

ACONDICIONAMIENTO DE LA SEÑAL DE VOLTAJE DEL ANEMÓMETRO EMPLEANDO UN SENSOR DE EFECTO HALL

El circuito de acondicionamiento que se diseñó para esta señal es el siguiente:

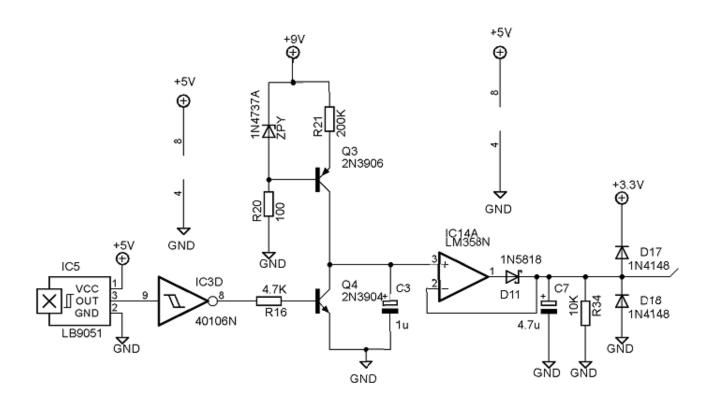


Figura 1: Acondicionamiento de la señal provenienete del sensor de Efecto Hall



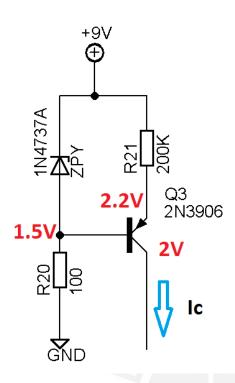


Figura 2: Fuente de corriente

En primer lugar, el circuito está compuesto por una fuente de corriente, formado por un transistor PNP 2N3906, un diodo Zener 1N4737A (7.5V), una resistencia de drenaje para este diodo (100 ohmios) y una resistencia en el emisor del transistor (200K).

El diodo zener mantiene un nivel de voltaje de 7.5V, por lo tanto se tiene 1.5V en la base del transistor. La diferencia de voltaje entre el emisor y la base del transistor es de 0.7V entonces el voltaje en el emisor del transistor será igual a 2.2V.

La corriente del colector lc será:

$$I_C = \frac{9V - 2.2V}{200K} = 3.4 \times 10^{-5}A$$

La caída de tensión entre el emisor y el colector es de 0.2V, entonces el voltaje en el colector será de 2V. Este voltaje será el máximo voltaje de carga del condensador C3.

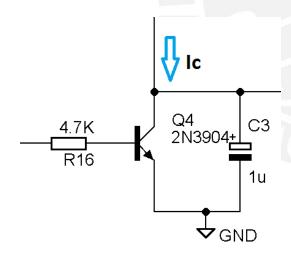


Figura 3: Circuito de carga del condensador

El circuito que se muestra en la figura 3 cargará el condensador C3 de 1uF con la corriente lc proveniente del transistor PNP hasta un voltaje máximo de 2V hallado en el circuito anterior. Se escogió el valor del condensador de 1uF debido a que el tiempo de carga es suficiente rápido para la velocidad de conmutación del transistor 2N3904.

La resistencia R16 de valor 4.7K proporciona la corriente necesaria para el correcto funcionamiento del transistor 2N3904. La carga y descarga del condensador es de la siguiente manera:

- Cuando se tiene un pulso positivo (5V) en la base del transistor entonces el transistor conduce y no carga el condensador ya que toda la corriente se va a tierra.
- Cuando se tiene un pulso de 0V en la base del transistor entonces el transistor no conduce y se carga el condensador C3 hasta un voltaje máximo de 2V.



De la ecuación de carga del condensador de obtiene la siguiente relación:

$$\begin{split} i(t) &= C \frac{dV}{dt} \\ v(t) &= \frac{1}{C} \int_{t_0}^t i(t) dt + v(t_0) \\ v_c(0) &= 0 \\ pero \ como \ i(t) \ es \ cte \ entonces : \ v_c(t) = \frac{1}{C} \int_0^t I dt + 0 \end{split}$$

Finalmente se obtiene la ecuación de carga del condensador con una fuente de corriente constante:

$$v_c(t) = \frac{1}{C}I\Delta t = \frac{I\Delta t}{C}$$

Donde I es la corriente que fluye por la resistencia 200K:

$$I = \frac{9 - V_{Emisor\ del\ transistor\ 2N3906}}{200K} = \frac{9 - 2.2}{200K} = 3.4 * 10^{-5}A$$

Reemplazando este valor en la ecuación de carga del condensador:

$$v_c(t) = \frac{1}{10^{-6}} * 3.4 * 10^{-5} \Delta t = 34 * \Delta t \rightarrow \Delta t = \frac{v_c}{34}$$

Donde Δt es el tiempo en el cual el voltaje en la base del transistor 2N3904 es 0V.

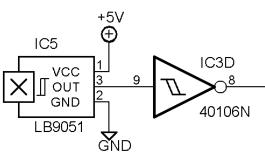


Figura 4: Sensor de Efecto Hall y compuerta inversora

El sensor de Efecto hall seleccionado fue el LB9051. La salida digital que proporciona varía desde 0.02V hasta 5V. El funcionamiento de este sensor es el siguiente:

Cuando las líneas de campo de polaridad norte (proporcionadas por el imán de polaridad norte) son perpendiculares a la superficie de este sensor entonces se tiene un nivel alto (5V) y debido a la histéresis de este dispositivo se mantiene en este nivel hasta que las líneas de campo de polaridad sur (proporcionadas por el imán de polaridad sur) son perpendiculares a la superficie de este dispositivo y en este caso la salida del sensor de Efecto Hall cambia a 0V.

Se quiere sensar el tiempo en el cual las líneas de campo de polaridad norte son perpendiculares a la superficie del sensor de Efecto Hall, sin embargo cuando esto sucede se tiene un nivel alto (5V) en la base del transistor 2N3904, por lo tanto el condensador no se carga. Para solucionar esto se ha colocado una compuerta inversora Schmitt trigger



HEF40106B. De esta manera, cuando las líneas de campo de polaridad norte sean perpendiculares a la superficie del dispositivo se tendrá un nivel alto en la salida del sensor (5V) y después de la compuerta inversora se tendrá 0V en la base del transistor 2N3904 entonces el condensador se cargara hasta un nivel que el proporcional al tiempo en el cual las líneas de campo de polaridad norte del imán son perpendiculares a la superficie del dispositivo.

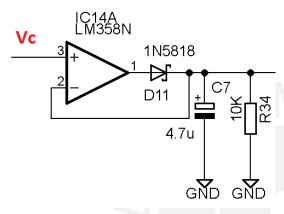


Figura 5: Seguidor de voltaje

mayor demora.

El voltaje de carga Vc del condensador C3 de 1uF ingresa a un buffer, en este caso se está empleando el amplificador operacional LM358N. Se colocó el diodo 1N5818 para evitar el paso de la corriente entre el capacitor y el amplificador operacional. El capacitor almacena la tensión de entrada Vc

La conexión entre la entrada del amplificador operacional (punto 2) y la salida del diodo 1N5818 es para que la tensión de entrada sea almacenada sin la caída de tensión del diodo. El amplificador va a compensar la caída en el diodo para que el voltaje en el

condensador C7 sea igual a Vc. Se ha colocado la resistencia R34 de 10K en paralelo con el condensador de C7 de 4.7uF para la descarga de este. La resistencia va a determinar el tiempo de descarga del capacitor. La corriente fluye del capacitor hacia el resistor para descargarlo. Mediante pruebas se verificó que para los valores seleccionados de C7 Y R34 el condensador se descarga rápidamente cuando se tiene la máxima variación en la entrada, proporcionando un adecuado seguimiento del valor en la entrada del amplificador. De esta manera podrá almacenar un nuevo valor de tensión ya sea menor o mayor sin

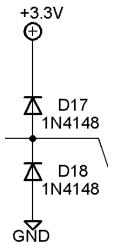


Figura 6: Diodos de protección

Asimismo, se adicionó unos diodos 1N4148 de alta velocidad de switching (4ns) que permiten proteger el puerto ADC del microcontrolador, en caso se presenten voltajes negativos o positivos mayores a 3.3V.



ACONDICIONAMIENTO DE LA SEÑAL DE VOLTAJE DE LA BATERÍA

La batería seleccionada proporciona una señal comprendida entre 11 y 13.5 voltios, esto debido a las siguientes razones:

 Se está empleando reguladores de la serie LM78XX, tales como los reguladores LM7809 y LM7805. Estos reguladores necesitan de una tensión de dropout de 2V, es decir, que la tensión de entrada debe ser al menos 2V mayor que la tensión de salida. De otra manera, el circuito dejará de funcionar.

reak Output Outretit	IFIN	1]-20 0	1.0	
Short-Circuit Current	Isc	VI=35V, Tj=25 C	250	mA
Dropout Voltage	Vd	Tj=25 C	2.0	V

Figura 1: Voltaje de Dropout de los reguladores LM7805 Y LM7809

A partir de lo anterior se concluye que es necesario un valor de voltaje mínimo de 11V en la entrada para el correcto funcionamiento del regulador LM7809 (9V).

 A partir de la figura 2, se observa que la tensión máxima de carga para una batería de 12V es un valor cercano a los 14V, 13.8V aproximadamente. Para este proyecto se ha considerado como voltaje máximo de carga para la batería el valor de 13.5V.

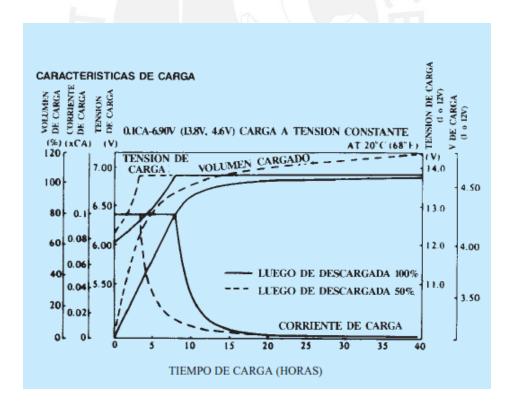


Figura 2: Curva de carga de la batería YUASA 12V



Esta señal no puede ser medida directamente por el microcontrolador, por lo cual se acondiciona con el divisor de voltaje que se muestra en la figura 3:

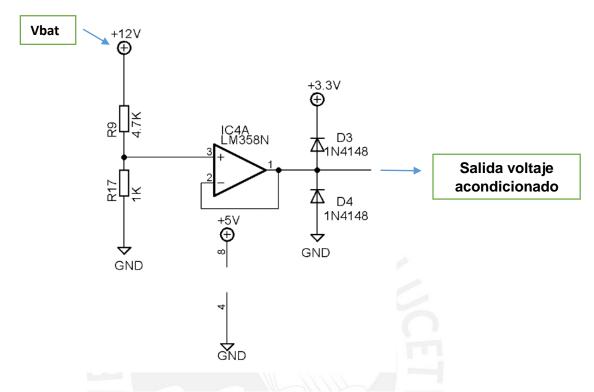


Figura 3: Acondicionamiento del voltaje de la batería

El voltaje en el punto 3 será:

$$V_3 = V_{bat} * \frac{1K}{1K + 4.7K} = V_{bat} * 0.1754$$

Para un valor de voltaje de 11V de la batería se tendrá en el punto 3:

$$V_3 = 11 * 0.1754 = 1.9294V$$

Para un valor de voltaje de 13.5V de la batería se tendrá en el punto 3:

$$V_3 = 13.5 * 0.1754 = 2.3679V$$

Por seguridad se medirá el voltaje que entrega la batería desde 0V hasta 15V. El voltaje máximo de salida acondicionado que se tendrá será:

$$V_{salida} = V_3 = 15 * 0.1754 = 2.631V < 3.3V$$
 (Máximo valor de entrada al puerto ADC)

Se adaptó las impedancias con un buffer (Op-amp con realimentación negativa) y como se observa en la figura 3 se adicionó unos diodos 1N4148 de alta velocidad de switching (4ns) que permiten proteger el puerto ADC del microcontrolador, en caso existan sobre voltajes, ya sean negativos (menores a 0 voltios) o positivos (mayores a 3.3 voltios).



Las resistencias 1K y 4.7K son resistencias de precisión, la cantidad de potencia máxima que disipan se muestran en los siguientes cálculos:

El voltaje máximo que entregue la batería será de 13.5V:

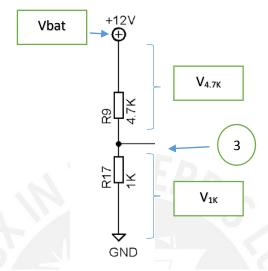


Figura 3: Divisor de voltaje

El voltaje en el punto 3 es:

$$V_3 = V_{hat} * 0.1754$$

La diferencia de potencial en la resistencia de 4.7K será:

$$V_{4.7K} = V_{bat} - V_{bat} * 0.1754 = V_{bat} * 0.8246$$

 $V_{4.7K} = 13.5 * 0.8246 = 11.1231V$

La potencia que disipará esta resistencia será:

$$P_{4.7K} = \frac{V_{4.7K}^2}{4.7K} = \frac{11.1231^2}{4.7K} = 0.02632W$$

Se escogió una resistencia de 0.5W.

La diferencia de potencial en la resistencia de 1K será:

$$V_{1K} = V_3 = V_{bat} * 0.1754 = 13.5 * 0.1754 = 2.3679V$$

La potencia que disipará esta resistencia será:

$$P_{1K} = \frac{V_{1K}^2}{1K} = \frac{2.3679^2}{1K} = 5.61 * 10^{-3}W$$

Se escogió una resistencia de 0.5W.



ACONDICIONAMIENTO DE LA SEÑAL DE VOLTAJE DEL PANEL SOLAR

El panel solar seleccionado es de 12V de voltaje nominal y proporciona una señal comprendida entre 0 y 18 voltios.

Esta señal no puede ser medida directamente por el microcontrolador, por lo cual se acondiciona con el divisor de voltaje que se muestra en la figura 1:

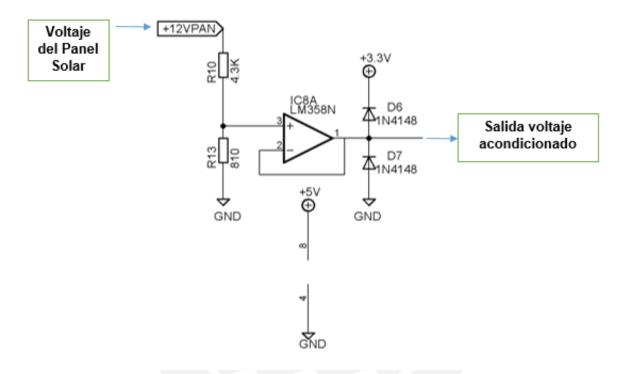


Figura 1: Acondicionamiento del voltaje del panel solar

El voltaje en el punto 3 será:

$$V_3 = V_{Panel\ Solar} * \frac{810}{810 + 4.3K} = V_{Panel\ Solar} * 0.1585$$

Para un valor de voltaje de 18V del panel solar se tendrá en el punto 3:

$$V_3 = 18 * 0.1585 = 2.853V$$

Por seguridad se medirá el voltaje que entrega el panel solar desde 0V hasta 20V. El voltaje máximo de salida acondicionado que se tendrá será:

$$V_{salida} = V_3 = 20 * 0.1585 = 3.17V < 3.3V$$
 (Máximo valor de entrada al puerto ADC)

Se adaptó las impedancias con un buffer (Op-amp con realimentación negativa) y como se observa en la figura 1 se adicionó unos diodos 1N4148 de alta velocidad de switching (4ns)



que permiten proteger el puerto ADC del microcontrolador, en caso existan sobre voltajes, ya sean negativos (menores a 0 voltios) o positivos (mayores a 3.3 voltios).

Las resistencias 810 ohms y 4.3K son resistencias de precisión, la cantidad de potencia máxima que disipan se muestran en los siguientes cálculos:

El voltaje máximo que entregará el panel solar será de 18V:

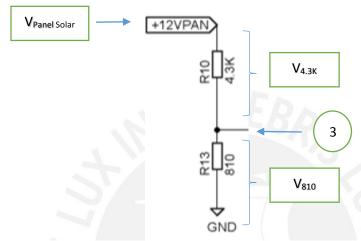


Figura 2: Divisor de voltaje

El voltaje en el punto 3 es:

$$V_3 = V_{Panel\ Solar} * 0.1585$$

La diferencia de potencial en la resistencia de 4.3K será:

$$V_{4.3K} = V_{Panel\ Solar} - V_{Panel\ Solar} * 0.1585 = V_{Panel\ Solar} * 0.8415$$

$$V_{4.3K} = 18 * 0.8415 = 15.147V$$

La potencia que disipará esta resistencia será:

$$P_{4.3K} = \frac{V_{4.3K}^2}{4.3K} = \frac{15.147^2}{4.3K} = 0.05335W$$

Se escogió una resistencia de 0.5W.

La diferencia de potencial en la resistencia de 810 ohms será:

$$V_{810} = V_3 = V_{Panel\ Solar} * 0.1585 = 18 * 0.1585 = 2.853V$$

La potencia que disipará esta resistencia será:

$$P_{810} = \frac{V_{810}^2}{810} = \frac{2.853^2}{810} = 0.010W$$

Se escogió una resistencia de 0.5W.



ACONDICIONAMIENTO DE LA SEÑAL DE SENSADO DE LA ORIENTACIÓN DEL VIENTO

Para el sensado de la orientación del viento se ha empleado un arreglo de interruptores Reed Switch los cuales se activarán ante la presencia del campo magnético del imán acoplado al eje del anemómetro cuando este pase por encima de alguno de estos interruptores de acuerdo a la velocidad del viento.

Esto se logrará mediante el arreglo de resistencias de precisión que se muestra en la figura 1:

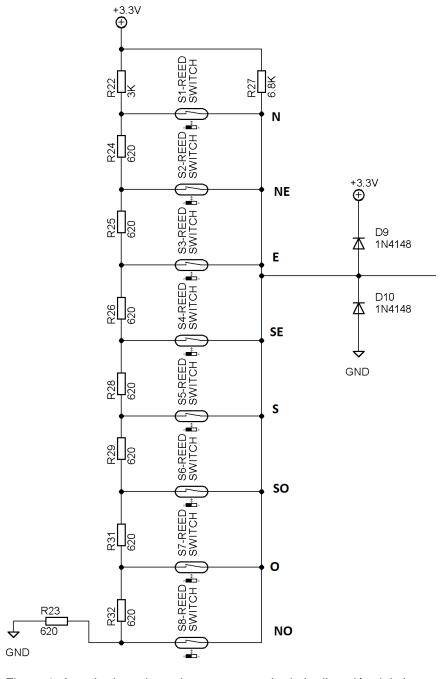
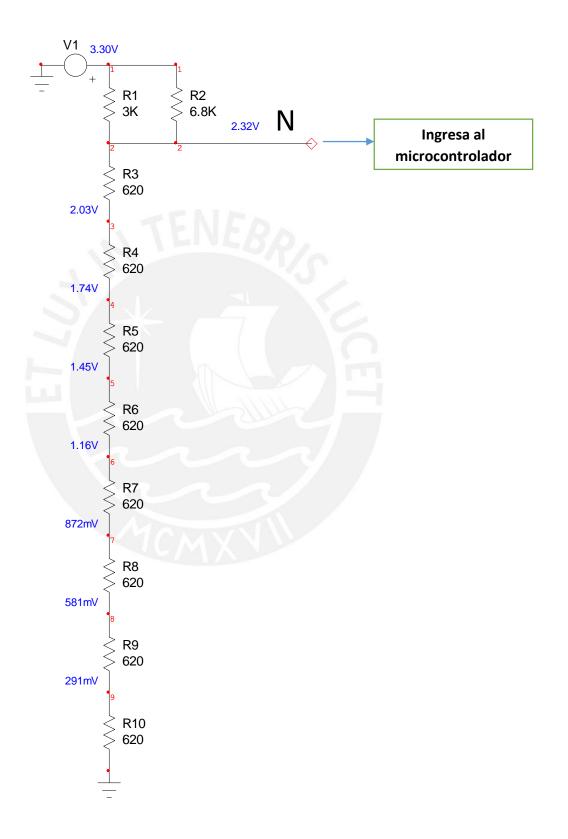


Figura 1: Arreglo de resistencias para sensado de la dirección del viento

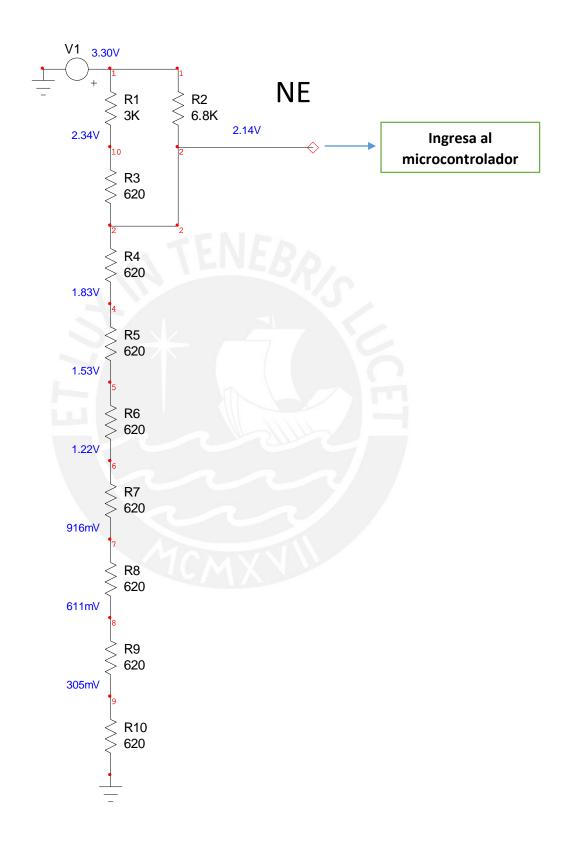


Caso 1: S1 se cierra → N



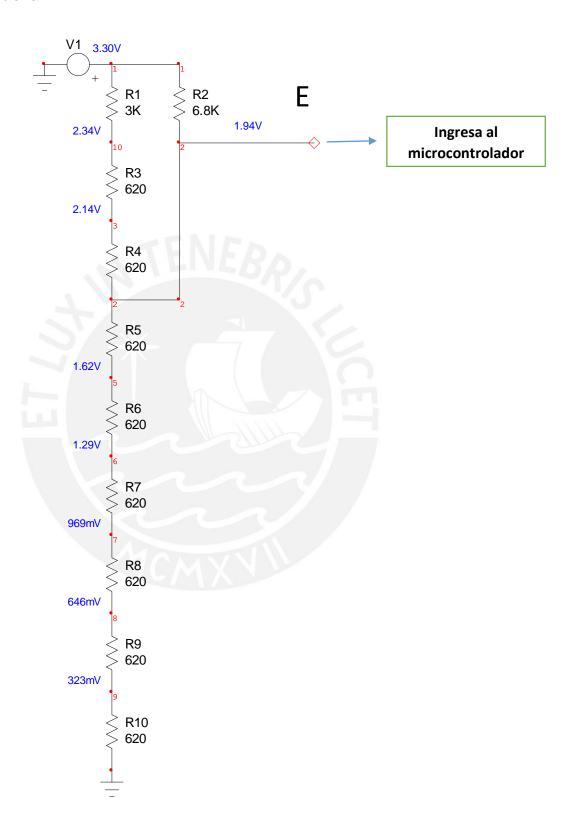


Caso 2: S2 se cierra → NE



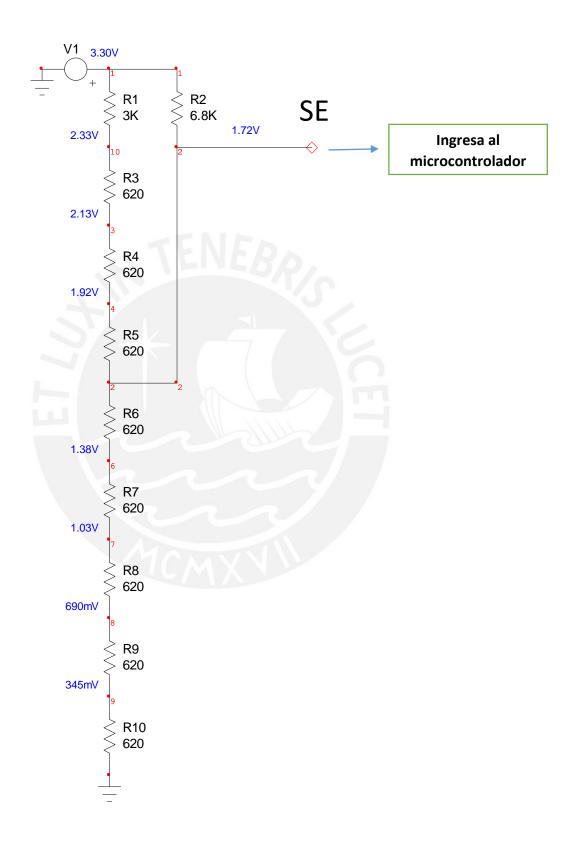


Caso 3: S3 se cierra → E



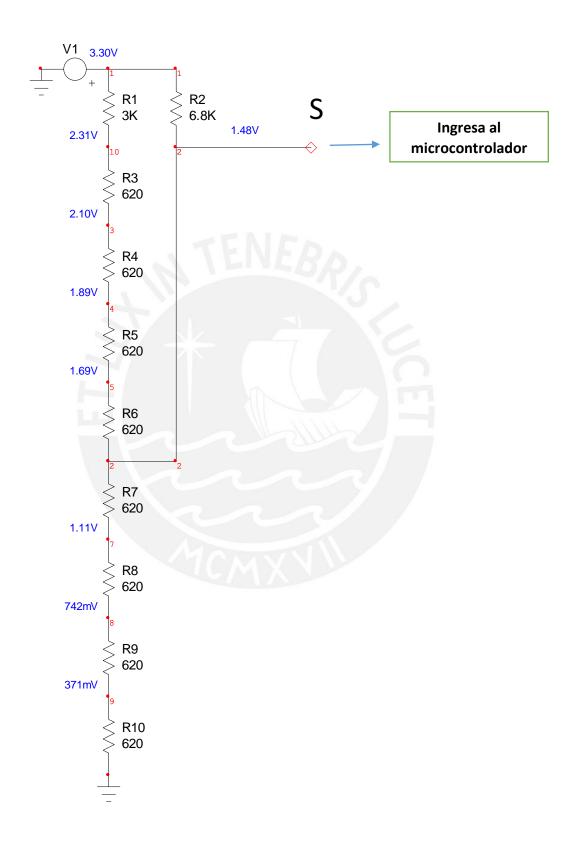


Caso 4: S4 se cierra → SE



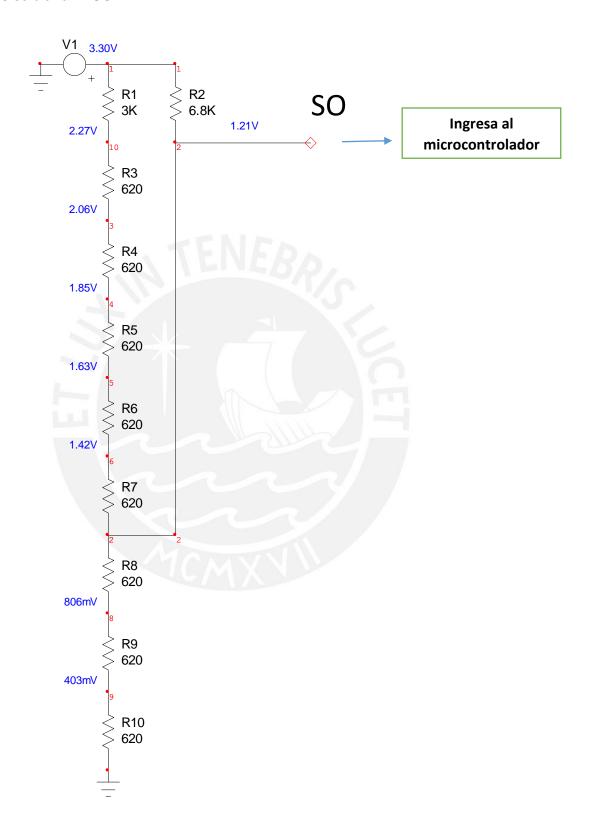


Caso 5: S5 se cierra → S



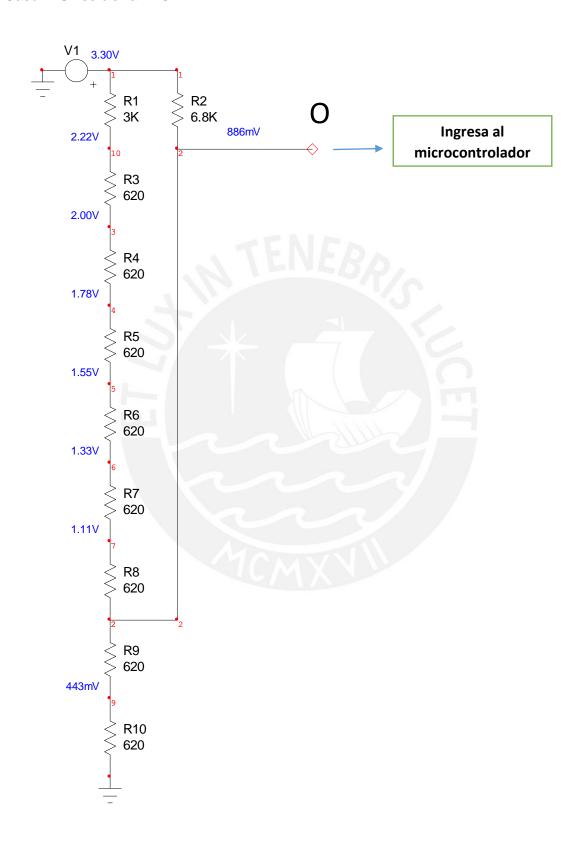


Caso 6: S6 se cierra → SO



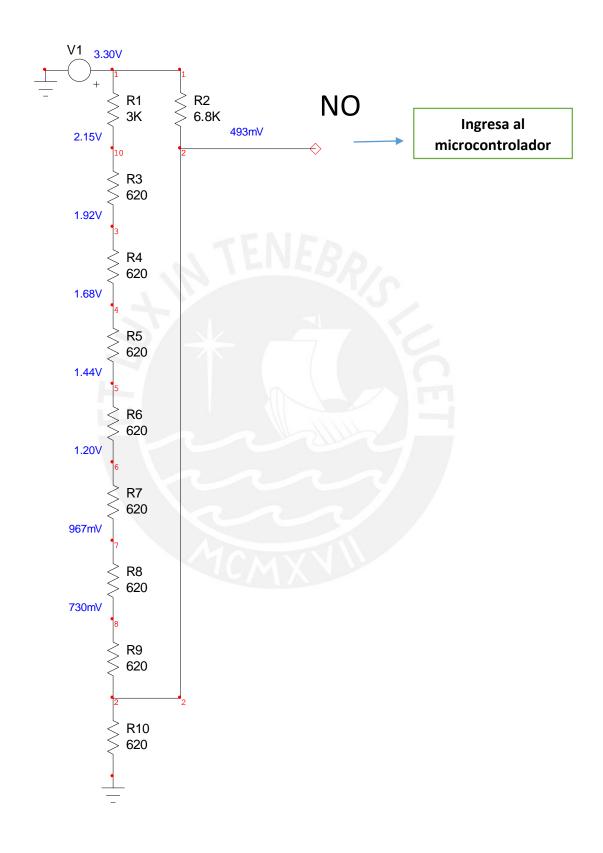


Caso 7: S7 se cierra → O



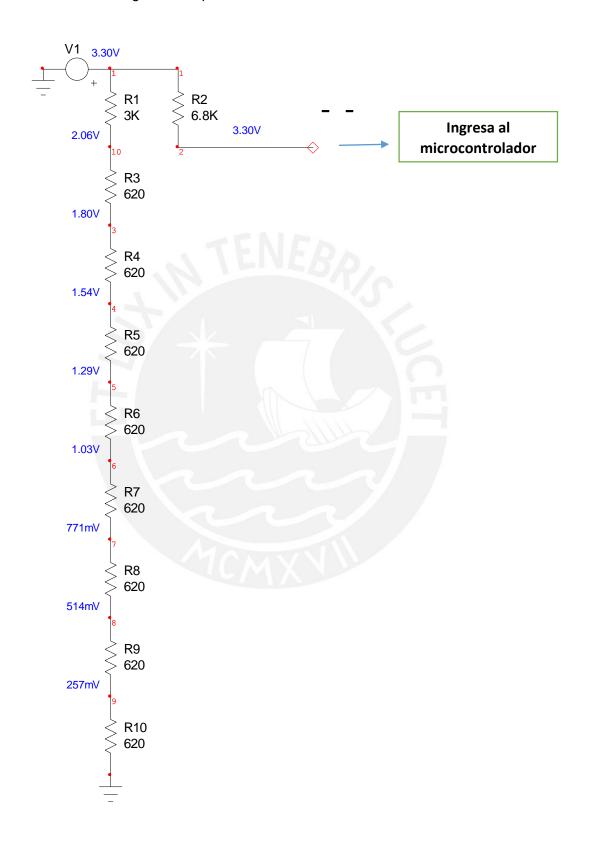


Caso 8: S8 se cierra → NO





Caso 9: No se cierra ningún interruptor Reed Switch → - -





Para cada uno de los casos se tiene un rango de voltaje distinto que ingresará al microcontrolador, se ha considerado sumar y restar 50mV a cada valor de voltaje hallado en los casos anteriores para establecer los rangos de voltaje, de esta manera se tienen los siguientes rangos:

Caso 1: N

$$V_{Divisor\ de\ voltaje} = 2.32V$$

 $Extremo\ inferior = 2.32V - 50mV = 2.27V$

 $Extremo\ superior = 2.32V + 50mV = 2.37V$

Rango de voltaje para N = 2.27V - 2.37V

Caso 2: NE

$$V_{Divisor\ de\ voltaje} = 2.14V$$

 $Extremo\ inferior = 2.14V - 50mV = 2.09V$

 $Extremo\ superior = 2.14V + 50mV = 2.19V$

Rango de voltaje para NE = 2.09V - 2.19V

Caso 3: E

$$V_{Divisor\ de\ volta\,ie} = 1.94V$$

 $Extremo\ inferior = 1.94V - 50mV = 1.89V$

 $Extremo\ superior = 1.94V + 50mV = 1.99V$

Rango de voltaje para E = 1.89V - 1.99V

Caso 4: SE

$$V_{Divisor\ de\ voltaje} = 1.72V$$

$$Extremo\ inferior = 1.72V - 50mV = 1.67V$$

$$Extremo\ superior = 1.72V + 50mV = 1.77V$$

Rango de voltaje para SE = 1.67V - 1.77V



Caso 5: S

$$V_{Divisor\ de\ voltaje} = 1.48V$$

 $Extremo\ inferior = 1.48V - 50mV = 1.43V$

 $Extremo\ superior = 1.48V + 50mV = 1.53V$

Rango de voltaje para S = 1.43V - 1.53V

Caso 6: SO

$$V_{Divisor\ de\ voltaje} = 1.21V$$

 $Extremo\ inferior = 1.21V - 50mV = 1.16V$

 $Extremo\ superior = 1.21V + 50mV = 1.26V$

Rango de voltaje para SO = 1.16V - 1.26V

Caso 7: O

$$V_{Divisor\ de\ voltaje} = 886mV$$

 $Extremo\ inferior = 886mV - 50mV = 836mV$

Extremo superior = 886mV + 50mV = 936mV

Rango de voltaje para 0 = 836mV - 936mV

Caso 8: NO

$$V_{Divisor\ de\ voltaje} = 493mV$$

 $Extremo\ inferior = 493mV - 50mV = 443mV$

 $Extremo\ superior = 493mV + 50mV = 543mV$

Rango de voltaje para NO = 443mV - 543mV

Caso 9: - -

$$V_{Divisor\ de\ voltaie} = 3.3V$$

Se ha empleado un divisor de tensión puesto que se quiere emplear un solo puerto ADC del microcontrolador.

Se adicionó unos diodos 1N4148 de alta velocidad de switching (4ns) que permiten proteger el puerto ADC del microcontrolador, en caso existan sobre voltajes, ya sean negativos (menores a 0 voltios) o positivos (mayores a 3.3 voltios).



DISEÑO DEL SISTEMA DE ALIMENTACIÓN AUTÓNOMO

Se emplea la energía de un panel solar de tensión nominal 12V-5W para cargar la batería que alimentará todo el sistema de medición. El circuito empleado para cargar la batería se muestra en la figura 1. El diodo D5 1N5818 se conecta entre el panel solar y la batería para evitar descargas.

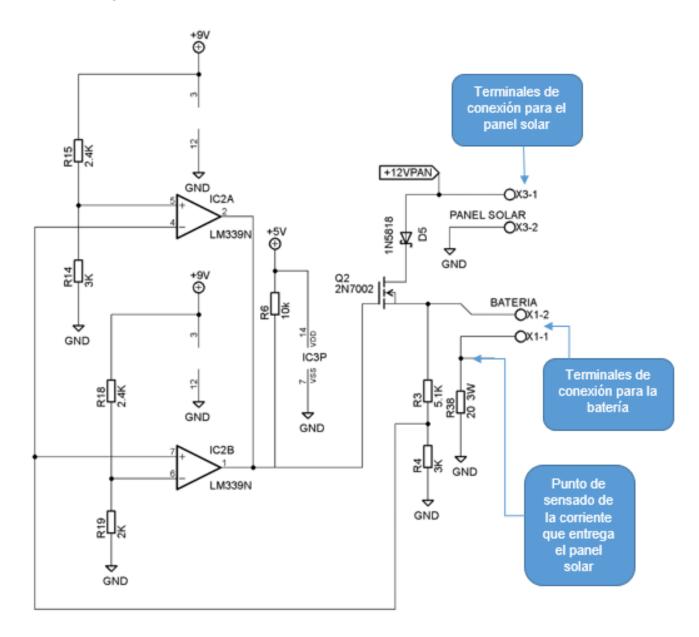


Figura 1: Diagrama esquemático del sistema de alimentación autónomo

Las resistencias R14 de 3K y R15 de 2.4K fijan el voltaje del límite superior del comparador de ventana.



$$V_5 = 9 \times \frac{3K}{3K + 2.4K} = 5V$$

El límite superior será de 5V. Las resistencias R18 de 2.4K y R19 de 2K fijan el voltaje del límite inferior del comparador de ventana.

$$V_6 = 9 \times \frac{2K}{2K + 2.4K} = 4V$$

El límite inferior será de 4V.

El comparador de ventana empleado está compuesto de dos amplificadores operaciones LM339N, los cuales tienen salida open collector por lo cual se requiere de una resistencia de Pull Up. La resistencia R6 de 10K es la resistencia de Pull Up.

Las resistencias R3 de 5.1K y R4 de 3K conforman un divisor de voltaje para el sensado del voltaje de la batería. De esta manera cuando se tenga 11V (límite mínimo de carga) en los terminales de la batería, se tendrá en el divisor de tensión el siguiente valor de voltaje:

$$V_{Divisor} = 11 \times \frac{3K}{3K + 5.1K} = 4V$$

Asimismo cuando se tenga 13.5V (límite máximo de carga) en los terminales de la batería, se tendrá en el divisor de tensión el siguiente valor de voltaje:

$$V_{Divisor} = 13.5 \times \frac{3K}{3K + 5.1K} = 5V$$

De esta manera, los límites de carga para la batería comprende el rango de voltaje de 11V hasta 13.5V, este valor de voltaje se reduce en el divisor de tensión y luego ingresa a los amplificadores operacionales (punto 4 y 7 de la figura 1).

Cuando el valor de voltaje en los terminales de la batería se encuentra entre 11V y 13.5V entonces se tendrá 4V y 5V en el divisor de tensión, el voltaje comprendido dentro de este rango proporciona una salida de 5V en el comparador de ventana y se dispara el mosfet 2N7002. De esta manera se permite el paso de la corriente entre drenador y surtidor, es decir, el panel solar entrega corriente para cargar la batería.

Cuando el valor de voltaje en los terminales de la batería se encuentra fuera del rango de 11V y 13.5V (ya sea menor o mayor) se tendrá una salida de 0V en el comparador de ventana y por lo tanto el mosfet 2N7002 no se dispara. Esto hace que no fluya corriente entre el drenador y surtidor, entonces el panel solar no carga la batería.

La potencia disipada por las resistencias R3 y R4 cuando la batería entrega el máximo voltaje (13.5V) se muestra a continuación.

$$P_{R3} = \frac{(13.5 - 5)^2}{5.1K} = 0.014W$$

$$P_{R4} = \frac{(5)^2}{3K} = 8.33 \times 10^{-3} W$$



ACONDICIONAMIENTO DE LA SEÑAL DE VOLTAJE DEL GENERADOR DC

El motor de corriente continua empleado como generador que fue seleccionado proporciona una señal en el rango comprendido desde -12V hasta 12V, esta señal no puede ser medida directamente por el microcontrolador por lo cual debe ser acondicionada para luego ser muestreada por ADC. Esto será posible mediante el arreglo de resistencias que se muestra en la figura 1:

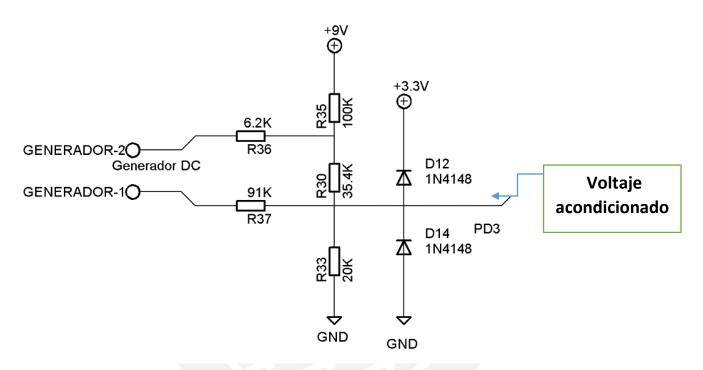
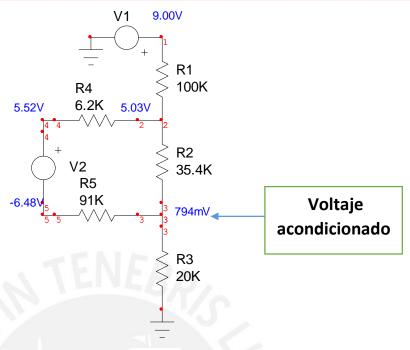


Figura 1: Arreglo de resistencias para el acondicionamiento de la señal de voltaje del generador DC

El generador entrega como máximo 12V, el voltaje acondicionado para este valor de tensión entregado por el generador sera igual a 794mV tal como se muestra en circuito de abajo:





A continuación se muestran los valores de potencia que disipa cada una de las resistencias:

$$P_{100K} = \frac{(9 - 5.03)^2}{100K} = 1.57609 \times 10^{-4} W$$

$$P_{6.2K} = \frac{(5.52 - 5.03)^2}{6.2K} = 3.8725 \times 10^{-5} W$$

$$P_{91K} = \frac{(794 \times 10^{-3} + 6.48)^2}{91K} = 5.8144 \times 10^{-4} W$$

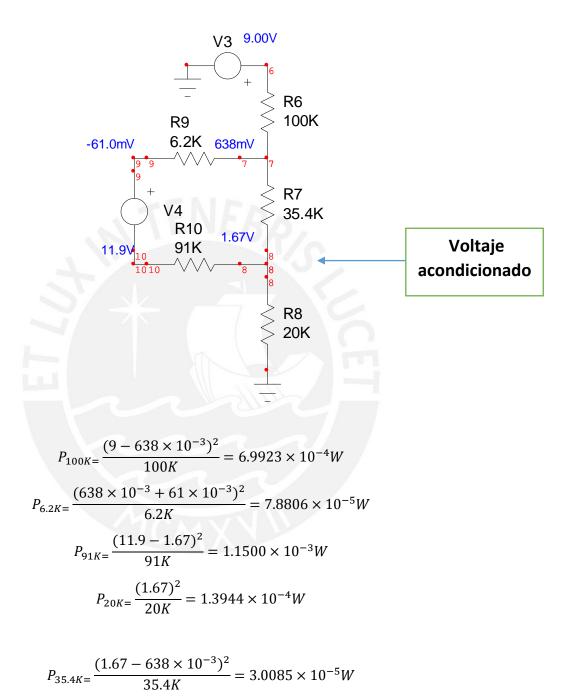
$$P_{20K} = \frac{(794 \times 10^{-3})^2}{20K} = 3.1521 \times 10^{-5} W$$

$$P_{35.4K} = \frac{(5.03 - 794 \times 10^{-3})^2}{35.4K} = 5.0688 \times 10^{-4}W$$

Se escogieron resistencias de 0.5W.



El generador entrega como mínimo -12V, el voltaje acondicionado para este valor de tensión entregado por el generador será igual a 1.67V tal como se muestra en circuito de abajo:



Se escogieron resistencias de 0.5W.



Mediante este arreglo de resistencias se reduce la amplitud de la tensión entregada por el generador y se tendrá una señal que va desde -366mV hasta 366mV. Para desplazar la señal y hacer que esta sea solo positiva, se requiere de una fuente de alimentación de 9 voltios, con lo cual se tendrá una señal que varía desde 794mV hasta 1.67V. Este rango es menor al rango del ADC, lo cual permite medir sobre tensiones en caso se presenten. Asimismo, se adicionó unos diodos 1N4148 de alta velocidad de switching (4ns) que permiten proteger el puerto ADC del microcontrolador, en caso se presenten voltajes negativos o positivos mayores a 3.3V





ACONDICIONAMIENTO DE LA SEÑAL DE SENSADO DE LA CORRIENTE QUE ENTREGA EL PANEL SOLAR

A partir de las mediciones realizadas al panel solar, la batería y la resistencia que componen el circuito que se muestra en la figura 1:

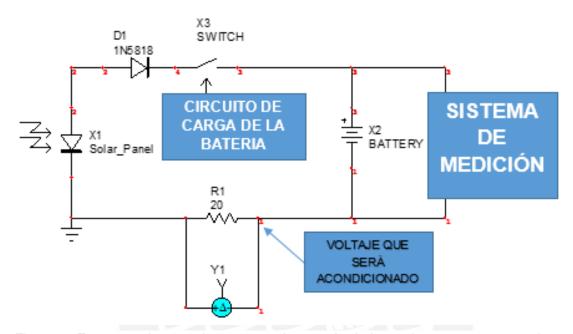


Figura 1: Esquema de conexiones para el sensado de la corriente que entrega el panel solar

Se obtuvieron los resultados que se muestran en la tabla 1:



Voltaje entregado por la	Voltaje entregado por el	Voltaje en la	Corriente que circula por la	Potencia que disipa	
bateria (V)	panel solar (V)	resistencia (V)	resistencia en el circuito (mA)	la resistencia (W)	
11	11.1	0.1	5	0.0005	
11	12	1	50	0.05	
11	13	2	100	0.2	
11	14	3	150	0.45	
11	16	5	250	1.25	
11	17	6	300	1.8	
11	18	7	350	2.45	
12	12.1	0.1	5	0.0005	
12	13	1	50	0.05	
12	14	2	100	0.2	
12	15	3	150	0.45	
12	17	5	250	1.25	
12	18	6	300	1.8	
13	13.1	0.1	5	0.0005	
13	14	1	50	0.05	
13	15	2	100	0.2	
13	16	3	150	0.45	
13	17	4	200	0.8	
13	18	5	250	1.25	
13.5	13.6	0.1	5	0.0005	
13.5	14	0.5	25	0.0125	
13.5	15	1.5	75	0.1125	
13.5	16	2.5	125	0.3125	
13.5	17	3.5	175	0.6125	
13.5	18	4.5	225	1.0125	

Tabla 1: Valores de voltaje y corriente del panel solar, la resistencia y la batería

El voltaje en la resistencia R1, no puede ser medido directamente por el microcontrolador, por lo cual requiere ser acondicionado, luego se calculará la corriente empleando la Ley de Ohm.

V = IR



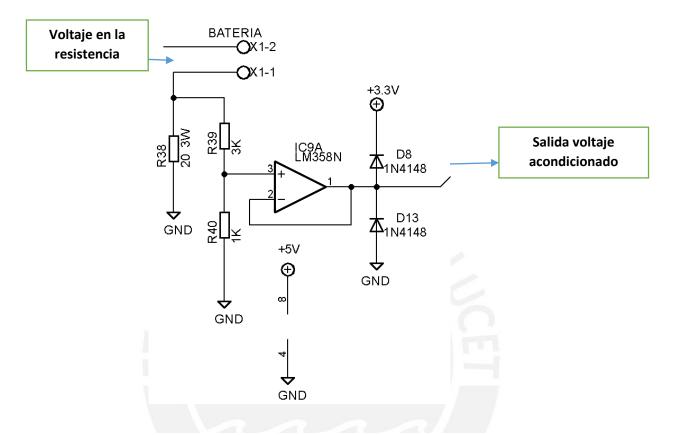


Figura 2: Acondicionamiento de la señal de voltaje en la resistencia de 20Ω 3W

A partir de los valores de voltaje y corriente que se muestran en la tabla 1, el voltaje máximo en la resistencia de 20Ω 3W será de 7V.

El voltaje en el punto 3 será:

$$V_3 = V_{Resistencia} * \frac{1K}{1K + 3K} = V_{Resistencia} * 0.25$$

Para un valor de voltaje de 7V en la resistencia se tendrá en el punto 3:

$$V_3 = 7 * 0.25 = 1.75V$$

 $V_{Salida\ acondicionado\ máximo} = 1.75V < 3.3V\ (Máximo\ valor\ de\ entrada\ al\ puerto\ ADC)$

Se adaptó las impedancias con un buffer (Op-amp con realimentación negativa) y como se observa en la figura 2 se adicionó unos diodos 1N4148 de alta velocidad de switching (4ns)



que permiten proteger el puerto ADC del microcontrolador, en caso existan sobre voltajes, ya sean negativos (menores a 0 voltios) o positivos (mayores a 3.3 voltios).

Las resistencias de 1K y 3K son resistencias de precisión, la cantidad de potencia máxima que disipan se muestran en los siguientes cálculos:

El voltaje máximo que se presentara en la resistencia de 20Ω 3W será de 7V:

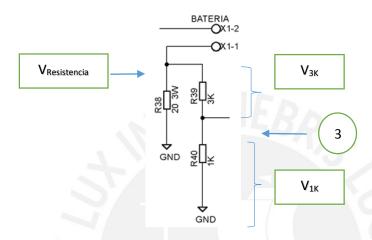


Figura 3: Divisor de voltaje

El voltaje en el punto 3 es:

$$V_3 = V_{Resistencia} * 0.25$$

La diferencia de potencial en la resistencia de 3K será:

$$V_{3K} = V_{Resistencia} - V_{Resistencia} * 0.25 = V_{Resistencia} * 0.75$$

$$V_{3K} = 7 * 0.75 = 5.25V$$

La potencia que disipará esta resistencia será:

$$P_{3K} = \frac{V_{3K}^2}{3K} = \frac{5.25^2}{3K} = 9.1875 * 10^{-3}W$$

Se escogió una resistencia de 0.5W.

La diferencia de potencial en la resistencia de 1K será:

$$V_{1K} = V_3 = V_{Resistencia} * 0.25 = 7 * 0.25 = 1.75V$$

La potencia que disipará esta resistencia será:

$$P_{1K} = \frac{V_{1K}^2}{1K} = \frac{1.75^2}{1K} = 3.0625 * 10^{-3} W$$

Se escogió una resistencia de 0.5W.



RATIOMETRIC, LINEAR HALL-EFFECT SENSORS

The UGN3503LT, UGN3503U, and UGN3503UA Hall-effect sensors accurately track extremely small changes in magnetic flux density—changes generally too small to operate Hall-effect switches.

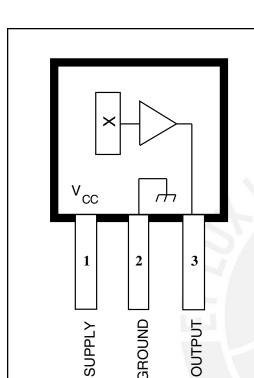
As motion detectors, gear tooth sensors, and proximity detectors, they are magnetically driven mirrors of mechanical events. As sensitive monitors of electromagnets, they can effectively measure a system's performance with negligible system loading while providing isolation from contaminated and electrically noisy environments.

Each Hall-effect integrated circuit includes a Hall sensing element, linear amplifier, and emitter-follower output stage. Problems associated with handling tiny analog signals are minimized by having the Hall cell and amplifier on a single chip.

Three package styles provide a magnetically optimized package for most applications. Package suffix 'LT' is a miniature SOT-89/TO-243AA transistor package for surface-mount applications; suffix 'U' is a miniature three-lead plastic SIP, while 'UA' is a three-lead ultra-mini-SIP. All devices are rated for continuous operation over the temperature range of -20°C to +85°C.

FEATURES

- **■** Extremely Sensitive
- Flat Response to 23 kHz
- Low-Noise Output
- 4.5 V to 6 V Operation
- Magnetically Optimized Package



Dwg. PH-006

Pinning is shown viewed from branded side.

ABSOLUTE MAXIMUM RATINGS

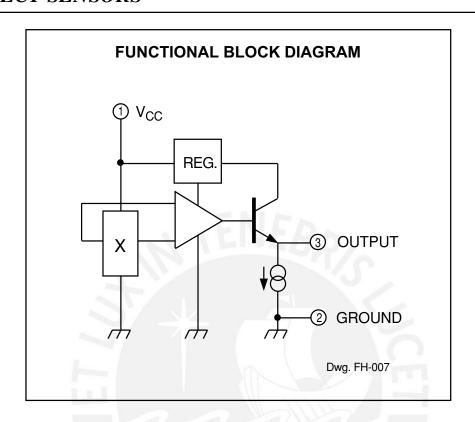
Supply Voltage, V_{CC}...... 8 V Magnetic Flux Density, B Unlimited Operating Temperature Range, T_A -20°C to +85°C Storage Temperature Range, T_s..... -65°C to +150°C

Always order by complete part number, e.g., UGN3503UA





LINEAR HALL-EFFECT SENSORS



ELECTRICAL CHARACTERISTICS at $T_A = +25$ °C, $V_{CC} = 5$ V

			Limits			
Characteristic	Symbol	Test Conditions	Min.	Тур.	Max.	Units
Operating Voltage	V _{cc}		4.5	_	6.0	V
Supply Current	I _{cc}	CMXV		9.0	13	mA
Quiescent Output Voltage	V _{OUT}	B = 0 G	2.25	2.50	2.75	V
Sensitivity	ΔV_{OUT}	B = 0 G to ±900 G	0.75	1.30	1.75	mV/G
Bandwidth (-3 dB)	BW			23	_	kHz
Broadband Output Noise	V _{out}	BW = 10 Hz to 10 kHz		90	_	μV
Output Resistance	R _{OUT}		_	50	220	Ω

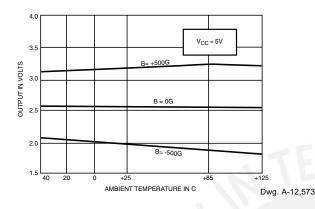
All output-voltage measurements are made with a voltmeter having an input impedance of at least 10 k Ω .

Magnetic flux density is measured at most sensitive area of device located 0.016" (0.41 mm) below the branded face of the 'U' package; 0.020" (0.51 mm) below the branded face of the 'UA' package; and 0.030" (0.76 mm) below the branded face of the 'LT' package.

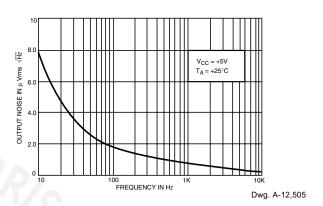


LINEAR HALL-EFFECT SENSORS

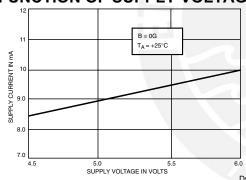
OUTPUT VOLTAGE AS A FUNCTION OF TEMPERATURE



OUTPUT NOISE AS A FUNCTION OF FREQUENCY

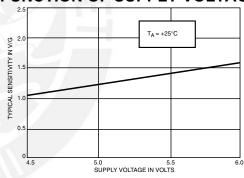


SUPPLY CURRENT AS A FUNCTION OF SUPPLY VOLTAGE



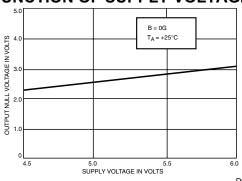
Dwg. A-12,506

DEVICE SENSITIVITY AS A FUNCTION OF SUPPLY VOLTAGE



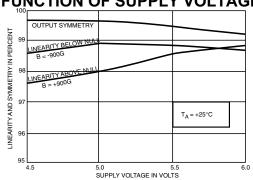
Dwg. A-12,507

OUTPUT NULL VOLTAGE AS A FUNCTION OF SUPPLY VOLTAGE



Dwg. A-12,508

LINEARITY AND SYMMETRY AS A FUNCTION OF SUPPLY VOLTAGE

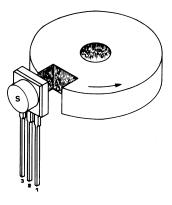


Dwg.A-12,509



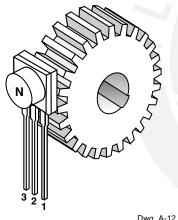
LINEAR HALL-EFFECT SENSORS

NOTCH SENSOR



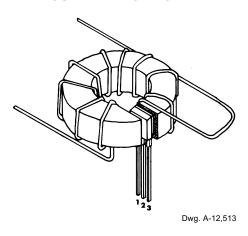
Dwg. A-12,574

GEAR TOOTH SENSOR



Dwg. A-12,512

CURRENT MONITOR



OPERATION

The output null voltage (B = 0 G) is nominally one-half the supply voltage. A south magnetic pole, presented to the branded face of the Halleffect sensor will drive the output higher than the null voltage level. A north magnetic pole will drive the output below the null level.

In operation, instantaneous and proportional output-voltage levels are dependent on magnetic flux density at the most sensitive area of the device. Greatest sensitivity is obtained with a supply voltage of 6 V, but at the cost of increased supply current and a slight loss of output symmetry. The sensor's output is usually capacitively coupled to an amplifier that boosts the output above the millivolt level.

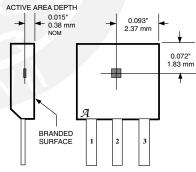
In two applications shown, a permanent bias magnet is attached with epoxy glue to the back of the epoxy package. The presence of ferrous material at the face of the package acts as a flux concentrator.

The south pole of a magnet is attached to the back of the package if the Hall-effect IC is to sense the presence of ferrous material. The north pole of a magnet is attached to the back surface if the integrated circuit is to sense the absence of ferrous matrial.

Calibrated linear Hall devices, which can be used to determine the actual flux density presented to the sensor in a particular application, are available.

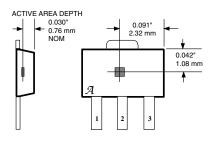
SENSOR LOCATIONS

SUFFIX "U"

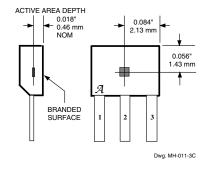


Dwg. MH-002-5C

SUFFIX "LT"



SUFFIX "UA"



Dwa. MH-008-9



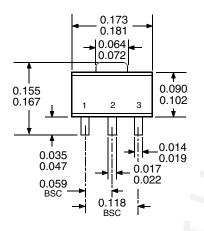
HALL-EFFECT SENSORS

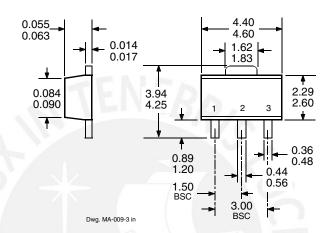
UGN3503LT

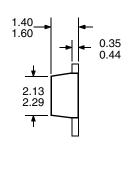
(SOT-89/TO-243AA)

Dimensions in Inches (for reference only)

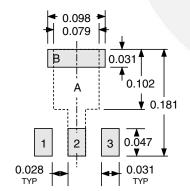
Dimensions in Millimeters (controlling dimensions)

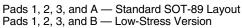






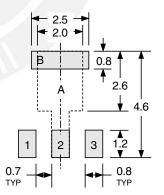
Dwg. MA-009-3 mm





Pads 1, 2, and 3 only — Lowest Stress, But Not Self Aligning

Dwg. MA-012-3 in



Pads 1, 2, 3, and A — Standard SOT-89 Layout Pads 1, 2, 3, and B — Low-Stress Version

Pads 1, 2, and 3 only — Lowest Stress, But Not Self Aligning

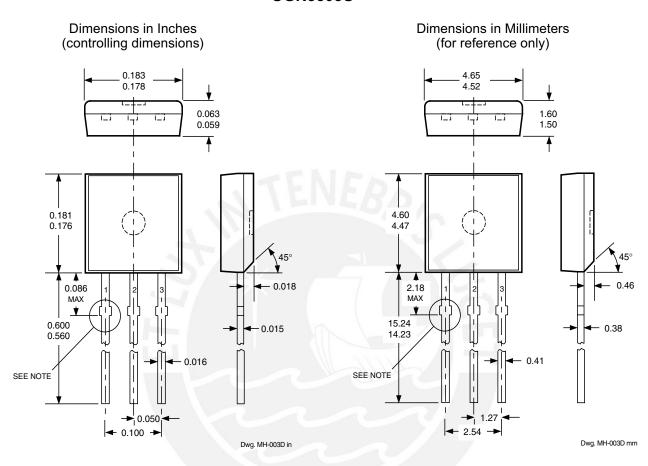
Dwg. MA-012-3 mm

NOTE: Exact body and lead configuration at vendor's option within limits shown.



LINEAR HALL-EFFECT SENSORS

UGN3503U



Devices in the 'U' package are NOT RECOMMENDED FOR NEW DESIGN

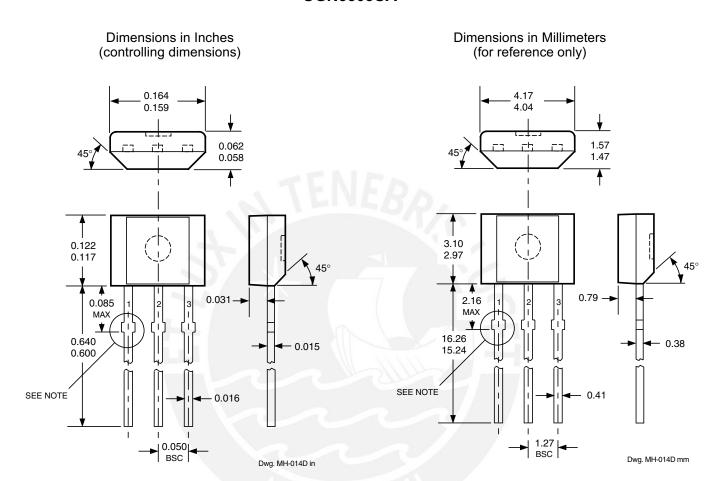
- NOTES: 1. Tolerances on package height and width represent allowable mold offsets. Dimensions given are measured at the widest point (parting line).
 - 2. Exact body and lead configuration at vendor's option within limits shown.
 - 3. Height does not include mold gate flash.
 - 4. Recommended minimum PWB hole diameter to clear transition area is 0.035" (0.89 mm).
 - 5. Minimum lead length was 0.500" (12.70 mm). If existing product to the original specifications is not acceptable, contact sales office before ordering.





LINEAR HALL-EFFECT SENSORS

UGN3503UA



NOTES: 1. Tolerances on package height and width represent allowable mold offsets. Dimensions given are measured at the widest point (parting line).

- 2. Exact body and lead configuration at vendor's option within limits shown.
- 3. Height does not include mold gate flash.
- 4. Minimum lead length was 0.500" (12.70 mm). If existing product to the original specifications is not acceptable, contact sales office before ordering.



LINEAR HALL-EFFECT SENSORS



Allegro MicroSystems, Inc. reserves the right to make, from time to time, such departures from the detail specifications as may be required to permit improvements in the design of its products.

The information included herein is believed to be accurate and reliable. However, Allegro MicroSystems, Inc. assumes no responsibility for its use; nor for any infringements of patents or other rights of third parties which may result from its use.





GENERAL PURPOSE SINGLE OPERATIONAL AMPLIFIER

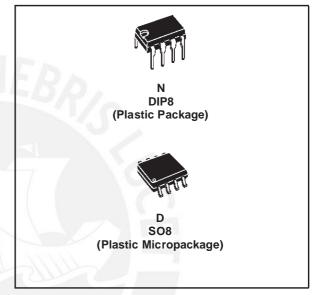
- LARGE INPUT VOLTAGE RANGE
- NO LATCH-UP
- HIGH GAIN
- SHORT-CIRCUIT PROTECTION
- NO FREQUENCY COMPENSATION
- REQUIRED
- SAME PIN CONFIGURATION AS THE UA709

DESCRIPTION

The UA741 is a high performance monolithic operational amplifier constructed on a single silicon chip. It is intented for a wide range of analog applications.

- Summing amplifier
- Voltage follower
- Integrator
- Active filter
- Function generator

The high gain and wide range of operating voltages provide superior performances in integrator, summing amplifier and general feedback applications. The internal compensation network (6dB/ octave) insures stability in closed loop circuits.

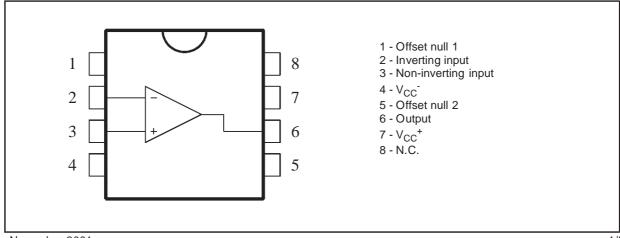


ORDER CODE

Part Number	Temperature Bange	Pack	age					
Part Number	Temperature Range	N	D					
UA741C	0°C, +70°C	•	•					
UA741I	-40°C, +105°C	•	•					
UA741M	-55°C, +125°C	•	•					
Example: UA7	Example: UA741CN							

 ${f N}$ = Dual in Line Package (DIP) ${f D}$ = Small Outline Package (SO) - also available in Tape & Reel (DT)

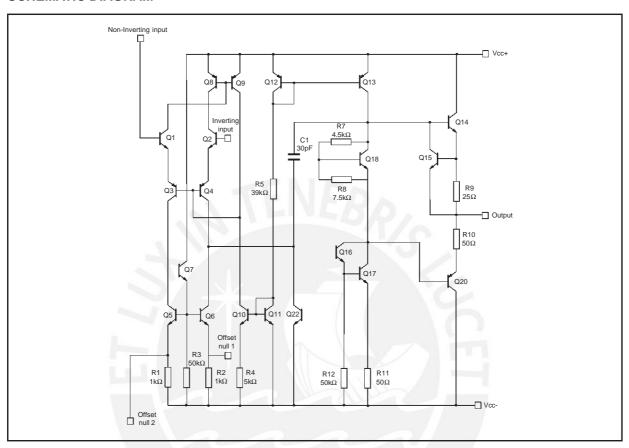
PIN CONNECTIONS (top view)



November 2001 1/5



SCHEMATIC DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	UA741M	UA741I	UA741C	Unit	
V _{CC}	Supply voltage		±22		V	
V _{id}	Differential Input Voltage		±30			
Vi	Input Voltage		±15			
P _{tot}	Power Dissipation 1)		500			
	Output Short-circuit Duration		Infinite			
T _{oper}	Operating Free-air Temperature Range	-55 to +125	-40 to +105	0 to +70	°C	
T _{stg}	Storage Temperature Range	-65 to +150			°C	

^{1.} Power dissipation must be considered to ensure maximum junction temperature (Tj) is not exceeded.



ELECTRICAL CHARACTERISTICS

 $V_{CC} = \pm 15V$, $T_{amb} = +25^{\circ}C$ (unless otherwise specified)

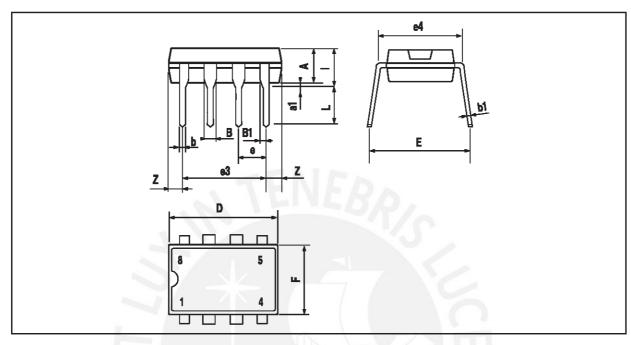
V_{io} Input Offset Voltage (R _s ≤ 10kΩ) $T_{amb} = +25^{\circ}C$ $T_{min} \le T_{amb} \le T_{max}$				1
		1	5 6	mV
$I_{io} \qquad \begin{array}{l} \text{Input Offset Current} \\ T_{amb} = +25^{\circ}\text{C} \\ T_{min} \leq T_{amb} \leq T_{max} \end{array}$		2	30 70	nA
$I_{ib} \begin{tabular}{ll} Input Bias Current \\ T_{amb} = +25^{\circ}C \\ T_{min} \leq T_{amb} \leq T_{max} \\ \end{tabular}$		10	100 200	nA
	50 25	200		V/mV
anno	77 77	90		dB
$I_{CC} \begin{tabular}{ll} Supply Current, no load \\ $T_{amb} = +25^{\circ}C$ \\ $T_{min} \le T_{amb} \le T_{max}$ \\ \end{tabular}$		1.7	2.8 3.3	mA
	:12			V
Gillio	70 70	90		dB
I _{OS} Output short Circuit Current 1	10	25	40	mA
$ \begin{vmatrix} \pm V_{opp} \\ T_{min} \le T_{amb} \le T_{max} \end{vmatrix} $	12 10 12 10	14 13		V
SR Slew Rate $V_i = \pm 10V$, $R_L = 2k\Omega$, $C_L = 100pF$, unity Gain 0.3	.25	0.5		V/μs
Rise Time $V_i = \pm 20 \text{mV}, R_L = 2 \text{k}\Omega, C_L = 100 \text{pF}, unity Gain}$		0.3		μs
K_{ov} Overshoot $V_i = 20$ mV, $R_L = 2$ kΩ, $C_L = 100$ pF, unity Gain		5		%
R _i Input Resistance 0	0.3	2		MΩ
$ \begin{array}{c c} & Gain \ Bandwith \ Product \\ & V_i = 10 mV, \ R_L = 2 k\Omega, \ C_L = 100 pF, \ f = 100 kHz \end{array} $).7	1		MHz
THD Total Harmonic Distortion $f = 1 \text{kHz}, A_v = 20 \text{dB}, R_L = 2 \text{k}\Omega, V_o = 2 \text{V}_{pp}, C_L = 100 \text{pF}, T_{amb} = +25 ^{\circ}\text{C}$		0.06		%
$\begin{array}{c} {\rm e_n} & {\rm Equivalent\ Input\ Noise\ Voltage} \\ {\rm f=1kHz,\ R_s=100\Omega} \end{array}$		23		n∨ √Hz
Øm Phase Margin	\dashv	50		Degrees





PACKAGE MECHANICAL DATA

8 PINS - PLASTIC DIP

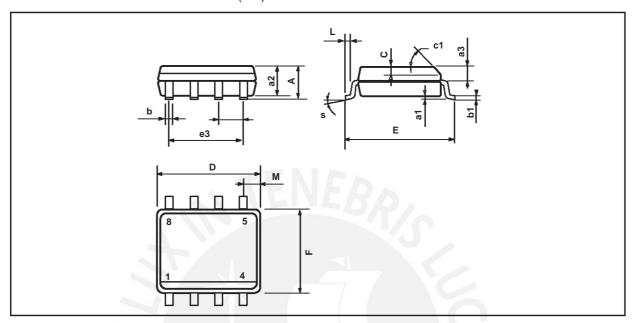


Dim	Millimeters				Inches	
Dim.	Min.	Тур.	Max.	Min.	Тур.	Max.
А		3.32			0.131	
a1	0.51		9 .00	0.020		
В	1.15		1.65	0.045	/	0.065
b	0.356		0.55	0.014		0.022
b1	0.204		0.304	0.008	/	0.012
D			10.92			0.430
Е	7.95	MA	9.75	0.313		0.384
е		2.54	AN X N	1	0.100	
e3		7.62			0.300	
e4		7.62			0.300	
F			6.6			0260
i			5.08			0.200
L	3.18		3.81	0.125		0.150
Z			1.52			0.060



PACKAGE MECHANICAL DATA

8 PINS - PLASTIC MICROPACKAGE (SO)



Dim		Millimeters			Inches		
Dim.	Min.	Тур.	Max.	Min.	Тур.	Max.	
А			1.75			0.069	
a1	0.1		0.25	0.004		0.010	
a2			1.65			0.065	
а3	0.65		0.85	0.026	1	0.033	
b	0.35		0.48	0.014		0.019	
b1	0.19		0.25	0.007	/	0.010	
С	0.25		0.5	0.010		0.020	
c1		MAA	45°	(typ.)	•	•	
D	4.8		5.0	0.189		0.197	
E	5.8		6.2	0.228		0.244	
е		1.27			0.050		
e3		3.81			0.150		
F	3.8		4.0	0.150		0.157	
L	0.4		1.27	0.016		0.050	
М			0.6			0.024	
S		•	8° (r	nax.)	•	•	

Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infring ement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of STMicroelectronics.

© The ST logo is a registered trademark of STMicroelectronics

© 2001 STMicroelectronics - Printed in Italy - All Rights Reserved STMicroelectronics GROUP OF COMPANIES

Australia - Brazil - Canada - China - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States © http://www.st.com





This datasheet has been download from:

 $\underline{www.datasheet catalog.com}$

Datasheets for electronics components.







SMALL SIGNAL NPN TRANSISTOR

PRELIMINARY DATA

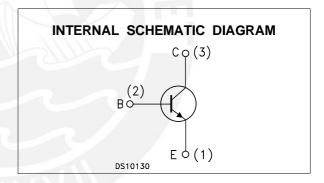
Туре	Marking
2N3904	2N3904

- SILICON EPITAXIAL PLANAR NPN TRANSISTOR
- TO-92 PACKAGE SUITABLE FOR THROUGH-HOLE PCB ASSEMBLY
- THE PNP COMPLEMENTARY TYPE IS 2N3906

APPLICATIONS

- WELL SUITABLE FOR TV AND HOME APPLIANCE EQUIPMENT
- SMALL LOAD SWITCH TRANSISTOR WITH HIGH GAIN AND LOW SATURATION VOLTAGE





ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{CBO}	Collector-Base Voltage (I _E = 0)	60	V
V_{CEO}	Collector-Emitter Voltage (I _B = 0)	40	V
V_{EBO}	Emitter-Base Voltage (I _C = 0)	6	V
Ic	Collector Current	200	mA
P _{tot}	Total Dissipation at T _C = 25 °C	625	mW
T _{stg}	Storage Temperature	-65 to 150	°C
Tj	Max. Operating Junction Temperature	150	°C

June 2002 1/4



THERMAL DATA

R _{thj-amb} •	Thermal Resistance Junction-Ambie	ent Max	200	°C/W	
R _{thj-case} •	Thermal Resistance Junction-Case	Max	83.3	°C/W	

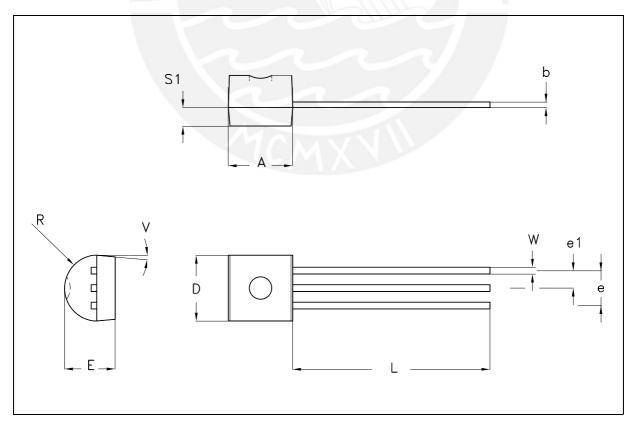
ELECTRICAL CHARACTERISTICS ($T_{case} = 25$ $^{\circ}C$ unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Тур.	Max.	Unit
I _{CEX}	Collector Cut-off Current (V _{BE} = -3 V)	V _{CE} = 30 V			50	nA
I _{BEX}	Base Cut-off Current (V _{BE} = -3 V)	V _{CE} = 30 V			50	nA
V _{(BR)CEO*}	Collector-Emitter Breakdown Voltage (I _B = 0)	I _C = 1 mA	40			V
V _{(BR)CBO}	Collector-Base Breakdown Voltage (I _E = 0)	I _C = 10 μA	60			V
V _{(BR)EBO}	Emitter-Base Breakdown Voltage (Ic = 0)	Ι _Ε = 10 μΑ	6			V
$V_{CE(sat)^*}$	Collector-Emitter Saturation Voltage	$\begin{array}{llllllllllllllllllllllllllllllllllll$			0.2 0.2	V V
V _{BE(sat)*}	Base-Emitter Saturation Voltage	I _C = 10 mA	0.65	7	0.85 0.95	V V
h _{FE} *	DC Current Gain	I _C = 0.1 mA	60 80 100 60 30		300	
f _T	Transition Frequency	$I_C = 10 \text{ mA } V_{CE} = 20 \text{ V } f = 100 \text{ MHz}$	250	270		MHz
Ссво	Collector-Base Capacitance	I _E = 0 V _{CB} = 10 V f = 1 MHz		4		pF
Сево	Emitter-Base Capacitance	$I_C = 0$ $V_{EB} = 0.5 \text{ V}$ $f = 1 \text{ MHz}$	7	18		pF
NF	Noise Figure	$V_{CE} = 5$ V $I_{C} = 0.1$ mA $f = 10$ Hz to 15.7 KHz $R_{G} = 1$ K Ω		5		dB
t _d t _r	Delay Time Rise Time	$I_C = 10 \text{ mA}$ $I_B = 1 \text{ mA}$ $V_{CC} = 30 \text{ V}$			35 35	ns ns
t _s	Storage Time Fall Time	$I_{C} = 10 \text{ mA}$ $I_{B1} = -I_{B2} = 1 \text{ mA}$ $V_{CC} = 30 \text{ V}$			200 50	ns ns

^{*} Pulsed: Pulse duration = 300 μs, duty cycle ≤ 2 %

TO-92 MECHANICAL DATA

DIM.		mm			inch		
Diwi.	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.	
А	4.32		4.95	0.170		0.195	
b	0.36		0.51	0.014		0.020	
D	4.45		4.95	0.175		0.194	
E	3.30		3.94	0.130		0.155	
е	2.41	-	2.67	0.095		0.105	
e1	1.14	1 10.	1.40	0.045		0.055	
L	12.70		15.49	0.500		0.609	
R	2.16		2.41	0.085		0.094	
S1	1.14	/ - 	1.52	0.045		0.059	
W	0.41		0.56	0.016		0.022	
V	4 degree		6 degree	4 degree		6 degree	







Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specification mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of STMicroelectronics.

The ST logo is a trademark of STMicroelectronics

© 2002 STMicroelectronics – Printed in Italy – All Rights Reserved STMicroelectronics GROUP OF COMPANIES

Australia - Brazil - Canada - China - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States.

http://www.st.com





This datasheet has been download from:

 $\underline{www.datasheet catalog.com}$

Datasheets for electronics components.







November 1995

2N7000 / 2N7002 / NDS7002A N-Channel Enhancement Mode Field Effect Transistor

General Description

These N-Channel enhancement mode field effect transistors are produced using Fairchild's proprietary, high cell density, DMOS technology. These products have been designed to minimize on-state resistance while provide rugged, reliable, and fast switching performance. They can be used in most applications requiring up to 400mA DC and can deliver pulsed currents up to 2A. These products are particularly suited for low voltage, low current applications such as small servo motor control, power MOSFET gate drivers, and other switching applications.

Features

- High density cell design for low R_{DS(ON)}.
- Voltage controlled small signal switch.
- Rugged and reliable.
- High saturation current capability.



Absolute Maximum Ratings T_A = 25°C unless otherwise noted

Symbol	Parameter	2N7000	2N7002	NDS7002A	Units
V _{DSS}	Drain-Source Voltage		60		V
V_{DGR}	Drain-Gate Voltage ($R_{GS} \le 1 M\Omega$)		60		V
V_{GSS}	Gate-Source Voltage - Continuous		±20		V
	- Non Repetitive (tp < 50μs)		±40		7
I _D	Maximum Drain Current - Continuous	200	115	280	mA
	- Pulsed	500	800	1500	
P_{D}	Maximum Power Dissipation	400	200	300	mW
	Derated above 25°C	3.2	1.6	2.4	mW/°C
T_J, T_{STG}	Operating and Storage Temperature Range	-55	to 150	-65 to 150	°C
T _L	Maximum Lead Temperature for Soldering Purposes, 1/16" from Case for 10 Seconds	300			
THERMA	L CHARACTERISTICS				
R _{θJA}	Thermal Resistance, Junction-to-Ambient	312.5	625	417	°C/W

© 1997 Fairchild Semiconductor Corporation



Symbol	Parameter	Conditions		Туре	Min	Тур	Max	Units
OFF CHA	ARACTERISTICS							
BV _{DSS}	Drain-Source Breakdown Voltage	$V_{GS} = 0 \text{ V}, I_{D} = 10 \mu\text{A}$		All	60			V
I _{DSS}	Zero Gate Voltage Drain Current	$V_{DS} = 48 \text{ V}, V_{GS} = 0 \text{ V}$		2N7000			1	μΑ
			T _J =125°C				1	mA
		$V_{DS} = 60 \text{ V}, V_{GS} = 0 \text{ V}$		2N7002			1	μΑ
			T _J =125°C	NDS7002A			0.5	mA
I_{GSSF}	Gate - Body Leakage, Forward	$V_{GS} = 15 \text{ V}, V_{DS} = 0 \text{ V}$		2N7000			10	nA
		$V_{GS} = 20 \text{ V}, V_{DS} = 0 \text{ V}$		2N7002 NDS7002A			100	nA
$I_{\rm GSSR}$	Gate - Body Leakage, Reverse	$V_{GS} = -15 \text{ V}, V_{DS} = 0 \text{ V}$		2N7000			-10	nA
		$V_{GS} = -20 \text{ V}, V_{DS} = 0 \text{ V}$		2N7002 NDS7002A			-100	nA
ON CHAI	RACTERISTICS (Note 1)	-CALL		, ,		ı		
$V_{\text{GS(th)}}$	Gate Threshold Voltage	$V_{DS} = V_{GS}, I_{D} = 1 \text{ mA}$	$K\Delta$	2N7000	8.0	2.1	3	V
	. 11	$V_{DS} = V_{GS}, I_{D} = 250 \mu A$		2N7002 NDS7002A	1	2.1	2.5	
$R_{DS(ON)}$	Static Drain-Source On-Resistance	$V_{GS} = 10 \text{ V}, I_D = 500 \text{ mA}$		2N7000		1.2	5	Ω
			T _J =125°C			1.9	9	
		$V_{GS} = 4.5 \text{ V}, I_{D} = 75 \text{ mA}$	7 A	AC		1.8	5.3	
		$V_{GS} = 10 \text{ V}, I_{D} = 500 \text{ mA}$		2N7002		1.2	7.5	
			T _J =100°C		- 17	1.7	13.5	
		$V_{GS} = 5.0 \text{ V}, I_{D} = 50 \text{ mA}$				1.7	7.5	
			$T_{J} = 100C$			2.4	13.5	
		$V_{GS} = 10 \text{ V}, I_{D} = 500 \text{ mA}$	7////	NDS7002A		1.2	2	
			T _J =125°C	7/		2	3.5	
		$V_{GS} = 5.0 \text{ V}, I_{D} = 50 \text{ mA}$		5//		1.7	3	
			T _J =125°C	×//		2.8	5	
$V_{DS(ON)}$	Drain-Source On-Voltage	$V_{GS} = 10 \text{ V}, I_{D} = 500 \text{ mA}$		2N7000		0.6	2.5	V
		$V_{GS} = 4.5 \text{ V}, I_{D} = 75 \text{ mA}$				0.14	0.4	
		$V_{GS} = 10 \text{ V}, I_{D} = 500 \text{mA}$		2N7002		0.6	3.75	
		$V_{GS} = 5.0 \text{ V}, I_{D} = 50 \text{ mA}$	1 11			0.09	1.5	
		$V_{GS} = 10 \text{ V}, I_{D} = 500 \text{mA}$		NDS7002A		0.6	1	
		$V_{GS} = 5.0 \text{ V}, I_{D} = 50 \text{ mA}$				0.09	0.15	



Symbol	Parameter	Conditions	Туре	Min	Тур	Max	Units
ON CHAP	RACTERISTICS Continued (Note 1)						
I _{D(ON)}	On-State Drain Current	$V_{GS} = 4.5 \text{ V}, \ V_{DS} = 10 \text{ V}$	2N7000	75	600		mA
		$V_{GS} = 10 \text{ V}, V_{DS} \ge 2 V_{DS(on)}$	2N7002	500	2700		
		$V_{GS} = 10 \text{ V}, V_{DS} \ge 2 V_{DS(on)}$	NDS7002A	500	2700		
g _{FS}	Forward Transconductance	$V_{DS} = 10 \text{ V}, I_{D} = 200 \text{ mA}$	2N7000	100	320		mS
		$V_{DS} \ge 2 V_{DS(on)}$, $I_D = 200 \text{ mA}$	2N7002	80	320		
		$V_{DS} \ge 2 V_{DS(on)}, I_D = 200 \text{ mA}$	NDS7002A	80	320		
DYNAMIC	CHARACTERISTICS						
C _{iss}	Input Capacitance	$V_{DS} = 25 \text{ V}, \ V_{GS} = 0 \text{ V}, $ f = 1.0 MHz	All		20	50	pF
C _{oss}	Output Capacitance	f = 1.0 MHz	All		11	25	pF
C _{rss}	Reverse Transfer Capacitance		All		4	5	pF
t _{on}	Turn-On Time	$V_{DD} = 15 \text{ V}, R_{L} = 25 \Omega,$ $I_{D} = 500 \text{ mA}, V_{GS} = 10 \text{ V},$ $R_{GEN} = 25$	2N7000			10	ns
	4	$\begin{aligned} & V_{\text{DD}} = 30 \text{ V}, \text{ R}_{\text{L}} = 150 \Omega, \\ & I_{\text{D}} = 200 \text{ mA}, \text{ V}_{\text{GS}} = 10 \text{ V}, \\ & R_{\text{GEN}} = 25 \Omega \end{aligned}$	2N7002 NDS7002A			20	
t _{off}	Turn-Off Time	$V_{DD} = 15 \text{ V}, R_{L} = 25 \Omega,$ $I_{D} = 500 \text{ mA}, V_{GS} = 10 \text{ V},$ $R_{GEN} = 25$	2N7000			10	ns
	H	$\begin{aligned} & V_{\text{DD}} = 30 \text{ V}, \text{ R}_{\text{L}} = 150 \Omega, \\ & I_{\text{D}} = 200 \text{ mA}, V_{\text{GS}} = 10 \text{ V}, \\ & R_{\text{GEN}} = 25 \Omega \end{aligned}$	2N7002 NDS7002A	7		20	
DRAIN-S	OURCE DIODE CHARACTERISTICS	S AND MAXIMUM RATINGS					
I _s	Maximum Continuous Drain-Sour	ce Diode Forward Current	2N7002			115	mA
			NDS7002A			280	
I _{SM}	Maximum Pulsed Drain-Source D	iode Forward Current	2N7002	7		0.8	Α
			NDS7002A	1		1.5	
V _{SD}	Drain-Source Diode Forward	V _{GS} = 0 V, I _S = 115 mA (Note 1)	2N7002		0.88	1.5	V
	Voltage	$V_{GS} = 0 \text{ V}, I_{S} = 400 \text{ mA} \text{ (Note 1)}$	NDS7002A		0.88	1.2	1

Note:
1. Pulse Test: Pulse Width ≤ 300µs, Duty Cycle ≤ 2.0%.



Typical Electrical Characteristics

2N7000 / 2N7002 / NDS7002A

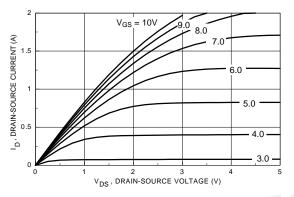


Figure 1. On-Region Characteristics

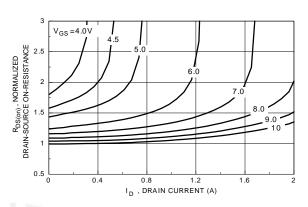


Figure 2. On-Resistance Variation with Gate Voltage and Drain Current

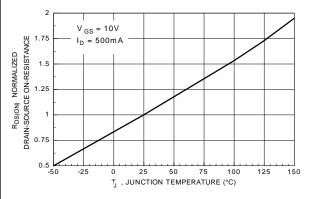


Figure 3. On-Resistance Variation with Temperature

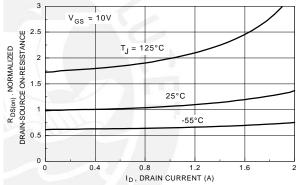


Figure 4. On-Resistance Variation with Drain Current and Temperature

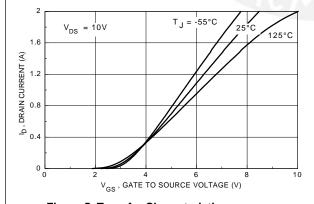


Figure 5. Transfer Characteristics

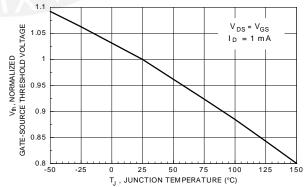
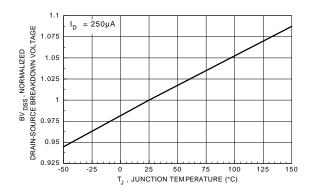


Figure 6. Gate Threshold Variation with Temperature



Typical Electrical Characteristics (continued)

2N7000 / 2N7002 /NDS7002A



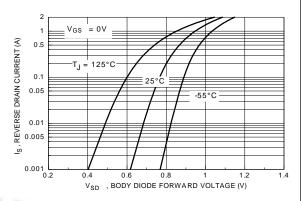
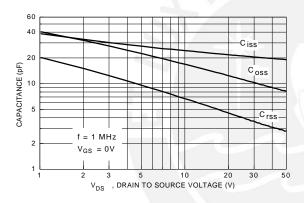


Figure 7. Breakdown Voltage Variation with Temperature

Figure 8. Body Diode Forward Voltage Variation with



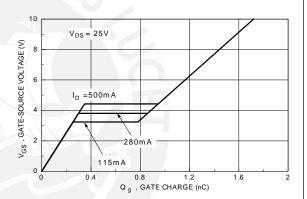
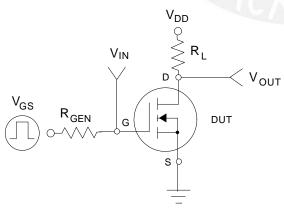


Figure 9. Capacitance Characteristics

Figure 10. Gate Charge Characteristics



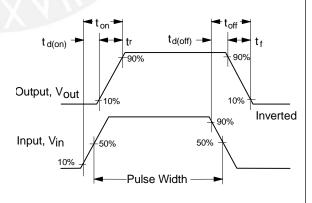
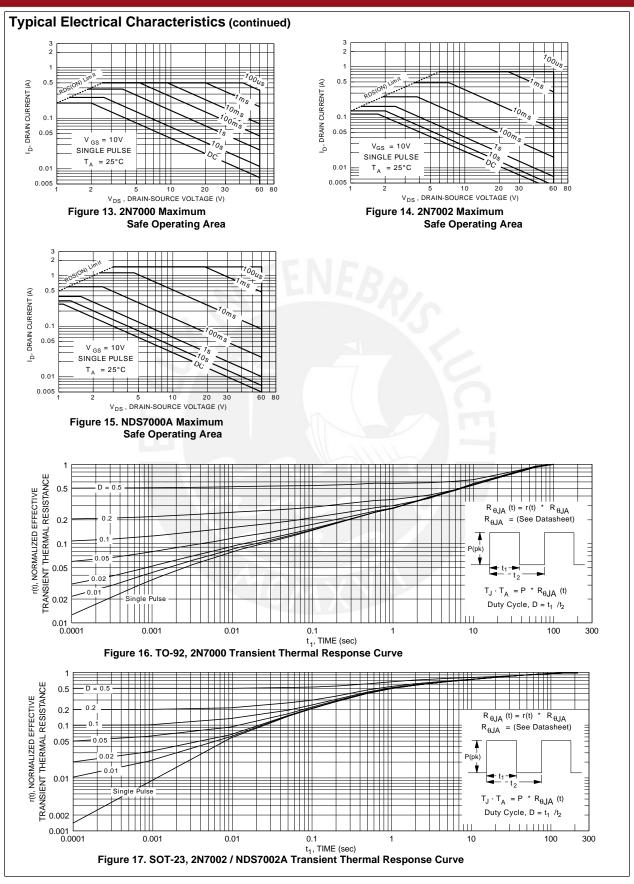


Figure 11.

Figure 12. Switching Waveforms







TRADEMARKS

The following are registered and unregistered trademarks Fairchild Semiconductor owns or is authorized to use and is not intended to be an exhaustive list of all such trademarks.

 $ACEx^{TM}$ FASTr™ PowerTrench® SyncFET™ Bottomless™ QFET™ TinyLogic™ GlobalOptoisolator™ QSTM UHC™ CoolFET™ **GTO™ VCX**TM $CROSSVOLT^{\mathsf{TM}}$ QT Optoelectronics™ HiSeC™

DOME™ ISOPLANAR™ Quiet Series™

DISCLAIMER

FAIRCHILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO IMPROVE RELIABILITY, FUNCTION OR DESIGN. FAIRCHILD DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS.

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF FAIRCHILD SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, or (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in significant injury to the user.

 A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

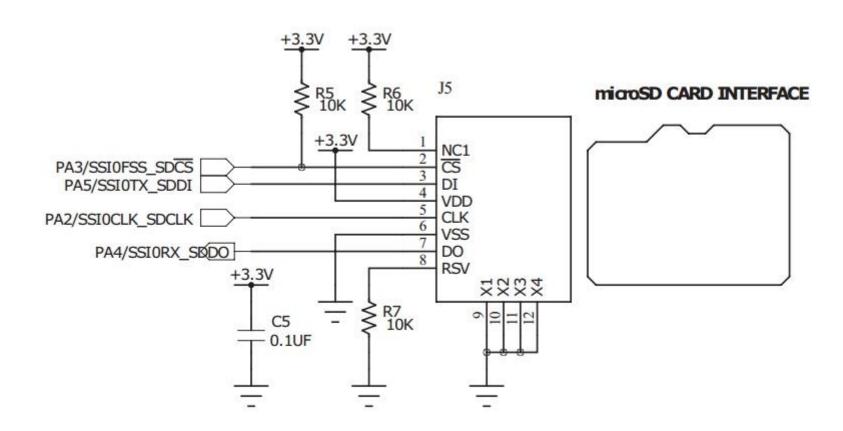
PRODUCT STATUS DEFINITIONS

Definition of Terms

Datasheet Identification	Product Status	Definition
Advance Information	Formative or In Design	This datasheet contains the design specifications for product development. Specifications may change in any manner without notice.
Preliminary	First Production	This datasheet contains preliminary data, and supplementary data will be published at a later date. Fairchild Semiconductor reserves the right to make changes at any time without notice in order to improve design.
No Identification Needed	Full Production	This datasheet contains final specifications. Fairchild Semiconductor reserves the right to make changes at any time without notice in order to improve design.
Obsolete	Not In Production	This datasheet contains specifications on a product that has been discontinued by Fairchild semiconductor. The datasheet is printed for reference information only.

Rev. G







Instruction Set Nomenclature

Status Register (SREG)

SREG: Status Register
C: Carry Flag
Z: Zero Flag
N: Negative Flag

V: Two's complement overflow indicator

S: $N \oplus V$, For signed tests

H: Half Carry Flag

T: Transfer bit used by BLD and BST instructions

I: Global Interrupt Enable/Disable Flag

Registers and Operands

Rd: Destination (and source) register in the Register File

Rr: Source register in the Register FileR: Result after instruction is executed

K: Constant datak: Constant address

b: Bit in the Register File or I/O Register (3-bit)

s: Bit in the Status Register (3-bit)

X,Y,Z: Indirect Address Register

(X=R27:R26, Y=R29:R28 and Z=R31:R30)

A: I/O location address

q: Displacement for direct addressing (6-bit)



8-bit **AVR**® Instruction Set

Rev. 0856I-AVR-07/10







I/O Registers

RAMPX, RAMPY, RAMPZ

Registers concatenated with the X-, Y-, and Z-registers enabling indirect addressing of the whole data space on MCUs with more than 64K bytes data space, and constant data fetch on MCUs with more than 64K bytes program space.

RAMPD

Register concatenated with the Z-register enabling direct addressing of the whole data space on MCUs with more than 64K bytes data space.

EIND

Register concatenated with the Z-register enabling indirect jump and call to the whole program space on MCUs with more than 64K words (128K bytes) program space.

Stack

STACK: Stack for return address and pushed registers

SP: Stack Pointer to STACK

Flags

⇔: Flag affected by instruction0: Flag cleared by instruction1: Flag set by instruction

-: Flag not affected by instruction

2 AVR Instruction Set



AVR Instruction Set

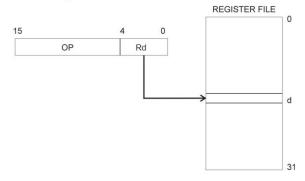
The Program and Data Addressing Modes

The AVR Enhanced RISC microcontroller supports powerful and efficient addressing modes for access to the Program memory (Flash) and Data memory (SRAM, Register file, I/O Memory, and Extended I/O Memory). This section describes the various addressing modes supported by the AVR architecture. In the following figures, OP means the operation code part of the instruction word. To simplify, not all figures show the exact location of the addressing bits. To generalize, the abstract terms RAMEND and FLASHEND have been used to represent the highest location in data and program space, respectively.

Note: Not all addressing modes are present in all devices. Refer to the device spesific instruction summary.

Register Direct, Single Register Rd

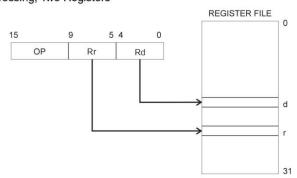
Figure 1. Direct Single Register Addressing



The operand is contained in register d (Rd).

Register Direct, Two Registers Rd and Rr

Figure 2. Direct Register Addressing, Two Registers



Operands are contained in register r (Rr) and d (Rd). The result is stored in register d (Rd).



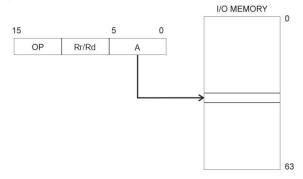
3





I/O Direct

Figure 3. I/O Direct Addressing

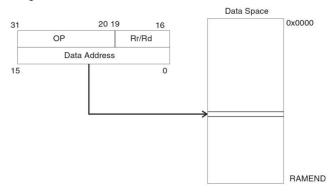


Operand address is contained in 6 bits of the instruction word. n is the destination or source register address.

Note: Some complex AVR Microcontrollers have more peripheral units than can be supported within the 64 locations reserved in the opcode for I/O direct addressing. The extended I/O memory from address 64 to 255 can only be reached by data addressing, not I/O addressing.

Data Direct

Figure 4. Direct Data Addressing



A 16-bit Data Address is contained in the 16 LSBs of a two-word instruction. Rd/Rr specify the destination or source register.

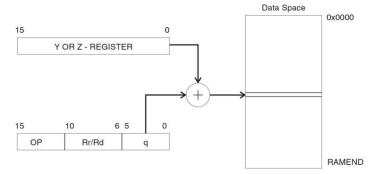
4 AVR Instruction Set



AVR Instruction Set

Data Indirect with Displacement

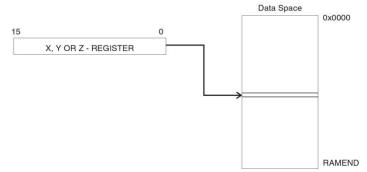
Figure 5. Data Indirect with Displacement



Operand address is the result of the Y- or Z-register contents added to the address contained in 6 bits of the instruction word. Rd/Rr specify the destination or source register.

Data Indirect

Figure 6. Data Indirect Addressing



Operand address is the contents of the X-, Y-, or the Z-register. In AVR devices without SRAM, Data Indirect Addressing is called Register Indirect Addressing. Register Indirect Addressing is a subset of Data Indirect Addressing since the data space form 0 to 31 is the Register File.

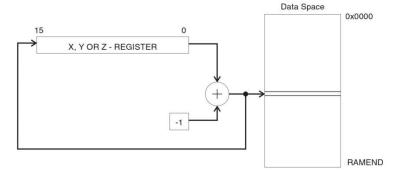






Data Indirect with Pre-decrement

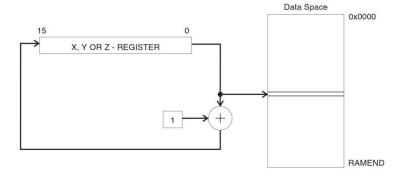
Figure 7. Data Indirect Addressing with Pre-decrement



The X,- Y-, or the Z-register is decremented before the operation. Operand address is the decremented contents of the X-, Y-, or the Z-register.

Data Indirect with Post-increment

Figure 8. Data Indirect Addressing with Post-increment



The X-, Y-, or the Z-register is incremented after the operation. Operand address is the content of the X-, Y-, or the Z-register prior to incrementing.

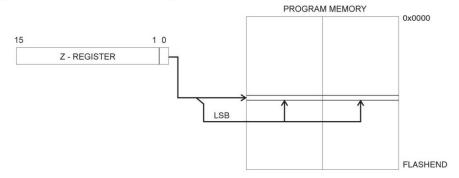
6 AVR Instruction Set _____



AVR Instruction Set

Program Memory Constant Addressing using the LPM, ELPM, and SPM Instructions

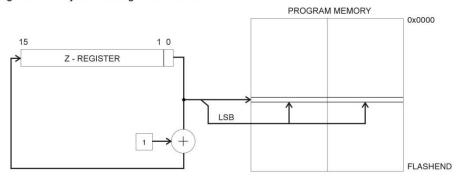
Figure 9. Program Memory Constant Addressing



Constant byte address is specified by the Z-register contents. The 15 MSBs select word address. For LPM, the LSB selects low byte if cleared (LSB = 0) or high byte if set (LSB = 1). For SPM, the LSB should be cleared. If ELPM is used, the RAMPZ Register is used to extend the Z-register.

Program Memory with Post-increment using the LPM Z+ and ELPM Z+ Instruction

Figure 10. Program Memory Addressing with Post-increment



Constant byte address is specified by the Z-register contents. The 15 MSBs select word address. The LSB selects low byte if cleared (LSB = 0) or high byte if set (LSB = 1). If ELPM Z+ is used, the RAMPZ Register is used to extend the Z-register.

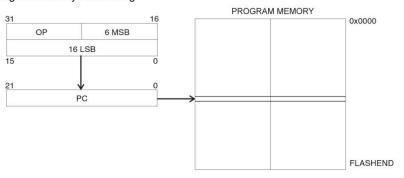






Direct Program Addressing, JMP and CALL

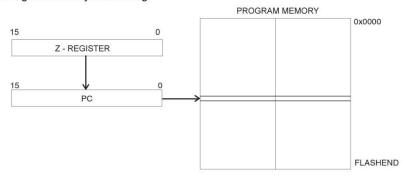
Figure 11. Direct Program Memory Addressing



Program execution continues at the address immediate in the instruction word.

Indirect Program Addressing, IJMP and ICALL

Figure 12. Indirect Program Memory Addressing



Program execution continues at address contained by the Z-register (i.e., the PC is loaded with the contents of the Z-register).

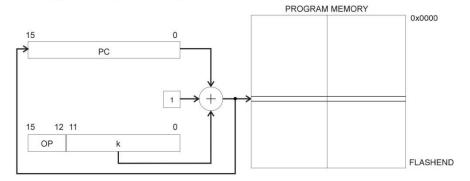
8 AVR Instruction Set



■ AVR Instruction Set

Relative Program Addressing, RJMP and RCALL

Figure 13. Relative Program Memory Addressing



Program execution continues at address PC + k + 1. The relative address k is from -2048 to 2047.







Conditional Branch Summary

Test	Boolean	Mnemonic	Complementary	Boolean	Mnemonic	Comment
Rd > Rr	Z•(N ⊕ V) = 0	BRLT ⁽¹⁾	Rd ≤ Rr	Z+(N ⊕ V) = 1	BRGE*	Signed
Rd □ Rr	(N ⊕ V) = 0	BRGE	Rd < Rr	(N ⊕ V) = 1	BRLT	Signed
Rd = Rr	Z = 1	BREQ	Rd ≠ Rr	Z = 0	BRNE	Signed
Rd ≤ Rr	Z+(N ⊕ V) = 1	BRGE ⁽¹⁾	Rd > Rr	Z•(N ⊕ V) = 0	BRLT*	Signed
Rd < Rr	(N ⊕ V) = 1	BRLT	Rd ≥ Rr	(N ⊕ V) = 0	BRGE	Signed
Rd > Rr	C + Z = 0	BRLO ⁽¹⁾	Rd ≤ Rr	C + Z = 1	BRSH*	Unsigned
Rd □ Rr	C = 0	BRSH/BRCC	Rd < Rr	C = 1	BRLO/BRCS	Unsigned
Rd = Rr	Z = 1	BREQ	Rd ≠ Rr	Z = 0	BRNE	Unsigned
Rd ≤ Rr	C + Z = 1	BRSH ⁽¹⁾	Rd > Rr	C + Z = 0	BRLO*	Unsigned
Rd < Rr	C = 1	BRLO/BRCS	Rd ≥ Rr	C = 0	BRSH/BRCC	Unsigned
Carry	C = 1	BRCS	No carry	C = 0	BRCC	Simple
Negative	N = 1	BRMI	Positive	N = 0	BRPL	Simple
Overflow	V = 1	BRVS	No overflow	V = 0	BRVC	Simple
Zero	Z = 1	BREQ	Not zero	Z = 0	BRNE	Simple

Note: 1. Interchange Rd and Rr in the operation before the test, i.e., CP Rd, Rr \rightarrow CP Rr, Rd



■ AVR Instruction Set

Complete Instruction Set Summary

Instruction Set Summary

Mnemonics	Operands	Description	Opera	ation		Flags	#Clocks	#Clocks XMEGA
		Arith	nmetic and Logic Instructions	6				
ADD	Rd, Rr	Add without Carry	Rd	←	Rd + Rr	Z,C,N,V,S,H	1	
ADC	Rd, Rr	Add with Carry	Rd	←	Rd + Rr + C	Z,C,N,V,S,H	1	
ADIW ⁽¹⁾	Rd, K	Add Immediate to Word	Rd	←	Rd + 1:Rd + K	Z,C,N,V,S	2	
SUB	Rd, Rr	Subtract without Carry	Rd	←	Rd - Rr	Z,C,N,V,S,H	1	
SUBI	Rd, K	Subtract Immediate	Rd	←	Rd - K	Z,C,N,V,S,H	1	
SBC	Rd, Rr	Subtract with Carry	Rd	←	Rd - Rr - C	Z,C,N,V,S,H	1	
SBCI	Rd, K	Subtract Immediate with Carry	Rd	←	Rd - K - C	Z,C,N,V,S,H	1	
SBIW ⁽¹⁾	Rd, K	Subtract Immediate from Word	Rd + 1:Rd	←	Rd + 1:Rd - K	Z,C,N,V,S	2	
AND	Rd, Rr	Logical AND	Rd	←	Rd • Rr	Z,N,V,S	1	
ANDI	Rd, K	Logical AND with Immediate	Rd	←	Rd • K	Z,N,V,S	1	
OR	Rd, Rr	Logical OR	Rd	←	Rd v Rr	Z,N,V,S	1	
ORI	Rd, K	Logical OR with Immediate	Rd	←	Rd v K	Z,N,V,S	1	
EOR	Rd, Rr	Exclusive OR	Rd	←	Rd ⊕ Rr	Z,N,V,S	1	
СОМ	Rd	One's Complement	Rd	←	\$FF - Rd	Z,C,N,V,S	1	
NEG	Rd	Two's Complement	Rd	←	\$00 - Rd	Z,C,N,V,S,H	1	
SBR	Rd,K	Set Bit(s) in Register	Rd	←	Rd v K	Z,N,V,S	1	
CBR	Rd,K	Clear Bit(s) in Register	Rd	←	Rd • (\$FFh - K)	Z,N,V,S	1	
INC	Rd	Increment	Rd	←	Rd + 1	Z,N,V,S	1	
DEC	Rd	Decrement	Rd	←	Rd - 1	Z,N,V,S	1	
TST	Rd	Test for Zero or Minus	Rd	←	Rd • Rd	Z,N,V,S	1	
CLR	Rd	Clear Register	Rd	←	Rd ⊕ Rd	Z,N,V,S	1	
SER	Rd	Set Register	Rd	←	\$FF	None	1	
MUL ⁽¹⁾	Rd,Rr	Multiply Unsigned	R1:R0	←	Rd x Rr (UU)	Z,C	2	
MULS ⁽¹⁾	Rd,Rr	Multiply Signed	R1:R0	←	Rd x Rr (SS)	Z,C	2	
MULSU ⁽¹⁾	Rd,Rr	Multiply Signed with Unsigned	R1:R0	←	Rd x Rr (SU)	Z,C	2	
FMUL ⁽¹⁾	Rd,Rr	Fractional Multiply Unsigned	R1:R0	←	Rd x Rr<<1 (UU)	Z,C	2	
FMULS ⁽¹⁾	Rd,Rr	Fractional Multiply Signed	R1:R0	←	Rd x Rr<<1 (SS)	Z,C	2	
FMULSU ⁽¹⁾	Rd,Rr	Fractional Multiply Signed with Unsigned	R1:R0	←	Rd x Rr<<1 (SU)	Z,C	2	
DES	К	Data Encryption	if (H = 0) then R15:R0 else if (H = 1) then R15:R0	←	Encrypt(R15:R0, K) Decrypt(R15:R0, K)			1/2
		Bra	nch Instructions					
RJMP	k	Relative Jump	PC	←	PC + k + 1	None	2	
IJMP ⁽¹⁾		Indirect Jump to (Z)	PC(15:0) PC(21:16)	←	Z, 0	None	2	
EIJMP ⁽¹⁾		Extended Indirect Jump to (Z)	PC(15:0) PC(21:16)	←	Z, EIND	None	2	
JMP ⁽¹⁾	k	Jump	PC	←	k	None	3	



11





Mnemonics	Operands	Description	Opera	ation		Flags	#Clocks	#Clocks XMEGA
RCALL	k	Relative Call Subroutine	PC	←	PC + k + 1	None	3 / 4 ⁽³⁾⁽⁵⁾	2/3(3)
ICALL ⁽¹⁾		Indirect Call to (Z)	PC(15:0) PC(21:16)	← ←	Z, 0	None	3 / 4 ⁽³⁾	2/3(3)
EICALL ⁽¹⁾		Extended Indirect Call to (Z)	PC(15:0) PC(21:16)	←	Z, EIND	None	4 (3)	3 (3)
CALL ⁽¹⁾	k	call Subroutine	PC	←	k	None	4 / 5(3)	3 / 4 ⁽³⁾
RET		Subroutine Return	PC	←	STACK	None	4 / 5 ⁽³⁾	
RETI		Interrupt Return	PC	←	STACK	Ĭ	4 / 5(3)	
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC	←	PC + 2 or 3	None	1/2/3	
CP	Rd,Rr	Compare	Rd - Rr			Z,C,N,V,S,H	1	
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C			Z,C,N,V,S,H	1	
CPI	Rd,K	Compare with Immediate	Rd - K			Z,C,N,V,S,H	1	
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b) = 0) PC	←	PC + 2 or 3	None	1/2/3	
SBRS	Rr, b	Skip if Bit in Register Set	if (Rr(b) = 1) PC	←	PC + 2 or 3	None	1/2/3	
SBIC	A, b	Skip if Bit in I/O Register Cleared	if (I/O(A,b) = 0) PC	←	PC + 2 or 3	None	1/2/3	2/3/4
SBIS	A, b	Skip if Bit in I/O Register Set	If (I/O(A,b) =1) PC	←	PC + 2 or 3	None	1/2/3	2/3/4
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC	←	PC + k + 1	None	1/2	
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC	←	PC + k + 1	None	1/2	
BREQ	k	Branch if Equal	if (Z = 1) then PC	←	PC + k + 1	None	1/2	
BRNE	k	Branch if Not Equal	if (Z = 0) then PC	←	PC + k + 1	None	1/2	
BRCS	k	Branch if Carry Set	if (C = 1) then PC	←	PC + k + 1	None	1/2	
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC	←	PC + k + 1	None	1/2	
BRSH	k	Branch if Same or Higher	if (C = 0) then PC	←	PC + k + 1	None	1/2	
BRLO	k	Branch if Lower	if (C = 1) then PC	←	PC + k + 1	None	1/2	
BRMI	k	Branch if Minus	if (N = 1) then PC	←	PC + k + 1	None	1/2	
BRPL	k	Branch if Plus	if (N = 0) then PC	←	PC + k + 1	None	1/2	
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC	←	PC + k + 1	None	1/2	
BRLT	k	Branch if Less Than, Signed	if (N ⊕ V= 1) then PC	←	PC + k + 1	None	1/2	
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC	←	PC + k + 1	None	1/2	
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC	←	PC + k + 1	None	1/2	
BRTS	k	Branch if T Flag Set	if (T = 1) then PC	←	PC + k + 1	None	1/2	
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC	←	PC + k + 1	None	1/2	
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC	←	PC + k + 1	None	1/2	
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC	←	PC + k + 1	None	1/2	
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC	←	PC + k + 1	None	1/2	
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC	←	PC + k + 1	None	1/2	
			ransfer Instructions		All long			-
MOV	Rd, Rr	Copy Register	Rd	←	Rr	None	1	
MOVW ⁽¹⁾	Rd, Rr	Copy Register Pair	Rd+1:Rd	←	Rr+1:Rr	None	1	
LDI	Rd, K	Load Immediate	Rd	←	К	None	1	
LDS ⁽¹⁾	Rd, k	Load Direct from data space	Rd	←	(k)	None	1 ⁽⁵⁾ /2 ⁽³⁾	2(3)(4)
LD ⁽²⁾	Rd, X	Load Indirect	Rd	←	(X)	None	1(5)2(3)	1(3)(4)

12 AVR Instruction Set



■ AVR Instruction Set

Mnemonics	Operands	Description	Opera	ation		Flags	#Clocks	#Clocks
LD ⁽²⁾	Rd, X+	Load Indirect and Post-Increment	Rd X	←	(X) X + 1	None	2 ⁽³⁾	1 ⁽³⁾⁽⁴⁾
LD ⁽²⁾	Rd, -X	Load Indirect and Pre-Decrement	$X \leftarrow X - 1$, $Rd \leftarrow (X)$	←	X - 1 (X)	None	2 ⁽³⁾ /3 ⁽⁵⁾	2(3)(4)
LD ⁽²⁾	Rd, Y	Load Indirect	$Rd \leftarrow (Y)$	←	(Y)	None	1 ⁽⁵⁾ /2 ⁽³⁾	1(3)(4)
LD ⁽²⁾	Rd, Y+	Load Indirect and Post-Increment	Rd Y	←	(Y) Y + 1	None	2 ⁽³⁾	1(3)(4)
LD ⁽²⁾	Rd, -Y	Load Indirect and Pre-Decrement	Y Rd	←	Y - 1 (Y)	None	2(3)/3(5)	2(3)(4)
LDD ⁽¹⁾	Rd, Y+q	Load Indirect with Displacement	Rd	←	(Y + q)	None	2(3)	2(3)(4)
LD ⁽²⁾	Rd, Z	Load Indirect	Rd	←	(Z)	None	1 ⁽⁵⁾ /2 ⁽³⁾	1(3)(4)
LD ⁽²⁾	Rd, Z+	Load Indirect and Post-Increment	Rd Z	←	(Z), Z+1	None	2 ⁽³⁾	1(3)(4)
LD ⁽²⁾	Rd, -Z	Load Indirect and Pre-Decrement	Z Rd	←	Z - 1, (Z)	None	2 ⁽³⁾ /3 ⁽⁵⁾	2(3)(4)
LDD ⁽¹⁾	Rd, Z+q	Load Indirect with Displacement	Rd	←	(Z + q)	None	2(3)	2(3)(4)
STS ⁽¹⁾	k, Rr	Store Direct to Data Space	(k)	←	Rd	None	1 ⁽⁵⁾ /2 ⁽³⁾	2 ⁽³⁾
ST ⁽²⁾	X, Rr	Store Indirect	(X)	←	Rr	None	1(5)/2(3)	1 ⁽³⁾
ST ⁽²⁾	X+, Rr	Store Indirect and Post-Increment	(X) X	←	Rr, X + 1	None	1 ⁽⁵⁾ /2 ⁽³⁾	1 ⁽³⁾
ST ⁽²⁾	-X, Rr	Store Indirect and Pre-Decrement	X (X)	←	X - 1, Rr	None	2 ⁽³⁾	2 ⁽³⁾
ST ⁽²⁾	Y, Rr	Store Indirect	(Y)	←	Rr	None	1(5)/2(3)	1 ⁽³⁾
ST ⁽²⁾	Y+, Rr	Store Indirect and Post-Increment	(Y)	←	Rr, Y + 1	None	1 ⁽⁵⁾ /2 ⁽³⁾	1 ⁽³⁾
ST ⁽²⁾	-Y, Rr	Store Indirect and Pre-Decrement	(Y)	←	Y - 1, Rr	None	2 ⁽³⁾	2 ⁽³⁾
STD ⁽¹⁾	Y+q, Rr	Store Indirect with Displacement	(Y + q)	←	Rr	None	2(3)	2(3)
ST ⁽²⁾	Z, Rr	Store Indirect	(Z)	←	Rr	None	1 ⁽⁵⁾ /2 ⁽³⁾	1 ⁽³⁾
ST ⁽²⁾	Z+, Rr	Store Indirect and Post-Increment	(Z) Z	←	Rr Z + 1	None	1 ⁽⁵⁾ /2 ⁽³⁾	1 ⁽³⁾
ST ⁽²⁾	-Z, Rr	Store Indirect and Pre-Decrement	z	←	Z - 1	None	2 ⁽³⁾	2 ⁽³⁾
STD ⁽¹⁾	Z+q,Rr	Store Indirect with Displacement	(Z + q)	←	Rr	None	2(3)	2(3)
LPM ⁽¹⁾⁽²⁾		Load Program Memory	RO	←	(Z)	None	3	3
LPM ⁽¹⁾⁽²⁾	Rd, Z	Load Program Memory	Rd	←	(Z)	None	3	3
LPM ⁽¹⁾⁽²⁾	Rd, Z+	Load Program Memory and Post- Increment	Rd Z	←	(Z), Z + 1	None	3	3
ELPM ⁽¹⁾		Extended Load Program Memory	R0	←	(RAMPZ:Z)	None	3	
ELPM ⁽¹⁾	Rd, Z	Extended Load Program Memory	Rd	←	(RAMPZ:Z)	None	3	
ELPM ⁽¹⁾	Rd, Z+	Extended Load Program Memory and Post-Increment	Rd Z	←	(RAMPZ:Z), Z + 1	None	3	
SPM ⁽¹⁾		Store Program Memory	(RAMPZ:Z)	←	R1:R0	None	-	
SPM ⁽¹⁾	Z+	Store Program Memory and Post- Increment by 2	(RAMPZ:Z) Z	←	R1:R0, Z+2	None	1.5	
IN	Rd, A	In From I/O Location	Rd	←	I/O(A)	None	1	
OUT	A, Rr	Out To I/O Location	I/O(A)	←	Rr	None	1	
PUSH ⁽¹⁾	Rr	Push Register on Stack	STACK	←	Rr	None	2	1 ⁽³⁾
POP ⁽¹⁾	Rd	Pop Register from Stack	Rd	←	STACK	None	2	2(3)







Mnemonics	Operands	Description	Opera	ation		Flags	#Clocks	#Clock
XCH	Z, Rd	Exchange	(Z) Rd	←	Rd, (Z)	None	1	
LAS	Z, Rd	Load and Set	(Z) Rd	←	Rd v (Z) (Z)	None	1	
LAC	Z, Rd	Load and Clear	(Z) Rd	←	(\$FF – Rd) • (Z) (Z)	None	1	
LAT	Z, Rd	Load and Toggle	(Z) Rd	←	Rd ⊕ (Z) (Z)	None	1	
		В	it and Bit-test Instructions					
LSL	Rd	Logical Shift Left	Rd(n+1) Rd(0) C	$\overset{\leftarrow}{\leftarrow}$	Rd(n), 0, Rd(7)	Z,C,N,V,H	1	
LSR	Rd	Logical Shift Right	Rd(n) Rd(7) C	← ←	Rd(n+1), 0, Rd(0)	Z,C,N,V	1	
ROL	Rd	Rotate Left Through Carry	Rd(0) Rd(n+1) C	← ←	C, Rd(n), Rd(7)	Z,C,N,V,H	1	
ROR	Rd	Rotate Right Through Carry	Rd(7) Rd(n) C	← ←	C, Rd(n+1), Rd(0)	Z,C,N,V	1	
ASR	Rd	Arithmetic Shift Right	Rd(n)	←	Rd(n+1), n=06	Z,C,N,V	1	
SWAP	Rd	Swap Nibbles	Rd(30)	\leftrightarrow	Rd(74)	None	1	
BSET	s	Flag Set	SREG(s)	←	1	SREG(s)	1	
BCLR	s	Flag Clear	SREG(s)	←	0	SREG(s)	1	
SBI	A, b	Set Bit in I/O Register	I/O(A, b)	←	1	None	1 ⁽⁵⁾ 2	1
СВІ	A, b	Clear Bit in I/O Register	I/O(A, b)	←	0	None	1 ⁽⁵⁾ /2	1
BST	Rr, b	Bit Store from Register to T	Т	←	Rr(b)	т	1	
BLD	Rd, b	Bit load from T to Register	Rd(b)	←	Т	None	1	
SEC		Set Carry	С	←	1	С	1	
CLC		Clear Carry	С	←	0	С	1	
SEN		Set Negative Flag	N	←	1	N	1	
CLN		Clear Negative Flag	N	←	0	N	1	
SEZ		Set Zero Flag	Z	←	1	Z	1	
CLZ		Clear Zero Flag	Z	←	0	Z	1	
SEI		Global Interrupt Enable	I	←	1	1	1	
CLI		Global Interrupt Disable	1	←	0	1	1	
SES		Set Signed Test Flag	S	←	1	S	1	
CLS		Clear Signed Test Flag	S	←	0	S	1	
SEV		Set Two's Complement Overflow	V	←	1	V	1	
CLV		Clear Two's Complement Overflow	V	←	0	V	1	
SET		Set T in SREG	Т	←	1	Т	1	
CLT		Clear T in SREG	Т	←	0	Т	1	
SEH		Set Half Carry Flag in SREG	Н	←	1	н	1	
CLH		Clear Half Carry Flag in SREG	Н	←	0	Н	1	
		MCU C	Control Instructions					
BREAK ⁽¹⁾		Break	(See specific de	scr. fo	r BREAK)	None	1	

14 AVR Instruction Set



Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks XMEGA
NOP		No Operation		None	1	
SLEEP		Sleep	(see specific descr. for Sleep)	None	1	
WDR		Watchdog Reset	(see specific descr. for WDR)	None	1	

- Notes: 1. This instruction is not available in all devices. Refer to the device specific instruction set summary.
 - 2. Not all variants of this instruction are available in all devices. Refer to the device specific instruction set summary.
 - 3. Cycle times for Data memory accesses assume internal memory accesses, and are not valid for accesses via the external
 - 4. One extra cycle must be added when accessing Internal SRAM.
 - 5. Number of clock cycles for Reduced Core tinyAVR.







ADC - Add with Carry

Description:

(i)

Adds two registers and the contents of the C Flag and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd + Rr + C$

Syntax:Operands:Program Counter:ADC Rd,Rr $0 \le d \le 31, 0 \le r \le 31$ $PC \leftarrow PC + 1$

16-bit Opcode:

0001	11rd	dddd	rrrr
------	------	------	------

Status Register (SREG) Boolean Formula:

I	Т	Н	S	V	N	Z	С	
-	-	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	

H: Rd3•Rr3+Rr3•R3•Rd3
Set if there was a carry from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: Rd7•Rr7•R7+Rd7•Rr7•R7

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$ Set if the result is \$00; cleared otherwise.

C: Rd7•Rr7+Rr7•R7+R7•Rd7

Set if there was carry from the MSB of the result; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

; Add R1:R0 to R3:R2

add r2,r0 ; Add low byte

adc r3,r1 ; Add with carry high byte

Words: 1 (2 bytes)

Cycles: 1

16 AVR Instruction Set I

0856I-AVR-07/10



ADD - Add without Carry

Description:

Adds two registers without the C Flag and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd + Rr$

16-bit Opcode:

0000	11rd	dddd	rrrr
------	------	------	------

Status Register (SREG) and Boolean Formula:

1	T	Н	S	V	N	Z	С
_	-	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow

H: Rd3•Rr3+Rr3•R3+R3•Rd3

Set if there was a carry from bit 3; cleared otherwise

- S: $N \oplus V$, For signed tests.
- V: Rd7•Rr7•R7+Rd7•Rr7•R7

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7• R6 •R5• R4 •R3 •R2 •R1 •R0

Set if the result is \$00; cleared otherwise.

C: Rd7 •Rr7 +Rr7 •R7+ R7 •Rd7

Set if there was carry from the MSB of the result; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

add r1,r2 ; Add r2 to r1 (r1=r1+r2) add r28,r28 ; Add r28 to itself (r28=r28+r28)

Words: 1 (2 bytes)

Cycles: 1







ADIW - Add Immediate to Word

Description:

(i)

Adds an immediate value (0 - 63) to a register pair and places the result in the register pair. This instruction operates on the upper four register pairs, and is well suited for operations on the pointer registers.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) $Rd+1:Rd \leftarrow Rd+1:Rd + K$

Syntax: Operands:

ADIW Rd+1:Rd,K $d \in \{24,26,28,30\}, 0 \le K \le 63$

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001	0110	KKdd	KKKK
------	------	------	------

Status Register (SREG) and Boolean Formula:

1	T	Н	S	V	N	Z	С
-	-	-	⇔	\Leftrightarrow	⇔	⇔	⇔

S: $N \oplus V$, For signed tests.

V: Rdh7 • R15

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R15

Set if MSB of the result is set; cleared otherwise.

Z: R15 •R14 •R13 •R12 •R11 •R10 •R9 •R8 •R7 • R6 • R5 • R4 • R3 • R2 •R1 • R0 Set if the result is \$0000; cleared otherwise.

C: R15 • Rdh7

Set if there was carry from the MSB of the result; cleared otherwise.

R (Result) equals Rdh:Rdl after the operation (Rdh7-Rdh0 = R15-R8, Rdl7-Rdl0=R7-R0).

Example:

```
adiw r25:24,1 ; Add 1 to r25:r24 adiw ZH:ZL,63 ; Add 63 to the Z-pointer(r31:r30)
```

Words: 1 (2 bytes)

Cycles: 2

18 AVR Instruction Set I

0856I-AVR-07/10



AND - Logical AND

Description:

Performs the logical AND between the contents of register Rd and register Rr and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd \bullet Rr$

16-bit Opcode:

0010	00rd	dddd	rrrr
------	------	------	------

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С	
-	-	=	\Leftrightarrow	0	\Leftrightarrow	⇔	-	7

S: $N \oplus V$, For signed tests.

V: 0

Cleared

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7 •R6 •R5 •R4 •R3 • R2 •R1 •R0
Set if the result is \$00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

and r2,r3 ; Bitwise and r2 and r3, result in r2
ldi r16,1 ; Set bitmask 0000 0001 in r16
and r2,r16 ; Isolate bit 0 in r2

Words: 1 (2 bytes)

Cycles: 1







ANDI – Logical AND with Immediate

Description:

Performs the logical AND between the contents of register Rd and a constant and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd \cdot K$

 Syntax:
 Operands:

 ANDI Rd,K
 $16 \le d \le 31, 0 \le K \le 255$

Program Counter:

PC ← PC + 1

16-bit Opcode:

0111 KKKK dddd KKKK

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С
-	-	-	\Leftrightarrow	0	\Leftrightarrow	\Leftrightarrow	-

S: $N \oplus V$, For signed tests.

V: 0

Cleared

N: R

Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$ Set if the result is \$00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

andi r17,\$0F ; Clear upper nibble of r17 andi r18,\$10 ; Isolate bit 4 in r18 andi r19,\$AA ; Clear odd bits of r19

Words: 1 (2 bytes)

Cycles: 1

20 AVR Instruction Set

0856I-AVR-07/10



ASR - Arithmetic Shift Right

Description:

Shifts all bits in Rd one place to the right. Bit 7 is held constant. Bit 0 is loaded into the C Flag of the SREG. This operation effectively divides a signed value by two without changing its sign. The Carry Flag can be used to round the result.

Operation:





Syntax: (i) ASR Rd Operands: $0 \le d \le 31$

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001 010d dddd 0101

Status Register (SREG) and Boolean Formula:

1	T	Н	S	V	N	Z	С	
-	-	820	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow]

S: $N \oplus V$, For signed tests.

V: N ⊕ C (For N and C after the shift)

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$ Set if the result is \$00; cleared otherwise.

C: Rd0

Set if, before the shift, the LSB of Rd was set; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

ldi r16,\$10 ; Load decimal 16 into r16
asr r16 ; r16=r16 / 2
ldi r17,\$FC ; Load -4 in r17
asr r17 ; r17=r17/2

Words: 1 (2 bytes)

Cycles: 1

AMEL

21





BCLR - Bit Clear in SREG

Description:

Clears a single Flag in SREG.

Operation:

(i) SREG(s) \leftarrow 0

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001 0100 1eee 1					
1001	00	100	1sss	0100	1001

Status Register (SREG) and Boolean Formula:

ı	Т	н	S	V	N	Z	С	
\Leftrightarrow	⇔	\Leftrightarrow	\Leftrightarrow	\$	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	

I: 0 if s = 7; Unchanged otherwise.

T: 0 if s = 6; Unchanged otherwise.

H: 0 if s = 5; Unchanged otherwise.

S: 0 if s = 4; Unchanged otherwise.

V: 0 if s = 3; Unchanged otherwise.

N: 0 if s = 2; Unchanged otherwise.

Z: 0 if s = 1; Unchanged otherwise.

C: 0 if s = 0; Unchanged otherwise.

Example:

bclr 0 ; Clear Carry Flag
bclr 7 ; Disable interrupts

Words: 1 (2 bytes)

Cycles: 1

22 AVR Instruction Set

0856I-AVR-07/10



BLD - Bit Load from the T Flag in SREG to a Bit in Register

Description:

(i)

Copies the T Flag in the SREG (Status Register) to bit b in register Rd.

Operation:

BLD Rd,b

(i) $Rd(b) \leftarrow T$

Syntax: Operands:

Program Counter:

 $PC \leftarrow PC + 1$

16 bit Opcode:

1111 100d dddd 0bbb

Status Register (SREG) and Boolean Formula:

I	Т	н	S	V	N	Z	С
-	-	-	-	=	-	-	=

 $0 \leq d \leq 31, \, 0 \leq b \leq 7$

Example:

; Copy bit

bst r1,2 ; Store bit 2 of r1 in T Flag bld r0,4 ; Load T Flag into bit 4 of r0

Words: 1 (2 bytes)

Cycles: 1







BRBC - Branch if Bit in SREG is Cleared

Description:

Conditional relative branch. Tests a single bit in SREG and branches relatively to PC if the bit is cleared. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form.

Operation:

(i) If SREG(s) = 0 then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

Syntax: Operands:

BRBC s,k $0 \le s \le 7, -64 \le k \le +63 \qquad \qquad PC \leftarrow PC + k + 1$

PC ← PC + 1, if condition is false

Program Counter:

16-bit Opcode:

1111	01kk	kkkk	ksss
1111	0 17171	16767676	, , , , ,

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С	
-	-	-	-	-	-	-	-	

Example:

(i)

cpi r20,5 ; Compare r20 to the value 5
brbc 1,noteq ; Branch if Zero Flag cleared

. . .

noteq:nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false 2 if condition is true

24 AVR Instruction Set



BRBS - Branch if Bit in SREG is Set

Description:

Conditional relative branch. Tests a single bit in SREG and branches relatively to PC if the bit is set. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form.

Operation:

(i) If SREG(s) = 1 then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

Syntax: Operands:

Program Counter:

(i) BRBS s,k

 $0 \leq s \leq 7, \ \text{-}64 \leq k \leq \text{+}63$

 $PC \leftarrow PC + k + 1$ $PC \leftarrow PC + 1$, if condition is false

16-bit Opcode:

IIII UUKK KKKK KSSS	1111	00kk	kkkk	ksss
---------------------------	------	------	------	------

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С
_	-	_	_	-	-	-	_

Example:

bst r0,3; Load T bit with bit 3 of r0

brbs 6,bitset ; Branch T bit was set

. . .

bitset: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false







BRCC - Branch if Carry Cleared

Description:

Conditional relative branch. Tests the Carry Flag (C) and branches relatively to PC if C is cleared. This instruction branches relatively to PC in either direction (PC - 63 ≤ destination ≤ PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 0,k).

Operation:

(i) If C = 0 then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Syntax: Operands: BRCC k $\textbf{-64} \leq k \leq \textbf{+63}$ **Program Counter:** $PC \leftarrow PC + k + 1$

 $PC \leftarrow PC + 1$, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k000
------	------	------	------

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С
_	-	_	_	_	-	-	-

Example:

(i)

; Add r23 to r22 add r22,r23

; Branch if carry cleared brcc nocarry

nocarry: nop

; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false



BRCS – Branch if Carry Set

Description:

Conditional relative branch. Tests the Carry Flag (C) and branches relatively to PC if C is set. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 0,k).

Operation:

(i) If C = 1 then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

> Syntax: Operands: BRCS k $\text{-}64 \leq k \leq \text{+}63$

Program Counter:

 $PC \leftarrow PC + k + 1$

 $PC \leftarrow PC + 1$, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k000
------	------	------	------

Status Register (SREG) and Boolean Formula:

	I	Т	Н	S	V	N	Z	С
Γ	_	-	-	_	-	_	-	_

Example:

(i)

cpi r26,\$56 ; Compare r26 with \$56 brcs carry ; Branch if carry set

; Branch destination (do nothing) carry: nop

Words: 1 (2 bytes)

Cycles: 1 if condition is false







BREAK - Break

Description:

The BREAK instruction is used by the On-chip Debug system, and is normally not used in the application software. When the BREAK instruction is executed, the AVR CPU is set in the Stopped Mode. This gives the On-chip Debugger access to internal resources.

If any Lock bits are set, or either the JTAGEN or OCDEN Fuses are unprogrammed, the CPU will treat the BREAK instruction as a NOP and will not enter the Stopped mode.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) On-chip Debug system break.

16-bit Opcode:

1001	0101	1001	1000

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С
_	-	_	-	-	-	_	I

Words: 1 (2 bytes)

Cycles: 1



BREQ – Branch if Equal

Description:

Conditional relative branch. Tests the Zero Flag (Z) and branches relatively to PC if Z is set. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the unsigned or signed binary number represented in Rd was equal to the unsigned or signed binary number represented in Rr. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 1,k).

Operation:

(i) If Rd = Rr (Z = 1) then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

Syntax: Operands: BREQ k $-64 \le k \le +63$

Program Counter:

 $PC \leftarrow PC + k + 1$ $PC \leftarrow PC + 1$, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k001
------	------	------	------

Status Register (SREG) and Boolean Formula:

1	T	Н	S	V	N	Z	С
-	-	-	-	-	-	-	-

Example:

(i)

cp r1,r0 ; Compare registers r1 and r0 breq equal ; Branch if registers equal

equal: nop

; Branch destination (do nothing)

Words: 1 (2 bytes)







BRGE – Branch if Greater or Equal (Signed)

Description:

Conditional relative branch. Tests the Signed Flag (S) and branches relatively to PC if S is cleared. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the signed binary number represented in Rd was greater than or equal to the signed binary number represented in Rr. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 4,k).

Operation:

(i) If $Rd \ge Rr$ (N \oplus V = 0) then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Syntax: Operands: BRGE k $-64 \le k \le +63$

Program Counter:

PC ← PC + k + 1

PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k100
1111	UIKK	KKKK	KIUU

Status Register (SREG) and Boolean Formula:

I	T	Н	S	V	N	Z	С
-	-	-	-	-	-	-	-

Example:

(i)

cp r11,r12 ; Compare registers r11 and r12 brge greateq ; Branch if r11 \geq r12 (signed)

greateq: nop

; Branch destination (do nothing)

Words: 1 (2 bytes)



BRHC – Branch if Half Carry Flag is Cleared

Description:

Conditional relative branch. Tests the Half Carry Flag (H) and branches relatively to PC if H is cleared. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 5,k).

Operation:

(i) If H = 0 then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Syntax: Operands: BRHC k $-64 \le k \le +63$

Program Counter:

 $PC \leftarrow PC + k + 1$

 $PC \leftarrow PC + 1$, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k101
------	------	------	------

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С
-	_	_	-	-	-	-	_

Example:

(i)

brhc hclear ; Branch if Half Carry Flag cleared

. . .

hclear: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)





BRHS - Branch if Half Carry Flag is Set

Description:

Conditional relative branch. Tests the Half Carry Flag (H) and branches relatively to PC if H is set. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 5,k).

Operation:

(i) If H = 1 then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Syntax: Operands: (i) BRHS k $-64 \le k \le +63$

Program Counter: PC \leftarrow PC + k + 1

 $PC \leftarrow PC + 1$, if condition is false

16-bit Opcode:

1111 00kk kkkk k101

Status Register (SREG) and Boolean Formula:

ı	Т	Н	S	V	N	Z	С
1-1	(— 7)	_	-	-	_	_	-

Example:

brhs hset ; Branch if Half Carry Flag set ...
nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

hset:



BRID – Branch if Global Interrupt is Disabled

Description:

Conditional relative branch. Tests the Global Interrupt Flag (I) and branches relatively to PC if I is cleared. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 7,k).

Operation:

(i) If I = 0 then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

Syntax: Operands: BRID k $-64 \le k \le +63$

Program Counter:

 $PC \leftarrow PC + k + 1$

 $PC \leftarrow PC + 1$, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k111
------	------	------	------

Status Register (SREG) and Boolean Formula:

	I	Т	Н	S	V	N	Z	С
Γ	_	-	-	_	-	_	-	_

Example:

(i)

brid intdis ; Branch if interrupt disabled

. . .

intdis: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)





BRIE – Branch if Global Interrupt is Enabled

Description:

Conditional relative branch. Tests the Global Interrupt Flag (I) and branches relatively to PC if I is set. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 7,k).

Operation:

(i) If I = 1 then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

Syntax: Operands: Program Counter:

(i) BRIE k $-64 \le k \le +63 \\ \text{PC} \leftarrow \text{PC} + k + 1 \\ \text{PC} \leftarrow \text{PC} + 1, \text{ if condition is false}$

16-bit Opcode:

1111	00kk	kkkk	k111

Status Register (SREG) and Boolean Formula:

1	T	н	S	V	N	Z	С
-	-		-	-	-	-	-

Example:

brie inten ; Branch if interrupt enabled

...

inten: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false



BRLO – Branch if Lower (Unsigned)

Description:

Conditional relative branch. Tests the Carry Flag (C) and branches relatively to PC if C is set. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the unsigned binary number represented in Rd was smaller than the unsigned binary number represented in Rr. This instruction branches relatively to PC in either direction (PC - $63 \le \text{destination} \le \text{PC} + 64$). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 0,k).

Operation:

(i) If Rd < Rr (C = 1) then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

Syntax: Operands: Program Counter:

(i) BRLO k $-64 \le k \le +63$ PC \leftarrow PC + k + 1

PC ← PC + 1, if condition is false

16-bit Opcode:

1111 00kk kkkk k0	00
-------------------	----

Status Register (SREG) and Boolean Formula:

ı	Т	Н	S	V	N	Z	С
_	_	_	-	_	-	_	_

Example:

eor r19,r19 ; Clear r19
loop: inc r19 ; Increase r19
...
cpi r19,\$10 ; Compare r19 with \$10
brlo loop ; Branch if r19 < \$10 (unsigned)
nop ; Exit from loop (do nothing)

Words: 1 (2 bytes)







BRLT – Branch if Less Than (Signed)

Description:

Conditional relative branch. Tests the Signed Flag (S) and branches relatively to PC if S is set. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the signed binary number represented in Rd was less than the signed binary number represented in Rr. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 4,k).

Operation:

(i) If Rd < Rr (N \oplus V = 1) then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

16-bit Opcode:

	1111	00kk	kkkk	k100
--	------	------	------	------

Status Register (SREG) and Boolean Formula:

ı	Т	Н	S	V	N	Z	С	
-	-	=	-	-	-	-	-	

Example:

cp r16,r1 ; Compare r16 to r1
brlt less ; Branch if r16 < r1 (signed)
...
less: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false 2 if condition is true

36 AVR Instruction Set



BRMI - Branch if Minus

Description:

Conditional relative branch. Tests the Negative Flag (N) and branches relatively to PC if N is set. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 2,k).

Operation:

(i) If N = 1 then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Syntax: Operands: BRMI k $-64 \le k \le +63$

Program Counter:

 $PC \leftarrow PC + k + 1$ $PC \leftarrow PC + 1$, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k010
------	------	------	------

Status Register (SREG) and Boolean Formula:

	I	Т	Н	S	V	N	Z	С
Г	-	-	-	_	-	-	-	_

Example:

(i)

subi r18,4 ; Subtract 4 from r18
brmi negative ; Branch if result negative
...
negative: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)







BRNE - Branch if Not Equal

Description:

Conditional relative branch. Tests the Zero Flag (Z) and branches relatively to PC if Z is cleared. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the unsigned or signed binary number represented in Rd was not equal to the unsigned or signed binary number represented in Rr. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 1,k).

Operation:

(i) If Rd \neq Rr (Z = 0) then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

Syntax:Operands:Program Counter:BRNE k $-64 \le k \le +63$ PC \leftarrow PC + k + 1

 $PC \leftarrow PC + 1$, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k001
0.0000000000000000000000000000000000000	260/2007/00/00	(300) (300) (300)	

Status Register (SREG) and Boolean Formula:

1	T	Н	S	V	N	Z	С
-	-	-	-	-	-	-	-

Example:

(i)

eor r27,r27 ; Clear r27

loop: inc r27 ; Increase r27

...

cpi r27,5 ; Compare r27 to 5

brne loop ; Branch if r27<>5

nop ; Loop exit (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false 2 if condition is true

38 AVR Instruction Set



BRPL - Branch if Plus

Description:

Conditional relative branch. Tests the Negative Flag (N) and branches relatively to PC if N is cleared. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 2,k).

Operation:

(i) If N = 0 then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

Program Counter: $PC \leftarrow PC + k + 1$

 $PC \leftarrow PC + 1$, if condition is false

16-bit Opcode:

k k010

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С
_	_	-		742	-	-	_

Example:

. . .

positive: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false 2 if condition is true



39





BRSH - Branch if Same or Higher (Unsigned)

Description:

Conditional relative branch. Tests the Carry Flag (C) and branches relatively to PC if C is cleared. If the instruction is executed immediately after execution of any of the instructions CP, CPI, SUB or SUBI the branch will occur if and only if the unsigned binary number represented in Rd was greater than or equal to the unsigned binary number represented in Rr. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 0,k).

Operation:

(i) If $Rd \ge Rr (C = 0)$ then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k000
------	------	------	------

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С
-	_	_	_	-	-	-	-

Example:

subi r19,4 ; Subtract 4 from r19

brsh highsm ; Branch if r19 >= 4 (unsigned)

...

highsm: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false 2 if condition is true

AVR Instruction Set •



BRTC – Branch if the T Flag is Cleared

Description:

Conditional relative branch. Tests the T Flag and branches relatively to PC if T is cleared. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 6,k).

Operation:

(i) If T = 0 then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

Program Counter: $PC \leftarrow PC + k + 1$

 $PC \leftarrow PC + 1$, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k110
------	------	------	------

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С	
_	_	(<u>—</u>)(-	_	_	_	_	

Example:

bst r3,5 ; Store bit 5 of r3 in T Flag
brtc tclear ; Branch if this bit was cleared

. . .

tclear: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false 2 if condition is true



41





BRTS - Branch if the T Flag is Set

Description:

Conditional relative branch. Tests the T Flag and branches relatively to PC if T is set. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 6,k).

Operation:

(i) If T = 1 then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

Program Counter:

 $\begin{array}{l} PC \leftarrow PC + k + 1 \\ PC \leftarrow PC + 1, \mbox{ if condition is false} \end{array}$

16-bit Opcode:

1111	00kk	kkkk	k110
------	------	------	------

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С	
_	_	-	-	-	_	_	_	

Example:

bst r3,5 ; Store bit 5 of r3 in T Flag
brts tset ; Branch if this bit was set

tset: nop

; Branch destination (do nothing)

Words: 1 (2 bytes)



BRVC - Branch if Overflow Cleared

Description:

Conditional relative branch. Tests the Overflow Flag (V) and branches relatively to PC if V is cleared. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 3,k).

Operation:

BRVC k

(i) If V = 0 then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Syntax: Operands:

 $-64 \le k \le +63$ PC \leftarrow PC + k + 1

 $PC \leftarrow PC + 1$, if condition is false

Program Counter:

16-bit Opcode:

1111	01kk	kkkk	k011

Status Register (SREG) and Boolean Formula:

I	Т	н	s	V	N	Z	С
(-)	-	-	-	-	-	=	-

Example:

(i)

add r3,r4 ; Add r4 to r3

brvc noover ; Branch if no overflow

...

noover: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false







BRVS - Branch if Overflow Set

Description:

Conditional relative branch. Tests the Overflow Flag (V) and branches relatively to PC if V is set. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 3,k).

Operation:

(i) If V = 1 then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Syntax: Operands: BRVS k $-64 \le k \le +63$

Program Counter:

 $PC \leftarrow PC + k + 1$ $PC \leftarrow PC + 1$, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k011
------	------	------	------

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С
_	-	_	_	-	-	-	_

Example:

(i)

add r3,r4 ; Add r4 to r3
brvs overf1 ; Branch if overflow

overfl: nop

; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false 2 if condition is true

44 AVR Instruction Set

0856I-AVR-07/10



BSET - Bit Set in SREG

Description:

Sets a single Flag or bit in SREG.

Operation:

(i) SREG(s) \leftarrow 1

16-bit Opcode:

1001	0100	0sss	1000

Status Register (SREG) and Boolean Formula:

ı	Т	н	S	V	N	Z	С	
\Leftrightarrow	l							

I: 1 if s = 7; Unchanged otherwise.

T: 1 if s = 6; Unchanged otherwise.

H: 1 if s = 5; Unchanged otherwise.

S: 1 if s = 4; Unchanged otherwise.

V: 1 if s = 3; Unchanged otherwise.

N: 1 if s = 2; Unchanged otherwise.

Z: 1 if s = 1; Unchanged otherwise.

C: 1 if s = 0; Unchanged otherwise.

Example:

bset 6 ; Set T Flag
bset 7 ; Enable interrupt

Words: 1 (2 bytes)

Cycles: 1







Program Counter:

 $PC \leftarrow PC + 1$

BST - Bit Store from Bit in Register to T Flag in SREG

Description:

Stores bit b from Rd to the T Flag in SREG (Status Register).

Operation:

(i) $T \leftarrow Rd(b)$

 Syntax:
 Operands:

 (i)
 BST Rd,b
 $0 \le d \le 31, 0 \le b \le 7$

16-bit Opcode:

1111 101d dddd 0bbb

Status Register (SREG) and Boolean Formula:

I	Т	н	S	V	N	Z	С	
-	\Leftrightarrow	=	-	=	-	-		7

T: 0 if bit b in Rd is cleared. Set to 1 otherwise.

Example:

; Copy bit

bst r1,2 ; Store bit 2 of r1 in T Flag bld r0,4 ; Load T into bit 4 of r0

Words: 1 (2 bytes)

Cycles: 1

46 AVR Instruction Set

0856I-AVR-07/10



CALL - Long Call to a Subroutine

Description:

Calls to a subroutine within the entire Program memory. The return address (to the instruction after the CALL) will be stored onto the Stack. (See also RCALL). The Stack Pointer uses a post-decrement scheme during CALL.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i)	$PC \leftarrow k$	Devices with 16 bits PC, 128K bytes Program memory maximum.
(ii)	$PC \leftarrow k$	Devices with 22 bits PC, 8M bytes Program memory maximum.

	Syntax:	Operands:	Program Counter	Stack:
(i)	CALL k	$0 \leq k < 64K$	PC ← k	STACK ← PC+2 SP ← SP-2, (2 bytes, 16 bits)

(ii) C	CALL k	0 ≤ k < 4M	PC ← k	STACK ← PC+2		
				CD / CD 2 /2 bytes 22 bit		

SP ← SP-3 (3 bytes, 22 bits)

32-bit Opcode:

1001	010k	kkkk	111k
kkkk	kkkk	kkkk	kkkk

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С
-	- -0	-	-	-		_	_

Example:

	mov	r16,r0	; C	opy r0 to r16
	call	check	; C	all subroutine
	nop		; C	ontinue (do nothing)
check:	cpi	r16,\$42	; C	heck if r16 has a special value
	breq	error	; B	ranch if equal
	ret		; R	eturn from subroutine
error:	rjmp	error	; I:	nfinite loop

Words : 2 (4 bytes)

Cycles : 4, devices with 16 bit PC

5, devices with 22 bit PC

Cycles XMEGA: 3, devices with 16 bit PC

4, devices with 22 bit PC







CBI - Clear Bit in I/O Register

Description:

(i)

Clears a specified bit in an I/O Register. This instruction operates on the lower 32 I/O Registers – addresses 0-31.

Operation:

(i) $I/O(A,b) \leftarrow 0$

Syntax: CBI A,b

Operands: $0 \leq A \leq 31, \ 0 \leq b \leq 7$ **Program Counter:**

 $PC \leftarrow PC + 1$

16-bit Opcode:

1000 1001 AAAA Abbb

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С
-	-	=	-	-	-	-	=

Example:

; Clear bit 7 in Port D \$12,7

Words: 1 (2 bytes)

2 Cycles: Cycles XMEGA: 1 Cycles Reduced Core tinyAVR:



CBR - Clear Bits in Register

Description:

Clears the specified bits in register Rd. Performs the logical AND between the contents of register Rd and the complement of the constant mask K. The result will be placed in register Rd.

Operation:

(i) $Rd \leftarrow Rd \bullet (\$FF - K)$

16-bit Opcode: (see ANDI with K complemented)

Status Register (SREG) and Boolean Formula:

Ţ	Т	Н	S	V	N	Z	С	
-	_	_	\Leftrightarrow	0	\Leftrightarrow	⇔	=	

S: $N \oplus V$, For signed tests.

V: 0

Cleared

N: R

Set if MSB of the result is set; cleared otherwise.

Z: R7 •R6 •R5• R4• R3 •R2• R1• R0
Set if the result is \$00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

cbr r16,\$F0 ; Clear upper nibble of r16
cbr r18,1 ; Clear bit 0 in r18

Words: 1 (2 bytes)

Cycles: 1







CLC - Clear Carry Flag

Description:

Clears the Carry Flag (C) in SREG (Status Register).

Operation:

(i) C ← 0

Syntax: Operands: (i) CLC None

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001 0100 1000 1000

Status Register (SREG) and Boolean Formula:

ı	Т	н	S	V	N	Z	С	
-	_	_	-	-	_	-	0	Ì

C: (

Carry Flag cleared

Example:

add r0,r0 ; Add r0 to itself clc ; Clear Carry Flag

Words: 1 (2 bytes)

Cycles: 1



CLH - Clear Half Carry Flag

Description:

Clears the Half Carry Flag (H) in SREG (Status Register).

Operation:

(i) H ← 0

16-bit Opcode:

1001 0100 1101 1000

Status Register (SREG) and Boolean Formula:

ı	Т	н	S	V	N	Z	С	
-	_	0	-	-	_	-	-	1

H: 0

Half Carry Flag cleared

Example:

clh ; Clear the Half Carry Flag

Words: 1 (2 bytes)







CLI - Clear Global Interrupt Flag

Description:

Clears the Global Interrupt Flag (I) in SREG (Status Register). The interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction.

Operation:

(i) I ← 0

16-bit Opcode:

1001	0100	1111	1000
A SECTION AND A		Name and Address of the Party o	N. P. SEC. A. L. STOLE .

Status Register (SREG) and Boolean Formula:

1	Т	н	S	V	N	z	С	
0	-	-	-	-	-	-	_	

I: 0
Global Interrupt Flag cleared

Example:

in temp, SREG ; Store SREG value (temp must be defined by user)
cli ; Disable interrupts during timed sequence
sbi EECR, EEMWE; Start EEPROM write
sbi EECR, EEWE
out SREG, temp ; Restore SREG value (I-Flag)

Words: 1 (2 bytes)

Cycles: 1

52 AVR Instruction Set



CLN - Clear Negative Flag

Description:

Clears the Negative Flag (N) in SREG (Status Register).

Operation:

(i) $N \leftarrow 0$

16-bit Opcode:

1001 0100 1010 1000

Status Register (SREG) and Boolean Formula:

ı	Т	н	S	V	N	Z	С
_	-	-	1-1	-	0	-	-

N: 0

Negative Flag cleared

Example:

add r2,r3 ; Add r3 to r2 cln ; Clear Negative Flag

Words: 1 (2 bytes)







CLR - Clear Register

Description:

Clears a register. This instruction performs an Exclusive OR between a register and itself. This will clear all bits in the register.

Operation:

(i) $Rd \leftarrow Rd \oplus Rd$

16-bit Opcode: (see EOR Rd,Rd)

0010	01dd	dddd	dddd

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С
_	-	_	0	0	0	1	_

S: 0

Cleared

V: 0

Cleared

N: 0

Cleared

Z: 1

Set

R (Result) equals Rd after the operation.

Example:

clr r18 ; clear r18
loop: inc r18 ; increase r18
...
cpi r18,\$50 ; Compare r18 to \$50
brne loop

Words: 1 (2 bytes)

Cycles: 1

54 AVR Instruction Set



CLS - Clear Signed Flag

Description:

Clears the Signed Flag (S) in SREG (Status Register).

Operation:

(i) $S \leftarrow 0$

16-bit Opcode:

1001 0100 1100 1000

Status Register (SREG) and Boolean Formula:

ı	Т	н	S	V	N	Z	С	
_	-	_	0	-	_	-	-]

S: 0

Signed Flag cleared

Example:

add r2,r3 ; Add r3 to r2 cls ; Clear Signed Flag

Words: 1 (2 bytes)





CLT - Clear T Flag

Description:

Clears the T Flag in SREG (Status Register).

Operation:

(i) $T \leftarrow 0$

16-bit Opcode:

1001 0100 1110 1000

Status Register (SREG) and Boolean Formula:

ı	Т	н	S	V	N	Z	С	
-	0	_	-	-	-	-	-]

T: (

T Flag cleared

Example:

:lt ; Clear T Flag

Words: 1 (2 bytes)



CLV - Clear Overflow Flag

Description:

Clears the Overflow Flag (V) in SREG (Status Register).

Operation:

(i) $V \leftarrow 0$

16-bit Opcode:

1001 0100 1011 1000

Status Register (SREG) and Boolean Formula:

ı	Т	н	S	V	N	Z	С	
-	-	-	_	0	_	_	_]

V: 0

Overflow Flag cleared

Example:

add r2,r3 ; Add r3 to r2

clv ; Clear Overflow Flag

Words: 1 (2 bytes)







CLZ - Clear Zero Flag

Description:

Clears the Zero Flag (Z) in SREG (Status Register).

Operation:

(i) $Z \leftarrow 0$

Syntax: Operands: (i) CLZ None

Program Counter:

PC ← PC + 1

16-bit Opcode:

1001	0100	1001	1000
------	------	------	------

Status Register (SREG) and Boolean Formula:

ı	Т	н	S	V	N	Z	С	
-	-	-	-	_	-	0	-	

Z: 0

Zero Flag cleared

Example:

add r2,r3 ; Add r3 to r2 clz ; Clear zero

Words: 1 (2 bytes)



COM - One's Complement

Description:

This instruction performs a One's Complement of register Rd.

Operation:

(i) $Rd \leftarrow \$FF - Rd$

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001	010d	dddd	0000

Status Register (SREG) and Boolean Formula:

ı	Т	Н	S	V	N	Z	С	
-	9-	-	\Leftrightarrow	0	⇔	\Leftrightarrow	1]

S: $N \oplus V$

For signed tests.

V: 0

Cleared.

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7 •R6• R5• R4 •R3 •R2• R1 •R0
Set if the result is \$00; Cleared otherwise.

C: 1 Set.

R (Result) equals Rd after the operation.

Example:

com r4 ; Take one's complement of r4 breq zero ; Branch if zero

breq zero , branen ir zer

zero: nop

; Branch destination (do nothing)

Words: 1 (2 bytes)







CP - Compare

Description:

This instruction performs a compare between two registers Rd and Rr. None of the registers are changed. All conditional branches can be used after this instruction.

Operation:

(i) Rd - Rr

 $\begin{tabular}{lll} \textbf{Syntax:} & \textbf{Operands:} \\ (i) & CP \ Rd, Rr & 0 \le d \le 31, \ 0 \le r \le 31 \end{tabular}$

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

0001	01rd	dddd	rrrr
0001	UIIG	aaaa	TITI

Status Register (SREG) and Boolean Formula:

ı	T	Н	S	V	N	Z	С
_	_	⇔	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\$

H: Rd3 •Rr3+ Rr3 •R3 +R3• Rd3

Set if there was a borrow from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: Rd7• Rr7 •R7+ Rd7 •Rr7 •R7

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7• R6 •R5• R4 •R3 •R2 •R1 •R0

Set if the result is \$00; cleared otherwise.

C: Rd7 •Rr7+ Rr7• R7 +R7• Rd7

Set if the absolute value of the contents of Rr is larger than the absolute value of Rd; cleared otherwise.

R (Result) after the operation.

Example:

```
cp r4,r19 ; Compare r4 with r19
brne noteq ; Branch if r4 <> r19
...
noteq: nop ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1

60 AVR Instruction Set



CPC – Compare with Carry

Description:

This instruction performs a compare between two registers Rd and Rr and also takes into account the previous carry. None of the registers are changed. All conditional branches can be used after this instruction.

Operation:

(i) Rd - Rr - C

0000 01rd dddd rrrr

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С	
-	-	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow]

H: Rd3 •Rr3+ Rr3 •R3 +R3 •Rd3

Set if there was a borrow from bit 3; cleared otherwise

- S: $N \oplus V$, For signed tests.
- V: Rd7 •Rr7 R7+ Rd7 Rr7 •R7

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7 •R6• R5• R4 •R3 •R2 •R1• R0 •Z

Previous value remains unchanged when the result is zero; cleared otherwise.

C: Rd7 •Rr7+ Rr7• R7 +R7 •Rd7

Set if the absolute value of the contents of Rr plus previous carry is larger than the absolute value of Rd; cleared otherwise.

R (Result) after the operation.

Example:

```
; Compare r3:r2 with r1:r0

cp r2,r0 ; Compare low byte

cpc r3,r1 ; Compare high byte

brne noteq ; Branch if not equal

...

noteq: nop ; Branch destination (do nothing)
```



61





Words: 1 (2 bytes)

Cycles: 1

62 AVR Instruction Set



CPI – Compare with Immediate

Description:

This instruction performs a compare between register Rd and a constant. The register is not changed. All conditional branches can be used after this instruction.

Operation:

(i) Rd - K

	- 22		
0011	KKKK	dddd	KKKK

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С
-	-	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow

- H: Rd3 •K3+ K3• R3+ R3 •Rd3
 Set if there was a borrow from bit 3; cleared otherwise
- S: $N \oplus V$, For signed tests.
- V: Rd7 •K7 •R7 +Rd7 •K7 •R7

 Set if two's complement overflow resulted from the operation; cleared otherwise.
- N: R7 Set if MSB of the result is set; cleared otherwise.
- Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$ Set if the result is \$00; cleared otherwise.
- C: Rd7 •K7 +K7 •R7+ R7 •Rd7

 Set if the absolute value of K is larger than the absolute value of Rd; cleared otherwise.

R (Result) after the operation.

Example:

cpi r19,3 ; Compare r19 with 3
brne error ; Branch if r19<>3
...
error: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1



63





CPSE – Compare Skip if Equal

Description:

This instruction performs a compare between two registers Rd and Rr, and skips the next instruction if Rd = Rr.

Operation:

(i) If Rd = Rr then PC \leftarrow PC + 2 (or 3) else PC \leftarrow PC + 1

Syntax: Operands:

(i) CPSE Rd,Rr $0 \le d \le 31, 0 \le r \le 31$

Program Counter: $PC \leftarrow PC + 1$, Condition false - no skip $PC \leftarrow PC + 2$, Skip a one word instruction

PC ← PC + 3, Skip a two word instruction

16-bit Opcode:

0001	00rd	dddd	rrrr
------	------	------	------

Status Register (SREG) and Boolean Formula:

ı	T	Н	S	V	N	Z	С	
-	-	-	-	-	-	-	-	1

Example:

inc r4 ; Increase r4
cpse r4,r0 ; Compare r4 to r0
neg r4 ; Only executed if r4<>r0
nop ; Continue (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false (no skip)

2 if condition is true (skip is executed) and the instruction skipped is 1 word 3 if condition is true (skip is executed) and the instruction skipped is 2 words

64 AVR Instruction Set —



DEC - Decrement

Description:

Subtracts one -1- from the contents of register Rd and places the result in the destination register Rd.

The C Flag in SREG is not affected by the operation, thus allowing the DEC instruction to be used on a loop counter in multiple-precision computations.

When operating on unsigned values, only BREQ and BRNE branches can be expected to perform consistently. When operating on two's complement values, all signed branches are available.

Operation:

(i) Rd ← Rd - 1

Syntax: Operands: Program Counter: (i) DEC Rd $0 \le d \le 31$ PC \leftarrow PC + 1

16-bit Opcode:

1001	010d	dddd	1010

Status Register and Boolean Formula:

ı	T	Н	S	V	N	Z	С
-	-	-	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	-

S: $N \oplus V$

For signed tests.

V: R7 •R6 •R5 •R4• R3• R2 •R1• R0

Set if two's complement overflow resulted from the operation; cleared otherwise. Two's complement overflow occurs if and only if Rd was \$80 before the operation.

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7 •R6• R5 •R4• R3• R2• R1• R0
Set if the result is \$00; Cleared otherwise.

R (Result) equals Rd after the operation.

Example:

ldi r17,\$10 ; Load constant in r17
loop: add r1,r2 ; Add r2 to r1
dec r17 ; Decrement r17
brne loop ; Branch if r17<>0
nop ; Continue (do nothing)

Words: 1 (2 bytes)

Cycles: 1



65





DES – Data Encryption Standard

Description:

The module is an instruction set extension to the AVR CPU, performing DES iterations. The 64-bit data block (plaintext or ciphertext) is placed in the CPU register file, registers R0-R7, where LSB of data is placed in LSB of R0 and MSB of data is placed in MSB of R7. The full 64-bit key (including parity bits) is placed in registers R8-R15, organized in the register file with LSB of key in LSB of R8 and MSB of key in MSB of R15. Executing one DES instruction performs one round in the DES algorithm. Sixteen rounds must be executed in increasing order to form the correct DES ciphertext or plaintext. Intermediate results are stored in the register file (R0-R15) after each DES instruction. The instruction's operand (K) determines which round is executed, and the half carry flag (H) determines whether encryption or decryption is performed.

The DES algorithm is described in "Specifications for the Data Encryption Standard" (Federal Information Processing Standards Publication 46). Intermediate results in this implementation differ from the standard because the initial permutation and the inverse initial permutation are performed each iteration. This does not affect the result in the final ciphertext or plaintext, but reduces execution time.

Operation:

(i) If H = 0 then Encrypt round (R7-R0, R15-R8, K) If H = 1 then Decrypt round (R7-R0, R15-R8, K)

Syntax: Operands: **Program Counter:** (i) 0x00≤K≤ 0x0F $PC \leftarrow PC + 1$

DES K

16-bit Opcode:

1001	0100	KKKK	1011

Example:

DES 0x00 DES 0x01 DES 0x0E DES 0x0F

Words: 1 Cycles: 1 (2⁽¹⁾)

Note: 1. If the DES instruction is succeeding a non-DES instruction, an extra cycle is inserted.

AVR Instruction Set =

0856I-AVR-07/10

66



EICALL – Extended Indirect Call to Subroutine

Description:

Indirect call of a subroutine pointed to by the Z (16 bits) Pointer Register in the Register File and the EIND Register in the I/O space. This instruction allows for indirect calls to the entire 4M (words) Program memory space. See also ICALL. The Stack Pointer uses a post-decrement scheme during EICALL.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) $PC(15:0) \leftarrow Z(15:0)$ $PC(21:16) \leftarrow EIND$

Syntax: Operands: Program Counter: Stack:

(i) EICALL None See Operation STACK \leftarrow PC + 1

 $SP \leftarrow SP - 3$ (3 bytes, 22 bits)

16-bit Opcode:

1001	0101	0001	1001
------	------	------	------

Status Register (SREG) and Boolean Formula:

I	T	Н	S	V	N	Z	С
-	-	-	_	_	-	_	_

Example:

ldi r16,\$05 ; Set up EIND and Z-pointer
out EIND,r16
ldi r30,\$00
ldi r31,\$10
eicall ; Call to \$051000

Words : 1 (2 bytes)

Cycles: 4 (only implemented in devices with 22 bit PC)

Cycles XMEGA: 3 (only implemented in devices with 22 bit PC)







EIJMP - Extended Indirect Jump

Description:

Indirect jump to the address pointed to by the Z (16 bits) Pointer Register in the Register File and the EIND Register in the I/O space. This instruction allows for indirect jumps to the entire 4M (words) Program memory space. See also IJMP.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) $PC(15:0) \leftarrow Z(15:0)$ $PC(21:16) \leftarrow EIND$

Syntax: Operands: Program Counter: Stack:
(i) EIJMP None See Operation Not Affected

16-bit Opcode:

1001	0100	0001	1001
------	------	------	------

Status Register (SREG) and Boolean Formula:

ı	Т	Н	S	V	N	Z	С
	-	-	-	-	-	-	-

Example:

Words: 1 (2 bytes)

Cycles: 2

68 AVR Instruction Set



ELPM – Extended Load Program Memory

Description:

Loads one byte pointed to by the Z-register and the RAMPZ Register in the I/O space, and places this byte in the destination register Rd. This instruction features a 100% space effective constant initialization or constant data fetch. The Program memory is organized in 16-bit words while the Z-pointer is a byte address. Thus, the least significant bit of the Z-pointer selects either low byte ($Z_{LSB} = 0$) or high byte ($Z_{LSB} = 1$). This instruction can address the entire Program memory space. The Z-pointer Register can either be left unchanged by the operation, or it can be incremented. The incrementation applies to the entire 24-bit concatenation of the RAMPZ and Z-pointer Registers.

Devices with Self-Programming capability can use the ELPM instruction to read the Fuse and Lock bit value. Refer to the device documentation for a detailed description.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

The result of these combinations is undefined:

ELPM r30, Z+ ELPM r31, Z+

Operation:

$\begin{array}{ll} \text{(i)} & \mathsf{R0} \leftarrow (\mathsf{RAMPZ:Z}) \\ \text{(ii)} & \mathsf{Rd} \leftarrow (\mathsf{RAMPZ:Z}) \\ \end{array}$

(iii) $Rd \leftarrow (RAMPZ:Z) \leftarrow (RAMPZ:Z) \leftarrow (RAMPZ:Z) + 1$

Comment:

RAMPZ:Z: Unchanged, R0 implied destination register

RAMPZ:Z: Unchanged

RAMPZ:Z: Post incremented

Syntax: Operands:

(i)	ELPM	None, R0 implied
(ii)	ELPM Rd, Z	$0 \le d \le 31$
(iii)	ELPM Rd, Z+	$0 \leq d \leq 31$

Program Counter:

 $PC \leftarrow PC + 1$ $PC \leftarrow PC + 1$ $PC \leftarrow PC + 1$

16 bit Opcode:

	(i)	1001	0101	1101	1000
Г	(ii)	1001	D000	dddd	0110
Г	(iii)	1001	000d	dddd	0111

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С	
-	-	-	-	=	-	-	-	

Example:

AMEL

69





. . .

Words: 1 (2 bytes)

Cycles: 3

70 AVR Instruction Set



EOR - Exclusive OR

Description:

Performs the logical EOR between the contents of register Rd and register Rr and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd \oplus Rr$

 $\begin{tabular}{lll} Syntax: & Operands: & Program Counter: \\ EOR Rd,Rr & 0 \le d \le 31, \ 0 \le r \le 31 & PC \leftarrow PC + 1 \\ & & 16\mbox{-bit Opcode:} \\ \end{tabular}$

0010	01rd	5555	rrrr
0010	olla	adda	TITI

Status Register (SREG) and Boolean Formula:

I	T	Н	S	V	N	Z	С	
-	_	-	⇔	0	\Leftrightarrow	⇔	_]

S: $N \oplus V$, For signed tests.

V: 0

Cleared

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7 •R6 •R5 •R4• R3• R2 •R1• R0
Set if the result is \$00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

eor r4,r4 ; Clear r4

eor r0,r22 ; Bitwise exclusive or between r0 and r22

Words: 1 (2 bytes)







FMUL - Fractional Multiply Unsigned

Description:

This instruction performs 8-bit × 8-bit → 16-bit unsigned multiplication and shifts the result one bit left.

Rd		Rr		R1	R0	
Multiplicand	×	Multiplier	Æ	Product High	Product Low	
- 8		8	-	1	6	

Let (N.Q) denote a fractional number with N binary digits left of the radix point, and Q binary digits right of the radix point. A multiplication between two numbers in the formats (N1.Q1) and (N2.Q2) results in the format ((N1+N2).(Q1+Q2)). For signal processing applications, the format (1.7) is widely used for the inputs, resulting in a (2.14) format for the product. A left shift is required for the high byte of the product to be in the same format as the inputs. The FMUL instruction incorporates the shift operation in the same number of cycles as MUL.

The (1.7) format is most commonly used with signed numbers, while FMUL performs an unsigned multiplication. This instruction is therefore most useful for calculating one of the partial products when performing a signed multiplication with 16-bit inputs in the (1.15) format, yielding a result in the (1.31) format. Note: the result of the FMUL operation may suffer from a 2's complement overflow if interpreted as a number in the (1.15) format. The MSB of the multiplication before shifting must be taken into account, and is found in the carry bit. See the following example.

The multiplicand Rd and the multiplier Rr are two registers containing unsigned fractional numbers where the implicit radix point lies between bit 6 and bit 7. The 16-bit unsigned fractional product with the implicit radix point between bit 14 and bit 15 is placed in R1 (high byte) and R0 (low byte).

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) R1:R0 \leftarrow Rd \times Rr (unsigned (1.15) \leftarrow unsigned (1.7) \times unsigned (1.7))

Syntax: Operands: Program Counter: (i) FMUL Rd,Rr $16 \le d \le 23$, $16 \le r \le 23$ PC \leftarrow PC + 1

16-bit Opcode:

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С	
_	_	_	-	-	-	\Leftrightarrow	\Leftrightarrow]

C: R16

Set if bit 15 of the result before left shift is set; cleared otherwise.

Z: R15 •R14 •R13 •R12 •R11 •R10 •R9 •R8 •R7 • R6 • R5 • R4 • R3 • R2 •R1 • R0 Set if the result is \$0000; cleared otherwise.

R (Result) equals R1,R0 after the operation.

72 AVR Instruction Set



Example:

```
;* DESCRIPTION
;*Signed fractional multiply of two 16-bit numbers with 32-bit result.
;*r19:r18:r17:r16 = ( r23:r22 * r21:r20 ) << 1
fmuls16x16_32:
 clrr2
 fmulsr23, r21;((signed)ah * (signed)bh) << 1</pre>
 movwr19:r18, r1:r0
 fmulr22, r20; (al * bl) << 1
 adcr18, r2
 movwr17:r16, r1:r0
 fmulsur23, r20; ((signed)ah * bl) << 1
 sbcr19, r2
 addr17, r0
 adcr18, r1
 adcr19, r2
 fmulsur21, r22;((signed)bh * al) << 1
 sbcr19, r2
 addr17, r0
 adcr18, r1
 adcr19, r2
```

Words: 1 (2 bytes)







FMULS - Fractional Multiply Signed

Description:

This instruction performs 8-bit × 8-bit → 16-bit signed multiplication and shifts the result one bit left.

Rd		Rr		R1	R0
Multiplicand	×	Multiplier	\rightarrow	Product High	Product Low
8		8		1	6

Let (N.Q) denote a fractional number with N binary digits left of the radix point, and Q binary digits right of the radix point. A multiplication between two numbers in the formats (N1.Q1) and (N2.Q2) results in the format ((N1+N2).(Q1+Q2)). For signal processing applications, the format (1.7) is widely used for the inputs, resulting in a (2.14) format for the product. A left shift is required for the high byte of the product to be in the same format as the inputs. The FMULS instruction incorporates the shift operation in the same number of cycles as MULS.

The multiplicand Rd and the multiplier Rr are two registers containing signed fractional numbers where the implicit radix point lies between bit 6 and bit 7. The 16-bit signed fractional product with the implicit radix point between bit 14 and bit 15 is placed in R1 (high byte) and R0 (low byte).

Note that when multiplying 0x80 (-1) with 0x80 (-1), the result of the shift operation is 0x8000 (-1). The shift operation thus gives a two's complement overflow. This must be checked and handled by software.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) R1:R0 \leftarrow Rd \times Rr (signed (1.15) \leftarrow signed (1.7) \times signed (1.7))

16-bit Opcode:

0000	0011	1ddd	0rrr
------	------	------	------

Status Register (SREG) and Boolean Formula:

ı	Т	Н	S	V	N	Z	С
_	-	-	-	-	1-1	\Leftrightarrow	\Leftrightarrow

C: R16

Set if bit 15 of the result before left shift is set; cleared otherwise.

Z: R15 •R14 •R13 •R12 •R11 •R10 •R9 •R8 •R7 • R6 • R5 • R4 • R3 • R2 •R1 • R0 Set if the result is \$0000; cleared otherwise.

R (Result) equals R1,R0 after the operation.

Example:

```
fmuls r23,r22 ; Multiply signed r23 and r22 in (1.7) format, result in (1.15) format movw r23:r22,r1:r0 ; Copy result back in r23:r22
```

74 AVR Instruction Set



Words: 1 (2 bytes)



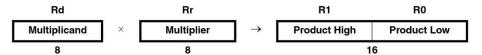




FMULSU - Fractional Multiply Signed with Unsigned

Description:

This instruction performs 8-bit × 8-bit → 16-bit signed multiplication and shifts the result one bit left.



Let (N.Q) denote a fractional number with N binary digits left of the radix point, and Q binary digits right of the radix point. A multiplication between two numbers in the formats (N1.Q1) and (N2.Q2) results in the format ((N1+N2).(Q1+Q2)). For signal processing applications, the format (1.7) is widely used for the inputs, resulting in a (2.14) format for the product. A left shift is required for the high byte of the product to be in the same format as the inputs. The FMULSU instruction incorporates the shift operation in the same number of cycles as MULSU.

The (1.7) format is most commonly used with signed numbers, while FMULSU performs a multiplication with one unsigned and one signed input. This instruction is therefore most useful for calculating two of the partial products when performing a signed multiplication with 16-bit inputs in the (1.15) format, yielding a result in the (1.31) format. Note: the result of the FMULSU operation may suffer from a 2's complement overflow if interpreted as a number in the (1.15) format. The MSB of the multiplication before shifting must be taken into account, and is found in the carry bit. See the following example.

The multiplicand Rd and the multiplier Rr are two registers containing fractional numbers where the implicit radix point lies between bit 6 and bit 7. The multiplicand Rd is a signed fractional number, and the multiplier Rr is an unsigned fractional number. The 16-bit signed fractional product with the implicit radix point between bit 14 and bit 15 is placed in R1 (high byte) and R0 (low byte).

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) R1:R0 \leftarrow Rd \times Rr (signed (1.15) \leftarrow signed (1.7) \times unsigned (1.7))

Syntax: Operands: Program Counter: (i) FMULSU Rd,Rr $16 \le d \le 23$, $16 \le r \le 23$ PC \leftarrow PC + 1

16-bit Opcode:

Status Register (SREG) and Boolean Formula:

Ī	Т	Н	S	V	N	Z	С
-	-	-	-		-	\Leftrightarrow	⇔

C: R16

Set if bit 15 of the result before left shift is set; cleared otherwise.

Z: R15 •R14 •R13 •R12 •R11 •R10 •R9 •R8 •R7 • R6 • R5 • R4 • R3 • R2 •R1 • R0 Set if the result is \$0000; cleared otherwise.

R (Result) equals R1,R0 after the operation.

76 AVR Instruction Set



Example:

```
; * DESCRIPTION
;*Signed fractional multiply of two 16-bit numbers with 32-bit result.
;*r19:r18:r17:r16 = ( r23:r22 * r21:r20 ) << 1
fmuls16x16_32:
 clrr2
 fmulsr23, r21;((signed)ah * (signed)bh) << 1</pre>
 movwr19:r18, r1:r0
 fmulr22, r20;(al * bl) << 1
 adcr18, r2
 movwr17:r16, r1:r0
 fmulsur23, r20;((signed)ah * bl) << 1
 sbcr19, r2
 addr17, r0
 adcr18, r1
 adcr19, r2
 fmulsur21, r22;((signed)bh * al) << 1
 sbcr19, r2
 addr17, r0
 adcr18, r1
 adcr19, r2
```

Words: 1 (2 bytes)







ICALL - Indirect Call to Subroutine

Description:

Calls to a subroutine within the entire 4M (words) Program memory. The return address (to the instruction after the CALL) will be stored onto the Stack. See also RCALL. The Stack Pointer uses a post-decrement scheme during CALL.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

- (i) PC(15:0) ← Z(15:0) Devices with 16 bits PC, 128K bytes Program memory maximum.
- (ii) PC(15:0) \leftarrow Z(15:0) Devices with 22 bits PC, 8M bytes Program memory maximum. PC(21:16) \leftarrow 0

	Syntax:	Operands:	Program Counter:	Stack:
(i)	ICALL	None	See Operation	STACK \leftarrow PC + 1 SP \leftarrow SP - 2 (2 bytes, 16 bits)
(ii)	ICALL	None	See Operation	STACK \leftarrow PC + 1 SP \leftarrow SP - 3 (3 bytes, 22 bits)

16-bit Opcode:

100	2.1	0101	0000	1001
100) T	0101	0000	1001

Status Register (SREG) and Boolean Formula:

1	T	Н	S	V	N	z	С
-	_	-	-	-	-	-	-

Example:

mov r30,r0 ; Set offset to call table

icall ; Call routine pointed to by r31:r30

Words: 1 (2 bytes)

Cycles: 3, devices with 16 bit PC

4, devices with 22 bit PC

Cycles XMEGA: 2, devices with 16 bit PC

3, devices with 22 bit PC



IJMP – Indirect Jump

Description:

Indirect jump to the address pointed to by the Z (16 bits) Pointer Register in the Register File. The Z-pointer Register is 16 bits wide and allows jump within the lowest 64K words (128K bytes) section of Program memory.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

- (i) PC \leftarrow Z(15:0) Devices with 16 bits PC, 128K bytes Program memory maximum.
- (ii) $PC(15:0) \leftarrow Z(15:0)$ Devices with 22 bits PC, 8M bytes Program memory maximum. $PC(21:16) \leftarrow 0$

Syntax: Operands: Program Counter: Stack:
(i),(ii) IJMP None See Operation Not Affected

16-bit Opcode:

1001	0100	0000	1001
1001	0100	0000	1001

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С	
-	-	-	-	=	-	-	-	

Example:

mov r30,r0 ; Set offset to jump table

ijmp ; Jump to routine pointed to by r31:r30

Words: 1 (2 bytes)







IN - Load an I/O Location to Register

Description:

Loads data from the I/O Space (Ports, Timers, Configuration Registers etc.) into register Rd in the Register File.

Operation:

(i) $Rd \leftarrow I/O(A)$

16-bit Opcode:

	1011	0AAd	dddd	AAAA
--	------	------	------	------

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С
-	-	-	-	-	-	-	-

Example:

in r25,\$16 ; Read Port B

cpi r25,4 ; Compare read value to constant

breq exit ; Branch if r25=4

...

exit: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)



INC - Increment

Description:

Adds one -1- to the contents of register Rd and places the result in the destination register Rd.

The C Flag in SREG is not affected by the operation, thus allowing the INC instruction to be used on a loop counter in multiple-precision computations.

When operating on unsigned numbers, only BREQ and BRNE branches can be expected to perform consistently. When operating on two's complement values, all signed branches are available.

Operation:

(i) $Rd \leftarrow Rd + 1$

16-bit Opcode:

	1001	010d	dddd	0011
--	------	------	------	------

Status Register and Boolean Formula:

I	Т	Н	S	V	N	Z	С
-	_	-	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	-

S: $N \oplus V$

For signed tests.

V: R7 •R6 •R5 •R4 •R3• R2 •R1 •R0

Set if two's complement overflow resulted from the operation; cleared otherwise. Two's complement overflow occurs if and only if Rd was \$7F before the operation.

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7 •R6 •R5 •R4•R3 •R2• R1• R0
Set if the result is \$00; Cleared otherwise.

R (Result) equals Rd after the operation.

Example:







Words: 1 (2 bytes)

Cycles: 1

82 AVR Instruction Set



JMP - Jump

Description:

Jump to an address within the entire 4M (words) Program memory. See also RJMP.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) $PC \leftarrow k$

	Syntax:	Operands:	Program Counter:	Stack:
(i)	JMP k	$0 \le k < 4M$	$PC \leftarrow k$	Unchanged

32-bit Opcode:

1001	010k	kkkk	110k
kkkk	kkkk	kkkk	kkkk

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С	
-	-	-	_	_	-	-	_	

Example:

Words: 2 (4 bytes)







Program Counter: PC ← PC + 1

LAC - Load And Clear

Description:

Operation:

(i) $(Z) \leftarrow Rd \cdot (\$FF - (Z))$

16-bit Opcode:

1001 001r rrrr 0110

Words: 1 (2 bytes)



Program Counter:

 $PC \leftarrow PC + 1$

LAS - Load And Set

Description:

Operation:

(i) $(Z) \leftarrow Rd \ v \ (Z), \ Rd \leftarrow (Z)$

16-bit Opcode:

1001 001r rrrr 0101

Words: 1 (2 bytes)







Program Counter:

 $PC \leftarrow PC + 1$

LAT – Load And Toggle

Description:

Operation:

(i) $(Z) \leftarrow Rd \oplus (Z), Rd \leftarrow (Z)$

16-bit Opcode:

1001 001r rrrr 0111

Words: 1 (2 bytes)

Cycles: 1

6 AVR Instruction Set



LD – Load Indirect from Data Space to Register using Index X

Description:

Loads one byte indirect from the data space to a register. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. In some parts the Flash Memory has been mapped to the data space and can be read using this command. The EEPROM has a separate address space.

The data location is pointed to by the X (16 bits) Pointer Register in the Register File. Memory access is limited to the current data segment of 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPX in register in the I/O area has to be changed.

The X-pointer Register can either be left unchanged by the operation, or it can be post-incremented or pre-decremented. These features are especially suited for accessing arrays, tables, and Stack Pointer usage of the X-pointer Register. Note that only the low byte of the X-pointer is updated in devices with no more than 256 bytes data space. For such devices, the high byte of the pointer is not used by this instruction and can be used for other purposes. The RAMPX Register in the I/O area is updated in parts with more than 64K bytes data space or more than 64K bytes Program memory, and the increment/decrement is added to the entire 24-bit address on such devices.

Not all variants of this instruction is available in all devices. Refer to the device specific instruction set summary.

In the Reduced Core tinyAVR the LD instruction can be used to achieve the same operation as LPM since the program memory is mapped to the data memory space.

Comment:

The result of these combinations is undefined:

LD r26, X+ LD r27, X+ LD r26, -X LD r27, -X

Operation:

Using the X-pointer:

(i)	$Rd \leftarrow (X)$		X: Unchanged
(ii)	$Rd \leftarrow (X)$	$X \leftarrow X + 1$	X: Post incremented
(iii)	X ← X - 1	$Rd \leftarrow (X)$	X: Pre decremented
	Syntax:	Operands:	Program Counter:
(i)	LD Rd, X	$0 \le d \le 31$	PC ← PC + 1
(ii)	LD Rd, X+	$0 \le d \le 31$	PC ← PC + 1
(iii)	LD RdX	$0 \le d \le 31$	PC ← PC + 1







16-bit Opcode:

(i)	1001	000d	dddd	1100
(ii)	1001	D000	dddd	1101
(iii)	1001	D000	dddd	1110

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С
-	-	-	-	-	-	-	-



Example:

```
clr r27
                 ; Clear X high byte
ldi r26,$60
                 ; Set X low byte to $60
ld
     r0,X+
                 ; Load r0 with data space loc. $60(X post inc)
1d
     r1,X
                 ; Load r1 with data space loc. $61
                ; Set X low byte to $63
ldi
    r26,$63
1d
                 ; Load r2 with data space loc. $63
    r2,X
ld r3,-X
                 ; Load r3 with data space loc. $62(X pre dec)
```

Words: 1 (2 bytes)

Cycles: (i) 1⁽²⁾

(ii) 2 (iii) 3⁽²⁾

Cycles XMEGA: (i) 1⁽¹⁾

(ii) 1⁽¹⁾ (iii) 2⁽¹⁾

Notes: 1. IF the LD instruction is accessing internal SRAM, one extra cycle is inserted.

2. LD instruction can load data from program memory since the flash is memory mapped. Loading data from the data memory takes 1 clock cycle, and loading from the program memory takes 2 clock cycles. But if an interrupt occur (before the last clock cycle) no additional clock cycles is necessary when loading from the program memory. Hence, the instruction takes only 1 clock cycle to execute.

LD instruction with pre-decrement can load data from program memory since the flash is memory mapped. Loading data from the data memory takes 2 clock cycles, and loading from the program memory takes 3 clock cycles. But if an interrupt occur (before the last clock cycle) no additional clock cycles is necessary when loading from the program memory. Hence, the instruction takes only 1 clock cycle to execute.







LD (LDD) – Load Indirect from Data Space to Register using Index Y

Description:

Loads one byte indirect with or without displacement from the data space to a register. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. In some parts the Flash Memory has been mapped to the data space and can be read using this command. The EEPROM has a separate address space.

The data location is pointed to by the Y (16 bits) Pointer Register in the Register File. Memory access is limited to the current data segment of 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPY in register in the I/O area has to be changed.

The Y-pointer Register can either be left unchanged by the operation, or it can be post-incremented or pre-decremented. These features are especially suited for accessing arrays, tables, and Stack Pointer usage of the Y-pointer Register. Note that only the low byte of the Y-pointer is updated in devices with no more than 256 bytes data space. For such devices, the high byte of the pointer is not used by this instruction and can be used for other purposes. The RAMPY Register in the I/O area is updated in parts with more than 64K bytes data space or more than 64K bytes Program memory, and the increment/decrement/displacement is added to the entire 24-bit address on such devices.

Not all variants of this instruction is available in all devices. Refer to the device specific instruction set summary.

In the Reduced Core tinyAVR the LD instruction can be used to achieve the same operation as LPM since the program memory is mapped to the data memory space.

The result of these combinations is undefined:

LD r28, Y+ LD r29, Y+ LD r28, -Y LD r29, -Y

Using the Y-pointer:

	Operation:		Comment:
(i)	$Rd \leftarrow (Y)$		Y: Unchanged
(ii)	$Rd \leftarrow (Y)$	$Y \leftarrow Y + 1$	Y: Post incremented
(iii)	$Y \leftarrow Y - 1$	$Rd \leftarrow (Y)$	Y: Pre decremented
(iv)	$Rd \leftarrow (Y+q)$		Y: Unchanged, q: Displacement
	Syntax:	Operands:	Program Counter:
(i)	LD Rd, Y	$0 \le d \le 31$	PC ← PC + 1
(ii)	LD Rd, Y+	$0 \le d \le 31$	PC ← PC + 1
(iii)	LD Rd, -Y	$0 \le d \le 31$	PC ← PC + 1
(iv)	LDD Rd, Y+q	$0 \le d \le 31, \ 0 \le q \le 63$	PC ← PC + 1

AVR Instruction Set

0856I-AVR-07/10

90



16-bit Opcode:

	(i)	1000	D000	dddd	1000
	(ii)	1001	D000	dddd	1001
	(iii)	1001	D000	dddd	1010
Г	(iv)	10q0	qq0d	dddd	1qqq

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С	
=	_	_	_	_	_	-	_	

Example:

clr r29 ; Clear Y high byte ldi r28,\$60 ; Set Y low byte to \$60 ld r0,Y+ ; Load r0 with data space loc. \$60(Y post inc) ld r1,Y ; Load r1 with data space loc. \$61 ldi r28,\$63 ; Set Y low byte to \$63 1d r2.Y ; Load r2 with data space loc. \$63 1d r3,-Y ; Load r3 with data space loc. \$62(Y pre dec) ldd r4,Y+2 ; Load r4 with data space loc. \$64

Words: 1 (2 bytes)

Cycles:

(i) 1⁽²⁾ (ii) 2

(iii) 3⁽²⁾

Cycles XMEGA:

(i) 1⁽¹⁾ (ii) 1⁽¹⁾

(iii) 2⁽¹⁾

(iv) 2⁽¹⁾

- Notes: 1. IF the LD instruction is accessing internal SRAM, one extra cycle is inserted.
 - 2. LD instruction can load data from program memory since the flash is memory mapped. Loading data from the data memory takes 1 clock cycle, and loading from the program memory takes 2 clock cycles. But if an interrupt occur (before the last clock cycle) no additional clock cycles is necessary when loading from the program memory. Hence, the instruction takes only 1 clock cycle to execute.

LD instruction with pre-decrement can load data from program memory since the flash is memory mapped. Loading data from the data memory takes 2 clock cycles, and loading from the program memory takes 3 clock cycles. But if an interrupt occur (before the last clock cycle) no additional clock cycles is necessary when loading from the program memory. Hence, the instruction takes only 1 clock cycle to execute.







LD (LDD) – Load Indirect From Data Space to Register using Index Z

Description:

Loads one byte indirect with or without displacement from the data space to a register. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. In some parts the Flash Memory has been mapped to the data space and can be read using this command. The EEPROM has a separate address space.

The data location is pointed to by the Z (16 bits) Pointer Register in the Register File. Memory access is limited to the current data segment of 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPZ in register in the I/O area has to be changed.

The Z-pointer Register can either be left unchanged by the operation, or it can be post-incremented or pre-decremented. These features are especially suited for Stack Pointer usage of the Z-pointer Register, however because the Z-pointer Register can be used for indirect subroutine calls, indirect jumps and table lookup, it is often more convenient to use the X or Y-pointer as a dedicated Stack Pointer. Note that only the low byte of the Z-pointer is updated in devices with no more than 256 bytes data space. For such devices, the high byte of the pointer is not used by this instruction and can be used for other purposes. The RAMPZ Register in the I/O area is updated in parts with more than 64K bytes data space or more than 64K bytes Program memory, and the increment/decrement/displacement is added to the entire 24-bit address on such devices.

Not all variants of this instruction is available in all devices. Refer to the device specific instruction set summary.

In the Reduced Core tinyAVR the LD instruction can be used to achieve the same operation as LPM since the program memory is mapped to the data memory space.

For using the Z-pointer for table lookup in Program memory see the LPM and ELPM instructions.

The result of these combinations is undefined:

LD r30, Z+ LD r31, Z+ LD r30, -Z LD r31, -Z

Using the Z-pointer:

	Operation:	Comment:	
(i)	$Rd \leftarrow (Z)$		Z: Unchanged
(ii)	$Rd \leftarrow (Z)$	$Z \leftarrow Z + 1$	Z: Post increment
(iii)	$Z \leftarrow Z - 1$	$Rd \leftarrow (Z)$	Z: Pre decrement
(iv)	$Rd \leftarrow (Z+q)$		Z: Unchanged, q: Displacement
	Syntax:	Operands:	Program Counter:
(i)	Syntax: LD Rd, Z	Operands: 0 ≤ d ≤ 31	Program Counter: PC ← PC + 1
(i) (ii)		est [®] rest	
	LD Rd, Z	$0 \le d \le 31$	PC ← PC + 1

92 AVR Instruction Set



16-bit Opcode:

(i)	1000	D000	dddd	0000
(ii)	1001	000d	dddd	0001
(iii)	1001	D000	dddd	0010
(iv)	10q0	qq0d	dddd	0qqq

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С	
=	_	_	_	_	_	-	_	

Example:

; Clear Z high byte clr r31 ldi r30,\$60 ; Set Z low byte to \$60 ld r0,Z+ ; Load r0 with data space loc. \$60(Z post inc) ld r1,Z ; Load r1 with data space loc. \$61 ldi r30,\$63 ; Set Z low byte to \$63 ld r2,Z ; Load r2 with data space loc. \$63 ld r3,-Z ; Load r3 with data space loc. \$62(Z pre dec) ldd r4,Z+2 ; Load r4 with data space loc. \$64

Words: 1 (2 bytes)

Cycles:

(i) 1⁽²⁾ (ii) 2

(iii) 3⁽²⁾

Cycles XMEGA:

(i) 1⁽¹⁾ (ii) 1⁽¹⁾

(iii) 2⁽¹⁾ (iv) 2⁽¹⁾

- Notes: 1. IF the LD instruction is accessing internal SRAM, one extra cycle is inserted.
 - 2. LD instruction can load data from program memory since the flash is memory mapped. Loading data from the data memory takes 1 clock cycle, and loading from the program memory takes 2 clock cycles. But if an interrupt occur (before the last clock cycle) no additional clock cycles is necessary when loading from the program memory. Hence, the instruction takes only 1 clock cycle to execute.

LD instruction with pre-decrement can load data from program memory since the flash is memory mapped. Loading data from the data memory takes 2 clock cycles, and loading from the program memory takes 3 clock cycles. But if an interrupt occur (before the last clock cycle) no additional clock cycles is necessary when loading from the program memory. Hence, the instruction takes only 1 clock cycle to execute.







LDI - Load Immediate

Description:

Loads an 8 bit constant directly to register 16 to 31.

Operation:

(i) $Rd \leftarrow K$

16-bit Opcode:

1110	KKKK	dddd	KKKK

Status Register (SREG) and Boolean Formula:

ı	Т	Н	s	V	N	Z	С	
-	_	-	-	_	-	-	_]

Example:

clr r31 ; Clear Z high byte
ldi r30,\$F0 ; Set Z low byte to \$F0
lpm ; Load constant from Program
; memory pointed to by Z

Words: 1 (2 bytes)

Cycles: 1



LDS - Load Direct from Data Space

Description:

Loads one byte from the data space to a register. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the register file only. The EEPROM has a separate address space.

A 16-bit address must be supplied. Memory access is limited to the current data segment of 64K bytes. The LDS instruction uses the RAMPD Register to access memory above 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPD in register in the I/O area has to be changed.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) $Rd \leftarrow (k)$

32-bit Opcode:

1001	000d	dddd	0000
kkkk	kkkk	kkkk	kkkk

Status Register (SREG) and Boolean Formula:

1	T	Н	s	V	N	Z	С	
_	_	-	_	-	_	_	_]

Example:

Words: 2 (4 bytes)

Cycles:

Cycles XMEGA: 2 If the LDS instruction is accessing internal SRAM, one extra cycle is inserted.







LDS (16-bit) - Load Direct from Data Space

Description:

Loads one byte from the data space to a register. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the register file only. In some parts the Flash memory has been mapped to the data space and can be read using this command. The EEPROM has a separate address space.

A 7-bit address must be supplied. The address given in the instruction is coded to a data space address as follows:

 $ADDR[7:0] = (\overline{INST[8]}, INST[8], INST[10], INST[9], INST[3], INST[2], INST[1], INST[0])$

Memory access is limited to the address range 0x40..0xbf.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) $Rd \leftarrow (k)$

16-bit Opcode:

1010	0kkk	dddd	kkkk
1010	0kkk	dddd	KKK.

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С
-	-	-	_	_	_	-	_

Example:

lds r16,\$00 ; Load r16 with the contents of data space location \$00 add r16,r17 ; add r17 to r16 ; Write result to the same address it was fetched from

Words: 1 (2 bytes)

Cycles: 1

Note: Registers r0..r15 are remapped to r16..r31.

AVR Instruction Set

0856I-AVR-07/10

96



LPM - Load Program Memory

Description:

Loads one byte pointed to by the Z-register into the destination register Rd. This instruction features a 100% space effective constant initialization or constant data fetch. The Program memory is organized in 16-bit words while the Z-pointer is a byte address. Thus, the least significant bit of the Z-pointer selects either low byte ($Z_{LSB} = 0$) or high byte ($Z_{LSB} = 1$). This instruction can address the first 64K bytes (32K words) of Program memory. The Z-pointer Register can either be left unchanged by the operation, or it can be incremented. The incrementation does not apply to the RAMPZ Register.

Devices with Self-Programming capability can use the LPM instruction to read the Fuse and Lock bit values. Refer to the device documentation for a detailed description.

The LPM instruction is not available in all devices. Refer to the device specific instruction set summary.

The result of these combinations is undefined:

LPM r30, Z+ LPM r31, Z+

Operation:

(i)	$R0 \leftarrow (Z)$
(ii)	$Rd \leftarrow (Z)$

LPM

(i)

(iii)	$Rd \leftarrow (Z)$	
	Syntax:	

$Z \leftarrow Z + 1$

Operands: None, R0 implied

LPM Rd, Z (ii) $0 \le d \le 31$ (iii) LPM Rd, Z+ $0 \le d \le 31$

16-bit Opcode:

(i)	1001	0101	1100	1000
(ii)	1001	D000	dddd	0100
(iii)	1001	D000	dddd	0101

Comment:

- Z: Unchanged, R0 implied destination register
- Z: Unchanged
- Z: Post incremented

Program Counter:

$PC \leftarrow PC +$	1
$PC \leftarrow PC +$	1
PC ← PC +	1

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С
-	_	-	-	_	_	_	_

Example:

```
ZH, high(Table_1<<1); Initialize Z-pointer
   ldi ZL, low(Table_1<<1)
                            ; Load constant from Program
   lpm r16, Z
                            ; Memory pointed to by Z (r31:r30)
Table_1:
.dw 0x5876
                            ; 0x76 is addresses when \rm Z_{LSB} = 0
                            ; 0x58 is addresses when \rm Z_{LSB} = 1
```



97





Words: 1 (2 bytes)

Cycles: 3

98 AVR Instruction Set



LSL - Logical Shift Left

Description:

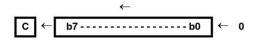
Shifts all bits in Rd one place to the left. Bit 0 is cleared. Bit 7 is loaded into the C Flag of the SREG. This operation effectively multiplies signed and unsigned values by two.

Program Counter:

 $PC \leftarrow PC + 1$

Operation:

(i)



11dd

Status Register (SR	REG) and Boolean Formula	a:

dddd

1	Т	Н	S	V	N	Z	С	
-	-	\Leftrightarrow	⇔	⇔	⇔	⇔	⇔	

dddd

H: Rd3

0000

S: $N \oplus V$, For signed tests.

V: N ⊕ C (For N and C after the shift)

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7• R6 •R5• R4• R3 •R2• R1• R0
Set if the result is \$00; cleared otherwise.

C: Rd7

Set if, before the shift, the MSB of Rd was set; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

add r0,r4 ; Add r4 to r0 lsl r0 ; Multiply r0 by 2

Words: 1 (2 bytes)

Cycles: 1





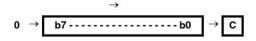


LSR - Logical Shift Right

Description:

Shifts all bits in Rd one place to the right. Bit 7 is cleared. Bit 0 is loaded into the C Flag of the SREG. This operation effectively divides an unsigned value by two. The C Flag can be used to round the result.

Operation:



Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001	010d	dddd	0110

Status Register (SREG) and Boolean Formula:

1	T	Н	S	V	N	z	С
-	=	-	\Leftrightarrow	\Leftrightarrow	0	\Leftrightarrow	\Leftrightarrow

S: $N \oplus V$, For signed tests.

 $V{:} \qquad N \oplus C \text{ (For N and C after the shift)} \\$

N: 0

Z: R7• R6 •R5• R4• R3 •R2• R1• R0
Set if the result is \$00; cleared otherwise.

C: Rd0

Set if, before the shift, the LSB of Rd was set; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

add r0,r4 ; Add r4 to r0 lsr r0 ; Divide r0 by 2

Words: 1 (2 bytes)

Cycles: 1

100 AVR Instruction Set



MOV - Copy Register

Description:

This instruction makes a copy of one register into another. The source register Rr is left unchanged, while the destination register Rd is loaded with a copy of Rr.

Operation:

(i) $Rd \leftarrow Rr$

16-bit Opcode:

0010	11rd	dddd	rrrr
------	------	------	------

Status Register (SREG) and Boolean Formula:

1	Т	н	S	V	N	Z	С	
-	_	-	-	3-3	-	-	×	

Example:

mov r16,r0 ; Copy r0 to r16
call check ; Call subroutine
...
check: cpi r16,\$11 ; Compare r16 to \$11
...
ret ; Return from subroutine

Words: 1 (2 bytes)

Cycles: 1





MOVW - Copy Register Word

Description:

This instruction makes a copy of one register pair into another register pair. The source register pair Rr+1:Rr is left unchanged, while the destination register pair Rd+1:Rd is loaded with a copy of Rr + 1:Rr.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) Rd+1:Rd ← Rr+1:Rr

Syntax: Operands: Program Counter:

(i) MOVW Rd+1:Rd,Rr+1Rrd \in {0,2,...,30}, $r \in$ {0,2,...,30}

 $PC \leftarrow PC + 1$

16-bit Opcode:

0000	0001	dddd	rrrr
------	------	------	------

Status Register (SREG) and Boolean Formula:

1	Т	н	S	V	N	Z	С
-)—.:	-	_		-	-	-

Example:

movw r17:16,r1:r0 ; Copy r1:r0 to r17:r16
call check ; Call subroutine
...
check: cpi r16,\$11 ; Compare r16 to \$11
...
cpi r17,\$32 ; Compare r17 to \$32
...
ret ; Return from subroutine

Words: 1 (2 bytes)

Cycles: 1

AVR Instruction Set

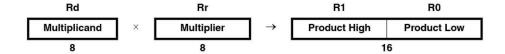
102



MUL – Multiply Unsigned

Description:

This instruction performs 8-bit \times 8-bit \rightarrow 16-bit unsigned multiplication.



The multiplicand Rd and the multiplier Rr are two registers containing unsigned numbers. The 16-bit unsigned product is placed in R1 (high byte) and R0 (low byte). Note that if the multiplicand or the multiplier is selected from R0 or R1 the result will overwrite those after multiplication.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) R1:R0 \leftarrow Rd \times Rr (unsigned \leftarrow unsigned \times unsigned)

16-bit Opcode:

1001 11rd dddd rrrr

Status Register (SREG) and Boolean Formula:

- 1	Т	Н	S	V	N	Z	С
-	_	-	_	_	-	⇔	⇔

C: R15

Set if bit 15 of the result is set; cleared otherwise.

Z: R15 •R14 •R13 •R12 •R11 •R10 •R9 •R8 •R7 • R6 • R5 • R4 • R3 • R2 •R1 • R0 Set if the result is \$0000; cleared otherwise.

R (Result) equals R1,R0 after the operation.

Example:

mul r5,r4 ; Multiply unsigned r5 and r4
movw r4,r0 ; Copy result back in r5:r4

Words: 1 (2 bytes)

Cycles: 2



103

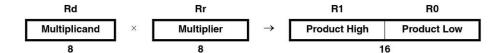




MULS – Multiply Signed

Description:

This instruction performs 8-bit \times 8-bit \rightarrow 16-bit signed multiplication.



The multiplicand Rd and the multiplier Rr are two registers containing signed numbers. The 16-bit signed product is placed in R1 (high byte) and R0 (low byte).

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation

(i) R1:R0 \leftarrow Rd \times Rr (signed \leftarrow signed \times signed)

Syntax: Operands: Program Counter: (i) MULS Rd,Rr $16 \le d \le 31$, $16 \le r \le 31$ PC \leftarrow PC + 1

16-bit Opcode:

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С	
-	-	-	-	-	_	⇔	\Leftrightarrow]

C: R15

Set if bit 15 of the result is set; cleared otherwise.

Z: R15 •R14 •R13 •R12 •R11 •R10 •R9 •R8 •R7 • R6 • R5 • R4 • R3 • R2 •R1 • R0 Set if the result is \$0000; cleared otherwise.

R (Result) equals R1,R0 after the operation.

Example:

muls r21,r20 ; Multiply signed r21 and r20 movw r20,r0 ; Copy result back in r21:r20

Words: 1 (2 bytes)
Cycles: 2

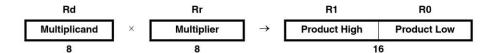
104 AVR Instruction Set



MULSU – Multiply Signed with Unsigned

Description:

This instruction performs 8-bit \times 8-bit \rightarrow 16-bit multiplication of a signed and an unsigned number.



The multiplicand Rd and the multiplier Rr are two registers. The multiplicand Rd is a signed number, and the multiplier Rr is unsigned. The 16-bit signed product is placed in R1 (high byte) and R0 (low byte).

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) R1:R0 \leftarrow Rd \times Rr (signed \leftarrow signed \times unsigned)

(i)	Synt MUL	ax: .SU Rd,Rr	Opera 16 ≤ d	nds: $1 \le 23$, $16 \le r \le 23$	Program Counter: PC ← PC + 1
	16-bi	it Opcode:			
	0000	0011	0ddd	0rrr	

Status Register (SREG) and Boolean Formula:

ı	Т	н	s	V	N	Z	С	
-	-	-	-	-	-	\Leftrightarrow	\Leftrightarrow	

- C: R15
 - Set if bit 15 of the result is set; cleared otherwise.
- Z: R15 •R14 •R13 •R12 •R11 •R10 •R9 •R8 •R7 R6 R5 R4 R3 R2 •R1 R0 Set if the result is \$0000; cleared otherwise.

R (Result) equals R1,R0 after the operation.

Example:

AIMEL

105





```
movwr19:r18, r1:r0
mulr22, r20; al * bl
movwr17:r16, r1:r0
mulsur23, r20; (signed)ah * bl
sbcr19, r2
addr17, r0
adcr18, r1
adcr19, r2
mulsur21, r22; (signed)bh * al
sbcr19, r2
addr17, r0
adcr18, r1
adcr19, r2
ret
```

Words: 1 (2 bytes)

Cycles: 2

106 AVR Instruction Set



NEG – Two's Complement

Description:

Replaces the contents of register Rd with its two's complement; the value \$80 is left unchanged.

Operation:

(i) Rd ← \$00 - Rd

16-bit Opcode:

1001	010d	dddd	0001
0.000,000,000,000	F16997771107058	200000000000000000000000000000000000000	

Status Register (SREG) and Boolean Formula:

1	T	Н	s	V	N	Z	С	
=	-	⇔	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	

H: R3 + Rd3

Set if there was a borrow from bit 3; cleared otherwise

S: N⊕V

For signed tests.

V: R7• R6 •R5• R4• R3 •R2• R1• R0

Set if there is a two's complement overflow from the implied subtraction from zero; cleared otherwise. A two's complement overflow will occur if and only if the contents of the Register after operation (Result) is \$80.

N: R7

Set if MSB of the result is set; cleared otherwise.

- Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$ Set if the result is \$00; Cleared otherwise.
- C: R7 + R6 + R5 + R4 + R3 + R2 + R1 + R0 Set if there is a borrow in the implied subtraction from zero; cleared otherwise. The C Flag will be set in all cases except when the contents of Register after operation is \$00.

R (Result) equals Rd after the operation.

Example:

```
sub r11,r0 ; Subtract r0 from r11
brpl positive ; Branch if result positive
neg r11 ; Take two's complement of r11
positive: nop ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1

<u>AIMEL</u>

107





NOP - No Operation

Description:

This instruction performs a single cycle No Operation.

Operation:

(i) No

Syntax: Operands: (i) NOP None

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

0000 0000 0000 0000

Status Register (SREG) and Boolean Formula:

1	Т	н	S	V	N	Z	С
_	-	(/ - -3	_	_	_	_	_

Example:

clr r16 ; Clear r16 ser r17 ; Set r17

out \$18,r16 ; Write zeros to Port B
nop ; Wait (do nothing)
out \$18,r17 ; Write ones to Port B

Words: 1 (2 bytes)

Cycles: 1



OR - Logical OR

Description:

Performs the logical OR between the contents of register Rd and register Rr and places the result in the destination register Rd

Operation:

(i) $Rd \leftarrow Rd \vee Rr$

0010 10rd dddd rrrr

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С
=	-	_	⇔	0	⇔	\Leftrightarrow	-

S: $N \oplus V$, For signed tests.

V: 0

Cleared

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7• R6 •R5• R4• R3 •R2• R1• R0
Set if the result is \$00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

ok:

or r15,r16 ; Do bitwise or between registers
bst r15,6 ; Store bit 6 of r15 in T Flag
brts ok ; Branch if T Flag set
...
nop ; Branch destination (do nothing)

Words: 1 (2 bytes)
Cycles: 1







ORI - Logical OR with Immediate

Description:

Performs the logical OR between the contents of register Rd and a constant and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd \vee K$

0.....

Syntax:Operands:ORI Rd,K $16 \le d \le 31$

Program Counter:

16-bit Opcode:

0110 KKKK dddd KKKK

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	٧	N	Z	С	
=	-	_	\Leftrightarrow	0	\Leftrightarrow	\Leftrightarrow	-	

S: $N \oplus V$, For signed tests.

V: 0

Cleared

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7• R6 •R5• R4• R3 •R2• R1• R0
Set if the result is \$00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

ori r16,\$F0 ; Set high nibble of r16 ori r17,1 ; Set bit 0 of r17

Words: 1 (2 bytes)

Cycles: 1

110 AVR Instruction Set



OUT - Store Register to I/O Location

Description:

Stores data from register Rr in the Register File to I/O Space (Ports, Timers, Configuration Registers etc.).

Operation:

(i) $I/O(A) \leftarrow Rr$

16-bit Opcode:

1011 1AAr rrrr AAAA

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С
-	·-	(-)	_	_	-	-	_

Example:

clr r16 ; Clear r16 ser r17 ; Set r17

Words: 1 (2 bytes)

Cycles: 1





POP - Pop Register from Stack

Description:

This instruction loads register Rd with a byte from the STACK. The Stack Pointer is pre-incremented by 1 before the POP. This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) Rd ← STACK

	Syntax:	Operands:	Program Counter:	Stack:
(i)	POP Rd	$0 \le d \le 31$	PC ← PC + 1	$SP \leftarrow SP + 1$

16-bit Opcode:

1001	D000	dddd	1111

Status Register (SREG) and Boolean Formula:

J	Т	Н	s	V	N	Z	С
_	_	-	-	-	-	-	-

Example:

call routine ; Call subroutine ...

routine: push r14 ; Save r14 on the Stack push r13 ; Save r13 on the Stack ...

pop r13 ; Restore r13 pop r14 ; Restore r14 ret ; Return from subroutine

Words: 1 (2 bytes)
Cycles: 2

112 AVR Instruction Set



PUSH - Push Register on Stack

Description:

This instruction stores the contents of register Rr on the STACK. The Stack Pointer is post-decremented by 1 after the PUSH.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) STACK ← Rr

16-bit Opcode:

1001	001d	dddd	1111
	L001	L001 001d	L001 001d dddd

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С
-	·	-	-	-	1-	-	7-7

Example:

routine; Call subroutine call ; Save r14 on the Stack routine: push r14 r13 ; Save r13 on the Stack push pop r13 ; Restore r13 pop r14 ; Restore r14 ; Return from subroutine

Words : 1 (2 bytes)

Cycles: 2
Cycles XMEGA: 1







RCALL – Relative Call to Subroutine

Description:

Relative call to an address within PC - 2K + 1 and PC + 2K (words). The return address (the instruction after the RCALL) is stored onto the Stack. See also CALL. For AVR microcontrollers with Program memory not exceeding 4K words (8K bytes) this instruction can address the entire memory from every address location. The Stack Pointer uses a post-decrement scheme during RCALL.

Operation:

(i) $PC \leftarrow PC + k + 1$ Devices with 16 bits PC, 128K bytes Program memory maximum.

(ii) PC ← PC + k + 1 Devices with 22 bits PC, 8M bytes Program memory maximum.

(i)	Syntax: RCALL k	Operands: $-2K \le k < 2K$	Program Counter: $PC \leftarrow PC + k + 1$	Stack: STACK ← PC + 1 SP ← SP - 2 (2 bytes, 16 bits)
(ii)	RCALL k	$-2K \le k < 2K$	$PC \leftarrow PC + k + 1$	STACK \leftarrow PC + 1 SP \leftarrow SP - 3 (3 bytes, 22 bits)

16-bit Opcode:

_				
	1101	kkkk	kkkk	kkkk

Status Register (SREG) and Boolean Formula:

1	Т	н	S	V	N	Z	С	
-	_	-	-		-	-	_	

Example:

rcall routine ; Call subroutine

. . .

routine: push r14

; Save r14 on the Stack

...

pop r14

; Restore r14

ret ; Return from subroutine

Words : 1 (2 bytes)

Cycles: 3, devices with 16 bit PC

4, devices with 22 bit PC

Cycles XMEGA: 2, devices with 16 bit PC

3, devices with 22 bit PC

Cycles Reduced Core tinyAVR:4

114 AVR Instruction Set



RET - Return from Subroutine

Description:

Returns from subroutine. The return address is loaded from the STACK. The Stack Pointer uses a pre-increment scheme during RET.

Operation:

- (i) PC(15:0) ← STACK Devices with 16 bits PC, 128K bytes Program memory maximum.
- (ii) PC(21:0) ← STACKDevices with 22 bits PC, 8M bytes Program memory maximum.

(i)	Sy RE	ntax: T	Ope Non	rands: e	Program Counter: See Operation	Stack: SP←SP + 2, (2bytes,16 bits)
(ii)	RE	ΞT	Non	е	See Operation	SP←SP + 3, (3bytes,22 bits)
	16-bit Opcode:					
	1001	0101	0000	1000		

Status Register (SREG) and Boolean Formula:

1	Т	н	s	V	N	Z	С	
-	-	·	_	-	_	1-	_	

Example:

call routine ; Call subroutine ...

routine: push r14 ; Save r14 on the Stack ...

pop r14 ; Restore r14 ret ; Return from subroutine

Words: 1 (2 bytes)

Cycles: 4 devices with 16-bit PC 5 devices with 22-bit PC







RETI – Return from Interrupt

Description:

Returns from interrupt. The return address is loaded from the STACK and the Global Interrupt Flag is set.

Note that the Status Register is not automatically stored when entering an interrupt routine, and it is not restored when returning from an interrupt routine. This must be handled by the application program. The Stack Pointer uses a pre-increment scheme during RETI.

Operation:

- (i) $PC(15:0) \leftarrow STACKDevices$ with 16 bits PC, 128K bytes Program memory maximum.
- (ii) PC(21:0) ← STACKDevices with 22 bits PC, 8M bytes Program memory maximum.

(i)	Syntax: RETI	Operands: None	Program Counter: See Operation	Stack SP ← SP + 2 (2 bytes, 16 bits)
(ii)	RETI	None	See Operation	$SP \leftarrow SP + 3$ (3 bytes, 22 bits)
	16-bit Opcode:			

Status Register (SREG) and Boolean Formula:

0001

1	Т	Н	s	V	N	Z	С
1	,—,	10-2	-	·-	_	-	()

1000

l: 1

1001

The I Flag is set.

0101

Example:

extint: push r0 ; Save r0 on the Stack
...
pop r0 ; Restore r0
reti ; Return and enable interrupts

Words: 1 (2 bytes)

Cycles: 4 devices with 16-bit PC 5 devices with 22-bit PC

116 AVR Instruction Set



RJMP - Relative Jump

Description:

Relative jump to an address within PC - 2K +1 and PC + 2K (words). For AVR microcontrollers with Program memory not exceeding 4K words (8K bytes) this instruction can address the entire memory from every address location. See also JMP.

Operation:

(i) $PC \leftarrow PC + k + 1$

	Syntax:	Operands:	Program Counter:	Stack
(i)	RJMP k	$-2K \le k < 2K$	$PC \leftarrow PC + k + 1$	Unchanged

16-bit Opcode:

	1100	kkkk	kkkk	kkkk
--	------	------	------	------

Status Register (SREG) and Boolean Formula:

1	Т	н	S	V	N	Z	С
	-	-	_	-	_	-	i-

Example:

cpi r16,\$42 ; Compare r16 to \$42
brne error ; Branch if r16 <> \$42
rjmp ok ; Unconditional branch
error: add r16,r17 ; Add r17 to r16
inc r16 ; Increment r16
ok: nop ; Destination for rjmp (do nothing)

Words: 1 (2 bytes)

Cycles: 2





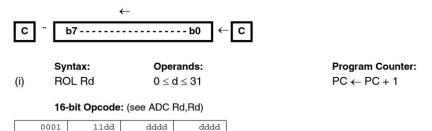


ROL - Rotate Left trough Carry

Description:

Shifts all bits in Rd one place to the left. The C Flag is shifted into bit 0 of Rd. Bit 7 is shifted into the C Flag. This operation, combined with LSL, effectively multiplies multi-byte signed and unsigned values by two.

Operation:



Status Register (SREG) and Boolean Formula:

1	Т	н	s	V	N	z	С	
-	-	⇔	⇔	⇔	⇔	\Leftrightarrow	⇔]

H: Rd3

S: $N \oplus V$, For signed tests.

V: N ⊕ C (For N and C after the shift)

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$ Set if the result is \$00; cleared otherwise.

C: Rd7

Set if, before the shift, the MSB of Rd was set; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
lsl r18 ; Multiply r19:r18 by two
rol r19 ; r19:r18 is a signed or unsigned two-byte integer
brcs oneenc ; Branch if carry set
...
oneenc: nop ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1

118 AVR Instruction Set I

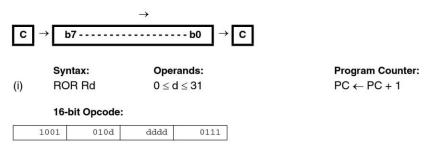


ROR – Rotate Right through Carry

Description:

Shifts all bits in Rd one place to the right. The C Flag is shifted into bit 7 of Rd. Bit 0 is shifted into the C Flag. This operation, combined with ASR, effectively divides multi-byte signed values by two. Combined with LSR it effectively divides multi-byte unsigned values by two. The Carry Flag can be used to round the result.

Operation:



Status Register (SREG) and Boolean Formula:

1	Т	н	s	V	N	Z	С	
_	_	_	⇔	⇔	⇔	\Leftrightarrow	\Leftrightarrow	

S: $N \oplus V$, For signed tests.

V: $N \oplus C$ (For N and C after the shift)

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7• R6 •R5• R4• R3 •R2• R1• R0
Set if the result is \$00; cleared otherwise.

C: Rd0

Set if, before the shift, the LSB of Rd was set; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
lsr r19 ; Divide r19:r18 by two
ror r18 ; r19:r18 is an unsigned two-byte integer
brcc zeroenc1 ; Branch if carry cleared
asr r17 ; Divide r17:r16 by two
ror r16 ; r17:r16 is a signed two-byte integer
brcc zeroenc2 ; Branch if carry cleared
...
zeroenc1: nop ; Branch destination (do nothing)
...
```

AIMEL

119





zeroenc1: nop

; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1

120 AVR Instruction Set



SBC - Subtract with Carry

Description:

Subtracts two registers and subtracts with the C Flag and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd - Rr - C$

16-bit Opcode:

000	0 10rd	dddd	rrrr
-----	--------	------	------

Status Register and Boolean Formula:

1	Т	н	s	V	N	Z	С	
-	-	⇔	\Leftrightarrow	\Leftrightarrow	⇔	\Leftrightarrow	\Leftrightarrow	

H: Rd3• Rr3 + Rr3• R3 + R3 • Rd3

Set if there was a borrow from bit 3; cleared otherwise

- S: N ⊕ V, For signed tests.
- V: Rd7 •Rr7 R7 +Rd7 •Rr7 •R7

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7• R6 •R5• R4• R3 •R2• R1• R0• Z

Previous value remains unchanged when the result is zero; cleared otherwise.

C: Rd7 •Rr7+ Rr7 •R7 +R7 •Rd7

Set if the absolute value of the contents of Rr plus previous carry is larger than the absolute value of the Rd; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

; Subtract r1:r0 from r3:r2

sub r2,r0 ; Subtract low byte

sbc r3,r1 ; Subtract with carry high byte

Words: 1 (2 bytes)

Cycles: 1



121





SBCI – Subtract Immediate with Carry

Description:

Subtracts a constant from a register and subtracts with the C Flag and places the result in the destination register Rd.

Operation:

(i) Rd ← Rd - K - C

16-bit Opcode:

0100	KKKK	5555	KKKK
0100	Terene	aaaa	Ittitit

Status Register and Boolean Formula:

1	Т	Н	s	V	N	Z	С	
=	-	\Leftrightarrow	\Leftrightarrow	⇔	\Leftrightarrow	⇔	⇔	

H: Rd3• K3 + K3• R3 + R3 • Rd3

Set if there was a borrow from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: Rd7 •K7• R7 +Rd7 •K7 •R7

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7• R6 •R5• R4• R3 •R2• R1• R0• Z

Previous value remains unchanged when the result is zero; cleared otherwise.

C: Rd7 •K7+ K7 • R7 +R7 •Rd7

Set if the absolute value of the constant plus previous carry is larger than the absolute value of Rd; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

; Subtract \$4F23 from r17:r16

subi r16,\$23 ; Subtract low byte

sbci r17,\$4F ; Subtract with carry high byte

Words: 1 (2 bytes)

Cycles: 1

122 AVR Instruction Set



SBI - Set Bit in I/O Register

Description:

Sets a specified bit in an I/O Register. This instruction operates on the lower 32 I/O Registers – addresses 0-31.

Operation:

(i) $I/O(A,b) \leftarrow 1$

16-bit Opcode:

1001 1010 AAAA Abbb

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С
-	·	(-	_	_	_	-	_

Example:

out \$1E,r0 ; Write EEPROM address sbi \$1C,0 ; Set read bit in EECR in r1,\$1D ; Read EEPROM data

Words: 1 (2 bytes)

Cycles XMEGA: 2
Cycles XMEGA: 1
Cycles Reduced Core tinyAVR:1







SBIC - Skip if Bit in I/O Register is Cleared

Description:

This instruction tests a single bit in an I/O Register and skips the next instruction if the bit is cleared. This instruction operates on the lower 32 I/O Registers – addresses 0-31.

Operation:

(i) If I/O(A,b) = 0 then $PC \leftarrow PC + 2$ (or 3) else $PC \leftarrow PC + 1$

Syntax: Operands: Program Counter:

(i) SBIC A,b $0 \le A \le 31, \ 0 \le b \le 7$ PC \leftarrow PC + 1, Condition false - no skip PC \leftarrow PC + 2, Skip a one word instruction PC \leftarrow PC + 3, Skip a two word instruction

16-bit Opcode:

1001	1001	AAAA	Abbb

Status Register (SREG) and Boolean Formula:

1	Т	н	s	V	N	Z	С	
-	_	-	_	-	_	-	_	

Example:

e2wait: sbic \$1C,1 ; Skip next inst. if EEWE cleared

rjmp e2wait ; EEPROM write not finished nop ; Continue (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false (no skip)

 ${\bf 2}$ if condition is true (skip is executed) and the instruction skipped is ${\bf 1}$ word

3 if condition is true (skip is executed) and the instruction skipped is 2 words

Cycles XMEGA: 2 if condition is false (no skip)

 ${\bf 3}$ if condition is true (skip is executed) and the instruction skipped is ${\bf 1}$ word

4 if condition is true (skip is executed) and the instruction skipped is 2 words

124 AVR Instruction Set



SBIS - Skip if Bit in I/O Register is Set

Description:

This instruction tests a single bit in an I/O Register and skips the next instruction if the bit is set. This instruction operates on the lower 32 I/O Registers – addresses 0-31.

Operation:

(i) If I/O(A,b) = 1 then $PC \leftarrow PC + 2$ (or 3) else $PC \leftarrow PC + 1$

Syntax: Operands: Program Counter:

(i) SBIS A,b $0 \le A \le 31, \ 0 \le b \le 7$ PC \leftarrow PC + 1, Condition false - no skip PC \leftarrow PC + 2, Skip a one word instruction PC \leftarrow PC + 3, Skip a two word instruction

16-bit Opcode:

1001	1011	AAAA	Abbb

Status Register (SREG) and Boolean Formula:

1	Т	н	s	V	N	Z	С
-	_	_	-	1	_	-	_

Example:

waitset: sbis \$10,0 ; Skip next inst. if bit 0 in Port D set

rjmp waitset ; Bit not set

nop ; Continue (do nothing)

Words : 1 (2 bytes)

Cycles: 1 if condition is false (no skip)

2 if condition is true (skip is executed) and the instruction skipped is 1 word

3 if condition is true (skip is executed) and the instruction skipped is 2 words

Cycles XMEGA: 2 if condition is false (no skip)

 ${\bf 3}$ if condition is true (skip is executed) and the instruction skipped is ${\bf 1}$ word

 $4\ \mbox{if condition}$ is true (skip is executed) and the instruction skipped is $2\ \mbox{words}$





SBIW - Subtract Immediate from Word

Description:

Subtracts an immediate value (0-63) from a register pair and places the result in the register pair. This instruction operates on the upper four register pairs, and is well suited for operations on the Pointer Registers.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) $Rd+1:Rd \leftarrow Rd+1:Rd - K$

16-bit Opcode:

1001	0111	KKdd	KKKK
		********	******

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С	
-	7-1		\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	⇔	

S: $N \oplus V$, For signed tests.

V: Rdh7 •R15

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R15

Set if MSB of the result is set; cleared otherwise.

- Z: R15• R14 •R13 •R12 •R11• R10• R9• R8• R7• R6 •R5• R4• R3 •R2• R1• R0 Set if the result is \$0000; cleared otherwise.
- C: R15• Rdh7

Set if the absolute value of K is larger than the absolute value of Rd; cleared otherwise.

R (Result) equals Rdh:Rdl after the operation (Rdh7-Rdh0 = R15-R8, Rdl7-Rdl0=R7-R0).

Example:

```
sbiw r25:r24,1 ; Subtract 1 from r25:r24
sbiw YH:YL,63 ; Subtract 63 from the Y-pointer(r29:r28)
```

Words: 1 (2 bytes)

Cycles: 2

126 AVR Instruction Set



SBR - Set Bits in Register

Description:

(i)

Sets specified bits in register Rd. Performs the logical ORI between the contents of register Rd and a constant mask K and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd \vee K$

O----

Syntax: Operands: SBR Rd,K $16 \le d \le 31, 0 \le K \le 255$

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

0110	KKKK	dddd	KKKK
	5-0000000000000000000000000000000000000	1.770.700.700.700	(2000)

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С
::	-	-	\Leftrightarrow	0	\Leftrightarrow	⇔	_

S: $N \oplus V$, For signed tests.

V: 0

Cleared

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7• R6 •R5• R4• R3 •R2• R1• R0
Set if the result is \$00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

sbr r16,3 ; Set bits 0 and 1 in r16 sbr r17,\$F0 ; Set 4 MSB in r17

Words: 1 (2 bytes)







SBRC - Skip if Bit in Register is Cleared

Description:

This instruction tests a single bit in a register and skips the next instruction if the bit is cleared.

Operation:

(i) If Rr(b) = 0 then $PC \leftarrow PC + 2$ (or 3) else $PC \leftarrow PC + 1$

Syntax: Operands: Program Counter:

(i) SBRC Rr,b $0 \le r \le 31, 0 \le b \le 7$ PC \leftarrow PC + 1, Condition false - no skip PC \leftarrow PC + 2, Skip a one word instruction PC \leftarrow PC + 3, Skip a two word instruction

16-bit Opcode:

1111	110r	rrrr	0bbb
------	------	------	------

Status Register (SREG) and Boolean Formula:

1	Т	н	s	V	N	Z	С	650
	-			0-0	-	-	-	

Example:

sub r0,r1 ; Subtract r1 from r0

sbrc r0,7 ; Skip if bit 7 in r0 cleared

sub r0,r1 ; Only executed if bit 7 in r0 not cleared

nop ; Continue (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false (no skip)

2 if condition is true (skip is executed) and the instruction skipped is 1 word

3 if condition is true (skip is executed) and the instruction skipped is 2 words

128 AVR Instruction Set



SBRS - Skip if Bit in Register is Set

Description:

(i)

This instruction tests a single bit in a register and skips the next instruction if the bit is set.

Operation:

(i) If Rr(b) = 1 then $PC \leftarrow PC + 2$ (or 3) else $PC \leftarrow PC + 1$

Syntax: Operands: Program Counter:

SBRS Rr,b $0 \le r \le 31, 0 \le b \le 7$ PC \leftarrow PC + 1, Condition false - no skip PC \leftarrow PC + 2, Skip a one word instruction PC \leftarrow PC + 3, Skip a two word instruction

16-bit Opcode:

1111	111r	rrrr	0bbb
------	------	------	------

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С
-	-	_	-	-	_	_	-

Example:

sub r0,r1 ; Subtract r1 from r0 sbrs r0,7 ; Skip if bit 7 in r0 set

neg r0 ; Only executed if bit 7 in r0 not set nop ; Continue (do nothing)

1.00 200 000 000

Words: 1 (2 bytes)

Cycles: 1 if condition is false (no skip)

2 if condition is true (skip is executed) and the instruction skipped is 1 word 3 if condition is true (skip is executed) and the instruction skipped is 2 words







SEC - Set Carry Flag

Description:

Sets the Carry Flag (C) in SREG (Status Register).

Operation:

(i) C ← 1

Syntax: Operands: (i) SEC None

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001 0100 0000 1000

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С
-	·	(-	_	_	_	-	1

C:

Carry Flag set

Example:

sec ; Set Carry Flag adc r0,r1 ; r0=r0+r1+1

Words: 1 (2 bytes)



SEH - Set Half Carry Flag

Description:

Sets the Half Carry (H) in SREG (Status Register).

Operation:

(i) H ← 1

Syntax: Operands: (i) SEH None

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001 0100 0101 1000

Status Register (SREG) and Boolean Formula:

I T H S V N Z C

H:

Half Carry Flag set

Example:

seh ; Set Half Carry Flag

Words: 1 (2 bytes)





SEI - Set Global Interrupt Flag

Description:

Sets the Global Interrupt Flag (I) in SREG (Status Register). The instruction following SEI will be executed before any pending interrupts.

Operation:

(i) I ← 1

16-bit Opcode:

	1001	0100	0111	1000
L.		500000000000000000000000000000000000000	100000000000000000000000000000000000000	

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С
1	_	_	_	_	_	_	-

.

Global Interrupt Flag set

Example:

sei ; set global interrupt enable sleep ; enter sleep, waiting for interrupt

; note: will enter sleep before any pending interrupt(s)

Words: 1 (2 bytes)

Cycles: 1

132 AVR Instruction Set



SEN - Set Negative Flag

Description:

Sets the Negative Flag (N) in SREG (Status Register).

Operation:

(i) $N \leftarrow 1$

Syntax: Operands: (i) SEN None

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001 0100 0010 1000

Status Register (SREG) and Boolean Formula:

I T H S V N Z C

N:

Negative Flag set

Example:

add r2,r19 ; Add r19 to r2 sen ; Set Negative Flag

Words: 1 (2 bytes)







SER - Set all Bits in Register

Description:

Loads \$FF directly to register Rd.

Operation:

(i) Rd ← \$FF

 Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

Г	1110	1111	dddd	1111
1			aaaa	

Status Register (SREG) and Boolean Formula:

1	Т	н	S	V	N	Z	С
_	_	(/ - 3	_	_	_	-	_

Example:

clr r16 ; Clear r16 ser r17 ; Set r17

out \$18,r16 ; Write zeros to Port B
nop ; Delay (do nothing)
out \$18,r17 ; Write ones to Port B

Words: 1 (2 bytes)



SES - Set Signed Flag

Description:

Sets the Signed Flag (S) in SREG (Status Register).

Operation:

(i) S ← 1

Syntax: Operands: (i) SES None

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001 0100 0100 1000

Status Register (SREG) and Boolean Formula:

1	T	Н	s	V	N	Z	С
-	=	-	1	-	-	_	_

S:

Signed Flag set

Example:

add r2,r19 ; Add r19 to r2 ses ; Set Negative Flag

Words: 1 (2 bytes)







SET - Set T Flag

Description:

Sets the T Flag in SREG (Status Register).

Operation:

(i) T ← 1

Syntax: Operands: (i) SET None

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001 0100 0110 1000

Status Register (SREG) and Boolean Formula:

I T H S V N Z C
- 1 - - - - - -

T:

T Flag set

Example:

set ; Set T Flag

Words: 1 (2 bytes)



SEV - Set Overflow Flag

Description:

Sets the Overflow Flag (V) in SREG (Status Register).

Operation:

(i) V ← 1

Syntax: Operands: (i) SEV None

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001 0100 0011 1000

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С	
_	·-	(/ -	_	1	_	_	_	

V:

Overflow Flag set

Example:

add r2,r19 ; Add r19 to r2 sev ; Set Overflow Flag

Words: 1 (2 bytes)







SEZ - Set Zero Flag

Description:

Sets the Zero Flag (Z) in SREG (Status Register).

Operation:

(i) $Z \leftarrow 1$

Syntax: Operands: (i) SEZ None

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

	1001	0100	0001	1000
1	1001	0100	0001	1000

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С	
-	9 — 9	0-3	_	_	_	1	_	

Z:

Zero Flag set

Example:

add r2,r19 ; Add r19 to r2 sez ; Set Zero Flag

Words: 1 (2 bytes)



SLEEP

Description:

This instruction sets the circuit in sleep mode defined by the MCU Control Register.

Operation:

Refer to the device documentation for detailed description of SLEEP usage.

Syntax:Operands:Program Counter:SLEEPNone $PC \leftarrow PC + 1$

; Put MCU in sleep mode

16-bit Opcode:

Г	1001	0101	1000	1000
	1001	0101	1000	1000

Status Register (SREG) and Boolean Formula:

1	Т	н	S	V	N	Z	С
_	-	(/ - 3	_	_	_	-	_

Example:

sleep

Words: 1 (2 bytes)







SPM – Store Program Memory

Description:

SPM can be used to erase a page in the Program memory, to write a page in the Program memory (that is already erased), and to set Boot Loader Lock bits. In some devices, the Program memory can be written one word at a time, in other devices an entire page can be programmed simultaneously after first filling a temporary page buffer. In all cases, the Program memory must be erased one page at a time. When erasing the Program memory, the RAMPZ and Z-register are used as page address. When writing the Program memory, the RAMPZ and Z-register are used as page or word address, and the R1:R0 register pair is used as data⁽¹⁾. When setting the Boot Loader Lock bits, the R1:R0 register pair is used as data. Refer to the device documentation for detailed description of SPM usage. This instruction can address the entire Program memory.

The SPM instruction is not available in all devices. Refer to the device specific instruction set summary.

Note: 1. R1 determines the instruction high byte, and R0 determines the instruction low byte.

O	p	e	ra	ti	0	n

- (i) $(RAMPZ:Z) \leftarrow \$ffff$
- (ii) (RAMPZ:Z) ← R1:R0
- (iii) $(RAMPZ:Z) \leftarrow R1:R0$
- (iv) $(RAMPZ:Z) \leftarrow TEMP$
- (v) BLBITS ← R1:R0

Syntax: Operands:

(i)-(v) SPM Z+

Comment:

Erase Program memory page
Write Program memory word
Write temporary page buffer
Write temporary page buffer to Program

Write temporary page buffer to Program memory

Set Boot Loader Lock bits

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001	0101	1110	1000
------	------	------	------

Status Register (SREG) and Boolean Formula:

1	T	н	s	V	N	z	С	
-	-	(-)	-	-	-	-	-	

Example:

; This example shows SPM write of one page for devices with page write

;- the routine writes one page of data from RAM to Flash

; the first data location in RAM is pointed to by the Y-pointer

; the first data location in Flash is pointed to by the Z-pointer

;- error handling is not included

;- the routine must be placed inside the boot space

; (at least the do_spm sub routine)

;- registers used: r0, r1, temp1, temp2, looplo, loophi, spmcrval

; (temp1, temp2, looplo, loophi, spmcrval must be defined by the user)

; storing and restoring of registers is not included in the routine ; register usage can be optimized at the expense of code size

.equPAGESIZEB = PAGESIZE*2; PAGESIZEB is page size in BYTES, not words

.org SMALLBOOTSTART

write_page:

140 AVR Instruction Set



```
;page erase
  ldispmcrval, (1<<PGERS) + (1<<SPMEN)
  calldo_spm
  ;transfer data from RAM to Flash page buffer
  ldilooplo, low(PAGESIZEB); init loop variable
  ldiloophi, high(PAGESIZEB); not required for PAGESIZEB<=256
wrloop:ldr0, Y+
 ldr1, Y+
 ldispmcrval, (1<<SPMEN)
 calldo_spm
 adiwZH:ZL, 2
  sbiwloophi:looplo, 2;use subi for PAGESIZEB<=256
 brnewrloop
  ; execute page write
 subiZL, low(PAGESIZEB); restore pointer
  sbciZH, high(PAGESIZEB); not required for PAGESIZEB<=256
  ldispmcrval, (1<<PGWRT) + (1<<SPMEN)
  calldo_spm
  ;read back and check, optional
 ldilooplo, low(PAGESIZEB);init loop variable
 ldiloophi, high(PAGESIZEB); not required for PAGESIZEB<=256
  subiYL, low(PAGESIZEB); restore pointer
  sbciYH, high(PAGESIZEB)
rdloop:1pmr0, Z+
 ldr1, Y+
  cpser0, r1
  sbiwloophi:looplo, 2;use subi for PAGESIZEB<=256
 brnerdloop
  ;return
  ret
do_spm:
  ;input: spmcrval determines SPM action
  ; disable interrupts if enabled, store status
 intemp2, SREG
 cli
  ; check for previous SPM complete
wait:intemp1, SPMCR
 sbrctemp1, SPMEN
 rjmpwait
 ;SPM timed sequence
 outSPMCR, spmcrval
 spm
 ;restore SREG (to enable interrupts if originally enabled)
 outSREG, temp2
```



AINEL 141





ret

Words: 1 (2 bytes)

Cycles: depends on the operation

142 AVR Instruction Set



SPM #2- Store Program Memory

Description:

SPM can be used to erase a page in the Program memory and to write a page in the Program memory (that is already erased). An entire page can be programmed simultaneously after first filling a temporary page buffer. The Program memory must be erased one page at a time. When erasing the Program memory, the RAMPZ and Z-register are used as page address. When writing the Program memory, the RAMPZ and Z-register are used as page or word address, and the R1:R0 register pair is used as data⁽¹⁾.

Refer to the device documentation for detailed description of SPM usage. This instruction can address the entire Program memory.

Note: 1. R1 determines the instruction high byte, and R0 determines the instruction low byte.

	Operation:		Comment:
(i)	$(RAMPZ:Z) \leftarrow \$ffff$		Erase Program memory page
(ii)	$(RAMPZ:Z) \leftarrow R1:R0$		Load Page Buffer
(iii)	$(RAMPZ:Z) \leftarrow BUFFER$		Write Page Buffer to Program memory
(iv)	$(RAMPZ:Z) \leftarrow \$fff$	$Z \leftarrow Z + 2$	Erase Program memory page, Z post incremented
(v)	$(RAMPZ:Z) \leftarrow R1:R0$	$Z \leftarrow Z + 2$	Load Page Buffer, Z post incremented
(vi)	(RAMPZ:Z) ←BUFFER	$Z \leftarrow Z + 2$	Write Page Buffer to Program memory,
			Z post incremented

	Syntax:	Operands:	Program Counter		
(i)-(iii)	SPM	None	PC ← PC + 1		
(iv)-(vi)	SPM Z+	None	PC ← PC + 1		

16-bit Opcode:

(i)-(iii)	1001	0101	1110	1000
(iv)-(vi)	1001	0101	1111	1000

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С
-	-	:	-	8—8	-	-	i —

Example:

TBD

Words: 1 (2 bytes)

Cycles: depends on the operation







ST – Store Indirect From Register to Data Space using Index X

Description:

Stores one byte indirect from a register to data space. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. The EEPROM has a separate address space.

The data location is pointed to by the X (16 bits) Pointer Register in the Register File. Memory access is limited to the current data segment of 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPX in register in the I/O area has to be changed.

The X-pointer Register can either be left unchanged by the operation, or it can be post-incremented or pre-decremented. These features are especially suited for accessing arrays, tables, and Stack Pointer usage of the X-pointer Register. Note that only the low byte of the X-pointer is updated in devices with no more than 256 bytes data space. For such devices, the high byte of the pointer is not used by this instruction and can be used for other purposes. The RAMPX Register in the I/O area is updated in parts with more than 64K bytes data space or more than 64K bytes Program memory, and the increment/ decrement is added to the entire 24-bit address on such devices.

Not all variants of this instruction is available in all devices. Refer to the device specific instruction set summary.

The result of these combinations is undefined:

ST X+, r26 ST X+, r27 ST -X, r26 ST -X, r27

Using the X-pointer:

Operation	1:
-----------	----

/V) , D.

(1)	$(X) \leftarrow Hr$	
(ii)	$(X) \leftarrow Rr$	$X \leftarrow X+1$
(iii)	X ← X - 1	$(X) \leftarrow Rr$

Syntax: Operands:

	Symax.	Operands.
(i)	ST X, Rr	$0 \le r \le 31$
(ii)	ST X+, Rr	$0 \le r \le 31$
(iii)	ST -X, Rr	$0 \le r \le 31$

Comment:

X: Unchanged
X: Post incremented
X: Pre decremented

Program Counter:

 $PC \leftarrow PC + 1$ $PC \leftarrow PC + 1$ $PC \leftarrow PC + 1$

16-bit Opcode:

(i)	1001	001r	rrrr	1100	
(ii)	1001	001r	rrrr	1101	
(iii)	1001	001r	rrrr	1110	

Status Register (SREG) and Boolean Formula:

1	Т	н	s	V	N	z	С
8	-	-	-	a-	-		-

144 AVR Instruction Set



Example:

```
clr
       r27
                  ; Clear X high byte
ldi
     r26,$60 ; Set X low byte to $60
      X+,r0
                  ; Store r0 in data space loc. $60(X post inc)
st
       X,r1
                  ; Store r1 in data space loc. $61
ldi
       r26,$63 ; Set X low byte to $63
       X,r2 ; Store r2 in data space loc. $63
-X,r3 ; Store r3 in 3-4
st
                   ; Store r3 in data space loc. $62(X pre dec)
st
```

Words: 1 (2 bytes)

 Cycles:
 2

 Cycles XMEGA:
 (i) 1

 (ii) 1
 (iii) 2

 Cycles Reduced Core tinyAVR:(i) 1

(ii) 1 (iii) 2







ST (STD) - Store Indirect From Register to Data Space using Index Y

Description:

Stores one byte indirect with or without displacement from a register to data space. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. The EEPROM has a separate address space.

The data location is pointed to by the Y (16 bits) Pointer Register in the Register File. Memory access is limited to the current data segment of 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPY in register in the I/O area has to be changed.

The Y-pointer Register can either be left unchanged by the operation, or it can be post-incremented or pre-decremented. These features are especially suited for accessing arrays, tables, and Stack Pointer usage of the Y-pointer Register. Note that only the low byte of the Y-pointer is updated in devices with no more than 256 bytes data space. For such devices, the high byte of the pointer is not used by this instruction and can be used for other purposes. The RAMPY Register in the I/O area is updated in parts with more than 64K bytes data space or more than 64K bytes Program memory, and the increment/ decrement/displacement is added to the entire 24-bit address on such devices.

Not all variants of this instruction is available in all devices. Refer to the device specific instruction set summary.

The result of these combinations is undefined:

ST Y+, r28 ST Y+, r29 ST -Y, r28 ST -Y, r29

Using the Y-pointer:

Operation:

(1)	$(Y) \leftarrow Hr$	
(ii)	$(Y) \leftarrow Rr$	$Y \leftarrow Y+1$
(iii)	$Y \leftarrow Y - 1$	$(Y) \leftarrow Rr$

(iv) $(Y+q) \leftarrow Rr$

Syntax: Operands: ST Y, Rr $0 \le r \le 31$ ST Y+, Rr $0 \le r \le 31$

(iv) STD Y+q, Rr $0 \le r \le 31$, $0 \le q \le 63$

16-bit Opcode:

	(i)	1000	001r	rrrr	1000
	(ii)	1001	001r	rrrr	1001
Г	(iii)	1001	001r	rrrr	1010
Г	(iv)	10q0	qq1r	rrrr	1qqq

Comment:

Y: Unchanged

Y: Post incremented

Y: Pre decremented

Y: Unchanged, q: Displacement

Program Counter:

 $PC \leftarrow PC + 1$ $PC \leftarrow PC + 1$

 $PC \leftarrow PC + 1$

PC ← PC + 1

AVR Instruction Set

146



Status Register (SREG) and Boolean Formula:

ı	T	Н	S	V	N	Z	С	
-	-	_	_	-	_	_	_]

Example:

r29 ; Clear Y high byte clr ldi r28,\$60 ; Set Y low byte to \$60 st Y+,r0; Store r0 in data space loc. \$60(Y post inc) Y, r1 ; Store r1 in data space loc. \$61 st ldi r28,\$63 ; Set Y low byte to \$63 st Y,r2 ; Store r2 in data space loc. \$63 -Y,r3 ; Store r3 in data space loc. \$62(Y pre dec) Y+2,r4 ; Store r4 in data space loc. \$64

Words: 1 (2 bytes)

Cycles: 2 Cycles XMEGA: (i) 1

(ii) 1 (iii) 2

(iv) 2

Cycles Reduced Core tinyAVR:(i) 1

(ii) 1

(iii) 2





ST (STD) – Store Indirect From Register to Data Space using Index Z

Description:

Stores one byte indirect with or without displacement from a register to data space. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. The EEPROM has a separate address space.

The data location is pointed to by the Z (16 bits) Pointer Register in the Register File. Memory access is limited to the current data segment of 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPZ in register in the I/O area has to be changed.

The Z-pointer Register can either be left unchanged by the operation, or it can be post-incremented or pre-decremented. These features are especially suited for Stack Pointer usage of the Z-pointer Register, however because the Z-pointer Register can be used for indirect subroutine calls, indirect jumps and table lookup, it is often more convenient to use the X or Y-pointer as a dedicated Stack Pointer. Note that only the low byte of the Z-pointer is updated in devices with no more than 256 bytes data space. For such devices, the high byte of the pointer is not used by this instruction and can be used for other purposes. The RAMPZ Register in the I/O area is updated in parts with more than 64K bytes data space or more than 64K bytes Program memory, and the increment/decrement/displacement is added to the entire 24-bit address on such devices.

Not all variants of this instruction is available in all devices. Refer to the device specific instruction set summary.

The result of these combinations is undefined:

ST Z+, r30

ST Z+, r31

ST -Z, r30

ST -Z, r31

Using the Z-pointer:

(i) (ii) (iii) (iv)	Operation: (Z) \leftarrow Rr (Z) \leftarrow Rr Z \leftarrow Z - 1 (Z+q) \leftarrow Rr	$Z \leftarrow Z+1$ $(Z) \leftarrow Rr$	Comment: Z: Unchanged Z: Post incremented Z: Pre decremented Z: Unchanged, q: Displacement
	Syntax:	Operands:	Program Counter:
(i)	ST Z, Rr	$0 \le r \le 31$	PC ← PC + 1
(ii)	ST Z+, Rr	$0 \le r \le 31$	PC ← PC + 1
(iii)	ST -Z, Rr	$0 \le r \le 31$	PC ← PC + 1
(iv)	STD Z+q, Rr	$0 \leq r \leq 31, \ 0 \leq q \leq 63$	PC ← PC + 1

AVR Instruction Set

148



16-bit Opcode:

	(i)	1000	001r	rrrr	0000
Г	(ii)	1001	001r	rrrr	0001
	(iii)	1001	001r	rrrr	0010
	(iv)	10q0	qq1r	rrrr	0qqq

Status Register (SREG) and Boolean Formula:

ı	Т	н	S	V	N	Z	С
-	-	=	-	-	-	-	-

Example:

clr r31 ; Clear Z high byte r30,\$60 ; Set Z low byte to \$60 ldi ; Store r0 in data space loc. \$60(Z post inc) Z+,r0 st st Z,r1 ; Store r1 in data space loc. \$61 ldi r30,\$63 ; Set Z low byte to \$63 ; Store r2 in data space loc. \$63 st Z,r2 ; Store r3 in data space loc. \$62(Z pre dec) st -Z,r3 Z+2,r4 ; Store r4 in data space loc. \$64

Words: 1 (2 bytes)

Cycles: Cycles XMEGA:

(i) 1

(ii) 1

2

(iii) 2

(iv) 2

(10) 2

Cycles Reduced Core tinyAVR:(i) 1

(ii) 1

(iii) 2





STS - Store Direct to Data Space

Description:

Stores one byte from a Register to the data space. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. The EEPROM has a separate address space.

A 16-bit address must be supplied. Memory access is limited to the current data segment of 64K bytes. The STS instruction uses the RAMPD Register to access memory above 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPD in register in the I/O area has to be changed.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) $(k) \leftarrow Rr$

32-bit Opcode:

1001	001d	dddd	0000
kkkk	kkkk	kkkk	kkkk

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С	
_	_	_	_	_	_	_	_	

Example:

lds r2,\$FF00 ; Load r2 with the contents of data space location \$FF00 add r2,r1 ; add r1 to r2 sts \$FF00,r2 ; Write back

Words: 2 (4 bytes)

Cycles: 2

AVR Instruction Set

150



STS (16-bit) - Store Direct to Data Space

Description:

Stores one byte from a Register to the data space. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. In some parts the Flash memory has been mapped to the data space and can be written using this command. The EEPROM has a separate address space.

A 7-bit address must be supplied. The address given in the instruction is coded to a data space address as follows:

ADDR[7:0] = (INST[8], INST[8], INST[10], INST[9], INST[3], INST[2], INST[1], INST[0])

Memory access is limited to the address range 0x40...0xbf of the data segment.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) $(k) \leftarrow Rr$

16-bit Opcode:

1.0	110	1 lelele	2222	lelelele
10	110	TKKK	aaaa	KKKK

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С	
-	-	-	-	-	-	_	_	

Example:

1ds r16,\$00 ; Load r16 with the contents of data space location \$00 add r16,r17 ; add r17 to r16 sts \$00,r16 ; Write result to the same address it was fetched from

Words: 1 (2 bytes)

Cycles: 1

Note: Registers r0..r15 are remaped to r16..r31







SUB - Subtract without Carry

Description:

Subtracts two registers and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd - Rr$

16-bit Opcode:

0001 10rd dddd rrrr

Status Register and Boolean Formula:

ı	T	Н	S	V	N	Z	С
-	-	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	⇔	\Leftrightarrow	⇔

H: Rd3• Rr3 +Rr3 •R3 +R3• Rd3
Set if there was a borrow from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: Rd7• Rr7 •R7 +Rd7 •Rr7• R7

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7 Set if MSB of the result is set; cleared otherwise.

Z: R7• R6 •R5• R4• R3 •R2• R1• R0
Set if the result is \$00; cleared otherwise.

C: Rd7• Rr7 +Rr7•R7 +R7• Rd7

Set if the absolute value of the contents of Rr is larger than the absolute value of Rd; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

sub r13,r12 ; Subtract r12 from r13
brne noteq ; Branch if r12<>r13
...
noteq: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1

152 AVR Instruction Set



SUBI - Subtract Immediate

Description:

Subtracts a register and a constant and places the result in the destination register Rd. This instruction is working on Register R16 to R31 and is very well suited for operations on the X, Y and Z-pointers.

Operation:

(i) $Rd \leftarrow Rd - K$

Syntax: Operands: Program Counter: i) SUBI Rd,K $16 \le d \le 31$, $0 \le K \le 255$ PC \leftarrow PC + 1

16-bit Opcode:

0101 KKKK dddd KKKK

Status Register and Boolean Formula:

I	Т	Н	s	V	N	Z	С
_	-	⇔	\Leftrightarrow	⇔	⇔	\Leftrightarrow	⇔

H: Rd3• K3+K3 •R3 +R3 •Rd3

Set if there was a borrow from bit 3; cleared otherwise

- S: $N \oplus V$, For signed tests.
- V: Rd7• K7 •R7 +Rd7• K7 •R7

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7

Set if MSB of the result is set; cleared otherwise.

- Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$ Set if the result is \$00; cleared otherwise.
- C: Rd7• K7 +K7 •R7 +R7• Rd7

 Set if the absolute value of K is larger than the absolute value of Rd; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

subi r22,\$11 ; Subtract \$11 from r22
brne noteq ; Branch if r22<>\$11
...
noteq: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1



153





SWAP - Swap Nibbles

Description:

Swaps high and low nibbles in a register.

Operation:

(i) $R(7:4) \leftarrow Rd(3:0), R(3:0) \leftarrow Rd(7:4)$

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

	1001	010d	dddd	0010
1		100100000000000000000000000000000000000	(20000000000	

Status Register and Boolean Formula:

1	Т	Н	s	V	N	Z	С
-	s=-x	0-1	_	—	-	-	_

R (Result) equals Rd after the operation.

Example:

inc r1 ; Increment r1
swap r1 ; Swap high and low nibble of r1
inc r1 ; Increment high nibble of r1
swap r1 ; Swap back

Words: 1 (2 bytes)



TST - Test for Zero or Minus

Description:

Tests if a register is zero or negative. Performs a logical AND between a register and itself. The register will remain unchanged.

Operation:

(i) $Rd \leftarrow Rd \bullet Rd$

16-bit Opcode: (see AND Rd, Rd)

	0010	DD00	dddd	dddd
--	------	------	------	------

Status Register and Boolean Formula:

I	Т	Н	S	V	N	Z	С
=	-	-	\Leftrightarrow	0	\Leftrightarrow	⇔	-

S: $N \oplus V$, For signed tests.

V: 0

Cleared

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$ Set if the result is \$00; cleared otherwise.

R (Result) equals Rd.

Example:

tst r0 ; Test r0
breq zero ; Branch if r0=0
...

zero: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)







WDR - Watchdog Reset

Description:

This instruction resets the Watchdog Timer. This instruction must be executed within a limited time given by the WD prescaler. See the Watchdog Timer hardware specification.

Operation:

(i) WD timer restart.

16-bit Opcode:

1001	0101	1010	1000
------	------	------	------

Status Register and Boolean Formula:

Ī	Т	Н	s	V	N	Z	С
-	-	-	-	-	i-	-	-

Example:

wdr ; Reset watchdog timer

Words: 1 (2 bytes)



XCH - Exchange

Description:

Operation:

(i) $(Z) \leftarrow Rd, Rd \leftarrow (Z)$

Syntax:Operands:XCH Z,Rd $0 \le d \le 31$

16-bit Opcode:

1001 001r rrrr 0100

Words: 1 (2 bytes)

Cycles: 1

(i)







Datasheet Revision History

Please note that the referring page numbers in this section are referred to this document. The referring revision in this section is referred to the document revision.

Rev.0856I - 07/10

- 1. Updated "Complete Instruction Set Summary" on page 11 with new instructions: LAC, LAS, LAT and XCH.
 - "LAC Load And Clear" on page 84
 - "LAS Load And Set" on page 85
 - "LAT Load And Toggle" on page 86
 - "XCH Exchange" on page 157
- 2. Updated number of clock cycles column to include Reduced Core tinyAVR.
 - (ATtiny replaced by Reduced Core tinyAVR).

Rev.0856H - 04/09

- 1. Updated "Complete Instruction Set Summary" on page 11:
 - Updated number of clock cycles column to include Reduced Core tinyAVR.
- 2. Updated sections for Reduced Core tinyAVR compatibility:
 - "CBI Clear Bit in I/O Register" on page 48
 - "LD Load Indirect from Data Space to Register using Index X" on page 87
 - "LD (LDD) Load Indirect from Data Space to Register using Index Y" on page 90
 - "LD (LDD) Load Indirect From Data Space to Register using Index Z" on page 92
 - "RCALL Relative Call to Subroutine" on page 114
 - "SBI Set Bit in I/O Register" on page 123
 - "ST Store Indirect From Register to Data Space using Index X" on page 144
 - "ST (STD) Store Indirect From Register to Data Space using Index Y" on page 146
 - "ST (STD) Store Indirect From Register to Data Space using Index Z" on page 148
- 3. Added sections for Reduced Core tinyAVR compatibility:
 - "LDS (16-bit) Load Direct from Data Space" on page 96
 - "STS (16-bit) Store Direct to Data Space" on page 151

Rev.0856G - 07/08

- 1. Inserted "Datasheet Revision History"
- 2. Updated "Cycles XMEGA" for ST, by removing (iv).
- 3. Updated "SPM #2" opcodes.

158 AVR Instruction Set



AVR Instruction Set

Rev.0856F - 05/08

1. This revision is based on the AVR Instruction Set 0856E-AVR-11/05

Changes done compared to AVR Instruction Set 0856E-AVR-11/05:

- Updated "Complete Instruction Set Summary" with DES and SPM #2.
- Updated AVR Instruction Set with XMEGA Clock cycles and Instruction Description.







Headquarters

Atmel Corporation

2325 Orchard Parkway San Jose, CA 95131 USA

Tel: 1(408) 441-0311 Fax: 1(408) 487-2600

International

Atmel Asia

Unit 1-5 & 16, 19/F BEA Tower, Millennium City 5 418 Kwun Tong Road Kwun Tong, Kowloon Hong Kong

Tel: (852) 2245-6100 Fax: (852) 2722-1369 Atmel Europe

Le Krebs 8, Rue Jean-Pierre Timbaud BP 309 78054 Saint-Quentin-en-Yvelines Cedex France

Tel: (33) 1-30-60-70-00 Fax: (33) 1-30-60-71-11 Atmel Japan

9F, Tonetsu Shinkawa Bldg. 1-24-8 Shinkawa Chuo-ku, Tokyo 104-0033 Japan

Tel: (81) 3-3523-3551 Fax: (81) 3-3523-7581

Product Contact

Web Site

www.atmel.com

Technical Support

avr@atmel.com

Sales Contact

www.atmel.com/contacts

Literature Requests

www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2010 Atmel Corporation. All rights reserved. Atmel ®, Atmel logo and combinations thereof, AVR®, AVR® logo, tinyAVR® and others are registered trademarks, XMEGATM and others are trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.

0856I-AVR-07/10



Features

- High-performance, Low-power Atmel® AVR® 8-bit Microcontroller
- Advanced RISC Architecture
 - 131 Powerful Instructions Most Single-clock Cycle Execution
 - 32 x 8 General Purpose Working Registers
 - Fully Static Operation
 - Up to 16 MIPS Throughput at 16 MHz
 - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
 - 16 Kbytes of In-System Self-programmable Flash program memory
 - 512 Bytes EEPROM
 - 1 Kbyte Internal SRAM
 - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
 - Data retention: 20 years at 85°C/100 years at 25°C(1)
 - Optional Boot Code Section with Independent Lock Bits In-System Programming by On-chip Boot Program True Read-While-Write Operation
 - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
 - Boundary-scan Capabilities According to the JTAG Standard
 - Extensive On-chip Debug Support
- Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
 - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
 - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
 - Real Time Counter with Separate Oscillator
 - Four PWM Channels
 - 8-channel, 10-bit ADC
 - 8 Single-ended Channels
 - 7 Differential Channels in TQFP Package Only
 - 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
 - Byte-oriented Two-wire Serial Interface
 - Programmable Serial USART
 - Master/Slave SPI Serial Interface
 - Programmable Watchdog Timer with Separate On-chip Oscillator
 - On-chip Analog Comparator
- Special Microcontroller Features
 - Power-on Reset and Programmable Brown-out Detection
 - Internal Calibrated RC Oscillator
 - External and Internal Interrupt Sources
 - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
 - 32 Programmable I/O Lines
 - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
 - 2.7V 5.5V for ATmega16L
 - 4.5V 5.5V for ATmega16
- Speed Grades
 - 0 8 MHz for ATmega16L
 - 0 16 MHz for ATmega16
- Power Consumption @ 1 MHz, 3V, and 25°C for ATmega16L
 - Active: 1.1 mA
 - Idle Mode: 0.35 mA
 - Power-down Mode: < 1 μA



8-bit AVR®
Microcontroller with 16K Bytes In-System
Programmable Flash

ATmega16 ATmega16L

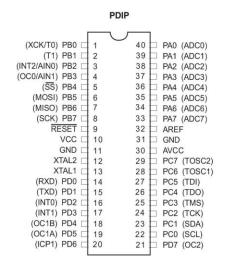
Rev. 2466T-AVR-07/10

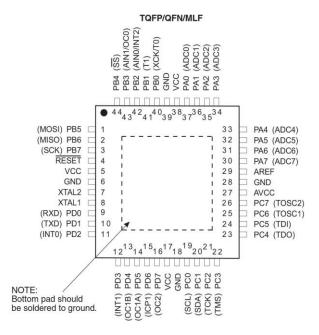




Pin Configurations

Figure 1. Pinout ATmega16





Disclaimer

Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.



2466T-AVR-07/10

2

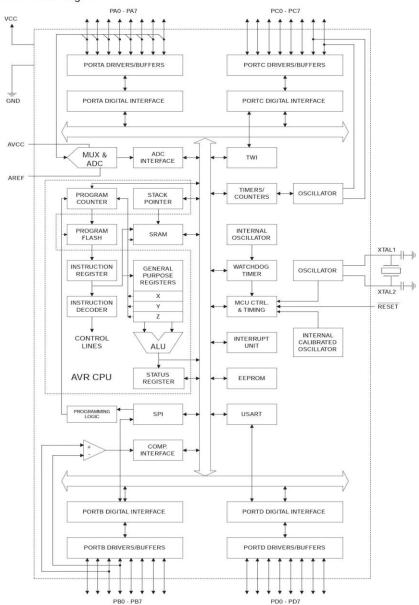


Overview

The ATmega16 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega16 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 2. Block Diagram





2466T-AVR-07/10



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega16 provides the following features: 16 Kbytes of In-System Programmable Flash Program memory with Read-While-Write capabilities, 512 bytes EEPROM, 1 Kbyte SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for Boundaryscan, On-chip Debugging support and programming, three flexible Timer/Counters with compare modes, Internal and External Interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain (TQFP package only), a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the USART, Two-wire interface, A/D Converter, SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next External Interrupt or Hardware Reset. In Power-save mode, the Asynchronous Timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run.

The device is manufactured using Atmel's high density nonvolatile memory technology. The Onchip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega16 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The ATmega16 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

Pin Descriptions

VCC Digital supply voltage.

GND Ground.

Port A (PA7..PA0) Port A serves as the analog inputs to the A/D Converter.

Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.





Port B (PB7..PB0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATmega16 as listed on page 58.

Port C (PC7..PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs.

Port C also serves the functions of the JTAG interface and other special features of the ATmega16 as listed on page 61.

Port D (PD7..PD0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATmega16 as listed on page 63

RESET

Reset Input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 15 on page 38. Shorter pulses are not guaranteed to generate a reset.

XTAL1

Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting Oscillator amplifier.

AVCC

AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to $V_{\rm CC}$, even if the ADC is not used. If the ADC is used, it should be connected to $V_{\rm CC}$ through a low-pass filter.

tillough a low-pa.

AREF

AREF is the analog reference pin for the A/D Converter.





Resources A comprehensive set of development tools, application notes and datasheets are available for

download on http://www.atmel.com/avr.

Data Retention Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85° C or 100 years at 25° C.





About Code Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. These code examples assume that the part specific header file is included before compilation. Be aware that not all C Compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Please confirm with the C Compiler documentation for more details.





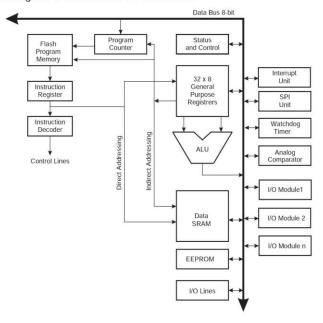
AVR CPU Core

Introduction

This section discusses the AVR core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

Architectural Overview

Figure 3. Block Diagram of the AVR MCU Architecture



In order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In-System Reprogrammable Flash memory.

The fast-access Register File contains 32×8 -bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in Flash Program memory. These added function registers are the 16-bit X-register, Y-register, and Z-register, described later in this section.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.





Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16-bit or 32-bit instruction.

Program Flash memory space is divided in two sections, the Boot program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM instruction that writes into the Application Flash memory section must reside in the Boot Program section.

During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the reset routine (before subroutines or interrupts are executed). The Stack Pointer SP is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional global interrupt enable bit in the Status Register. All interrupts have a separate interrupt vector in the interrupt vector table. The interrupts have priority in accordance with their interrupt vector position. The lower the interrupt vector address, the higher the priority.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, SPI, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the Register File, \$20 - \$5F.

ALU – Arithmetic Logic Unit

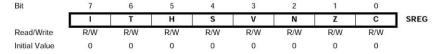
The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the "Instruction Set" section for a detailed description.

Status Register

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set Reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

The Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

The AVR Status Register - SREG - is defined as:



Bit 7 – I: Global Interrupt Enable

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.



9

2466T-AVR-07/10



· Bit 6 - T: Bit Copy Storage

The Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit. A bit from a register in the Register File can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

· Bit 5 - H: Half Carry Flag

The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half Carry is useful in BCD arithmetic. See the "Instruction Set Description" for detailed information.

Bit 4 – S: Sign Bit, S = N ⊕ V

The S-bit is always an exclusive or between the Negative Flag N and the Two's Complement Overflow Flag V. See the "Instruction Set Description" for detailed information.

Bit 3 – V: Two's Complement Overflow Flag

The Two's Complement Overflow Flag V supports two's complement arithmetics. See the "Instruction Set Description" for detailed information.

Bit 2 – N: Negative Flag

The Negative Flag N indicates a negative result in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.

Bit 1 – Z: Zero Flag

The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.

• Bit 0 - C: Carry Flag

The Carry Flag C indicates a carry in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.





General Purpose Register File

The Register File is optimized for the AVR Enhanced RISC instruction set. In order to achieve the required performance and flexibility, the following input/output schemes are supported by the Register File:

- One 8-bit output operand and one 8-bit result input
- Two 8-bit output operands and one 8-bit result input
- Two 8-bit output operands and one 16-bit result input
- One 16-bit output operand and one 16-bit result input

Figure 4 shows the structure of the 32 general purpose working registers in the CPU.

Figure 4. AVR CPU General Purpose Working Registers

Addr. R0 \$00 R1 \$01 R2 \$02 \$0D R13 R14 \$0E R15 \$0F R16 \$10 R17 \$11 R26 \$1A X-register Low Byte R27 X-register High Byte \$1B R28 \$1C Y-register Low Byte R29 \$1D Y-register High Byte R30 \$1E Z-register Low Byte R31 Z-register High Byte \$1F

General Working Registers

Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 4, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y-, and Z-pointer Registers can be set to index any register in the file.



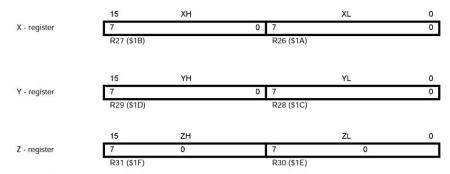
11



The X-register, Y-register and Z-register

The registers R26..R31 have some added functions to their general purpose usage. These registers are 16-bit address pointers for indirect addressing of the Data Space. The three indirect address registers X, Y, and Z are defined as described in Figure 5.

Figure 5. The X-register, Y-register, and Z-register



In the different addressing modes these address registers have functions as fixed displacement, automatic increment, and automatic decrement (see the Instruction Set Reference for details).

Stack Pointer

The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The Stack Pointer Register always points to the top of the Stack. Note that the Stack is implemented as growing from higher memory locations to lower memory locations. This implies that a Stack PUSH command decreases the Stack Pointer. If software reads the Program Counter from the Stack after a call or an interrupt, unused bits (15:13) should be masked out.

The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above \$60. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call or interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

The AVR Stack Pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation dependent. Note that the data space in some implementations of the AVR architecture is so small that only SPL is needed. In this case, the SPH Register will not be present.

Bit	15	14	13	12	11	10	9	8	
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	





Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The AVR CPU is driven by the CPU clock clk_{CPU} , directly generated from the selected clock source for the chip. No internal clock division is used.

Figure 6 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access Register File concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Figure 6. The Parallel Instruction Fetches and Instruction Executions

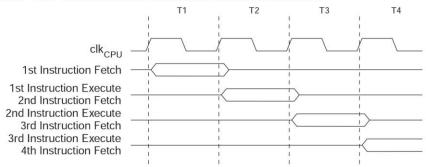
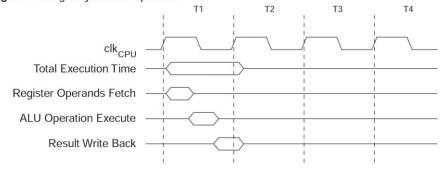


Figure 7 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 7. Single Cycle ALU Operation



Reset and Interrupt Handling

The AVR provides several different interrupt sources. These interrupts and the separate reset vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt. Depending on the Program Counter value, interrupts may be automatically disabled when Boot Lock bits BLB02 or BLB12 are programmed. This feature improves software security. See the section "Memory Programming" on page 259 for details.

The lowest addresses in the program memory space are by default defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in "Interrupts" on page 45. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INTO – the External Interrupt Request



13



0. The Interrupt Vectors can be moved to the start of the Boot Flash section by setting the IVSEL bit in the General Interrupt Control Register (GICR). Refer to "Interrupts" on page 45 for more information. The Reset Vector can also be moved to the start of the boot Flash section by programming the BOOTRST Fuse, see "Boot Loader Support – Read-While-Write Self-Programming" on page 246.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared and all interrupts are disabled. The user software can write logic one to the I-bit to enable nested interrupts. All enabled interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a Return from Interrupt instruction – RETI – is executed.

There are basically two types of interrupts. The first type is triggered by an event that sets the Interrupt Flag. For these interrupts, the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, and hardware clears the corresponding Interrupt Flag. Interrupt Flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared. If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the Interrupt Flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software. Similarly, if one or more interrupt conditions occur while the Global Interrupt Enable bit is cleared, the corresponding Interrupt Flag(s) will be set and remembered until the global interrupt enable bit is set, and will then be executed by order of priority.

The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have Interrupt Flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered.

When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

Note that the Status Register is not automatically stored when entering an interrupt routine, nor restored when returning from an interrupt routine. This must be handled by software.

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence.

```
in r16, SREG  ; store SREG value
  cli  ; disable interrupts during timed sequence
  sbi EECR, EEMWE  ; start EEPROM write
  sbi EECR, EEWE
  out SREG, r16  ; restore SREG value (I-bit)

C Code Example

  char cSREG;
  cSREG = SREG; /* store SREG value */
  /* disable interrupts during timed sequence */
   _CLI();
  EECR |= (1<<EEMWE); /* start EEPROM write */
  EECR |= (1<<EEWE);
  SREG = cSREG; /* restore SREG value (I-bit) */</pre>
```





When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in this example.

```
Assembly Code Example

sei ; set global interrupt enable
sleep; enter sleep, waiting for interrupt
; note: will enter sleep before any pending
; interrupt(s)

C Code Example

_SEI(); /* set global interrupt enable */
_SLEEP(); /* enter sleep, waiting for interrupt */
/* note: will enter sleep before any pending interrupt(s) */
```

Interrupt Response Time

The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum. After four clock cycles the program vector address for the actual interrupt handling routine is executed. During this four clock cycle period, the Program Counter is pushed onto the Stack. The vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by four clock cycles. This increase comes in addition to the start-up time from the selected sleep mode.

A return from an interrupt handling routine takes four clock cycles. During these four clock cycles, the Program Counter (two bytes) is popped back from the Stack, the Stack Pointer is incremented by two, and the I-bit in SREG is set.



2466T-AVR-07/10



AVR ATmega16 Memories

This section describes the different memories in the ATmega16. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, the ATmega16 features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

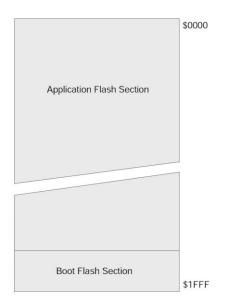
In-System Reprogrammable Flash Program Memory The ATmega16 contains 16 Kbytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 8K \times 16. For software security, the Flash Program memory space is divided into two sections, Boot Program section and Application Program section.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega16 Program Counter (PC) is 13 bits wide, thus addressing the 8K program memory locations. The operation of Boot Program section and associated Boot Lock bits for software protection are described in detail in "Boot Loader Support – Read-While-Write Self-Programming" on page 246. "Memory Programming" on page 259 contains a detailed description on Flash data serial downloading using the SPI pins or the JTAG interface.

Constant tables can be allocated within the entire program memory address space (see the LPM – Load Program Memory Instruction Description).

Timing diagrams for instruction fetch and execution are presented in "Instruction Execution Timing" on page 13.

Figure 8. Program Memory Map







SRAM Data Memory

Figure 9 shows how the ATmega16 SRAM Memory is organized.

The lower 1120 Data Memory locations address the Register File, the I/O Memory, and the internal data SRAM. The first 96 locations address the Register File and I/O Memory, and the next 1024 locations address the internal data SRAM.

The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement, and Indirect with Post-increment. In the Register File, registers R26 to R31 feature the indirect addressing pointer registers.

The direct addressing reaches the entire data space.

The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y-register or Z-register.

When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented.

The 32 general purpose working registers, 64 I/O Registers, and the 1024 bytes of internal data SRAM in the ATmega16 are all accessible through all these addressing modes. The Register File is described in "General Purpose Register File" on page 11.

Figure 9. Data Memory Map

Register File	Data Address Space
R0	\$0000
R1	\$0001
R2	\$0002
7	***
R29	\$001D
R30	\$001E
R31	\$001F
I/O Registers	
\$00	\$0020
\$01	\$0021
\$02	\$0022
300	****
\$3D	\$005D
\$3E	\$005E
\$3F	\$005F
	Internal SRAM
	\$0060
	\$0061

	\$045E
	\$045F

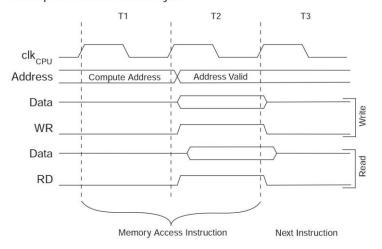




Data Memory Access Times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two clk_{CPU} cycles as described in Figure 10.

Figure 10. On-chip Data SRAM Access Cycles



EEPROM Data Memory

The ATmega16 contains 512 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

For a detailed description of SPI, JTAG, and Parallel data downloading to the EEPROM, see page 273, page 278, and page 262, respectively.

EEPROM Read/Write Access

The EEPROM Access Registers are accessible in the I/O space.

The write access time for the EEPROM is given in Table 1. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, $V_{\rm CC}$ is likely to rise or fall slowly on Power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. See "Preventing EEPROM Corruption" on page 22 for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this.

When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.





——— ATmega16(L)

The EEPROM Address Register – EEARH and EEARL

Bit	15	14	13	12	11	10	9	8	
	-	-	-	-	-	-	-	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	•
Read/Write	R	R	R	R	R	R	R	R/W	
	R/W								
Initial Value	0	0	0	0	0	0	0	X	
	X	X	X	X	X	X	X	X	

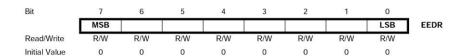
· Bits 15..9 - Res: Reserved Bits

These bits are reserved bits in the ATmega16 and will always read as zero.

• Bits 8..0 - EEAR8..0: EEPROM Address

The EEPROM Address Registers - EEARH and EEARL - specify the EEPROM address in the 512 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 511. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

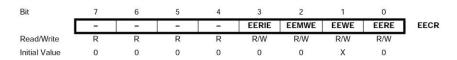
The EEPROM Data Register – EEDR



· Bits 7..0 - EEDR7.0: EEPROM Data

For the EEPROM write operation, the EEDR Register contains the data to be written to the EEPROM in the address given by the EEAR Register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

The EEPROM Control Register – EECR



Bits 7..4 – Res: Reserved Bits

These bits are reserved bits in the ATmega16 and will always read as zero.

· Bit 3 - EERIE: EEPROM Ready Interrupt Enable

Writing EERIE to one enables the EEPROM Ready Interrupt if the I bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready interrupt generates a constant interrupt when EEWE is cleared.

· Bit 2 - EEMWE: EEPROM Master Write Enable

The EEMWE bit determines whether setting EEWE to one causes the EEPROM to be written. When EEMWE is set, setting EEWE within four clock cycles will write data to the EEPROM at the selected address If EEMWE is zero, setting EEWE will have no effect. When EEMWE has been written to one by software, hardware clears the bit to zero after four clock cycles. See the description of the EEWE bit for an EEPROM write procedure.



19

2466T-AVR-07/10



Bit 1 – EEWE: EEPROM Write Enable

The EEPROM Write Enable Signal EEWE is the write strobe to the EEPROM. When address and data are correctly set up, the EEWE bit must be written to one to write the value into the EEPROM. The EEMWE bit must be written to one before a logical one is written to EEWE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 3 and 4 is not essential):

- Wait until EEWE becomes zero.
- 2. Wait until SPMEN in SPMCR becomes zero.
- 3. Write new EEPROM address to EEAR (optional).
- 4. Write new EEPROM data to EEDR (optional).
- 5. Write a logical one to the EEMWE bit while writing a zero to EEWE in EECR.
- 6. Within four clock cycles after setting EEMWE, write a logical one to EEWE.

The EEPROM can not be programmed during a CPU write to the Flash memory. The software must check that the Flash programming is completed before initiating a new EEPROM write. Step 2 is only relevant if the software contains a Boot Loader allowing the CPU to program the Flash. If the Flash is never being updated by the CPU, step 2 can be omitted. See "Boot Loader Support – Read-While-Write Self-Programming" on page 246 for details about boot programming.

Caution: An interrupt between step 5 and step 6 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM Access, the EEAR or EEDR reGister will be modified, causing the interrupted EEPROM Access to fail. It is recommended to have the Global Interrupt Flag cleared during all the steps to avoid these problems.

When the write access time has elapsed, the EEWE bit is cleared by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEWE has been set, the CPU is halted for two cycles before the next instruction is executed.

• Bit 0 - EERE: EEPROM Read Enable

The EEPROM Read Enable Signal – EERE – is the read strobe to the EEPROM. When the correct address is set up in the EEAR Register, the EERE bit must be written to a logic one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed.

The user should poll the EEWE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR Register.

The calibrated Oscillator is used to time the EEPROM accesses. Table 1 lists the typical programming time for EEPROM access from the CPU.

Table 1. EEPROM Programming Time

Symbol	Number of Calibrated RC Oscillator Cycles ⁽¹⁾	Typ Programming Time
EEPROM write (from CPU)	8448	8.5 ms

Note: 1. Uses 1 MHz clock, independent of CKSEL Fuse setting.

The following code examples show one assembly and one C function for writing to the EEPROM. The examples assume that interrupts are controlled (for example by disabling interrupts globally) so that no interrupts will occur during execution of these functions. The examples



20



also assume that no Flash Boot Loader is present in the software. If such code is present, the EEPROM write function must also wait for any ongoing SPM command to finish.

```
Assembly Code Example
   EEPROM write:
     ; Wait for completion of previous write
     sbic EECR, EEWE
     rjmp EEPROM_write
     ; Set up address (r18:r17) in address register
     out EEARH, r18
     out EEARL, r17
     ; Write data (r16) to data register
     out EEDR, r16
     ; Write logical one to EEMWE
     sbi EECR, EEMWE
     ; Start eeprom write by setting EEWE
     sbi EECR, EEWE
     ret
C Code Example
   void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
     /* Wait for completion of previous write */
     while(EECR & (1<<EEWE))
      ;
     /\star Set up address and data registers \star/
     EEAR = uiAddress;
     EEDR = ucData;
     /* Write logical one to EEMWE */
     EECR |= (1<<EEMWE);
     /* Start eeprom write by setting EEWE */
     EECR |= (1<<EEWE);</pre>
```





The next code examples show assembly and C functions for reading the EEPROM. The examples assume that interrupts are controlled so that no interrupts will occur during execution of these functions.

```
Assembly Code Example
   EEPROM_read:
     ; Wait for completion of previous write
     sbic EECR. EEWE
     rjmp EEPROM_read
     ; Set up address (r18:r17) in address register
     out EEARH, r18
     out EEARL, r17
     ; Start eeprom read by writing EERE
     sbi EECR, EERE
     ; Read data from data register
     in r16.EEDR
     ret
C Code Example
   unsigned char EEPROM_read(unsigned int uiAddress)
     /* Wait for completion of previous write */
     while(EECR & (1<<EEWE))
     /* Set up address register */
     EEAR = uiAddress;
     /* Start eeprom read by writing EERE */
     EECR |= (1<<EERE);
     /* Return data from data register */
     return EEDR;
```

EEPROM Write During Power-down Sleep Mode When entering Power-down Sleep mode while an EEPROM write operation is active, the EEPROM write operation will continue, and will complete before the Write Access time has passed. However, when the write operation is completed, the Oscillator continues running, and as a consequence, the device does not enter Power-down entirely. It is therefore recommended to verify that the EEPROM write operation is completed before entering Power-down.

Preventing EEPROM Corruption

During periods of low V_{CC} , the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board level systems using EEPROM, and the same design solutions should be applied.

An EEPROM data corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage is too low.





EEPROM data corruption can easily be avoided by following this design recommendation:

Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD). If the detection level of the internal BOD does not match the needed detection level, an external low $V_{\rm CC}$ Reset Protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.

I/O Memory

The I/O space definition of the ATmega16 is shown in "Register Summary" on page 331.

All ATmega16 I/Os and peripherals are placed in the I/O space. The I/O locations are accessed by the IN and OUT instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range \$00 - \$1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the Instruction Set section for more details. When using the I/O specific commands IN and OUT, the I/O addresses \$00 - \$3F must be used. When addressing I/O Registers as data space using LD and ST instructions, \$20 must be added to these addresses.

For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

Some of the Status Flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O Register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers \$00 to \$1F only.

The I/O and Peripherals Control Registers are explained in later sections.

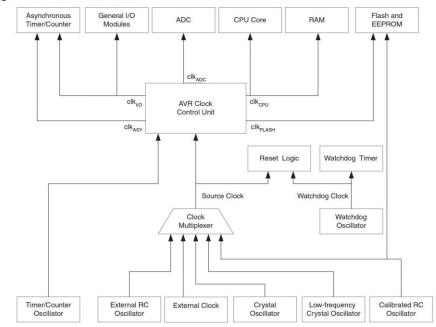




System Clock and Clock Options

Clock Systems and their Distribution Figure 11 presents the principal clock systems in the AVR and their distribution. All of the clocks need not be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes, as described in "Power Management and Sleep Modes" on page 32. The clock systems are detailed Figure 11.

Figure 11. Clock Distribution



CPU Clock - clk_{CPU}

The CPU clock is routed to parts of the system concerned with operation of the AVR core. Examples of such modules are the General Purpose Register File, the Status Register and the data memory holding the Stack Pointer. Halting the CPU clock inhibits the core from performing general operations and calculations.

I/O Clock - clk_{I/O}

The I/O clock is used by the majority of the I/O modules, like Timer/Counters, SPI, and USART. The I/O clock is also used by the External Interrupt module, but note that some external interrupts are detected by asynchronous logic, allowing such interrupts to be detected even if the I/O clock is halted. Also note that address recognition in the TWI module is carried out asynchronously when clk_{I/O} is halted, enabling TWI address reception in all sleep modes.

Flash Clock - clk_{FLASH}

The Flash clock controls operation of the Flash interface. The Flash clock is usually active simultaneously with the CPU clock.

Asynchronous Timer Clock – clk_{ASY} The Asynchronous Timer clock allows the Asynchronous Timer/Counter to be clocked directly from an external 32 kHz clock crystal. The dedicated clock domain allows using this Timer/Counter as a real-time counter even when the device is in sleep mode.



24

2466T-AVR-07/10



ADC Clock - clk_{ADC}

The ADC is provided with a dedicated clock domain. This allows halting the CPU and I/O clocks in order to reduce noise generated by digital circuitry. This gives more accurate ADC conversion results.

Clock Sources

The device has the following clock source options, selectable by Flash Fuse bits as shown below. The clock from the selected source is input to the AVR clock generator, and routed to the appropriate modules.

Table 2. Device Clocking Options Select(1)

Device Clocking Option	CKSEL30
External Crystal/Ceramic Resonator	1111 - 1010
External Low-frequency Crystal	1001
External RC Oscillator	1000 - 0101
Calibrated Internal RC Oscillator	0100 - 0001
External Clock	0000

Note: 1. For all fuses "1" means unprogrammed while "0" means programmed.

The various choices for each clocking option is given in the following sections. When the CPU wakes up from Power-down or Power-save, the selected clock source is used to time the start-up, ensuring stable Oscillator operation before instruction execution starts. When the CPU starts from Reset, there is as an additional delay allowing the power to reach a stable level before commencing normal operation. The Watchdog Oscillator is used for timing this real-time part of the start-up time. The number of WDT Oscillator cycles used for each time-out is shown in Table 3. The frequency of the Watchdog Oscillator is voltage dependent as shown in "ATmega16 Typical Characteristics" on page 299.

Table 3. Number of Watchdog Oscillator Cycles

Typ Time-out (V _{CC} = 5.0V)	Typ Time-out (V _{CC} = 3.0V)	Number of Cycles
4.1 ms	4.3 ms	4K (4,096)
65 ms	69 ms	64K (65,536)

Default Clock Source

The device is shipped with CKSEL = "0001" and SUT = "10". The default clock source setting is therefore the 1 MHz Internal RC Oscillator with longest startup time. This default setting ensures that all users can make their desired clock source setting using an In-System or Parallel Programmer.

Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, as shown in Figure 12. Either a quartz crystal or a ceramic resonator may be used. The CKOPT Fuse selects between two different Oscillator amplifier modes. When CKOPT is programmed, the Oscillator output will oscillate will a full rail-to-rail swing on the output. This mode is suitable when operating in a very noisy environment or when the output from XTAL2 drives a second clock buffer. This mode has a wide frequency range. When CKOPT is unprogrammed, the Oscillator has a smaller output swing. This reduces power consumption considerably. This mode has a limited frequency range and it can not be used to drive other clock buffers.

For resonators, the maximum frequency is 8 MHz with CKOPT unprogrammed and 16 MHz with CKOPT programmed. C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator in use, the amount of stray capacitance, and the electromagnetic noise of the environment. Some initial guidelines for

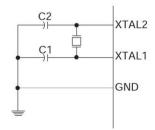


25



choosing capacitors for use with crystals are given in Table 4. For ceramic resonators, the capacitor values given by the manufacturer should be used.

Figure 12. Crystal Oscillator Connections



The Oscillator can operate in three different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL3..1 as shown in Table 4.

Table 4. Crystal Oscillator Operating Modes

СКОРТ	CKSEL31	Frequency Range (MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
1	101 ⁽¹⁾	0.4 - 0.9	-
1	110	0.9 - 3.0	12 - 22
1	111	3.0 - 8.0	12 - 22
0	101, 110, 111	1.0 ≤	12 - 22

Note: 1. This option should not be used with crystals, only with ceramic resonators.



The CKSEL0 Fuse together with the SUT1..0 Fuses select the start-up times as shown in Table 5.

Table 5. Start-up Times for the Crystal Oscillator Clock Selection

CKSEL0	SUT10	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V _{CC} = 5.0V)	Recommended Usage
0	00	258 CK ⁽¹⁾	4.1 ms	Ceramic resonator, fast rising power
0	01	258 CK ⁽¹⁾	65 ms	Ceramic resonator, slowly rising power
0	10	1K CK ⁽²⁾	_	Ceramic resonator, BOD enabled
0	11	1K CK ⁽²⁾	4.1 ms	Ceramic resonator, fast rising power
1	00	1K CK ⁽²⁾	65 ms	Ceramic resonator, slowly rising power
1	01	16K CK	-	Crystal Oscillator, BOD enabled
1	10	16K CK	4.1 ms	Crystal Oscillator, fast rising power
1	11	16K CK	65 ms	Crystal Oscillator, slowly rising power

Notes: 1. These options should only be used when not operating close to the maximum frequency of the device, and only if frequency stability at start-up is not important for the application. These options are not suitable for crystals.



These options are intended for use with ceramic resonators and will ensure frequency stability at start-up. They can also be used with crystals when not operating close to the maximum frequency of the device, and if frequency stability at start-up is not important for the application.



Low-frequency Crystal Oscillator

To use a 32.768 kHz watch crystal as the clock source for the device, the Low-frequency Crystal Oscillator must be selected by setting the CKSEL Fuses to "1001". The crystal should be connected as shown in Figure 12. By programming the CKOPT Fuse, the user can enable internal capacitors on XTAL1 and XTAL2, thereby removing the need for external capacitors. The internal capacitors have a nominal value of 36 pF.

When this Oscillator is selected, start-up times are determined by the SUT Fuses as shown in Table 6.

Table 6. Start-up Times for the Low-frequency Crystal Oscillator Clock Selection

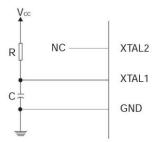
SUT10	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V _{cc} = 5.0V)	Recommended Usage	
00	1K CK ⁽¹⁾	4.1 ms	Fast rising power or BOD enabled	
01	1K CK ⁽¹⁾	65 ms	Slowly rising power	
10	32K CK	65 ms	Stable frequency at start-up	
11	Reserved			

Note: 1. These options should only be used if frequency stability at start-up is not important for the application.

External RC Oscillator

For timing insensitive applications, the external RC configuration shown in Figure 13 can be used. The frequency is roughly estimated by the equation f = 1/(3RC). C should be at least 22 pF. By programming the CKOPT Fuse, the user can enable an internal 36 pF capacitor between XTAL1 and GND, thereby removing the need for an external capacitor. For more information on Oscillator operation and details on how to choose R and C, refer to the External RC Oscillator application note.

Figure 13. External RC Configuration



The Oscillator can operate in four different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL3..0 as shown in Table 7.

Table 7. External RC Oscillator Operating Modes

CKSEL30	Frequency Range (MHz)
0101	0.1 ≤ 0.9
0110	0.9 - 3.0
0111	3.0 - 8.0
1000	8.0 - 12.0



28



When this Oscillator is selected, start-up times are determined by the SUT Fuses as shown in Table 8.

Table 8. Start-up Times for the External RC Oscillator Clock Selection

SUT10	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V _{CC} = 5.0V)	Recommended Usage
00	18 CK	_	BOD enabled
01	18 CK	4.1 ms	Fast rising power
10	18 CK	65 ms	Slowly rising power
11	6 CK ⁽¹⁾	4.1 ms	Fast rising power or BOD enabled

Note: 1. This option should not be used when operating close to the maximum frequency of the device.

Calibrated Internal RC Oscillator

The Calibrated Internal RC Oscillator provides a fixed 1.0 MHz, 2.0 MHz, 4.0 MHz, or 8.0 MHz clock. All frequencies are nominal values at 5V and 25°C. This clock may be selected as the system-clock by programming the CKSEL Fuses as shown in Table 9. If selected, it will operate with no external components. The CKOPT Fuse should always be unpro-grammed when using this clock option. During Reset, hardware loads the calibration byte into the OSCCAL Register and thereby automatically calibrates the RC Oscillator. At 5V, 25°C and 1.0 MHz, 2.0 MHz, 4.0 MHz or 8.0 MHz Oscillator frequency selected, this calibration gives a frequency within $\pm 3\%$ of the nominal frequency. Using calibration methods as described in application notes available at www.atmel.com/avr it is possible to achieve $\pm 1\%$ accuracy at any given V_{CC} and Temperature. When this Oscillator is used as the Chip Clock, the Watchdog Oscillator will still be used for the Watchdog Timer and for the reset time-out. For more information on the pre-programmed calibration value, see the section "Calibration Byte" on page 261.

Table 9. Internal Calibrated RC Oscillator Operating Modes

CKSEL30	Nominal Frequency (MHz)
0001 ⁽¹⁾	1.0
0010	2.0
0011	4.0
0100	8.0

Note: 1. The device is shipped with this option selected.

When this Oscillator is selected, start-up times are determined by the SUT Fuses as shown in Table 10. XTAL1 and XTAL2 should be left unconnected (NC).

Table 10. Start-up Times for the Internal Calibrated RC Oscillator Clock Selection

SUT10	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V _{CC} = 5.0V)	Recommended Usage
00	6 CK	_	BOD enabled
01	6 CK	4.1 ms	Fast rising power
10 ⁽¹⁾	6 CK	65 ms	Slowly rising power
11		Reserved	

Note: 1. The device is shipped with this option selected.



29



Oscillator Calibration Register – OSCCAL

Bit	7	6	5	4	3	2	1	0	
	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	OSCCAL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	•
Initial Value			Devi	ce Specific	Calibration \	/alue			

Bits 7..0 - CAL7..0: Oscillator Calibration Value

Writing the calibration byte to this address will trim the Internal Oscillator to remove process variations from the Oscillator frequency. This is done automatically during Chip Reset. When OSCCAL is zero, the lowest available frequency is chosen. Writing non-zero values to this register will increase the frequency of the Internal Oscillator. Writing \$FF to the register gives the highest available frequency. The calibrated Oscillator is used to time EEPROM and Flash access. If EEPROM or Flash is written, do not calibrate to more than 10% above the nominal frequency. Otherwise, the EEPROM or Flash write may fail. Note that the Oscillator is intended for calibration to 1.0 MHz, 2.0 MHz, 4.0 MHz, or 8.0 MHz. Tuning to other values is not guaranteed, as indicated in Table 11.

Table 11. Internal RC Oscillator Frequency Range.

OSCCAL Value	Min Frequency in Percentage of Nominal Frequency (%)	Max Frequency in Percentage of Nominal Frequency (%)			
\$00	50	100			
\$7F	75	150			
\$FF	100	200			

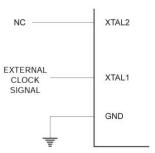




External Clock

To drive the device from an external clock source, XTAL1 should be driven as shown in Figure 14. To run the device on an external clock, the CKSEL Fuses must be programmed to "0000". By programming the CKOPT Fuse, the user can enable an internal 36 pF capacitor between XTAL1 and GND.

Figure 14. External Clock Drive Configuration



When this clock source is selected, start-up times are determined by the SUT Fuses as shown in Table 12.

Table 12. Start-up Times for the External Clock Selection

SUT10	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V _{CC} = 5.0V)	Recommended Usage
00	6 CK	-	BOD enabled
01	6 CK	4.1 ms	Fast rising power
10	6 CK	65 ms	Slowly rising power
11		Reserved	

When applying an external clock, it is required to avoid sudden changes in the applied clock frequency to ensure stable operation of the MCU. A variation in frequency of more than 2% from one clock cycle to the next can lead to unpredictable behavior. It is required to ensure that the MCU is kept in reset during such changes in the clock frequency.

Timer/Counter Oscillator

For AVR microcontrollers with Timer/Counter Oscillator pins (TOSC1 and TOSC2), the crystal is connected directly between the pins. No external capacitors are needed. The Oscillator is optimized for use with a 32.768 kHz watch crystal. Applying an external clock source to TOSC1 is not recommended.

Iote: The Timer/Counter Oscillator uses the same type of crystal oscillator as Low-Frequency Oscillator and the internal capacitors have the same nominal value of 36 pF.





Power Management and Sleep Modes

Sleep modes enable the application to shut down unused modules in the MCU, thereby saving power. The AVR provides various sleep modes allowing the user to tailor the power consumption to the application's requirements.

To enter any of the six sleep modes, the SE bit in MCUCR must be written to logic one and a SLEEP instruction must be executed. The SM2, SM1, and SM0 bits in the MCUCR Register select which sleep mode (Idle, ADC Noise Reduction, Power-down, Power-save, Standby, or Extended Standby) will be activated by the SLEEP instruction. See Table 13 for a summary. If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU wakes up. The MCU is then halted for four cycles in addition to the start-up time, it executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the Register File and SRAM are unaltered when the device wakes up from sleep. If a Reset occurs during sleep mode, the MCU wakes up and executes from the Reset Vector.

Figure 11 on page 24 presents the different clock systems in the ATmega16, and their distribution. The figure is helpful in selecting an appropriate sleep mode.

MCU Control Register – MCUCR

The MCU Control Register contains control bits for power management.

Bit	7	6	5	4	3	2	1	0	
	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	•
Initial Value	0	0	0	0	0	0	0	0	

Bits 7, 5, 4 – SM2..0: Sleep Mode Select Bits 2, 1, and 0

These bits select between the six available sleep modes as shown in Table 13.

Table 13. Sleep Mode Select

SM2	SM1	SM0	Sleep Mode	
0	0	0	Idle	
0	0	1	ADC Noise Reduction	
0	1	0	Power-down	
0	1	1	Power-save	
1	0	0	Reserved	
1	0	1	Reserved	
1	1	0	Standby ⁽¹⁾	
1	1	1	Extended Standby ⁽¹⁾	

 Standby mode and Extended Standby mode are only available with external crystals or resonators.

· Bit 6 - SE: Sleep Enable

The SE bit must be written to logic one to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmers purpose, it is recommended to write the Sleep Enable (SE) bit to one just before the execution of the SLEEP instruction and to clear it immediately after waking up.





Idle Mode

When the SM2..0 bits are written to 000, the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing SPI, USART, Analog Comparator, ADC, Two-wire Serial Interface, Timer/Counters, Watchdog, and the interrupt system to continue operating. This sleep mode basically halts clk_{CPU} and clk_{FLASH}, while allowing the other clocks to run.

Idle mode enables the MCU to wake up from external triggered interrupts as well as internal ones like the Timer Overflow and USART Transmit Complete interrupts. If wake-up from the Analog Comparator interrupt is not required, the Analog Comparator can be powered down by setting the ACD bit in the Analog Comparator Control and Status Register – ACSR. This will reduce power consumption in Idle mode. If the ADC is enabled, a conversion starts automatically when this mode is entered.

ADC Noise Reduction Mode

When the SM2..0 bits are written to 001, the SLEEP instruction makes the MCU enter ADC Noise Reduction mode, stopping the CPU but allowing the ADC, the External Interrupts, the Two-wire Serial Interface address watch, Timer/Counter2 and the Watchdog to continue operating (if enabled). This sleep mode basically halts $clk_{I/O}$, clk_{CPU} , and clk_{FLASH} , while allowing the other clocks to run.

This improves the noise environment for the ADC, enabling higher resolution measurements. If the ADC is enabled, a conversion starts automatically when this mode is entered. Apart form the ADC Conversion Complete interrupt, only an External Reset, a Watchdog Reset, a Brown-out Reset, a Two-wire Serial Interface Address Match Interrupt, a Timer/Counter2 interrupt, an SPM/EEPROM ready interrupt, an External level interrupt on INT0 or INT1, or an external interrupt on INT2 can wake up the MCU from ADC Noise Reduction mode.

Power-down Mode

When the SM2..0 bits are written to 010, the SLEEP instruction makes the MCU enter Power-down mode. In this mode, the External Oscillator is stopped, while the External interrupts, the Two-wire Serial Interface address watch, and the Watchdog continue operating (if enabled). Only an External Reset, a Watchdog Reset, a Brown-out Reset, a Two-wire Serial Interface address match interrupt, an External level interrupt on INT0 or INT1, or an External interrupt on INT2 can wake up the MCU. This sleep mode basically halts all generated clocks, allowing operation of asynchronous modules only.

Note that if a level triggered interrupt is used for wake-up from Power-down mode, the changed level must be held for some time to wake up the MCU. Refer to "External Interrupts" on page 68 for details.

When waking up from Power-down mode, there is a delay from the wake-up condition occurs until the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is defined by the same CKSEL Fuses that define the reset time-out period, as described in "Clock Sources" on page 25.

Power-save Mode

When the SM2..0 bits are written to 011, the SLEEP instruction makes the MCU enter Powersave mode. This mode is identical to Power-down, with one exception:

If Timer/Counter2 is clocked asynchronously, that is, the AS2 bit in ASSR is set, Timer/Counter2 will run during sleep. The device can wake up from either Timer Overflow or Output Compare event from Timer/Counter2 if the corresponding Timer/Counter2 interrupt enable bits are set in TIMSK, and the Global Interrupt Enable bit in SREG is set.

If the Asynchronous Timer is NOT clocked asynchronously, Power-down mode is recommended instead of Power-save mode because the contents of the registers in the Asynchronous Timer should be considered undefined after wake-up in Power-save mode if AS2 is 0.

This sleep mode basically halts all clocks except clk_{ASY}, allowing operation only of asynchronous modules, including Timer/Counter2 if clocked asynchronously.





Standby Mode

When the SM2..0 bits are 110 and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter Standby mode. This mode is identical to Power-down with the exception that the Oscillator is kept running. From Standby mode, the device wakes up in six clock cycles.

Extended Standby Mode

When the SM2..0 bits are 111 and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter Extended Standby mode. This mode is identical to Power-save mode with the exception that the Oscillator is kept running. From Extended Standby mode, the device wakes up in six clock cycles..

Table 14. Active Clock Domains and Wake Up Sources in the Different Sleep Modes

		Active C	lock d	lomains	6	Oscilla	Wake-up Sources						
Sleep Mode	clk _{CPU}	clk _{FLASH}	clk _{io}	clk _{ADC}	clk _{ASY}	Main Clock Source Enabled	Timer Osc. Enabled	INT2 INT1 INT0	TWI Address Match	Timer 2	SPM / EEPROM Ready	ADC	Other I/O
Idle			Х	Х	Х	Х	X ⁽²⁾	Х	Х	Х	Х	Х	Х
ADC Noise Redu- ction				х	х	х	X ⁽²⁾	X ⁽³⁾	х	х	X	х	
Power Down								X ⁽³⁾	Х				
Power Save					X ⁽²⁾		X ⁽²⁾	X ⁽³⁾	Х	X ⁽²⁾			
Standby ⁽¹⁾						Х		X ⁽³⁾	Х				
Exten- ded Standby ⁽¹⁾					X ⁽²⁾	х	X ⁽²⁾	X ⁽³⁾	Х	X ⁽²⁾			

Notes: 1. External Crystal or resonator selected as clock source.

- 2. If AS2 bit in ASSR is set.
- 3. Only INT2 or level interrupt INT1 and INT0.





Minimizing Power Consumption

There are several issues to consider when trying to minimize the power consumption in an AVR controlled system. In general, sleep modes should be used as much as possible, and the sleep mode should be selected so that as few as possible of the device's functions are operating. All functions not needed should be disabled. In particular, the following modules may need special consideration when trying to achieve the lowest possible power consumption.

Analog to Digital Converter

If enabled, the ADC will be enabled in all sleep modes. To save power, the ADC should be disabled before entering any sleep mode. When the ADC is turned off and on again, the next conversion will be an extended conversion. Refer to "Analog to Digital Converter" on page 204 for details on ADC operation.

Analog Comparator

When entering Idle mode, the Analog Comparator should be disabled if not used. When entering ADC Noise Reduction mode, the Analog Comparator should be disabled. In the other sleep modes, the Analog Comparator is automatically disabled. However, if the Analog Comparator is set up to use the Internal Voltage Reference as input, the Analog Comparator should be disabled in all sleep modes. Otherwise, the Internal Voltage Reference will be enabled, independent of sleep mode. Refer to "Analog Comparator" on page 201 for details on how to configure the Analog Comparator.

Brown-out Detector

If the Brown-out Detector is not needed in the application, this module should be turned off. If the Brown-out Detector is enabled by the BODEN Fuse, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to "Brown-out Detection" on page 40 for details on how to configure the Brown-out Detector.

Internal Voltage Reference

The Internal Voltage Reference will be enabled when needed by the Brown-out Detector, the Analog Comparator or the ADC. If these modules are disabled as described in the sections above, the internal voltage reference will be disabled and it will not be consuming power. When turned on again, the user must allow the reference to start up before the output is used. If the reference is kept on in sleep mode, the output can be used immediately. Refer to "Internal Voltage Reference" on page 42 for details on the start-up time.

Watchdog Timer

If the Watchdog Timer is not needed in the application, this module should be turned off. If the Watchdog Timer is enabled, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to "Watchdog Timer" on page 42 for details on how to configure the Watchdog Timer.

Port Pins

When entering a sleep mode, all port pins should be configured to use minimum power. The most important thing is then to ensure that no pins drive resistive loads. In sleep modes where the both the I/O clock (clk_{I/O}) and the ADC clock (clk_{ADC}) are stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed for detecting wake-up conditions, and it will then be enabled. Refer to the section "Digital Input Enable and Sleep Modes" on page 54 for details on which pins are enabled. If the input buffer is enabled and the input signal is left floating or have an analog signal level close to $V_{\rm CC}/2$, the input buffer will use excessive power.





JTAG Interface and On-chip Debug System If the On-chip debug system is enabled by the OCDEN Fuse and the chip enter Power down or Power save sleep mode, the main clock source remains enabled. In these sleep modes, this will contribute significantly to the total current consumption. There are three alternative ways to avoid this:

- Disable OCDEN Fuse.
- Disable JTAGEN Fuse.
- Write one to the JTD bit in MCUCSR.

The TDO pin is left floating when the JTAG interface is enabled while the JTAG TAP controller is not shifting data. If the hardware connected to the TDO pin does not pull up the logic level, power consumption will increase. Note that the TDI pin for the next device in the scan chain contains a pull-up that avoids this problem. Writing the JTD bit in the MCUCSR register to one or leaving the JTAG fuse unprogrammed disables the JTAG interface.





System Control and Reset

Resetting the AVR

During Reset, all I/O Registers are set to their initial values, and the program starts execution from the Reset Vector. The instruction placed at the Reset Vector must be a JMP – absolute jump – instruction to the reset handling routine. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa. The circuit diagram in Figure 15 shows the reset logic. Table 15 defines the electrical parameters of the reset circuitry.

The I/O ports of the AVR are immediately reset to their initial state when a reset source goes active. This does not require any clock source to be running.

After all reset sources have gone inactive, a delay counter is invoked, stretching the Internal Reset. This allows the power to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the CKSEL Fuses. The different selections for the delay period are presented in "Clock Sources" on page 25.

Reset Sources

The ATmega16 has five sources of reset:

- Power-on Reset.
 - The MCU is reset when the supply voltage is below the Power-on Reset threshold (V_{POT}).
- External Reset.
 - The MCU is reset when a low level is present on the RESET pin for longer than the minimum pulse length.
- Watchdog Reset.
 - The MCU is reset when the Watchdog Timer period expires and the Watchdog is enabled.
- Brown-out Reset.
 - The MCU is reset when the supply voltage $V_{\rm CC}$ is below the Brown-out Reset threshold $(V_{\rm BOT})$ and the Brown-out Detector is enabled.
- JTAG AVR Reset.

The MCU is reset as long as there is a logic one in the Reset Register, one of the scan chains of the JTAG system. Refer to the section "IEEE 1149.1 (JTAG) Boundary-scan" on page 228 for details.





Figure 15. Reset Logic

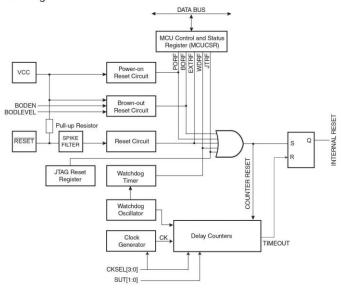


Table 15. Reset Characteristics

Symbol	Parameter	Condition	Min	Тур	Max	Units	
	Power-on Reset Threshold Voltage (rising)			1.4	2.3		
V _{POT}	Power-on Reset Threshold Voltage (falling) ⁽¹⁾			1.3	2.3	V	
V _{RST}	RESET Pin Threshold Voltage		0.1V _{CC}		0.9V _{CC}		
t _{RST}	Minimum pulse width on RESET Pin				1.5	μs	
	Brown-out Reset	BODLEVEL = 1	2.5	2.7	3.2	.,	
V_{BOT}	Threshold Voltage ⁽²⁾	BODLEVEL = 0	3.6	4.0	4.5	V	
	Minimum low voltage	BODLEVEL = 1		2		μs	
t _{BOD}	period for Brown-out Detection	BODLEVEL = 0		2			
V_{HYST}	Brown-out Detector hysteresis			50		mV	

Notes: 1. The Power-on Reset will not work unless the supply voltage has been below V_{POT} (falling).

^{2.} V_{BOT} may be below nominal minimum operating voltage for some devices. For devices where this is the case, the device is tested down to V_{CC} = V_{BOT} during the production test. This guarantees that a Brown-out Reset will occur before V_{CC} drops to a voltage where correct operation of the microcontroller is no longer guaranteed. The test is performed using BODLEVEL = 1 for ATmega16L and BODLEVEL = 0 for ATmega16. BODLEVEL = 1 is not applicable for ATmega16.





Power-on Reset

A Power-on Reset (POR) pulse is generated by an On-chip detection circuit. The detection level is defined in Table 15. The POR is activated whenever V_{CC} is below the detection level. The POR circuit can be used to trigger the Start-up Reset, as well as to detect a failure in supply voltage.

A Power-on Reset (POR) circuit ensures that the device is reset from Power-on. Reaching the Power-on Reset threshold voltage invokes the delay counter, which determines how long the device is kept in RESET after V_{CC} rise. The RESET signal is activated again, without any delay, when V_{CC} decreases below the detection level.

Figure 16. MCU Start-up, RESET Tied to V_{CC}.

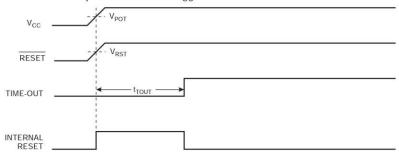
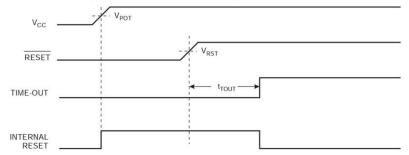


Figure 17. MCU Start-up, RESET Extended Externally





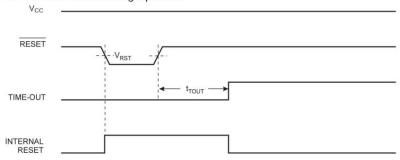
2466T-AVR-07/10



External Reset

An External Reset is generated by a low level on the $\overline{\text{RESET}}$ pin. Reset pulses longer than the minimum pulse width (see Table 15) will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset. When the applied signal reaches the Reset Threshold Voltage – V_{RST} – on its positive edge, the delay counter starts the MCU after the Time-out period t_{TOUT} has expired.

Figure 18. External Reset During Operation



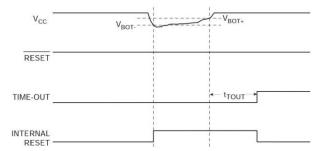
Brown-out Detection

ATmega16 has an On-chip Brown-out Detection (BOD) circuit for monitoring the V_{CC} level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the fuse BODLEVEL to be 2.7V (BODLEVEL unprogrammed), or 4.0V (BODLEVEL programmed). The trigger level has a hysteresis to ensure spike free Brown-out Detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST}/2$ and $V_{BOT-} = V_{BOT} - V_{HYST}/2$.

The BOD circuit can be enabled/disabled by the fuse BODEN. When the BOD is enabled (BODEN programmed), and V_{CC} decreases to a value below the trigger level (V_{BOT} in Figure 19), the Brown-out Reset is immediately activated. When V_{CC} increases above the trigger level (V_{BOT} in Figure 19), the delay counter starts the MCU after the Time-out period t_{TOUT} has expired.

The BOD circuit will only detect a drop in V_{CC} if the voltage stays below the trigger level for longer than t_{BOD} given in Table 15.

Figure 19. Brown-out Reset During Operation



AIMEL

No olvide citar esta tesis

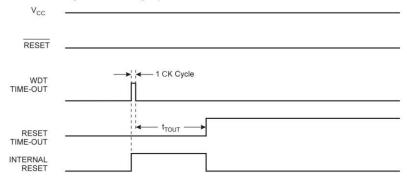
2466T-AVR-07/10



Watchdog Reset

When the Watchdog times out, it will generate a short reset pulse of one CK cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT} . Refer to page 42 for details on operation of the Watchdog Timer.

Figure 20. Watchdog Reset During Operation



MCU Control and Status Register – MCUCSR

The MCU Control and Status Register provides information on which reset source caused an MCU Reset.

Bit	7	6	5	4	3	2	1	0	
	JTD	ISC2	-	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	•
Initial Value	0	0	0						

Bit 4 – JTRF: JTAG Reset Flag

This bit is set if a reset is being caused by a logic one in the JTAG Reset Register selected by the JTAG instruction AVR_RESET. This bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

· Bit 3 - WDRF: Watchdog Reset Flag

This bit is set if a Watchdog Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

Bit 2 – BORF: Brown-out Reset Flag

This bit is set if a Brown-out Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

· Bit 1 - EXTRF: External Reset Flag

This bit is set if an External Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

Bit 0 – PORF: Power-on Reset Flag

This bit is set if a Power-on Reset occurs. The bit is reset only by writing a logic zero to the flag.

To make use of the Reset Flags to identify a reset condition, the user should read and then reset the MCUCSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the Reset Flags.





Internal Voltage Reference

ATmega16 features an internal bandgap reference. This reference is used for Brown-out Detection, and it can be used as an input to the Analog Comparator or the ADC. The 2.56V reference to the ADC is generated from the internal bandgap reference.

Voltage Reference Enable Signals and Start-up Time The voltage reference has a start-up time that may influence the way it should be used. The start-up time is given in Table 16. To save power, the reference is not always turned on. The reference is on during the following situations:

- 1. When the BOD is enabled (by programming the BODEN Fuse).
- When the bandgap reference is connected to the Analog Comparator (by setting the ACBG bit in ACSR).
- 3. When the ADC is enabled.

Thus, when the BOD is not enabled, after setting the ACBG bit or enabling the ADC, the user must always allow the reference to start up before the output from the Analog Comparator or ADC is used. To reduce power consumption in Power-down mode, the user can avoid the three conditions above to ensure that the reference is turned off before entering Power-down mode.

Table 16. Internal Voltage Reference Characteristics

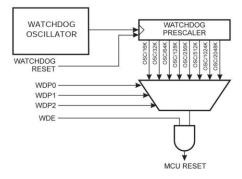
Symbol	Parameter	Min	Тур	Max	Units
V_{BG}	Bandgap reference voltage	1.15	1.23	1.4	V
t_{BG}	Bandgap reference start-up time		40	70	μs
I_{BG}	Bandgap reference current consumption		10		μΑ

Watchdog Timer

The Watchdog Timer is clocked from a separate On-chip Oscillator which runs at 1 MHz. This is the typical value at $V_{\rm CC}$ = 5V. See characterization data for typical values at other $V_{\rm CC}$ levels. By controlling the Watchdog Timer prescaler, the Watchdog Reset interval can be adjusted as shown in Table 17 on page 43. The WDR – Watchdog Reset – instruction resets the Watchdog Timer. The Watchdog Timer is also reset when it is disabled and when a Chip Reset occurs. Eight different clock cycle periods can be selected to determine the reset period. If the reset period expires without another Watchdog Reset, the ATmega16 resets and executes from the Reset Vector. For timing details on the Watchdog Reset, refer to page 41.

To prevent unintentional disabling of the Watchdog, a special turn-off sequence must be followed when the Watchdog is disabled. Refer to the description of the Watchdog Timer Control Register for details.

Figure 21. Watchdog Timer







——— ATmega16(L)

Watchdog Timer Control Register – WDTCR

Bit	7	6	5	4	3	2	1	0	
	-	-	-	WDTOE	WDE	WDP2	WDP1	WDP0	WDTCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

· Bits 7..5 - Res: Reserved Bits

These bits are reserved bits in the ATmega16 and will always read as zero.

· Bit 4 - WDTOE: Watchdog Turn-off Enable

This bit must be set when the WDE bit is written to logic zero. Otherwise, the Watchdog will not be disabled. Once written to one, hardware will clear this bit after four clock cycles. Refer to the description of the WDE bit for a Watchdog disable procedure.

· Bit 3 - WDE: Watchdog Enable

When the WDE is written to logic one, the Watchdog Timer is enabled, and if the WDE is written to logic zero, the Watchdog Timer function is disabled. WDE can only be cleared if the WDTOE bit has logic level one. To disable an enabled Watchdog Timer, the following procedure must be followed:

- 1. In the same operation, write a logic one to WDTOE and WDE. A logic one must be written to WDE even though it is set to one before the disable operation starts.
- 2. Within the next four clock cycles, write a logic 0 to WDE. This disables the Watchdog.

• Bits 2..0 - WDP2, WDP1, WDP0: Watchdog Timer Prescaler 2, 1, and 0

The WDP2, WDP1, and WDP0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding Timeout Periods are shown in Table 17.

Table 17. Watchdog Timer Prescale Select

WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V _{CC} = 3.0V	Typical Time-out at V _{CC} = 5.0V
0	0	0	16K (16,384)	17.1 ms	16.3 ms
0	0	1	32K (32,768)	34.3 ms	32.5 ms
0	1	0	64K (65,536)	68.5 ms	65 ms
0	1	1	128K (131,072)	0.14 s	0.13 s
1	0	0	256K (262,144)	0.27 s	0.26 s
1	0	1	512K (524,288)	0.55 s	0.52 s
1	1	0	1,024K (1,048,576)	1.1 s	1.0 s
1	1	1	2,048K (2,097,152)	2.2 s	2.1 s





The following code example shows one assembly and one C function for turning off the WDT. The example assumes that interrupts are controlled (for example by disabling interrupts globally) so that no interrupts will occur during execution of these functions.

```
Assembly Code Example
   WDT_off:
     ; Reset WDT
     ; Write logical one to WDTOE and WDE
     in r16, WDTCR
     ori r16, (1<<WDTOE) | (1<<WDE)
     out WDTCR, r16
     ; Turn off WDT
     ldi r16, (0<<WDE)
     out WDTCR, r16
C Code Example
   void WDT_off(void)
     /* Reset WDT*/
     _WDR();
     /* Write logical one to WDTOE and WDE */
     WDTCR |= (1<<WDTOE) | (1<<WDE);
     /* Turn off WDT */
     WDTCR = 0x00;
```





Interrupts

This section describes the specifics of the interrupt handling as performed in ATmega16. For a general explanation of the AVR interrupt handling, refer to "Reset and Interrupt Handling" on page 13.

Interrupt Vectors in ATmega16

Table 18. Reset and Interrupt Vectors

Vector No.	Program Address ⁽²⁾	Source	Interrupt Definition
1	\$000 ⁽¹⁾	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	\$002	INTO	External Interrupt Request 0
3	\$004	INT1	External Interrupt Request 1
4	\$006	TIMER2 COMP	Timer/Counter2 Compare Match
5	\$008	TIMER2 OVF	Timer/Counter2 Overflow
6	\$00A	TIMER1 CAPT	Timer/Counter1 Capture Event
7	\$00C	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	\$00E	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	\$010	TIMER1 OVF	Timer/Counter1 Overflow
10	\$012	TIMER0 OVF	Timer/Counter0 Overflow
11	\$014	SPI, STC	Serial Transfer Complete
12	\$016	USART, RXC	USART, Rx Complete
13	\$018	USART, UDRE	USART Data Register Empty
14	\$01A	USART, TXC	USART, Tx Complete
15	\$01C	ADC	ADC Conversion Complete
16	\$01E	EE_RDY	EEPROM Ready
17	\$020	ANA_COMP	Analog Comparator
18	\$022	TWI	Two-wire Serial Interface
19	\$024	INT2	External Interrupt Request 2
20	\$026	TIMERO COMP	Timer/Counter0 Compare Match
21	\$028	SPM_RDY	Store Program Memory Ready

Notes: 1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see "Boot Loader Support – Read-While-Write Self-Programming" on page 246.

Table 19 shows Reset and Interrupt Vectors placement for the various combinations of BOOTRST and IVSEL settings. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa.



When the IVSEL bit in GICR is set, interrupt vectors will be moved to the start of the Boot Flash section. The address of each Interrupt Vector will then be the address in this table added to the start address of the Boot Flash section.



Table 19. Reset and Interrupt Vectors Placement⁽¹⁾

BOOTRST	IVSEL	Reset address	Interrupt Vectors Start Address
1	0	\$0000	\$0002
1	1	\$0000	Boot Reset Address + \$0002
0	0	Boot Reset Address	\$0002
0	1	Boot Reset Address	Boot Reset Address + \$0002

Note: 1. The Boot Reset Address is shown in Table 100 on page 257. For the BOOTRST Fuse "1" means unprogrammed while "0" means programmed.

The most typical and general program setup for the Reset and Interrupt Vector Addresses in ATmega16 is:

 moga io io	•				
Address	Labels	Code		Co	omments
\$000		jmp	RESET	;	Reset Handler
\$002		jmp	EXT_INT0	;	IRQ0 Handler
\$004		jmp	EXT_INT1	;	IRQ1 Handler
\$006		jmp	TIM2_COMP	;	Timer2 Compare Handler
\$008		jmp	TIM2_OVF	;	Timer2 Overflow Handler
\$00A		jmp	TIM1_CAPT	;	Timerl Capture Handler
\$00C		jmp	TIM1_COMPA	;	Timer1 CompareA Handler
\$00E		jmp	TIM1_COMPB	;	Timer1 CompareB Handler
\$010		jmp	TIM1_OVF	;	Timer1 Overflow Handler
\$012		jmp	TIMO_OVF	;	Timer0 Overflow Handler
\$014		jmp	SPI_STC	;	SPI Transfer Complete Handler
\$016		jmp	USART_RXC	;	USART RX Complete Handler
\$018		jmp	USART_UDRE	;	UDR Empty Handler
\$01A		jmp	USART_TXC	;	USART TX Complete Handler
\$01C		jmp	ADC	;	ADC Conversion Complete Handler
\$01E		jmp	EE_RDY	;	EEPROM Ready Handler
\$020		jmp	ANA_COMP	;	Analog Comparator Handler
\$022		jmp	TWSI	;	Two-wire Serial Interface Handler
\$024		jmp	EXT_INT2	;	IRQ2 Handler
\$026		jmp	TIMO_COMP	;	Timer0 Compare Handler
\$028		jmp	SPM_RDY	;	Store Program Memory Ready Handler
;					
\$02A	RESET:	ldi	r16, high (RAMEND)	;	Main program start
\$02B		out	SPH,r16	;	Set Stack Pointer to top of RAM
\$02C		ldi	r16,low(RAMEND)		
\$02D		out	SPL,r16		
\$02E		sei		;	Enable interrupts
\$02F		<inst< td=""><td>r> xxx</td><td></td><td></td></inst<>	r> xxx		





When the BOOTRST Fuse is unprogrammed, the Boot section size set to 2 Kbytes and the IVSEL bit in the GICR Register is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
Address Labels
                Code
                                      Comments
        RESET: ldi r16,high(RAMEND); Main program start
$000
$001
                out SPH,r16
                                      ; Set Stack Pointer to top of RAM
$002
               ldi r16, low(RAMEND)
$003
                out SPL, r16
$004
                 sei
                                      ; Enable interrupts
$005
                 <instr> xxx
;
.org $1C02
                 jmp EXT_INTO
                                      ; IRQ0 Handler
$1C02
                 jmp EXT_INT1
                                      ; IRQ1 Handler
$1C04
. . .
$1C28
                 jmp SPM_RDY
                                      ; Store Program Memory Ready Handler
```

When the BOOTRST Fuse is programmed and the Boot section size set to 2 Kbytes, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
Address Labels
                       Code
                                              Comments
.org $002
$002
                                       ; IRQ0 Handler
                 jmp EXT_INT0
$004
                 jmp EXT_INT1
                                       ; IRQ1 Handler
. . .
                 . .
$028
                      SPM_RDY
                                       ; Store Program Memory Ready Handler
                 jmp
.org $1C00
                 ldi r16, high (RAMEND) ; Main program start
$1C00
$1C01
                 out SPH, r16
                                ; Set Stack Pointer to top of RAM
                 ldi r16, low(RAMEND)
$1C02
                 out SPL, r16
$1C03
$1C04
                 sei
                                       ; Enable interrupts
$1C05
                 <instr> xxx
```

When the BOOTRST Fuse is programmed, the Boot section size set to 2 Kbytes and the IVSEL bit in the GICR Register is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
Address Labels Code
.org $1C00
$1C00
                 jmp RESET
                                   ; Reset handler
                                   ; IRQ0 Handler
                 jmp EXT_INTO
$1C02
                                   ; IRQ1 Handler
$1C04
                 jmp EXT_INT1
. . .
                                    ;
$1C28
                 jmp
                       SPM_RDY
                                   ; Store Program Memory Ready Handler
$1C2A
         RESET:
                 ldi r16, high (RAMEND) ; Main program start
                 out SPH, r16
                                  ; Set Stack Pointer to top of RAM
$1C2B
$1C2C
                 ldi r16, low(RAMEND)
$1C2D
                 out SPL, r16
$1C2E
                 sei
                                       : Enable interrupts
$1C2F
                 <instr> xxx
```





Moving Interrupts Between Application and Boot Space

General Interrupt Control Register – GICR The General Interrupt Control Register controls the placement of the Interrupt Vector table.

Bit	7	6	5	4	3	2	1	0	
	INT1	INTO	INT2	-	-	-	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	•
Initial Value	0	0	0	0	0	0	0	0	

· Bit 1 - IVSEL: Interrupt Vector Select

When the IVSEL bit is cleared (zero), the Interrupt Vectors are placed at the start of the Flash memory. When this bit is set (one), the interrupt vectors are moved to the beginning of the Boot Loader section of the Flash. The actual address of the start of the Boot Flash section is determined by the BOOTSZ Fuses. Refer to the section "Boot Loader Support – Read-While-Write Self-Programming" on page 246 for details. To avoid unintentional changes of Interrupt Vector tables, a special write procedure must be followed to change the IVSEL bit:

- 1. Write the Interrupt Vector Change Enable (IVCE) bit to one.
- 2. Within four cycles, write the desired value to IVSEL while writing a zero to IVCE.

Interrupts will automatically be disabled while this sequence is executed. Interrupts are disabled in the cycle IVCE is set, and they remain disabled until after the instruction following the write to IVSEL. If IVSEL is not written, interrupts remain disabled for four cycles. The I-bit in the Status Register is unaffected by the automatic disabling.

Note: If Interrupt Vectors are placed in the Boot Loader section and Boot Lock bit BLB02 is programmed, interrupts are disabled while executing from the Application section. If Interrupt Vectors are placed in the Application section and Boot Lock bit BLB12 is programed, interrupts are disabled while executing from the Boot Loader section. Refer to the section "Boot Loader Support – Read-While-Write Self-Programming" on page 246 for details on Boot Lock bits.

· Bit 0 - IVCE: Interrupt Vector Change Enable

The IVCE bit must be written to logic one to enable change of the IVSEL bit. IVCE is cleared by hardware four cycles after it is written or when IVSEL is written. Setting the IVCE bit will disable interrupts, as explained in the IVSEL description above. See Code Example below





```
Assembly Code Example

Move_interrupts:
    ; Enable change of interrupt vectors

ldi r16, (1<<IVCE)
    out GICR, r16
    ; Move interrupts to boot Flash section

ldi r16, (1<<IVSEL)
    out GICR, r16
    ret

C Code Example

void Move_interrupts(void)
{
    /* Enable change of interrupt vectors */
    GICR = (1<<IVCE);
    /* Move interrupts to boot Flash section */
    GICR = (1<<IVSEL);
}
```



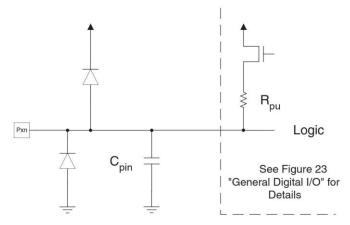


I/O Ports

Introduction

All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both $V_{\rm CC}$ and Ground as indicated in Figure 22. Refer to "Electrical Characteristics" on page 291 for a complete list of parameters.

Figure 22. I/O Pin Equivalent Schematic



All registers and bit references in this section are written in general form. A lower case "x" represents the numbering letter for the port, and a lower case "n" represents the bit number. However, when using the register or bit defines in a program, the precise form must be used, that is, PORTB3 for bit no. 3 in Port B, here documented generally as PORTxn. The physical I/O Registers and bit locations are listed in "Register Description for I/O Ports" on page 66.

Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. In addition, the Pull-up Disable – PUD bit in SFIOR disables the pull-up function for all pins in all ports when set.

Using the I/O port as General Digital I/O is described in "Ports as General Digital I/O" on page 50. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in "Alternate Port Functions" on page 55. Refer to the individual module sections for a full description of the alternate functions.

Note that enabling the alternate function of some of the port pins does not affect the use of the other pins in the port as general digital I/O.

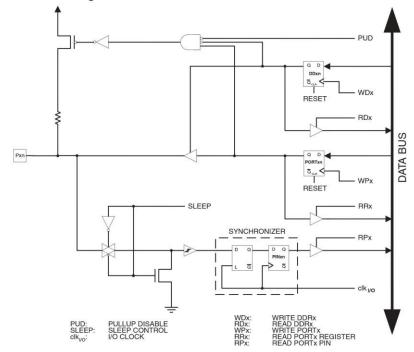
Ports as General Digital I/O

The ports are bi-directional I/O ports with optional internal pull-ups. Figure 23 shows a functional description of one I/O-port pin, here generically called Pxn.





Figure 23. General Digital I/O(1)



 WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. clk_{I/O}, SLEEP, and PUD are common to all ports.

Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in "Register Description for I/O Ports" on page 66, the DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, and the PINxn bits at the PINx I/O address.

The DDxn bit in the DDRx Register selects the direction of this pin. If DDxn is written logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is configured as an input pin.

If PORTxn is written logic one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written logic zero or the pin has to be configured as an output pin. The port pins are tri-stated when a reset condition becomes active, even if no clocks are running.

If PORTxn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

When switching between tri-state ({DDxn, PORTxn} = 0b00) and output high ({DDxn, PORTxn} = 0b11), an intermediate state with either pull-up enabled ({DDxn, PORTxn} = 0b01) or output low ({DDxn, PORTxn} = 0b10) must occur. Normally, the pull-up enabled state is fully acceptable, as a high-impedant environment will not notice the difference between a strong high driver and a pull-up. If this is not the case, the PUD bit in the SFIOR Register can be set to disable all pull-ups in all ports.





Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state ({DDxn, PORTxn} = 0b00) or the output high state ({DDxn, PORTxn} = 0b11) as an intermediate step.

Table 20 summarizes the control signals for the pin value.

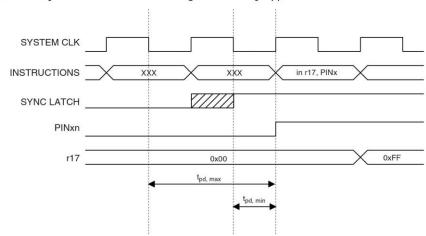
Table 20. Port Pin Configurations

DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	Х	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	Х	Output	No	Output Low (Sink)
1	1	Х	Output	No	Output High (Source)

Reading the Pin Value

Independent of the setting of Data Direction bit DDxn, the port pin can be read through the PINxn Register bit. As shown in Figure 23, the PINxn Register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. Figure 24 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{\rm pd,max}$ and $t_{\rm pd,min}$ respectively.

Figure 24. Synchronization when Reading an Externally Applied Pin Value



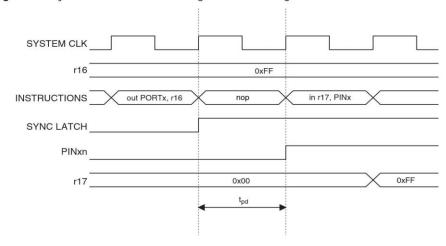
Consider the clock period starting shortly *after* the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the "SYNC LATCH" signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn Register at the succeeding positive clock edge. As indicated by the two arrows $t_{pd,max}$ and $t_{pd,min'}$ a single signal transition on the pin will be delayed between ½ and 1½ system clock period depending upon the time of assertion.





When reading back a software assigned pin value, a *nop* instruction must be inserted as indicated in Figure 25. The *out* instruction sets the "SYNC LATCH" signal at the positive edge of the clock. In this case, the delay t_{pd} through the synchronizer is one system clock period.

Figure 25. Synchronization when Reading a Software Assigned Pin Value







The following code example shows how to set port B pins 0 and 1 high, 2 and 3 low, and define the port pins from 4 to 7 as input with pull-ups assigned to port pins 6 and 7. The resulting pin values are read back again, but as previously discussed, a *nop* instruction is included to be able to read back the value recently assigned to some of the pins.

```
Assembly Code Example<sup>(1)</sup>
      ; Define pull-ups and set outputs high
      ; Define directions for port pins
      ldi r16, (1<<PB7) | (1<<PB6) | (1<<PB1) | (1<<PB0)
      ldi r17, (1<<DDB3) | (1<<DDB2) | (1<<DDB1) | (1<<DDB0)
      out PORTB. r16
      out DDRB.r17
      ; Insert nop for synchronization
      ; Read port pins
           r16, PINB
C Code Example<sup>(1)</sup>
    unsigned char i;
      /* Define pull-ups and set outputs high */
      /* Define directions for port pins */
      PORTB = (1<<PB7) | (1<<PB6) | (1<<PB1) | (1<<PB0);
      DDRB = (1<<DDB3) | (1<<DDB2) | (1<<DDB1) | (1<<DDB0);
      /* Insert nop for synchronization*/
      _NOP();
      /* Read port pins */
      i = PINB;
```

Note:

 For the assembly program, two temporary registers are used to minimize the time from pullups are set on pins 0, 1, 6, and 7, until the direction bits are correctly set, defining bit 2 and 3 as low and redefining bits 0 and 1 as strong high drivers.

Digital Input Enable and Sleep Modes

As shown in Figure 23, the digital input signal can be clamped to ground at the input of the schmitt-trigger. The signal denoted SLEEP in the figure, is set by the MCU Sleep Controller in Power-down mode, Power-save mode, Standby mode, and Extended Standby mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{\rm CC}/2$.

SLEEP is overridden for port pins enabled as External Interrupt pins. If the External Interrupt Request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in "Alternate Port Functions" on page 55.

If a logic high level ("one") is present on an Asynchronous External Interrupt pin configured as "Interrupt on Rising Edge, Falling Edge, or Any Logic Change on Pin" while the External Interrupt is *not* enabled, the corresponding External Interrupt Flag will be set when resuming from the above mentioned sleep modes, as the clamping in these sleep modes produces the requested logic change.





Unconnected pins

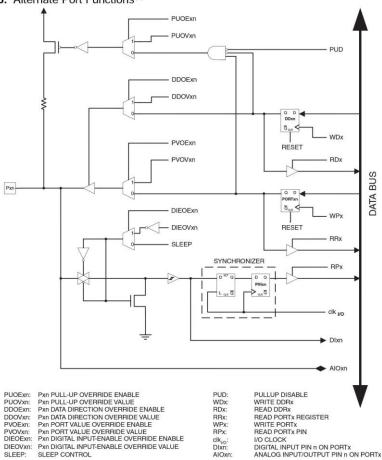
If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled (Reset, Active mode and Idle mode).

The simplest method to ensure a defined level of an unused pin, is to enable the internal pull-up. In this case, the pull-up will be disabled during reset. If low power consumption during reset is important, it is recommended to use an external pull-up or pull-down. Connecting unused pins directly to $V_{\rm CC}$ or GND is not recommended, since this may cause excessive currents if the pin is accidentally configured as an output.

Alternate Port Functions

Most port pins have alternate functions in addition to being General Digital I/Os. Figure 26 shows how the port pin control signals from the simplified Figure 23 can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the AVR microcontroller family.

Figure 26. Alternate Port Functions(1)



 WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. clk_{I/O}, SLEEP, and PUD are common to all ports. All other signals are unique for each pin.





Table 21 summarizes the function of the overriding signals. The pin and port indexes from Figure 26 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 21. Generic Description of Overriding Signals for Alternate Functions

Signal Name	Full Name	Description
PUOE	Pull-up Override Enable	If this signal is set, the pull-up enable is controlled by the PUOV signal. If this signal is cleared, the pull-up is enabled when {DDxn, PORTxn, PUD} = 0b010.
PUOV	Pull-up Override Value	If PUOE is set, the pull-up is enabled/disabled when PUOV is set/cleared, regardless of the setting of the DDxn, PORTxn, and PUD Register bits.
DDOE	Data Direction Override Enable	If this signal is set, the Output Driver Enable is controlled by the DDOV signal. If this signal is cleared, the Output driver is enabled by the DDxn Register bit.
DDOV	Data Direction Override Value	If DDOE is set, the Output Driver is enabled/disabled when DDOV is set/cleared, regardless of the setting of the DDxn Register bit.
PVOE	Port Value Override Enable	If this signal is set and the Output Driver is enabled, the port value is controlled by the PVOV signal. If PVOE is cleared, and the Output Driver is enabled, the port Value is controlled by the PORTxn Register bit.
PVOV	Port Value Override Value	If PVOE is set, the port value is set to PVOV, regardless of the setting of the PORTxn Register bit.
DIEOE	Digital Input Enable Override Enable	If this bit is set, the Digital Input Enable is controlled by the DIEOV signal. If this signal is cleared, the Digital Input Enable is determined by MCU-state (Normal Mode, sleep modes).
DIEOV	Digital Input Enable Override Value	If DIEOE is set, the Digital Input is enabled/disabled when DIEOV is set/cleared, regardless of the MCU state (Normal Mode, sleep modes).
DI	Digital Input	This is the Digital Input to alternate functions. In the figure, the signal is connected to the output of the schmitt trigger but before the synchronizer. Unless the Digital Input is used as a clock source, the module with the alternate function will use its own synchronizer.
AIO	Analog Input/ output	This is the Analog Input/output to/from alternate functions. The signal is connected directly to the pad, and can be used bi-directionally.

The following subsections shortly describe the alternate functions for each port, and relate the overriding signals to the alternate function. Refer to the alternate function description for further details.





Special Function I/O Register – SFIOR

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	40	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	•
Initial Value	0	0	0	0	0	0	0	0	

· Bit 2 - PUD: Pull-up disable

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01). See "Configuring the Pin" on page 51 for more details about this feature.

Alternate Functions of Port A

Port A has an alternate function as analog input for the ADC as shown in Table 22. If some Port A pins are configured as outputs, it is essential that these do not switch when a conversion is in progress. This might corrupt the result of the conversion.

Table 22. Port A Pins Alternate Functions

Port Pin	Alternate Function
PA7	ADC7 (ADC input channel 7)
PA6	ADC6 (ADC input channel 6)
PA5	ADC5 (ADC input channel 5)
PA4	ADC4 (ADC input channel 4)
PA3	ADC3 (ADC input channel 3)
PA2	ADC2 (ADC input channel 2)
PA1	ADC1 (ADC input channel 1)
PA0	ADC0 (ADC input channel 0)

Table 23 and Table 24 relate the alternate functions of Port A to the overriding signals shown in Figure 26 on page 55.

Table 23. Overriding Signals for Alternate Functions in PA7..PA4

Signal Name	PA7/ADC7	PA6/ADC6	PA5/ADC5	PA4/ADC4
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	1-1	_	-	×
AIO	ADC7 INPUT	ADC6 INPUT	ADC5 INPUT	ADC4 INPUT





Table 24. Overriding Signals for Alternate Functions in PA3..PA0

Signal Name	PA3/ADC3	PA2/ADC2	PA1/ADC1	PA0/ADC0
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	-		-	-
AIO	ADC3 INPUT	ADC2 INPUT	ADC1 INPUT	ADC0 INPUT

Port B

Alternate Functions of The Port B pins with alternate functions are shown in Table 25.

Table 25. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB7	SCK (SPI Bus Serial Clock)
PB6	MISO (SPI Bus Master Input/Slave Output)
PB5	MOSI (SPI Bus Master Output/Slave Input)
PB4	SS (SPI Slave Select Input)
PB3	AIN1 (Analog Comparator Negative Input) OC0 (Timer/Counter0 Output Compare Match Output)
PB2	AINO (Analog Comparator Positive Input) INT2 (External Interrupt 2 Input)
PB1	T1 (Timer/Counter1 External Counter Input)
PB0	T0 (Timer/Counter0 External Counter Input) XCK (USART External Clock Input/Output)

The alternate pin configuration is as follows:

SCK – Port B, Bit 7

SCK: Master Clock output, Slave Clock input pin for SPI channel. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB7. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB7. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB7 bit.

· MISO - Port B, Bit 6

MISO: Master Data input, Slave Data output pin for SPI channel. When the SPI is enabled as a Master, this pin is configured as an input regardless of the setting of DDB6. When the SPI is enabled as a Slave, the data direction of this pin is controlled by DDB6. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB6 bit.





· MOSI - Port B, Bit 5

MOSI: SPI Master Data output, Slave Data input for SPI channel. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB5. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB5. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB5 bit.

SS – Port B, Bit 4

SS: Slave Select input. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB4. As a Slave, the SPI is activated when this pin is driven low. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB4. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB4 bit.

AIN1/OC0 – Port B, Bit 3

AIN1, Analog Comparator Negative Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the analog comparator.

OC0, Output Compare Match output: The PB3 pin can serve as an external output for the Timer/Counter0 Compare Match. The PB3 pin has to be configured as an output (DDB3 set (one)) to serve this function. The OC0 pin is also the output pin for the PWM mode timer function.

· AIN0/INT2 - Port B, Bit 2

AINO, Analog Comparator Positive input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.

INT2, External Interrupt Source 2: The PB2 pin can serve as an external interrupt source to the MCU.

T1 – Port B, Bit 1

T1, Timer/Counter1 Counter Source.

· T0/XCK - Port B, Bit 0

T0, Timer/Counter0 Counter Source.

XCK, USART External Clock. The Data Direction Register (DDB0) controls whether the clock is output (DDB0 set) or input (DDB0 cleared). The XCK pin is active only when the USART operates in Synchronous mode.

Table 26 and Table 27 relate the alternate functions of Port B to the overriding signals shown in Figure 26 on page 55. SPI MSTR INPUT and SPI SLAVE OUTPUT constitute the MISO signal, while MOSI is divided into SPI MSTR OUTPUT and SPI SLAVE INPUT.





Table 26. Overriding Signals for Alternate Functions in PB7..PB4

Signal Name	PB7/SCK	PB6/MISO	PB5/MOSI	PB4/SS
PUOE	SPE • MSTR	SPE • MSTR	SPE • MSTR	SPE • MSTR
PUOV	PORTB7 • PUD	PORTB6 • PUD	PORTB5 • PUD	PORTB4 • PUD
DDOE	SPE • MSTR	SPE • MSTR	SPE • MSTR	SPE • MSTR
DDOV	0	0	0	0
PVOE	SPE • MSTR	SPE • MSTR	SPE • MSTR	0
PVOV	SCK OUTPUT	SPI SLAVE OUTPUT	SPI MSTR OUTPUT	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	SCK INPUT	SPI MSTR INPUT	SPI SLAVE INPUT	SPI SS
AIO	_	15	_	_

Table 27. Overriding Signals for Alternate Functions in PB3..PB0

Signal Name	PB3/OC0/AIN1	PB2/INT2/AIN0	PB1/T1	PB0/T0/XCK
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC0 ENABLE	0	0	UMSEL
PVOV	OC0	0	0	XCK OUTPUT
DIEOE	0	INT2 ENABLE	0	0
DIEOV	0	1	0	0
DI	_	INT2 INPUT	T1 INPUT	XCK INPUT/T0 INPUT
AIO	AIN1 INPUT	AINO INPUT	-	_



2466T-AVR-07/10



Alternate Functions of Port C

The Port C pins with alternate functions are shown in Table 28. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs.

Table 28. Port C Pins Alternate Functions

Port Pin	Alternate Function			
PC7	TOSC2 (Timer Oscillator Pin 2)			
PC6	TOSC1 (Timer Oscillator Pin 1)			
PC5	TDI (JTAG Test Data In)			
PC4	TDO (JTAG Test Data Out)			
PC3	TMS (JTAG Test Mode Select)			
PC2	TCK (JTAG Test Clock)			
PC1	SDA (Two-wire Serial Bus Data Input/Output Line)			
PC0	SCL (Two-wire Serial Bus Clock Line)			

The alternate pin configuration is as follows:

TOSC2 – Port C, Bit 7

TOSC2, Timer Oscillator pin 2: When the AS2 bit in ASSR is set (one) to enable asynchronous clocking of Timer/Counter2, pin PC7 is disconnected from the port, and becomes the inverting output of the Oscillator amplifier. In this mode, a Crystal Oscillator is connected to this pin, and the pin can not be used as an I/O pin.

• TOSC1 - Port C, Bit 6

TOSC1, Timer Oscillator pin 1: When the AS2 bit in ASSR is set (one) to enable asynchronous clocking of Timer/Counter2, pin PC6 is disconnected from the port, and becomes the input of the inverting Oscillator amplifier. In this mode, a Crystal Oscillator is connected to this pin, and the pin can not be used as an I/O pin.

TDI – Port C, Bit 5

TDI, JTAG Test Data In: Serial input data to be shifted in to the Instruction Register or Data Register (scan chains). When the JTAG interface is enabled, this pin can not be used as an I/O pin.

· TDO - Port C, Bit 4

TDO, JTAG Test Data Out: Serial output data from Instruction Register or Data Register. When the JTAG interface is enabled, this pin can not be used as an I/O pin.

The TD0 pin is tri-stated unless TAP states that shifts out data are entered.

TMS – Port C, Bit 3

TMS, JTAG Test Mode Select: This pin is used for navigating through the TAP-controller state machine. When the JTAG interface is enabled, this pin can not be used as an I/O pin.

· TCK - Port C, Bit 2

TCK, JTAG Test Clock: JTAG operation is synchronous to TCK. When the JTAG interface is enabled, this pin can not be used as an I/O pin.



61

2466T-AVR-07/10



· SDA - Port C, Bit 1

SDA, Two-wire Serial Interface Data: When the TWEN bit in TWCR is set (one) to enable the Two-wire Serial Interface, pin PC1 is disconnected from the port and becomes the Serial Data I/O pin for the Two-wire Serial Interface. In this mode, there is a spike filter on the pin to suppress spikes shorter than 50 ns on the input signal, and the pin is driven by an open drain driver with slew-rate limitation. When this pin is used by the Two-wire Serial Interface, the pull-up can still be controlled by the PORTC1 bit.

· SCL - Port C, Bit 0

SCL, Two-wire Serial Interface Clock: When the TWEN bit in TWCR is set (one) to enable the Two-wire Serial Interface, pin PC0 is disconnected from the port and becomes the Serial Clock I/O pin for the Two-wire Serial Interface. In this mode, there is a spike filter on the pin to suppress spikes shorter than 50 ns on the input signal, and the pin is driven by an open drain driver with slew-rate limitation. When this pin is used by the Two-wire Serial Interface, the pull-up can still be controlled by the PORTC0 bit.

Table 29 and Table 30 relate the alternate functions of Port C to the overriding signals shown in Figure 26 on page 55.

Table 29. Overriding Signals for Alternate Functions in PC7..PC4

Signal Name	PC7/TOSC2	PC6/TOSC1	PC5/TDI	PC4/TDO
PUOE	AS2	AS2	JTAGEN	JTAGEN
PUOV	0	0	1	0
DDOE	AS2	AS2	JTAGEN	JTAGEN
DDOV	0	0	0	SHIFT_IR + SHIFT_DR
PVOE	0	0	0	JTAGEN
PVOV	0	0	0	TDO
DIEOE	AS2	AS2	JTAGEN	JTAGEN
DIEOV	0	0	0	0
DI	-	-	-	-
AIO	T/C2 OSC OUTPUT	T/C2 OSC INPUT	TDI	-





Table 30. Overriding Signals for Alternate Functions in PC3..PC0⁽¹⁾

Signal Name	PC3/TMS	PC2/TCK	PC1/SDA	PC0/SCL
PUOE	JTAGEN	JTAGEN	TWEN	TWEN
PUOV	1	1	PORTC1 • PUD	PORTC0 • PUD
DDOE	JTAGEN	JTAGEN	TWEN	TWEN
DDOV	0	0	SDA_OUT	SCL_OUT
PVOE	0	0	TWEN	TWEN
PVOV	0	0	0	0
DIEOE	JTAGEN	JTAGEN	0	0
DIEOV	0	0	0	0
DI	_	_	-	-
AIO	TMS	TCK	SDA INPUT	SCL INPUT

Note: 1. When enabled, the Two-wire Serial Interface enables slew-rate controls on the output pins PC0 and PC1. This is not shown in the figure. In addition, spike filters are connected between the AIO outputs shown in the port figure and the digital logic of the TWI module.

Alternate Functions of Port D

Alternate Functions of The Port D pins with alternate functions are shown in Table 31.

Table 31. Port D Pins Alternate Functions

Port Pin	Alternate Function			
PD7	OC2 (Timer/Counter2 Output Compare Match Output)			
PD6	ICP1 (Timer/Counter1 Input Capture Pin)			
PD5	OC1A (Timer/Counter1 Output Compare A Match Output)			
PD4	OC1B (Timer/Counter1 Output Compare B Match Output)			
PD3	INT1 (External Interrupt 1 Input)			
PD2	INTO (External Interrupt 0 Input)			
PD1	TXD (USART Output Pin)			
PD0	RXD (USART Input Pin)			

The alternate pin configuration is as follows:

OC2 – Port D, Bit 7

OC2, Timer/Counter2 Output Compare Match output: The PD7 pin can serve as an external output for the Timer/Counter2 Output Compare. The pin has to be configured as an output (DDD7 set (one)) to serve this function. The OC2 pin is also the output pin for the PWM mode timer function.

· ICP1 - Port D, Bit 6

ICP1 – Input Capture Pin: The PD6 pin can act as an Input Capture pin for Timer/Counter1.





· OC1A - Port D, Bit 5

OC1A, Output Compare Match A output: The PD5 pin can serve as an external output for the Timer/Counter1 Output Compare A. The pin has to be configured as an output (DDD5 set (one)) to serve this function. The OC1A pin is also the output pin for the PWM mode timer function.

· OC1B - Port D, Bit 4

OC1B, Output Compare Match B output: The PD4 pin can serve as an external output for the Timer/Counter1 Output Compare B. The pin has to be configured as an output (DDD4 set (one)) to serve this function. The OC1B pin is also the output pin for the PWM mode timer function.

• INT1 - Port D, Bit 3

INT1, External Interrupt Source 1: The PD3 pin can serve as an external interrupt source.

· INT0 - Port D. Bit 2

INTO, External Interrupt Source 0: The PD2 pin can serve as an external interrupt source.

· TXD - Port D, Bit 1

TXD, Transmit Data (Data output pin for the USART). When the USART Transmitter is enabled, this pin is configured as an output regardless of the value of DDD1.

· RXD - Port D, Bit 0

RXD, Receive Data (Data input pin for the USART). When the USART Receiver is enabled this pin is configured as an input regardless of the value of DDD0. When the USART forces this pin to be an input, the pull-up can still be controlled by the PORTD0 bit.

Table 32 and Table 33 relate the alternate functions of Port D to the overriding signals shown in Figure 26 on page 55.

Table 32. Overriding Signals for Alternate Functions PD7..PD4

Signal Name	PD7/OC2	PD6/ICP1	PD5/OC1A	PD4/OC1B
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC2 ENABLE	0	OC1A ENABLE	OC1B ENABLE
PVOV	OC2	0	OC1A	OC1B
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	-	ICP1 INPUT	-	-
AIO	-	-	-	-





Table 33. Overriding Signals for Alternate Functions in PD3..PD0

Signal Name	PD3/INT1	PD2/INT0	PD1/TXD	PD0/RXD
PUOE	0	0	TXEN	RXEN
PUOV	0	0	0	PORTD0 ⋅ PUD
DDOE	0	0	TXEN	RXEN
DDOV	0	0	1	0
PVOE	0	0	TXEN	0
PVOV	0	0	TXD	0
DIEOE	INT1 ENABLE	INTO ENABLE	0	0
DIEOV	1	1	0	0
DI	INT1 INPUT	INTO INPUT	=	RXD
AIO	_	_	=	-





Register Description for I/O **Ports**

Port A Data Register -**PORTA**

Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W								
Initial Value	0	0	0	0	n	0	0	0	

Port A Data Direction Register - DDRA

Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Read/Write	R/W	.5							
Initial Value	0	0	0	0	0	0	0	0	

Port A Input Pins Address - PINA

Bit	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	•
Initial Value	N/A								

Port B Data Register -**PORTB**

Bit	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

Port B Data Direction Register - DDRB

Die	7	6	-		2	2	1	0	
Bit	,	0	5	4	3	2	2224	0	
2 Had 3 HOLD (1997)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

Port B Input Pins Address - PINB

Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	•
Initial Value	N/A								





Port C Data	Register -
PORTC	_

Bit	7	6	5	4	3	2	1	0	
	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
Read/Write	R/W	ı							
Initial Value	0	0	0	0	0	0	0	0	

Port C Data Direction Register – DDRC

Bit	7	6	5	4	3	2	1	0	
	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
Read/Write	R/W	•,							
Initial Value	0	0	0	0	0	0	0	0	

Port C Input Pins Address – PINC

Bit	7	6	5	4	3	2	1	0	
	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A								

Port D Data Register – PORTD

Bit	7	6	5	4	3	2	1	0	
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W	k.							
Initial Value	0	0	0	0	0	0	0	0	

Port D Data Direction Register – DDRD

Bit	7	6	5	4	3	2	1	0	
	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	•							
Initial Value	0	0	0	0	0	0	0	0	

Port D Input Pins Address – PIND

Bit	7	6	5	4	3	2	1	0	
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIN
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A								





External Interrupts

The External Interrupts are triggered by the INTO, INT1, and INT2 pins. Observe that, if enabled, the interrupts will trigger even if the INTO..2 pins are configured as outputs. This feature provides a way of generating a software interrupt. The external interrupts can be triggered by a falling or rising edge or a low level (INT2 is only an edge triggered interrupt). This is set up as indicated in the specification for the MCU Control Register – MCUCR – and MCU Control and Status Register – MCUCSR. When the external interrupt is enabled and is configured as level triggered (only INTO/INT1), the interrupt will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INTO and INT1 requires the presence of an I/O clock, described in "Clock Systems and their Distribution" on page 24. Low level interrupts on INTO/INT1 and the edge interrupt on INT2 are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode. The I/O clock is halted in all sleep modes except Idle mode.

Note that if a level triggered interrupt is used for wake-up from Power-down mode, the changed level must be held for some time to wake up the MCU. This makes the MCU less sensitive to noise. The changed level is sampled twice by the Watchdog Oscillator clock. The period of the Watchdog Oscillator is 1 µs (nominal) at 5.0V and 25°C. The frequency of the Watchdog Oscillator is voltage dependent as shown in "Electrical Characteristics" on page 291. The MCU will wake up if the input has the required level during this sampling or if it is held until the end of the start-up time. The start-up time is defined by the SUT Fuses as described in "System Clock and Clock Options" on page 24. If the level is sampled twice by the Watchdog Oscillator clock but disappears before the end of the start-up time, the MCU will still wake up, but no interrupt will be generated. The required level must be held long enough for the MCU to complete the wake up to trigger the level interrupt.

MCU Control Register – MCUCR

The MCU Control Register contains control bits for interrupt sense control and general MCU functions.

Bit	7	6	5	4	3	2	1	0	
	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	n	0	0	0	

Bit 3, 2 – ISC11, ISC10: Interrupt Sense Control 1 Bit 1 and Bit 0

The External Interrupt 1 is activated by the external pin INT1 if the SREG I-bit and the corresponding interrupt mask in the GICR are set. The level and edges on the external INT1 pin that activate the interrupt are defined in Table 34. The value on the INT1 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 34. Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.





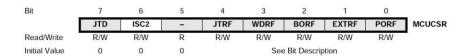
• Bit 1, 0 - ISC01, ISC00: Interrupt Sense Control 0 Bit 1 and Bit 0

The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in Table 35. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 35. Interrupt 0 Sense Control

ISC01	ISC00	Description				
0	0	The low level of INTO generates an interrupt request.				
0	1	Any logical change on INT0 generates an interrupt request.				
1	0	The falling edge of INTO generates an interrupt request.				
1	1	The rising edge of INTO generates an interrupt request.				

MCU Control and Status Register – MCUCSR



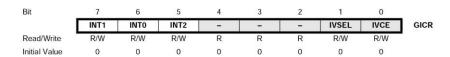
· Bit 6 - ISC2: Interrupt Sense Control 2

The Asynchronous External Interrupt 2 is activated by the external pin INT2 if the SREG I-bit and the corresponding interrupt mask in GICR are set. If ISC2 is written to zero, a falling edge on INT2 activates the interrupt. If ISC2 is written to one, a rising edge on INT2 activates the interrupt. Edges on INT2 are registered asynchronously. Pulses on INT2 wider than the minimum pulse width given in Table 36 will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. When changing the ISC2 bit, an interrupt can occur. Therefore, it is recommended to first disable INT2 by clearing its Interrupt Enable bit in the GICR Register. Then, the ISC2 bit can be changed. Finally, the INT2 Interrupt Flag should be cleared by writing a logical one to its Interrupt Flag bit (INTF2) in the GIFR Register before the interrupt is re-enabled.

Table 36. Asynchronous External Interrupt Characteristics

Symbol	Parameter	Condition	Min	Тур	Max	Units
t _{INT}	Minimum pulse width for asynchronous external interrupt			50		ns

General Interrupt Control Register – GICR



• Bit 7 - INT1: External Interrupt Request 1 Enable

When the INT1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control1 bits 1/0 (ISC11 and ISC10) in the MCU General Control Register (MCUCR) define whether the External Interrupt is activated on rising

ATMEL

69

2466T-AVR-07/10



and/or falling edge of the INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The corresponding interrupt of External Interrupt Request 1 is executed from the INT1 interrupt Vector.

Bit 6 – INT0: External Interrupt Request 0 Enable

When the INT0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control0 bits 1/0 (ISC01 and ISC00) in the MCU General Control Register (MCUCR) define whether the External Interrupt is activated on rising and/or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of External Interrupt Request 0 is executed from the INT0 interrupt vector.

· Bit 5 - INT2: External Interrupt Request 2 Enable

When the INT2 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control2 bit (ISC2) in the MCU Control and Status Register (MCUCSR) defines whether the External Interrupt is activated on rising or falling edge of the INT2 pin. Activity on the pin will cause an interrupt request even if INT2 is configured as an output. The corresponding interrupt of External Interrupt Request 2 is executed from the INT2 Interrupt Vector.

General Interrupt Flag Register – GIFR



Bit 7 – INTF1: External Interrupt Flag 1

When an edge or logic change on the INT1 pin triggers an interrupt request, INTF1 becomes set (one). If the I-bit in SREG and the INT1 bit in GICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT1 is configured as a level interrupt.

· Bit 6 - INTF0: External Interrupt Flag 0

When an edge or logic change on the INTO pin triggers an interrupt request, INTFO becomes set (one). If the I-bit in SREG and the INTO bit in GICR are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INTO is configured as a level interrupt.

• Bit 5 - INTF2: External Interrupt Flag 2

When an event on the INT2 pin triggers an interrupt request, INTF2 becomes set (one). If the I-bit in SREG and the INT2 bit in GICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. Note that when entering some sleep modes with the INT2 interrupt disabled, the input buffer on this pin will be disabled. This may cause a logic change in internal signals which will set the INTF2 Flag. See "Digital Input Enable and Sleep Modes" on page 54 for more information.





8-bit Timer/Counter0 with PWM

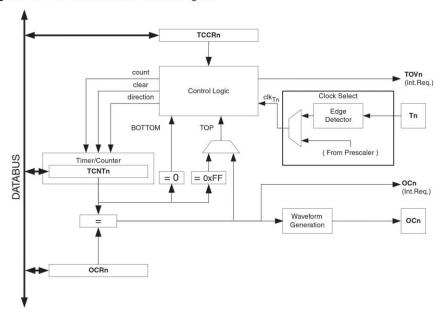
Timer/Counter0 is a general purpose, single compare unit, 8-bit Timer/Counter module. The main features are:

- · Single Compare Unit Counter
- · Clear Timer on Compare Match (Auto Reload)
- · Glitch-free, Phase Correct Pulse Width Modulator (PWM)
- · Frequency Generator
- · External Event Counter
- 10-bit Clock Prescaler
- Overflow and Compare Match Interrupt Sources (TOV0 and OCF0)

Overview

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 27. For the actual placement of I/O pins, refer to "Pinout ATmega16" on page 2. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the "8-bit Timer/Counter Register Description" on page 83.

Figure 27. 8-bit Timer/Counter Block Diagram



Registers

The Timer/Counter (TCNT0) and Output Compare Register (OCR0) are 8-bit registers. Interrupt request (abbreviated to Int.Req. in the figure) signals are all visible in the Timer Interrupt Flag Register (TIFR). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK). TIFR and TIMSK are not shown in the figure since these registers are shared by other timer units.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the T0 pin. The Clock Select logic block controls which clock source and edge the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock (clk_{T0}).

The double buffered Output Compare Register (OCR0) is compared with the Timer/Counter value at all times. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the Output Compare Pin (OC0). See "Output Compare





Unit" on page 73. for details. The compare match event will also set the Compare Flag (OCF0) which can be used to generate an output compare interrupt request.

Definitions

Many register and bit references in this document are written in general form. A lower case "n" replaces the Timer/Counter number, in this case 0. However, when using the register or bit defines in a program, the precise form must be used, that is, TCNT0 for accessing Timer/Counter0 counter value and so on.

The definitions in Table 37 are also used extensively throughout the document.

Table 37. Definitions

воттом	The counter reaches the BOTTOM when it becomes 0x00.
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
ТОР	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR0 Register. The assignment is dependent on the mode of operation.

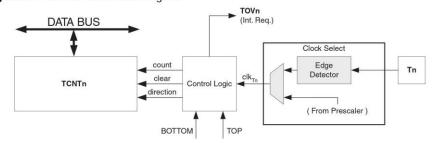
Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the clock select logic which is controlled by the clock select (CS02:0) bits located in the Timer/Counter Control Register (TCCR0). For details on clock sources and prescaler, see "Timer/Counter0 and Timer/Counter1 Prescalers" on page 87.

Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 28 shows a block diagram of the counter and its surroundings.

Figure 28. Counter Unit Block Diagram



Signal description (internal signals):

count Increment or decrement TCNT0 by 1.direction Select between increment and decrement.

clear TCNT0 (set all bits to zero).

clk_{Tn} Timer/Counter clock, referred to as clk_{T0} in the following.
 TOP Signalize that TCNT0 has reached maximum value.
 BOTTOM Signalize that TCNT0 has reached minimum value (zero).

Depending of the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T0}) . clk_{T0} can be generated from an external or internal clock source, selected by the Clock Select bits (CS02:0). When no clock source is selected (CS02:0 = 0) the timer is stopped. However, the TCNT0 value can be accessed by the CPU, regardless of





whether ${\rm clk_{T0}}$ is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the WGM01 and WGM00 bits located in the Timer/Counter Control Register (TCCR0). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare output OC0. For more details about advanced counting sequences and waveform generation, see "Modes of Operation" on page 76.

The Timer/Counter Overflow (TOV0) Flag is set according to the mode of operation selected by the WGM01:0 bits. TOV0 can be used for generating a CPU interrupt.

Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the Output Compare Register (OCR0). Whenever TCNT0 equals OCR0, the comparator signals a match. A match will set the Output Compare Flag (OCF0) at the next timer clock cycle. If enabled (OCIE0 = 1 and Global Interrupt Flag in SREG is set), the Output Compare Flag generates an output compare interrupt. The OCF0 Flag is automatically cleared when the interrupt is executed. Alternatively, the OCF0 Flag can be cleared by software by writing a logical one to its I/O bit location. The waveform generator uses the match signal to generate an output according to operating mode set by the WGM01:0 bits and Compare Output mode (COM01:0) bits. The max and bottom signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation (See "Modes of Operation" on page 76.).

Figure 29 shows a block diagram of the output compare unit.

DATA BUS

OCRn

TCNTn

OCFn (Int.Req.)

bottom
FOCn

Waveform Generator
FOCn

OCN

OCN

Figure 29. Output Compare Unit, Block Diagram

The OCRO Register is double buffered when using any of the Pulse Width Modulation (PWM) modes. For the normal and Clear Timer on Compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCRO Compare Register to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.



The OCR0 Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR0 Buffer Register, and if double buffering is disabled the CPU will access the OCR0 directly.

Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the Force Output Compare (FOC0) bit. Forcing compare match will not set the OCF0 Flag or reload/clear the timer, but the OC0 pin will be updated as if a real compare match had occurred (the COM01:0 bits settings define whether the OC0 pin is set, cleared or toggled).

Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNT0 Register will block any compare match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCR0 to be initialized to the same value as TCNT0 without triggering an interrupt when the Timer/Counter clock is enabled.

Using the Output Compare Unit

Since writing TCNT0 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT0 when using the output compare unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT0 equals the OCR0 value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT0 value equal to BOTTOM when the counter is downcounting.

The setup of the OC0 should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC0 value is to use the Force Output Compare (FOC0) strobe bits in Normal mode. The OC0 Register keeps its value even when changing between waveform generation modes.

Be aware that the COM01:0 bits are not double buffered together with the compare value. Changing the COM01:0 bits will take effect immediately.

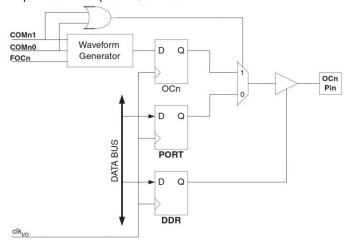
Compare Match Output Unit

The Compare Output mode (COM01:0) bits have two functions. The Waveform Generator uses the COM01:0 bits for defining the Output Compare (OC0) state at the next compare match. Also, the COM01:0 bits control the OC0 pin output source. Figure 30 shows a simplified schematic of the logic affected by the COM01:0 bit setting. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O port Control Registers (DDR and PORT) that are affected by the COM01:0 bits are shown. When referring to the OC0 state, the reference is for the internal OC0 Register, not the OC0 pin. If a System Reset occur, the OC0 Register is reset to "0".





Figure 30. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the Output Compare (OC0) from the Waveform Generator if either of the COM01:0 bits are set. However, the OC0 pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. The Data Direction Register bit for the OC0 pin (DDR_OC0) must be set as output before the OC0 value is visible on the pin. The port override function is independent of the Waveform Generation mode.

The design of the output compare pin logic allows initialization of the OC0 state before the output is enabled. Note that some COM01:0 bit settings are reserved for certain modes of operation. See "8-bit Timer/Counter Register Description" on page 83.

Compare Output Mode and Waveform Generation The Waveform Generator uses the COM01:0 bits differently in normal, CTC, and PWM modes. For all modes, setting the COM01:0 = 0 tells the waveform generator that no action on the OCO Register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 39 on page 84. For fast PWM mode, refer to Table 40 on page 84, and for phase correct PWM refer to Table 41 on page 84.

A change of the COM01:0 bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC0 strobe bits.





Modes of Operation

The mode of operation, that is, the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the Waveform Generation mode (WGM01:0) and Compare Output mode (COM01:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM01:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM01:0 bits control whether the output should be set, cleared, or toggled at a compare match (See "Compare Match Output Unit" on page 74.).

For detailed timing information refer to Figure 34, Figure 35, Figure 36 and Figure 37 in "Timer/Counter Timing Diagrams" on page 81.

Normal Mode

The simplest mode of operation is the normal mode (WGM01:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter Overflow Flag (TOV0) will be set in the same timer clock cycle as the TCNT0 becomes zero. The TOV0 Flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV0 Flag, the timer resolution can be increased by software. There are no special cases to consider in the normal mode, a new counter value can be written anytime.

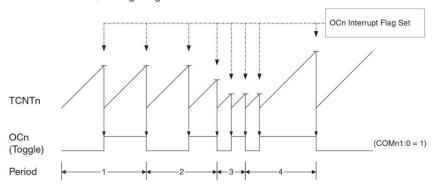
The output compare unit can be used to generate interrupts at some given time. Using the output compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM01:0 = 2), the OCR0 Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT0) matches the OCR0. The OCR0 defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 31. The counter value (TCNT0) increases until a compare match occurs between TCNT0 and OCR0, and then counter (TCNT0) is cleared.

Figure 31. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCF0 Flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have





the double buffering feature. If the new value written to OCR0 is lower than the current value of TCNT0, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFF) and wrap around starting at 0x00 before the compare match can occur.

For generating a waveform output in CTC mode, the OC0 output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COM01:0 = 1). The OC0 value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{\rm OC0} = f_{\rm clk_I/O}/2$ when OCR0 is set to zero (0x00). The waveform frequency is defined by the following equation:

$$f_{OCn} = \frac{f_{\text{clk_I/O}}}{2 \cdot N \cdot (1 + OCRn)}$$

The N variable represents the prescale factor (1, 8, 64, 256, or 1024).

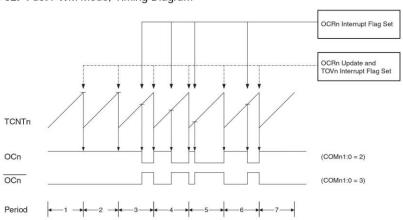
As for the Normal mode of operation, the TOV0 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

Fast PWM Mode

The fast Pulse Width Modulation or fast PWM mode (WGM01:0 = 3) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to MAX then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC0) is cleared on the compare match between TCNT0 and OCR0, and set at BOTTOM. In inverting Compare Output mode, the output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct PWM mode that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

In fast PWM mode, the counter is incremented until the counter value matches the MAX value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 32. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent compare matches between OCR0 and TCNT0.

Figure 32. Fast PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV0) is set each time the counter reaches MAX. If the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.





In fast PWM mode, the compare unit allows generation of PWM waveforms on the OC0 pin. Setting the COM01:0 bits to 2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM01:0 to 3 (See Table 40 on page 84). The actual OC0 value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC0 Register at the compare match between OCR0 and TCNT0, and clearing (or setting) the OC0 Register at the timer clock cycle the counter is cleared (changes from MAX to BOTTOM).

The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnPWM} = \frac{f_{\text{clk_I/O}}}{N \cdot 256}$$

The N variable represents the prescale factor (1, 8, 64, 256, or 1024).

The extreme values for the OCR0 Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR0 is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCR0 equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM01:0 bits.)



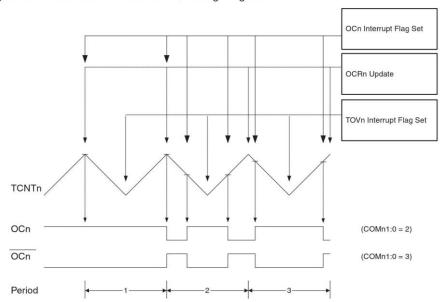


Phase Correct PWM Mode

The phase correct PWM mode (WGM01:0 = 1) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to MAX and then from MAX to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC0) is cleared on the compare match between TCNTO and OCR0 while upcounting, and set on the compare match while downcounting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The PWM resolution for the phase correct PWM mode is fixed to eight bits. In phase correct PWM mode the counter is incremented until the counter value matches MAX. When the counter reaches MAX, it changes the count direction. The TCNT0 value will be equal to MAX for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 33. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent compare matches between OCR0 and TCNT0.

Figure 33. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV0) is set each time the counter reaches BOTTOM. The Interrupt Flag can be used to generate an interrupt each time the counter reaches the BOTTOM value.

In phase correct PWM mode, the compare unit allows generation of PWM waveforms on the OC0 pin. Setting the COM01:0 bits to 2 will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM01:0 to 3 (see Table 41 on page 84). The actual OC0 value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC0 Register at the compare match between OCR0 and TCNT0 when the counter increments, and setting (or clearing) the OC0 Register at compare match between OCR0 and TCNT0 when the counter decrements. The





PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnPCPWM} = \frac{f_{\text{clk_I/O}}}{N \cdot 510}$$

The N variable represents the prescale factor (1, 8, 64, 256, or 1024).

The extreme values for the OCR0 Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR0 is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

At the very start of Period 2 in Figure 33 OCn has a transition from high to low even though there is no Compare Match. The point of this transition is to guarantee symmetry around BOTTOM. There are two cases that give a transition without Compare Match:

- OCR0A changes its value from MAX, like in Figure 33. When the OCR0A value is MAX the
 OCn pin value is the same as the result of a down-counting Compare Match. To ensure
 symmetry around BOTTOM the OCn value at MAX must be correspond to the result of an
 up-counting Compare Match.
- The Timer starts counting from a value higher than the one in OCROA, and for that reason misses the Compare Match and hence the OCn change that would have happened on the way up.





Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T0}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set. Figure 34 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 34. Timer/Counter Timing Diagram, no Prescaling

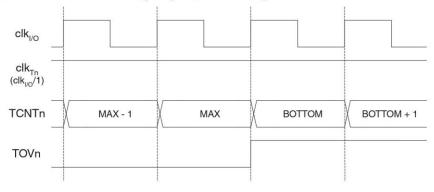


Figure 35 shows the same timing data, but with the prescaler enabled.

Figure 35. Timer/Counter Timing Diagram, with Prescaler ($f_{clk_I/O}/8$)

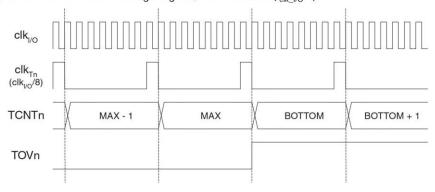


Figure 36 shows the setting of OCF0 in all modes except CTC mode.

AIMEL



Figure 36. Timer/Counter Timing Diagram, Setting of OCF0, with Prescaler ($f_{clk_I/O}/8$)

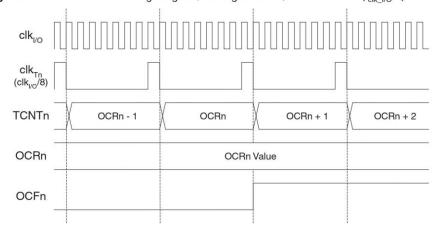
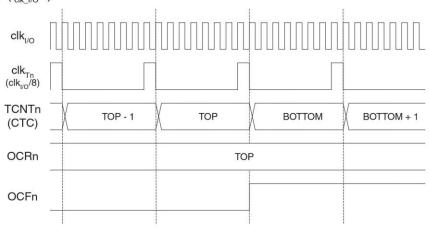


Figure 37 shows the setting of OCF0 and the clearing of TCNT0 in CTC mode.

Figure 37. Timer/Counter Timing Diagram, Clear Timer on Compare Match Mode, with Prescaler $(f_{clk_I/O}/8)$







8-bit Timer/Counter Register Description

Timer/Counter Control Register – TCCR0

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	-
Initial Value	0	0	0	0	0	0	0	0	

· Bit 7 - FOC0: Force Output Compare

The FOC0 bit is only active when the WGM00 bit specifies a non-PWM mode. However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR0 is written when operating in PWM mode. When writing a logical one to the FOC0 bit, an immediate compare match is forced on the Waveform Generation unit. The OC0 output is changed according to its COM01:0 bits setting. Note that the FOC0 bit is implemented as a strobe. Therefore it is the value present in the COM01:0 bits that determines the effect of the forced compare.

A FOC0 strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0 as TOP.

The FOC0 bit is always read as zero.

· Bit 3, 6 - WGM01:0: Waveform Generation Mode

These bits control the counting sequence of the counter, the source for the maximum (TOP) counter value, and what type of Waveform Generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode, Clear Timer on Compare Match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes. See Table 38 and "Modes of Operation" on page 76.

Table 38. Waveform Generation Mode Bit Description⁽¹⁾

WGM01 WGM00 Mode (CTC0) (PWM0)			Timer/Counter Mode of Operation	Update of OCR0	TOV0 Flag Set-on			
0	0	0	Normal	0xFF	Immediate	MAX		
1	0	1	PWM, Phase Correct	0xFF	TOP	воттом		
2	1	0	СТС	OCR0	Immediate	MAX		
3	1	1	Fast PWM	0xFF	воттом	MAX		

Note: 1. The CTC0 and PWM0 bit definition names are now obsolete. Use the WGM01:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

Bit 5:4 – COM01:0: Compare Match Output Mode

These bits control the Output Compare pin (OC0) behavior. If one or both of the COM01:0 bits are set, the OC0 output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC0 pin must be set in order to enable the output driver.





When OC0 is connected to the pin, the function of the COM01:0 bits depends on the WGM01:0 bit setting. Table 39 shows the COM01:0 bit functionality when the WGM01:0 bits are set to a normal or CTC mode (non-PWM).

Table 39. Compare Output Mode, non-PWM Mode

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Toggle OC0 on compare match
1	0	Clear OC0 on compare match
1	1	Set OC0 on compare match

Table 40 shows the COM01:0 bit functionality when the WGM01:0 bits are set to fast PWM mode.

Table 40. Compare Output Mode, Fast PWM Mode⁽¹⁾

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match, set OC0 at BOTTOM, (non-inverting mode)
1	1	Set OC0 on compare match, clear OC0 at BOTTOM, (inverting mode)

A special case occurs when OCR0 equals TOP and COM01 is set. In this case, the compare
match is ignored, but the set or clear is done at TOP. See "Fast PWM Mode" on page 77 for
more details.

Table 41 shows the COM01:0 bit functionality when the WGM01:0 bits are set to phase correct PWM mode.

Table 41. Compare Output Mode, Phase Correct PWM Mode⁽¹⁾

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match when up-counting. Set OC0 on compare match when downcounting.
1	1	Set OC0 on compare match when up-counting. Clear OC0 on compare match when downcounting.

 A special case occurs when OCR0 equals TOP and COM01 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See "Phase Correct PWM Mode" on page 79 for more details.





Bit 2:0 - CS02:0: Clock Select

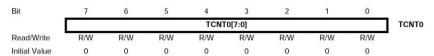
The three Clock Select bits select the clock source to be used by the Timer/Counter.

Table 42. Clock Select Bit Description

CS02	CS01	CS00	Description					
0	0	0	No clock source (Timer/Counter stopped).					
0	0	1	_{/O} /(No prescaling)					
0	1	0	_O /8 (From prescaler)					
0	1	1	lk _{I/O} /64 (From prescaler)					
1	0	0	K _{I/O} /256 (From prescaler)					
1	0	1	_{N/O} /1024 (From prescaler)					
1	1	0	External clock source on T0 pin. Clock on falling edge.					
1	1	1	External clock source on T0 pin. Clock on rising edge.					

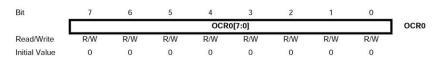
If external pin modes are used for the Timer/Counter0, transitions on the T0 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

Timer/Counter Register – TCNT0



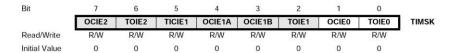
The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT0 Register blocks (removes) the compare match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a compare match between TCNT0 and the OCR0 Register.

Output Compare Register – OCR0



The Output Compare Register contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC0 pin.

Timer/Counter Interrupt Mask Register – TIMSK



• Bit 1 – OCIE0: Timer/Counter0 Output Compare Match Interrupt Enable

When the OCIE0 bit is written to one, and the I-bit in the Status Register is set (one), the Timer/Counter0 Compare Match interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter0 occurs, that is, when the OCF0 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.



85



• Bit 0 - TOIE0: Timer/Counter0 Overflow Interrupt Enable

When the TOIE0 bit is written to one, and the I-bit in the Status Register is set (one), the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, that is, when the TOV0 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

Timer/Counter Interrupt Flag Register – TIFR

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	25
Initial Value	0	0	0	0	0	0	0		

Bit 1 – OCF0: Output Compare Flag 0

The OCF0 bit is set (one) when a compare match occurs between the Timer/Counter0 and the data in OCR0 – Output Compare Register0. OCF0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0 is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0 (Timer/Counter0 Compare Match Interrupt Enable), and OCF0 are set (one), the Timer/Counter0 Compare Match Interrupt is executed.

· Bit 0 - TOV0: Timer/Counter0 Overflow Flag

The bit TOV0 is set (one) when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set (one), the Timer/Counter0 Overflow interrupt is executed. In phase correct PWM mode, this bit is set when Timer/Counter0 changes counting direction at \$00.





Timer/Counter0 and Timer/Counter1 Prescalers

Timer/Counter1 and Timer/Counter0 share the same prescaler module, but the Timer/Counters can have different prescaler settings. The description below applies to both Timer/Counter1 and Timer/Counter0.

Internal Clock Source

The Timer/Counter can be clocked directly by the system clock (by setting the CSn2:0 = 1). This provides the fastest operation, with a maximum Timer/Counter clock frequency equal to system clock frequency ($f_{CLK_I/O}$). Alternatively, one of four taps from the prescaler can be used as a clock source. The prescaled clock has a frequency of either $f_{CLK_I/O}/8$, $f_{CLK_I/O}/64$, $f_{CLK_I/O}/256$, or $f_{CLK_I/O}/1024$.

Prescaler Reset

The prescaler is free running, that is, operates independently of the clock select logic of the Timer/Counter, and it is shared by Timer/Counter1 and Timer/Counter0. Since the prescaler is not affected by the Timer/Counter's clock select, the state of the prescaler will have implications for situations where a prescaled clock is used. One example of prescaling artifacts occurs when the timer is enabled and clocked by the prescaler (6 > CSn2:0 > 1). The number of system clock cycles from when the timer is enabled to the first count occurs can be from 1 to N+1 system clock cycles, where N equals the prescaler divisor (8, 64, 256, or 1024).

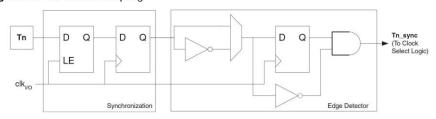
It is possible to use the Prescaler Reset for synchronizing the Timer/Counter to program execution. However, care must be taken if the other Timer/Counter that shares the same prescaler also uses prescaling. A prescaler reset will affect the prescaler period for all Timer/Counters it is connected to.

External Clock Source

An external clock source applied to the T1/T0 pin can be used as Timer/Counter clock (clk_{T1}/clk_{T0}). The T1/T0 pin is sampled once every system clock cycle by the pin synchronization logic. The synchronized (sampled) signal is then passed through the edge detector. Figure 38 shows a functional equivalent block diagram of the T1/T0 synchronization and edge detector logic. The registers are clocked at the positive edge of the internal system clock (clk_{I/O}). The latch is transparent in the high period of the internal system clock.

The edge detector generates one clk_{T_1}/clk_{T_0} pulse for each positive (CSn2:0 = 7) or negative (CSn2:0 = 6) edge it detects.

Figure 38. T1/T0 Pin Sampling



The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge has been applied to the T1/T0 pin to the counter is updated.

Enabling and disabling of the clock input must be done when T1/T0 has been stable for at least one system clock cycle, otherwise it is a risk that a false Timer/Counter clock pulse is generated.

Each half period of the external clock applied must be longer than one system clock cycle to ensure correct sampling. The external clock must be guaranteed to have less than half the system clock frequency ($f_{\text{ExtClk}} < f_{\text{clk_I/O}}/2$) given a 50/50% duty cycle. Since the edge detector uses

AMEL

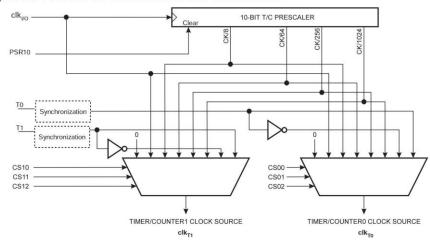
87



sampling, the maximum frequency of an external clock it can detect is half the sampling frequency (Nyquist sampling theorem). However, due to variation of the system clock frequency and duty cycle caused by Oscillator source (crystal, resonator, and capacitors) tolerances, it is recommended that maximum frequency of an external clock source is less than $f_{\text{clk I/O}}/2.5$.

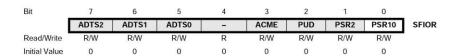
An external clock source can not be prescaled.

Figure 39. Prescaler for Timer/Counter0 and Timer/Counter1⁽¹⁾



Note: 1. The synchronization logic on the input pins (T1/T0) is shown in Figure 38.

Special Function IO Register – SFIOR



• Bit 0 - PSR10: Prescaler Reset Timer/Counter1 and Timer/Counter0

When this bit is written to one, the Timer/Counter1 and Timer/Counter0 prescaler will be reset. The bit will be cleared by hardware after the operation is performed. Writing a zero to this bit will have no effect. Note that Timer/Counter1 and Timer/Counter0 share the same prescaler and a reset of this prescaler will affect both timers. This bit will always be read as zero.





16-bit Timer/Counter1

The 16-bit Timer/Counter unit allows accurate program execution timing (event management), wave generation, and signal timing measurement. The main features are:

- True 16-bit Design (that is, allows 16-bit PWM)
- Two Independent Output Compare Units
- · Double Buffered Output Compare Registers
- · One Input Capture Unit
- · Input Capture Noise Canceler
- · Clear Timer on Compare Match (Auto Reload)
- · Glitch-free, Phase Correct Pulse Width Modulator (PWM)
- Variable PWM Period
- Frequency Generator
- External Event Counter
- · Four Independent Interrupt Sources (TOV1, OCF1A, OCF1B, and ICF1)

Overview

Most register and bit references in this section are written in general form. A lower case "n" replaces the Timer/Counter number, and a lower case "x" replaces the output compare unit. However, when using the register or bit defines in a program, the precise form must be used (that is, TCNT1 for accessing Timer/Counter1 counter value and so on).

A simplified block diagram of the 16-bit Timer/Counter is shown in Figure 40. For the actual placement of I/O pins, refer to Figure 1 on page 2. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device specific I/O Register and bit locations are listed in the "16-bit Timer/Counter Register Description" on page 110.





Count TOVn (Int.Req.) Clear Control Logic Clock Select Direction Edge Tn Detector BOTTOM TOP (From Prescale TCNTn = 0OCnA (Int.Req.) Waveform OCnA **OCRnA** OCnB (Int.Req.) TOP DATABUS Values Waveform OCnB Generation **OCRnB** (From Analog Comparator Ouput) ►ICFn (Int.Req.) Edge ICRn ICPn TCCRnA TCCRnB

Figure 40. 16-bit Timer/Counter Block Diagram⁽¹⁾

 Refer to Figure 1 on page 2, Table 25 on page 58, and Table 31 on page 63 for Timer/Counter1 pin placement and description.

Registers

The Timer/Counter (TCNT1), Output Compare Registers (OCR1A/B), and Input Capture Register (ICR1) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in the section "Accessing 16-bit Registers" on page 92. The Timer/Counter Control Registers (TCCR1A/B) are 8-bit registers and have no CPU access restrictions. Interrupt requests (abbreviated to Int.Req. in the figure) signals are all visible in the Timer Interrupt Flag Register (TIFR). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK). TIFR and TIMSK are not shown in the figure since these registers are shared by other timer units.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the T1 pin. The Clock Select logic block controls which clock source and edge the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the clock select logic is referred to as the timer clock (clk_{T_1}).

The double buffered Output Compare Registers (OCR1A/B) are compared with the Timer/Counter value at all time. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pin (OC1A/B). See "Output Compare Units" on page 98. The compare match event will also set the Compare Match Flag (OCF1A/B) which can be used to generate an output compare interrupt request.





91

The Input Capture Register can capture the Timer/Counter value at a given external (edge triggered) event on either the Input Capture Pin (ICP1) or on the Analog Comparator pins (See "Analog Comparator" on page 201.) The Input Capture unit includes a digital filtering unit (Noise Canceler) for reducing the chance of capturing noise spikes.

The TOP value, or maximum Timer/Counter value, can in some modes of operation be defined by either the OCR1A Register, the ICR1 Register, or by a set of fixed values. When using OCR1A as TOP value in a PWM mode, the OCR1A Register can not be used for generating a PWM output. However, the TOP value will in this case be double buffered allowing the TOP value to be changed in run time. If a fixed TOP value is required, the ICR1 Register can be used as an alternative, freeing the OCR1A to be used as PWM output.

Definitions

The following definitions are used extensively throughout the document:

Table 43. Definitions

воттом	The counter reaches the BOTTOM when it becomes 0x0000.
MAX	The counter reaches its MAXimum when it becomes 0xFFFF (decimal 65535).
ТОР	The counter reaches the <i>TOP</i> when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be one of the fixed values: 0x00FF, 0x01FF, or 0x03FF, or to the value stored in the OCR1A or ICR1 Register. The assignment is dependent of the mode of operation.

Compatibility

The 16-bit Timer/Counter has been updated and improved from previous versions of the 16-bit AVR Timer/Counter. This 16-bit Timer/Counter is fully compatible with the earlier version regarding:

- All 16-bit Timer/Counter related I/O Register address locations, including Timer Interrupt Registers.
- Bit locations inside all 16-bit Timer/Counter Registers, including Timer Interrupt Registers.
- Interrupt Vectors.

The following control bits have changed name, but have same functionality and register location:

- PWM10 is changed to WGM10.
- PWM11 is changed to WGM11.
- CTC1 is changed to WGM12.

The following bits are added to the 16-bit Timer/Counter Control Registers:

- FOC1A and FOC1B are added to TCCR1A.
- WGM13 is added to TCCR1B.

The 16-bit Timer/Counter has improvements that will affect the compatibility in some special cases.





Accessing 16-bit Registers

The TCNT1, OCR1A/B, and ICR1 are 16-bit registers that can be accessed by the AVR CPU via the 8-bit data bus. The 16-bit register must be byte accessed using two read or write operations. Each 16-bit timer has a single 8-bit register for temporary storing of the High byte of the 16-bit access. The same temporary register is shared between all 16-bit registers within each 16-bit timer. Accessing the Low byte triggers the 16-bit read or write operation. When the Low byte of a 16-bit register is written by the CPU, the High byte stored in the temporary register, and the Low byte of a 16-bit register is read by the CPU, the High byte of the 16-bit register is copied into the temporary register is read by the CPU, the High byte of the 16-bit register is copied into the temporary register in the same clock cycle as the Low byte is read.

Not all 16-bit accesses uses the temporary register for the High byte. Reading the OCR1A/B 16-bit registers does not involve using the temporary register.

To do a 16-bit write, the High byte must be written before the Low byte. For a 16-bit read, the Low byte must be read before the High byte.

The following code examples show how to access the 16-bit Timer Registers assuming that no interrupts updates the temporary register. The same principle can be used directly for accessing the OCR1A/B and ICR1 Registers. Note that when using "C", the compiler handles the 16-bit access.

```
Assembly Code Example<sup>(1)</sup>
      ; Set TCNT1 to 0x01FF
     ldi r17,0x01
     ldi r16,0xFF
      out TCNT1H, r17
      out TCNT1L, r16
      ; Read TCNT1 into r17:r16
      in r16,TCNT1L
      in r17,TCNT1H
C Code Example<sup>(1)</sup>
      unsigned int i;
      /* Set TCNT1 to 0x01FF */
      TCNT1 = 0x1FF;
      /* Read TCNT1 into i */
      i = TCNT1;
      . . .
```

Note: 1. See "About Code Examples" on page 7.

The assembly code example returns the TCNT1 value in the r17:r16 register pair.

It is important to notice that accessing 16-bit registers are atomic operations. If an interrupt occurs between the two instructions accessing the 16-bit register, and the interrupt code updates the temporary register by accessing the same or any other of the 16-bit Timer Registers, then the result of the access outside the interrupt will be corrupted. Therefore, when both the main code and the interrupt code update the temporary register, the main code must disable the interrupts during the 16-bit access.





The following code examples show how to do an atomic read of the TCNT1 Register contents. Reading any of the OCR1A/B or ICR1 Registers can be done by using the same principle.

```
Assembly Code Example<sup>(1)</sup>
   TIM16 ReadTCNT1:
     ; Save global interrupt flag
     in r18,SREG
     ; Disable interrupts
     cli
     ; Read TCNT1 into r17:r16
     in r16,TCNT1L
     in r17,TCNT1H
     ; Restore global interrupt flag
     out SREG, r18
     ret
C Code Example<sup>(1)</sup>
   unsigned int TIM16_ReadTCNT1( void )
     unsigned char sreg;
     unsigned int i;
     /* Save global interrupt flag */
     sreg = SREG;
     /* Disable interrupts */
     _CLI();
     /* Read TCNT1 into i */
     i = TCNT1;
     /* Restore global interrupt flag */
     SREG = sreg;
     return i;
```

Note: 1. See "About Code Examples" on page 7.

The assembly code example returns the TCNT1 value in the r17:r16 register pair.





The following code examples show how to do an atomic write of the TCNT1 Register contents. Writing any of the OCR1A/B or ICR1 Registers can be done by using the same principle.

```
Assembly Code Example(1)
   TIM16 WriteTCNT1:
     ; Save global interrupt flag
     in r18,SREG
     ; Disable interrupts
     cli
     ; Set TCNT1 to r17:r16
     out TCNT1H, r17
     out TCNT1L, r16
     ; Restore global interrupt flag
     out SREG, r18
     ret
C Code Example<sup>(1)</sup>
   void TIM16_WriteTCNT1 ( unsigned int i )
     unsigned char sreg;
     unsigned int i;
     /* Save global interrupt flag */
     sreg = SREG;
     /* Disable interrupts */
     _CLI();
     /* Set TCNT1 to i */
     TCNT1 = i;
     /* Restore global interrupt flag */
     SREG = sreg;
```

Note: 1. See "About Code Examples" on page 7.

The assembly code example requires that the r17:r16 register pair contains the value to be written to TCNT1.

Reusing the Temporary High Byte Register If writing to more than one 16-bit register where the High byte is the same for all registers written, then the High byte only needs to be written once. However, note that the same rule of atomic operation described previously also applies in this case.

Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the *Clock Select* (CS12:0) bits located in the *Timer/Counter Control Register B* (TCCR1B). For details on clock sources and prescaler, see "Timer/Counter0 and Timer/Counter1 Prescalers" on page 87.

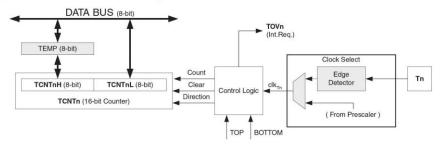




Counter Unit

The main part of the 16-bit Timer/Counter is the programmable 16-bit bi-directional counter unit. Figure 41 shows a block diagram of the counter and its surroundings.

Figure 41. Counter Unit Block Diagram



Signal description (internal signals):

Count Increment or decrement TCNT1 by 1.

Direction Select between increment and decrement.

Clear TCNT1 (set all bits to zero).

clk_{T1} Timer/Counter clock.

TOP Signalize that TCNT1 has reached maximum value.

BOTTOM Signalize that TCNT1 has reached minimum value (zero).

The 16-bit counter is mapped into two 8-bit I/O memory locations: *Counter High* (TCNT1H) containing the upper eight bits of the counter, and *Counter Low* (TCNT1L) containing the lower 8 bits. The TCNT1H Register can only be indirectly accessed by the CPU. When the CPU does an access to the TCNT1H I/O location, the CPU accesses the High byte temporary register (TEMP). The temporary register is updated with the TCNT1H value when the TCNT1L is read, and TCNT1H is updated with the temporary register value when TCNT1L is written. This allows the CPU to read or write the entire 16-bit counter value within one clock cycle via the 8-bit data bus. It is important to notice that there are special cases of writing to the TCNT1 Register when the counter is counting that will give unpredictable results. The special cases are described in the sections where they are of importance.

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each *timer clock* (clk_{T1}). The clk_{T1} can be generated from an external or internal clock source, selected by the *Clock Select* bits (CS12:0). When no clock source is selected (CS12:0 = 0) the timer is stopped. However, the TCNT1 value can be accessed by the CPU, independent of whether clk_{T1} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the *Waveform Generation Mode* bits (WGM13:0) located in the *Timer/Counter Control Registers* A and B (TCCR1A and TCCR1B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC1x. For more details about advanced counting sequences and waveform generation, see "Modes of Operation" on page 101.

The *Timer/Counter Overflow* (TOV1) Flag is set according to the mode of operation selected by the WGM13:0 bits. TOV1 can be used for generating a CPU interrupt.



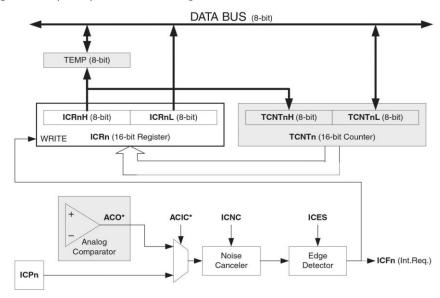


Input Capture Unit

The Timer/Counter incorporates an Input Capture unit that can capture external events and give them a time-stamp indicating time of occurrence. The external signal indicating an event, or multiple events, can be applied via the ICP1 pin or alternatively, via the Analog Comparator unit. The time-stamps can then be used to calculate frequency, duty-cycle, and other features of the signal applied. Alternatively the time-stamps can be used for creating a log of the events.

The Input Capture unit is illustrated by the block diagram shown in Figure 42. The elements of the block diagram that are not directly a part of the Input Capture unit are gray shaded. The small "n" in register and bit names indicates the Timer/Counter number.

Figure 42. Input Capture Unit Block Diagram



When a change of the logic level (an event) occurs on the *Input Capture pin* (ICP1), alternatively on the *Analog Comparator output* (ACO), and this change confirms to the setting of the edge detector, a capture will be triggered. When a capture is triggered, the 16-bit value of the counter (TCNT1) is written to the *Input Capture Register* (ICR1). The *Input Capture Flag* (ICF1) is set at the same system clock as the TCNT1 value is copied into ICR1 Register. If enabled (TICIE1 = 1), the Input Capture Flag generates an Input Capture Interrupt. The ICF1 Flag is automatically cleared when the interrupt is executed. Alternatively the ICF1 Flag can be cleared by software by writing a logical one to its I/O bit location.

Reading the 16-bit value in the *Input Capture Register* (ICR1) is done by first reading the Low byte (ICR1L) and then the High byte (ICR1H). When the Low byte is read the High byte is copied into the High byte temporary register (TEMP). When the CPU reads the ICR1H I/O location it will access the TEMP Register.

The ICR1 Register can only be written when using a Waveform Generation mode that utilizes the ICR1 Register for defining the counter's TOP value. In these cases the *Waveform Generation mode* (WGM13:0) bits must be set before the TOP value can be written to the ICR1 Register. When writing the ICR1 Register the High byte must be written to the ICR1H I/O location before the Low byte is written to ICR1L.





For more information on how to access the 16-bit registers refer to "Accessing 16-bit Registers" on page 92.

Input Capture Pin Source

The main trigger source for the Input Capture unit is the *Input Capture pin* (ICP1). Timer/Counter1 can alternatively use the Analog Comparator output as trigger source for the Input Capture unit. The Analog Comparator is selected as trigger source by setting the *Analog Comparator Input Capture* (ACIC) bit in the *Analog Comparator Control and Status Register* (ACSR). Be aware that changing trigger source can trigger a capture. The Input Capture Flag must therefore be cleared after the change.

Both the *Input Capture pin* (ICP1) and the *Analog Comparator output* (ACO) inputs are sampled using the same technique as for the T1 pin (Figure 38 on page 87). The edge detector is also identical. However, when the noise canceler is enabled, additional logic is inserted before the edge detector, which increases the delay by four system clock cycles. Note that the input of the noise canceler and edge detector is always enabled unless the Timer/Counter is set in a waveform generation mode that uses ICR1 to define TOP.

An Input Capture can be triggered by software by controlling the port of the ICP1 pin.

Noise Canceler

The noise canceler improves noise immunity by using a simple digital filtering scheme. The noise canceler input is monitored over four samples, and all four must be equal for changing the output that in turn is used by the edge detector.

The noise canceler is enabled by setting the *Input Capture Noise Canceler* (ICNC1) bit in *Timer/Counter Control Register B* (TCCR1B). When enabled the noise canceler introduces additional four system clock cycles of delay from a change applied to the input, to the update of the ICR1 Register. The noise canceler uses the system clock and is therefore not affected by the prescaler.

Using the Input Capture Unit

The main challenge when using the Input Capture unit is to assign enough processor capacity for handling the incoming events. The time between two events is critical. If the processor has not read the captured value in the ICR1 Register before the next event occurs, the ICR1 will be overwritten with a new value. In this case the result of the capture will be incorrect.

When using the Input Capture Interrupt, the ICR1 Register should be read as early in the interrupt handler routine as possible. Even though the Input Capture Interrupt has relatively high priority, the maximum interrupt response time is dependent on the maximum number of clock cycles it takes to handle any of the other interrupt requests.

Using the Input Capture unit in any mode of operation when the TOP value (resolution) is actively changed during operation, is not recommended.

Measurement of an external signal's duty cycle requires that the trigger edge is changed after each capture. Changing the edge sensing must be done as early as possible after the ICR1 Register has been read. After a change of the edge, the Input Capture Flag (ICF1) must be cleared by software (writing a logical one to the I/O bit location). For measuring frequency only, the clearing of the ICF1 Flag is not required (if an interrupt handler is used).





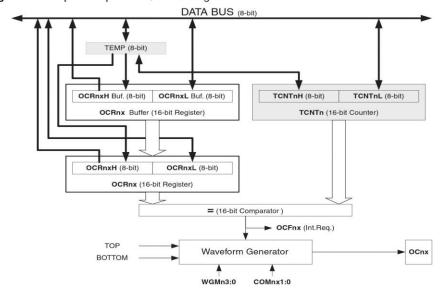
Output Compare Units

The 16-bit comparator continuously compares TCNT1 with the *Output Compare Register* (OCR1x). If TCNT equals OCR1x the comparator signals a match. A match will set the *Output Compare Flag* (OCF1x) at the next timer clock cycle. If enabled (OCIE1x = 1), the Output Compare Flag generates an output compare interrupt. The OCF1x Flag is automatically cleared when the interrupt is executed. Alternatively the OCF1x Flag can be cleared by software by writing a logical one to its I/O bit location. The Waveform Generator uses the match signal to generate an output according to operating mode set by the *Waveform Generation mode* (WGM13:0) bits and *Compare Output mode* (COM1x1:0) bits. The TOP and BOTTOM signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation (See "Modes of Operation" on page 101.)

A special feature of output compare unit A allows it to define the Timer/Counter TOP value (that is, counter resolution). In addition to the counter resolution, the TOP value defines the period time for waveforms generated by the Waveform Generator.

Figure 43 shows a block diagram of the output compare unit. The small "n" in the register and bit names indicates the device number (n = 1 for Timer/Counter1), and the "x" indicates output compare unit (A/B). The elements of the block diagram that are not directly a part of the output compare unit are gray shaded.

Figure 43. Output Compare Unit, Block Diagram



The OCR1x Register is double buffered when using any of the twelve *Pulse Width Modulation* (PWM) modes. For the normal and *Clear Timer on Compare* (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR1x Compare Register to either TOP or BOTTOM of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR1x Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR1x Buffer Register, and if double buffering is disabled the CPU will access the OCR1x directly. The content of the OCR1x (Buffer or Compare) Register is only changed by a write operation (the Timer/Counter does not update this register automatically as the TCNT1 and ICR1 Register). Therefore OCR1x is not read via the High byte





temporary register (TEMP). However, it is a good practice to read the Low byte first as when accessing other 16-bit registers. Writing the OCR1x Registers must be done via the TEMP Register since the compare of all 16 bits is done continuously. The High byte (OCR1xH) has to be written first. When the High byte I/O location is written by the CPU, the TEMP Register will be updated by the value written. Then when the Low byte (OCR1xL) is written to the lower eight bits, the High byte will be copied into the upper 8-bits of either the OCR1x buffer or OCR1x Compare Register in the same system clock cycle.

For more information of how to access the 16-bit registers refer to "Accessing 16-bit Registers" on page 92.

Force Output Compare

In non-PWM Waveform Generation modes, the match output of the comparator can be forced by writing a one to the *Force Output Compare* (FOC1x) bit. Forcing compare match will not set the OCF1x Flag or reload/clear the timer, but the OC1x pin will be updated as if a real compare match had occurred (the COM1x1:0 bits settings define whether the OC1x pin is set, cleared or toggled).

Compare Match Blocking by TCNT1 Write

All CPU writes to the TCNT1 Register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR1x to be initialized to the same value as TCNT1 without triggering an interrupt when the Timer/Counter clock is enabled.

Using the Output Compare Unit

Since writing TCNT1 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT1 when using any of the output compare units, independent of whether the Timer/Counter is running or not. If the value written to TCNT1 equals the OCR1x value, the compare match will be missed, resulting in incorrect waveform generation. Do not write the TCNT1 equal to TOP in PWM modes with variable TOP values. The compare match for the TOP will be ignored and the counter will continue to 0xFFFF. Similarly, do not write the TCNT1 value equal to BOTTOM when the counter is downcounting.

The setup of the OC1x should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC1x value is to use the force output compare (FOC1x) strobe bits in Normal mode. The OC1x Register keeps its value even when changing between waveform generation modes.

Be aware that the COM1x1:0 bits are not double buffered together with the compare value. Changing the COM1x1:0 bits will take effect immediately.

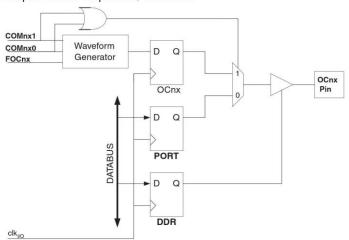




Compare Match Output Unit

The Compare Output mode (COM1x1:0) bits have two functions. The Waveform Generator uses the COM1x1:0 bits for defining the Output Compare (OC1x) state at the next compare match. Secondly the COM1x1:0 bits control the OC1x pin output source. Figure 44 shows a simplified schematic of the logic affected by the COM1x1:0 bit setting. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O Port Control Registers (DDR and PORT) that are affected by the COM1x1:0 bits are shown. When referring to the OC1x state, the reference is for the internal OC1x Register, not the OC1x pin. If a System Reset occur, the OC1x Register is reset to "0".

Figure 44. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the Output Compare (OC1x) from the Waveform Generator if either of the COM1x1:0 bits are set. However, the OC1x pin direction (input or output) is still controlled by the *Data Direction Register* (DDR) for the port pin. The Data Direction Register bit for the OC1x pin (DDR_OC1x) must be set as output before the OC1x value is visible on the pin. The port override function is generally independent of the Waveform Generation mode, but there are some exceptions. Refer to Table 44, Table 45 and Table 46 for details.

The design of the output compare pin logic allows initialization of the OC1x state before the output is enabled. Note that some COM1x1:0 bit settings are reserved for certain modes of operation. See "16-bit Timer/Counter Register Description" on page 110.

The COM1x1:0 bits have no effect on the Input Capture unit.

Compare Output Mode and Waveform Generation The Waveform Generator uses the COM1x1:0 bits differently in normal, CTC, and PWM modes. For all modes, setting the COM1x1:0 = 0 tells the Waveform Generator that no action on the OC1x Register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 44 on page 110. For fast PWM mode refer to Table 45 on page 111, and for phase correct and phase and frequency correct PWM refer to Table 46 on page 111.

A change of the COM1x1:0 bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC1x strobe bits.



100



Modes of Operation

The mode of operation, that is, the behavior of the Timer/Counter and the output compare pins, is defined by the combination of the *Waveform Generation mode* (WGM13:0) and *Compare Output mode* (COM1x1:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM1x1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM1x1:0 bits control whether the output should be set, cleared or toggle at a compare match (See "Compare Match Output Unit" on page 100.)

For detailed timing information refer to "Timer/Counter Timing Diagrams" on page 108.

Normal Mode

The simplest mode of operation is the *Normal* mode (WGM13:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 16-bit value (MAX = 0xFFFF) and then restarts from the BOTTOM (0x0000). In normal operation the *Timer/Counter Overflow Flag* (TOV1) will be set in the same timer clock cycle as the TCNT1 becomes zero. The TOV1 Flag in this case behaves like a 17th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV1 Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

The Input Capture unit is easy to use in Normal mode. However, observe that the maximum interval between the external events must not exceed the resolution of the counter. If the interval between events are too long, the timer overflow interrupt or the prescaler must be used to extend the resolution for the capture unit.

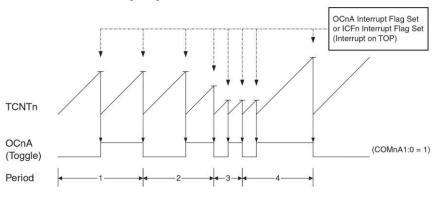
The output compare units can be used to generate interrupts at some given time. Using the output compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM13:0 = 4 or 12), the OCR1A or ICR1 Register are used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT1) matches either the OCR1A (WGM13:0 = 4) or the ICR1 (WGM13:0 = 12). The OCR1A or ICR1 define the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 45. The counter value (TCNT1) increases until a compare match occurs with either OCR1A or ICR1, and then counter (TCNT1) is cleared.

Figure 45. CTC Mode, Timing Diagram



<u>AIMEL</u>

101



An interrupt can be generated at each time the counter value reaches the TOP value by either using the OCF1A or ICF1 Flag according to the register used to define the TOP value. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing the TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR1A or ICR1 is lower than the current value of TCNT1, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. In many cases this feature is not desirable. An alternative will then be to use the fast PWM mode using OCR1A for defining TOP (WGM13:0 = 15) since the OCR1A then will be double buffered.

For generating a waveform output in CTC mode, the OC1A output can be set to toggle its logical level on each compare match by setting the compare output mode bits to toggle mode (COM1A1:0 = 1). The OC1A value will not be visible on the port pin unless the data direction for the pin is set to output (DDR_OC1A = 1). The waveform generated will have a maximum frequency of $f_{\text{OC1A}} = f_{\text{clk_1/O}}/2$ when OCR1A is set to zero (0x0000). The waveform frequency is defined by the following equation:

$$f_{OCnA} = \frac{f_{\text{clk_I/O}}}{2 \cdot N \cdot (1 + OCRnA)}$$

The N variable represents the prescaler factor (1, 8, 64, 256, or 1024).

As for the Normal mode of operation, the TOV1 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x0000.

Fast PWM Mode

The fast Pulse Width Modulation or fast PWM mode (WGM13:0 = 5,6,7,14, or 15) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM options by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x, and set at BOTTOM. In inverting Compare Output mode output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct and phase and frequency correct PWM modes that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), hence reduces total system cost.

The PWM resolution for fast PWM can be fixed to 8-bit, 9-bit, or 10-bit, or defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated by using the following equation:

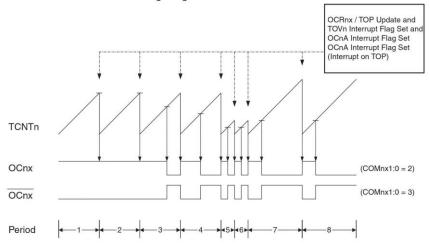
$$R_{FPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

In fast PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM13:0 = 5, 6, or 7), the value in ICR1 (WGM13:0 = 14), or the value in OCR1A (WGM13:0 = 15). The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 46. The figure shows fast PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x Interrupt Flag will be set when a compare match occurs.





Figure 46. Fast PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV1) is set each time the counter reaches TOP. In addition the OC1A or ICF1 Flag is set at the same timer clock cycle as TOV1 is set when either OCR1A or ICR1 is used for defining the TOP value. If one of the interrupts are enabled, the interrupt handler routine can be used for updating the TOP and compare values.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the Compare Registers. If the TOP value is lower than any of the Compare Registers, a compare match will never occur between the TCNT1 and the OCR1x. Note that when using fixed TOP values the unused bits are masked to zero when any of the OCR1x Registers are written.

The procedure for updating ICR1 differs from updating OCR1A when used for defining the TOP value. The ICR1 Register is not double buffered. This means that if ICR1 is changed to a low value when the counter is running with none or a low prescaler value, there is a risk that the new ICR1 value written is lower than the current value of TCNT1. The result will then be that the counter will miss the compare match at the TOP value. The counter will then have to count to the MAX value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. The OCR1A Register however, is double buffered. This feature allows the OCR1A I/O location to be written anytime. When the OCR1A I/O location is written the value written will be put into the OCR1A Buffer Register. The OCR1A Compare Register will then be updated with the value in the Buffer Register at the next timer clock cycle the TCNT1 matches TOP. The update is done at the same timer clock cycle as the TCNT1 is cleared and the TOV1 Flag is set.

Using the ICR1 Register for defining TOP works well when using fixed TOP values. By using ICR1, the OCR1A Register is free to be used for generating a PWM output on OC1A. However, if the base PWM frequency is actively changed (by changing the TOP value), using the OCR1A as TOP is clearly a better choice due to its double buffer feature.

In fast PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to 2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to 3 (See Table 44 on page 110). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x Register at the compare match between OCR1x and TCNT1, and clearing (or setting) the OC1x Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).





The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnxPWM} = \frac{f_{\text{clk_I/O}}}{N \cdot (1 + TOP)}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1x Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR1x is set equal to BOTTOM (0x0000) the output will be a narrow spike for each TOP+1 timer clock cycle. Setting the OCR1x equal to TOP will result in a constant high or low output (depending on the polarity of the output set by the COM1x1:0 bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC1A to toggle its logical level on each compare match (COM1A1:0 = 1). This applies only if OCR1A is used to define the TOP value (WGM13:0 = 15). The waveform generated will have a maximum frequency of $f_{\text{OC1A}} = f_{\text{clk_I/O}}/2$ when OCR1A is set to zero (0x0000). This feature is similar to the OC1A toggle in CTC mode, except the double buffer feature of the output compare unit is enabled in the fast PWM mode.

Phase Correct PWM Mode The phase correct Pulse Width Modulation or phase correct PWM mode (WGM13:0 = 1,2,3,10, or 11) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is, like the phase and frequency correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x while upcounting, and set on the compare match while downcounting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The PWM resolution for the phase correct PWM mode can be fixed to 8-bit, 9-bit, or 10-bit, or defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated by using the following equation:

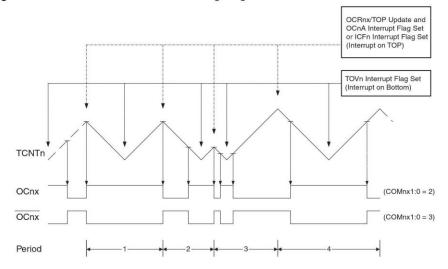
$$R_{PCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

In phase correct PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM13:0 = 1, 2, or 3), the value in ICR1 (WGM13:0 = 10), or the value in OCR1A (WGM13:0 = 11). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 47. The figure shows phase correct PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x Interrupt Flag will be set when a compare match occurs.





Figure 47. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV1) is set each time the counter reaches BOTTOM. When either OCR1A or ICR1 is used for defining the TOP value, the OC1A or ICF1 Flag is set accordingly at the same timer clock cycle as the OCR1x Registers are updated with the double buffer value (at TOP). The Interrupt Flags can be used to generate an interrupt each time the counter reaches the TOP or BOTTOM value.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the Compare Registers. If the TOP value is lower than any of the Compare Registers, a compare match will never occur between the TCNT1 and the OCR1x. Note that when using fixed TOP values, the unused bits are masked to zero when any of the OCR1x Registers are written. As the third period shown in Figure 47 illustrates, changing the TOP actively while the Timer/Counter is running in the phase correct mode can result in an unsymmetrical output. The reason for this can be found in the time of update of the OCR1x Register. Since the OCR1x update occurs at TOP, the PWM period starts and ends at TOP. This implies that the length of the falling slope is determined by the previous TOP value, while the length of the rising slope is determined by the new TOP value. When these two values differ the two slopes of the period will differ in length. The difference in length gives the unsymmetrical result on the output.

It is recommended to use the phase and frequency correct mode instead of the phase correct mode when changing the TOP value while the Timer/Counter is running. When using a static TOP value there are practically no differences between the two modes of operation.

In phase correct PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to 2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to 3 (See Table 44 on page 110). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x Register at the compare match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OC1x Register at compare match between OCR1x and TCNT1 when





the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk_I/O}}{2 \cdot N \cdot TOP}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1x Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR1x is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCR1A is used to define the TOP value (WGM13:0 = 11) and COM1A1:0 = 1, the OC1A output will togqle with a 50% duty cycle.

Phase and Frequency Correct PWM Mode

The phase and frequency correct Pulse Width Modulation, or phase and frequency correct PWM mode (WGM13:0 = 8 or 9) provides a high resolution phase and frequency correct PWM waveform generation option. The phase and frequency correct PWM mode is, like the phase correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x while upcounting, and set on the compare match while downcounting. In inverting Compare Output mode, the operation is inverted. The dual-slope operation gives a lower maximum operation frequency compared to the single-slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The main difference between the phase correct, and the phase and frequency correct PWM mode is the time the OCR1x Register is updated by the OCR1x Buffer Register, (see Figure 47 and Figure 48).

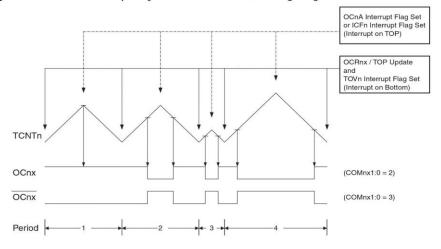
The PWM resolution for the phase and frequency correct PWM mode can be defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated using the following equation:

$$R_{PFCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

In phase and frequency correct PWM mode the counter is incremented until the counter value matches either the value in ICR1 (WGM13:0 = 8), or the value in OCR1A (WGM13:0 = 9). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct and frequency correct PWM mode is shown on Figure 48. The figure shows phase and frequency correct PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x Interrupt Flag will be set when a compare match occurs.



Figure 48. Phase and Frequency Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV1) is set at the same timer clock cycle as the OCR1x Registers are updated with the double buffer value (at BOTTOM). When either OCR1A or ICR1 is used for defining the TOP value, the OC1A or ICF1 Flag set when TCNT1 has reached TOP. The Interrupt Flags can then be used to generate an interrupt each time the counter reaches the TOP or BOTTOM value.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the Compare Registers. If the TOP value is lower than any of the Compare Registers, a compare match will never occur between the TCNT1 and the OCR1x.

As Figure 48 shows the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Since the OCR1x Registers are updated at BOTTOM, the length of the rising and the falling slopes will always be equal. This gives symmetrical output pulses and is therefore frequency correct.

Using the ICR1 Register for defining TOP works well when using fixed TOP values. By using ICR1, the OCR1A Register is free to be used for generating a PWM output on OC1A. However, if the base PWM frequency is actively changed by changing the TOP value, using the OCR1A as TOP is clearly a better choice due to its double buffer feature.

In phase and frequency correct PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to 2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to 3 (See Table on page 111). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x Register at the compare match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OC1x Register at compare match between OCR1x and TCNT1 when the counter decrements. The PWM frequency for the output when using phase and frequency correct PWM can be calculated by the following equation:

$$f_{OCnxPFCPWM} = \frac{f_{clk_l/O}}{2 \cdot N \cdot TOP}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1x Register represents special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR1x is set equal to BOTTOM the



107

2466T-AVR-07/10



output will be continuously low and if set equal to TOP the output will be set to high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCR1A is used to define the TOP value (WGM13:0 = 9) and COM1A1:0 = 1, the OC1A output will toggle with a 50% duty cycle.

Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T1}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set, and when the OCR1x Register is updated with the OCR1x buffer value (only for modes utilizing double buffering). Figure 49 shows a timing diagram for the setting of OCF1x.

Figure 49. Timer/Counter Timing Diagram, Setting of OCF1x, No Prescaling

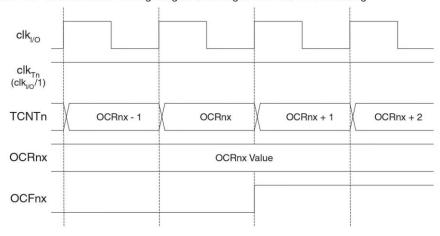


Figure 50 shows the same timing data, but with the prescaler enabled.

Figure 50. Timer/Counter Timing Diagram, Setting of OCF1x, with Prescaler ($f_{clk_l/O}/8$)

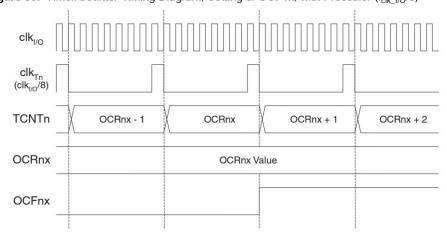


Figure 51 shows the count sequence close to TOP in various modes. When using phase and frequency correct PWM mode the OCR1x Register is updated at BOTTOM. The timing diagrams



108

2466T-AVR-07/10



will be the same, but TOP should be replaced by BOTTOM, TOP-1 by BOTTOM+1 and so on. The same renaming applies for modes that set the TOV1 Flag at BOTTOM.

Figure 51. Timer/Counter Timing Diagram, no Prescaling

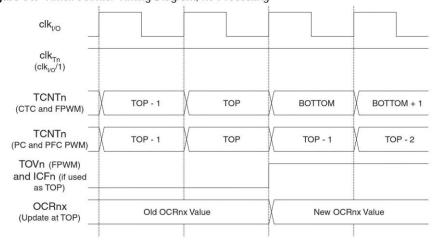
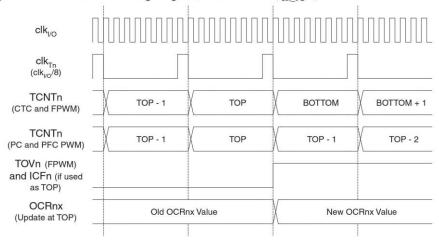


Figure 52 shows the same timing data, but with the prescaler enabled.

Figure 52. Timer/Counter Timing Diagram, with Prescaler ($f_{clk_I/O}/8$)







16-bit Timer/Counter Register Description

Timer/Counter1 Control Register A – TCCR1A

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W	-
Initial Value	0	0	0	0	0	0	0	0	

- Bit 7:6 COM1A1:0: Compare Output Mode for Channel A
- Bit 5:4 COM1B1:0: Compare Output Mode for Channel B

The COM1A1:0 and COM1B1:0 control the Output Compare pins (OC1A and OC1B respectively) behavior. If one or both of the COM1A1:0 bits are written to one, the OC1A output overrides the normal port functionality of the I/O pin it is connected to. If one or both of the COM1B1:0 bit are written to one, the OC1B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the *Data Direction Register* (DDR) bit corresponding to the OC1A or OC1B pin must be set in order to enable the output driver.

When the OC1A or OC1B is connected to the pin, the function of the COM1x1:0 bits is dependent of the WGM13:0 bits setting. Table 44 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to a normal or a CTC mode (non-PWM).

Table 44. Compare Output Mode, non-PWM

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	Toggle OC1A/OC1B on compare match
1	0	Clear OC1A/OC1B on compare match (Set output to low level)
1	1	Set OC1A/OC1B on compare match (Set output to high level)

Table 45 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to the fast PWM mode.





Table 45. Compare Output Mode, Fast PWM(1)

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 15: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM13:0 settings, normal port operation, OCnA/OCnB disconnected.
1	0	Clear OC1A/OC1B on compare match, set OC1A/OC1B at BOTTOM, (non-inverting mode)
1	1	Set OC1A/OC1B on compare match, clear OC1A/OC1B at BOTTOM, (inverting mode)

Note: 1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. In this case the compare match is ignored, but the set or clear is done at BOTTOM. See "Fast PWM Mode" on page 102. for more details.

Table 46 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to the phase correct or the phase and frequency correct, PWM mode.

Table 46. Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM (1)

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 9 or 14: Toggle OCnA on Compare Match, OCnB disconnected (normal port operation). For all other WGM13:0 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on compare match when up-counting. Set OC1A/OC1B on compare match when downcounting.
1	1	Set OC1A/OC1B on compare match when up- counting. Clear OC1A/OC1B on compare match when downcounting.

Note: 1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. See "Phase Correct PWM Mode" on page 104. for more details.

- · Bit 3 FOC1A: Force Output Compare for Channel A
- · Bit 2 FOC1B: Force Output Compare for Channel B

The FOC1A/FOC1B bits are only active when the WGM13:0 bits specifies a non-PWM mode. However, for ensuring compatibility with future devices, these bits must be set to zero when TCCR1A is written when operating in a PWM mode. When writing a logical one to the FOC1A/FOC1B bit, an immediate compare match is forced on the Waveform Generation unit. The OC1A/OC1B output is changed according to its COM1x1:0 bits setting. Note that the FOC1A/FOC1B bits are implemented as strobes. Therefore it is the value present in the COM1x1:0 bits that determine the effect of the forced compare.





A FOC1A/FOC1B strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare match (CTC) mode using OCR1A as TOP.

The FOC1A/FOC1B bits are always read as zero.

· Bit 1:0 - WGM11:0: Waveform Generation Mode

Combined with the WGM13:2 bits found in the TCCR1B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 47. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes. (See "Modes of Operation" on page 101.)

Table 47. Waveform Generation Mode Bit Description⁽¹⁾

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	тор	Update of OCR1X	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	воттом
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	воттом
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	воттом
4	0	1	0	0	стс	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	воттом	ТОР
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	воттом	ТОР
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	воттом	ТОР
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	воттом	воттом
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	воттом	воттом
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	воттом
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	воттом
12	1	1	0	0	стс	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	-	_	_
14	1	1	1	0	Fast PWM	ICR1	воттом	ТОР
15	1	1	1	1	Fast PWM	OCR1A	воттом	ТОР

Note: 1. The CTC1 and PWM11:0 bit definition names are obsolete. Use the WGM12:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.





Timer/Counter1 Control Register B – TCCR1B

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	_	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	•
Initial Value	0	0	0	0	0	0	0	0	

· Bit 7 - ICNC1: Input Capture Noise Canceler

Setting this bit (to one) activates the Input Capture Noise Canceler. When the Noise Canceler is activated, the input from the Input Capture Pin (ICP1) is filtered. The filter function requires four successive equal valued samples of the ICP1 pin for changing its output. The Input Capture is therefore delayed by four Oscillator cycles when the Noise Canceler is enabled.

• Bit 6 - ICES1: Input Capture Edge Select

This bit selects which edge on the Input Capture Pin (ICP1) that is used to trigger a capture event. When the ICES1 bit is written to zero, a falling (negative) edge is used as trigger, and when the ICES1 bit is written to one, a rising (positive) edge will trigger the capture.

When a capture is triggered according to the ICES1 setting, the counter value is copied into the Input Capture Register (ICR1). The event will also set the Input Capture Flag (ICF1), and this can be used to cause an Input Capture Interrupt, if this interrupt is enabled.

When the ICR1 is used as TOP value (see description of the WGM13:0 bits located in the TCCR1A and the TCCR1B Register), the ICP1 is disconnected and consequently the Input Capture function is disabled.

· Bit 5 - Reserved Bit

This bit is reserved for future use. For ensuring compatibility with future devices, this bit must be written to zero when TCCR1B is written.

• Bit 4:3 - WGM13:2: Waveform Generation Mode

See TCCR1A Register description.

Bit 2:0 – CS12:0: Clock Select

The three Clock Select bits select the clock source to be used by the Timer/Counter, see Figure 49 and Figure 50.

Table 48. Clock Select Bit Description

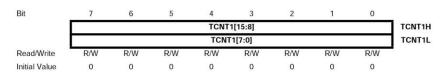
CS12	CS11	CS10	Description	
0	0	0	No clock source (Timer/Counter stopped).	
0	0	1	clk _{I/O} /1 (No prescaling)	
0	1	0	clk _{I/O} /8 (From prescaler)	
0	1	1	clk _{I/O} /64 (From prescaler)	
1	0	0	clk _{I/O} /256 (From prescaler)	
1	0	1	clk _{I/O} /1024 (From prescaler)	
1	1	0	External clock source on T1 pin. Clock on falling edge.	
1	1	1	External clock source on T1 pin. Clock on rising edge.	





If external pin modes are used for the Timer/Counter1, transitions on the T1 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

Timer/Counter1 – TCNT1H and TCNT1L

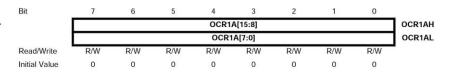


The two *Timer/Counter* I/O locations (TCNT1H and TCNT1L, combined TCNT1) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and Low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See "Accessing 16-bit Registers" on page 92.

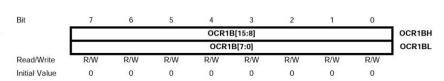
Modifying the counter (TCNT1) while the counter is running introduces a risk of missing a compare match between TCNT1 and one of the OCR1x Registers.

Writing to the TCNT1 Register blocks (removes) the compare match on the following timer clock for all compare units.

Output Compare Register 1 A – OCR1AH and OCR1AL



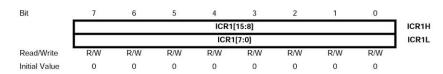
Output Compare Register 1 B – OCR1BH and OCR1BL



The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC1x pin.

The Output Compare Registers are 16-bit in size. To ensure that both the high and Low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See "Accessing 16-bit Registers" on page 92.

Input Capture Register 1 – ICR1H and ICR1L



ATMEL

114

2466T-AVR-07/10



The Input Capture is updated with the counter (TCNT1) value each time an event occurs on the ICP1 pin (or optionally on the analog comparator output for Timer/Counter1). The Input Capture can be used for defining the counter TOP value.

The Input Capture Register is 16-bit in size. To ensure that both the high and Low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See "Accessing 16-bit Registers" on page 92.

Timer/Counter Interrupt Mask Register – TIMSK⁽¹⁾

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	•):
Initial Value	0	0	0	0	0	0	0	0	

Note: 1. This register contains interrupt control bits for several Timer/Counters, but only Timer1 bits are described in this section. The remaining bits are described in their respective timer sections.

· Bit 5 - TICIE1: Timer/Counter1, Input Capture Interrupt Enable

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Input Capture Interrupt is enabled. The corresponding Interrupt Vector (See "Interrupts" on page 45.) is executed when the ICF1 Flag, located in TIFR, is set.

• Bit 4 - OCIE1A: Timer/Counter1, Output Compare A Match Interrupt Enable

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Output Compare A match interrupt is enabled. The corresponding Interrupt Vector (See "Interrupts" on page 45.) is executed when the OCF1A Flag, located in TIFR, is set.

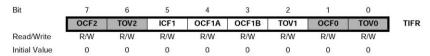
• Bit 3 - OCIE1B: Timer/Counter1, Output Compare B Match Interrupt Enable

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Output Compare B match interrupt is enabled. The corresponding Interrupt Vector (See "Interrupts" on page 45.) is executed when the OCF1B Flag, located in TIFR, is set.

Bit 2 – TOIE1: Timer/Counter1, Overflow Interrupt Enable

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Overflow Interrupt is enabled. The corresponding Interrupt Vector (See "Interrupts" on page 45.) is executed when the TOV1 Flag, located in TIFR, is set.

Timer/Counter Interrupt Flag Register – TIFR



Note: This register contains flag bits for several Timer/Counters, but only Timer1 bits are described in this section. The remaining bits are described in their respective timer sections.

AIMEL



· Bit 5 - ICF1: Timer/Counter1, Input Capture Flag

This flag is set when a capture event occurs on the ICP1 pin. When the Input Capture Register (ICR1) is set by the WGM13:0 to be used as the TOP value, the ICF1 Flag is set when the counter reaches the TOP value.

ICF1 is automatically cleared when the Input Capture Interrupt Vector is executed. Alternatively, ICF1 can be cleared by writing a logic one to its bit location.

· Bit 4 - OCF1A: Timer/Counter1, Output Compare A Match Flag

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register A (OCR1A).

Note that a Forced Output Compare (FOC1A) strobe will not set the OCF1A Flag.

OCF1A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF1A can be cleared by writing a logic one to its bit location.

Bit 3 – OCF1B: Timer/Counter1, Output Compare B Match Flag

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register B (OCR1B).

Note that a forced output compare (FOC1B) strobe will not set the OCF1B Flag.

OCF1B is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCF1B can be cleared by writing a logic one to its bit location.

Bit 2 – TOV1: Timer/Counter1, Overflow Flag

The setting of this flag is dependent of the WGM13:0 bits setting. In normal and CTC modes, the TOV1 Flag is set when the timer overflows. Refer to Table 47 on page 112 for the TOV1 Flag behavior when using another WGM13:0 bit setting.

TOV1 is automatically cleared when the Timer/Counter1 Overflow interrupt vector is executed. Alternatively, TOV1 can be cleared by writing a logic one to its bit location.





8-bit

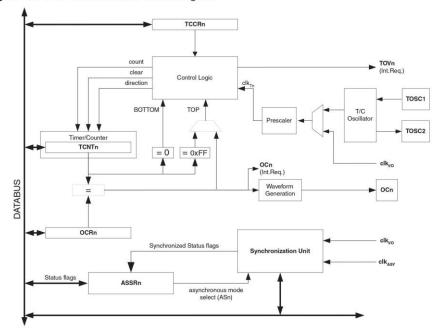
Timer/Counter2 with PWM and Asynchronous Operation Timer/Counter2 is a general purpose, single compare unit, 8-bit Timer/Counter module. The main features are:

- · Single Compare unit Counter
- · Clear Timer on Compare Match (Auto Reload)
- · Glitch-free, Phase Correct Pulse Width Modulator (PWM)
- Frequency Generator
- 10-bit Clock Prescaler
- Overflow and Compare Match Interrupt Sources (TOV2 and OCF2)
- Allows clocking from External 32 kHz Watch Crystal Independent of the I/O Clock

Overview

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 53. For the actual placement of I/O pins, refer to "Pinout ATmega16" on page 2. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the "8-bit Timer/Counter Register Description" on page 128.

Figure 53. 8-bit Timer/Counter Block Diagram



Registers

The Timer/Counter (TCNT2) and Output Compare Register (OCR2) are 8-bit registers. Interrupt request (shorten as Int.Req.) signals are all visible in the Timer Interrupt Flag Register (TIFR). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK). TIFR and TIMSK are not shown in the figure since these registers are shared by other timer units.

The Timer/Counter can be clocked internally, via the prescaler, or asynchronously clocked from the TOSC1/2 pins, as detailed later in this section. The asynchronous operation is controlled by the Asynchronous Status Register (ASSR). The Clock Select logic block controls which clock source the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock (clk_{T2}).





The double buffered Output Compare Register (OCR2) is compared with the Timer/Counter value at all times. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the Output Compare Pin (OC2). See "Output Compare Unit" on page 119. for details. The compare match event will also set the Compare Flag (OCF2) which can be used to generate an output compare interrupt request.

Definitions

Many register and bit references in this document are written in general form. A lower case "n" replaces the Timer/Counter number, in this case 2. However, when using the register or bit defines in a program, the precise form must be used (that is, TCNT2 for accessing Timer/Counter2 counter value and so on). The definitions in Table 49 are also used extensively throughout the document.

Table 49. Definitions

BOTTOM	The counter reaches the BOTTOM when it becomes zero (0x00).
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR2 Register. The assignment is dependent on the mode of operation.

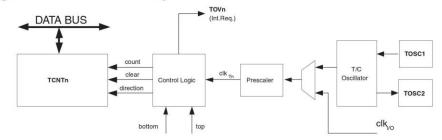
Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal synchronous or an external asynchronous clock source. The clock source clk_{T2} is by default equal to the MCU clock, $clk_{I/O}$. When the AS2 bit in the ASSR Register is written to logic one, the clock source is taken from the Timer/Counter Oscillator connected to TOSC1 and TOSC2. For details on asynchronous operation, see "Asynchronous Status Register – ASSR" on page 131. For details on clock sources and prescaler, see "Timer/Counter Prescaler" on page 134.

Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 54 shows a block diagram of the counter and its surrounding environment.

Figure 54. Counter Unit Block Diagram



Signal description (internal signals):

count Increment or decrement TCNT2 by 1.direction Selects between increment and decrement.

clear TCNT2 (set all bits to zero).

clk_{T2} Timer/Counter clock.

top Signalizes that TCNT2 has reached maximum value.



118

2466T-AVR-07/10



bottom Signalizes that TCNT2 has reached minimum value (zero).

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T2}) . clk_{T2} can be generated from an external or internal clock source, selected by the Clock Select bits (CS22:0). When no clock source is selected (CS22:0 = 0) the timer is stopped. However, the TCNT2 value can be accessed by the CPU, regardless of whether clk_{T2} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

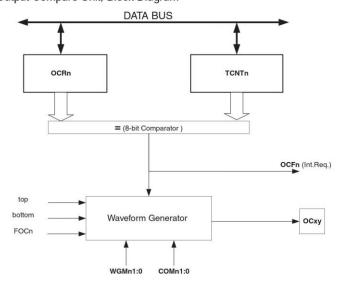
The counting sequence is determined by the setting of the WGM21 and WGM20 bits located in the Timer/Counter Control Register (TCCR2). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare output OC2. For more details about advanced counting sequences and waveform generation, see "Modes of Operation" on page 122.

The Timer/Counter Overflow (TOV2) Flag is set according to the mode of operation selected by the WGM21:0 bits. TOV2 can be used for generating a CPU interrupt.

Output Compare Unit

The 8-bit comparator continuously compares TCNT2 with the Output Compare Register (OCR2). Whenever TCNT2 equals OCR2, the comparator signals a match. A match will set the Output Compare Flag (OCF2) at the next timer clock cycle. If enabled (OCIE2 = 1), the Output Compare Flag generates an output compare interrupt. The OCF2 Flag is automatically cleared when the interrupt is executed. Alternatively, the OCF2 Flag can be cleared by software by writing a logical one to its I/O bit location. The waveform generator uses the match signal to generate an output according to operating mode set by the WGM21:0 bits and Compare Output mode (COM21:0) bits. The max and bottom signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation ("Modes of Operation" on page 122). Figure 55 shows a block diagram of the output compare unit.

Figure 55. Output Compare Unit, Block Diagram



The OCR2 Register is double buffered when using any of the Pulse Width Modulation (PWM) modes. For the normal and Clear Timer on Compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR2 Compare Register





to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR2 Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR2 Buffer Register, and if double buffering is disabled the CPU will access the OCR2 directly.

Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the Force Output Compare (FOC2) bit. Forcing compare match will not set the OCF2 Flag or reload/clear the timer, but the OC2 pin will be updated as if a real compare match had occurred (the COM21:0 bits settings define whether the OC2 pin is set, cleared or toggled).

Compare Match Blocking by TCNT2 Write

All CPU write operations to the TCNT2 Register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR2 to be initialized to the same value as TCNT2 without triggering an interrupt when the Timer/Counter clock is enabled.

Using the Output Compare Unit

Since writing TCNT2 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT2 when using the output compare unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT2 equals the OCR2 value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT2 value equal to BOTTOM when the counter is downcounting.

The setup of the OC2 should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC2 value is to use the Force Output Compare (FOC2) strobe bit in Normal mode. The OC2 Register keeps its value even when changing between Waveform Generation modes.

Be aware that the COM21:0 bits are not double buffered together with the compare value. Changing the COM21:0 bits will take effect immediately.

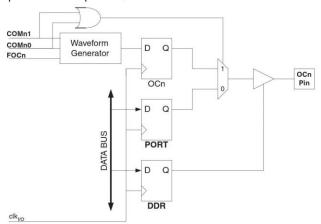




Compare Match Output Unit

The Compare Output mode (COM21:0) bits have two functions. The Waveform Generator uses the COM21:0 bits for defining the Output Compare (OC2) state at the next compare match. Also, the COM21:0 bits control the OC2 pin output source. Figure 56 shows a simplified schematic of the logic affected by the COM21:0 bit setting. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O Port Control Registers (DDR and PORT) that are affected by the COM21:0 bits are shown. When referring to the OC2 state, the reference is for the internal OC2 Register, not the OC2 pin.

Figure 56. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the Output Compare (OC2) from the waveform generator if either of the COM21:0 bits are set. However, the OC2 pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. The Data Direction Register bit for the OC2 pin (DDR_OC2) must be set as output before the OC2 value is visible on the pin. The port override function is independent of the Waveform Generation mode.

The design of the output compare pin logic allows initialization of the OC2 state before the output is enabled. Note that some COM21:0 bit settings are reserved for certain modes of operation. See "8-bit Timer/Counter Register Description" on page 128.

Compare Output Mode and Waveform Generation The waveform generator uses the COM21:0 bits differently in Normal, CTC, and PWM modes. For all modes, setting the COM21:0 = 0 tells the Waveform Generator that no action on the OC2 Register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 51 on page 129. For fast PWM mode, refer to Table 52 on page 129, and for phase correct PWM refer to Table 53 on page 129.

A change of the COM21:0 bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC2 strobe bits.





Modes of Operation

The mode of operation, that is, the behavior of the Timer/Counter and the output compare pins, is defined by the combination of the Waveform Generation mode (WGM21:0) and Compare Output mode (COM21:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM21:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM21:0 bits control whether the output should be set, cleared, or toggled at a compare match (See "Compare Match Output Unit" on page 121.).

For detailed timing information refer to "Timer/Counter Timing Diagrams" on page 126.

Normal Mode

The simplest mode of operation is the Normal mode (WGM21:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter Overflow Flag (TOV2) will be set in the same timer clock cycle as the TCNT2 becomes zero. The TOV2 Flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV2 Flag, the timer resolution can be increased by software. There are no special cases to consider in the normal mode, a new counter value can be written anytime.

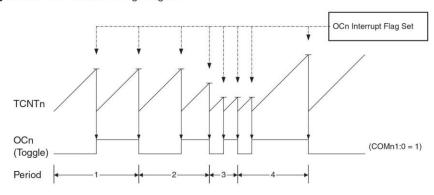
The Output Compare unit can be used to generate interrupts at some given time. Using the output compare to generate waveforms in normal mode is not recommended, since this will occupy too much of the CPU time.

Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM21:0 = 2), the OCR2 Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT2) matches the OCR2. The OCR2 defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 57. The counter value (TCNT2) increases until a compare match occurs between TCNT2 and OCR2, and then counter (TCNT2) is cleared.

Figure 57. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCF2 Flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing the TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR2 is lower than the current

AIMEL

122

2466T-AVR-07/10



value of TCNT2, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFF) and wrap around starting at 0x00 before the compare match can occur.

For generating a waveform output in CTC mode, the OC2 output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COM21:0 = 1). The OC2 value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{\rm OC2} = f_{\rm clk_I/O}/2$ when OCR2 is set to zero (0x00). The waveform frequency is defined by the following equation:

$$f_{OCn} = \frac{f_{\text{clk_I/O}}}{2 \cdot N \cdot (1 + OCRn)}$$

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024).

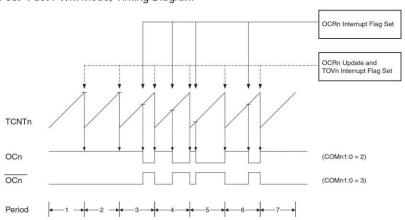
As for the Normal mode of operation, the TOV2 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

Fast PWM Mode

The fast Pulse Width Modulation or fast PWM mode (WGM21:0 = 3) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to MAX then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC2) is cleared on the compare match between TCNT2 and OCR2, and set at BOTTOM. In inverting Compare Output mode, the output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct PWM mode that uses dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

In fast PWM mode, the counter is incremented until the counter value matches the MAX value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 58. The TCNT2 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT2 slopes represent compare matches between OCR2 and TCNT2.

Figure 58. Fast PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV2) is set each time the counter reaches MAX. If the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.



123

2466T-AVR-07/10



In fast PWM mode, the compare unit allows generation of PWM waveforms on the OC2 pin. Setting the COM21:0 bits to 2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM21:0 to 3 (see Table 52 on page 129). The actual OC2 value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC2 Register at the compare match between OCR2 and TCNT2, and clearing (or setting) the OC2 Register at the timer clock cycle the counter is cleared (changes from MAX to BOTTOM).

The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnPWM} = \frac{f_{\text{clk_I/O}}}{N \cdot 256}$$

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024).

The extreme values for the OCR2 Register represent special cases when generating a PWM waveform output in the fast PWM mode. If the OCR2 is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCR2 equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM21:0 bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC2 to toggle its logical level on each compare match (COM21:0 = 1). The waveform generated will have a maximum frequency of $f_{oc2} = f_{clk_I/O}/2$ when OCR2 is set to zero. This feature is similar to the OC2 toggle in CTC mode, except the double buffer feature of the output compare unit is enabled in the fast PWM mode.

Phase Correct PWM Mode

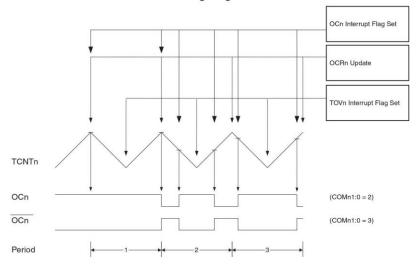
The phase correct PWM mode (WGM21:0 = 1) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to MAX and then from MAX to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC2) is cleared on the compare match between TCNT2 and OCR2 while upcounting, and set on the compare match while downcounting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The PWM resolution for the phase correct PWM mode is fixed to 8 bits. In phase correct PWM mode the counter is incremented until the counter value matches MAX. When the counter reaches MAX, it changes the count direction. The TCNT2 value will be equal to MAX for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 59. The TCNT2 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT2 slopes represent compare matches between OCR2 and TCNT2.





Figure 59. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV2) is set each time the counter reaches BOTTOM. The Interrupt Flag can be used to generate an interrupt each time the counter reaches the BOTTOM value.

In phase correct PWM mode, the compare unit allows generation of PWM waveforms on the OC2 pin. Setting the COM21:0 bits to 2 will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM21:0 to 3 (see Table 53 on page 129). The actual OC2 value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC2 Register at the compare match between OCR2 and TCNT2 when the counter increments, and setting (or clearing) the OC2 Register at compare match between OCR2 and TCNT2 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnPCPWM} = \frac{f_{\text{clk_I/O}}}{N \cdot 510}$$

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024).

The extreme values for the OCR2 Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR2 is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

At the very start of Period 2 in Figure 59 OCn has a transition from high to I ow even though there is no Compare Match. The point of this transition is to guarantee symmetry around BOTTOM. There are two cases that will give transition without Compare Match:

- OCR2A changes its value from Max, like in Figure 59. When the OCR2A value is MAX the
 OCn pin value is the same as the result of a down-counting Compare Match. To ensure
 symmetry around BOTTOM the OCn value at MAX must be correspond the the result of an
 up-counting Compare Match.
- The Timer starts counting from a value higher than the one in OCR2A, and for that reason misses the Compare Match and hence the OCn that would have happened on the way up.





Timer/Counter Timing Diagrams

The following figures show the Timer/Counter in Synchronous mode, and the timer clock (clk_{T2}) is therefore shown as a clock enable signal. In Asynchronous mode, $clk_{I/O}$ should be replaced by the Timer/Counter Oscillator clock. The figures include information on when Interrupt Flags are set. Figure 60 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 60. Timer/Counter Timing Diagram, no Prescaling

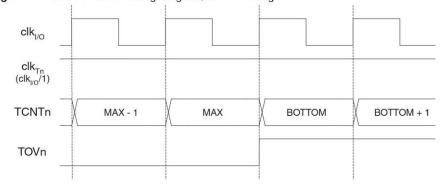


Figure 61 shows the same timing data, but with the prescaler enabled.

Figure 61. Timer/Counter Timing Diagram, with Prescaler ($f_{\text{clk_I/O}}/8$)

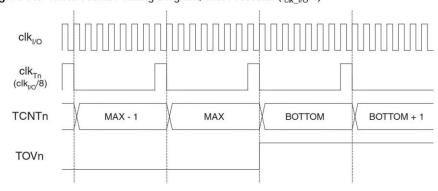


Figure 62 shows the setting of OCF2 in all modes except CTC mode.

AIMEL



Figure 62. Timer/Counter Timing Diagram, Setting of OCF2, with Prescaler ($f_{clk_I/O}/8$)

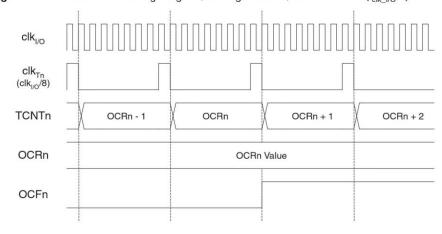
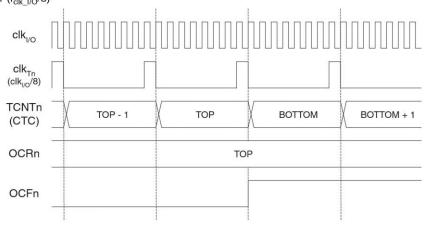


Figure 63 shows the setting of OCF2 and the clearing of TCNT2 in CTC mode.

Figure 63. Timer/Counter Timing Diagram, Clear Timer on Compare Match Mode, with Prescaler ($f_{clk_I/O}/8$)







8-bit Timer/Counter Register Description

Timer/Counter Control Register – TCCR2

Bit	7	6	5	4	3	2	1	0	
	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	TCCR2
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	•
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 - FOC2: Force Output Compare

The FOC2 bit is only active when the WGM bits specify a non-PWM mode. However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR2 is written when operating in PWM mode. When writing a logical one to the FOC2 bit, an immediate compare match is forced on the waveform generation unit. The OC2 output is changed according to its COM21:0 bits setting. Note that the FOC2 bit is implemented as a strobe. Therefore it is the value present in the COM21:0 bits that determines the effect of the forced compare.

A FOC2 strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR2 as TOP.

The FOC2 bit is always read as zero.

Bit 3, 6 – WGM21:0: Waveform Generation Mode

These bits control the counting sequence of the counter, the source for the maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode, Clear Timer on Compare match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes. See Table 50 and "Modes of Operation" on page 122.

Table 50. Waveform Generation Mode Bit Description⁽¹⁾

Mode	WGM21 (CTC2)	WGM20 (PWM2)	Timer/Counter Mode of Operation	тор	Update of OCR2	TOV2 Flag Set on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	воттом
2	1	0	СТС	OCR2	Immediate	MAX
3	1	1	Fast PWM	0xFF	воттом	MAX

Note: 1. The CTC2 and PWM2 bit definition names are now obsolete. Use the WGM21:0 definitions.

However, the functionality and location of these bits are compatible with previous versions of the times.

• Bit 5:4 - COM21:0: Compare Match Output Mode

These bits control the Output Compare pin (OC2) behavior. If one or both of the COM21:0 bits are set, the OC2 output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to OC2 pin must be set in order to enable the output driver.

AIMEL



When OC2 is connected to the pin, the function of the COM21:0 bits depends on the WGM21:0 bit setting. Table 51 shows the COM21:0 bit functionality when the WGM21:0 bits are set to a normal or CTC mode (non-PWM).

Table 51. Compare Output Mode, non-PWM Mode

COM21	COM20	Description			
0	0 Normal port operation, OC2 disconnected.				
0	1	Toggle OC2 on compare match			
1	0	Clear OC2 on compare match			
1	1	Set OC2 on compare match			

Table 52 shows the COM21:0 bit functionality when the WGM21:0 bits are set to fast PWM mode.

Table 52. Compare Output Mode, Fast PWM Mode⁽¹⁾

COM21	COM20	Description
0	0	Normal port operation, OC2 disconnected.
0	1	Reserved
1	0	Clear OC2 on compare match, set OC2 at BOTTOM, (non-inverting mode)
1	1	Set OC2 on compare match, clear OC2 at BOTTOM, (inverting mode)

Note: 1. A special case occurs when OCR2 equals TOP and COM21 is set. In this case, the compare match is ignored, but the set or clear is done at BOTTOM. See "Fast PWM Mode" on page 123 for more details.

Table 53 shows the COM21:0 bit functionality when the WGM21:0 bits are set to phase correct PWM mode

Table 53. Compare Output Mode, Phase Correct PWM Mode⁽¹⁾

COM21	COM20	Description
0	0	Normal port operation, OC2 disconnected.
0	1	Reserved
1	0	Clear OC2 on compare match when up-counting. Set OC2 on compare match when downcounting.
1	1	Set OC2 on compare match when up-counting. Clear OC2 on compare match when downcounting.

Note: 1. A special case occurs when OCR2 equals TOP and COM21 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See "Phase Correct PWM Mode" on page 124 for more details.





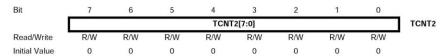
Bit 2:0 – CS22:0: Clock Select

The three Clock Select bits select the clock source to be used by the Timer/Counter, see Table 54.

Table 54. Clock Select Bit Description

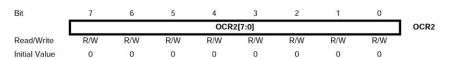
CS22	CS21	CS20	Description	
0	0	0	No clock source (Timer/Counter stopped).	
0	0	1	clk _{T2S} /(No prescaling)	
0	1	0	clk _{T2S} /8 (From prescaler)	
0	1	1	clk _{T2S} /32 (From prescaler)	
1	0	0	clk _{T2S} /64 (From prescaler)	
1	0	1	clk _{T2S} /128 (From prescaler)	
1	1	0	clk _{T2S} /256 (From prescaler)	
1	1	1	clk _{T2S} /1024 (From prescaler)	

Timer/Counter Register – TCNT2



The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT2 Register blocks (removes) the compare match on the following timer clock. Modifying the counter (TCNT2) while the counter is running, introduces a risk of missing a compare match between TCNT2 and the OCR2 Register.

Output Compare Register - OCR2



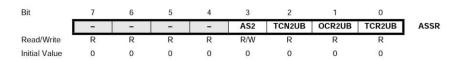
The Output Compare Register contains an 8-bit value that is continuously compared with the counter value (TCNT2). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC2 pin.





Asynchronous Operation of the Timer/Counter

Asynchronous Status Register – ASSR



Bit 3 – AS2: Asynchronous Timer/Counter2

When AS2 is written to zero, Timer/Counter 2 is clocked from the I/O clock, clk_{I/O}. When AS2 is written to one, Timer/Counter2 is clocked from a Crystal Oscillator connected to the Timer Oscillator 1 (TOSC1) pin. When the value of AS2 is changed, the contents of TCNT2, OCR2, and TCCR2 might be corrupted.

Bit 2 – TCN2UB: Timer/Counter2 Update Busy

When Timer/Counter2 operates asynchronously and TCNT2 is written, this bit becomes set. When TCNT2 has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCNT2 is ready to be updated with a new value.

• Bit 1 - OCR2UB: Output Compare Register2 Update Busy

When Timer/Counter2 operates asynchronously and OCR2 is written, this bit becomes set. When OCR2 has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that OCR2 is ready to be updated with a new value.

Bit 0 – TCR2UB: Timer/Counter Control Register2 Update Busy

When Timer/Counter2 operates asynchronously and TCCR2 is written, this bit becomes set. When TCCR2 has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCCR2 is ready to be updated with a new value.

If a write is performed to any of the three Timer/Counter2 Registers while its update busy flag is set, the updated value might get corrupted and cause an unintentional interrupt to occur.

The mechanisms for reading TCNT2, OCR2, and TCCR2 are different. When reading TCNT2, the actual timer value is read. When reading OCR2 or TCCR2, the value in the temporary storage register is read.

Asynchronous Operation of Timer/Counter2 When Timer/Counter2 operates asynchronously, some considerations must be taken.

- Warning: When switching between asynchronous and synchronous clocking of Timer/Counter2, the Timer Registers TCNT2, OCR2, and TCCR2 might be corrupted. A safe procedure for switching clock source is:
 - 1. Disable the Timer/Counter2 interrupts by clearing OCIE2 and TOIE2.
 - 2. Select clock source by setting AS2 as appropriate.
 - 3. Write new values to TCNT2, OCR2, and TCCR2.
 - 4. To switch to asynchronous operation: Wait for TCN2UB, OCR2UB, and TCR2UB.
 - 5. Clear the Timer/Counter2 Interrupt Flags.
 - Enable interrupts, if needed.





- The Oscillator is optimized for use with a 32.768 kHz watch crystal. Applying an external clock to the TOSC1 pin may result in incorrect Timer/Counter2 operation. The CPU main clock frequency must be more than four times the Oscillator frequency.
- When writing to one of the registers TCNT2, OCR2, or TCCR2, the value is transferred to a
 temporary register, and latched after two positive edges on TOSC1. The user should not
 write a new value before the contents of the temporary register have been transferred to its
 destination. Each of the three mentioned registers have their individual temporary register,
 which means for example that writing to TCNT2 does not disturb an OCR2 write in progress.
 To detect that a transfer to the destination register has taken place, the Asynchronous Status
 Register ASSR has been implemented.
- When entering Power-save or Extended Standby mode after having written to TCNT2, OCR2, or TCCR2, the user must wait until the written register has been updated if Timer/Counter2 is used to wake up the device. Otherwise, the MCU will enter sleep mode before the changes are effective. This is particularly important if the Output Compare2 interrupt is used to wake up the device, since the output compare function is disabled during writing to OCR2 or TCNT2. If the write cycle is not finished, and the MCU enters sleep mode before the OCR2UB bit returns to zero, the device will never receive a compare match interrupt, and the MCU will not wake up.
- If Timer/Counter2 is used to wake the device up from Power-save or Extended Standby
 mode, precautions must be taken if the user wants to re-enter one of these modes: The
 interrupt logic needs one TOSC1 cycle to be reset. If the time between wake-up and reentering sleep mode is less than one TOSC1 cycle, the interrupt will not occur, and the
 device will fail to wake up. If the user is in doubt whether the time before re-entering Powersave or Extended Standby mode is sufficient, the following algorithm can be used to ensure
 that one TOSC1 cycle has elapsed:
 - 1. Write a value to TCCR2, TCNT2, or OCR2.
 - 2. Wait until the corresponding Update Busy Flag in ASSR returns to zero.
 - 3. Enter Power-save or Extended Standby mode.
- When the asynchronous operation is selected, the 32.768 kHz Oscillator for Timer/Counter2 is always running, except in Power-down and Standby modes. After a Power-up Reset or wake-up from Power-down or Standby mode, the user should be aware of the fact that this Oscillator might take as long as one second to stabilize. The user is advised to wait for at least one second before using Timer/Counter2 after power-up or wake-up from Power-down or Standby mode. The contents of all Timer/Counter2 Registers must be considered lost after a wake-up from Power-down or Standby mode due to unstable clock signal upon start-up, no matter whether the Oscillator is in use or a clock signal is applied to the TOSC1 pin.
- Description of wake up from Power-save or Extended Standby mode when the timer is clocked asynchronously: When the interrupt condition is met, the wake up process is started on the following cycle of the timer clock, that is, the timer is always advanced by at least one before the processor can read the counter value. After wake-up, the MCU is halted for four cycles, it executes the interrupt routine, and resumes execution from the instruction following SLEEP.
- Reading of the TCNT2 Register shortly after wake-up from Power-save may give an incorrect result. Since TCNT2 is clocked on the asynchronous TOSC clock, reading TCNT2 must be done through a register synchronized to the internal I/O clock domain. Synchronization takes place for every rising TOSC1 edge. When waking up from Power-save mode, and the I/O clock (clk_{I/O}) again becomes active, TCNT2 will read as the previous value (before entering sleep) until the next rising TOSC1 edge. The phase of the TOSC clock after waking up from Power-save mode is essentially unpredictable, as it depends on the wake-up time. The recommended procedure for reading TCNT2 is thus as follows:





- 1. Write any value to either of the registers OCR2 or TCCR2.
- 2. Wait for the corresponding Update Busy Flag to be cleared.
- 3. Read TCNT2.
- During asynchronous operation, the synchronization of the Interrupt Flags for the
 asynchronous timer takes three processor cycles plus one timer cycle. The timer is therefore
 advanced by at least one before the processor can read the timer value causing the setting
 of the Interrupt Flag. The output compare pin is changed on the timer clock and is not
 synchronized to the processor clock.

Timer/Counter Interrupt Mask Register – TIMSK

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	0.
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 - OCIE2: Timer/Counter2 Output Compare Match Interrupt Enable

When the OCIE2 bit is written to one and the I-bit in the Status Register is set (one), the Timer/Counter2 Compare Match interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter2 occurs, that is, when the OCF2 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

• Bit 6 - TOIE2: Timer/Counter2 Overflow Interrupt Enable

When the TOIE2 bit is written to one and the I-bit in the Status Register is set (one), the Timer/Counter2 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter2 occurs, that is, when the TOV2 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

Timer/Counter Interrupt Flag Register – TIFR

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	72
Initial Value	0	0	0	0	0	0	0	0	

Bit 7 – OCF2: Output Compare Flag 2

The OCF2 bit is set (one) when a compare match occurs between the Timer/Counter2 and the data in OCR2 – Output Compare Register2. OCF2 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF2 is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE2 (Timer/Counter2 Compare match Interrupt Enable), and OCF2 are set (one), the Timer/Counter2 Compare match Interrupt is executed.

· Bit 6 - TOV2: Timer/Counter2 Overflow Flag

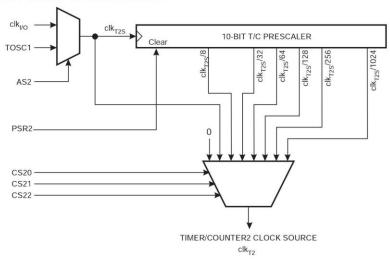
The TOV2 bit is set (one) when an overflow occurs in Timer/Counter2. TOV2 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV2 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE2 (Timer/Counter2 Overflow Interrupt Enable), and TOV2 are set (one), the Timer/Counter2 Overflow interrupt is executed. In PWM mode, this bit is set when Timer/Counter2 changes counting direction at \$00.





Timer/Counter Prescaler

Figure 64. Prescaler for Timer/Counter2



The clock source for Timer/Counter2 is named clk_{T2S} , clk_{T2S} is by default connected to the main system I/O clock clk_{IO} . By setting the AS2 bit in ASSR, Timer/Counter2 is asynchronously clocked from the TOSC1 pin. This enables use of Timer/Counter2 as a Real Time Counter (RTC). When AS2 is set, pins TOSC1 and TOSC2 are disconnected from Port C. A crystal can then be connected between the TOSC1 and TOSC2 pins to serve as an independent clock source for Timer/Counter2. The Oscillator is optimized for use with a 32.768 kHz crystal. Applying an external clock source to TOSC1 is not recommended.

For Timer/Counter2, the possible prescaled selections are: $clk_{T2S}/8$, $clk_{T2S}/32$, $clk_{T2S}/64$, $clk_{T2S}/128$, $clk_{T2S}/256$, and $clk_{T2S}/1024$. Additionally, clk_{T2S} as well as 0 (stop) may be selected. Setting the PSR2 bit in SFIOR resets the prescaler. This allows the user to operate with a predictable prescaler.

Special Function IO Register – SFIOR



Bit 1 – PSR2: Prescaler Reset Timer/Counter2

When this bit is written to one, the Timer/Counter2 prescaler will be reset. The bit will be cleared by hardware after the operation is performed. Writing a zero to this bit will have no effect. This bit will always be read as zero if Timer/Counter2 is clocked by the internal CPU clock. If this bit is written when Timer/Counter2 is operating in asynchronous mode, the bit will remain one until the prescaler has been reset.



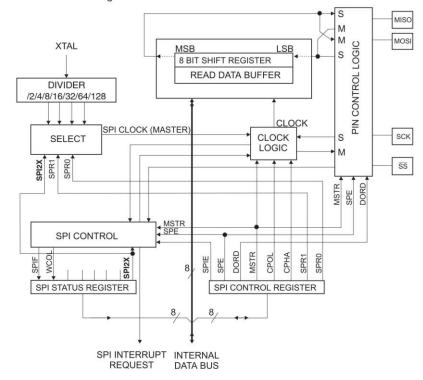


Serial Peripheral Interface – SPI

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATmega16 and peripheral devices or between several AVR devices. The ATmega16 SPI includes the following features:

- · Full-duplex, Three-wire Synchronous Data Transfer
- · Master or Slave Operation
- · LSB First or MSB First Data Transfer
- · Seven Programmable Bit Rates
- · End of Transmission Interrupt Flag
- Write Collision Flag Protection
- · Wake-up from Idle Mode
- Double Speed (CK/2) Master SPI Mode

Figure 65. SPI Block Diagram⁽¹⁾



Note: 1. Refer to Figure 1 on page 2, and Table 25 on page 58 for SPI pin placement.

The interconnection between Master and Slave CPUs with SPI is shown in Figure 66. The system consists of two Shift Registers, and a Master clock generator. The SPI Master initiates the communication cycle when pulling low the Slave Select \overline{SS} pin of the desired Slave. Master and Slave prepare the data to be sent in their respective Shift Registers, and the Master generates the required clock pulses on the SCK line to interchange data. Data is always shifted from Master to Slave on the Master Out – Slave In, MOSI, line, and from Slave to Master on the Master In – Slave Out, MISO, line. After each data packet, the Master will synchronize the Slave by pulling high the Slave Select, \overline{SS} , line.

When configured as a Master, the SPI interface has no automatic control of the \overline{SS} line. This must be handled by user software before communication can start. When this is done, writing a

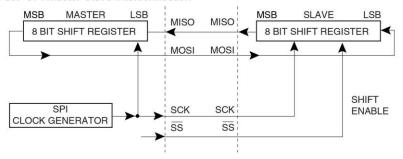




byte to the SPI Data Register starts the SPI clock generator, and the hardware shifts the eight bits into the Slave. After shifting one byte, the SPI clock generator stops, setting the end of Transmission Flag (SPIF). If the SPI Interrupt Enable bit (SPIE) in the SPCR Register is set, an interrupt is requested. The Master may continue to shift the next byte by writing it into SPDR, or signal the end of packet by pulling high the Slave Select, \overline{SS} line. The last incoming byte will be kept in the Buffer Register for later use.

When configured as a Slave, the SPI interface will remain sleeping with MISO tri-stated as long as the \overline{SS} pin is driven high. In this state, software may update the contents of the SPI Data Register, SPDR, but the data will not be shifted out by incoming clock pulses on the SCK pin until the \overline{SS} pin is driven low. As one byte has been completely shifted, the end of Transmission Flag, SPIF is set. If the SPI Interrupt Enable bit, SPIE, in the SPCR Register is set, an interrupt is requested. The Slave may continue to place new data to be sent into SPDR before reading the incoming data. The last incoming byte will be kept in the Buffer Register for later use.

Figure 66. SPI Master-Slave Interconnection



The system is single buffered in the transmit direction and double buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SPI Data Register before the entire shift cycle is completed. When receiving data, however, a received character must be read from the SPI Data Register before the next character has been completely shifted in. Otherwise, the first byte is lost.

In SPI Slave mode, the control logic will sample the incoming signal of the SCK pin. To ensure correct sampling of the clock signal, the minimum low and high periods should be:

Low periods: Longer than 2 CPU clock cycles.

High periods: Longer than 2 CPU clock cycles.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and \overline{SS} pins is overridden according to Table 55 on page 136. For more details on automatic port overrides, refer to "Alternate Port Functions" on page 55.

Table 55. SPI Pin Overrides

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
SS	User Defined	Input





Note: See "Alternate Functions of Port B" on page 58 for a detailed description of how to define the direction of the user defined SPI pins.

The following code examples show how to initialize the SPI as a Master and how to perform a simple transmission. DDR_SPI in the examples must be replaced by the actual Data Direction Register controlling the SPI pins. DD_MOSI, DD_MISO and DD_SCK must be replaced by the actual data direction bits for these pins. For example if MOSI is placed on pin PB5, replace DD_MOSI with DDB5 and DDR_SPI with DDRB.





```
Assembly Code Example<sup>(1)</sup>
   SPI_MasterInit:
     ; Set MOSI and SCK output, all others input
     ldi r17, (1<<DD_MOSI) | (1<<DD_SCK)
     out DDR_SPI,r17
     ; Enable SPI, Master, set clock rate fck/16
     ldi r17, (1<<SPE) | (1<<MSTR) | (1<<SPR0)
     out SPCR, r17
     ret
   SPI_MasterTransmit:
     ; Start transmission of data (r16)
     out SPDR, r16
   Wait_Transmit:
     ; Wait for transmission complete
     sbis SPSR, SPIF
     rjmp Wait_Transmit
C Code Example<sup>(1)</sup>
    void SPI_MasterInit(void)
     /\star Set MOSI and SCK output, all others input \star/
     DDR_SPI = (1<<DD_MOSI) | (1<<DD_SCK);
     /* Enable SPI, Master, set clock rate fck/16 */
     SPCR = (1<<SPE) | (1<<MSTR) | (1<<SPR0);
   void SPI_MasterTransmit(char cData)
     /* Start transmission */
     SPDR = cData;
     /* Wait for transmission complete */
     while(!(SPSR & (1<<SPIF)))</pre>
```

Note: 1. See "About Code Examples" on page 7.





The following code examples show how to initialize the SPI as a Slave and how to perform a simple reception.

```
Assembly Code Example<sup>(1)</sup>
   SPI SlaveInit:
     ; Set MISO output, all others input
     ldi r17, (1<<DD_MISO)
     out DDR_SPI,r17
     ; Enable SPI
     ldi r17, (1<<SPE)
     out SPCR, r17
   SPI SlaveReceive:
     ; Wait for reception complete
     sbis SPSR, SPIF
     rjmp SPI_SlaveReceive
     ; Read received data and return
     in r16,SPDR
     ret
C Code Example<sup>(1)</sup>
   void SPI_SlaveInit(void)
     /\star Set MISO output, all others input \star/
     DDR_SPI = (1<<DD_MISO);
     /* Enable SPI */
     SPCR = (1 < < SPE);
   char SPI_SlaveReceive(void)
     /* Wait for reception complete */
     while(!(SPSR & (1<<SPIF)))</pre>
     /* Return data register */
     return SPDR;
```

Note: 1. See "About Code Examples" on page 7.





——— ATmega16(L)

SS Pin Functionality

Slave Mode

When the SPI is configured as a Slave, the Slave Select (\overline{SS}) pin is always input. When \overline{SS} is held low, the SPI is activated, and MISO becomes an output if configured so by the user. All other pins are inputs. When \overline{SS} is driven high, all pins are inputs except MISO which can be user configured as an output, and the SPI is passive, which means that it will not receive incoming data. Note that the SPI logic will be reset once the \overline{SS} pin is driven high.

The \overline{SS} pin is useful for packet/byte synchronization to keep the Slave Bit Counter synchronous with the Master Clock generator. When the \overline{SS} pin is driven high, the SPI Slave will immediately reset the send and receive logic, and drop any partially received data in the Shift Register.

Master Mode

When the SPI is configured as a Master (MSTR in SPCR is set), the user can determine the direction of the \overline{SS} pin.

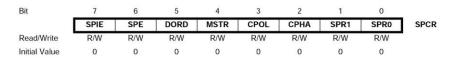
If \overline{SS} is configured as an output, the pin is a general output pin which does not affect the SPI system. Typically, the pin will be driving the \overline{SS} pin of the SPI Slave.

If \overline{SS} is configured as an input, it must be held high to ensure Master SPI operation. If the \overline{SS} pin is driven low by peripheral circuitry when the SPI is configured as a Master with the \overline{SS} pin defined as an input, the SPI system interprets this as another Master selecting the SPI as a Slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:

- The MSTR bit in SPCR is cleared and the SPI system becomes a Slave. As a result of the SPI becoming a Slave, the MOSI and SCK pins become inputs.
- 2. The SPIF Flag in SPSR is set, and if the SPI interrupt is enabled, and the I-bit in SREG is set, the interrupt routine will be executed.

Thus, when interrupt-driven SPI transmission is used in Master mode, and there exists a possibility that \overline{SS} is driven low, the interrupt should always check that the MSTR bit is still set. If the MSTR bit has been cleared by a Slave Select, it must be set by the user to re-enable SPI Master mode.

SPI Control Register – SPCR



Bit 7 – SPIE: SPI Interrupt Enable

This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and the if the global interrupt enable bit in SREG is set.

• Bit 6 - SPE: SPI Enable

When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

· Bit 5 - DORD: Data Order

When the DORD bit is written to one, the LSB of the data word is transmitted first.

When the DORD bit is written to zero, the MSB of the data word is transmitted first.

AIMEL

140

2466T-AVR-07/10



· Bit 4 - MSTR: Master/Slave Select

This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero. If \overline{SS} is configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI Master mode.

· Bit 3 - CPOL: Clock Polarity

When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to Figure 67 and Figure 68 for an example. The CPOL functionality is summarized below:

Table 56. CPOL Functionality

CPOL	Leading Edge	Trailing Edge
0	Rising	Falling
1	Falling	Rising

· Bit 2 - CPHA: Clock Phase

The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to Figure 67 and Figure 68 for an example. The CPHA functionality is summarized below:

Table 57. CPHA Functionality

СРНА	Leading Edge	Trailing Edge
0	Sample	Setup
1	Setup	Sample

· Bits 1, 0 - SPR1, SPR0: SPI Clock Rate Select 1 and 0

These two bits control the SCK rate of the device configured as a Master. SPR1 and SPR0 have no effect on the Slave. The relationship between SCK and the Oscillator Clock frequency $f_{\rm osc}$ is shown in the following table:

Table 58. Relationship Between SCK and the Oscillator Frequency

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	f _{osc} /4
0	0	1	f _{osc} /16
0	1	0	f _{osc} /64
0	1	1	f _{osc} /128
1	0	0	f _{osc} /2
1	0	1	f _{osc} /8
1	1	0	f _{osc} /32
1	1	1	f _{osc} /32 f _{osc} /64





SPI Status Register – SPSR

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	-	-		-	-	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	,
Initial Value	0	0	0	0	0	0	0	0	

· Bit 7 - SPIF: SPI Interrupt Flag

When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If \overline{SS} is an input and is driven low when the SPI is in Master mode, this will also set the SPIF Flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPDR).

· Bit 6 - WCOL: Write COLlision Flag

The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set, and then accessing the SPI Data Register.

· Bit 5..1 - Res: Reserved Bits

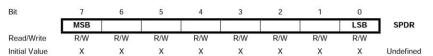
These bits are reserved bits in the ATmega16 and will always read as zero.

· Bit 0 - SPI2X: Double SPI Speed Bit

When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (see Table 58). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at $f_{\rm osc}/4$ or lower.

The SPI interface on the ATmega16 is also used for program memory and EEPROM downloading or uploading. See page 273 for SPI Serial Programming and Verification.

SPI Data Register – SPDR



The SPI Data Register is a read/write register used for data transfer between the Register File and the SPI Shift Register. Writing to the register initiates data transmission. Reading the register causes the Shift Register Receive buffer to be read.





Data Modes

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 67 and Figure 68. Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize. This is clearly seen by summarizing Table 56 and Table 57, as done below:

Table 59. CPOL and CPHA Functionality

	Leading Edge	Trailing Edge	SPI Mode
CPOL = 0, CPHA = 0	Sample (Rising)	Setup (Falling)	0
CPOL = 0, CPHA = 1	Setup (Rising)	Sample (Falling)	1
CPOL = 1, CPHA = 0	Sample (Falling)	Setup (Rising)	2
CPOL = 1, CPHA = 1	Setup (Falling)	Sample (Rising)	3

Figure 67. SPI Transfer Format with CPHA = 0

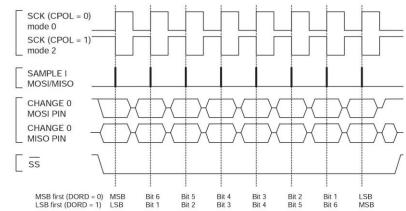
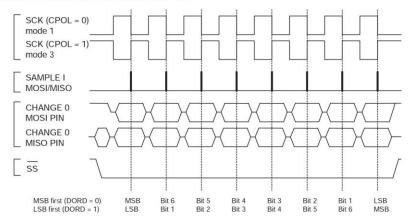


Figure 68. SPI Transfer Format with CPHA = 1







USART

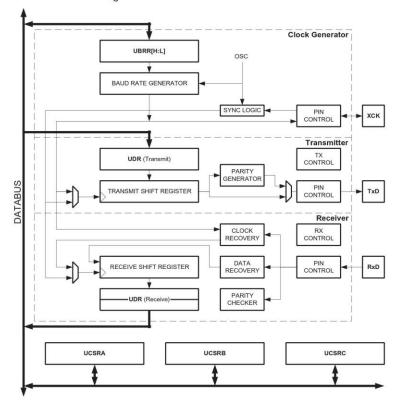
The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a highly flexible serial communication device. The main features are:

- Full Duplex Operation (Independent Serial Receive and Transmit Registers)
- · Asynchronous or Synchronous Operation
- · Master or Slave Clocked Synchronous Operation
- High Resolution Baud Rate Generator
- · Supports Serial Frames with 5, 6, 7, 8, or 9 Data Bits and 1 or 2 Stop Bits
- · Odd or Even Parity Generation and Parity Check Supported by Hardware
- · Data OverRun Detection
- · Framing Error Detection
- Noise Filtering Includes False Start Bit Detection and Digital Low Pass Filter
- · Three Separate Interrupts on TX Complete, TX Data Register Empty, and RX Complete
- · Multi-processor Communication Mode
- Double Speed Asynchronous Communication Mode

Overview

A simplified block diagram of the USART transmitter is shown in Figure 69. CPU accessible I/O Registers and I/O pins are shown in bold.

Figure 69. USART Block Diagram⁽¹⁾



Note: 1. Refer to Figure 1 on page 2, Table 33 on page 65, and Table 27 on page 60 for USART pin placement.

AIMEL

144



The dashed boxes in the block diagram separate the three main parts of the USART (listed from the top): Clock Generator, Transmitter and Receiver. Control Registers are shared by all units. The clock generation logic consists of synchronization logic for external clock input used by synchronous Slave operation, and the baud rate generator. The XCK (Transfer Clock) pin is only used by Synchronous Transfer mode. The Transmitter consists of a single write buffer, a serial Shift Register, parity generator and control logic for handling different serial frame formats. The write buffer allows a continuous transfer of data without any delay between frames. The Receiver is the most complex part of the USART module due to its clock and data recovery units. The receiver units are used for asynchronous data reception. In addition to the recovery units, the receiver includes a parity checker, control logic, a Shift Register and a two level receive buffer (UDR). The receiver supports the same frame formats as the transmitter, and can detect frame error, data overrun and parity errors.

AVR USART vs. AVR UART – Compatibility

The USART is fully compatible with the AVR UART regarding:

- Bit locations inside all USART Registers
- Baud Rate Generation
- Transmitter Operation
- Transmit Buffer Functionality
- Receiver Operation

However, the receive buffering has two improvements that will affect the compatibility in some special cases:

- A second Buffer Register has been added. The two Buffer Registers operate as a circular FIFO buffer. Therefore the UDR must only be read once for each incoming data! More important is the fact that the Error Flags (FE and DOR) and the 9th data bit (RXB8) are buffered with the data in the receive buffer. Therefore the status bits must always be read before the UDR Register is read. Otherwise the error status will be lost since the buffer state is lost.
- The receiver Shift Register can now act as a third buffer level. This is done by allowing the
 received data to remain in the serial Shift Register (see Figure 69) if the Buffer Registers are
 full, until a new start bit is detected. The USART is therefore more resistant to Data OverRun
 (DOR) error conditions.

The following control bits have changed name, but have same functionality and register location:

- CHR9 is changed to UCSZ2
- OR is changed to DOR

Clock Generation

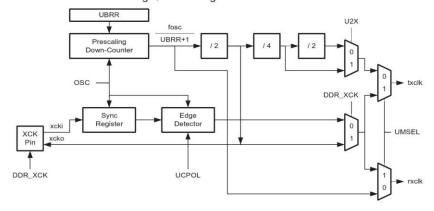
The clock generation logic generates the base clock for the Transmitter and Receiver. The USART supports four modes of clock operation: Normal Asynchronous, Double Speed Asynchronous, Master Synchronous and Slave Synchronous mode. The UMSEL bit in USART Control and Status Register C (UCSRC) selects between asynchronous and synchronous operation. Double Speed (Asynchronous mode only) is controlled by the U2X found in the UCSRA Register. When using Synchronous mode (UMSEL = 1), the Data Direction Register for the XCK pin (DDR_XCK) controls whether the clock source is internal (Master mode) or external (Slave mode). The XCK pin is only active when using Synchronous mode.

Figure 70 shows a block diagram of the clock generation logic.

AIMEL



Figure 70. Clock Generation Logic, Block Diagram



Signal description:

txclk Transmitter clock (Internal Signal).

rxclk Receiver base clock (Internal Signal).

xcki Input from XCK pin (Internal Signal). Used for synchronous Slave operation.

xcko Clock output to XCK pin (Internal Signal). Used for synchronous Master operation.

fosc XTAL pin frequency (System Clock).

Internal Clock Generation – The Baud Rate Generator Internal clock generation is used for the asynchronous and the synchronous Master modes of operation. The description in this section refers to Figure 70.

The USART Baud Rate Register (UBRR) and the down-counter connected to it function as a programmable prescaler or baud rate generator. The down-counter, running at system clock (fosc), is loaded with the UBRR value each time the counter has counted down to zero or when the UBRRL Register is written. A clock is generated each time the counter reaches zero. This clock is the baud rate generator clock output (= fosc/(UBRR+1)). The Transmitter divides the baud rate generator clock output by 2, 8 or 16 depending on mode. The baud rate generator output is used directly by the receiver's clock and data recovery units. However, the recovery units use a state machine that uses 2, 8 or 16 states depending on mode set by the state of the UMSEL, U2X and DDR_XCK bits.

Table 60 contains equations for calculating the baud rate (in bits per second) and for calculating the UBRR value for each mode of operation using an internally generated clock source.



Table 60. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate ⁽¹⁾	Equation for Calculating UBRR Value
Asynchronous Normal Mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(\textit{UBRR} + 1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed Mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(\textit{UBRR} + 1)}$	$UBRR = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master Mode	$BAUD = \frac{f_{OSC}}{2(UBRR + 1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps).

BAUD Baud rate (in bits per second, bps)

f_{OSC} System Oscillator clock frequency

UBRR Contents of the UBRRH and UBRRL Registers, (0 - 4095)

Some examples of UBRR values for some system clock frequencies are found in Table 68 (see page 168).

Double Speed Operation (U2X)

The transfer rate can be doubled by setting the U2X bit in UCSRA. Setting this bit only has effect for the asynchronous operation. Set this bit to zero when using synchronous operation.

Setting this bit will reduce the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication. Note however that the receiver will in this case only use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery, and therefore a more accurate baud rate setting and system clock are required when this mode is used. For the Transmitter, there are no downsides.

External Clock

External clocking is used by the synchronous Slave modes of operation. The description in this section refers to Figure 70 for details.

External clock input from the XCK pin is sampled by a synchronization register to minimize the chance of meta-stability. The output from the synchronization register must then pass through an edge detector before it can be used by the Transmitter and receiver. This process introduces a two CPU clock period delay and therefore the maximum external XCK clock frequency is limited by the following equation:

$$f_{XCK} < \frac{f_{OSC}}{4}$$

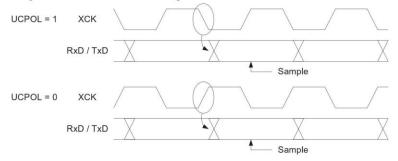
Note that f_{osc} depends on the stability of the system clock source. It is therefore recommended to add some margin to avoid possible loss of data due to frequency variations.



Synchronous Clock Operation

When Synchronous mode is used (UMSEL = 1), the XCK pin will be used as either clock input (Slave) or clock output (Master). The dependency between the clock edges and data sampling or data change is the same. The basic principle is that data input (on RxD) is sampled at the opposite XCK clock edge of the edge the data output (TxD) is changed.

Figure 71. Synchronous Mode XCK Timing.



The UCPOL bit UCRSC selects which XCK clock edge is used for data sampling and which is used for data change. As Figure 71 shows, when UCPOL is zero the data will be changed at rising XCK edge and sampled at falling XCK edge. If UCPOL is set, the data will be changed at falling XCK edge and sampled at rising XCK edge.

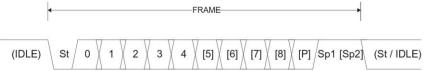
Frame Formats

A serial frame is defined to be one character of data bits with synchronization bits (start and stop bits), and optionally a parity bit for error checking. The USART accepts all 30 combinations of the following as valid frame formats:

- 1 start bit
- 5, 6, 7, 8, or 9 data bits
- · no, even or odd parity bit
- · 1 or 2 stop bits

A frame starts with the start bit followed by the least significant data bit. Then the next data bits, up to a total of nine, are succeeding, ending with the most significant bit. If enabled, the parity bit is inserted after the data bits, before the stop bits. When a complete frame is transmitted, it can be directly followed by a new frame, or the communication line can be set to an idle (high) state. Figure 72 illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

Figure 72. Frame Formats



- St Start bit, always low.
- (n) Data bits (0 to 8).
- P Parity bit. Can be odd or even.
- Sp Stop bit, always high.

AMEL

148



IDLE No transfers on the communication line (RxD or TxD). An IDLE line must be high.

The frame format used by the USART is set by the UCSZ2:0, UPM1:0, and USBS bits in UCSRB and UCSRC. The Receiver and Transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

The USART Character SiZe (UCSZ2:0) bits select the number of data bits in the frame. The USART Parity mode (UPM1:0) bits enable and set the type of parity bit. The selection between one or two stop bits is done by the USART Stop Bit Select (USBS) bit. The receiver ignores the second stop bit. An FE (Frame Error) will therefore only be detected in the cases where the first stop bit is zero.

Parity Bit Calculation

The parity bit is calculated by doing an exclusive-or of all the data bits. If odd parity is used, the result of the exclusive or is inverted. The relation between the parity bit and data bits is as follows::

$$\begin{array}{l} P_{even} = \, d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0 \\ P_{odd} = \, d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1 \end{array}$$

Peven Parity bit using even parity
 Podd Parity bit using odd parity
 Data bit n of the character

If used, the parity bit is located between the last data bit and first stop bit of a serial frame.

USART Initialization

The USART has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting frame format and enabling the Transmitter or the Receiver depending on the usage. For interrupt driven USART operation, the Global Interrupt Flag should be cleared (and interrupts globally disabled) when doing the initialization.

Before doing a re-initialization with changed baud rate or frame format, be sure that there are no ongoing transmissions during the period the registers are changed. The TXC Flag can be used to check that the Transmitter has completed all transfers, and the RXC Flag can be used to check that there are no unread data in the receive buffer. Note that the TXC Flag must be cleared before each transmission (before UDR is written) if it is used for this purpose.

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume asynchronous operation using polling (no interrupts enabled) and a fixed frame format. The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 registers. When the function writes to the UCSRC Register, the URSEL bit (MSB) must be set due to the sharing of I/O location by UBRRH and UCSRC.



```
Assembly Code Example(1)
    USART Init:
     ; Set baud rate
     out UBRRH, r17
     out UBRRL, r16
     ; Enable receiver and transmitter
     ldi r16, (1<<RXEN) | (1<<TXEN)
     out UCSRB, r16
     ; Set frame format: 8data, 2stop bit
     ldi r16, (1<<URSEL) | (1<<USBS) | (3<<UCSZ0)
          UCSRC, r16
     out
     ret
C Code Example<sup>(1)</sup>
    #define FOSC 1843200// Clock Speed
    #define BAUD 9600
    #define MYUBRR FOSC/16/BAUD-1
    void main ( void )
     USART_Init ( MYUBRR );
    void USART_Init( unsigned int ubrr)
     /* Set baud rate */
     UBRRH = (unsigned char) (ubrr>>8);
     UBRRL = (unsigned char)ubrr;
     /* Enable receiver and transmitter */
     UCSRB = (1<<RXEN) | (1<<TXEN);
     /* Set frame format: 8data, 2stop bit */
     UCSRC = (1<<URSEL) | (1<<USBS) | (3<<UCSZ0);
```

Note: 1. See "About Code Examples" on page 7.

More advanced initialization routines can be made that include frame format as parameters, disable interrupts and so on. However, many applications use a fixed setting of the Baud and Control Registers, and for these types of applications the initialization code can be placed directly in the main routine, or be combined with initialization code for other I/O modules.

Data Transmission - The USART Transmitter

The USART Transmitter is enabled by setting the *Transmit Enable* (TXEN) bit in the UCSRB Register. When the Transmitter is enabled, the normal port operation of the TxD pin is overridden by the USART and given the function as the transmitter's serial output. The baud rate, mode of operation and frame format must be set up once before doing any transmissions. If synchronous operation is used, the clock on the XCK pin will be overridden and used as transmission clock.

Sending Frames with 5 to 8 Data Bit

A data transmission is initiated by loading the transmit buffer with the data to be transmitted. The CPU can load the transmit buffer by writing to the UDR I/O location. The buffered data in the transmit buffer will be moved to the Shift Register when the Shift Register is ready to send a new

AIMEL

150



frame. The Shift Register is loaded with new data if it is in idle state (no ongoing transmission) or immediately after the last stop bit of the previous frame is transmitted. When the Shift Register is loaded with new data, it will transfer one complete frame at the rate given by the Baud Register, U2X bit or by XCK depending on mode of operation.

The following code examples show a simple USART transmit function based on polling of the *Data Register Empty* (UDRE) Flag. When using frames with less than eight bits, the most significant bits written to the UDR are ignored. The USART has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in Register R16

```
Assembly Code Example(1)

USART_Transmit:
; Wait for empty transmit buffer
sbis UCSRA,UDRE
rjmp USART_Transmit
; Put data (r16) into buffer, sends the data
out UDR,r16
ret

C Code Example(1)

void USART_Transmit( unsigned char data )
{
    /* Wait for empty transmit buffer */
    while ( !( UCSRA & (1<<UDRE)) )
        ;
    /* Put data into buffer, sends the data */
    UDR = data;
}
```

Note: 1. See "About Code Examples" on page 7.

The function simply waits for the transmit buffer to be empty by checking the UDRE Flag, before loading it with new data to be transmitted. If the Data Register Empty Interrupt is utilized, the interrupt routine writes the data into the buffer.





Sending Frames with 9 Data Bit

If 9-bit characters are used (UCSZ = 7), the ninth bit must be written to the TXB8 bit in UCSRB before the Low byte of the character is written to UDR. The following code examples show a transmit function that handles 9-bit characters. For the assembly code, the data to be sent is assumed to be stored in Registers R17:R16.

```
Assembly Code Example<sup>(1)</sup>
    USART Transmit:
     ; Wait for empty transmit buffer
     sbis UCSRA, UDRE
     rjmp USART_Transmit
      ; Copy 9th bit from r17 to TXB8
     cbi UCSRB, TXB8
     sbrc r17,0
     sbi UCSRB, TXB8
      ; Put LSB data (r16) into buffer, sends the data
     out UDR.r16
     ret
C Code Example<sup>(1)</sup>
    void USART_Transmit( unsigned int data )
      /* Wait for empty transmit buffer */
     while ( !( UCSRA & (1<<UDRE))) )</pre>
      /* Copy 9th bit to TXB8 */
     UCSRB &= ~(1<<TXB8):
     if ( data & 0x0100 )
       UCSRB |= (1<<TXB8);
      /* Put data into buffer, sends the data */
     UDR = data;
```

Note: 1. These transmit functions are written to be general functions. They can be optimized if the contents of the UCSRB is static. (that is, only the TXB8 bit of the UCSRB Register is used after initialization).

The ninth bit can be used for indicating an address frame when using multi processor communication mode or for other protocol handling as for example synchronization.

Transmitter Flags and Interrupts

The USART transmitter has two flags that indicate its state: USART Data Register Empty (UDRE) and Transmit Complete (TXC). Both flags can be used for generating interrupts.

The Data Register Empty (UDRE) Flag indicates whether the transmit buffer is ready to receive new data. This bit is set when the transmit buffer is empty, and cleared when the transmit buffer contains data to be transmitted that has not yet been moved into the Shift Register. For compatibility with future devices, always write this bit to zero when writing the UCSRA Register.

When the Data Register empty Interrupt Enable (UDRIE) bit in UCSRB is written to one, the USART Data Register Empty Interrupt will be executed as long as UDRE is set (provided that global interrupts are enabled). UDRE is cleared by writing UDR. When interrupt-driven data transmission is used, the Data Register Empty Interrupt routine must either write new data to UDR in order to clear UDRE or disable the Data Register empty Interrupt, otherwise a new interrupt will occur once the interrupt routine terminates.



152



The Transmit Complete (TXC) Flag bit is set one when the entire frame in the transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer. The TXC Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC Flag is useful in half-duplex communication interfaces (like the RS485 standard), where a transmitting application must enter receive mode and free the communication bus immediately after completing the transmission.

When the Transmit Compete Interrupt Enable (TXCIE) bit in UCSRB is set, the USART Transmit Complete Interrupt will be executed when the TXC Flag becomes set (provided that global interrupts are enabled). When the transmit complete interrupt is used, the interrupt handling routine does not have to clear the TXC Flag, this is done automatically when the interrupt is executed.

Parity Generator

The parity generator calculates the parity bit for the serial frame data. When parity bit is enabled (UPM1 = 1), the transmitter control logic inserts the parity bit between the last data bit and the first stop bit of the frame that is sent.

Disabling the Transmitter The disabling of the transmitter (setting the TXEN to zero) will not become effective until ongoing and pending transmissions are completed, that is, when the transmit Shift Register and transmit Buffer Register do not contain data to be transmitted. When disabled, the transmitter will no longer override the TxD pin.





Data Reception – The USART Receiver

The USART Receiver is enabled by writing the Receive Enable (RXEN) bit in the UCSRB Register to one. When the receiver is enabled, the normal pin operation of the RxD pin is overridden by the USART and given the function as the receiver's serial input. The baud rate, mode of operation and frame format must be set up once before any serial reception can be done. If synchronous operation is used, the clock on the XCK pin will be used as transfer clock.

Receiving Frames with 5 to 8 Data Bits

The receiver starts data reception when it detects a valid start bit. Each bit that follows the start bit will be sampled at the baud rate or XCK clock, and shifted into the receive Shift Register until the first stop bit of a frame is received. A second stop bit will be ignored by the receiver. When the first stop bit is received, that is, a complete serial frame is present in the receive Shift Register, the contents of the Shift Register will be moved into the receive buffer. The receive buffer can then be read by reading the UDR I/O location.

The following code example shows a simple USART receive function based on polling of the Receive Complete (RXC) Flag. When using frames with less than eight bits the most significant bits of the data read from the UDR will be masked to zero. The USART has to be initialized before the function can be used.

```
Assembly Code Example(1)
    USART Receive:
      ; Wait for data to be received
     sbis UCSRA, RXC
     rjmp USART Receive
      ; Get and return received data from buffer
          r16, UDR
     in
     ret
C Code Example<sup>(1)</sup>
    unsigned char USART Receive ( void )
      /* Wait for data to be received */
     while ( !(UCSRA & (1<<RXC)) )
      /* Get and return received data from buffer */
      return UDR;
    }
```

Note: 1. See "About Code Examples" on page 7.

The function simply waits for data to be present in the receive buffer by checking the RXC Flag, before reading the buffer and returning the value.





Receiving Frames with 9 Databits

If 9 bit characters are used (UCSZ=7) the ninth bit must be read from the RXB8 bit in UCSRB **before** reading the low bits from the UDR. This rule applies to the FE, DOR and PE status Flags as well. Read status from UCSRA, then data from UDR. Reading the UDR I/O location will change the state of the receive buffer FIFO and consequently the TXB8, FE, DOR and PE bits, which all are stored in the FIFO, will change.

The following code example shows a simple USART receive function that handles both 9-bit characters and the status bits.

```
Assembly Code Example<sup>(1)</sup>
   USART Receive:
     ; Wait for data to be received
     sbis UCSRA, RXC
     rjmp USART Receive
     ; Get status and 9th bit, then data from buffer
     in r18, UCSRA
          r17, UCSRB
          r16, UDR
     ; If error, return -1
     andi r18, (1<<FE) | (1<<DOR) | (1<<PE)
     breq USART ReceiveNoError
     ldi r17, HIGH(-1)
     ldi r16, LOW(-1)
   USART ReceiveNoError:
     ; Filter the 9th bit, then return
     lsr r17
     andi r17, 0x01
     ret
C Code Example<sup>(1)</sup>
    unsigned int USART Receive ( void )
     unsigned char status, resh, resl;
     /* Wait for data to be received */
     while ( !(UCSRA & (1<<RXC)) )
      /* Get status and 9th bit, then data */
     /* from buffer */
     status = UCSRA;
     resh = UCSRB:
     resl = UDR;
      /* If error, return -1 */
     if ( status & (1<<FE) | (1<<DOR) | (1<<PE) )</pre>
       return -1;
      /* Filter the 9th bit, then return */
     resh = (resh >> 1) \& 0x01;
      return ((resh << 8) | resl);</pre>
```

Note: 1. See "About Code Examples" on page 7.





The receive function example reads all the I/O Registers into the Register File before any computation is done. This gives an optimal receive buffer utilization since the buffer location read will be free to accept new data as early as possible.

Receive Compete Flag and Interrupt

The USART Receiver has one flag that indicates the receiver state.

The Receive Complete (RXC) Flag indicates if there are unread data present in the receive buffer. This flag is one when unread data exist in the receive buffer, and zero when the receive buffer is empty (that is, does not contain any unread data). If the receiver is disabled (RXEN = 0), the receive buffer will be flushed and consequently the RXC bit will become zero.

When the Receive Complete Interrupt Enable (RXCIE) in UCSRB is set, the USART Receive Complete Interrupt will be executed as long as the RXC Flag is set (provided that global interrupts are enabled). When interrupt-driven data reception is used, the receive complete routine must read the received data from UDR in order to clear the RXC Flag, otherwise a new interrupt will occur once the interrupt routine terminates.

Receiver Error Flags

The USART Receiver has three Error Flags: Frame Error (FE), Data OverRun (DOR) and Parity Error (PE). All can be accessed by reading UCSRA. Common for the Error Flags is that they are located in the receive buffer together with the frame for which they indicate the error status. Due to the buffering of the Error Flags, the UCSRA must be read before the receive buffer (UDR), since reading the UDR I/O location changes the buffer read location. Another equality for the Error Flags is that they can not be altered by software doing a write to the flag location. However, all flags must be set to zero when the UCSRA is written for upward compatibility of future USART implementations. None of the Error Flags can generate interrupts.

The Frame Error (FE) Flag indicates the state of the first stop bit of the next readable frame stored in the receive buffer. The FE Flag is zero when the stop bit was correctly read (as one), and the FE Flag will be one when the stop bit was incorrect (zero). This flag can be used for detecting out-of-sync conditions, detecting break conditions and protocol handling. The FE Flag is not affected by the setting of the USBS bit in UCSRC since the receiver ignores all, except for the first, stop bits. For compatibility with future devices, always set this bit to zero when writing to UCSRA.

The Data OverRun (DOR) Flag indicates data loss due to a receiver buffer full condition. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the receive Shift Register, and a new start bit is detected. If the DOR Flag is set there was one or more serial frame lost between the frame last read from UDR, and the next frame read from UDR. For compatibility with future devices, always write this bit to zero when writing to UCSRA. The DOR Flag is cleared when the frame received was successfully moved from the Shift Register to the receive buffer.

The Parity Error (PE) Flag indicates that the next frame in the receive buffer had a parity error when received. If parity check is not enabled the PE bit will always be read zero. For compatibility with future devices, always set this bit to zero when writing to UCSRA. For more details see "Parity Bit Calculation" on page 149 and "Parity Checker" on page 156.

Parity Checker

The Parity Checker is active when the high USART Parity mode (UPM1) bit is set. Type of parity check to be performed (odd or even) is selected by the UPM0 bit. When enabled, the parity checker calculates the parity of the data bits in incoming frames and compares the result with the parity bit from the serial frame. The result of the check is stored in the receive buffer together with the received data and stop bits. The Parity Error (PE) Flag can then be read by software to check if the frame had a parity error.

The PE bit is set if the next character that can be read from the receive buffer had a parity error when received and the parity checking was enabled at that point (UPM1 = 1). This bit is valid until the receive buffer (UDR) is read.

AIMEL

156



Disabling the Receiver

In contrast to the Transmitter, disabling of the Receiver will be immediate. Data from ongoing receptions will therefore be lost. When disabled (that is, the RXEN is set to zero) the Receiver will no longer override the normal function of the RxD port pin. The receiver buffer FIFO will be flushed when the receiver is disabled. Remaining data in the buffer will be lost

Flushing the Receive Buffer

The receiver buffer FIFO will be flushed when the Receiver is disabled, that is, the buffer will be emptied of its contents. Unread data will be lost. If the buffer has to be flushed during normal operation, due to for instance an error condition, read the UDR I/O location until the RXC Flag is cleared. The following code example shows how to flush the receive buffer.

```
Assembly Code Example(1)

USART_Flush:
sbis UCSRA, RXC
ret
in r16, UDR
rjmp USART_Flush

C Code Example(1)

void USART_Flush( void )
{
 unsigned char dummy;
 while ( UCSRA & (1<<RXC) ) dummy = UDR;
}
```

Note: 1. See "About Code Examples" on page 7.

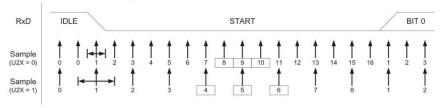
Asynchronous Data Reception

The USART includes a clock recovery and a data recovery unit for handling asynchronous data reception. The clock recovery logic is used for synchronizing the internally generated baud rate clock to the incoming asynchronous serial frames at the RxD pin. The data recovery logic samples and low pass filters each incoming bit, thereby improving the noise immunity of the receiver. The asynchronous reception operational range depends on the accuracy of the internal baud rate clock, the rate of the incoming frames, and the frame size in number of bits.

Asynchronous Clock Recovery

The clock recovery logic synchronizes internal clock to the incoming serial frames. Figure 73 illustrates the sampling process of the start bit of an incoming frame. The sample rate is 16 times the baud rate for Normal mode, and 8 times the baud rate for Double Speed mode. The horizontal arrows illustrate the synchronization variation due to the sampling process. Note the larger time variation when using the double speed mode (U2X = 1) of operation. Samples denoted zero are samples done when the RxD line is idle (that is, no communication activity).

Figure 73. Start Bit Sampling



When the clock recovery logic detects a high (idle) to low (start) transition on the RxD line, the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample as shown in the figure. The clock recovery logic then uses samples 8, 9, and 10 for Normal mode, and samples 4, 5, and 6 for Double Speed mode (indicated with sample numbers inside boxes on the

AIMEL

157

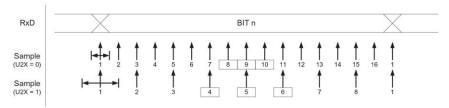


figure), to decide if a valid start bit is received. If two or more of these three samples have logical high levels (the majority wins), the start bit is rejected as a noise spike and the receiver starts looking for the next high to low-transition. If however, a valid start bit is detected, the clock recovery logic is synchronized and the data recovery can begin. The synchronization process is repeated for each start bit.

Asynchronous Data Recovery

When the receiver clock is synchronized to the start bit, the data recovery can begin. The data recovery unit uses a state machine that has 16 states for each bit in normal mode and 8 states for each bit in Double Speed mode. Figure 74 shows the sampling of the data bits and the parity bit. Each of the samples is given a number that is equal to the state of the recovery unit.

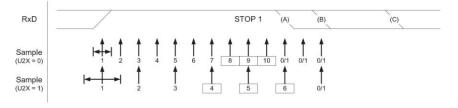
Figure 74. Sampling of Data and Parity Bit



The decision of the logic level of the received bit is taken by doing a majority voting of the logic value to the three samples in the center of the received bit. The center samples are emphasized on the figure by having the sample number inside boxes. The majority voting process is done as follows: If two or all three samples have high levels, the received bit is registered to be a logic 1. If two or all three samples have low levels, the received bit is registered to be a logic 0. This majority voting process acts as a low pass filter for the incoming signal on the RxD pin. The recovery process is then repeated until a complete frame is received. Including the first stop bit. Note that the receiver only uses the first stop bit of a frame.

Figure 75 shows the sampling of the stop bit and the earliest possible beginning of the start bit of the next frame.

Figure 75. Stop Bit Sampling and Next Start Bit Sampling



The same majority voting is done to the stop bit as done for the other bits in the frame. If the stop bit is registered to have a logic 0 value, the Frame Error (FE) Flag will be set.

A new high to low transition indicating the start bit of a new frame can come right after the last of the bits used for majority voting. For Normal Speed mode, the first low level sample can be at point marked (A) in Figure 75. For Double Speed mode the first low level must be delayed to (B). (C) marks a stop bit of full length. The early start bit detection influences the operational range of the receiver.

AIMEL

158



Asynchronous Operational Range

The operational range of the receiver is dependent on the mismatch between the received bit rate and the internally generated baud rate. If the Transmitter is sending frames at too fast or too slow bit rates, or the internally generated baud rate of the receiver does not have a similar (see Table 61) base frequency, the receiver will not be able to synchronize the frames to the start bit.

The following equations can be used to calculate the ratio of the incoming data rate and internal receiver baud rate.

$$R_{slow} = \frac{(D+1)S}{S-1+D\cdot S+S_F}$$

$$R_{fast} = \frac{(D+2)S}{(D+1)S + S_M}$$

- D Sum of character size and parity size (D = 5 to 10 bit)
- S Samples per bit. S = 16 for Normal Speed mode and S = 8 for Double Speed mode.
- S_F First sample number used for majority voting. S_F = 8 for Normal Speed and S_F = 4 for Double Speed mode.
- S_M Middle sample number used for majority voting. S_M = 9 for Normal Speed and S_M = 5 for Double Speed mode.
- R_{slow} is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate. R_{fast} is the ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate.

Table 61 and Table 62 list the maximum receiver baud rate error that can be tolerated. Note that Normal Speed mode has higher toleration of baud rate variations.

Table 61. Recommended Maximum Receiver Baud Rate Error for Normal Speed Mode (U2X = 0)

D # (Data+Parity Bit)	R _{slow} (%)	R _{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	93.20	106.67	+6.67/-6.8	±3.0
6	94.12	105.79	+5.79/-5.88	±2.5
7	94.81	105.11	+5.11/-5.19	±2.0
8	95.36	104.58	+4.58/-4.54	±2.0
9	95.81	104.14	+4.14/-4.19	±1.5
10	96.17	103.78	+3.78/-3.83	±1.5

Table 62. Recommended Maximum Receiver Baud Rate Error for Double Speed Mode (U2X = 1)

D # (Data+Parity Bit)	R _{slow} (%)	R _{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	94.12	105.66	+5.66/-5.88	±2.5
6	94.92	104.92	+4.92/-5.08	±2.0
7	95.52	104.35	+4.35/-4.48	±1.5





Table 62. Recommended Maximum Receiver Baud Rate Error for Double Speed Mode (U2X = 1)

D # (Data+Parity Bit)	R _{slow} (%)	R _{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
8	96.00	103.90	+3.90/-4.00	±1.5
9	96.39	103.53	+3.53/-3.61	±1.5
10	96.70	103.23	+3.23/-3.30	±1.0

The recommendations of the maximum receiver baud rate error was made under the assumption that the receiver and transmitter equally divides the maximum total error.

There are two possible sources for the receivers baud rate error. The receiver's system clock (XTAL) will always have some minor instability over the supply voltage range and the temperature range. When using a crystal to generate the system clock, this is rarely a problem, but for a resonator the system clock may differ more than 2% depending of the resonators tolerance. The second source for the error is more controllable. The baud rate generator can not always do an exact division of the system frequency to get the baud rate wanted. In this case an UBRR value that gives an acceptable low error can be used if possible.





Multi-processor Communication Mode

Setting the Multi-processor Communication mode (MPCM) bit in UCSRA enables a filtering function of incoming frames received by the USART Receiver. Frames that do not contain address information will be ignored and not put into the receive buffer. This effectively reduces the number of incoming frames that has to be handled by the CPU, in a system with multiple MCUs that communicate via the same serial bus. The Transmitter is unaffected by the MPCM setting, but has to be used differently when it is a part of a system utilizing the Multi-processor Communication mode.

If the receiver is set up to receive frames that contain 5 to 8 data bits, then the first stop bit indicates if the frame contains data or address information. If the receiver is set up for frames with nine data bits, then the ninth bit (RXB8) is used for identifying address and data frames. When the frame type bit (the first stop or the ninth bit) is one, the frame contains an address. When the frame type bit is zero the frame is a data frame.

The Multi-processor Communication mode enables several Slave MCUs to receive data from a Master MCU. This is done by first decoding an address frame to find out which MCU has been addressed. If a particular Slave MCU has been addressed, it will receive the following data frames as normal, while the other Slave MCUs will ignore the received frames until another address frame is received.

Using MPCM

For an MCU to act as a Master MCU, it can use a 9-bit character frame format (UCSZ = 7). The ninth bit (TXB8) must be set when an address frame (TXB8 = 1) or cleared when a data frame (TXB = 0) is being transmitted. The Slave MCUs must in this case be set to use a 9-bit character frame format.

The following procedure should be used to exchange data in Multi-processor Communication mode:

- 1. All Slave MCUs are in Multi-processor Communication mode (MPCM in UCSRA is set).
- The Master MCU sends an address frame, and all Slaves receive and read this frame. In the Slave MCUs, the RXC Flag in UCSRA will be set as normal.
- Each Slave MCU reads the UDR Register and determines if it has been selected. If so, it clears the MPCM bit in UCSRA, otherwise it waits for the next address byte and keeps the MPCM setting.
- The addressed MCU will receive all data frames until a new address frame is received.
 The other Slave MCUs, which still have the MPCM bit set, will ignore the data frames.
- When the last data frame is received by the addressed MCU, the addressed MCU sets the MPCM bit and waits for a new address frame from Master. The process then repeats from 2.

Using any of the 5-bit to 8-bit character frame formats is possible, but impractical since the receiver must change between using n and n+1 character frame formats. This makes full-duplex operation difficult since the transmitter and receiver uses the same character size setting. If 5-bit to 8-bit character frames are used, the transmitter must be set to use two stop bit (USBS = 1) since the first stop bit is used for indicating the frame type.

Do not use Read-Modify-Write instructions (SBI and CBI) to set or clear the MPCM bit. The MPCM bit shares the same I/O location as the TXC Flag and this might accidentally be cleared when using SBI or CBI instructions.





Accessing UBRRH/ UCSRC Registers

The UBRRH Register shares the same I/O location as the UCSRC Register. Therefore some special consideration must be taken when accessing this I/O location.

Write Access

When doing a write access of this I/O location, the high bit of the value written, the USART Register Select (URSEL) bit, controls which one of the two registers that will be written. If URSEL is zero during a write operation, the UBRRH value will be updated. If URSEL is one, the UCSRC setting will be updated.

The following code examples show how to access the two registers.

```
Assembly Code Example(1)

...
; Set UBRRH to 2
ldi r16,0x02
out UBRRH,r16
...
; Set the USBS and the UCSZ1 bit to one, and
; the remaining bits to zero.
ldi r16, (1<<URSEL) | (1<<USBS) | (1<<UCSZ1)
out UCSRC,r16
...

C Code Example(1)

...

/* Set UBRRH to 2 */
UBRRH = 0x02;
...

/* Set the USBS and the UCSZ1 bit to one, and */
/* the remaining bits to zero. */
UCSRC = (1<<URSEL) | (1<<USBS) | (1<<UCSZ1);
...
```

Note: 1. See "About Code Examples" on page 7.

As the code examples illustrate, write accesses of the two registers are relatively unaffected of the sharing of I/O location.

Read Access

Doing a read access to the UBRRH or the UCSRC Register is a more complex operation. However, in most applications, it is rarely necessary to read any of these registers.

The read access is controlled by a timed sequence. Reading the I/O location once returns the UBRRH Register contents. If the register location was read in previous system clock cycle, reading the register in the current clock cycle will return the UCSRC contents. Note that the timed sequence for reading the UCSRC is an atomic operation. Interrupts must therefore be controlled (for example by disabling interrupts globally) during the read operation.





The following code example shows how to read the UCSRC Register contents.

```
Assembly Code Example(1)

USART_ReadUCSRC:
; Read UCSRC
in r16,UBRRH
in r16,UCSRC
ret

C Code Example(1)

unsigned char USART_ReadUCSRC( void )
{
 unsigned char ucsrc;
   /* Read UCSRC */
   ucsrc = UBRRH;
   ucsrc = UCSRC;
   return ucsrc;
}
```

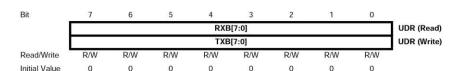
Note: 1. See "About Code Examples" on page 7.

The assembly code example returns the UCSRC value in r16.

Reading the UBRRH contents is not an atomic operation and therefore it can be read as an ordinary register, as long as the previous instruction did not access the register location.

USART Register Description

USART I/O Data Register – UDR



The USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDR. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDR Register location. Reading the UDR Register location will return the contents of the Receive Data Buffer Register (RXB).

For 5-bit, 6-bit, or 7-bit characters the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver.

The transmit buffer can only be written when the UDRE Flag in the UCSRA Register is set. Data written to UDR when the UDRE Flag is not set, will be ignored by the USART Transmitter. When data is written to the transmit buffer, and the Transmitter is enabled, the Transmitter will load the data into the transmit Shift Register when the Shift Register is empty. Then the data will be serially transmitted on the TxD pin.

The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use read modify write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.



163



USART Control and Status Register A – UCSRA

Bit	7		5	4	4 3		1	0	
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	10
Initial Value	0	0	1	0	0	0	0	0	

· Bit 7 - RXC: USART Receive Complete

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (that is, does not contain any unread data). If the receiver is disabled, the receive buffer will be flushed and consequently the RXC bit will become zero. The RXC Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE bit).

• Bit 6 - TXC: USART Transmit Complete

This flag bit is set when the entire frame in the transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR). The TXC Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC Flag can generate a Transmit Complete interrupt (see description of the TXCIE bit).

· Bit 5 - UDRE: USART Data Register Empty

The UDRE Flag indicates if the transmit buffer (UDR) is ready to receive new data. If UDRE is one, the buffer is empty, and therefore ready to be written. The UDRE Flag can generate a Data Register empty Interrupt (see description of the UDRIE bit).

UDRE is set after a reset to indicate that the transmitter is ready.

· Bit 4 - FE: Frame Error

This bit is set if the next character in the receive buffer had a Frame Error when received. that is, when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDR) is read. The FE bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSRA.

· Bit 3 - DOR: Data OverRun

This bit is set if a Data OverRun condition is detected. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the receive Shift Register, and a new start bit is detected. This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

Bit 2 – PE: Parity Error

This bit is set if the next character in the receive buffer had a Parity Error when received and the parity checking was enabled at that point (UPM1 = 1). This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

Bit 1 – U2X: Double the USART Transmission Speed

This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.





Bit 0 – MPCM: Multi-processor Communication Mode

This bit enables the Multi-processor Communication mode. When the MPCM bit is written to one, all the incoming frames received by the USART receiver that do not contain address information will be ignored. The transmitter is unaffected by the MPCM setting. For more detailed information see "Multi-processor Communication Mode" on page 161.

USART Control and Status Register B – UCSRB

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	•
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 - RXCIE: RX Complete Interrupt Enable

Writing this bit to one enables interrupt on the RXC Flag. A USART Receive Complete Interrupt will be generated only if the RXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC bit in UCSRA is set.

• Bit 6 - TXCIE: TX Complete Interrupt Enable

Writing this bit to one enables interrupt on the TXC Flag. A USART Transmit Complete Interrupt will be generated only if the TXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC bit in UCSRA is set.

· Bit 5 - UDRIE: USART Data Register Empty Interrupt Enable

Writing this bit to one enables interrupt on the UDRE Flag. A Data Register Empty Interrupt will be generated only if the UDRIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE bit in UCSRA is set.

· Bit 4 - RXEN: Receiver Enable

Writing this bit to one enables the USART Receiver. The Receiver will override normal port operation for the RxD pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FE, DOR, and PE Flags.

• Bit 3 - TXEN: Transmitter Enable

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD pin when enabled. The disabling of the Transmitter (writing TXEN to zero) will not become effective until ongoing and pending transmissions are completed, that is, when the transmit Shift Register and transmit Buffer Register do not contain data to be transmitted. When disabled, the transmitter will no longer override the TxD port.

· Bit 2 - UCSZ2: Character Size

The UCSZ2 bits combined with the UCSZ1:0 bit in UCSRC sets the number of data bits (Character Size) in a frame the receiver and transmitter use.

· Bit 1 - RXB8: Receive Data Bit 8

RXB8 is the ninth data bit of the received character when operating with serial frames with nine data bits. Must be read before reading the low bits from UDR.





· Bit 0 - TXB8: Transmit Data Bit 8

TXB8 is the ninth data bit in the character to be transmitted when operating with serial frames with nine data bits. Must be written before writing the low bits to UDR.

USART Control and Status Register C – UCSRC

Bit	7	6	5	4	3	2	1	0	
	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	1.5
Initial Value	1	0	0	0	0	1	1	0	

The UCSRC Register shares the same I/O location as the UBRRH Register. See the "Accessing UBRRH/ UCSRC Registers" on page 162 section which describes how to access this register.

· Bit 7 - URSEL: Register Select

This bit selects between accessing the UCSRC or the UBRRH Register. It is read as one when reading UCSRC. The URSEL must be one when writing the UCSRC.

· Bit 6 - UMSEL: USART Mode Select

This bit selects between Asynchronous and Synchronous mode of operation.

Table 63. UMSEL Bit Settings

UMSEL	Mode
0	Asynchronous Operation
1	Synchronous Operation

· Bit 5:4 - UPM1:0: Parity Mode

These bits enable and set type of parity generation and check. If enabled, the transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data and compare it to the UPM0 setting. If a mismatch is detected, the PE Flag in UCSRA will be set.

Table 64. UPM Bits Settings

_			
	UPM1	UPM0	Parity Mode
	0	0	Disabled
	0	1	Reserved
	1	0	Enabled, Even Parity
	1	1	Enabled, Odd Parity

· Bit 3 - USBS: Stop Bit Select

This bit selects the number of Stop Bits to be inserted by the Transmitter. The Receiver ignores this setting.

Table 65. USBS Bit Settings

USBS	Stop Bit(s)
0	1-bit
1	2-bit





· Bit 2:1 - UCSZ1:0: Character Size

The UCSZ1:0 bits combined with the UCSZ2 bit in UCSRB sets the number of data bits (Character Size) in a frame the Receiver and Transmitter use.

Table 66. UCSZ Bits Settings

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

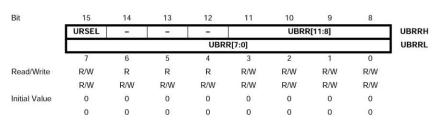
· Bit 0 - UCPOL: Clock Polarity

This bit is used for Synchronous mode only. Write this bit to zero when Asynchronous mode is used. The UCPOL bit sets the relationship between data output change and data input sample, and the synchronous clock (XCK).

Table 67. UCPOL Bit Settings

UCPOL	Transmitted Data Changed (Output of TxD Pin)	Received Data Sampled (Input on RxD Pin)
0	Rising XCK Edge	Falling XCK Edge
1	Falling XCK Edge	Rising XCK Edge

USART Baud Rate Registers – UBRRL and UBRRH



The UBRRH Register shares the same I/O location as the UCSRC Register. See the "Accessing UBRRH/ UCSRC Registers" on page 162 section which describes how to access this register.

· Bit 15 - URSEL: Register Select

This bit selects between accessing the UBRRH or the UCSRC Register. It is read as zero when reading UBRRH. The URSEL must be zero when writing the UBRRH.

· Bit 14:12 - Reserved Bits

These bits are reserved for future use. For compatibility with future devices, these bit must be written to zero when UBRRH is written.

<u>AIMEL</u>

167



· Bit 11:0 - UBRR11:0: USART Baud Rate Register

This is a 12-bit register which contains the USART baud rate. The UBRRH contains the four most significant bits, and the UBRRL contains the 8 least significant bits of the USART baud rate. Ongoing transmissions by the transmitter and receiver will be corrupted if the baud rate is changed. Writing UBRRL will trigger an immediate update of the baud rate prescaler.

Examples of Baud Rate Setting

For standard crystal and resonator frequencies, the most commonly used baud rates for asynchronous operation can be generated by using the UBRR settings in Table 68. UBRR values which yield an actual baud rate differing less than 0.5% from the target baud rate, are bold in the table. Higher error ratings are acceptable, but the receiver will have less noise resistance when the error ratings are high, especially for large serial frames (see "Asynchronous Operational Range" on page 159). The error values are calculated using the following equation:

$$Error[\%] = \left(\frac{BaudRate_{Closest\ Match}}{BaudRate} - 1\right) \bullet 100\%$$

Table 68. Examples of UBRR Settings for Commonly Used Oscillator Frequencies

		f _{osc} = 1.0	000 MHz			f _{osc} = 1.8	432 MHz		f _{osc} = 2.0000 MHz				
Baud Rate	U2X = 0 U		U2)	(= 1	U2X = 0		U2X = 1		U2X = 0		U2X = 1		
(bps)	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	
2400	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%	
4800	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%	
9600	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%	
14.4k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	
19.2k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	
28.8k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	
38.4k	1	-18.6%	2	8.5%	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	
57.6k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%	
76.8k	-	-	1	-18.6%	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%	
115.2k	-	-	0	8.5%	0	0.0%	1	0.0%	0	8.5%	1	8.5%	
230.4k	1-1	-	-	_	_	_	0	0.0%	_	_	-	_	
250k		_	_	_	_	_	_	_	_	_	0	0.0%	
Max (1)	62.5	Kbps	125	Kbps	115.2	Kbps	230.4	Kbps	125	Kbps	250	Kbps	

^{1.} UBRR = 0, Error = 0.0%





Table 69. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

		f _{osc} = 3.6	864 MHz			f _{osc} = 4.0	0000 MHz		f _{osc} = 7.3728 MHz				
Baud Rate	U2X = 0		U2X	U2X = 1		U2X = 0		U2X = 1		U2X = 0		(= 1	
(bps)	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	
2400	95	0.0%	191	0.0%	103	0.2%	207	0.2%	191	0.0%	383	0.0%	
4800	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%	
9600	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%	
14.4k	15	0.0%	31	0.0%	16	2.1%	34	-0.8%	31	0.0%	63	0.0%	
19.2k	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%	
28.8k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%	
38.4k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	
57.6k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	
76.8k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	
115.2k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%	
230.4k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%	
250k	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%	
0.5M	_	_	0	-7.8%	_	_	0	0.0%	0	-7.8%	1	-7.8%	
1M	_	_	_	_	_	_	_	_	_	-	0	-7.8%	
Max (1)	230.4	Kbps	460.8	Kbps	250	Kbps	0.5 N	Лbps	460.8	Kbps	921.6	Kbps	

^{1.} UBRR = 0, Error = 0.0%





Table 70. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

220		f _{osc} = 8.0	000 MHz			f _{osc} = 11.0	0592 MHz		f _{osc} = 14.7456 MHz				
Baud Rate	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1		
(bps)	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%	
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%	
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%	
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%	
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%	
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%	
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%	
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%	
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%	
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%	
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%	
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%	
0.5M	0	0.0%	1	0.0%	_	_	2	-7.8%	1	-7.8%	3	-7.8%	
1M		_	0	0.0%	_	_	_	_	0	-7.8%	1	-7.8%	
Max (1)	0.5 N	Иbps	1 M	bps	691.2	Kbps	1.3824	1 Mbps	921.6	Kbps	1.8432 Mbps		

^{1.} UBRR = 0, Error = 0.0%





Table 71. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

		f _{osc} = 16.0	0000 MHz			f _{osc} = 18.	4320 MHz		f _{osc} = 20.0000 MHz				
Baud Rate	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1		
(bps)	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%	
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%	
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%	
14.4k	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%	
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%	
28.8k	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%	
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%	
57.6k	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%	
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%	
115.2k	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%	
230.4k	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%	
250k	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%	
0.5M	1	0.0%	3	0.0%	-	=	4	-7.8%	1_	-	4	0.0%	
1M	0	0.0%	1	0.0%	_	_	_	_	_	_		_	
Max (1)	1 M	bps	2 M	bps	1.152	Mbps	2.304	Mbps	1.25	Mbps	2.5	Mbps	

^{1.} UBRR = 0, Error = 0.0%





Two-wire Serial Interface

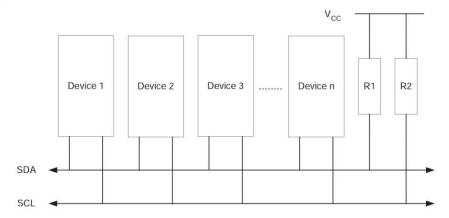
Features

- · Simple Yet Powerful and Flexible Communication Interface, Only Two Bus Lines Needed
- Both Master and Slave Operation Supported
- · Device Can Operate as Transmitter or Receiver
- · 7-bit Address Space allows up to 128 Different Slave Addresses
- · Multi-master Arbitration Support
- · Up to 400 kHz Data Transfer Speed
- Slew-rate Limited Output Drivers
- Noise Suppression Circuitry Rejects Spikes on Bus Lines
- · Fully Programmable Slave Address with General Call Support
- · Address Recognition causes Wake-up when AVR is in Sleep Mode

Two-wire Serial Interface Bus Definition

The Two-wire Serial Interface (TWI) is ideally suited for typical microcontroller applications. The TWI protocol allows the systems designer to interconnect up to 128 different devices using only two bi-directional bus lines, one for clock (SCL) and one for data (SDA). The only external hardware needed to implement the bus is a single pull-up resistor for each of the TWI bus lines. All devices connected to the bus have individual addresses, and mechanisms for resolving bus contention are inherent in the TWI protocol.

Figure 76. TWI Bus Interconnection



TWI Terminology

The following definitions are frequently encountered in this section.

Table 72. TWI Terminology

Term	Description
Master	The device that initiates and terminates a transmission. The Master also generates the SCL clock.
Slave	The device addressed by a Master.
Transmitter	The device placing data on the bus.
Receiver	The device reading data from the bus.





Electrical Interconnection

As depicted in Figure 76, both bus lines are connected to the positive supply voltage through pull-up resistors. The bus drivers of all TWI-compliant devices are open-drain or open-collector. This implements a wired-AND function which is essential to the operation of the interface. A low level on a TWI bus line is generated when one or more TWI devices output a zero. A high level is output when all TWI devices tri-state their outputs, allowing the pull-up resistors to pull the line high. Note that all AVR devices connected to the TWI bus must be powered in order to allow any bus operation.

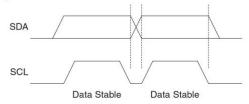
The number of devices that can be connected to the bus is only limited by the bus capacitance limit of 400 pF and the 7-bit Slave address space. A detailed specification of the electrical characteristics of the TWI is given in "Two-wire Serial Interface Characteristics" on page 294. Two different sets of specifications are presented there, one relevant for bus speeds below 100 kHz, and one valid for bus speeds up to 400 kHz.

Data Transfer and Frame Format

Transferring Bits

Each data bit transferred on the TWI bus is accompanied by a pulse on the clock line. The level of the data line must be stable when the clock line is high. The only exception to this rule is for generating start and stop conditions.

Figure 77. Data Validity



Data Change

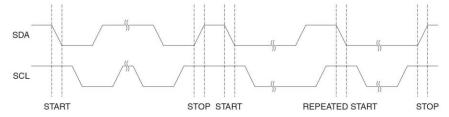
START and STOP Conditions

The Master initiates and terminates a data transmission. The transmission is initiated when the Master issues a START condition on the bus, and it is terminated when the Master issues a STOP condition. Between a START and a STOP condition, the bus is considered busy, and no other Master should try to seize control of the bus. A special case occurs when a new START condition is issued between a START and STOP condition. This is referred to as a REPEATED START condition, and is used when the Master wishes to initiate a new transfer without releasing control of the bus. After a REPEATED START, the bus is considered busy until the next STOP. This is identical to the START behavior, and therefore START is used to describe both START and REPEATED START for the remainder of this datasheet, unless otherwise noted. As depicted below, START and STOP conditions are signalled by changing the level of the SDA line when the SCL line is high.





Figure 78. START, REPEATED START, and STOP Conditions



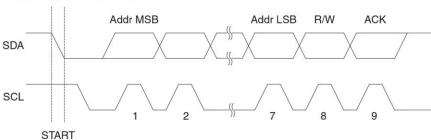
Address Packet Format All address packets transmitted on the TWI bus are nine bits long, consisting of seven address bits, one READ/WRITE control bit and an acknowledge bit. If the READ/WRITE bit is set, a read operation is to be performed, otherwise a write operation should be performed. When a Slave recognizes that it is being addressed, it should acknowledge by pulling SDA low in the ninth SCL (ACK) cycle. If the addressed Slave is busy, or for some other reason can not service the Master's request, the SDA line should be left high in the ACK clock cycle. The Master can then transmit a STOP condition, or a REPEATED START condition to initiate a new transmission. An address packet consisting of a Slave address and a READ or a WRITE bit is called SLA+R or SLA+W, respectively.

The MSB of the address byte is transmitted first. Slave addresses can freely be allocated by the designer, but the address 0000 000 is reserved for a general call.

When a general call is issued, all Slaves should respond by pulling the SDA line low in the ACK cycle. A general call is used when a Master wishes to transmit the same message to several Slaves in the system. When the general call address followed by a Write bit is transmitted on the bus, all Slaves set up to acknowledge the general call will pull the SDA line low in the ack cycle. The following data packets will then be received by all the Slaves that acknowledged the general call. Note that transmitting the general call address followed by a Read bit is meaningless, as this would cause contention if several Slaves started transmitting different data.

All addresses of the format 1111 xxx should be reserved for future purposes.

Figure 79. Address Packet Format

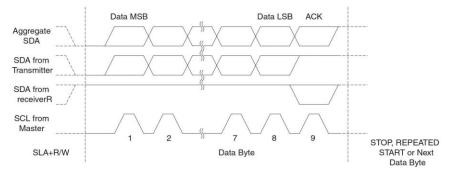




Data Packet Format

All data packets transmitted on the TWI bus are nine bits long, consisting of one data byte and an acknowledge bit. During a data transfer, the Master generates the clock and the START and STOP conditions, while the receiver is responsible for acknowledging the reception. An Acknowledge (ACK) is signalled by the receiver pulling the SDA line low during the ninth SCL cycle. If the receiver leaves the SDA line high, a NACK is signalled. When the receiver has received the last byte, or for some reason cannot receive any more bytes, it should inform the transmitter by sending a NACK after the final byte. The MSB of the data byte is transmitted first.

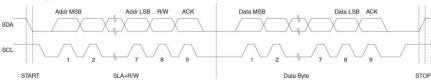
Figure 80. Data Packet Format



Combining Address and Data Packets into a Transmission A transmission basically consists of a START condition, a SLA+R/W, one or more data packets and a STOP condition. An empty message, consisting of a START followed by a STOP condition, is illegal. Note that the wired-ANDing of the SCL line can be used to implement handshaking between the Master and the Slave. The Slave can extend the SCL low period by pulling the SCL line low. This is useful if the clock speed set up by the Master is too fast for the Slave, or the Slave needs extra time for processing between the data transmissions. The Slave extending the SCL low period will not affect the SCL high period, which is determined by the Master. As a consequence, the Slave can reduce the TWI data transfer speed by prolonging the SCL duty cycle.

Figure 81 shows a typical data transmission. Note that several data bytes can be transmitted between the SLA+R/W and the STOP condition, depending on the software protocol implemented by the application software.

Figure 81. Typical Data Transmission





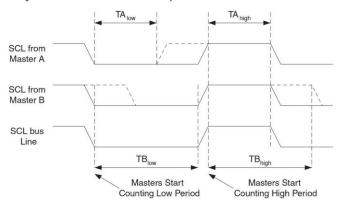


Multi-master Bus Systems, Arbitration and Synchronization The TWI protocol allows bus systems with several Masters. Special concerns have been taken in order to ensure that transmissions will proceed as normal, even if two or more Masters initiate a transmission at the same time. Two problems arise in multi-master systems:

- An algorithm must be implemented allowing only one of the Masters to complete the
 transmission. All other Masters should cease transmission when they discover that they
 have lost the selection process. This selection process is called arbitration. When a
 contending Master discovers that it has lost the arbitration process, it should immediately
 switch to Slave mode to check whether it is being addressed by the winning Master. The fact
 that multiple Masters have started transmission at the same time should not be detectable to
 the Slaves, that is, the data being transferred on the bus must not be corrupted.
- Different Masters may use different SCL frequencies. A scheme must be devised to synchronize the serial clocks from all Masters, in order to let the transmission proceed in a lockstep fashion. This will facilitate the arbitration process.

The wired-ANDing of the bus lines is used to solve both these problems. The serial clocks from all Masters will be wired-ANDed, yielding a combined clock with a high period equal to the one from the Master with the shortest high period. The low period of the combined clock is equal to the low period of the Master with the longest low period. Note that all Masters listen to the SCL line, effectively starting to count their SCL high and low time-out periods when the combined SCL line goes high or low, respectively.

Figure 82. SCL Synchronization between Multiple Masters

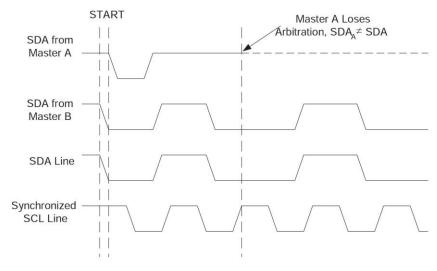


Arbitration is carried out by all Masters continuously monitoring the SDA line after outputting data. If the value read from the SDA line does not match the value the Master had output, it has lost the arbitration. Note that a Master can only lose arbitration when it outputs a high SDA value while another Master outputs a low value. The losing Master should immediately go to Slave mode, checking if it is being addressed by the winning Master. The SDA line should be left high, but losing Masters are allowed to generate a clock signal until the end of the current data or address packet. Arbitration will continue until only one Master remains, and this may take many bits. If several Masters are trying to address the same Slave, arbitration will continue into the data packet.





Figure 83. Arbitration between Two Masters



Note that arbitration is not allowed between:

- · A REPEATED START condition and a data bit
- · A STOP condition and a data bit
- · A REPEATED START and a STOP condition

It is the user software's responsibility to ensure that these illegal arbitration conditions never occur. This implies that in multi-master systems, all data transfers must use the same composition of SLA+R/W and data packets. In other words: All transmissions must contain the same number of data packets, otherwise the result of the arbitration is undefined.

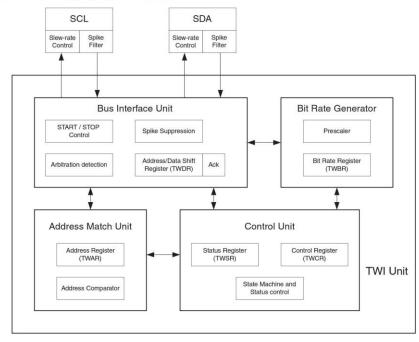




Overview of the TWI Module

The TWI module is comprised of several submodules, as shown in Figure 84. All registers drawn in a thick line are accessible through the AVR data bus.

Figure 84. Overview of the TWI Module



SCL and SDA Pins

These pins interface the AVR TWI with the rest of the MCU system. The output drivers contain a slew-rate limiter in order to conform to the TWI specification. The input stages contain a spike suppression unit removing spikes shorter than 50 ns. Note that the internal pull-ups in the AVR pads can be enabled by setting the PORT bits corresponding to the SCL and SDA pins, as explained in the I/O Port section. The internal pull-ups can in some systems eliminate the need for external ones.

Bit Rate Generator Unit

This unit controls the period of SCL when operating in a Master mode. The SCL period is controlled by settings in the TWI Bit Rate Register (TWBR) and the Prescaler bits in the TWI Status Register (TWSR). Slave operation does not depend on Bit Rate or Prescaler settings, but the CPU clock frequency in the Slave must be at least 16 times higher than the SCL frequency. Note that Slaves may prolong the SCL low period, thereby reducing the average TWI bus clock period. The SCL frequency is generated according to the following equation:

SCL frequency =
$$\frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot 4^{TWPS}}$$

- TWBR = Value of the TWI Bit Rate Register
- TWPS = Value of the prescaler bits in the TWI Status Register

Note: Note: Pull-up resistor values should be selected according to the SCL frequency and the capacitive bus line load. See Table 120 on page 294 for value of pull-up resistor.

Bus Interface Unit

This unit contains the Data and Address Shift Register (TWDR), a START/STOP Controller and Arbitration detection hardware. The TWDR contains the address or data bytes to be transmitted,

<u>AIMEL</u>

178



or the address or data bytes received. In addition to the 8-bit TWDR, the Bus Interface Unit also contains a register containing the (N)ACK bit to be transmitted or received. This (N)ACK Register is not directly accessible by the application software. However, when receiving, it can be set or cleared by manipulating the TWI Control Register (TWCR). When in Transmitter mode, the value of the received (N)ACK bit can be determined by the value in the TWSR.

The START/STOP Controller is responsible for generation and detection of START, REPEATED START, and STOP conditions. The START/STOP controller is able to detect START and STOP conditions even when the AVR MCU is in one of the sleep modes, enabling the MCU to wake up if addressed by a Master.

If the TWI has initiated a transmission as Master, the Arbitration Detection hardware continuously monitors the transmission trying to determine if arbitration is in process. If the TWI has lost an arbitration, the Control Unit is informed. Correct action can then be taken and appropriate status codes generated.

Address Match Unit

The Address Match unit checks if received address bytes match the 7-bit address in the TWI Address Register (TWAR). If the TWI General Call Recognition Enable (TWGCE) bit in the TWAR is written to one, all incoming address bits will also be compared against the General Call address. Upon an address match, the Control Unit is informed, allowing correct action to be taken. The TWI may or may not acknowledge its address, depending on settings in the TWCR. The Address Match unit is able to compare addresses even when the AVR MCU is in sleep mode, enabling the MCU to wake up if addressed by a Master.

Control Unit

The Control unit monitors the TWI bus and generates responses corresponding to settings in the TWI Control Register (TWCR). When an event requiring the attention of the application occurs on the TWI bus, the TWI Interrupt Flag (TWINT) is asserted. In the next clock cycle, the TWI Status Register (TWSR) is updated with a status code identifying the event. The TWSR only contains relevant status information when the TWI Interrupt Flag is asserted. At all other times, the TWSR contains a special status code indicating that no relevant status information is available. As long as the TWINT Flag is set, the SCL line is held low. This allows the application software to complete its tasks before allowing the TWI transmission to continue.

The TWINT Flag is set in the following situations:

- · After the TWI has transmitted a START/REPEATED START condition
- · After the TWI has transmitted SLA+R/W
- · After the TWI has transmitted an address byte
- · After the TWI has lost arbitration
- After the TWI has been addressed by own Slave address or general call
- After the TWI has received a data byte
- After a STOP or REPEATED START has been received while still addressed as a Slave.
- When a bus error has occurred due to an illegal START or STOP condition





TWI Register Description

TWI Bit Rate Register – TWBR

Bit	7	6	5	4	3	2	1	0	
	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

· Bits 7..0 - TWI Bit Rate Register

TWBR selects the division factor for the bit rate generator. The bit rate generator is a frequency divider which generates the SCL clock frequency in the Master modes. See "Bit Rate Generator Unit" on page 178 for calculating bit rates.

TWI Control Register – TWCR

Bit	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	•
Initial Value	0	0	0	0	0	0	0	0	

The TWCR is used to control the operation of the TWI. It is used to enable the TWI, to initiate a Master access by applying a START condition to the bus, to generate a receiver acknowledge, to generate a stop condition, and to control halting of the bus while the data to be written to the bus are written to the TWDR. It also indicates a write collision if data is attempted written to TWDR while the register is inaccessible.

· Bit 7 - TWINT: TWI Interrupt Flag

This bit is set by hardware when the TWI has finished its current job and expects application software response. If the I-bit in SREG and TWIE in TWCR are set, the MCU will jump to the TWI Interrupt Vector. While the TWINT Flag is set, the SCL low period is stretched. The TWINT Flag must be cleared by software by writing a logic one to it. Note that this flag is not automatically cleared by hardware when executing the interrupt routine. Also note that clearing this flag starts the operation of the TWI, so all accesses to the TWI Address Register (TWAR), TWI Status Register (TWSR), and TWI Data Register (TWDR) must be complete before clearing this flag.

· Bit 6 - TWEA: TWI Enable Acknowledge Bit

The TWEA bit controls the generation of the acknowledge pulse. If the TWEA bit is written to one, the ACK pulse is generated on the TWI bus if the following conditions are met:

- 1. The device's own Slave address has been received.
- 2. A general call has been received, while the TWGCE bit in the TWAR is set.
- 3. A data byte has been received in Master Receiver or Slave Receiver mode.

By writing the TWEA bit to zero, the device can be virtually disconnected from the Two-wire Serial Bus temporarily. Address recognition can then be resumed by writing the TWEA bit to one again.

· Bit 5 - TWSTA: TWI START Condition Bit

The application writes the TWSTA bit to one when it desires to become a Master on the Twowire Serial Bus. The TWI hardware checks if the bus is available, and generates a START condition on the bus if it is free. However, if the bus is not free, the TWI waits until a STOP condition

AIMEL

180



is detected, and then generates a new START condition to claim the bus Master status. TWSTA must be cleared by software when the START condition has been transmitted.

• Bit 4 - TWSTO: TWI STOP Condition Bit

Writing the TWSTO bit to one in Master mode will generate a STOP condition on the Two-wire Serial Bus. When the STOP condition is executed on the bus, the TWSTO bit is cleared automatically. In Slave mode, setting the TWSTO bit can be used to recover from an error condition. This will not generate a STOP condition, but the TWI returns to a well-defined unaddressed Slave mode and releases the SCL and SDA lines to a high impedance state.

· Bit 3 - TWWC: TWI Write Collision Flag

The TWWC bit is set when attempting to write to the TWI Data Register – TWDR when TWINT is low. This flag is cleared by writing the TWDR Register when TWINT is high.

· Bit 2 - TWEN: TWI Enable Bit

The TWEN bit enables TWI operation and activates the TWI interface. When TWEN is written to one, the TWI takes control over the I/O pins connected to the SCL and SDA pins, enabling the slew-rate limiters and spike filters. If this bit is written to zero, the TWI is switched off and all TWI transmissions are terminated, regardless of any ongoing operation.

· Bit 1 - Res: Reserved Bit

This bit is a reserved bit and will always read as zero.

• Bit 0 - TWIE: TWI Interrupt Enable

When this bit is written to one, and the I-bit in SREG is set, the TWI interrupt request will be activated for as long as the TWINT Flag is high.

TWI Status Register – TWSR

Bit	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	10
Initial Value	1	1	1	1	1	0	0	0	

· Bits 7..3 - TWS: TWI Status

These five bits reflect the status of the TWI logic and the Two-wire Serial Bus. The different status codes are described later in this section. Note that the value read from TWSR contains both the 5-bit status value and the 2-bit prescaler value. The application designer should mask the prescaler bits to zero when checking the Status bits. This makes status checking independent of prescaler setting. This approach is used in this datasheet, unless otherwise noted.

• Bit 2 - Res: Reserved Bit

This bit is reserved and will always read as zero.





· Bits 1..0 - TWPS: TWI Prescaler Bits

These bits can be read and written, and control the bit rate prescaler.

Table 73. TWI Bit Rate Prescaler

TWPS1	TWPS0	Prescaler Value
0	0	1
0	1	4
1	0	16
1	1	64

To calculate bit rates, see "Bit Rate Generator Unit" on page 178. The value of TWPS1..0 is used in the equation.

TWI Data Register – TWDR

Bit	7	6	5	4	3	2	1	0	20
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W								
Initial Value	1	1	1	1	1	1	1	1	

In Transmit mode, TWDR contains the next byte to be transmitted. In Receive mode, the TWDR contains the last byte received. It is writable while the TWI is not in the process of shifting a byte. This occurs when the TWI Interrupt Flag (TWINT) is set by hardware. Note that the Data Register cannot be initialized by the user before the first interrupt occurs. The data in TWDR remains stable as long as TWINT is set. While data is shifted out, data on the bus is simultaneously shifted in. TWDR always contains the last byte present on the bus, except after a wake up from a sleep mode by the TWI interrupt. In this case, the contents of TWDR is undefined. In the case of a lost bus arbitration, no data is lost in the transition from Master to Slave. Handling of the ACK bit is controlled automatically by the TWI logic, the CPU cannot access the ACK bit directly.

· Bits 7..0 - TWD: TWI Data Register

These eight bits contain the next data byte to be transmitted, or the latest data byte received on the Two-wire Serial Bus.

TWI (Slave) Address Register – TWAR

Bit	7	6	5	4	3	2	1	0	
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
Read/Write	R/W								
Initial Value	1	1	1	1	1	1	1	0	

The TWAR should be loaded with the 7-bit Slave address (in the seven most significant bits of TWAR) to which the TWI will respond when programmed as a Slave Transmitter or receiver. In multi-master systems, TWAR must be set in Masters which can be addressed as Slaves by other Masters.

The LSB of TWAR is used to enable recognition of the general call address (\$00). There is an associated address comparator that looks for the Slave address (or general call address if enabled) in the received serial address. If a match is found, an interrupt request is generated.

· Bits 7..1 - TWA: TWI (Slave) Address Register

These seven bits constitute the Slave address of the TWI unit.

AMEL

182



Bit 0 – TWGCE: TWI General Call Recognition Enable Bit

If set, this bit enables the recognition of a General Call given over the Two-wire Serial Bus.

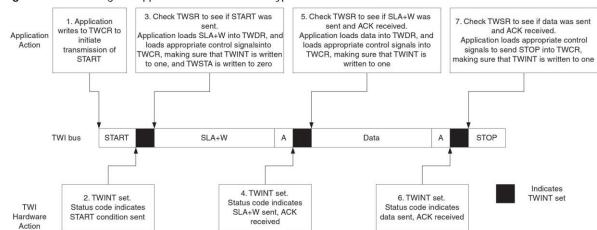
Using the TWI

The AVR TWI is byte-oriented and interrupt based. Interrupts are issued after all bus events, like reception of a byte or transmission of a START condition. Because the TWI is interrupt-based, the application software is free to carry on other operations during a TWI byte transfer. Note that the TWI Interrupt Enable (TWIE) bit in TWCR together with the Global Interrupt Enable bit in SREG allow the application to decide whether or not assertion of the TWINT Flag should generate an interrupt request. If the TWIE bit is cleared, the application must poll the TWINT Flag in order to detect actions on the TWI bus.

When the TWINT Flag is asserted, the TWI has finished an operation and awaits application response. In this case, the TWI Status Register (TWSR) contains a value indicating the current state of the TWI bus. The application software can then decide how the TWI should behave in the next TWI bus cycle by manipulating the TWCR and TWDR Registers.

Figure 85 is a simple example of how the application can interface to the TWI hardware. In this example, a Master wishes to transmit a single data byte to a Slave. This description is quite abstract, a more detailed explanation follows later in this section. A simple code example implementing the desired behavior is also presented.

Figure 85. Interfacing the Application to the TWI in a Typical Transmission



- The first step in a TWI transmission is to transmit a START condition. This is done by
 writing a specific value into TWCR, instructing the TWI hardware to transmit a START
 condition. Which value to write is described later on. However, it is important that the
 TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will
 not start any operation as long as the TWINT bit in TWCR is set. Immediately after the
 application has cleared TWINT, the TWI will initiate transmission of the START condition.
- When the START condition has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the START condition has successfully been sent.
- 3. The application software should now examine the value of TWSR, to make sure that the START condition was successfully transmitted. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must load SLA+W into TWDR. Remember that TWDR is used both for address and data. After TWDR has been loaded with the



183



desired SLA+W, a specific value must be written to TWCR, instructing the TWI hardware to transmit the SLA+W present in TWDR. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the address packet.

- 4. When the address packet has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the address packet has successfully been sent. The status code will also reflect whether a Slave acknowledged the packet or not.
- 5. The application software should now examine the value of TWSR, to make sure that the address packet was successfully transmitted, and that the value of the ACK bit was as expected. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must load a data packet into TWDR. Subsequently, a specific value must be written to TWCR, instructing the TWI hardware to transmit the data packet present in TWDR. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the data packet.
- 6. When the data packet has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the data packet has successfully been sent. The status code will also reflect whether a Slave acknowledged the packet or not.
- 7. The application software should now examine the value of TWSR, to make sure that the data packet was successfully transmitted, and that the value of the ACK bit was as expected. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must write a specific value to TWCR, instructing the TWI hardware to transmit a STOP condition. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the STOP condition. Note that TWINT is NOT set after a STOP condition has been sent.

Even though this example is simple, it shows the principles involved in all TWI transmissions. These can be summarized as follows:

- When the TWI has finished an operation and expects application response, the TWINT Flag is set. The SCL line is pulled low until TWINT is cleared.
- When the TWINT Flag is set, the user must update all TWI Registers with the value relevant for the next TWI bus cycle. As an example, TWDR must be loaded with the value to be transmitted in the next bus cycle.
- After all TWI Register updates and other pending application software tasks have been completed, TWCR is written. When writing TWCR, the TWINT bit should be set. Writing a one to TWINT clears the flag. The TWI will then commence executing whatever operation was specified by the TWCR setting.

In the following an assembly and C implementation of the example is given. Note that the code below assumes that several definitions have been made, for example by using include-files.





П	Assembly	code example	C example	Comments
1	ldi	r16, (1< <twint) (1<<twsta)="" th="" ="" <=""><th>TWCR = (1<<twint) (1<<twsta)="" th="" ="" <=""><th>Send START condition</th></twint)></th></twint)>	TWCR = (1< <twint) (1<<twsta)="" th="" ="" <=""><th>Send START condition</th></twint)>	Send START condition
		(1< <twen)< th=""><th>(1<<twen)< th=""><th></th></twen)<></th></twen)<>	(1< <twen)< th=""><th></th></twen)<>	
Ш	out	TWCR, r16		
2	wait1	:	while (!(TWCR & (1< <twint)))< th=""><th>Wait for TWINT Flag set. This indicates</th></twint)))<>	Wait for TWINT Flag set. This indicates
	in	r16,TWCR	,	that the START condition has been
	sbrs	r16,TWINT		transmitted
	rjmp	wait1		
3	in	r16,TWSR	if ((TWSR & 0xF8) != START)	Check value of TWI Status Register. Mask
	andi	r16, 0xF8	ERROR();	prescaler bits. If status different from
	cpi	r16, START		START go to ERROR
	brne	ERROR		
	ldi	r16, SLA_W	TWDR = SLA_W;	Load SLA_W into TWDR Register. Clear
	out	TWDR, r16	TWCR = (1< <twint) (1<<twen);<="" th="" =""><th>TWINT bit in TWCR to start transmission</th></twint)>	TWINT bit in TWCR to start transmission
	ldi	r16, (1< <twint) (1<<twen)<="" th="" =""><th></th><th>of address</th></twint)>		of address
	out	TWCR, r16		
4	wait2	:	while (!(TWCR & (1< <twint)))< th=""><th>Wait for TWINT Flag set. This indicates</th></twint)))<>	Wait for TWINT Flag set. This indicates
	in	r16,TWCR	,	that the SLA+W has been transmitted,
	sbrs	r16,TWINT		and ACK/NACK has been received.
Ш	rjmp	wait2		
5	in	r16,TWSR	if ((TWSR & 0xF8) != MT_SLA_ACK)	Check value of TWI Status Register. Mask
	andi	r16, 0xF8	ERROR();	prescaler bits. If status different from
	cpi	r16, MT_SLA_ACK		MT_SLA_ACK go to ERROR
	040.0000	ERROR		
	ldi	r16, DATA	TWDR = DATA;	Load DATA into TWDR Register. Clear
	out	TWDR, r16	TWCR = (1< <twint) (1<<twen);<="" th="" =""><th>TWINT bit in TWCR to start transmission</th></twint)>	TWINT bit in TWCR to start transmission
	ldi	r16, (1< <twint) (1<<twen)<="" th="" =""><th></th><th>of data</th></twint)>		of data
Ш	out	TWCR, r16		
6	wait3		while (!(TWCR & (1< <twint)))< th=""><th>Wait for TWINT Flag set. This indicates</th></twint)))<>	Wait for TWINT Flag set. This indicates
	in	r16,TWCR	,	that the DATA has been transmitted, and
		r16,TWINT		ACK/NACK has been received.
H		wait3	Microson will describe which is about the property of the control of the property of the prope	1000 1000 100 100 100 100 100 100 100 1
7	in	r16,TWSR	if ((TWSR & 0xF8) != MT_DATA_ACK)	Check value of TWI Status Register. Mask
		r16, 0xF8	ERROR();	prescaler bits. If status different from
	cpi	r16, MT_DATA_ACK		MT_DATA_ACK go to ERROR
	231/07/2013/2013	ERROR		
	ldi	r16, (1< <twint) (1<<twen)="" th="" ="" <=""><th>TWCR = (1<<twint) (1<<twen)="" th="" ="" <=""><th>Transmit STOP condition</th></twint)></th></twint)>	TWCR = (1< <twint) (1<<twen)="" th="" ="" <=""><th>Transmit STOP condition</th></twint)>	Transmit STOP condition
		(1< <twsto)< th=""><th>(1<<twsto);< th=""><th></th></twsto);<></th></twsto)<>	(1< <twsto);< th=""><th></th></twsto);<>	
Ц	out	TWCR, r16		





Transmission Modes

The TWI can operate in one of four major modes. These are named Master Transmitter (MT), Master Receiver (MR), Slave Transmitter (ST) and Slave Receiver (SR). Several of these modes can be used in the same application. As an example, the TWI can use MT mode to write data into a TWI EEPROM, MR mode to read the data back from the EEPROM. If other Masters are present in the system, some of these might transmit data to the TWI, and then SR mode would be used. It is the application software that decides which modes are legal.

The following sections describe each of these modes. Possible status codes are described along with figures detailing data transmission in each of the modes. These figures contain the following abbreviations:

S: START condition

Rs: REPEATED START condition R: Read bit (high level at SDA)

W: Write bit (low level at SDA)

A: Acknowledge bit (low level at SDA)

A: Not acknowledge bit (high level at SDA)

Data: 8-bit data byte P: STOP condition SLA: Slave Address

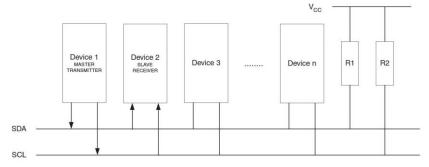
In Figure 87 to Figure 93, circles are used to indicate that the TWINT Flag is set. The numbers in the circles show the status code held in TWSR, with the prescaler bits masked to zero. At these points, actions must be taken by the application to continue or complete the TWI transfer. The TWI transfer is suspended until the TWINT Flag is cleared by software.

When the TWINT Flag is set, the status code in TWSR is used to determine the appropriate software action. For each status code, the required software action and details of the following serial transfer are given in Table 74 to Table 77. Note that the prescaler bits are masked to zero in these tables.

Master Transmitter Mode

In the Master Transmitter mode, a number of data bytes are transmitted to a Slave Receiver (see Figure 86). In order to enter a Master mode, a START condition must be transmitted. The format of the following address packet determines whether Master Transmitter or Master Receiver mode is to be entered. If SLA+W is transmitted, MT mode is entered, if SLA+R is transmitted, MR mode is entered. All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 86. Data Transfer in Master Transmitter Mode



ATMEL

186



A START condition is sent by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	1	Х	1	0	Х	1	0	Х

TWEN must be set to enable the Two-wire Serial Interface, TWSTA must be written to one to transmit a START condition and TWINT must be written to one to clear the TWINT Flag. The TWI will then test the Two-wire Serial Bus and generate a START condition as soon as the bus becomes free. After a START condition has been transmitted, the TWINT Flag is set by hardware, and the status code in TWSR will be \$08 (See Table 74). In order to enter MT mode, SLA+W must be transmitted. This is done by writing SLA+W to TWDR. Thereafter the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	1	Х	0	0	Х	1	0	Х

When SLA+W have been transmitted and an acknowledgement bit has been received, TWINT is set again and a number of status codes in TWSR are possible. Possible status codes in Master mode are \$18, \$20, or \$38. The appropriate action to be taken for each of these status codes is detailed in Table 74.

When SLA+W has been successfully transmitted, a data packet should be transmitted. This is done by writing the data byte to TWDR. TWDR must only be written when TWINT is high. If not, the access will be discarded, and the Write Collision bit (TWWC) will be set in the TWCR Register. After updating TWDR, the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	1	Х	0	0	Х	1	0	Х

This scheme is repeated until the last byte has been sent and the transfer is ended by generating a STOP condition or a repeated START condition. A STOP condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	1	Х	0	1	Х	1	0	Х

A REPEATED START condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	1	Х	1	0	Х	1	0	Х

After a repeated START condition (state \$10) the Two-wire Serial Interface can access the same Slave again, or a new Slave without transmitting a STOP condition. Repeated START enables the Master to switch between Slaves, Master Transmitter mode and Master Receiver mode without losing control of the bus.

Table 74. Status Codes for Master Transmitter Mode

Status Code		Applica	tion Softw	are Resp	onse		
(TWSR) Prescaler Bits	Status of the Two-wire Serial Bus and Two-wire Serial Inter-		To TWCR				
are 0	face Hardware	To/from TWDR	STA	STO	TWINT	TWEA	Next Action Taken by TWI Hardware
\$08	A START condition has been transmitted	Load SLA+W	0	0	1	Х	SLA+W will be transmitted; ACK or NOT ACK will be received
\$10	A repeated START condition has been transmitted	Load SLA+W or	0	0	1	Х	SLA+W will be transmitted; ACK or NOT ACK will be received
	3 (30) (3 (3))	Load SLA+R	0	0	1	Х	SLA+R will be transmitted; Logic will switch to Master Receiver mode





Table 74. Status Codes for Master Transmitter Mode

\$18	SLA+W has been transmitted; ACK has been received	Load data byte or	0	0	1	Х	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	Х	STOP condition will be transmitted and TWSTO Flag will be Reset
		No TWDR action	1	1	1	Х	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be Reset
\$20	SLA+W has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	х	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	Х	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	Х	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
\$28	Data byte has been transmitted; ACK has been received	Load data byte or	0	0	1	Х	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	Х	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	Х	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
\$30	Data byte has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	Х	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	Х	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	Х	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
\$38	Arbitration lost in SLA+W or data bytes	No TWDR action or	0	0	1	Х	Two-wire Serial Bus will be released and not addressed Slave mode entered
		No TWDR action	1	0	1	Х	A START condition will be transmitted when the bus becomes free





Successfull transmission to a slave receiver DATA S SLA W \$18 \$08 \$28 Next transfer started with a repeated start condition W R_{S} SLA (\$10) R Not acknowledge Р received after the slave address \$20 Not acknowledge received after a data byte MR \$30 Arbitration lost in slave A or A address or data byte \$38 \$38 Arbitration lost and addressed as slave \$68 \$78 \$B0 Any number of data bytes DATA From master to slave This number (contained in TWSR) corresponds to a defined state of the Two-wire Serial Bus. The prescaler bits are zero or masked to zero From slave to master (n)

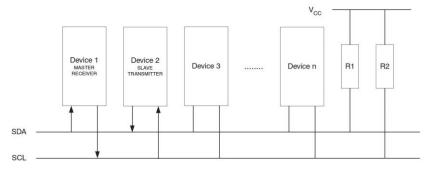
Figure 87. Formats and States in the Master Transmitter Mode

Master Receiver Mode

In the Master Receiver mode, a number of data bytes are received from a Slave Transmitter (see Figure 88). In order to enter a Master mode, a START condition must be transmitted. The format of the following address packet determines whether Master Transmitter or Master Receiver mode is to be entered. If SLA+W is transmitted, MT mode is entered, if SLA+R is transmitted, MR mode is entered. All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.



Figure 88. Data Transfer in Master Receiver Mode



A START condition is sent by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	1	Х	1	0	Х	1	0	Х

TWEN must be written to one to enable the Two-wire Serial Interface, TWSTA must be written to one to transmit a START condition and TWINT must be set to clear the TWINT Flag. The TWI will then test the Two-wire Serial Bus and generate a START condition as soon as the bus becomes free. After a START condition has been transmitted, the TWINT Flag is set by hardware, and the status code in TWSR will be \$08 (See Table 74). In order to enter MR mode, SLA+R must be transmitted. This is done by writing SLA+R to TWDR. Thereafter the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	1	Х	0	0	Х	1	0	Х

When SLA+R have been transmitted and an acknowledgement bit has been received, TWINT is set again and a number of status codes in TWSR are possible. Possible status codes in Master mode are \$38, \$40, or \$48. The appropriate action to be taken for each of these status codes is detailed in Table 75. Received data can be read from the TWDR Register when the TWINT Flag is set high by hardware. This scheme is repeated until the last byte has been received. After the last byte has been received, the MR should inform the ST by sending a NACK after the last received data byte. The transfer is ended by generating a STOP condition or a repeated START condition. A STOP condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	1	Х	0	1	Х	1	0	Х

A REPEATED START condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	1	Х	1	0	Х	1	0	Х

After a repeated START condition (state \$10) the Two-wire Serial Interface can access the same Slave again, or a new Slave without transmitting a STOP condition. Repeated START enables the Master to switch between Slaves, Master Transmitter mode and Master Receiver mode without losing control over the bus.

Table 75. Status Codes for Master Receiver Mode

-								
	Status Code		Applicat	tion Softw	are Resp	onse		
	(TWSR) Prescaler Bits	Status of the Two-wire Serial Bus and Two-wire Serial Inter-	1.7 . 6 . 11 .		To TWCR			
	are 0	face Hardware	To/from TWDR	STA	STO	TWINT	TWEA	Next Action Taken by TWI Hardware

ATMEL

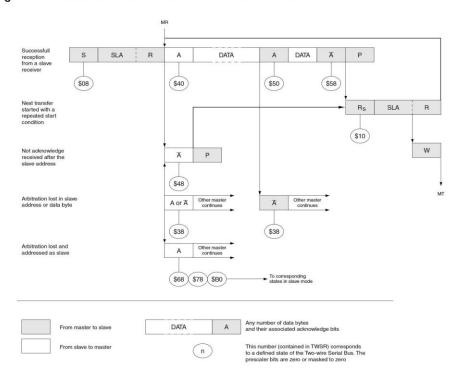
190



Table 75. Status Codes for Master Receiver Mode (Continued)

\$08	A START condition has been transmitted	Load SLA+R	0	0	1	Х	SLA+R will be transmitted ACK or NOT ACK will be received
\$10	A repeated START condition has been transmitted	Load SLA+R or Load SLA+W	0	0	1	x x	SLA+R will be transmitted ACK or NOT ACK will be received SLA+W will be transmitted Logic will switch to masTer Transmitter mode
\$38	Arbitration lost in SLA+R or NOT ACK bit	No TWDR action or No TWDR action	0	0	1	x x	Two-wire Serial Bus will be released and not addressed Slave mode will be entered A START condition will be transmitted when the bus becomes free
\$40	SLA+R has been transmitted; ACK has been received	No TWDR action or No TWDR action	0	0	1	0	Data byte will be received and NOT ACK will be returned Data byte will be received and ACK will be returned
\$48	SLA+R has been transmitted; NOT ACK has been received	No TWDR action or No TWDR action or No TWDR action	1 0	0 1 1	1 1 1	X X	Repeated START will be transmitted STOP condition will be transmitted and TWSTO Flag will be reset STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
\$50	Data byte has been received; ACK has been returned	Read data byte or Read data byte	0	0	1	0	Data byte will be received and NOT ACK will be returned Data byte will be received and ACK will be returned
\$58	Data byte has been received; NOT ACK has been returned	Read data byte or Read data byte or Read data byte	1 0	0 1	1 1 1	X X	Repeated START will be transmitted STOP condition will be transmitted and TWSTO Flag will be reset STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset

Figure 89. Formats and States in the Master Receiver Mode





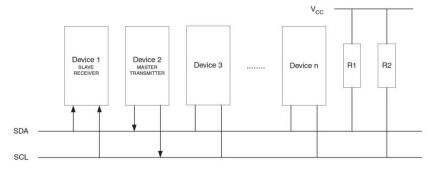
191



Slave Receiver Mode

In the Slave Receiver mode, a number of data bytes are received from a Master Transmitter (see Figure 90). All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 90. Data Transfer in Slave Receiver Mode



To initiate the Slave Receiver mode, TWAR and TWCR must be initialized as follows:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Value			Device's	s Own Slave	Address			

The upper seven bits are the address to which the Two-wire Serial Interface will respond when addressed by a Master. If the LSB is set, the TWI will respond to the general call address (\$00), otherwise it will ignore the general call address.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	. 	TWIE
Value	0	1	0	0	0	1	0	Х

TWEN must be written to one to enable the TWI. The TWEA bit must be written to one to enable the acknowledgement of the device's own Slave address or the general call address. TWSTA and TWSTO must be written to zero.

When TWAR and TWCR have been initialized, the TWI waits until it is addressed by its own Slave address (or the general call address if enabled) followed by the data direction bit. If the direction bit is "0" (write), the TWI will operate in SR mode, otherwise ST mode is entered. After its own Slave address and the write bit have been received, the TWINT Flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate software action. The appropriate action to be taken for each status code is detailed in Table 76. The Slave Receiver mode may also be entered if arbitration is lost while the TWI is in the Master mode (see states \$68 and \$78).

If the TWEA bit is reset during a transfer, the TWI will return a "Not Acknowledge" ("1") to SDA after the next received data byte. This can be used to indicate that the Slave is not able to receive any more bytes. While TWEA is zero, the TWI does not acknowledge its own Slave address. However, the Two-wire Serial Bus is still monitored and address recognition may resume at any time by setting TWEA. This implies that the TWEA bit may be used to temporarily isolate the TWI from the Two-wire Serial Bus.

In all sleep modes other than Idle Mode, the clock system to the TWI is turned off. If the TWEA bit is set, the interface can still acknowledge its own Slave address or the general call address by using the Two-wire Serial Bus clock as a clock source. The part will then wake up from sleep and the TWI will hold the SCL clock low during the wake up and until the TWINT Flag is cleared (by writing it to one). Further data reception will be carried out as normal, with the AVR clocks running as normal. Observe that if the AVR is set up with a long start-up time, the SCL line may be held low for a long time, blocking other data transmissions.





Note that the Two-wire Serial Interface Data Register – TWDR does not reflect the last byte present on the bus when waking up from these sleep modes.





Table 76. Status Codes for Slave Receiver Mode

Status Code		Applica	tion Softw	are Resp	onse		
(TWSR) Prescaler Bits	Status of the Two-wire Serial Bus and Two-wire Serial Interface	To/from TWDR	CTA		TWINT	TAFA	
are 0	Hardware		STA	STO	A. (4) (4) (4) (4)	TWEA	Next Action Taken by TWI Hardware
\$60	Own SLA+W has been received; ACK has been returned	No TWDR action or No TWDR action	X	0	1	0	Data byte will be received and NOT ACK will be returned Data byte will be received and ACK will be returned
\$68	Arbitration lost in SLA+R/W as	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be
\$00	Master; own SLA+W has been received; ACK has been returned	No TWDR action	X	0	1	1	returned Data byte will be received and NOT ACK will be returned
\$70	General call address has been	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be
*	received; ACK has been returned	No TWDR action	Х	0	1	1	returned Data byte will be received and ACK will be returned
\$78	Arbitration lost in SLA+R/W as	No TWDR action or	Х	0	1	0	Data byte will be received and NOT ACK will be
	Master; General call address has been received; ACK has been returned	No TWDR action	Х	0	1	1	returned Data byte will be received and ACK will be returned
\$80	Previously addressed with own SLA+W; data has been received;	Read data byte or	Х	0	1	0	Data byte will be received and NOT ACK will be returned
COSTONAGA	ACK has been returned	Read data byte	Х	0	1	1	Data byte will be received and ACK will be returned
\$88	Previously addressed with own SLA+W; data has been received;	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
	NOT ACK has been returned	Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free
\$90	Previously addressed with general call; data has been re-	Read data byte or	Х	0	1	0	Data byte will be received and NOT ACK will be returned
	ceived; ACK has been returned	Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
\$98	Previously addressed with general call; data has been	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
	received; NOT ACK has been returned	Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free
\$A0	A STOP condition or repeated START condition has been	No action	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
	received while still addressed as Slave		0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
			1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
			1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free



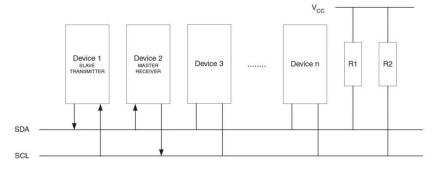


Reception of the own slave address and one or more data bytes. All are acknowledged W DATA S SLA DATA PorS \$60 \$80 (\$A0) \$80 Last data byte received is not acknowledged A PorS \$88 Arbitration lost as master Α \$68 General Call DATA DATA PorS A A \$70 \$90 (\$90) (\$A0) Last data byte received is not acknowledged PorS \$98 Arbitration lost as master and addressed as slave by general call Α (\$78) Any number of data bytes and their associated acknowledge bits DATA This number (contained in TWSR) corresponds to a defined state of the Two-wire Serial Bus. The prescaler bits are zero or masked to zero From slave to master (n)

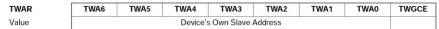
Figure 91. Formats and States in the Slave Receiver Mode

Slave Transmitter Mode In the Slave Transmitter mode, a number of data bytes are transmitted to a Master Receiver (see Figure 92). All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 92. Data Transfer in Slave Transmitter Mode



To initiate the Slave Transmitter mode, TWAR and TWCR must be initialized as follows:



AIMEL

195



The upper seven bits are the address to which the Two-wire Serial Interface will respond when addressed by a Master. If the LSB is set, the TWI will respond to the general call address (\$00), otherwise it will ignore the general call address.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	0	1	0	0	0	1	0	Х

TWEN must be written to one to enable the TWI. The TWEA bit must be written to one to enable the acknowledgement of the device's own Slave address or the general call address. TWSTA and TWSTO must be written to zero.

When TWAR and TWCR have been initialized, the TWI waits until it is addressed by its own Slave address (or the general call address if enabled) followed by the data direction bit. If the direction bit is "1" (read), the TWI will operate in ST mode, otherwise SR mode is entered. After its own Slave address and the write bit have been received, the TWINT Flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate software action. The appropriate action to be taken for each status code is detailed in Table 77. The Slave Transmitter mode may also be entered if arbitration is lost while the TWI is in the Master mode (see state \$B0).

If the TWEA bit is written to zero during a transfer, the TWI will transmit the last byte of the transfer. State \$C0 or state \$C8 will be entered, depending on whether the Master Receiver transmits a NACK or ACK after the final byte. The TWI is switched to the not addressed Slave mode, and will ignore the Master if it continues the transfer. Thus the Master Receiver receives all "1" as serial data. State \$C8 is entered if the Master demands additional data bytes (by transmitting ACK), even though the Slave has transmitted the last byte (TWEA zero and expecting NACK from the Master).

While TWEA is zero, the TWI does not respond to its own Slave address. However, the Twowire Serial Bus is still monitored and address recognition may resume at any time by setting TWEA. This implies that the TWEA bit may be used to temporarily isolate the TWI from the Twowire Serial Bus.

In all sleep modes other than Idle mode, the clock system to the TWI is turned off. If the TWEA bit is set, the interface can still acknowledge its own Slave address or the general call address by using the Two-wire Serial Bus clock as a clock source. The part will then wake up from sleep and the TWI will hold the SCL clock will low during the wake up and until the TWINT Flag is cleared (by writing it to one). Further data transmission will be carried out as normal, with the AVR clocks running as normal. Observe that if the AVR is set up with a long start-up time, the SCL line may be held low for a long time, blocking other data transmissions.

Note that the Two-wire Serial Interface Data Register – TWDR does not reflect the last byte present on the bus when waking up from these sleep modes.





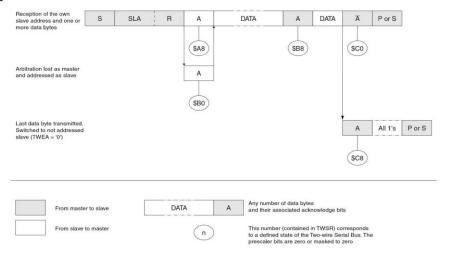
Table 77. Status Codes for Slave Transmitter Mode

Status Code		Applica	tion Softv	vare Resp	oonse		
(TWSR) Prescaler Bits	Status of the Two-wire Serial Bus and Two-wire Serial Interface		To TWCR				
are 0	Hardware	To/from TWDR	STA	STO	TMINT	TWEA	Next Action Taken by TWI Hardware
\$A8	Own SLA+R has been received; ACK has been returned	Load data byte or Load data byte	X X	0	1	0	Last data byte will be transmitted and NOT ACK should be received Data byte will be transmitted and ACK should be re-
							ceived
\$B0	Arbitration lost in SLA+R/W as Master; own SLA+R has been	Load data byte or	Х	0	1	0	Last data byte will be transmitted and NOT ACK should be received
	received; ACK has been returned	Load data byte	Х	0	1	1	Data byte will be transmitted and ACK should be re- ceived
\$B8	Data byte in TWDR has been transmitted; ACK has been	Load data byte or	Х	0	1	0	Last data byte will be transmitted and NOT ACK should be received
	received	Load data byte	Х	0	1	1	Data byte will be transmitted and ACK should be re- ceived
\$C0	Data byte in TWDR has been transmitted; NOT ACK has been received	No TWDR action or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		No TWDR action or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		No TWDR action or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free
\$C8	Last data byte in TWDR has been transmitted (TWEA = "0"); ACK	No TWDR action or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
	has been received	No TWDR action or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		No TWDR action or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free





Figure 93. Formats and States in the Slave Transmitter Mode



Miscellaneous States

There are two status codes that do not correspond to a defined TWI state, see Table 78.

Status \$F8 indicates that no relevant information is available because the TWINT Flag is not set. This occurs between other states, and when the TWI is not involved in a serial transfer.

Status \$00 indicates that a bus error has occurred during a Two-wire Serial Bus transfer. A bus error occurs when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte, or an acknowledge bit. When a bus error occurs, TWINT is set. To recover from a bus error, the TWSTO Flag must set and TWINT must be cleared by writing a logic one to it. This causes the TWI to enter the not addressed Slave mode and to clear the TWSTO Flag (no other bits in TWCR are affected). The SDA and SCL lines are released, and no STOP condition is transmitted.

Table 78. Miscellaneous States

Status Code		Application Software Response						
(TWSR) Prescaler Bits	Status of the Two-wire Serial To TWCR							
are 0		To/from TWDR	STA	STO	TWINT	TWEA	Next Action Taken by TWI Hardware	
\$F8	No relevant state information available; TWINT = "0"	No TWDR action	No TWCR action			Wait or proceed current transfer		
\$00	Bus error due to an illegal START or STOP condition	No TWDR action	0	1	1	Х	Only the internal hardware is affected, no STOP condition is sent on the bus. In all cases, the bus is released and TWSTO is cleared.	

ATMEL



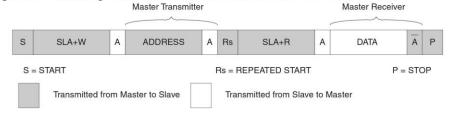
Combining Several TWI Modes

In some cases, several TWI modes must be combined in order to complete the desired action. Consider for example reading data from a serial EEPROM. Typically, such a transfer involves the following steps:

- 1. The transfer must be initiated
- 2. The EEPROM must be instructed what location should be read
- 3. The reading must be performed
- 4. The transfer must be finished

Note that data is transmitted both from Master to Slave and vice versa. The Master must instruct the Slave what location it wants to read, requiring the use of the MT mode. Subsequently, data must be read from the Slave, implying the use of the MR mode. Thus, the transfer direction must be changed. The Master must keep control of the bus during all these steps, and the steps should be carried out as an atomical operation. If this principle is violated in a multi-master system, another Master can alter the data pointer in the EEPROM between steps 2 and 3, and the Master will read the wrong data location. Such a change in transfer direction is accomplished by transmitting a REPEATED START between the transmission of the address byte and reception of the data. After a REPEATED START, the Master keeps ownership of the bus. The following figure shows the flow in this transfer.

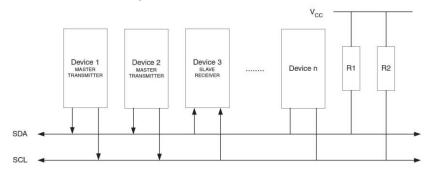
Figure 94. Combining Several TWI Modes to Access a Serial EEPROM



Multi-master Systems and Arbitration

If multiple Masters are connected to the same bus, transmissions may be initiated simultaneously by one or more of them. The TWI standard ensures that such situations are handled in such a way that one of the Masters will be allowed to proceed with the transfer, and that no data will be lost in the process. An example of an arbitration situation is depicted below, where two Masters are trying to transmit data to a Slave Receiver.

Figure 95. An Arbitration Example



ATMEL

199

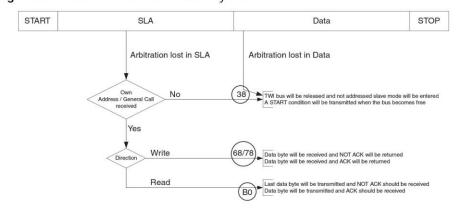


Several different scenarios may arise during arbitration, as described below:

- Two or more Masters are performing identical communication with the same Slave. In this
 case, neither the Slave nor any of the Masters will know about the bus contention.
- Two or more Masters are accessing the same Slave with different data or direction bit. In this
 case, arbitration will occur, either in the READ/WRITE bit or in the data bits. The Masters
 trying to output a one on SDA while another Master outputs a zero will lose the arbitration.
 Losing Masters will switch to not addressed Slave mode or wait until the bus is free and
 transmit a new START condition, depending on application software action.
- Two or more Masters are accessing different Slaves. In this case, arbitration will occur in the SLA bits. Masters trying to output a one on SDA while another Master outputs a zero will lose the arbitration. Masters losing arbitration in SLA will switch to Slave mode to check if they are being addressed by the winning Master. If addressed, they will switch to SR or ST mode, depending on the value of the READ/WRITE bit. If they are not being addressed, they will switch to not addressed Slave mode or wait until the bus is free and transmit a new START condition, depending on application software action.

This is summarized in Figure 96. Possible status values are given in circles.

Figure 96. Possible Status Codes Caused by Arbitration



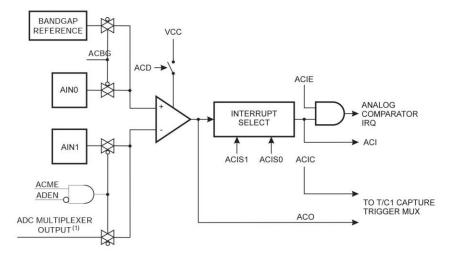




Analog Comparator

The Analog Comparator compares the input values on the positive pin AIN0 and negative pin AIN1. When the voltage on the positive pin AIN0 is higher than the voltage on the negative pin AIN1, the Analog Comparator Output, ACO, is set. The comparator's output can be set to trigger the Timer/Counter1 Input Capture function. In addition, the comparator can trigger a separate interrupt, exclusive to the Analog Comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 97.

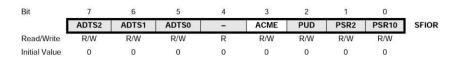
Figure 97. Analog Comparator Block Diagram⁽²⁾



Notes: 1. See Table 80 on page 203.

2. Refer to Figure 1 on page 2 and Table 25 on page 58 for Analog Comparator pin placement.

Special Function IO Register – SFIOR



• Bit 3 - ACME: Analog Comparator Multiplexer Enable

When this bit is written logic one and the ADC is switched off (ADEN in ADCSRA is zero), the ADC multiplexer selects the negative input to the Analog Comparator. When this bit is written logic zero, AIN1 is applied to the negative input of the Analog Comparator. For a detailed description of this bit, see "Analog Comparator Multiplexed Input" on page 203.





Analog Comparator Control and Status Register – ACSR

Bit	7	6	5	4	3	2	1	0	
	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	1
Initial Value	0	0	N/A	0	0	0	0	0	

· Bit 7 - ACD: Analog Comparator Disable

When this bit is written logic one, the power to the Analog Comparator is switched off. This bit can be set at any time to turn off the Analog Comparator. This will reduce power consumption in active and Idle mode. When changing the ACD bit, the Analog Comparator Interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise an interrupt can occur when the bit is changed.

Bit 6 – ACBG: Analog Comparator Bandgap Select

When this bit is set, a fixed bandgap reference voltage replaces the positive input to the Analog Comparator. When this bit is cleared, AIN0 is applied to the positive input of the Analog Comparator. See "Internal Voltage Reference" on page 42.

· Bit 5 - ACO: Analog Comparator Output

The output of the Analog Comparator is synchronized and then directly connected to ACO. The synchronization introduces a delay of 1 - 2 clock cycles.

Bit 4 – ACI: Analog Comparator Interrupt Flag

This bit is set by hardware when a comparator output event triggers the interrupt mode defined by ACIS1 and ACIS0. The Analog Comparator Interrupt routine is executed if the ACIE bit is set and the I-bit in SREG is set. ACI is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ACI is cleared by writing a logic one to the flag.

• Bit 3 - ACIE: Analog Comparator Interrupt Enable

When the ACIE bit is written logic one and the I-bit in the Status Register is set, the Analog Comparator Interrupt is activated. When written logic zero, the interrupt is disabled.

• Bit 2 - ACIC: Analog Comparator Input Capture Enable

When written logic one, this bit enables the Input Capture function in Timer/Counter1 to be triggered by the Analog Comparator. The comparator output is in this case directly connected to the Input Capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter1 Input Capture interrupt. When written logic zero, no connection between the Analog Comparator and the Input Capture function exists. To make the comparator trigger the Timer/Counter1 Input Capture interrupt, the TICIE1 bit in the Timer Interrupt Mask Register (TIMSK) must be set.

Bits 1, 0 – ACIS1, ACIS0: Analog Comparator Interrupt Mode Select

These bits determine which comparator events that trigger the Analog Comparator interrupt. The different settings are shown in Table 79.





Table 79. ACIS1/ACIS0 Settings

ACIS1	ACIS0	Interrupt Mode
0	0	Comparator Interrupt on Output Toggle
0	1	Reserved
1	0	Comparator Interrupt on Falling Output Edge
1	1	Comparator Interrupt on Rising Output Edge

When changing the ACIS1/ACIS0 bits, the Analog Comparator Interrupt must be disabled by clearing its Interrupt Enable bit in the ACSR Register. Otherwise an interrupt can occur when the bits are changed.

Analog Comparator Multiplexed Input It is possible to select any of the ADC7..0 pins to replace the negative input to the Analog Comparator. The ADC multiplexer is used to select this input, and consequently, the ADC must be switched off to utilize this feature. If the Analog Comparator Multiplexer Enable bit (ACME in SFIOR) is set and the ADC is switched off (ADEN in ADCSRA is zero), MUX2..0 in ADMUX select the input pin to replace the negative input to the Analog Comparator, as shown in Table 80. If ACME is cleared or ADEN is set, AIN1 is applied to the negative input to the Analog Comparator.

Table 80. Analog Comparator Multiplexed Input

ACME	ADEN	MUX20	Analog Comparator Negative Input
0	х	xxx	AIN1
1	1	xxx	AIN1
1	0	000	ADC0
1	0	001	ADC1
1	0	010	ADC2
1	0	011	ADC3
1	0	100	ADC4
1	0	101	ADC5
1	0	110	ADC6
1	0	111	ADC7





Analog to Digital Converter

Features

- 10-bit Resolution
- · 0.5 LSB Integral Non-linearity
- · ±2 LSB Absolute Accuracy
- 13 μs- 260 μs Conversion Time
- · Up to 15 kSPS at Maximum Resolution
- 8 Multiplexed Single Ended Input Channels
- · 7 Differential Input Channels
- · 2 Differential Input Channels with Optional Gain of 10x and 200x
- · Optional Left adjustment for ADC Result Readout
- 0 V_{CC} ADC Input Voltage Range
- Selectable 2.56V ADC Reference Voltage
- Free Running or Single Conversion Mode
- ADC Start Conversion by Auto Triggering on Interrupt Sources
- Interrupt on ADC Conversion Complete
- · Sleep Mode Noise Canceler

The ATmega16 features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer which allows 8 single-ended voltage inputs constructed from the pins of Port A. The single-ended voltage inputs refer to 0V (GND).

The device also supports 16 differential voltage input combinations. Two of the differential inputs (ADC1, ADC0 and ADC3, ADC2) are equipped with a programmable gain stage, providing amplification steps of 0 dB (1x), 20 dB (10x), or 46 dB (200x) on the differential input voltage before the A/D conversion. Seven differential analog input channels share a common negative terminal (ADC1), while any other ADC input can be selected as the positive input terminal. If 1x or 10x gain is used, 8-bit resolution can be expected. If 200x gain is used, 7-bit resolution can be expected.

The ADC contains a Sample and Hold circuit which ensures that the input voltage to the ADC is held at a constant level during conversion. A block diagram of the ADC is shown in Figure 98.

The ADC has a separate analog supply voltage pin, AVCC. AVCC must not differ more than $\pm 0.3V$ from V_{CC} . See the paragraph "ADC Noise Canceler" on page 211 on how to connect this pin.

Internal reference voltages of nominally 2.56V or AVCC are provided On-chip. The voltage reference may be externally decoupled at the AREF pin by a capacitor for better noise performance.





ADC CONVERSION COMPLETE IRQ INTERRUPT ADTS[2:0 8-BIT DATA BUS ADIF ADIE ADC MULTIPLEXER SELECT (ADMUX) ADC CTRL. & STATUS REGISTER (ADCSRA MUX3 MUX2 MUX1 MUX DECODER PRESCALER CONVERSION LOGIC INTERNAL 2.56V REFERENCE 10-BIT DAC BANDGAP REFERENCE SINGLE ENDED / DIFFERENTIAL SELECTION ADCE ADC4 ADC1

Figure 98. Analog to Digital Converter Block Schematic

Operation

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents GND and the maximum value represents the voltage on the AREF pin minus 1 LSB. Optionally, AVCC or an internal 2.56V reference voltage may be connected to the AREF pin by writing to the REFSn bits in the ADMUX Register. The internal voltage reference may thus be decoupled by an external capacitor at the AREF pin to improve noise immunity.

The analog input channel and differential gain are selected by writing to the MUX bits in ADMUX. Any of the ADC input pins, as well as GND and a fixed bandgap voltage reference, can be selected as single ended inputs to the ADC. A selection of ADC input pins can be selected as positive and negative inputs to the differential gain amplifier.

If differential channels are selected, the differential gain stage amplifies the voltage difference between the selected input channel pair by the selected gain factor. This amplified value then



205



becomes the analog input to the ADC. If single ended channels are used, the gain amplifier is bypassed altogether.

The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSRA. Voltage reference and input channel selections will not go into effect until ADEN is set. The ADC does not consume power when ADEN is cleared, so it is recommended to switch off the ADC before entering power saving sleep modes.

The ADC generates a 10-bit result which is presented in the ADC Data Registers, ADCH and ADCL. By default, the result is presented right adjusted, but can optionally be presented left adjusted by setting the ADLAR bit in ADMUX.

If the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH, to ensure that the content of the Data Registers belongs to the same conversion. Once ADCL is read, ADC access to Data Registers is blocked. This means that if ADCL has been read, and a conversion completes before ADCH is read, neither register is updated and the result from the conversion is lost. When ADCH is read, ADC access to the ADCH and ADCL Registers is re-enabled.

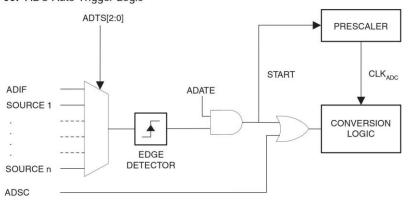
The ADC has its own interrupt which can be triggered when a conversion completes. When ADC access to the Data Registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

Starting a Conversion

A single conversion is started by writing a logical one to the ADC Start Conversion bit, ADSC. This bit stays high as long as the conversion is in progress and will be cleared by hardware when the conversion is completed. If a different data channel is selected while a conversion is in progress, the ADC will finish the current conversion before performing the channel change.

Alternatively, a conversion can be triggered automatically by various sources. Auto Triggering is enabled by setting the ADC Auto Trigger Enable bit, ADATE in ADCSRA. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in SFIOR (see description of the ADTS bits for a list of the trigger sources). When a positive edge occurs on the selected trigger signal, the ADC prescaler is reset and a conversion is started. This provides a method of starting conversions at fixed intervals. If the trigger signal still is set when the conversion completes, a new conversion will not be started. If another positive edge occurs on the trigger signal during conversion, the edge will be ignored. Note that an Interrupt Flag will be set even if the specific interrupt is disabled or the global interrupt enable bit in SREG is cleared. A conversion can thus be triggered without causing an interrupt. However, the Interrupt Flag must be cleared in order to trigger a new conversion at the next interrupt event.

Figure 99. ADC Auto Trigger Logic



ATMEL

206

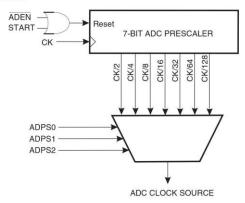


Using the ADC Interrupt Flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in Free Running mode, constantly sampling and updating the ADC Data Register. The first conversion must be started by writing a logical one to the ADSC bit in ADCSRA. In this mode the ADC will perform successive conversions independently of whether the ADC Interrupt Flag, ADIF is cleared or not.

If Auto Triggering is enabled, single conversions can be started by writing ADSC in ADCSRA to one. ADSC can also be used to determine if a conversion is in progress. The ADSC bit will be read as one during a conversion, independently of how the conversion was started.

Prescaling and Conversion Timing

Figure 100. ADC Prescaler



By default, the successive approximation circuitry requires an input clock frequency between 50 kHz and 200 kHz to get maximum resolution. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be higher than 200 kHz to get a higher sample rate.

The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100 kHz. The prescaling is set by the ADPS bits in ADCSRA. The prescaler starts counting from the moment the ADC is switched on by setting the ADEN bit in ADCSRA. The prescaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

When initiating a single ended conversion by setting the ADSC bit in ADCSRA, the conversion starts at the following rising edge of the ADC clock cycle. See "Differential Gain Channels" on page 209 for details on differential conversion timing.

A normal conversion takes 13 ADC clock cycles. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 25 ADC clock cycles in order to initialize the analog circuitry.

The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 13.5 ADC clock cycles after the start of a first conversion. When a conversion is complete, the result is written to the ADC Data Registers, and ADIF is set. In single conversion mode, ADSC is cleared simultaneously. The software may then set ADSC again, and a new conversion will be initiated on the first rising ADC clock edge.

When Auto Triggering is used, the prescaler is reset when the trigger event occurs. This assures a fixed delay from the trigger event to the start of conversion. In this mode, the sample-and-hold takes place 2 ADC clock cycles after the rising edge on the trigger source signal. Three additional CPU clock cycles are used for synchronization logic. When using Differential mode, along with Auto triggering from a source other than the ADC Conversion Complete, each conversion



207



will require 25 ADC clocks. This is because the ADC must be disabled and re-enabled after every conversion.

In Free Running mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. For a summary of conversion times, see Table 81.

Figure 101. ADC Timing Diagram, First Conversion (Single Conversion Mode)

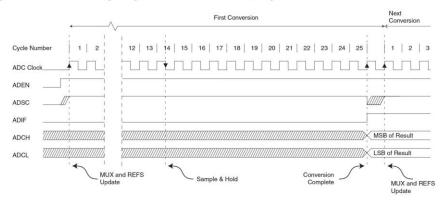


Figure 102. ADC Timing Diagram, Single Conversion

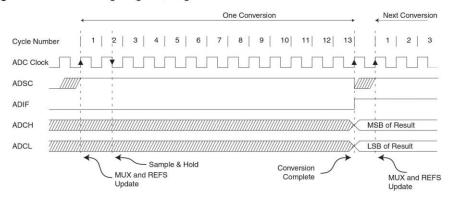






Figure 103. ADC Timing Diagram, Auto Triggered Conversion

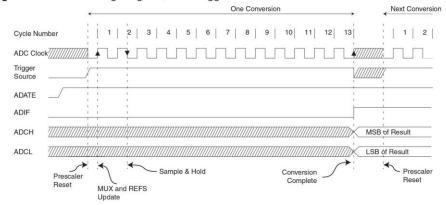


Figure 104. ADC Timing Diagram, Free Running Conversion

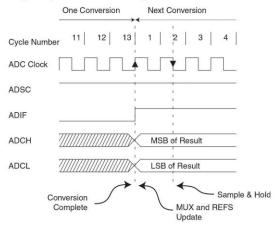


Table 81. ADC Conversion Time

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	13.5	25
Normal conversions, single ended	1.5	13
Auto Triggered conversions	2	13.5
Normal conversions, differential	1.5/2.5	13/14

Differential Gain Channels When using differential gain channels, certain aspects of the conversion need to be taken into consideration.

Differential conversions are synchronized to the internal clock CK_{ADC2} equal to half the ADC clock. This synchronization is done automatically by the ADC interface in such a way that the sample-and-hold occurs at a specific phase of CK_{ADC2} . A conversion initiated by the user (that is,

AIMEL

209



all single conversions, and the first free running conversion) when CK_{ADC2} is low will take the same amount of time as a single ended conversion (13 ADC clock cycles from the next prescaled clock cycle). A conversion initiated by the user when CK_{ADC2} is high will take 14 ADC clock cycles due to the synchronization mechanism. In Free Running mode, a new conversion is initiated immediately after the previous conversion completes, and since CK_{ADC2} is high at this time, all automatically started (that is, all but the first) free running conversions will take 14 ADC clock cycles.

The gain stage is optimized for a bandwidth of 4 kHz at all gain settings. Higher frequencies may be subjected to non-linear amplification. An external low-pass filter should be used if the input signal contains higher frequency components than the gain stage bandwidth. Note that the ADC clock frequency is independent of the gain stage bandwidth limitation. For example, the ADC clock period may be 6 μ s, allowing a channel to be sampled at 12 kSPS, regardless of the bandwidth of this channel.

If differential gain channels are used and conversions are started by Auto Triggering, the ADC must be switched off between conversions. When Auto Triggering is used, the ADC prescaler is reset before the conversion is started. Since the gain stage is dependent of a stable ADC clock prior to the conversion, this conversion will not be valid. By disabling and then re-enabling the ADC between each conversion (writing ADEN in ADCSRA to "0" then to "1"), only extended conversions are performed. The result from the extended conversions will be valid. See "Prescaling and Conversion Timing" on page 207 for timing details.

Changing Channel or Reference Selection

The MUXn and REFS1:0 bits in the ADMUX Register are single buffered through a temporary register to which the CPU has random access. This ensures that the channels and reference selection only takes place at a safe point during the conversion. The channel and reference selection is continuously updated until a conversion is started. Once the conversion starts, the channel and reference selection is locked to ensure a sufficient sampling time for the ADC. Continuous updating resumes in the last ADC clock cycle before the conversion completes (ADIF in ADCSRA is set). Note that the conversion starts on the following rising ADC clock edge after ADSC is written. The user is thus advised not to write new channel or reference selection values to ADMUX until one ADC clock cycle after ADSC is written.

If Auto Triggering is used, the exact time of the triggering event can be indeterministic. Special care must be taken when updating the ADMUX Register, in order to control which conversion will be affected by the new settings.

If both ADATE and ADEN is written to one, an interrupt event can occur at any time. If the ADMUX Register is changed in this period, the user cannot tell if the next conversion is based on the old or the new settings. ADMUX can be safely updated in the following ways:

- When ADATE or ADEN is cleared.
- During conversion, minimum one ADC clock cycle after the trigger event.
- 3. After a conversion, before the Interrupt Flag used as trigger source is cleared.

When updating ADMUX in one of these conditions, the new settings will affect the next ADC conversion.

Special care should be taken when changing differential channels. Once a differential channel has been selected, the gain stage may take as much as 125 μ s to stabilize to the new value. Thus conversions should not be started within the first 125 μ s after selecting a new differential channel. Alternatively, conversion results obtained within this period should be discarded.

The same settling time should be observed for the first differential conversion after changing ADC reference (by changing the REFS1:0 bits in ADMUX).





ADC Input Channels

When changing channel selections, the user should observe the following guidelines to ensure that the correct channel is selected:

In Single Conversion mode, always select the channel before starting the conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the conversion to complete before changing the channel selection.

In Free Running mode, always select the channel before starting the first conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the first conversion to complete, and then change the channel selection. Since the next conversion has already started automatically, the next result will reflect the previous channel selection. Subsequent conversions will reflect the new channel selection.

When switching to a differential gain channel, the first conversion result may have a poor accuracy due to the required settling time for the automatic offset cancellation circuitry. The user should preferably disregard the first conversion result.

ADC Voltage Reference

The reference voltage for the ADC (V_{REF}) indicates the conversion range for the ADC. Single ended channels that exceed V_{REF} will result in codes close to 0x3FF. V_{REF} can be selected as either AVCC, internal 2.56V reference, or external AREF pin.

AVCC is connected to the ADC through a passive switch. The internal 2.56V reference is generated from the internal bandgap reference (V_{BG}) through an internal amplifier. In either case, the external AREF pin is directly connected to the ADC, and the reference voltage can be made more immune to noise by connecting a capacitor between the AREF pin and ground. V_{REF} can also be measured at the AREF pin with a high impedant voltmeter. Note that V_{REF} is a high impedant source, and only a capacitive load should be connected in a system.

If the user has a fixed voltage source connected to the AREF pin, the user may not use the other reference voltage options in the application, as they will be shorted to the external voltage. If no external voltage is applied to the AREF pin, the user may switch between AVCC and 2.56V as reference selection. The first ADC conversion result after switching reference voltage source may be inaccurate, and the user is advised to discard this result.

If differential channels are used, the selected reference should not be closer to AVCC than indicated in Table 122 on page 297.

ADC Noise Canceler

The ADC features a noise canceler that enables conversion during sleep mode to reduce noise induced from the CPU core and other I/O peripherals. The noise canceler can be used with ADC Noise Reduction and Idle mode. To make use of this feature, the following procedure should be used:

- Make sure that the ADC is enabled and is not busy converting. Single Conversion Mode must be selected and the ADC conversion complete interrupt must be enabled.
- Enter ADC Noise Reduction mode (or Idle mode). The ADC will start a conversion once the CPU has been halted.
- 3. If no other interrupts occur before the ADC conversion completes, the ADC interrupt will wake up the CPU and execute the ADC Conversion Complete interrupt routine. If another interrupt wakes up the CPU before the ADC conversion is complete, that interrupt will be executed, and an ADC Conversion Complete interrupt request will be generated when the ADC conversion completes. The CPU will remain in active mode until a new sleep command is executed.

Note that the ADC will not be automatically turned off when entering other sleep modes than Idle mode and ADC Noise Reduction mode. The user is advised to write zero to ADEN before entering such sleep modes to avoid excessive power consumption. If the ADC is enabled in such





sleep modes and the user wants to perform differential conversions, the user is advised to switch the ADC off and on after waking up from sleep to prompt an extended conversion to get a valid result.

Analog Input Circuitry

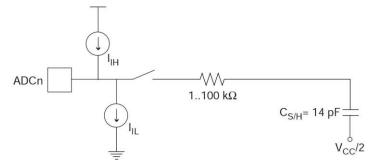
The Analog Input Circuitry for single ended channels is illustrated in Figure 105. An analog source applied to ADCn is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is selected as input for the ADC. When the channel is selected, the source must drive the S/H capacitor through the series resistance (combined resistance in the input path).

The ADC is optimized for analog signals with an output impedance of approximately 10 k Ω or less. If such a source is used, the sampling time will be negligible. If a source with higher impedance is used, the sampling time will depend on how long time the source needs to charge the S/H capacitor, with can vary widely. The user is recommended to only use low impedant sources with slowly varying signals, since this minimizes the required charge transfer to the S/H capacitor.

If differential gain channels are used, the input circuitry looks somewhat different, although source impedances of a few hundred $k\Omega$ or less is recommended.

Signal components higher than the Nyquist frequency (f_{ADC}/2) should not be present for either kind of channels, to avoid distortion from unpredictable signal convolution. The user is advised to remove high frequency components with a low-pass filter before applying the signals as inputs to the ADC.

Figure 105. Analog Input Circuitry



Analog Noise Canceling Techniques

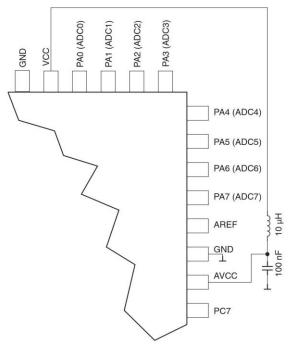
Digital circuitry inside and outside the device generates EMI which might affect the accuracy of analog measurements. If conversion accuracy is critical, the noise level can be reduced by applying the following techniques:

- Keep analog signal paths as short as possible. Keep them well away from highspeed switching digital tracks.
- The AVCC pin on the device should be connected to the digital V_{CC} supply voltage via an LC network as shown in Figure 106.
- 3. Use the ADC noise canceler function to reduce induced noise from the CPU.
- 4. If any ADC port pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.





Figure 106. ADC Power Connections



Offset Compensation Schemes

The gain stage has a built-in offset cancellation circuitry that nulls the offset of differential measurements as much as possible. The remaining offset in the analog path can be measured directly by selecting the same channel for both differential inputs. This offset residue can be then subtracted in software from the measurement results. Using this kind of software based offset correction, offset on any channel can be reduced below one LSB.

ADC Accuracy Definitions

An n-bit single-ended ADC converts a voltage linearly between GND and V_{REF} in 2^n steps (LSBs). The lowest code is read as 0, and the highest code is read as 2^{n} -1.

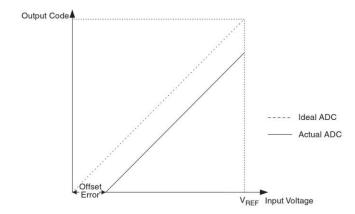
Several parameters describe the deviation from the ideal behavior:

 Offset: The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB.



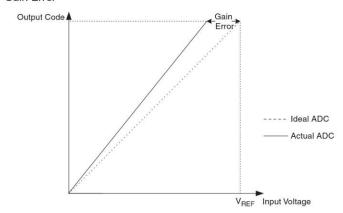


Figure 107. Offset Error



 Gain Error: After adjusting for offset, the Gain Error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB

Figure 108. Gain Error

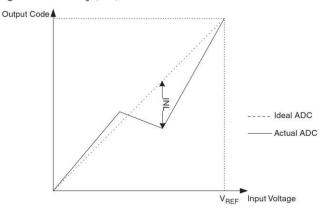


Integral Non-linearity (INL): After adjusting for offset and gain error, the INL is the maximum
deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0
LSB.



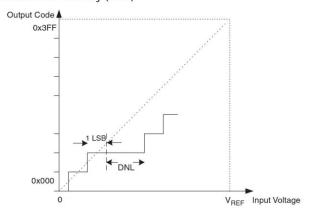


Figure 109. Integral Non-linearity (INL)



 Differential Non-linearity (DNL): The maximum deviation of the actual code width (the interval between two adjacent transitions) from the ideal code width (1 LSB). Ideal value: 0 LSB.

Figure 110. Differential Non-linearity (DNL)



- Quantization Error: Due to the quantization of the input voltage into a finite number of codes, a range of input voltages (1 LSB wide) will code to the same value. Always ±0.5 LSB.
- Absolute Accuracy: The maximum deviation of an actual (unadjusted) transition compared to an ideal transition for any code. This is the compound effect of Offset, Gain Error, Differential Error, Non-linearity, and Quantization Error. Ideal value: ±0.5 LSB.





ADC Conversion Result

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC Result Registers (ADCL, ADCH).

For single ended conversion, the result is

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

where V_{IN} is the voltage on the selected input pin and V_{REF} the selected voltage reference (see Table 83 on page 217 and Table 84 on page 218). 0x000 represents ground, and 0x3FF represents the selected reference voltage minus one LSB.

If differential channels are used, the result is

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot GAIN \cdot 512}{V_{REF}}$$

where V_{POS} is the voltage on the positive input pin, V_{NEG} the voltage on the negative input pin, GAIN the selected gain factor, and V_{REF} the selected voltage reference. The result is presented in two's complement form, from 0x200 (-512d) through 0x1FF (+511d). Note that if the user wants to perform a quick polarity check of the results, it is sufficient to read the MSB of the result (ADC9 in ADCH). If this bit is one, the result is negative, and if this bit is zero, the result is positive. Figure 111 shows the decoding of the differential input range.

Table 82 shows the resulting output codes if the differential input channel pair (ADCn - ADCm) is selected with a gain of GAIN and a reference voltage of $V_{\rm REF}$.

Figure 111. Differential Measurement Range

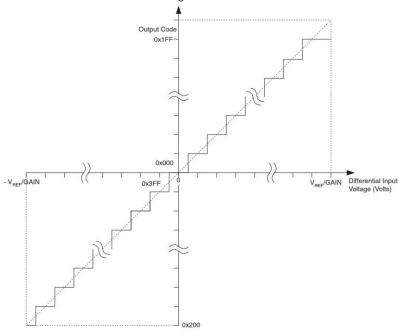




Table 82. Correlation between Input Voltage and Output Codes

V _{ADCn}	Read code	Corresponding Decimal Value
V _{ADCm} + V _{REF} /GAIN	0x1FF	511
V _{ADCm} + 511/512 V _{REF} /GAIN	0x1FF	511
V _{ADCm} + 510/512 V _{REF} /GAIN	0x1FE	510
V _{ADCm} + 1/512 V _{REF} /GAIN	0x001	1
V_{ADCm}	0x000	0
V _{ADCm} - 1/512 V _{REF} /GAIN	0x3FF	-1
V _{ADCm} - 511/512 V _{REF} /GAIN	0x201	-511
V _{ADCm} - V _{REF} /GAIN	0x200	-512

Example:

ADMUX = 0xED (ADC3 - ADC2, 10x gain, 2.56V reference, left adjusted result)

Voltage on ADC3 is 300 mV, voltage on ADC2 is 500 mV.

 $ADCR = 512 \times 10 \times (300 - 500) / 2560 = -400 = 0x270$

ADCL will thus read 0x00, and ADCH will read 0x9C. Writing zero to ADLAR right adjusts the result: ADCL = 0x70, ADCH = 0x02.

ADC Multiplexer Selection Register – ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	•
Initial Value	0	0	0	0	0	0	0	0	

· Bit 7:6 - REFS1:0: Reference Selection Bits

These bits select the voltage reference for the ADC, as shown in Table 83. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 83. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

· Bit 5 - ADLAR: ADC Left Adjust Result

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conver-





sions. For a complete description of this bit, see "The ADC Data Register – ADCL and ADCH" on page 220.

• Bits 4:0 - MUX4:0: Analog Channel and Gain Selection Bits

The value of these bits selects which combination of analog inputs are connected to the ADC. These bits also select the gain for the differential channels. See Table 84 for details. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set).

Table 84. Input Channel and Gain Selections

MUX40	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
00000	ADC0		`	
00001	ADC1			
00010	ADC2			
00011	ADC3	N/A		
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000		ADC0	ADC0	10x
01001		ADC1	ADC0	10x
01010		ADC0	ADC0	200x
01011		ADC1	ADC0	200x
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	10x
01110		ADC2	ADC2	200x
01111		ADC3	ADC2	200x
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	1x
10010	N/A	ADC2	ADC1	1x
10011		ADC3	ADC1	1x
10100		ADC4	ADC1	1x
10101		ADC5	ADC1	1x
10110		ADC6	ADC1	1x
10111		ADC7	ADC1	1x
11000		ADC0	ADC2	1x
11001		ADC1	ADC2	1x
11010		ADC2	ADC2	1x
11011		ADC3	ADC2	1x
11100		ADC4	ADC2	1x





Table 84. Input Channel and Gain Selections (Continued)

MUX40	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
11101		ADC5	ADC2	1x
11110	1.22V (V _{BG})	N/A		
11111	0 V (GND)			

ADC Control and Status Register A – ADCSRA

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	•
Initial Value	0	0	0	0	0	0	0	0	

· Bit 7 - ADEN: ADC Enable

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

· Bit 6 - ADSC: ADC Start Conversion

In Single Conversion mode, write this bit to one to start each conversion. In Free Running Mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

• Bit 5 - ADATE: ADC Auto Trigger Enable

When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in SFIOR.

• Bit 4 - ADIF: ADC Interrupt Flag

This bit is set when an ADC conversion completes and the Data Registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

· Bit 3 - ADIE: ADC Interrupt Enable

When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

· Bits 2:0 - ADPS2:0: ADC Prescaler Select Bits

These bits determine the division factor between the XTAL frequency and the input clock to the ADC.

AIMEL



Table 85. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

The ADC Data Register – ADCL and ADCH

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
	-	-	-	-	-	-	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	-	-	-	-	-	-	ADCL
	7	6	5	4	3	2	1	0	•
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When an ADC conversion is complete, the result is found in these two registers. If differential channels are used, the result is presented in two's complement form.

When ADCL is read, the ADC Data Register is not updated until ADCH is read. Consequently, if the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH.

The ADLAR bit in ADMUX, and the MUXn bits in ADMUX affect the way the result is read from the registers. If ADLAR is set, the result is left adjusted. If ADLAR is cleared (default), the result is right adjusted.

· ADC9:0: ADC Conversion Result

These bits represent the result from the conversion, as detailed in "ADC Conversion Result" on page 216.



220

2466T-AVR-07/10



Special FunctionIO Register – SFIOR

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	-	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	Li iir
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7:5 - ADTS2:0: ADC Auto Trigger Source

If ADATE in ADCSRA is written to one, the value of these bits selects which source will trigger an ADC conversion. If ADATE is cleared, the ADTS2:0 settings will have no effect. A conversion will be triggered by the rising edge of the selected Interrupt Flag. Note that switching from a trigger source that is cleared to a trigger source that is set, will generate a positive edge on the trigger signal. If ADEN in ADCSRA is set, this will start a conversion. Switching to Free Running mode (ADTS[2:0]=0) will not cause a trigger event, even if the ADC Interrupt Flag is set.

Table 86. ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

· Bit 4 - Res: Reserved Bit

This bit is reserved for future use. To ensure compatibility with future devices, this bit must be written to zero when SFIOR is written.





JTAG Interface and On-chip Debug System

Features

- · JTAG (IEEE std. 1149.1 Compliant) Interface
- · Boundary-scan Capabilities According to the IEEE std. 1149.1 (JTAG) Standard
- · Debugger Access to:
 - All Internal Peripheral Units
 - Internal and External RAM
 - The Internal Register File
 - Program Counter
 - EEPROM and Flash Memories
 - Extensive On-chip Debug Support for Break Conditions, Including
 - AVR Break Instruction
 - Break on Change of Program Memory Flow
 - Single Step Break
 - Program Memory Breakpoints on Single Address or Address Range
 - Data Memory Breakpoints on Single Address or Address Range
- · Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- On-chip Debugging Supported by AVR Studio[®]

Overview

The AVR IEEE std. 1149.1 compliant JTAG interface can be used for

- · Testing PCBs by using the JTAG Boundary-scan capability
- · Programming the non-volatile memories, Fuses and Lock bits
- · On-chip Debugging

A brief description is given in the following sections. Detailed descriptions for Programming via the JTAG interface, and using the Boundary-scan Chain can be found in the sections "Programming via the JTAG Interface" on page 278 and "IEEE 1149.1 (JTAG) Boundary-scan" on page 228, respectively. The On-chip Debug support is considered being private JTAG instructions, and distributed within ATMEL and to selected third party vendors only.

Figure 112 shows a block diagram of the JTAG interface and the On-chip Debug system. The TAP Controller is a state machine controlled by the TCK and TMS signals. The TAP Controller selects either the JTAG Instruction Register or one of several Data Registers as the scan chain (Shift Register) between the TDI input and TDO output. The Instruction Register holds JTAG instructions controlling the behavior of a Data Register.

The ID-Register, Bypass Register, and the Boundary-scan Chain are the Data Registers used for board-level testing. The JTAG Programming Interface (actually consisting of several physical and virtual Data Registers) is used for JTAG Serial Programming via the JTAG interface. The Internal Scan Chain and Break Point Scan Chain are used for On-chip Debugging only.

Test Access Port – TAP

The JTAG interface is accessed through four of the AVR's pins. In JTAG terminology, these pins constitute the Test Access Port – TAP. These pins are:

- TMS: Test Mode Select. This pin is used for navigating through the TAP-controller state machine.
- TCK: Test Clock. JTAG operation is synchronous to TCK.
- TDI: Test Data In. Serial input data to be shifted in to the Instruction Register or Data Register (Scan Chains).
- TDO: Test Data Out. Serial output data from Instruction register or Data Register.



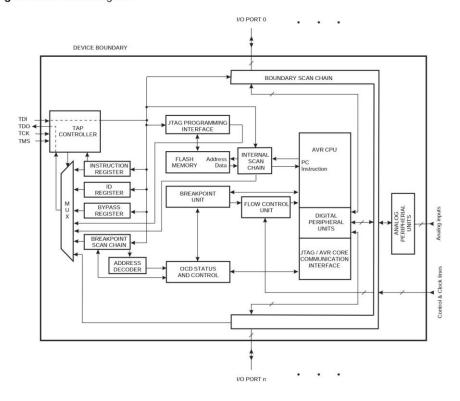


The IEEE std. 1149.1 also specifies an optional TAP signal; TRST – Test ReSeT – which is not provided.

When the JTAGEN fuse is unprogrammed, these four TAP pins are normal port pins and the TAP controller is in reset. When programmed and the JTD bit in MCUCSR is cleared, the TAP input signals are internally pulled high and the JTAG is enabled for Boundary-scan and programming. In this case, the TAP output pin (TDO) is left floating in states where the JTAG TAP controller is not shifting data, and must therefore be connected to a pull-up resistor or other hardware having pull-ups (for instance the TDI-input of the next device in the scan chain). The device is shipped with this fuse programmed.

For the On-chip Debug system, in addition to the JTAG interface pins, the $\overline{\text{RESET}}$ pin is monitored by the debugger to be able to detect external reset sources. The debugger can also pull the $\overline{\text{RESET}}$ pin low to reset the whole system, assuming only open collectors on the reset line are used in the application.

Figure 112. Block Diagram







——— ATmega16(L)

Test-Logic-Reset 0 Select-DR Scan Select-IR Scan Run-Test/Idle 0 0 Capture-DR Capture-IR 0 0 Shift-DR 0 Shift-IR 0 (Exit1-DR Exit1-IR 0 0 Pause-DR 0 0 Pause-IR 0 Exit2-DR Exit2-IR 1 1 Update-DR Update-IR 0 0

Figure 113. TAP Controller State Diagram

TAP Controller

The TAP controller is a 16-state finite state machine that controls the operation of the Boundary-scan circuitry, JTAG programming circuitry, or On-chip Debug system. The state transitions depicted in Figure 113 depend on the signal present on TMS (shown adjacent to each state transition) at the time of the rising edge at TCK. The initial state after a Power-On Reset is Test-Logic-Reset.

As a definition in this document, the LSB is shifted in and out first for all Shift Registers.

Assuming Run-Test/Idle is the present state, a typical scenario for using the JTAG interface is:

 At the TMS input, apply the sequence 1, 1, 0, 0 at the rising edges of TCK to enter the Shift Instruction Register – Shift-IR state. While in this state, shift the four bits of the JTAG instructions into the JTAG Instruction Register from the TDI input at the rising edge of TCK. The TMS input must be held low during input of the 3 LSBs in order to remain in the Shift-IR state. The MSB of the instruction is shifted in when this state is left by setting TMS high. While the instruction is shifted in from the TDI pin, the captured IR-state 0x01 is shifted out

AIMEL

224

2466T-AVR-07/10



on the TDO pin. The JTAG Instruction selects a particular Data Register as path between TDI and TDO and controls the circuitry surrounding the selected Data Register.

- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. The instruction is latched onto the parallel output from the Shift Register path in the Update-IR state. The Exit-IR, Pause-IR, and Exit2-IR states are only used for navigating the state machine.
- At the TMS input, apply the sequence 1, 0, 0 at the rising edges of TCK to enter the Shift
 Data Register Shift-DR state. While in this state, upload the selected Data Register
 (selected by the present JTAG instruction in the JTAG Instruction Register) from the TDI
 input at the rising edge of TCK. In order to remain in the Shift-DR state, the TMS input must
 be held low during input of all bits except the MSB. The MSB of the data is shifted in when
 this state is left by setting TMS high. While the Data Register is shifted in from the TDI pin,
 the parallel inputs to the Data Register captured in the Capture-DR state is shifted out on the
 TDO pin.
- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. If the selected Data Register has a latched parallel-output, the latching takes place in the Update-DR state. The Exit-DR, Pause-DR, and Exit2-DR states are only used for navigating the state machine.

As shown in the state diagram, the Run-Test/Idle state need not be entered between selecting JTAG instruction and using Data Registers, and some JTAG instructions may select certain functions to be performed in the Run-Test/Idle, making it unsuitable as an Idle state.

Note: Independent of the initial state of the TAP Controller, the Test-Logic-Reset state can always be entered by holding TMS high for five TCK clock periods.

For detailed information on the JTAG specification, refer to the literature listed in "Bibliography" on page 227.

Using the Boundary-scan Chain

A complete description of the Boundary-scan capabilities are given in the section "IEEE 1149.1 (JTAG) Boundary-scan" on page 228.

Using the On-chip Debug System

As shown in Figure 112, the hardware support for On-chip Debugging consists mainly of:

- A scan chain on the interface between the internal AVR CPU and the internal peripheral units
- Break Point unit
- Communication interface between the CPU and JTAG system

All read or modify/write operations needed for implementing the Debugger are done by applying AVR instructions via the internal AVR CPU Scan Chain. The CPU sends the result to an I/O memory mapped location which is part of the communication interface between the CPU and the JTAG system.

The Break Point Unit implements Break on Change of Program Flow, Single Step Break, 2 Program Memory Break Points, and 2 combined Break Points. Together, the 4 Break Points can be configured as either:

- · 4 single Program Memory Break Points
- · 3 Single Program Memory Break Point + 1 single Data Memory Break Point
- 2 single Program Memory Break Points + 2 single Data Memory Break Points
- 2 single Program Memory Break Points + 1 Program Memory Break Point with mask ("range Break Point")
- 2 single Program Memory Break Points + 1 Data Memory Break Point with mask ("range Break Point")





A debugger, like the AVR Studio, may however use one or more of these resources for its internal purpose, leaving less flexibility to the end-user.

A list of the On-chip Debug Specific JTAG instructions is given in "On-chip Debug Specific JTAG Instructions" on page 226.

The JTAGEN Fuse must be programmed to enable the JTAG Test Access Port. In addition, the OCDEN Fuse must be programmed and no Lock bits must be set for the On-chip Debug system to work. As a security feature, the On-chip Debug system is disabled when *any* Lock bits are set. Otherwise, the On-chip Debug system would have provided a back-door into a secured device.

The AVR JTAG ICE from Atmel is a powerful development tool for On-chip Debugging of all AVR 8-bit RISC Microcontrollers with IEEE 1149.1 compliant JTAG interface. The JTAG ICE and the AVR Studio user interface give the user complete control of the internal resources of the microcontroller, helping to reduce development time by making debugging easier. The JTAG ICE performs real-time emulation of the microcontroller while it is running in a target system.

Please refer to the Support Tools section on the AVR pages on www.atmel.com for a full description of the AVR JTEG ICE. AVR Studio can be downloaded free from Software section on the same web site.

All necessary execution commands are available in AVR Studio, both on source level and on disassembly level. The user can execute the program, single step through the code either by tracing into or stepping over functions, step out of functions, place the cursor on a statement and execute until the statement is reached, stop the execution, and reset the execution target. In addition, the user can have an unlimited number of code breakpoints (using the BREAK instruction) and up to two data memory breakpoints, alternatively combined as a mask (range) Break Point.

On-chip Debug Specific JTAG Instructions	The On-chip Debug support is considered being private JTAG instructions, and distributed within ATMEL and to selected third party vendors only. Instruction opcodes are listed for reference.
PRIVATE0; \$8	Private JTAG instruction for accessing On-chip Debug system.
PRIVATE1; \$9	Private JTAG instruction for accessing On-chip Debug system.
PRIVATE2; \$A	Private JTAG instruction for accessing On-chip Debug system.



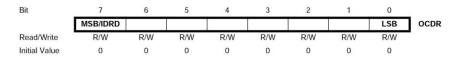
Private JTAG instruction for accessing On-chip Debug system.

PRIVATE3; \$B



On-chip Debug Related Register in I/O Memory

On-chip Debug Register – OCDR



The OCDR Register provides a communication channel from the running program in the micro-controller to the debugger. The CPU can transfer a byte to the debugger by writing to this location. At the same time, an Internal Flag; I/O Debug Register Dirty – IDRD – is set to indicate to the debugger that the register has been written. When the CPU reads the OCDR Register the 7 LSB will be from the OCDR Register, while the MSB is the IDRD bit. The debugger clears the IDRD bit when it has read the information.

In some AVR devices, this register is shared with a standard I/O location. In this case, the OCDR Register can only be accessed if the OCDEN Fuse is programmed, and the debugger enables access to the OCDR Register. In all other cases, the standard I/O location is accessed.

Refer to the debugger documentation for further information on how to use this register.

Using the JTAG Programming Capabilities

Programming of AVR parts via JTAG is performed via the 4-pin JTAG port, TCK, TMS, TDI and TDO. These are the only pins that need to be controlled/observed to perform JTAG programming (in addition to power pins). It is not required to apply 12V externally. The JTAGEN Fuse must be programmed and the JTD bit in the MCUSR Register must be cleared to enable the JTAG Test Access Port.

The JTAG programming capability supports:

- · Flash programming and verifying
- · EEPROM programming and verifying
- Fuse programming and verifying
- · Lock bit programming and verifying

The Lock bit security is exactly as in Parallel Programming mode. If the Lock bits LB1 or LB2 are programmed, the OCDEN Fuse cannot be programmed unless first doing a chip erase. This is a security feature that ensures no back-door exists for reading out the content of a secured device.

The details on programming through the JTAG interface and programming specific JTAG instructions are given in the section "Programming via the JTAG Interface" on page 278.

Bibliography

For more information about general Boundary-scan, the following literature can be consulted:

- IEEE: IEEE Std. 1149.1-1990. IEEE Standard Test Access Port and Boundary-scan Architecture, IEEE, 1993
- Colin Maunder: The Board Designers Guide to Testable Logic Circuits, Addison-Wesley, 1992





IEEE 1149.1 (JTAG) Boundary-scan

Features

- · JTAG (IEEE std. 1149.1 Compliant) Interface
- · Boundary-scan Capabilities According to the JTAG Standard
- · Full Scan of all Port Functions as well as Analog Circuitry having Off-chip Connections
- · Supports the Optional IDCODE Instruction
- Additional Public AVR_RESET Instruction to Reset the AVR

System Overview

The Boundary-scan chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having Off-chip connections. At system level, all ICs having JTAG capabilities are connected serially by the TDI/TDO signals to form a long Shift Register. An external controller sets up the devices to drive values at their output pins, and observe the input values received from other devices. The controller compares the received data with the expected result. In this way, Boundary-scan provides a mechanism for testing interconnections and integrity of components on Printed Circuits Boards by using the four TAP signals only.

The four IEEE 1149.1 defined mandatory JTAG instructions IDCODE, BYPASS, SAMPLE/PRELOAD, and EXTEST, as well as the AVR specific public JTAG instruction AVR_RESET can be used for testing the Printed Circuit Board. Initial scanning of the Data Register path will show the ID-code of the device, since IDCODE is the default JTAG instruction. It may be desirable to have the AVR device in Reset during Test mode. If not reset, inputs to the device may be determined by the scan operations, and the internal software may be in an undetermined state when exiting the Test mode. Entering reset, the outputs of any Port Pin will instantly enter the high impedance state, making the HIGHZ instruction redundant. If needed, the BYPASS instruction can be issued to make the shortest possible scan chain through the device. The device can be set in the reset state either by pulling the external RESET pin low, or issuing the AVR_RESET instruction with appropriate setting of the Reset Data Register.

The EXTEST instruction is used for sampling external pins and loading output pins with data. The data from the output latch will be driven out on the pins as soon as the EXTEST instruction is loaded into the JTAG IR-Register. Therefore, the SAMPLE/PRELOAD should also be used for setting initial values to the scan ring, to avoid damaging the board when issuing the EXTEST instruction for the first time. SAMPLE/PRELOAD can also be used for taking a snapshot of the external pins during normal operation of the part.

The JTAGEN Fuse must be programmed and the JTD bit in the I/O Register MCUCSR must be cleared to enable the JTAG Test Access Port.

When using the JTAG interface for Boundary-scan, using a JTAG TCK clock frequency higher than the internal chip frequency is possible. The chip clock is not required to run.

Data Registers

The Data Registers relevant for Boundary-scan operations are:

- · Bypass Register
- Device Identification Register
- Reset Register
- · Boundary-scan Chain

Bypass Register

The Bypass Register consists of a single Shift Register stage. When the Bypass Register is selected as path between TDI and TDO, the register is reset to 0 when leaving the Capture-DR



228

2466T-AVR-07/10



controller state. The Bypass Register can be used to shorten the scan chain on a system when the other devices are to be tested.

Device Identification Register

Figure 114 shows the structure of the Device Identification Register.

Figure 114. The Format of the Device Identification Register

	MSB						LSB
Bit	31	28	27	12	11	1	0
Device ID	Vers	sion	Part N	umber	Manufa	cturer ID	1
	4 h	oits	16	hits	11	bits	1 bit

Version

Version is a 4-bit number identifying the revision of the component. The JTAG version number follows the revision of the device. Revision A is 0x0, revision B is 0x1 and so on. However, some revisions deviate from this rule, and the relevant version number is shown in Table 87.

Table 87. JTAG Version Numbers

Version	JTAG Version Number (Hex)
ATmega16 revision G	0x6
ATmega16 revision H	0xE
ATmega16 revision I	0x8
ATmega16 revision J	0x9
ATmega16 revision K	0xA
ATmega16 revision L	0xB

Part Number

The part number is a 16-bit code identifying the component. The JTAG Part Number for ATmega16 is listed in Table 88.

Table 88. AVR JTAG Part Number

Part Number	JTAG Part Number (Hex)		
ATmega16	0x9403		

Manufacturer ID

The Manufacturer ID is a 11 bit code identifying the manufacturer. The JTAG manufacturer ID for ATMEL is listed in Table 89.

Table 89. Manufacturer ID

Manufacturer	JTAG Manufacturer ID (Hex)		
ATMEL	0x01F		

Reset Register

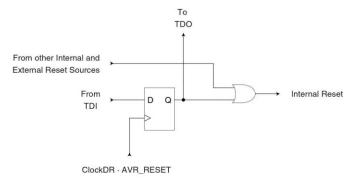
The Reset Register is a Test Data Register used to reset the part. Since the AVR tri-states Port Pins when reset, the Reset Register can also replace the function of the unimplemented optional JTAG instruction HIGHZ.

A high value in the Reset Register corresponds to pulling the External Reset low. The part is reset as long as there is a high value present in the Reset Register. Depending on the Fuse settings for the clock options, the part will remain reset for a Reset Time-Out Period (refer to "Clock Sources" on page 25) after releasing the Reset Register. The output from this Data Register is not latched, so the reset will take place immediately, as shown in Figure 115.

AMEL



Figure 115. Reset Register



Boundary-scan Chain

The Boundary-scan Chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having Off-chip connections.

See "Boundary-scan Chain" on page 232 for a complete description.

Boundary-scan Specific JTAG Instructions

The instruction register is 4-bit wide, supporting up to 16 instructions. Listed below are the JTAG instructions useful for Boundary-scan operation. Note that the optional HIGHZ instruction is not implemented, but all outputs with tri-state capability can be set in high-impedant state by using the AVR_RESET instruction, since the initial state for all port pins is tri-state.

As a definition in this datasheet, the LSB is shifted in and out first for all Shift Registers.

The OPCODE for each instruction is shown behind the instruction name in hex format. The text describes which Data Register is selected as path between TDI and TDO for each instruction.

EXTEST; \$0

Mandatory JTAG instruction for selecting the Boundary-scan Chain as Data Register for testing circuitry external to the AVR package. For port-pins, Pull-up Disable, Output Control, Output Data, and Input Data are all accessible in the scan chain. For Analog circuits having Off-chip connections, the interface between the analog and the digital logic is in the scan chain. The contents of the latched outputs of the Boundary-scan chain is driven out as soon as the JTAG IR-register is loaded with the EXTEST instruction.

The active states are:

- · Capture-DR: Data on the external pins are sampled into the Boundary-scan Chain.
- · Shift-DR: The Internal Scan Chain is shifted by the TCK input.
- · Update-DR: Data from the scan chain is applied to output pins.

IDCODE; \$1

Optional JTAG instruction selecting the 32-bit ID-register as Data Register. The ID-register consists of a version number, a device number and the manufacturer code chosen by JEDEC. This is the default instruction after power-up.

The active states are:

- · Capture-DR: Data in the IDCODE-register is sampled into the Boundary-scan Chain.
- Shift-DR: The IDCODE scan chain is shifted by the TCK input.

SAMPLE_PRELOAD; \$2

Mandatory JTAG instruction for pre-loading the output latches and talking a snap-shot of the input/output pins without affecting the system operation. However, the output latches are not connected to the pins. The Boundary-scan Chain is selected as Data Register.

AIMEL

230

2466T-AVR-07/10



The active states are:

- · Capture-DR: Data on the external pins are sampled into the Boundary-scan Chain.
- · Shift-DR: The Boundary-scan Chain is shifted by the TCK input.
- Update-DR: Data from the Boundary-scan Chain is applied to the output latches. However, the output latches are not connected to the pins.

AVR_RESET; \$C

The AVR specific public JTAG instruction for forcing the AVR device into the Reset mode or releasing the JTAG Reset source. The TAP controller is not reset by this instruction. The one bit Reset Register is selected as Data Register. Note that the reset will be active as long as there is a logic 'one' in the Reset Chain. The output from this chain is not latched.

The active states are:

· Shift-DR: The Reset Register is shifted by the TCK input.

BYPASS; \$F

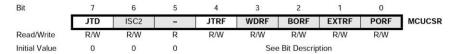
Mandatory JTAG instruction selecting the Bypass Register for Data Register.

The active states are:

- · Capture-DR: Loads a logic "0" into the Bypass Register.
- · Shift-DR: The Bypass Register cell between TDI and TDO is shifted.

Boundary-scan Related Register in I/O Memory

MCU Control and Status Register – MCUCSR The MCU Control and Status Register contains control bits for general MCU functions, and provides information on which reset source caused an MCU Reset.



· Bit 7 - JTD: JTAG Interface Disable

When this bit is zero, the JTAG interface is enabled if the JTAGEN Fuse is programmed. If this bit is one, the JTAG interface is disabled. In order to avoid unintentional disabling or enabling of the JTAG interface, a timed sequence must be followed when changing this bit: The application software must write this bit to the desired value twice within four cycles to change its value.

If the JTAG interface is left unconnected to other JTAG circuitry, the JTD bit should be set to one. The reason for this is to avoid static current at the TDO pin in the JTAG interface.

· Bit 4 - JTRF: JTAG Reset Flag

This bit is set if a reset is being caused by a logic one in the JTAG Reset Register selected by the JTAG instruction AVR_RESET. This bit is reset by a Power-on Reset, or by writing a logic zero to the flag.





Boundary-scan Chain

The Boundary-scan chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having Off-chip connection.

Scanning the Digital Port Pins

Figure 116 shows the Boundary-scan Cell for a bi-directional port pin with pull-up function. The cell consists of a standard Boundary-scan cell for the Pull-up Enable – PUExn – function, and a bi-directional pin cell that combines the three signals Output Control – OCxn, Output Data – ODxn, and Input Data – IDxn, into only a two-stage Shift Register. The port and pin indexes are not used in the following description.

The Boundary-scan logic is not included in the figures in the datasheet. Figure 117 shows a simple digital Port Pin as described in the section "I/O Ports" on page 50. The Boundary-scan details from Figure 116 replaces the dashed box in Figure 117.

When no alternate port function is present, the Input Data – ID – corresponds to the PINxn Register value (but ID has no synchronizer), Output Data corresponds to the PORT Register, Output Control corresponds to the Data Direction – DD Register, and the Pull-up Enable – PUExn – corresponds to logic expression $\overline{\text{PUD}} \cdot \overline{\text{DDxn}} \cdot \text{PORTxn}$.

Digital alternate port functions are connected outside the dotted box in Figure 117 to make the scan chain read the actual pin value. For Analog function, there is a direct connection from the external pin to the analog circuit, and a scan chain is inserted on the interface between the digital logic and the analog circuitry.

ShiftDR To Next Cell EXTEST Pullup Enable (PUE) FF2 LD2 D Output Control (OC) LD1 D D G Output Data (OD) FFO LDO Port Pin (PXn) G Input Data (ID) ◄ From Last Cell ClockDR UpdateDR

Figure 116. Boundary-scan Cell for Bidirectional Port Pin with Pull-up Function.





PUExn 5 0 4 WDx RESET **OCxn** RDx DATA BUS ODxn āৣ IDxn RESET SLEEP RRx SYNCHRONIZER ₫ UP DISABLE
UP ENABLE for pin Pxn
PUT CONTROL for pin Pxn
PUT DATA to pin Pxn
OATA from pin Pxn WDx: RDx: WPx: RRx: RPx: CLK 1/0 WRITE DDRX READ DDRX WRITE PORTX READ PORTX REGISTER READ PORTX PIN I/O CLOCK PUD: PUExn: OCxn: ODxn: IDxn: SLEEP

Figure 117. General Port Pin Schematic Diagram⁽¹⁾

Note: 1. See Boundary-scan description for details.

Boundary-scan and the Two-wire Interface

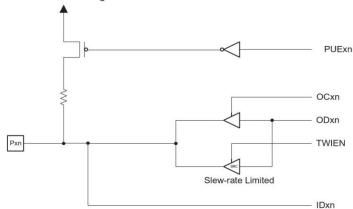
The 2 Two-wire Interface pins SCL and SDA have one additional control signal in the scanchain; Two-wire Interface Enable – TWIEN. As shown in Figure 118, the TWIEN signal enables a tri-state buffer with slew-rate control in parallel with the ordinary digital port pins. A general scan cell as shown in Figure 122 is attached to the TWIEN signal.

Notes: 1. A separate scan chain for the 50 ns spike filter on the input is not provided. The ordinary scan support for digital port pins suffice for connectivity tests. The only reason for having TWIEN in the scan path, is to be able to disconnect the slew-rate control buffer when doing boundary-scan.

Make sure the OC and TWIEN signals are not asserted simultaneously, as this will lead to drive contention.



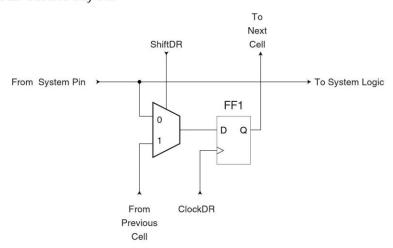
Figure 118. Additional Scan Signal for the Two-wire Interface



Scanning the RESET Pin

The RESET pin accepts 5V active low logic for standard reset operation, and 12V active high logic for High Voltage Parallel Programming. An observe-only cell as shown in Figure 119 is inserted both for the 5V reset signal; RSTT, and the 12V reset signal; RSTHV.

Figure 119. Observe-only Cell



Scanning the Clock Pins

The AVR devices have many clock options selectable by fuses. These are: Internal RC Oscillator, External RC, External Clock, (High Frequency) Crystal Oscillator, Low Frequency Crystal Oscillator, and Ceramic Resonator.

Figure 120 shows how each Oscillator with external connection is supported in the scan chain. The Enable signal is supported with a general boundary-scan cell, while the Oscillator/Clock output is attached to an observe-only cell. In addition to the main clock, the Timer Oscillator is scanned in the same way. The output from the internal RC Oscillator is not scanned, as this Oscillator does not have external connections.

AMEL



Figure 120. Boundary-scan Cells for Oscillators and Clock Options

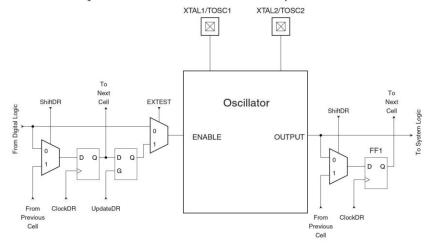


Table 90 summaries the scan registers for the external clock pin XTAL1, Oscillators with XTAL1/XTAL2 connections as well as 32 kHz Timer Oscillator.

Table 90. Scan Signals for the Oscillators (1)(2)(3)

Enable Signal	Scanned Clock Line	Clock Option	Scanned Clock Line when not Used
EXTCLKEN	EXTCLK (XTAL1)	External Clock	0
OSCON	OSCCK	External Crystal External Ceramic Resonator	0
RCOSCEN	RCCK	External RC	1
OSC32EN	OSC32CK	Low Freq. External Crystal	0
TOSKON	TOSCK	32 kHz Timer Oscillator	0

- Notes: 1. Do not enable more than one clock source as main clock at a time.
 - 2. Scanning an Oscillator output gives unpredictable results as there is a frequency drift between the Internal Oscillator and the JTAG TCK clock. If possible, scanning an external clock is
 - 3. The clock configuration is programmed by fuses. As a fuse is not changed run-time, the clock configuration is considered fixed for a given application. The user is advised to scan the same clock option as to be used in the final system. The enable signals are supported in the scan chain because the system logic can disable clock options in sleep modes, thereby disconnecting the Oscillator pins from the scan path if not provided. The INTCAP Fuses are not supported in the scan-chain, so the boundary scan chain can not make a XTAL Oscillator requiring internal capacitors to run unless the fuse is correctly programmed.

Scanning the Analog Comparator

The relevant Comparator signals regarding Boundary-scan are shown in Figure 121. The Boundary-scan cell from Figure 122 is attached to each of these signals. The signals are described in Table 91.

The Comparator need not be used for pure connectivity testing, since all analog inputs are shared with a digital port pin as well.



Figure 121. Analog Comparator

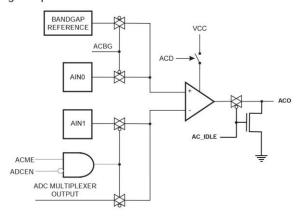


Figure 122. General Boundary-scan Cell used for Signals for Comparator and ADC

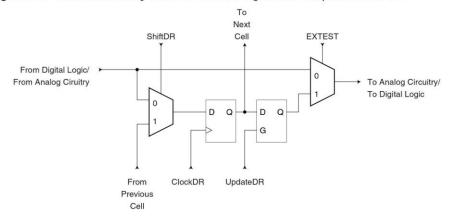






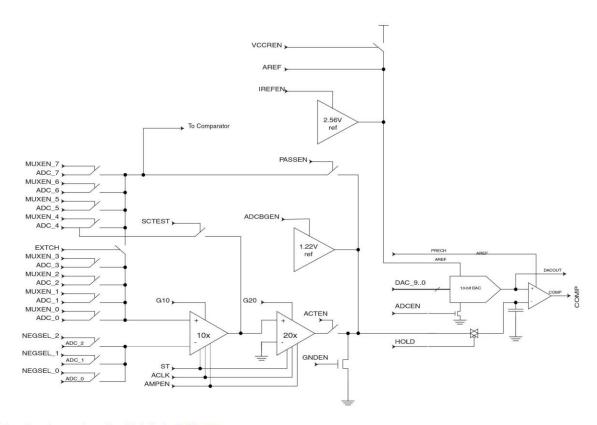
Table 91. Boundary-scan Signals for the Analog Comparator

Signal Name	Direction as Seen from the Comparator	Description	Recommended Input when Not in Use	Output Values when Recommended Inputs are Used
AC_IDLE	Input	Turns off Analog comparator when true	1	Depends upon μC code being executed
ACO	Output	Analog Comparator Output	Will become input to µC code being executed	0
ACME	Input	Uses output signal from ADC mux when true	0	Depends upon μC code being executed
ACBG	Input	Bandgap Reference enable	0	Depends upon μC code being executed

Scanning the ADC

Figure 123 shows a block diagram of the ADC with all relevant control and observe signals. The Boundary-scan cell from Figure 122 is attached to each of these signals. The ADC need not be used for pure connectivity testing, since all analog inputs are shared with a digital port pin as well.

Figure 123. Analog to Digital Converter



The signals are described briefly in Table 92.

ATMEL

237

2466T-AVR-07/10



Table 92. Boundary-scan Signals for the ADC

Signal Name	Direction as Seen from the ADC			Output Values when Recommended Inputs are used, and CPU is not Using the ADC
COMP	Output	Comparator Output	0	0
ACLK	Input	Clock signal to gain stages implemented as Switch-cap filters	0	0
ACTEN	Input	Enable path from gain stages to the comparator	0	0
ADCBGEN	Input	Enable Band-gap reference as negative input to comparator	0	0
ADCEN	Input	Power-on signal to the ADC	0	0
AMPEN	Input	Power-on signal to the gain stages	0	0
DAC_9	Input	Bit 9 of digital value to DAC	1	1
DAC_8	Input	Bit 8 of digital value to DAC	0	0
DAC_7	Input	Bit 7 of digital value to DAC	0	0
DAC_6	Input	Bit 6 of digital value to DAC	0	0
DAC_5	Input	Bit 5 of digital value to DAC	0	0
DAC_4	Input	Bit 4 of digital value to DAC	0	0
DAC_3	Input	Bit 3 of digital value to DAC	0	0
DAC_2	Input	Bit 2 of digital value to DAC	0	0
DAC_1	Input	Bit 1 of digital value to DAC	0	0
DAC_0	Input	Bit 0 of digital value to DAC	0	0
EXTCH	Input	Connect ADC channels 0 - 3 to by- pass path around gain stages	1	1
G10	Input	Enable 10x gain	0	0
G20	Input	Enable 20x gain	0	0
GNDEN	Input	Ground the negative input to comparator when true	0	0
HOLD	Input	Sample&Hold signal. Sample analog signal when low. Hold signal when high. If gain stages are used, this signal must go active when ACLK is high.	1	1
IREFEN	Input	Enables Band-gap reference as AREF signal to DAC	0	0
MUXEN_7	Input	Input Mux bit 7	0	0
MUXEN_6	Input	Input Mux bit 6	0	0
MUXEN_5	Input	Input Mux bit 5	0	0
MUXEN_4	Input	Input Mux bit 4	0	0
MUXEN_3	Input	Input Mux bit 3	0	0





Table 92. Boundary-scan Signals for the ADC (Continued)

Signal Name	Direction as Seen from the ADC	Description	Recommended Input when Not in Use	Output Values when Recommended Inputs are used, and CPU is not Using the ADC
MUXEN_2	Input	Input Mux bit 2	0	0
MUXEN_1	Input	Input Mux bit 1	0	0
MUXEN_0	Input	Input Mux bit 0	1	1
NEGSEL_2	Input	Input Mux for negative input for differential signal, bit 2	0	0
NEGSEL_1	Input	Input Mux for negative input for differential signal, bit 1	0	0
NEGSEL_0	Input	Input Mux for negative input for differential signal, bit 0	0	0
PASSEN	Input	Enable pass-gate of gain stages.	1	1
PRECH	Input	Precharge output latch of comparator. (Active low)	1	1
SCTEST	Input	Switch-cap TEST enable. Output from x10 gain stage send out to Port Pin having ADC_4	0	0
ST	Input	Output of gain stages will settle faster if this signal is high first two ACLK periods after AMPEN goes high.	0	0
VCCREN	Input	Selects Vcc as the ACC reference voltage.	0	0

Note: Incorrect setting of the switches in Figure 123 will make signal contention and may damage the part. There are several input choices to the S&H circuitry on the negative input of the output comparator in Figure 123. Make sure only one path is selected from either one ADC pin, Bandgap reference source, or Ground.

If the ADC is not to be used during scan, the recommended input values from Table 92 should be used. The user is recommended **not** to use the Differential Gain stages during scan. Switchcap based gain stages require fast operation and accurate timing which is difficult to obtain when used in a scan chain. Details concerning operations of the differential gain stage is therefore not provided.

The AVR ADC is based on the analog circuitry shown in Figure 123 with a successive approximation algorithm implemented in the digital logic. When used in Boundary-scan, the problem is usually to ensure that an applied analog voltage is measured within some limits. This can easily be done without running a successive approximation algorithm: apply the lower limit on the digital DAC[9:0] lines, make sure the output from the comparator is low, then apply the upper limit on the digital DAC[9:0] lines, and verify the output from the comparator to be high.

The ADC need not be used for pure connectivity testing, since all analog inputs are shared with a digital port pin as well.

When using the ADC, remember the following:

- The Port Pin for the ADC channel in use must be configured to be an input with pull-up disabled to avoid signal contention.
- In Normal mode, a dummy conversion (consisting of 10 comparisons) is performed when enabling the ADC. The user is advised to wait at least 200 ns after enabling the ADC before





controlling/observing any ADC signal, or perform a dummy conversion before using the first result.

 The DAC values must be stable at the midpoint value 0x200 when having the HOLD signal low (Sample mode).

As an example, consider the task of verifying a 1.5V $\pm 5\%$ input signal at ADC channel 3 when the power supply is 5.0V and AREF is externally connected to V_{CC} .

The recommended values from Table 92 are used unless other values are given in the algorithm in Table 93. Only the DAC and Port Pin values of the Scan-chain are shown. The column "Actions" describes what JTAG instruction to be used before filling the Boundary-scan Register with the succeeding columns. The verification should be done on the data scanned out when scanning in the data on the same row in the table.

Table 93. Algorithm for Using the ADC

Step	Actions	ADCEN	DAC	MUXEN	HOLD	PRECH	PA3. Data	PA3. Control	PA3. Pullup_ Enable
1	SAMPLE _PRELO AD	1	0x200	0x08	1	1	0	0	0
2	EXTEST	1	0x200	0x08	0	1	0	0	0
3		1	0x200	0x08	1	1	0	0	0
4		1	0x123	0x08	1	1	0	0	0
5		1	0x123	0x08	1	0	0	0	0
6	Verify the COMP bit scanned out to be 0	1	0x200	80x0	1	1	0	0	0
7		1	0x200	0x08	0	1	0	0	0
8		1	0x200	0x08	1	1	0	0	0
9		1	0x143	0x08	1	1	0	0	0
10		1	0x143	0x08	1	0	0	0	0
11	Verify the COMP bit scanned out to be 1	1	0x200	80x0	1	1	0	0	0

Using this algorithm, the timing constraint on the HOLD signal constrains the TCK clock frequency. As the algorithm keeps HOLD high for five steps, the TCK clock frequency has to be at least five times the number of scan bits divided by the maximum hold time, $t_{\text{hold,max}}$.





ATmega16 Boundary-scan Order Table 94 shows the scan order between TDI and TDO when the Boundary-scan chain is selected as data path. Bit 0 is the LSB; the first bit scanned in, and the first bit scanned out. The scan order follows the pin-out order as far as possible. Therefore, the bits of Port A is scanned in the opposite bit order of the other ports. Exceptions from the rules are the Scan chains for the analog circuits, which constitute the most significant bits of the scan chain regardless of which physical pin they are connected to. In Figure 116, PXn. Data corresponds to FF0, PXn. Control corresponds to FF1, and PXn. Pullup_enable corresponds to FF2. Bit 2, 3, 4, and 5 of Port C is not in the scan chain, since these pins constitute the TAP pins when the JTAG is enabled.

Table 94. ATmega16 Boundary-scan Order

Bit Number	Signal Name	Module
140	AC_IDLE	Comparator
139	ACO	
138	ACME	
137	ACBG	
136	COMP	ADC
135	PRIVATE_SIGNAL1(1)	
134	ACLK	
133	ACTEN	
132	PRIVATE_SIGNAL2(2)	
131	ADCBGEN	
130	ADCEN	
129	AMPEN	
128	DAC_9	
127	DAC_8	
126	DAC_7	
125	DAC_6	
124	DAC_5	
123	DAC_4	
122	DAC_3	
121	DAC_2	
120	DAC_1	
119	DAC_0	
118	EXTCH	
117	G10	
116	G20	
115	GNDEN	
114	HOLD	
113	IREFEN	
112	MUXEN_7	





Table 94. ATmega16 Boundary-scan Order (Continued)

Bit Number	Signal Name	Module
111	MUXEN_6	
110	MUXEN_5	
109	MUXEN_4	
108	MUXEN_3	
107	MUXEN_2	
106	MUXEN_1	
105	MUXEN_0	
104	NEGSEL_2	
103	NEGSEL_1	
102	NEGSEL_0	
101	PASSEN	
100	PRECH	
99	SCTEST	
98	ST	
97	VCCREN	
96	PB0.Data	Port B
95	PB0.Control	
94	PB0.Pullup_Enable	
93	PB1.Data	
92	PB1.Control	
91	PB1.Pullup_Enable	
90	PB2.Data	
89	PB2.Control	
88	PB2.Pullup_Enable	
87	PB3.Data	
86	PB3.Control	
85	PB3.Pullup_Enable	
84	PB4.Data	
83	PB4.Control	
82	PB4.Pullup_Enable	
81	PB5.Data	
80	PB5.Control	
79	PB5.Pullup_Enable	
78	PB6.Data	
77	PB6.Control	
76	PB6.Pullup_Enable	





Table 94. ATmega16 Boundary-scan Order (Continued)

Bit Number	Signal Name	Module
75	PB7.Data	
74	PB7.Control	
73	PB7.Pullup_Enable	
72	RSTT	Reset Logic
71	RSTHV	(Observe-Only)
70	EXTCLKEN	Enable signals for main clock/Oscillators
69	OSCON	
68	RCOSCEN	
67	OSC32EN	
66	EXTCLK (XTAL1)	Clock input and Oscillators for the main clock
65	OSCCK	(Observe-Only)
64	RCCK	
63	OSC32CK	
62	TWIEN	TWI
61	PD0.Data	Port D
60	PD0.Control	
59	PD0.Pullup_Enable	
58	PD1.Data	
57	PD1.Control	
56	PD1.Pullup_Enable	
55	PD2.Data	
54	PD2.Control	
53	PD2.Pullup_Enable	
52	PD3.Data	
51	PD3.Control	
50	PD3.Pullup_Enable	
49	PD4.Data	
48	PD4.Control	
47	PD4.Pullup_Enable	
46	PD5.Data	
45	PD5.Control	
44	PD5.Pullup_Enable	
43	PD6.Data	
42	PD6.Control	
41	PD6.Pullup_Enable	
40	PD7.Data	





Table 94. ATmega16 Boundary-scan Order (Continued)

Bit Number	Signal Name	Module
39	PD7.Control	
38	PD7.Pullup_Enable	
37	PC0.Data	Port C
36	PC0.Control	
35	PC0.Pullup_Enable	
34	PC1.Data	
33	PC1.Control	
32	PC1.Pullup_Enable	
31	PC6.Data	
30	PC6.Control	
29	PC6.Pullup_Enable	
28	PC7.Data	
27	PC7.Control	
26	PC7.Pullup_Enable	
25	TOSC	32 kHz Timer Oscillator
24	TOSCON	
23	PA7.Data	Port A
22	PA7.Control	
21	PA7.Pullup_Enable	
20	PA6.Data	
19	PA6.Control	
18	PA6.Pullup_Enable	
17	PA5.Data	
16	PA5.Control	
15	PA5.Pullup_Enable	
14	PA4.Data	
13	PA4.Control	
12	PA4.Pullup_Enable	
11	PA3.Data	
10	PA3.Control	
9	PA3.Pullup_Enable	
8	PA2.Data	
7	PA2.Control	
6	PA2.Pullup_Enable	
5	PA1.Data	





Table 94. ATmega16 Boundary-scan Order (Continued)

Bit Number	Signal Name	Module
4	PA1.Control	
3	PA1.Pullup_Enable	
2	PA0.Data	
1	PA0.Control	
0	PA0.Pullup_Enable	

PRIVATE_SIGNAL1 should always be scanned in as zero.
 PRIVATE:SIGNAL2 should always be scanned in as zero.

Boundary-scan Description Language Files

Boundary-scan Description Language (BSDL) files describe Boundary-scan capable devices in a standard format used by automated test-generation software. The order and function of bits in the Boundary-scan Data Register are included in this description. A BSDL file for ATmega16 is available.





Boot Loader Support – Read-While-Write Self-Programming

The Boot Loader Support provides a real Read-While-Write Self-Programming mechanism for downloading and uploading program code by the MCU itself. This feature allows flexible application software updates controlled by the MCU using a Flash-resident Boot Loader program. The Boot Loader program can use any available data interface and associated protocol to read code and write (program) that code into the Flash memory, or read the code from the Program memory. The program code within the Boot Loader section has the capability to write into the entire Flash, including the Boot Loader memory. The Boot Loader can thus even modify itself, and it can also erase itself from the code if the feature is not needed anymore. The size of the Boot Loader memory is configurable with Fuses and the Boot Loader has two separate sets of Boot Lock bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

Features

- Read-While-Write Self-Programming
- · Flexible Boot Memory size
- · High Security (Separate Boot Lock Bits for a Flexible Protection)
- · Separate Fuse to Select Reset Vector
- Optimized Page⁽¹⁾ Size
- · Code Efficient Algorithm
- · Efficient Read-Modify-Write Support

Note: 1. A page is a section in the flash consisting of several bytes (see Table 107 on page 262) used during programming. The page organization does not affect normal operation.

Application and Boot Loader Flash Sections

The Flash memory is organized in two main sections, the Application section and the Boot Loader section (see Figure 125). The size of the different sections is configured by the BOOTSZ Fuses as shown in Table 100 on page 257 and Figure 125. These two sections can have different level of protection since they have different sets of Lock bits.

Application Section

The Application section is the section of the Flash that is used for storing the application code. The protection level for the application section can be selected by the Application Boot Lock bits (Boot Lock bits 0), see Table 96 on page 249. The Application section can never store any Boot Loader code since the SPM instruction is disabled when executed from the Application section.

BLS – Boot Loader Section

While the Application section is used for storing the application code, the The Boot Loader software must be located in the BLS since the SPM instruction can initiate a programming when executing from the BLS only. The SPM instruction can access the entire Flash, including the BLS itself. The protection level for the Boot Loader section can be selected by the Boot Loader Lock bits (Boot Lock bits 1), see Table 97 on page 249.

Read-While-Write and no Read-While-Write Flash Sections

Whether the CPU supports Read-While-Write or if the CPU is halted during a Boot Loader software update is dependent on which address that is being programmed. In addition to the two sections that are configurable by the BOOTSZ Fuses as described above, the Flash is also divided into two fixed sections, the Read-While-Write (RWW) section and the No Read-While-Write (NRWW) section. The limit between the RWW- and NRWW sections is given in Table 101 on page 257 and Figure 125 on page 248. The main difference between the two sections is:

- When erasing or writing a page located inside the RWW section, the NRWW section can be read during the operation.
- When erasing or writing a page located inside the NRWW section, the CPU is halted during the entire operation.

Note that the user software can never read any code that is located inside the RWW section during a Boot Loader software operation. The syntax "Read-While-Write section" refers to which section that is being programmed (erased or written), not which section that actually is being read during a Boot Loader software update.





RWW – Read-While-Write Section If a Boot Loader software update is programming a page inside the RWW section, it is possible to read code from the Flash, but only code that is located in the NRWW section. During an ongoing programming, the software must ensure that the RWW section never is being read. If the user software is trying to read code that is located inside the RWW section (that is, by a call/jmp/lpm or an interrupt) during programming, the software might end up in an unknown state. To avoid this, the interrupts should either be disabled or moved to the Boot Loader section. The Boot Loader section is always located in the NRWW section. The RWW Section Busy bit (RWWSB) in the Store Program Memory Control Register (SPMCR) will be read as logical one as long as the RWW section is blocked for reading. After a programming is completed, the RWWSB must be cleared by software before reading code located in the RWW section. See "Store Program Memory Control Register – SPMCR" on page 250. for details on how to clear RWWSB.

NRWW - No Read-While-Write Section The code located in the NRWW section can be read when the Boot Loader software is updating a page in the RWW section. When the Boot Loader code updates the NRWW section, the CPU is halted during the entire page erase or page write operation.

Table 95. Read-While-Write Features

Which Section does the Z- pointer Address during the Programming?	Which Section can be Read during Programming?	Is the CPU Halted?	Read-While- Write Supported?	
RWW section	NRWW section	No	Yes	
NRWW section	None	Yes	No	

Figure 124. Read-While-Write vs. No Read-While-Write

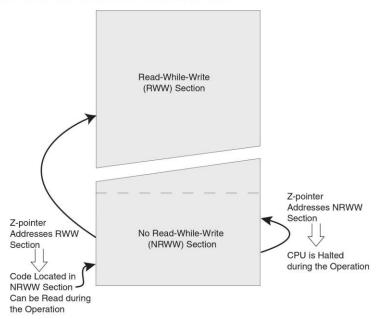
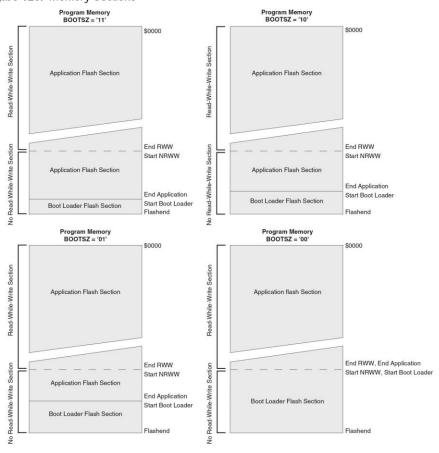






Figure 125. Memory Sections⁽¹⁾



Note: 1. The parameters in the figure above are given in Table 100 on page 257.

Boot Loader Lock Bits

If no Boot Loader capability is needed, the entire Flash is available for application code. The Boot Loader has two separate sets of Boot Lock bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

The user can select:

- To protect the entire Flash from a software update by the MCU
- To protect only the Boot Loader Flash section from a software update by the MCU
- To protect only the Application Flash section from a software update by the MCU
- Allow software update in the entire Flash

See Table 96 and Table 97 for further details. The Boot Lock bits can be set in software and in Serial or Parallel Programming mode, but they can be cleared by a Chip Erase command only. The general Write Lock (Lock Bit mode 2) does not control the programming of the Flash memory by SPM instruction. Similarly, the general Read/Write Lock (Lock Bit mode 3) does not control reading nor writing by LPM/SPM, if it is attempted.

ATMEL



Table 96. Boot Lock BitO Protection Modes (Application Section)(1)

BLB0 Mode	BLB02	BLB01	Protection
1	1	1	No restrictions for SPM or LPM accessing the Application section.
2	1	0	SPM is not allowed to write to the Application section.
3	0	0	SPM is not allowed to write to the Application section, and LPM executing from the Boot Loader section is not allowed to read from the Application section. If interrupt vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.
4	0	1	LPM executing from the Boot Loader section is not allowed to read from the Application section. If interrupt vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.

Note: 1. "1" means unprogrammed, "0" means programmed

Table 97. Boot Lock Bit1 Protection Modes (Boot Loader Section)⁽¹⁾

BLB1 mode	BLB12	BLB11	Protection
1	1	1	No restrictions for SPM or LPM accessing the Boot Loader section.
2	1	0	SPM is not allowed to write to the Boot Loader section.
3	0	0	SPM is not allowed to write to the Boot Loader section, and LPM executing from the Application section is not allowed to read from the Boot Loader section. If interrupt vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.
4	0	1	LPM executing from the Application section is not allowed to read from the Boot Loader section. If interrupt vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.

Note: 1. "1" means unprogrammed, "0" means programmed

Entering the Boot Loader Program

Entering the Boot Loader takes place by a jump or call from the application program. This may be initiated by a trigger such as a command received via USART, or SPI interface. Alternatively, the Boot Reset Fuse can be programmed so that the Reset Vector is pointing to the Boot Flash start address after a reset. In this case, the Boot Loader is started after a reset. After the application code is loaded, the program can start executing the application code. Note that the fuses cannot be changed by the MCU itself. This means that once the Boot Reset Fuse is programmed, the Reset Vector will always point to the Boot Loader Reset and the fuse can only be changed through the serial or parallel programming interface.





Table 98. Boot Reset Fuse⁽¹⁾

BOOTRST	Reset Address
1	Reset Vector = Application reset (address \$0000)
0	Reset Vector = Boot Loader reset (see Table 100 on page 257)

Note: 1. "1" means unprogrammed, "0" means programmed

Store Program Memory Control Register – SPMCR The Store Program Memory Control Register contains the control bits needed to control the Boot Loader operations.

Bit	7	6	5	4	3	2	1	0	
	SPMIE	RWWSB	-	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	SPMCR
Read/Write	R/W	R	R	R/W	R/W	R/W	R/W	R/W	•
Initial value	0	0	0	0	0	0	0	0	

• Bit 7 - SPMIE: SPM Interrupt Enable

When the SPMIE bit is written to one, and the I-bit in the Status Register is set (one), the SPM ready interrupt will be enabled. The SPM ready Interrupt will be executed as long as the SPMEN bit in the SPMCR Register is cleared.

· Bit 6 - RWWSB: Read-While-Write Section Busy

When a self-programming (Page Erase or Page Write) operation to the RWW section is initiated, the RWWSB will be set (one) by hardware. When the RWWSB bit is set, the RWW section cannot be accessed. The RWWSB bit will be cleared if the RWWSRE bit is written to one after a Self-Programming operation is completed. Alternatively the RWWSB bit will automatically be cleared if a page load operation is initiated.

· Bit 5 - Res: Reserved Bit

This bit is a reserved bit in the ATmega16 and always read as zero.

· Bit 4 - RWWSRE: Read-While-Write Section Read Enable

When programming (Page Erase or Page Write) to the RWW section, the RWW section is blocked for reading (the RWWSB will be set by hardware). To re-enable the RWW section, the user software must wait until the programming is completed (SPMEN will be cleared). Then, if the RWWSRE bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles re-enables the RWW section. The RWW section cannot be re-enabled while the Flash is busy with a page erase or a page write (SPMEN is set). If the RWWSRE bit is written while the Flash is being loaded, the Flash load operation will abort and the data loaded will be lost.

· Bit 3 - BLBSET: Boot Lock Bit Set

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles sets Boot Lock bits, according to the data in R0. The data in R1 and the address in the Z-pointer are ignored. The BLBSET bit will automatically be cleared upon completion of the Lock bit set, or if no SPM instruction is executed within four clock cycles.

An LPM instruction within three cycles after BLBSET and SPMEN are set in the SPMCR Register, will read either the Lock bits or the Fuse bits (depending on Z0 in the Z-pointer) into the destination register. See "Reading the Fuse and Lock Bits from Software" on page 254 for details.





· Bit 2 - PGWRT: Page Write

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes Page Write, with the data stored in the temporary buffer. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGWRT bit will auto-clear upon completion of a page write, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire page write operation if the NRWW section is addressed.

· Bit 1 - PGERS: Page Erase

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes Page Erase. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGERS bit will auto-clear upon completion of a page erase, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire page write operation if the NRWW section is addressed.

· Bit 0 - SPMEN: Store Program Memory Enable

This bit enables the SPM instruction for the next four clock cycles. If written to one together with either RWWSRE, BLBSET, PGWRT' or PGERS, the following SPM instruction will have a special meaning, see description above. If only SPMEN is written, the following SPM instruction will store the value in R1:R0 in the temporary page buffer addressed by the Z-pointer. The LSB of the Z-pointer is ignored. The SPMEN bit will auto-clear upon completion of an SPM instruction, or if no SPM instruction is executed within four clock cycles. During page erase and page write, the SPMEN bit remains high until the operation is completed.

Writing any other combination than "10001", "01001", "00101", "00001" or "00001" in the lower five bits will have no effect.

Addressing the Flash during Self-Programming

The Z-pointer is used to address the SPM commands.

Bit	15	14	13	12	11	10	9	8
ZH (R31)	Z15	Z14	Z13	Z12	Z11	Z10	Z9	Z8
ZL (R30)	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0
	7	6		1	2	2	- 1	0

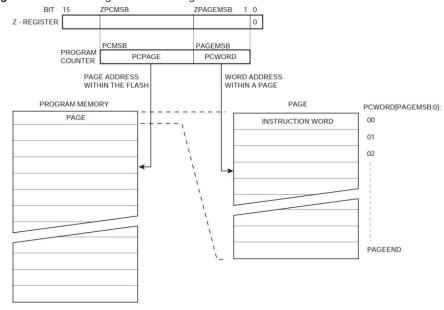
Since the Flash is organized in pages (see Table 107 on page 262), the Program Counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is shown in Figure 126. Note that the Page Erase and Page Write operations are addressed independently. Therefore it is of major importance that the Boot Loader software addresses the same page in both the Page Erase and Page Write operation. Once a programming operation is initiated, the address is latched and the Z-pointer can be used for other operations.

The only SPM operation that does not use the Z-pointer is Setting the Boot Loader Lock bits. The content of the Z-pointer is ignored and will have no effect on the operation. The LPM instruction does also use the Z pointer to store the address. Since this instruction addresses the Flash byte by byte, also the LSB (bit Z0) of the Z-pointer is used.





Figure 126. Addressing the Flash during SPM⁽¹⁾



- Notes: 1. The different variables used in Figure 126 are listed in Table 102 on page 258.
 - 2. PCPAGE and PCWORD are listed in Table 107 on page 262.

Self-Programming the Flash

The program memory is updated in a page by page fashion. Before programming a page with the data stored in the temporary page buffer, the page must be erased. The temporary page buffer is filled one word at a time using SPM and the buffer can be filled either before the page erase command or between a page erase and a page write operation:

Alternative 1, fill the buffer before a Page Erase

- · Fill temporary page buffer
- · Perform a Page Erase
- · Perform a Page Write

Alternative 2, fill the buffer after Page Erase

- · Perform a Page Erase
- · Fill temporary page buffer
- · Perform a Page Write

If only a part of the page needs to be changed, the rest of the page must be stored (for example in the temporary page buffer) before the erase, and then be rewritten. When using alternative 1, the Boot Loader provides an effective Read-Modify-Write feature which allows the user software to first read the page, do the necessary changes, and then write back the modified data. If alternative 2 is used, it is not possible to read the old data while loading since the page is already erased. The temporary page buffer can be accessed in a random sequence. It is essential that the page address used in both the page erase and page write operation is addressing the same page. See "Simple Assembly Code Example for a Boot Loader" on page 256 for an assembly code example.



252



Performing Page Erase by SPM

To execute Page Erase, set up the address in the Z-pointer, write "X0000011" to SPMCR and execute SPM within four clock cycles after writing SPMCR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE in the Z-register. Other bits in the Z-pointer must be written zero during this operation.

- Page Erase to the RWW section: The NRWW section can be read during the page erase.
- Page Erase to the NRWW section: The CPU is halted during the operation.

Filling the Temporary Buffer (Page Loading)

To write an instruction word, set up the address in the Z-pointer and data in R1:R0, write "00000001" to SPMCR and execute SPM within four clock cycles after writing SPMCR. The content of PCWORD in the Z-register is used to address the data in the temporary buffer. The temporary buffer will auto-erase after a page write operation or by writing the RWWSRE bit in SPMCR. It is also erased after a system reset. Note that it is not possible to write more than one time to each address without erasing the temporary buffer.

Note: If the EEPROM is written in the middle of an SPM Page Load operation, all data loaded will be lost.

Performing a Page Write

To execute Page Write, set up the address in the Z-pointer, write "X0000101" to SPMCR and execute SPM within four clock cycles after writing SPMCR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE. Other bits in the Z-pointer must be written zero during this operation.

- Page Write to the RWW section: The NRWW section can be read during the Page Write.
- · Page Write to the NRWW section: The CPU is halted during the operation.

Using the SPM Interrupt

If the SPM interrupt is enabled, the SPM interrupt will generate a constant interrupt when the SPMEN bit in SPMCR is cleared. This means that the interrupt can be used instead of polling the SPMCR Register in software. When using the SPM interrupt, the Interrupt Vectors should be moved to the BLS section to avoid that an interrupt is accessing the RWW section when it is blocked for reading. How to move the interrupts is described in "Interrupts" on page 45.

Consideration while Updating BLS

Special care must be taken if the user allows the Boot Loader section to be updated by leaving Boot Lock bit11 unprogrammed. An accidental write to the Boot Loader itself can corrupt the entire Boot Loader, and further software updates might be impossible. If it is not necessary to change the Boot Loader software itself, it is recommended to program the Boot Lock bit11 to protect the Boot Loader software from any internal software changes.

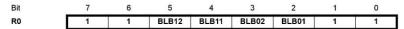
Prevent Reading the RWW Section during Self-Programming

During Self-Programming (either Page Erase or Page Write), the RWW section is always blocked for reading. The user software itself must prevent that this section is addressed during the Self-Programming operation. The RWWSB in the SPMCR will be set as long as the RWW section is busy. During self-programming the Interrupt Vector table should be moved to the BLS as described in "Interrupts" on page 45, or the interrupts must be disabled. Before addressing the RWW section after the programming is completed, the user software must clear the RWWSB by writing the RWWSRE. See "Simple Assembly Code Example for a Boot Loader" on page 256 for an example.





Setting the Boot Loader Lock Bits by SPM To set the Boot Loader Lock bits, write the desired data to R0, write "X0001001" to SPMCR and execute SPM within four clock cycles after writing SPMCR. The only accessible Lock bits are the Boot Lock bits that may prevent the Application and Boot Loader section from any software update by the MCU.



See Table 96 and Table 97 for how the different settings of the Boot Loader bits affect the Flash access.

If bits 5..2 in R0 are cleared (zero), the corresponding Boot Lock bit will be programmed if an SPM instruction is executed within four cycles after BLBSET and SPMEN are set in SPMCR. The Z-pointer is don't care during this operation, but for future compatibility it is recommended to load the Z-pointer with \$0001 (same as used for reading the Lock bits). For future compatibility It is also recommended to set bits 7, 6, 1, and 0 in R0 to "1" when writing the Lock bits. When programming the Lock bits the entire Flash can be read during the operation.

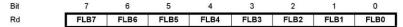
EEPROM Write Prevents Writing to SPMCR Note that an EEPROM write operation will block all software programming to Flash. Reading the Fuses and Lock bits from software will also be prevented during the EEPROM write operation. It is recommended that the user checks the status bit (EEWE) in the EECR Register and verifies that the bit is cleared before writing to the SPMCR Register.

Reading the Fuse and Lock Bits from Software

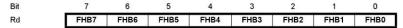
It is possible to read both the Fuse and Lock bits from software. To read the Lock bits, load the Z-pointer with \$0001 and set the BLBSET and SPMEN bits in SPMCR. When an LPM instruction is executed within three CPU cycles after the BLBSET and SPMEN bits are set in SPMCR, the value of the Lock bits will be loaded in the destination register. The BLBSET and SPMEN bits will auto-clear upon completion of reading the Lock bits or if no LPM instruction is executed within three CPU cycles or no SPM instruction is executed within four CPU cycles. When BLB-SET and SPMEN are cleared, LPM will work as described in the Instruction set Manual.

Bit	7	6	5	4	3	2	1	0
Rd	-	-	BLB12	BLB11	BLB02	BLB01	LB2	LB1

The algorithm for reading the Fuse Low bits is similar to the one described above for reading the Lock bits. To read the Fuse Low bits, load the Z-pointer with \$0000 and set the BLBSET and SPMEN bits in SPMCR. When an LPM instruction is executed within three cycles after the BLB-SET and SPMEN bits are set in the SPMCR, the value of the Fuse Low bits (FLB) will be loaded in the destination register as shown below. Refer to Table 106 on page 261 for a detailed description and mapping of the Fuse Low bits.



Similarly, when reading the Fuse High bits, load \$0003 in the Z-pointer. When an LPM instruction is executed within three cycles after the BLBSET and SPMEN bits are set in the SPMCR, the value of the Fuse High bits (FHB) will be loaded in the destination register as shown below. Refer to Table 105 on page 260 for detailed description and mapping of the Fuse High bits.



Fuse and Lock bits that are programmed, will be read as zero. Fuse and Lock bits that are unprogrammed, will be read as one.

Preventing Flash Corruption During periods of low V_{CC} , the Flash program can be corrupted because the supply voltage is too low for the CPU and the Flash to operate properly. These issues are the same as for board level systems using the Flash, and the same design solutions should be applied.

<u>AIMEL</u>

254



A Flash program corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the Flash requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage for executing instructions is too low.

Flash corruption can easily be avoided by following these design recommendations (one is sufficient):

- If there is no need for a Boot Loader update in the system, program the Boot Loader Lock bits to prevent any Boot Loader software updates.
- Keep the AVR RESET active (low) during periods of insufficient power supply voltage.
 This can be done by enabling the internal Brown-out Detector (BOD) if the operating voltage matches the detection level. If not, an external low V_{CC} Reset Protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.
- 3. Keep the AVR core in Power-down Sleep mode during periods of low V_{CC} . This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the SPMCR Register and thus the Flash from unintentional writes.

Programming Time for Flash when using SPM

The Calibrated RC Oscillator is used to time Flash accesses. Table 99 shows the typical programming time for Flash accesses from the CPU.

Table 99. SPM Programming Time.

Symbol	Min Programming Time	Max Programming Time
Flash write (Page Erase, Page Write, and write Lock bits by SPM)	3.7 ms	4.5 ms





Simple Assembly Code Example for a Boot Loader

```
;-the routine writes one page of data from RAM to Flash
 ; the first data location in RAM is pointed to by the Y pointer
 ; the first data location in Flash is pointed to by the Z pointer
 ;-error handling is not included
 ;-the routine must be placed inside the boot space
 ; (at least the Do_spm sub routine). Only code inside NRWW section can
 ; be read during self-programming (page erase and page write).
 ;-registers used: r0, r1, temp1 (r16), temp2 (r17), looplo (r24),
 ; loophi (r25), spmcrval (r20)
 ; storing and restoring of registers is not included in the routine
 ; register usage can be optimized at the expense of code size
 ;-It is assumed that either the interrupt table is moved to the Boot
 ; loader section or that the interrupts are disabled.
.equ PAGESIZEB = PAGESIZE*2
                                  ; PAGESIZEB is page size in BYTES, not
                                    ; words
.org SMALLBOOTSTART
Write_page:
 ; page erase
 ldi spmcrval, (1<<PGERS) | (1<<SPMEN) call Do_spm
 ; re-enable the RWW section
 ldi spmcrval, (1<<RWWSRE) | (1<<SPMEN)
 call Do_spm
 ; transfer data from RAM to Flash page buffer
                                ;init loop variable
 ldi looplo, low(PAGESIZEB)
 ldi loophi, high(PAGESIZEB)
                                   ;not required for PAGESIZEB<=256
Wrloop:
 ld r0, Y+
 ld r1, Y+
ldi spmcrval, (1<<SPMEN)</pre>
 call Do_spm
 adiw ZH:ZL, 2
 sbiw loophi:looplo, 2
                                  ;use subi for PAGESIZEB<=256
 brne Wrloop
 ; execute page write
 subi ZL, low(PAGESIZEB)
                                   ; restore pointer
 sbci ZH, high(PAGESIZEB)
                                    ;not required for PAGESIZEB<=256
 ldi spmcrval, (1<<PGWRT) | (1<<SPMEN)
 call Do_spm
 ; re-enable the RWW section
 ldi spmcrval, (1<<RWWSRE) | (1<<SPMEN)
 call Do_spm
  ; read back and check, optional
 ldi looplo, low(PAGESIZEB)
                                   ; init loop variable
 ldi loophi, high(PAGESIZEB)
                                   ;not required for PAGESIZEB<=256
 subi YL, low(PAGESIZEB)
sbci YH, high(PAGESIZEB)
                                 ;restore pointer
Rdloop:
 lpm r0, Z+
ld r1, Y+
 cose r0, r1
 jmp Error
 sbiw loophi:looplo, 1
                                 ;use subi for PAGESIZEB<=256
 brne Rdloop
 ; return to RWW section
 ; verify that RWW section is safe to read
Return:
```



temp1, SPMCR

in

256



```
sbrs temp1, RWWSB
                                 ; If RWWSB is set, the RWW section is not
                                 ; ready yet
  ; re-enable the RWW section
  ldi spmcrval, (1<<RWWSRE) | (1<<SPMEN)
 call Do_spm
  rjmp Return
Do spm:
 ; check for previous SPM complete
Wait_spm:
 in temp1, SPMCR
  sbrc temp1, SPMEN
 rjmp Wait_spm
 ; input: spmcrval determines SPM action
  ; disable interrupts if enabled, store status
 in temp2, SREG
 cli
  ; check that no EEPROM write access is present
Wait_ee:
  sbic EECR, EEWE
  rjmp Wait_ee
  ; SPM timed sequence
 out SPMCR, spmcrval
 spm
  ; restore SREG (to enable interrupts if originally enabled)
 out SREG, temp2
  ret
```

ATmega16 Boot Loader Parameters

In Table 100 through Table 102, the parameters used in the description of the self programming are given.

Table 100. Boot Size Configuration⁽¹⁾

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Section	Boot Loader Flash Section	End Application section	Boot Reset Address (start Boot Loader Section)
1	1	128 words	2	\$0000 - \$1F7F	\$1F80 - \$1FFF	\$1F7F	\$1F80
1	0	256 words	4	\$0000 - \$1EFF	\$1F00 - \$1FFF	\$1EFF	\$1F00
0	1	512 words	8	\$0000 - \$1DFF	\$1E00 - \$1FFF	\$1DFF	\$1E00
0	0	1024 words	16	\$0000 - \$1BFF	\$1C00 - \$1FFF	\$1BFF	\$1C00

Note: 1. The different BOOTSZ Fuse configurations are shown in Figure 125

Table 101. Read-While-Write Limit⁽¹⁾

Section	Pages	Address
Read-While-Write section (RWW)	112	\$0000 - \$1BFF
No Read-While-Write section (NRWW)	16	\$1C00 - \$1FFF

Note: 1. For details about these two section, see "NRWW – No Read-While-Write Section" on page 247 and "RWW – Read-While-Write Section" on page 247



257



Table 102. Explanation of Different Variables used in Figure 126 and the Mapping to the Z-pointer

Variable		Corresponding Z-value ⁽¹⁾	Description
PCMSB	12		Most significant bit in the Program Counter. (The Program Counter is 13 bits PC[12:0])
PAGEMSB	5		Most significant bit which is used to address the words within one page (64 words in a page requires 6 bits PC [5:0]).
ZPCMSB		Z13	Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1.
ZPAGEMSB		Z6	Bit in Z-register that is mapped to PAGEMSB. Because Z0 is not used, the ZPAGEMSB equals PAGEMSB + 1.
PCPAGE	PC[12:6]	Z13:Z7	Program Counter page address: Page select, for Page Erase and Page Write
PCWORD	PC[5:0]	Z6:Z1	Program Counter word address: Word select, for filling temporary buffer (must be zero during page write operation)

Note: 1. Z15:Z14: always ignored

Z0: should be zero for all SPM commands, byte select for the LPM instruction. See "Addressing the Flash during Self-Programming" on page 251 for details about the use of Z-pointer during Self-Programming.





Memory Programming

Program And Data Memory Lock Bits

The ATmega16 provides six Lock bits which can be left unprogrammed ("1") or can be programmed ("0") to obtain the additional features listed in Table 104. The Lock bits can only be erased to "1" with the Chip Erase command.

Table 103. Lock Bit Byte⁽¹⁾

Lock Bit Byte	Bit No.	Description	Default Value
	7	-	1 (unprogrammed)
	6	-	1 (unprogrammed)
BLB12	5	Boot Lock bit	1 (unprogrammed)
BLB11	4	Boot Lock bit	1 (unprogrammed)
BLB02	3	Boot Lock bit	1 (unprogrammed)
BLB01	2	Boot Lock bit	1 (unprogrammed)
LB2	1	Lock bit	1 (unprogrammed)
LB1	0	Lock bit	1 (unprogrammed)

Note: 1. "1" means unprogrammed, "0" means programmed

Table 104. Lock Bit Protection Modes

Memory	Lock Bit	s ⁽²⁾	Protection Type
LB Mode	LB2	LB1	
1	1	1	No memory lock features enabled.
2	1	0	Further programming of the Flash and EEPROM is disabled in Parallel and SPI/JTAG Serial Programming mode. The Fuse bits are locked in both Serial and Parallel Programming mode. ⁽¹⁾
3	0	0	Further programming and verification of the Flash and EEPROM is disabled in Parallel and SPI/JTAG Serial Programming mode. The Fuse bits are locked in both Serial and Parallel Programming mode. (1)
BLB0 Mode	BLB02	BLB01	
1	1	1	No restrictions for SPM or LPM accessing the Application section.
2	1	0	SPM is not allowed to write to the Application section.
3	0	0	SPM is not allowed to write to the Application section, and LPM executing from the Boot Loader section is not allowed to read from the Application section. If interrupt vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.
4	0	1	LPM executing from the Boot Loader section is not allowed to read from the Application section. If interrupt vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.





Table 104. Lock Bit Protection Modes (Continued)

Memory	Memory Lock Bits ⁽²⁾		Protection Type
BLB1 Mode	BLB12	BLB11	
1	1	1	No restrictions for SPM or LPM accessing the Boot Loader section.
2	1	0	SPM is not allowed to write to the Boot Loader section.
3	0	0	SPM is not allowed to write to the Boot Loader section, and LPM executing from the Application section is not allowed to read from the Boot Loader section. If interrupt vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.
4	0	1	LPM executing from the Application section is not allowed to read from the Boot Loader section. If interrupt vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.

otes: 1. Program the Fuse bits before programming the Lock bits.

2. "1" means unprogrammed, "0" means programmed

Fuse Bits

The ATmega16 has two fuse bytes. Table 105 and Table 106 describe briefly the functionality of all the fuses and how they are mapped into the fuse bytes. Note that the fuses are read as logical zero, "0", if they are programmed.

Table 105. Fuse High Byte

Fuse High Byte	Bit No.	Description	Default Value
OCDEN ⁽⁴⁾	7	Enable OCD	1 (unprogrammed, OCD disabled)
JTAGEN ⁽⁵⁾	6	Enable JTAG	0 (programmed, JTAG enabled)
SPIEN ⁽¹⁾	5	Enable SPI Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
CKOPT ⁽²⁾	4	Oscillator options	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BOOTSZ1	2	Select Boot Size (see Table 100 for details)	0 (programmed) ⁽³⁾
BOOTSZ0	1	Select Boot Size (see Table 100 for details)	0 (programmed) ⁽³⁾
BOOTRST	0	Select reset vector	1 (unprogrammed)

Notes: 1. The SPIEN Fuse is not accessible in SPI Serial Programming mode.

- The CKOPT Fuse functionality depends on the setting of the CKSEL bits. See See "Clock Sources" on page 25. for details.
- 3. The default value of BOOTSZ1..0 results in maximum Boot Size. See Table 100 on page 257.
- 4. Never ship a product with the OCDEN Fuse programmed regardless of the setting of Lock bits and the JTAGEN Fuse. A programmed OCDEN Fuse enables some parts of the clock system to be running in all sleep modes. This may increase the power consumption.
- If the JTAG interface is left unconnected, the JTAGEN fuse should if possible be disabled. This to avoid static current at the TDO pin in the JTAG interface.



260



Table 106. Fuse Low Byte

Fuse Low Byte	Bit No.	Description	Default Value
BODLEVEL	7	Brown-out Detector trigger level	1 (unprogrammed)
BODEN	6	Brown-out Detector enable	1 (unprogrammed, BOD disabled)
SUT1	5	Select start-up time	1 (unprogrammed) ⁽¹⁾
SUT0	4	Select start-up time	0 (programmed) ⁽¹⁾
CKSEL3	3	Select Clock source	0 (programmed) ⁽²⁾
CKSEL2	2	Select Clock source	0 (programmed) ⁽²⁾
CKSEL1	1	Select Clock source	0 (programmed) ⁽²⁾
CKSEL0	0	Select Clock source	1 (unprogrammed) ⁽²⁾

Notes: 1. The default value of SUT1..0 results in maximum start-up time. SeeTable 10 on page 29 for details

The default setting of CKSEL3..0 results in internal RC Oscillator @ 1 MHz. See Table 2 on page 25 for details.

The status of the Fuse bits is not affected by Chip Erase. Note that the Fuse bits are locked if Lock bit1 (LB1) is programmed. Program the Fuse bits before programming the Lock bits.

Latching of Fuses

The Fuse values are latched when the device enters programming mode and changes of the Fuse values will have no effect until the part leaves Programming mode. This does not apply to the EESAVE Fuse which will take effect once it is programmed. The fuses are also latched on Power-up in Normal mode.

Signature Bytes

All Atmel microcontrollers have a three-byte signature code which identifies the device. This code can be read in both serial and parallel mode, also when the device is locked. The three bytes reside in a separate address space.

For the ATmega16 the signature bytes are:

- 1. \$000: \$1E (indicates manufactured by Atmel)
- 2. \$001: \$94 (indicates 16 Kbytes Flash memory)
- 3. \$002: \$03 (indicates ATmega16 device when \$001 is \$94)

Calibration Byte

The ATmega16 stores four different calibration values for the internal RC Oscillator. These bytes resides in the signature row High Byte of the addresses 0x0000, 0x0001, 0x0002, and 0x0003 for 1 MHz, 2 MHz, 4 MHz, and 8 Mhz respectively. During Reset, the 1 MHz value is automatically loaded into the OSCCAL Register. If other frequencies are used, the calibration value has to be loaded manually, see "Oscillator Calibration Register – OSCCAL" on page 30 for details.





Page Size

Table 107. No. of Words in a Page and no. of Pages in the Flash

Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
8K words (16 Kbytes)	64 words	PC[5:0]	128	PC[12:6]	12

Table 108. No. of Words in a Page and no. of Pages in the EEPROM

EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
512 bytes	4 bytes	EEA[1:0]	128	EEA[8:2]	8

Parallel Programming Parameters, Pin Mapping, and Commands This section describes how to parallel program and verify Flash Program memory, EEPROM Data memory, Memory Lock bits, and Fuse bits in the ATmega16. Pulses are assumed to be at least 250 ns unless otherwise noted.

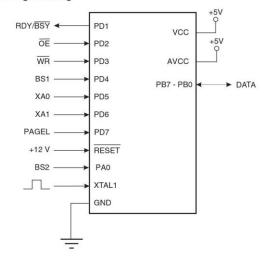
Signal Names

In this section, some pins of the ATmega16 are referenced by signal names describing their functionality during parallel programming, see Figure 127 and Table 109. Pins not described in the following table are referenced by pin names.

The XA1/XA0 pins determine the action executed when the XTAL1 pin is given a positive pulse. The bit coding is shown in Table 111.

When pulsing \overline{WR} or \overline{OE} , the command loaded determines the action executed. The different Commands are shown in Table 112.

Figure 127. Parallel Programming



AMEL



Table 109. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
RDY/BSY	PD1	0	0: Device is busy programming, 1: Device is ready for new command
ŌĒ	PD2	1	Output Enable (Active low)
WR	PD3	1	Write Pulse (Active low)
BS1	PD4	1	Byte Select 1 ("0" selects Low byte, "1" selects High byte)
XA0	PD5	I	XTAL Action Bit 0
XA1	PD6	1	XTAL Action Bit 1
PAGEL	PD7	1	Program Memory and EEPROM data Page Load
BS2	PA0	I	Byte Select 2 ("0" selects Low byte, "1" selects 2'nd High byte)
DATA	PB7-0	I/O	Bidirectional Data bus (Output when OE is low)

Table 110. Pin Values used to Enter Programming Mode

Pin	Symbol	Value
PAGEL	Prog_enable[3]	0
XA1	Prog_enable[2]	0
XA0	Prog_enable[1]	0
BS1	Prog_enable[0]	0

Table 111. XA1 and XA0 Coding

XA1	XA0	Action when XTAL1 is Pulsed
0	0	Load Flash or EEPROM Address (High or low address byte determined by BS1)
0	1	Load Data (High or Low data byte for Flash determined by BS1)
1	0	Load Command
1	1	No Action, Idle

Table 112. Command Byte Bit Coding

Command Byte	Command Executed
1000 0000	Chip Erase
0100 0000	Write Fuse Bits
0010 0000	Write Lock Bits
0001 0000	Write Flash
0001 0001	Write EEPROM





Table 112. Command Byte Bit Coding

Command Byte	Command Executed
0000 1000	Read Signature Bytes and Calibration byte
0000 0100	Read Fuse and Lock bits
0000 0010	Read Flash
0000 0011	Read EEPROM





Parallel Programming

Enter Programming Mode

The following algorithm puts the device in Parallel Programming mode:

- 1. Apply 4.5V 5.5V between V_{CC} and GND, and wait at least 100 μs .
- 2. Set RESET to "0" and toggle XTAL1 at least 6 times
- 3. Set the Prog_enable pins listed in Table 110 on page 263 to "0000" and wait at least 100 ns
- Apply 11.5V 12.5V to RESET. Any activity on Prog_enable pins within 100 ns after +12V has been applied to RESET, will cause the device to fail entering Programming mode.

Note, if External Crystal or External RC configuration is selected, it may not be possible to apply qualified XTAL1 pulses. In such cases, the following algorithm should be followed:

- 1. Set Prog_enable pins listed in Table 110 on page 263 to "0000".
- Apply 4.5V 5.5V between V_{CC} and GND simultaneously as 11.5V 12.5V is applied to RESET.
- 3. Wait 100 µs.
- Re-program the fuses to ensure that External Clock is selected as clock source (CKSEL3:0 = 0b0000) If Lock bits are programmed, a Chip Erase command must be executed before changing the fuses.
- 5. Exit Programming mode by power the device down or by bringing RESET pin to 0b0.
- 6. Entering Programming mode with the original algorithm, as described above.

Considerations for Efficient Programming

The loaded command and address are retained in the device during programming. For efficient programming, the following should be considered.

- The command needs only be loaded once when writing or reading multiple memory locations.
- Skip writing the data value \$FF, that is the contents of the entire EEPROM (unless the EESAVE Fuse is programmed) and Flash after a Chip Erase.
- Address High byte needs only be loaded before programming or reading a new 256 word window in Flash or 256 byte EEPROM. This consideration also applies to Signature bytes reading.

Chip Erase

The Chip Erase will erase the Flash and EEPROM⁽¹⁾ memories plus Lock bits. The Lock bits are not reset until the program memory has been completely erased. The Fuse bits are not changed. A Chip Erase must be performed before the Flash and/or the EEPROM are reprogrammed.

Note: 1. The EEPRPOM memory is preserved during chip erase if the EESAVE Fuse is programmed.

Load Command "Chip Erase"

- 1. Set XA1, XA0 to "10". This enables command loading.
- 2. Set BS1 to "0".
- 3. Set DATA to "1000 0000". This is the command for Chip Erase.
- 4. Give XTAL1 a positive pulse. This loads the command.
- 5. Give WR a negative pulse. This starts the Chip Erase. RDY/BSY goes low.
- 6. Wait until RDY/BSY goes high before loading a new command.



265



Programming the Flash

The Flash is organized in pages, see Table 107 on page 262. When programming the Flash, the program data is latched into a page buffer. This allows one page of program data to be programmed simultaneously. The following procedure describes how to program the entire Flash memory:

A. Load Command "Write Flash"

- 1. Set XA1, XA0 to "10". This enables command loading.
- 2. Set BS1 to "0".
- 3. Set DATA to "0001 0000". This is the command for Write Flash.
- 4. Give XTAL1 a positive pulse. This loads the command.
- B. Load Address Low byte
- 1. Set XA1, XA0 to "00". This enables address loading.
- 2. Set BS1 to "0". This selects low address.
- 3. Set DATA = Address Low byte (\$00 \$FF).
- 4. Give XTAL1 a positive pulse. This loads the address Low byte.
- C. Load Data Low Byte
- 1. Set XA1, XA0 to "01". This enables data loading.
- 2. Set DATA = Data Low byte (\$00 \$FF).
- 3. Give XTAL1 a positive pulse. This loads the data byte.
- D. Load Data High Byte
- 1. Set BS1 to "1". This selects high data byte.
- 2. Set XA1, XA0 to "01". This enables data loading.
- 3. Set DATA = Data High byte (\$00 \$FF).
- 4. Give XTAL1 a positive pulse. This loads the data byte.
- E. Latch Data
- 1. Set BS1 to "1". This selects high data byte.
- Give PAGEL a positive pulse. This latches the data bytes. (See Figure 129 for signal waveforms)
- F. Repeat B through E until the entire buffer is filled or until all data within the page is loaded.

While the lower bits in the address are mapped to words within the page, the higher bits address the pages within the FLASH. This is illustrated in Figure 128 on page 267. Note that if less than 8 bits are required to address words in the page (pagesize < 256), the most significant bit(s) in the address Low byte are used to address the page when performing a page write.

G. Load Address High byte

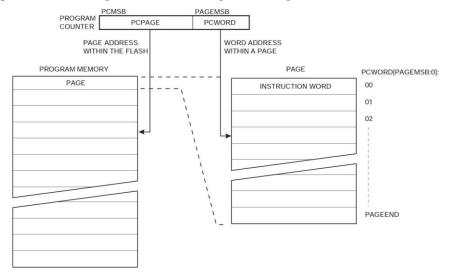
- 1. Set XA1, XA0 to "00". This enables address loading.
- 2. Set BS1 to "1". This selects high address.
- 3. Set DATA = Address High byte (\$00 \$FF).
- 4. Give XTAL1 a positive pulse. This loads the address High byte.
- H. Program Page
- 1. Set BS1 = "0"
- Give WR a negative pulse. This starts programming of the entire page of data. RDY/BSY goes low.
- 3. Wait until RDY/BSY goes high. (See Figure 129 for signal waveforms)





- I. Repeat B through H until the entire Flash is programmed or until all data has been programmed.
- J. End Page Programming
- 1. 1. Set XA1, XA0 to "10". This enables command loading.
- 2. Set DATA to "0000 0000". This is the command for No Operation.
- Give XTAL1 a positive pulse. This loads the command, and the internal write signals are reset.

Figure 128. Addressing the Flash which is Organized in Pages

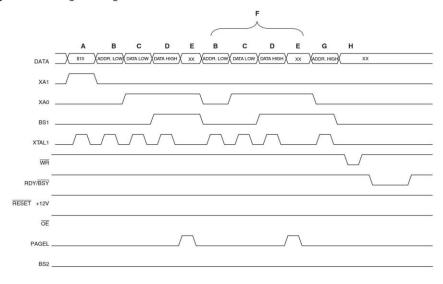


Note: 1. PCPAGE and PCWORD are listed in Table 107 on page 262.





Figure 129. Programming the Flash Waveforms⁽¹⁾



Note: 1. "XX" is don't care. The letters refer to the programming description above.

Programming the EEPROM

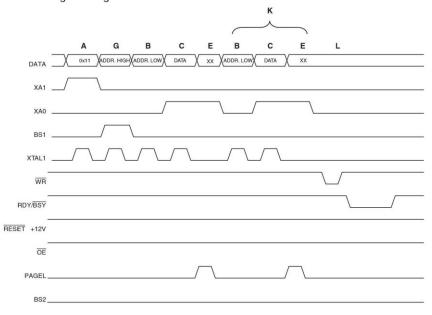
The EEPROM is organized in pages, see Table 108 on page 262. When programming the EEPROM, the program data is latched into a page buffer. This allows one page of data to be programmed simultaneously. The programming algorithm for the EEPROM data memory is as follows (refer to "Programming the Flash" on page 266 for details on Command, Address and Data loading):

- 1. A: Load Command "0001 0001".
- 2. G: Load Address High Byte (\$00 \$FF)
- 3. B: Load Address Low Byte (\$00 \$FF)
- 4. C: Load Data (\$00 \$FF)
- 5. E: Latch data (give PAGEL a positive pulse)
- K: Repeat 3 through 5 until the entire buffer is filled
- L: Program EEPROM page
- 1. Set BS1 to "0".
- Give WR a negative pulse. This starts programming of the EEPROM page. RDY/BSY
 qoes low.
- Wait until to RDY/BSY goes high before programming the next page. (See Figure 130 for signal waveforms)





Figure 130. Programming the EEPROM Waveforms



Reading the Flash

The algorithm for reading the Flash memory is as follows (refer to "Programming the Flash" on page 266 for details on Command and Address loading):

- 1. A: Load Command "0000 0010".
- 2. G: Load Address High Byte (\$00 \$FF)
- 3. B: Load Address Low Byte (\$00 \$FF)
- 4. Set \overline{OE} to "0", and BS1 to "0". The Flash word Low byte can now be read at DATA.
- Set BS1 to "1". The Flash word High byte can now be read at DATA.
- 6. Set OE to "1".

Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to "Programming the Flash" on page 266 for details on Command and Address loading):

- 1. A: Load Command "0000 0011".
- 2. G: Load Address High Byte (\$00 \$FF)
- B: Load Address Low Byte (\$00 \$FF)
- 4. Set \overline{OE} to "0", and BS1 to "0". The EEPROM Data byte can now be read at DATA.
- 5. Set OE to "1".

Programming the Fuse Low Bits

The algorithm for programming the Fuse Low bits is as follows (refer to "Programming the Flash" on page 266 for details on Command and Data loading):

- 1. A: Load Command "0100 0000".
- 2. C: Load Data Low Byte. Bit n = "0" programs and bit n = "1" erases the Fuse bit.
- 3. Set BS1 to "0" and BS2 to "0".
- 4. Give WR a negative pulse and wait for RDY/BSY to go high.



269

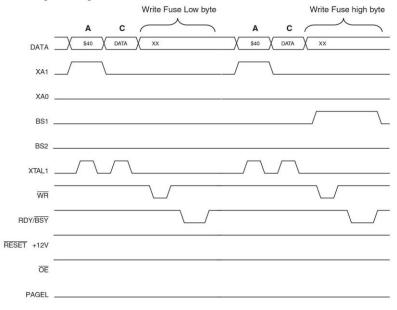


Programming the Fuse High Bits

The algorithm for programming the Fuse high bits is as follows (refer to "Programming the Flash" on page 266 for details on Command and Data loading):

- 1. A: Load Command "0100 0000".
- 2. C: Load Data Low Byte. Bit n = "0" programs and bit n = "1" erases the Fuse bit.
- 3. Set BS1 to "1" and BS2 to "0". This selects high data byte.
- 4. Give WR a negative pulse and wait for RDY/BSY to go high.
- 5. Set BS1 to "0". This selects low data byte.

Figure 131. Programming the Fuses



Programming the Lock Bits

The algorithm for programming the Lock bits is as follows (refer to "Programming the Flash" on page 266 for details on Command and Data loading):

- 1. A: Load Command "0010 0000".
- 2. C: Load Data Low Byte. Bit n = "0" programs the Lock bit.
- 3. Give WR a negative pulse and wait for RDY/BSY to go high.

The Lock bits can only be cleared by executing Chip Erase.

Reading the Fuse and Lock Bits

The algorithm for reading the Fuse and Lock bits is as follows (refer to "Programming the Flash" on page 266 for details on Command loading):

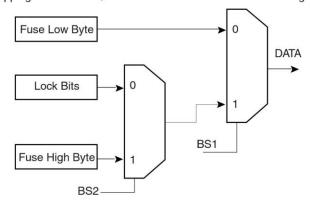
- 1. A: Load Command "0000 0100".
- Set OE to "0", BS2 to "0" and BS1 to "0". The status of the Fuse Low bits can now be read at DATA ("0" means programmed).
- Set OE to "0", BS2 to "1" and BS1 to "1". The status of the Fuse High bits can now be read at DATA ("0" means programmed).
- Set OE to "0", BS2 to "0" and BS1 to "1". The status of the Lock bits can now be read at DATA ("0" means programmed).
- 5. Set OE to "1".



270



Figure 132. Mapping between BS1, BS2 and the Fuse- and Lock Bits during Read



Reading the Signature Bytes

The algorithm for reading the Signature bytes is as follows (refer to "Programming the Flash" on page 266 for details on Command and Address loading):

- 1. A: Load Command "0000 1000".
- 2. B: Load Address Low Byte (\$00 \$02).
- 3. Set OE to "0", and BS1 to "0". The selected Signature byte can now be read at DATA.
- 4. Set OE to "1".

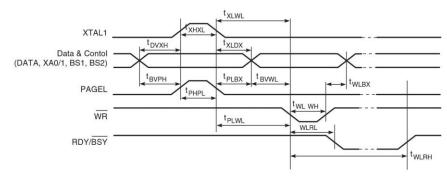
Reading the Calibration Byte

The algorithm for reading the Calibration byte is as follows (refer to "Programming the Flash" on page 266 for details on Command and Address loading):

- 1. A: Load Command "0000 1000".
- 2. B: Load Address Low Byte, \$00.
- 3. Set \overline{OE} to "0", and BS1 to "1". The Calibration byte can now be read at DATA.
- 4. Set OE to "1".

Parallel Programming Characteristics

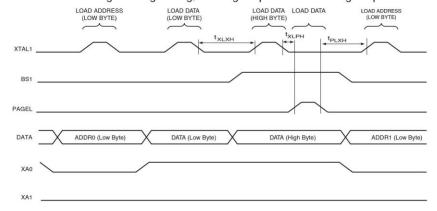
Figure 133. Parallel Programming Timing, Including some General Timing Requirements



AIMEL

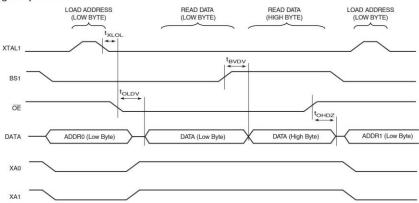


Figure 134. Parallel Programming Timing, Loading Sequence with Timing Requirements⁽¹⁾



Note: 1. The timing requirements shown in Figure 133 (that is, t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to loading operation.

Figure 135. Parallel Programming Timing, Reading Sequence (within the Same Page) with Timing Requirements⁽¹⁾



Note: 1. The timing requirements shown in Figure 133 (that is, t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to reading operation.

Table 113. Parallel Programming Characteristics, $V_{CC} = 5V \pm 10\%$

Symbol	Parameter	Min	Тур	Max	Units
V_{PP}	Programming Enable Voltage	11.5		12.5	V
I _{PP}	Programming Enable Current			250	μА





Table 113. Parallel Programming Characteristics, $V_{CC} = 5V \pm 10\%$ (Continued)

Symbol	Parameter	Min	Тур	Max	Units
t _{DVXH}	Data and Control Valid before XTAL1 High	67			
t _{XLXH}	XTAL1 Low to XTAL1 High	200			
t _{XHXL}	XTAL1 Pulse Width High	150			
t _{XLDX}	Data and Control Hold after XTAL1 Low	67			
t _{XLWL}	XTAL1 Low to WR Low	0			
t _{XLPH}	XTAL1 Low to PAGEL high	0			
t _{PLXH}	PAGEL low to XTAL1 high	150			
t _{BVPH}	BS1 Valid before PAGEL High	67			ns
t _{PHPL}	PAGEL Pulse Width High	150			
t _{PLBX}	BS1 Hold after PAGEL Low	67			
t _{WLBX}	BS2/1 Hold after WR Low	67			
t _{PLWL}	PAGEL Low to WR Low	67			
t _{BVWL}	BS1 Valid to WR Low	67			
t _{WLWH}	WR Pulse Width Low	150			
t _{WLRL}	WR Low to RDY/BSY Low	0		1	μs
t _{WLRH}	WR Low to RDY/BSY High ⁽¹⁾	3.7		4.5	
t _{WLRH_CE}	WR Low to RDY/BSY High for Chip Erase ⁽²⁾	7.5		9	ms
t _{XLOL}	XTAL1 Low to OE Low	0			
t _{BVDV}	BS1 Valid to DATA valid	0		250	
t _{OLDV}	OE Low to DATA Valid			250	ns
t _{OHDZ}	OE High to DATA Tri-stated			250	

Notes: 1. t_{WLRH} is valid for the Write Flash, Write EEPROM, Write Fuse bits and Write Lock bits commands.

2. t_{WLRH_CE} is valid for the Chip Erase command.

Serial Downloading

Both the Flash and EEPROM memory arrays can be programmed using the serial SPI bus while RESET is pulled to GND. The serial interface consists of pins SCK, MOSI (input), and MISO (output). After RESET is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed. NOTE, in Table 114 on page 273, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

SPI Serial Programming Pin Mapping

Table 114. Pin Mapping SPI Serial Programming

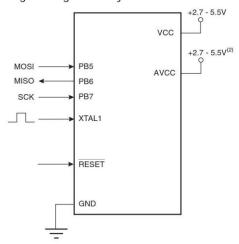
Symbol	Pins	I/O	Description	
MOSI	PB5	1	Serial Data in	
MISO	PB6	0	Serial Data out	
SCK	PB7	1	Serial Clock	



273



Figure 136. SPI Serial Programming and Verify⁽¹⁾



Notes: 1. If the device is clocked by the Internal Oscillator, it is no need to connect a clock source to the XTAL1 pin.

2. V_{CC} -0.3V < AVCC < V_{CC} +0.3V, however, AVCC should always be within 2.7V - 5.5V

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the serial mode ONLY) and there is no need to first execute the Chip Erase instruction. The Chip Erase operation turns the content of every memory location in both the Program and EEPROM arrays into \$FF.

Depending on CKSEL Fuses, a valid clock must be present. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 2 CPU clock cycles for f_{ck} < 12 MHz, 3 CPU clock cycles for f_{ck} \ge 12 MHz

High: > 2 CPU clock cycles for f_{ck} < 12 MHz, 3 CPU clock cycles for f_{ck} \ge 12 MHz

SPI Serial Programming Algorithm When writing serial data to the ATmega16, data is clocked on the rising edge of SCK.

When reading data from the ATmega16, data is clocked on the falling edge of SCK. See Figure 138 for timing details.

To program and verify the ATmega16 in the SPI Serial Programming mode, the following sequence is recommended (See four byte instruction formats in Figure 116 on page 276):

- Power-up sequence: Apply power between V_{CC} and GND while RESET and SCK are set to "0". In some systems, the programmer can not guarantee that SCK is held low during power-up. In this case, RESET must be given a positive pulse of at least two CPU clock cycles duration after SCK has been set to "0".
- 2. Wait for at least 20 ms and enable SPI Serial Programming by sending the Programming Enable serial instruction to pin MOSI.
- 3. The SPI Serial Programming instructions will not work if the communication is out of synchronization. When in sync. the second byte (\$53), will echo back when issuing the third byte of the Programming Enable instruction. Whether the echo is correct or not, all four bytes of the instruction must be transmitted. If the \$53 did not echo back, give RESET a positive pulse and issue a new Programming Enable command.





- 4. The Flash is programmed one page at a time. The page size is found in Table 107 on page 262. The memory page is loaded one byte at a time by supplying the 6 LSB of the address and data together with the Load Program Memory Page instruction. To ensure correct loading of the page, the data Low byte must be loaded before data High byte is applied for a given address. The Program Memory Page is stored by loading the Write Program Memory Page instruction with the 7 MSB of the address. If polling is not used, the user must wait at least t_{WD_FLASH} before issuing the next page. (See Table 115). Accessing the SPI Serial Programming interface before the Flash write operation completes can result in incorrect programming.
- 5. The EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. An EEPROM memory location is first automatically erased before new data is written. If polling is not used, the user must wait at least t_{WD_EEPROM} before issuing the next byte. (See Table 115). In a chip erased device, no \$FFs in the data file(s) need to be programmed.
- Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO.
- At the end of the programming session, RESET can be set high to commence normal operation.
- Power-off sequence (if needed): Set RESET to "1". Turn V_{CC} power off.

Data Polling Flash

When a page is being programmed into the Flash, reading an address location within the page being programmed will give the value \$FF. At the time the device is ready for a new page, the programmed value will read correctly. This is used to determine when the next page can be written. Note that the entire page is written simultaneously and any address within the page can be used for polling. Data polling of the Flash will not work for the value \$FF, so when programming this value, the user will have to wait for at least t_{WD_FLASH} before programming the next page. As a chip erased device contains \$FF in all locations, programming of addresses that are meant to contain \$FF, can be skipped. See Table 115 for t_{WD_FLASH} value

Data Polling EEPROM

When a new byte has been written and is being programmed into EEPROM, reading the address location being programmed will give the value \$FF. At the time the device is ready for a new byte, the programmed value will read correctly. This is used to determine when the next byte can be written. This will not work for the value \$FF, but the user should have the following in mind: As a chip erased device contains \$FF in all locations, programming of addresses that are meant to contain \$FF, can be skipped. This does not apply if the EEPROM is re-programmed without chip erasing the device. In this case, data polling cannot be used for the value \$FF, and the user will have to wait at least t_{WD_EEPROM} before programming the next byte. See Table 115 for t_{WD_EEPROM} value.





Table 115. Minimum Wait Delay before Writing the Next Flash or EEPROM Location

Symbol	Minimum Wait Delay
t _{WD_FUSE}	4.5 ms
t _{WD_FLASH}	4.5 ms
t _{WD_EEPROM}	9.0 ms
t _{WD_ERASE}	9.0 ms

Serial Programming Instruction set Table 116 on page 276 and Figure 137 on page 277 describes the Instruction set.

Table 116. Serial Programming Instruction Set (Hexadecimal values)

		Instructi	on Format	
Instruction ⁽¹⁾ /Operation	Byte 1	Byte 2	Byte 3	Byte 4
Programming Enable	\$AC	\$53	\$00	\$00
Chip Erase (Program Memory/EEPROM)	\$AC	\$80	\$00	\$00
Poll RDY/BSY	\$FO	\$00	\$00	data byte out
Load Instructions		'		<u>'</u>
Load Extended Address byte ⁽¹⁾	\$4D	\$00	Extended adr	\$00
Load Program Memory Page, High byte	\$48	adr MSB	adr LSB	high data byte in
Load Program Memory Page, Low byte	\$40	adr MSB	adr LSB	low data byte in
Load EEPROM Memory Page (page access) ⁽¹⁾	\$C1	\$00	adr LSB	data byte in
Read Instructions		<u> </u>		<u>'</u>
Read Program Memory, High byte	\$28	adr MSB	adr LSB	high data byte out
Read Program Memory, Low byte	\$20	adr MSB	adr LSB	low data byte out
Read EEPROM Memory	\$A0	adr MSB	adr LSB	data byte out
Read Lock bits	\$58	\$00	\$00	data byte out
Read Signature Byte	\$30	\$00	0000 000aa	data byte out
Read Fuse bits	\$50	\$00	\$00	data byte out
Read Fuse High bits	\$58	\$08	\$00	data byte out
Read Extended Fuse Bits	\$50	\$08	\$00	data byte out
Read Calibration Byte	\$38	\$00	\$0b00 000bb	data byte out
Write Instructions			*	do.
Write Program Memory Page	\$4C	000a aaaa	aa00 0000	\$00
Write EEPROM Memory	\$C0	adr MSB	adr LSB	data byte in
Write EEPROM Memory Page (page access) ⁽¹⁾	\$C2	adr MSB	adr LSB	\$00
Write Lock bits	\$AC	\$E0	\$00	data byte in
Write Fuse bits	\$AC	\$A0	\$00	data byte in
Write Fuse High bits	\$AC	\$A8	\$00	data byte in
Write Extended Fuse Bits	\$AC	\$A4	\$00	data byte in
			1	



276



- Notes: 1. Not all instructions are applicable for all parts.
 - 2. a = address
 - 3. Bits are programmed '0', unprogrammed '1'.
 - 4. To ensure future compatibility, unused Fuses and Lock bits should be unprogrammed ('1') .
 - Refer to the correspondig section for Fuse and Lock bits, Calibration and Signature bytes and Page size.
 - 6. See htt://www.atmel.com/avr for Application Notes regarding programming and programmers.

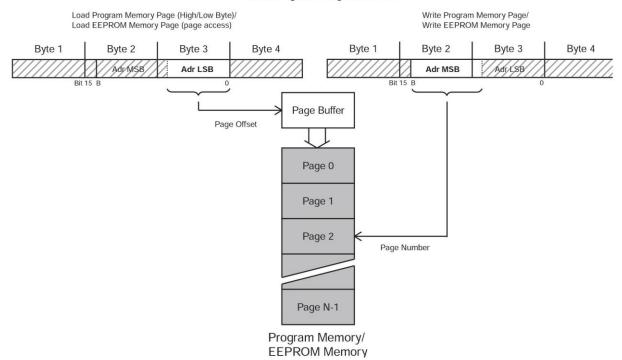
If the LSB in RDY/BSY data byte out is '1', a programming operation is still pending. Wait until this bit returns '0' before the next instruction is carried out.

Within the same page, the low data byte must be loaded prior to the high data byte.

After data is loaded to the page buffer, program the EEPROM page, see Figure 137 on page 277.

Figure 137. Serial Programming Instruction example

Serial Programming Instruction

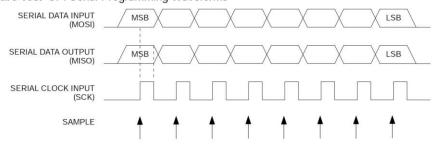






SPI Serial Programming Characteristics For characteristics of the SPI module, see "SPI Timing Characteristics" on page 295.

Figure 138. SPI Serial Programming Waveforms



Programming via the JTAG Interface

Programming through the JTAG interface requires control of the four JTAG specific pins: TCK, TMS, TDI and TDO. Control of the reset and clock pins is not required.

To be able to use the JTAG interface, the JTAGEN Fuse must be programmed. The device is default shipped with the fuse programmed. In addition, the JTD bit in MCUCSR must be cleared. Alternatively, if the JTD bit is set, the External Reset can be forced low. Then, the JTD bit will be cleared after two chip clocks, and the JTAG pins are available for programming. This provides a means of using the JTAG pins as normal port pins in running mode while still allowing In-System Programming via the JTAG interface. Note that this technique can not be used when using the JTAG pins for Boundary-scan or On-chip Debug. In these cases the JTAG pins must be dedicated for this purpose.

As a definition in this datasheet, the LSB is shifted in and out first of all Shift Registers.

Programming Specific JTAG Instructions

The instruction register is 4-bit wide, supporting up to 16 instructions. The JTAG instructions useful for Programming are listed below.

The OPCODE for each instruction is shown behind the instruction name in hex format. The text describes which Data Register is selected as path between TDI and TDO for each instruction.

The Run-Test/Idle state of the TAP controller is used to generate internal clocks. It can also be used as an idle state between JTAG sequences. The state machine sequence for changing the instruction word is shown in Figure 139.





Test-Logic-Reset -0 Select-DR Scan Select-IR Scan Run-Test/Idle 0 0 Capture-DR Capture-IR 0 0**V**...... Shift-DR Shift-IR 0 1 1 Exit1-DR Exit1-IR 0 0 Pause-IR Pause-DR 1 Exit2-DR Exit2-IR 1 Update-DR Update-IR 0 1

Figure 139. State Machine Sequence for Changing the Instruction Word

AVR_RESET (\$C)

The AVR specific public JTAG instruction for setting the AVR device in the Reset mode or taking the device out from the Reset Mode. The TAP controller is not reset by this instruction. The one bit Reset Register is selected as Data Register. Note that the Reset will be active as long as there is a logic "one" in the Reset Chain. The output from this chain is not latched.

The active states are:

· Shift-DR: The Reset Register is shifted by the TCK input.

PROG_ENABLE (\$4)

The AVR specific public JTAG instruction for enabling programming via the JTAG port. The 16-bit Programming Enable Register is selected as Data Register. The active states are the following:

- · Shift-DR: The programming enable signature is shifted into the Data Register.
- Update-DR: The programming enable signature is compared to the correct value, and Programming mode is entered if the signature is valid.



279



PROG_COMMANDS (\$5)

The AVR specific public JTAG instruction for entering programming commands via the JTAG port. The 15-bit Programming Command Register is selected as Data Register. The active states are the following:

- Capture-DR: The result of the previous command is loaded into the Data Register.
- Shift-DR: The Data Register is shifted by the TCK input, shifting out the result of the previous command and shifting in the new command.
- · Update-DR: The programming command is applied to the Flash inputs
- Run-Test/Idle: One clock cycle is generated, executing the applied command (not always required, see Table 117 below).

PROG_PAGELOAD (\$6)

The AVR specific public JTAG instruction to directly load the Flash data page via the JTAG port. The 1024 bit Virtual Flash Page Load Register is selected as Data Register. This is a virtual scan chain with length equal to the number of bits in one Flash page. Internally the Shift Register is 8-bit. Unlike most JTAG instructions, the Update-DR state is not used to transfer data from the Shift Register. The data are automatically transferred to the Flash page buffer byte by byte in the Shift-DR state by an internal state machine. This is the only active state:

 Shift-DR: Flash page data are shifted in from TDI by the TCK input, and automatically loaded into the Flash page one byte at a time.

Note: The JTAG instruction PROG_PAGELOAD can only be used if the AVR device is the first device in JTAG scan chain. If the AVR cannot be the first device in the scan chain, the byte-wise programming algorithm must be used.

PROG_PAGEREAD (\$7)

The AVR specific public JTAG instruction to read one full Flash data page via the JTAG port. The 1032 bit Virtual Flash Page Read Register is selected as Data Register. This is a virtual scan chain with length equal to the number of bits in one Flash page plus 8. Internally the Shift Register is 8-bit. Unlike most JTAG instructions, the Capture-DR state is not used to transfer data to the Shift Register. The data are automatically transferred from the Flash page buffer byte by byte in the Shift-DR state by an internal state machine. This is the only active state:

 Shift-DR: Flash data are automatically read one byte at a time and shifted out on TDO by the TCK input. The TDI input is ignored.

Note: The JTAG instruction PROG_PAGEREAD can only be used if the AVR device is the first device in JTAG scan chain. If the AVR cannot be the first device in the scan chain, the byte-wise programming algorithm must be used.

Data Registers

The Data Registers are selected by the JTAG Instruction Registers described in section "Programming Specific JTAG Instructions" on page 278. The Data Registers relevant for programming operations are:

- Reset Register
- · Programming Enable Register
- Programming Command Register
- Virtual Flash Page Load Register
- Virtual Flash Page Read Register





Reset Register

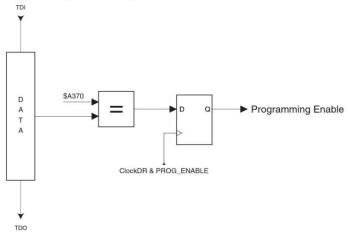
The Reset Register is a Test Data Register used to reset the part during programming. It is required to reset the part before entering programming mode.

A high value in the Reset Register corresponds to pulling the external Reset low. The part is reset as long as there is a high value present in the Reset Register. Depending on the Fuse settings for the clock options, the part will remain reset for a Reset Time-out Period (refer to "Clock Sources" on page 25) after releasing the Reset Register. The output from this Data Register is not latched, so the reset will take place immediately, as shown in Figure 115 on page 230.

Programming Enable Register

The Programming Enable Register is a 16-bit register. The contents of this register is compared to the programming enable signature, binary code 1010_0011_0111_0000. When the contents of the register is equal to the programming enable signature, programming via the JTAG port is enabled. The register is reset to 0 on Power-on Reset, and should always be reset when leaving Programming mode.

Figure 140. Programming Enable Register







Programming Command Register

The Programming Command Register is a 15-bit register. This register is used to serially shift in programming commands, and to serially shift out the result of the previous command, if any. The JTAG Programming Instruction Set is shown in Table 117. The state sequence when shifting in the programming commands is illustrated in Figure 142.

Figure 141. Programming Command Register

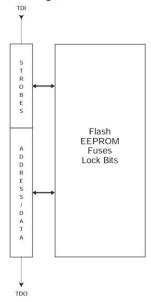






Table 117.JTAG Programming Instruction Seta = address high bits, b = address low bits, H = 0 - Low byte, 1 - High Byte, O = C at a out, O = C at a in, O

Instruction	TDI sequence	TDO sequence	Notes
1a. Chip erase	0100011_10000000	xxxxxxx_xxxxxxxx	
	0110001_10000000	xxxxxxx_xxxxxxx	
	0110011_10000000	XXXXXXX_XXXXXXX	
Al- Dell ferrolling and a second as	0110011_10000000	XXXXXXX_XXXXXXX	(0)
1b. Poll for chip erase complete	0110011_10000000	XXXXX O X_XXXXXXXX	(2)
2a. Enter Flash Write	0100011_00010000	XXXXXXX_XXXXXXX	
2b. Load Address High Byte	0000111_aaaaaaaa	XXXXXXX_XXXXXXX	(9)
2c. Load Address Low Byte	0000011_ bbbbbbbb	xxxxxxx_xxxxxxx	
2d. Load Data Low Byte	0010011_ iiiiiiii	xxxxxxx_xxxxxxxx	
2e. Load Data High Byte	0010111_ iiiiiiii	xxxxxxx_xxxxxxxx	
2f. Latch Data	0110111_00000000	xxxxxxx_xxxxxxxx	(1)
	1110111_00000000	xxxxxxx_xxxxxxx	
	0110111_00000000	XXXXXXX_XXXXXXX	
2g. Write Flash Page	0110111_00000000	XXXXXXX_XXXXXXXX	(1)
	0110101_00000000	xxxxxxx_xxxxxxx	
	0110111_00000000 0110111_00000000	XXXXXXX_XXXXXXXX	
2h Doll for Dago Write complete		XXXXXXX_XXXXXXXX	(2)
2h. Poll for Page Write complete	0110111_00000000	xxxxxox_xxxxxxx	(2)
3a. Enter Flash Read	0100011_00000010	XXXXXXX_XXXXXXX	2.0
3b. Load Address High Byte	0000111_aaaaaaaa	XXXXXXX_XXXXXXX	(9)
3c. Load Address Low Byte	0000011_ bbbbbbbb	xxxxxxx_xxxxxxx	
3d. Read Data Low and High Byte	0110010_00000000	xxxxxxx_xxxxxxx	
	0110110_00000000	XXXXXXX_00000000	Low byte
	0110111_00000000	XXXXXXX_00000000	High byte
4a. Enter EEPROM Write	0100011_00010001	XXXXXXX_XXXXXXX	
4b. Load Address High Byte	0000111_aaaaaaaa	XXXXXXX_XXXXXXX	(9)
4c. Load Address Low Byte	0000011_ bbbbbbbb	xxxxxxx_xxxxxxxx	
4d. Load Data Byte	0010011_iiiiiiiii	xxxxxxx_xxxxxxxx	
4e. Latch Data	0110111_00000000	xxxxxxx_xxxxxxxx	(1)
	1110111_00000000	xxxxxxx_xxxxxxx	
	0110111_00000000	xxxxxxx_xxxxxxx	
4f. Write EEPROM Page	0110011_00000000	xxxxxxx_xxxxxxx	(1)
	0110001_00000000	xxxxxxx_xxxxxxx	
	0110011_00000000	XXXXXXX_XXXXXXXX	
As Doll for Dogo Write complete	0110011_00000000	XXXXXXX_XXXXXXXX	(2)
4g. Poll for Page Write complete	0110011_00000000	xxxxxox_xxxxxxx	(2)
5a. Enter EEPROM Read	0100011_00000011	XXXXXXX_XXXXXXX	7-9
5b. Load Address High Byte	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	(9)
5c. Load Address Low Byte	0000011_ bbbbbbb	xxxxxxx_xxxxxxxx	



283



Table 117. JTAG Programming Instruction Set (Continued)

 \mathbf{a} = address high bits, \mathbf{b} = address low bits, \mathbf{H} = 0 – Low byte, 1 – High Byte, \mathbf{o} = data out, \mathbf{i} = data in, \mathbf{x} = don't care

Instruction	TDI sequence	TDO sequence	Notes
5d. Read Data Byte	0110011_ bbbbbbb 0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_ooooooo	
6a. Enter Fuse Write	0100011_01000000	xxxxxxx_xxxxxxxx	
6b. Load Data Low Byte ⁽⁶⁾	0010011_ iiiiiiii	xxxxxxx_xxxxxxx	(3)
6c. Write Fuse High byte	0110111_00000000 0110101_00000000 0110111_00000000	XXXXXXX_XXXXXXXX XXXXXXX_XXXXXXXX XXXXXX	(1)
6d. Poll for Fuse Write complete	0110111_00000000	xxxxx o x_xxxxxxx	(2)
6e. Load Data Low Byte ⁽⁷⁾	0010011_ iiiiiiii	xxxxxxx_xxxxxxxx	(3)
6f. Write Fuse Low byte	0110011_00000000 0110001_0000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxx	(1)
6g. Poll for Fuse Write complete	0110011_00000000	xxxxx o x_xxxxxxxx	(2)
7a. Enter Lock Bit Write	0100011_00100000	xxxxxxx_xxxxxxxx	
7b. Load Data Byte ⁽⁸⁾	0010011_11 iiiiii	xxxxxxx_xxxxxxxx	(4)
7c. Write Lock Bits	0110011_00000000 0110001_00000000 0110011_00000000	XXXXXXX_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	(1)
7d. Poll for Lock Bit Write complete	0110011_00000000	xxxxx o x_xxxxxxx	(2)
8a. Enter Fuse/Lock Bit Read	0100011_00000100	xxxxxxx_xxxxxxxx	
8b. Read Fuse High Byte ⁽⁶⁾	0111110_00000000 0111111_00000000	xxxxxxx_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	
8c. Read Fuse Low Byte ⁽⁷⁾	0110010_00000000 0110011_00000000	xxxxxxx_ooooooo	
8d. Read Lock Bits ⁽⁸⁾	0110110_00000000 0110111_00000000	xxxxxxx_xxxooooo	(5)
8e. Read Fuses and Lock Bits	0111110_00000000 0110010_00000000 0110110_00000000	xxxxxxx_oooooooooooooooooooooooooooooo	(5) Fuse High Byte Fuse Low Byte Lock bits
9a. Enter Signature Byte Read	0100011_00001000	xxxxxxx_xxxxxxxx	
9b. Load Address Byte	0000011_ bbbbbbb	xxxxxxx_xxxxxxxx	
9c. Read Signature Byte	0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_ 00000000	
10a. Enter Calibration Byte Read	0100011_00001000	xxxxxxx_xxxxxxxx	





Table 117. JTAG Programming Instruction Set (Continued)

a = address high bits, b = address low bits, H = 0 - Low byte, D - Low

Instruction	TDI sequence	TDO sequence	Notes
10b. Load Address Byte	0000011_ bbbbbbb	xxxxxxx_xxxxxxxx	
10c. Read Calibration Byte	0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	
11a. Load No Operation Command	0100011_00000000 0110011_00000000	xxxxxxx_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	

Notes: 1. This command sequence is not required if the seven MSB are correctly set by the previous command sequence (which is normally the case).

- 2. Repeat until o = "1".
- 3. Set bits to "0" to program the corresponding fuse, "1" to unprogram the fuse.
- 4. Set bits to "0" to program the corresponding lock bit, "1" to leave the lock bit unchanged.
- 5. "0" = programmed, "1" = unprogrammed.
- 6. The bit mapping for Fuses High byte is listed in Table 105 on page 260
- 7. The bit mapping for Fuses Low byte is listed in Table 106 on page 261
- 8. The bit mapping for Lock bits byte is listed in Table 103 on page 259
- 9. Address bits exceeding PCMSB and EEAMSB (Table 107 and Table 108) are don't care





Test-Logic-Reset 0 Run-Test/Idle Select-DR Scan Select-IR Scan 0 0 Capture-IR Capture-DR 0 0 Shift-DR Shift-IR 1 Exit1-DR Exit1-IR 0 0 Pause-DR Pause-IR 1 1 Exit2-DR Exit2-IR 1 Update-IR Update-DR 0

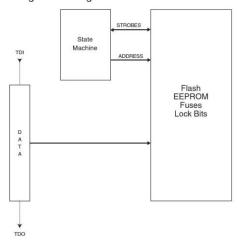
Figure 142. State Machine Sequence for Changing/Reading the Data Word

Virtual Flash Page Load Register The Virtual Flash Page Load Register is a virtual scan chain with length equal to the number of bits in one Flash page. Internally the Shift Register is 8-bit, and the data are automatically transferred to the Flash page buffer byte by byte. Shift in all instruction words in the page, starting with the LSB of the first instruction in the page and ending with the MSB of the last instruction in the page. This provides an efficient way to load the entire Flash page buffer before executing Page Write.



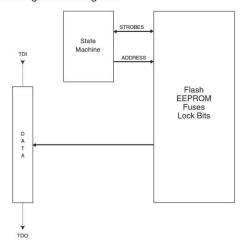


Figure 143. Virtual Flash Page Load Register



Virtual Flash Page Read Register The Virtual Flash Page Read Register is a virtual scan chain with length equal to the number of bits in one Flash page plus 8. Internally the Shift Register is 8-bit, and the data are automatically transferred from the Flash data page byte by byte. The first 8 cycles are used to transfer the first byte to the internal Shift Register, and the bits that are shifted out during these 8 cycles should be ignored. Following this initialization, data are shifted out starting with the LSB of the first instruction in the page and ending with the MSB of the last instruction in the page. This provides an efficient way to read one full Flash page to verify programming.

Figure 144. Virtual Flash Page Read Register





287



Programming Algorithm

All references below of type "1a", "1b", and so on, refer to Table 117.

Entering Programming Mode

- 1. Enter JTAG instruction AVR RESET and shift 1 in the Reset Register.
- Enter instruction PROG_ENABLE and shift 1010_0011_0111_0000 in the Programming Enable Register.

Leaving Programming Mode

- Enter JTAG instruction PROG COMMANDS.
- 2. Disable all programming instructions by using no operation instruction 11a.
- Enter instruction PROG_ENABLE and shift 0000_0000_0000 in the programming Enable Register.
- 4. Enter JTAG instruction AVR_RESET and shift 0 in the Reset Register.

Performing Chip Erase

- Enter JTAG instruction PROG_COMMANDS.
- 2. Start chip erase using programming instruction 1a.
- Poll for Chip Erase complete using programming instruction 1b, or wait for t_{WLRH_CE} (refer to Table 113 on page 272).

Programming the Flash

Before programming the Flash a Chip Erase must be performed. See "Performing Chip Erase" on page 288.

- 1. Enter JTAG instruction PROG_COMMANDS.
- 2. Enable Flash write using programming instruction 2a.
- 3. Load address High byte using programming instruction 2b.
- 4. Load address Low byte using programming instruction 2c.
- 5. Load data using programming instructions 2d, 2e and 2f.
- Repeat steps 4 and 5 for all instruction words in the page.
- 7. Write the page using programming instruction 2g.
- Poll for Flash write complete using programming instruction 2h, or wait for t_{WLRH} (refer to Table 113 on page 272).
- Repeat steps 3 to 7 until all data have been programmed.

A more efficient data transfer can be achieved using the PROG_PAGELOAD instruction:

- 1. Enter JTAG instruction PROG_COMMANDS.
- 2. Enable Flash write using programming instruction 2a.
- Load the page address using programming instructions 2b and 2c. PCWORD (refer to Table 107 on page 262) is used to address within one page and must be written as 0.
- 4. Enter JTAG instruction PROG_PAGELOAD.
- Load the entire page by shifting in all instruction words in the page, starting with the LSB of the first instruction in the page and ending with the MSB of the last instruction in the page.
- 6. Enter JTAG instruction PROG_COMMANDS.
- 7. Write the page using programming instruction 2g.
- Poll for Flash write complete using programming instruction 2h, or wait for t_{WLRH} (refer to Table 113 on page 272).
- 9. Repeat steps 3 to 8 until all data have been programmed.





Reading the Flash

- Enter JTAG instruction PROG_COMMANDS.
- 2. Enable Flash read using programming instruction 3a.
- 3. Load address using programming instructions 3b and 3c.
- 4. Read data using programming instruction 3d.
- 5. Repeat steps 3 and 4 until all data have been read.

A more efficient data transfer can be achieved using the PROG_PAGEREAD instruction:

- 1. Enter JTAG instruction PROG_COMMANDS.
- 2. Enable Flash read using programming instruction 3a.
- Load the page address using programming instructions 3b and 3c. PCWORD (refer to Table 107 on page 262) is used to address within one page and must be written as 0.
- 4. Enter JTAG instruction PROG_PAGEREAD.
- Read the entire page by shifting out all instruction words in the page, starting with the LSB of the first instruction in the page and ending with the MSB of the last instruction in the page. Remember that the first 8 bits shifted out should be ignored.
- 6. Enter JTAG instruction PROG_COMMANDS.
- 7. Repeat steps 3 to 6 until all data have been read.

Programming the EEPROM

Before programming the EEPROM a Chip Erase must be performed. See "Performing Chip Erase" on page 288.

- 1. Enter JTAG instruction PROG_COMMANDS.
- 2. Enable EEPROM write using programming instruction 4a.
- 3. Load address High byte using programming instruction 4b.
- Load address Low byte using programming instruction 4c.
- Load data using programming instructions 4d and 4e.
- 6. Repeat steps 4 and 5 for all data bytes in the page.
- Write the data using programming instruction 4f.
- Poll for EEPROM write complete using programming instruction 4g, or wait for t_{WLRH} (refer to Table 113 on page 272).
- 9. Repeat steps 3 to 8 until all data have been programmed.

Note that the PROG_PAGELOAD instruction can not be used when programming the EEPROM

Reading the EEPROM

- 1. Enter JTAG instruction PROG_COMMANDS.
- 2. Enable EEPROM read using programming instruction 5a.
- 3. Load address using programming instructions 5b and 5c.
- Read data using programming instruction 5d.
- 5. Repeat steps 3 and 4 until all data have been read.

Note that the PROG_PAGEREAD instruction can not be used when reading the EEPROM





Programming the Fuses

- Enter JTAG instruction PROG_COMMANDS.
- 2. Enable Fuse write using programming instruction 6a.
- 3. Load data High byte using programming instructions 6b. A bit value of "0" will program the corresponding fuse, a "1" will unprogram the fuse.
- 4. Write Fuse High byte using programming instruction 6c.
- Poll for Fuse write complete using programming instruction 6d, or wait for t_{WLRH} (refer to Table 113 on page 272).
- Load data Low byte using programming instructions 6e. A "0" will program the fuse, a "1" will unprogram the fuse.
- 7. Write Fuse Low byte using programming instruction 6f.
- Poll for Fuse write complete using programming instruction 6g, or wait for t_{WLRH} (refer to Table 113 on page 272).

Programming the Lock Bits

- Programming the Lock 1. Enter JTAG instruction PROG_COMMANDS.
 - 2. Enable Lock bit write using programming instruction 7a.
 - Load data using programming instructions 7b. A bit value of "0" will program the corresponding Lock bit, a "1" will leave the Lock bit unchanged.
 - 4. Write Lock bits using programming instruction 7c.
 - Poll for Lock bit write complete using programming instruction 7d, or wait for t_{WLRH} (refer to Table 113 on page 272).

Reading the Fuses and Lock Bits

- 1. Enter JTAG instruction PROG_COMMANDS.
- 2. Enable Fuse/Lock bit read using programming instruction 8a.
- To read all Fuses and Lock bits, use programming instruction 8e.
 To only read Fuse High byte, use programming instruction 8b.
 To only read Fuse Low byte, use programming instruction 8c.
 To only read Lock bits, use programming instruction 8d.

Reading the Signature Bytes

- Enter JTAG instruction PROG_COMMANDS.
- 2. Enable Signature byte read using programming instruction 9a.
- 3. Load address \$00 using programming instruction 9b.
- 4. Read first signature byte using programming instruction 9c.
- Repeat steps 3 and 4 with address \$01 and address \$02 to read the second and third signature bytes, respectively.

Reading the Calibration Byte

- 1. Enter JTAG instruction PROG_COMMANDS.
- 2. Enable Calibration byte read using programming instruction 10a.
- 3. Load address \$00 using programming instruction 10b.
- 4. Read the calibration byte using programming instruction 10c.





Electrical Characteristics

Absolute Maximum Ratings*

V
Operating Temperature55°C to +125°C
Storage Temperature65°C to +150°C
Voltage on any Pin except RESET with respect to Ground0.5V to V _{CC} +0.5V
Voltage on RESET with respect to Ground0.5V to +13.0V
Maximum Operating Voltage 6.0V
DC Current per I/O Pin40.0 mA
DC Current V _{CC} and GND Pins200.0 mA PDIP and
400.0 mA TQFP/MLF

*NOTICE:

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

 T_A = -40°C to 85°C, V_{CC} = 2.7V to 5.5V (Unless Otherwise Noted)

Symbol	Parameter	Condition	Min	Тур	Max	Units	
V _{IL}	Input Low Voltage except XTAL1 and RESET pins	V _{CC} = 2.7 - 5.5	-0.5		0.2 V _{CC} ⁽¹⁾		
V _{IH}	Input High Voltage except XTAL1 and RESET pins	V _{CC} = 2.7 - 5.5	0.6 V _{CC} ⁽²⁾		V _{CC} +0.5		
V _{IH1}	Input High Voltage XTAL1 pin	V _{CC} = 2.7 - 5.5	0.7 V _{cc} ⁽²⁾		V _{CC} +0.5		
V _{IL1}	Input Low Voltage XTAL1 pin	V _{CC} = 2.7 - 5.5	-0.5		0.1 V _{CC} ⁽¹⁾	V	
V _{IH2}	Input High Voltage RESET pin	V _{CC} = 2.7 - 5.5	0.9 V _{CC} ⁽²⁾		V _{CC} +0.5		
V _{IL2}	Input Low Voltage RESET pin	V _{CC} = 2.7 - 5.5	-0.5		0.2 V _{CC}		
V _{OL}	Output Low Voltage ⁽³⁾ (Ports A,B,C,D)	I _{OL} = 20 mA, V _{CC} = 5V I _{OL} = 10 mA, V _{CC} = 3V			0.7 0.5	V V	
V _{OH}	Output High Voltage ⁽⁴⁾ (Ports A,B,C,D)	I_{OH} = -20 mA, V_{CC} = 5V I_{OH} = -10 mA, V_{CC} = 3V	4.2 2.2			V V	
I _{IL}	Input Leakage Current I/O Pin	Vcc = 5.5V, pin low (absolute value)			1		
I _{IH}	Input Leakage Current I/O Pin	Vcc = 5.5V, pin high (absolute value)			1	μА	
R _{RST}	Reset Pull-up Resistor		30		60	kO	
R _{pu}	I/O Pin Pull-up Resistor		20		50	kΩ	





 $T_A = -40$ °C to 85°C, $V_{CC} = 2.7$ V to 5.5V (Unless Otherwise Noted) (Continued)

Symbol	Parameter	Condition	Min	Тур	Max	Units
	Power Supply Current Power-down Mode ⁽⁵⁾ Analog Comparator Input Offset Voltage	Active 1 MHz, V _{CC} = 3V (ATmega16L)		1.1		
		Active 4 MHz, V _{CC} = 3V (ATmega16L)		3.8	5	
		Active 8 MHz, V _{CC} = 5V (ATmega16)		12	15	
I _{CC}		Idle 1 MHz, V _{CC} = 3V (ATmega16L)		0.35		- mA
		Idle 4 MHz, V _{CC} = 3V (ATmega16L)		1.2	2	
		Idle 8 MHz, V _{CC} = 5V (ATmega16)		5.5	7	
		WDT enabled, V _{CC} = 3V		<8	15	
		WDT disabled, V _{CC} = 3V		< 1	4	μΑ
V _{ACIO}		$V_{CC} = 5V$ $V_{in} = V_{CC}/2$			40	mV
I _{ACLK}	Analog Comparator Input Leakage Current	$V_{CC} = 5V$ $V_{in} = V_{CC}/2$	-50		50	nA
t _{ACPD}	Analog Comparator Propagation Delay	V _{CC} = 2.7V V _{CC} = 4.0V		750 500		ns

- Notes: 1. "Max" means the highest value where the pin is guaranteed to be read as low
 - 2. "Min" means the lowest value where the pin is guaranteed to be read as high
 - 3. Although each I/O port can sink more than the test conditions (20 mA at Vcc = 5V, 10 mA at Vcc = 3V) under steady state conditions (non-transient), the following must be observed: PDIP Package:
 - 1] The sum of all IOL, for all ports, should not exceed 200 mA.
 - 2] The sum of all IOL, for port A0 A7, should not exceed 100 mA.
 - 3] The sum of all IOL, for ports B0 B7,C0 C7, D0 D7 and XTAL2, should not exceed 100 mA.

TQFP and QFN/MLF Package:

- 1] The sum of all IOL, for all ports, should not exceed 400 mA.
- 2] The sum of all IOL, for ports A0 A7, should not exceed 100 mA.
- 3] The sum of all IOL, for ports B0 B4, should not exceed 100 mA.
- 4] The sum of all IOL, for ports B3 B7, XTAL2, D0 D2, should not exceed 100 mA.
- 5] The sum of all IOL, for ports D3 D7, should not exceed 100 mA.
- 6] The sum of all IOL, for ports C0 C7, should not exceed 100 mA.
- If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
- 4. Although each I/O port can source more than the test conditions (20 mA at Vcc = 5V, 10 mA at Vcc = 3V) under steady state conditions (non-transient), the following must be observed:
 - PDIP Package:
 - 1] The sum of all IOH, for all ports, should not exceed 200 mA.
 - 2] The sum of all IOH, for port A0 A7, should not exceed 100 mA.
 - 3] The sum of all IOH, for ports B0 B7,C0 C7, D0 D7 and XTAL2, should not exceed 100 mA.

TQFP and QFN/MLF Package:

- 1] The sum of all IOH, for all ports, should not exceed 400 mA.
- 2] The sum of all IOH, for ports A0 A7, should not exceed 100 mA.
- 3] The sum of all IOH, for ports B0 B4, should not exceed 100 mA.
- 4] The sum of all IOH, for ports B3 B7, XTAL2, D0 D2, should not exceed 100 mA.



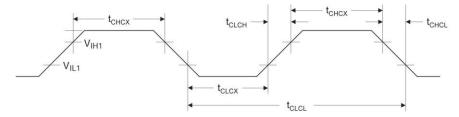
292



- 5] The sum of all IOH, for ports D3 D7, should not exceed 100 mA.
- 6] The sum of all IOH, for ports C0 C7, should not exceed 100 mA.If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
- 5. Minimum V_{CC} for Power-down is 2.5V.

External Clock Drive Waveforms

Figure 145. External Clock Drive Waveforms



External Clock Drive

Table 118. External Clock Drive(1)

		$V_{CC} = 2.7$	7V to 5.5V	$V_{CC} = 4.5V \text{ to } 5.5V$		
Symbol	ol Parameter	Min	Max	Min	Max	Units
1/t _{CLCL}	Oscillator Frequency	0	8	0	16	MHz
t _{CLCL}	Clock Period	125		62.5		
t _{CHCX}	High Time	50		25		ns
t _{CLCX}	Low Time	50		25		
t _{CLCH}	Rise Time		1.6		0.5	
t _{CHCL}	Fall Time		1.6		0.5	μS
Δt_{CLCL}	Change in period from one clock cycle to the next		2		2	%

Note: 1. Refer to "External Clock" on page 31 for details.

Table 119. External RC Oscillator, Typical Frequencies ($V_{CC} = 5$)

R [kΩ] ⁽¹⁾	C [pF]	f ⁽²⁾
33	22	650 kHz
10	22	2.0 MHz

Notes: 1. R should be in the range 3 k Ω - 100 k Ω , and C should be at least 20 pF.

2. The frequency will vary with package type and board layout.





Two-wire Serial Interface Characteristics

Table 120 describes the requirements for devices connected to the Two-wire Serial Bus. The ATmega16 Two-wire Serial Interface meets or exceeds these requirements under the noted conditions.

Timing symbols refer to Figure 146.

Table 120. Two-wire Serial Bus Requirements

Symbol	Parameter	Condition	Min	Max	Units	
V _{IL}	Input Low-voltage		-0.5	0.3 V _{CC}		
V _{IH}	Input High-voltage		0.7 V _{CC}	$V_{CC} + 0.5$	V	
V _{hys} ⁽¹⁾	Hysteresis of Schmitt Trigger Inputs		0.05 V _{CC} ⁽²⁾	-	V	
V _{OL} ⁽¹⁾	Output Low-voltage	3 mA sink current	0	0.4		
(1) T	Rise Time for both SDA and SCL		20 + 0.1C _b ⁽³⁾⁽²⁾	300		
(1) of	Output Fall Time from V _{IHmin} to V _{ILmax}	10 pF < C _b < 400 pF ⁽³⁾	20 + 0.1C _b ⁽³⁾⁽²⁾	250	ns	
t _{SP} ⁽¹⁾	Spikes Suppressed by Input Filter		0	50 ⁽²⁾		
li	Input Current each I/O Pin	$0.1V_{CC} < V_i < 0.9V_{CC}$	-10	10	μA	
C _i ⁽¹⁾	Capacitance for each I/O Pin		-	10	pF	
f _{SCL}	SCL Clock Frequency	$f_{CK}^{(4)} > max(16f_{SCL}, 250kHz)^{(5)}$	0	400	kHz	
	Value of Pull-up resistor	f _{SCL} ≤ 100 kHz	$\frac{V_{CC} - 0.4V}{3 \text{ mA}}$	$\frac{1000 \mathrm{ns}}{C_b}$		
Rp		f _{SCL} > 100 kHz	$\frac{V_{CC} - 0.4V}{3 \text{ mA}}$	$\frac{300 \text{ns}}{C_b}$	Ω	
HD;STA	Hold Time (repeated) START Condition	f _{SCL} ≤ 100 kHz	4.0	_		
HD;SIA	, , , , , , , , , , , , , , , , , , ,	f _{SCL} > 100 kHz	0.6	-		
LOW	Low Period of the SCL Clock	f _{SCL} ≤ 100 kHz	4.7	-		
LOW		f _{SCL} > 100 kHz	1.3	_		
HIGH	High period of the SCL clock	f _{SCL} ≤ 100 kHz	4.0		7000	
niGn		f _{SCL} > 100 kHz	0.6	_	μs	
SU;STA		f _{SCL} ≤ 100 kHz	4.7	_		
30,31A	Set-up time for a repeated START condition	f _{SCL} > 100 kHz	0.6	=		
HD;DAT	Data hold time	f _{SCL} ≤ 100 kHz	0	3.45		
nu,uai		f _{SCL} > 100 kHz	0	0.9		
SU:DAT	Data setup time	f _{SCL} ≤ 100 kHz	250	=	100000	
JU,DAI		f _{SCL} > 100 kHz 100		-	ns	
SU:STO	Setup time for STOP condition	f _{SCL} ≤ 100 kHz	4.0	-		
30;310		f _{SCL} > 100 kHz	0.6	_		
t _{BUF}	Bus free time between a STOP and START	f _{SCL} ≤ 100 kHz	4.7	_	μs	
DUF	condition	f _{SCL} > 100 kHz	1.3	_		

- Notes: 1. In ATmega16, this parameter is characterized and not 100% tested.

 - Required only for f_{SCL} > 100 kHz.
 C_b = capacitance of one bus line in pF.
 f_{CK} = CPU clock frequency



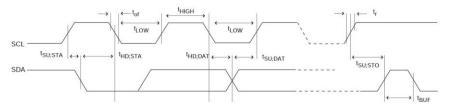
294

2466T-AVR-07/10



 This requirement applies to all ATmega16 Two-wire Serial Interface operation. Other devices connected to the Two-wire Serial Bus need only obey the general f_{SCL} requirement.

Figure 146. Two-wire Serial Bus Timing



SPI Timing Characteristics

See Figure 147 and Figure 148 for details.

Table 121. SPI Timing Parameters

	Description	Mode	Min	Тур	Max	
1	SCK period	Master		See Table 58		
2	SCK high/low	Master		50% duty cycle		
3	Rise/Fall time	Master		3.6		
4	Setup	Master		10		
5	Hold	Master		10		
6	Out to SCK	Master		0.5 • t _{SCK}		ns
7	SCK to out	Master		10		
8	SCK to out high	Master		10		
9	SS low to out	Slave		15		
10	SCK period	Slave	4 • t _{SCK}			
11	SCK high/low	Slave	2 · t _{SCK}			
12	Rise/Fall time	Slave			1.6	μs
13	Setup	Slave	10			
14	Hold	Slave	10			
15	SCK to out	Slave		15		
16	SCK to SS high	Slave	20			ns
17	SS high to tri-state	Slave		10		
18	SS low to SCK	Slave	2 · t _{SCK}			





Figure 147. SPI Interface Timing Requirements (Master Mode)

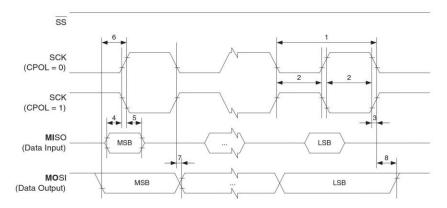
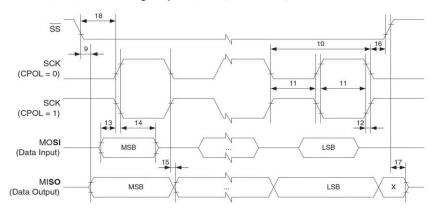


Figure 148. SPI Interface Timing Requirements (Slave Mode)







ADC Characteristics

Table 122. ADC Characteristics

Symbol	Parameter	Condition	Min ⁽¹⁾	Typ ⁽¹⁾	Max ⁽¹⁾	Units	
		Single Ended Conversion		10			
	Resolution	Differential Conversion Gain = 1x or 10x		8		Bits	
		Differential Conversion Gain = 200x		7			
		Single Ended Conversion $V_{REF} = 4V$, $V_{CC} = 4V$ ADC clock = 200 kHz		1.5	2.5		
	Absolute Accuracy (Including INL, DNL, Quantization Error, Gain, and Offset Error).	Single Ended Conversion V _{REF} = 4V, V _{CC} = 4V ADC clock = 1 MHz		3	4		
		Single Ended Conversion $V_{REF} = 4V$, $V_{CC} = 4V$ ADC clock = 200 kHz Noise Reduction mode		1.5			
		Single Ended Conversion $V_{REF} = 4V$, $V_{CC} = 4V$ ADC clock = 1 MHz Noise Reduction mode		3		LSB	
	Integral Non-linearity (INL)	Single Ended Conversion $V_{REF} = 4V$, $V_{CC} = 4V$ ADC clock = 200 kHz		1			
	Differential Non-linearity (DNL)	Single Ended Conversion $V_{REF} = 4V$, $V_{CC} = 4V$ ADC clock = 200 kHz		0.5			
	Gain Error	Single Ended Conversion V _{REF} = 4V, V _{CC} = 4V ADC clock = 200 kHz		1			
	Offset Error	Single Ended Conversion V _{REF} = 4V, V _{CC} = 4V ADC clock = 200 kHz					
	Conversion Time	Free Running Conversion	13		260	μs	
	Clock Frequency		50		1000	kHz	
AVCC	Analog Supply Voltage		V _{CC} - 0.3 ⁽²⁾		$V_{CC} + 0.3^{(3)}$		
V_{REF}	Reference Voltage	Single Ended Conversion	2.0		AVCC		
* REF	Reference voltage	Differential Conversion	2.0		AVCC - 0.2	V	
	Input voltage	Single ended channels	GND		V _{REF}		
V _{IN}		Differential channels	0		V _{REF}		
V IN	Input bandwidth	Single ended channels		38.5		kHz	
	The salidition	Differential channels		4		IN 12	





Table 122. ADC Characteristics (Continued)

Symbol	Parameter	Condition	Min ⁽¹⁾	Typ ⁽¹⁾	Max ⁽¹⁾	Units
V_{INT}	Internal Voltage Reference		2.3	2.6	2.9	V
R _{REF}	Reference Input Resistance			32		kΩ
R _{AIN}	Analog Input Resistance			100		MΩ

- Notes: 1. Values are guidelines only.
 - 2. Minimum for AVCC is 2.7V.
 - 3. Maximum for AVCC is 5.5V.





ATmega16 Typical Characteristics

The following charts show typical behavior. These figures are not tested during manufacturing. All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled. A sine wave generator with rail-to-rail output is used as clock source.

The power consumption in Power-down mode is independent of clock selection.

The current consumption is a function of several factors such as: operating voltage, operating frequency, loading of I/O pins, switching rate of I/O pins, code executed and ambient temperature. The dominating factors are operating voltage and frequency.

The current drawn from capacitive loaded pins may be estimated (for one pin) as $C_L*V_{cc}*f$ where C_L = load capacitance, V_{cc} = operating voltage and f = average switching frequency of I/O pin.

The parts are characterized at frequencies higher than test limits. Parts are not guaranteed to function properly at frequencies higher than the ordering code indicates.

The difference between current consumption in Power-down mode with Watchdog Timer enabled and Power-down mode with Watchdog Timer disabled represents the differential current drawn by the Watchdog Timer.

Active Supply Current

Figure 149. Active Supply Current vs. Frequency (0.1 MHz - 1.0 MHz

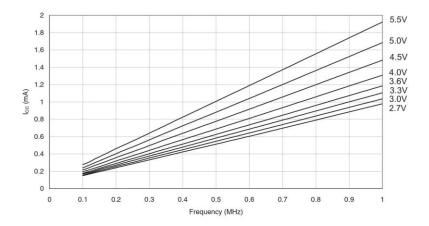






Figure 150. Active Supply Current vs. Frequency (1 MHz - 20 MHz)

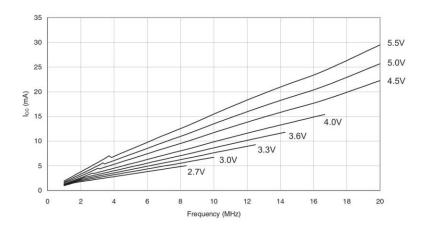


Figure 151. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)

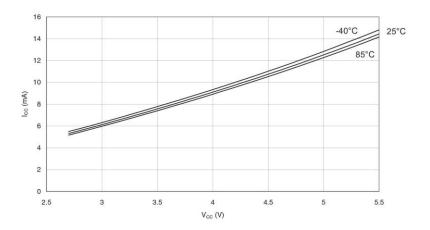




Figure 152. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 4 MHz)

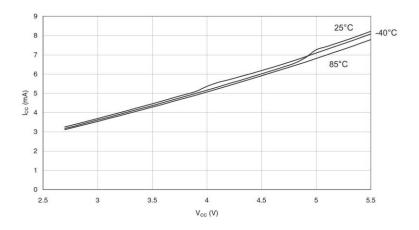


Figure 153. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 2 MHz)

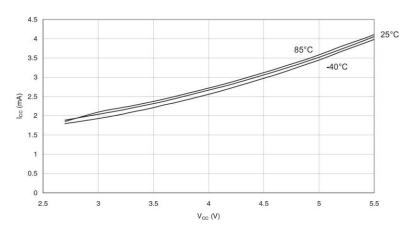






Figure 154. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 1 MHz)

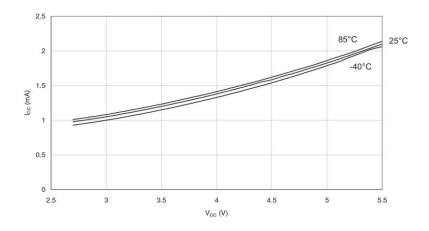
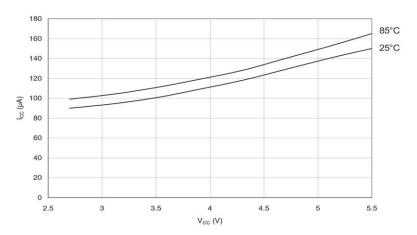


Figure 155. Active Supply Current vs. V_{CC} (32 kHz External Oscillator)





Idle Supply Current

Figure 156. Idle Supply Current vs. Frequency (0.1 MHz - 1.0 MHz)

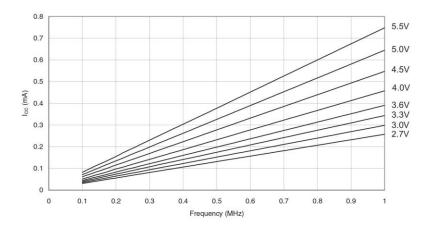


Figure 157. Idle Supply Current vs. Frequency (1 MHz - 20 MHz)

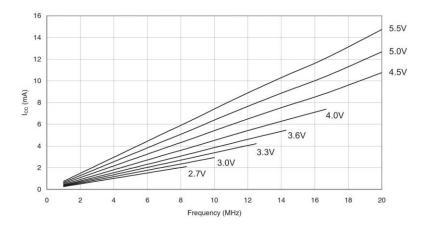




Figure 158. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)

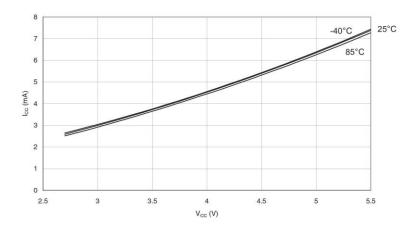


Figure 159. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 4 MHz)

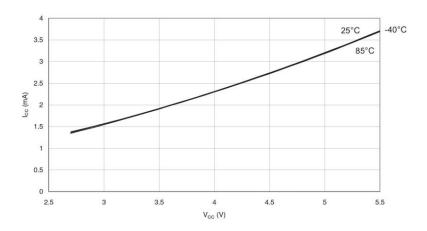




Figure 160. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 2 MHz)

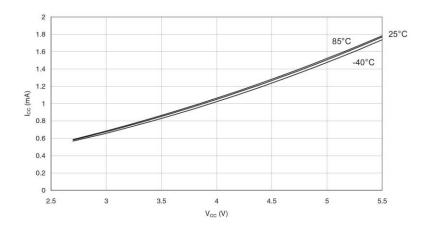


Figure 161. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 1 MHz)

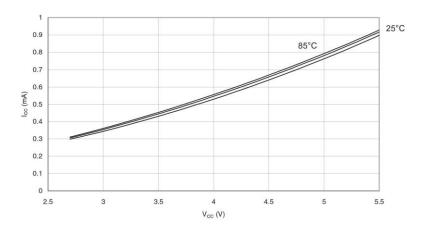
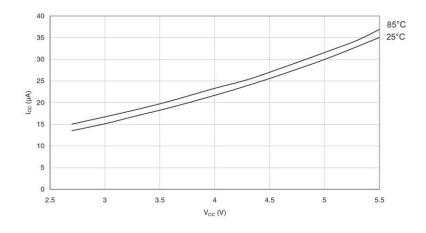




Figure 162. Idle Supply Current vs. V_{CC} (32 kHz External Oscillator)



Power-Down Supply Current

Figure 163. Power-Down Supply Current vs. V_{CC} (Watchdog Timer Disabled)

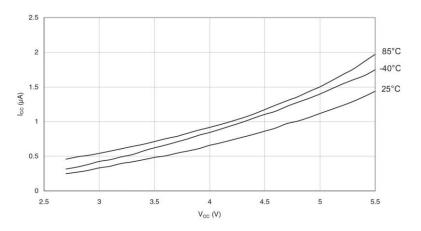
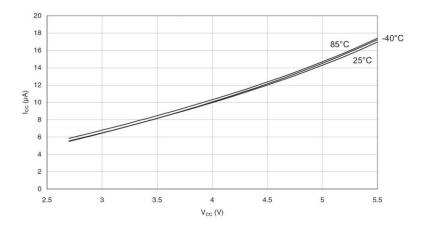


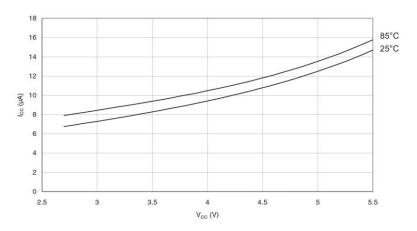


Figure 164. Power-Down Supply Current vs. V_{CC} (Watchdog Timer Enabled)



Power-Save Supply Current

 $\textbf{Figure 165.} \ \ \text{Power-Save Supply Current vs. V}_{\text{CC}} \ \ (\text{Watchdog Timer Disabled})$





Standby Supply Current

Figure 166. Standby Supply Current vs. V_{CC} (455 kHz Resonator, Watchdog Timer Disabled)

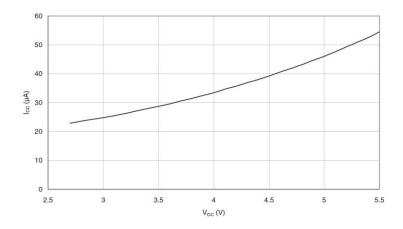


Figure 167. Standby Supply Current vs. V_{CC} (1 MHz Resonator, Watchdog Timer Disabled)

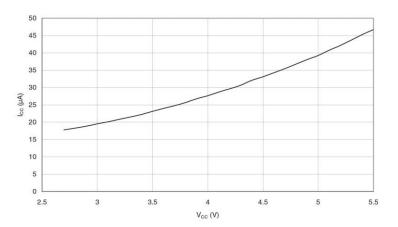






Figure 168. Standby Supply Current vs. V_{CC} (2 MHz Resonator, Watchdog Timer Disabled)

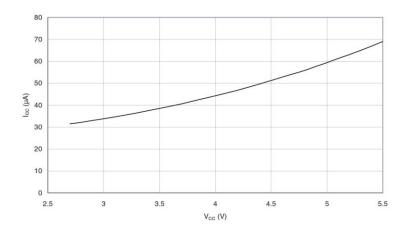


Figure 169. Standby Supply Current vs. V_{CC} (2 MHz Xtal, Watchdog Timer Disabled)

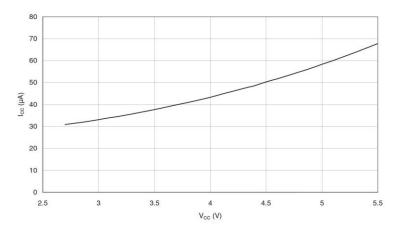




Figure 170. Standby Supply Current vs. V_{CC} (4 MHz Resonator, Watchdog Timer Disabled)

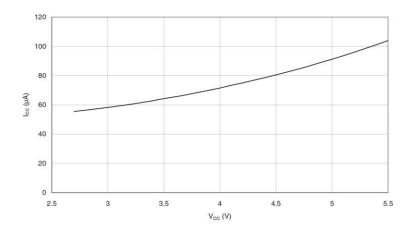


Figure 171. Standby Supply Current vs. V_{CC} (4 MHz Xtal, Watchdog Timer Disabled)

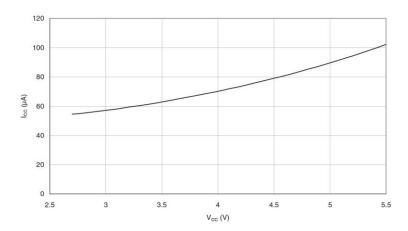




Figure 172. Standby Supply Current vs. V_{CC} (6 MHz Resonator, Watchdog Timer Disabled)

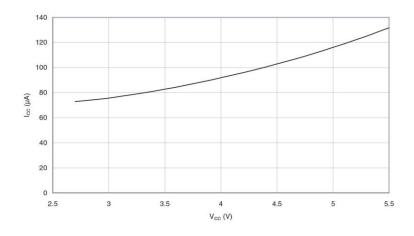
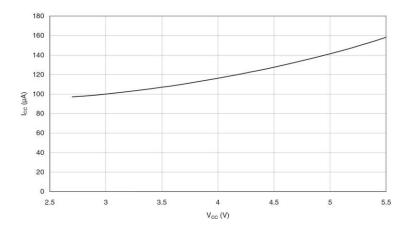


Figure 173. Standby Supply Current vs. V_{CC} (6 MHz Xtal, Watchdog Timer Disabled)





Pin Pullup

Figure 174. I/O Pin Pull-Up Resistor Current vs. Input Voltage (V_{CC} = 5V)

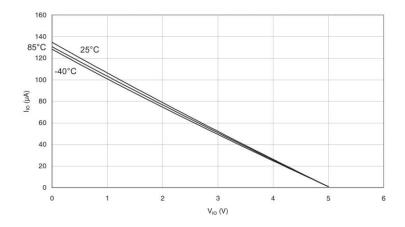


Figure 175. I/O Pin Pull-Up Resistor Current vs. Input Voltage ($V_{CC} = 2.7V$)

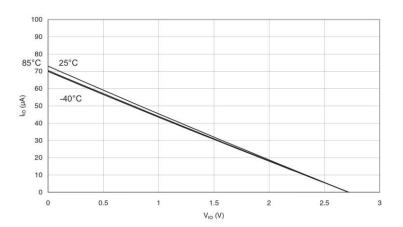




Figure 176. Reset Pull-Up Resistor Current vs. Reset Pin Voltage (V_{CC} = 5V)

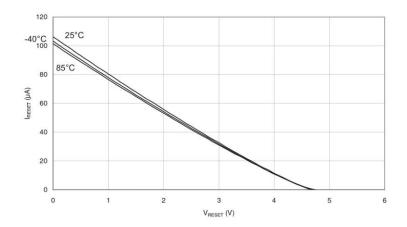
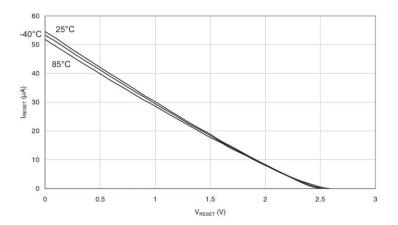


Figure 177. Reset Pull-Up Resistor Current vs. Reset Pin Voltage (V_{CC} = 2.7V)





Pin Driver Strength Figure 178. I/O Pin Source Current vs. Output Voltage (V_{CC} = 5V)

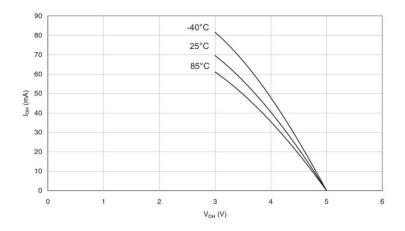


Figure 179. I/O Pin Source Current vs. Output Voltage ($V_{CC} = 2.7V$)

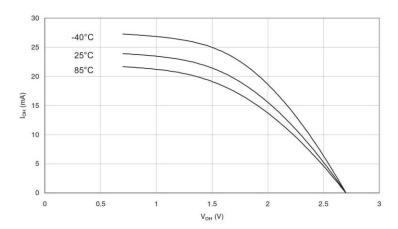




Figure 180. I/O Pin Sink Current vs. Output Voltage ($V_{CC} = 5V$)

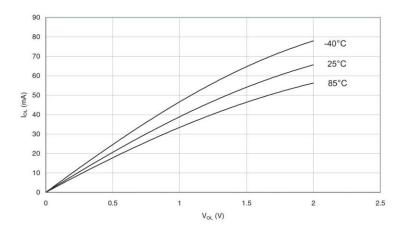
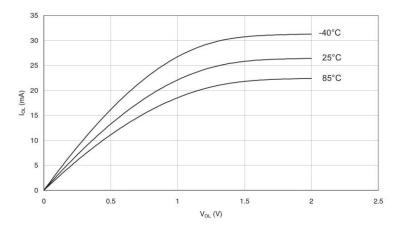


Figure 181. I/O Pin Sink Current vs. Output Voltage ($V_{CC} = 2.7V$)





Pin Thresholds And Hysteresis

Figure 182. I/O Pin Input Threshold Voltage vs. V_{CC} (V_{IH} , I/O Pin Read As '1')

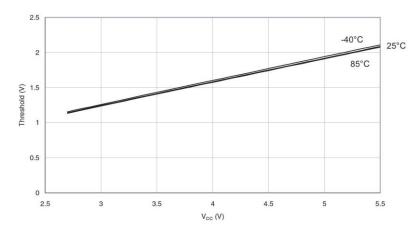


Figure 183. I/O Pin Input Threshold Voltage vs. V_{CC} (V_{IL} , I/O Pin Read As '0')

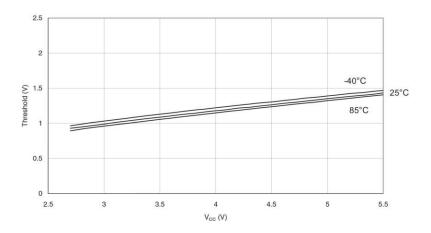




Figure 184. I/O Pin Input Hysteresis vs. V_{CC}

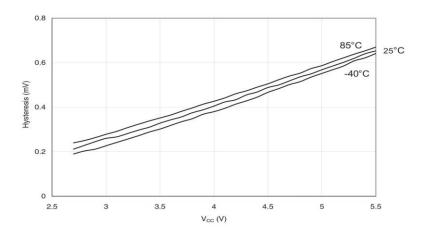


Figure 185. Reset Input Threshold Voltage vs. V_{CC} (V_{IH} , Reset Pin Read As '1')

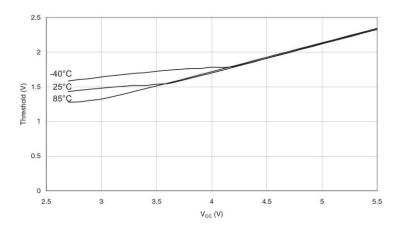




Figure 186. Reset Input Threshold Voltage vs. V_{CC} (V_{IL}, Reset Pin Read As '0')

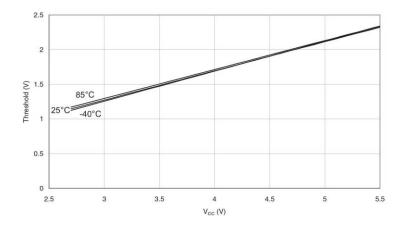
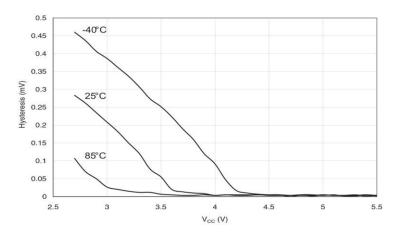


Figure 187. Reset Input Pin Hysteresis vs. $V_{\rm CC}$





Bod Thresholds And Analog Comparator Offset

Figure 188. Bod Thresholds vs. Temperature (Bodlevel is 4.0V)

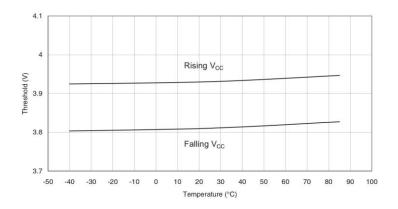


Figure 189. Bod Thresholds vs. Temperature (Bodlevel is 2.7V)

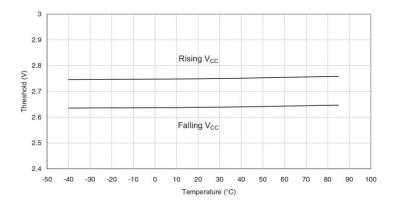






Figure 190. Bandgap Voltage vs. V_{CC}

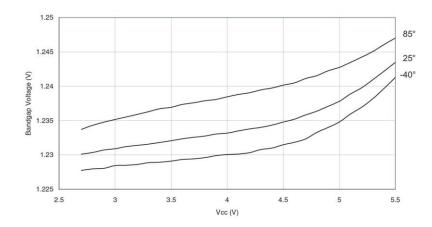


Figure 191. Analog Comparator Offset Voltage vs. Common Mode Voltage ($V_{CC} = 5V$)

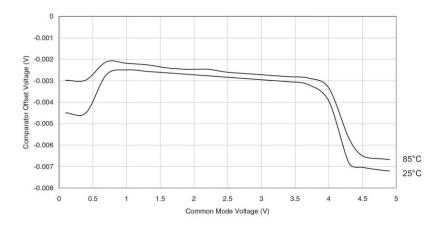
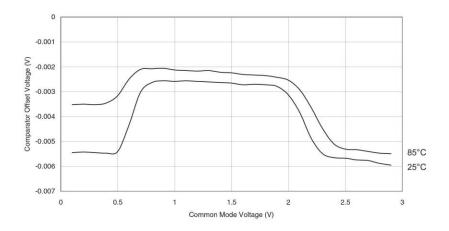




Figure 192. Analog Comparator Offset Voltage vs. Common Mode Voltage ($V_{CC} = 3V$)



Internal Oscillator Speed

Figure 193. Watchdog Oscillator Frequency vs. V_{CC}

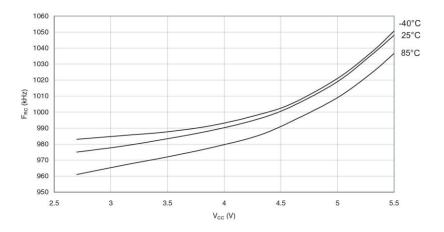




Figure 194. Calibrated 8 MHz RC Oscillator Frequency vs. Temperature

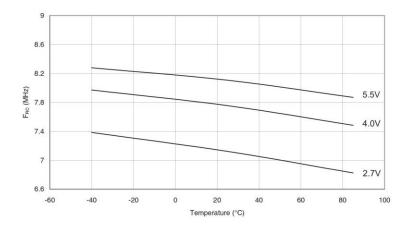


Figure 195. Calibrated 8 MHz RC Oscillator Frequency vs. $V_{\rm CC}$

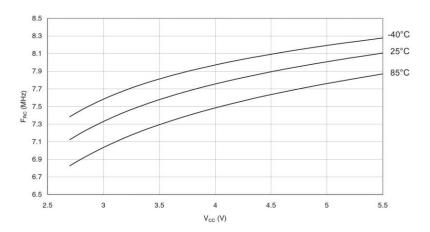




Figure 196. Calibrated 8 MHz RC Oscillator Frequency vs. Osccal Value

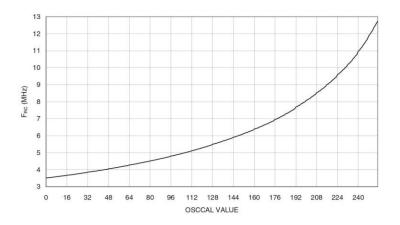


Figure 197. Calibrated 4 MHz RC Oscillator Frequency vs. Temperature

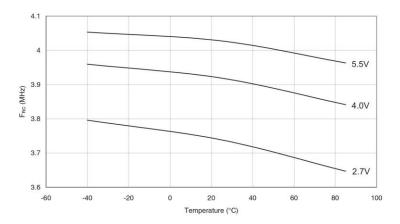




Figure 198. Calibrated 4 MHz RC Oscillator Frequency vs. $V_{\rm CC}$

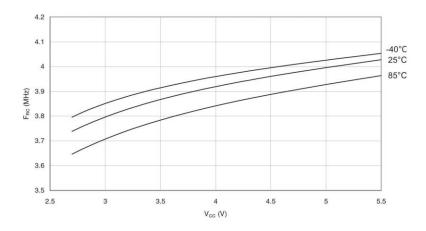


Figure 199. Calibrated 4 MHz RC Oscillator Frequency vs. Osccal Value

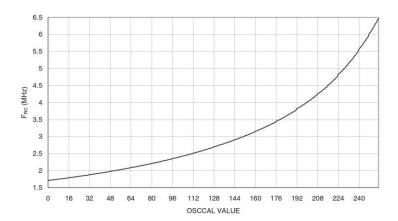






Figure 200. Calibrated 2 MHz RC Oscillator Frequency vs. Temperature

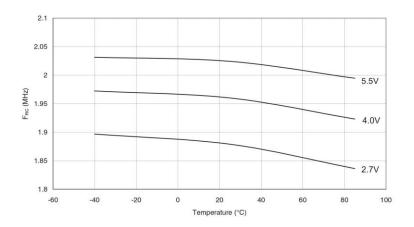


Figure 201. Calibrated 2 MHz RC Oscillator Frequency vs. $V_{\rm CC}$

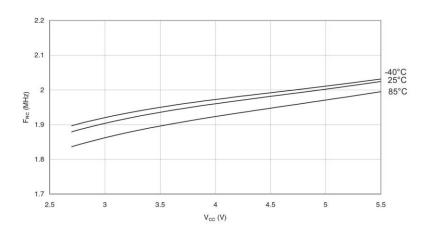




Figure 202. Calibrated 2 MHz RC Oscillator Frequency vs. Osccal Value

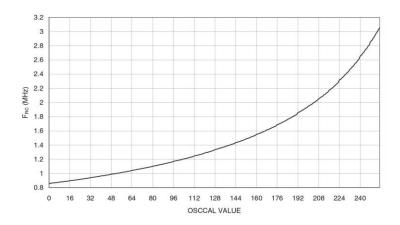


Figure 203. Calibrated 1 MHz RC Oscillator Frequency vs. Temperature

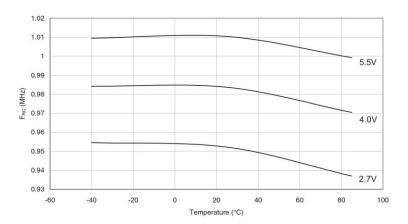




Figure 204. Calibrated 1 MHz RC Oscillator Frequency vs. $V_{\rm CC}$

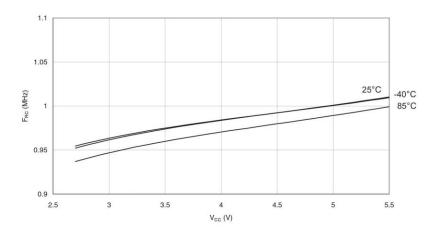
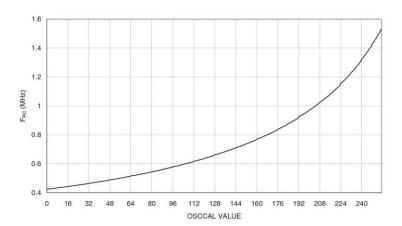


Figure 205. Calibrated 1 MHz RC Oscillator Frequency vs. Osccal Value





Current Consumption Of Peripheral Units

Figure 206. Brownout Detector Current vs. V_{CC}

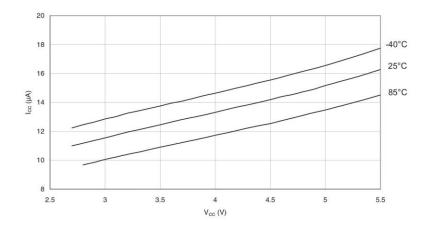


Figure 207. ADC Current vs. V_{CC}(Aref = AVCC)

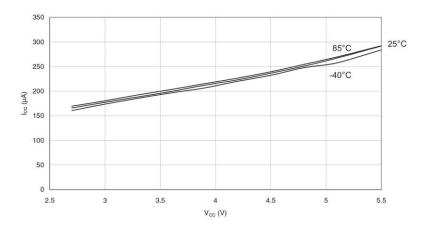




Figure 208. Aref External Reference Current vs. V_{CC}

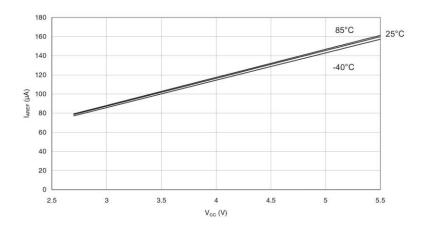


Figure 209. 32khz Tosc Current vs. $V_{\rm CC}$ (Watchdog Timer Disabled)

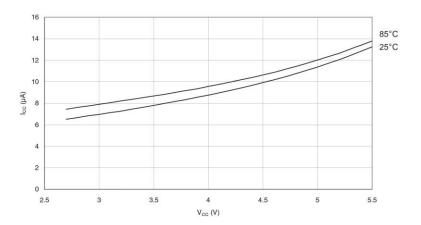




Figure 210. Watchdog Timer Current vs. V_{CC}

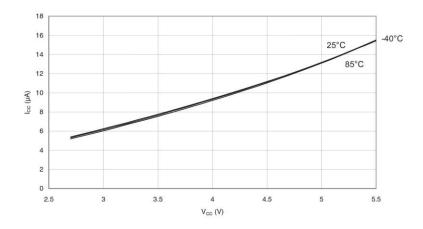
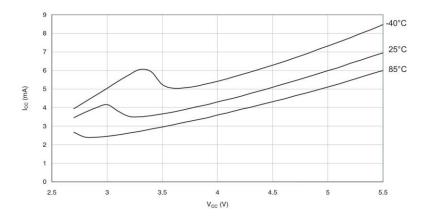


Figure 211. Programming Current vs. V_{CC}





Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
\$3F (\$5F)	SREG	1	Т	Н	s	V	N	Z	С	9
\$3E (\$5E)	SPH	<u></u>	<u>-</u>	N <u>+</u>	20		SP10	SP9	SP8	12
\$3D (\$5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	12
\$3C (\$5C)	OCR0	Timer/Counter	r0 Output Compa	re Register					•	85
\$3B (\$5B)	GICR	INT1	INTO	INT2	-x	_	-	IVSEL	IVCE	48, 69
\$3A (\$5A)	GIFR	INTF1	INTF0	INTF2	_	-	_	_	-	70
\$39 (\$59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	85, 115, 133
\$38 (\$58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	86, 115, 133
\$37 (\$57)	SPMCR	SPMIE	RWWSB	-	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	250
\$36 (\$56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	180
\$35 (\$55)	MCUCR	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	32, 68
\$34 (\$54)	MCUCSR	JTD	ISC2	-	JTRF	WDRF	BORF	EXTRF	PORF	41, 69, 231
\$33 (\$53)	TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	83
\$32 (\$52)	TCNT0	Timer/Counter								85
\$31(1) (\$51)(1)	OSCCAL		bration Register							30
22200-1-22002	OCDR	On-Chip Debu								227
\$30 (\$50)	SFIOR	ADTS2	ADTS1	ADTS0	-	ACME	PUD	PSR2	PSR10	57,88,134,201,221
\$2F (\$4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	110
\$2E (\$4E)	TCCR1B	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	113
\$2D (\$4D)	TCNT1H		r1 – Counter Reg							114
\$2C (\$4C)	TCNT1L		r1 – Counter Reg		- b D. t-					114
\$2B (\$4B)	OCR1AH			are Register A H						114
\$2A (\$4A)	OCR1AL	2002 2002 2002		pare Register A Lo						114
\$29 (\$49)	OCR1BH			oare Register B Hi						114
\$28 (\$48)	OCR1BL			pare Register B Lo						114
\$27 (\$47)	ICR1H			Register High B						114
\$26 (\$46)	ICR1L TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	114 128
\$25 (\$45) \$24 (\$44)	TCNT2	Timer/Counter		CONIZI	COIVIZO	VVGIVIZI	U322	C321	C320	130
\$23 (\$43)	OCR2	_	r2 Output Compa	re Register						130
\$23 (\$43)	ASSR	- Inner/Counter		L Legister	2.	AS2	TCN2UB	OCR2UB	TCR2UB	131
\$21 (\$41)	WDTCR	_	_	-	WDTOE	WDE	WDP2	WDP1	WDP0	43
	UBRRH	URSEL	_	-	-	VVDL		R[11:8]	VVDI 0	167
\$20 ⁽²⁾ (\$40) ⁽²⁾	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	166
\$1F (\$3F)	EEARH	-	-	-	-	-	-	_	EEAR8	19
\$1E (\$3E)	EEARL	EEPROM Add	dress Register Lov	w Byte	I E					19
\$1D (\$3D)	EEDR	EEPROM Dat								19
\$1C (\$3C)	EECR	-	_	1 -	_	EERIE	EEMWE	EEWE	EERE	19
\$1B (\$3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	66
\$1A (\$3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	66
\$19 (\$39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	66
\$18 (\$38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	66
\$17 (\$37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	66
\$16 (\$36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	66
\$15 (\$35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	67
\$14 (\$34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	67
\$13 (\$33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	67
\$12 (\$32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	67
\$11 (\$31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	67
\$10 (\$30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	67
\$0F (\$2F)	SPDR	SPI Data Reg				1				142
\$0E (\$2E)	SPSR	SPIF	WCOL	-	-	-	-	-	SPI2X	142
\$0D (\$2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	140
\$0C (\$2C)	UDR	USART I/O D		10.2mm	02000	0. <u></u> 0		2,900.00		163
\$0B (\$2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	164
\$0A (\$2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	165
\$09 (\$29)	UBRRL		Rate Register Lo	T		1	1 45:5	1.0:-:	10:	167
\$08 (\$28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	202
\$07 (\$27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	217
\$06 (\$26)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	219
\$05 (\$25)	ADCH		gister High Byte							220
\$04 (\$24)	ADCL		gister Low Byte	Doubleton.						220
\$03 (\$23)	TWDR TWAR	Two-wire Seri	al Interface Data		TWA3	TWA2	TWA1	TWAO	TWGCE	182 182
\$02 (\$22)	IVVAR	IVVAb	I VVA5	TWA4	I VVA3	TVVA2	I VVAT	I VVAU	TWGCE	182





Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
\$01 (\$21)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	_	TWPS1	TWPS0	181
\$00 (\$20)	TWBR	Two-wire Seria	Two-wire Serial Interface Bit Rate Register						180	

- Notes: 1. When the OCDEN Fuse is unprogrammed, the OSCCAL Register is always accessed on this address. Refer to the debugger specific documentation for details on how to use the OCDR Register.
 - 2. Refer to the USART description for details on how to access UBRRH and UCSRC.
 - 3. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
 - 4. Some of the Status Flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O Register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers \$00 to \$1F only.





Instruction Set Summary

ADC ADIW SUB SUB SUB SUB SUB SBC SBCI SBIW AND ANDI OR ORI EOR COM NEG SBR CBR INC DEC TST CLR SER MUL	Rd, Rr Rd, K Rd, Rr Rd Rd Rd Rd Rd Rd Rd	Add two Registers Add with Carry two Registers Add Immediate to Word Subtract two Registers Subtract Constant from Register Subtract with Carry two Registers Subtract with Carry two Registers Subtract with Carry Constant from Reg. Subtract Immediate from Word Logical AND Registers Logical AND Registers Logical OR Registers Logical OR Registers Logical OR Registers Clogical OR Registers One's Complement Two's Complement Set Bit(s) in Register	$Rd \leftarrow Rd + Rr$ $Rd \leftarrow Rd + Rr + C$ $Rdh_Rdl \leftarrow Rdh_Rdl + K$ $Rd \leftarrow Rd - Rr$ $Rd \leftarrow Rd - K$ $Rd \leftarrow Rd - K - C$ $Rdh_Rdl \leftarrow Rdh_Rdl + K$ $Rd \leftarrow Rd - K - C$ $Rdh_Rdl \leftarrow Rdh_Rdl - K$ $Rd \leftarrow Rd - K - C$ $Rdh_Rdl \leftarrow Rdh_Rdl - K$ $Rd \leftarrow Rd - K - K$ $Rd \leftarrow Rd - K$	Z.C.N.V.H Z.C.N.V.H Z.C.N.V.S Z.C.N.V.H Z.C.N.V.H Z.C.N.V.H Z.C.N.V.H Z.C.N.V.S Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V	1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
ADD ADC ADIW SUB SUBI SBC SBCI SBIW AND ANDI OR ORI EOR COM NEG SBR CIBR INC DEC TST CLR SER MUL	Rd, Rr Rd, Rr Rd, Kr Rd, K Rd, K Rd, Rr Rd, K Rd, Rr Rd, K Rd, K Rd, Rr Rd	Add two Registers Add with Carry two Registers Add Immediate to Word Subtract two Registers Subtract Constant from Register Subtract with Carry two Registers Subtract with Carry two Registers Subtract with Carry Constant from Reg. Subtract Immediate from Word Logical AND Registers Logical AND Registers Logical OR Register and Constant Logical OR Registers Logical OR Register and Constant Exclusive OR Registers One's Complement Two's Complement Set Bit(s) in Register	$Rd \leftarrow Rd + Rr + C$ $Rdh:Rdl \leftarrow Rdh:Rdl + K$ $Rd \leftarrow Rd - Rr$ $Rd \leftarrow Rd - K$ $Rd \leftarrow Rd - K - C$ $Rd \leftarrow Rd - K - C$ $Rdh:Rdl \leftarrow Rdh:Rdl - K$ $Rd \leftarrow Rd \bullet K - C$ $Rd \leftarrow Rd \bullet R$ $Rd \leftarrow Rd \bullet R$	Z.C.N.V.H Z.C.N.V.S Z.C.N.V.H Z.C.N.V.H Z.C.N.V.H Z.C.N.V.H Z.C.N.V.S Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V	1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
ADIW SUB SUBI SUBI SBCI SBIW AND ANDI OR OR ORI EOR COM NEG SBR INC DEC TST CLR SER MUL	Rdi,K Rd, Rr Rd, K Rd, Rr Rd, K Rdi,K Rdi,K Rd, Rr Rd, K Rd, Rr Rd	Add Immediate to Word Subtract two Registers Subtract Constant from Register Subtract with Carry two Registers Subtract with Carry Constant from Reg. Subtract mediate from Word Logical AND Registers Logical AND Registers Logical OR Registers Logical OR Registers Cogical OR Registers Cogical OR Registers Subtract mediate from Word Logical OR Registers Logical OR Registers Cogical OR Registers Subtract Months of Complement Two's Complement Set Bit(s) in Register	$\begin{aligned} Rdh_{1}Rdl &\leftarrow Rdh_{1}Rdl + K \\ Rd &\leftarrow Rd - Rr \\ Rd &\leftarrow Rd - K \\ Rd &\leftarrow Rd - K - C \\ Rd &\leftarrow Rd - K - C \\ Rdh_{1}Rdl &\leftarrow Rdh_{1}Rdl - K \\ Rd &\leftarrow Rd \bullet K \\ Rd &\leftarrow Rd \bullet K \\ Rd &\leftarrow Rd \bullet K \\ Rd &\leftarrow Rd \circ FF - Rd \end{aligned}$	Z.C.N.V.S Z.C.N.V.H Z.C.N.V.H Z.C.N.V.H Z.C.N.V.H Z.C.N.V.S Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V	2 1 1 1 1 2 1 1 1 1
SUB SUBI SUBI SBC SBCI SBIW AND ANDI OR OR COM NEG SBR INC CER INC CLR SER MUL	Rd, Rr Rd, K Rd, Rr Rd, K Rdl, K Rdl, K Rd, Rr Rd, K Rd, Rr Rd, K Rd, Rr Rd, R Rd Rd Rd Rd Rd Rd Rd, R Rd, R Rd, R Rd Rd Rd Rd Rd, R Rd, R Rd, R Rd Rd Rd Rd, R	Subtract two Registers Subtract Constant from Register Subtract with Carry two Registers Subtract with Carry Constant from Reg. Subtract Immediate from Word Logical AND Registers Logical AND Register and Constant Logical OR Register and Constant Exclusive OR Register and Constant Exclusive OR Registers One's Complement Two's Complement Set Bit(s) in Register	Rd ← Rd - Rr Rd ← Rd - K Rd ← Rd - Rr - C Rd ← Rd - K - C Rdn-Rdl ← Rdn-Rdl - K Rd ← Rd • Rr Rd ← Rd • K Rd ← Rd v K Rd ← Rd v K Rd ← Rd ⊕ Rr Rd ← Rd ⊕ Rr Rd ← Rd ⊕ Rr	Z.C.N.V.H Z.C.N.V.H Z.C.N.V.H Z.C.N.V.H Z.C.N.V.S Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V Z.N.V	1 1 1 1 2 1 1 1
SUBI SBC SBCI SBCI SBIW AND ANDI OR ORI EOR COM NEG SBR CBR INC DEC TST CLR SER MUL	Rd, K Rd, Rr Rd, K Rdi, K Rd, Rr Rd, K Rd, Rr Rd, K Rd, Rr Rd, K Rd, Rr Rd, R Rd, R Rd, R Rd	Subtract Constant from Register Subtract with Carry two Registers Subtract with Carry Constant from Reg. Subtract Immediate from Word Logical AND Registers Logical AND Register and Constant Logical OR Registers Logical OR Register and Constant Exclusive OR Registers One's Complement Two's Complement Set Bit(s) in Register	$\begin{split} Rd \leftarrow Rd \cdot K \\ Rd \leftarrow Rd \cdot Rr \cdot C \\ Rd \leftarrow Rd \cdot K \cdot C \\ Rd \leftarrow Rdh \cdot K \cdot C \\ Rdh \cdot Rdl \leftarrow Rdh \cdot Rdl \cdot K \\ Rd \leftarrow Rd \cdot K $	Z,C,N,V,H Z,C,N,V,H Z,C,N,V,S Z,N,V Z,N,V Z,N,V Z,N,V Z,N,V Z,N,V Z,N,V Z,N,V	1 1 1 2 1 1 1
SBC SBCI SBIW AND AND OR ORI EOR COM NEG SBR CBR INC DEC TST CLR SER MUL	Rd, Rr Rd, K RdI,K Rd, Rr Rd, K Rd, Rr Rd, K Rd, Rr Rd, K Rd, Rr Rd, R Rd Rd Rd Rd Rd Rd Rd, R	Subtract with Carry two Registers Subtract with Carry Constant from Reg. Subtract Immediate from Word Logical AND Registers Logical AND Register and Constant Logical OR Registers Logical OR Register and Constant Exclusive OR Register and Constant Exclusive OR Registers One's Complement Two's Complement Set Bit(s) in Register	$\begin{aligned} Rd &\leftarrow Rd \cdot Rr \cdot C \\ Rd &\leftarrow Rd \cdot K \cdot C \\ Rdh \cdot Rdl &\leftarrow Rdh \cdot Rdl \cdot K \\ Rd &\leftarrow Rd \bullet Rr \\ Rd &\leftarrow Rd \bullet K \\ Rd &\leftarrow Rd \vee Rr \\ Rd &\leftarrow Rd \vee K \\ Rd &\leftarrow Rd \vee K \\ Rd &\leftarrow Rd \vee K \\ Rd &\leftarrow Rd \oplus Rr \\ Rd &\leftarrow SFF - Rd \end{aligned}$	Z,C,N,V,H Z,C,N,V,H Z,C,N,V,S Z,N,V Z,N,V Z,N,V Z,N,V Z,N,V Z,N,V	1 1 2 1 1 1 1
SBCI SBIW AND AND OR OR ORI EOR COM NEG SBR CBR INC DEC TST CLR SER MUL	Rd, K RdI,K Rd, Rr Rd, K Rd, Rr Rd, K Rd, Rr Rd, K Rd, Rr Rd Rd,K	Subtract with Carry Constant from Reg. Subtract Immediate from Word Logical AND Registers Logical AND Register and Constant Logical OR Registers Logical OR Register and Constant Exclusive OR Registers One's Complement Two's Complement Set Bit(s) in Register	$\begin{split} Rd \leftarrow Rd - K - C \\ Rdh: Rdl \leftarrow Rdh: Rdl - K \\ Rd \leftarrow Rd \bullet Rr \\ Rd \leftarrow Rd \bullet K \\ Rd \leftarrow Rd VR \\ Rd \leftarrow Rd VR \\ Rd \leftarrow Rd VK \\ Rd \leftarrow Rd VK \\ Rd \leftarrow Rd \Theta Rr \\ Rd \leftarrow SFF - Rd \\ \end{split}$	Z,C,N,V,H Z,C,N,V,S Z,N,V Z,N,V Z,N,V Z,N,V Z,N,V Z,N,V	1 2 1 1 1 1
SBIW AND ANDI OR OR OR COM NEG SBR CER INC DEC TST CLR SER MUL	Rdi,K Rd, Rr Rd, K Rd, Rr Rd, K Rd, Rr Rd Rd Rd Rd Rd, Rr Rd,K Rd,K	Subtract Immediate from Word Logical AND Registers Logical AND Register and Constant Logical OR Registers Logical OR Registers Logical OR Register and Constant Exclusive OR Registers One's Complement Two's Complement Set Bit(s) in Register	$\begin{aligned} Rdh.RdI &\leftarrow Rdh.RdI - K \\ Rd &\leftarrow Rd \bullet Rr \\ Rd &\leftarrow Rd \bullet K \\ Rd &\leftarrow Rd \bullet K \\ Rd &\leftarrow Rd \vee Rr \\ Rd &\leftarrow Rd \vee K \\ Rd &\leftarrow Rd \oplus Rr \\ Rd &\leftarrow SFF - Rd \end{aligned}$	Z,C,N,V,S Z,N,V Z,N,V Z,N,V Z,N,V Z,N,V	2 1 1 1 1
AND ANDI OR OR COM NEG SBR CPR INC DEC TST CLR SER MUL	Rd, Rr Rd, K Rd, Rr Rd, K Rd, Rr Rd Rd Rd Rd,K Rd,K Rd,K	Logical AND Registers Logical AND Register and Constant Logical OR Registers Logical OR Register and Constant Exclusive OR Registers One's Complement Two's Complement Set Bit(s) in Register	$Rd \leftarrow Rd \bullet Rr$ $Rd \leftarrow Rd \bullet K$ $Rd \leftarrow Rd \lor Rr$ $Rd \leftarrow Rd \lor K$ $Rd \leftarrow Rd \oplus Rr$ $Rd \leftarrow SFF - Rd$	Z,N,V Z,N,V Z,N,V Z,N,V Z,N,V	1 1 1
ANDI OR ORI EOR COM NEG SBR CBR INC DEC TST CLR SER MUL	Rd, K Rd, Rr Rd, K Rd, Rr Rd Rd Rd Rd Rd, K Rd, K Rd, K Rd, K	Logical AND Register and Constant Logical OR Registers Logical OR Register and Constant Exclusive OR Registers One's Complement Two's Complement Set Bit(s) in Register	$Rd \leftarrow Rd \bullet K$ $Rd \leftarrow Rd \lor Rr$ $Rd \leftarrow Rd \lor K$ $Rd \leftarrow Rd \oplus Rr$ $Rd \leftarrow SFF - Rd$	Z,N,V Z,N,V Z,N,V Z,N,V	1 1 1
OR ORI EOR COM NEG SBR CBR INC DEC TST CLR SER MUL	Rd, Rr Rd, K Rd, Rr Rd Rd Rd,K Rd,K Rd,K	Logical OR Registers Logical OR Register and Constant Exclusive OR Registers One's Complement Two's Complement Set Bit(s) in Register	$\begin{tabular}{ll} Rd \leftarrow Rd \lor Rr \\ Rd \leftarrow Rd \lor K \\ Rd \leftarrow Rd \oplus Rr \\ Rd \leftarrow SFF - Rd \\ \end{tabular}$	Z,N,V Z,N,V Z,N,V	1
ORI EOR COM NEG SBR CBR INC DEC TST CLR SER MUL	Rd, K Rd, Rr Rd Rd Rd,K Rd,K Rd,K	Logical OR Register and Constant Exclusive OR Registers One's Complement Two's Complement Set Bit(s) in Register	$Rd \leftarrow Rd \lor K$ $Rd \leftarrow Rd \oplus Rr$ $Rd \leftarrow \$FF - Rd$	Z,N,V Z,N,V	1
EOR COM NEG SBR CBR INC DEC TST CLR SER MUL	Rd, Rr Rd Rd Rd,K Rd,K Rd,K	Exclusive OR Registers One's Complement Two's Complement Set Bit(s) in Register	$Rd \leftarrow Rd \oplus Rr$ $Rd \leftarrow \$FF - Rd$	Z,N,V	-
COM NEG SBR CBR INC DEC TST CLR SER MUL	Rd Rd, Rd,K Rd,K Rd	One's Complement Two's Complement Set Bit(s) in Register	Rd ← \$FF – Rd		1
NEG SBR CBR INC DEC TST CLR SER MUL	Rd Rd,K Rd,K Rd	Two's Complement Set Bit(s) in Register		Z,C,N,V	
SBR CBR INC DEC TST CLR SER MUL	Rd,K Rd,K Rd	Set Bit(s) in Register	Rd ← \$00 – Rd	1	1
CBR INC DEC TST CLR SER MUL	Rd,K Rd			Z,C,N,V,H	1
INC DEC TST CLR SER MUL	Rd	Clear Rit(s) in Register	$Rd \leftarrow Rd \vee K$	Z,N,V	1
DEC TST CLR SER MUL			$Rd \leftarrow Rd \cdot (\$FF - K)$	Z,N,V	1
TST CLR SER MUL	Rd I	Increment	Rd ← Rd + 1	Z,N,V	1
CLR SER MUL	100000	Decrement	Rd ← Rd – 1	Z,N,V	1
SER MUL	Rd	Test for Zero or Minus	$Rd \leftarrow Rd \bullet Rd$	Z,N,V	1
MUL	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
	Rd	Set Register	Rd ← \$FF	None	1
	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
578750000707	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2
	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
BRANCH INSTRUCTION		Deletive luma	PC ← PC + k + 1	Ness	
	k	Relative Jump		None	2
IJMP	ν.	Indirect Jump to (Z)	PC ← Z	None	2
	k	Direct Jump	PC ← k PC ← PC + k + 1	None	3
RCALL ICALL	k	Relative Subroutine Call		None	3
	6	Indirect Call to (Z)	PC ← Z	None	4
	k	Direct Subroutine Call	PC ← k	None	4
RETI RETI		Subroutine Return Interrupt Return	PC ← STACK PC ← STACK	None	4
0.0000000	Rd,Rr	2.34	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
3.50,750,750	Rd,Rr	Compare, Skip if Equal Compare	Rd – Rr	Z, N,V,C,H	1
	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1
	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,H	1
	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
	P, b	Skip if Bit in I/O Register Set	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k+1	None	1/2
	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Gleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Set Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
	k	Branch if 1 Flag Cleared Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1/2





Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2
DATA TRANSFER	INSTRUCTIONS				
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	$Rd \leftarrow (X)$	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	$Rd \leftarrow (X), X \leftarrow X + 1$	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	$X \leftarrow X - 1$, $Rd \leftarrow (X)$	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	$Rd \leftarrow (Y), Y \leftarrow Y + 1$	None	2
LDD	Rd, - Y	Load Indirect with Displacement	$Y \leftarrow Y - 1, Rd \leftarrow (Y)$	None	2
LDD	Rd,Y+q Rd, Z	Load Indirect with Displacement Load Indirect	$Rd \leftarrow (Y + q)$ $Rd \leftarrow (Z)$	None None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	$Rd \leftarrow (Z)$ $Rd \leftarrow (Z), Z \leftarrow Z+1$	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	$Z \leftarrow Z - 1$, $Rd \leftarrow (Z)$	None	2
LDD	Rd, Z+q	Load Indirect and Pie-Dec.	$Rd \leftarrow (Z + q)$	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	$Y \leftarrow Y - 1$, $(Y) \leftarrow Rr$	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	$Z \leftarrow Z - 1$, $(Z) \leftarrow Rr$	None	2
STD	Z+q,Rr	Store Indirect with Displacement	$(Z + q) \leftarrow Rr$	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	$Rd \leftarrow (Z), Z \leftarrow Z+1$	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP BIT AND BIT-TEST	Rd	Pop Register from Stack	Rd ← STACK	None	2
SBI	100000	Set Bit in I/O Benieter	1000 1	None	2
CBI	P,b P,b	Set Bit in I/O Register Clear Bit in I/O Register	$I/O(P,b) \leftarrow 1$ $I/O(P,b) \leftarrow 0$	None	2
LSL	Rd	Logical Shift Left	$Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0$	Z,C,N,V	1
LSR	Rd	Logical Shift Right	$Rd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0$	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	$Rd(0)\leftarrow C,Rd(n+1)\leftarrow Rd(n),C\leftarrow Rd(7)$	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	$Rd(7) \leftarrow C, Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0)$	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=06	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(30)←Rd(74),Rd(74)←Rd(30)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	$SREG(s) \leftarrow 0$	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	$T \leftarrow Rr(b)$	Т	1
BLD	Rd, b	Bit load from T to Register	$Rd(b) \leftarrow T$	None	1
SEC		Set Carry	C←1	С	1
CLC		Clear Carry	C ← 0	С	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	1←1	1	1
CLI	1	Global Interrupt Disable	1←0	- 1	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV	+	Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET	+	Set T in SREG	T ← 1	T	1
CLT	+	Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1





Mnemonics	Operands	Description	Operation	Flags	#Clocks
CLH		Clear Half Carry Flag in SREG	H ← 0	Н	1
MCU CONTROL	INSTRUCTIONS				
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-Chip Debug Only	None	N/A





Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
8	2.7V - 5.5V	ATmega16L-8AU ⁽¹⁾ ATmega16L-8PU ⁽¹⁾ ATmega16L-8MU ⁽¹⁾	44A 40P6 44M1	Industrial (-40°C to 85°C)
16	4.5V - 5.5V	ATmega16-16AU ⁽¹⁾ ATmega16-16PU ⁽¹⁾ ATmega16-16MU ⁽¹⁾	44A 40P6 44M1	Industrial (-40°C to 85°C)

Note: 1. Pb-free packaging complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.

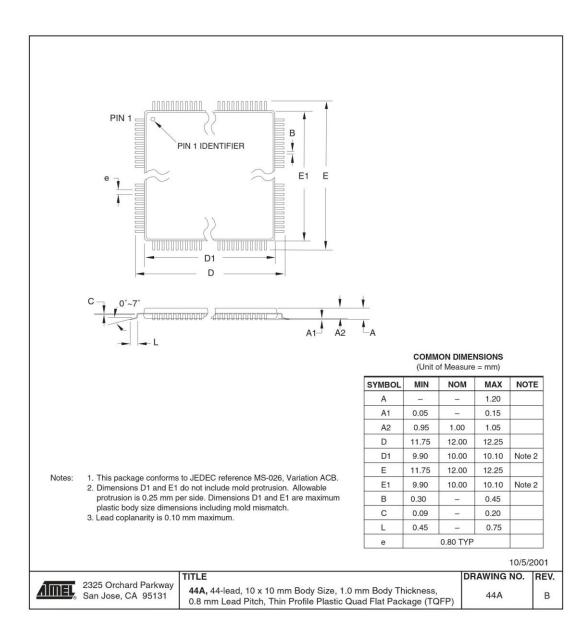
	Package Type
44A	44-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flat Package (TQFP)
40P6	40-pin, 0.600" Wide, Plastic Dual Inline Package (PDIP)
44M1	44-pad, 7 × 7 × 1.0 mm body, lead pitch 0.50 mm, Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)





Packaging Information

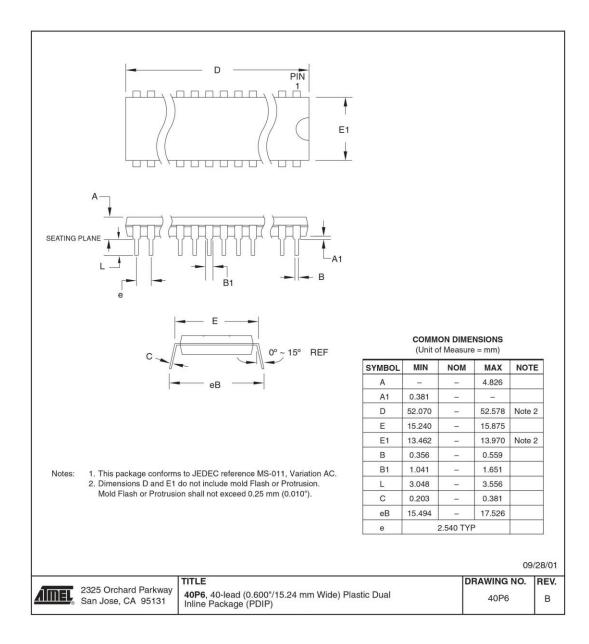
44A







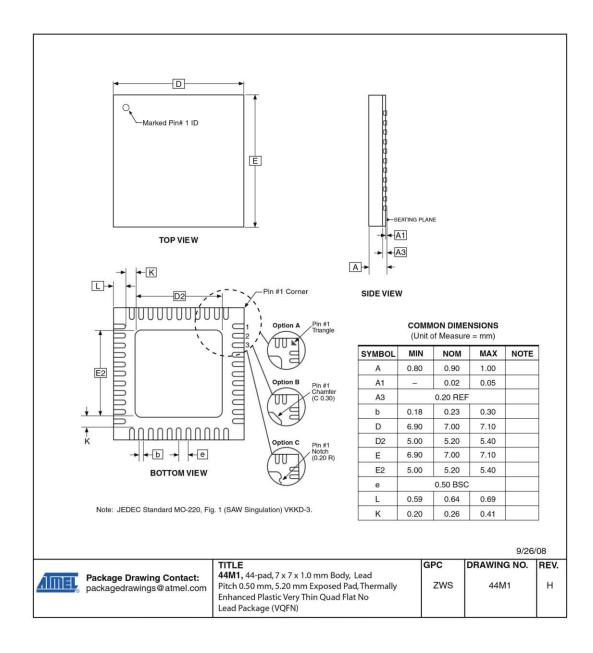
40P6







44M1







Errata

The revision letter in this section refers to the revision of the ATmega16 device.

ATmega16(L) Rev. M

- · First Analog Comparator conversion may be delayed
- · Interrupts may be lost when writing the timer registers in the asynchronous timer
- · IDCODE masks data from TDI input
- · Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request

1. First Analog Comparator conversion may be delayed

If the device is powered by a slow rising V_{CC} , the first Analog Comparator conversion will take longer than expected on some devices.

Problem Fix/Workaround

When the device has been powered or reset, disable then enable the Analog Comparator before the first conversion.

2. Interrupts may be lost when writing the timer registers in the asynchronous timer

The interrupt will be lost if a timer register that is synchronized to the asynchronous timer clock is written when the asynchronous Timer/Counter register(TCNTx) is 0x00.

Problem Fix / Workaround

Always check that the asynchronous Timer/Counter register neither have the value 0xFF nor 0x00 before writing to the asynchronous Timer Control Register(TCCRx), asynchronous Timer Counter Register(TCNTx), or asynchronous Output Compare Register(OCRx).

3. IDCODE masks data from TDI input

The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

Problem Fix / Workaround

- If ATmega16 is the only device in the scan chain, the problem is not visible.
- Select the Device ID Register of the ATmega16 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega16 while reading the Device ID Registers of preceding devices of the boundary scan chain.
- If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega16 must be the fist device in the chain.

Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request.

Reading EEPROM by using the ST or STS command to set the EERE bit in the EECR register triggers an unexpected EEPROM interrupt request.

Problem Fix / Workaround

Always use OUT or SBI to set EERE in EECR.

ATmega16(L) Rev.

- · First Analog Comparator conversion may be delayed
- · Interrupts may be lost when writing the timer registers in the asynchronous timer
- · IDCODE masks data from TDI input
- · Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request

1. First Analog Comparator conversion may be delayed

If the device is powered by a slow rising V_{CC} , the first Analog Comparator conversion will take longer than expected on some devices.



340

2466T-AVR-07/10



Problem Fix/Workaround

When the device has been powered or reset, disable then enable the Analog Comparator before the first conversion.

2. Interrupts may be lost when writing the timer registers in the asynchronous timer

The interrupt will be lost if a timer register that is synchronized to the asynchronous timer clock is written when the asynchronous Timer/Counter register(TCNTx) is 0x00.

Problem Fix / Workaround

Always check that the asynchronous Timer/Counter register neither have the value 0xFF nor 0x00 before writing to the asynchronous Timer Control Register(TCCRx), asynchronous Timer Counter Register(TCNTx), or asynchronous Output Compare Register(OCRx).

3. IDCODE masks data from TDI input

The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

Problem Fix / Workaround

- If ATmega16 is the only device in the scan chain, the problem is not visible.
- Select the Device ID Register of the ATmega16 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega16 while reading the Device ID Registers of preceding devices of the boundary scan chain.
- If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega16 must be the fist device in the chain.

Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request.

Reading EEPROM by using the ST or STS command to set the EERE bit in the EECR register triggers an unexpected EEPROM interrupt request.

Problem Fix / Workaround

Always use OUT or SBI to set EERE in EECR.

ATmega16(L) Rev.

- · First Analog Comparator conversion may be delayed
- · Interrupts may be lost when writing the timer registers in the asynchronous timer
- · IDCODE masks data from TDI input
- · Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request

1. First Analog Comparator conversion may be delayed

If the device is powered by a slow rising $V_{\rm CC}$, the first Analog Comparator conversion will take longer than expected on some devices.

Problem Fix/Workaround

When the device has been powered or reset, disable then enable the Analog Comparator before the first conversion.

2. Interrupts may be lost when writing the timer registers in the asynchronous timer

The interrupt will be lost if a timer register that is synchronized to the asynchronous timer clock is written when the asynchronous Timer/Counter register(TCNTx) is 0x00.



341



Problem Fix / Workaround

Always check that the asynchronous Timer/Counter register neither have the value 0xFF nor 0x00 before writing to the asynchronous Timer Control Register(TCCRx), asynchronous Timer Counter Register(TCNTx), or asynchronous Output Compare Register(OCRx).

3. IDCODE masks data from TDI input

The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

Problem Fix / Workaround

- If ATmega16 is the only device in the scan chain, the problem is not visible.
- Select the Device ID Register of the ATmega16 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega16 while reading the Device ID Registers of preceding devices of the boundary scan chain.
- If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega16 must be the fist device in the chain.

Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request.

Reading EEPROM by using the ST or STS command to set the EERE bit in the EECR register triggers an unexpected EEPROM interrupt request.

Problem Fix / Workaround

Always use OUT or SBI to set EERE in EECR.

ATmega16(L) Rev.

- · First Analog Comparator conversion may be delayed
- · Interrupts may be lost when writing the timer registers in the asynchronous timer
- IDCODE masks data from TDI input
- · Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request

1. First Analog Comparator conversion may be delayed

If the device is powered by a slow rising V_{CC} , the first Analog Comparator conversion will take longer than expected on some devices.

Problem Fix/Workaround

When the device has been powered or reset, disable then enable the Analog Comparator before the first conversion.

2. Interrupts may be lost when writing the timer registers in the asynchronous timer

The interrupt will be lost if a timer register that is synchronized to the asynchronous timer clock is written when the asynchronous Timer/Counter register(TCNTx) is 0x00.

Problem Fix / Workaround

Always check that the asynchronous Timer/Counter register neither have the value 0xFF nor 0x00 before writing to the asynchronous Timer Control Register(TCCRx), asynchronous Timer Counter Register(TCNTx), or asynchronous Output Compare Register(OCRx).

3. IDCODE masks data from TDI input

The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

Problem Fix / Workaround



342

2466T-AVR-07/10



- If ATmega16 is the only device in the scan chain, the problem is not visible.
- Select the Device ID Register of the ATmega16 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega16 while reading the Device ID Registers of preceding devices of the boundary scan chain.
- If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega16 must be the fist device in the chain.

Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request.

Reading EEPROM by using the ST or STS command to set the EERE bit in the EECR register triggers an unexpected EEPROM interrupt request.

Problem Fix / Workaround

Always use OUT or SBI to set EERE in EECR.

ATmega16(L) Rev.

- · First Analog Comparator conversion may be delayed
- · Interrupts may be lost when writing the timer registers in the asynchronous timer
- IDCODE masks data from TDI input
- · Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request

1. First Analog Comparator conversion may be delayed

If the device is powered by a slow rising V_{CC} , the first Analog Comparator conversion will take longer than expected on some devices.

Problem Fix/Workaround

When the device has been powered or reset, disable then enable the Analog Comparator before the first conversion.

2. Interrupts may be lost when writing the timer registers in the asynchronous timer

The interrupt will be lost if a timer register that is synchronized to the asynchronous timer clock is written when the asynchronous Timer/Counter register(TCNTx) is 0x00.

Problem Fix / Workaround

Always check that the asynchronous Timer/Counter register neither have the value 0xFF nor 0x00 before writing to the asynchronous Timer Control Register(TCCRx), asynchronous Timer Counter Register(TCNTx), or asynchronous Output Compare Register(OCRx).

3. IDCODE masks data from TDI input

The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

Problem Fix / Workaround

- If ATmega16 is the only device in the scan chain, the problem is not visible.
- Select the Device ID Register of the ATmega16 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega16 while reading the Device ID Registers of preceding devices of the boundary scan chain.
- If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega16 must be the fist device in the chain.



343



Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request.

Reading EEPROM by using the ST or STS command to set the EERE bit in the EECR register triggers an unexpected EEPROM interrupt request.

Problem Fix / Workaround

Always use OUT or SBI to set EERE in EECR.

ATmega16(L) Rev. H

- · First Analog Comparator conversion may be delayed
- · Interrupts may be lost when writing the timer registers in the asynchronous timer
- · IDCODE masks data from TDI input
- · Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request

1. First Analog Comparator conversion may be delayed

If the device is powered by a slow rising V_{CC} , the first Analog Comparator conversion will take longer than expected on some devices.

Problem Fix/Workaround

When the device has been powered or reset, disable then enable theAnalog Comparator before the first conversion.

2. Interrupts may be lost when writing the timer registers in the asynchronous timer

The interrupt will be lost if a timer register that is synchronized to the asynchronous timer clock is written when the asynchronous Timer/Counter register(TCNTx) is 0x00.

Problem Fix / Workaround

Always check that the asynchronous Timer/Counter register neither have the value 0xFF nor 0x00 before writing to the asynchronous Timer Control Register(TCCRx), asynchronous Timer Counter Register(TCNTx), or asynchronous Output Compare Register(OCRx).

3. IDCODE masks data from TDI input

The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

Problem Fix / Workaround

- If ATmega16 is the only device in the scan chain, the problem is not visible.
- Select the Device ID Register of the ATmega16 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega16 while reading the Device ID Registers of preceding devices of the boundary scan chain.
- If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega16 must be the fist device in the chain.

Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request.

Reading EEPROM by using the ST or STS command to set the EERE bit in the EECR register triggers an unexpected EEPROM interrupt request.

Problem Fix / Workaround

Always use OUT or SBI to set EERE in EECR.





Datasheet Revision History

Please note that the referring page numbers in this section are referred to this document. The referring revision in this section are referring to the document revision.

Rev. 2466T-07/10

- Corrected use of comma in formula Rp in Table 120, "Two-wire Serial Bus Requirements," on page 294.
- 2. Updated document according to Atmel's Technical Terminology
- Note 6 and Note 7 under Table 120, "Two-wire Serial Bus Requirements," on page 294 have been removed.

Rev. 2466S-05/09

- 1. Updated "Errata" on page 340.
- 2. Updated the last page with Atmel's new adresses.

Rev. 2466R-06/08

1. Added "Not recommended for new designs" note in Figure on page 1.

Rev. 2466Q-05/08

- Updated "Fast PWM Mode" on page 77 in "8-bit Timer/Counter0 with PWM" on page 71:
 - Removed the last section describing how to achieve a frequency with 50% duty cycle waveform output in fast PWM mode.
- 2. Removed note from Feature list in "Analog to Digital Converter" on page 204.
- 3. Removed note from Table 84 on page 218.
- 4. Updated "Ordering Information" on page 336:
 - Commercial ordering codes removed.
 - Non Pb-free package option removed.

Rev. 2466P-08/07

- 1. Updated "Features" on page 1.
- 2. Added "Data Retention" on page 6.
- 3. Updated "Errata" on page 340.
- 4. Updated "Slave Mode" on page 140.

Rev. 2466O-03/07

- Updated "Calibrated Internal RC Oscillator" on page 29.
- 2. Updated C code example in "USART Initialization" on page 149.
- 3. Updated "ATmega16 Boundary-scan Order" on page 241.
- 4. Removed "premilinary" from "ADC Characteristics" on page 297.
- 5. Updated from V to mV in "I/O Pin Input Hysteresis vs. VCC" on page 317.
- 6. Updated from V to mV in "Reset Input Pin Hysteresis vs. VCC" on page 318.



345



Rev. 2466N-10/06

- 1. Updated "Timer/Counter Oscillator" on page 31.
- 2. Updated "Fast PWM Mode" on page 102.
- 3. Updated Table 38 on page 83, Table 40 on page 84, Table 45 on page 111, Table 47 on page 112, Table 50 on page 128 and Table 52 on page 129.
- 4. Updated C code example in "USART Initialization" on page 149.
- 5. Updated "Errata" on page 340.

Rev. 2466M-04/06

- 1. Updated typos.
- 2. Updated "Serial Peripheral Interface SPI" on page 135.
- Updated Table 86 on page 221, Table 116 on page 276, Table 121 on page 295 and Table 122 on page 297.

Rev. 2466L-06/05

- 1. Updated note in "Bit Rate Generator Unit" on page 178.
- 2. Updated values for V_{INT} in "ADC Characteristics" on page 297.
- 3. Updated "Serial Programming Instruction set" on page 276.
- 4. Updated USART init C-code example in "USART" on page 144.

Rev. 2466K-04/05

- 1. Updated "Ordering Information" on page 336.
- MLF-package alternative changed to "Quad Flat No-Lead/Micro Lead Frame Package QFN/MLF".
- 3. Updated "Electrical Characteristics" on page 291.

Rev. 2466J-10/04

1. Updated "Ordering Information" on page 336.

Rev. 2466I-10/04

- 1. Removed references to analog ground.
- 2. Updated Table 7 on page 28, Table 15 on page 38, Table 16 on page 42, Table 81 on page 209, Table 116 on page 276, and Table 119 on page 293.
- 3. Updated "Pinout ATmega16" on page 2.
- 4. Updated features in "Analog to Digital Converter" on page 204.
- 5. Updated "Version" on page 229.
- 6. Updated "Calibration Byte" on page 261.
- 7. Added "Page Size" on page 262.

Rev. 2466H-12/03

1. Updated "Calibrated Internal RC Oscillator" on page 29.



346



Rev. 2466G-10/03

- 1. Removed "Preliminary" from the datasheet.
- 2. Changed ICP to ICP1 in the datasheet.
- 3. Updated "JTAG Interface and On-chip Debug System" on page 36.
- Updated assembly and C code examples in "Watchdog Timer Control Register WDTCR" on page 43.
- 5. Updated Figure 46 on page 103.
- 6. Updated Table 15 on page 38, Table 82 on page 217 and Table 115 on page 276.
- 7. Updated "Test Access Port TAP" on page 222 regarding JTAGEN.
- 8. Updated description for the JTD bit on page 231.
- 9. Added note 2 to Figure 126 on page 252.
- 10. Added a note regarding JTAGEN fuse to Table 105 on page 260.
- Updated Absolute Maximum Ratings* and DC Characteristics in "Electrical Characteristics" on page 291.
- 12. Updated "ATmega16 Typical Characteristics" on page 299.
- 13. Fixed typo for 16 MHz QFN/MLF package in "Ordering Information" on page 336.
- 14. Added a proposal for solving problems regarding the JTAG instruction IDCODE in "Errata" on page 340.

Rev. 2466F-02/03

- 1. Added note about masking out unused bits when reading the Program Counter in "Stack Pointer" on page 12.
- Added Chip Erase as a first step in "Programming the Flash" on page 288 and "Programming the EEPROM" on page 289.
- 3. Added the section "Unconnected pins" on page 55.
- Added tips on how to disable the OCD system in "On-chip Debug System" on page 34.
- Removed reference to the "Multi-purpose Oscillator" application note and "32 kHz Crystal Oscillator" application note, which do not exist.
- 6. Added information about PWM symmetry for Timer0 and Timer2.
- 7. Added note in "Filling the Temporary Buffer (Page Loading)" on page 253 about writing to the EEPROM during an SPM Page Load.
- 8. Removed ADHSM completely.





- Added Table 73, "TWI Bit Rate Prescaler," on page 182 to describe the TWPS bits in the "TWI Status Register – TWSR" on page 181.
- 10. Added section "Default Clock Source" on page 25.
- Added note about frequency variation when using an external clock. Note added in "External Clock" on page 31. An extra row and a note added in Table 118 on page 293.
- 12. Various minor TWI corrections.
- 13. Added "Power Consumption" data in "Features" on page 1.
- 14. Added section "EEPROM Write During Power-down Sleep Mode" on page 22.
- Added note about Differential Mode with Auto Triggering in "Prescaling and Conversion Timing" on page 207.
- 16. Added updated "Packaging Information" on page 337.
- Rev. 2466E-10/02
- 1. Updated "DC Characteristics" on page 291.
- Rev. 2466D-09/02
- 1. Changed all Flash write/erase cycles from 1,000 to 10,000.
- Updated the following tables: Table 4 on page 26, Table 15 on page 38, Table 42 on page 85, Table 45 on page 111, Table 46 on page 111, Table 59 on page 143, Table 67 on page 167, Table 90 on page 235, Table 102 on page 258, "DC Characteristics" on page 291, Table 119 on page 293, Table 121 on page 295, and Table 122 on page 297.
- 3. Updated "Errata" on page 340.
- Rev. 2466C-03/02
- 1. Updated typical EEPROM programming time, Table 1 on page 20.
- 2. Updated typical start-up time in the following tables:

Table 3 on page 25, Table 5 on page 27, Table 6 on page 28, Table 8 on page 29, Table 9 on page 29, and Table 10 on page 29.

- 3. Updated Table 17 on page 43 with typical WDT Time-out.
- 4. Added Some Preliminary Test Limits and Characterization Data.

Removed some of the TBD's in the following tables and pages:

Table 15 on page 38, Table 16 on page 42, Table 116 on page 272 (table removed in document review #D), "Electrical Characteristics" on page 291, Table 119 on page 293, Table 121 on page 295, and Table 122 on page 297.

5. Updated TWI Chapter.

Added the note at the end of the "Bit Rate Generator Unit" on page 178.

- Corrected description of ADSC bit in "ADC Control and Status Register A ADCSRA" on page 219.
- Improved description on how to do a polarity check of the ADC doff results in "ADC Conversion Result" on page 216.



348



- 8. Added JTAG version number for rev. H in Table 87 on page 229.
- 9. Added not regarding OCDEN Fuse below Table 105 on page 260.
- 10. Updated Programming Figures:

Figure 127 on page 262 and Figure 136 on page 274 are updated to also reflect that AVCC must be connected during Programming mode. Figure 131 on page 270 added to illustrate how to program the fuses.

- 11. Added a note regarding usage of the "PROG_PAGELOAD (\$6)" on page 280 and "PROG_PAGEREAD (\$7)" on page 280.
- **12.** Removed alternative algortihm for leaving JTAG Programming mode. See "Leaving Programming Mode" on page 288.
- Added Calibrated RC Oscillator characterization curves in section "ATmega16 Typical Characteristics" on page 299.
- Corrected ordering code for QFN/MLF package (16 MHz) in "Ordering Information" on page 336.
- 15. Corrected Table 90, "Scan Signals for the Oscillators(1)(2)(3)," on page 235.





Table of Contents

Features 1

Pin Configurations 2

Disclaimer 2

Overview 3

Block Diagram 3 Pin Descriptions 4

Resources 6

Data Retention 6

About Code Examples 7

AVR CPU Core 8

Introduction 8
Architectural Overview 8
ALU – Arithmetic Logic Unit 9
Status Register 9
General Purpose Register File 11
Stack Pointer 12
Instruction Execution Timing 13
Reset and Interrupt Handling 13

AVR ATmega16 Memories 16

In-System Reprogrammable Flash Program Memory 16 SRAM Data Memory 17 EEPROM Data Memory 18 I/O Memory 23

System Clock and Clock Options 24

Clock Systems and their Distribution 24
Clock Sources 25
Default Clock Source 25
Crystal Oscillator 25
Low-frequency Crystal Oscillator 28
External RC Oscillator 28
Calibrated Internal RC Oscillator 29
External Clock 31
Timer/Counter Oscillator 31

Power Management and Sleep Modes 32



2466T-AVR-07/10





Idle Mode 33
ADC Noise Reduction Mode 33
Power-down Mode 33
Power-save Mode 33
Standby Mode 34
Extended Standby Mode 34
Minimizing Power Consumption 35

System Control and Reset 37

Internal Voltage Reference 42 Watchdog Timer 42

Interrupts 45

Interrupt Vectors in ATmega16 45

I/O Ports 50

Introduction 50
Ports as General Digital I/O 50
Alternate Port Functions 55
Register Description for I/O Ports 66

External Interrupts 68

8-bit Timer/Counter0 with PWM 71

Overview 71
Timer/Counter Clock Sources 72
Counter Unit 72
Output Compare Unit 73
Compare Match Output Unit 74
Modes of Operation 76
Timer/Counter Timing Diagrams 81
8-bit Timer/Counter Register Description 83

Timer/Counter0 and Timer/Counter1 Prescalers 87

16-bit Timer/Counter1 89

Overview 89
Accessing 16-bit Registers 92
Timer/Counter Clock Sources 94
Counter Unit 95
Input Capture Unit 96
Output Compare Units 98
Compare Match Output Unit 100
Modes of Operation 101
Timer/Counter Timing Diagrams 108
16-bit Timer/Counter Register Description 110

ATmega16(L) ■

2466T-AVR-07/10



8-bit Timer/Counter2 with PWM and Asynchronous Operation 117

Overview 117

Timer/Counter Clock Sources 118

Counter Unit 118

Output Compare Unit 119

Compare Match Output Unit 121

Modes of Operation 122

Timer/Counter Timing Diagrams 126

8-bit Timer/Counter Register Description 128

Asynchronous Operation of the Timer/Counter 131

Timer/Counter Prescaler 134

Serial Peripheral Interface - SPI 135

SS Pin Functionality 140

Data Modes 143

USART 144

Overview 144

Clock Generation 145

Frame Formats 148

USART Initialization 149

Data Reception - The USART Receiver 154

Asynchronous Data Reception 157

Multi-processor Communication Mode 161

Accessing UBRRH/ UCSRC Registers 162

USART Register Description 163

Examples of Baud Rate Setting 168

Two-wire Serial Interface 172

Features 172

Two-wire Serial Interface Bus Definition 172

Data Transfer and Frame Format 173

Multi-master Bus Systems, Arbitration and Synchronization 176

Overview of the TWI Module 178

TWI Register Description 180

Using the TWI 183

Transmission Modes 186

Multi-master Systems and Arbitration 199

Analog Comparator 201

Analog Comparator Multiplexed Input 203

Analog to Digital Converter 204

Features 204

Operation 205

Starting a Conversion 206



iii





Prescaling and Conversion Timing 207
Changing Channel or Reference Selection 210
ADC Noise Canceler 211
ADC Conversion Result 216

JTAG Interface and On-chip Debug System 222

Features 222

Overview 222

Test Access Port - TAP 222

TAP Controller 224

Using the Boundary-scan Chain 225

Using the On-chip Debug System 225

On-chip Debug Specific JTAG Instructions 226

On-chip Debug Related Register in I/O Memory 227

Using the JTAG Programming Capabilities 227

Bibliography 227

IEEE 1149.1 (JTAG) Boundary-scan 228

Features 228

System Overview 228

Data Registers 228

Boundary-scan Specific JTAG Instructions 230

Boundary-scan Chain 232

ATmega16 Boundary-scan Order 241

Boundary-scan Description Language Files 245

Boot Loader Support - Read-While-Write Self-Programming 246

Features 246

Application and Boot Loader Flash Sections 246

Read-While-Write and no Read-While-Write Flash Sections 246

Boot Loader Lock Bits 248

Entering the Boot Loader Program 249

Addressing the Flash during Self-Programming 251

Self-Programming the Flash 252

Memory Programming 259

Program And Data Memory Lock Bits 259

Fuse Bits 260

Signature Bytes 261

Calibration Byte 261

Page Size 262

Parallel Programming Parameters, Pin Mapping, and Commands 262

Parallel Programming 265

Serial Downloading 273

Programming via the JTAG Interface 278

ATmega16(L)

2466T-AVR-07/10

iv



Electrical Characteristics 291

Absolute Maximum Ratings* 291
DC Characteristics 291
External Clock Drive Waveforms 293
External Clock Drive 293
Two-wire Serial Interface Characteristics 294
SPI Timing Characteristics 295
ADC Characteristics 297

ATmega16 Typical Characteristics 299

Register Summary 331

Instruction Set Summary 333

Ordering Information 336

Packaging Information 337

44A 337 40P6 338 44M1 339

Errata 340

ATmega16(L) Rev. M 340 ATmega16(L) Rev. L 340 ATmega16(L) Rev. K 341 ATmega16(L) Rev. J 342 ATmega16(L) Rev. I 343 ATmega16(L) Rev. H 344

Datasheet Revision History 345

Rev. 2466T-07/10 345 Rev. 2466S-05/09 345 Rev. 2466R-06/08 345 Rev. 2466P-08/07 345 Rev. 2466P-08/07 345 Rev. 2466O-03/07 345 Rev. 2466N-10/06 346 Rev. 2466M-04/06 346 Rev. 2466L-06/05 346 Rev. 2466L-06/05 346 Rev. 2466J-10/04 346 Rev. 2466I-10/04 346 Rev. 2466H-12/03 346 Rev. 2466G-10/03 347 Rev. 2466F-02/03 347







Rev. 2466E-10/02 348 Rev. 2466D-09/02 348 Rev. 2466C-03/02 348

Table of Contents i

vi ATmega16(L) =









Headquarters

Atmel Corporation

2325 Orchard Parkway San Jose, CA 95131 USA

Tel: 1(408) 441-0311 Fax: 1(408) 487-2600

International

Atmel Asia

Unit 1-5 & 16, 19/F BEA Tower, Millennium City 5 418 Kwun Tong Road Kwun Tong, Kowloon Hong Kong

Tel: (852) 2245-6100

Fax: (852) 2722-1369

Atmel Europe

Le Krebs 8, Rue Jean-Pierre Timbaud BP 309

78054 Saint-Quentin-en-Yvelines Cedex

France

Tel: (33) 1-30-60-70-00 Fax: (33) 1-30-60-71-11 Atmel Japan

9F, Tonetsu Shinkawa Bldg. 1-24-8 Shinkawa Chuo-ku, Tokyo 104-0033 Japan

Tel: (81) 3-3523-3551 Fax: (81) 3-3523-7581

Product Contact

Web Site

www.atmel.com

Technical Support avr@atmel.com

Sales Contact

www.atmel.com/contacts

Literature Requests

www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life

@ 2010 Atmel Corporation. All rights reserved. Atmel®, Atmel logo and combinations thereof, AVR® and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.

2466T-AVR-07/10



Using a DC Motor as a Tachometer



Team 358 - Hauppauge, NY

Here is an out of the ordinary use for a small motor, turning normal motor usage on it's head by running the motor as a weak generator and source of electrical energy rather than as a source of mechanical power and user of electricity. Speed is proportional to the electric current generated by the device. No power is necessary or desired when the motor is hooked up as a tachometer (tach) and basically becomes a generator. You cannot recharge your batteries and it would be silly to attempt to do so, but when the voltage produced is scaled by an adjustable circuit to be within the 0-5 volt range of the FIRST robot controller analog input you have yourself a simple speedometer. Commercial tachometers can be purchased for the job, but tend to be expensive in return for high accuracy and efficiency.

It's main advantages are: instantaneous speed measurement, best suited for high revolution per minute (rpm) measurement, easy to integrate mechanically & as a program analog input, easy to adjust, no extra gearing necessary, inexpensive, no processing overhead, and hardy. Disadvantages over other sensors include being bulkier.

A tachometer...

Tells you how fast something is spinning. It returns a value directly proportional to the rotational speed of a spinning mechanism such as a wheel, motor, or shaft. You see them on some cars telling you the rpm of the engine. A tachometer is not used to count wheel turns, but is a measure of the speed alone and is often used on mechanisms that spin in only one direction.

Tachometer feedback is useful for:

- > Setting or limiting the speed of a spinning object
- Monitoring the spin rate for dangerous conditions
- Acting when rpm's reach predetermined values, for example, automatic gear shifting

Tachometers can be made from other sensors, not just DC motors. For example, encoders or light beam sensors counting revolutions, with some processing overhead, can be used to produce a tachometer-like rotary speed measurement. However, a real tach measures instantaneous speed while counting revolutions indicates only average speed over some time interval. Every sensor has unique drawbacks as well as advantages of their own.

Design considerations:

- > Judge the suitability of a tach for your application. If you're making sure a wheel doesn't spin too fast a tach is great, however, if you need to measure how many times the wheel has spun then a tach isn't the answer
- Match maximum tach motor rpm vs. rpm to be measured chose a motor to match the maximum rated rpm of your rotating object. You want one that generates a healthy voltage at the maximum rpm. You will cut back excess voltage with resistors, but it's difficult to beef up voltage that just isn't there.
- ➤ It isn't difficult to match motor characteristics to the minimum/maximum rpm operating range since whatever the voltage resistors will map it into the 0-5v range of the FIRST Robot Controller (RC). Your choice of resistors can emphasize any particular range of speeds that are important to your application.



For tach applications just experiment with spinning motors at the maximum rpm to see what voltage each produces.

- ➤ How to physically mount the tachometer and mechanically engaging it with the rotating wheel, shaft, motor, etc. Will the motor you chose be required to take side loads?
- ➤ Is a one direction or a bi-directional tach necessary? In either case the tach must be wired to receive positive voltage at the RC. A sample circuit is shown later for a tach that only spins in one direction, but a modified circuit that includes directional diodes is required for converting both positive & negative tach voltage into the RC's positive 0-5v range.
- Tradeoffs such as tachometer noise vs. accuracy / quick response vs software filtering. A DC motor tach doesn't return a pure flat-line value, but one that has tiny fluctuations due to the motor brushes constantly making and breaking contact with the commutator. This might be a consideration if your application requires high accuracy, but typically you'll only notice it on a detailed plot of the tach data...
- You can borrow a commercial tach to calibrate yours if you want to know exactly how each voltage 0-5v corresponds to a specific rpm.

Wiring:

Spin the tach at the maximum rpm you expect to see and measure the voltage produced directly from the motor terminals with a multi-meter. Then it's a simple calculation to figure the resistance required to produce a positive voltage in the range 0 to 5 volts when it turns in the desired direction. You can easily test spin candidate tach motors by attaching them to a variable-speed drill so you can sample the voltage each generates.

- The voltage generated by the tach during it's full range of motion must be normalized to the RC's range of 0-5v via resistors or an adjustable pot before connecting it to the RC. Calculate the resistance(R) needed using Ohm's Law, V=IR or R=V/I and your own measurement of the current produced.
- If the tach is dedicated to one direction then use a diode to limit current direction and pass positive voltage only for the RC.
- An adjustable potentiometer makes the tach easily tunable if the rpm's of your spinning system are still being changed. If desired the potentiometer can later be swapped for fixed resistors when the mechanical design has settled down and you've found the perfect resistant value.

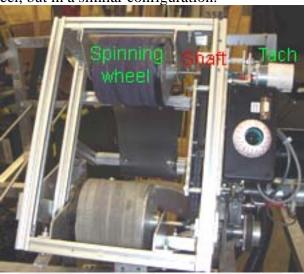
Software:

No special software is required to use a DC motor tach – one of the reasons a tach can be a simple solution. The tach input comes to the RC via an analog input that your software polls. At it's simplest, poll when you want to know the current speed. The analog input directly corresponds to speed. The resolution of the analog input (1024) means that each analog point will equal some number of rpm. The exact rpm can only be determined by validating with a commercial tach, but it's usually not necessary to convert the analog value into human terms such as rpm. The software will check for and act on pre-determined analog values. Because it's really a motor the tach demonstrates regular minor reading fluctuations as the commutator brushes cross breaks. These can be smoothed or otherwise filtered out or ignored in software.



One Implementation

FIRST team 358, the Robotic Eagles, used this motor-based tachometer feedback to control our robot's shooter wheels in 2006's Aim High ensuring a steady, constant wheel speed of 2900rpm before the launch of each poof ball. The tach can be seen in an early mount on the top right of the following photo of our shooting system driven directly off the shaft of the top shooter wheel. The tach was later relocated to the bottom shooter wheel, but in a similar configuration.



DC Motor:

We used the small Matchubi motor that has come loose in the FIRST Kit-of-Parts for several years. It has an operational maximum of 4900 rpm, perfectly matched to our max. theoretical designed shooter wheel rpm. The shooter wheels were designed to be driven at half speed, so operationally the max rpm was roughly half that, but we designed the tach to operate up to the maximum possible designed speed. In practice with mechanical losses the maximum wheel rpm was actually around 4500.

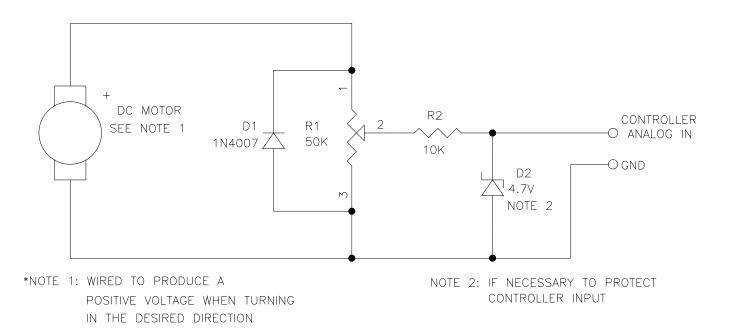




Wiring:

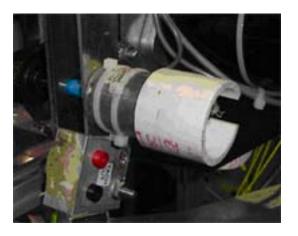
The circuit on the next page is for a one directional tach feeding an analog input on the robot controller. The DC motor connections are wired to produce a positive voltage when it turns in the desired direction. If by accident the tach is run backwards a diode limits current direction and permit positive voltage only to the RC analog input. An adjustable potentiometer limits our maximum voltage to the 5v handled by the RC and makes the resistance easily tunable, but fixed resistors can be used. The end result of this circuit is that in it's final configuration each analog input of 1 represented ~4.76rpm of our shooter wheel.





Mounting:

The mounting was quite simple, one of the primary advantages of using a tachometer. The tach motor was attached to a mounting plate, the pinion passed through a hole in the plate and the pinion was connected directly in line to the wheel drive shaft with a piece of pneumatic tubing. To protect the end of the motor/wiring from risk of physical damage a short piece of PVC, the same diameter as the motor casing, was added over the motor terminals and the circuit at the back of the motor housing. The circuit components were secured by a liberal amount of hot melt glue (making it hard to show the pot and diodes in the photographs). Cooling isn't an issue with the tach motor as there isn't any stall condition and no heat is generated.







Direct tach readout:

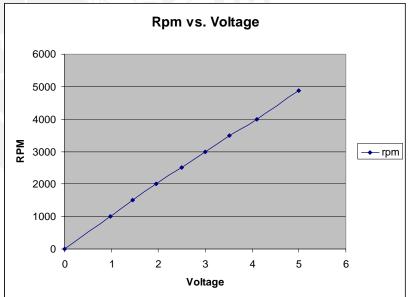
We added an optional permanent electrical connector mount (visible in the right photo above) where we could attach standard multi-meter or oscilloscope probes for a direct readout of the voltage produced by the tachometer while it was operating. These made it simple for the electrical team to fine tune the variable potentiometer, quickly verify correct operation, trouble shoot, and monitor the tach output directly.

Software:

The tach reading comes into the robot controller via an analog input that was simply polled by the software at a convenient rate. A moving average was experimented with, but found to unnecessarily slow response for little gain in accuracy. The tach input was smoothed of minor fluctuations by zeroing the last two bits of the analog input ($Get_Analog_Value(rc_ana_in02) \& \sim 3$), essentially giving us a deadband of 3 analog points which in this case represented an rpm deadband of 14 rpm accurate to half a percent.

The following table are measurements we took of our tachometer validated with a commercial hand-held tachometer. In practical terms, the mapping of rpm to a specific analog input value was unnecessary and was not used. The desired speed of our spinning poof ball throwing wheels was found by repeatedly throwing balls until they consistently hit their target. Then we recorded and hard coded in software the corresponding analog input value to be maintained. The plot of Rpm vs. Voltage shows the relationship is very linear.

Volts	rpm	Analog input value
0	0	0
0.98	1000	200
1.45	1500	297
1.95	2000	399
2.5	2500	512
3	3000	614
3.52	3500	720
4.1	4000	839
5	4878	1023





Silicon Epitaxial Planar Zener Diodes for Stabilized Power Supply

HITACHI

ADE-208-136C (Z)

Rev.3 Sep. 2000

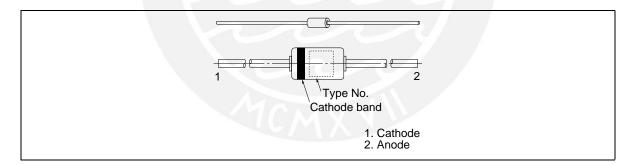
Features

- Glass package DO-41 structure ensures high reliability.
- Wide spectrum from 3.3V through 36V of zener voltage provide flexible application.

Ordering Information

Type No.	Mark	Package Code	
1N4728A through 1N4753A	Type No.	DO-41	

Outline



Absolute Maximum Ratings

 $(Ta = 25^{\circ}C)$

Item	Symbol	Value	Unit	
Power dissipation	Pd *1	1.0	W	
Junction temperature	Tj	200	°C	
Storage temperature	Tstg	-65 to +200	°C	

Note: 1. See Fig.3





Electrical Characteristics

 $(Ta = 25^{\circ}C)$

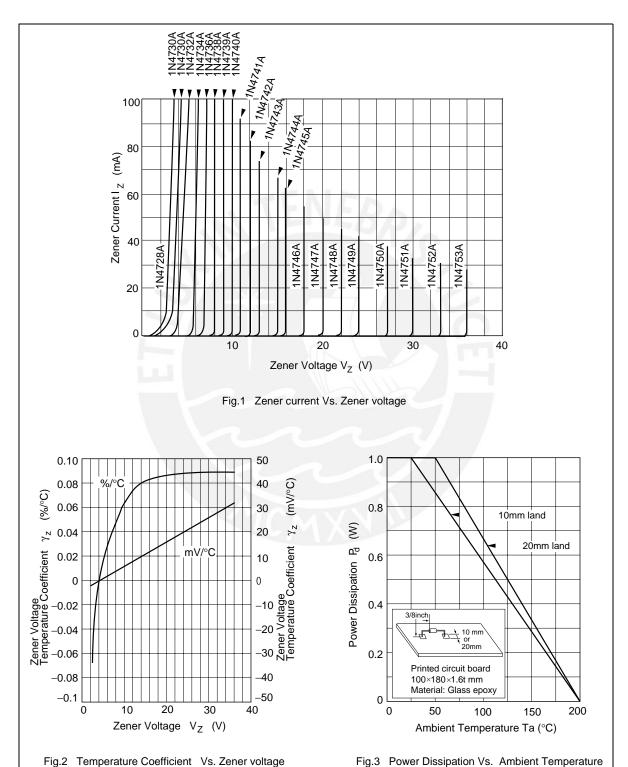
2		I _R (μΑ)		$Z_{zr}(\Omega)$)	$Z_{zK}(\Omega)$		I_{RSM} (mA)* ²
	Test Condition	l	Test Condition		Test Condition		Test Condition	
Max	I _z (mA)	Max	V _R (V)	Max	I _{zt} (mA)	Max	I _{zk} (mA)	Max
3.3 ± 5 (%)	76	100	1.0	10	76	400	1.0	1380
3.6 ± 5 (%)	69	100	1.0	10	69	400	1.0	1260
3.9 ± 5 (%)	64	50	1.0	9	64	400	1.0	1190
4.3 ± 5 (%)	58	10	1.0	9	58	400	1.0	1070
4.7 ± 5 (%)	53	10	1.0	8	53	500	1.0	970
5.1 ± 5 (%)	49	10	1.0	7	49	550	1.0	890
5.6 ± 5 (%)	45	10	2.0	5	45	600	1.0	810
6.2 ± 5 (%)	41	10	3.0	2	41	700	1.0	730
6.8 ± 5 (%)	37	10	4.0	3.5	37	700	1.0	660
7.5 ± 5 (%)	34	10	5.0	4	34	700	0.5	605
8.2 ± 5 (%)	31	10	6.0	4.5	31	700	0.5	550
9.1 ± 5 (%)	28	10	7.0	5	28	700	0.5	500
10 ± 5 (%)	25	10	7.6	7	25	700	0.25	454
11 ± 5 (%)	23	5	8.4	8	23	700	0.25	414
12 ± 5 (%)	21	5	9.1	9	21	700	0.25	380
13 ± 5 (%)	19	5	9.9	10	19	700	0.25	344
15 ± 5 (%)	17	5	11.4	14	17	700	0.25	304
16 ± 5 (%)	15.5	5	12.2	16	15.5	750	0.25	285
18 ± 5 (%)	14.0	5	13.7	20	14.0	750	0.25	250
20 ± 5 (%)	12.5	5	15.2	22	12.5	750	0.25	225
22 ± 5 (%)	11.5	5	16.7	23	11.5	750	0.25	205
24 ± 5 (%)	10.5	5	18.2	25	10.5	750	0.25	190
27 ± 5 (%)	9.5	5	20.6	35	9.5	750	0.25	170
30 ± 5 (%)	8.5	5	22.8	40	8.5	1000	0.25	150
33 ± 5 (%)	7.5	5	25.1	45	7.5	1000	0.25	135
36 ± 5 (%)	7.0	5	27.4	50	7.0	1000	0.25	125
	3.3 ± 5 (%) 3.6 ± 5 (%) 4.3 ± 5 (%) 4.7 ± 5 (%) 5.1 ± 5 (%) 6.2 ± 5 (%) 6.2 ± 5 (%) 7.5 ± 5 (%) 9.1 ± 5 (%) 11 ± 5 (%) 12 ± 5 (%) 13 ± 5 (%) 15 ± 5 (%) 22 ± 5 (%) 22 ± 5 (%) 27 ± 5 (%) 33 ± 5 (%)	Max I_z (mA) 3.3 ± 5 (%) 76 3.6 ± 5 (%) 69 3.9 ± 5 (%) 64 4.3 ± 5 (%) 58 4.7 ± 5 (%) 53 5.1 ± 5 (%) 49 5.6 ± 5 (%) 45 6.2 ± 5 (%) 41 6.8 ± 5 (%) 37 7.5 ± 5 (%) 34 8.2 ± 5 (%) 31 9.1 ± 5 (%) 28 10 ± 5 (%) 25 11 ± 5 (%) 23 12 ± 5 (%) 19 15 ± 5 (%) 19 15 ± 5 (%) 17 16 ± 5 (%) 15.5 18 ± 5 (%) 14.0 20 ± 5 (%) 12.5 22 ± 5 (%) 11.5 24 ± 5 (%) 10.5 27 ± 5 (%) 8.5 33 ± 5 (%) 7.5	Test Condition Max I_z (mA) Max 3.3 ± 5 (%) 76 100 3.6 ± 5 (%) 69 100 3.9 ± 5 (%) 64 50 4.3 ± 5 (%) 58 10 4.7 ± 5 (%) 53 10 5.1 ± 5 (%) 49 10 5.6 ± 5 (%) 45 10 6.2 ± 5 (%) 41 10 6.8 ± 5 (%) 37 10 7.5 ± 5 (%) 34 10 8.2 ± 5 (%) 31 10 9.1 ± 5 (%) 28 10 10 ± 5 (%) 25 10 11 ± 5 (%) 23 5 12 ± 5 (%) 21 5 13 ± 5 (%) 19 5 15 ± 5 (%) 19 5 15 ± 5 (%) 15.5 5 18 ± 5 (%) 14.0 5 20 ± 5 (%) 12.5 5 22 ± 5 (%) 11.5 5	Max I_z (mA) Max V_R (V) 3.3 ± 5 (%) 76 100 1.0 3.6 ± 5 (%) 69 100 1.0 3.9 ± 5 (%) 64 50 1.0 4.3 ± 5 (%) 58 10 1.0 4.7 ± 5 (%) 53 10 1.0 5.1 ± 5 (%) 49 10 1.0 5.6 ± 5 (%) 45 10 2.0 6.2 ± 5 (%) 41 10 3.0 6.8 ± 5 (%) 37 10 4.0 7.5 ± 5 (%) 34 10 5.0 8.2 ± 5 (%) 31 10 6.0 9.1 ± 5 (%) 28 10 7.0 10 ± 5 (%) 25 10 7.6 11 ± 5 (%) 23 5 8.4 12 ± 5 (%) 17 5 11.4 16 ± 5 (%) 15.5 5 12.2 18 ± 5 (%) 14.0 5 13.7 20 ± 5 (%)	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	$\begin{array}{ c c c c c c c }\hline \textbf{Max} & \textbf{I}_{\textbf{z}} (\textbf{mA}) & \textbf{Max} & \textbf{V}_{\textbf{R}} (\textbf{V}) & \textbf{Max} & \textbf{I}_{\textbf{zr}} (\textbf{mA}) & \textbf{Max}\\\hline \textbf{3}.3 \pm 5 (\%) & 76 & 100 & 1.0 & 10 & 76 & 400\\\hline \textbf{3}.6 \pm 5 (\%) & 69 & 100 & 1.0 & 10 & 69 & 400\\\hline \textbf{3}.9 \pm 5 (\%) & 64 & 50 & 1.0 & 9 & 64 & 400\\\hline \textbf{4}.3 \pm 5 (\%) & 58 & 10 & 1.0 & 9 & 58 & 400\\\hline \textbf{4}.7 \pm 5 (\%) & 53 & 10 & 1.0 & 8 & 53 & 500\\\hline \textbf{5}.1 \pm 5 (\%) & 49 & 10 & 1.0 & 7 & 49 & 550\\\hline \textbf{5}.6 \pm 5 (\%) & 45 & 10 & 2.0 & 5 & 45 & 600\\\hline \textbf{6}.2 \pm 5 (\%) & 41 & 10 & 3.0 & 2 & 41 & 700\\\hline \textbf{6}.8 \pm 5 (\%) & 37 & 10 & 4.0 & 3.5 & 37 & 700\\\hline \textbf{7}.5 \pm 5 (\%) & 34 & 10 & 5.0 & 4 & 34 & 700\\\hline \textbf{8}.2 \pm 5 (\%) & 31 & 10 & 6.0 & 4.5 & 31 & 700\\\hline \textbf{9}.1 \pm 5 (\%) & 28 & 10 & 7.0 & 5 & 28 & 700\\\hline \textbf{10} \pm 5 (\%) & 23 & 5 & 8.4 & 8 & 23 & 700\\\hline \textbf{11} \pm 5 (\%) & 21 & 5 & 9.1 & 9 & 21 & 700\\\hline \textbf{13} \pm 5 (\%) & 19 & 5 & 9.9 & 10 & 19 & 700\\\hline \textbf{15} \pm 5 (\%) & 17 & 5 & 11.4 & 14 & 17 & 700\\\hline \textbf{16} \pm 5 (\%) & 12.5 & 5 & 12.2 & 16 & 15.5 & 750\\\hline \textbf{22} \pm 5 (\%) & 11.5 & 5 & 16.7 & 23 & 11.5 & 750\\\hline \textbf{24} \pm 5 (\%) & 10.5 & 5 & 18.2 & 25 & 10.5 & 750\\\hline \textbf{27} \pm 5 (\%) & 9.5 & 5 & 20.6 & 35 & 9.5 & 750\\\hline \textbf{30} \pm 5 (\%) & 8.5 & 5 & 22.8 & 40 & 8.5 & 1000\\\hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 25.1 & 45 & 7.5 & 1000\\\hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 25.1 & 45 & 7.5 & 1000\\\hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 25.1 & 45 & 7.5 & 1000\\\hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 25.1 & 45 & 7.5 & 1000\\\hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 25.1 & 45 & 7.5 & 1000\\\hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 25.1 & 45 & 7.5 & 1000\\\hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 25.1 & 45 & 7.5 & 1000\\\hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 25.1 & 45 & 7.5 & 1000\\\hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 25.1 & 45 & 7.5 & 1000\\\hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 25.1 & 45 & 7.5 & 1000\\\hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 25.1 & 45 & 7.5 & 1000\\\hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 25.1 & 45 & 7.5 & 1000\\\hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 25.1 & 45 & 7.5 & 1000\\\hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 25.1 & 45 & 7.5 & 1000\\\hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 25.1 & 45 & 7.5 & 1000\\\hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 25.1 & 45 & 7.5 & 1000\\\hline 33$	$\begin{array}{ c c c c c c c }\hline \textbf{Max} & \textbf{I}_{\textbf{z}}(\textbf{mA}) & \textbf{Max} & \textbf{V}_{\textbf{x}}(\textbf{V}) & \textbf{Max} & \textbf{I}_{\textbf{zr}}(\textbf{mA}) & \textbf{Max} & \textbf{I}_{\textbf{zx}}(\textbf{mA}) \\ \hline \textbf{Max} & \textbf{I}_{\textbf{z}}(\textbf{mA}) & \textbf{Max} & \textbf{V}_{\textbf{x}}(\textbf{V}) & \textbf{Max} & \textbf{I}_{\textbf{zr}}(\textbf{mA}) & \textbf{Max} & \textbf{I}_{\textbf{zx}}(\textbf{mA}) \\ \hline \textbf{3}.3 \pm 5 (\%) & 76 & 100 & 1.0 & 10 & 69 & 400 & 1.0 \\ \hline \textbf{3}.6 \pm 5 (\%) & 69 & 100 & 1.0 & 10 & 69 & 400 & 1.0 \\ \hline \textbf{3}.9 \pm 5 (\%) & 64 & 50 & 1.0 & 9 & 64 & 400 & 1.0 \\ \hline \textbf{4}.3 \pm 5 (\%) & 58 & 10 & 1.0 & 9 & 58 & 400 & 1.0 \\ \hline \textbf{4}.7 \pm 5 (\%) & 53 & 10 & 1.0 & 8 & 53 & 500 & 1.0 \\ \hline \textbf{5}.1 \pm 5 (\%) & 49 & 10 & 1.0 & 7 & 49 & 550 & 1.0 \\ \hline \textbf{5}.6 \pm 5 (\%) & 45 & 10 & 2.0 & 5 & 45 & 600 & 1.0 \\ \hline \textbf{6}.2 \pm 5 (\%) & 41 & 10 & 3.0 & 2 & 41 & 700 & 1.0 \\ \hline \textbf{6}.8 \pm 5 (\%) & 37 & 10 & 4.0 & 3.5 & 37 & 700 & 1.0 \\ \hline \textbf{7}.5 \pm 5 (\%) & 34 & 10 & 5.0 & 4 & 34 & 700 & 0.5 \\ \hline \textbf{8}.2 \pm 5 (\%) & 31 & 10 & 6.0 & 4.5 & 31 & 700 & 0.5 \\ \hline \textbf{9}.1 \pm 5 (\%) & 28 & 10 & 7.0 & 5 & 28 & 700 & 0.25 \\ \hline \textbf{11} \pm 5 (\%) & 23 & 5 & 8.4 & 8 & 23 & 700 & 0.25 \\ \hline \textbf{12} \pm 5 (\%) & 19 & 5 & 9.9 & 10 & 19 & 700 & 0.25 \\ \hline \textbf{13} \pm 5 (\%) & 19 & 5 & 9.9 & 10 & 19 & 700 & 0.25 \\ \hline \textbf{18} \pm 5 (\%) & 17 & 5 & 11.4 & 14 & 17 & 700 & 0.25 \\ \hline \textbf{18} \pm 5 (\%) & 12.5 & 5 & 15.2 & 22 & 12.5 & 750 & 0.25 \\ \hline \textbf{22} \pm 5 (\%) & 11.5 & 5 & 16.7 & 23 & 11.5 & 750 & 0.25 \\ \hline \textbf{22} \pm 5 (\%) & 10.5 & 5 & 18.2 & 25 & 10.5 & 750 & 0.25 \\ \hline \textbf{27} \pm 5 (\%) & 9.5 & 5 & 20.6 & 35 & 9.5 & 750 & 0.25 \\ \hline \textbf{27} \pm 5 (\%) & 9.5 & 5 & 20.6 & 35 & 9.5 & 750 & 0.25 \\ \hline \textbf{27} \pm 5 (\%) & 8.5 & 5 & 22.8 & 40 & 8.5 & 1000 & 0.25 \\ \hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 5 & 5.1 & 45 & 7.5 & 1000 & 0.25 \\ \hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 5 & 5.1 & 45 & 7.5 & 1000 & 0.25 \\ \hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 5 & 5.1 & 45 & 7.5 & 1000 & 0.25 \\ \hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 5 & 5.1 & 5.1 & 45 & 7.5 & 1000 & 0.25 \\ \hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 5.1 & 5.1 & 45 & 7.5 & 1000 & 0.25 \\ \hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 5.1 & 5.1 & 45 & 7.5 & 1000 & 0.25 \\ \hline \textbf{33} \pm 5 (\%) & 7.5 & 5 & 5.1 & 5.1 & 45 & 7.5 & 1000 & 0.25 \\ \hline \textbf$

Notes: 1. Tested with DC

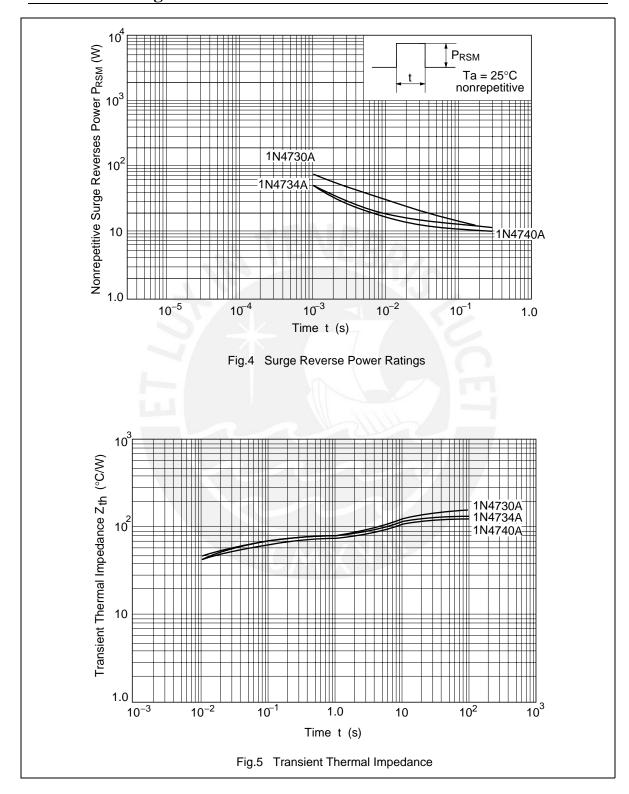
2. t = 1/120 sec reverse direction 1 pulse



Main Characteristic

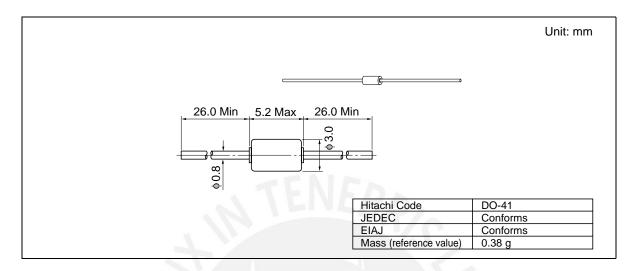








Package Dimensions





Disclaimer

- 1. Hitachi neither warrants nor grants licenses of any rights of Hitachi's or any third party's patent, copyright, trademark, or other intellectual property rights for information contained in this document. Hitachi bears no responsibility for problems that may arise with third party's rights, including intellectual property rights, in connection with use of the information contained in this document.
- 2. Products and product specifications may be subject to change without notice. Confirm that you have received the latest product standards or specifications before final design, purchase or use.
- 3. Hitachi makes every attempt to ensure that its products are of high quality and reliability. However, contact Hitachi's sales office before using the product in an application that demands especially high quality and reliability or where its failure or malfunction may directly threaten human life or cause risk of bodily injury, such as aerospace, aeronautics, nuclear power, combustion control, transportation, traffic, safety equipment or medical equipment for life support.
- 4. Design your application so that the product is used within the ranges guaranteed by Hitachi particularly for maximum rating, operating supply voltage range, heat radiation characteristics, installation conditions and other characteristics. Hitachi bears no responsibility for failure or damage when used beyond the guaranteed ranges. Even within the guaranteed ranges, consider normally foreseeable failure rates or failure modes in semiconductor devices and employ systemic measures such as fail-safes, so that the equipment incorporating Hitachi product does not cause bodily injury, fire or other consequential damage due to operation of the Hitachi product.
- 5. This product is not designed to be radiation resistant.
- 6. No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without written approval from Hitachi.
- 7. Contact Hitachi's sales office for any questions regarding this document or Hitachi semiconductor products.

Sales Offices

HITACHI

Hitachi, Ltd.

Semiconductor & Integrated Circuits Nippon Bldg., 2-6-2, Ohte-machi, Chiyoda-ku, Tokyo 100-0004, Japan Tel: (03) 3270-2111 Ftax: (03) 3270-5109

URL

NorthAmerica : http://semiconductor.hitachi.com/
Europe : http://www.hitachi-eu.com/hel/ecg
Asia : http://sicapac.hitachi-asia.com
Japan : http://www.hitachi.co.jp/Sicd/indx.htm

For further information write to:

Hitachi Semiconductor (America) Inc. 179 East Tasman Drive San Jose,CA 95134 Tel: <1> (408) 433-1990 Fax: <1>(408) 433-0223

Hitachi Europe Ltd.
Electronic Components Group
Whitebrook Park
Lower Cookham Road
Maidenhead
Berkshire SL6 8YA, United Kingdom
Tel: <44> (1628) 585000
Fax: <44> (1628) 585200

Hitachi Europe GmbH Electronic Components Group Dornacher Straße 3 D-85622 Feldkirchen, Munich Germany Tel: <49> (89) 9 9180-0 Fax: <49> (89) 9 29 30 00 Hitachi Asia Ltd. Hitachi Tower 16 Collyer Quay #20-00 Singapore 049318 Tel: <65>-538-6533/538-8577 Fax: <65>-538-6933/538-3877 URL: http://www.hitachi.com.sg

Hitachi Asia Ltd. (Taipei Branch Office) 4/F, No. 167, Tun Hwa North Road Hung-Kuo Building Taipei (105), Taiwan Tel: <886>-(2)-2718-3666

Tel: <886>-(2)-2/18-3666 Fax: <886>-(2)-2718-8180 Telex: 23222 HAS-TP URL: http://www.hitachi.com.tw Hitachi Asia (Hong Kong) Ltd. Group III (Electronic Components) 7/F., North Tower World Finance Centre, Harbour City, Canton Road Tam Sha Tsui, Kowloon Hong Kong

Tel : <852>-(2)-735-9218 Fax : <852>-(2)-730-0281 URL : http://semiconductor.hitachi.com.hk

URL: http://semiconductor.nitacni.com.ni

Copyright © Hitachi, Ltd., 2001. All rights reserved. Printed in Japan.

Colophon 4.0



This datasheet has been download from:

 $\underline{www.datasheet catalog.com}$

Datasheets for electronics components.







April 2014

2N3906 / MMBT3906 / PZT3906 PNP General-Purpose Amplifier

Description

This device is designed for general-purpose amplifier and switching applications at collector currents of 10 mA to 100 mA.



Ordering Information

Part Number	Marking	Package	Packing Method	Pack Quantity
2N3906BU	2N3906	TO-92 3L	Bulk	10000
2N3906TA	2N3906	TO-92 3L	Ammo	2000
2N3906TAR	2N3906	TO-92 3L	Ammo	2000
2N3906TF	2N3906	TO-92 3L	Tape and Reel	2000
2N3906TFR	2N3906	TO-92 3L	Tape and Reel	2000
MMBT3906	2A	SOT-23 3L	Tape and Reel	3000
PZT3906	3906	SOT-223 4L	Tape and Reel	2500

© 2010 Fairchild Semiconductor Corporation 2N3906 / MMBT3906 / PZT3906 Rev. 1.2.2



Absolute Maximum Ratings(1)

Stresses exceeding the absolute maximum ratings may damage the device. The device may not function or be operable above the recommended operating conditions and stressing the parts to these levels is not recommended. In addition, extended exposure to stresses above the recommended operating conditions may affect device reliability. The absolute maximum ratings are stress ratings only. Values are at $T_A = 25$ °C unless otherwise noted.

Symbol	Parameter	Value	Unit
V_{CEO}	Collector-Emitter Voltage	-40	V
V_{CBO}	Collector-Base Voltage	-40	V
V_{EBO}	Emitter-Base Voltage	-5.0	V
I _C	Collector Current - Continuous	-200	mA
T _{J,} T _{STG}	Operating and Storage Junction Temperature Range	-55 to +150	°C

Note:

These ratings are based on a maximum junction temperature of 150°C.
 These are steady-state limits. Fairchild Semiconductor should be consulted on applications involving pulsed or low-duty cycle operations.

Thermal Characteristics

Values are at T_A = 25°C unless otherwise noted.

Symbol	Parameter	Maximum			Unit
Symbol Parameter	Parameter	2N3906 ⁽³⁾	MMBT3906 ⁽²⁾	PZT3906 ⁽³⁾	Offic
D	Total Device Dissipation	625	350	1,000	mW
P_{D}	Derate Above 25°C	5.0	2.8	8.0	mW/°C
$R_{\theta JC}$	Thermal Resistance, Junction to Case	83.3			°C/W
$R_{\theta JA}$	Thermal Resistance, Junction to Ambient	200	357	125	°C/W

Notes:

- 2. Device is mounted on FR-4 PCB 1.6 inch X 1.6 inch X 0.06 inch.
- 3. PCB size: FR-4, 76 mm x 114 mm x 1.57 mm (3.0 inch x 4.5 inch x 0.062 inch) with minimum land pattern size.

© 2010 Fairchild Semiconductor Corporation 2N3906 / MMBT3906 / PZT3906 Rev. 1.2.2



Electrical Characteristics

Values are at $T_A = 25$ °C unless otherwise noted.

Symbol	Parameter	Conditions	Min.	Max.	Unit
OFF CHAR	ACTERISTICS				
V _{(BR)CEO}	Collector-Emitter Breakdown Voltage ⁽⁴⁾	I _C = -1.0 mA, I _B = 0	-40		V
V _{(BR)CBO}	Collector-Base Breakdown Voltage	$I_C = -10 \mu\text{A}, I_E = 0$	-40		V
V _{(BR)EBO}	Emitter-Base Breakdown Voltage	$I_E = -10 \mu A, I_C = 0$	-5.0		V
I _{BL}	Base Cut-Off Current	$V_{CE} = -30 \text{ V}, V_{BE} = 3.0 \text{ V}$		-50	nA
I _{CEX}	Collector Cut-Off Current	V _{CE} = -30 V, V _{BE} = 3.0 V		-50	nA
ON CHARA	CTERISTICS				
		$I_C = -0.1 \text{ mA}, V_{CE} = -1.0 \text{ V}$	60		
		I _C = -1.0 mA, V _{CE} = -1.0 V	80		
h _{FE}	DC Current Gain ⁽⁴⁾	I _C = -10 mA, V _{CE} = -1.0 V	100	300	
		$I_C = -50 \text{ mA}, V_{CE} = -1.0 \text{ V}$	60		
		I _C = -100 mA, V _{CE} = -1.0V	30		
\/ (aat)	Collector-Emitter Saturation	$I_C = -10 \text{ mA}, I_B = -1.0 \text{ mA}$		-0.25	V
V _{CE} (sat)	Voltage	$I_C = -50 \text{ mA}, I_B = -5.0 \text{ mA}$		-0.40	\ \
\/ (cot)	Page Emitter Seturation Voltage	I _C = -10 mA, I _B = -1.0 mA	-0.65	-0.85	V
V _{BE} (sat)	Base-Emitter Saturation Voltage	$I_C = -50 \text{ mA}, I_B = -5.0 \text{ mA}$		-0.95	_ v
SMALL SIG	NAL CHARACTERISTICS				
f _T	Current Gain - Bandwidth Product	I _C = -10 mA, V _{CE} = -20 V, f = 100 MHz	250		МН
C _{obo}	Output Capacitance	V _{CB} = -5.0 V, I _E = 0, f = 100 kHz		4.5	pF
C _{ibo}	Input Capacitance	V _{EB} = -0.5 V, I _C = 0, f = 100 kHz		10.0	pF
NF	Noise Figure	$\begin{split} I_C &= \text{-}100~\mu\text{A}, V_{CE} = \text{-}5.0~\text{V}, \\ R_S &= 1.0~\text{k}\Omega, \\ f &= 10~\text{Hz} \text{ to } 15.7~\text{kHz} \end{split}$		4.0	dB
SWITCHING	G CHARACTERISTICS				
t _d	Delay Time	V _{CC} = -3.0 V, V _{BE} = -0.5 V		35	ns
t _r	Rise Time	I _C = -10 mA, I _{B1} = -1.0 mA		35	ns
t _s	Storage Time	$V_{CC} = -3.0 \text{ V, } I_{C} = -10 \text{ mA,}$		225	ns
t _f	Fall Time	I _{B1} = I _{B2} = -1.0 mA		75	ns

Note

4. Pulse test: pulse width \leq 300 μ s, duty cycle \leq 2.0%.



Typical Performance Characteristics

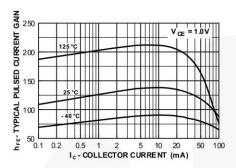


Figure 1. Typical Pulsed Current Gain vs. Collector Current

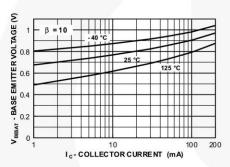


Figure 3. Base-Emitter Saturation Voltage vs. Collector Current

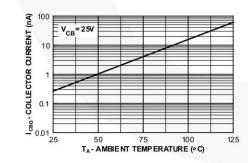


Figure 5. Collector Cut-Off Current vs. Ambient Temperature

4

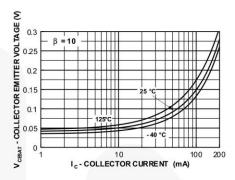


Figure 2. Collector-Emitter Saturation Voltage vs. Collector Current

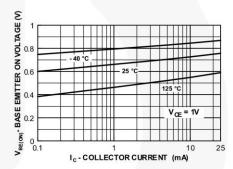


Figure 4. Base-Emitter On Voltage vs. Collector Current

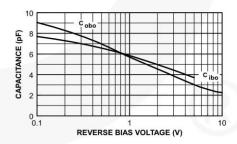


Figure 6. Common-Base Open Circuit Input and Output Capacitance vs. Reverse Bias Voltage

© 2010 Fairchild Semiconductor Corporation 2N3906 / MMBT3906 / PZT3906 Rev. 1.2.2



Typical Performance Characteristics (Continued)

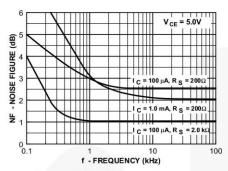


Figure 7. Noise Figure vs. Frequency

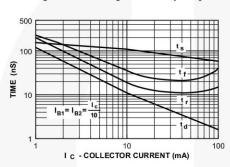


Figure 9. Switching Times vs. Collector Current

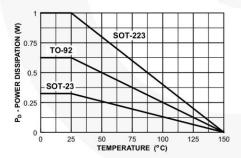


Figure 11. Power Dissipation vs. Ambient Temperature

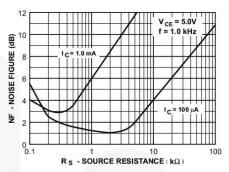


Figure 8. Noise Figure vs. Source Resistance

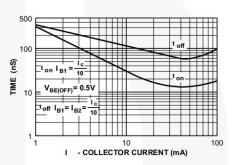


Figure 10. Turn-On and Turn-Off Times vs. Collector Current



Typical Performance Characteristics (Continued)

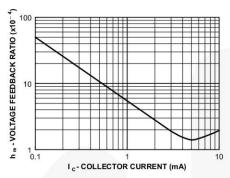


Figure 12. Voltage Feedback Ratio

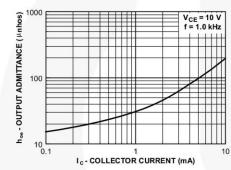


Figure 14. Output Admittance

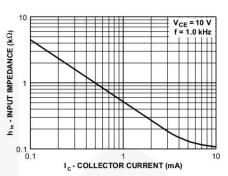


Figure 13. Input Impedance

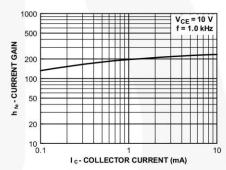


Figure 15. Current Gain



Physical Dimensions

TO-92 (Bulk)

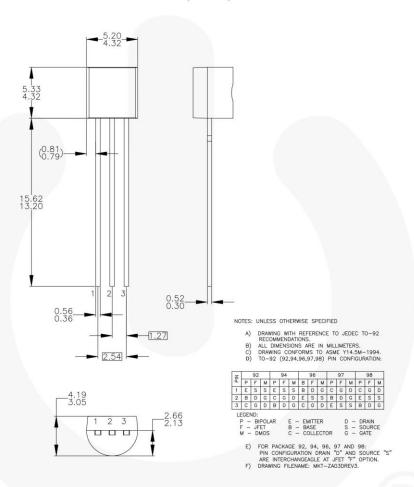


Figure 16. 3-LEAD, TO92, JEDEC TO-92 COMPLIANT STRAIGHT LEAD CONFIGURATION (OLD TO92AM3)

Package drawings are provided as a service to customers considering Fairchild components. Drawings may change in any manner without notice. Please note the revision and/or date on the drawing and contact a Fairchild Semiconductor representative to verify or obtain the most recent revision. Package specifications do not expand the terms of Fairchild's worldwide terms and conditions, specifically the warranty therein, which covers Fairchild products.

7

Always visit Fairchild Semiconductor's online packaging area for the most recent package drawings: http://www.fairchildsemi.com/dwg/ZA/ZAO3D.pdf.

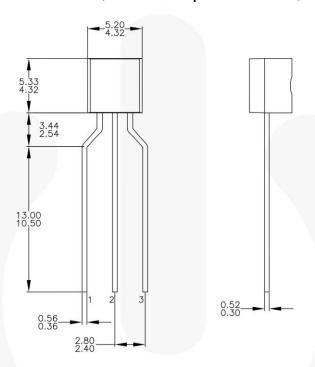
For current tape and reel specifications, visit Fairchild Semiconductor's online packaging area: http://www.fairchildsemi.com/packing_dwg/PKG-ZA03D_BK.pdf.

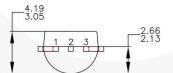
© 2010 Fairchild Semiconductor Corporation 2N3906 / MMBT3906 / PZT3906 Rev. 1.2.2



Physical Dimensions (Continued)

TO-92 (Ammo, Tape and Reel)





NOTES: UNLESS OTHERWISE SPECIFIED

- DRAWING CONFORMS TO JEDEC MS-013, VARIATION AC.
 ALL DIMENSIONS ARE IN MILLIMETERS.
 DRAWING CONFORMS TO ASME Y14.5M-2:
 DRAWING FILENAME: MKT-ZAO3FREV3.
 FAIRCHILD SEMICONDUCTOR

Figure 17. 3-LEAD, TO92, MOLDED 0.200 IN LINE SPACING LEAD FORM (J61Z OPTION)

Package drawings are provided as a service to customers considering Fairchild components. Drawings may change in any manner without notice. Please note the revision and/or date on the drawing and contact a Fairchild Semiconductor representative to verify or obtain the most recent revision. Package specifications do not expand the terms of Fairchild's worldwide terms and conditions, specifically the warranty therein, which covers Fairchild products.

Always visit Fairchild Semiconductor's online packaging area for the most recent package drawings: http://www.fairchildsemi.com/dwg/ZA/ZA03F.pdf.

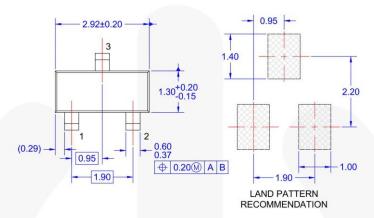
For current tape and reel specifications, visit Fairchild Semiconductor's online packaging area: http://www.fairchildsemi.com/packing_dwg/PKG-ZA03F_BK.pdf.

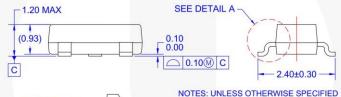
© 2010 Fairchild Semiconductor Corporation 2N3906 / MMBT3906 / PZT3906 Rev. 1.2.2

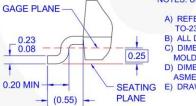


Physical Dimensions (Continued)

SOT-23







- A) REFERENCE JEDEC REGISTRATION
- TO-236, VARIATION AB, ISSUE H.
 B) ALL DIMENSIONS ARE IN MILLIMETERS.
 C) DIMENSIONS ARE INCLUSIVE OF BURRS,
 MOLD FLASH AND TIE BAR EXTRUSIONS.
- D) DIMENSIONING AND TOLERANCING PER ASME Y14.5M - 1994. SEATING E) DRAWING FILE NAME: MA03DREV10
- PLANE

Figure 18. 3-LEAD, SOT23, JEDEC TO-236, LOW PROFILE

Package drawings are provided as a service to customers considering Fairchild components. Drawings may change in any manner without notice. Please note the revision and/or date on the drawing and contact a Fairchild Semiconductor representative to verify or obtain the most recent revision. Package specifications do not expand the terms of Fairchild's worldwide terms and conditions, specifically the warranty therein, which covers Fairchild products.

Always visit Fairchild Semiconductor's online packaging area for the most recent package drawings: http://www.fairchildsemi.com/dwg/MA/MA03D.pdf.

DETAIL A

For current tape and reel specifications, visit Fairchild Semiconductor's online packaging area: http://www.fairchildsemi.com/packing_dwg/PKG-MA03D.pdf.

© 2010 Fairchild Semiconductor Corporation 2N3906 / MMBT3906 / PZT3906 Rev. 1.2.2



Physical Dimensions (Continued)

SOT-223 4L

6.70

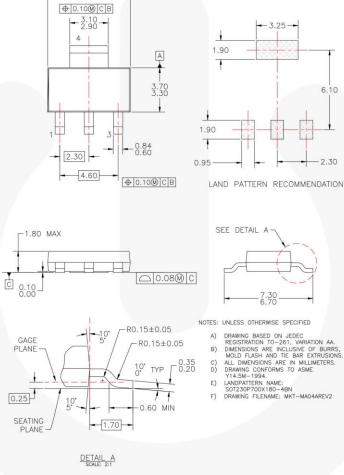


Figure 19. MOLDED PACKAGE, SOT-223, 4-LEAD

Package drawings are provided as a service to customers considering Fairchild components. Drawings may change in any manner without notice. Please note the revision and/or date on the drawing and contact a Fairchild Semiconductor representative to verify or obtain the most recent revision. Package specifications do not expand the terms of Fairchild's worldwide terms and conditions, specifically the warranty therein, which covers Fairchild products.

Always visit Fairchild Semiconductor's online packaging area for the most recent package drawings: http://www.fairchildsemi.com/dwg/MA/MA04A.pdf.

For current tape and reel specifications, visit Fairchild Semiconductor's online packaging area: http://www.fairchildsemi.com/packing_dwg/PKG-MA04A_BK.pdf.

© 2010 Fairchild Semiconductor Corporation 2N3906 / MMBT3906 / PZT3906 Rev. 1.2.2







TRADEMARKS

The following includes registered and unregistered trademarks and service marks, owned by Fairchild Semiconductor and/or its global subsidiaries, and is not intended to be an exhaustive list of all such trademarks.

AccuPower™ AX-CAP®* F-PFS™ FRFET® Global Power ResourceSM BitSiC™ GreenBridge™ Build it Now™ CorePLUS™ Green FPS™ CorePOWER™ Green FPS™ e-Series™ **CROSSVOLT**TM Gmax™ CTL™

GTO™ Current Transfer Logic™ IntelliMAX™ ISOPLANAR™ DEUXPEED⁶ Dual Cool™ Making Small Speakers Sound Louder EcoSPARK® EfficientMax™

ESBC™ Fairchild® Fairchild Semiconductor® FACT Quiet Series™ FACT[®] FastvCore™

and Better MegaBuck™ MICROCOUPLER™ MicroFET™ MicroPak™ MicroPak2™ MillerDrive™ MotionMax™ mWSaver® OPTOLOGIC® OPTOPLANAR®

PowerTrench® PowerXS™ Programmable Active Droop™ QS™

Quiet Series™ RapidConfigure™

> Saving our world, 1mW/W/kW at a time™ SignalWise™

SmartMax™ SMART START™ Solutions for Your Success $^{\text{TM}}$ SPM $^{\otimes}$

STEALTH™ SuperFET® SuperSOT™-3 SuperSOT™-6 SuperSOT™-8 SupreMOS® SyncFET™ Sync-Lock™

SYSTEM STERNER ST. TinyBoost[®] TinyBuck[®] TinyCalc™ TinyLogic[®] TINYOPTO™ TinyPower™ TinyPWM™ TinyWire™ TranSiC™ TriFault Detect TRUECURRENT®* μSerDes™

W SerDes UHC[®] Ultra FRFET™ UniFET™ **VCX™** VisualMax™ VoltagePlus™ XSTM 仙童™

DISCLAIMER

FETBench™

FPS™

FAIRCHILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO IMPROVE RELIABILITY, FUNCTION, OR DESIGN. FAIRCHILD DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN: NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS. THESE SPECIFICATIONS DO NOT EXPAND THE TERMS OF FAIRCHILD'S WORLDWIDE TERMS AND CONDITIONS, SPECIFICALLY THE WARRANTY THEREIN, WHICH COVERS THESE PRODUCTS.

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF FAIRCHILD SEMICONDUCTOR CORPORATION.

- 1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body or (b) support or sustain life, and (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury of the user.
- 2. A critical component in any component of a life support, device, or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness

ANTI-COUNTERFEITING POLICY

Fairchild Semiconductor Corporation's Anti-Counterfeiting Policy. Fairchild's Anti-Counterfeiting Policy is also stated on our external website, www.fairchildsemi.com, under Sales Support.

Counterfeiting of semiconductor parts is a growing problem in the industry. All manufacturers of semiconductor products are experiencing counterfeiting of their parts. Customers who inadvertently purchase counterfeit parts experience many problems such as loss of brand reputation, substandard performance, failed applications, and increased cost of production and manufacturing delays. Fairchild is taking strong measures to protect ourselves and our customers from the proliferation of counterfeit parts. Fairchild strongly encourages customers to purchase Fairchild parts either directly from Fairchild or from Authorized Fairchild Distributors who are listed by country on our web page cited above. Products customers buy either from Fairchild directly or from Authorized Fairchild Distributors are genuine parts, have full traceability, meet Fairchild's quality standards for handling and storage and provide access to Fairchild's full range of up-to-date technical and product information. Fairchild and our Authorized Distributors will stand behind all warranties and will appropriately address any warranty issues that may arise. Fairchild will not provide any warranty coverage or other assistance for parts bought from Unauthorized Sources. Fairchild is committed to combat this global problem and encourage our customers to do their part in stopping this practice by buying direct or from authorized distributors.

PRODUCT STATUS DEFINITIONS

Datasheet Identification	Product Status	Definition
Advance Information	Formative / In Design	Datasheet contains the design specifications for product development. Specifications may change in any manner without notice.
Preliminary	First Production	Datasheet contains preliminary data; supplementary data will be published at a later date. Fairchild Semiconductor reserves the right to make changes at any time without notice to improve design.
No Identification Needed	Full Production	Datasheet contains final specifications. Fairchild Semiconductor reserves the right to make changes at any time without notice to improve the design.
Obsolete	Not In Production	Datasheet contains specifications on a product that is discontinued by Fairchild Semiconductor. The datasheet is for reference information only.

Rev. 168

© Fairchild Semiconductor Corporation

^{*} Trademarks of System General Corporation, used under license by Fairchild Semiconductor.





November 1995

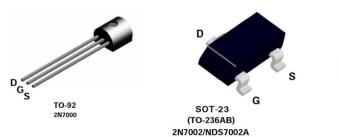
2N7000 / 2N7002 / NDS7002A N-Channel Enhancement Mode Field Effect Transistor

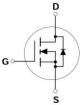
General Description

These N-Channel enhancement mode field effect transistors are produced using Fairchild's proprietary, high cell density, DMOS technology. These products have been designed to minimize on-state resistance while provide rugged, reliable, and fast switching performance. They can be used in most applications requiring up to 400mA DC and can deliver pulsed currents up to 2A. These products are particularly suited for low voltage, low current applications such as small servo motor control, power MOSFET gate drivers, and other switching applications.

Features

- High density cell design for low R_{DS(ON)}.
- Voltage controlled small signal switch.
- Rugged and reliable.
- High saturation current capability.





Absolute Maximum Ratings	T _A = 25°C unless otherwise noted
--------------------------	--

Symbol	Parameter	2N7000	2N7002	NDS7002A	Units	
V _{DSS}	Drain-Source Voltage		60 60 ±20 ±40 115 280 800 1500 200 300 1.6 2.4	<u> </u>	V	
V_{DGR}	Drain-Gate Voltage (R _{GS} ≤ 1 MW)		60		V	
V_{GSS}	Gate-Source Voltage - Continuous	±20			V	
	- Non Repetitive (tp < 50μs)		±40			
I _D	Maximum Drain Current - Continuous	200	115	280	mA	
	- Pulsed	500	800	1500	7	
P _D	Maximum Power Dissipation	400	200	300	mW	
	Derated above 25°C	3.2	1.6	2.4	mW/°C	
T_J, T_{STG}	Operating and Storage Temperature Range	-55 t	o 150	-65 to 150	°C	
T _L	Maximum Lead Temperature for Soldering Purposes, 1/16" from Case for 10 Seconds		300		°C	
THERMA	L CHARACTERISTICS					
R _{qJA}	Thermal Resistance, Junction-to-Ambient	312.5	625	417	°C/W	

^{© 1997} Fairchild Semiconductor Corporation



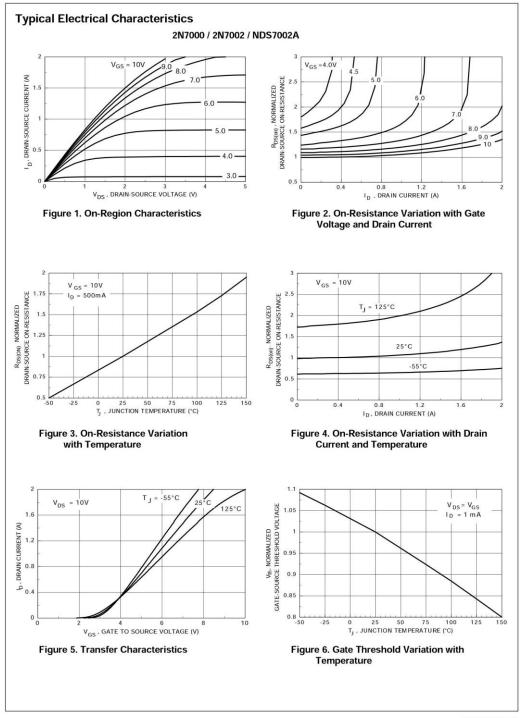
Symbol	cal Characteristics T _A = 25°C un Parameter	Conditions		Туре	Min	Тур	Max	Units
	RACTERISTICS			, ,				l
BV _{DSS}	Drain-Source Breakdown Voltage	$V_{GS} = 0 \text{ V}, I_{D} = 10 \mu\text{A}$		All	60			V
I _{DSS}	Zero Gate Voltage Drain Current	V _{DS} = 48 V, V _{GS} = 0 V		2N7000			1	μA
555			T ₁ =125°C				1	mA
		V _{DS} = 60 V, V _{GS} = 0 V		2N7002			1	μA
			T _J =125°C	NDS7002A			0.5	mA
I _{GSSF}	Gate - Body Leakage, Forward	V _{GS} = 15 V, V _{DS} = 0 V		2N7000			10	nA
		$V_{GS} = 20 \text{ V}, V_{DS} = 0 \text{ V}$		2N7002 NDS7002A			100	nA
GSSR	Gate - Body Leakage, Reverse	$V_{GS} = -15 \text{ V}, V_{DS} = 0 \text{ V}$		2N7000			-10	nA
		V_{GS} = -20 V, V_{DS} = 0 V		2N7002 NDS7002A			-100	nA
ON CHAR	RACTERISTICS (Note 1)							
$V_{GS(th)}$	Gate Threshold Voltage	$V_{DS} = V_{GS}$, $I_{D} = 1 \text{ mA}$		2N7000	8.0	2.1	3	V
		$V_{DS} = V_{GS}$, $I_D = 250 \mu A$		2N7002 NDS7002A	1	2.1	2.5	
R _{DS(ON)}	Static Drain-Source On-Resistance	$V_{GS} = 10 \text{ V}, I_{D} = 500 \text{ mA}$		2N7000	1	1.2	5	W
		T _J =125°C				1.9	9	
		$V_{GS} = 4.5 \text{ V}, I_{D} = 75 \text{ mA}$]		1.8	5.3	
		$V_{GS} = 10 \text{ V}, I_{D} = 500 \text{ mA}$		2N7002		1.2	7.5	
		0.000	T _J =100°C			1.7	13.5	
		$V_{GS} = 5.0 \text{ V}, I_{D} = 50 \text{ mA}$				1.7	7.5	1
			T _J =100C			2.4	13.5	
		$V_{GS} = 10 \text{ V}, I_{D} = 500 \text{ mA}$				1.2	2	
			T _J =125°C			2	3.5	
		$V_{GS} = 5.0 \text{ V}, I_{D} = 50 \text{ mA}$				1.7	3	
		-	T _J =125°C			2.8	5	
$V_{DS(ON)}$	Drain-Source On-Voltage	$V_{GS} = 10 \text{ V}, I_{D} = 500 \text{ mA}$		2N7000		0.6	2.5	V
		$V_{GS} = 4.5 \text{ V}, I_{D} = 75 \text{ mA}$	3			0.14	0.4	
		$V_{GS} = 10 \text{ V}, I_{D} = 500 \text{mA}$		2N7002		0.6	3.75	
		$V_{GS} = 5.0 \text{ V}, I_{D} = 50 \text{ mA}$				0.09	1.5	
		$V_{GS} = 10 \text{ V}, I_{D} = 500 \text{mA}$		NDS7002A		0.6	1	
		$V_{GS} = 5.0 \text{ V}, I_D = 50 \text{ mA}$		1		0.09	0.15	



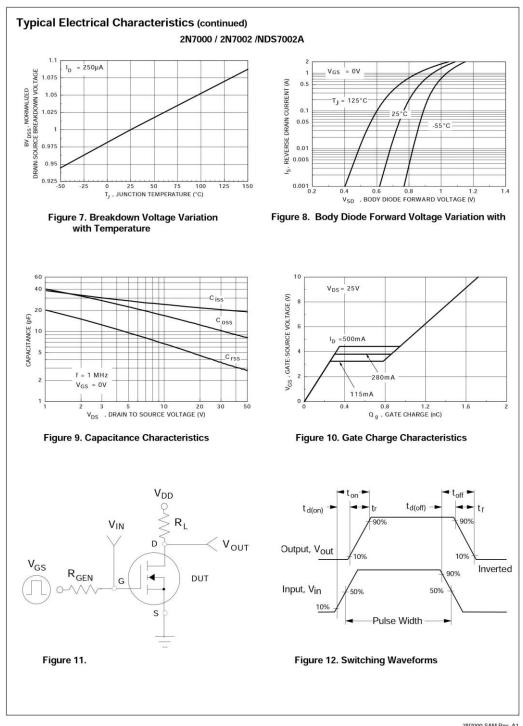
Symbol	Parameter	Conditions	Туре	Min	Тур	Max	Units
ON CHAP	RACTERISTICS Continued (Note 1)						
I _{D(ON)}	On-State Drain Current	$V_{GS} = 4.5 \text{ V}, \ V_{DS} = 10 \text{ V}$	2N7000	75	600		mA
		$V_{GS} = 10 \text{ V}, V_{DS} \ge 2 V_{DS(on)}$	2N7002	500	2700		
		$V_{GS} = 10 \text{ V}, V_{DS} \ge 2 V_{DS(on)}$	NDS7002A	500	2700		
g _{FS}	Forward Transconductance	$V_{DS} = 10 \text{ V}, I_{D} = 200 \text{ mA}$	2N7000	100	320		mS
		$V_{DS} \ge 2 V_{DS(on)'} I_D = 200 \text{ mA}$	2N7002	80	320		
		$V_{DS} \ge 2 V_{DS(on)}$, $I_D = 200 \text{ mA}$	NDS7002A	80	320		
DYNAMIC	CHARACTERISTICS						
C _{iss}	Input Capacitance	$V_{DS} = 25 \text{ V}, V_{GS} = 0 \text{ V},$ f = 1.0 MHz	All		20	50	pF
Coss	Output Capacitance		All		11	25	pF
C _{rss}	Reverse Transfer Capacitance		All		4	5	pF
t _{on}	Turn-On Time	$V_{DD} = 15 \text{ V}, R_L = 25 \text{ W}, I_D = 500 \text{ mA}, V_{GS} = 10 \text{ V}, R_{GEN} = 25$	2N7000			10	ns
		$V_{DD} = 30 \text{ V}, R_L = 150 \text{ W},$ $I_D = 200 \text{ mA}, V_{GS} = 10 \text{ V},$ $R_{GEN} = 25 \text{ W}$	2N7002 NDS7002A			20	
t _{off}	Turn-Off Time	$V_{DD} = 15 \text{ V}, R_L = 25 \text{ W},$ $I_D = 500 \text{ mA}, V_{GS} = 10 \text{ V},$ $R_{GEN} = 25$	2N7000			10	ns
		$V_{DD} = 30 \text{ V}, R_L = 150 \text{ W}, \\ I_D = 200 \text{ mA}, V_{GS} = 10 \text{ V}, \\ R_{GEN} = 25 \text{ W}$	2N7002 NDS7002A			20	
DRAIN-S	OURCE DIODE CHARACTERISTICS	AND MAXIMUM RATINGS					
I _s	Maximum Continuous Drain-Source	ce Diode Forward Current	2N7002			115	mA
			NDS7002A			280	
I _{SM}	Maximum Pulsed Drain-Source Di	ode Forward Current	2N7002			0.8	Α
			NDS7002A			1.5	
V _{SD}	Drain-Source Diode Forward	$V_{GS} = 0 \text{ V}, I_{S} = 115 \text{ mA (Note 1)}$	2N7002		0.88	1.5	V
	Voltage	$V_{GS} = 0 \text{ V}, I_{S} = 400 \text{ mA (Note 1)}$	NDS7002A		0.88	1.2	

Note: 1. Pulse Test: Pulse Width ≤ 300µs, Duty Cycle ≤ 2.0%.

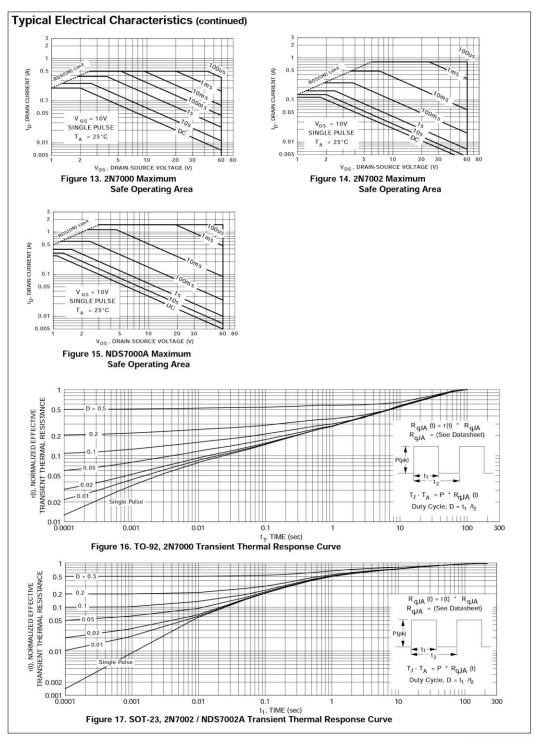














TRADEMARKS

The following are registered and unregistered trademarks Fairchild Semiconductor owns or is authorized to use and is not intended to be an exhaustive list of all such trademarks.

SyncFET™ **ACEx™** FASTr™ PowerTrench TinyLogic™ Bottomless™ QFET™ GlobalOptoisolator™ CoolFET™ OS™ **UHCTM** GTO™ $HiSeC^{TM}$ **VCX**TM QT Optoelectronics™ CROSSVOLT™ ISOPLANAR™ DOME™ Ouiet Series™ E2CMOS™ MICROWIRE™ SILENT SWITCHER EnSigna™ OPTOLOGIC™ SMART START™ FACT™ OPTOPLANAR™ SuperSOT™-3 SuperSOT™-6 FACT Quiet Series™ PACMAN™ SuperSOT™-8 **FAST** РОР™

DISCLAIMER

FAIRCHILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO IMPROVE RELIABILITY, FUNCTION OR DESIGN. FAIRCHILD DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS.

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF FAIRCHILD SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, or (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in significant injury to the A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

PRODUCT STATUS DEFINITIONS

Definition of Terms

Datasheet Identification	Product Status	Definition
Advance Information	Formative or In Design	This datasheet contains the design specifications for product development. Specifications may change in any manner without notice.
Preliminary	First Production	This datasheet contains preliminary data, and supplementary data will be published at a later date. Fairchild Semiconductor reserves the right to make changes at any time without notice in order to improve design.
No Identification Needed	Full Production	This datasheet contains final specifications. Fairchild Semiconductor reserves the right to make changes at any time without notice in order to improve design.
Obsolete	Not In Production	This datasheet contains specifications on a product that has been discontinued by Fairchild semiconductor. The datasheet is printed for reference information only.

Rev. G



2N7002P

60 V, 360 mA N-channel Trench MOSFET Rev. 02 — 29 July 2010

Product data sheet

1. Product profile

1.1 General description

N-channel enhancement mode Field-Effect Transistor (FET) in a small SOT23 (TO-236AB) Surface-Mounted Device (SMD) plastic package using Trench MOSFET technology.

1.2 Features and benefits

- AEC-Q101 qualified
- Logic-level compatible
- Trench MOSFET technology
- Very fast switching

1.3 Applications

- High-speed line driver
- Low-side loadswitch

- Relay driver
- Switching circuits

1.4 Quick reference data

Table 1. Quick reference data

Symbol	Parameter	Conditions		Min	Тур	Max	Unit
V_{DS}	drain-source voltage	T _{amb} = 25 °C		-	-	60	V
V_{GS}	gate-source voltage			-20	-	20	V
I _D	drain current	V_{GS} = 10 V; T_{amb} = 25 °C	[1]	-	9	360	mΑ
Static cha	racteristics						
R _{DSon}	drain-source on-state resistance	V_{GS} = 10 V; I_D = 500 mA; T_j = 25 °C; pulsed; t_p ≤ 300 µs; δ ≤ 0.01		-	1	1.6	Ω

^[1] Device mounted on an FR4 PCB, single-sided copper, tin-plated, mounting pad for drain 1 cm².





2N7002P

60 V, 360 mA N-channel Trench MOSFET

2. Pinning information

Table 2. Pinning information

Pin	Symbol	Description	Simplified outline	Graphic symbol
1	G	gate		_
2	S	source	3	B
3	D	drain	1 2	G (F)
			SOT23 (TO-236AB)	mbb076 S

3. Ordering information

Table 3. Ordering information

Type number	Package		
	Name	Description	Version
2N7002P	TO-236AB	plastic surface-mounted package; 3 leads	SOT23

4. Marking

Table 4. Marking codes

Type number	Marking code[1]	
2N7002P	LW%	

 $^{[1] \}quad \% = -: made \ in \ Hong \ Kong; \ \% = p: made \ in \ Hong \ Kong; \ \% = t: made \ in \ Malaysia; \ \% = W: made \ in \ China \ Malaysia; \ \% = W: made \ in \ Malaysia; \ \% = W: made \ in \ Malaysia; \ \% = W: made \ in \ Malaysia; \ \% = W: made \ in \ Malaysia; \ \% = W: made \ in \ Malaysia; \ \% = W: made \ in \ Malaysia; \ \% = W: made \ in \ Malaysia; \ \% = W: made \ in \ Malaysia; \ \% = W: made \ in \ Malaysia; \ Malaysia;$

5. Limiting values

Table 5. Limiting values

In accordance with the Absolute Maximum Rating System (IEC 60134).

Parameter	Conditions		Min	Max	Unit
drain-source voltage	T _{amb} = 25 °C		-	60	V
gate-source voltage			-20	20	V
drain current	V_{GS} = 10 V; T_{amb} = 25 °C	[1]	-	360	mA
	V_{GS} = 10 V; T_{amb} = 100 °C	[1]	-	280	mA
peak drain current	T_{amb} = 25 °C; single pulse; $t_p \le 10 \mu s$		=	1.2	Α
total power dissipation	T _{amb} = 25 °C	[2]	-	350	mW
		[1]	-	420	mW
	$T_{sp} = 25 ^{\circ}C$			1140	mW
junction temperature			-	150	°C
ambient temperature			-55	150	°C
storage temperature			-65	150	°C
in diode					
source current	T _{amb} = 25 °C	[1]	2	360	mA
	drain-source voltage gate-source voltage drain current peak drain current total power dissipation junction temperature ambient temperature storage temperature in diode	$\begin{array}{ll} \mbox{drain-source voltage} & T_{amb} = 25 \ ^{\circ}\mbox{C} \\ \mbox{gate-source voltage} \\ \mbox{drain current} & V_{GS} = 10 \ \mbox{V; } T_{amb} = 25 \ ^{\circ}\mbox{C} \\ \mbox{V}_{GS} = 10 \ \mbox{V; } T_{amb} = 100 \ ^{\circ}\mbox{C} \\ \mbox{peak drain current} & T_{amb} = 25 \ ^{\circ}\mbox{C; single pulse; } t_p \leq 10 \ \mu s \\ \mbox{total power dissipation} & T_{amb} = 25 \ ^{\circ}\mbox{C} \\ T_{sp} = 25 \ ^{\circ}\mbox{C} \\ \mbox{junction temperature} \\ \mbox{ambient temperature} \\ \mbox{storage temperature} \\ \mbox{in diode} & \end{array}$	$ \begin{array}{c} \text{drain-source voltage} \\ \text{gate-source voltage} \\ \text{drain current} \\ \\ \end{array} \begin{array}{c} V_{GS} = 10 \text{ V; } T_{amb} = 25 \text{ °C} \\ \hline V_{GS} = 10 \text{ V; } T_{amb} = 25 \text{ °C} \\ \hline V_{GS} = 10 \text{ V; } T_{amb} = 100 \text{ °C} \\ \hline \end{array} \begin{array}{c} \text{I1} \\ \text{peak drain current} \\ \text{total power dissipation} \\ \hline \\ T_{amb} = 25 \text{ °C; single pulse; } t_p \leq 10 \text{ µs} \\ \hline \\ T_{amb} = 25 \text{ °C} \\ \hline \\ \text{I1} \\ \hline \\ T_{Sp} = 25 \text{ °C} \\ \hline \\ \text{junction temperature} \\ \text{ambient temperature} \\ \text{storage temperature} \\ \hline \\ \text{in diode} \\ \hline \end{array} $	$\begin{array}{c} \text{drain-source voltage} \\ \text{gate-source voltage} \\ \end{array} \begin{array}{c} T_{amb} = 25 \ ^{\circ}\text{C} \\ \\ \text{gate-source voltage} \\ \end{array} \begin{array}{c} -20 \\ \\ \text{drain current} \\ \end{array} \begin{array}{c} V_{GS} = 10 \ \text{V}; \ T_{amb} = 25 \ ^{\circ}\text{C} \\ \hline V_{GS} = 10 \ \text{V}; \ T_{amb} = 100 \ ^{\circ}\text{C} \\ \end{array} \begin{array}{c} 11 \\ \text{l} \\ \text{-} \\ \end{array} \\ \text{peak drain current} \\ \text{peak drain current} \\ \end{array} \begin{array}{c} T_{amb} = 25 \ ^{\circ}\text{C}; \ \text{single pulse}; \ t_p \leq 10 \ \mu\text{s} \\ \hline T_{amb} = 25 \ ^{\circ}\text{C} \\ \hline \text{-} \\ \text{-} \\ \hline \text{-} \\ T_{Sp} = 25 \ ^{\circ}\text{C} \\ \end{array} \begin{array}{c} 2 \\ \text{-} \\ \hline \text{-} \\ \text{-} \\ \text{-} \\ \text{-} \\ \text{storage temperature} \\ \end{array} \begin{array}{c} -55 \\ \text{-} \\ \text{-}$	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$

^[1] Device mounted on an FR4 PCB, single-sided copper, tin-plated, mounting pad for drain 1 cm².

002P All information provided in this document is subject to legal disclaimers.

ONYP B.V. 2010. All rights reserved.

Product data sheet

Rev. 02 — 29 July 2010



2N7002P

60 V, 360 mA N-channel Trench MOSFET

[2] Device mounted on an FR4 PCB, single-sided copper, tin-plated and standard footprint.

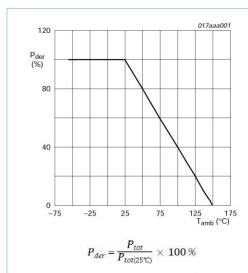


Fig 1. Normalized total power dissipation as a function of ambient temperature

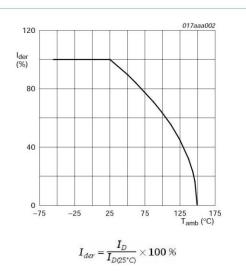
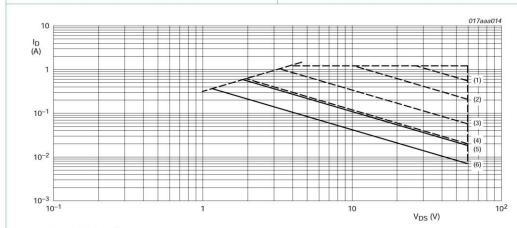


Fig 2. Normalized continuous drain current as a function of ambient temperature



I_{DM} = single pulse

(1) $t_p = 100 \, \mu s$

(2) $t_p = 1 \text{ ms}$

(3) $t_p = 10 \text{ ms}$

(4) $t_p = 100 \text{ ms}$

(5) DC; $T_{sp} = 25 \, ^{\circ}C$

(6) DC; T_{amb} = 25 °C; drain mounting pad 1 cm²

Fig 3. Safe operating area; junction to ambient; continuous and peak drain currents as a function of drain-source voltage

2N7002F

All information provided in this document is subject to legal disclaimers.

NXP B.V. 2010. All rights reserved.

Product data sheet

Rev. 02 — 29 July 2010



2N7002P

60 V, 360 mA N-channel Trench MOSFET

Thermal characteristics

Table 6. Thermal characteristics

I dibit o.	Thermal characteristics						
Symbol	Parameter	Conditions		Min	Тур	Max	Unit
$R_{th(j-a)}$	thermal resistance	in free air	[1]	-	310	370	K/W
	from junction to ambient	Ţ	[2]	-	260	300	K/W
$R_{th(j-sp)}$	thermal resistance from junction to solder point			-	-	115	K/W

- [1] Device mounted on an FR4 PCB, single-sided copper, tin-plated and standard footprint.
- [2] Device mounted on an FR4 PCB, single-sided copper, tin-plated, mounting pad for drain 1 cm².

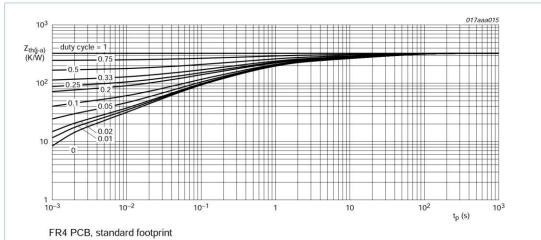
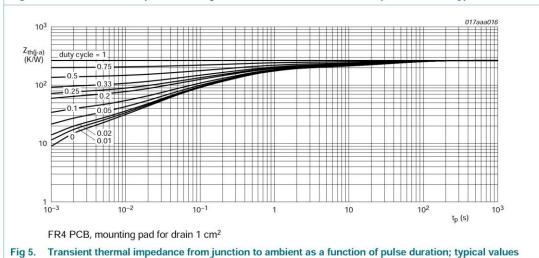


Fig 4. Transient thermal impedance from junction to ambient as a function of pulse duration; typical values



All information provided in this document is subject to legal disclaimers. Product data sheet Rev. 02 — 29 July 2010 4 of 15



2N7002P

60 V, 360 mA N-channel Trench MOSFET

7. Characteristics

Symbol	Parameter	Conditions	Min	Тур	Max	Unit
Static chara	cteristics					
V _{(BR)DSS}	drain-source breakdown voltage	I_D = 10 μ A; V_{GS} = 0 V; T_j = 25 °C	60	-	-	V
V_{GSth}	gate-source threshold voltage	$I_D = 250 \ \mu A; \ V_{DS} = V_{GS}; \ T_j = 25 \ ^{\circ}C$	1.1	1.75	2.4	V
I _{DSS}	drain leakage current	V_{DS} = 60 V; V_{GS} = 0 V; T_j = 25 °C	-	-	1	μΑ
		V_{DS} = 60 V; V_{GS} = 0 V; T_j = 150 °C	-	-	10	μΑ
I _{GSS} gate leakage current	V_{GS} = 20 V; V_{DS} = 0 V; T_j = 25 °C	-	-	100	nΑ	
		V_{GS} = -20 V; V_{DS} = 0 V; T_j = 25 °C	-	-	100	nA
R _{DSon}	drain-source on-state resistance	$V_{GS} = 5 \text{ V; } I_D = 50 \text{ mA; pulsed;}$ $t_p \le 300 \text{ µs; } \delta \le 0.01 \text{ ; } T_j = 25 \text{ °C}$		1.3	2	Ω
		V_{GS} = 10 V; I_{D} = 500 mA; pulsed; $t_{p} \le 300 \ \mu s$; $\delta \le 0.01 \ ; T_{j}$ = 25 °C	12	1	1.6	Ω
g _{fs}	forward transconductance	V_{DS} = 10 V; I_{D} = 200 mA; pulsed; $t_{p} \le 300 \ \mu s$; $\delta \le 0.01 \ ; T_{j}$ = 25 °C	-	400	Ē	mS
Dynamic ch	aracteristics					
Q _{G(tot)}	total gate charge	I_D = 300 mA; V_{DS} = 30 V; V_{GS} = 4.5 V;	(=)	0.6	0.8	nC
Q_{GS}	gate-source charge	$T_j = 25 ^{\circ}\text{C}$	920	0.2	1/2	nC
Q_{GD}	gate-drain charge		-	0.2	-	nC
C _{iss}	input capacitance	$V_{GS} = 0 \text{ V}; V_{DS} = 10 \text{ V}; f = 1 \text{ MHz};$	17	30	50	pF
C _{oss}	output capacitance	$T_j = 25 ^{\circ}\text{C}$	(-)	7		pF
C _{rss}	reverse transfer capacitance		-	4	-	pF
t _{d(on)}	turn-on delay time	V_{DS} = 50 V; R_L = 250 Ω ; V_{GS} = 10 V;	-	3	6	ns
t _r	rise time	$R_{G(ext)} = 6 \Omega$; $T_j = 25 °C$		4	-	ns
t _{d(off)}	turn-off delay time			10	20	ns
t _f	fall time			5	-	ns

2N7002F

 V_{SD}

source-drain voltage

All information provided in this document is subject to legal disclaimers.

 I_S = 115 mA; V_{GS} = 0 V; T_j = 25 °C

NXP B.V. 2010. All rights reserved.

0.47 0.75 1.1

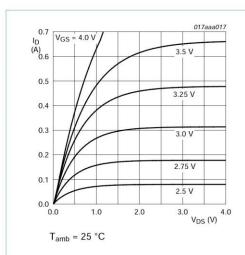
Product data sheet

Rev. 02 — 29 July 2010



2N7002P

60 V, 360 mA N-channel Trench MOSFET



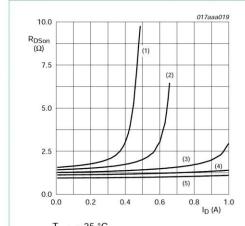
10⁻³ 017aaa018 I_D (A) (A) (2) (3) (3) (10⁻⁶ 10⁻⁶ 0 1 2 V_{GS} (V)

 $T_{amb} = 25 \, ^{\circ}C; \, V_{DS} = 5 \, V$

- (1) minimum values
- (2) typical values
- (3) maximum values

Fig 6. Output characteristics: drain current as a function of drain-source voltage; typical values





 $T_{amb} = 25 \, ^{\circ}C$

(1) $V_{GS} = 3.25 \text{ V}$

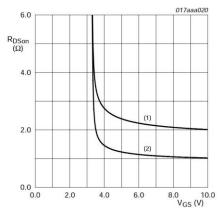
(2) $V_{GS} = 3.5 \text{ V}$

(3) $V_{GS} = 4 V$

(4) $V_{GS} = 5 V$

(5) $V_{GS} = 10 \text{ V}$





 $I_D = 500 \text{ mA}$

(1) T_{amb} = 150 °C

(2) T_{amb} = 25 °C

Fig 9. Drain-source on-state resistance as a function of gate-source voltage; typical values

2N7002P

All information provided in this document is subject to legal disclaimers.

NXP B.V. 2010. All rights reserved.

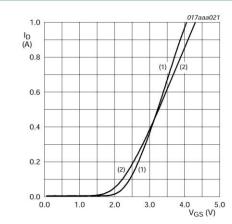
Product data sheet

Rev. 02 — 29 July 2010



2N7002P

60 V, 360 mA N-channel Trench MOSFET

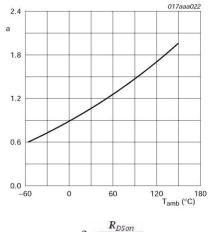


 $V_{DS} > I_D \times R_{DSon}$

(1) $T_{amb} = 25 \, ^{\circ}C$

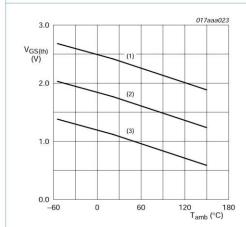
(2) $T_{amb} = 150 \, ^{\circ}C$

Fig 10. Transfer characteristics: drain current as a function of gate-source voltage; typical values



 $a = \frac{R_{DSon}}{R_{DSon(25^*C)}}$

Fig 11. Normalized drain-source on-state resistance as a function of ambient temperature; typical values



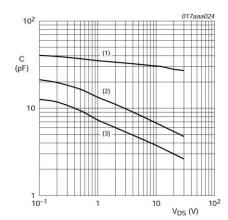
 I_D = 0.25 mA; V_{DS} = V_{GS}

(1) maximum values

(2) typical values

(3) minimum values

Fig 12. Gate-source threshold voltage as a function of ambient temperature



 $f = 1 \text{ MHz}; V_{GS} = 0 \text{ V}$

(1) C_{iss}

(2) C_{oss}

(3) C_{rss}

Fig 13. Input, output and reverse transfer capacitances as a function of drain-source voltage; typical values

2N7002

All information provided in this document is subject to legal disclaimers

© NXP B.V. 2010. All rights reserved.

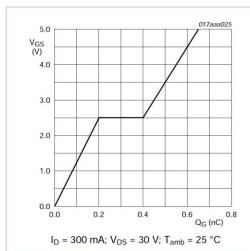
Product data sheet

Rev. 02 — 29 July 2010



2N7002P

60 V, 360 mA N-channel Trench MOSFET



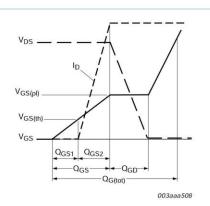
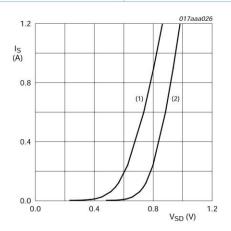


Fig 14. Gate-source voltage as a function of gate charge; typical values

Fig 15. Gate charge waveform definitions



 $V_{GS} = 0 V$

(1) $T_{amb} = 150 \, ^{\circ}C$

(2) T_{amb} = 25 °C

Fig 16. Source current as a function of source-drain voltage; typical values

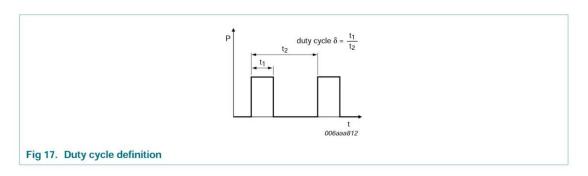
Product data sheet



2N7002P

60 V, 360 mA N-channel Trench MOSFET

8. Test information



Product data sheet



2N7002P

60 V, 360 mA N-channel Trench MOSFET

9. Package outline

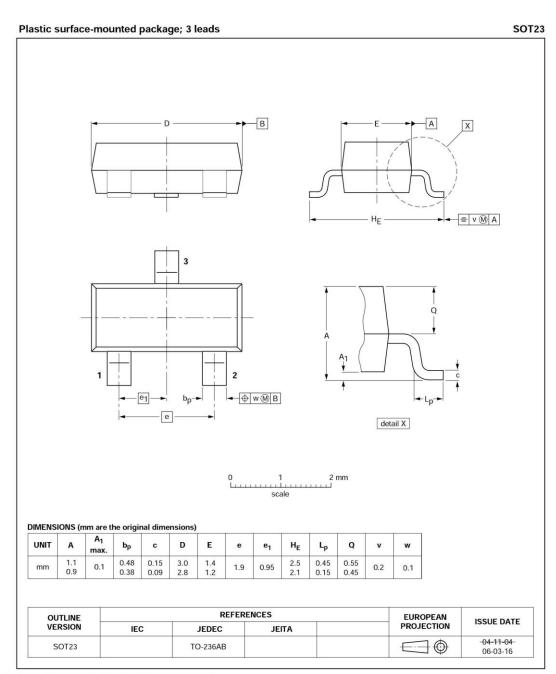


Fig 18. Package outline SOT23 (TO-236AB)

2N7002P All information provided in this document is subject to legal disclaimers. • NXP B.V. 2010. All rights reserved.

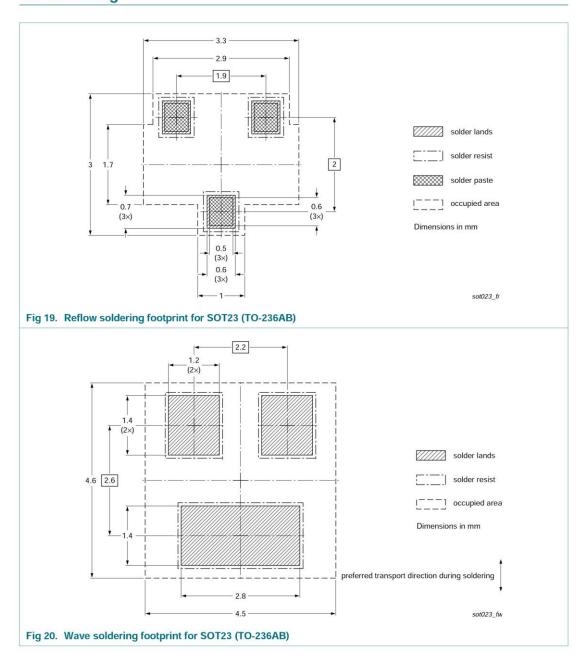
Product data sheet Rev. 02 — 29 July 2010 10 of 15



2N7002P

60 V, 360 mA N-channel Trench MOSFET

10. Soldering



2N7002

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2010. All rights reserved.

Product data sheet

Rev. 02 — 29 July 2010

11 of 15



2N7002P

60 V, 360 mA N-channel Trench MOSFET

11. Revision history

Table 8. Revision	on history					
Document ID	Release date	Data sheet status	Change notice	Supersedes		
2N7002P v.2	20100729	Product data sheet	-	2N7002P_1		
Modifications:	 Correction of t 	Correction of thermal values.				
	 Correction of v 	arious characteristics value	s including related grap	hs.		
2N7002P_1	20100419	Product data sheet	(4)	2		



2N7002P

60 V, 360 mA N-channel Trench MOSFET

12. Legal information

12.1 Data sheet status

Document status[1][2]	Product status[3]	Definition
Objective [short] data sheet	Development	This document contains data from the objective specification for product development.
Preliminary [short] data sheet	Qualification	This document contains data from the preliminary specification.
Product [short] data sheet	Production	This document contains the product specification.

- [1] Please consult the most recently issued document before initiating or completing a design.
- [2] The term 'short data sheet' is explained in section "Definitions".
- [3] The product status of device(s) described in this document may have changed since this document was published and may differ in case of multiple devices. The latest product status information is available on the Internet at URL http://www.nxp.com.

12.2 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

Short data sheet — A short data sheet is an extract from a full data sheet with the same product type number(s) and title. A short data sheet is intended for quick reference only and should not be relied upon to contain detailed and full information. For detailed and full information see the relevant full data sheet, which is available on request via the local NXP Semiconductors sales office. In case of any inconsistency or conflict with the short data sheet, the full data sheet shall prevail.

Product specification — The information and data provided in a Product data sheet shall define the specification of the product as agreed between NXP Semiconductors and its customer, unless NXP Semiconductors and customer have explicitly agreed otherwise in writing. In no event however, shall an agreement be valid in which the NXP Semiconductors product is deemed to offer functions and qualities beyond those described in the Product data sheet.

12.3 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Limiting values — Stress above one or more limiting values (as defined in the Absolute Maximum Ratings System of IEC 60134) will cause permanent damage to the device. Limiting values are stress ratings only and (proper) operation of the device at these or any other conditions above those given in the Recommended operating conditions section (if present) or the Characteristics sections of this document is not warranted. Constant or repeated exposure to limiting values will permanently and irreversibly affect the quality and reliability of the device.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

2N7002F

All information provided in this document is subject to legal disclaimers.

NXP B.V. 2010. All rights reserved

Product data sheet

Rev. 02 — 29 July 2010

13 of 15



2N7002P

60 V, 360 mA N-channel Trench MOSFET

No offer to sell or license — Nothing in this document may be interpreted or construed as an offer to sell products that is open for acceptance or the grant, conveyance or implication of any license under any copyrights, patents or other industrial or intellectual property rights.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from national authorities.

Quick reference data — The Quick reference data is an extract of the product data given in the Limiting values and Characteristics sections of this document, and as such is not complete, exhaustive or legally binding.

12.4 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

Adelante, Bitport, Bitsound, CoolFlux, CoReUse, DESFire, EZ-HV, FabKey, GreenChip, HiPerSmart, HITAG, IPC-bus logo, ICODE, I-CODE, ITEC, Labelution, MIFARE, MIFARE Plus, MIFARE Ultralight, MoReUse, QLPAK, Silicon Tuner, SiliconMAX, SmartXA, STARplug, TOPFET, TrenchMOS, TriMedia and UCODE — are trademarks of NXP B.V.

HD Radio and **HD Radio** logo — are trademarks of iBiquity Digital Corporation.

13. Contact information

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com



2N7002P

60 V, 360 mA N-channel Trench MOSFET

14. Contents

1	Product profile
1.1	General description
1.2	Features and benefits
1.3	Applications
1.4	Quick reference data
2	Pinning information
3	Ordering information
4	Marking
5	Limiting values
6	Thermal characteristics
7	Characteristics
8	Test information
9	Package outline
10	Soldering11
11	Revision history12
12	Legal information13
12.1	Data sheet status
12.2	Definitions13
12.3	Disclaimers
12.4	Trademarks14
13	Contact information 14

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© NXP B.V. 2010.

All rights reserved.

For more information, please visit: http://www.nxp.com For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 29 July 2010 Document identifier: 2N7002P





ST – Store Indirect From Register to Data Space using Index X

Description:

Stores one byte indirect from a register to data space. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. The EEPROM has a separate address space.

The data location is pointed to by the X (16 bits) Pointer Register in the Register File. Memory access is limited to the current data segment of 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPX in register in the I/O area has to be changed.

The X-pointer Register can either be left unchanged by the operation, or it can be post-incremented or pre-decremented. These features are especially suited for accessing arrays, tables, and Stack Pointer usage of the X-pointer Register. Note that only the low byte of the X-pointer is updated in devices with no more than 256 bytes data space. For such devices, the high byte of the pointer is not used by this instruction and can be used for other purposes. The RAMPX Register in the I/O area is updated in parts with more than 64K bytes data space or more than 64K bytes Program memory, and the increment/ decrement is added to the entire 24-bit address on such devices.

Not all variants of this instruction is available in all devices. Refer to the device specific instruction set summary.

The result of these combinations is undefined:

ST X+, r26 ST X+, r27 ST -X, r26 ST -X, r27

Using the X-pointer:

	Operation:		Comment:
(i)	$(X) \leftarrow Rr$		X: Unchanged
(ii)	$(X) \leftarrow Rr$	X ← X+1	X: Post incremented
(iii)	X ← X - 1	$(X) \leftarrow Rr$	X: Pre decremented
	Syntax:	Operands:	Program Counter:
(i)	ST X, Rr	$0 \le r \le 31$	PC ← PC + 1
(ii)	ST X+, Rr	$0 \le r \le 31$	PC ← PC + 1
(iii)	ST -X, Rr	$0 \le r \le 31$	$PC \leftarrow PC + 1$

16-bit Opcode:

(i)	1001	001r	rrrr	1100
(ii)	1001	001r	rrrr	1101
(iii)	1001	001r	rrrr	1110

Status Register (SREG) and Boolean Formula:

1	T	Н	s	V	N	Z	С	
-	-	-	-	-	-	-	1-0	

144 AVR Instruction Set



■ AVR Instruction Set

Example:

```
clr
      r27
                 ; Clear X high byte
ldi
    r26,$60
                 ; Set X low byte to $60
st
      X+,r0
                 ; Store r0 in data space loc. $60(X post inc)
st
      X,r1
                 ; Store r1 in data space loc. $61
                 ; Set X low byte to $63
ldi
      r26,$63
                 ; Store r2 in data space loc. $63
      X,r2
st
                  ; Store r3 in data space loc. $62(X pre dec)
st
      -X,r3
```

Words: 1 (2 bytes)

 Cycles:
 2

 Cycles XMEGA:
 (i) 1

 (ii) 1
 (iii) 2

 Cycles Reduced Core tinyAVR:(i) 1
 1

(ii) 1 (iii) 2







ST (STD) – Store Indirect From Register to Data Space using Index Y

Description:

Stores one byte indirect with or without displacement from a register to data space. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. The EEPROM has a separate address space.

The data location is pointed to by the Y (16 bits) Pointer Register in the Register File. Memory access is limited to the current data segment of 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPY in register in the I/O area has to be changed.

The Y-pointer Register can either be left unchanged by the operation, or it can be post-incremented or pre-decremented. These features are especially suited for accessing arrays, tables, and Stack Pointer usage of the Y-pointer Register. Note that only the low byte of the Y-pointer is updated in devices with no more than 256 bytes data space. For such devices, the high byte of the pointer is not used by this instruction and can be used for other purposes. The RAMPY Register in the I/O area is updated in parts with more than 64K bytes data space or more than 64K bytes Program memory, and the increment/ decrement/displacement is added to the entire 24-bit address on such devices.

Not all variants of this instruction is available in all devices. Refer to the device specific instruction set summary.

The result of these combinations is undefined:

ST Y+, r28 ST Y+, r29 ST -Y, r28 ST -Y, r29

Using the Y-pointer:

Operation:

(i)	$(Y) \leftarrow Rr$	
(ii)	$(Y) \leftarrow Rr$	$Y \leftarrow Y+1$
(iii)	$Y \leftarrow Y - 1$	$(Y) \leftarrow Rr$

(iv) $(Y+q) \leftarrow Rr$

	Syntax:	Operands:
i)	STY, Rr	$0 \le r \le 31$
ii)	ST Y+ Br	0 < r < 31

(iii) ST -Y, Rr $0 \le r \le 31$ (iv) STD Y+q, Rr $0 \le r \le 31, 0 \le q \le 63$

16-bit Opcode:

Г	(i)	1000	001r	rrrr	1000
Г	(ii)	1001	001r	rrrr	1001
Т	(iii)	1001	001r	rrrr	1010
	(iv)	10q0	qq1r	rrrr	1qqq

Comment:

Y: Unchanged Y: Post incremented Y: Pre decremented

Y: Unchanged, q: Displacement

Program Counter:

 $PC \leftarrow PC + 1$ $PC \leftarrow PC + 1$ $PC \leftarrow PC + 1$ $PC \leftarrow PC + 1$

146 AVR Instruction Set



■ AVR Instruction Set

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С	
-	8-1	-	1 - 1	-	—	-	1-1	

Example:

; Clear Y high byte r29 clr r28,\$60 ; Set Y low byte to \$60 ldi st Y+,r0 ; Store r0 in data space loc. \$60(Y post inc) Y,r1 ; Store r1 in data space loc. \$61 ldi r28,\$63 ; Set Y low byte to \$63 ; Store r2 in data space loc. \$63 st Y,r2 -Y,r3 st ; Store r3 in data space loc. \$62(Y pre dec) std Y+2,r4 ; Store r4 in data space loc. \$64

Words: 1 (2 bytes)

Cycles: 2 Cycles XMEGA: (i) 1

(ii) 1 (iii) 2

(iv) 2

Cycles Reduced Core tinyAVR:(i) 1

(ii) 1

(iii) 2





ST (STD) – Store Indirect From Register to Data Space using Index Z

Description:

Stores one byte indirect with or without displacement from a register to data space. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. The EEPROM has a separate address space.

The data location is pointed to by the Z (16 bits) Pointer Register in the Register File. Memory access is limited to the current data segment of 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPZ in register in the I/O area has to be changed.

The Z-pointer Register can either be left unchanged by the operation, or it can be post-incremented or pre-decremented. These features are especially suited for Stack Pointer usage of the Z-pointer Register, however because the Z-pointer Register can be used for indirect subroutine calls, indirect jumps and table lookup, it is often more convenient to use the X or Y-pointer as a dedicated Stack Pointer. Note that only the low byte of the Z-pointer is updated in devices with no more than 256 bytes data space. For such devices, the high byte of the pointer is not used by this instruction and can be used for other purposes. The RAMPZ Register in the I/O area is updated in parts with more than 64K bytes data space or more than 64K bytes Program memory, and the increment/decrement/displacement is added to the entire 24-bit address on such devices.

Not all variants of this instruction is available in all devices. Refer to the device specific instruction set summary.

The result of these combinations is undefined:

ST Z+, r30

ST Z+, r31

ST -Z, r30

ST -Z, r31

Using the Z-pointer:

(i) (ii) (iii) (iv)	Operation: (Z) \leftarrow Rr (Z) \leftarrow Rr Z \leftarrow Z - 1 (Z+q) \leftarrow Rr	$Z \leftarrow Z+1$ $(Z) \leftarrow Rr$	Comment: Z: Unchanged Z: Post incremented Z: Pre decremented Z: Unchanged, q: Displacement
	Syntax:	Operands:	Program Counter:
(i)	ST Z, Rr	$0 \le r \le 31$	PC ← PC + 1
(ii)	ST Z+, Rr	$0 \le r \le 31$	PC ← PC + 1
(iii)	ST -Z, Rr	$0 \le r \le 31$	PC ← PC + 1
(iv)	STD Z+q, Rr	$0 \leq r \leq 31, 0 \leq q \leq 63$	PC ← PC + 1

AVR Instruction Set

0856I-AVR-07/10

148



AVR Instruction Set

16-bit Opcode:

(i)	1000	001r	rrrr	0000
(ii)	1001	001r	rrrr	0001
(iii)	1001	001r	rrrr	0010
(iv)	10q0	qq1r	rrrr	0qqq

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С	
-		_	-	-	-	_	-	

Example:

clr r31 ; Clear Z high byte ldi r30,\$60 ; Set Z low byte to \$60 Z+,r0 ; Store r0 in data space loc. \$60(Z post inc) st Z,r1 ; Store r1 in data space loc. \$61 ldi r30,\$63 ; Set Z low byte to \$63 st Z,r2 ; Store r2 in data space loc. \$63 -Z,r3 ; Store r3 in data space loc. \$62(Z pre dec) st Z+2,r4 ; Store r4 in data space loc. \$64 std

Words: 1 (2 bytes)

Cycles: 2

Cycles XMEGA: (i) 1

(ii) 1 (iii) 2

(iv) 2

(iv) 2

Cycles Reduced Core tinyAVR:(i) 1

(ii) 1

(iii) 2





STS - Store Direct to Data Space

Description:

Stores one byte from a Register to the data space. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. The EEPROM has a separate address space.

A 16-bit address must be supplied. Memory access is limited to the current data segment of 64K bytes. The STS instruction uses the RAMPD Register to access memory above 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPD in register in the I/O area has to be changed.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) $(k) \leftarrow Rr$

Syntax: Operands: Program Counter: (i) STS k,Rr $0 \le r \le 31, 0 \le k \le 65535$ PC \leftarrow PC + 2

32-bit Opcode:

1001	001d	dddd	0000
kkkk	kkkk	kkkk	kkkk

Status Register (SREG) and Boolean Formula:

1	T	Н	s	V	N	Z	С	
_	1-	-	-	-	_	_	-]

Example:

lds r2,\$FF00 ; Load r2 with the contents of data space location \$FF00
add r2,r1 ; add r1 to r2
sts \$FF00,r2 ; Write back

Words: 2 (4 bytes)

Cycles: 2

150 AVR Instruction Set



AVR Instruction Set

STS (16-bit) - Store Direct to Data Space

Description:

Stores one byte from a Register to the data space. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. In some parts the Flash memory has been mapped to the data space and can be written using this command. The EEPROM has a separate address space.

A 7-bit address must be supplied. The address given in the instruction is coded to a data space address as follows:

 $ADDR[7:0] = (\overline{INST[8]}, INST[8], INST[10], INST[9], INST[3], INST[2], INST[1], INST[0])$

Memory access is limited to the address range 0x40...0xbf of the data segment.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) $(k) \leftarrow Rr$

16-bit Opcode:

	1010	1kkk	dddd	kkkk
--	------	------	------	------

Status Register (SREG) and Boolean Formula:

1	T	н	s	V	N	Z	С
-	-	-	-	-	-	-	-

Example:

lds r16,\$00 ; Load r16 with the contents of data space location \$00 add r16,r17 ; add r17 to r16 sts \$00,r16 ; Write result to the same address it was fetched from

Words: 1 (2 bytes)

Cycles: 1

Note: Registers r0..r15 are remaped to r16..r31







SUB - Subtract without Carry

Description:

Subtracts two registers and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd - Rr$

16-bit Opcode:

0001 10rd dddd rrrr

Status Register and Boolean Formula:

1	Т	Н	S	V	N	Z	С	
_	-	⇔	\Leftrightarrow	\Leftrightarrow	⇔	⇔	\Leftrightarrow	

H: Rd3• Rr3 +Rr3 •R3 +R3• Rd3

Set if there was a borrow from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: Rd7• Rr7 •R7 +Rd7 •Rr7• R7

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7• R6 •R5• R4• R3 •R2• R1• R0

Set if the result is \$00; cleared otherwise.

C: Rd7• Rr7 +Rr7 •R7 +R7• Rd7

Set if the absolute value of the contents of Rr is larger than the absolute value of Rd; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

sub r13,r12 ; Subtract r12 from r13
brne noteq ; Branch if r12<>r13
...

noteq: no

; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1

152 AVR Instruction Set



AVR Instruction Set

SUBI - Subtract Immediate

Description:

Subtracts a register and a constant and places the result in the destination register Rd. This instruction is working on Register R16 to R31 and is very well suited for operations on the X, Y and Z-pointers.

Operation:

(i) $Rd \leftarrow Rd - K$

 $\begin{array}{ccc} \text{Syntax:} & \text{Operands:} & \text{Program Counter:} \\ \text{(i)} & \text{SUBI Rd,K} & 16 \leq d \leq 31, \, 0 \leq K \leq 255 & \text{PC} \leftarrow \text{PC} + 1 \\ \end{array}$

16-bit Opcode:

0101	KKKK	dddd	KKKK
50.00	2011012012011	100000000000000000000000000000000000000	

Status Register and Boolean Formula:

1	Т	Н	s	V	N	Z	С	
n = //	-	⇔	⇔	⇔	⇔	⇔	\Leftrightarrow]

H: Rd3• K3+K3 •R3 +R3 •Rd3

Set if there was a borrow from bit 3; cleared otherwise

- S: $N \oplus V$, For signed tests.
- V: Rd7• K7 •R7 +Rd7• K7 •R7

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7

Set if MSB of the result is set; cleared otherwise.

- Z: R7• R6 •R5• R4• R3 •R2• R1• R0
 - Set if the result is \$00; cleared otherwise.
- C: Rd7 K7 +K7 •R7 +R7 Rd7
 Set if the absolute value of K is larger than the absolute value of Rd; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
subi r22,$11 ; Subtract $11 from r22
brne noteq ; Branch if r22<>$11
...
noteq: nop ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1



153





SWAP - Swap Nibbles

Description:

Swaps high and low nibbles in a register.

Operation:

(i) $R(7:4) \leftarrow Rd(3:0), R(3:0) \leftarrow Rd(7:4)$

Syntax: Operands: **Program Counter:** (i) SWAP Rd $0 \le d \le 31$ $PC \leftarrow PC + 1$

16-bit Opcode:

١	1001	010d	dddd	0010
	1001	0100	adda	0010

Status Register and Boolean Formula:

1	Т	н	s	V	N	Z	С
-	-	-	-	-	-	-	-

R (Result) equals Rd after the operation.

Example:

inc r1 ; Increment r1

r1 ; Swap high and low nibble of r1 inc ; Increment high nibble of r1 r1 ; Swap back

Words: 1 (2 bytes)

Cycles: 1

AVR Instruction Set 154



AVR Instruction Set

TST - Test for Zero or Minus

Description:

Tests if a register is zero or negative. Performs a logical AND between a register and itself. The register will remain unchanged.

Operation:

(i) $Rd \leftarrow Rd \bullet Rd$

16-bit Opcode: (see AND Rd, Rd)

	0010	00dd	dddd	dddd
--	------	------	------	------

Status Register and Boolean Formula:

ı	Т	Н	S	V	N	Z	C	
_	-	-	⇔	0	\Leftrightarrow	\Leftrightarrow	:-::	

S: $N \oplus V$, For signed tests.

V: 0

Cleared

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$ Set if the result is \$00; cleared otherwise.

R (Result) equals Rd.

Example:

tst r0 ; Test r0 breq zero ; Branch if r0=0

. . .

nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

zero:

Cycles: 1







WDR - Watchdog Reset

Description:

This instruction resets the Watchdog Timer. This instruction must be executed within a limited time given by the WD prescaler. See the Watchdog Timer hardware specification.

Operation:

(i) WD timer restart.

Syntax: Operands: (i) WDR None

Program Counter:

PC ← PC + 1

16-bit Opcode:

1001 0101 1010 1000

Status Register and Boolean Formula:

1		Т	Н	s	V	N	Z	С	
	_	-	-	-	_	-	_	-	

Example:

wdr ; Reset watchdog timer

Words: 1 (2 bytes)

Cycles: 1



■ AVR Instruction Set

XCH - Exchange

Description:

Operation:

(i) $(Z) \leftarrow Rd, Rd \leftarrow (Z)$

Syntax:Operands:XCH Z,Rd $0 \le d \le 31$

16-bit Opcode:

1001 001r rrrr 0100

Words: 1 (2 bytes)

Cycles: 1

(i)







Datasheet Revision History

Please note that the referring page numbers in this section are referred to this document. The referring revision in this section is referred to the document revision.

Rev.0856I - 07/10

- 1. Updated "Complete Instruction Set Summary" on page 11 with new instructions: LAC, LAS, LAT and XCH.
 - "LAC Load And Clear" on page 84
 - "LAS Load And Set" on page 85
 - "LAT Load And Toggle" on page 86
 - "XCH Exchange" on page 157
- 2. Updated number of clock cycles column to include Reduced Core tinyAVR.
 - (ATtiny replaced by Reduced Core tinyAVR).

Rev.0856H - 04/09

- 1. Updated "Complete Instruction Set Summary" on page 11:
 - Updated number of clock cycles column to include Reduced Core tinyAVR.
- 2. Updated sections for Reduced Core tinyAVR compatibility:
 - "CBI Clear Bit in I/O Register" on page 48
 - "LD Load Indirect from Data Space to Register using Index X" on page 87
 - "LD (LDD) Load Indirect from Data Space to Register using Index Y" on page 90
 - "LD (LDD) Load Indirect From Data Space to Register using Index Z" on page 92
 - "RCALL Relative Call to Subroutine" on page 114
 - "SBI Set Bit in I/O Register" on page 123
 - "ST Store Indirect From Register to Data Space using Index X" on page 144
 - "ST (STD) Store Indirect From Register to Data Space using Index Y" on page 146
 - "ST (STD) Store Indirect From Register to Data Space using Index Z" on page 148
- 3. Added sections for Reduced Core tinyAVR compatibility:
 - "LDS (16-bit) Load Direct from Data Space" on page 96
 - "STS (16-bit) Store Direct to Data Space" on page 151

Rev.0856G - 07/08

- 1. Inserted "Datasheet Revision History"
- 2. Updated "Cycles XMEGA" for ST, by removing (iv).
- 3. Updated "SPM #2" opcodes.

158 AVR Instruction Set



■ AVR Instruction Set

Rev.0856F - 05/08

1. This revision is based on the AVR Instruction Set 0856E-AVR-11/05

Changes done compared to AVR Instruction Set 0856E-AVR-11/05:

- Updated "Complete Instruction Set Summary" with DES and SPM #2.
- Updated AVR Instruction Set with XMEGA Clock cycles and Instruction Description.







Headquarters

Atmel Corporation

2325 Orchard Parkway San Jose, CA 95131

USA

Tel: 1(408) 441-0311 Fax: 1(408) 487-2600

International

Atmel Asia

Unit 1-5 & 16, 19/F BEA Tower, Millennium City 5 418 Kwun Tong Road Kwun Tong, Kowloon Hong Kong

Tel: (852) 2245-6100

Fax: (852) 2722-1369

Atmel Europe

Le Krebs 8, Rue Jean-Pierre Timbaud BP 309

78054 Saint-Quentin-en-Yvelines Cedex France

Tel: (33) 1-30-60-70-00 Fax: (33) 1-30-60-71-11 Atmel Japan

9F, Tonetsu Shinkawa Bldg. 1-24-8 Shinkawa Chuo-ku, Tokyo 104-0033 Japan

Tel: (81) 3-3523-3551 Fax: (81) 3-3523-7581

Product Contact

Web Site

www.atmel.com

Technical Support

avr@atmel.com

Sales Contact

www.atmel.com/contacts

Literature Requests

www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2010 Atmel Corporation. All rights reserved. Atmel®, Atmel logo and combinations thereof, AVR®, AVR® logo, tinyAVR® and others are registered trademarks, XMEGATM and others are trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.



Instruction Set Nomenclature

Status Register (SREG)

SREG: Status Register
C: Carry Flag
Z: Zero Flag
N: Negative Flag

V: Two's complement overflow indicator

S: $N \oplus V$, For signed tests

H: Half Carry Flag

T: Transfer bit used by BLD and BST instructions

I: Global Interrupt Enable/Disable Flag

Registers and Operands

Rd: Destination (and source) register in the Register File

Rr: Source register in the Register File
R: Result after instruction is executed

K: Constant datak: Constant address

b: Bit in the Register File or I/O Register (3-bit)

s: Bit in the Status Register (3-bit)

X,Y,Z: Indirect Address Register

(X=R27:R26, Y=R29:R28 and Z=R31:R30)

A: I/O location address

q: Displacement for direct addressing (6-bit)



8-bit **AVR**® Instruction Set

Rev. 0856I-AVR-07/10







I/O Registers

RAMPX, RAMPY, RAMPZ

Registers concatenated with the X-, Y-, and Z-registers enabling indirect addressing of the whole data space on MCUs with more than 64K bytes data space, and constant data fetch on MCUs with more than 64K bytes program space.

RAMPD

Register concatenated with the Z-register enabling direct addressing of the whole data space on MCUs with more than 64K bytes data space.

EIND

Register concatenated with the Z-register enabling indirect jump and call to the whole program space on MCUs with more than 64K words (128K bytes) program space.

Stack

STACK: Stack for return address and pushed registers

SP: Stack Pointer to STACK

Flags

⇔: Flag affected by instruction
 0: Flag cleared by instruction
 1: Flag set by instruction

-: Flag not affected by instruction

2 AVR Instruction Set



AVR Instruction Set

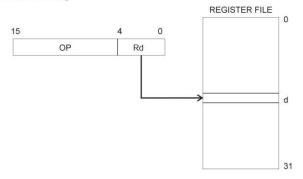
The Program and Data Addressing Modes

The AVR Enhanced RISC microcontroller supports powerful and efficient addressing modes for access to the Program memory (Flash) and Data memory (SRAM, Register file, I/O Memory, and Extended I/O Memory). This section describes the various addressing modes supported by the AVR architecture. In the following figures, OP means the operation code part of the instruction word. To simplify, not all figures show the exact location of the addressing bits. To generalize, the abstract terms RAMEND and FLASHEND have been used to represent the highest location in data and program space, respectively.

Note: Not all addressing modes are present in all devices. Refer to the device spesific instruction summary.

Register Direct, Single Register Rd

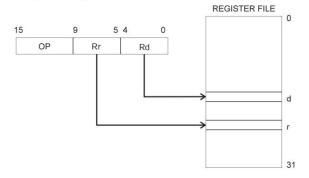
Figure 1. Direct Single Register Addressing



The operand is contained in register d (Rd).

Register Direct, Two Registers Rd and Rr

Figure 2. Direct Register Addressing, Two Registers



Operands are contained in register r (Rr) and d (Rd). The result is stored in register d (Rd).

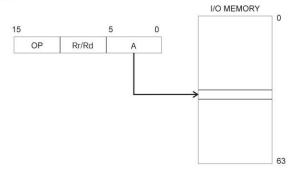






I/O Direct

Figure 3. I/O Direct Addressing

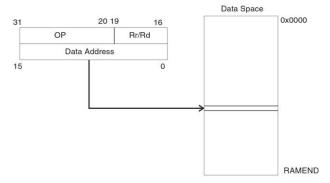


Operand address is contained in 6 bits of the instruction word. n is the destination or source register address.

Note: Some complex AVR Microcontrollers have more peripheral units than can be supported within the 64 locations reserved in the opcode for I/O direct addressing. The extended I/O memory from address 64 to 255 can only be reached by data addressing, not I/O addressing.

Data Direct

Figure 4. Direct Data Addressing



A 16-bit Data Address is contained in the 16 LSBs of a two-word instruction. Rd/Rr specify the destination or source register.

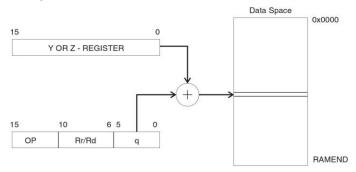
4 AVR Instruction Set



AVR Instruction Set

Data Indirect with Displacement

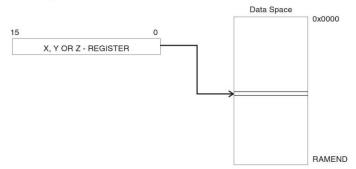
Figure 5. Data Indirect with Displacement



Operand address is the result of the Y- or Z-register contents added to the address contained in 6 bits of the instruction word. Rd/Rr specify the destination or source register.

Data Indirect

Figure 6. Data Indirect Addressing



Operand address is the contents of the X-, Y-, or the Z-register. In AVR devices without SRAM, Data Indirect Addressing is called Register Indirect Addressing. Register Indirect Addressing is a subset of Data Indirect Addressing since the data space form 0 to 31 is the Register File.

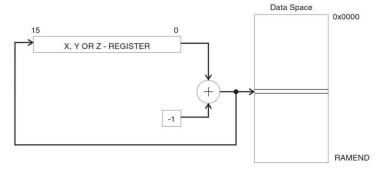






Data Indirect with Pre-decrement

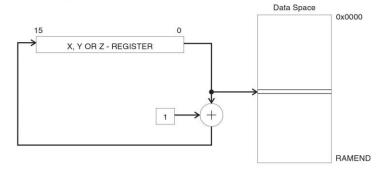
Figure 7. Data Indirect Addressing with Pre-decrement



The X,- Y-, or the Z-register is decremented before the operation. Operand address is the decremented contents of the X-, Y-, or the Z-register.

Data Indirect with Post-increment

Figure 8. Data Indirect Addressing with Post-increment



The X-, Y-, or the Z-register is incremented after the operation. Operand address is the content of the X-, Y-, or the Z-register prior to incrementing.

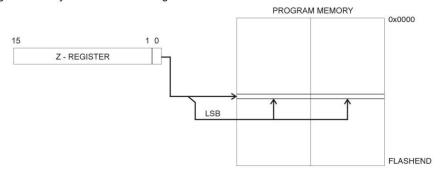
6 AVR Instruction Set



AVR Instruction Set

Program Memory Constant Addressing using the LPM, ELPM, and SPM Instructions

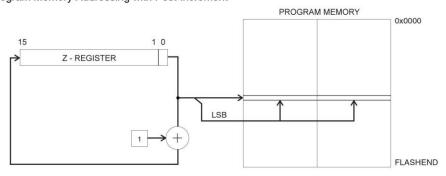
Figure 9. Program Memory Constant Addressing



Constant byte address is specified by the Z-register contents. The 15 MSBs select word address. For LPM, the LSB selects low byte if cleared (LSB = 0) or high byte if set (LSB = 1). For SPM, the LSB should be cleared. If ELPM is used, the RAMPZ Register is used to extend the Z-register.

Program Memory with Post-increment using the LPM Z+ and ELPM Z+ Instruction

Figure 10. Program Memory Addressing with Post-increment



Constant byte address is specified by the Z-register contents. The 15 MSBs select word address. The LSB selects low byte if cleared (LSB = 0) or high byte if set (LSB = 1). If ELPM Z+ is used, the RAMPZ Register is used to extend the Z-register.

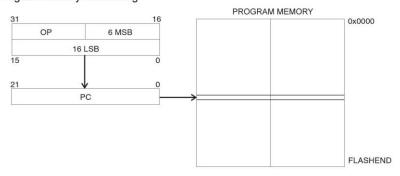






Direct Program Addressing, JMP and CALL

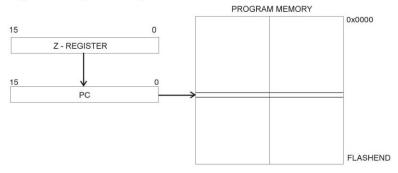
Figure 11. Direct Program Memory Addressing



Program execution continues at the address immediate in the instruction word.

Indirect Program Addressing, IJMP and ICALL

Figure 12. Indirect Program Memory Addressing



Program execution continues at address contained by the Z-register (i.e., the PC is loaded with the contents of the Z-register).

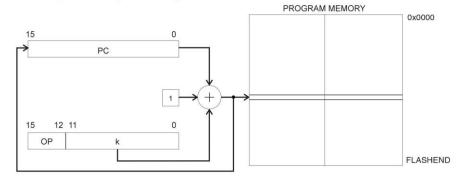
8 AVR Instruction Set



■ AVR Instruction Set

Relative Program Addressing, RJMP and RCALL

Figure 13. Relative Program Memory Addressing



Program execution continues at address PC + k + 1. The relative address k is from -2048 to 2047.







Conditional Branch Summary

Test	Boolean	Mnemonic	Complementary	Boolean	Mnemonic	Comment
Rd > Rr	Z•(N ⊕ V) = 0	BRLT ⁽¹⁾	Rd ≤ Rr	Z+(N ⊕ V) = 1	BRGE*	Signed
Rd □ Rr	(N ⊕ V) = 0	BRGE	Rd < Rr	(N ⊕ V) = 1	BRLT	Signed
Rd = Rr	Z = 1	BREQ	Rd ≠ Rr	Z = 0	BRNE	Signed
Rd ≤ Rr	Z+(N ⊕ V) = 1	BRGE ⁽¹⁾	Rd > Rr	Z•(N ⊕ V) = 0	BRLT*	Signed
Rd < Rr	(N ⊕ V) = 1	BRLT	Rd ≥ Rr	(N ⊕ V) = 0	BRGE	Signed
Rd > Rr	C + Z = 0	BRLO ⁽¹⁾	Rd ≤ Rr	C + Z = 1	BRSH*	Unsigned
Rd 🗆 Rr	C = 0	BRSH/BRCC	Rd < Rr	C = 1	BRLO/BRCS	Unsigned
Rd = Rr	Z = 1	BREQ	Rd ≠ Rr	Z = 0	BRNE	Unsigned
$Rd \leq Rr$	C + Z = 1	BRSH ⁽¹⁾	Rd > Rr	C + Z = 0	BRLO*	Unsigned
Rd < Rr	C = 1	BRLO/BRCS	Rd ≥ Rr	C = 0	BRSH/BRCC	Unsigned
Carry	C = 1	BRCS	No carry	C = 0	BRCC	Simple
Negative	N = 1	BRMI	Positive	N = 0	BRPL	Simple
Overflow	V = 1	BRVS	No overflow	V = 0	BRVC	Simple
Zero	Z = 1	BREQ	Not zero	Z = 0	BRNE	Simple

Note: 1. Interchange Rd and Rr in the operation before the test, i.e., CP Rd,Rr \rightarrow CP Rr,Rd



■ AVR Instruction Set

Complete Instruction Set Summary

Instruction Set Summary

Mnemonics	Operands	Description	Opera	ation		Flags	#Clocks	#Clocks XMEGA
		Ariti	nmetic and Logic Instructions	3				
ADD	Rd, Rr	Add without Carry	Rd	←	Rd + Rr	Z,C,N,V,S,H	1	
ADC	Rd, Rr	Add with Carry	Rd	←	Rd + Rr + C	Z,C,N,V,S,H	1	
ADIW ⁽¹⁾	Rd, K	Add Immediate to Word	Rd	←	Rd + 1:Rd + K	Z,C,N,V,S	2	
SUB	Rd, Rr	Subtract without Carry	Rd	←	Rd - Rr	Z,C,N,V,S,H	1	
SUBI	Rd, K	Subtract Immediate	Rd	←	Rd - K	Z,C,N,V,S,H	1	
SBC	Rd, Rr	Subtract with Carry	Rd	←	Rd - Rr - C	Z,C,N,V,S,H	1	
SBCI	Rd, K	Subtract Immediate with Carry	Rd	←	Rd - K - C	Z,C,N,V,S,H	1	
SBIW ⁽¹⁾	Rd, K	Subtract Immediate from Word	Rd + 1:Rd	←	Rd + 1:Rd - K	Z,C,N,V,S	2	
AND	Rd, Rr	Logical AND	Rd	←	Rd • Rr	Z,N,V,S	1	
ANDI	Rd, K	Logical AND with Immediate	Rd	←	Rd • K	Z,N,V,S	1	
OR	Rd, Rr	Logical OR	Rd	←	Rd v Rr	Z,N,V,S	1	
ORI	Rd, K	Logical OR with Immediate	Rd	←	Rd v K	Z,N,V,S	1	
EOR	Rd, Rr	Exclusive OR	Rd	←	Rd ⊕ Rr	Z,N,V,S	1	
СОМ	Rd	One's Complement	Rd	←	\$FF - Rd	Z,C,N,V,S	1	
NEG	Rd	Two's Complement	Rd	←	\$00 - Rd	Z,C,N,V,S,H	1	
SBR	Rd,K	Set Bit(s) in Register	Rd	←	Rd v K	Z,N,V,S	1	
CBR	Rd,K	Clear Bit(s) in Register	Rd	←	Rd • (\$FFh - K)	Z,N,V,S	1	
INC	Rd	Increment	Rd	←	Rd + 1	Z,N,V,S	1	
DEC	Rd	Decrement	Rd	←	Rd - 1	Z,N,V,S	1	
TST	Rd	Test for Zero or Minus	Rd	←	Rd • Rd	Z,N,V,S	1	
CLR	Rd	Clear Register	Rd	←	Rd ⊕ Rd	Z,N,V,S	1	
SER	Rd	Set Register	Rd	←	\$FF	None	1	
MUL ⁽¹⁾	Rd,Rr	Multiply Unsigned	R1:R0	←	Rd x Rr (UU)	Z,C	2	
MULS ⁽¹⁾	Rd,Rr	Multiply Signed	R1:R0	←	Rd x Rr (SS)	Z,C	2	
MULSU ⁽¹⁾	Rd,Rr	Multiply Signed with Unsigned	R1:R0	←	Rd x Rr (SU)	Z,C	2	
FMUL ⁽¹⁾	Rd,Rr	Fractional Multiply Unsigned	R1:R0	←	Rd x Rr<<1 (UU)	Z,C	2	
FMULS ⁽¹⁾	Rd,Rr	Fractional Multiply Signed	R1:R0	←	Rd x Rr<<1 (SS)	Z,C	2	
FMULSU ⁽¹⁾	Rd,Rr	Fractional Multiply Signed with Unsigned	R1:R0	←	Rd x Rr<<1 (SU)	Z,C	2	
DES	К	Data Encryption	if (H = 0) then R15:R0 else if (H = 1) then R15:R0	←	Encrypt(R15:R0, K) Decrypt(R15:R0, K)			1/2
		Bra	nch Instructions					
RJMP	k	Relative Jump	PC	←	PC + k + 1	None	2	
IJMP ⁽¹⁾		Indirect Jump to (Z)	PC(15:0) PC(21:16)	←	Z, 0	None	2	
EIJMP ⁽¹⁾		Extended Indirect Jump to (Z)	PC(15:0) PC(21:16)	←	Z, EIND	None	2	
JMP ⁽¹⁾	k	Jump	PC	←	k	None	3	







Mnemonics				Flags	#Clocks	#Clocks XMEGA		
RCALL	k	Relative Call Subroutine	PC	←	PC + k + 1	None	3 / 4 ⁽³⁾⁽⁵⁾	2 / 3(3)
ICALL ⁽¹⁾		Indirect Call to (Z)	PC(15:0) PC(21:16)	←	Z, 0	None	3 / 4 ⁽³⁾	2/3 ⁽³⁾
EICALL ⁽¹⁾		Extended Indirect Call to (Z)	PC(15:0) PC(21:16)	←	Z, EIND	None	4 (3)	3 (3)
CALL ⁽¹⁾	k	call Subroutine	PC	←	k	None	4 / 5 ⁽³⁾	3 / 4 ⁽³⁾
RET		Subroutine Return	PC	←	STACK	None	4 / 5 ⁽³⁾	
RETI		Interrupt Return	PC	←	STACK	1	4 / 5(3)	
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC	←	PC + 2 or 3	None	1/2/3	
СР	Rd,Rr	Compare	Rd - Rr			Z,C,N,V,S,H	1	
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C			Z,C,N,V,S,H	1	
CPI	Rd,K	Compare with Immediate	Rd - K			Z,C,N,V,S,H	1	
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b) = 0) PC	←	PC + 2 or 3	None	1/2/3	
SBRS	Rr, b	Skip if Bit in Register Set	if (Rr(b) = 1) PC	←	PC + 2 or 3	None	1/2/3	
SBIC	A, b	Skip if Bit in I/O Register Cleared	if (I/O(A,b) = 0) PC	←	PC + 2 or 3	None	1/2/3	2/3/4
SBIS	A, b	Skip if Bit in I/O Register Set	If (I/O(A,b) =1) PC	←	PC + 2 or 3	None	1/2/3	2/3/4
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC	←	PC + k + 1	None	1/2	
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC	←	PC + k + 1	None	1/2	
BREQ	k	Branch if Equal	if (Z = 1) then PC	←	PC + k + 1	None	1/2	
BRNE	k	Branch if Not Equal	if (Z = 0) then PC	←	PC + k + 1	None	1/2	
BRCS	k	Branch if Carry Set	if (C = 1) then PC	←	PC + k + 1	None	1/2	
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC	←	PC + k + 1	None	1/2	
BRSH	k	Branch if Same or Higher	if (C = 0) then PC	←	PC + k + 1	None	1/2	
BRLO	k	Branch if Lower	if (C = 1) then PC	←	PC + k + 1	None	1/2	
BRMI	k	Branch if Minus	if (N = 1) then PC	←	PC + k + 1	None	1/2	
BRPL	k	Branch if Plus	if (N = 0) then PC	←	PC + k + 1	None	1/2	
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC	←	PC + k + 1	None	1/2	
BRLT	k	Branch if Less Than, Signed	if (N ⊕ V= 1) then PC	←	PC + k + 1	None	1/2	
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC	←	PC + k + 1	None	1/2	
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC	←	PC + k + 1	None	1/2	
BRTS	k	Branch if T Flag Set	if (T = 1) then PC	←	PC + k + 1	None	1/2	
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC	←	PC + k + 1	None	1/2	
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC	←	PC + k + 1	None	1/2	
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC	←	PC + k + 1	None	1/2	
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC	←	PC + k + 1	None	1/2	
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC	←	PC + k + 1	None	1/2	
		Data Ti	ansfer Instructions					
MOV	Rd, Rr	Copy Register	Rd	←	Rr	None	1	
MOVW ⁽¹⁾	Rd, Rr	Copy Register Pair	Rd+1:Rd	←	Rr+1:Rr	None	1	
LDI	Rd, K	Load Immediate	Rd	←	к	None	1	
LDS ⁽¹⁾	Rd, k	Load Direct from data space	Rd	←	(k)	None	1(5)/2(3)	2(3)(4)
LD ⁽²⁾	Rd, X	Load Indirect	Rd	←	(X)	None	1(5)2(3)	1(3)(4)



Mnemonics	Operands	Description	Opera	ation		Flags	#Clocks	#Clocks XMEGA
LD ⁽²⁾	Rd, X+	Load Indirect and Post-Increment	Rd X	←	(X) X + 1	None	2(3)	1(3)(4)
LD ⁽²⁾	Rd, -X	Load Indirect and Pre-Decrement	$X \leftarrow X - 1$, $Rd \leftarrow (X)$	←	X - 1 (X)	None	2(3)/3(5)	2(3)(4)
LD ⁽²⁾	Rd, Y	Load Indirect	$Rd \leftarrow (Y)$	←	(Y)	None	1(5)/2(3)	1(3)(4)
LD ⁽²⁾	Rd, Y+	Load Indirect and Post-Increment	Rd Y	←	(Y) Y + 1	None	2 ⁽³⁾	1(3)(4)
LD ⁽²⁾	Rd, -Y	Load Indirect and Pre-Decrement	Y Rd	←	Y - 1 (Y)	None	2(3)/3(5)	2(3)(4)
LDD ⁽¹⁾	Rd, Y+q	Load Indirect with Displacement	Rd	←	(Y + q)	None	2 ⁽³⁾	2(3)(4)
LD ⁽²⁾	Rd, Z	Load Indirect	Rd	←	(Z)	None	1 ⁽⁵⁾ /2 ⁽³⁾	1(3)(4)
LD ⁽²⁾	Rd, Z+	Load Indirect and Post-Increment	Rd Z	←	(Z), Z+1	None	2 ⁽³⁾	1 ⁽³⁾⁽⁴⁾
LD ⁽²⁾	Rd, -Z	Load Indirect and Pre-Decrement	Z Rd	←	Z - 1, (Z)	None	2 ⁽³⁾ /3 ⁽⁵⁾	2 ⁽³⁾⁽⁴⁾
LDD ⁽¹⁾	Rd, Z+q	Load Indirect with Displacement	Rd	←	(Z + q)	None	2 ⁽³⁾	2(3)(4)
STS ⁽¹⁾	k, Rr	Store Direct to Data Space	(k)	←	Rd	None	1(5)/2(3)	2 ⁽³⁾
ST ⁽²⁾	X, Rr	Store Indirect	(X)	←	Rr	None	1 ⁽⁵⁾ /2 ⁽³⁾	1 ⁽³⁾
ST ⁽²⁾	X+, Rr	Store Indirect and Post-Increment	(X) X	←	Rr, X + 1	None	1 ⁽⁵⁾ /2 ⁽³⁾	1 ⁽³⁾
ST ⁽²⁾	-X, Rr	Store Indirect and Pre-Decrement	X (X)	←	X - 1, Rr	None	2 ⁽³⁾	2(3)
ST ⁽²⁾	Y, Rr	Store Indirect	(Y)	←	Rr	None	1(5)/2(3)	1 ⁽³⁾
ST ⁽²⁾	Y+, Rr	Store Indirect and Post-Increment	(4)	←	Rr, Y + 1	None	1 ⁽⁵⁾ /2 ⁽³⁾	1 ⁽³⁾
ST ⁽²⁾	-Y, Rr	Store Indirect and Pre-Decrement	Y (Y)	←	Y - 1, Rr	None	2(3)	2(3)
STD ⁽¹⁾	Y+q, Rr	Store Indirect with Displacement	(Y + q)	←	Rr	None	2(3)	2(3)
ST ⁽²⁾	Z, Rr	Store Indirect	(Z)	←	Rr	None	1(5)/2(3)	1(3)
ST ⁽²⁾	Z+, Rr	Store Indirect and Post-Increment	(Z) Z	←	Rr Z + 1	None	1 ⁽⁵⁾ /2 ⁽³⁾	1 ⁽³⁾
ST ⁽²⁾	-Z, Rr	Store Indirect and Pre-Decrement	Z	←	Z - 1	None	2 ⁽³⁾	2 ⁽³⁾
STD ⁽¹⁾	Z+q,Rr	Store Indirect with Displacement	(Z + q)	←	Rr	None	2(3)	2(3)
LPM ⁽¹⁾⁽²⁾		Load Program Memory	R0	←	(Z)	None	3	3
LPM ⁽¹⁾⁽²⁾	Rd, Z	Load Program Memory	Rd	←	(Z)	None	3	3
LPM ⁽¹⁾⁽²⁾	Rd, Z+	Load Program Memory and Post- Increment	Rd Z	←	(Z), Z + 1	None	3	3
ELPM ⁽¹⁾		Extended Load Program Memory	R0	←	(RAMPZ:Z)	None	3	
ELPM ⁽¹⁾	Rd, Z	Extended Load Program Memory	Rd	←	(RAMPZ:Z)	None	3	
ELPM ⁽¹⁾	Rd, Z+	Extended Load Program Memory and Post-Increment	Rd Z	←	(RAMPZ:Z), Z + 1	None	3	
SPM ⁽¹⁾		Store Program Memory	(RAMPZ:Z)	←	R1:R0	None	-	-
SPM ⁽¹⁾	Z+	Store Program Memory and Post- Increment by 2	(RAMPZ:Z) Z	←	R1:R0, Z+2	None	() 	
IN	Rd, A	In From I/O Location	Rd	←	I/O(A)	None	1	
OUT	A, Rr	Out To I/O Location	I/O(A)	←	Rr	None	1	
PUSH ⁽¹⁾	Rr	Push Register on Stack	STACK	←	Rr	None	2	1 ⁽³⁾
POP ⁽¹⁾	Rd	Pop Register from Stack	Rd	←	STACK	None	2	2(3)







Mnemonics	Operands	Description	Opera	ition		Flags	#Clocks	#Clocks
XCH	Z, Rd	Exchange	(Z) Rd	←	Rd, (Z)	None	1	
LAS	Z, Rd	Load and Set	(Z) Rd	←	Rd v (Z) (Z)	None	1	
LAC	Z, Rd	Load and Clear	(Z) Rd	←	(\$FF - Rd) • (Z) (Z)	None	1	
LAT	Z, Rd	Load and Toggle	(Z) Rd	←	Rd ⊕ (Z) (Z)	None	1	
		В	it and Bit-test Instructions					
LSL	Rd	Logical Shift Left	Rd(n+1) Rd(0) C	← ←	Rd(n), 0, Rd(7)	Z,C,N,V,H	1	
LSR	Rd	Logical Shift Right	Rd(n) Rd(7) C	← ←	Rd(n+1), 0, Rd(0)	Z,C,N,V	1	
ROL	Rd	Rotate Left Through Carry	Rd(0) Rd(n+1) C	← ←	C, Rd(n), Rd(7)	Z,C,N,V,H	1	
ROR	Rd	Rotate Right Through Carry	Rd(7) Rd(n) C	← ←	C, Rd(n+1), Rd(0)	Z,C,N,V	1	
ASR	Rd	Arithmetic Shift Right	Rd(n)	←	Rd(n+1), n=06	Z,C,N,V	1	
SWAP	Rd	Swap Nibbles	Rd(30)	\leftrightarrow	Rd(74)	None	1	
BSET	s	Flag Set	SREG(s)	←	1	SREG(s)	1	
BCLR	s	Flag Clear	SREG(s)	←	0	SREG(s)	1	
SBI	A, b	Set Bit in I/O Register	I/O(A, b)	←	1	None	1 ⁽⁵⁾ 2	1
СВІ	A, b	Clear Bit in I/O Register	I/O(A, b)	←	0	None	1 ⁽⁵⁾ /2	1
BST	Rr, b	Bit Store from Register to T	Т	←	Rr(b)	Т	1	
BLD	Rd, b	Bit load from T to Register	Rd(b)	←	Т	None	1	
SEC		Set Carry	С	←	1	С	1	
CLC		Clear Carry	С	←	0	С	1	
SEN		Set Negative Flag	N	←	1	N	1	
CLN		Clear Negative Flag	N	←	0	N	1	
SEZ		Set Zero Flag	Z	←	1	Z	1	
CLZ		Clear Zero Flag	Z	←	0	Z	1	
SEI		Global Interrupt Enable	1	←	1	1	1	
CLI		Global Interrupt Disable	1	←	0	1	1	
SES		Set Signed Test Flag	S	←	1	S	1	
CLS		Clear Signed Test Flag	S	←	0	S	1	
SEV		Set Two's Complement Overflow	V	←	1	V	1	
CLV		Clear Two's Complement Overflow	V	←	0	V	1	
SET		Set T in SREG	Т	←	1	т	1	
CLT		Clear T in SREG	Т	←	0	Т	1	
SEH		Set Half Carry Flag in SREG	Н	←	1	н	1	
CLH		Clear Half Carry Flag in SREG	Н	←	0	н	1	
	E	MCU C	Control Instructions				-000	
BREAK ⁽¹⁾		Break	(See specific de	scr. fo	r BREAK)	None	1	



Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks XMEGA
NOP		No Operation		None	1	
SLEEP		Sleep	(see specific descr. for Sleep)	None	1	
WDR		Watchdog Reset	(see specific descr. for WDR)	None	1	

- Notes: 1. This instruction is not available in all devices. Refer to the device specific instruction set summary.
 - 2. Not all variants of this instruction are available in all devices. Refer to the device specific instruction set summary.
 - 3. Cycle times for Data memory accesses assume internal memory accesses, and are not valid for accesses via the external
 - 4. One extra cycle must be added when accessing Internal SRAM.
 - 5. Number of clock cycles for Reduced Core tinyAVR.







ADC - Add with Carry

Description:

(i)

Adds two registers and the contents of the C Flag and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd + Rr + C$

Syntax:Operands:Program Counter:ADC Rd,Rr $0 \le d \le 31, 0 \le r \le 31$ $PC \leftarrow PC + 1$

16-bit Opcode:

0001	11rd	dddd	rrrr

Status Register (SREG) Boolean Formula:

1	Т	Н	s	V	N	Z	С	
10-11	-	\Leftrightarrow	\Leftrightarrow	⇔	⇔	⇔	\Leftrightarrow	1

H: Rd3•Rr3+Rr3•R3•Rd3
Set if there was a carry from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: Rd7•Rr7•R7+Rd7•Rr7•R7

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7• R6 •R5• R4 •R3 •R2 •R1 •R0
Set if the result is \$00; cleared otherwise.

C: Rd7•Rr7+Rr7•R7+R7•Rd7

Set if there was carry from the MSB of the result; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

; Add R1:R0 to R3:R2

add r2,r0 ; Add low byte

adc r3,r1 ; Add with carry high byte

Words: 1 (2 bytes)

Cycles: 1

16 AVR Instruction Set



ADD - Add without Carry

Description:

Adds two registers without the C Flag and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd + Rr$

Syntax: Operands: Program Counter: (i) ADD Rd,Rr $0 \le d \le 31, 0 \le r \le 31$ PC \leftarrow PC + 1

16-bit Opcode:

		2222	201002223333
0000	11rd	dddd	rrrr

Status Register (SREG) and Boolean Formula:

1	Т	н	S	V	N	Z	С
1::	-	⇔	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow

H: Rd3•Rr3+Rr3•R3+R3•Rd3

Set if there was a carry from bit 3; cleared otherwise

- S: $N \oplus V$, For signed tests.
- V: Rd7•Rr7•R7+Rd7•Rr7•R7

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R

Set if MSB of the result is set; cleared otherwise.

Z: R7• R6 •R5• R4 •R3 •R2 •R1 •R0

Set if the result is \$00; cleared otherwise.

C: Rd7 •Rr7 +Rr7 •R7+ R7 •Rd7

Set if there was carry from the MSB of the result; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

add r1,r2 ; Add r2 to r1 (r1=r1+r2) add r28,r28 ; Add r28 to itself (r28=r28+r28)

Words: 1 (2 bytes)

Cycles: 1







ADIW - Add Immediate to Word

Description:

(i)

Adds an immediate value (0 - 63) to a register pair and places the result in the register pair. This instruction operates on the upper four register pairs, and is well suited for operations on the pointer registers.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) $Rd+1:Rd \leftarrow Rd+1:Rd + K$

> Syntax: Operands: **Program Counter:** ADIW Rd+1:Rd,K $d \in \{24,26,28,30\}, 0 \le K \le 63$

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001	0110	KKdd	KKKK
------	------	------	------

Status Register (SREG) and Boolean Formula:

I	Т	н	s	V	N	Z	С
-	1.—1	1 -	\Leftrightarrow	⇔	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow

S: N ⊕ V, For signed tests.

V: Rdh7 • R15

Set if two's complement overflow resulted from the operation; cleared otherwise.

N:

Set if MSB of the result is set; cleared otherwise.

- R15 •R14 •R13 •R12 •R11 •R10 •R9 •R8 •R7 R6 R5 R4 R3 R2 •R1 R0 Z: Set if the result is \$0000; cleared otherwise.
- C: R15 • Rdh7 Set if there was carry from the MSB of the result; cleared otherwise.

R (Result) equals Rdh:Rdl after the operation (Rdh7-Rdh0 = R15-R8, Rdl7-Rdl0=R7-R0).

Example:

```
adiw r25:24,1 ; Add 1 to r25:r24
adiw ZH:ZL,63 ; Add 63 to the Z-pointer(r31:r30)
```

Words: 1 (2 bytes) Cycles: 2

AVR Instruction Set 18



AND - Logical AND

Description:

Performs the logical AND between the contents of register Rd and register Rr and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd \bullet Rr$

Syntax: Operands:

Program Counter:

(i) AND Rd,Rr $0 \le d \le 31, 0 \le r \le 31$

 $PC \leftarrow PC + 1$

16-bit Opcode:

0010	00rd	dddd	rrrr
71.000 vit.7532000	100000000000000000000000000000000000000	110000000000000000000000000000000000000	

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С	
_	·	-	⇔	0	\Leftrightarrow	⇔	-	7

S: $N \oplus V$, For signed tests.

V: 0

Cleared

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7 •R6 •R5 •R4 •R3 • R2 •R1 •R0
Set if the result is \$00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

and r2,r3 ; Bitwise and r2 and r3, result in r2
ldi r16,1 ; Set bitmask 0000 0001 in r16
and r2,r16 ; Isolate bit 0 in r2

Words: 1 (2 bytes)

Cycles: 1

AIMEL





ANDI - Logical AND with Immediate

Description:

(i)

Performs the logical AND between the contents of register Rd and a constant and places the result in the destination register Rd.

(i) $Rd \leftarrow Rd \bullet K$

Syntax:

Operands: ANDI Rd,K $16 \le d \le 31, \ 0 \le K \le 255$ **Program Counter:**

 $PC \leftarrow PC + 1$

16-bit Opcode:

0111		2222	
0111	KKKK	dddd	KKKK

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С	
0-0	_	-	\Leftrightarrow	0	\Leftrightarrow	⇔	2:	7

S: N ⊕ V, For signed tests.

V:

Cleared

N:

Set if MSB of the result is set; cleared otherwise.

R7 •R6• R5•R4 •R3• R2• R1• R0 Z: Set if the result is \$00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

andi r17,\$0F ; Clear upper nibble of r17 andi r18,\$10 ; Isolate bit 4 in r18 andi r19,\$AA ; Clear odd bits of r19

Words: 1 (2 bytes)

Cycles: 1

AVR Instruction Set = 20



ASR – Arithmetic Shift Right

Description:

Shifts all bits in Rd one place to the right. Bit 7 is held constant. Bit 0 is loaded into the C Flag of the SREG. This operation effectively divides a signed value by two without changing its sign. The Carry Flag can be used to round the result.

Operation:



 $\begin{tabular}{lll} \mbox{Syntax:} & \mbox{Operands:} \\ \mbox{(i)} & \mbox{ASR Rd} & \mbox{0} \le d \le 31 \\ \end{tabular}$

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001	010d	dddd	0101
		•	

Status Register (SREG) and Boolean Formula:

ı	Т	н	S	V	N	Z	С	
-	-	-	⇔	⇔	⇔	\Leftrightarrow	\Leftrightarrow]

S: $N \oplus V$, For signed tests.

V: N ⊕ C (For N and C after the shift)

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7 •R6 •R5• R4 •R3 •R2• R1• R0
Set if the result is \$00; cleared otherwise.

C: Rd0

Set if, before the shift, the LSB of Rd was set; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

ldi r16,\$10 ; Load decimal 16 into r16
asr r16 ; r16=r16 / 2
ldi r17,\$FC ; Load -4 in r17
asr r17 ; r17=r17/2

Words: 1 (2 bytes)

Cycles: 1



21





BCLR - Bit Clear in SREG

Description:

(i)

Clears a single Flag in SREG.

Operation:

(i) SREG(s) $\leftarrow 0$

Syntax:Operands:BCLR s $0 \le s \le 7$

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001	0100	1sss	1000
------	------	------	------

Status Register (SREG) and Boolean Formula:

1	Т	н	S	V	N	Z	С	
\Leftrightarrow	⇔	\Leftrightarrow	\Leftrightarrow	⇔	⇔	\Leftrightarrow	\Leftrightarrow]

I: 0 if s = 7; Unchanged otherwise.

T: 0 if s = 6; Unchanged otherwise.

H: 0 if s = 5; Unchanged otherwise.

S: 0 if s = 4; Unchanged otherwise.

V: 0 if s = 3; Unchanged otherwise.

N: 0 if s = 2; Unchanged otherwise.

Z: 0 if s = 1; Unchanged otherwise.

C: 0 if s = 0; Unchanged otherwise.

Example:

bclr 0 ; Clear Carry Flag bclr 7 ; Disable interrupts

Words: 1 (2 bytes)

Cycles: 1

22 AVR Instruction Set



BLD - Bit Load from the T Flag in SREG to a Bit in Register

Description:

Copies the T Flag in the SREG (Status Register) to bit b in register Rd.

Operation:

(i) $Rd(b) \leftarrow T$

 $\begin{tabular}{llll} \textbf{Syntax:} & \textbf{Operands:} & \textbf{Program Counter:} \\ \textbf{(i)} & BLD \ Rd,b & 0 \le d \le 31, \ 0 \le b \le 7 & PC \leftarrow PC + 1 \\ \end{tabular}$

16 bit Opcode:

1111 100d dddd 0bbb

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С	
-	-	-	-	_	_	-	10-0]

Example:

; Copy bit

bst r1,2 ; Store bit 2 of r1 in T Flag bld r0,4 ; Load T Flag into bit 4 of r0

Words: 1 (2 bytes)

Cycles: 1







BRBC - Branch if Bit in SREG is Cleared

Description:

Conditional relative branch. Tests a single bit in SREG and branches relatively to PC if the bit is cleared. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form.

Operation:

(i) If SREG(s) = 0 then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

Syntax: Operands: Program Counter:

(i) BRBC s,k $0 \le s \le 7, -64 \le k \le +63 \\ PC \leftarrow PC + k + 1 \\ PC \leftarrow PC + 1, \text{ if condition is false}$

16-bit Opcode:

1111 01kk kkkk ksss

Status Register (SREG) and Boolean Formula:

I	T	Н	s	V	N	Z	С
10-00	3-0	1 - 1	-	8 8	-	_	_

Example:

cpi r20,5 ; Compare r20 to the value 5 brbc 1,noteq ; Branch if Zero Flag cleared

• • •

noteq:nop ; Branch destination (do nothing)

Words: 1 (2 bytes)



BRBS - Branch if Bit in SREG is Set

Description:

Conditional relative branch. Tests a single bit in SREG and branches relatively to PC if the bit is set. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form.

Operation:

(i) If SREG(s) = 1 then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

Syntax: Operands: Program Counter:

(i) BRBS s,k $0 \le s \le 7, -64 \le k \le +63 \\ PC \leftarrow PC + k + 1 \\ PC \leftarrow PC + 1, \text{ if condition is false}$

16-bit Opcode:

1111 00kk kkkk ksss

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С
-	-	-	1	_	-	_	-

Example:

bst r0,3 ; Load T bit with bit 3 of r0

brbs 6, bitset ; Branch T bit was set

• • •

bitset: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)







BRCC – Branch if Carry Cleared

Description:

Conditional relative branch. Tests the Carry Flag (C) and branches relatively to PC if C is cleared. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 0,k).

Operation:

(i) If C = 0 then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

PC ← PC + 1, if condition is false

16-bit Opcode:

1111 01kk kkkk k000

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С	
_	-	-	-	_	-	-	-	

Example:

add r22,r23 ; Add r23 to r22

brcc nocarry ; Branch if carry cleared

• • •

nocarry: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false 2 if condition is true

26



BRCS - Branch if Carry Set

Description:

Conditional relative branch. Tests the Carry Flag (C) and branches relatively to PC if C is set. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 0,k).

Operation:

(i) If C = 1 then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

PC ← PC + 1, if condition is false

16-bit Opcode:

0.01/2/2	kkkk	k000
	00kk	00kk kkkk

Status Register (SREG) and Boolean Formula:

1	Т	н	s	V	N	Z	С
-	-	-	-	-	-	-	-

Example:

cpi r26,\$56 ; Compare r26 with \$56
brcs carry ; Branch if carry set

carry: nop ; Branch des

; Branch destination (do nothing)

Words: 1 (2 bytes)







BREAK - Break

Description:

The BREAK instruction is used by the On-chip Debug system, and is normally not used in the application software. When the BREAK instruction is executed, the AVR CPU is set in the Stopped Mode. This gives the On-chip Debugger access to internal resources.

If any Lock bits are set, or either the JTAGEN or OCDEN Fuses are unprogrammed, the CPU will treat the BREAK instruction as a NOP and will not enter the Stopped mode.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) On-chip Debug system break.

16-bit Opcode:

1001	0101	1001	1000

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С
: - :	1,-2	3 — 3	:-:	_	-	-	-

Words: 1 (2 bytes)

Cycles: 1



BREQ – Branch if Equal

Description:

Conditional relative branch. Tests the Zero Flag (Z) and branches relatively to PC if Z is set. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the unsigned or signed binary number represented in Rd was equal to the unsigned or signed binary number represented in Rr. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 1,k).

Operation:

BREQ k

(i) If Rd = Rr (Z = 1) then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

 $\textbf{-64} \leq k \leq \textbf{+63}$

Syntax: Operands:

Program Counter:

PC ← PC + k + 1

 $PC \leftarrow PC + 1$, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k001

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С
-	-	-	-	-	-	-	-

Example:

(i)

cp r1,r0 ; Compare registers r1 and r0
breq equal ; Branch if registers equal

equal: nop

; Branch destination (do nothing)

Words: 1 (2 bytes)







BRGE – Branch if Greater or Equal (Signed)

Description:

Conditional relative branch. Tests the Signed Flag (S) and branches relatively to PC if S is cleared. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the signed binary number represented in Rd was greater than or equal to the signed binary number represented in Rr. This instruction branches relatively to PC in either direction (PC - $63 \le \text{destination} \le \text{PC} + 64$). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 4,k).

Operation:

(i) If $Rd \ge Rr (N \oplus V = 0)$ then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k100

Status Register (SREG) and Boolean Formula:

1	Т	н	s	V	N	Z	С
-	-	-	-	-	-	-	-

Example:

cp r11,r12 ; Compare registers r11 and r12
brge greateq ; Branch if r11 ≥ r12 (signed)
...

greateg: nop

; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false 2 if condition is true

30 AVR Instruction Set ■



BRHC - Branch if Half Carry Flag is Cleared

Description:

Conditional relative branch. Tests the Half Carry Flag (H) and branches relatively to PC if H is cleared. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 5,k).

Operation:

(i) If H = 0 then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Syntax: Operands: Program Counter: (i) BRHC k $-64 \le k \le +63$ PC \leftarrow PC + k + 1

PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k101

Status Register (SREG) and Boolean Formula:

1	Т	н	S	V	N	Z	С
-	_	_	_	-	_	-	-

Example:

brhc hclear ; Branch if Half Carry Flag cleared

. . .

hclear: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)







BRHS - Branch if Half Carry Flag is Set

Description:

Conditional relative branch. Tests the Half Carry Flag (H) and branches relatively to PC if H is set. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 5,k).

Operation:

(i) If H = 1 then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Program Counter: $PC \leftarrow PC + k + 1$

PC ← PC + 1, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k101

Status Register (SREG) and Boolean Formula:

I	Т	н	s	V	N	Z	С
_	-	-	-	-	-	-	-

Example:

brhs hset ; Branch if Half Carry Flag set

. . .

hset: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false

2 if condition is true

32



BRID – Branch if Global Interrupt is Disabled

Description:

Conditional relative branch. Tests the Global Interrupt Flag (I) and branches relatively to PC if I is cleared. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 7,k).

Operation:

(i) If I = 0 then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

Syntax:Operands:Program Counter:BRID k $-64 \le k \le +63$ PC \leftarrow PC + k + 1

PC ← PC + 1, if condition is false

16-bit Opcode:

	01kk	1111	2 0 0
1111	O L K K	kkkk	k11

Status Register (SREG) and Boolean Formula:

1	Т	н	s	V	N	Z	С
-	-	-	-	-	-	-	-

Example:

(i)

brid intdis ; Branch if interrupt disabled

...

intdis: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)







BRIE – Branch if Global Interrupt is Enabled

Description:

Conditional relative branch. Tests the Global Interrupt Flag (I) and branches relatively to PC if I is set. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 7,k).

Operation:

(i) If I = 1 then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

 $PC \leftarrow PC + 1$, if condition is false

16-bit Opcode:

	0.022	lelelele	
1111	1 00kk l	KKKK I	K11

Status Register (SREG) and Boolean Formula:

- 1	Т	н	s	V	N	Z	С
-	-	-	-	-	-	-	-

Example:

brie inten ; Branch if interrupt enabled

...

inten: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false 2 if condition is true

34 AVR Instruction Set



BRLO - Branch if Lower (Unsigned)

Description:

Conditional relative branch. Tests the Carry Flag (C) and branches relatively to PC if C is set. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the unsigned binary number represented in Rd was smaller than the unsigned binary number represented in Rr. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 0,k).

Operation:

(i) If Rd < Rr (C = 1) then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

Syntax: Operands: Program Counter: BRLO k $-64 \le k \le +63$ PC \leftarrow PC + k + 1

PC ← PC + 1, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k000
1111	OOM	TETETE TE	1000

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С	
-	_	-	_	-	-	-	-	1

Example:

(i)

eor r19,r19 ; Clear r19
loop: inc r19 ; Increase r19
...
cpi r19,\$10 ; Compare r19 with \$10
brlo loop ; Branch if r19 < \$10 (unsigned)
nop ; Exit from loop (do nothing)</pre>

Words: 1 (2 bytes)







BRLT – Branch if Less Than (Signed)

Description:

Conditional relative branch. Tests the Signed Flag (S) and branches relatively to PC if S is set. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the signed binary number represented in Rd was less than the signed binary number represented in Rr. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 4,k).

Operation:

(i) If Rd < Rr (N \oplus V = 1) then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

PC ← PC + 1, if condition is false

16-bit Opcode:

1111 00kk kkkk k100

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С	
-	-	_	-	-	-	-	: - :]

Example:

cp r16,r1 ; Compare r16 to r1

brlt less ; Branch if r16 < r1 (signed)

less: nop

; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false

2 if condition is true

36



BRMI - Branch if Minus

Description:

Conditional relative branch. Tests the Negative Flag (N) and branches relatively to PC if N is set. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 2,k).

Operation:

(i) If N = 1 then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

PC ← PC + 1, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k010

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С
: - 3	-	-	-	-	-	_	_

Example:

subi r18,4 ; Subtract 4 from r18
brmi negative ; Branch if result negative

• • •

negative: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)







BRNE – Branch if Not Equal

Description:

Conditional relative branch. Tests the Zero Flag (Z) and branches relatively to PC if Z is cleared. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the unsigned or signed binary number represented in Rd was not equal to the unsigned or signed binary number represented in Rr. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 1,k).

Operation:

(i) If Rd \neq Rr (Z = 0) then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

Syntax: Operands: BRNE k $-64 \le k \le +63$

Program Counter:

 $\begin{aligned} & PC \leftarrow PC + k + 1 \\ & PC \leftarrow PC + 1, & \text{if condition is false} \end{aligned}$

16-bit Opcode:

1111 01kk kkkk k001

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С
-	_	-	_	-	-	-	-

Example:

(i)

Words: 1 (2 bytes)

Cycles: 1 if condition is false 2 if condition is true

38 AVR Instruction Set



BRPL - Branch if Plus

Description:

Conditional relative branch. Tests the Negative Flag (N) and branches relatively to PC if N is cleared. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 2,k).

Operation:

(i) If N = 0 then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Syntax: Operands: Program Counter:

(i) BRPL k $-64 \le k \le +63 \\ PC \leftarrow PC + k + 1 \\ PC \leftarrow PC + 1, \text{ if condition is false}$

16-bit Opcode:

1111 01kk kkkk k010

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С	
_	-	-	-	_	-	-	-	

Example:

subi r26,\$50 ; Subtract \$50 from r26 brpl positive ; Branch if r26 positive

...

positive: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)







BRSH – Branch if Same or Higher (Unsigned)

Description:

Conditional relative branch. Tests the Carry Flag (C) and branches relatively to PC if C is cleared. If the instruction is executed immediately after execution of any of the instructions CP, CPI, SUB or SUBI the branch will occur if and only if the unsigned binary number represented in Rd was greater than or equal to the unsigned binary number represented in Rr. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 0,k).

Operation:

(i) If Rd \geq Rr (C = 0) then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

Syntax: Operands: Program Counter: (i) BRSH k $-64 \le k \le +63$ PC \leftarrow PC + k + 1

PC ← PC + 1, if condition is false

16-bit Opcode:

1111 01kk kkkk k000

Status Register (SREG) and Boolean Formula:

1	Т	н	s	V	N	Z	С
-	-	-	-	-	-	-	-

Example:

subi r19,4 ; Subtract 4 from r19

brsh highsm ; Branch if r19 >= 4 (unsigned)

...

highsm: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false 2 if condition is true

40 AVR Instruction Set



BRTC – Branch if the T Flag is Cleared

Description:

Conditional relative branch. Tests the T Flag and branches relatively to PC if T is cleared. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 6,k).

Operation:

If T = 0 then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1 (i)

Syntax: Operands: **Program Counter:** $-64 \le k \le +63$ $PC \leftarrow PC + k + 1$ (i) BRTC k

PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k110		

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С	
-	-	-	1	-	-	_	-	1

Example:

; Store bit 5 of r3 in T Flag ; Branch if this bit was cleared

tclear: ; Branch destination (do nothing) nop

Words: 1 (2 bytes)







BRTS – Branch if the T Flag is Set

Description:

Conditional relative branch. Tests the T Flag and branches relatively to PC if T is set. This instruction branches relatively to PC in either direction (PC - 63 ≤ destination ≤ PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 6,k).

Operation:

If T = 1 then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1 (i)

Syntax: Operands:

 $-64 \le k \le +63$ $PC \leftarrow PC + k + 1$ (i) BRTS k

PC ← PC + 1, if condition is false

Program Counter:

16-bit Opcode:

4444	00kk	bbbb	1440
1111	OUKK	KKKK	KIIO

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С
-	-	-	1	_	-	_	-

Example:

; Store bit 5 of r3 in T Flag ; Branch if this bit was set

tset: ; Branch destination (do nothing) nop

Words: 1 (2 bytes)



BRVC - Branch if Overflow Cleared

Description:

Conditional relative branch. Tests the Overflow Flag (V) and branches relatively to PC if V is cleared. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 3,k).

Operation:

(i) If V = 0 then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Syntax: Operands: BRVC k $-64 \le k \le +63$

Program Counter: $PC \leftarrow PC + k + 1$

PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k011
------	------	------	------

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С	
_	-	-	-	-	_	_	_	

Example:

(i)

add r3,r4 ; Add r4 to r3

brvc noover ; Branch if no overflow

noover: nop

; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false

2 if condition is true







BRVS - Branch if Overflow Set

Description:

Conditional relative branch. Tests the Overflow Flag (V) and branches relatively to PC if V is set. This instruction branches relatively to PC in either direction (PC - 63 ≤ destination ≤ PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 3,k).

Operation:

(i) If V = 1 then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Operands: **Program Counter:** Syntax: $PC \leftarrow PC + k + 1$ (i) BRVS k $-64 \le k \le +63$

PC ← PC + 1, if condition is false

16-bit Opcode:

00kk kkkk 1111 k011

Status Register (SREG) and Boolean Formula:

ı	Т	Н	s	V	N	Z	С
_	_	_	_	_	_	_	_

Example:

; Add r4 to r3 add r3,r4 brvs overfl ; Branch if overflow

overfl: ; Branch destination (do nothing) nop

Words: 1 (2 bytes)

Cycles: 1 if condition is false 2 if condition is true

AVR Instruction Set 44



BSET - Bit Set in SREG

Description:

(i)

Sets a single Flag or bit in SREG.

Operation:

(i) SREG(s) \leftarrow 1

Syntax: Operands: BSET s $0 \le s \le 7$

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001 0100 0sss 1000

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С
\Leftrightarrow							

I: 1 if s = 7; Unchanged otherwise.

T: 1 if s = 6; Unchanged otherwise.

H: 1 if s = 5; Unchanged otherwise.

S: 1 if s = 4; Unchanged otherwise.

V: 1 if s = 3; Unchanged otherwise.

N: 1 if s = 2; Unchanged otherwise.

Z: 1 if s = 1; Unchanged otherwise.

C: 1 if s = 0; Unchanged otherwise.

Example:

bset 6 ; Set T Flag bset 7 ; Enable interrupt

Words: 1 (2 bytes)

Cycles: 1







BST - Bit Store from Bit in Register to T Flag in SREG

Description:

Stores bit b from Rd to the T Flag in SREG (Status Register).

Operation:

(i) $T \leftarrow Rd(b)$

 $\begin{tabular}{lll} \textbf{Syntax:} & \textbf{Operands:} \\ \textbf{BST Rd,b} & 0 \le d \le 31, \ 0 \le b \le 7 \end{tabular}$

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1111 101d dddd 0bbb

Status Register (SREG) and Boolean Formula:

Ţ	T	Н	s	V	N	Z	С
.—.	\Leftrightarrow	-	-	:: - :		:: - ->	-

T: 0 if bit b in Rd is cleared. Set to 1 otherwise.

Example:

; Copy bit

bst r1,2 ; Store bit 2 of r1 in T Flag bld r0,4 ; Load T into bit 4 of r0

Words: 1 (2 bytes)

Cycles: 1

46 AVR Instruction Set



CALL – Long Call to a Subroutine

Description:

Calls to a subroutine within the entire Program memory. The return address (to the instruction after the CALL) will be stored onto the Stack. (See also RCALL). The Stack Pointer uses a post-decrement scheme during CALL.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) PC ← k
 (ii) PC ← k
 Devices with 16 bits PC, 128K bytes Program memory maximum.
 (iii) PC ← k
 Devices with 22 bits PC, 8M bytes Program memory maximum.

	Syntax:	Operands:	Program Counter	Stack:
(i)	CALL k	$0 \leq k < 64K$	PC ← k	STACK ← PC+2 SP ← SP-2, (2 bytes, 16 bits)
(ii)	CALL k	$0 \leq k < 4M$	$PC \leftarrow k$	STACK ← PC+2 SP ← SP-3 (3 bytes, 22 bits)

32-bit Opcode:

1001	010k	kkkk	111k	
kkkk	kkkk	kkkk	kkkk	

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	٧	N	Z	С
-	-	-	-	-	-	-	-

Example:

r16,r0 ; Copy r0 to r16 mov call check ; Call subroutine ; Continue (do nothing) nop check: cpi r16,\$42 ; Check if r16 has a special value breq error ; Branch if equal ; Return from subroutine error: rjmp ; Infinite loop

Words : 2 (4 bytes)

Cycles : 4, devices with 16 bit PC

5, devices with 22 bit PC

Cycles XMEGA: 3, devices with 16 bit PC

4, devices with 22 bit PC







CBI - Clear Bit in I/O Register

Description:

Clears a specified bit in an I/O Register. This instruction operates on the lower 32 I/O Registers - addresses 0-31.

Operation:

(i) $I/O(A,b) \leftarrow 0$

 Syntax:
 Operands:

 CBI A,b
 $0 \le A \le 31, \ 0 \le b \le 7$

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001 1000 AAAA Abbb

Status Register (SREG) and Boolean Formula:

Ţ	T	Н	s	V	N	Z	С
-	n - n	s - -		:: - :		:	-

Example:

cbi \$12,7 ; Clear bit 7 in Port D

Words : 1 (2 bytes)

Cycles : 2

Cycles XMEGA: 1
Cycles Reduced Core tinyAVR: 1

48



CBR - Clear Bits in Register

Description:

Clears the specified bits in register Rd. Performs the logical AND between the contents of register Rd and the complement of the constant mask K. The result will be placed in register Rd.

Operation:

(i) $Rd \leftarrow Rd \bullet (\$FF - K)$

 $\begin{array}{ccc} \text{Syntax:} & \text{Operands:} & \text{Program Counter:} \\ \text{(i)} & \text{CBR Rd,K} & 16 \leq d \leq 31, \, 0 \leq K \leq 255 & \text{PC} \leftarrow \text{PC} + 1 \\ \end{array}$

16-bit Opcode: (see ANDI with K complemented)

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С	
-	-	_	⇔	0	⇔	⇔	_]

S: $N \oplus V$, For signed tests.

V: 0

Cleared

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7 •R6 •R5• R4• R3 •R2• R1• R0
Set if the result is \$00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

cbr r16,\$F0 ; Clear upper nibble of r16 cbr r18,1 ; Clear bit 0 in r18

Words: 1 (2 bytes)







CLC – Clear Carry Flag

Description:

Clears the Carry Flag (C) in SREG (Status Register).

Operation:

(i) $C \leftarrow 0$

(i)

Syntax: Operands: CLC None

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001 0100 1000 1000	1001	0100	1000	1000
---------------------	------	------	------	------

Status Register (SREG) and Boolean Formula:

1	T	Н	S	V	N	Z	С
-	-	_	_	_	8-1	-	0

C: 0

Carry Flag cleared

Example:

add r0,r0 ; Add r0 to itself clc ; Clear Carry Flag

Words: 1 (2 bytes)



CLH - Clear Half Carry Flag

Description:

Clears the Half Carry Flag (H) in SREG (Status Register).

Operation:

(i) $H \leftarrow 0$

16-bit Opcode:

1001 0100 1101 1000

Status Register (SREG) and Boolean Formula:

- 1	T	н	S	V	N	Z	С	
_	-	0	-	-	-	j=.	(3 —)]

H: (

Half Carry Flag cleared

Example:

clh ; Clear the Half Carry Flag

Words: 1 (2 bytes)







CLI - Clear Global Interrupt Flag

Description:

Clears the Global Interrupt Flag (I) in SREG (Status Register). The interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction.

Operation:

(i) I ← 0

16-bit Opcode:

1001	0100	1111	1000

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С
0	-	-	-	-	-	-	-

I: 0
Global Interrupt Flag cleared

Example:

in temp, SREG ; Store SREG value (temp must be defined by user)
cli ; Disable interrupts during timed sequence
sbi EECR, EEMWE; Start EEPROM write
sbi EECR, EEWE
out SREG, temp ; Restore SREG value (I-Flag)

Words: 1 (2 bytes)

Cycles: 1

52 AVR Instruction Set



CLN - Clear Negative Flag

Description:

Clears the Negative Flag (N) in SREG (Status Register).

Operation:

(i) $N \leftarrow 0$

16-bit Opcode:

1001 0100 1010 1000

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С
=	-	-	-	-	0	-	-

N: (

Negative Flag cleared

Example:

add r2,r3 ; Add r3 to r2 cln ; Clear Negative Flag

Words: 1 (2 bytes)







CLR - Clear Register

Description:

Clears a register. This instruction performs an Exclusive OR between a register and itself. This will clear all bits in the register.

Operation:

(i) $Rd \leftarrow Rd \oplus Rd$

16-bit Opcode: (see EOR Rd,Rd)

0010	01dd	dddd	dddd
------	------	------	------

Status Register (SREG) and Boolean Formula:

1	Т	н	S	V	N	Z	С
-	-	-	0	0	0	1	-

S: 0

Cleared

V: 0

Cleared

N: C

Cleared

Z: 1

Set

R (Result) equals Rd after the operation.

Example:

```
clr r18 ; clear r18
loop: inc r18 ; increase r18
...
cpi r18,$50 ; Compare r18 to $50
brne loop
```

Words: 1 (2 bytes)

Cycles: 1

54 AVR Instruction Set



CLS - Clear Signed Flag

Description:

Clears the Signed Flag (S) in SREG (Status Register).

Operation:

(i) $S \leftarrow 0$

(i)

Syntax:Operands:Program Counter:CLSNone $PC \leftarrow PC + 1$

16-bit Opcode:

	1001	0100	1100	100
--	------	------	------	-----

Status Register (SREG) and Boolean Formula:

ı	Т	н	s	V	N	Z	С
=	-	-	0	-	-	-	-

S: 0 Signed Flag cleared

Example:

add r2,r3 ; Add r3 to r2 cls ; Clear Signed Flag

Words: 1 (2 bytes)
Cycles: 1





CLT - Clear T Flag

Description:

Clears the T Flag in SREG (Status Register).

Operation:

(i) $T \leftarrow 0$

(i)

Syntax: Operands: CLT None

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001 0100 1110 1000

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С
-	0	-	-	-	-	=	-

T: 0

T Flag cleared

Example:

clt ; Clear T Flag

Words: 1 (2 bytes)

Cycles: 1

AVR Instruction Set

0856I-AVR-07/10

56



CLV - Clear Overflow Flag

Description:

Clears the Overflow Flag (V) in SREG (Status Register).

Operation:

(i) $V \leftarrow 0$

(i)

Syntax:Operands:Program Counter:CLVNone $PC \leftarrow PC + 1$

16-bit Opcode:

1001	0100	1011	1000

Status Register (SREG) and Boolean Formula:

I	Т	н	s	V	N	Z	С	
-	-	-	-	0	-	-	-	

/: (

Overflow Flag cleared

Example:

add r2,r3 ; Add r3 to r2 clv ; Clear Overflow Flag

Words: 1 (2 bytes)





CLZ - Clear Zero Flag

Description:

Clears the Zero Flag (Z) in SREG (Status Register).

Operation:

(i) $Z \leftarrow 0$

16-bit Opcode:

1001 0100 1001 1000

Status Register (SREG) and Boolean Formula:

1	Т	н	s	V	N	Z	С	
=	-	-	-	-	-	0	_	7

Z: 0 Zero Flag cleared

Example:

add r2,r3 ; Add r3 to r2 clz ; Clear zero

Words: 1 (2 bytes)

Cycles: 1

58 AVR Instruction Set



COM - One's Complement

Description:

This instruction performs a One's Complement of register Rd.

Operation:

(i) $Rd \leftarrow \$FF - Rd$

	Syntax:	Operands:	Program Counter:
(i)	COM Rd	$0 \leq d \leq 31$	PC ← PC + 1

16-bit Opcode:

1001	010d	dddd	0000

Status Register (SREG) and Boolean Formula:

1	Т	н	S	V	N	Z	С	
=	-	-	⇔	0	⇔	⇔	1	1

S: $N \oplus V$

For signed tests.

V: 0

Cleared.

N: R

Set if MSB of the result is set; cleared otherwise.

Z: R7 •R6• R5• R4 •R3 •R2• R1 •R0
Set if the result is \$00; Cleared otherwise.

C: 1 Set.

R (Result) equals Rd after the operation.

Example:

```
com r4 ; Take one's complement of r4
breq zero ; Branch if zero
...
zero: nop ; Branch destination (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1







CP - Compare

Description:

This instruction performs a compare between two registers Rd and Rr. None of the registers are changed. All conditional branches can be used after this instruction.

Operation:

(i) Rd - Rr

16-bit Opcode:

0001	01rd	dddd	rrrr
------	------	------	------

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С	
n = //	-	⇔	\Leftrightarrow	⇔	⇔	⇔	\Leftrightarrow]

H: Rd3 •Rr3+ Rr3 •R3 +R3• Rd3

Set if there was a borrow from bit 3; cleared otherwise

- S: $N \oplus V$, For signed tests.
- V: Rd7• Rr7 •R7+ Rd7 •Rr7 •R7

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7

Set if MSB of the result is set; cleared otherwise.

- Z: R7• R6 •R5• R4 •R3 •R2 •R1 •R0
- Set if the result is \$00; cleared otherwise.
- C: Rd7 •Rr7+ Rr7• R7 +R7• Rd7
 Set if the absolute value of the contents of Rr is larger than the absolute value of Rd; cleared otherwise.

R (Result) after the operation.

Example:

```
cp r4,r19 ; Compare r4 with r19
brne noteq ; Branch if r4 <> r19
...
noteq: nop ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1

60 AVR Instruction Set



CPC – Compare with Carry

Description:

This instruction performs a compare between two registers Rd and Rr and also takes into account the previous carry. None of the registers are changed. All conditional branches can be used after this instruction.

Operation:

(i) Rd - Rr - C

16-bit Opcode:

0000	01rd	dddd	rrrr
------	------	------	------

Status Register (SREG) and Boolean Formula:

I	T	Н	s	V	N	Z	С
-	-	⇔	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	⇔

H: Rd3 •Rr3+ Rr3 •R3 +R3 •Rd3

Set if there was a borrow from bit 3; cleared otherwise

- S: $N \oplus V$, For signed tests.
- V: Rd7 •Rr7 R7+ Rd7 Rr7 •R7

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7 •R6• R5• R4 •R3 •R2 •R1• R0 •Z

Previous value remains unchanged when the result is zero; cleared otherwise.

C: Rd7 •Rr7+ Rr7• R7 +R7 •Rd7

Set if the absolute value of the contents of Rr plus previous carry is larger than the absolute value of Rd; cleared otherwise.

R (Result) after the operation.

Example:

```
; Compare r3:r2 with r1:r0

cp r2,r0 ; Compare low byte

cpc r3,r1 ; Compare high byte

brne noteq ; Branch if not equal

...

noteq: nop ; Branch destination (do nothing)
```



61





Words: 1 (2 bytes)

Cycles: 1

62 AVR Instruction Set



CPI – Compare with Immediate

Description:

This instruction performs a compare between register Rd and a constant. The register is not changed. All conditional branches can be used after this instruction.

Operation:

(i) Rd - K

	Syntax:	Operands:	Program Counter:
(i)	CPI Rd,K	$16 \leq d \leq 31,~0 \leq K \leq 255$	PC ← PC + 1

16-bit Opcode:

0011	KKKK	dddd	KKKK
1. 000000000000000000000000000000000000	COSTO COSTO DE LA COSTO DEL COSTO DE LA COSTO DE LA COSTO DEL COSTO DE LA COSTO DEL COSTO DELA COSTO DE LA COSTO DEL COSTO DE LA COSTO DEL COSTO DEL COSTO DEL COSTO DE LA COS	200000000000000000000000000000000000000	200111111111111111111111111111111111111

Status Register (SREG) and Boolean Formula:

1	Т	н	s	V	N	Z	С
-	-	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow

H: Rd3 •K3+ K3• R3+ R3 •Rd3

Set if there was a borrow from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: Rd7 •K7 •R7 +Rd7 •K7 •R7

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7 •R6• R5 •R4• R3• R2 •R1 •R0

Set if the result is \$00; cleared otherwise.

C: Rd7 •K7 +K7 •R7+ R7 •Rd7

Set if the absolute value of K is larger than the absolute value of Rd; cleared otherwise.

R (Result) after the operation.

Example:

```
cpi r19,3 ; Compare r19 with 3
brne error ; Branch if r19<>3
...
error: nop ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1



63





CPSE – Compare Skip if Equal

Description:

This instruction performs a compare between two registers Rd and Rr, and skips the next instruction if Rd = Rr.

Operation:

(i) If Rd = Rr then PC \leftarrow PC + 2 (or 3) else PC \leftarrow PC + 1

> Syntax: Operands:

Program Counter: CPSE Rd,Rr $0 \leq d \leq 31, \ 0 \leq r \leq 31$ PC ← PC + 1, Condition false - no skip (i)

PC ← PC + 2, Skip a one word instruction

PC ← PC + 3, Skip a two word instruction

16-bit Opcode:

0001 00	rd dddd	rrrr
---------	---------	------

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	z	С
-	_	=	=	=	=	-	-

Example:

r4 inc ; Increase r4 r4,r0 ; Compare r4 to r0 cpse r4 ; Only executed if r4<>r0 neg ; Continue (do nothing) nop

Words: 1 (2 bytes)

Cycles: 1 if condition is false (no skip)

2 if condition is true (skip is executed) and the instruction skipped is 1 word 3 if condition is true (skip is executed) and the instruction skipped is 2 words



DEC - Decrement

Description:

Subtracts one -1- from the contents of register Rd and places the result in the destination register Rd.

The C Flag in SREG is not affected by the operation, thus allowing the DEC instruction to be used on a loop counter in multiple-precision computations.

When operating on unsigned values, only BREQ and BRNE branches can be expected to perform consistently. When operating on two's complement values, all signed branches are available.

Operation:

(i) Rd ← Rd - 1

16-bit Opcode:

1001	010d	dddd	1010
------	------	------	------

Status Register and Boolean Formula:

ı	Т	Н	s	V	N	Z	С	
-	-	-	⇔	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow		

S: $N \oplus V$

For signed tests.

V: R7 •R6 •R5 •R4• R3• R2 •R1• R0

Set if two's complement overflow resulted from the operation; cleared otherwise. Two's complement overflow occurs if and only if Rd was \$80 before the operation.

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7 •R6• R5 •R4• R3• R2• R1• R0
Set if the result is \$00; Cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
ldi r17,$10 ; Load constant in r17
loop: add r1,r2 ; Add r2 to r1
dec r17 ; Decrement r17
brne loop ; Branch if r17<>0
nop ; Continue (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1



65





DES - Data Encryption Standard

Description:

The module is an instruction set extension to the AVR CPU, performing DES iterations. The 64-bit data block (plaintext or ciphertext) is placed in the CPU register file, registers R0-R7, where LSB of data is placed in LSB of R0 and MSB of data is placed in MSB of R7. The full 64-bit key (including parity bits) is placed in registers R8-R15, organized in the register file with LSB of key in LSB of R8 and MSB of key in MSB of R15. Executing one DES instruction performs one round in the DES algorithm. Sixteen rounds must be executed in increasing order to form the correct DES ciphertext or plaintext. Intermediate results are stored in the register file (R0-R15) after each DES instruction. The instruction's operand (K) determines which round is executed, and the half carry flag (H) determines whether encryption or decryption is performed.

The DES algorithm is described in "Specifications for the Data Encryption Standard" (Federal Information Processing Standards Publication 46). Intermediate results in this implementation differ from the standard because the initial permutation and the inverse initial permutation are performed each iteration. This does not affect the result in the final ciphertext or plaintext, but reduces execution time.

Operation:

(i) If H = 0 then Encrypt round (R7-R0, R15-R8, K)
If H = 1 then Decrypt round (R7-R0, R15-R8, K)

Syntax: Operands: Program Counter:

(i) DES K $0x00 \le K \le 0x0F$ PC \leftarrow PC + 1

16-bit Opcode:

	- 4		
1001	0100	KKKK	1011

Example:

DES 0x00
DES 0x01
...
DES 0x0E
DES 0x0F

Words: 1 Cycles: 1 (2⁽¹⁾)

Note: 1. If the DES instruction is succeeding a non-DES instruction, an extra cycle is inserted.

AVR Instruction Set

0856I-AVR-07/10

66



EICALL – Extended Indirect Call to Subroutine

Description:

Indirect call of a subroutine pointed to by the Z (16 bits) Pointer Register in the Register File and the EIND Register in the I/O space. This instruction allows for indirect calls to the entire 4M (words) Program memory space. See also ICALL. The Stack Pointer uses a post-decrement scheme during EICALL.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) $PC(15:0) \leftarrow Z(15:0)$ $PC(21:16) \leftarrow EIND$

Syntax: Operands: Program Counter: Stack:

(i) EICALL None See Operation STACK \leftarrow PC + 1

 $SP \leftarrow SP - 3$ (3 bytes, 22 bits)

16-bit Opcode:

1001	0101	0001	1001
------	------	------	------

Status Register (SREG) and Boolean Formula:

1	т	н	S	V	N	Z	С
i —	-	_	-	-	-	_	23-0

Example:

Words: 1 (2 bytes)

Cycles: 4 (only implemented in devices with 22 bit PC)
Cycles XMEGA: 3 (only implemented in devices with 22 bit PC)







EIJMP - Extended Indirect Jump

Description:

Indirect jump to the address pointed to by the Z (16 bits) Pointer Register in the Register File and the EIND Register in the I/O space. This instruction allows for indirect jumps to the entire 4M (words) Program memory space. See also IJMP.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) PC(15:0) ← Z(15:0) PC(21:16) ← EIND

Syntax: Operands: Program Counter: Stack:
(i) EIJMP None See Operation Not Affected

16-bit Opcode:

1001	0100	0001	1001
------	------	------	------

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С
-	_	-	-	-	-	-	-

Example:

Words: 1 (2 bytes)

Cycles: 2

68



ELPM – Extended Load Program Memory

Description:

Loads one byte pointed to by the Z-register and the RAMPZ Register in the I/O space, and places this byte in the destination register Rd. This instruction features a 100% space effective constant initialization or constant data fetch. The Program memory is organized in 16-bit words while the Z-pointer is a byte address. Thus, the least significant bit of the Z-pointer selects either low byte $(Z_{LSB} = 0)$ or high byte $(Z_{LSB} = 1)$. This instruction can address the entire Program memory space. The Z-pointer Register can either be left unchanged by the operation, or it can be incremented. The incrementation applies to the entire 24-bit concatenation of the RAMPZ and Z-pointer Registers.

Devices with Self-Programming capability can use the ELPM instruction to read the Fuse and Lock bit value. Refer to the device documentation for a detailed description.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

The result of these combinations is undefined:

ELPM r30, Z+ ELPM r31, Z+

Operation:

(i) $R0 \leftarrow (RAMPZ:Z)$

 $Rd \leftarrow (RAMPZ:Z)$ (ii)

(iii) $Rd \leftarrow (RAMPZ:Z)$ $(RAMPZ:Z) \leftarrow (RAMPZ:Z) + 1$

Comment:

RAMPZ:Z: Unchanged, R0 implied destination register

RAMPZ:Z: Unchanged

RAMPZ:Z: Post incremented

Syntax: Operands: **ELPM** None, R0 implied

$PC \leftarrow PC + 1$ PC ← PC + 1 PC ← PC + 1

Program Counter:

ELPM Rd, Z $0 \le d \le 31$ (ii) (iii) ELPM Rd, Z+ $0 \le d \le 31$

16 bit Opcode:

(i)	1001	0101	1101	1000
(ii)	1001	D000	dddd	0110
(iii)	1001	D000	dddd	0111

Status Register (SREG) and Boolean Formula:

ı	Т	н	s	V	N	Z	С
_	_	-	-	-	-	-	-

Example:

(i)

```
ldi ZL, byte3(Table_1<<1); Initialize Z-pointer
         RAMPZ, ZL
        ZH, byte2(Table_1<<1)
   ldi ZL, byte1(Table_1<<1)
   elpm r16, Z+
                              ; Load constant from Program
                               ; memory pointed to by RAMPZ:Z (Z is r31:r30)
Table_1:
                               ; 0x38 is addressed when \rm Z_{LSB} = 0
.dw 0x3738
                               ; 0x37 is addressed when Z_{LSB} = 1
```

69





. . .

Words: 1 (2 bytes) Cycles: 3

70 AVR Instruction Set



EOR - Exclusive OR

Description:

Performs the logical EOR between the contents of register Rd and register Rr and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd \oplus Rr$

16-bit Opcode:

0010	01rd	dddd	rrrr
0020	0 2 2 0	or or or or	

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С
1-1	-	=	⇔	0	\Leftrightarrow	⇔	-

S: $N \oplus V$, For signed tests.

V: 0

Cleared

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7 •R6 •R5 •R4• R3• R2 •R1• R0
Set if the result is \$00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

eor r4,r4 ; Clear r4

eor ${\tt r0,r22}$; Bitwise exclusive or between ${\tt r0}$ and ${\tt r22}$

Words: 1 (2 bytes)







FMUL - Fractional Multiply Unsigned

Description:

This instruction performs 8-bit \times 8-bit \rightarrow 16-bit unsigned multiplication and shifts the result one bit left.

Rd		Rr		R1	R0
Multiplicand	×	Multiplier	Æ	Product High	Product Low
- 8		8	_	1	6

Let (N.Q) denote a fractional number with N binary digits left of the radix point, and Q binary digits right of the radix point. A multiplication between two numbers in the formats (N1.Q1) and (N2.Q2) results in the format ((N1+N2).(Q1+Q2)). For signal processing applications, the format (1.7) is widely used for the inputs, resulting in a (2.14) format for the product. A left shift is required for the high byte of the product to be in the same format as the inputs. The FMUL instruction incorporates the shift operation in the same number of cycles as MUL.

The (1.7) format is most commonly used with signed numbers, while FMUL performs an unsigned multiplication. This instruction is therefore most useful for calculating one of the partial products when performing a signed multiplication with 16-bit inputs in the (1.15) format, yielding a result in the (1.31) format. Note: the result of the FMUL operation may suffer from a 2's complement overflow if interpreted as a number in the (1.15) format. The MSB of the multiplication before shifting must be taken into account, and is found in the carry bit. See the following example.

The multiplicand Rd and the multiplier Rr are two registers containing unsigned fractional numbers where the implicit radix point lies between bit 6 and bit 7. The 16-bit unsigned fractional product with the implicit radix point between bit 14 and bit 15 is placed in R1 (high byte) and R0 (low byte).

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) R1:R0 \leftarrow Rd \times Rr (unsigned (1.15) \leftarrow unsigned (1.7) \times unsigned (1.7))

Syntax: Operands: Program Counter: (i) FMUL Rd,Rr $16 \le d \le 23$, $16 \le r \le 23$ PC \leftarrow PC + 1

16-bit Opcode:

Status Register (SREG) and Boolean Formula:

I	T	Н	s	V	N	Z	С	
-	-	-	-	-	-	⇔	\Leftrightarrow	

C: R16

Set if bit 15 of the result before left shift is set; cleared otherwise.

Z: R15 •R14 •R13 •R12 •R11 •R10 •R9 •R8 •R7 • R6 • R5 • R4 • R3 • R2 •R1 • R0 Set if the result is \$0000; cleared otherwise.

R (Result) equals R1,R0 after the operation.

72 AVR Instruction Set



Example:

```
; * DESCRIPTION
;*Signed fractional multiply of two 16-bit numbers with 32-bit result.
;*r19:r18:r17:r16 = ( r23:r22 * r21:r20 ) << 1
fmuls16x16_32:
 clrr2
 fmulsr23, r21;((signed)ah * (signed)bh) << 1
 movwr19:r18, r1:r0
 fmulr22, r20;(al * bl) << 1
 adcr18, r2
 movwr17:r16, r1:r0
 fmulsur23, r20;((signed)ah * bl) << 1
 sbcr19, r2
 addr17, r0
 adcr18, r1
 adcr19, r2
 fmulsur21, r22;((signed)bh * al) << 1
 sbcr19, r2
 addr17, r0
 adcr18, r1
 adcr19, r2
```

Words: 1 (2 bytes)







FMULS – Fractional Multiply Signed

Description:

This instruction performs 8-bit × 8-bit → 16-bit signed multiplication and shifts the result one bit left.



Let (N.Q) denote a fractional number with N binary digits left of the radix point, and Q binary digits right of the radix point. A multiplication between two numbers in the formats (N1.Q1) and (N2.Q2) results in the format ((N1+N2).(Q1+Q2)). For signal processing applications, the format (1.7) is widely used for the inputs, resulting in a (2.14) format for the product. A left shift is required for the high byte of the product to be in the same format as the inputs. The FMULS instruction incorporates the shift operation in the same number of cycles as MULS.

The multiplicand Rd and the multiplier Rr are two registers containing signed fractional numbers where the implicit radix point lies between bit 6 and bit 7. The 16-bit signed fractional product with the implicit radix point between bit 14 and bit 15 is placed in R1 (high byte) and R0 (low byte).

Note that when multiplying 0x80 (-1) with 0x80 (-1), the result of the shift operation is 0x8000 (-1). The shift operation thus gives a two's complement overflow. This must be checked and handled by software.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) R1:R0 \leftarrow Rd \times Rr (signed (1.15) \leftarrow signed (1.7) \times signed (1.7))

0rrr

Status Register (SREG) and Boolean Formula:

1ddd

0011

- 1	Т	Н	S	V	N	Z	С
8 - -8	-	3:-3	1-	N=1	-	⇔	⇔

C: R16

0000

Set if bit 15 of the result before left shift is set; cleared otherwise.

Z: R15 •R14 •R13 •R12 •R11 •R10 •R9 •R8 •R7• R6• R5• R4• R3• R2 •R1• R0 Set if the result is \$0000; cleared otherwise.

R (Result) equals R1,R0 after the operation.

Example:

```
fmuls r23,r22 ; Multiply signed r23 and r22 in (1.7) format, result in (1.15) format movw r23:r22,r1:r0 ; Copy result back in r23:r22
```

74 AVR Instruction Set



Words: 1 (2 bytes)







FMULSU – Fractional Multiply Signed with Unsigned

Description:

This instruction performs 8-bit × 8-bit → 16-bit signed multiplication and shifts the result one bit left.



Let (N.Q) denote a fractional number with N binary digits left of the radix point, and Q binary digits right of the radix point. A multiplication between two numbers in the formats (N1.Q1) and (N2.Q2) results in the format ((N1+N2).(Q1+Q2)). For signal processing applications, the format (1.7) is widely used for the inputs, resulting in a (2.14) format for the product. A left shift is required for the high byte of the product to be in the same format as the inputs. The FMULSU instruction incorporates the shift operation in the same number of cycles as MULSU.

The (1.7) format is most commonly used with signed numbers, while FMULSU performs a multiplication with one unsigned and one signed input. This instruction is therefore most useful for calculating two of the partial products when performing a signed multiplication with 16-bit inputs in the (1.15) format, yielding a result in the (1.31) format. Note: the result of the FMULSU operation may suffer from a 2's complement overflow if interpreted as a number in the (1.15) format. The MSB of the multiplication before shifting must be taken into account, and is found in the carry bit. See the following example.

The multiplicand Rd and the multiplier Rr are two registers containing fractional numbers where the implicit radix point lies between bit 6 and bit 7. The multiplicand Rd is a signed fractional number, and the multiplier Rr is an unsigned fractional number. The 16-bit signed fractional product with the implicit radix point between bit 14 and bit 15 is placed in R1 (high byte) and R0 (low byte).

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) R1:R0 \leftarrow Rd \times Rr (signed (1.15) \leftarrow signed (1.7) \times unsigned (1.7))

(i)	Synt FMU	ax: ILSU Rd,Ri	Opera r 16 ≤ 0	nds: I ≤ 23, 16≤ r ≤ 23	Program Counter: PC ← PC + 1
	16-bi	t Opcode:			
	0000	0011	1ddd	1rrr	

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С	
-	_	-	_	-	-	⇔	⇔	

- C: R16
 - Set if bit 15 of the result before left shift is set; cleared otherwise.
- Z: R15 •R14 •R13 •R12 •R11 •R10 •R9 •R8 •R7• R6• R5• R4• R3• R2 •R1• R0 Set if the result is \$0000; cleared otherwise.

R (Result) equals R1,R0 after the operation.

76 AVR Instruction Set



Example:

```
; * DESCRIPTION
;*Signed fractional multiply of two 16-bit numbers with 32-bit result.
;*r19:r18:r17:r16 = ( r23:r22 * r21:r20 ) << 1
fmuls16x16_32:
clrr2
 fmulsr23, r21;((signed)ah * (signed)bh) << 1
movwr19:r18, r1:r0
 fmulr22, r20; (al * bl) << 1
 adcr18, r2
 movwr17:r16, r1:r0
 fmulsur23, r20; ((signed)ah * bl) << 1
 sbcr19, r2
 addr17, r0
 adcr18, r1
 adcr19, r2
 fmulsur21, r22;((signed)bh * al) << 1
 sbcr19, r2
 addr17, r0
 adcr18, r1
 adcr19, r2
```

Words: 1 (2 bytes)







ICALL - Indirect Call to Subroutine

Description:

Calls to a subroutine within the entire 4M (words) Program memory. The return address (to the instruction after the CALL) will be stored onto the Stack. See also RCALL. The Stack Pointer uses a post-decrement scheme during CALL.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

- (i) $PC(15:0) \leftarrow Z(15:0)$ Devices with 16 bits PC, 128K bytes Program memory maximum.
- (ii) PC(15:0) ← Z(15:0) Devices with 22 bits PC, 8M bytes Program memory maximum.

 $PC(21:16) \leftarrow 0$

(i)	Syntax: ICALL	Operands: None	Program Counter: See Operation	Stack: STACK \leftarrow PC + 1 SP \leftarrow SP - 2 (2 bytes, 16 bits)
(ii)	ICALL	None	See Operation	STACK \leftarrow PC + 1 SP \leftarrow SP - 3 (3 bytes, 22 bits)

16-b	it Opcode:		
1001	0101	0000	

Status Register (SREG) and Boolean Formula:

ı	T	н	S	V	N	Z	С	
_	_	-	-	_	_	-	_	

1001

Example:

mov

r30,r0 ; Set offset to call table

icall ; Call routine pointed to by r31:r30

Words: 1 (2 bytes)

Cycles: 3, devices with 16 bit PC

4, devices with 22 bit PC

Cycles XMEGA: 2, devices with 16 bit PC

3, devices with 22 bit PC

78



IJMP - Indirect Jump

Description:

Indirect jump to the address pointed to by the Z (16 bits) Pointer Register in the Register File. The Z-pointer Register is 16 bits wide and allows jump within the lowest 64K words (128K bytes) section of Program memory.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

- (i) $PC \leftarrow Z(15:0)$ Devices with 16 bits PC, 128K bytes Program memory maximum.
- (ii) PC(15:0) ← Z(15:0) Devices with 22 bits PC, 8M bytes Program memory maximum.

 $PC(21:16) \leftarrow 0$

Syntax: Operands: Program Counter: Stack:
(i),(ii) IJMP None See Operation Not Affected

16-bit Opcode:

1001	0100	0000	1001
------	------	------	------

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С
-	-	-	1.—	-	-	-	-

Example:

v r30,r0 ; Set offset to jump table

ijmp ; Jump to routine pointed to by r31:r30

Words: 1 (2 bytes)







IN - Load an I/O Location to Register

Description:

Loads data from the I/O Space (Ports, Timers, Configuration Registers etc.) into register Rd in the Register File.

Operation:

(i) $Rd \leftarrow I/O(A)$

16-bit Opcode:

1011	0AAd	dddd	AAAA
------	------	------	------

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С
-	-	- S	-	_	-	-	a—2

Example:

in r25,\$16 ; Read Port B

cpi r25,4 ; Compare read value to constant

breq exit ; Branch if r25=4

. . .

exit: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1

80 AVR Instruction Set



INC - Increment

Description:

Adds one -1- to the contents of register Rd and places the result in the destination register Rd.

The C Flag in SREG is not affected by the operation, thus allowing the INC instruction to be used on a loop counter in multiple-precision computations.

When operating on unsigned numbers, only BREQ and BRNE branches can be expected to perform consistently. When operating on two's complement values, all signed branches are available.

Operation:

(i) $Rd \leftarrow Rd + 1$

16-bit Opcode:

1001	010d	dddd	0011
------	------	------	------

Status Register and Boolean Formula:

1	T	н	s	V	N	Z	С
-	-	-	\Leftrightarrow	\Leftrightarrow	⇔	\Leftrightarrow	-

S: $N \oplus V$

For signed tests.

V: R7 •R6 •R5 •R4 •R3• R2 •R1 •R0

Set if two's complement overflow resulted from the operation; cleared otherwise. Two's complement overflow occurs if and only if Rd was \$7F before the operation.

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7 •R6 •R5 •R4•R3 •R2• R1• R0
Set if the result is \$00; Cleared otherwise.

R (Result) equals Rd after the operation.

Example:



81





Words: 1 (2 bytes)

Cycles: 1

82 AVR Instruction Set



JMP - Jump

Description:

Jump to an address within the entire 4M (words) Program memory. See also RJMP.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) $PC \leftarrow k$

	Syntax:	Operands:	Program Counter:	Stack:
(i)	JMP k	$0 \le k < 4M$	$PC \leftarrow k$	Unchanged

32-bit Opcode:

1001	010k	kkkk	110k
kkkk	kkkk	kkkk	kkkk

Status Register (SREG) and Boolean Formula:

I	T	Н	S	V	N	Z	С
_	_	-	_	_	_	_	-

Example:

mov r1,r0 ; Copy r0 to r1
jmp farplc ; Unconditional jump
...
farplc: nop ; Jump destination (do nothing)

Words: 2 (4 bytes)







Program Counter:

 $PC \leftarrow PC + 1$

LAC - Load And Clear

Description:

Operation:

(i) $(Z) \leftarrow Rd \cdot (\$FF - (Z))$

16-bit Opcode:

1001 001r rrrr 0110

Words: 1 (2 bytes)

Cycles: 1

84 AVR Instruction Set



Program Counter:

 $PC \leftarrow PC + 1$

LAS - Load And Set

Description:

Operation:

(i) $(Z) \leftarrow Rd \ v \ (Z), \ Rd \leftarrow (Z)$

16-bit Opcode:

1001 001r rrrr 0101

Words: 1 (2 bytes)

Cycles: 1







Program Counter:

 $PC \leftarrow PC + 1$

LAT - Load And Toggle

Description:

Operation:

(i) $(Z) \leftarrow Rd \oplus (Z), Rd \leftarrow (Z)$

16-bit Opcode:

1001 001r rrrr 0111

Words: 1 (2 bytes)

Cycles: 1

AVR Instruction Set

0856I-AVR-07/10



LD – Load Indirect from Data Space to Register using Index X

Description:

Loads one byte indirect from the data space to a register. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. In some parts the Flash Memory has been mapped to the data space and can be read using this command. The EEPROM has a separate address space.

The data location is pointed to by the X (16 bits) Pointer Register in the Register File. Memory access is limited to the current data segment of 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPX in register in the I/O area has to be changed.

The X-pointer Register can either be left unchanged by the operation, or it can be post-incremented or pre-decremented. These features are especially suited for accessing arrays, tables, and Stack Pointer usage of the X-pointer Register. Note that only the low byte of the X-pointer is updated in devices with no more than 256 bytes data space. For such devices, the high byte of the pointer is not used by this instruction and can be used for other purposes. The RAMPX Register in the I/O area is updated in parts with more than 64K bytes data space or more than 64K bytes Program memory, and the increment/decrement is added to the entire 24-bit address on such devices.

Not all variants of this instruction is available in all devices. Refer to the device specific instruction set summary.

In the Reduced Core tinyAVR the LD instruction can be used to achieve the same operation as LPM since the program memory is mapped to the data memory space.

The result of these combinations is undefined:

LD r26, X+

LD r27, X+

LD r26, -X

LD r27, -X

Using the X-pointer:

	Operation:		Comment:
(i)	$Rd \leftarrow (X)$		X: Unchanged
(ii)	$Rd \leftarrow (X)$	$X \leftarrow X + 1$	X: Post incremented
(iii)	X ← X - 1	$Rd \leftarrow (X)$	X: Pre decremented
	Syntax:	Operands:	Program Counter:
(i)	Syntax: LD Rd, X	Operands: 0 ≤ d ≤ 31	Program Counter: PC ← PC + 1
(i) (ii)			







16-bit Opcode:

(i)	1001	D000	dddd	1100
(ii)	1001	D000	dddd	1101
(iii)	1001	D000	dddd	1110

Status Register (SREG) and Boolean Formula:

1	Т	н	S	V	N	Z	С	
-	-	-	-	-	-	_	_	

AVR Instruction Set

No olvide citar esta tesis



Example:

```
clr r27 ; Clear X high byte

ldi r26,$60 ; Set X low byte to $60

ld r0,X+ ; Load r0 with data space loc. $60(X post inc)

ld r1,X ; Load r1 with data space loc. $61

ldi r26,$63 ; Set X low byte to $63

ld r2,X ; Load r2 with data space loc. $63

ld r3,-X ; Load r3 with data space loc. $62(X pre dec)
```

Words: 1 (2 bytes)

Cycles: (i) 1⁽²⁾

(ii) 2

(iii) 3⁽²⁾

Cycles XMEGA: (i) 1⁽¹⁾

(ii) 1⁽¹⁾

(iii) 2⁽¹⁾

Notes: 1. IF the LD instruction is accessing internal SRAM, one extra cycle is inserted.

2. LD instruction can load data from program memory since the flash is memory mapped. Loading data from the data memory takes 1 clock cycle, and loading from the program memory takes 2 clock cycles. But if an interrupt occur (before the last clock cycle) no additional clock cycles is necessary when loading from the program memory. Hence, the instruction takes only 1 clock cycle to execute.

LD instruction with pre-decrement can load data from program memory since the flash is memory mapped. Loading data from the data memory takes 2 clock cycles, and loading from the program memory takes 3 clock cycles. But if an interrupt occur (before the last clock cycle) no additional clock cycles is necessary when loading from the program memory. Hence, the instruction takes only 1 clock cycle to execute.







LD (LDD) – Load Indirect from Data Space to Register using Index Y

Description:

Loads one byte indirect with or without displacement from the data space to a register. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. In some parts the Flash Memory has been mapped to the data space and can be read using this command. The EEPROM has a separate address space.

The data location is pointed to by the Y (16 bits) Pointer Register in the Register File. Memory access is limited to the current data segment of 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPY in register in the I/O area has to be changed.

The Y-pointer Register can either be left unchanged by the operation, or it can be post-incremented or pre-decremented. These features are especially suited for accessing arrays, tables, and Stack Pointer usage of the Y-pointer Register. Note that only the low byte of the Y-pointer is updated in devices with no more than 256 bytes data space. For such devices, the high byte of the pointer is not used by this instruction and can be used for other purposes. The RAMPY Register in the I/O area is updated in parts with more than 64K bytes data space or more than 64K bytes Program memory, and the increment/decrement/displacement is added to the entire 24-bit address on such devices.

Not all variants of this instruction is available in all devices. Refer to the device specific instruction set summary.

In the Reduced Core tinyAVR the LD instruction can be used to achieve the same operation as LPM since the program memory is mapped to the data memory space.

The result of these combinations is undefined:

LD r28, Y+ LD r29, Y+ LD r28, -Y LD r29, -Y

Using the Y-pointer:

	Operation:		Comment:
(i)	$Rd \leftarrow (Y)$		Y: Unchanged
(ii)	$Rd \leftarrow (Y)$	Y ← Y + 1	Y: Post incremented
(iii)	$Y \leftarrow Y - 1$	$Rd \leftarrow (Y)$	Y: Pre decremented
(iv)	$Rd \leftarrow (Y+q)$		Y: Unchanged, q: Displacement
	Syntax:	Operands:	Program Counter:
(i)	LD Rd, Y	$0 \le d \le 31$	PC ← PC + 1
(ii)	LD Rd, Y+	$0 \le d \le 31$	PC ← PC + 1
(iii)	LD Rd, -Y	$0 \le d \le 31$	PC ← PC + 1
(iv)	LDD Rd, Y+q	$0 \leq d \leq 31, 0 \leq q \leq 63$	PC ← PC + 1

AVR Instruction Set

0856I-AVR-07/10



16-bit Opcode:

(i)	1000	D000	dddd	1000
(ii)	1001	D000	dddd	1001
(iii)	1001	D000	dddd	1010
(iv)	10q0	qq0d	dddd	1qqq

Status Register (SREG) and Boolean Formula:

	Т	Н	S	V	N	Z	С
1=	-	-	-	=	-	-	-

Example:

```
clr
    r29
                 ; Clear Y high byte
ldi
    r28,$60
                  ; Set Y low byte to $60
     r0,Y+
                  ; Load r0 with data space loc. $60(Y post inc)
                  ; Load r1 with data space loc. $61
    r28,$63
                  ; Set Y low byte to $63
1d
     r2,Y
                  ; Load r2 with data space loc. $63
ld
     r3,-Y
                  ; Load r3 with data space loc. $62(Y pre dec)
ldd r4,Y+2
                  ; Load r4 with data space loc. $64
```

Words: 1 (2 bytes)

Cycles:

(i) 1⁽²⁾ (ii) 2

(iii) 3⁽²⁾

Cycles XMEGA:

(i) 1⁽¹⁾ (ii) 1⁽¹⁾

(iii) 2⁽¹⁾ (iv) 2⁽¹⁾

- Notes: 1. IF the LD instruction is accessing internal SRAM, one extra cycle is inserted.
 - 2. LD instruction can load data from program memory since the flash is memory mapped. Loading data from the data memory takes 1 clock cycle, and loading from the program memory takes 2 clock cycles. But if an interrupt occur (before the last clock cycle) no additional clock cycles is necessary when loading from the program memory. Hence, the instruction takes only 1 clock cycle to execute.

LD instruction with pre-decrement can load data from program memory since the flash is memory mapped. Loading data from the data memory takes 2 clock cycles, and loading from the program memory takes 3 clock cycles. But if an interrupt occur (before the last clock cycle) no additional clock cycles is necessary when loading from the program memory. Hence, the instruction takes only 1 clock cycle to execute.







LD (LDD) – Load Indirect From Data Space to Register using Index Z

Description:

Loads one byte indirect with or without displacement from the data space to a register. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. In some parts the Flash Memory has been mapped to the data space and can be read using this command. The EEPROM has a separate address space.

The data location is pointed to by the Z (16 bits) Pointer Register in the Register File. Memory access is limited to the current data segment of 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPZ in register in the I/O area has to be changed.

The Z-pointer Register can either be left unchanged by the operation, or it can be post-incremented or pre-decremented. These features are especially suited for Stack Pointer usage of the Z-pointer Register, however because the Z-pointer Register can be used for indirect subroutine calls, indirect jumps and table lookup, it is often more convenient to use the X or Y-pointer as a dedicated Stack Pointer. Note that only the low byte of the Z-pointer is updated in devices with no more than 256 bytes data space. For such devices, the high byte of the pointer is not used by this instruction and can be used for other purposes. The RAMPZ Register in the I/O area is updated in parts with more than 64K bytes data space or more than 64K bytes Program memory, and the increment/decrement/displacement is added to the entire 24-bit address on such devices.

Not all variants of this instruction is available in all devices. Refer to the device specific instruction set summary.

In the Reduced Core tinyAVR the LD instruction can be used to achieve the same operation as LPM since the program memory is mapped to the data memory space.

For using the Z-pointer for table lookup in Program memory see the LPM and ELPM instructions.

The result of these combinations is undefined:

LD r30, Z+ LD r31, Z+ LD r30, -Z LD r31, -Z

Using the Z-pointer:

	Operation:	Comment:	
(i)	$Rd \leftarrow (Z)$		Z: Unchanged
(ii)	$Rd \leftarrow (Z)$	$Z \leftarrow Z + 1$	Z: Post increment
(iii)	$Z \leftarrow Z - 1$	$Rd \leftarrow (Z)$	Z: Pre decrement
(iv)	$Rd \leftarrow (Z+q)$		Z: Unchanged, q: Displacement
	Syntax:	Operands:	Program Counter:
(i)	LD Rd, Z	$0 \le d \le 31$	PC ← PC + 1
(ii)	LD Rd, Z+	$0 \le d \le 31$	PC ← PC + 1
(iii)	LD Rd, -Z	$0 \le d \le 31$	PC ← PC + 1
(iv)	LDD Rd, Z+q	$0 \leq d \leq 31, 0 \leq q \leq 63$	PC ← PC + 1

AVR Instruction Set

0856I-AVR-07/10



16-bit Opcode:

(i)	1000	D000	dddd	0000
(ii)	1001	D000	dddd	0001
(iii)	1001	D000	dddd	0010
(iv)	10q0	qq0d	dddd	0qqq

Status Register (SREG) and Boolean Formula:

	Т	Н	S	V	N	Z	С
1=	-	-	-	=	-	-	-

Example:

```
; Clear Z high byte
    r30,$60 ; Set Z low byte to $60
    r0,Z+
            ; Load r0 with data space loc. $60(Z post inc)
1d
    r1,Z
             ; Load r1 with data space loc. $61
ldi r30,$63 ; Set Z low byte to $63
1d
    r2.7
            ; Load r2 with data space loc. $63
            ; Load r3 with data space loc. $62(Z pre dec)
ld r3,-Z
ldd r4,Z+2 ; Load r4 with data space loc. $64
```

Words: 1 (2 bytes)

Cycles:

(i) 1⁽²⁾ (ii) 2 (iii) 3⁽²⁾

Cycles XMEGA:

(i) 1⁽¹⁾ (ii) 1⁽¹⁾ (iii) 2⁽¹⁾

(iv) 2⁽¹⁾

- Notes: 1. IF the LD instruction is accessing internal SRAM, one extra cycle is inserted.
 - 2. LD instruction can load data from program memory since the flash is memory mapped. Loading data from the data memory takes 1 clock cycle, and loading from the program memory takes 2 clock cycles. But if an interrupt occur (before the last clock cycle) no additional clock cycles is necessary when loading from the program memory. Hence, the instruction takes only 1 clock cycle to execute.

LD instruction with pre-decrement can load data from program memory since the flash is memory mapped. Loading data from the data memory takes 2 clock cycles, and loading from the program memory takes 3 clock cycles. But if an interrupt occur (before the last clock cycle) no additional clock cycles is necessary when loading from the program memory. Hence, the instruction takes only 1 clock cycle to execute.







LDI - Load Immediate

Description:

Loads an 8 bit constant directly to register 16 to 31.

Operation:

(i) $Rd \leftarrow K$

(i)

 Syntax:
 Operands:

 LDI Rd,K
 $16 \le d \le 31, 0 \le K \le 255$

Program Counter: PC ← PC + 1

16-bit Opcode:

1110 KKKK dddd KKKK

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С
-	-	-	-	_	-	_	_

Example:

clr r31 ; Clear Z high byte
ldi r30,\$F0 ; Set Z low byte to \$F0
lpm ; Load constant from Program
; memory pointed to by Z

Words: 1 (2 bytes)

Cycles: 1



LDS – Load Direct from Data Space

Description:

Loads one byte from the data space to a register. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the register file only. The EEPROM has a separate address space.

A 16-bit address must be supplied. Memory access is limited to the current data segment of 64K bytes. The LDS instruction uses the RAMPD Register to access memory above 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPD in register in the I/O area has to be changed.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) $Rd \leftarrow (k)$

32-bit Opcode:

1001	D000	dddd	0000
kkkk	kkkk	kkkk	kkkk

Status Register (SREG) and Boolean Formula:

1	T	Н	s	V	N	Z	С
_	_	_	_	-	_	_	-

Example:

1ds r2,\$FF00 ; Load r2 with the contents of data space location \$FF00
add r2,r1 ; add r1 to r2
sts \$FF00,r2 ; Write back

Words: 2 (4 bytes)

Cycles:

Cycles XMEGA: 2 If the LDS instruction is accessing internal SRAM, one extra cycle is inserted.







LDS (16-bit) - Load Direct from Data Space

Description:

Loads one byte from the data space to a register. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the register file only. In some parts the Flash memory has been mapped to the data space and can be read using this command. The EEPROM has a separate address space.

A 7-bit address must be supplied. The address given in the instruction is coded to a data space address as follows:

 $ADDR[7:0] = (\overline{INST[8]}, INST[8], INST[10], INST[9], INST[3], INST[2], INST[1], INST[0])$

Memory access is limited to the address range 0x40..0xbf.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) $Rd \leftarrow (k)$

16-bit Opcode:

	1010	0kkk	dddd	kkkk
--	------	------	------	------

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С
-	-	-	-	-	_	-	-

Example:

lds r16,\$00 ; Load r16 with the contents of data space location \$00 add r16,r17 ; add r17 to r16 sts \$00,r16 ; Write result to the same address it was fetched from

Words: 1 (2 bytes)

Cycles: 1

Note: Registers r0..r15 are remapped to r16..r31.

AVR Instruction Set =

0856I-AVR-07/10



LPM - Load Program Memory

Description:

Loads one byte pointed to by the Z-register into the destination register Rd. This instruction features a 100% space effective constant initialization or constant data fetch. The Program memory is organized in 16-bit words while the Z-pointer is a byte address. Thus, the least significant bit of the Z-pointer selects either low byte ($Z_{LSB} = 0$) or high byte ($Z_{LSB} = 1$). This instruction can address the first 64K bytes (32K words) of Program memory. The Z-pointer Register can either be left unchanged by the operation, or it can be incremented. The incrementation does not apply to the RAMPZ Register.

Devices with Self-Programming capability can use the LPM instruction to read the Fuse and Lock bit values. Refer to the device documentation for a detailed description.

The LPM instruction is not available in all devices. Refer to the device specific instruction set summary.

The result of these combinations is undefined:

LPM r30, Z+ LPM r31, Z+

Operation:

- (i) $R0 \leftarrow (Z)$
- (ii) $Rd \leftarrow (Z)$
- (iii) $Rd \leftarrow (Z)$ $Z \leftarrow Z + 1$

Comment:

- Z: Unchanged, R0 implied destination register
- Z: Unchanged
- Z: Post incremented

Syntax: Operands:

(i)	LPM	None, R0 implied
(ii)	LPM Rd, Z	$0 \le d \le 31$
(iii)	LPM Rd, Z+	$0 \leq d \leq 31$

Program Counter:

 $PC \leftarrow PC + 1$ $PC \leftarrow PC + 1$ $PC \leftarrow PC + 1$

16-bit Opcode:

	(i)	1001	0101	1100	1000
	(ii)	1001	D000	dddd	0100
Г	(iii)	1001	D000	dddd	0101

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С	
-	-	-	-	-	-	-	-	

Example:



97





Words: 1 (2 bytes) Cycles: 3

98 AVR Instruction Set



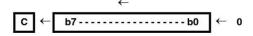
LSL - Logical Shift Left

Description:

Shifts all bits in Rd one place to the left. Bit 0 is cleared. Bit 7 is loaded into the C Flag of the SREG. This operation effectively multiplies signed and unsigned values by two.

Operation:

(i)



Program Counter: PC ← PC + 1

Status Register (SREG) and Boolean Formula:

I	T	н	s	V	N	Z	С	
-	-	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	

H: Rd3

S: $N \oplus V$, For signed tests.

V: N ⊕ C (For N and C after the shift)

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7• R6 •R5• R4• R3 •R2• R1• R0

Set if the result is \$00; cleared otherwise.

C: Rd7

Set if, before the shift, the MSB of Rd was set; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

add r0,r4 ; Add r4 to r0 lsl r0 ; Multiply r0 by 2

Words: 1 (2 bytes)

Cycles: 1





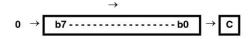


LSR - Logical Shift Right

Description:

Shifts all bits in Rd one place to the right. Bit 7 is cleared. Bit 0 is loaded into the C Flag of the SREG. This operation effectively divides an unsigned value by two. The C Flag can be used to round the result.

Operation:



Program Counter: PC ← PC + 1

16-bit Opcode:

1001 010d	dddd	0110
-----------	------	------

Status Register (SREG) and Boolean Formula:

I	Т	н	s	V	N	Z	С	
-	-	=	⇔	⇔	0	\Leftrightarrow	⇔	

S: $N \oplus V$, For signed tests.

V: $N \oplus C$ (For N and C after the shift)

N: 0

Z: R7• R6 •R5• R4• R3 •R2• R1• R0
Set if the result is \$00; cleared otherwise.

C: Rd0 Set if, before the shift, the LSB of Rd was set; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

add r0,r4 ; Add r4 to r0 lsr r0 ; Divide r0 by 2

Words: 1 (2 bytes)

Cycles: 1

100 AVR Instruction Set



MOV - Copy Register

Description:

This instruction makes a copy of one register into another. The source register Rr is left unchanged, while the destination register Rd is loaded with a copy of Rr.

Operation:

(i) $Rd \leftarrow Rr$

16-bit Opcode:

0010	11rd	bbbb	rrrr
0020		Section of	

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С	
-	-	_	_	_	-	-	_	

Example:

mov r16,r0 ; Copy r0 to r16
call check ; Call subroutine
...
check: cpi r16,\$11 ; Compare r16 to \$11
...
ret ; Return from subroutine

Words: 1 (2 bytes)

Cycles: 1







MOVW - Copy Register Word

Description:

This instruction makes a copy of one register pair into another register pair. The source register pair Rr+1:Rr is left unchanged, while the destination register pair Rd+1:Rd is loaded with a copy of Rr + 1:Rr.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) $Rd+1:Rd \leftarrow Rr+1:Rr$

Syntax: Operands: (i) MOVW Rd+1:Rd,Rr+1Rrd \in {0,2,...,30}, r \in {0,2,...,30}

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

	0000	0001	dddd	rrrr
--	------	------	------	------

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С
-	_	_	_	_	_	_	_

Example:

movw r17:16,r1:r0 ; Copy r1:r0 to r17:r16
call check ; Call subroutine
...
check: cpi r16,\$11 ; Compare r16 to \$11
...
cpi r17,\$32 ; Compare r17 to \$32
...
ret ; Return from subroutine

Words: 1 (2 bytes)

Cycles: 1

AVR Instruction Set

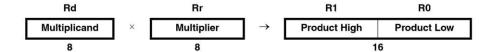
0856I-AVR-07/10



MUL - Multiply Unsigned

Description:

This instruction performs 8-bit \times 8-bit \rightarrow 16-bit unsigned multiplication.



The multiplicand Rd and the multiplier Rr are two registers containing unsigned numbers. The 16-bit unsigned product is placed in R1 (high byte) and R0 (low byte). Note that if the multiplicand or the multiplier is selected from R0 or R1 the result will overwrite those after multiplication.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) R1:R0 \leftarrow Rd \times Rr (unsigned \leftarrow unsigned \times unsigned)

(i)	Syntax: MUL Rd,Rr 16-bit Opcode:		Opera 0 ≤ d :	nds: ≤ 31, 0 ≤ r ≤ 31	Program Counter: PC ← PC + 1
	1001	11rd	dddd	rrrr	

Status Register (SREG) and Boolean Formula:

1	T	Н	S	V	N	Z	С	
-	-	-	-	_	_	\Leftrightarrow	\Leftrightarrow]

- C: R15
 - Set if bit 15 of the result is set; cleared otherwise.
- Z: R15 •R14 •R13 •R12 •R11 •R10 •R9 •R8 •R7 R6 R5 R4 R3 R2 •R1 R0 Set if the result is \$0000; cleared otherwise.

R (Result) equals R1,R0 after the operation.

Example:

mul r5,r4 ; Multiply unsigned r5 and r4
movw r4,r0 ; Copy result back in r5:r4

Words: 1 (2 bytes)

Cycles: 2



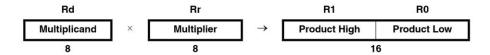




MULS – Multiply Signed

Description:

This instruction performs 8-bit \times 8-bit \rightarrow 16-bit signed multiplication.



The multiplicand Rd and the multiplier Rr are two registers containing signed numbers. The 16-bit signed product is placed in R1 (high byte) and R0 (low byte).

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) R1:R0 \leftarrow Rd \times Rr (signed \leftarrow signed \times signed)

0000	0010	dddd	rrrr
	0000	0000 0010	0000 0010 dddd

Status Register (SREG) and Boolean Formula:

1	Ţ	Н	s	V	N	Z	С	
-	-	-	-	-	-	\Leftrightarrow	\Leftrightarrow	

C: R15

Set if bit 15 of the result is set; cleared otherwise.

Z: R15 •R14 •R13 •R12 •R11 •R10 •R9 •R8 •R7 • R6 • R5 • R4 • R3 • R2 •R1 • R0 Set if the result is \$0000; cleared otherwise.

R (Result) equals R1,R0 after the operation.

Example:

muls r21,r20 ; Multiply signed r21 and r20 movw r20,r0 ; Copy result back in r21:r20

Words: 1 (2 bytes)

Cycles: 2

104 AVR Instruction Set



MULSU – Multiply Signed with Unsigned

Description:

This instruction performs 8-bit \times 8-bit \rightarrow 16-bit multiplication of a signed and an unsigned number.



The multiplicand Rd and the multiplier Rr are two registers. The multiplicand Rd is a signed number, and the multiplier Rr is unsigned. The 16-bit signed product is placed in R1 (high byte) and R0 (low byte).

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) R1:R0 ← Rd × Rr (signed ← signed × unsigned)

Status Register (SREG) and Boolean Formula:

1	Т	н	s	V	N	Z	С	
_	7-1	_	_	_		\Leftrightarrow	\Leftrightarrow]

- C: R15
 - Set if bit 15 of the result is set; cleared otherwise.
- Z: R15 •R14 •R13 •R12 •R11 •R10 •R9 •R8 •R7• R6• R5• R4• R3• R2 •R1• R0 Set if the result is \$0000; cleared otherwise.

R (Result) equals R1,R0 after the operation.

Example:

<u>AIMEL</u>

105





```
movwr19:r18, r1:r0
mulr22, r20; al * bl
movwr17:r16, r1:r0
mulsur23, r20; (signed)ah * bl
sbcr19, r2
addr17, r0
adcr18, r1
adcr19, r2
mulsur21, r22; (signed)bh * al
sbcr19, r2
addr17, r0
adcr18, r1
adcr19, r2
ret
```

Words: 1 (2 bytes)

Cycles: 2

106 AVR Instruction Set



NEG – Two's Complement

Description:

Replaces the contents of register Rd with its two's complement; the value \$80 is left unchanged.

Operation:

(i) Rd ← \$00 - Rd

	Syntax:	Operands:	Program Counter:
(i)	NEG Rd	$0 \le d \le 31$	PC ← PC + 1

16-bit Opcode:

1001 010d	dddd 0001
-----------	-----------

Status Register (SREG) and Boolean Formula:

I	T	Н	s	V	N	Z	С	
-	_	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	⇔	\Leftrightarrow	\Leftrightarrow	

H: R3 + Rd3

Set if there was a borrow from bit 3; cleared otherwise

S: $N \oplus V$

For signed tests.

V: R7• R6 •R5• R4• R3 •R2• R1• R0

Set if there is a two's complement overflow from the implied subtraction from zero; cleared otherwise. A two's complement overflow will occur if and only if the contents of the Register after operation (Result) is \$80.

N: R7

Set if MSB of the result is set; cleared otherwise.

- Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$ Set if the result is \$00; Cleared otherwise.
- C: R7 + R6 + R5 + R4 + R3 + R2 + R1 + R0
 Set if there is a borrow in the implied subtraction from zero; cleared otherwise. The C Flag will be set in all cases except when the contents of Register after operation is \$00.

R (Result) equals Rd after the operation.

Example:

```
sub r11,r0 ; Subtract r0 from r11
brpl positive ; Branch if result positive
neg r11 ; Take two's complement of r11
positive: nop ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1

<u>AIMEL</u>

107





NOP - No Operation

Description:

This instruction performs a single cycle No Operation.

Operation:

(i) No

16-bit Opcode:

0000 0000 0000 0000

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С	
-	-	-	-	-	_	-	-	

Example:

clr r16 ; Clear r16 ser r17 ; Set r17 out \$18,r16 ; Write zeros to Port B

nop ; Wait (do nothing)
out \$18,r17 ; Write ones to Port B

Words: 1 (2 bytes)

Cycles: 1

108 AVR Instruction Set



OR - Logical OR

Description:

Performs the logical OR between the contents of register Rd and register Rr and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd \vee Rr$

16-bit Opcode:

0010	10rd	dddd	rrrr
------	------	------	------

Status Register (SREG) and Boolean Formula:

I	T	Н	s	V	N	Z	С	
-	-	_	\Leftrightarrow	0	\Leftrightarrow	\Leftrightarrow	:-:]

S: $N \oplus V$, For signed tests.

V: 0

Cleared

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7• R6 •R5• R4• R3 •R2• R1• R0
Set if the result is \$00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

or r15,r16 ; Do bitwise or between registers
bst r15,6 ; Store bit 6 of r15 in T Flag
brts ok ; Branch if T Flag set
...
nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1

ok:







ORI – Logical OR with Immediate

Description:

Performs the logical OR between the contents of register Rd and a constant and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd \vee K$

225. 0

Syntax: (i) ORI Rd,K

 $16 \leq d \leq 31,\, 0 \leq K \leq 255$

Operands:

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

0110 KKKK dddd KKKK

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С	
n = //	-	=	⇔	0	⇔	⇔	-	

S: $N \oplus V$, For signed tests.

V: 0

Cleared

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7• R6 •R5• R4• R3 •R2• R1• R0
Set if the result is \$00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

ori r16,\$F0 ; Set high nibble of r16 ori r17,1 ; Set bit 0 of r17

Words: 1 (2 bytes)

Cycles: 1

110 AVR Instruction Set



OUT - Store Register to I/O Location

Description:

Stores data from register Rr in the Register File to I/O Space (Ports, Timers, Configuration Registers etc.).

Operation:

(i) $I/O(A) \leftarrow Rr$

16-bit Opcode:

1011	1AAr	rrrr	AAAA

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С	
-	-	-	-	-	_	-	_	

Example:

clr r16 ; Clear r16 ser r17 ; Set r17

out \$18,r16 ; Write zeros to Port B
nop ; Wait (do nothing)
out \$18,r17 ; Write ones to Port B

Words: 1 (2 bytes)

Cycles: 1







POP - Pop Register from Stack

Description:

This instruction loads register Rd with a byte from the STACK. The Stack Pointer is pre-incremented by 1 before the POP. This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) $Rd \leftarrow STACK$

16-bit Opcode:

	1001	D000	dddd	1111
--	------	------	------	------

Status Register (SREG) and Boolean Formula:

ı	Т	Н	S	V	N	Z	С
-	-	_	-	-	-	-	-

Example:

call routine ; Call subroutine routine: r14 ; Save r14 on the Stack push push r13 ; Save r13 on the Stack r13 pop ; Restore r13 r14 ; Restore r14 pop ret ; Return from subroutine

Words: 1 (2 bytes)

Cycles: 2

112 AVR Instruction Set



PUSH - Push Register on Stack

Description:

This instruction stores the contents of register Rr on the STACK. The Stack Pointer is post-decremented by 1 after the PUSH.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) STACK ← Rr

16-bit Opcode:

		50000-0000	
1001	001d	dddd	1111

Status Register (SREG) and Boolean Formula:

ı	Т	н	S	V	N	Z	С	
-	_	_	_	_	_	_	_	

Example:

call routine; Call subroutine routine: r14 ; Save r14 on the Stack push ; Save r13 on the Stack push r13 r13 pop ; Restore r13 r14 ; Restore r14 pop ; Return from subroutine

 Words :
 1 (2 bytes)

 Cycles :
 2

 Cycles XMEGA:
 1







RCALL – Relative Call to Subroutine

Description:

Relative call to an address within PC - 2K + 1 and PC + 2K (words). The return address (the instruction after the RCALL) is stored onto the Stack. See also CALL. For AVR microcontrollers with Program memory not exceeding 4K words (8K bytes) this instruction can address the entire memory from every address location. The Stack Pointer uses a post-decrement scheme during RCALL.

Operation:

(i) $PC \leftarrow PC + k + 1$ Devices with 16 bits PC, 128K bytes Program memory maximum.

(ii) $PC \leftarrow PC + k + 1$ Devices with 22 bits PC, 8M bytes Program memory maximum.

(i)	Syntax: RCALL k	Operands: $-2K \le k < 2K$	Program Counter: $PC \leftarrow PC + k + 1$	Stack: STACK ← PC + 1 SP ← SP - 2 (2 bytes, 16 bits)
(ii)	RCALL k	$-2K \leq k < 2K$	$PC \leftarrow PC + k + 1$	STACK \leftarrow PC + 1 SP \leftarrow SP - 3 (3 bytes, 22 bits)

16-bit Opcode:

1101	kkkk	kkkk	kkkk	

Status Register (SREG) and Boolean Formula:

I	T	Н	S	V	N	Z	С
-	-	-	-	-	-	-	-

Example:

rcall routine ; Call subroutine

routine: push r14 ; Save r14 on the Stack

pop r14 ; Restore r14

ret ; Return from subroutine

Words : 1 (2 bytes)

Cycles: 3, devices with 16 bit PC

4, devices with 22 bit PC

Cycles XMEGA: 2, devices with 16 bit PC

3, devices with 22 bit PC

Cycles Reduced Core tinyAVR:4

114 AVR Instruction Set



RET – Return from Subroutine

Description:

Returns from subroutine. The return address is loaded from the STACK. The Stack Pointer uses a pre-increment scheme during RET.

Operation:

- (i) PC(15:0) ← STACK Devices with 16 bits PC, 128K bytes Program memory maximum.
- (ii) PC(21:0) ← STACKDevices with 22 bits PC, 8M bytes Program memory maximum.

Syntax:	Operands:	Program Counter:	Stack:
Syriax.	operarias.	r rogram obanici.	otuck.

(i) RET None See Operation $SP \leftarrow SP + 2$, (2bytes, 16 bits)

(ii) RET None See Operation SP←SP + 3, (3bytes,22 bits)

16-bit Opcode:

	1001	0101	0000	1000
--	------	------	------	------

Status Register (SREG) and Boolean Formula:

push

1	T	Н	S	V	N	Z	С	
-	_	_	-	_	-	_	-	

Example:

call routine ; Call subroutine

...

r14

pop r14 ; Restore r14

pop r14 ; Restore r14 ret ; Return from subroutine

; Save r14 on the Stack

Words: 1 (2 bytes)

routine:

Cycles: 4 devices with 16-bit PC 5 devices with 22-bit PC







RETI – Return from Interrupt

Description:

Returns from interrupt. The return address is loaded from the STACK and the Global Interrupt Flag is set.

Note that the Status Register is not automatically stored when entering an interrupt routine, and it is not restored when returning from an interrupt routine. This must be handled by the application program. The Stack Pointer uses a pre-increment scheme during RETI.

Operation:

- (i) PC(15:0) ← STACK Devices with 16 bits PC, 128K bytes Program memory maximum.
- (ii) PC(21:0) ← STACKDevices with 22 bits PC, 8M bytes Program memory maximum.

Syntax:		Operands:	Program Counter:	Stack		
(i)	RETI	None	See Operation	$SP \leftarrow SP + 2 (2 \text{ bytes, } 16 \text{ bits})$		
(ii)	RETI	None	See Operation	$SP \leftarrow SP + 3 (3 \text{ bytes, 22 bits})$		
	16-bit Opcode:					
	1001 0101	0001 1000				

Status Register (SREG) and Boolean Formula:

1	Т	н	s	V	N	z	С	
1	-	-	-	-	-	-	-	

l: 1

The I Flag is set.

Example:

extint: push r0 ; Save r0 on the Stack
...
pop r0 ; Restore r0
reti ; Return and enable interrupts

Words: 1 (2 bytes)

Cycles: 4 devices with 16-bit PC 5 devices with 22-bit PC



RJMP - Relative Jump

Description:

Relative jump to an address within PC - 2K +1 and PC + 2K (words). For AVR microcontrollers with Program memory not exceeding 4K words (8K bytes) this instruction can address the entire memory from every address location. See also JMP.

Operation:

(i) $PC \leftarrow PC + k + 1$

	Syntax:	Operands:	Program Counter:	Stack
(i)	RJMP k	$-2K \le k < 2K$	$PC \leftarrow PC + k + 1$	Unchanged

16-bit Opcode:

1100	kkkk	kkkk	kkkk
------	------	------	------

Status Register (SREG) and Boolean Formula:

1	Т	н	s	V	N	Z	С
-	-	-	_	-	1-	1-1	-

Example:

cpi r16,\$42 ; Compare r16 to \$42

brne error ; Branch if r16 <> \$42

rjmp ok ; Unconditional branch

error: add r16,r17 ; Add r17 to r16

inc r16 ; Increment r16

ok: nop ; Destination for rjmp (do nothing)

Words: 1 (2 bytes)

Cycles: 2





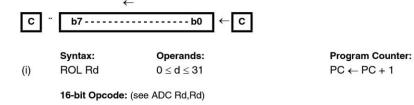


ROL – Rotate Left trough Carry

Description:

Shifts all bits in Rd one place to the left. The C Flag is shifted into bit 0 of Rd. Bit 7 is shifted into the C Flag. This operation, combined with LSL, effectively multiplies multi-byte signed and unsigned values by two.

Operation:



dddd

Status Register (SREG) and Boolean Formula:

dddd

11dd

1	Т	Н	S	V	N	Z	С	
-		\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	⇔	\Leftrightarrow	\Leftrightarrow]

H: Rd3

0001

S: $N \oplus V$, For signed tests.

V: N ⊕ C (For N and C after the shift)

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7• R6 •R5• R4• R3 •R2• R1• R0
Set if the result is \$00; cleared otherwise.

C: Rd7

Set if, before the shift, the MSB of Rd was set; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
lsl r18 ; Multiply r19:r18 by two
rol r19 ; r19:r18 is a signed or unsigned two-byte integer
brcs oneenc ; Branch if carry set
...
oneenc: nop ; Branch destination (do nothing)
```

Words: 1 (2 bytes) Cycles: 1

118 AVR Instruction Set =

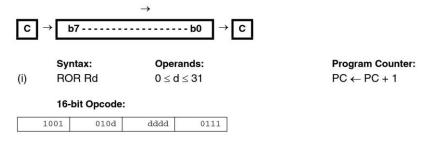


ROR – Rotate Right through Carry

Description:

Shifts all bits in Rd one place to the right. The C Flag is shifted into bit 7 of Rd. Bit 0 is shifted into the C Flag. This operation, combined with ASR, effectively divides multi-byte signed values by two. Combined with LSR it effectively divides multi-byte unsigned values by two. The Carry Flag can be used to round the result.

Operation:



Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С	
-	=	12 de	⇔	\Leftrightarrow	\Leftrightarrow	⇔	\Leftrightarrow	

S: $N \oplus V$, For signed tests.

V: N ⊕ C (For N and C after the shift)

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7• R6•R5• R4• R3•R2• R1• R0
Set if the result is \$00; cleared otherwise.

C: Rd0

Set if, before the shift, the LSB of Rd was set; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
lsr r19 ; Divide r19:r18 by two
ror r18 ; r19:r18 is an unsigned two-byte integer
brcc zeroenc1 ; Branch if carry cleared
asr r17 ; Divide r17:r16 by two
ror r16 ; r17:r16 is a signed two-byte integer
brcc zeroenc2 ; Branch if carry cleared
...
zeroenc1: nop ; Branch destination (do nothing)
...
```

ATMEL

119





zeroencl: nop ; Branch destination (do nothing)

Words: 1 (2 bytes)

Cycles: 1

120 AVR Instruction Set



SBC - Subtract with Carry

Description:

Subtracts two registers and subtracts with the C Flag and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd - Rr - C$

16-bit Opcode:

	0000	10rd	dddd	rrrr
1	0000	1014	aaaa	TITI

Status Register and Boolean Formula:

I	Т	Н	s	V	N	Z	С	
1-1	-	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	

H: Rd3• Rr3 + Rr3• R3 + R3 • Rd3

Set if there was a borrow from bit 3; cleared otherwise

- S: $N \oplus V$, For signed tests.
- V: Rd7 •Rr7 R7 +Rd7 •Rr7 •R7

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7• R6 •R5• R4• R3 •R2• R1• R0• Z

Previous value remains unchanged when the result is zero; cleared otherwise.

C: Rd7 •Rr7 •R7 •R7 +R7 •Rd7

Set if the absolute value of the contents of Rr plus previous carry is larger than the absolute value of the Rd; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

; Subtract r1:r0 from r3:r2
sub r2,r0 ; Subtract low byte
sbc r3,r1 ; Subtract with carry high byte

Words: 1 (2 bytes)

Cycles: 1



121





SBCI - Subtract Immediate with Carry

Description:

Subtracts a constant from a register and subtracts with the C Flag and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd - K - C$

 $\begin{array}{ccc} \text{Syntax:} & \text{Operands:} & \text{Program Counter:} \\ \text{(i)} & \text{SBCI Rd,K} & 16 \leq d \leq 31, \, 0 \leq K \leq 255 & \text{PC} \leftarrow \text{PC} + 1 \\ \end{array}$

16-bit Opcode:

0100	KKKK	dddd	KKKK

Status Register and Boolean Formula:

ı	Т	Н	s	V	N	Z	С	
-	-	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	\Leftrightarrow	

H: Rd3• K3 + K3• R3 + R3 • Rd3

Set if there was a borrow from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: Rd7 •K7• R7 +Rd7 •K7 •R7

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: R7• R6 •R5• R4• R3 •R2• R1• R0• Z

Previous value remains unchanged when the result is zero; cleared otherwise.

C: Rd7 •K7+ K7 • R7 +R7 •Rd7

Set if the absolute value of the constant plus previous carry is larger than the absolute value of Rd; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

; Subtract \$4F23 from r17:r16 subi r16,\$23 ; Subtract low byte

sbci r17,\$4F ; Subtract with carry high byte

Words: 1 (2 bytes)

Cycles: 1

122 AVR Instruction Set



SBI - Set Bit in I/O Register

Description:

 $Sets\ a\ specified\ bit\ in\ an\ I/O\ Register.\ This\ instruction\ operates\ on\ the\ lower\ 32\ I/O\ Registers\ -\ addresses\ 0-31.$

Operation:

(i) $I/O(A,b) \leftarrow 1$

16-bit Opcode:

1001	1010	AAAA	Abbb
1001	1010	I III III I	ADDD

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С	
-	-	-	-	-	-	-	-	

Example:

out \$1E,r0 ; Write EEPROM address sbi \$1C,0 ; Set read bit in EECR in r1,\$1D ; Read EEPROM data

Words : 1 (2 bytes)

Cycles : 2
Cycles XMEGA: 1
Cycles Reduced Core tinyAVR:1







SBIC - Skip if Bit in I/O Register is Cleared

Description:

This instruction tests a single bit in an I/O Register and skips the next instruction if the bit is cleared. This instruction operates on the lower 32 I/O Registers – addresses 0-31.

Operation:

(i) If I/O(A,b) = 0 then $PC \leftarrow PC + 2$ (or 3) else $PC \leftarrow PC + 1$

Syntax: Operands: Program Counter:

(i) SBIC A,b $0 \le A \le 31, \ 0 \le b \le 7 \\ PC \leftarrow PC + 1, \ Condition \ false - no \ skip \\ PC \leftarrow PC + 2, \ Skip \ a \ one \ word \ instruction$

PC ← PC + 3, Skip a two word instruction

16-bit Opcode:

1001	1001	AAAA	Abbb

Status Register (SREG) and Boolean Formula:

Ī	Т	Н	s	V	N	Z	С	
-	-	-	-	-	-	-	-	

Example:

e2wait: sbic \$1C,1 ; Skip next inst. if EEWE cleared

rjmp e2wait ; EEPROM write not finished
nop ; Continue (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false (no skip)

2 if condition is true (skip is executed) and the instruction skipped is 1 word

3 if condition is true (skip is executed) and the instruction skipped is 2 words

Cycles XMEGA: 2 if condition is false (no skip)

3 if condition is true (skip is executed) and the instruction skipped is 1 word

4 if condition is true (skip is executed) and the instruction skipped is 2 words

124 AVR Instruction Set



SBIS – Skip if Bit in I/O Register is Set

Description:

This instruction tests a single bit in an I/O Register and skips the next instruction if the bit is set. This instruction operates on the lower 32 I/O Registers – addresses 0-31.

Operation:

(i) If I/O(A,b) = 1 then $PC \leftarrow PC + 2$ (or 3) else $PC \leftarrow PC + 1$

Syntax: Operands: Program Counter:

(i) SBIS A,b $0 \le A \le 31, \ 0 \le b \le 7 \\ PC \leftarrow PC + 1, \ Condition \ false - no \ skip \\ PC \leftarrow PC + 2, \ Skip \ a \ one \ word \ instruction$

PC ← PC + 3, Skip a two word instruction

16-bit Opcode:

1001	1011	AAAA	Abbb
1001	1011	AAAA	ADDD

Status Register (SREG) and Boolean Formula:

I	T	Н	s	V	N	Z	С
-	-	-	-	-	-	-	-

Example:

waitset: sbis \$10,0 ; Skip next inst. if bit 0 in Port D set

rjmp waitset ; Bit not set

nop ; Continue (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false (no skip)

2 if condition is true (skip is executed) and the instruction skipped is 1 word

3 if condition is true (skip is executed) and the instruction skipped is 2 words

Cycles XMEGA: 2 if condition is false (no skip)

 ${\bf 3}$ if condition is true (skip is executed) and the instruction skipped is ${\bf 1}$ word

4 if condition is true (skip is executed) and the instruction skipped is 2 words







SBIW - Subtract Immediate from Word

Description:

Subtracts an immediate value (0-63) from a register pair and places the result in the register pair. This instruction operates on the upper four register pairs, and is well suited for operations on the Pointer Registers.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) Rd+1:Rd ← Rd+1:Rd - K

16-bit Opcode:

	1001	0111	KKdd	KKKK
--	------	------	------	------

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	Z	С	
_	_	_	⇔	\Leftrightarrow	\Leftrightarrow	⇔	⇔	

- S: $N \oplus V$, For signed tests.
- V: Rdh7 •R15

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R15

Set if MSB of the result is set; cleared otherwise.

- Z: R15• R14 •R13 •R12 •R11• R10• R9• R8• R7• R6 •R5• R4• R3 •R2• R1• R0 Set if the result is \$0000; cleared otherwise.
- C: R15• Rdh7

Set if the absolute value of K is larger than the absolute value of Rd; cleared otherwise.

R (Result) equals Rdh:Rdl after the operation (Rdh7-Rdh0 = R15-R8, Rdl7-Rdl0=R7-R0).

Example:

```
sbiw r25:r24,1 ; Subtract 1 from r25:r24
sbiw YH:YL,63 ; Subtract 63 from the Y-pointer(r29:r28)
```

Words: 1 (2 bytes)

Cycles: 2

126 AVR Instruction Set



SBR - Set Bits in Register

Description:

Sets specified bits in register Rd. Performs the logical ORI between the contents of register Rd and a constant mask K and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd \vee K$

16-bit Opcode:

0110	*******	5555	*******
0110	KKKK	aaaa	KKKK

Status Register (SREG) and Boolean Formula:

ı	Т	н	S	V	N	Z	С	
-	a—.		⇔	0	\Leftrightarrow	⇔]

S: $N \oplus V$, For signed tests.

V: 0

Cleared

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$ Set if the result is \$00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

sbr r16,3 ; Set bits 0 and 1 in r16
sbr r17,\$F0 ; Set 4 MSB in r17

Words: 1 (2 bytes)

Cycles: 1







SBRC - Skip if Bit in Register is Cleared

Description:

This instruction tests a single bit in a register and skips the next instruction if the bit is cleared.

Operation

(i) If Rr(b) = 0 then $PC \leftarrow PC + 2$ (or 3) else $PC \leftarrow PC + 1$

Syntax: Operands:

(i) SBRC Rr,b $0 \le r \le 31, 0 \le b \le 7$

 $PC \leftarrow PC + 1$, Condition false - no skip $PC \leftarrow PC + 2$, Skip a one word instruction $PC \leftarrow PC + 3$, Skip a two word instruction

Program Counter:

16-bit Opcode:

	1111	110r	rrrr	0bbb
--	------	------	------	------

Status Register (SREG) and Boolean Formula:

t	T	Н	S	V	N	Z	С
-	_	-	9-9	-	-	-	-

Example:

sub r0,r1 ; Subtract r1 from r0
sbrc r0,7 ; Skip if bit 7 in r0 cleared
sub r0,r1 ; Only executed if bit 7 in r0 not cleared
nop ; Continue (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false (no skip)

2 if condition is true (skip is executed) and the instruction skipped is 1 word 3 if condition is true (skip is executed) and the instruction skipped is 2 words

128



SBRS - Skip if Bit in Register is Set

Description:

This instruction tests a single bit in a register and skips the next instruction if the bit is set.

Operation

(i) If Rr(b) = 1 then $PC \leftarrow PC + 2$ (or 3) else $PC \leftarrow PC + 1$

Syntax: Operands:

(i) SBRS Rr,b $0 \le r \le 31, 0 \le b \le 7$

 $PC \leftarrow PC + 1$, Condition false - no skip $PC \leftarrow PC + 2$, Skip a one word instruction

Program Counter:

PC ← PC + 3, Skip a two word instruction

16-bit Opcode:

1111	1111	rrrr	0bbb
TTTT	TITE	TITI	dadu

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С
-	-	-	-	-	-	-	-

Example:

sub r0,r1 ; Subtract r1 from r0
sbrs r0,7 ; Skip if bit 7 in r0 set
neg r0 ; Only executed if bit 7 in r0 not set
nop ; Continue (do nothing)

Words: 1 (2 bytes)

Cycles: 1 if condition is false (no skip)

2 if condition is true (skip is executed) and the instruction skipped is 1 word 3 if condition is true (skip is executed) and the instruction skipped is 2 words







SEC - Set Carry Flag

Description:

Sets the Carry Flag (C) in SREG (Status Register).

Operation:

(i) C ← 1

(i)

Syntax:Operands:Program Counter:SECNone $PC \leftarrow PC + 1$

16-bit Opcode:

1001 0100 0000 1000

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С	
-	-	-	-	-	-	-	1	

O:

Carry Flag set

Example:

sec ; Set Carry Flag adc r0,r1 ; r0=r0+r1+1

Words: 1 (2 bytes)

Cycles: 1



Program Counter:

 $PC \leftarrow PC + 1$

SEH - Set Half Carry Flag

Description:

Sets the Half Carry (H) in SREG (Status Register).

Operation:

(i) H ← 1

Syntax: Operands:
(i) SEH None

16-bit Opcode:

1001 0100 0101 1000

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С
-	-	1	-	-	-	-	i-

H: 1

Half Carry Flag set

Example:

seh ; Set Half Carry Flag

Words: 1 (2 bytes)

Cycles: 1







SEI - Set Global Interrupt Flag

Description:

Sets the Global Interrupt Flag (I) in SREG (Status Register). The instruction following SEI will be executed before any pending interrupts.

Operation:

(i) I ← 1

16-bit Opcode:

	1001	0100	0111	1000
--	------	------	------	------

Status Register (SREG) and Boolean Formula:

1	Т	Н	s	V	N	z	С
1	-	-	-	_	-	-	-

l:

Global Interrupt Flag set

Example:

sei ; set global interrupt enable sleep ; enter sleep, waiting for interrupt

; note: will enter sleep before any pending interrupt(s)

Words: 1 (2 bytes)

Cycles: 1

132 AVR Instruction Set



Program Counter:

 $PC \leftarrow PC + 1$

SEN - Set Negative Flag

Description:

Sets the Negative Flag (N) in SREG (Status Register).

Operation:

(i) $N \leftarrow 1$

Syntax: Operands: (i) SEN None

16-bit Opcode:

1001 0100 0010 1000

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С
_	-	_	-	_	1	-	_

N: 1

Negative Flag set

Example:

add r2,r19 ; Add r19 to r2 sen ; Set Negative Flag

Words: 1 (2 bytes)

Cycles: 1







SER - Set all Bits in Register

Description:

Loads \$FF directly to register Rd.

Operation:

(i) $Rd \leftarrow \$FF$

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1110 1111 dddd 1111

Status Register (SREG) and Boolean Formula:

T.	Т	Н	S	V	N	Z	С
-	_	-	_	-	-	_	-

Example:

clr r16 ; Clear r16 ser r17 ; Set r17

out \$18,r16 ; Write zeros to Port B
nop ; Delay (do nothing)
out \$18,r17 ; Write ones to Port B

Words: 1 (2 bytes)

Cycles: 1

134 AVR Instruction Set



SES - Set Signed Flag

Description:

Sets the Signed Flag (S) in SREG (Status Register).

Operation:

(i) S ← 1

(i)

Syntax: Operands: SES None

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001 0100 0100 1000

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С
-	_	-	1	-	-	-	-

3:

Signed Flag set

Example:

add r2,r19 ; Add r19 to r2 ses ; Set Negative Flag

Words: 1 (2 bytes)

Cycles: 1







SET - Set T Flag

Description:

Sets the T Flag in SREG (Status Register).

Operation:

(i) T ← 1

(i)

Syntax: Operands: SET None

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001 0100 0110 1000

Status Register (SREG) and Boolean Formula:

ľ	Т	Н	S	V	N	Z	С	
-	1	-	_	-	-	_	-	

T: 1

T Flag set

Example:

set ; Set T Flag

Words: 1 (2 bytes)

Cycles: 1

136



SEV - Set Overflow Flag

Description:

Sets the Overflow Flag (V) in SREG (Status Register).

Operation:

(i) V ← 1

(i)

Syntax:Operands:Program Counter:SEVNone $PC \leftarrow PC + 1$

16-bit Opcode:

1001 0100 0011 1000

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С
-	j-	7-1	-	1	-	-	2-2

V:

Overflow Flag set

Example:

add r2,r19 ; Add r19 to r2 sev ; Set Overflow Flag

Words: 1 (2 bytes)

Cycles: 1





SEZ - Set Zero Flag

Description:

Sets the Zero Flag (Z) in SREG (Status Register).

Operation:

(i) $Z \leftarrow 1$

Syntax: Operands: (i) SEZ None

Program Counter:

 $PC \leftarrow PC + 1$

16-bit Opcode:

1001 0100 0001 1000

Status Register (SREG) and Boolean Formula:

1	Т	Н	S	V	N	Z	С
-	_	1 - 1	-	-	-	1	8-9

Z:

Zero Flag set

Example:

add r2,r19 ; Add r19 to r2 sez ; Set Zero Flag

Words: 1 (2 bytes)

Cycles: 1

138 AVR Instruction Set



SLEEP

Description:

This instruction sets the circuit in sleep mode defined by the MCU Control Register.

Operation:

Refer to the device documentation for detailed description of SLEEP usage.

16-bit Opcode:

1000	1000	0101	1001	
	1000	0101	1001	

Status Register (SREG) and Boolean Formula:

I	Т	Н	s	V	N	Z	С
-	-	-	-	-	-	-	-

Example:

 mov
 r0,r11
 ; Copy r11 to r0

 ldi
 r16,(1<<SE)</td>
 ; Enable sleep mode

 out
 MCUCR, r16

 sleep
 ; Put MCU in sleep mode

Words: 1 (2 bytes)

Cycles: 1







SPM – Store Program Memory

Description:

SPM can be used to erase a page in the Program memory, to write a page in the Program memory (that is already erased), and to set Boot Loader Lock bits. In some devices, the Program memory can be written one word at a time, in other devices an entire page can be programmed simultaneously after first filling a temporary page buffer. In all cases, the Program memory must be erased one page at a time. When erasing the Program memory, the RAMPZ and Z-register are used as page address. When writing the Program memory, the RAMPZ and Z-register are used as page or word address, and the R1:R0 register pair is used as data⁽¹⁾. When setting the Boot Loader Lock bits, the R1:R0 register pair is used as data. Refer to the device documentation for detailed description of SPM usage. This instruction can address the entire Program

The SPM instruction is not available in all devices. Refer to the device specific instruction set summary.

1. R1 determines the instruction high byte, and R0 determines the instruction low byte.

	Operation:	Comment:
(i)	$(RAMPZ:Z) \leftarrow \$ffff$	Erase Program memory page
(ii)	(RAMPZ:Z) ← R1:R0	Write Program memory word
(iii)	(RAMPZ:Z) ← R1:R0	Write temporary page buffer

(RAMPZ:Z) ← TEMP Write temporary page buffer to Program memory (iv) (v)

BLBITS ← R1:R0 Set Boot Loader Lock bits

Syntax: Operands: **Program Counter:** $PC \leftarrow PC + 1$ (i)-(v)SPM

16-bit Opcode:

	1001	0101	1110	1000
1	1001	0101	1110	1000

Status Register (SREG) and Boolean Formula:

1	T	Н	s	V	N	Z	С	
-	-	-	-	1000	-	_	_]

Example:

; This example shows SPM write of one page for devices with page write

; - the routine writes one page of data from RAM to Flash

; the first data location in RAM is pointed to by the Y-pointer

; the first data location in Flash is pointed to by the Z-pointer

; - error handling is not included

;- the routine must be placed inside the boot space

; (at least the do_spm sub routine)

;- registers used: r0, r1, temp1, temp2, looplo, loophi, spmcrval

; (temp1, temp2, looplo, loophi, spmcrval must be defined by the user)

; storing and restoring of registers is not included in the routine

; register usage can be optimized at the expense of code size

.equPAGESIZEB = PAGESIZE*2; PAGESIZEB is page size in BYTES, not words

.org SMALLBOOTSTART

write_page:

AVR Instruction Set 140



```
; page erase
 ldispmcrval, (1<<PGERS) + (1<<SPMEN)
 calldo_spm
 ;transfer data from RAM to Flash page buffer
 ldilooplo, low(PAGESIZEB);init loop variable
 ldiloophi, high(PAGESIZEB); not required for PAGESIZEB<=256
wrloop:ldr0, Y+
 ldr1, Y+
 ldispmcrval, (1<<SPMEN)
 calldo_spm
 adiwZH:ZL, 2
 sbiwloophi:looplo, 2;use subi for PAGESIZEB<=256
 brnewrloop
 ; execute page write
 subiZL, low(PAGESIZEB); restore pointer
 sbciZH, high(PAGESIZEB); not required for PAGESIZEB<=256
 ldispmcrval, (1<<PGWRT) + (1<<SPMEN)
 calldo_spm
 ; read back and check, optional
 ldilooplo, low(PAGESIZEB); init loop variable
 ldiloophi, high(PAGESIZEB); not required for PAGESIZEB<=256
 subiYL, low(PAGESIZEB); restore pointer
 sbciYH, high(PAGESIZEB)
rdloop:lpmr0, Z+
 ldr1, Y+
 cpser0, r1
 jmperror
 sbiwloophi:looplo, 2;use subi for PAGESIZEB<=256
 brnerdloop
 ;return
 ret
do spm:
 ;input: spmcrval determines SPM action
 ; disable interrupts if enabled, store status
 intemp2, SREG
 cli
 ; check for previous SPM complete
wait:intemp1, SPMCR
 sbrctemp1, SPMEN
 rjmpwait
 ;SPM timed sequence
 outSPMCR, spmcrval
 ;restore SREG (to enable interrupts if originally enabled)
 outSREG, temp2
```



0856I-AVR-07/10

141





ret

Words: 1 (2 bytes)

Cycles: depends on the operation

142 AVR Instruction Set



SPM #2- Store Program Memory

Description:

SPM can be used to erase a page in the Program memory and to write a page in the Program memory (that is already erased). An entire page can be programmed simultaneously after first filling a temporary page buffer. The Program memory must be erased one page at a time. When erasing the Program memory, the RAMPZ and Z-register are used as page address. When writing the Program memory, the RAMPZ and Z-register are used as page or word address, and the R1:R0 register pair is used as data⁽¹⁾.

Refer to the device documentation for detailed description of SPM usage. This instruction can address the entire Program memory.

Note: 1. R1 determines the instruction high byte, and R0 determines the instruction low byte.

	Operation:		Comment:
(i)	$(RAMPZ:Z) \leftarrow \$ffff$		Erase Program memory page
(ii)	(RAMPZ:Z) ← R1:R0		Load Page Buffer
(iii)	(RAMPZ:Z) ← BUFFER		Write Page Buffer to Program memory
(iv)	$(RAMPZ:Z) \leftarrow \$fff$	$Z \leftarrow Z + 2$	Erase Program memory page, Z post incremented
(v)	(RAMPZ:Z) ← R1:R0	$Z \leftarrow Z + 2$	Load Page Buffer, Z post incremented
(vi)	(RAMPZ:Z) ←BUFFER	$Z \leftarrow Z + 2$	Write Page Buffer to Program memory.

Z post incremented

16-bit Opcode:

(i)-(iii)	1001	0101	1110	1000
(iv)-(vi)	1001	0101	1111	1000

Status Register (SREG) and Boolean Formula:

ľ	Т	Н	s	V	N	Z	С	
-	(-)	-	-	-	-	-	-	

Example:

TBD

Words: 1 (2 bytes)

Cycles: depends on the operation





Features

- High-performance, Low-power Atmel® AVR® 8-bit Microcontroller
- **Advanced RISC Architecture**
 - 131 Powerful Instructions Most Single-clock Cycle Execution
 - 32 x 8 General Purpose Working Registers
 - Fully Static Operation
 - Up to 16 MIPS Throughput at 16 MHz
 - On-chip 2-cycle Multiplier
- · High Endurance Non-volatile Memory segments
 - 16 Kbytes of In-System Self-programmable Flash program memory
 - 512 Bytes EEPROM
 - 1 Kbyte Internal SRAM
 - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
 - Data retention: 20 years at 85°C/100 years at 25°C
 - Optional Boot Code Section with Independent Lock Bits In-System Programming by On-chip Boot Program True Read-While-Write Operation
 - Programming Lock for Software Security
- · JTAG (IEEE std. 1149.1 Compliant) Interface
 - Boundary-scan Capabilities According to the JTAG Standard
 - Extensive On-chip Debug Support
 - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
 - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
 - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture
 - Real Time Counter with Separate Oscillator
 - Four PWM Channels
 - 8-channel, 10-bit ADC
 - 8 Single-ended Channels
 - 7 Differential Channels in TQFP Package Only
 - 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
 - Byte-oriented Two-wire Serial Interface
 - Programmable Serial USART
 - Master/Slave SPI Serial Interface
 - Programmable Watchdog Timer with Separate On-chip Oscillator
 - On-chip Analog Comparator
- Special Microcontroller Features
 - Power-on Reset and Programmable Brown-out Detection
 - Internal Calibrated RC Oscillator
 - External and Internal Interrupt Sources
 - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
 - 32 Programmable I/O Lines
 - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
 - 2.7V 5.5V for ATmega16L
 - 4.5V 5.5V for ATmega16
- **Speed Grades**
 - 0 8 MHz for ATmega16L
- 0 16 MHz for ATmega16
 Power Consumption @ 1 MHz, 3V, and 25°C for ATmega16L
 - Active: 1.1 mA
 - Idle Mode: 0.35 mA
 - Power-down Mode: < 1 μA



8-bit **AVR**® Microcontroller with 16K Bytes **In-System Programmable** Flash

ATmega16 ATmega16L

Rev. 2466T-AVR-07/10

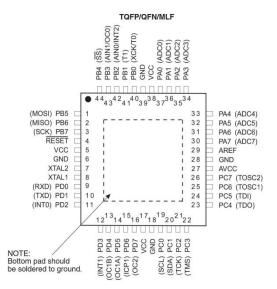




Pin Configurations

Figure 1. Pinout ATmega16

PDIP												
1												
(XCK/T0) PB0	1	40	PA0 (ADC0)									
(T1) PB1 🗆	2	39	PA1 (ADC1)									
(INT2/AIN0) PB2	3	38	PA2 (ADC2)									
(OC0/AIN1) PB3	4	37	PA3 (ADC3)									
(SS) PB4 □	5	36	PA4 (ADC4)									
(MOSI) PB5	6	35	PA5 (ADC5)									
(MISO) PB6	7	34	PA6 (ADC6)									
(SCK) PB7	8	33	PA7 (ADC7)									
RESET	9	32	AREF									
VCC 🗆	10	31	GND									
GND □	11	30	□ AVCC									
XTAL2	12	29	□ PC7 (TOSC2)									
XTAL1	13	28	PC6 (TOSC1)									
(RXD) PD0	14	27	□ PC5 (TDI)									
(TXD) PD1	15	26	□ PC4 (TDO)									
(INT0) PD2	16	25	□ PC3 (TMS)									
(INT1) PD3	17	24	□ PC2 (TCK)									
(OC1B) PD4	18	23	□ PC1 (SDA)									
(OC1A) PD5	19	22	□ PC0 (SCL)									
(ICP1) PD6	20	21	□ PD7 (OC2)									



Disclaimer

Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

AIMEL

2

2466T-AVR-07/10

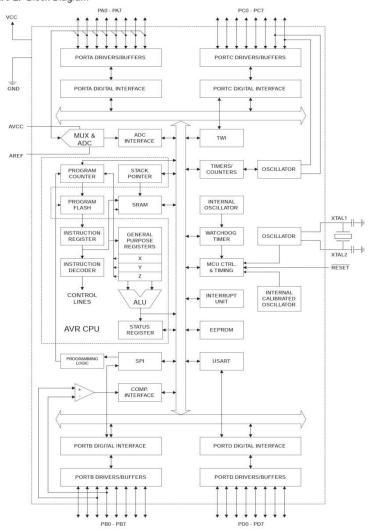


Overview

The ATmega16 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega16 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 2. Block Diagram





2466T-AVR-07/10

3



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega16 provides the following features: 16 Kbytes of In-System Programmable Flash Program memory with Read-While-Write capabilities, 512 bytes EEPROM, 1 Kbyte SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for Boundaryscan, On-chip Debugging support and programming, three flexible Timer/Counters with compare modes, Internal and External Interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain (TQFP package only), a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the USART, Two-wire interface, A/D Converter, SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next External Interrupt or Hardware Reset. In Power-save mode, the Asynchronous Timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run.

The device is manufactured using Atmel's high density nonvolatile memory technology. The Onchip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega16 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The ATmega16 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

Pin Descriptions

VCC Digital supply voltage.

GND Ground.

Port A (PA7..PA0) Port A serves as the analog inputs to the A/D Converter.

Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.



4

2466T-AVR-07/10



Port B (PB7..PB0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATmega16 as listed on page 58.

Port C (PC7..PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs.

Port C also serves the functions of the JTAG interface and other special features of the ATmega16 as listed on page 61.

Port D (PD7..PD0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATmega16 as listed on page 63

RESET

Reset Input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 15 on page 38. Shorter pulses are not guaranteed to generate a reset.

XTAL1 Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

XTAL2 Output from the inverting Oscillator amplifier.

AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally con-

nected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC}

through a low-pass filter.

AREF is the analog reference pin for the A/D Converter.





Resources A comprehensive set of development tools, application notes and datasheets are available for

download on http://www.atmel.com/avr.

Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85° C or 100 years at 25° C. **Data Retention**



2466T-AVR-07/10

6



About Code Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. These code examples assume that the part specific header file is included before compilation. Be aware that not all C Compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Please confirm with the C Compiler documentation for more details.



2466T-AVR-07/10

7



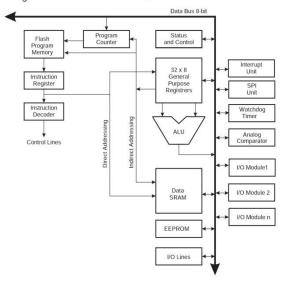
AVR CPU Core

Introduction

This section discusses the AVR core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

Architectural Overview

Figure 3. Block Diagram of the AVR MCU Architecture



In order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In-System Reprogrammable Flash memory.

The fast-access Register File contains 32×8 -bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in Flash Program memory. These added function registers are the 16-bit X-register, Y-register, and Z-register, described later in this section.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.



8

2466T-AVR-07/10



Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16-bit or 32-bit instruction.

Program Flash memory space is divided in two sections, the Boot program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM instruction that writes into the Application Flash memory section must reside in the Boot Program section.

During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the reset routine (before subroutines or interrupts are executed). The Stack Pointer SP is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional global interrupt enable bit in the Status Register. All interrupts have a separate interrupt vector in the interrupt vector table. The interrupts have priority in accordance with their interrupt vector position. The lower the interrupt vector address, the higher the priority.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, SPI, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the Register File, \$20 - \$5F.

ALU - Arithmetic Logic Unit

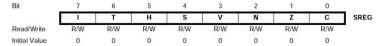
The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the "Instruction Set" section for a detailed description.

Status Register

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set Reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

The Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

The AVR Status Register - SREG - is defined as:



• Bit 7 - I: Global Interrupt Enable

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.



9

2466T-AVR-07/10



· Bit 6 - T: Bit Copy Storage

The Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit. A bit from a register in the Register File can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

· Bit 5 - H: Half Carry Flag

The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half Carry is useful in BCD arithmetic. See the "Instruction Set Description" for detailed information.

• Bit 4 - S: Sign Bit, S = N ⊕ V

The S-bit is always an exclusive or between the Negative Flag N and the Two's Complement Overflow Flag V. See the "Instruction Set Description" for detailed information.

• Bit 3 - V: Two's Complement Overflow Flag

The Two's Complement Overflow Flag V supports two's complement arithmetics. See the "Instruction Set Description" for detailed information.

· Bit 2 - N: Negative Flag

The Negative Flag N indicates a negative result in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.

· Bit 1 - Z: Zero Flag

The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.

· Bit 0 - C: Carry Flag

The Carry Flag C indicates a carry in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.





General Purpose Register File

The Register File is optimized for the AVR Enhanced RISC instruction set. In order to achieve the required performance and flexibility, the following input/output schemes are supported by the Register File:

- · One 8-bit output operand and one 8-bit result input
- · Two 8-bit output operands and one 8-bit result input
- Two 8-bit output operands and one 16-bit result input
- One 16-bit output operand and one 16-bit result input

Figure 4 shows the structure of the 32 general purpose working registers in the CPU.

Figure 4. AVR CPU General Purpose Working Registers

General

Purpose

Working

Registers

Addr R0 \$00 R1 R2 \$02 R13 \$0D \$0E R14 R15 \$OF R16 \$10 R17 \$11 R26 \$1A X-register Low Byte R27 \$1B X-register High Byte R28 \$1C Y-register Low Byte \$1D Y-register High Byte R29 R30 \$1E Z-register Low Byte R31 Z-register High Byte

Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 4, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y-, and Z-pointer Registers can be set to index any register in the file.



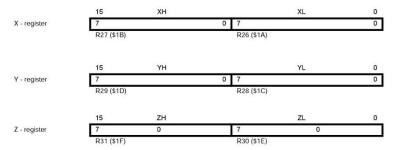


——— ATmega16(L)

The X-register, Y-register and Z-register

The registers R26..R31 have some added functions to their general purpose usage. These registers are 16-bit address pointers for indirect addressing of the Data Space. The three indirect address registers X, Y, and Z are defined as described in Figure 5.

Figure 5. The X-register, Y-register, and Z-register



In the different addressing modes these address registers have functions as fixed displacement, automatic increment, and automatic decrement (see the Instruction Set Reference for details).

Stack Pointer

The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The Stack Pointer Register always points to the top of the Stack. Note that the Stack is implemented as growing from higher memory locations to lower memory locations. This implies that a Stack PUSH command decreases the Stack Pointer. If software reads the Program Counter from the Stack after a call or an interrupt, unused bits (15:13) should be masked out.

The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above \$60. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call or interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

The AVR Stack Pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation dependent. Note that the data space in some implementations of the AVR architecture is so small that only SPL is needed. In this case, the SPH Register will not be present.

Bit	15	14	13	12	11	10	9	8	
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	-
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	



12

2466T-AVR-07/10



Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The AVR CPU is driven by the CPU clock clk_{CPU}, directly generated from the selected clock source for the chip. No internal clock division is used.

Figure 6 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access Register File concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Figure 6. The Parallel Instruction Fetches and Instruction Executions

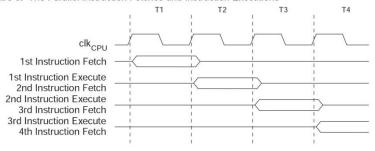
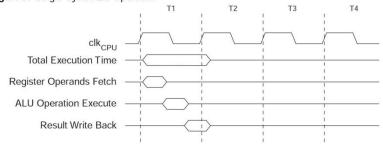


Figure 7 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 7. Single Cycle ALU Operation



Reset and Interrupt Handling

The AVR provides several different interrupt sources. These interrupts and the separate reset vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt. Depending on the Program Counter value, interrupts may be automatically disabled when Boot Lock bits BLB02 or BLB12 are programmed. This feature improves software security. See the section "Memory Programming" on page 259 for details.

The lowest addresses in the program memory space are by default defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in "Interrupts" on page 45. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INTO – the External Interrupt Request



13



0. The Interrupt Vectors can be moved to the start of the Boot Flash section by setting the IVSEL bit in the General Interrupt Control Register (GICR). Refer to "Interrupts" on page 45 for more information. The Reset Vector can also be moved to the start of the boot Flash section by programming the BOOTRST Fuse, see "Boot Loader Support – Read-While-Write Self-Programming" on page 246.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared and all interrupts are disabled. The user software can write logic one to the I-bit to enable nested interrupts. All enabled interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a Return from Interrupt instruction – RETI – is executed.

There are basically two types of interrupts. The first type is triggered by an event that sets the Interrupt Flag. For these interrupts, the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, and hardware clears the corresponding Interrupt Flag. Interrupt Flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared. If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the Interrupt Flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software. Similarly, if one or more interrupt conditions occur while the Global Interrupt Enable bit is cleared, the corresponding Interrupt Flag(s) will be set and remembered until the global interrupt enable bit is set, and will then be executed by order of priority.

The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have Interrupt Flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered.

When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

Note that the Status Register is not automatically stored when entering an interrupt routine, nor restored when returning from an interrupt routine. This must be handled by software.

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence.

```
Assembly Code Example
   in r16, SREG
                     ; store SREG value
   cli ; disable interrupts during timed sequence
   sbi EECR, EEMWE
                    ; start EEPROM write
   sbi EECR, EEWE
   out SREG, r16
                     ; restore SREG value (I-bit)
C Code Example
   char cSREG;
   cSREG = SREG; /* store SREG value */
   /* disable interrupts during timed sequence */
   EECR |= (1<<EEMWE); /* start EEPROM write */
   EECR |= (1<<EEWE);
   SREG = cSREG; /* restore SREG value (I-bit) */
```



14



When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in this example.

```
Assembly Code Example

sei ; set global interrupt enable

sleep; enter sleep, waiting for interrupt
; note: will enter sleep before any pending
; interrupt(s)

C Code Example

_SEI(); /* set global interrupt enable */
_SLEEP(); /* enter sleep, waiting for interrupt */
/* note: will enter sleep before any pending interrupt(s) */
```

Interrupt Response Time

The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum. After four clock cycles the program vector address for the actual interrupt handling routine is executed. During this four clock cycle period, the Program Counter is pushed onto the Stack. The vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by four clock cycles. This increase comes in addition to the start-up time from the selected sleep mode.

A return from an interrupt handling routine takes four clock cycles. During these four clock cycles, the Program Counter (two bytes) is popped back from the Stack, the Stack Pointer is incremented by two, and the I-bit in SREG is set.





AVR ATmega16 Memories

This section describes the different memories in the ATmega16. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, the ATmega16 features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

In-System Reprogrammable Flash Program Memory

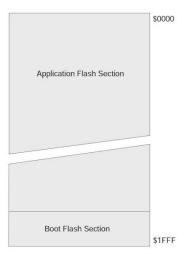
The ATmega16 contains 16 Kbytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as $8K \times 16$. For software security, the Flash Program memory space is divided into two sections, Boot Program section and Application Program section.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega16 Program Counter (PC) is 13 bits wide, thus addressing the 8K program memory locations. The operation of Boot Program section and associated Boot Lock bits for software protection are described in detail in "Boot Loader Support – Read-While-Write Self-Programming" on page 246. "Memory Programming" on page 259 contains a detailed description on Flash data serial downloading using the SPI pins or the JTAG interface.

Constant tables can be allocated within the entire program memory address space (see the LPM – Load Program Memory Instruction Description).

Timing diagrams for instruction fetch and execution are presented in "Instruction Execution Timing" on page 13.

Figure 8. Program Memory Map







SRAM Data Memory

Figure 9 shows how the ATmega16 SRAM Memory is organized.

The lower 1120 Data Memory locations address the Register File, the I/O Memory, and the internal data SRAM. The first 96 locations address the Register File and I/O Memory, and the next 1024 locations address the internal data SRAM.

The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement, and Indirect with Post-increment. In the Register File, registers R26 to R31 feature the indirect addressing pointer registers.

The direct addressing reaches the entire data space.

The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y-register or Z-register.

When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented.

The 32 general purpose working registers, 64 I/O Registers, and the 1024 bytes of internal data SRAM in the ATmega16 are all accessible through all these addressing modes. The Register File is described in "General Purpose Register File" on page 11.

Figure 9. Data Memory Map

Register File	Data Address Space
R0	\$0000
R1	\$0001
R2	\$0002
R29	\$001D
R30	\$001E
R31	\$001F
I/O Registers	
\$00	\$0020
\$01	\$0021
\$02	\$0022

\$3D	\$005D
\$3E	\$005E
\$3F	\$005F
	Internal SRAM
	\$0060

Internal SRAM \$0060 \$0061 ... \$045E

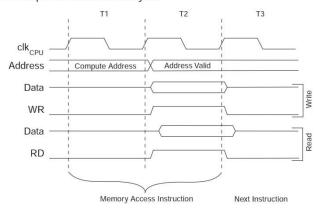




Data Memory Access Times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two clk_{CPU} cycles as described in Figure 10.

Figure 10. On-chip Data SRAM Access Cycles



EEPROM Data Memory

The ATmega16 contains 512 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

For a detailed description of SPI, JTAG, and Parallel data downloading to the EEPROM, see page 273, page 278, and page 262, respectively.

EEPROM Read/Write Access

The EEPROM Access Registers are accessible in the I/O space.

The write access time for the EEPROM is given in Table 1. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, V_{CC} is likely to rise or fall slowly on Power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. See "Preventing EEPROM Corruption" on page 22 for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this.

When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.



18



The EEPROM Address Register – EEARH and EEARL

Bit	15	14	13	12	11	10	9	8	
	-	-	-	-	-	-	-	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	•
Read/Write	R	R	R	R	R	R	R	R/W	
	R/W								
Initial Value	0	0	0	0	0	0	0	X	
	X	X	X	X	X	X	X	X	

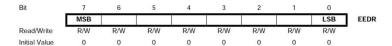
· Bits 15..9 - Res: Reserved Bits

These bits are reserved bits in the ATmega16 and will always read as zero.

• Bits 8..0 - EEAR8..0: EEPROM Address

The EEPROM Address Registers - EEARH and EEARL - specify the EEPROM address in the 512 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 511. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

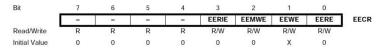
The EEPROM Data Register – EEDR



• Bits 7..0 - EEDR7.0: EEPROM Data

For the EEPROM write operation, the EEDR Register contains the data to be written to the EEPROM in the address given by the EEAR Register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

The EEPROM Control Register – EECR



· Bits 7..4 - Res: Reserved Bits

These bits are reserved bits in the ATmega16 and will always read as zero.

• Bit 3 - EERIE: EEPROM Ready Interrupt Enable

Writing EERIE to one enables the EEPROM Ready Interrupt if the I bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready interrupt generates a constant interrupt when EEWE is cleared.

• Bit 2 - EEMWE: EEPROM Master Write Enable

The EEMWE bit determines whether setting EEWE to one causes the EEPROM to be written. When EEMWE is set, setting EEWE within four clock cycles will write data to the EEPROM at the selected address If EEMWE is zero, setting EEWE will have no effect. When EEMWE has been written to one by software, hardware clears the bit to zero after four clock cycles. See the description of the EEWE bit for an EEPROM write procedure.



19



· Bit 1 - EEWE: EEPROM Write Enable

The EEPROM Write Enable Signal EEWE is the write strobe to the EEPROM. When address and data are correctly set up, the EEWE bit must be written to one to write the value into the EEPROM. The EEMWE bit must be written to one before a logical one is written to EEWE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 3 and 4 is not essential):

- 1. Wait until EEWE becomes zero.
- 2. Wait until SPMEN in SPMCR becomes zero.
- Write new EEPROM address to EEAR (optional).
- Write new EEPROM data to EEDR (optional).
- 5. Write a logical one to the EEMWE bit while writing a zero to EEWE in EECR.
- 6. Within four clock cycles after setting EEMWE, write a logical one to EEWE.

The EEPROM can not be programmed during a CPU write to the Flash memory. The software must check that the Flash programming is completed before initiating a new EEPROM write. Step 2 is only relevant if the software contains a Boot Loader allowing the CPU to program the Flash. If the Flash is never being updated by the CPU, step 2 can be omitted. See "Boot Loader Support – Read-While-Write Self-Programming" on page 246 for details about boot programming.

Caution: An interrupt between step 5 and step 6 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM Access, the EEAR or EEDR reGister will be modified, causing the interrupted EEPROM Access to fail. It is recommended to have the Global Interrupt Flag cleared during all the steps to avoid these problems.

When the write access time has elapsed, the EEWE bit is cleared by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEWE has been set, the CPU is halted for two cycles before the next instruction is executed.

· Bit 0 - EERE: EEPROM Read Enable

The EEPROM Read Enable Signal – EERE – is the read strobe to the EEPROM. When the correct address is set up in the EEAR Register, the EERE bit must be written to a logic one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed.

The user should poll the EEWE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR Register.

The calibrated Oscillator is used to time the EEPROM accesses. Table 1 lists the typical programming time for EEPROM access from the CPU.

Table 1. EEPROM Programming Time

Symbol	Number of Calibrated RC Oscillator Cycles ⁽¹⁾	Typ Programming Time
EEPROM write (from CPU)	8448	8.5 ms

Note: 1. Uses 1 MHz clock, independent of CKSEL Fuse setting.

The following code examples show one assembly and one C function for writing to the EEPROM. The examples assume that interrupts are controlled (for example by disabling interrupts globally) so that no interrupts will occur during execution of these functions. The examples

AMEL

20



also assume that no Flash Boot Loader is present in the software. If such code is present, the EEPROM write function must also wait for any ongoing SPM command to finish.

```
Assembly Code Example
   EEPROM_write:
     ; Wait for completion of previous write
     sbic EECR, EEWE
    rjmp EEPROM_write
     ; Set up address (r18:r17) in address register
     out EEARH, r18
     out EEARL, r17
     ; Write data (r16) to data register
     out EEDR, r16
     ; Write logical one to EEMWE
     sbi EECR, EEMWE
     ; Start eeprom write by setting EEWE
     sbi EECR, EEWE
     ret
C Code Example
    void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
     /* Wait for completion of previous write */
     while(EECR & (1<<EEWE))
     /* Set up address and data registers */
     EEAR = uiAddress;
     EEDR = ucData;
     /* Write logical one to EEMWE */
     EECR |= (1<<EEMWE);
     /* Start eeprom write by setting EEWE */
     EECR |= (1<<EEWE);
   }
```



21



The next code examples show assembly and C functions for reading the EEPROM. The examples assume that interrupts are controlled so that no interrupts will occur during execution of these functions.

```
Assembly Code Example
   EEPROM read:
     ; Wait for completion of previous write
     sbic EECR, EEWE
     rimp EEPROM read
     ; Set up address (r18:r17) in address register
     out EEARH, r18
     out EEARL, r17
     ; Start eeprom read by writing EERE
     sbi EECR, EERE
     ; Read data from data register
     in r16,EEDR
     ret
C Code Example
   unsigned char EEPROM_read(unsigned int uiAddress)
     /* Wait for completion of previous write */
     while(EECR & (1<<EEWE))
     /* Set up address register */
     EEAR = uiAddress:
     /* Start eeprom read by writing EERE */
     EECR |= (1<<EERE);
     /* Return data from data register */
     return EEDR;
```

EEPROM Write During Power-down Sleep Mode When entering Power-down Sleep mode while an EEPROM write operation is active, the EEPROM write operation will continue, and will complete before the Write Access time has passed. However, when the write operation is completed, the Oscillator continues running, and as a consequence, the device does not enter Power-down entirely. It is therefore recommended to verify that the EEPROM write operation is completed before entering Power-down.

Preventing EEPROM Corruption

During periods of low V_{CC} , the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board level systems using EEPROM, and the same design solutions should be applied.

An EEPROM data corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage is too low.



22



——— ATmega16(L)

EEPROM data corruption can easily be avoided by following this design recommendation:

Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD). If the detection level of the internal BOD does not match the needed detection level, an external low $V_{\rm CC}$ Reset Protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.

I/O Memory

The I/O space definition of the ATmega16 is shown in "Register Summary" on page 331.

All ATmega16 I/Os and peripherals are placed in the I/O space. The I/O locations are accessed by the IN and OUT instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range \$00 - \$1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the Instruction Set section for more details. When using the I/O specific commands IN and OUT, the I/O addresses \$00 - \$3F must be used. When addressing I/O Registers as data space using LD and ST instructions, \$20 must be added to these addresses.

For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

Some of the Status Flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O Register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers \$00 to \$1F only.

The I/O and Peripherals Control Registers are explained in later sections.



23

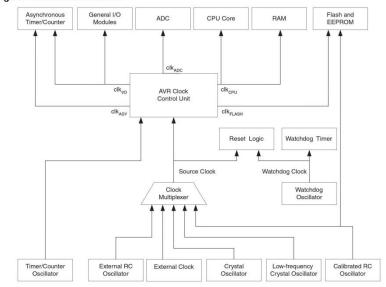


System Clock and Clock Options

Clock Systems and their Distribution

Figure 11 presents the principal clock systems in the AVR and their distribution. All of the clocks need not be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes, as described in "Power Management and Sleep Modes" on page 32. The clock systems are detailed Figure 11.

Figure 11. Clock Distribution



CPU Clock - clk_{CPU}

The CPU clock is routed to parts of the system concerned with operation of the AVR core. Examples of such modules are the General Purpose Register File, the Status Register and the data memory holding the Stack Pointer. Halting the CPU clock inhibits the core from performing general operations and calculations.

I/O Clock - clk_{I/O}

The I/O clock is used by the majority of the I/O modules, like Timer/Counters, SPI, and USART. The I/O clock is also used by the External Interrupt module, but note that some external interrupts are detected by asynchronous logic, allowing such interrupts to be detected even if the I/O clock is halted. Also note that address recognition in the TWI module is carried out asynchronously when clk_{I/O} is halted, enabling TWI address reception in all sleep modes.

Flash Clock - clk_{FLASH}

The Flash clock controls operation of the Flash interface. The Flash clock is usually active simultaneously with the CPU clock.

Asynchronous Timer Clock – clk_{ASY} The Asynchronous Timer clock allows the Asynchronous Timer/Counter to be clocked directly from an external 32 kHz clock crystal. The dedicated clock domain allows using this Timer/Counter as a real-time counter even when the device is in sleep mode.



24



ADC Clock - clk_{ADC}

The ADC is provided with a dedicated clock domain. This allows halting the CPU and I/O clocks in order to reduce noise generated by digital circuitry. This gives more accurate ADC conversion results.

Clock Sources

The device has the following clock source options, selectable by Flash Fuse bits as shown below. The clock from the selected source is input to the AVR clock generator, and routed to the appropriate modules.

Table 2. Device Clocking Options Select(1)

Device Clocking Option	CKSEL30
External Crystal/Ceramic Resonator	1111 - 1010
External Low-frequency Crystal	1001
External RC Oscillator	1000 - 0101
Calibrated Internal RC Oscillator	0100 - 0001
External Clock	0000

Note: 1. For all fuses "1" means unprogrammed while "0" means programmed.

The various choices for each clocking option is given in the following sections. When the CPU wakes up from Power-down or Power-save, the selected clock source is used to time the startup, ensuring stable Oscillator operation before instruction execution starts. When the CPU starts from Reset, there is as an additional delay allowing the power to reach a stable level before commencing normal operation. The Watchdog Oscillator is used for timing this real-time part of the start-up time. The number of WDT Oscillator cycles used for each time-out is shown in Table 3. The frequency of the Watchdog Oscillator is voltage dependent as shown in "ATmega16 Typical Characteristics" on page 299.

Table 3. Number of Watchdog Oscillator Cycles

Typ Time-out (V _{CC} = 5.0V)	Typ Time-out (V _{CC} = 3.0V)	Number of Cycles
4.1 ms	4.3 ms	4K (4,096)
65 ms	69 ms	64K (65,536)

Default Clock Source

The device is shipped with CKSEL = "0001" and SUT = "10". The default clock source setting is therefore the 1 MHz Internal RC Oscillator with longest startup time. This default setting ensures that all users can make their desired clock source setting using an In-System or Parallel Programmer.

Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, as shown in Figure 12. Either a quartz crystal or a ceramic resonator may be used. The CKOPT Fuse selects between two different Oscillator amplifier modes. When CKOPT is programmed, the Oscillator output will oscillate will a full rail-to-rail swing on the output. This mode is suitable when operating in a very noisy environment or when the output from XTAL2 drives a second clock buffer. This mode has a wide frequency range. When CKOPT is unprogrammed, the Oscillator has a smaller output swing. This reduces power consumption considerably. This mode has a limited frequency range and it can not be used to drive other clock buffers.

For resonators, the maximum frequency is 8 MHz with CKOPT unprogrammed and 16 MHz with CKOPT programmed. C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator in use, the amount of stray capacitance, and the electromagnetic noise of the environment. Some initial guidelines for

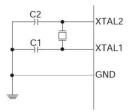


25



choosing capacitors for use with crystals are given in Table 4. For ceramic resonators, the capacitor values given by the manufacturer should be used.

Figure 12. Crystal Oscillator Connections



The Oscillator can operate in three different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL3..1 as shown in Table 4.

Table 4. Crystal Oscillator Operating Modes

СКОРТ	CKSEL31	Frequency Range (MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
1	101 ⁽¹⁾	0.4 - 0.9	n-
1	110	0.9 - 3.0	12 - 22
1	111	3.0 - 8.0	12 - 22
0	101, 110, 111	1.0 ≤	12 - 22

Note: 1. This option should not be used with crystals, only with ceramic resonators.



The CKSEL0 Fuse together with the SUT1..0 Fuses select the start-up times as shown in Table 5.

Table 5. Start-up Times for the Crystal Oscillator Clock Selection

CKSEL0	SUT10	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V _{CC} = 5.0V)	Recommended Usage
0	00	258 CK ⁽¹⁾	4.1 ms	Ceramic resonator, fast rising power
0	01	258 CK ⁽¹⁾	65 ms	Ceramic resonator, slowly rising power
0	10	1K CK ⁽²⁾		Ceramic resonator, BOD enabled
0	11	1K CK ⁽²⁾	4.1 ms	Ceramic resonator, fast rising power
1	00	1K CK ⁽²⁾	65 ms	Ceramic resonator, slowly rising power
1	01	16K CK	=	Crystal Oscillator, BOD enabled
1	10	16K CK	4.1 ms	Crystal Oscillator, fast rising power
1	11	16K CK	65 ms	Crystal Oscillator, slowly rising power

Notes: 1. These options should only be used when not operating close to the maximum frequency of the device, and only if frequency stability at start-up is not important for the application. These options are not suitable for crystals.

These options are intended for use with ceramic resonators and will ensure frequency stability at start-up. They can also be used with crystals when not operating close to the maximum frequency of the device, and if frequency stability at start-up is not important for the application.



Low-frequency Crystal Oscillator

To use a 32.768 kHz watch crystal as the clock source for the device, the Low-frequency Crystal Oscillator must be selected by setting the CKSEL Fuses to "1001". The crystal should be connected as shown in Figure 12. By programming the CKOPT Fuse, the user can enable internal capacitors on XTAL1 and XTAL2, thereby removing the need for external capacitors. The internal capacitors have a nominal value of 36 pF.

When this Oscillator is selected, start-up times are determined by the SUT Fuses as shown in Table 6.

Table 6. Start-up Times for the Low-frequency Crystal Oscillator Clock Selection

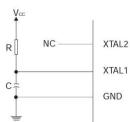
SUT10	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V _{CC} = 5.0V)	Recommended Usage
00	1K CK ⁽¹⁾	4.1 ms	Fast rising power or BOD enabled
01	1K CK ⁽¹⁾	65 ms	Slowly rising power
10	32K CK	65 ms	Stable frequency at start-up
11		Reserve	ed

Note: 1. These options should only be used if frequency stability at start-up is not important for the application.

External RC Oscillator

For timing insensitive applications, the external RC configuration shown in Figure 13 can be used. The frequency is roughly estimated by the equation f = 1/(3RC). C should be at least 22 pF. By programming the CKOPT Fuse, the user can enable an internal 36 pF capacitor between XTAL1 and GND, thereby removing the need for an external capacitor. For more information on Oscillator operation and details on how to choose R and C, refer to the External RC Oscillator application note.

Figure 13. External RC Configuration



The Oscillator can operate in four different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL3..0 as shown in Table 7.

Table 7. External RC Oscillator Operating Modes

CKSEL30	Frequency Range (MHz)
0101	0.1 ≤ 0.9
0110	0.9 - 3.0
0111	3.0 - 8.0
1000	8.0 - 12.0



28



When this Oscillator is selected, start-up times are determined by the SUT Fuses as shown in Table 8.

Table 8. Start-up Times for the External RC Oscillator Clock Selection

SUT10	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V _{CC} = 5.0V)	Recommended Usage
00	18 CK	<u> </u>	BOD enabled
01	18 CK	4.1 ms	Fast rising power
10	18 CK	65 ms	Slowly rising power
11	6 CK ⁽¹⁾	4.1 ms	Fast rising power or BOD enabled

Note: 1. This option should not be used when operating close to the maximum frequency of the device.

Calibrated Internal RC Oscillator

The Calibrated Internal RC Oscillator provides a fixed 1.0 MHz, 2.0 MHz, 4.0 MHz, or 8.0 MHz clock. All frequencies are nominal values at 5V and $25\,^{\circ}\text{C}$. This clock may be selected as the system-clock by programming the CKSEL Fuses as shown in Table 9. If selected, it will operate with no external components. The CKOPT Fuse should always be unpro-grammed when using this clock option. During Reset, hardware loads the calibration byte into the OSCCAL Register and thereby automatically calibrates the RC Oscillator. At 5V, $25\,^{\circ}\text{C}$ and 1.0 MHz, 2.0 MHz, 4.0 MHz or 8.0 MHz Oscillator frequency selected, this calibration gives a frequency within $\pm 3\%$ of the nominal frequency. Using calibration methods as described in application notes available at www.atmel.com/avr it is possible to achieve $\pm 1\%$ accuracy at any given V_{CC} and Temperature. When this Oscillator is used as the Chip Clock, the Watchdog Oscillator will still be used for the Watchdog Timer and for the reset time-out. For more information on the pre-programmed calibration value, see the section "Calibration Byte" on page 261.

Table 9. Internal Calibrated RC Oscillator Operating Modes

CKSEL30	Nominal Frequency (MHz)
0001 ⁽¹⁾	1.0
0010	2.0
0011	4.0
0100	8.0

Note: 1. The device is shipped with this option selected.

When this Oscillator is selected, start-up times are determined by the SUT Fuses as shown in Table 10. XTAL1 and XTAL2 should be left unconnected (NC).

Table 10. Start-up Times for the Internal Calibrated RC Oscillator Clock Selection

SUT10	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V _{CC} = 5.0V)	Recommended Usage	
00	6 CK	-	BOD enabled	
01	6 CK	4.1 ms	Fast rising power	
10 ⁽¹⁾	6 CK	65 ms	Slowly rising power	
11		Reserved		

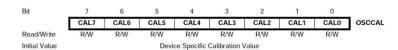
Note: 1. The device is shipped with this option selected.



29



Oscillator Calibration Register – OSCCAL



· Bits 7..0 - CAL7..0: Oscillator Calibration Value

Writing the calibration byte to this address will trim the Internal Oscillator to remove process variations from the Oscillator frequency. This is done automatically during Chip Reset. When OSCCAL is zero, the lowest available frequency is chosen. Writing non-zero values to this register will increase the frequency of the Internal Oscillator. Writing \$FF to the register gives the highest available frequency. The calibrated Oscillator is used to time EEPROM and Flash access. If EEPROM or Flash is written, do not calibrate to more than 10% above the nominal frequency. Otherwise, the EEPROM or Flash write may fail. Note that the Oscillator is intended for calibration to 1.0 MHz, 2.0 MHz, 4.0 MHz, or 8.0 MHz. Tuning to other values is not guaranteed, as indicated in Table 11.

Table 11. Internal RC Oscillator Frequency Range.

OSCCAL Value	Min Frequency in Percentage of Nominal Frequency (%)	Max Frequency in Percentage of Nominal Frequency (%)			
\$00	50	100			
\$7F	75	150			
\$FF	100	200			

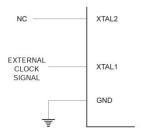




External Clock

To drive the device from an external clock source, XTAL1 should be driven as shown in Figure 14. To run the device on an external clock, the CKSEL Fuses must be programmed to "0000". By programming the CKOPT Fuse, the user can enable an internal 36 pF capacitor between XTAL1 and GND.

Figure 14. External Clock Drive Configuration



When this clock source is selected, start-up times are determined by the SUT Fuses as shown in Table 12.

Table 12. Start-up Times for the External Clock Selection

SUT10	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V _{CC} = 5.0V)	Recommended Usage
00	6 CK	-	BOD enabled
01	6 CK	4.1 ms	Fast rising power
10	6 CK	65 ms	Slowly rising power
11		Reserved	

When applying an external clock, it is required to avoid sudden changes in the applied clock frequency to ensure stable operation of the MCU. A variation in frequency of more than 2% from one clock cycle to the next can lead to unpredictable behavior. It is required to ensure that the MCU is kept in reset during such changes in the clock frequency.

Timer/Counter Oscillator

For AVR microcontrollers with Timer/Counter Oscillator pins (TOSC1 and TOSC2), the crystal is connected directly between the pins. No external capacitors are needed. The Oscillator is optimized for use with a 32.768 kHz watch crystal. Applying an external clock source to TOSC1 is not recommended.

Note: The Timer/Counter Oscillator uses the same type of crystal oscillator as Low-Frequency Oscillator and the internal capacitors have the same nominal value of 36 pF.



31



Power Management and Sleep Modes

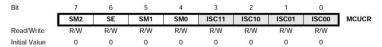
Sleep modes enable the application to shut down unused modules in the MCU, thereby saving power. The AVR provides various sleep modes allowing the user to tailor the power consumption to the application's requirements.

To enter any of the six sleep modes, the SE bit in MCUCR must be written to logic one and a SLEEP instruction must be executed. The SM2, SM1, and SM0 bits in the MCUCR Register select which sleep mode (Idle, ADC Noise Reduction, Power-down, Power-save, Standby, or Extended Standby) will be activated by the SLEEP instruction. See Table 13 for a summary. If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU wakes up. The MCU is then halted for four cycles in addition to the start-up time, it executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the Register File and SRAM are unaltered when the device wakes up from sleep. If a Reset occurs during sleep mode, the MCU wakes up and executes from the Reset Vector.

Figure 11 on page 24 presents the different clock systems in the ATmega16, and their distribution. The figure is helpful in selecting an appropriate sleep mode.

MCU Control Register – MCUCR

The MCU Control Register contains control bits for power management.



• Bits 7, 5, 4 - SM2..0: Sleep Mode Select Bits 2, 1, and 0

These bits select between the six available sleep modes as shown in Table 13.

Table 13. Sleep Mode Select

SM2	SM1	SM0	Sleep Mode	
0	0	0	Idle	
0	0	1	ADC Noise Reduction	
0	1	0	Power-down	
0	1	1	Power-save	
1	0	0	Reserved	
1	0	1	Reserved	
1	1	0	Standby ⁽¹⁾	
1	1	1	Extended Standby ⁽¹⁾	

Note: 1. Standby mode and Extended Standby mode are only available with external crystals or resonators.

• Bit 6 - SE: Sleep Enable

The SE bit must be written to logic one to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmers purpose, it is recommended to write the Sleep Enable (SE) bit to one just before the execution of the SLEEP instruction and to clear it immediately after waking up.



32



Idle Mode

When the SM2..0 bits are written to 000, the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing SPI, USART, Analog Comparator, ADC, Two-wire Serial Interface, Timer/Counters, Watchdog, and the interrupt system to continue operating. This sleep mode basically halts ${\rm clk_{CPU}}$ and ${\rm clk_{FLASH}}$, while allowing the other clocks to run.

Idle mode enables the MCU to wake up from external triggered interrupts as well as internal ones like the Timer Overflow and USART Transmit Complete interrupts. If wake-up from the Analog Comparator interrupt is not required, the Analog Comparator can be powered down by setting the ACD bit in the Analog Comparator Control and Status Register – ACSR. This will reduce power consumption in Idle mode. If the ADC is enabled, a conversion starts automatically when this mode is entered.

ADC Noise Reduction Mode

When the SM2..0 bits are written to 001, the SLEEP instruction makes the MCU enter ADC Noise Reduction mode, stopping the CPU but allowing the ADC, the External Interrupts, the Two-wire Serial Interface address watch, Timer/Counter2 and the Watchdog to continue operating (if enabled). This sleep mode basically halts clk_{I/O}, clk_{CPU}, and clk_{FLASH}, while allowing the other clocks to run.

This improves the noise environment for the ADC, enabling higher resolution measurements. If the ADC is enabled, a conversion starts automatically when this mode is entered. Apart form the ADC Conversion Complete interrupt, only an External Reset, a Watchdog Reset, a Brown-out Reset, a Two-wire Serial Interface Address Match Interrupt, a Timer/Counter2 interrupt, an SPM/EEPROM ready interrupt, an External level interrupt on INT0 or INT1, or an external interrupt on INT2 can wake up the MCU from ADC Noise Reduction mode.

Power-down Mode

When the SM2..0 bits are written to 010, the SLEEP instruction makes the MCU enter Power-down mode. In this mode, the External Oscillator is stopped, while the External interrupts, the Two-wire Serial Interface address watch, and the Watchdog continue operating (if enabled). Only an External Reset, a Watchdog Reset, a Brown-out Reset, a Two-wire Serial Interface address match interrupt, an External level interrupt on INT0 or INT1, or an External interrupt on INT2 can wake up the MCU. This sleep mode basically halts all generated clocks, allowing operation of asynchronous modules only.

Note that if a level triggered interrupt is used for wake-up from Power-down mode, the changed level must be held for some time to wake up the MCU. Refer to "External Interrupts" on page 68 for details.

When waking up from Power-down mode, there is a delay from the wake-up condition occurs until the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is defined by the same CKSEL Fuses that define the reset time-out period, as described in "Clock Sources" on page 25.

Power-save Mode

When the SM2..0 bits are written to 011, the SLEEP instruction makes the MCU enter Powersave mode. This mode is identical to Power-down, with one exception:

If Timer/Counter2 is clocked asynchronously, that is, the AS2 bit in ASSR is set, Timer/Counter2 will run during sleep. The device can wake up from either Timer Overflow or Output Compare event from Timer/Counter2 if the corresponding Timer/Counter2 interrupt enable bits are set in TIMSK, and the Global Interrupt Enable bit in SREG is set.

If the Asynchronous Timer is NOT clocked asynchronously, Power-down mode is recommended instead of Power-save mode because the contents of the registers in the Asynchronous Timer should be considered undefined after wake-up in Power-save mode if AS2 is 0.

This sleep mode basically halts all clocks except clk_{ASY} , allowing operation only of asynchronous modules, including Timer/Counter2 if clocked asynchronously.



33



Standby Mode

When the SM2..0 bits are 110 and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter Standby mode. This mode is identical to Power-down with the exception that the Oscillator is kept running. From Standby mode, the device wakes up in six clock cycles.

Extended Standby Mode

When the SM2..0 bits are 111 and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter Extended Standby mode. This mode is identical to Power-save mode with the exception that the Oscillator is kept running. From Extended Standby mode, the device wakes up in six clock cycles..

Table 14. Active Clock Domains and Wake Up Sources in the Different Sleep Modes

		Active Clock domains			Oscilla	Oscillators			Wake-up Sources					
Sleep Mode	clk _{CPU}	clk _{FLASH}	clk _{io}	clk _{ADC}	clk _{ASY}	Main Clock Source Enabled	Timer Osc. Enabled	INT2 INT1 INT0	TWI Address Match	Timer 2	SPM / EEPROM Ready	ADC	Other I/O	
Idle			Х	Х	Х	Х	X ⁽²⁾	Х	Х	Х	Х	Х	Х	
ADC Noise Redu- ction				х	х	х	X ⁽²⁾	X ⁽³⁾	Х	х	Х	х		
Power Down								X ⁽³⁾	Х					
Power Save					X ⁽²⁾		X ⁽²⁾	X ⁽³⁾	Х	X ⁽²⁾				
Standby ⁽¹⁾						х		X ⁽³⁾	Х					
Exten- ded Standby ⁽¹⁾					X ⁽²⁾	х	X ⁽²⁾	X ⁽³⁾	х	X ⁽²⁾				

Notes: 1. External Crystal or resonator selected as clock source.

- 2. If AS2 bit in ASSR is set.
- 3. Only INT2 or level interrupt INT1 and INT0.





Minimizing Power Consumption

There are several issues to consider when trying to minimize the power consumption in an AVR controlled system. In general, sleep modes should be used as much as possible, and the sleep mode should be selected so that as few as possible of the device's functions are operating. All functions not needed should be disabled. In particular, the following modules may need special consideration when trying to achieve the lowest possible power consumption.

Analog to Digital Converter

If enabled, the ADC will be enabled in all sleep modes. To save power, the ADC should be disabled before entering any sleep mode. When the ADC is turned off and on again, the next conversion will be an extended conversion. Refer to "Analog to Digital Converter" on page 204 for details on ADC operation.

Analog Comparator

When entering Idle mode, the Analog Comparator should be disabled if not used. When entering ADC Noise Reduction mode, the Analog Comparator should be disabled. In the other sleep modes, the Analog Comparator is automatically disabled. However, if the Analog Comparator is set up to use the Internal Voltage Reference as input, the Analog Comparator should be disabled in all sleep modes. Otherwise, the Internal Voltage Reference will be enabled, independent of sleep mode. Refer to "Analog Comparator" on page 201 for details on how to configure the Analog Comparator.

Brown-out Detector

If the Brown-out Detector is not needed in the application, this module should be turned off. If the Brown-out Detector is enabled by the BODEN Fuse, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to "Brown-out Detection" on page 40 for details on how to configure the Brown-out Detector.

Internal Voltage Reference

The Internal Voltage Reference will be enabled when needed by the Brown-out Detector, the Analog Comparator or the ADC. If these modules are disabled as described in the sections above, the internal voltage reference will be disabled and it will not be consuming power. When turned on again, the user must allow the reference to start up before the output is used. If the reference is kept on in sleep mode, the output can be used immediately. Refer to "Internal Voltage Reference" on page 42 for details on the start-up time.

Watchdog Timer

If the Watchdog Timer is not needed in the application, this module should be turned off. If the Watchdog Timer is enabled, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to "Watchdog Timer" on page 42 for details on how to configure the Watchdog Timer.

Port Pins

When entering a sleep mode, all port pins should be configured to use minimum power. The most important thing is then to ensure that no pins drive resistive loads. In sleep modes where the both the I/O clock ($clk_{I/O}$) and the ADC clock (clk_{ADC}) are stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed for detecting wake-up conditions, and it will then be enabled. Refer to the section "Digital Input Enable and Sleep Modes" on page 54 for details on which pins are enabled. If the input buffer is enabled and the input signal is left floating or have an analog signal level close to $V_{CC}/2$, the input buffer will use excessive power.





JTAG Interface and On-chip Debug System If the On-chip debug system is enabled by the OCDEN Fuse and the chip enter Power down or Power save sleep mode, the main clock source remains enabled. In these sleep modes, this will contribute significantly to the total current consumption. There are three alternative ways to avoid this:

- · Disable OCDEN Fuse.
- · Disable JTAGEN Fuse.
- · Write one to the JTD bit in MCUCSR.

The TDO pin is left floating when the JTAG interface is enabled while the JTAG TAP controller is not shifting data. If the hardware connected to the TDO pin does not pull up the logic level, power consumption will increase. Note that the TDI pin for the next device in the scan chain contains a pull-up that avoids this problem. Writing the JTD bit in the MCUCSR register to one or leaving the JTAG fuse unprogrammed disables the JTAG interface.



36



——— ATmega16(L)

System Control and Reset

Resetting the AVR

During Reset, all I/O Registers are set to their initial values, and the program starts execution from the Reset Vector. The instruction placed at the Reset Vector must be a JMP – absolute jump – instruction to the reset handling routine. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa. The circuit diagram in Figure 15 shows the reset logic. Table 15 defines the electrical parameters of the reset circuitry.

The I/O ports of the AVR are immediately reset to their initial state when a reset source goes active. This does not require any clock source to be running.

After all reset sources have gone inactive, a delay counter is invoked, stretching the Internal Reset. This allows the power to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the CKSEL Fuses. The different selections for the delay period are presented in "Clock Sources" on page 25.

Reset Sources

The ATmega16 has five sources of reset:

- · Power-on Reset.
 - The MCU is reset when the supply voltage is below the Power-on Reset threshold (V_{POT}).
- · External Reset.
 - The MCU is reset when a low level is present on the $\overline{\text{RESET}}$ pin for longer than the minimum pulse length.
- Watchdog Reset.
 - The MCU is reset when the Watchdog Timer period expires and the Watchdog is enabled.
- Brown-out Reset.
 - The MCU is reset when the supply voltage $V_{\rm CC}$ is below the Brown-out Reset threshold $(V_{\rm BOT})$ and the Brown-out Detector is enabled.
- JTAG AVR Reset

The MCU is reset as long as there is a logic one in the Reset Register, one of the scan chains of the JTAG system. Refer to the section "IEEE 1149.1 (JTAG) Boundary-scan" on page 228 for details.





Figure 15. Reset Logic

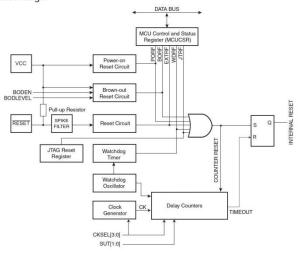


Table 15. Reset Characteristics

Symbol	Parameter	Condition	Min	Тур	Max	Units	
	Power-on Reset Threshold Voltage (rising)			1.4	2.3		
V _{POT}	Power-on Reset Threshold Voltage (falling) ⁽¹⁾			1.3	2.3	V	
V _{RST}	RESET Pin Threshold Voltage		0.1V _{CC}		0.9V _{CC}		
t _{RST}	Minimum pulse width on RESET Pin				1.5	μs	
	Brown-out Reset	BODLEVEL = 1	2.5	2.7	3.2	v	
V_{BOT}	Threshold Voltage ⁽²⁾	BODLEVEL = 0	3.6	4.0	4.5		
	Minimum low voltage	BODLEVEL = 1	2				
t _{BOD}	period for Brown-out Detection	BODLEVEL = 0		2		μs	
V _{HYST}	Brown-out Detector hysteresis			50		mV	

Notes: 1. The Power-on Reset will not work unless the supply voltage has been below V_{POT} (falling).

2. V_{BOT} may be below nominal minimum operating voltage for some devices. For devices where this is the case, the device is tested down to V_{CC} = V_{BOT} during the production test. This guarantees that a Brown-out Reset will occur before V_{CC} drops to a voltage where correct operation of the microcontroller is no longer guaranteed. The test is performed using BODLEVEL = 1 for ATmega16L and BODLEVEL = 0 for ATmega16. BODLEVEL = 1 is not applicable for ATmega16.



38



Power-on Reset

A Power-on Reset (POR) pulse is generated by an On-chip detection circuit. The detection level is defined in Table 15. The POR is activated whenever V_{CC} is below the detection level. The POR circuit can be used to trigger the Start-up Reset, as well as to detect a failure in supply voltage.

A Power-on Reset (POR) circuit ensures that the device is reset from Power-on. Reaching the Power-on Reset threshold voltage invokes the delay counter, which determines how long the device is kept in RESET after $V_{\mathbb{C}\mathbb{C}}$ rise. The RESET signal is activated again, without any delay, when $V_{\mathbb{C}\mathbb{C}}$ decreases below the detection level.

Figure 16. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC} .

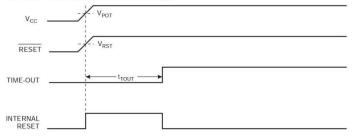
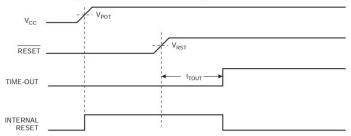


Figure 17. MCU Start-up, RESET Extended Externally





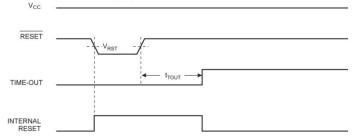
39



External Reset

An External Reset is generated by a low level on the $\overline{\text{RESET}}$ pin. Reset pulses longer than the minimum pulse width (see Table 15) will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset. When the applied signal reaches the Reset Threshold Voltage – V_{RST} – on its positive edge, the delay counter starts the MCU after the Time-out period t_{TOUT} has expired.

Figure 18. External Reset During Operation



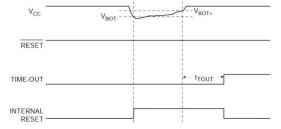
Brown-out Detection

ATmega16 has an On-chip Brown-out Detection (BOD) circuit for monitoring the V_{CC} level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the fuse BODLEVEL to be 2.7V (BODLEVEL unprogrammed), or 4.0V (BODLEVEL programmed). The trigger level has a hysteresis to ensure spike free Brown-out Detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST}/2$ and $V_{BOT-} = V_{BOT} - V_{HYST}/2$.

The BOD circuit can be enabled/disabled by the fuse BODEN. When the BOD is enabled (BODEN programmed), and V_{CC} decreases to a value below the trigger level (V_{BOT-} in Figure 19), the Brown-out Reset is immediately activated. When V_{CC} increases above the trigger level (V_{BOT+} in Figure 19), the delay counter starts the MCU after the Time-out period t_{TOUT} has expired.

The BOD circuit will only detect a drop in V_{CC} if the voltage stays below the trigger level for longer than t_{BOD} given in Table 15.

Figure 19. Brown-out Reset During Operation





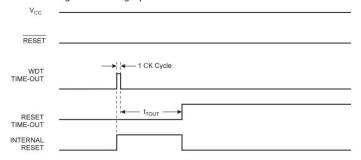
40



Watchdog Reset

When the Watchdog times out, it will generate a short reset pulse of one CK cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT} . Refer to page 42 for details on operation of the Watchdog Timer.

Figure 20. Watchdog Reset During Operation



MCU Control and Status Register – MCUCSR The MCU Control and Status Register provides information on which reset source caused an MCU Reset.



· Bit 4 - JTRF: JTAG Reset Flag

This bit is set if a reset is being caused by a logic one in the JTAG Reset Register selected by the JTAG instruction AVR_RESET. This bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

· Bit 3 - WDRF: Watchdog Reset Flag

This bit is set if a Watchdog Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

• Bit 2 - BORF: Brown-out Reset Flag

This bit is set if a Brown-out Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

· Bit 1 - EXTRF: External Reset Flag

This bit is set if an External Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

· Bit 0 - PORF: Power-on Reset Flag

This bit is set if a Power-on Reset occurs. The bit is reset only by writing a logic zero to the flag.

To make use of the Reset Flags to identify a reset condition, the user should read and then reset the MCUCSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the Reset Flags.



41



Internal Voltage Reference

ATmega16 features an internal bandgap reference. This reference is used for Brown-out Detection, and it can be used as an input to the Analog Comparator or the ADC. The 2.56V reference to the ADC is generated from the internal bandgap reference.

Voltage Reference Enable Signals and Start-up Time The voltage reference has a start-up time that may influence the way it should be used. The start-up time is given in Table 16. To save power, the reference is not always turned on. The reference is on during the following situations:

- 1. When the BOD is enabled (by programming the BODEN Fuse).
- When the bandgap reference is connected to the Analog Comparator (by setting the ACBG bit in ACSR).
- 3. When the ADC is enabled.

Thus, when the BOD is not enabled, after setting the ACBG bit or enabling the ADC, the user must always allow the reference to start up before the output from the Analog Comparator or ADC is used. To reduce power consumption in Power-down mode, the user can avoid the three conditions above to ensure that the reference is turned off before entering Power-down mode.

Table 16. Internal Voltage Reference Characteristics

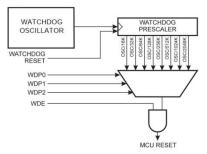
Symbol	Parameter	Min	Тур	Max	Units
V_{BG}	Bandgap reference voltage	1.15	1.23	1.4	V
t _{BG}	Bandgap reference start-up time		40	70	μs
I _{BG}	Bandgap reference current consumption		10		μΑ

Watchdog Timer

The Watchdog Timer is clocked from a separate On-chip Oscillator which runs at 1 MHz. This is the typical value at V_{CC} = 5V. See characterization data for typical values at other V_{CC} levels. By controlling the Watchdog Timer prescaler, the Watchdog Reset interval can be adjusted as shown in Table 17 on page 43. The WDR – Watchdog Reset – instruction resets the Watchdog Timer. The Watchdog Timer is also reset when it is disabled and when a Chip Reset occurs. Eight different clock cycle periods can be selected to determine the reset period. If the reset period expires without another Watchdog Reset, the ATmega16 resets and executes from the Reset Vector. For timing details on the Watchdog Reset, refer to page 41.

To prevent unintentional disabling of the Watchdog, a special turn-off sequence must be followed when the Watchdog is disabled. Refer to the description of the Watchdog Timer Control Register for details.

Figure 21. Watchdog Timer





42



Watchdog Timer Control Register – WDTCR

Bit	7	6	5	4	3	2	1	0	
	-	-	-	WDTOE	WDE	WDP2	WDP1	WDP0	WDTCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	•
Initial Value	0	0	0	0	0	0	0	0	

· Bits 7..5 - Res: Reserved Bits

These bits are reserved bits in the ATmega16 and will always read as zero.

• Bit 4 - WDTOE: Watchdog Turn-off Enable

This bit must be set when the WDE bit is written to logic zero. Otherwise, the Watchdog will not be disabled. Once written to one, hardware will clear this bit after four clock cycles. Refer to the description of the WDE bit for a Watchdog disable procedure.

• Bit 3 - WDE: Watchdog Enable

When the WDE is written to logic one, the Watchdog Timer is enabled, and if the WDE is written to logic zero, the Watchdog Timer function is disabled. WDE can only be cleared if the WDTOE bit has logic level one. To disable an enabled Watchdog Timer, the following procedure must be followed:

- In the same operation, write a logic one to WDTOE and WDE. A logic one must be written to WDE even though it is set to one before the disable operation starts.
- 2. Within the next four clock cycles, write a logic 0 to WDE. This disables the Watchdog.

• Bits 2..0 - WDP2, WDP1, WDP0: Watchdog Timer Prescaler 2, 1, and 0

The WDP2, WDP1, and WDP0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding Timeout Periods are shown in Table 17.

Table 17. Watchdog Timer Prescale Select

WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V _{CC} = 3.0V	Typical Time-out at V _{CC} = 5.0V
0	0	0	16K (16,384)	17.1 ms	16.3 ms
0	0	1	32K (32,768)	34.3 ms	32.5 ms
0	1	0	64K (65,536)	68.5 ms	65 ms
0	1	1	128K (131,072)	0.14 s	0.13 s
1	0	0	256K (262,144)	0.27 s	0.26 s
1	0	1	512K (524,288)	0.55 s	0.52 s
1	1	0	1,024K (1,048,576)	1.1 s	1.0 s
1	1	1	2,048K (2,097,152)	2.2 s	2.1 s





The following code example shows one assembly and one C function for turning off the WDT. The example assumes that interrupts are controlled (for example by disabling interrupts globally) so that no interrupts will occur during execution of these functions.

```
Assembly Code Example
   WDT_off:
     ; Reset WDT
     WDR
     ; Write logical one to WDTOE and WDE
     in r16, WDTCR
     ori r16, (1<<WDTOE) | (1<<WDE)
     out WDTCR, r16
     ; Turn off WDT
     ldi r16, (0<<WDE)
     out WDTCR, r16
     ret
C Code Example
   void WDT_off(void)
     /* Reset WDT*/
     _WDR();
     /\star Write logical one to WDTOE and WDE \star/
     WDTCR |= (1<<WDTOE) | (1<<WDE);
     /* Turn off WDT */
     WDTCR = 0 \times 00;
```



44



Interrupts

This section describes the specifics of the interrupt handling as performed in ATmega16. For a general explanation of the AVR interrupt handling, refer to "Reset and Interrupt Handling" on page 13.

Interrupt Vectors in ATmega16

Table 18. Reset and Interrupt Vectors

Vector No.	Program Address ⁽²⁾	Source	Interrupt Definition
1	\$000 ⁽¹⁾	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	\$002	INTO	External Interrupt Request 0
3	\$004	INT1	External Interrupt Request 1
4	\$006	TIMER2 COMP	Timer/Counter2 Compare Match
5	\$008	TIMER2 OVF	Timer/Counter2 Overflow
6	\$00A	TIMER1 CAPT	Timer/Counter1 Capture Event
7	\$00C	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	\$00E	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	\$010	TIMER1 OVF	Timer/Counter1 Overflow
10	\$012	TIMER0 OVF	Timer/Counter0 Overflow
11	\$014	SPI, STC	Serial Transfer Complete
12	\$016	USART, RXC	USART, Rx Complete
13	\$018	USART, UDRE	USART Data Register Empty
14	\$01A	USART, TXC	USART, Tx Complete
15	\$01C	ADC	ADC Conversion Complete
16	\$01E	EE_RDY	EEPROM Ready
17	\$020	ANA_COMP	Analog Comparator
18	\$022	TWI	Two-wire Serial Interface
19	\$024	INT2	External Interrupt Request 2
20	\$026	TIMER0 COMP	Timer/Counter0 Compare Match
21	\$028	SPM_RDY	Store Program Memory Ready

Notes: 1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see "Boot Loader Support – Read-While-Write Self-Programming" on page 246.

When the IVSEL bit in GICR is set, interrupt vectors will be moved to the start of the Boot Flash section. The address of each Interrupt Vector will then be the address in this table added to the start address of the Boot Flash section.

Table 19 shows Reset and Interrupt Vectors placement for the various combinations of BOOTRST and IVSEL settings. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa.



45



Table 19. Reset and Interrupt Vectors Placement⁽¹⁾

BOOTRST	IVSEL	Reset address	Interrupt Vectors Start Address
1	0	\$0000	\$0002
1	1	\$0000	Boot Reset Address + \$0002
0	0	Boot Reset Address	\$0002
0	1	Boot Reset Address	Boot Reset Address + \$0002

Note: 1. The Boot Reset Address is shown in Table 100 on page 257. For the BOOTRST Fuse "1" means unprogrammed while "0" means programmed.

The most typical and general program setup for the Reset and Interrupt Vector Addresses in ATmega16 is:

Address	Labels	Code		Co	omments
\$000		jmp	RESET	;	Reset Handler
\$002		jmp	EXT_INTO	;	IRQ0 Handler
\$004		jmp	EXT_INT1	;	IRQ1 Handler
\$006		jmp	TIM2_COMP	;	Timer2 Compare Handler
\$008		jmp	TIM2_OVF	;	Timer2 Overflow Handler
\$00A		jmp	TIM1_CAPT	;	Timer1 Capture Handler
\$00C		jmp	TIM1_COMPA	;	Timerl CompareA Handler
\$00E		jmp	TIM1_COMPB	;	Timer1 CompareB Handler
\$010		jmp	TIM1_OVF	;	Timer1 Overflow Handler
\$012		jmp	TIMO_OVF	;	Timer0 Overflow Handler
\$014		jmp	SPI_STC	;	SPI Transfer Complete Handler
\$016		jmp	USART_RXC	;	USART RX Complete Handler
\$018		jmp	USART_UDRE	;	UDR Empty Handler
\$01A		jmp	USART_TXC	;	USART TX Complete Handler
\$01C		jmp	ADC	;	ADC Conversion Complete Handler
\$01E		jmp	EE_RDY	;	EEPROM Ready Handler
\$020		jmp	ANA_COMP	;	Analog Comparator Handler
\$022		jmp	TWSI	;	Two-wire Serial Interface Handler
\$024		jmp	EXT_INT2	;	IRQ2 Handler
\$026		jmp	TIMO_COMP	;	Timer0 Compare Handler
\$028		jmp	SPM_RDY	;	Store Program Memory Ready Handler
;					
\$02A	RESET:	ldi	r16, high (RAMEND)	;	Main program start
\$02B		out	SPH,r16	;	Set Stack Pointer to top of RAM
\$02C		ldi	r16, low(RAMEND)		
\$02D		out	SPL,r16		
\$02E		sei		;	Enable interrupts
\$02F		<inst< td=""><td>r> xxx</td><td></td><td></td></inst<>	r> xxx		





When the BOOTRST Fuse is unprogrammed, the Boot section size set to 2 Kbytes and the IVSEL bit in the GICR Register is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
Address Labels Code
$000 RESET: ldi r16,high(RAMEND) ; Main program start
$001
                out SPH, r16
                                    ; Set Stack Pointer to top of RAM
$002
              ldi r16, low(RAMEND)
$003
               out SPL, r16
$004
                sei
                                     ; Enable interrupts
$005
                <instr> xxx
.org $1C02
$1C02
                jmp EXT_INTO
                                     ; IRQ0 Handler
$1C04
                jmp EXT_INT1
                                     ; IRQ1 Handler
. . .
                . .
$1C28
                jmp SPM_RDY
                                     ; Store Program Memory Ready Handler
```

When the BOOTRST Fuse is programmed and the Boot section size set to 2 Kbytes, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
Address Labels
                      Code
                                              Comments
.org $002
$002
                 jmp EXT_INT0
                                       ; IRQ0 Handler
$004
                jmp EXT_INT1
                                       ; IRQ1 Handler
$028
                jmp SPM_RDY
                                       ; Store Program Memory Ready Handler
.org $1C00
        RESET: ldi r16, high (RAMEND) ; Main program start
$1C00
$1C01
                 out SPH, r16
                                      ; Set Stack Pointer to top of RAM
$1C02
                 ldi r16.low(RAMEND)
$1C03
$1C04
                 sei
                                       ; Enable interrupts
                 <instr> xxx
```

When the BOOTRST Fuse is programmed, the Boot section size set to 2 Kbytes and the IVSEL bit in the GICR Register is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
Labels
                 Code
.org $1C00
                      RESET
                                   ; Reset handler
$1C00
                 jmp
$1C02
                 jmp EXT_INTO
                                   ; IRQ0 Handler
                                   ; IRQ1 Handler
                 jmp EXT_INT1
$1C04
$1C28
                 jmp
                      SPM_RDY
                                   ; Store Program Memory Ready Handler
$1C2A
        RESET: ldi r16, high (RAMEND) ; Main program start
$1C2B
                 out SPH, r16
                                    ; Set Stack Pointer to top of RAM
$1C2C
                 ldi
                      r16, low(RAMEND)
$1C2D
                 out SPL, r16
$1C2E
                 sei
                                      ; Enable interrupts
$1C2F
                 <instr> xxx
```





Moving Interrupts Between Application and Boot Space The General Interrupt Control Register controls the placement of the Interrupt Vector table.

General Interrupt Control Register – GICR

Bit	7	6	5	4	3	2	1	0	
	INT1	INTO	INT2	-	-	-	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

· Bit 1 - IVSEL: Interrupt Vector Select

When the IVSEL bit is cleared (zero), the Interrupt Vectors are placed at the start of the Flash memory. When this bit is set (one), the interrupt vectors are moved to the beginning of the Boot Loader section of the Flash. The actual address of the start of the Boot Flash section is determined by the BOOTSZ Fuses. Refer to the section "Boot Loader Support – Read-While-Write Self-Programming" on page 246 for details. To avoid unintentional changes of Interrupt Vector tables, a special write procedure must be followed to change the IVSEL bit:

- 1. Write the Interrupt Vector Change Enable (IVCE) bit to one.
- 2. Within four cycles, write the desired value to IVSEL while writing a zero to IVCE.

Interrupts will automatically be disabled while this sequence is executed. Interrupts are disabled in the cycle IVCE is set, and they remain disabled until after the instruction following the write to IVSEL. If IVSEL is not written, interrupts remain disabled for four cycles. The I-bit in the Status Register is unaffected by the automatic disabling.

Note: If Interrupt Vectors are placed in the Boot Loader section and Boot Lock bit BLB02 is programmed, interrupts are disabled while executing from the Application section. If Interrupt Vectors are placed in the Application section and Boot Lock bit BLB12 is programed, interrupts are disabled while executing from the Boot Loader section. Refer to the section "Boot Loader Support – Read-While-Write Self-Programming" on page 246 for details on Boot Lock bits.

· Bit 0 - IVCE: Interrupt Vector Change Enable

The IVCE bit must be written to logic one to enable change of the IVSEL bit. IVCE is cleared by hardware four cycles after it is written or when IVSEL is written. Setting the IVCE bit will disable interrupts, as explained in the IVSEL description above. See Code Example below





```
Assembly Code Example
   Move_interrupts:
     ; Enable change of interrupt vectors
    ldi r16, (1<<IVCE)
     out GICR, r16
     ; Move interrupts to boot Flash section
     ldi r16, (1<<IVSEL)
     out GICR, r16
     ret
C Code Example
   void Move_interrupts(void)
     /* Enable change of interrupt vectors */
    GICR = (1<<IVCE);
    /* Move interrupts to boot Flash section */
    GICR = (1<<IVSEL);
```



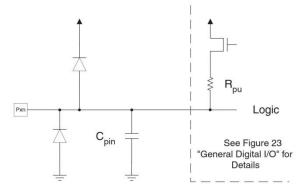


I/O Ports

Introduction

All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both $\rm V_{CC}$ and Ground as indicated in Figure 22. Refer to "Electrical Characteristics" on page 291 for a complete list of parameters.

Figure 22. I/O Pin Equivalent Schematic



All registers and bit references in this section are written in general form. A lower case "x" represents the numbering letter for the port, and a lower case "n" represents the bit number. However, when using the register or bit defines in a program, the precise form must be used, that is, PORTB3 for bit no. 3 in Port B, here documented generally as PORTxn. The physical I/O Registers and bit locations are listed in "Register Description for I/O Ports" on page 66.

Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. In addition, the Pull-up Disable – PUD bit in SFIOR disables the pull-up function for all pins in all norts when set

Using the I/O port as General Digital I/O is described in "Ports as General Digital I/O" on page 50. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in "Alternate Port Functions" on page 55. Refer to the individual module sections for a full description of the alternate functions.

Note that enabling the alternate function of some of the port pins does not affect the use of the other pins in the port as general digital I/O.

Ports as General Digital I/O

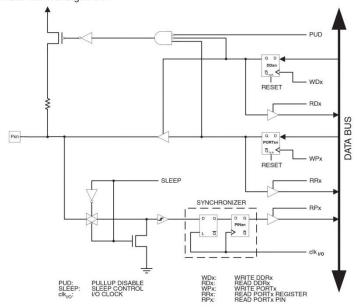
The ports are bi-directional I/O ports with optional internal pull-ups. Figure 23 shows a functional description of one I/O-port pin, here generically called Pxn.



50



Figure 23. General Digital I/O(1)



Note: 1. WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. clk_{I/O}, SLEEP, and PUD are common to all ports.

Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in "Register Description for I/O Ports" on page 66, the DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, and the PINxn bits at the PINx I/O address.

The DDxn bit in the DDRx Register selects the direction of this pin. If DDxn is written logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is configured as an input pin.

If PORTxn is written logic one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written logic zero or the pin has to be configured as an output pin. The port pins are tri-stated when a reset condition becomes active, even if no clocks are running.

If PORTxn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

When switching between tri-state ({DDxn, PORTxn} = 0b00) and output high ({DDxn, PORTxn} = 0b11), an intermediate state with either pull-up enabled ({DDxn, PORTxn} = 0b01) or output low ({DDxn, PORTxn} = 0b10) must occur. Normally, the pull-up enabled state is fully acceptable, as a high-impedant environment will not notice the difference between a strong high driver and a pull-up. If this is not the case, the PUD bit in the SFIOR Register can be set to disable all pull-ups in all ports.



51



Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state ({DDxn, PORTxn} = 0b00) or the output high state ({DDxn, PORTxn} = 0b11) as an intermediate step.

Table 20 summarizes the control signals for the pin value.

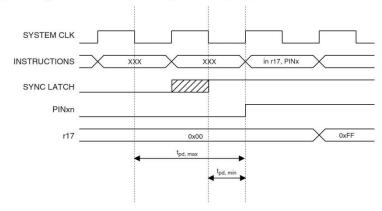
Table 20. Port Pin Configurations

DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	Х	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	Х	Output	No	Output Low (Sink)
1	1	Х	Output	No	Output High (Source)

Reading the Pin Value

Independent of the setting of Data Direction bit DDxn, the port pin can be read through the PINxn Register bit. As shown in Figure 23, the PINxn Register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. Figure 24 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{\text{pd,max}}$ and $t_{\text{pd,min}}$ respectively.

Figure 24. Synchronization when Reading an Externally Applied Pin Value



Consider the clock period starting shortly *after* the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the "SYNC LATCH" signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn Register at the succeeding positive clock edge. As indicated by the two arrows $t_{pd,max}$ and $t_{pd,min}$, a single signal transition on the pin will be delayed between ½ and 1½ system clock period depending upon the time of assertion.

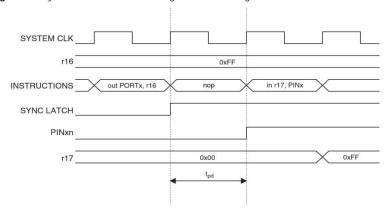


52



When reading back a software assigned pin value, a nop instruction must be inserted as indicated in Figure 25. The out instruction sets the "SYNC LATCH" signal at the positive edge of the clock. In this case, the delay $t_{\rm pd}$ through the synchronizer is one system clock period.

Figure 25. Synchronization when Reading a Software Assigned Pin Value







The following code example shows how to set port B pins 0 and 1 high, 2 and 3 low, and define the port pins from 4 to 7 as input with pull-ups assigned to port pins 6 and 7. The resulting pin values are read back again, but as previously discussed, a *nop* instruction is included to be able to read back the value recently assigned to some of the pins.

```
Assembly Code Example<sup>(1)</sup>
     ; Define pull-ups and set outputs high
     ; Define directions for port pins
     ldi r16,(1<<PB7) | (1<<PB6) | (1<<PB1) | (1<<PB0)
     ldi r17, (1<<DDB3) | (1<<DDB2) | (1<<DDB1) | (1<<DDB0)
     out PORTB, r16
     out DDRB, r17
     ; Insert nop for synchronization
     nop
     ; Read port pins
     in
          r16, PINB
C Code Example<sup>(1)</sup>
   unsigned char i;
     /* Define pull-ups and set outputs high */
     /* Define directions for port pins */
     PORTB = (1<<PB7) | (1<<PB6) | (1<<PB1) | (1<<PB0);
     DDRB = (1<<DDB3) | (1<<DDB2) | (1<<DDB1) | (1<<DDB0);
     /* Insert nop for synchronization*/
     _NOP();
     /* Read port pins */
     i = PINB;
```

Note: 1. For the assembly program, two temporary registers are used to minimize the time from pullups are set on pins 0, 1, 6, and 7, until the direction bits are correctly set, defining bit 2 and 3 as low and redefining bits 0 and 1 as strong high drivers.

Digital Input Enable and Sleep Modes

As shown in Figure 23, the digital input signal can be clamped to ground at the input of the schmitt-trigger. The signal denoted SLEEP in the figure, is set by the MCU Sleep Controller in Power-down mode, Power-save mode, Standby mode, and Extended Standby mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{\rm CC}/2$.

SLEEP is overridden for port pins enabled as External Interrupt pins. If the External Interrupt Request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in "Alternate Port Functions" on page 55.

If a logic high level ("one") is present on an Asynchronous External Interrupt pin configured as "Interrupt on Rising Edge, Falling Edge, or Any Logic Change on Pin" while the External Interrupt is *not* enabled, the corresponding External Interrupt Flag will be set when resuming from the above mentioned sleep modes, as the clamping in these sleep modes produces the requested logic change.



54



Unconnected pins

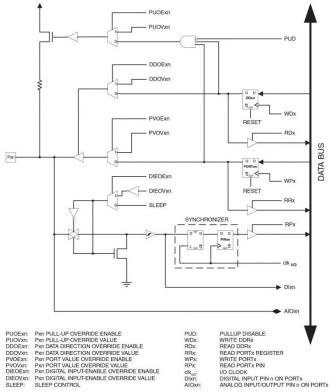
If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled (Reset, Active mode and Idle mode).

The simplest method to ensure a defined level of an unused pin, is to enable the internal pull-up. In this case, the pull-up will be disabled during reset. If low power consumption during reset is important, it is recommended to use an external pull-up or pull-down. Connecting unused pins directly to $V_{\rm CC}$ or GND is not recommended, since this may cause excessive currents if the pin is accidentally configured as an output.

Alternate Port Functions

Most port pins have alternate functions in addition to being General Digital I/Os. Figure 26 shows how the port pin control signals from the simplified Figure 23 can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the AVR microcontroller family.

Figure 26. Alternate Port Functions(1)



Note: 1. WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. clk_{I/O}, SLEEP, and PUD are common to all ports. All other signals are unique for each pin.



55



Table 21 summarizes the function of the overriding signals. The pin and port indexes from Figure 26 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 21. Generic Description of Overriding Signals for Alternate Functions

Signal Name	Full Name	Description
PUOE	Pull-up Override Enable	If this signal is set, the pull-up enable is controlled by the PUOV signal. If this signal is cleared, the pull-up is enabled when {DDxn, PORTxn, PUD} = 0b010.
PUOV	Pull-up Override Value	If PUOE is set, the pull-up is enabled/disabled when PUOV is set/cleared, regardless of the setting of the DDxn, PORTxn, and PUD Register bits.
DDOE	Data Direction Override Enable	If this signal is set, the Output Driver Enable is controlled by the DDOV signal. If this signal is cleared, the Output driver is enabled by the DDxn Register bit.
DDOV	Data Direction Override Value	If DDOE is set, the Output Driver is enabled/disabled when DDOV is set/cleared, regardless of the setting of the DDxn Register bit.
PVOE	Port Value Override Enable	If this signal is set and the Output Driver is enabled, the port value is controlled by the PVOV signal. If PVOE is cleared, and the Output Driver is enabled, the port Value is controlled by the PORTxn Register bit.
PVOV	Port Value Override Value	If PVOE is set, the port value is set to PVOV, regardless of the setting of the PORTxn Register bit.
DIEOE	Digital Input Enable Override Enable	If this bit is set, the Digital Input Enable is controlled by the DIEOV signal. If this signal is cleared, the Digital Input Enable is determined by MCU-state (Normal Mode, sleep modes).
DIEOV	Digital Input Enable Override Value	If DIEOE is set, the Digital Input is enabled/disabled when DIEOV is set/cleared, regardless of the MCU state (Normal Mode, sleep modes).
DI	Digital Input	This is the Digital Input to alternate functions. In the figure, the signal is connected to the output of the schmitt trigger but before the synchronizer. Unless the Digital Input is used as a clock source, the module with the alternate function will use its own synchronizer.
AIO	Analog Input/ output	This is the Analog Input/output to/from alternate functions. The signal is connected directly to the pad, and can be used bi-directionally.

The following subsections shortly describe the alternate functions for each port, and relate the overriding signals to the alternate function. Refer to the alternate function description for further details.





Special Function I/O Register – SFIOR

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	-	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

· Bit 2 - PUD: Pull-up disable

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01). See "Configuring the Pin" on page 51 for more details about this feature.

Alternate Functions of Port A

Port A has an alternate function as analog input for the ADC as shown in Table 22. If some Port A pins are configured as outputs, it is essential that these do not switch when a conversion is in progress. This might corrupt the result of the conversion.

Table 22. Port A Pins Alternate Functions

Port Pin	Alternate Function	
PA7	ADC7 (ADC input channel 7)	
PA6	ADC6 (ADC input channel 6)	
PA5	ADC5 (ADC input channel 5)	
PA4	ADC4 (ADC input channel 4)	
PA3	PA3 ADC3 (ADC input channel 3)	
PA2 ADC2 (ADC input channel 2)		
PA1 ADC1 (ADC input channel 1)		
PA0	ADC0 (ADC input channel 0)	

Table 23 and Table 24 relate the alternate functions of Port A to the overriding signals shown in Figure 26 on page 55.

Table 23. Overriding Signals for Alternate Functions in PA7..PA4

Signal Name	PA7/ADC7	PA6/ADC6	PA5/ADC5	PA4/ADC4
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	-	-	-	1.—.1
AIO	ADC7 INPUT	ADC6 INPUT	ADC5 INPUT	ADC4 INPUT



57



Table 24. Overriding Signals for Alternate Functions in PA3..PA0

Signal Name	PA3/ADC3	PA2/ADC2	PA1/ADC1	PA0/ADC0
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	_	_	-	_
AIO	ADC3 INPUT	ADC2 INPUT	ADC1 INPUT	ADC0 INPUT

Alternate Functions of Port B

Alternate Functions of The Port B pins with alternate functions are shown in Table 25.

Table 25. Port B Pins Alternate Functions

Port Pin	Alternate Functions	
PB7	SCK (SPI Bus Serial Clock)	
PB6	MISO (SPI Bus Master Input/Slave Output)	
PB5	MOSI (SPI Bus Master Output/Slave Input)	
PB4	SS (SPI Slave Select Input)	
PB3	AIN1 (Analog Comparator Negative Input) OC0 (Timer/Counter0 Output Compare Match Output)	
PB2	AIN0 (Analog Comparator Positive Input) INT2 (External Interrupt 2 Input)	
PB1	T1 (Timer/Counter1 External Counter Input)	
PB0	T0 (Timer/Counter0 External Counter Input) XCK (USART External Clock Input/Output)	

The alternate pin configuration is as follows:

· SCK - Port B, Bit 7

SCK: Master Clock output, Slave Clock input pin for SPI channel. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB7. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB7. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB7 bit.

· MISO - Port B, Bit 6

MISO: Master Data input, Slave Data output pin for SPI channel. When the SPI is enabled as a Master, this pin is configured as an input regardless of the setting of DDB6. When the SPI is enabled as a Slave, the data direction of this pin is controlled by DDB6. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB6 bit.



58



· MOSI - Port B, Bit 5

MOSI: SPI Master Data output, Slave Data input for SPI channel. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB5. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB5. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB5 bit.

• SS - Port B, Bit 4

SS: Slave Select input. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB4. As a Slave, the SPI is activated when this pin is driven low. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB4. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB4 bit.

· AIN1/OC0 - Port B, Bit 3

AIN1, Analog Comparator Negative Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the analog comparator.

OCO, Output Compare Match output: The PB3 pin can serve as an external output for the Timer/Counter0 Compare Match. The PB3 pin has to be configured as an output (DDB3 set (one)) to serve this function. The OCO pin is also the output pin for the PWM mode timer function.

· AIN0/INT2 - Port B, Bit 2

AINO, Analog Comparator Positive input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.

INT2, External Interrupt Source 2: The PB2 pin can serve as an external interrupt source to the MCU.

T1 – Port B, Bit 1

T1, Timer/Counter1 Counter Source.

· T0/XCK - Port B, Bit 0

T0, Timer/Counter0 Counter Source.

XCK, USART External Clock. The Data Direction Register (DDB0) controls whether the clock is output (DDB0 set) or input (DDB0 cleared). The XCK pin is active only when the USART operates in Synchronous mode.

Table 26 and Table 27 relate the alternate functions of Port B to the overriding signals shown in Figure 26 on page 55. SPI MSTR INPUT and SPI SLAVE OUTPUT constitute the MISO signal, while MOSI is divided into SPI MSTR OUTPUT and SPI SLAVE INPUT.





Table 26. Overriding Signals for Alternate Functions in PB7..PB4

Signal Name	PB7/SCK	PB6/MISO	PB5/MOSI	PB4/SS
PUOE	SPE • MSTR	SPE • MSTR	SPE • MSTR	SPE • MSTR
PUOV	PORTB7 • PUD	PORTB6 • PUD	PORTB5 • PUD	PORTB4 • PUD
DDOE	SPE • MSTR	SPE • MSTR	SPE • MSTR	SPE • MSTR
DDOV	0	0	0	0
PVOE	SPE • MSTR	SPE • MSTR	SPE • MSTR	0
PVOV	SCK OUTPUT	SPI SLAVE OUTPUT	SPI MSTR OUTPUT	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	SCK INPUT	SPI MSTR INPUT	SPI SLAVE INPUT	SPI SS
AIO	_	-	-	-

Table 27. Overriding Signals for Alternate Functions in PB3..PB0

Signal Name	PB3/OC0/AIN1	PB2/INT2/AIN0	PB1/T1	PB0/T0/XCK
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC0 ENABLE	0	0	UMSEL
PVOV	OC0	0	0	XCK OUTPUT
DIEOE	0	INT2 ENABLE	0	0
DIEOV	0	1	0	0
DI	_	INT2 INPUT	T1 INPUT	XCK INPUT/T0 INPUT
AIO	AIN1 INPUT	AINO INPUT	_	_





Alternate Functions of Port C

The Port C pins with alternate functions are shown in Table 28. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs.

Table 28. Port C Pins Alternate Functions

Port Pin	Alternate Function	
PC7	TOSC2 (Timer Oscillator Pin 2)	
PC6	TOSC1 (Timer Oscillator Pin 1)	
PC5	TDI (JTAG Test Data In)	
PC4	TDO (JTAG Test Data Out)	
PC3	TMS (JTAG Test Mode Select)	
PC2	TCK (JTAG Test Clock)	
PC1	SDA (Two-wire Serial Bus Data Input/Output Line)	
PC0	SCL (Two-wire Serial Bus Clock Line)	

The alternate pin configuration is as follows:

· TOSC2 - Port C, Bit 7

TOSC2, Timer Oscillator pin 2: When the AS2 bit in ASSR is set (one) to enable asynchronous clocking of Timer/Counter2, pin PC7 is disconnected from the port, and becomes the inverting output of the Oscillator amplifier. In this mode, a Crystal Oscillator is connected to this pin, and the pin can not be used as an I/O pin.

· TOSC1 - Port C, Bit 6

TOSC1, Timer Oscillator pin 1: When the AS2 bit in ASSR is set (one) to enable asynchronous clocking of Timer/Counter2, pin PC6 is disconnected from the port, and becomes the input of the inverting Oscillator amplifier. In this mode, a Crystal Oscillator is connected to this pin, and the pin can not be used as an I/O pin.

TDI – Port C, Bit 5

TDI, JTAG Test Data In: Serial input data to be shifted in to the Instruction Register or Data Register (scan chains). When the JTAG interface is enabled, this pin can not be used as an I/O pin.

· TDO - Port C, Bit 4

TDO, JTAG Test Data Out: Serial output data from Instruction Register or Data Register. When the JTAG interface is enabled, this pin can not be used as an I/O pin.

The TD0 pin is tri-stated unless TAP states that shifts out data are entered.

· TMS - Port C, Bit 3

TMS, JTAG Test Mode Select: This pin is used for navigating through the TAP-controller state machine. When the JTAG interface is enabled, this pin can not be used as an I/O pin.

TCK – Port C, Bit 2

TCK, JTAG Test Clock: JTAG operation is synchronous to TCK. When the JTAG interface is enabled, this pin can not be used as an I/O pin.



61



· SDA - Port C, Bit 1

SDA, Two-wire Serial Interface Data: When the TWEN bit in TWCR is set (one) to enable the Two-wire Serial Interface, pin PC1 is disconnected from the port and becomes the Serial Data I/O pin for the Two-wire Serial Interface. In this mode, there is a spike filter on the pin to suppress spikes shorter than 50 ns on the input signal, and the pin is driven by an open drain driver with slew-rate limitation. When this pin is used by the Two-wire Serial Interface, the pull-up can still be controlled by the PORTC1 bit.

· SCL - Port C, Bit 0

SCL, Two-wire Serial Interface Clock: When the TWEN bit in TWCR is set (one) to enable the Two-wire Serial Interface, pin PC0 is disconnected from the port and becomes the Serial Clock I/O pin for the Two-wire Serial Interface. In this mode, there is a spike filter on the pin to suppress spikes shorter than 50 ns on the input signal, and the pin is driven by an open drain driver with slew-rate limitation. When this pin is used by the Two-wire Serial Interface, the pull-up can still be controlled by the PORTCO bit.

Table 29 and Table 30 relate the alternate functions of Port C to the overriding signals shown in Figure 26 on page 55.

Table 29. Overriding Signals for Alternate Functions in PC7..PC4

Signal Name	PC7/TOSC2	PC6/TOSC1	PC5/TDI	PC4/TDO
PUOE	AS2	AS2	JTAGEN	JTAGEN
PUOV	0	0	1	0
DDOE	AS2	AS2	JTAGEN	JTAGEN
DDOV	0	0	0	SHIFT_IR + SHIFT_DR
PVOE	0	0	0	JTAGEN
PVOV	0	0	0	TDO
DIEOE	AS2	AS2	JTAGEN	JTAGEN
DIEOV	0	0	0	0
DI	-	-	_	-
AIO	T/C2 OSC OUTPUT	T/C2 OSC INPUT	TDI	_





Table 30. Overriding Signals for Alternate Functions in PC3..PC0⁽¹⁾

Signal Name	PC3/TMS	PC2/TCK	PC1/SDA	PC0/SCL
PUOE	JTAGEN	JTAGEN	TWEN	TWEN
PUOV	1	1	PORTC1 • PUD	PORTC0 • PUD
DDOE	JTAGEN	JTAGEN	TWEN	TWEN
DDOV	0	0	SDA_OUT	SCL_OUT
PVOE	0	0	TWEN	TWEN
PVOV	0	0	0	0
DIEOE	JTAGEN	JTAGEN	0	0
DIEOV	0	0	0	0
DI	_	_	7-1	_
AIO	TMS	TCK	SDA INPUT	SCL INPUT

Note: 1. When enabled, the Two-wire Serial Interface enables slew-rate controls on the output pins PC0 and PC1. This is not shown in the figure. In addition, spike filters are connected between the AIO outputs shown in the port figure and the digital logic of the TWI module.

Alternate Functions of Port D

Alternate Functions of The Port D pins with alternate functions are shown in Table 31.

Table 31. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD7	OC2 (Timer/Counter2 Output Compare Match Output)
PD6	ICP1 (Timer/Counter1 Input Capture Pin)
PD5	OC1A (Timer/Counter1 Output Compare A Match Output)
PD4	OC1B (Timer/Counter1 Output Compare B Match Output)
PD3	INT1 (External Interrupt 1 Input)
PD2	INTO (External Interrupt 0 Input)
PD1	TXD (USART Output Pin)
PD0	RXD (USART Input Pin)

The alternate pin configuration is as follows:

• OC2 - Port D, Bit 7

OC2, Timer/Counter2 Output Compare Match output: The PD7 pin can serve as an external output for the Timer/Counter2 Output Compare. The pin has to be configured as an output (DDD7 set (one)) to serve this function. The OC2 pin is also the output pin for the PWM mode timer function.

ICP1 – Port D, Bit 6

ICP1 – Input Capture Pin: The PD6 pin can act as an Input Capture pin for Timer/Counter1.



63



· OC1A - Port D, Bit 5

OC1A, Output Compare Match A output: The PD5 pin can serve as an external output for the Timer/Counter1 Output Compare A. The pin has to be configured as an output (DDD5 set (one)) to serve this function. The OC1A pin is also the output pin for the PWM mode timer function.

· OC1B - Port D, Bit 4

OC1B, Output Compare Match B output: The PD4 pin can serve as an external output for the Timer/Counter1 Output Compare B. The pin has to be configured as an output (DDD4 set (one)) to serve this function. The OC1B pin is also the output pin for the PWM mode timer function.

• INT1 - Port D, Bit 3

INT1, External Interrupt Source 1: The PD3 pin can serve as an external interrupt source.

· INT0 - Port D, Bit 2

INTO, External Interrupt Source 0: The PD2 pin can serve as an external interrupt source.

TXD – Port D, Bit 1

TXD, Transmit Data (Data output pin for the USART). When the USART Transmitter is enabled, this pin is configured as an output regardless of the value of DDD1.

· RXD - Port D, Bit 0

RXD, Receive Data (Data input pin for the USART). When the USART Receiver is enabled this pin is configured as an input regardless of the value of DDD0. When the USART forces this pin to be an input, the pull-up can still be controlled by the PORTD0 bit.

Table 32 and Table 33 relate the alternate functions of Port D to the overriding signals shown in Figure 26 on page 55.

Table 32. Overriding Signals for Alternate Functions PD7..PD4

Signal Name	PD7/OC2	PD6/ICP1	PD5/OC1A	PD4/OC1B
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC2 ENABLE	0	OC1A ENABLE	OC1B ENABLE
PVOV	OC2	0	OC1A	OC1B
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	-	ICP1 INPUT	_	-
AIO	-	_	-	-



64



 Table 33. Overriding Signals for Alternate Functions in PD3..PD0

Signal Name	PD3/INT1	PD2/INT0	PD1/TXD	PD0/RXD
PUOE	0	0	TXEN	RXEN
PUOV	0	0	0	PORTD0 • PUD
DDOE	0	0	TXEN	RXEN
DDOV	0	0	1	0
PVOE	0	0	TXEN	0
PVOV	0	0	TXD	0
DIEOE	INT1 ENABLE	INTO ENABLE	0	0
DIEOV	1	1	0	0
DI	INT1 INPUT	INTO INPUT	_	RXD
AIO	_	-	_	-





Register Description for I/O Ports

Port A Data Register – PORTA

Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

Port A Data Direction Register – DDRA

Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

Port A Input Pins Address – PINA

Bit	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	•
Initial Value	N/A								

Port B Data Register – PORTB

Bit	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

Port B Data Direction Register – DDRB

Bit	7	6	5	4	3	2	1	0	
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	2							
Indiana Malana	0	0	0	0	0	0	0	0	

Port B Input Pins Address – PINB

Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A								





Port	C Da	ata	Reg	ister	_
POP	TC				

Bit	7	6	5	4	3	2	1	0	
	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

Port C Data Direction Register – DDRC

Bit	7	6	5	4	3	2	1	0	
	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
Read/Write	R/W	51							
Initial Value	0	0	0	0	0	0	0	0	

Port C Input Pins Address – PINC

Bit	7	6	5	4	3	2	1	0	
	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A								

Port D Data Register – PORTD

Bit	7	6	5	4	3	2	1	0	
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

Port D Data Direction Register – DDRD

Bit	7	6	5	4	3	2	1	0	
	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	•							
Initial Value	0	0	0	0	0	0	0	0	

Port D Input Pins Address – PIND

Bit	7	6	5	4	3	2	1	0	
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	ti.
Initial Value	N/A								





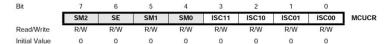
External Interrupts

The External Interrupts are triggered by the INTO, INT1, and INT2 pins. Observe that, if enabled, the interrupts will trigger even if the INTO..2 pins are configured as outputs. This feature provides a way of generating a software interrupt. The external interrupts can be triggered by a falling or rising edge or a low level (INT2 is only an edge triggered interrupt). This is set up as indicated in the specification for the MCU Control Register – MCUCR – and MCU Control and Status Register – MCUCSR. When the external interrupt is enabled and is configured as level triggered (only INTO/INT1), the interrupt will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INTO and INT1 requires the presence of an I/O clock, described in "Clock Systems and their Distribution" on page 24. Low level interrupts on INTO/INT1 and the edge interrupt on INT2 are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode. The I/O clock is halted in all sleep modes except Idle mode.

Note that if a level triggered interrupt is used for wake-up from Power-down mode, the changed level must be held for some time to wake up the MCU. This makes the MCU less sensitive to noise. The changed level is sampled twice by the Watchdog Oscillator clock. The period of the Watchdog Oscillator is 1 μs (nominal) at 5.0V and 25°C. The frequency of the Watchdog Oscillator is voltage dependent as shown in "Electrical Characteristics" on page 291. The MCU will wake up if the input has the required level during this sampling or if it is held until the end of the start-up time. The start-up time is defined by the SUT Fuses as described in "System Clock and Clock Options" on page 24. If the level is sampled twice by the Watchdog Oscillator clock but disappears before the end of the start-up time, the MCU will still wake up, but no interrupt will be generated. The required level must be held long enough for the MCU to complete the wake up to trigger the level interrupt.

MCU Control Register – MCUCR

The MCU Control Register contains control bits for interrupt sense control and general MCU functions.



· Bit 3, 2 - ISC11, ISC10: Interrupt Sense Control 1 Bit 1 and Bit 0

The External Interrupt 1 is activated by the external pin INT1 if the SREG I-bit and the corresponding interrupt mask in the GICR are set. The level and edges on the external INT1 pin that activate the interrupt are defined in Table 34. The value on the INT1 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 34. Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.



68



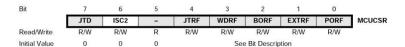
· Bit 1, 0 - ISC01, ISC00: Interrupt Sense Control 0 Bit 1 and Bit 0

The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in Table 35. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 35. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INTO generates an interrupt request.
0	1	Any logical change on INTO generates an interrupt request.
1	0	The falling edge of INTO generates an interrupt request.
1	1	The rising edge of INTO generates an interrupt request.

MCU Control and Status Register – MCUCSR



· Bit 6 - ISC2: Interrupt Sense Control 2

The Asynchronous External Interrupt 2 is activated by the external pin INT2 if the SREG I-bit and the corresponding interrupt mask in GICR are set. If ISC2 is written to zero, a falling edge on INT2 activates the interrupt. If ISC2 is written to one, a rising edge on INT2 activates the interrupt. Edges on INT2 are registered asynchronously. Pulses on INT2 wider than the minimum pulse width given in Table 36 will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. When changing the ISC2 bit, an interrupt can occur. Therefore, it is recommended to first disable INT2 by clearing its Interrupt Enable bit in the GICR Register. Then, the ISC2 bit can be changed. Finally, the INT2 Interrupt Flag should be cleared by writing a logical one to its Interrupt Flag bit (INTF2) in the GIFR Register before the interrupt is re-enabled.

Table 36. Asynchronous External Interrupt Characteristics

Symbol	Parameter	Condition	Min	Тур	Max	Units
t _{INT}	Minimum pulse width for asynchronous external interrupt			50		ns

General Interrupt Control Register – GICR



• Bit 7 - INT1: External Interrupt Request 1 Enable

When the INT1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control1 bits 1/0 (ISC11 and ISC10) in the MCU General Control Register (MCUCR) define whether the External Interrupt is activated on rising

AMEL

69



and/or falling edge of the INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The corresponding interrupt of External Interrupt Request 1 is executed from the INT1 interrupt Vector.

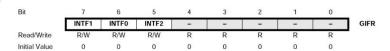
· Bit 6 - INT0: External Interrupt Request 0 Enable

When the INTO bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control0 bits 1/0 (ISC01 and ISC00) in the MCU General Control Register (MCUCR) define whether the External Interrupt is activated on rising and/or falling edge of the INTO pin or level sensed. Activity on the pin will cause an interrupt request even if INTO is configured as an output. The corresponding interrupt of External Interrupt Request 0 is executed from the INTO interrupt vector.

Bit 5 – INT2: External Interrupt Request 2 Enable

When the INT2 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control2 bit (ISC2) in the MCU Control and Status Register (MCUCSR) defines whether the External Interrupt is activated on rising or falling edge of the INT2 pin. Activity on the pin will cause an interrupt request even if INT2 is configured as an output. The corresponding interrupt of External Interrupt Request 2 is executed from the INT2 Interrupt Vector.

General Interrupt Flag Register – GIFR



• Bit 7 - INTF1: External Interrupt Flag 1

When an edge or logic change on the INT1 pin triggers an interrupt request, INTF1 becomes set (one). If the I-bit in SREG and the INT1 bit in GICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT1 is configured as a level interrupt.

• Bit 6 - INTF0: External Interrupt Flag 0

When an edge or logic change on the INTO pin triggers an interrupt request, INTFO becomes set (one). If the I-bit in SREG and the INTO bit in GICR are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INTO is configured as a level interrupt.

Bit 5 – INTF2: External Interrupt Flag 2

When an event on the INT2 pin triggers an interrupt request, INTF2 becomes set (one). If the I-bit in SREG and the INT2 bit in GICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. Note that when entering some sleep modes with the INT2 interrupt disabled, the input buffer on this pin will be disabled. This may cause a logic change in internal signals which will set the INTF2 Flag. See "Digital Input Enable and Sleep Modes" on page 54 for more information.



70



8-bit Timer/Counter0 with PWM

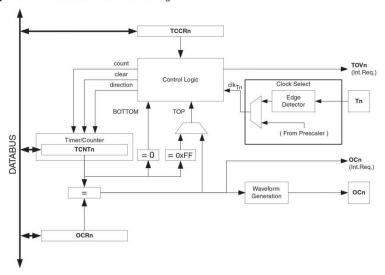
Timer/Counter0 is a general purpose, single compare unit, 8-bit Timer/Counter module. The main features are:

- · Single Compare Unit Counter
- · Clear Timer on Compare Match (Auto Reload)
- · Glitch-free, Phase Correct Pulse Width Modulator (PWM)
- Frequency Generator
- External Event Counter
- 10-bit Clock Prescaler
- · Overflow and Compare Match Interrupt Sources (TOV0 and OCF0)

Overview

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 27. For the actual placement of I/O pins, refer to "Pinout ATmega16" on page 2. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the "8-bit Timer/Counter Register Description" on page 83.

Figure 27. 8-bit Timer/Counter Block Diagram



Registers

The Timer/Counter (TCNT0) and Output Compare Register (OCR0) are 8-bit registers. Interrupt request (abbreviated to Int.Req. in the figure) signals are all visible in the Timer Interrupt Flag Register (TIFR). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK). TIFR and TIMSK are not shown in the figure since these registers are shared by other timer units.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the T0 pin. The Clock Select logic block controls which clock source and edge the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock (clk_{T0}).

The double buffered Output Compare Register (OCR0) is compared with the Timer/Counter value at all times. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the Output Compare Pin (OC0). See "Output Compare



71



Unit" on page 73. for details. The compare match event will also set the Compare Flag (OCF0) which can be used to generate an output compare interrupt request.

Definitions

Many register and bit references in this document are written in general form. A lower case "n" replaces the Timer/Counter number, in this case 0. However, when using the register or bit defines in a program, the precise form must be used, that is, TCNT0 for accessing Timer/Counter0 counter value and so on.

The definitions in Table 37 are also used extensively throughout the document.

Table 37. Definitions

	Z4104177.1.1.7.0
ВОТТОМ	The counter reaches the BOTTOM when it becomes 0x00.
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR0 Register. The assignment is dependent on the mode of operation.

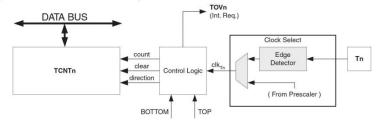
Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the clock select logic which is controlled by the clock select (CS02:0) bits located in the Timer/Counter Control Register (TCCR0). For details on clock sources and prescaler, see "Timer/Counter0 and Timer/Counter1 Prescalers" on page 87.

Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 28 shows a block diagram of the counter and its surroundings.

Figure 28. Counter Unit Block Diagram



Signal description (internal signals):

count Increment or decrement TCNT0 by 1.direction Select between increment and decrement.

clear TCNT0 (set all bits to zero).

clk_{Tn} Timer/Counter clock, referred to as clk_{To} in the following.
 TOP Signalize that TCNT0 has reached maximum value.
 BOTTOM Signalize that TCNT0 has reached minimum value (zero).

Depending of the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T0}) . clk_{T0} can be generated from an external or internal clock source, selected by the Clock Select bits (CS02:0). When no clock source is selected (CS02:0 = 0) the timer is stopped. However, the TCNT0 value can be accessed by the CPU, regardless of



72



whether ${\rm clk}_{\rm T0}$ is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the WGM01 and WGM00 bits located in the Timer/Counter Control Register (TCCR0). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare output OC0. For more details about advanced counting sequences and waveform generation, see "Modes of Operation" on page 76.

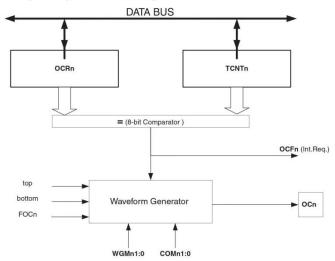
The Timer/Counter Overflow (TOV0) Flag is set according to the mode of operation selected by the WGM01:0 bits. TOV0 can be used for generating a CPU interrupt.

Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the Output Compare Register (OCR0). Whenever TCNT0 equals OCR0, the comparator signals a match. A match will set the Output Compare Flag (OCF0) at the next timer clock cycle. If enabled (OCIE0 = 1 and Global Interrupt Flag in SREG is set), the Output Compare Flag generates an output compare interrupt. The OCF0 Flag is automatically cleared when the interrupt is executed. Alternatively, the OCF0 Flag can be cleared by software by writing a logical one to its I/O bit location. The waveform generator uses the match signal to generate an output according to operating mode set by the WGM01:0 bits and Compare Output mode (COM01:0) bits. The max and bottom signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation (See "Modes of Operation" on page 76.).

Figure 29 shows a block diagram of the output compare unit.

Figure 29. Output Compare Unit, Block Diagram



The OCR0 Register is double buffered when using any of the Pulse Width Modulation (PWM) modes. For the normal and Clear Timer on Compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR0 Compare Register to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.



73



The OCR0 Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR0 Buffer Register, and if double buffering is disabled the CPU will access the OCR0 directly.

Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the Force Output Compare (FOC0) bit. Forcing compare match will not set the OCF0 Flag or reload/clear the timer, but the OC0 pin will be updated as if a real compare match had occurred (the COM01:0 bits settings define whether the OC0 pin is set, cleared or toggled).

Compare Match Blocking by TCNT0 Write All CPU write operations to the TCNTO Register will block any compare match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCRO to be initialized to the same value as TCNTO without triggering an interrupt when the Timer/Counter clock is enabled

Using the Output Compare Unit Since writing TCNT0 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT0 when using the output compare unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT0 equals the OCR0 value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT0 value equal to BOTTOM when the counter is downcounting.

The setup of the OC0 should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC0 value is to use the Force Output Compare (FOC0) strobe bits in Normal mode. The OC0 Register keeps its value even when changing between waveform generation modes.

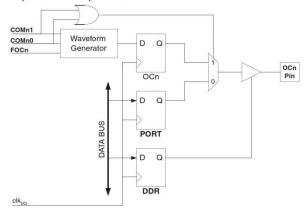
Be aware that the COM01:0 bits are not double buffered together with the compare value. Changing the COM01:0 bits will take effect immediately.

Compare Match Output Unit

The Compare Output mode (COM01:0) bits have two functions. The Waveform Generator uses the COM01:0 bits for defining the Output Compare (OC0) state at the next compare match. Also, the COM01:0 bits control the OC0 pin output source. Figure 30 shows a simplified schematic of the logic affected by the COM01:0 bit setting. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O port Control Registers (DDR and PORT) that are affected by the COM01:0 bits are shown. When referring to the OC0 state, the reference is for the internal OC0 Register, not the OC0 pin. If a System Reset occur, the OC0 Register is reset to "0".



Figure 30. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the Output Compare (OC0) from the Waveform Generator if either of the COM01:0 bits are set. However, the OC0 pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. The Data Direction Register bit for the OC0 pin (DDR_OC0) must be set as output before the OC0 value is visible on the pin. The port override function is independent of the Waveform Generation mode.

The design of the output compare pin logic allows initialization of the OC0 state before the output is enabled. Note that some COM01:0 bit settings are reserved for certain modes of operation. See "8-bit Timer/Counter Register Description" on page 83.

Compare Output Mode and Waveform Generation The Waveform Generator uses the COM01:0 bits differently in normal, CTC, and PWM modes. For all modes, setting the COM01:0 = 0 tells the waveform generator that no action on the OCO Register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 39 on page 84. For fast PWM mode, refer to Table 40 on page 84, and for phase correct PWM refer to Table 41 on page 84.

A change of the COM01:0 bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC0 strobe bits.





Modes of Operation

The mode of operation, that is, the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the Waveform Generation mode (WGM01:0) and Compare Output mode (COM01:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM01:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM01:0 bits control whether the output should be set, cleared, or toggled at a compare match (See "Compare Match Output Unit" on page 74.).

For detailed timing information refer to Figure 34, Figure 35, Figure 36 and Figure 37 in "Timer/Counter Timing Diagrams" on page 81.

Normal Mode

The simplest mode of operation is the normal mode (WGM01:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter Overflow Flag (TOV0) will be set in the same timer clock cycle as the TCNT0 becomes zero. The TOV0 Flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV0 Flag, the timer resolution can be increased by software. There are no special cases to consider in the normal mode, a new counter value can be written anytime.

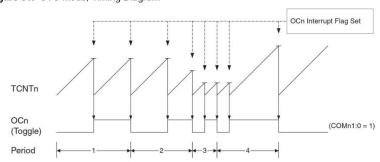
The output compare unit can be used to generate interrupts at some given time. Using the output compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM01:0 = 2), the OCR0 Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT0) matches the OCR0. The OCR0 defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 31. The counter value (TCNT0) increases until a compare match occurs between TCNT0 and OCR0, and then counter (TCNT0) is cleared.

Figure 31. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCF0 Flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have

AIMEL

76



the double buffering feature. If the new value written to OCR0 is lower than the current value of TCNT0, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFF) and wrap around starting at 0x00 before the compare match can occur.

For generating a waveform output in CTC mode, the OC0 output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COM01:0 = 1). The OC0 value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{\text{OC0}} = f_{\text{clk_VO}}/2$ when OCR0 is set to zero (0x00). The waveform frequency is defined by the following equation:

$$f_{OCn} = \frac{f_{\text{clk_I/O}}}{2 \cdot N \cdot (1 + OCRn)}$$

The N variable represents the prescale factor (1, 8, 64, 256, or 1024).

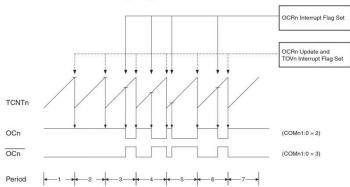
As for the Normal mode of operation, the TOV0 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

Fast PWM Mode

The fast Pulse Width Modulation or fast PWM mode (WGM01:0 = 3) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to MAX then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC0) is cleared on the compare match between TCNT0 and OCR0, and set at BOTTOM. In inverting Compare Output mode, the output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct PWM mode that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

In fast PWM mode, the counter is incremented until the counter value matches the MAX value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 32. The TCNTO value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNTO slopes represent compare matches between OCRO and TCNTO.

Figure 32. Fast PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV0) is set each time the counter reaches MAX. If the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.

ATMEL

77



In fast PWM mode, the compare unit allows generation of PWM waveforms on the OC0 pin. Setting the COM01:0 bits to 2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM01:0 to 3 (See Table 40 on page 84). The actual OC0 value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC0 Register at the compare match between OCR0 and TCNT0, and clearing (or setting) the OC0 Register at the timer clock cycle the counter is cleared (changes from MAX to BOTTOM).

The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnPWM} = \frac{f_{\text{clk_I/O}}}{N \cdot 256}$$

The N variable represents the prescale factor (1, 8, 64, 256, or 1024).

The extreme values for the OCR0 Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR0 is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCR0 equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM01:0 bits.)





(COMn1:0 = 2)

(COMn1:0 = 3)

Phase Correct PWM Mode

The phase correct PWM mode (WGM01:0 = 1) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to MAX and then from MAX to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC0) is cleared on the compare match between TCNT0 and OCR0 while upcounting, and set on the compare match while downcounting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The PWM resolution for the phase correct PWM mode is fixed to eight bits. In phase correct PWM mode the counter is incremented until the counter value matches MAX. When the counter reaches MAX, it changes the count direction. The TCNTO value will be equal to MAX for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 33. The TCNTO value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNTO slopes represent compare matches between OCRO and TCNTO.

OCR Interrupt Flag Set

OCR Update

TOVn Interrupt Flag Set

Figure 33. Phase Correct PWM Mode, Timing Diagram

OCn

OCn

Period

The Timer/Counter Overflow Flag (TOV0) is set each time the counter reaches BOTTOM. The Interrupt Flag can be used to generate an interrupt each time the counter reaches the BOTTOM value.

In phase correct PWM mode, the compare unit allows generation of PWM waveforms on the OCO pin. Setting the COM01:0 bits to 2 will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM01:0 to 3 (see Table 41 on page 84). The actual OCO value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OCO Register at the compare match between OCR0 and TCNT0 when the counter increments, and setting (or clearing) the OCO Register at compare match between OCR0 and TCNT0 when the counter decrements. The



79



PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnPCPWM} = \frac{f_{\text{clk_I/O}}}{N \cdot 510}$$

The N variable represents the prescale factor (1, 8, 64, 256, or 1024).

The extreme values for the OCR0 Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR0 is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

At the very start of Period 2 in Figure 33 OCn has a transition from high to low even though there is no Compare Match. The point of this transition is to guarantee symmetry around BOTTOM. There are two cases that give a transition without Compare Match:

- OCR0A changes its value from MAX, like in Figure 33. When the OCR0A value is MAX the
 OCn pin value is the same as the result of a down-counting Compare Match. To ensure
 symmetry around BOTTOM the OCn value at MAX must be correspond to the result of an
 up-counting Compare Match.
- The Timer starts counting from a value higher than the one in OCR0A, and for that reason misses the Compare Match and hence the OCn change that would have happened on the way up.





Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T0}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set. Figure 34 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 34. Timer/Counter Timing Diagram, no Prescaling

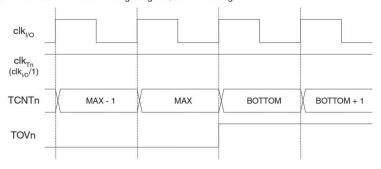


Figure 35 shows the same timing data, but with the prescaler enabled.

Figure 35. Timer/Counter Timing Diagram, with Prescaler ($f_{\text{clk_I/O}}/8$)

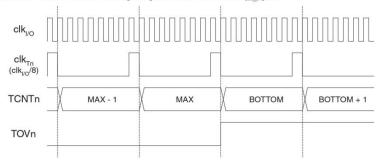


Figure 36 shows the setting of OCF0 in all modes except CTC mode.

AIMEL

81



Figure 36. Timer/Counter Timing Diagram, Setting of OCF0, with Prescaler ($f_{clk_I/O}/8$)

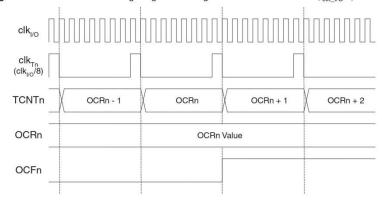
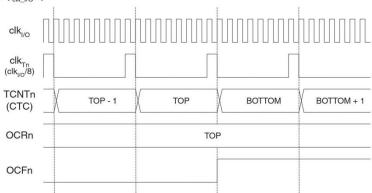


Figure 37 shows the setting of OCF0 and the clearing of TCNT0 in CTC mode.

Figure 37. Timer/Counter Timing Diagram, Clear Timer on Compare Match Mode, with Prescaler ($f_{clk_l/O}/8$)



<u>AIMEL</u>

82



8-bit Timer/Counter Register Description

Timer/Counter Control Register – TCCR0

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	33
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 - FOC0: Force Output Compare

The FOC0 bit is only active when the WGM00 bit specifies a non-PWM mode. However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR0 is written when operating in PWM mode. When writing a logical one to the FOC0 bit, an immediate compare match is forced on the Waveform Generation unit. The OC0 output is changed according to its COM01:0 bits setting. Note that the FOC0 bit is implemented as a strobe. Therefore it is the value present in the COM01:0 bits that determines the effect of the forced compare.

A FOC0 strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0 as TOP.

The FOC0 bit is always read as zero.

• Bit 3, 6 - WGM01:0: Waveform Generation Mode

These bits control the counting sequence of the counter, the source for the maximum (TOP) counter value, and what type of Waveform Generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode, Clear Timer on Compare Match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes. See Table 38 and "Modes of Operation" on page 76.

Table 38. Waveform Generation Mode Bit Description⁽¹⁾

Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation	ТОР	Update of OCR0	TOV0 Flag Set-on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	воттом
2	1	0	СТС	OCR0	Immediate	MAX
3	1	1	Fast PWM	0xFF	воттом	MAX

Note: 1. The CTC0 and PWM0 bit definition names are now obsolete. Use the WGM01:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

• Bit 5:4 - COM01:0: Compare Match Output Mode

These bits control the Output Compare pin (OC0) behavior. If one or both of the COM01:0 bits are set, the OC0 output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC0 pin must be set in order to enable the output driver.



83



When OC0 is connected to the pin, the function of the COM01:0 bits depends on the WGM01:0 bit setting. Table 39 shows the COM01:0 bit functionality when the WGM01:0 bits are set to a normal or CTC mode (non-PWM).

Table 39. Compare Output Mode, non-PWM Mode

COM01	COM00	Description	
0	0	Normal port operation, OC0 disconnected.	
0	1	Toggle OC0 on compare match	
1	0	Clear OC0 on compare match	
1	1	Set OC0 on compare match	

Table 40 shows the COM01:0 bit functionality when the WGM01:0 bits are set to fast PWM mode.

Table 40. Compare Output Mode, Fast PWM Mode⁽¹⁾

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match, set OC0 at BOTTOM, (non-inverting mode)
1	1	Set OC0 on compare match, clear OC0 at BOTTOM, (inverting mode)

Note: 1. A special case occurs when OCR0 equals TOP and COM01 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See "Fast PWM Mode" on page 77 for more details.

Table 41 shows the COM01:0 bit functionality when the WGM01:0 bits are set to phase correct PWM mode.

Table 41. Compare Output Mode, Phase Correct PWM Mode⁽¹⁾

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match when up-counting. Set OC0 on compare match when downcounting.
1	1	Set OC0 on compare match when up-counting. Clear OC0 on compare match when downcounting.

Note: 1. A special case occurs when OCR0 equals TOP and COM01 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See "Phase Correct PWM Mode" on page 79 for more details.





· Bit 2:0 - CS02:0: Clock Select

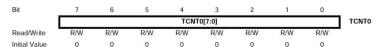
The three Clock Select bits select the clock source to be used by the Timer/Counter.

Table 42. Clock Select Bit Description

CS02	CS01	CS00	Description				
0	0	0	No clock source (Timer/Counter stopped).				
0	0	1	clk _{I/O} /(No prescaling)				
0	1	0	clk _{I/O} /8 (From prescaler)				
0	1	1	k _{I/O} /64 (From prescaler)				
1	0	0	clk _{I/O} /256 (From prescaler)				
1	0	1	clk _{I/O} /1024 (From prescaler)				
1	1	0	External clock source on T0 pin. Clock on falling edge.				
1	1	1	External clock source on T0 pin. Clock on rising edge.				

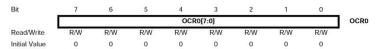
If external pin modes are used for the Timer/Counter0, transitions on the T0 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

Timer/Counter Register – TCNT0



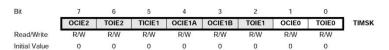
The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT0 Register blocks (removes) the compare match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a compare match between TCNT0 and the OCR0 Register.

Output Compare Register – OCR0



The Output Compare Register contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC0 pin.

Timer/Counter Interrupt Mask Register – TIMSK



• Bit 1 - OCIE0: Timer/Counter0 Output Compare Match Interrupt Enable

When the OCIE0 bit is written to one, and the I-bit in the Status Register is set (one), the Timer/Counter0 Compare Match interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter0 occurs, that is, when the OCF0 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.



85



• Bit 0 - TOIE0: Timer/Counter0 Overflow Interrupt Enable

When the TOIE0 bit is written to one, and the I-bit in the Status Register is set (one), the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, that is, when the TOV0 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

Timer/Counter Interrupt Flag Register – TIFR

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 1 - OCF0: Output Compare Flag 0

The OCF0 bit is set (one) when a compare match occurs between the Timer/Counter0 and the data in OCR0 – Output Compare Register0. OCF0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0 is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0 (Timer/Counter0 Compare Match Interrupt Enable), and OCF0 are set (one), the Timer/Counter0 Compare Match Interrupt is executed.

· Bit 0 - TOV0: Timer/Counter0 Overflow Flag

The bit TOV0 is set (one) when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set (one), the Timer/Counter0 Overflow interrupt is executed. In phase correct PWM mode, this bit is set when Timer/Counter0 changes counting direction at \$00.000.





Timer/Counter0 and Timer/Counter1 Prescalers

Timer/Counter1 and Timer/Counter0 share the same prescaler module, but the Timer/Counters can have different prescaler settings. The description below applies to both Timer/Counter1 and Timer/Counter0.

Internal Clock Source

The Timer/Counter can be clocked directly by the system clock (by setting the CSn2:0 = 1). This provides the fastest operation, with a maximum Timer/Counter clock frequency equal to system clock frequency ($f_{CLK_I/O}$). Alternatively, one of four taps from the prescaler can be used as a clock source. The prescaled clock has a frequency of either $f_{CLK_I/O}/8$, $f_{CLK_I/O}/64$, $f_{CLK_I/O}/256$, or $f_{CLK_I/O}/1024$.

Prescaler Reset

The prescaler is free running, that is, operates independently of the clock select logic of the Timer/Counter, and it is shared by Timer/Counter1 and Timer/Counter0. Since the prescaler is not affected by the Timer/Counter's clock select, the state of the prescaler will have implications for situations where a prescaled clock is used. One example of prescaling artifacts occurs when the timer is enabled and clocked by the prescaler (6 > CSn2:0 > 1). The number of system clock cycles from when the timer is enabled to the first count occurs can be from 1 to N+1 system clock cycles, where N equals the prescaler divisor (8, 64, 256, or 1024).

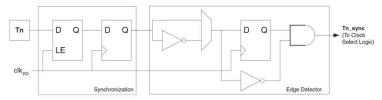
It is possible to use the Prescaler Reset for synchronizing the Timer/Counter to program execution. However, care must be taken if the other Timer/Counter that shares the same prescaler also uses prescaling. A prescaler reset will affect the prescaler period for all Timer/Counters it is connected to.

External Clock Source

An external clock source applied to the T1/T0 pin can be used as Timer/Counter clock (clk_{T1}/clk_{T0}). The T1/T0 pin is sampled once every system clock cycle by the pin synchronization logic. The synchronized (sampled) signal is then passed through the edge detector. Figure 38 shows a functional equivalent block diagram of the T1/T0 synchronization and edge detector logic. The registers are clocked at the positive edge of the internal system clock ($clk_{I/O}$). The latch is transparent in the high period of the internal system clock.

The edge detector generates one clk_{T1}/clk_{T0} pulse for each positive (CSn2:0 = 7) or negative (CSn2:0 = 6) edge it detects.

Figure 38. T1/T0 Pin Sampling



The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge has been applied to the T1/T0 pin to the counter is updated.

Enabling and disabling of the clock input must be done when T1/T0 has been stable for at least one system clock cycle, otherwise it is a risk that a false Timer/Counter clock pulse is generated.

Each half period of the external clock applied must be longer than one system clock cycle to ensure correct sampling. The external clock must be guaranteed to have less than half the system clock frequency ($f_{\text{ExtClk}} < f_{\text{clk_I/O}}/2$) given a 50/50% duty cycle. Since the edge detector uses



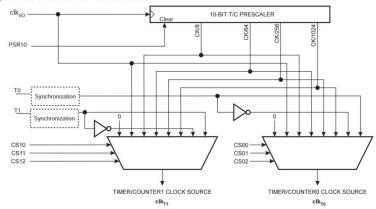
87



sampling, the maximum frequency of an external clock it can detect is half the sampling frequency (Nyquist sampling theorem). However, due to variation of the system clock frequency and duty cycle caused by Oscillator source (crystal, resonator, and capacitors) tolerances, it is recommended that maximum frequency of an external clock source is less than $f_{clk} _{I/O}/2.5$.

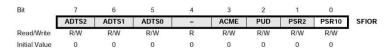
An external clock source can not be prescaled.

Figure 39. Prescaler for Timer/Counter0 and Timer/Counter1⁽¹⁾



Note: 1. The synchronization logic on the input pins (T1/T0) is shown in Figure 38.

Special Function IO Register – SFIOR



• Bit 0 – PSR10: Prescaler Reset Timer/Counter1 and Timer/Counter0

When this bit is written to one, the Timer/Counter1 and Timer/Counter0 prescaler will be reset. The bit will be cleared by hardware after the operation is performed. Writing a zero to this bit will have no effect. Note that Timer/Counter1 and Timer/Counter0 share the same prescaler and a reset of this prescaler will affect both timers. This bit will always be read as zero.





16-bit Timer/Counter1

The 16-bit Timer/Counter unit allows accurate program execution timing (event management), wave generation, and signal timing measurement. The main features are:

- True 16-bit Design (that is, allows 16-bit PWM)
 Two Independent Output Compare Units
 Double Buffered Output Compare Registers

- One Input Capture Unit
- Input Capture Noise Canceler
- · Clear Timer on Compare Match (Auto Reload)
- Glitch-free, Phase Correct Pulse Width Modulator (PWM)
 Variable PWM Period
- Frequency Generator
- External Event Counter
- · Four Independent Interrupt Sources (TOV1, OCF1A, OCF1B, and ICF1)

Overview

Most register and bit references in this section are written in general form. A lower case "n" replaces the Timer/Counter number, and a lower case "x" replaces the output compare unit. However, when using the register or bit defines in a program, the precise form must be used (that is, TCNT1 for accessing Timer/Counter1 counter value and so on).

A simplified block diagram of the 16-bit Timer/Counter is shown in Figure 40. For the actual placement of I/O pins, refer to Figure 1 on page 2. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device specific I/O Register and bit locations are listed in the "16-bit Timer/Counter Register Description" on page 110.





Count TOVn (Int.Req.) Control Logic Clock Select Edge Detector Tn воттом TOP (From Prescaler) **TCNTn** = 0OCnA (Int.Req.) OCnA Generation OCRnA Fixed OCnB (Int.Req.) DATABUS Values Waveform OCnB OCRnB (From Analog Comparator Ouput) ►ICFn (Int.Req.) ICPn TCCRnA TCCRnB

Figure 40. 16-bit Timer/Counter Block Diagram⁽¹⁾

Note: 1. Refer to Figure 1 on page 2, Table 25 on page 58, and Table 31 on page 63 for Timer/Counter1 pin placement and description.

Registers

The Timer/Counter (TCNT1), Output Compare Registers (OCR1A/B), and Input Capture Register (ICR1) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in the section "Accessing 16-bit Registers" on page 92. The Timer/Counter Control Registers (TCCR1A/B) are 8-bit registers and have no CPU access restrictions. Interrupt requests (abbreviated to Int.Req. in the figure) signals are all visible in the Timer Interrupt Flag Register (TIFR). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK). TIFR and TIMSK are not shown in the figure since these registers are shared by other timer units.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the T1 pin. The Clock Select logic block controls which clock source and edge the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the clock select logic is referred to as the timer clock (clk_{T1}).

The double buffered Output Compare Registers (OCR1A/B) are compared with the Timer/Counter value at all time. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pin (OC1A/B). See "Output Compare Units" on page 98. The compare match event will also set the Compare Match Flag (OCF1A/B) which can be used to generate an output compare interrupt request.



90



The Input Capture Register can capture the Timer/Counter value at a given external (edge triggered) event on either the Input Capture Pin (ICP1) or on the Analog Comparator pins (See "Analog Comparator" on page 201.) The Input Capture unit includes a digital filtering unit (Noise Canceler) for reducing the chance of capturing noise spikes.

The TOP value, or maximum Timer/Counter value, can in some modes of operation be defined by either the OCR1A Register, the ICR1 Register, or by a set of fixed values. When using OCR1A as TOP value in a PWM mode, the OCR1A Register can not be used for generating a PWM output. However, the TOP value will in this case be double buffered allowing the TOP value to be changed in run time. If a fixed TOP value is required, the ICR1 Register can be used as an alternative, freeing the OCR1A to be used as PWM output.

Definitions

The following definitions are used extensively throughout the document:

Table 43. Definitions

BOTTOM	The counter reaches the BOTTOM when it becomes 0x0000.
MAX	The counter reaches its MAXimum when it becomes 0xFFFF (decimal 65535).
TOP	The counter reaches the <i>TOP</i> when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be one of the fixed values: 0x00FF, 0x01FF, or 0x03FF, or to the value stored in the OCR1A or ICR1 Register. The assignment is dependent of the mode of operation.

Compatibility

The 16-bit Timer/Counter has been updated and improved from previous versions of the 16-bit AVR Timer/Counter. This 16-bit Timer/Counter is fully compatible with the earlier version regarding:

- All 16-bit Timer/Counter related I/O Register address locations, including Timer Interrupt Registers.
- · Bit locations inside all 16-bit Timer/Counter Registers, including Timer Interrupt Registers.
- · Interrupt Vectors.

The following control bits have changed name, but have same functionality and register location:

- PWM10 is changed to WGM10.
- · PWM11 is changed to WGM11.
- CTC1 is changed to WGM12.

The following bits are added to the 16-bit Timer/Counter Control Registers:

- FOC1A and FOC1B are added to TCCR1A.
- · WGM13 is added to TCCR1B.

The 16-bit Timer/Counter has improvements that will affect the compatibility in some special cases.





Accessing 16-bit Registers

The TCNT1, OCR1A/B, and ICR1 are 16-bit registers that can be accessed by the AVR CPU via the 8-bit data bus. The 16-bit register must be byte accessed using two read or write operations. Each 16-bit timer has a single 8-bit register for temporary storing of the High byte of the 16-bit access. The same temporary register is shared between all 16-bit registers within each 16-bit timer. Accessing the Low byte triggers the 16-bit read or write operation. When the Low byte of a 16-bit register is written by the CPU, the High byte stored in the temporary register, and the Low byte written are both copied into the 16-bit register in the same clock cycle. When the Low byte of a 16-bit register is read by the CPU, the High byte of the 16-bit register is copied into the temporary register in the same clock cycle as the Low byte is read.

Not all 16-bit accesses uses the temporary register for the High byte. Reading the OCR1A/B 16-bit registers does not involve using the temporary register.

To do a 16-bit write, the High byte must be written before the Low byte. For a 16-bit read, the Low byte must be read before the High byte.

The following code examples show how to access the 16-bit Timer Registers assuming that no interrupts updates the temporary register. The same principle can be used directly for accessing the OCR1A/B and ICR1 Registers. Note that when using "C", the compiler handles the 16-bit access.

```
Assembly Code Example(1)

...

; Set TCNT1 to 0x01FF

ldi r17,0x01

ldi r16,0xFF

out TCNT1L,r17

out TCNT1L,r16

; Read TCNT1 into r17:r16

in r16,TCNT1L

in r17,TCNT1H

...

C Code Example(1)

unsigned int i;

...

/* Set TCNT1 to 0x01FF */

TCNT1 = 0x1FF;

/* Read TCNT1 into i */

i = TCNT1;

...
```

Note: 1. See "About Code Examples" on page 7.

The assembly code example returns the TCNT1 value in the r17:r16 register pair.

It is important to notice that accessing 16-bit registers are atomic operations. If an interrupt occurs between the two instructions accessing the 16-bit register, and the interrupt code updates the temporary register by accessing the same or any other of the 16-bit Timer Registers, then the result of the access outside the interrupt will be corrupted. Therefore, when both the main code and the interrupt code update the temporary register, the main code must disable the interrupts during the 16-bit access.



92



The following code examples show how to do an atomic read of the TCNT1 Register contents. Reading any of the OCR1A/B or ICR1 Registers can be done by using the same principle.

```
Assembly Code Example<sup>(1)</sup>
    TIM16_ReadTCNT1:
     ; Save global interrupt flag
     in r18, SREG
     ; Disable interrupts
     cli
     ; Read TCNT1 into r17:r16
     in r16,TCNT1L
     in r17, TCNT1H
     ; Restore global interrupt flag
     out SREG, r18
C Code Example<sup>(1)</sup>
    unsigned int TIM16 ReadTCNT1( void )
     unsigned char sreg;
     unsigned int i;
     /* Save global interrupt flag */
     sreg = SREG;
     /* Disable interrupts */
     _CLI();
     /* Read TCNT1 into i */
     i = TCNT1;
     /* Restore global interrupt flag */
     SREG = sreg;
```

Note: 1. See "About Code Examples" on page 7.

The assembly code example returns the TCNT1 value in the r17:r16 register pair.



93



The following code examples show how to do an atomic write of the TCNT1 Register contents. Writing any of the OCR1A/B or ICR1 Registers can be done by using the same principle.

```
Assembly Code Example(1)
    TIM16 WriteTCNT1:
     ; Save global interrupt flag
     in r18, SREG
     ; Disable interrupts
     cli
     ; Set TCNT1 to r17:r16
     out TCNT1H, r17
     out TCNT1L, r16
     ; Restore global interrupt flag
     out SREG, r18
C Code Example<sup>(1)</sup>
    void TIM16 WriteTCNT1 ( unsigned int i )
     unsigned char sreq;
     unsigned int i;
     /* Save global interrupt flag */
     sreg = SREG;
     /* Disable interrupts */
     _CLI();
     /* Set TCNT1 to i */
     TCNT1 = i;
     /* Restore global interrupt flag */
     SREG = sreg;
```

Note: 1. See "About Code Examples" on page 7.

The assembly code example requires that the r17:r16 register pair contains the value to be written to TCNT1.

Reusing the Temporary High Byte Register If writing to more than one 16-bit register where the High byte is the same for all registers written, then the High byte only needs to be written once. However, note that the same rule of atomic operation described previously also applies in this case.

Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the *Clock Select* (CS12:0) bits located in the *Timer/Counter Control Register B* (TCCR1B). For details on clock sources and prescaler, see "Timer/Counter0 and Timer/Counter1 Prescalers" on page 87.

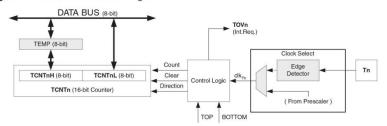




Counter Unit

The main part of the 16-bit Timer/Counter is the programmable 16-bit bi-directional counter unit. Figure 41 shows a block diagram of the counter and its surroundings.

Figure 41. Counter Unit Block Diagram



Signal description (internal signals):

Count Increment or decrement TCNT1 by 1.

Direction Select between increment and decrement.

Clear TCNT1 (set all bits to zero).

clk_{T1} Timer/Counter clock.

TOP Signalize that TCNT1 has reached maximum value.BOTTOM Signalize that TCNT1 has reached minimum value (zero).

The 16-bit counter is mapped into two 8-bit I/O memory locations: Counter High (TCNT1H) containing the upper eight bits of the counter, and Counter Low (TCNT1L) containing the lower 8 bits. The TCNT1H Register can only be indirectly accessed by the CPU. When the CPU does an access to the TCNT1H I/O location, the CPU accesses the High byte temporary register (TEMP). The temporary register is updated with the TCNT1H value when the TCNT1L is read, and TCNT1H is updated with the temporary register value when TCNT1L is written. This allows the CPU to read or write the entire 16-bit counter value within one clock cycle via the 8-bit data bus. It is important to notice that there are special cases of writing to the TCNT1 Register when the counter is counting that will give unpredictable results. The special cases are described in the sections where they are of importance.

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each *timer clock* (clk_{T_1}). The clk_{T_1} can be generated from an external or internal clock source, selected by the *Clock Select* bits (CS12:0). When no clock source is selected (CS12:0 = 0) the timer is stopped. However, the TCNT1 value can be accessed by the CPU, independent of whether clk_{T_1} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the *Waveform Generation Mode* bits (WGM13:0) located in the *Timer/Counter Control Registers* A and B (TCCR1A and TCCR1B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC1x. For more details about advanced counting sequences and waveform generation, see "Modes of Operation" on page 101.

The *Timer/Counter Overflow* (TOV1) Flag is set according to the mode of operation selected by the WGM13:0 bits. TOV1 can be used for generating a CPU interrupt.



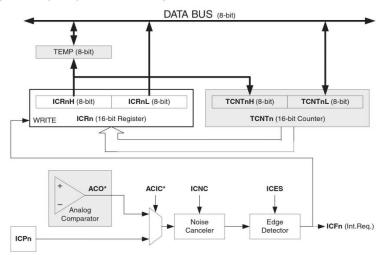
95



Input Capture Unit The Timer/Counter incorporates an Input Capture unit that can capture external events and give them a time-stamp indicating time of occurrence. The external signal indicating an event, or multiple events, can be applied via the ICP1 pin or alternatively, via the Analog Comparator unit. The time-stamps can then be used to calculate frequency, duty-cycle, and other features of the signal applied. Alternatively the time-stamps can be used for creating a log of the events.

> The Input Capture unit is illustrated by the block diagram shown in Figure 42. The elements of the block diagram that are not directly a part of the Input Capture unit are gray shaded. The small "n" in register and bit names indicates the Timer/Counter number.

Figure 42. Input Capture Unit Block Diagram



When a change of the logic level (an event) occurs on the Input Capture pin (ICP1), alternatively on the Analog Comparator output (ACO), and this change confirms to the setting of the edge detector, a capture will be triggered. When a capture is triggered, the 16-bit value of the counter (TCNT1) is written to the Input Capture Register (ICR1). The Input Capture Flag (ICF1) is set at the same system clock as the TCNT1 value is copied into ICR1 Register. If enabled (TICIE1 = 1), the Input Capture Flag generates an Input Capture Interrupt. The ICF1 Flag is automatically cleared when the interrupt is executed. Alternatively the ICF1 Flag can be cleared by software by writing a logical one to its I/O bit location.

Reading the 16-bit value in the Input Capture Register (ICR1) is done by first reading the Low byte (ICR1L) and then the High byte (ICR1H). When the Low byte is read the High byte is copied into the High byte temporary register (TEMP). When the CPU reads the ICR1H I/O location it will access the TEMP Register.

The ICR1 Register can only be written when using a Waveform Generation mode that utilizes the ICR1 Register for defining the counter's TOP value. In these cases the Waveform Generation mode (WGM13:0) bits must be set before the TOP value can be written to the ICR1 Register. When writing the ICR1 Register the High byte must be written to the ICR1H I/O location before the Low byte is written to ICR1L.



96



For more information on how to access the 16-bit registers refer to "Accessing 16-bit Registers" on page 92.

Input Capture Pin Source

The main trigger source for the Input Capture unit is the *Input Capture pin* (ICP1). Timer/Counter1 can alternatively use the Analog Comparator output as trigger source for the Input Capture unit. The Analog Comparator is selected as trigger source by setting the *Analog Comparator Input Capture* (ACIC) bit in the *Analog Comparator Control and Status Register* (ACSR). Be aware that changing trigger source can trigger a capture. The Input Capture Flag must therefore be cleared after the change.

Both the *Input Capture pin* (ICP1) and the *Analog Comparator output* (ACO) inputs are sampled using the same technique as for the T1 pin (Figure 38 on page 87). The edge detector is also identical. However, when the noise canceler is enabled, additional logic is inserted before the edge detector, which increases the delay by four system clock cycles. Note that the input of the noise canceler and edge detector is always enabled unless the Timer/Counter is set in a waveform generation mode that uses ICR1 to define TOP.

An Input Capture can be triggered by software by controlling the port of the ICP1 pin.

Noise Canceler

The noise canceler improves noise immunity by using a simple digital filtering scheme. The noise canceler input is monitored over four samples, and all four must be equal for changing the output that in turn is used by the edge detector.

The noise canceler is enabled by setting the *Input Capture Noise Canceler* (ICNC1) bit in *Timer/Counter Control Register B* (TCCR1B). When enabled the noise canceler introduces additional four system clock cycles of delay from a change applied to the input, to the update of the ICR1 Register. The noise canceler uses the system clock and is therefore not affected by the prescaler.

Using the Input Capture Unit

The main challenge when using the Input Capture unit is to assign enough processor capacity for handling the incoming events. The time between two events is critical. If the processor has not read the captured value in the ICR1 Register before the next event occurs, the ICR1 will be overwritten with a new value. In this case the result of the capture will be incorrect.

When using the Input Capture Interrupt, the ICR1 Register should be read as early in the interrupt handler routine as possible. Even though the Input Capture Interrupt has relatively high priority, the maximum interrupt response time is dependent on the maximum number of clock cycles it takes to handle any of the other interrupt requests.

Using the Input Capture unit in any mode of operation when the TOP value (resolution) is actively changed during operation, is not recommended.

Measurement of an external signal's duty cycle requires that the trigger edge is changed after each capture. Changing the edge sensing must be done as early as possible after the ICR1 Register has been read. After a change of the edge, the Input Capture Flag (ICF1) must be cleared by software (writing a logical one to the I/O bit location). For measuring frequency only, the clearing of the ICF1 Flag is not required (if an interrupt handler is used).





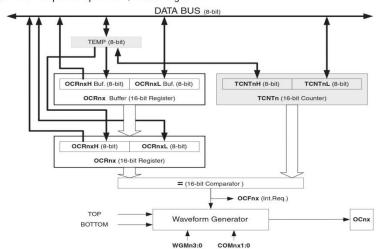
Output Compare Units

The 16-bit comparator continuously compares TCNT1 with the *Output Compare Register* (OCR1x). If TCNT equals OCR1x the comparator signals a match. A match will set the *Output Compare Flag* (OCF1x) at the next timer clock cycle. If enabled (OCIE1x = 1), the Output Compare Flag generates an output compare interrupt. The OCF1x Flag is automatically cleared when the interrupt is executed. Alternatively the OCF1x Flag can be cleared by software by writing a logical one to its I/O bit location. The Waveform Generator uses the match signal to generate an output according to operating mode set by the *Waveform Generation mode* (WGM13:0) bits and *Compare Output mode* (COM1x1:0) bits. The TOP and BOTTOM signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation (See "Modes of Operation" on page 101.)

A special feature of output compare unit A allows it to define the Timer/Counter TOP value (that is, counter resolution). In addition to the counter resolution, the TOP value defines the period time for waveforms generated by the Waveform Generator.

Figure 43 shows a block diagram of the output compare unit. The small "n" in the register and bit names indicates the device number (n = 1 for Timer/Counter1), and the "x" indicates output compare unit (A/B). The elements of the block diagram that are not directly a part of the output compare unit are gray shaded.

Figure 43. Output Compare Unit, Block Diagram



The OCR1x Register is double buffered when using any of the twelve *Pulse Width Modulation* (PWM) modes. For the normal and *Clear Timer on Compare* (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR1x Compare Register to either TOP or BOTTOM of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR1x Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR1x Buffer Register, and if double buffering is disabled the CPU will access the OCR1x directly. The content of the OCR1x (Buffer or Compare) Register is only changed by a write operation (the Timer/Counter does not update this register automatically as the TCNT1 and ICR1 Register). Therefore OCR1x is not read via the High byte



98



temporary register (TEMP). However, it is a good practice to read the Low byte first as when accessing other 16-bit registers. Writing the OCR1x Registers must be done via the TEMP Register since the compare of all 16 bits is done continuously. The High byte (OCR1xH) has to be written first. When the High byte I/O location is written by the CPU, the TEMP Register will be updated by the value written. Then when the Low byte (OCR1xL) is written to the lower eight bits, the High byte will be copied into the upper 8-bits of either the OCR1x buffer or OCR1x Compare Register in the same system clock cycle.

For more information of how to access the 16-bit registers refer to "Accessing 16-bit Registers" on page 92.

Force Output Compare

In non-PWM Waveform Generation modes, the match output of the comparator can be forced by writing a one to the *Force Output Compare* (FOC1x) bit. Forcing compare match will not set the OCF1x Flag or reload/clear the timer, but the OC1x pin will be updated as if a real compare match had occurred (the COM1x1:0 bits settings define whether the OC1x pin is set, cleared or toggled).

Compare Match Blocking by TCNT1 Write All CPU writes to the TCNT1 Register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR1x to be initialized to the same value as TCNT1 without triggering an interrupt when the Timer/Counter clock is enabled.

Using the Output Compare Unit Since writing TCNT1 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT1 when using any of the output compare units, independent of whether the Timer/Counter is running or not. If the value written to TCNT1 equals the OCR1x value, the compare match will be missed, resulting in incorrect waveform generation. Do not write the TCNT1 equal to TOP in PWM modes with variable TOP values. The compare match for the TOP will be ignored and the counter will continue to 0xFFFF. Similarly, do not write the TCNT1 value equal to BOTTOM when the counter is downcounting.

The setup of the OC1x should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC1x value is to use the force output compare (FOC1x) strobe bits in Normal mode. The OC1x Register keeps its value even when changing between waveform generation modes.

Be aware that the COM1x1:0 bits are not double buffered together with the compare value. Changing the COM1x1:0 bits will take effect immediately.

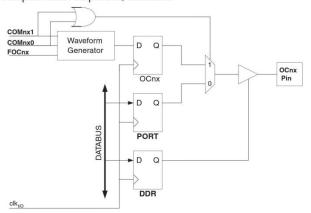




Compare Match Output Unit

The Compare Output mode (COM1x1:0) bits have two functions. The Waveform Generator uses the COM1x1:0 bits for defining the Output Compare (OC1x) state at the next compare match. Secondly the COM1x1:0 bits control the OC1x pin output source. Figure 44 shows a simplified schematic of the logic affected by the COM1x1:0 bit setting. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O Port Control Registers (DDR and PORT) that are affected by the COM1x1:0 bits are shown. When referring to the OC1x state, the reference is for the internal OC1x Register, not the OC1x pin. If a System Reset occur, the OC1x Register is reset to "0".

Figure 44. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the Output Compare (OC1x) from the Waveform Generator if either of the COM1x1:0 bits are set. However, the OC1x pin direction (input or output) is still controlled by the *Data Direction Register* (DDR) for the port pin. The Data Direction Register bit for the OC1x pin (DDR_OC1x) must be set as output before the OC1x value is visible on the pin. The port override function is generally independent of the Waveform Generation mode, but there are some exceptions. Refer to Table 44, Table 45 and Table 46 for details.

The design of the output compare pin logic allows initialization of the OC1x state before the output is enabled. Note that some COM1x1:0 bit settings are reserved for certain modes of operation. See "16-bit Timer/Counter Register Description" on page 110.

The COM1x1:0 bits have no effect on the Input Capture unit.

Compare Output Mode and Waveform Generation The Waveform Generator uses the COM1x1:0 bits differently in normal, CTC, and PWM modes. For all modes, setting the COM1x1:0 = 0 tells the Waveform Generator that no action on the OC1x Register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 44 on page 110. For fast PWM mode refer to Table 45 on page 111, and for phase correct and phase and frequency correct PWM refer to Table 46 on page 111

A change of the COM1x1:0 bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC1x strobe bits.



100



Modes of Operation

The mode of operation, that is, the behavior of the Timer/Counter and the output compare pins, is defined by the combination of the *Waveform Generation mode* (WGM13:0) and *Compare Output mode* (CGM1x1:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM1x1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM1x1:0 bits control whether the output should be set, cleared or toggle at a compare match (See "Compare Match Output Unit" on page 100.)

For detailed timing information refer to "Timer/Counter Timing Diagrams" on page 108.

Normal Mode

The simplest mode of operation is the *Normal* mode (WGM13:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 16-bit value (MAX = 0xFFFF) and then restarts from the BOTTOM (0x0000). In normal operation the *Timer/Counter Overflow Flag* (TOV1) will be set in the same timer clock cycle as the TCNT1 becomes zero. The TOV1 Flag in this case behaves like a 17th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV1 Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

The Input Capture unit is easy to use in Normal mode. However, observe that the maximum interval between the external events must not exceed the resolution of the counter. If the interval between events are too long, the timer overflow interrupt or the prescaler must be used to extend the resolution for the capture unit.

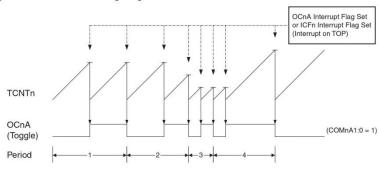
The output compare units can be used to generate interrupts at some given time. Using the output compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM13:0 = 4 or 12), the OCR1A or ICR1 Register are used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT1) matches either the OCR1A (WGM13:0 = 4) or the ICR1 (WGM13:0 = 12). The OCR1A or ICR1 define the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 45. The counter value (TCNT1) increases until a compare match occurs with either OCR1A or ICR1, and then counter (TCNT1) is cleared.

Figure 45. CTC Mode, Timing Diagram



101



An interrupt can be generated at each time the counter value reaches the TOP value by either using the OCF1A or ICF1 Flag according to the register used to define the TOP value. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing the TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR1A or ICR1 is lower than the current value of TCNT1, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. In many cases this feature is not desirable. An alternative will then be to use the fast PWM mode using OCR1A for defining TOP (WGM13:0 = 15) since the OCR1A then will be double buffered.

For generating a waveform output in CTC mode, the OC1A output can be set to toggle its logical level on each compare match by setting the compare output mode bits to toggle mode (COM1A1:0 = 1). The OC1A value will not be visible on the port pin unless the data direction for the pin is set to output (DDR_OC1A = 1). The waveform generated will have a maximum frequency of $f_{\rm OC1A} = f_{\rm clk_I/O}/2$ when OCR1A is set to zero (0x0000). The waveform frequency is defined by the following equation:

$$f_{OCnA} = \frac{f_{\text{clk_I/O}}}{2 \cdot N \cdot (1 + OCRnA)}$$

The N variable represents the prescaler factor (1, 8, 64, 256, or 1024).

As for the Normal mode of operation, the TOV1 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x0000.

Fast PWM Mode

The fast Pulse Width Modulation or fast PWM mode (WGM13:0 = 5,6,7,14, or 15) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM options by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x, and set at BOTTOM. In inverting Compare Output mode output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct and phase and frequency correct PWM modes that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), hence reduces total system cost.

The PWM resolution for fast PWM can be fixed to 8-bit, 9-bit, or 10-bit, or defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated by using the following equation:

$$R_{FPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

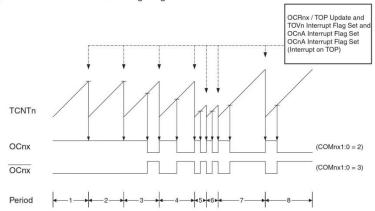
In fast PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM13:0 = 5, 6, or 7), the value in ICR1 (WGM13:0 = 14), or the value in OCR1A (WGM13:0 = 15). The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 46. The figure shows fast PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x Interrupt Flag will be set when a compare match occurs.



102



Figure 46. Fast PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV1) is set each time the counter reaches TOP. In addition the OC1A or ICF1 Flag is set at the same timer clock cycle as TOV1 is set when either OCR1A or ICR1 is used for defining the TOP value. If one of the interrupts are enabled, the interrupt handler routine can be used for updating the TOP and compare values.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the Compare Registers. If the TOP value is lower than any of the Compare Registers, a compare match will never occur between the TCNT1 and the OCR1x. Note that when using fixed TOP values the unused bits are masked to zero when any of the OCR1x Registers are written.

The procedure for updating ICR1 differs from updating OCR1A when used for defining the TOP value. The ICR1 Register is not double buffered. This means that if ICR1 is changed to a low value when the counter is running with none or a low prescaler value, there is a risk that the new ICR1 value written is lower than the current value of TCNT1. The result will then be that the counter will miss the compare match at the TOP value. The counter will then have to count to the MAX value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. The OCR1A Register however, is double buffered. This feature allows the OCR1A I/O location to be written anytime. When the OCR1A I/O location is written the value written will be put into the OCR1A Buffer Register. The OCR1A Compare Register will then be updated with the value in the Buffer Register at the next timer clock cycle the TCNT1 matches TOP. The update is done at the same timer clock cycle as the TCNT1 is cleared and the TOV1 Flag is set.

Using the ICR1 Register for defining TOP works well when using fixed TOP values. By using ICR1, the OCR1A Register is free to be used for generating a PWM output on OC1A. However, if the base PWM frequency is actively changed (by changing the TOP value), using the OCR1A as TOP is clearly a better choice due to its double buffer feature.

In fast PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to 2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to 3 (See Table 44 on page 110). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x Register at the compare match between OCR1x and TCNT1, and clearing (or setting) the OC1x Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).



103



The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnxPWM} = \frac{f_{clk_I/O}}{N \cdot (1 + TOP)}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1x Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR1x is set equal to BOTTOM (0x0000) the output will be a narrow spike for each TOP+1 timer clock cycle. Setting the OCR1x equal to TOP will result in a constant high or low output (depending on the polarity of the output set by the COM1x1:0 bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC1A to toggle its logical level on each compare match (COM1A1:0 = 1). This applies only if OCR1A is used to define the TOP value (WGM13:0 = 15). The waveform generated will have a maximum frequency of $f_{\rm OC1A} = f_{\rm clk_I/O}/2$ when OCR1A is set to zero (0x0000). This feature is similar to the OC1A toggle in CTC mode, except the double buffer feature of the output compare unit is enabled in the fast PWM mode.

Phase Correct PWM Mode The phase correct Pulse Width Modulation or phase correct PWM mode (WGM13:0 = 1,2,3,10, or 11) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is, like the phase and frequency correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x while upcounting, and set on the compare match while downcounting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The PWM resolution for the phase correct PWM mode can be fixed to 8-bit, 9-bit, or 10-bit, or defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated by using the following equation:

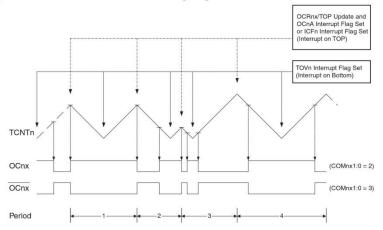
$$R_{PCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

In phase correct PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM13:0 = 1, 2, or 3), the value in ICR1 (WGM13:0 = 10), or the value in OCR1A (WGM13:0 = 11). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 47. The figure shows phase correct PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x Interrupt Flag will be set when a compare match occurs.





Figure 47. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV1) is set each time the counter reaches BOTTOM. When either OCR1A or ICR1 is used for defining the TOP value, the OC1A or ICF1 Flag is set accordingly at the same timer clock cycle as the OCR1x Registers are updated with the double buffer value (at TOP). The Interrupt Flags can be used to generate an interrupt each time the counter reaches the TOP or BOTTOM value.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the Compare Registers. If the TOP value is lower than any of the Compare Registers, a compare match will never occur between the TCNT1 and the OCR1x. Note that when using fixed TOP values, the unused bits are masked to zero when any of the OCR1x Registers are written. As the third period shown in Figure 47 illustrates, changing the TOP actively while the Timer/Counter is running in the phase correct mode can result in an unsymmetrical output. The reason for this can be found in the time of update of the OCR1x Register. Since the OCR1x update occurs at TOP, the PWM period starts and ends at TOP. This implies that the length of the falling slope is determined by the previous TOP value, while the length of the rising slope is determined by the new TOP value. When these two values differ the two slopes of the period will differ in length. The difference in length gives the unsymmetrical result on the output.

It is recommended to use the phase and frequency correct mode instead of the phase correct mode when changing the TOP value while the Timer/Counter is running. When using a static TOP value there are practically no differences between the two modes of operation.

In phase correct PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to 2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to 3 (See Table 44 on page 110). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x Register at the compare match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OC1x Register at compare match between OCR1x and TCNT1 when



105



the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk_l/O}}{2 \cdot N \cdot TOP}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1x Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR1x is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCR1A is used to define the TOP value (WGM13:0 = 11) and COM1A1:0 = 1, the OC1A output will toggle with a 50% duty cycle.

Phase and Frequency Correct PWM Mode

The phase and frequency correct Pulse Width Modulation, or phase and frequency correct PWM mode (WGM13:0 = 8 or 9) provides a high resolution phase and frequency correct PWM waveform generation option. The phase and frequency correct PWM mode is, like the phase correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is cleared on the compare match between TCN11 and OCR1x while upcounting, and set on the compare match while downcounting. In inverting Compare Output mode, the operation is inverted. The dual-slope operation gives a lower maximum operation frequency compared to the single-slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The main difference between the phase correct, and the phase and frequency correct PWM mode is the time the OCR1x Register is updated by the OCR1x Buffer Register, (see Figure 47 and Figure 48).

The PWM resolution for the phase and frequency correct PWM mode can be defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated using the following equation:

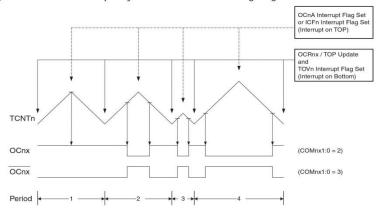
$$R_{PFCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

In phase and frequency correct PWM mode the counter is incremented until the counter value matches either the value in ICR1 (WGM13:0 = 8), or the value in OCR1A (WGM13:0 = 9). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct and frequency correct PWM mode is shown on Figure 48. The figure shows phase and frequency correct PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x Interrupt Flag will be set when a compare match occurs.





Figure 48. Phase and Frequency Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV1) is set at the same timer clock cycle as the OCR1x Registers are updated with the double buffer value (at BOTTOM). When either OCR1A or ICR1 is used for defining the TOP value, the OC1A or ICF1 Flag set when TCNT1 has reached TOP. The Interrupt Flags can then be used to generate an interrupt each time the counter reaches the TOP or BOTTOM value.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the Compare Registers. If the TOP value is lower than any of the Compare Registers, a compare match will never occur between the TCNT1 and the OCR1x.

As Figure 48 shows the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Since the OCR1x Registers are updated at BOTTOM, the length of the rising and the falling slopes will always be equal. This gives symmetrical output pulses and is therefore frequency correct.

Using the ICR1 Register for defining TOP works well when using fixed TOP values. By using ICR1, the OCR1A Register is free to be used for generating a PWM output on OC1A. However, if the base PWM frequency is actively changed by changing the TOP value, using the OCR1A as TOP is clearly a better choice due to its double buffer feature.

In phase and frequency correct PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to 2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to 3 (See Table on page 111). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x Register at the compare match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OC1x Register at compare match between OCR1x and TCNT1 when the counter decrements. The PWM frequency for the output when using phase and frequency correct PWM can be calculated by the following equation:

$$f_{OCnxPFCPWM} = \frac{f_{clk_l/O}}{2 \cdot N \cdot TOP}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1x Register represents special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR1x is set equal to BOTTOM the



107



output will be continuously low and if set equal to TOP the output will be set to high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCR1A is used to define the TOP value (WGM13:0 = 9) and COM1A1:0 = 1, the OC1A output will toggle with a 50% duty cycle.

Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T1}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set, and when the OCR1x Register is updated with the OCR1x buffer value (only for modes utilizing double buffering). Figure 49 shows a timing diagram for the setting of OCF1x.

Figure 49. Timer/Counter Timing Diagram, Setting of OCF1x, No Prescaling

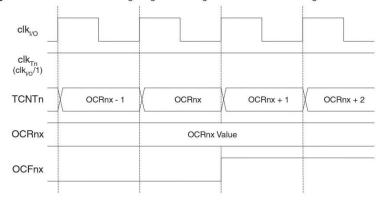


Figure 50 shows the same timing data, but with the prescaler enabled.

Figure 50. Timer/Counter Timing Diagram, Setting of OCF1x, with Prescaler ($f_{clk_l/O}/8$)

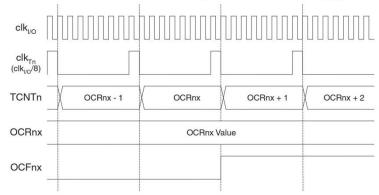


Figure 51 shows the count sequence close to TOP in various modes. When using phase and frequency correct PWM mode the OCR1x Register is updated at BOTTOM. The timing diagrams

AIMEL

108



will be the same, but TOP should be replaced by BOTTOM, TOP-1 by BOTTOM+1 and so on. The same renaming applies for modes that set the TOV1 Flag at BOTTOM.

Figure 51. Timer/Counter Timing Diagram, no Prescaling

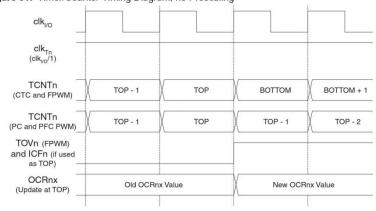
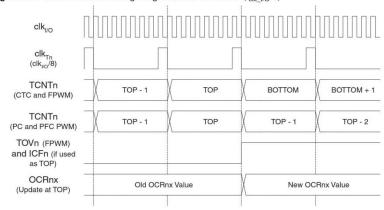


Figure 52 shows the same timing data, but with the prescaler enabled.

Figure 52. Timer/Counter Timing Diagram, with Prescaler ($f_{clk_I/O}/8$)



ATMEL

109



16-bit Timer/Counter Register Description

Timer/Counter1 Control Register A – TCCR1A

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 7:6 COM1A1:0: Compare Output Mode for Channel A
- Bit 5:4 COM1B1:0: Compare Output Mode for Channel B

The COM1A1:0 and COM1B1:0 control the Output Compare pins (OC1A and OC1B respectively) behavior. If one or both of the COM1A1:0 bits are written to one, the OC1A output overrides the normal port functionality of the I/O pin it is connected to. If one or both of the COM1B1:0 bit are written to one, the OC1B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the *Data Direction Register* (DDR) bit corresponding to the OC1A or OC1B pin must be set in order to enable the output driver.

When the OC1A or OC1B is connected to the pin, the function of the COM1x1:0 bits is dependent of the WGM13:0 bits setting. Table 44 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to a normal or a CTC mode (non-PWM).

Table 44. Compare Output Mode, non-PWM

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	Toggle OC1A/OC1B on compare match
1	0	Clear OC1A/OC1B on compare match (Set output to low level)
1	1	Set OC1A/OC1B on compare match (Set output to high level)

Table 45 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to the fast PWM mode.



Table 45. Compare Output Mode, Fast PWM(1)

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 15: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM13:0 settings, normal port operation, OCnA/OCnB disconnected.
1	0	Clear OC1A/OC1B on compare match, set OC1A/OC1B at BOTTOM, (non-inverting mode)
1	1	Set OC1A/OC1B on compare match, clear OC1A/OC1B at BOTTOM, (inverting mode)

Note: 1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. In this case the compare match is ignored, but the set or clear is done at BOTTOM. See "Fast PWM Mode" on page 102. for more details.

Table 46 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to the phase correct or the phase and frequency correct, PWM mode.

Table 46. Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM (1)

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 9 or 14: Toggle OCnA on Compare Match, OCnB disconnected (normal port operation). For all other WGM13:0 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on compare match when up-counting. Set OC1A/OC1B on compare match when downcounting.
1	1	Set OC1A/OC1B on compare match when up- counting. Clear OC1A/OC1B on compare match when downcounting.

Note: 1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. See "Phase Correct PWM Mode" on page 104. for more details.

- · Bit 3 FOC1A: Force Output Compare for Channel A
- Bit 2 FOC1B: Force Output Compare for Channel B

The FOC1A/FOC1B bits are only active when the WGM13:0 bits specifies a non-PWM mode. However, for ensuring compatibility with future devices, these bits must be set to zero when TCCR1A is written when operating in a PWM mode. When writing a logical one to the FOC1A/FOC1B bit, an immediate compare match is forced on the Waveform Generation unit. The OC1A/OC1B output is changed according to its COM1x1:0 bits setting. Note that the FOC1A/FOC1B bits are implemented as strobes. Therefore it is the value present in the COM1x1:0 bits that determine the effect of the forced compare.



111



A FOC1A/FOC1B strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare match (CTC) mode using OCR1A as TOP.

The FOC1A/FOC1B bits are always read as zero.

• Bit 1:0 - WGM11:0: Waveform Generation Mode

Combined with the WGM13:2 bits found in the TCCR1B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 47. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes. (See "Modes of Operation" on page 101.)

Table 47. Waveform Generation Mode Bit Description⁽¹⁾

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	тор	Update of OCR1X	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	воттом
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	воттом
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	воттом
4	0	1	0	0	стс	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	воттом	ТОР
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	воттом	ТОР
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	воттом	ТОР
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	воттом	воттом
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	воттом	воттом
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	воттом
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	воттом
12	1	1	0	0	стс	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	-	-	-
14	1	1	1	0	Fast PWM	ICR1	воттом	ТОР
15	1	1	1	1	Fast PWM	OCR1A	воттом	TOP

Note: 1. The CTC1 and PWM11:0 bit definition names are obsolete. Use the WGM12:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.





Timer/Counter1 Control Register B – TCCR1B

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	•3
Initial Value	0	0	0	0	0	0	0	0	

· Bit 7 - ICNC1: Input Capture Noise Canceler

Setting this bit (to one) activates the Input Capture Noise Canceler. When the Noise Canceler is activated, the input from the Input Capture Pin (ICP1) is filtered. The filter function requires four successive equal valued samples of the ICP1 pin for changing its output. The Input Capture is therefore delayed by four Oscillator cycles when the Noise Canceler is enabled.

· Bit 6 - ICES1: Input Capture Edge Select

This bit selects which edge on the Input Capture Pin (ICP1) that is used to trigger a capture event. When the ICES1 bit is written to zero, a falling (negative) edge is used as trigger, and when the ICES1 bit is written to one, a rising (positive) edge will trigger the capture.

When a capture is triggered according to the ICES1 setting, the counter value is copied into the Input Capture Register (ICR1). The event will also set the Input Capture Flag (ICF1), and this can be used to cause an Input Capture Interrupt, if this interrupt is enabled.

When the ICR1 is used as TOP value (see description of the WGM13:0 bits located in the TCCR1A and the TCCR1B Register), the ICP1 is disconnected and consequently the Input Capture function is disabled.

· Bit 5 - Reserved Bit

This bit is reserved for future use. For ensuring compatibility with future devices, this bit must be written to zero when TCCR1B is written.

• Bit 4:3 - WGM13:2: Waveform Generation Mode

See TCCR1A Register description.

• Bit 2:0 - CS12:0: Clock Select

The three Clock Select bits select the clock source to be used by the Timer/Counter, see Figure 49 and Figure 50.

Table 48. Clock Select Bit Description

CS12	CS11	CS10	Description				
0	0	0	No clock source (Timer/Counter stopped).				
0	0	1	clk _{I/O} /1 (No prescaling)				
0	1	0	clk _{I/O} /8 (From prescaler)				
0	1	1	clk _{I/O} /64 (From prescaler)				
1	0	0	clk _{I/O} /256 (From prescaler)				
1	0	1	clk _{I/O} /1024 (From prescaler)				
1	1	0	External clock source on T1 pin. Clock on falling edge.				
1	1	1	External clock source on T1 pin. Clock on rising edge.				

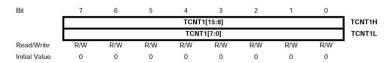


113



If external pin modes are used for the Timer/Counter1, transitions on the T1 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

Timer/Counter1 – TCNT1H and TCNT1L

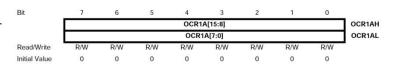


The two *Timer/Counter I/O* locations (TCNT1H and TCNT1L, combined TCNT1) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and Low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See "Accessing 16-bit Registers" on page 92.

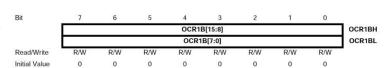
Modifying the counter (TCNT1) while the counter is running introduces a risk of missing a compare match between TCNT1 and one of the OCR1x Registers.

Writing to the TCNT1 Register blocks (removes) the compare match on the following timer clock for all compare units.

Output Compare Register 1 A – OCR1AH and OCR1AL



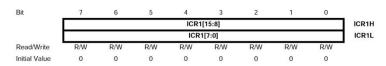
Output Compare Register 1 B – OCR1BH and OCR1BL



The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC1x pin.

The Output Compare Registers are 16-bit in size. To ensure that both the high and Low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See "Accessing 16-bit Registers" on page 92.

Input Capture Register 1 – ICR1H and ICR1L



AIMEL

114



The Input Capture is updated with the counter (TCNT1) value each time an event occurs on the ICP1 pin (or optionally on the analog comparator output for Timer/Counter1). The Input Capture can be used for defining the counter TOP value.

The Input Capture Register is 16-bit in size. To ensure that both the high and Low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See "Accessing 16-bit Registers" on page 92.

Timer/Counter Interrupt Mask Register – TIMSK⁽¹⁾

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	98
Initial Value	0	0	0	0	0	0	0	0	

Note: 1. This register contains interrupt control bits for several Timer/Counters, but only Timer1 bits are described in this section. The remaining bits are described in their respective timer sections.

• Bit 5 - TICIE1: Timer/Counter1, Input Capture Interrupt Enable

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Input Capture Interrupt is enabled. The corresponding Interrupt Vector (See "Interrupts" on page 45.) is executed when the ICF1 Flag, located in TIFR, is set.

• Bit 4 - OCIE1A: Timer/Counter1, Output Compare A Match Interrupt Enable

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Output Compare A match interrupt is enabled. The corresponding Interrupt Vector (See "Interrupts" on page 45.) is executed when the OCF1A Flag, located in TIFR. is set.

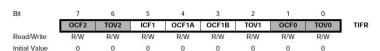
• Bit 3 - OCIE1B: Timer/Counter1, Output Compare B Match Interrupt Enable

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Output Compare B match interrupt is enabled. The corresponding Interrupt Vector (See "Interrupts" on page 45.) is executed when the OCF1B Flag, located in TIFR, is set.

• Bit 2 - TOIE1: Timer/Counter1, Overflow Interrupt Enable

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Overflow Interrupt is enabled. The corresponding Interrupt Vector (See "Interrupts" on page 45.) is executed when the TOV1 Flag, located in TIFR, is set.

Timer/Counter Interrupt Flag Register – TIFR



Note: This register contains flag bits for several Timer/Counters, but only Timer1 bits are described in this section. The remaining bits are described in their respective timer sections.

AIMEL

115



· Bit 5 - ICF1: Timer/Counter1, Input Capture Flag

This flag is set when a capture event occurs on the ICP1 pin. When the Input Capture Register (ICR1) is set by the WGM13:0 to be used as the TOP value, the ICF1 Flag is set when the counter reaches the TOP value.

ICF1 is automatically cleared when the Input Capture Interrupt Vector is executed. Alternatively, ICF1 can be cleared by writing a logic one to its bit location.

· Bit 4 - OCF1A: Timer/Counter1, Output Compare A Match Flag

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register A (OCR1A).

Note that a Forced Output Compare (FOC1A) strobe will not set the OCF1A Flag.

OCF1A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF1A can be cleared by writing a logic one to its bit location.

• Bit 3 - OCF1B: Timer/Counter1, Output Compare B Match Flag

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register B (OCR1B).

Note that a forced output compare (FOC1B) strobe will not set the OCF1B Flag.

OCF1B is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCF1B can be cleared by writing a logic one to its bit location.

• Bit 2 - TOV1: Timer/Counter1, Overflow Flag

The setting of this flag is dependent of the WGM13:0 bits setting. In normal and CTC modes, the TOV1 Flag is set when the timer overflows. Refer to Table 47 on page 112 for the TOV1 Flag behavior when using another WGM13:0 bit setting.

TOV1 is automatically cleared when the Timer/Counter1 Overflow interrupt vector is executed. Alternatively, TOV1 can be cleared by writing a logic one to its bit location.





8-bit

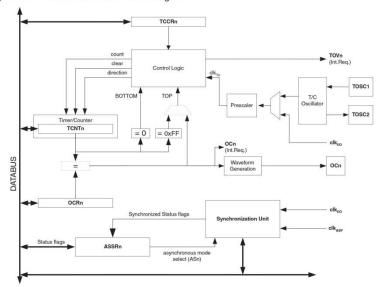
Timer/Counter2 with PWM and Asynchronous Operation Timer/Counter2 is a general purpose, single compare unit, 8-bit Timer/Counter module. The main features are:

- · Single Compare unit Counter
- · Clear Timer on Compare Match (Auto Reload)
- · Glitch-free, Phase Correct Pulse Width Modulator (PWM)
- Frequency Generator
- 10-bit Clock Prescaler
- Overflow and Compare Match Interrupt Sources (TOV2 and OCF2)
- Allows clocking from External 32 kHz Watch Crystal Independent of the I/O Clock

Overview

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 53. For the actual placement of I/O pins, refer to "Pinout ATmega16" on page 2. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the "8-bit Timer/Counter Register Description" on page 128.

Figure 53. 8-bit Timer/Counter Block Diagram



Registers

The Timer/Counter (TCNT2) and Output Compare Register (OCR2) are 8-bit registers. Interrupt request (shorten as Int.Req.) signals are all visible in the Timer Interrupt Flag Register (TIFR). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK). TIFR and TIMSK are not shown in the figure since these registers are shared by other timer units.

The Timer/Counter can be clocked internally, via the prescaler, or asynchronously clocked from the TOSC1/2 pins, as detailed later in this section. The asynchronous operation is controlled by the Asynchronous Status Register (ASSR). The Clock Select logic block controls which clock source the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock (clk_{T2}).



117



The double buffered Output Compare Register (OCR2) is compared with the Timer/Counter value at all times. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the Output Compare Pin (OC2). See "Output Compare Unit" on page 119. for details. The compare match event will also set the Compare Flag (OCF2) which can be used to generate an output compare interrupt request.

Definitions

Many register and bit references in this document are written in general form. A lower case "n" replaces the Timer/Counter number, in this case 2. However, when using the register or bit defines in a program, the precise form must be used (that is, TCNT2 for accessing Timer/Counter2 counter value and so on). The definitions in Table 49 are also used extensively throughout the document.

Table 49. Definitions

воттом	The counter reaches the BOTTOM when it becomes zero (0x00).
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR2 Register. The assignment is dependent on the mode of operation.

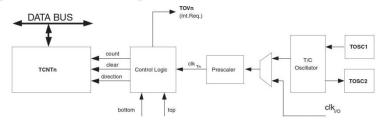
Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal synchronous or an external asynchronous clock source. The clock source ${\rm clk}_{\rm T2}$ is by default equal to the MCU clock, ${\rm clk}_{\rm I/O}$. When the AS2 bit in the ASSR Register is written to logic one, the clock source is taken from the Timer/Counter Oscillator connected to TOSC1 and TOSC2. For details on asynchronous operation, see "Asynchronous Status Register – ASSR" on page 131. For details on clock sources and prescaler, see "Timer/Counter Prescaler" on page 134.

Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 54 shows a block diagram of the counter and its surrounding environment.

Figure 54. Counter Unit Block Diagram



Signal description (internal signals):

count Increment or decrement TCNT2 by 1.direction Selects between increment and decrement.

clear TCNT2 (set all bits to zero).

clk_{T2} Timer/Counter clock.

top Signalizes that TCNT2 has reached maximum value.

AIMEL

118



bottom Signalizes that TCNT2 has reached minimum value (zero).

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock $(clk_{T2}).\ clk_{T2}$ can be generated from an external or internal clock source, selected by the Clock Select bits (CS22:0). When no clock source is selected (CS22:0 = 0) the timer is stopped. However, the TCNT2 value can be accessed by the CPU, regardless of whether clk_{T2} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

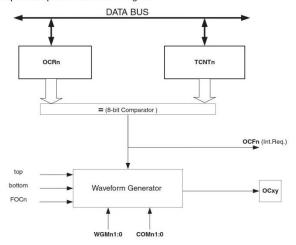
The counting sequence is determined by the setting of the WGM21 and WGM20 bits located in the Timer/Counter Control Register (TCCR2). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare output OC2. For more details about advanced counting sequences and waveform generation, see "Modes of Operation" on page 122.

The Timer/Counter Overflow (TOV2) Flag is set according to the mode of operation selected by the WGM21:0 bits. TOV2 can be used for generating a CPU interrupt.

Output Compare Unit

The 8-bit comparator continuously compares TCNT2 with the Output Compare Register (OCR2). Whenever TCNT2 equals OCR2, the comparator signals a match. A match will set the Output Compare Flag (OCF2) at the next timer clock cycle. If enabled (OCIE2 = 1), the Output Compare Flag generates an output compare interrupt. The OCF2 Flag is automatically cleared when the interrupt is executed. Alternatively, the OCF2 Flag can be cleared by software by writing a logical one to its I/O bit location. The waveform generator uses the match signal to generate an output according to operating mode set by the WGM21:0 bits and Compare Output mode (COM21:0) bits. The max and bottom signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation ("Modes of Operation" on page 122). Figure 55 shows a block diagram of the output compare unit.

Figure 55. Output Compare Unit, Block Diagram



The OCR2 Register is double buffered when using any of the Pulse Width Modulation (PWM) modes. For the normal and Clear Timer on Compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR2 Compare Register

<u>AIMEL</u>

119



to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR2 Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR2 Buffer Register, and if double buffering is disabled the CPU will access the OCR2 directly.

Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the Force Output Compare (FOC2) bit. Forcing compare match will not set the OCF2 Flag or reload/clear the timer, but the OC2 pin will be updated as if a real compare match had occurred (the COM21:0 bits settings define whether the OC2 pin is set, cleared or toggled).

Compare Match Blocking by TCNT2 Write All CPU write operations to the TCNT2 Register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR2 to be initialized to the same value as TCNT2 without triggering an interrupt when the Timer/Counter clock is enabled.

Using the Output Compare Unit Since writing TCNT2 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT2 when using the output compare unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT2 equals the OCR2 value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT2 value equal to BOTTOM when the counter is downcounting.

The setup of the OC2 should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC2 value is to use the Force Output Compare (FOC2) strobe bit in Normal mode. The OC2 Register keeps its value even when changing between Waveform Generation modes.

Be aware that the COM21:0 bits are not double buffered together with the compare value. Changing the COM21:0 bits will take effect immediately.

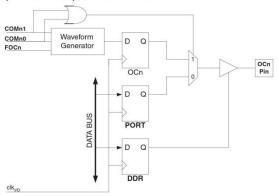




Compare Match Output Unit

The Compare Output mode (COM21:0) bits have two functions. The Waveform Generator uses the COM21:0 bits for defining the Output Compare (OC2) state at the next compare match. Also, the COM21:0 bits control the OC2 pin output source. Figure 56 shows a simplified schematic of the logic affected by the COM21:0 bit setting. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O Port Control Registers (DDR and PORT) that are affected by the COM21:0 bits are shown. When referring to the OC2 state, the reference is for the internal OC2 Register, not the OC2 pin.

Figure 56. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the Output Compare (OC2) from the waveform generator if either of the COM21:0 bits are set. However, the OC2 pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. The Data Direction Register bit for the OC2 pin (DDR_OC2) must be set as output before the OC2 value is visible on the pin. The port override function is independent of the Waveform Generation mode.

The design of the output compare pin logic allows initialization of the OC2 state before the output is enabled. Note that some COM21:0 bit settings are reserved for certain modes of operation. See "8-bit Timer/Counter Register Description" on page 128.

Compare Output Mode and Waveform Generation The waveform generator uses the COM21:0 bits differently in Normal, CTC, and PWM modes. For all modes, setting the COM21:0 = 0 tells the Waveform Generator that no action on the OC2 Register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 51 on page 129. For fast PWM mode, refer to Table 52 on page 129, and for phase correct PWM refer to Table 53 on page 129.

A change of the COM21:0 bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC2 strobe bits.





Modes of Operation

The mode of operation, that is, the behavior of the Timer/Counter and the output compare pins, is defined by the combination of the Waveform Generation mode (WGM21:0) and Compare Output mode (COM21:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM21:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM21:0 bits control whether the output should be set, cleared, or toggled at a compare match (See "Compare Match Output Unit" on page 121.).

For detailed timing information refer to "Timer/Counter Timing Diagrams" on page 126.

Normal Mode

The simplest mode of operation is the Normal mode (WGM21:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter Overflow Flag (TOV2) will be set in the same timer clock cycle as the TCNT2 becomes zero. The TOV2 Flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV2 Flag, the timer resolution can be increased by software. There are no special cases to consider in the normal mode, a new counter value can be written anytime.

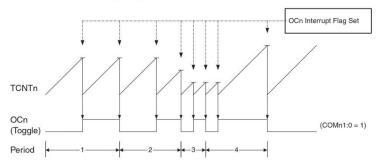
The Output Compare unit can be used to generate interrupts at some given time. Using the output compare to generate waveforms in normal mode is not recommended, since this will occupy too much of the CPU time.

Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM21:0 = 2), the OCR2 Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT2) matches the OCR2. The OCR2 defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 57. The counter value (TCNT2) increases until a compare match occurs between TCNT2 and OCR2, and then counter (TCNT2) is cleared.

Figure 57. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCF2 Flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing the TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR2 is lower than the current



122



value of TCNT2, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFF) and wrap around starting at 0x00 before the compare match can occur.

For generating a waveform output in CTC mode, the OC2 output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COM21:0 = 1). The OC2 value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{\text{OC2}} = f_{\text{clk_VO}}/2$ when OCR2 is set to zero (0x00). The waveform frequency is defined by the following equation:

$$f_{OCn} = \frac{f_{\text{clk_I/O}}}{2 \cdot N \cdot (1 + OCRn)}$$

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024).

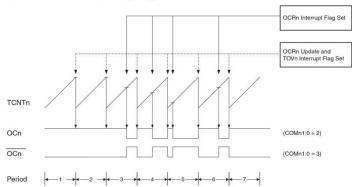
As for the Normal mode of operation, the TOV2 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

Fast PWM Mode

The fast Pulse Width Modulation or fast PWM mode (WGM21:0 = 3) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to MAX then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC2) is cleared on the compare match between TCNT2 and OCR2, and set at BOTTOM. In inverting Compare Output mode, the output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct PWM mode that uses dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

In fast PWM mode, the counter is incremented until the counter value matches the MAX value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 58. The TCNT2 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT2 slopes represent compare matches between OCR2 and TCNT2.

Figure 58. Fast PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV2) is set each time the counter reaches MAX. If the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.

AIMEL

123



In fast PWM mode, the compare unit allows generation of PWM waveforms on the OC2 pin. Setting the COM21:0 bits to 2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM21:0 to 3 (see Table 52 on page 129). The actual OC2 value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC2 Register at the compare match between OCR2 and TCNT2, and clearing (or setting) the OC2 Register at the timer clock cycle the counter is cleared (changes from MAX to BOTTOM).

The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnPWM} = \frac{f_{\text{clk_I/O}}}{N \cdot 256}$$

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024).

The extreme values for the OCR2 Register represent special cases when generating a PWM waveform output in the fast PWM mode. If the OCR2 is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCR2 equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM21:0 bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC2 to toggle its logical level on each compare match (COM21:0 = 1). The waveform generated will have a maximum frequency of $f_{oc2} = f_{clk_I/O}/2$ when OCR2 is set to zero. This feature is similar to the OC2 toggle in CTC mode, except the double buffer feature of the output compare unit is enabled in the fast PWM mode.

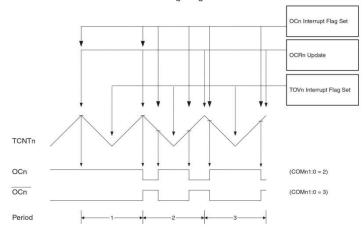
Phase Correct PWM Mode

The phase correct PWM mode (WGM21:0 = 1) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to MAX and then from MAX to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC2) is cleared on the compare match between TCNT2 and OCR2 while upcounting, and set on the compare match while downcounting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The PWM resolution for the phase correct PWM mode is fixed to 8 bits. In phase correct PWM mode the counter is incremented until the counter value matches MAX. When the counter reaches MAX, it changes the count direction. The TCNT2 value will be equal to MAX for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 59. The TCNT2 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT2 slopes represent compare matches between OCR2 and TCNT2.



Figure 59. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV2) is set each time the counter reaches BOTTOM. The Interrupt Flag can be used to generate an interrupt each time the counter reaches the BOTTOM value.

In phase correct PWM mode, the compare unit allows generation of PWM waveforms on the OC2 pin. Setting the COM21:0 bits to 2 will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM21:0 to 3 (see Table 53 on page 129). The actual OC2 value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC2 Register at the compare match between OCR2 and TCNT2 when the counter increments, and setting (or clearing) the OC2 Register at compare match between OCR2 and TCNT2 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnPCPWM} = \frac{f_{\text{clk_I/O}}}{N \cdot 510}$$

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024).

The extreme values for the OCR2 Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR2 is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

At the very start of Period 2 in Figure 59 OCn has a transition from high to I ow even though there is no Compare Match. The point of this transition is to guarantee symmetry around BOTTOM. There are two cases that will give transition without Compare Match:

- OCR2A changes its value from Max, like in Figure 59. When the OCR2A value is MAX the
 OCn pin value is the same as the result of a down-counting Compare Match. To ensure
 symmetry around BOTTOM the OCn value at MAX must be correspond the the result of an
 up-counting Compare Match.
- The Timer starts counting from a value higher than the one in OCR2A, and for that reason misses the Compare Match and hence the OCn that would have happened on the way up.





Timer/Counter Timing Diagrams

The following figures show the Timer/Counter in Synchronous mode, and the timer clock (clk_{T2}) is therefore shown as a clock enable signal. In Asynchronous mode, $clk_{I/O}$ should be replaced by the Timer/Counter Oscillator clock. The figures include information on when Interrupt Flags are set. Figure 60 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 60. Timer/Counter Timing Diagram, no Prescaling

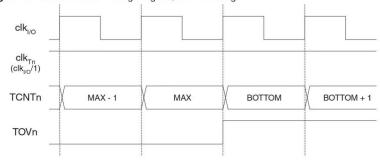


Figure 61 shows the same timing data, but with the prescaler enabled.

Figure 61. Timer/Counter Timing Diagram, with Prescaler (f_{clk_I/O}/8)

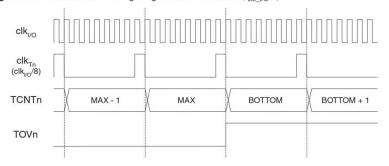


Figure 62 shows the setting of OCF2 in all modes except CTC mode.

AIMEL

126



 $\textbf{Figure 62}. \ \, \textbf{Timer/Counter Timing Diagram, Setting of OCF2, with Prescaler} \, \, (f_{\text{clk_I/O}}/8)$

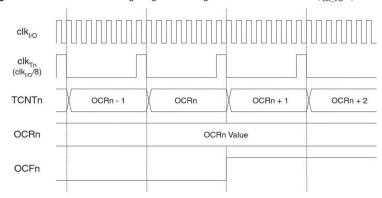
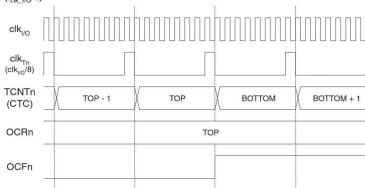


Figure 63 shows the setting of OCF2 and the clearing of TCNT2 in CTC mode.

Figure 63. Timer/Counter Timing Diagram, Clear Timer on Compare Match Mode, with Prescaler ($f_{clk_I/O}/8$)



AIMEL

127



8-bit Timer/Counter Register Description

Timer/Counter Control Register – TCCR2

Bit	7	6	5	4	3	2	1	0	
	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	TCCR2
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 - FOC2: Force Output Compare

The FOC2 bit is only active when the WGM bits specify a non-PWM mode. However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR2 is written when operating in PWM mode. When writing a logical one to the FOC2 bit, an immediate compare match is forced on the waveform generation unit. The OC2 output is changed according to its COM21:0 bits setting. Note that the FOC2 bit is implemented as a strobe. Therefore it is the value present in the COM21:0 bits that determines the effect of the forced compare.

A FOC2 strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR2 as TOP.

The FOC2 bit is always read as zero.

• Bit 3, 6 - WGM21:0: Waveform Generation Mode

These bits control the counting sequence of the counter, the source for the maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode, Clear Timer on Compare match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes. See Table 50 and "Modes of Operation" on page 122.

Table 50. Waveform Generation Mode Bit Description⁽¹⁾

Mode	WGM21 (CTC2)	WGM20 (PWM2)	Timer/Counter Mode of Operation	ТОР	Update of OCR2	TOV2 Flag Set on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	воттом
2	1	0	СТС	OCR2	Immediate	MAX
3	1	1	Fast PWM	0xFF	воттом	MAX

Note: 1. The CTC2 and PWM2 bit definition names are now obsolete. Use the WGM21:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

• Bit 5:4 - COM21:0: Compare Match Output Mode

These bits control the Output Compare pin (OC2) behavior. If one or both of the COM21:0 bits are set, the OC2 output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to OC2 pin must be set in order to enable the output driver.



128



When OC2 is connected to the pin, the function of the COM21:0 bits depends on the WGM21:0 bit setting. Table 51 shows the COM21:0 bit functionality when the WGM21:0 bits are set to a normal or CTC mode (non-PWM).

Table 51. Compare Output Mode, non-PWM Mode

COM21	COM20	Description
0	0	Normal port operation, OC2 disconnected.
0	1	Toggle OC2 on compare match
1	0	Clear OC2 on compare match
1	1	Set OC2 on compare match

Table 52 shows the COM21:0 bit functionality when the WGM21:0 bits are set to fast PWM mode.

Table 52. Compare Output Mode, Fast PWM Mode⁽¹⁾

COM21	COM20	Description
0	0	Normal port operation, OC2 disconnected.
0	1	Reserved
1	0	Clear OC2 on compare match, set OC2 at BOTTOM, (non-inverting mode)
1	1	Set OC2 on compare match, clear OC2 at BOTTOM, (inverting mode)

Note: 1. A special case occurs when OCR2 equals TOP and COM21 is set. In this case, the compare match is ignored, but the set or clear is done at BOTTOM. See "Fast PWM Mode" on page 123 for more details.

Table 53 shows the COM21:0 bit functionality when the WGM21:0 bits are set to phase correct PWM mode

Table 53. Compare Output Mode, Phase Correct PWM Mode⁽¹⁾

COM21	COM20	Description
0	0	Normal port operation, OC2 disconnected.
0	1	Reserved
1	0	Clear OC2 on compare match when up-counting. Set OC2 on compare match when downcounting.
1	1	Set OC2 on compare match when up-counting. Clear OC2 on compare match when downcounting.

 A special case occurs when OCR2 equals TOP and COM21 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See "Phase Correct PWM Mode" on page 124 for more details.





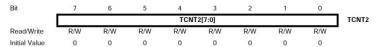
· Bit 2:0 - CS22:0: Clock Select

The three Clock Select bits select the clock source to be used by the Timer/Counter, see Table 54.

Table 54. Clock Select Bit Description

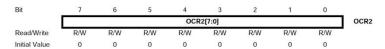
CS22	CS21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk _{T2S} /(No prescaling)
0	1	0	clk _{T2S} /8 (From prescaler)
0	1	1	clk _{T2S} /32 (From prescaler)
1	0	0	clk _{T2S} /64 (From prescaler)
1	0	1	clk _{T2S} /128 (From prescaler)
1	1	0	clk _{T2S} /256 (From prescaler)
1	1	1	clk _{T2S} /1024 (From prescaler)

Timer/Counter Register – TCNT2



The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT2 Register blocks (removes) the compare match on the following timer clock. Modifying the counter (TCNT2) while the counter is running, introduces a risk of missing a compare match between TCNT2 and the OCR2 Register.

Output Compare Register – OCR2

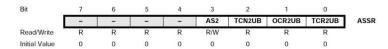


The Output Compare Register contains an 8-bit value that is continuously compared with the counter value (TCNT2). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC2 pin.



Asynchronous Operation of the Timer/Counter

Asynchronous Status Register – ASSR



· Bit 3 - AS2: Asynchronous Timer/Counter2

When AS2 is written to zero, Timer/Counter 2 is clocked from the I/O clock, $clk_{I/O}$. When AS2 is written to one, Timer/Counter2 is clocked from a Crystal Oscillator connected to the Timer Oscillator 1 (TOSC1) pin. When the value of AS2 is changed, the contents of TCNT2, OCR2, and TCCR2 might be corrupted.

• Bit 2 - TCN2UB: Timer/Counter2 Update Busy

When Timer/Counter2 operates asynchronously and TCNT2 is written, this bit becomes set. When TCNT2 has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCNT2 is ready to be updated with a new value.

Bit 1 – OCR2UB: Output Compare Register2 Update Busy

When Timer/Counter2 operates asynchronously and OCR2 is written, this bit becomes set. When OCR2 has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that OCR2 is ready to be updated with a new value.

· Bit 0 - TCR2UB: Timer/Counter Control Register2 Update Busy

When Timer/Counter2 operates asynchronously and TCCR2 is written, this bit becomes set. When TCCR2 has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCCR2 is ready to be updated with a new value.

If a write is performed to any of the three Timer/Counter2 Registers while its update busy flag is set, the updated value might get corrupted and cause an unintentional interrupt to occur.

The mechanisms for reading TCNT2, OCR2, and TCCR2 are different. When reading TCNT2, the actual timer value is read. When reading OCR2 or TCCR2, the value in the temporary storage register is read.

Asynchronous Operation of Timer/Counter2

When Timer/Counter2 operates asynchronously, some considerations must be taken.

- Warning: When switching between asynchronous and synchronous clocking of Timer/Counter2, the Timer Registers TCNT2, OCR2, and TCCR2 might be corrupted. A safe procedure for switching clock source is:
 - 1. Disable the Timer/Counter2 interrupts by clearing OCIE2 and TOIE2.
 - 2. Select clock source by setting AS2 as appropriate.
 - 3. Write new values to TCNT2, OCR2, and TCCR2.
 - 4. To switch to asynchronous operation: Wait for TCN2UB, OCR2UB, and TCR2UB.
 - 5. Clear the Timer/Counter2 Interrupt Flags.
 - 6. Enable interrupts, if needed.



131



- The Oscillator is optimized for use with a 32.768 kHz watch crystal. Applying an external clock to the TOSC1 pin may result in incorrect Timer/Counter2 operation. The CPU main clock frequency must be more than four times the Oscillator frequency.
- When writing to one of the registers TCNT2, OCR2, or TCCR2, the value is transferred to a temporary register, and latched after two positive edges on TOSC1. The user should not write a new value before the contents of the temporary register have been transferred to its destination. Each of the three mentioned registers have their individual temporary register, which means for example that writing to TCNT2 does not disturb an OCR2 write in progress. To detect that a transfer to the destination register has taken place, the Asynchronous Status Register ASSR has been implemented.
- When entering Power-save or Extended Standby mode after having written to TCNT2, OCR2, or TCCR2, the user must wait until the written register has been updated if Timer/Counter2 is used to wake up the device. Otherwise, the MCU will enter sleep mode before the changes are effective. This is particularly important if the Output Compare2 interrupt is used to wake up the device, since the output compare function is disabled during writing to OCR2 or TCNT2. If the write cycle is not finished, and the MCU enters sleep mode before the OCR2UB bit returns to zero, the device will never receive a compare match interrupt, and the MCU will not wake up.
- If Timer/Counter2 is used to wake the device up from Power-save or Extended Standby
 mode, precautions must be taken if the user wants to re-enter one of these modes: The
 interrupt logic needs one TOSC1 cycle to be reset. If the time between wake-up and reentering sleep mode is less than one TOSC1 cycle, the interrupt will not occur, and the
 device will fail to wake up. If the user is in doubt whether the time before re-entering Powersave or Extended Standby mode is sufficient, the following algorithm can be used to ensure
 that one TOSC1 cycle has elapsed:
 - 1. Write a value to TCCR2, TCNT2, or OCR2.
 - 2. Wait until the corresponding Update Busy Flag in ASSR returns to zero.
 - 3. Enter Power-save or Extended Standby mode.
- When the asynchronous operation is selected, the 32.768 kHz Oscillator for Timer/Counter2 is always running, except in Power-down and Standby modes. After a Power-up Reset or wake-up from Power-down or Standby mode, the user should be aware of the fact that this Oscillator might take as long as one second to stabilize. The user is advised to wait for at least one second before using Timer/Counter2 after power-up or wake-up from Power-down or Standby mode. The contents of all Timer/Counter2 Registers must be considered lost after a wake-up from Power-down or Standby mode due to unstable clock signal upon start-up, no matter whether the Oscillator is in use or a clock signal is applied to the TOSC1 pin.
- Description of wake up from Power-save or Extended Standby mode when the timer is clocked asynchronously: When the interrupt condition is met, the wake up process is started on the following cycle of the timer clock, that is, the timer is always advanced by at least one before the processor can read the counter value. After wake-up, the MCU is halted for four cycles, it executes the interrupt routine, and resumes execution from the instruction following SLEEP.
- Reading of the TCNT2 Register shortly after wake-up from Power-save may give an incorrect result. Since TCNT2 is clocked on the asynchronous TOSC clock, reading TCNT2 must be done through a register synchronized to the internal I/O clock domain. Synchronization takes place for every rising TOSC1 edge. When waking up from Powersave mode, and the I/O clock (clk_{I/O}) again becomes active, TCNT2 will read as the previous value (before entering sleep) until the next rising TOSC1 edge. The phase of the TOSC clock after waking up from Power-save mode is essentially unpredictable, as it depends on the wake-up time. The recommended procedure for reading TCNT2 is thus as follows:



132



- 1. Write any value to either of the registers OCR2 or TCCR2.
- 2. Wait for the corresponding Update Busy Flag to be cleared.
- 3. Read TCNT2.
- During asynchronous operation, the synchronization of the Interrupt Flags for the
 asynchronous timer takes three processor cycles plus one timer cycle. The timer is therefore
 advanced by at least one before the processor can read the timer value causing the setting
 of the Interrupt Flag. The output compare pin is changed on the timer clock and is not
 synchronized to the processor clock.

Timer/Counter Interrupt Mask Register – TIMSK

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 - OCIE2: Timer/Counter2 Output Compare Match Interrupt Enable

When the OCIE2 bit is written to one and the I-bit in the Status Register is set (one), the Timer/Counter2 Compare Match interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter2 occurs, that is, when the OCF2 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

• Bit 6 - TOIE2: Timer/Counter2 Overflow Interrupt Enable

When the TOIE2 bit is written to one and the I-bit in the Status Register is set (one), the Timer/Counter2 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter2 occurs, that is, when the TOV2 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

Timer/Counter Interrupt Flag Register – TIFR



• Bit 7 - OCF2: Output Compare Flag 2

The OCF2 bit is set (one) when a compare match occurs between the Timer/Counter2 and the data in OCR2 – Output Compare Register2. OCF2 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF2 is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE2 (Timer/Counter2 Compare match Interrupt Enable), and OCF2 are set (one), the Timer/Counter2 Compare match Interrupt is executed.

• Bit 6 - TOV2: Timer/Counter2 Overflow Flag

The TOV2 bit is set (one) when an overflow occurs in Timer/Counter2. TOV2 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV2 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE2 (Timer/Counter2 Overflow Interrupt Enable), and TOV2 are set (one), the Timer/Counter2 Overflow interrupt is executed. In PWM mode, this bit is set when Timer/Counter2 changes counting direction at \$00.

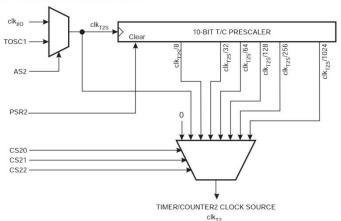


133



Timer/Counter Prescaler

Figure 64. Prescaler for Timer/Counter2



The clock source for Timer/Counter2 is named clk_{T2S} . clk_{T2S} is by default connected to the main system I/O clock clk_{IO} . By setting the AS2 bit in ASSR, Timer/Counter2 is asynchronously clocked from the TOSC1 pin. This enables use of Timer/Counter2 as a Real Time Counter (RTC). When AS2 is set, pins TOSC1 and TOSC2 are disconnected from Port C. A crystal can then be connected between the TOSC1 and TOSC2 pins to serve as an independent clock source for Timer/Counter2. The Oscillator is optimized for use with a 32.768 kHz crystal. Applying an external clock source to TOSC1 is not recommended.

For Timer/Counter2, the possible prescaled selections are: $clk_{T2S}/8$, $clk_{T2S}/32$, $clk_{T2S}/64$, $clk_{T2S}/128$, $clk_{T2S}/256$, and $clk_{T2S}/1024$. Additionally, clk_{T2S} as well as 0 (stop) may be selected. Setting the PSR2 bit in SFIOR resets the prescaler. This allows the user to operate with a predictable prescaler.

Special Function IO Register – SFIOR



• Bit 1 - PSR2: Prescaler Reset Timer/Counter2

When this bit is written to one, the Timer/Counter2 prescaler will be reset. The bit will be cleared by hardware after the operation is performed. Writing a zero to this bit will have no effect. This bit will always be read as zero if Timer/Counter2 is clocked by the internal CPU clock. If this bit is written when Timer/Counter2 is operating in asynchronous mode, the bit will remain one until the prescaler has been reset.

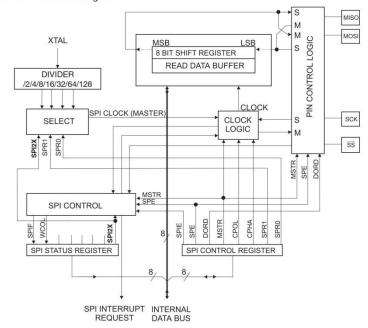


Serial Peripheral Interface – SPI

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATmega16 and peripheral devices or between several AVR devices. The ATmega16 SPI includes the following features:

- · Full-duplex, Three-wire Synchronous Data Transfer
- Master or Slave Operation
- · LSB First or MSB First Data Transfer
- Seven Programmable Bit Rates
- · End of Transmission Interrupt Flag
- · Write Collision Flag Protection
- Wake-up from Idle Mode
- Double Speed (CK/2) Master SPI Mode

Figure 65. SPI Block Diagram⁽¹⁾



Note: 1. Refer to Figure 1 on page 2, and Table 25 on page 58 for SPI pin placement.

The interconnection between Master and Slave CPUs with SPI is shown in Figure 66. The system consists of two Shift Registers, and a Master clock generator. The SPI Master initiates the communication cycle when pulling low the Slave Select \$\overline{S}\$ pin of the desired Slave. Master and Slave prepare the data to be sent in their respective Shift Registers, and the Master generates the required clock pulses on the SCK line to interchange data. Data is always shifted from Master to Slave on the Master Out – Slave In, MOSI, line, and from Slave to Master on the Master In – Slave Out, MISO, line. After each data packet, the Master will synchronize the Slave by pulling high the Slave Select, \$\overline{S}\$, line.

When configured as a Master, the SPI interface has no automatic control of the \overline{SS} line. This must be handled by user software before communication can start. When this is done, writing a



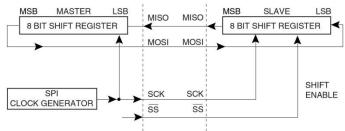
135



byte to the SPI Data Register starts the SPI clock generator, and the hardware shifts the eight bits into the Slave. After shifting one byte, the SPI clock generator stops, setting the end of Transmission Flag (SPIF). If the SPI Interrupt Enable bit (SPIE) in the SPCR Register is set, an interrupt is requested. The Master may continue to shift the next byte by writing it into SPDR, or signal the end of packet by pulling high the Slave Select, \overline{SS} line. The last incoming byte will be kept in the Buffer Register for later use.

When configured as a Slave, the SPI interface will remain sleeping with MISO tri-stated as long as the \overline{SS} pin is driven high. In this state, software may update the contents of the SPI Data Register, SPDR, but the data will not be shifted out by incoming clock pulses on the SCK pin until the \overline{SS} pin is driven low. As one byte has been completely shifted, the end of Transmission Flag, SPIF is set. If the SPI Interrupt Enable bit, SPIE, in the SPCR Register is set, an interrupt is requested. The Slave may continue to place new data to be sent into SPDR before reading the incoming data. The last incoming byte will be kept in the Buffer Register for later use.

Figure 66. SPI Master-Slave Interconnection



The system is single buffered in the transmit direction and double buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SPI Data Register before the entire shift cycle is completed. When receiving data, however, a received character must be read from the SPI Data Register before the next character has been completely shifted in. Otherwise, the first byte is lost.

In SPI Slave mode, the control logic will sample the incoming signal of the SCK pin. To ensure correct sampling of the clock signal, the minimum low and high periods should be:

Low periods: Longer than 2 CPU clock cycles.

High periods: Longer than 2 CPU clock cycles.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and \overline{SS} pins is overridden according to Table 55 on page 136. For more details on automatic port overrides, refer to "Alternate Port Functions" on page 55.

Table 55. SPI Pin Overrides

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
SS	User Defined	Input



136



Note: See "Alternate Functions of Port B" on page 58 for a detailed description of how to define the direction of the user defined SPI pins.

The following code examples show how to initialize the SPI as a Master and how to perform a simple transmission. DDR_SPI in the examples must be replaced by the actual Data Direction Register controlling the SPI pins. DD_MOSI, DD_MISO and DD_SCK must be replaced by the actual data direction bits for these pins. For example if MOSI is placed on pin PB5, replace DD_MOSI with DDB5 and DDR_SPI with DDRB.



137



```
Assembly Code Example(1)
   SPI_MasterInit:
     ; Set MOSI and SCK output, all others input
     ldi r17, (1<<DD_MOSI) | (1<<DD_SCK)
     out DDR_SPI,r17
     ; Enable SPI, Master, set clock rate fck/16
     ldi r17,(1<<SPE) | (1<<MSTR) | (1<<SPR0)
     out SPCR, r17
     ret
   SPI_MasterTransmit:
     ; Start transmission of data (r16)
     out SPDR, r16
   Wait_Transmit:
     ; Wait for transmission complete
     sbis SPSR, SPIF
     rjmp Wait_Transmit
     ret
C Code Example<sup>(1)</sup>
   void SPI_MasterInit(void)
     /\star Set MOSI and SCK output, all others input \star/
     DDR_SPI = (1<<DD_MOSI) | (1<<DD_SCK);
     /* Enable SPI, Master, set clock rate fck/16 */
     SPCR = (1<<SPE) | (1<<MSTR) | (1<<SPR0);
   void SPI_MasterTransmit(char cData)
     /* Start transmission */
     SPDR = cData;
     /* Wait for transmission complete */
     while(!(SPSR & (1<<SPIF)))
```

Note: 1. See "About Code Examples" on page 7.



138



The following code examples show how to initialize the SPI as a Slave and how to perform a simple reception.

```
Assembly Code Example<sup>(1)</sup>
   SPI_SlaveInit:
     ; Set MISO output, all others input
     ldi r17, (1<<DD_MISO)
     out DDR_SPI,r17
     ; Enable SPI
     ldi r17, (1<<SPE)
     out SPCR, r17
     ret
   SPI_SlaveReceive:
     ; Wait for reception complete
     sbis SPSR, SPIF
     rjmp SPI_SlaveReceive
     ; Read received data and return
     in r16,SPDR
     ret
C Code Example<sup>(1)</sup>
   void SPI_SlaveInit(void)
     /* Set MISO output, all others input */
    DDR_SPI = (1<<DD_MISO);
     /* Enable SPI */
     SPCR = (1 << SPE);
   char SPI_SlaveReceive(void)
     /* Wait for reception complete */
     while(!(SPSR & (1<<SPIF)))</pre>
     /* Return data register */
     return SPDR;
```

Note: 1. See "About Code Examples" on page 7.



139



SS Pin Functionality

Slave Mode

When the SPI is configured as a Slave, the Slave Select $\overline{(SS)}$ pin is always input. When \overline{SS} is held low, the SPI is activated, and MISO becomes an output if configured so by the user. All other pins are inputs. When \overline{SS} is driven high, all pins are inputs except MISO which can be user configured as an output, and the SPI is passive, which means that it will not receive incoming data. Note that the SPI logic will be reset once the \overline{SS} pin is driven high.

The \overline{SS} pin is useful for packet/byte synchronization to keep the Slave Bit Counter synchronous with the Master Clock generator. When the \overline{SS} pin is driven high, the SPI Slave will immediately reset the send and receive logic, and drop any partially received data in the Shift Register.

Master Mode

When the SPI is configured as a Master (MSTR in SPCR is set), the user can determine the direction of the \overline{SS} pin.

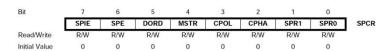
If \overline{SS} is configured as an output, the pin is a general output pin which does not affect the SPI system. Typically, the pin will be driving the \overline{SS} pin of the SPI Slave.

If \overline{SS} is configured as an input, it must be held high to ensure Master SPI operation. If the \overline{SS} pin is driven low by peripheral circuitry when the SPI is configured as a Master with the \overline{SS} pin defined as an input, the SPI system interprets this as another Master selecting the SPI as a Slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:

- The MSTR bit in SPCR is cleared and the SPI system becomes a Slave. As a result of the SPI becoming a Slave, the MOSI and SCK pins become inputs.
- The SPIF Flag in SPSR is set, and if the SPI interrupt is enabled, and the I-bit in SREG is set, the interrupt routine will be executed.

Thus, when interrupt-driven SPI transmission is used in Master mode, and there exists a possibility that \overline{SS} is driven low, the interrupt should always check that the MSTR bit is still set. If the MSTR bit has been cleared by a Slave Select, it must be set by the user to re-enable SPI Master mode.

SPI Control Register – SPCR



Bit 7 – SPIE: SPI Interrupt Enable

This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and the if the global interrupt enable bit in SREG is set.

· Bit 6 - SPE: SPI Enable

When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

· Bit 5 - DORD: Data Order

When the DORD bit is written to one, the LSB of the data word is transmitted first.

When the DORD bit is written to zero, the MSB of the data word is transmitted first.



140



· Bit 4 - MSTR: Master/Slave Select

This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero. If \overline{SS} is configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI Master mode.

· Bit 3 - CPOL: Clock Polarity

When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to Figure 67 and Figure 68 for an example. The CPOL functionality is summarized below:

Table 56. CPOL Functionality

CPOL	Leading Edge	Trailing Edge		
0	Rising	Falling		
1	Falling	Rising		

· Bit 2 - CPHA: Clock Phase

The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to Figure 67 and Figure 68 for an example. The CPHA functionality is summarized below:

Table 57. CPHA Functionality

СРНА	Leading Edge	Trailing Edge			
0	Sample	Setup			
1	Setup	Sample			

• Bits 1, 0 - SPR1, SPR0: SPI Clock Rate Select 1 and 0

These two bits control the SCK rate of the device configured as a Master. SPR1 and SPR0 have no effect on the Slave. The relationship between SCK and the Oscillator Clock frequency $f_{\rm osc}$ is shown in the following table:

Table 58. Relationship Between SCK and the Oscillator Frequency

SPI2X	SPR1	SPR0	SCK Frequency	
0	0	0	f _{osc} /4	
0	0	1	f _{osc} /16	
0	1	0	f _{osc} /64	
0	1	1	f _{osc} /128	
1	0	0	f _{osc} /2	
1	0	1	f _{osc} /8	
1	1	0	f _{osc} /32 f _{osc} /64	
1	1	1	f _{osc} /64	



141



SPI Status Register – SPSR

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	-) -	-	-	-	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	,
Initial Value	0	0	0	0	0	0	0	0	

· Bit 7 - SPIF: SPI Interrupt Flag

When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If \overline{SS} is an input and is driven low when the SPI is in Master mode, this will also set the SPIF Flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPDR).

• Bit 6 - WCOL: Write COLlision Flag

The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set, and then accessing the SPI Data Register.

· Bit 5..1 - Res: Reserved Bits

These bits are reserved bits in the ATmega16 and will always read as zero.

· Bit 0 - SPI2X: Double SPI Speed Bit

When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (see Table 58). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at $f_{\rm osc}/4$ or lower.

The SPI interface on the ATmega16 is also used for program memory and EEPROM downloading or uploading. See page 273 for SPI Serial Programming and Verification.

SPI Data Register – SPDR



The SPI Data Register is a read/write register used for data transfer between the Register File and the SPI Shift Register. Writing to the register initiates data transmission. Reading the register causes the Shift Register Receive buffer to be read.



Data Modes

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 67 and Figure 68. Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize. This is clearly seen by summarizing Table 56 and Table 57, as done below:

Table 59. CPOL and CPHA Functionality

	Leading Edge	Trailing Edge	SPI Mode
CPOL = 0, CPHA = 0	Sample (Rising)	Setup (Falling)	0
CPOL = 0, CPHA = 1	Setup (Rising)	Sample (Falling)	1
CPOL = 1, CPHA = 0	Sample (Falling) Setup (Rising)		2
CPOL = 1, CPHA = 1	Setup (Falling)	Sample (Rising)	3

Figure 67. SPI Transfer Format with CPHA = 0

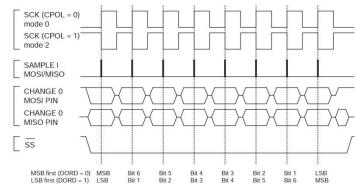
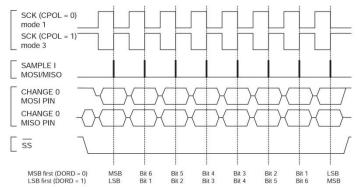


Figure 68. SPI Transfer Format with CPHA = 1





143



USART

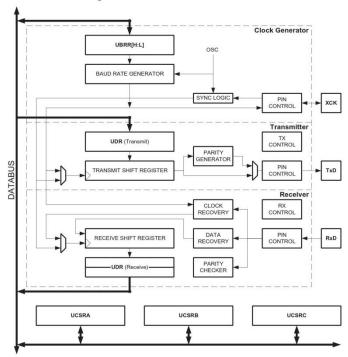
The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a highly flexible serial communication device. The main features are:

- Full Duplex Operation (Independent Serial Receive and Transmit Registers)
- Asynchronous or Synchronous Operation
 Master or Slave Clocked Synchronous Operation
 High Resolution Baud Rate Generator
- Supports Serial Frames with 5, 6, 7, 8, or 9 Data Bits and 1 or 2 Stop Bits Odd or Even Parity Generation and Parity Check Supported by Hardware
- **Data OverRun Detection**
- Framing Error Detection
- Noise Filtering Includes False Start Bit Detection and Digital Low Pass Filter
- Three Separate Interrupts on TX Complete, TX Data Register Empty, and RX Complete
 Multi-processor Communication Mode
- Double Speed Asynchronous Communication Mode

Overview

A simplified block diagram of the USART transmitter is shown in Figure 69. CPU accessible I/O Registers and I/O pins are shown in bold.

Figure 69. USART Block Diagram⁽¹⁾



1. Refer to Figure 1 on page 2, Table 33 on page 65, and Table 27 on page 60 for USART pin

144



The dashed boxes in the block diagram separate the three main parts of the USART (listed from the top): Clock Generator, Transmitter and Receiver. Control Registers are shared by all units. The clock generation logic consists of synchronization logic for external clock input used by synchronous Slave operation, and the baud rate generator. The XCK (Transfer Clock) pin is only used by Synchronous Transfer mode. The Transmitter consists of a single write buffer, a serial Shift Register, parity generator and control logic for handling different serial frame formats. The write buffer allows a continuous transfer of data without any delay between frames. The Receiver is the most complex part of the USART module due to its clock and data recovery units. The recovery units are used for asynchronous data reception. In addition to the recovery units, the receiver includes a parity checker, control logic, a Shift Register and a two level receive buffer (UDR). The receiver supports the same frame formats as the transmitter, and can detect frame error, data overrun and parity errors.

AVR USART vs. AVR UART – Compatibility

The USART is fully compatible with the AVR UART regarding:

- · Bit locations inside all USART Registers
- · Baud Rate Generation
- Transmitter Operation
- · Transmit Buffer Functionality
- Receiver Operation

However, the receive buffering has two improvements that will affect the compatibility in some special cases:

- A second Buffer Register has been added. The two Buffer Registers operate as a circular FIFO buffer. Therefore the UDR must only be read once for each incoming data! More important is the fact that the Error Flags (FE and DOR) and the 9th data bit (RXB8) are buffered with the data in the receive buffer. Therefore the status bits must always be read before the UDR Register is read. Otherwise the error status will be lost since the buffer state is lost.
- The receiver Shift Register can now act as a third buffer level. This is done by allowing the
 received data to remain in the serial Shift Register (see Figure 69) if the Buffer Registers are
 full, until a new start bit is detected. The USART is therefore more resistant to Data OverRun
 (DOR) error conditions.

The following control bits have changed name, but have same functionality and register location:

- CHR9 is changed to UCSZ2
- · OR is changed to DOR

Clock Generation

The clock generation logic generates the base clock for the Transmitter and Receiver. The USART supports four modes of clock operation: Normal Asynchronous, Double Speed Asynchronous, Master Synchronous and Slave Synchronous mode. The UMSEL bit in USART Control and Status Register C (UCSRC) selects between asynchronous and synchronous operation. Double Speed (Asynchronous mode only) is controlled by the U2X found in the UCSRA Register. When using Synchronous mode (UMSEL = 1), the Data Direction Register for the XCK pin (DDR_XCK) controls whether the clock source is internal (Master mode) or external (Slave mode). The XCK pin is only active when using Synchronous mode.

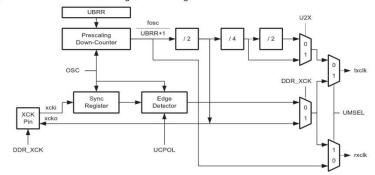
Figure 70 shows a block diagram of the clock generation logic.



145



Figure 70. Clock Generation Logic, Block Diagram



Signal description:

txclk Transmitter clock (Internal Signal).

rxclk Receiver base clock (Internal Signal).

xcki Input from XCK pin (Internal Signal). Used for synchronous Slave operation.

xcko Clock output to XCK pin (Internal Signal). Used for synchronous Master operation.

fosc XTAL pin frequency (System Clock).

Internal Clock Generation – The Baud Rate Generator Internal clock generation is used for the asynchronous and the synchronous Master modes of operation. The description in this section refers to Figure 70.

The USART Baud Rate Register (UBRR) and the down-counter connected to it function as a programmable prescaler or baud rate generator. The down-counter, running at system clock (fosc), is loaded with the UBRR value each time the counter has counted down to zero or when the UBRRL Register is written. A clock is generated each time the counter reaches zero. This clock is the baud rate generator clock output (= fosc/(UBRR+1)). The Transmitter divides the baud rate generator clock output by 2, 8 or 16 depending on mode. The baud rate generator output is used directly by the receiver's clock and data recovery units. However, the recovery units use a state machine that uses 2, 8 or 16 states depending on mode set by the state of the UMSEL, U2X and DDR_XCK bits.

Table 60 contains equations for calculating the baud rate (in bits per second) and for calculating the UBRR value for each mode of operation using an internally generated clock source.



Table 60. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate ⁽¹⁾	Equation for Calculating UBRR Value	
Asynchronous Normal Mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR + 1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$	
Asynchronous Double Speed Mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(UBRR + 1)}$	$UBRR = \frac{f_{OSC}}{8BAUD} - 1$	
Synchronous Master Mode	$BAUD = \frac{f_{OSC}}{2(UBRR + 1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$	

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps).

BAUD Baud rate (in bits per second, bps)

f_{OSC} System Oscillator clock frequency

UBRR Contents of the UBRRH and UBRRL Registers, (0 - 4095)

Some examples of UBRR values for some system clock frequencies are found in Table 68 (see page 168).

Double Speed Operation (U2X)

The transfer rate can be doubled by setting the U2X bit in UCSRA. Setting this bit only has effect for the asynchronous operation. Set this bit to zero when using synchronous operation.

Setting this bit will reduce the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication. Note however that the receiver will in this case only use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery, and therefore a more accurate baud rate setting and system clock are required when this mode is used. For the Transmitter, there are no downsides.

External Clock

External clocking is used by the synchronous Slave modes of operation. The description in this section refers to Figure 70 for details.

External clock input from the XCK pin is sampled by a synchronization register to minimize the chance of meta-stability. The output from the synchronization register must then pass through an edge detector before it can be used by the Transmitter and receiver. This process introduces a two CPU clock period delay and therefore the maximum external XCK clock frequency is limited by the following equation:

$$f_{XCK} < \frac{f_{OSC}}{4}$$

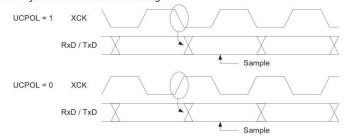
Note that $f_{\rm osc}$ depends on the stability of the system clock source. It is therefore recommended to add some margin to avoid possible loss of data due to frequency variations.



Synchronous Clock Operation

When Synchronous mode is used (UMSEL = 1), the XCK pin will be used as either clock input (Slave) or clock output (Master). The dependency between the clock edges and data sampling or data change is the same. The basic principle is that data input (on RxD) is sampled at the opposite XCK clock edge of the edge the data output (TxD) is changed.

Figure 71. Synchronous Mode XCK Timing



The UCPOL bit UCRSC selects which XCK clock edge is used for data sampling and which is used for data change. As Figure 71 shows, when UCPOL is zero the data will be changed at rising XCK edge and sampled at falling XCK edge. If UCPOL is set, the data will be changed at falling XCK edge and sampled at rising XCK edge.

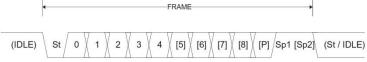
Frame Formats

A serial frame is defined to be one character of data bits with synchronization bits (start and stop bits), and optionally a parity bit for error checking. The USART accepts all 30 combinations of the following as valid frame formats:

- 1 start bit
- 5, 6, 7, 8, or 9 data bits
- · no, even or odd parity bit
- 1 or 2 stop bits

A frame starts with the start bit followed by the least significant data bit. Then the next data bits, up to a total of nine, are succeeding, ending with the most significant bit. If enabled, the parity bit is inserted after the data bits, before the stop bits. When a complete frame is transmitted, it can be directly followed by a new frame, or the communication line can be set to an idle (high) state. Figure 72 illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

Figure 72. Frame Formats



- St Start bit, always low.
- (n) Data bits (0 to 8).
- P Parity bit. Can be odd or even.
- **Sp** Stop bit, always high.



148



IDLE No transfers on the communication line (RxD or TxD). An IDLE line must be high.

The frame format used by the USART is set by the UCSZ2:0, UPM1:0, and USBS bits in UCSRB and UCSRC. The Receiver and Transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

The USART Character SiZe (UCSZ2:0) bits select the number of data bits in the frame. The USART Parity mode (UPM1:0) bits enable and set the type of parity bit. The selection between one or two stop bits is done by the USART Stop Bit Select (USBS) bit. The receiver ignores the second stop bit. An FE (Frame Error) will therefore only be detected in the cases where the first stop bit is zero.

Parity Bit Calculation

The parity bit is calculated by doing an exclusive-or of all the data bits. If odd parity is used, the result of the exclusive or is inverted. The relation between the parity bit and data bits is as follows::

$$\begin{array}{l} P_{even} = \, d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0 \\ P_{odd} = \, d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1 \end{array}$$

Peven Parity bit using even parity

Podd Parity bit using odd parity

d_n Data bit n of the character

If used, the parity bit is located between the last data bit and first stop bit of a serial frame.

USART Initialization

The USART has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting frame format and enabling the Transmitter or the Receiver depending on the usage. For interrupt driven USART operation, the Global Interrupt Flag should be cleared (and interrupts globally disabled) when doing the initialization.

Before doing a re-initialization with changed baud rate or frame format, be sure that there are no ongoing transmissions during the period the registers are changed. The TXC Flag can be used to check that the Transmitter has completed all transfers, and the RXC Flag can be used to check that there are no unread data in the receive buffer. Note that the TXC Flag must be cleared before each transmission (before UDR is written) if it is used for this purpose.

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume asynchronous operation using polling (no interrupts enabled) and a fixed frame format. The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 registers. When the function writes to the UCSRC Register, the URSEL bit (MSB) must be set due to the sharing of I/O location by UBRRH and UCSRC.



```
Assembly Code Example(1)
   USART_Init:
     ; Set baud rate
     out UBRRH, r17
     out UBRRL, r16
     ; Enable receiver and transmitter
     ldi r16, (1<<RXEN) | (1<<TXEN)
     out UCSRB, r16
     ; Set frame format: 8data, 2stop bit
     ldi r16, (1<<URSEL) | (1<<USBS) | (3<<UCSZ0)
     out UCSRC, r16
     ret
C Code Example<sup>(1)</sup>
   #define FOSC 1843200// Clock Speed
   #define BAUD 9600
   #define MYUBRR FOSC/16/BAUD-1
   void main( void )
     USART_Init ( MYUBRR );
   void USART_Init( unsigned int ubrr)
     /* Set baud rate */
     UBRRH = (unsigned char) (ubrr>>8);
     UBRRL = (unsigned char)ubrr;
     /* Enable receiver and transmitter */
     UCSRB = (1<<RXEN) | (1<<TXEN);
     /* Set frame format: 8data, 2stop bit */
     UCSRC = (1<<URSEL) | (1<<USBS) | (3<<UCSZ0);
```

Note: 1. See "About Code Examples" on page 7.

More advanced initialization routines can be made that include frame format as parameters, disable interrupts and so on. However, many applications use a fixed setting of the Baud and Control Registers, and for these types of applications the initialization code can be placed directly in the main routine, or be combined with initialization code for other I/O modules.

Data Transmission - The USART Transmitter

The USART Transmitter is enabled by setting the *Transmit Enable* (TXEN) bit in the UCSRB Register. When the Transmitter is enabled, the normal port operation of the TxD pin is overridden by the USART and given the function as the transmitter's serial output. The baud rate, mode of operation and frame format must be set up once before doing any transmissions. If synchronous operation is used, the clock on the XCK pin will be overridden and used as transmission clock.

Sending Frames with 5 to 8 Data Bit

A data transmission is initiated by loading the transmit buffer with the data to be transmitted. The CPU can load the transmit buffer by writing to the UDR I/O location. The buffered data in the transmit buffer will be moved to the Shift Register when the Shift Register is ready to send a new



150



frame. The Shift Register is loaded with new data if it is in idle state (no ongoing transmission) or immediately after the last stop bit of the previous frame is transmitted. When the Shift Register is loaded with new data, it will transfer one complete frame at the rate given by the Baud Register, U2X bit or by XCK depending on mode of operation.

The following code examples show a simple USART transmit function based on polling of the *Data Register Empty* (UDRE) Flag. When using frames with less than eight bits, the most significant bits written to the UDR are ignored. The USART has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in Register R16

```
Assembly Code Example(1)

USART_Transmit:
; Wait for empty transmit buffer
sbis UCSRA, UDRE
rjmp USART_Transmit
; Put data (r16) into buffer, sends the data
out UDR,r16
ret

C Code Example(1)

void USART_Transmit( unsigned char data )
{
    /* Wait for empty transmit buffer */
    while (!(UCSRA & (1<<UDRE)))
    ;
    /* Put data into buffer, sends the data */
    UDR = data;
}
```

Note: 1. See "About Code Examples" on page 7.

The function simply waits for the transmit buffer to be empty by checking the UDRE Flag, before loading it with new data to be transmitted. If the Data Register Empty Interrupt is utilized, the interrupt routine writes the data into the buffer.



151



Sending Frames with 9 Data Bit

If 9-bit characters are used (UCSZ = 7), the ninth bit must be written to the TXB8 bit in UCSRB before the Low byte of the character is written to UDR. The following code examples show a transmit function that handles 9-bit characters. For the assembly code, the data to be sent is assumed to be stored in Registers R17:R16.

```
Assembly Code Example(1)
   USART_Transmit:
     ; Wait for empty transmit buffer
     sbis UCSRA, UDRE
     rimp USART Transmit
     ; Copy 9th bit from r17 to TXB8
     cbi UCSRB. TXB8
     sbrc r17,0
     sbi UCSRB, TXB8
     ; Put LSB data (r16) into buffer, sends the data
     out UDR, r16
     ret
C Code Example<sup>(1)</sup>
   void USART_Transmit( unsigned int data )
     /* Wait for empty transmit buffer */
     while ( !( UCSRA & (1<<UDRE))) )
          ;
     /* Copy 9th bit to TXB8 */
     UCSRB &= ~(1<<TXB8);
     if ( data & 0x0100 )
      UCSRB |= (1<<TXB8);
     /* Put data into buffer, sends the data */
     UDR = data;
```

Note: 1. These transmit functions are written to be general functions. They can be optimized if the contents of the UCSRB is static. (that is, only the TXB8 bit of the UCSRB Register is used after initial register.)

The ninth bit can be used for indicating an address frame when using multi processor communication mode or for other protocol handling as for example synchronization.

Transmitter Flags and Interrupts

The USART transmitter has two flags that indicate its state: USART Data Register Empty (UDRE) and Transmit Complete (TXC). Both flags can be used for generating interrupts.

The Data Register Empty (UDRE) Flag indicates whether the transmit buffer is ready to receive new data. This bit is set when the transmit buffer is empty, and cleared when the transmit buffer contains data to be transmitted that has not yet been moved into the Shift Register. For compatibility with future devices, always write this bit to zero when writing the UCSRA Register.

When the Data Register empty Interrupt Enable (UDRIE) bit in UCSRB is written to one, the USART Data Register Empty Interrupt will be executed as long as UDRE is set (provided that global interrupts are enabled). UDRE is cleared by writing UDR. When interrupt-driven data transmission is used, the Data Register Empty Interrupt routine must either write new data to UDR in order to clear UDRE or disable the Data Register empty Interrupt, otherwise a new interrupt will occur once the interrupt routine terminates.



152



The Transmit Complete (TXC) Flag bit is set one when the entire frame in the transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer. The TXC Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC Flag is useful in half-duplex communication interfaces (like the RS485 standard), where a transmitting application must enter receive mode and free the communication bus immediately after completing the transmission.

When the Transmit Compete Interrupt Enable (TXCIE) bit in UCSRB is set, the USART Transmit Complete Interrupt will be executed when the TXC Flag becomes set (provided that global interrupts are enabled). When the transmit complete interrupt is used, the interrupt handling routine does not have to clear the TXC Flag, this is done automatically when the interrupt is executed.

Parity Generator

The parity generator calculates the parity bit for the serial frame data. When parity bit is enabled (UPM1 = 1), the transmitter control logic inserts the parity bit between the last data bit and the first stop bit of the frame that is sent.

Disabling the Transmitter The disabling of the transmitter (setting the TXEN to zero) will not become effective until ongoing and pending transmissions are completed, that is, when the transmit Shift Register and transmit Buffer Register do not contain data to be transmitted. When disabled, the transmitter will no longer override the TxD pin.



153



Data Reception – The USART Receiver The USART Receiver is enabled by writing the Receive Enable (RXEN) bit in the UCSRB Register to one. When the receiver is enabled, the normal pin operation of the RxD pin is overridden by the USART and given the function as the receiver's serial input. The baud rate, mode of operation and frame format must be set up once before any serial reception can be done. If synchronous operation is used, the clock on the XCK pin will be used as transfer clock.

Receiving Frames with 5 to 8 Data Bits

The receiver starts data reception when it detects a valid start bit. Each bit that follows the start bit will be sampled at the baud rate or XCK clock, and shifted into the receive Shift Register until the first stop bit of a frame is received. A second stop bit will be ignored by the receiver. When the first stop bit is received, that is, a complete serial frame is present in the receive Shift Register, the contents of the Shift Register will be moved into the receive buffer. The receive buffer can then be read by reading the UDR I/O location.

The following code example shows a simple USART receive function based on polling of the Receive Complete (RXC) Flag. When using frames with less than eight bits the most significant bits of the data read from the UDR will be masked to zero. The USART has to be initialized before the function can be used.

```
Assembly Code Example(1)

USART_Receive:
; Wait for data to be received
sbis UCSRA, RXC
rjmp USART_Receive
; Get and return received data from buffer
in r16, UDR
ret

C Code Example(1)

unsigned char USART_Receive( void )
{
    /* Wait for data to be received */
    while (!(UCSRA & (1<<RXC)))
    ;
    /* Get and return received data from buffer */
    return UDR;
}
```

Note: 1. See "About Code Examples" on page 7.

The function simply waits for data to be present in the receive buffer by checking the RXC Flag, before reading the buffer and returning the value.





Receiving Frames with 9 Databits

If 9 bit characters are used (UCSZ=7) the ninth bit must be read from the RXB8 bit in UCSRB **before** reading the low bits from the UDR. This rule applies to the FE, DOR and PE status Flags as well. Read status from UCSRA, then data from UDR. Reading the UDR I/O location will change the state of the receive buffer FIFO and consequently the TXB8, FE, DOR and PE bits, which all are stored in the FIFO, will change.

The following code example shows a simple USART receive function that handles both 9-bit characters and the status bits.

```
Assembly Code Example<sup>(1)</sup>
   USART Receive:
     ; Wait for data to be received
     sbis UCSRA, RXC
     rjmp USART_Receive
     ; Get status and 9th bit, then data from buffer
     in r18, UCSRA
     in r17, UCSRB
     in r16, UDR
     ; If error, return -1
     andi r18,(1<<FE) | (1<<DOR) | (1<<PE)
     breq USART_ReceiveNoError
     ldi r17, HIGH(-1)
     ldi r16, LOW(-1)
   USART_ReceiveNoError:
     ; Filter the 9th bit, then return
     lsr r17
     andi r17, 0x01
     ret
C Code Example<sup>(1)</sup>
    unsigned int USART_Receive( void )
     unsigned char status, resh, resl;
     /* Wait for data to be received */
     while ( !(UCSRA & (1<<RXC)) )
     /* Get status and 9th bit, then data */
     /* from buffer */
     status = UCSRA;
     resh = UCSRB;
     resl = UDR;
     /* If error, return -1 */
     if ( status & (1<<FE) | (1<<DOR) | (1<<PE) )</pre>
       return -1:
      /* Filter the 9th bit, then return */
     resh = (resh >> 1) & 0x01;
     return ((resh << 8) | resl);
```

Note: 1. See "About Code Examples" on page 7.



155



The receive function example reads all the I/O Registers into the Register File before any computation is done. This gives an optimal receive buffer utilization since the buffer location read will be free to accept new data as early as possible.

Receive Compete Flag and Interrupt

The USART Receiver has one flag that indicates the receiver state.

The Receive Complete (RXC) Flag indicates if there are unread data present in the receive buffer. This flag is one when unread data exist in the receive buffer, and zero when the receive buffer is empty (that is, does not contain any unread data). If the receiver is disabled (RXEN = 0), the receive buffer will be flushed and consequently the RXC bit will become zero.

When the Receive Complete Interrupt Enable (RXCIE) in UCSRB is set, the USART Receive Complete Interrupt will be executed as long as the RXC Flag is set (provided that global interrupts are enabled). When interrupt-driven data reception is used, the receive complete routine must read the received data from UDR in order to clear the RXC Flag, otherwise a new interrupt will occur once the interrupt routine terminates.

Receiver Error Flags

The USART Receiver has three Error Flags: Frame Error (FE), Data OverRun (DOR) and Parity Error (PE). All can be accessed by reading UCSRA. Common for the Error Flags is that they are located in the receive buffer together with the frame for which they indicate the error status. Due to the buffering of the Error Flags, the UCSRA must be read before the receive buffer (UDR), since reading the UDR I/O location changes the buffer read location. Another equality for the Error Flags is that they can not be altered by software doing a write to the flag location. However, all flags must be set to zero when the UCSRA is written for upward compatibility of future USART implementations. None of the Error Flags can generate interrupts.

The Frame Error (FE) Flag indicates the state of the first stop bit of the next readable frame stored in the receive buffer. The FE Flag is zero when the stop bit was correctly read (as one), and the FE Flag will be one when the stop bit was incorrect (zero). This flag can be used for detecting out-of-sync conditions, detecting break conditions and protocol handling. The FE Flag is not affected by the setting of the USBS bit in UCSRC since the receiver ignores all, except for the first, stop bits. For compatibility with future devices, always set this bit to zero when writing to LICSRA

The Data OverRun (DOR) Flag indicates data loss due to a receiver buffer full condition. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the receive Shift Register, and a new start bit is detected. If the DOR Flag is set there was one or more serial frame lost between the frame last read from UDR, and the next frame read from UDR. For compatibility with future devices, always write this bit to zero when writing to UCSRA. The DOR Flag is cleared when the frame received was successfully moved from the Shift Register to the receive buffer.

The Parity Error (PE) Flag indicates that the next frame in the receive buffer had a parity error when received. If parity check is not enabled the PE bit will always be read zero. For compatibility with future devices, always set this bit to zero when writing to UCSRA. For more details see "Parity Bit Calculation" on page 149 and "Parity Checker" on page 156.

Parity Checker

The Parity Checker is active when the high USART Parity mode (UPM1) bit is set. Type of parity check to be performed (odd or even) is selected by the UPM0 bit. When enabled, the parity checker calculates the parity of the data bits in incoming frames and compares the result with the parity bit from the serial frame. The result of the check is stored in the receive buffer together with the received data and stop bits. The Parity Error (PE) Flag can then be read by software to check if the frame had a parity error.

The PE bit is set if the next character that can be read from the receive buffer had a parity error when received and the parity checking was enabled at that point (UPM1 = 1). This bit is valid until the receive buffer (UDR) is read.



156



Disabling the Receiver

In contrast to the Transmitter, disabling of the Receiver will be immediate. Data from ongoing receptions will therefore be lost. When disabled (that is, the RXEN is set to zero) the Receiver will no longer override the normal function of the RxD port pin. The receiver buffer FIFO will be flushed when the receiver is disabled. Remaining data in the buffer will be lost

Flushing the Receive Buffer

The receiver buffer FIFO will be flushed when the Receiver is disabled, that is, the buffer will be emptied of its contents. Unread data will be lost. If the buffer has to be flushed during normal operation, due to for instance an error condition, read the UDR I/O location until the RXC Flag is cleared. The following code example shows how to flush the receive buffer.

```
Assembly Code Example(1)

USART_Flush:
    sbis UCSRA, RXC
    ret
    in    r16, UDR
    rjmp USART_Flush

C Code Example(1)

void USART_Flush( void )
{
    unsigned char dummy;
    while ( UCSRA & (1<<RXC) ) dummy = UDR;
}
```

Note: 1. See "About Code Examples" on page 7.

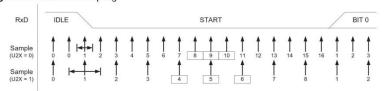
Asynchronous Data Reception

The USART includes a clock recovery and a data recovery unit for handling asynchronous data reception. The clock recovery logic is used for synchronizing the internally generated baud rate clock to the incoming asynchronous serial frames at the RxD pin. The data recovery logic samples and low pass filters each incoming bit, thereby improving the noise immunity of the receiver. The asynchronous reception operational range depends on the accuracy of the internal baud rate clock, the rate of the incoming frames, and the frame size in number of bits.

Asynchronous Clock Recovery

The clock recovery logic synchronizes internal clock to the incoming serial frames. Figure 73 illustrates the sampling process of the start bit of an incoming frame. The sample rate is 16 times the baud rate for Normal mode, and 8 times the baud rate for Double Speed mode. The horizontal arrows illustrate the synchronization variation due to the sampling process. Note the larger time variation when using the double speed mode (U2X = 1) of operation. Samples denoted zero are samples done when the RxD line is idle (that is, no communication activity).

Figure 73. Start Bit Sampling



When the clock recovery logic detects a high (idle) to low (start) transition on the RxD line, the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample as shown in the figure. The clock recovery logic then uses samples 8, 9, and 10 for Normal mode, and samples 4, 5, and 6 for Double Speed mode (indicated with sample numbers inside boxes on the

AIMEL

157

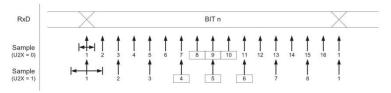


figure), to decide if a valid start bit is received. If two or more of these three samples have logical high levels (the majority wins), the start bit is rejected as a noise spike and the receiver starts looking for the next high to low-transition. If however, a valid start bit is detected, the clock recovery logic is synchronized and the data recovery can begin. The synchronization process is repeated for each start bit.

Asynchronous Data Recovery

When the receiver clock is synchronized to the start bit, the data recovery can begin. The data recovery unit uses a state machine that has 16 states for each bit in normal mode and 8 states for each bit in Double Speed mode. Figure 74 shows the sampling of the data bits and the parity bit. Each of the samples is given a number that is equal to the state of the recovery unit.

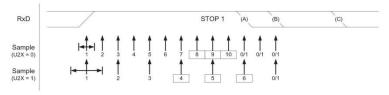
Figure 74. Sampling of Data and Parity Bit



The decision of the logic level of the received bit is taken by doing a majority voting of the logic value to the three samples in the center of the received bit. The center samples are emphasized on the figure by having the sample number inside boxes. The majority voting process is done as follows: If two or all three samples have high levels, the received bit is registered to be a logic 1. If two or all three samples have low levels, the received bit is registered to be a logic 0. This majority voting process acts as a low pass filter for the incoming signal on the RxD pin. The recovery process is then repeated until a complete frame is received. Including the first stop bit. Note that the receiver only uses the first stop bit of a frame.

Figure 75 shows the sampling of the stop bit and the earliest possible beginning of the start bit of the next frame.

Figure 75. Stop Bit Sampling and Next Start Bit Sampling



The same majority voting is done to the stop bit as done for the other bits in the frame. If the stop bit is registered to have a logic 0 value, the Frame Error (FE) Flag will be set.

A new high to low transition indicating the start bit of a new frame can come right after the last of the bits used for majority voting. For Normal Speed mode, the first low level sample can be at point marked (A) in Figure 75. For Double Speed mode the first low level must be delayed to (B). (C) marks a stop bit of full length. The early start bit detection influences the operational range of the receiver.

<u>AIMEL</u>

158



Asynchronous Operational Range

The operational range of the receiver is dependent on the mismatch between the received bit rate and the internally generated baud rate. If the Transmitter is sending frames at too fast or too slow bit rates, or the internally generated baud rate of the receiver does not have a similar (see Table 61) base frequency, the receiver will not be able to synchronize the frames to the start bit.

The following equations can be used to calculate the ratio of the incoming data rate and internal receiver baud rate.

$$R_{slow} = \frac{(D+1)S}{S-1+D\cdot S+S_F}$$

$$R_{fast} = \frac{(D+2)S}{(D+1)S + S_M}$$

- D Sum of character size and parity size (D = 5 to 10 bit)
- S Samples per bit. S = 16 for Normal Speed mode and S = 8 for Double Speed mode.
- S_F First sample number used for majority voting. S_F = 8 for Normal Speed and S_F = 4 for Double Speed mode.
- $S_{\rm M}$ Middle sample number used for majority voting. $S_{\rm M}$ = 9 for Normal Speed and $S_{\rm M}$ = 5 for Double Speed mode.
- R_{slow} is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate. R_{fast} is the ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate.

Table 61 and Table 62 list the maximum receiver baud rate error that can be tolerated. Note that Normal Speed mode has higher toleration of baud rate variations.

Table 61. Recommended Maximum Receiver Baud Rate Error for Normal Speed Mode (U2X = 0)

D # (Data+Parity Bit)	R _{slow} (%)	R _{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	93.20	106.67	+6.67/-6.8	±3.0
6	94.12	105.79	+5.79/-5.88	±2.5
7	94.81	105.11	+5.11/-5.19	±2.0
8	95.36	104.58	+4.58/-4.54	±2.0
9	95.81	104.14	+4.14/-4.19	±1.5
10	96.17	103.78	+3.78/-3.83	±1.5

Table 62. Recommended Maximum Receiver Baud Rate Error for Double Speed Mode (U2X = 1)

D # (Data+Parity Bit)	R _{slow} (%)	R _{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	94.12	105.66	+5.66/-5.88	±2.5
6	94.92	104.92	+4.92/-5.08	±2.0
7	95.52	104.35	+4.35/-4.48	±1.5



159



Table 62. Recommended Maximum Receiver Baud Rate Error for Double Speed Mode (U2X = 1)

D # (Data+Parity Bit)	R _{slow} (%)	R _{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
8	96.00	103.90	+3.90/-4.00	±1.5
9	96.39	103.53	+3.53/-3.61	±1.5
10	96.70	103.23	+3.23/-3.30	±1.0

The recommendations of the maximum receiver baud rate error was made under the assumption that the receiver and transmitter equally divides the maximum total error.

There are two possible sources for the receivers baud rate error. The receiver's system clock (XTAL) will always have some minor instability over the supply voltage range and the temperature range. When using a crystal to generate the system clock, this is rarely a problem, but for a resonator the system clock may differ more than 2% depending of the resonators tolerance. The second source for the error is more controllable. The baud rate generator can not always do an exact division of the system frequency to get the baud rate wanted. In this case an UBRR value that gives an acceptable low error can be used if possible.



160



Multi-processor Communication Mode

Setting the Multi-processor Communication mode (MPCM) bit in UCSRA enables a filtering function of incoming frames received by the USART Receiver. Frames that do not contain address information will be ignored and not put into the receive buffer. This effectively reduces the number of incoming frames that has to be handled by the CPU, in a system with multiple MCUs that communicate via the same serial bus. The Transmitter is unaffected by the MPCM setting, but has to be used differently when it is a part of a system utilizing the Multi-processor Communication mode.

If the receiver is set up to receive frames that contain 5 to 8 data bits, then the first stop bit indicates if the frame contains data or address information. If the receiver is set up for frames with nine data bits, then the ninth bit (RXB8) is used for identifying address and data frames. When the frame type bit (the first stop or the ninth bit) is one, the frame contains an address. When the frame type bit is zero the frame is a data frame.

The Multi-processor Communication mode enables several Slave MCUs to receive data from a Master MCU. This is done by first decoding an address frame to find out which MCU has been addressed. If a particular Slave MCU has been addressed, it will receive the following data frames as normal, while the other Slave MCUs will ignore the received frames until another address frame is received.

Using MPCM

For an MCU to act as a Master MCU, it can use a 9-bit character frame format (UCSZ = 7). The ninth bit (TXB8) must be set when an address frame (TXB8 = 1) or cleared when a data frame (TXB = 0) is being transmitted. The Slave MCUs must in this case be set to use a 9-bit character frame format.

The following procedure should be used to exchange data in Multi-processor Communication mode:

- 1. All Slave MCUs are in Multi-processor Communication mode (MPCM in UCSRA is set).
- 2. The Master MCU sends an address frame, and all Slaves receive and read this frame. In the Slave MCUs, the RXC Flag in UCSRA will be set as normal.
- Each Slave MCU reads the UDR Register and determines if it has been selected. If so, it clears the MPCM bit in UCSRA, otherwise it waits for the next address byte and keeps the MPCM setting.
- The addressed MCU will receive all data frames until a new address frame is received.
 The other Slave MCUs, which still have the MPCM bit set, will ignore the data frames.
- When the last data frame is received by the addressed MCU, the addressed MCU sets
 the MPCM bit and waits for a new address frame from Master. The process then repeats
 from 2

Using any of the 5-bit to 8-bit character frame formats is possible, but impractical since the receiver must change between using n and n+1 character frame formats. This makes full-duplex operation difficult since the transmitter and receiver uses the same character size setting. If 5-bit to 8-bit character frames are used, the transmitter must be set to use two stop bit (USBS = 1) since the first stop bit is used for indicating the frame type.

Do not use Read-Modify-Write instructions (SBI and CBI) to set or clear the MPCM bit. The MPCM bit shares the same I/O location as the TXC Flag and this might accidentally be cleared when using SBI or CBI instructions.





Accessing UBRRH/ UCSRC Registers

The UBRRH Register shares the same I/O location as the UCSRC Register. Therefore some special consideration must be taken when accessing this I/O location.

Write Access

When doing a write access of this I/O location, the high bit of the value written, the USART Register Select (URSEL) bit, controls which one of the two registers that will be written. If URSEL is zero during a write operation, the UBRRH value will be updated. If URSEL is one, the UCSRC setting will be updated.

The following code examples show how to access the two registers.

```
Assembly Code Example(1)

...
; Set UBRRH to 2
ldi r16,0x02
out UBRRH,r16
...
; Set the USBS and the UCSZ1 bit to one, and
; the remaining bits to zero.
ldi r16,(1<<URSEL)|(1<<UCSZ1)
out UCSRC,r16
...

C Code Example(1)

...

/* Set UBRRH to 2 */
UBRRH = 0x02;
...

/* Set the USBS and the UCSZ1 bit to one, and */
/* the remaining bits to zero. */
UCSRC = (1<<URSEL)|(1<<USBS)|(1<<UCSZ1);
...
```

Note: 1. See "About Code Examples" on page 7.

As the code examples illustrate, write accesses of the two registers are relatively unaffected of the sharing of I/O location.

Read Access

Doing a read access to the UBRRH or the UCSRC Register is a more complex operation. However, in most applications, it is rarely necessary to read any of these registers.

The read access is controlled by a timed sequence. Reading the I/O location once returns the UBRRH Register contents. If the register location was read in previous system clock cycle, reading the register in the current clock cycle will return the UCSRC contents. Note that the timed sequence for reading the UCSRC is an atomic operation. Interrupts must therefore be controlled (for example by disabling interrupts globally) during the read operation.





The following code example shows how to read the UCSRC Register contents.

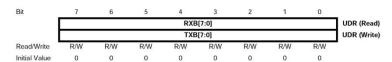
Note: 1. See "About Code Examples" on page 7.

The assembly code example returns the UCSRC value in r16.

Reading the UBRRH contents is not an atomic operation and therefore it can be read as an ordinary register, as long as the previous instruction did not access the register location.

USART Register Description

USART I/O Data Register – UDR



The USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDR. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDR Register location. Reading the UDR Register location will return the contents of the Receive Data Buffer Register (RXB).

For 5-bit, 6-bit, or 7-bit characters the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver.

The transmit buffer can only be written when the UDRE Flag in the UCSRA Register is set. Data written to UDR when the UDRE Flag is not set, will be ignored by the USART Transmitter. When data is written to the transmit buffer, and the Transmitter is enabled, the Transmitter will load the data into the transmit Shift Register when the Shift Register is empty. Then the data will be serially transmitted on the TxD pin.

The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use read modify write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.



163



USART Control and Status Register A – UCSRA

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	•
Initial Value	0	0	1	0	0	0	0	0	

· Bit 7 - RXC: USART Receive Complete

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (that is, does not contain any unread data). If the receiver is disabled, the receive buffer will be flushed and consequently the RXC bit will become zero. The RXC Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE bit).

• Bit 6 - TXC: USART Transmit Complete

This flag bit is set when the entire frame in the transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR). The TXC Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC Flag can generate a Transmit Complete interrupt (see description of the TXCIE bit).

· Bit 5 - UDRE: USART Data Register Empty

The UDRE Flag indicates if the transmit buffer (UDR) is ready to receive new data. If UDRE is one, the buffer is empty, and therefore ready to be written. The UDRE Flag can generate a Data Register empty Interrupt (see description of the UDRIE bit).

UDRE is set after a reset to indicate that the transmitter is ready.

· Bit 4 - FE: Frame Error

This bit is set if the next character in the receive buffer had a Frame Error when received. that is, when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDR) is read. The FE bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSRA.

• Bit 3 - DOR: Data OverRun

This bit is set if a Data OverRun condition is detected. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the receive Shift Register, and a new start bit is detected. This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

• Bit 2 - PE: Parity Error

This bit is set if the next character in the receive buffer had a Parity Error when received and the parity checking was enabled at that point (UPM1 = 1). This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

Bit 1 – U2X: Double the USART Transmission Speed

This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.



164



· Bit 0 - MPCM: Multi-processor Communication Mode

This bit enables the Multi-processor Communication mode. When the MPCM bit is written to one, all the incoming frames received by the USART receiver that do not contain address information will be ignored. The transmitter is unaffected by the MPCM setting. For more detailed information see "Multi-processor Communication Mode" on page 161.

USART Control and Status Register B – UCSRB

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	•
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 - RXCIE: RX Complete Interrupt Enable

Writing this bit to one enables interrupt on the RXC Flag. A USART Receive Complete Interrupt will be generated only if the RXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC bit in UCSRA is set.

• Bit 6 - TXCIE: TX Complete Interrupt Enable

Writing this bit to one enables interrupt on the TXC Flag. A USART Transmit Complete Interrupt will be generated only if the TXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC bit in UCSRA is set.

• Bit 5 – UDRIE: USART Data Register Empty Interrupt Enable

Writing this bit to one enables interrupt on the UDRE Flag. A Data Register Empty Interrupt will be generated only if the UDRIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE bit in UCSRA is set.

· Bit 4 - RXEN: Receiver Enable

Writing this bit to one enables the USART Receiver. The Receiver will override normal port operation for the RxD pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FE, DOR, and PE Flags.

Bit 3 – TXEN: Transmitter Enable

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD pin when enabled. The disabling of the Transmitter (writing TXEN to zero) will not become effective until ongoing and pending transmissions are completed, that is, when the transmit Shift Register and transmit Buffer Register do not contain data to be transmitted. When disabled, the transmitter will no longer override the TxD port.

• Bit 2 - UCSZ2: Character Size

The UCSZ2 bits combined with the UCSZ1:0 bit in UCSRC sets the number of data bits (Character Size) in a frame the receiver and transmitter use.

• Bit 1 - RXB8: Receive Data Bit 8

RXB8 is the ninth data bit of the received character when operating with serial frames with nine data bits. Must be read before reading the low bits from UDR.



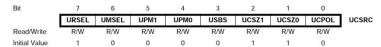
165



· Bit 0 - TXB8: Transmit Data Bit 8

TXB8 is the ninth data bit in the character to be transmitted when operating with serial frames with nine data bits. Must be written before writing the low bits to UDR.

USART Control and Status Register C – UCSRC



The UCSRC Register shares the same I/O location as the UBRRH Register. See the "Accessing UBRRH/ UCSRC Registers" on page 162 section which describes how to access this register.

· Bit 7 - URSEL: Register Select

This bit selects between accessing the UCSRC or the UBRRH Register. It is read as one when reading UCSRC. The URSEL must be one when writing the UCSRC.

· Bit 6 - UMSEL: USART Mode Select

This bit selects between Asynchronous and Synchronous mode of operation.

Table 63. UMSEL Bit Settings

UMSEL	Mode
0	Asynchronous Operation
1	Synchronous Operation

· Bit 5:4 - UPM1:0: Parity Mode

These bits enable and set type of parity generation and check. If enabled, the transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data and compare it to the UPM0 setting. If a mismatch is detected, the PE Flag in UCSRA will be set.

Table 64. UPM Bits Settings

UPM1	UPM0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

· Bit 3 - USBS: Stop Bit Select

This bit selects the number of Stop Bits to be inserted by the Transmitter. The Receiver ignores this setting.

Table 65. USBS Bit Settings

USBS	Stop Bit(s)
0	1-bit
1	2-bit



166



· Bit 2:1 - UCSZ1:0: Character Size

The UCSZ1:0 bits combined with the UCSZ2 bit in UCSRB sets the number of data bits (Character Size) in a frame the Receiver and Transmitter use.

Table 66. UCSZ Bits Settings

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

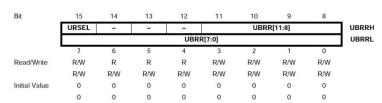
· Bit 0 - UCPOL: Clock Polarity

This bit is used for Synchronous mode only. Write this bit to zero when Asynchronous mode is used. The UCPOL bit sets the relationship between data output change and data input sample, and the synchronous clock (XCK).

Table 67. UCPOL Bit Settings

UCPOL	Transmitted Data Changed (Output of TxD Pin)	Received Data Sampled (Input on RxD Pin)
0	Rising XCK Edge	Falling XCK Edge
1	Falling XCK Edge	Rising XCK Edge

USART Baud Rate Registers – UBRRL and UBRRH



The UBRRH Register shares the same I/O location as the UCSRC Register. See the "Accessing UBRRH/ UCSRC Registers" on page 162 section which describes how to access this register.

· Bit 15 - URSEL: Register Select

This bit selects between accessing the UBRRH or the UCSRC Register. It is read as zero when reading UBRRH. The URSEL must be zero when writing the UBRRH.

· Bit 14:12 - Reserved Bits

These bits are reserved for future use. For compatibility with future devices, these bit must be written to zero when UBRRH is written.



167



· Bit 11:0 - UBRR11:0: USART Baud Rate Register

This is a 12-bit register which contains the USART baud rate. The UBRRH contains the four most significant bits, and the UBRRL contains the 8 least significant bits of the USART baud rate. Ongoing transmissions by the transmitter and receiver will be corrupted if the baud rate is changed. Writing UBRRL will trigger an immediate update of the baud rate prescaler.

Examples of Baud Rate Setting

For standard crystal and resonator frequencies, the most commonly used baud rates for asynchronous operation can be generated by using the UBRR settings in Table 68. UBRR values which yield an actual baud rate differing less than 0.5% from the target baud rate, are bold in the table. Higher error ratings are acceptable, but the receiver will have less noise resistance when the error ratings are high, especially for large serial frames (see "Asynchronous Operational Range" on page 159). The error values are calculated using the following equation:

$$Error[\%] = \left(\frac{BaudRate_{Closest\ Match}}{BaudRate} - 1\right) \bullet 100\%$$

Table 68. Examples of UBRR Settings for Commonly Used Oscillator Frequencies

		f _{osc} = 1.0	000 MHz			f _{osc} = 1.8	432 MHz		$f_{\rm osc}$ = 2.0000 MHz			
Baud Rate	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
(bps)	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
4800	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
9600	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
14.4k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
19.2k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%
28.8k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
38.4k	1	-18.6%	2	8.5%	2	0.0%	5	0.0%	2	8.5%	6	-7.0%
57.6k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
76.8k	-	1-2	1	-18.6%	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%
115.2k	_	_	0	8.5%	0	0.0%	1	0.0%	0	8.5%	1	8.5%
230.4k	_	-	_	_	_	_	0	0.0%	_	-	_	-
250k	_	N=8	_	25-3	-		-	::	_	_	0	0.0%
Max (1)	62.5 Kbps 125 Kbps		115.2	Kbps	230.4	Kbps	125	Kbps	250	Kbps		

^{1.} UBRR = 0, Error = 0.0%



168



Table 69. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

		f _{osc} = 3.6	864 MHz			$f_{\rm osc} = 4.0$	0000 MHz		f _{osc} = 7.3728 MHz				
Baud Rate	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X	(= 1	
(bps)	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	
2400	95	0.0%	191	0.0%	103	0.2%	207	0.2%	191	0.0%	383	0.0%	
4800	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%	
9600	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%	
14.4k	15	0.0%	31	0.0%	16	2.1%	34	-0.8%	31	0.0%	63	0.0%	
19.2k	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%	
28.8k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%	
38.4k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	
57.6k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	
76.8k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	
115.2k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%	
230.4k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%	
250k	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%	
0.5M	_	_	0	-7.8%	_	_	0	0.0%	0	-7.8%	1	-7.8%	
1M	_		_	_	_	-	_	-	-	-	0	-7.8%	
Max (1)	230.4	Kbps	460.8	Kbps	250	Kbps	0.5 N	Иbps	460.8	Kbps	921.6	921.6 Kbps	

^{1.} UBRR = 0, Error = 0.0%





Table 70. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

		f _{osc} = 8.0	0000 MHz			f _{osc} = 11.	0592 MHz			f _{osc} = 14.	7456 MHz	
Baud Rate	U2X	(= O	U2X	U2X = 1		(= 0	U2X	(= 1	U2X	(= 0	U2X	(= 1
(bps)	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	_	-	2	-7.8%	1	-7.8%	3	-7.8%
1M	_		0	0.0%	_	_	_	_	0	-7.8%	1	-7.8%
Max (1)	0.5 N	Лbps	1 N	bps	691.2	Kbps	1.3824	Mbps	921.6	Kbps	1.8432	2 Mbps

^{1.} UBRR = 0, Error = 0.0%





Table 71. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

		f _{osc} = 16.	0000 MHz			f _{osc} = 18.	4320 MHz		$f_{\rm osc}$ = 20.0000 MHz			
Baud Rate	U2X	(= 0	U2X	U2X = 1		(= 0	U2X	(= 1	U2X	(= 0	U2X	(= 1
(bps)	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4k	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6k	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2k	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4k	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%
250k	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	_	-	4	-7.8%	_	-	4	0.0%
1M	0	0.0%	1	0.0%	_	-	_	_	-	0-0	_	_
Max (1)	1 M	bps	2 M	bps	1.152	Mbps	2.304	Mbps	1.25	Mbps	2.5	Mbps

^{1.} UBRR = 0, Error = 0.0%





Two-wire Serial Interface

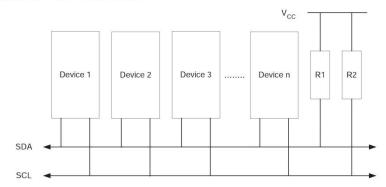
Features

- Simple Yet Powerful and Flexible Communication Interface, Only Two Bus Lines Needed
- Both Master and Slave Operation Supported
- · Device Can Operate as Transmitter or Receiver
- 7-bit Address Space allows up to 128 Different Slave Addresses
- · Multi-master Arbitration Support
- Up to 400 kHz Data Transfer Speed
- Slew-rate Limited Output Drivers
- · Noise Suppression Circuitry Rejects Spikes on Bus Lines
- · Fully Programmable Slave Address with General Call Support
- · Address Recognition causes Wake-up when AVR is in Sleep Mode

Two-wire Serial Interface Bus Definition

The Two-wire Serial Interface (TWI) is ideally suited for typical microcontroller applications. The TWI protocol allows the systems designer to interconnect up to 128 different devices using only two bi-directional bus lines, one for clock (SCL) and one for data (SDA). The only external hardware needed to implement the bus is a single pull-up resistor for each of the TWI bus lines. All devices connected to the bus have individual addresses, and mechanisms for resolving bus contention are inherent in the TWI protocol.

Figure 76. TWI Bus Interconnection



TWI Terminology

The following definitions are frequently encountered in this section.

Table 72. TWI Terminology

Term	Description
Master	The device that initiates and terminates a transmission. The Master also generates the SCL clock.
Slave	The device addressed by a Master.
Transmitter	The device placing data on the bus.
Receiver	The device reading data from the bus.

AIMEL

172



Electrical Interconnection

As depicted in Figure 76, both bus lines are connected to the positive supply voltage through pull-up resistors. The bus drivers of all TWI-compliant devices are open-drain or open-collector. This implements a wired-AND function which is essential to the operation of the interface. A low level on a TWI bus line is generated when one or more TWI devices output a zero. A high level is output when all TWI devices tri-state their outputs, allowing the pull-up resistors to pull the line high. Note that all AVR devices connected to the TWI bus must be powered in order to allow any bus operation.

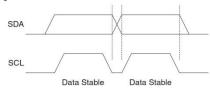
The number of devices that can be connected to the bus is only limited by the bus capacitance limit of 400 pF and the 7-bit Slave address space. A detailed specification of the electrical characteristics of the TWI is given in "Two-wire Serial Interface Characteristics" on page 294. Two different sets of specifications are presented there, one relevant for bus speeds below 100 kHz, and one valid for bus speeds up to 400 kHz.

Data Transfer and Frame Format

Transferring Bits

Each data bit transferred on the TWI bus is accompanied by a pulse on the clock line. The level of the data line must be stable when the clock line is high. The only exception to this rule is for generating start and stop conditions.

Figure 77. Data Validity



Data Change

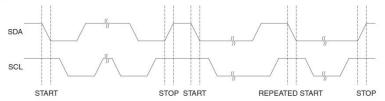
START and STOP Conditions

The Master initiates and terminates a data transmission. The transmission is initiated when the Master issues a START condition on the bus, and it is terminated when the Master issues a STOP condition. Between a START and a STOP condition, the bus is considered busy, and no other Master should try to seize control of the bus. A special case occurs when a new START condition is issued between a START and STOP condition. This is referred to as a REPEATED START condition, and is used when the Master wishes to initiate a new transfer without releasing control of the bus. After a REPEATED START, the bus is considered busy until the next STOP. This is identical to the START behavior, and therefore START is used to describe both START and REPEATED START for the remainder of this datasheet, unless otherwise noted. As depicted below, START and STOP conditions are signalled by changing the level of the SDA line when the SCL line is high.





Figure 78. START, REPEATED START, and STOP Conditions



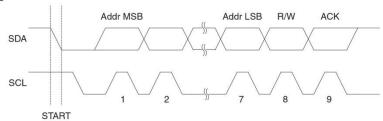
Address Packet Format All address packets transmitted on the TWI bus are nine bits long, consisting of seven address bits, one READ/WRITE control bit and an acknowledge bit. If the READ/WRITE bit is set, a read operation is to be performed, otherwise a write operation should be performed. When a Slave recognizes that it is being addressed, it should acknowledge by pulling SDA low in the ninth SCL (ACK) cycle. If the addressed Slave is busy, or for some other reason can not service the Master's request, the SDA line should be left high in the ACK clock cycle. The Master can then transmit a STOP condition, or a REPEATED START condition to initiate a new transmission. An address packet consisting of a Slave address and a READ or a WRITE bit is called SLA+R or SLA+W, respectively.

The MSB of the address byte is transmitted first. Slave addresses can freely be allocated by the designer, but the address 0000 000 is reserved for a general call.

When a general call is issued, all Slaves should respond by pulling the SDA line low in the ACK cycle. A general call is used when a Master wishes to transmit the same message to several Slaves in the system. When the general call address followed by a Write bit is transmitted on the bus, all Slaves set up to acknowledge the general call will pull the SDA line low in the ack cycle. The following data packets will then be received by all the Slaves that acknowledged the general call. Note that transmitting the general call address followed by a Read bit is meaningless, as this would cause contention if several Slaves started transmitting different data.

All addresses of the format 1111 xxx should be reserved for future purposes.

Figure 79. Address Packet Format



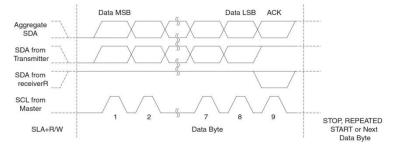




Data Packet Format

All data packets transmitted on the TWI bus are nine bits long, consisting of one data byte and an acknowledge bit. During a data transfer, the Master generates the clock and the START and STOP conditions, while the receiver is responsible for acknowledging the reception. An Acknowledge (ACK) is signalled by the receiver pulling the SDA line low during the ninth SCL cycle. If the receiver leaves the SDA line high, a NACK is signalled. When the receiver has received the last byte, or for some reason cannot receive any more bytes, it should inform the transmitter by sending a NACK after the final byte. The MSB of the data byte is transmitted first.

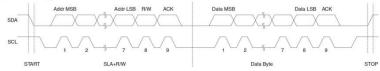
Figure 80. Data Packet Format



Combining Address and Data Packets into a Transmission A transmission basically consists of a START condition, a SLA+R/W, one or more data packets and a STOP condition. An empty message, consisting of a START followed by a STOP condition, is illegal. Note that the wired-ANDing of the SCL line can be used to implement handshaking between the Master and the Slave. The Slave can extend the SCL low period by pulling the SCL line low. This is useful if the clock speed set up by the Master is too fast for the Slave, or the Slave needs extra time for processing between the data transmissions. The Slave extending the SCL low period will not affect the SCL high period, which is determined by the Master. As a consequence, the Slave can reduce the TWI data transfer speed by prolonging the SCL duty cycle.

Figure 81 shows a typical data transmission. Note that several data bytes can be transmitted between the SLA+R/W and the STOP condition, depending on the software protocol implemented by the application software.

Figure 81. Typical Data Transmission



AMEL

175



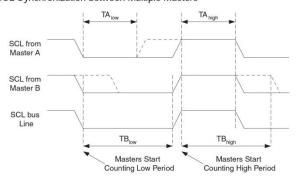
Multi-master Bus Systems, Arbitration and Synchronization

The TWI protocol allows bus systems with several Masters. Special concerns have been taken in order to ensure that transmissions will proceed as normal, even if two or more Masters initiate a transmission at the same time. Two problems arise in multi-master systems:

- An algorithm must be implemented allowing only one of the Masters to complete the transmission. All other Masters should cease transmission when they discover that they have lost the selection process. This selection process is called arbitration. When a contending Master discovers that it has lost the arbitration process, it should immediately switch to Slave mode to check whether it is being addressed by the winning Master. The fact that multiple Masters have started transmission at the same time should not be detectable to the Slaves, that is, the data being transferred on the bus must not be corrupted.
- Different Masters may use different SCL frequencies. A scheme must be devised to synchronize the serial clocks from all Masters, in order to let the transmission proceed in a lockstep fashion. This will facilitate the arbitration process.

The wired-ANDing of the bus lines is used to solve both these problems. The serial clocks from all Masters will be wired-ANDed, yielding a combined clock with a high period equal to the one from the Master with the shortest high period. The low period of the combined clock is equal to the low period of the Master with the longest low period. Note that all Masters listen to the SCL line, effectively starting to count their SCL high and low time-out periods when the combined SCL line goes high or low, respectively.

Figure 82. SCL Synchronization between Multiple Masters



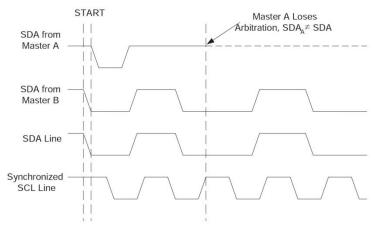
Arbitration is carried out by all Masters continuously monitoring the SDA line after outputting data. If the value read from the SDA line does not match the value the Master had output, it has lost the arbitration. Note that a Master can only lose arbitration when it outputs a high SDA value while another Master outputs a low value. The losing Master should immediately go to Slave mode, checking if it is being addressed by the winning Master. The SDA line should be left high, but losing Masters are allowed to generate a clock signal until the end of the current data or address packet. Arbitration will continue until only one Master remains, and this may take many bits. If several Masters are trying to address the same Slave, arbitration will continue into the data packet.



176



Figure 83. Arbitration between Two Masters



Note that arbitration is not allowed between:

- · A REPEATED START condition and a data bit
- · A STOP condition and a data bit
- · A REPEATED START and a STOP condition

It is the user software's responsibility to ensure that these illegal arbitration conditions never occur. This implies that in multi-master systems, all data transfers must use the same composition of SLA+R/W and data packets. In other words: All transmissions must contain the same number of data packets, otherwise the result of the arbitration is undefined.



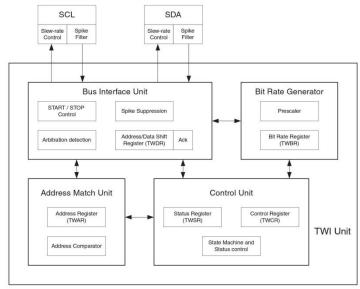


——— ATmega16(L)

Overview of the TWI Module

The TWI module is comprised of several submodules, as shown in Figure 84. All registers drawn in a thick line are accessible through the AVR data bus.

Figure 84. Overview of the TWI Module



SCL and SDA Pins

These pins interface the AVR TWI with the rest of the MCU system. The output drivers contain a slew-rate limiter in order to conform to the TWI specification. The input stages contain a spike suppression unit removing spikes shorter than 50 ns. Note that the internal pull-ups in the AVR pads can be enabled by setting the PORT bits corresponding to the SCL and SDA pins, as explained in the I/O Port section. The internal pull-ups can in some systems eliminate the need for external ones.

Bit Rate Generator Unit

This unit controls the period of SCL when operating in a Master mode. The SCL period is controlled by settings in the TWI Bit Rate Register (TWBR) and the Prescaler bits in the TWI Status Register (TWSR). Slave operation does not depend on Bit Rate or Prescaler settings, but the CPU clock frequency in the Slave must be at least 16 times higher than the SCL frequency. Note that Slaves may prolong the SCL low period, thereby reducing the average TWI bus clock period. The SCL frequency is generated according to the following equation:

SCL frequency =
$$\frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot 4^{TWPS}}$$

- TWBR = Value of the TWI Bit Rate Register
- · TWPS = Value of the prescaler bits in the TWI Status Register

Note: Note: Pull-up resistor values should be selected according to the SCL frequency and the capacitive bus line load. See Table 120 on page 294 for value of pull-up resistor.

Bus Interface Unit

This unit contains the Data and Address Shift Register (TWDR), a START/STOP Controller and Arbitration detection hardware. The TWDR contains the address or data bytes to be transmitted,

<u>AIMEL</u>

178



or the address or data bytes received. In addition to the 8-bit TWDR, the Bus Interface Unit also contains a register containing the (N)ACK bit to be transmitted or received. This (N)ACK Register is not directly accessible by the application software. However, when receiving, it can be set or cleared by manipulating the TWI Control Register (TWCR). When in Transmitter mode, the value of the received (N)ACK bit can be determined by the value in the TWSR.

The START/STOP Controller is responsible for generation and detection of START, REPEATED START, and STOP conditions. The START/STOP controller is able to detect START and STOP conditions even when the AVR MCU is in one of the sleep modes, enabling the MCU to wake up if addressed by a Master.

If the TWI has initiated a transmission as Master, the Arbitration Detection hardware continuously monitors the transmission trying to determine if arbitration is in process. If the TWI has lost an arbitration, the Control Unit is informed. Correct action can then be taken and appropriate status codes generated.

Address Match Unit

The Address Match unit checks if received address bytes match the 7-bit address in the TWI Address Register (TWAR). If the TWI General Call Recognition Enable (TWGCE) bit in the TWAR is written to one, all incoming address bits will also be compared against the General Call address. Upon an address match, the Control Unit is informed, allowing correct action to be taken. The TWI may or may not acknowledge its address, depending on settings in the TWCR. The Address Match unit is able to compare addresses even when the AVR MCU is in sleep mode, enabling the MCU to wake up if addressed by a Master.

Control Unit

The Control unit monitors the TWI bus and generates responses corresponding to settings in the TWI Control Register (TWCR). When an event requiring the attention of the application occurs on the TWI bus, the TWI Interrupt Flag (TWINT) is asserted. In the next clock cycle, the TWI Status Register (TWSR) is updated with a status code identifying the event. The TWSR only contains relevant status information when the TWI Interrupt Flag is asserted. At all other times, the TWSR contains a special status code indicating that no relevant status information is available. As long as the TWINT Flag is set, the SCL line is held low. This allows the application software to complete its tasks before allowing the TWI transmission to continue.

The TWINT Flag is set in the following situations:

- · After the TWI has transmitted a START/REPEATED START condition
- After the TWI has transmitted SLA+R/W
- · After the TWI has transmitted an address byte
- · After the TWI has lost arbitration
- · After the TWI has been addressed by own Slave address or general call
- After the TWI has received a data byte
- · After a STOP or REPEATED START has been received while still addressed as a Slave.
- When a bus error has occurred due to an illegal START or STOP condition

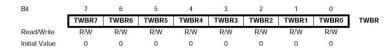


179



TWI Register Description

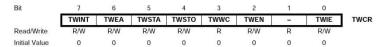
TWI Bit Rate Register – TWBR



· Bits 7..0 - TWI Bit Rate Register

TWBR selects the division factor for the bit rate generator. The bit rate generator is a frequency divider which generates the SCL clock frequency in the Master modes. See "Bit Rate Generator Unit" on page 178 for calculating bit rates.

TWI Control Register – TWCR



The TWCR is used to control the operation of the TWI. It is used to enable the TWI, to initiate a Master access by applying a START condition to the bus, to generate a receiver acknowledge, to generate a stop condition, and to control halting of the bus while the data to be written to the bus are written to the TWDR. It also indicates a write collision if data is attempted written to TWDR while the register is inaccessible.

• Bit 7 - TWINT: TWI Interrupt Flag

This bit is set by hardware when the TWI has finished its current job and expects application software response. If the I-bit in SREG and TWIE in TWCR are set, the MCU will jump to the TWI Interrupt Vector. While the TWINT Flag is set, the SCL low period is stretched. The TWINT Flag must be cleared by software by writing a logic one to it. Note that this flag is not automatically cleared by hardware when executing the interrupt routine. Also note that clearing this flag starts the operation of the TWI, so all accesses to the TWI Address Register (TWAR), TWI Status Register (TWSR), and TWI Data Register (TWDR) must be complete before clearing this flag.

· Bit 6 - TWEA: TWI Enable Acknowledge Bit

The TWEA bit controls the generation of the acknowledge pulse. If the TWEA bit is written to one, the ACK pulse is generated on the TWI bus if the following conditions are met:

- 1. The device's own Slave address has been received.
- 2. A general call has been received, while the TWGCE bit in the TWAR is set.
- 3. A data byte has been received in Master Receiver or Slave Receiver mode.

By writing the TWEA bit to zero, the device can be virtually disconnected from the Two-wire Serial Bus temporarily. Address recognition can then be resumed by writing the TWEA bit to one again.

• Bit 5 - TWSTA: TWI START Condition Bit

The application writes the TWSTA bit to one when it desires to become a Master on the Twowire Serial Bus. The TWI hardware checks if the bus is available, and generates a START condition on the bus if it is free. However, if the bus is not free, the TWI waits until a STOP condition



180



is detected, and then generates a new START condition to claim the bus Master status. TWSTA must be cleared by software when the START condition has been transmitted.

· Bit 4 - TWSTO: TWI STOP Condition Bit

Writing the TWSTO bit to one in Master mode will generate a STOP condition on the Two-wire Serial Bus. When the STOP condition is executed on the bus, the TWSTO bit is cleared automatically. In Slave mode, setting the TWSTO bit can be used to recover from an error condition. This will not generate a STOP condition, but the TWI returns to a well-defined unaddressed Slave mode and releases the SCL and SDA lines to a high impedance state.

· Bit 3 - TWWC: TWI Write Collision Flag

The TWWC bit is set when attempting to write to the TWI Data Register – TWDR when TWINT is low. This flag is cleared by writing the TWDR Register when TWINT is high.

· Bit 2 - TWEN: TWI Enable Bit

The TWEN bit enables TWI operation and activates the TWI interface. When TWEN is written to one, the TWI takes control over the I/O pins connected to the SCL and SDA pins, enabling the slew-rate limiters and spike filters. If this bit is written to zero, the TWI is switched off and all TWI transmissions are terminated, regardless of any ongoing operation.

· Bit 1 - Res: Reserved Bit

This bit is a reserved bit and will always read as zero.

• Bit 0 - TWIE: TWI Interrupt Enable

When this bit is written to one, and the I-bit in SREG is set, the TWI interrupt request will be activated for as long as the TWINT Flag is high.

TWI Status Register – TWSR

Bit	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

• Bits 7..3 - TWS: TWI Status

These five bits reflect the status of the TWI logic and the Two-wire Serial Bus. The different status codes are described later in this section. Note that the value read from TWSR contains both the 5-bit status value and the 2-bit prescaler value. The application designer should mask the prescaler bits to zero when checking the Status bits. This makes status checking independent of prescaler setting. This approach is used in this datasheet, unless otherwise noted.

· Bit 2 - Res: Reserved Bit

This bit is reserved and will always read as zero.



181



· Bits 1..0 - TWPS: TWI Prescaler Bits

These bits can be read and written, and control the bit rate prescaler.

Table 73. TWI Bit Rate Prescaler

TWPS1	TWPS0	Prescaler Value
0	0	1
0	1	4
1	0	16
1	1	64

To calculate bit rates, see "Bit Rate Generator Unit" on page 178. The value of TWPS1..0 is used in the equation.

TWI Data Register – TWDR

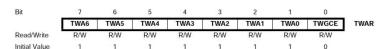
Bit	7	6	5	4	3	2	1	0	
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W								
Initial Value	1	1	1	1	1	1	1	1	

In Transmit mode, TWDR contains the next byte to be transmitted. In Receive mode, the TWDR contains the last byte received. It is writable while the TWI is not in the process of shifting a byte. This occurs when the TWI Interrupt Flag (TWINT) is set by hardware. Note that the Data Register cannot be initialized by the user before the first interrupt occurs. The data in TWDR remains stable as long as TWINT is set. While data is shifted out, data on the bus is simultaneously shifted in. TWDR always contains the last byte present on the bus, except after a wake up from a sleep mode by the TWI interrupt. In this case, the contents of TWDR is undefined. In the case of a lost bus arbitration, no data is lost in the transition from Master to Slave. Handling of the ACK bit is controlled automatically by the TWI logic, the CPU cannot access the ACK bit directly.

• Bits 7..0 - TWD: TWI Data Register

These eight bits contain the next data byte to be transmitted, or the latest data byte received on the Two-wire Serial Bus.

TWI (Slave) Address Register – TWAR



The TWAR should be loaded with the 7-bit Slave address (in the seven most significant bits of TWAR) to which the TWI will respond when programmed as a Slave Transmitter or receiver. In multi-master systems, TWAR must be set in Masters which can be addressed as Slaves by other Masters.

The LSB of TWAR is used to enable recognition of the general call address (\$00). There is an associated address comparator that looks for the Slave address (or general call address if enabled) in the received serial address. If a match is found, an interrupt request is generated.

· Bits 7..1 - TWA: TWI (Slave) Address Register

These seven bits constitute the Slave address of the TWI unit.

AIMEL

182



· Bit 0 - TWGCE: TWI General Call Recognition Enable Bit

If set, this bit enables the recognition of a General Call given over the Two-wire Serial Bus.

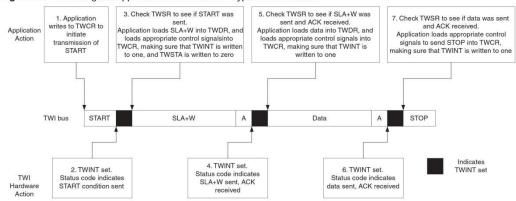
Using the TWI

The AVR TWI is byte-oriented and interrupt based. Interrupts are issued after all bus events, like reception of a byte or transmission of a START condition. Because the TWI is interrupt-based, the application software is free to carry on other operations during a TWI byte transfer. Note that the TWI Interrupt Enable (TWIE) bit in TWCR together with the Global Interrupt Enable bit in SREG allow the application to decide whether or not assertion of the TWINT Flag should generate an interrupt request. If the TWIE bit is cleared, the application must poll the TWINT Flag in order to detect actions on the TWI bus.

When the TWINT Flag is asserted, the TWI has finished an operation and awaits application response. In this case, the TWI Status Register (TWSR) contains a value indicating the current state of the TWI bus. The application software can then decide how the TWI should behave in the next TWI bus cycle by manipulating the TWCR and TWDR Registers.

Figure 85 is a simple example of how the application can interface to the TWI hardware. In this example, a Master wishes to transmit a single data byte to a Slave. This description is quite abstract, a more detailed explanation follows later in this section. A simple code example implementing the desired behavior is also presented.

Figure 85. Interfacing the Application to the TWI in a Typical Transmission



- 1. The first step in a TWI transmission is to transmit a START condition. This is done by writing a specific value into TWCR, instructing the TWI hardware to transmit a START condition. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the START condition.
- When the START condition has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the START condition has successfully been sent.
- 3. The application software should now examine the value of TWSR, to make sure that the START condition was successfully transmitted. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must load SLA+W into TWDR. Remember that TWDR is used both for address and data. After TWDR has been loaded with the



183



desired SLA+W, a specific value must be written to TWCR, instructing the TWI hardware to transmit the SLA+W present in TWDR. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the address packet.

- 4. When the address packet has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the address packet has successfully been sent. The status code will also reflect whether a Slave acknowledged the packet or not.
- 5. The application software should now examine the value of TWSR, to make sure that the address packet was successfully transmitted, and that the value of the ACK bit was as expected. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must load a data packet into TWDR. Subsequently, a specific value must be written to TWCR, instructing the TWI hardware to transmit the data packet present in TWDR. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the data packet.
- 6. When the data packet has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the data packet has successfully been sent. The status code will also reflect whether a Slave acknowledged the packet or not.
- 7. The application software should now examine the value of TWSR, to make sure that the data packet was successfully transmitted, and that the value of the ACK bit was as expected. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must write a specific value to TWCR, instructing the TWI hardware to transmit a STOP condition. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the STOP condition. Note that TWINT is NOT set after a STOP condition has been sent.

Even though this example is simple, it shows the principles involved in all TWI transmissions. These can be summarized as follows:

- When the TWI has finished an operation and expects application response, the TWINT Flag is set. The SCL line is pulled low until TWINT is cleared.
- When the TWINT Flag is set, the user must update all TWI Registers with the value relevant for the next TWI bus cycle. As an example, TWDR must be loaded with the value to be transmitted in the next bus cycle.
- After all TWI Register updates and other pending application software tasks have been completed, TWCR is written. When writing TWCR, the TWINT bit should be set. Writing a one to TWINT clears the flag. The TWI will then commence executing whatever operation was specified by the TWCR setting.

In the following an assembly and C implementation of the example is given. Note that the code below assumes that several definitions have been made, for example by using include-files.



184



As	ssembly code example	C example	Comments
1	ldi r16, (1< <twint) (1<<twsta)="" td="" ="" <=""><td>TWCR = (1<<twint) (1<<twen)<="" (1<<twsta)="" td="" =""><td>Send START condition</td></twint)></td></twint)>	TWCR = (1< <twint) (1<<twen)<="" (1<<twsta)="" td="" =""><td>Send START condition</td></twint)>	Send START condition
2	waitl: in r16,TWCR sbrs r16,TWINT rjmp waitl	while (!(TWCR & (1< <twint))) ;<="" td=""><td>Wait for TWINT Flag set. This indicates that the START condition has been transmitted</td></twint)))>	Wait for TWINT Flag set. This indicates that the START condition has been transmitted
3	in r16,TWSR andi r16, 0xF8 cpi r16, START brne ERROR	<pre>if ((TWSR & 0xF8) != START)</pre>	Check value of TWI Status Register. Mask prescaler bits. If status different from START go to ERROR
	ldi r16, SLA_W out TWDR, r16 ldi r16, (1< <twint) (1<<twen)="" out="" r16<="" td="" twcr,="" =""><td>TWDR = SLA_W; TWCR = (1<<twint) (1<<twen);<="" td="" =""><td>Load SLA_W into TWDR Register. Clear TWINT bit in TWCR to start transmission of address</td></twint)></td></twint)>	TWDR = SLA_W; TWCR = (1< <twint) (1<<twen);<="" td="" =""><td>Load SLA_W into TWDR Register. Clear TWINT bit in TWCR to start transmission of address</td></twint)>	Load SLA_W into TWDR Register. Clear TWINT bit in TWCR to start transmission of address
4	wait2: in r16,TWCR sbrs r16,TWINT rimp wait2	<pre>while (!(TWCR & (1<<twint))) ;<="" pre=""></twint)))></pre>	Wait for TWINT Flag set. This indicates that the SLA+W has been transmitted, and ACK/NACK has been received.
5	in r16,TWSR andi r16, 0xF8 cpi r16, MT_SLA_ACK brne ERROR	<pre>if ((TWSR & 0xF8) != MT_SLA_ACK)</pre>	Check value of TWI Status Register. Mask prescaler bits. If status different from MT_SLA_ACK go to ERROR
	ldi r16, DATA out TWDR, r16 ldi r16, (1< <twint) (1<<twen)="" out="" r16<="" td="" twcr,="" =""><td>TWDR = DATA; TWCR = (1<<twint) (1<<twen);<="" td="" =""><td>Load DATA into TWDR Register. Clear TWINT bit in TWCR to start transmission of data</td></twint)></td></twint)>	TWDR = DATA; TWCR = (1< <twint) (1<<twen);<="" td="" =""><td>Load DATA into TWDR Register. Clear TWINT bit in TWCR to start transmission of data</td></twint)>	Load DATA into TWDR Register. Clear TWINT bit in TWCR to start transmission of data
6	wait3: in r16,TWCR sbrs r16,TWINT rimp wait3	while (!(TWCR & (1< <twint))) ;<="" td=""><td>Wait for TWINT Flag set. This indicates that the DATA has been transmitted, and ACK/NACK has been received.</td></twint)))>	Wait for TWINT Flag set. This indicates that the DATA has been transmitted, and ACK/NACK has been received.
7	in r16,TWSR andi r16, 0xF8 cpi r16, MT_DATA_ACK brne ERROR	<pre>if ((TWSR & 0xF8) != MT_DATA_ACK)</pre>	Check value of TWI Status Register. Mask prescaler bits. If status different from MT_DATA_ACK go to ERROR
	ldi r16, (1< <twint) (1<<twen)="" td="" ="" <=""><td>TWCR = (1<<twint) (1<<twen)="" (1<<twsto);<="" td="" =""><td>Transmit STOP condition</td></twint)></td></twint)>	TWCR = (1< <twint) (1<<twen)="" (1<<twsto);<="" td="" =""><td>Transmit STOP condition</td></twint)>	Transmit STOP condition





Transmission Modes

The TWI can operate in one of four major modes. These are named Master Transmitter (MT), Master Receiver (MR), Slave Transmitter (ST) and Slave Receiver (SR). Several of these modes can be used in the same application. As an example, the TWI can use MT mode to write data into a TWI EEPROM, MR mode to read the data back from the EEPROM. If other Masters are present in the system, some of these might transmit data to the TWI, and then SR mode would be used. It is the application software that decides which modes are legal.

The following sections describe each of these modes. Possible status codes are described along with figures detailing data transmission in each of the modes. These figures contain the following abbreviations:

S: START condition

Rs: REPEATED START condition

R: Read bit (high level at SDA)

W: Write bit (low level at SDA)

A: Acknowledge bit (low level at SDA)

A: Not acknowledge bit (high level at SDA)

Data: 8-bit data byte P: STOP condition SLA: Slave Address

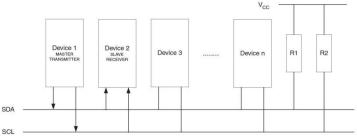
In Figure 87 to Figure 93, circles are used to indicate that the TWINT Flag is set. The numbers in the circles show the status code held in TWSR, with the prescaler bits masked to zero. At these points, actions must be taken by the application to continue or complete the TWI transfer. The TWI transfer is suspended until the TWINT Flag is cleared by software.

When the TWINT Flag is set, the status code in TWSR is used to determine the appropriate software action. For each status code, the required software action and details of the following serial transfer are given in Table 74 to Table 77. Note that the prescaler bits are masked to zero in these tables.

Master Transmitter Mode

In the Master Transmitter mode, a number of data bytes are transmitted to a Slave Receiver (see Figure 86). In order to enter a Master mode, a START condition must be transmitted. The format of the following address packet determines whether Master Transmitter or Master Receiver mode is to be entered. If SLA+W is transmitted, MT mode is entered, if SLA+R is transmitted, MR mode is entered. All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 86. Data Transfer in Master Transmitter Mode





186



A START condition is sent by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	1	Х	1	0	X	1	0	Х

TWEN must be set to enable the Two-wire Serial Interface, TWSTA must be written to one to transmit a START condition and TWINT must be written to one to clear the TWINT Flag. The TWI will then test the Two-wire Serial Bus and generate a START condition as soon as the bus becomes free. After a START condition has been transmitted, the TWINT Flag is set by hardware, and the status code in TWSR will be \$08 (See Table 74). In order to enter MT mode, SLA+W must be transmitted. This is done by writing SLA+W to TWDR. Thereafter the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	1	Х	0	0	Х	1	0	Х

When SLA+W have been transmitted and an acknowledgement bit has been received, TWINT is set again and a number of status codes in TWSR are possible. Possible status codes in Master mode are \$18, \$20, or \$38. The appropriate action to be taken for each of these status codes is detailed in Table 74.

When SLA+W has been successfully transmitted, a data packet should be transmitted. This is done by writing the data byte to TWDR. TWDR must only be written when TWINT is high. If not, the access will be discarded, and the Write Collision bit (TWWC) will be set in the TWCR Register. After updating TWDR, the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	1	Х	0	0	Х	1	0	Х

This scheme is repeated until the last byte has been sent and the transfer is ended by generating a STOP condition or a repeated START condition. A STOP condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	1	X	0	1	X	1	0	Х

A REPEATED START condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	1	Х	1	0	Х	1	0	Х

After a repeated START condition (state \$10) the Two-wire Serial Interface can access the same Slave again, or a new Slave without transmitting a STOP condition. Repeated START enables the Master to switch between Slaves, Master Transmitter mode and Master Receiver mode without losing control of the bus.

Table 74. Status Codes for Master Transmitter Mode

Status Code			ation Softw	vare Resp	onse			
(TWSR) Status of the Two-wire Serial				To	TWCR			
are 0	Prescaler Bits Bus and Two-wire Serial Inter- re 0 face Hardware		STA	STO	TWINT	TWEA	Next Action Taken by TWI Hardware	
\$08	A START condition has been transmitted	Load SLA+W	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received	
\$10	A repeated START condition has been transmitted	Load SLA+W or	0	0	1	Х	SLA+W will be transmitted; ACK or NOT ACK will be received	
		Load SLA+R	0	0	1	X	SLA+R will be transmitted; Logic will switch to Master Receiver mode	



187



Table 74. Status Codes for Master Transmitter Mode

\$18	SLA+W has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	Х	STOP condition will be transmitted and TWSTO Flag will be Reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be Reset
\$20	SLA+W has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	х	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	Х	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
\$28	Data byte has been transmitted; ACK has been received	Load data byte or	0	0	1	х	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	Х	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	Х	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
\$30	Data byte has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
	STAGEN AND SANDS CONCERNS WAS TO CONTRACT AND STAGE ST	No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	Х	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
\$38	Arbitration lost in SLA+W or data bytes	No TWDR action or	0	0	1	X	Two-wire Serial Bus will be released and not addressed Slave mode entered
		No TWDR action	1	0	1	Х	A START condition will be transmitted when the bus be- comes free



188



DATA \$08 \$18 \$28 SLA W R_S (\$10) Not acknowledge received after the slave address R Р \$20 \$30 A or \overline{A} A or A \$38 \$38 \$68 \$78 \$B0 DATA This number (contained in TWSR) corresponds to a defined state of the Two-wire Serial Bus. The prescaler bits are zero or masked to zero From slave to master (n)

Figure 87. Formats and States in the Master Transmitter Mode

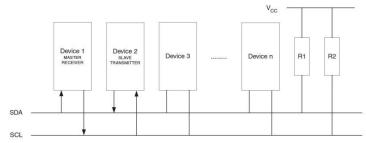
Master Receiver Mode

In the Master Receiver mode, a number of data bytes are received from a Slave Transmitter (see Figure 88). In order to enter a Master mode, a START condition must be transmitted. The format of the following address packet determines whether Master Transmitter or Master Receiver mode is to be entered. If SLA+W is transmitted, MT mode is entered, if SLA+R is transmitted, MR mode is entered. All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.





Figure 88. Data Transfer in Master Receiver Mode



A START condition is sent by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	1	Х	1	0	Х	1	0	Х

TWEN must be written to one to enable the Two-wire Serial Interface, TWSTA must be written to one to transmit a START condition and TWINT must be set to clear the TWINT Flag. The TWI will then test the Two-wire Serial Bus and generate a START condition as soon as the bus becomes free. After a START condition has been transmitted, the TWINT Flag is set by hardware, and the status code in TWSR will be \$08 (See Table 74). In order to enter MR mode, SLA+R must be transmitted. This is done by writing SLA+R to TWDR. Thereafter the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	1	Х	0	0	Х	1	0	Х

When SLA+R have been transmitted and an acknowledgement bit has been received, TWINT is set again and a number of status codes in TWSR are possible. Possible status codes in Master mode are \$38, \$40, or \$48. The appropriate action to be taken for each of these status codes is detailed in Table 75. Received data can be read from the TWDR Register when the TWINT Flag is set high by hardware. This scheme is repeated until the last byte has been received. After the last byte has been received, the MR should inform the ST by sending a NACK after the last received data byte. The transfer is ended by generating a STOP condition or a repeated START condition. A STOP condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	1	Х	0	1	X	1	0	Х

A REPEATED START condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	1	Х	1	0	Х	1	0	Х

After a repeated START condition (state \$10) the Two-wire Serial Interface can access the same Slave again, or a new Slave without transmitting a STOP condition. Repeated START enables the Master to switch between Slaves, Master Transmitter mode and Master Receiver mode without losing control over the bus.

Table 75. Status Codes for Master Receiver Mode

Status Code	Status of the Two-wire Serial Bus and Two-wire Serial Inter-	Applica	tion Softw	are Resp	onse		
(TWSR) Prescaler Bits			To TWCR				
are 0	face Hardware	To/from TWDR	STA	STO	TWINT	TWEA	Next Action Taken by TWI Hardware

ATTEL

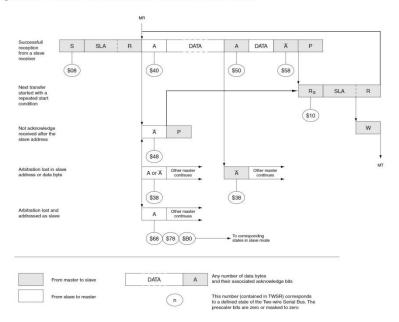
190



Table 75. Status Codes for Master Receiver Mode (Continued)

\$08	A START condition has been transmitted	Load SLA+R	0	0	1	Х	SLA+R will be transmitted ACK or NOT ACK will be received
\$10	A repeated START condition has been transmitted	Load SLA+R or	0	0	1	x	SLA+R will be transmitted ACK or NOT ACK will be received
		Load SLA+W	0	0	1	Х	SLA+W will be transmitted Logic will switch to masTer Transmitter mode
\$38	Arbitration lost in SLA+R or NOT ACK bit	No TWDR action or	0	0	1	Х	Two-wire Serial Bus will be released and not addressed Slave mode will be entered
		No TWDR action	1	0	1	X	A START condition will be transmitted when the bus becomes free
\$40	SLA+R has been transmitted; ACK has been received	No TWDR action or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	0	0	1	1	Data byte will be received and ACK will be returned
\$48	SLA+R has been transmitted;	No TWDR action or	1	0	1	X	Repeated START will be transmitted
	NOT ACK has been received	No TWDR action or	0	1	1	Х	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
\$50	Data byte has been received; ACK has been returned	Read data byte or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	0	0	1	1	Data byte will be received and ACK will be returned
\$58	Data byte has been received;	Read data byte or	1	0	1	Х	Repeated START will be transmitted
	NOT ACK has been returned	Read data byte or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		Read data byte	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset

Figure 89. Formats and States in the Master Receiver Mode





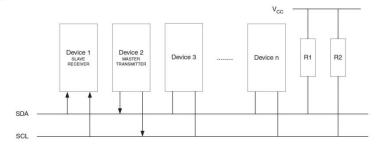
191



Slave Receiver Mode

In the Slave Receiver mode, a number of data bytes are received from a Master Transmitter (see Figure 90). All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 90. Data Transfer in Slave Receiver Mode



To initiate the Slave Receiver mode, TWAR and TWCR must be initialized as follows:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Value			Device's	s Own Slave	Address			

The upper seven bits are the address to which the Two-wire Serial Interface will respond when addressed by a Master. If the LSB is set, the TWI will respond to the general call address (\$00), otherwise it will ignore the general call address.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	0	1	0	0	0	1	0	Х

TWEN must be written to one to enable the TWI. The TWEA bit must be written to one to enable the acknowledgement of the device's own Slave address or the general call address. TWSTA and TWSTO must be written to zero.

When TWAR and TWCR have been initialized, the TWI waits until it is addressed by its own Slave address (or the general call address if enabled) followed by the data direction bit. If the direction bit is "0" (write), the TWI will operate in SR mode, otherwise ST mode is entered. After its own Slave address and the write bit have been received, the TWINT Flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate software action. The appropriate action to be taken for each status code is detailed in Table 76. The Slave Receiver mode may also be entered if arbitration is lost while the TWI is in the Master mode (see states \$68 and \$78).

If the TWEA bit is reset during a transfer, the TWI will return a "Not Acknowledge" ("1") to SDA after the next received data byte. This can be used to indicate that the Slave is not able to receive any more bytes. While TWEA is zero, the TWI does not acknowledge its own Slave address. However, the Two-wire Serial Bus is still monitored and address recognition may resume at any time by setting TWEA. This implies that the TWEA bit may be used to temporarily isolate the TWI from the Two-wire Serial Bus.

In all sleep modes other than Idle Mode, the clock system to the TWI is turned off. If the TWEA bit is set, the interface can still acknowledge its own Slave address or the general call address by using the Two-wire Serial Bus clock as a clock source. The part will then wake up from sleep and the TWI will hold the SCL clock low during the wake up and until the TWINT Flag is cleared (by writing it to one). Further data reception will be carried out as normal, with the AVR clocks running as normal. Observe that if the AVR is set up with a long start-up time, the SCL line may be held low for a long time, blocking other data transmissions.



192



Note that the Two-wire Serial Interface Data Register – TWDR does not reflect the last byte present on the bus when waking up from these sleep modes.



193



Table 76. Status Codes for Slave Receiver Mode

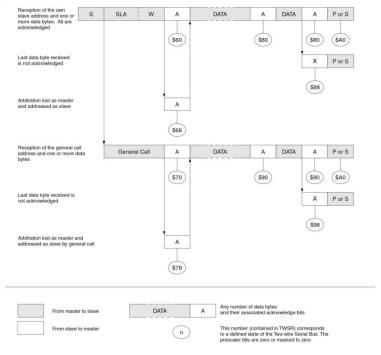
Status Code		Applica	tion Softw	are Resp	onse		
(TWSR) Prescaler Bits	Status of the Two-wire Serial Bus and Two-wire Serial Interface	To/from TWDR			TWCR		
are 0	Hardware	TO/ITOITITIVER	STA	STO	TWINT	TWEA	Next Action Taken by TWI Hardware
\$60	Own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
***	A Lin Lin Lin CLA DAY	No TWDR action	X		- 15		Data byte will be received and ACK will be returned
\$68	Arbitration lost in SLA+R/W as Master; own SLA+W has been received: ACK has been returned	No TWDR action or No TWDR action	X	0	1	0	Data byte will be received and NOT ACK will be returned Data byte will be received and ACK will be returned
\$70	General call address has been	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be
370	received; ACK has been returned	No TWDR action	x	0	1	1	returned Data byte will be received and ACK will be returned
\$78	Arbitration lost in SLA+R/W as Master; General call address has	No TWDR action or	Х	0	1	0	Data byte will be received and NOT ACK will be returned
	been received; ACK has been returned	No TWDR action	Х	0	1	1	Data byte will be received and ACK will be returned
\$80	Previously addressed with own SLA+W; data has been received;	Read data byte or	Х	0	1	0	Data byte will be received and NOT ACK will be returned
	ACK has been returned	Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
\$88	Previously addressed with own SLA+W; data has been received;	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
	NOT ACK has been returned	Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free
\$90	Previously addressed with general call; data has been re-	Read data byte or	Х	0	1	0	Data byte will be received and NOT ACK will be returned
	ceived; ACK has been returned	Read data byte	Х	0	1	1	Data byte will be received and ACK will be returned
\$98	Previously addressed with general call; data has been	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
	received; NOT ACK has been returned	Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free
\$A0	A STOP condition or repeated START condition has been	No action	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
received while still addressed as Slave		0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"	
			1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
			1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free



194

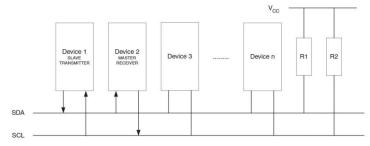


Figure 91. Formats and States in the Slave Receiver Mode



Slave Transmitter Mode In the Slave Transmitter mode, a number of data bytes are transmitted to a Master Receiver (see Figure 92). All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 92. Data Transfer in Slave Transmitter Mode



To initiate the Slave Transmitter mode, TWAR and TWCR must be initialized as follows:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Value			Device'	s Own Slave	Address			

<u>AMEL</u>

195



The upper seven bits are the address to which the Two-wire Serial Interface will respond when addressed by a Master. If the LSB is set, the TWI will respond to the general call address (\$00), otherwise it will ignore the general call address.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Value	0	1	0	0	0	1	0	Х

TWEN must be written to one to enable the TWI. The TWEA bit must be written to one to enable the acknowledgement of the device's own Slave address or the general call address. TWSTA and TWSTO must be written to zero.

When TWAR and TWCR have been initialized, the TWI waits until it is addressed by its own Slave address (or the general call address if enabled) followed by the data direction bit. If the direction bit is "1" (read), the TWI will operate in ST mode, otherwise SR mode is entered. After its own Slave address and the wite bit have been received, the TWINT Flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate software action. The appropriate action to be taken for each status code is detailed in Table 77. The Slave Transmitter mode may also be entered if arbitration is lost while the TWI is in the Master mode (see state \$BO).

If the TWEA bit is written to zero during a transfer, the TWI will transmit the last byte of the transfer. State \$C0 or state \$C8 will be entered, depending on whether the Master Receiver transmits a NACK or ACK after the final byte. The TWI is switched to the not addressed Slave mode, and will ignore the Master if it continues the transfer. Thus the Master Receiver receives all "1" as serial data. State \$C8 is entered if the Master demands additional data bytes (by transmitting ACK), even though the Slave has transmitted the last byte (TWEA zero and expecting NACK from the Master).

While TWEA is zero, the TWI does not respond to its own Slave address. However, the Twowire Serial Bus is still monitored and address recognition may resume at any time by setting TWEA. This implies that the TWEA bit may be used to temporarily isolate the TWI from the Twowire Serial Bus

In all sleep modes other than Idle mode, the clock system to the TWI is turned off. If the TWEA bit is set, the interface can still acknowledge its own Slave address or the general call address by using the Two-wire Serial Bus clock as a clock source. The part will then wake up from sleep and the TWI will hold the SCL clock will low during the wake up and until the TWINT Flag is cleared (by writing it to one). Further data transmission will be carried out as normal, with the AVR clocks running as normal. Observe that if the AVR is set up with a long start-up time, the SCL line may be held low for a long time, blocking other data transmissions.

Note that the Two-wire Serial Interface Data Register – TWDR does not reflect the last byte present on the bus when waking up from these sleep modes.





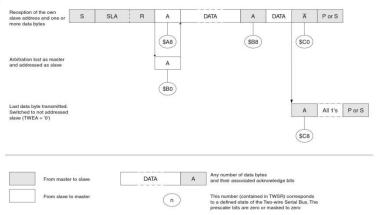
Table 77. Status Codes for Slave Transmitter Mode

Status Code		Applica	tion Softv	vare Resp	oonse		
(TWSR)	Status of the Two-wire Serial Bus			To	TWCR		
Prescaler Bits are 0	and Two-wire Serial Interface Hardware	To/from TWDR	STA	STO	TWINT	TWEA	Next Action Taken by TWI Hardware
\$A8	Own SLA+R has been received; ACK has been returned	Load data byte or	Х	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	Х	0	1	1	Data byte will be transmitted and ACK should be re- ceived
\$B0	Arbitration lost in SLA+R/W as Master; own SLA+R has been	Load data byte or	Х	0	1	0	Last data byte will be transmitted and NOT ACK should be received
	received; ACK has been returned	Load data byte	Х	0	1	1	Data byte will be transmitted and ACK should be re- ceived
\$B8	Data byte in TWDR has been transmitted; ACK has been	Load data byte or	Х	0	1	0	Last data byte will be transmitted and NOT ACK should be received
	received	Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be re- ceived
Data byte in TWDR has been transmitted; NOT ACK has been received		No TWDR action or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
	received	No TWDR action or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		No TWDR action or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free
\$C8	Last data byte in TWDR has been transmitted (TWEA = "0"); ACK	No TWDR action or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
	has been received	No TWDR action or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		No TWDR action or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free





Figure 93. Formats and States in the Slave Transmitter Mode



Miscellaneous States

There are two status codes that do not correspond to a defined TWI state, see Table 78.

Status \$F8 indicates that no relevant information is available because the TWINT Flag is not set. This occurs between other states, and when the TWI is not involved in a serial transfer.

Status \$00 indicates that a bus error has occurred during a Two-wire Serial Bus transfer. A bus error occurs when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte, or an acknowledge bit. When a bus error occurs, TWINT is set. To recover from a bus error, the TWSTO Flag must set and TWINT must be cleared by writing a logic one to it. This causes the TWI to enter the not addressed Slave mode and to clear the TWSTO Flag (no other bits in TWCR are affected). The SDA and SCL lines are released, and no STOP condition is transmitted.

Table 78. Miscellaneous States

Status Code			ation Softv	vare Resp	onse		
(TWSR) Prescaler Bits	Status of the Two-wire Serial Bus and Two-wire Serial Inter-			To	TWCR		
are 0	face Hardware	To/from TWDR	STA	STO	TWINT	TWEA	Next Action Taken by TWI Hardware
\$F8	No relevant state information available; TWINT = "0"	No TWDR action		No TW	CR action		Wait or proceed current transfer
\$00	Bus error due to an illegal START or STOP condition	No TWDR action	0	1	1	X	Only the internal hardware is affected, no STOP condition is sent on the bus. In all cases, the bus is released and TWSTO is cleared.



198



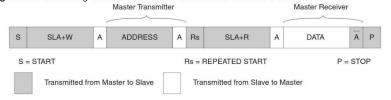
Combining Several TWI Modes

In some cases, several TWI modes must be combined in order to complete the desired action. Consider for example reading data from a serial EEPROM. Typically, such a transfer involves the following steps:

- 1. The transfer must be initiated
- 2. The EEPROM must be instructed what location should be read
- 3. The reading must be performed
- 4. The transfer must be finished

Note that data is transmitted both from Master to Slave and vice versa. The Master must instruct the Slave what location it wants to read, requiring the use of the MT mode. Subsequently, data must be read from the Slave, implying the use of the MR mode. Thus, the transfer direction must be changed. The Master must keep control of the bus during all these steps, and the steps should be carried out as an atomical operation. If this principle is violated in a multi-master system, another Master can alter the data pointer in the EEPROM between steps 2 and 3, and the Master will read the wrong data location. Such a change in transfer direction is accomplished by transmitting a REPEATED START between the transmission of the address byte and reception of the data. After a REPEATED START, the Master keeps ownership of the bus. The following figure shows the flow in this transfer.

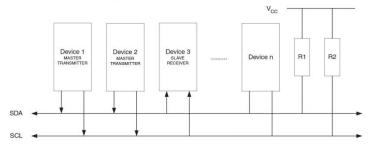
Figure 94. Combining Several TWI Modes to Access a Serial EEPROM



Multi-master Systems and Arbitration

If multiple Masters are connected to the same bus, transmissions may be initiated simultaneously by one or more of them. The TWI standard ensures that such situations are handled in such a way that one of the Masters will be allowed to proceed with the transfer, and that no data will be lost in the process. An example of an arbitration situation is depicted below, where two Masters are trying to transmit data to a Slave Receiver.

Figure 95. An Arbitration Example



AMEL

199

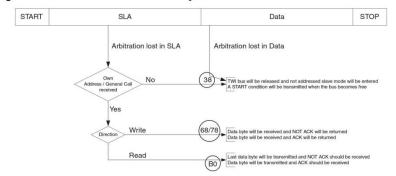


Several different scenarios may arise during arbitration, as described below:

- Two or more Masters are performing identical communication with the same Slave. In this
 case, neither the Slave nor any of the Masters will know about the bus contention.
- Two or more Masters are accessing the same Slave with different data or direction bit. In this
 case, arbitration will occur, either in the READ/WRITE bit or in the data bits. The Masters
 trying to output a one on SDA while another Master outputs a zero will lose the arbitration.
 Losing Masters will switch to not addressed Slave mode or wait until the bus is free and
 transmit a new START condition, depending on application software action.
- Two or more Masters are accessing different Slaves. In this case, arbitration will occur in the SLA bits. Masters trying to output a one on SDA while another Master outputs a zero will lose the arbitration. Masters losing arbitration in SLA will switch to Slave mode to check if they are being addressed by the winning Master. If addressed, they will switch to SR or ST mode, depending on the value of the READ/WRITE bit. If they are not being addressed, they will switch to not addressed Slave mode or wait until the bus is free and transmit a new START condition, depending on application software action.

This is summarized in Figure 96. Possible status values are given in circles.

Figure 96. Possible Status Codes Caused by Arbitration





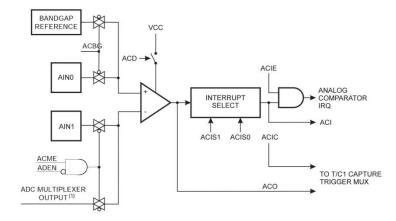
200



Analog Comparator

The Analog Comparator compares the input values on the positive pin AIN0 and negative pin AIN1. When the voltage on the positive pin AIN0 is higher than the voltage on the negative pin AIN1, the Analog Comparator Output, ACO, is set. The comparator's output can be set to trigger the Timer/Counter1 Input Capture function. In addition, the comparator can trigger a separate interrupt, exclusive to the Analog Comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 97.

Figure 97. Analog Comparator Block Diagram⁽²⁾



- Notes: 1. See Table 80 on page 203.
 - 2. Refer to Figure 1 on page 2 and Table 25 on page 58 for Analog Comparator pin placement.

Special Function IO Register – SFIOR



· Bit 3 - ACME: Analog Comparator Multiplexer Enable

When this bit is written logic one and the ADC is switched off (ADEN in ADCSRA is zero), the ADC multiplexer selects the negative input to the Analog Comparator. When this bit is written logic zero, AIN1 is applied to the negative input of the Analog Comparator. For a detailed description of this bit, see "Analog Comparator Multiplexed Input" on page 203.



201



Analog Comparator Control and Status Register – ACSR

Bit	7	6	5	4	3	2	1	0	
	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	0	0	0	0	0	

· Bit 7 - ACD: Analog Comparator Disable

When this bit is written logic one, the power to the Analog Comparator is switched off. This bit can be set at any time to turn off the Analog Comparator. This will reduce power consumption in active and Idle mode. When changing the ACD bit, the Analog Comparator Interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise an interrupt can occur when the bit is changed.

Bit 6 – ACBG: Analog Comparator Bandgap Select

When this bit is set, a fixed bandgap reference voltage replaces the positive input to the Analog Comparator. When this bit is cleared, AIN0 is applied to the positive input of the Analog Comparator. See "Internal Voltage Reference" on page 42.

• Bit 5 - ACO: Analog Comparator Output

The output of the Analog Comparator is synchronized and then directly connected to ACO. The synchronization introduces a delay of 1 - 2 clock cycles.

· Bit 4 - ACI: Analog Comparator Interrupt Flag

This bit is set by hardware when a comparator output event triggers the interrupt mode defined by ACIS1 and ACIS0. The Analog Comparator Interrupt routine is executed if the ACIE bit is set and the I-bit in SREG is set. ACI is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ACI is cleared by writing a logic one to the flag.

· Bit 3 - ACIE: Analog Comparator Interrupt Enable

When the ACIE bit is written logic one and the I-bit in the Status Register is set, the Analog Comparator Interrupt is activated. When written logic zero, the interrupt is disabled.

Bit 2 – ACIC: Analog Comparator Input Capture Enable

When written logic one, this bit enables the Input Capture function in Timer/Counter1 to be triggered by the Analog Comparator. The comparator output is in this case directly connected to the Input Capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter1 Input Capture interrupt. When written logic zero, no connection between the Analog Comparator and the Input Capture function exists. To make the comparator trigger the Timer/Counter1 Input Capture interrupt, the TICIE1 bit in the Timer Interrupt Mask Register (TIMSK) must be set.

• Bits 1, 0 - ACIS1, ACIS0: Analog Comparator Interrupt Mode Select

These bits determine which comparator events that trigger the Analog Comparator interrupt. The different settings are shown in Table 79.



202



Table 79. ACIS1/ACIS0 Settings

ACIS1	ACIS0	Interrupt Mode	
0	0	Comparator Interrupt on Output Toggle	
0	1	Reserved	
1	0	Comparator Interrupt on Falling Output Edge	
1	1	Comparator Interrupt on Rising Output Edge	

When changing the ACIS1/ACIS0 bits, the Analog Comparator Interrupt must be disabled by clearing its Interrupt Enable bit in the ACSR Register. Otherwise an interrupt can occur when the bits are changed.

Analog Comparator Multiplexed Input It is possible to select any of the ADC7..0 pins to replace the negative input to the Analog Comparator. The ADC multiplexer is used to select this input, and consequently, the ADC must be switched off to utilize this feature. If the Analog Comparator Multiplexer Enable bit (ACME in SFIOR) is set and the ADC is switched off (ADEN in ADCSRA is zero), MUX2..0 in ADMUX select the input pin to replace the negative input to the Analog Comparator, as shown in Table 80. If ACME is cleared or ADEN is set, AIN1 is applied to the negative input to the Analog Comparator.

Table 80. Analog Comparator Multiplexed Input

ACME	ADEN	MUX20	Analog Comparator Negative Input
0	х	xxx	AIN1
1	1	XXX	AIN1
1	0	000	ADC0
1	0	001	ADC1
1	0	010	ADC2
1	0	011	ADC3
1	0	100	ADC4
1	0	101	ADC5
1	0	110	ADC6
1	0	111	ADC7





Analog to **Digital** Converter

Features

- · 10-bit Resolution
- · 0.5 LSB Integral Non-linearity
- ±2 LSB Absolute Accuracy
- 13 µs- 260 µs Conversion Time
 Up to 15 kSPS at Maximum Resolution
- 8 Multiplexed Single Ended Input Channels
- 7 Differential Input Channels
- · 2 Differential Input Channels with Optional Gain of 10x and 200x
- Optional Left adjustment for ADC Result Readout
- 0 V_{CC} ADC Input Voltage Range
- Selectable 2.56V ADC Reference Voltage
- Free Running or Single Conversion Mode
- ADC Start Conversion by Auto Triggering on Interrupt Sources
- Interrupt on ADC Conversion Complete
- · Sleep Mode Noise Canceler

The ATmega16 features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer which allows 8 single-ended voltage inputs constructed from the pins of Port A. The single-ended voltage inputs refer to 0V (GND).

The device also supports 16 differential voltage input combinations. Two of the differential inputs (ADC1, ADC0 and ADC3, ADC2) are equipped with a programmable gain stage, providing amplification steps of 0 dB (1x), 20 dB (10x), or 46 dB (200x) on the differential input voltage before the A/D conversion. Seven differential analog input channels share a common negative terminal (ADC1), while any other ADC input can be selected as the positive input terminal. If 1x or 10x gain is used, 8-bit resolution can be expected. If 200x gain is used, 7-bit resolution can be expected.

The ADC contains a Sample and Hold circuit which ensures that the input voltage to the ADC is held at a constant level during conversion. A block diagram of the ADC is shown in Figure 98.

The ADC has a separate analog supply voltage pin, AVCC. AVCC must not differ more than ±0.3V from V_{CC}. See the paragraph "ADC Noise Canceler" on page 211 on how to connect this

Internal reference voltages of nominally 2.56V or AVCC are provided On-chip. The voltage reference may be externally decoupled at the AREF pin by a capacitor for better noise performance.





AVCC MITERNAL 2.56V REFERENCE

ADD AULTIFLEXER SELECT (ADMIX)

B SIT DATA BUS

ADD AULTIFLEXER SELECT (ADMIX)

ADD AULTIFLEXER SELECT (ADMIX)

B SIT DATA BUS

ADD AULTIFLEXER SELECT (ADMIX)

ADD AULTIFLEXER SELECT (ADMIX)

B SIT DATA BUS

ADD AULTIFLEXER SELECT (ADMIX)

ADD AULTIFLEXER OUTFUT

ADD AUTFUT

ADD AULTIFLEXER OUTFUT

ADD AUTFUT

ADD AULTIFLEXER OUTFUT

ADD AUTFUT

ADD AULTIFLEXER OUTFUT

ADD AULTIFLEXER OUTFUT

ADD AUTFUT

ADD AULTIFLEXER OUTFUT

ADD AULTIFLEXER OUTFUT

ADD AUTFUT

ADD

Figure 98. Analog to Digital Converter Block Schematic

Operation

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents GND and the maximum value represents the voltage on the AREF pin minus 1 LSB. Optionally, AVCC or an internal 2.56V reference voltage may be connected to the AREF pin by writing to the REFSn bits in the ADMUX Register. The internal voltage reference may thus be decoupled by an external capacitor at the AREF pin to improve noise immunity.

The analog input channel and differential gain are selected by writing to the MUX bits in ADMUX. Any of the ADC input pins, as well as GND and a fixed bandgap voltage reference, can be selected as single ended inputs to the ADC. A selection of ADC input pins can be selected as positive and negative inputs to the differential gain amplifier.

If differential channels are selected, the differential gain stage amplifies the voltage difference between the selected input channel pair by the selected gain factor. This amplified value then

AIMEL

205



becomes the analog input to the ADC. If single ended channels are used, the gain amplifier is bypassed altogether.

The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSRA. Voltage reference and input channel selections will not go into effect until ADEN is set. The ADC does not consume power when ADEN is cleared, so it is recommended to switch off the ADC before entering power saving sleep modes.

The ADC generates a 10-bit result which is presented in the ADC Data Registers, ADCH and ADCL. By default, the result is presented right adjusted, but can optionally be presented left adjusted by setting the ADLAR bit in ADMUX.

If the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH, to ensure that the content of the Data Registers belongs to the same conversion. Once ADCL is read, ADC access to Data Registers is blocked. This means that if ADCL has been read, and a conversion completes before ADCH is read, neither register is updated and the result from the conversion is lost. When ADCH is read, ADC access to the ADCH and ADCL Registers is re-enabled.

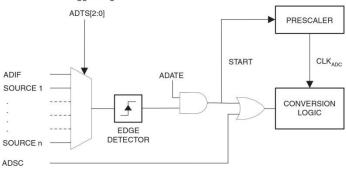
The ADC has its own interrupt which can be triggered when a conversion completes. When ADC access to the Data Registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

Starting a Conversion

A single conversion is started by writing a logical one to the ADC Start Conversion bit, ADSC. This bit stays high as long as the conversion is in progress and will be cleared by hardware when the conversion is completed. If a different data channel is selected while a conversion is in progress, the ADC will finish the current conversion before performing the channel change.

Alternatively, a conversion can be triggered automatically by various sources. Auto Triggering is enabled by setting the ADC Auto Trigger Enable bit, ADATE in ADCSRA. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in SFIOR (see description of the ADTS bits for a list of the trigger sources). When a positive edge occurs on the selected trigger signal, the ADC prescaler is reset and a conversion is started. This provides a method of starting conversions at fixed intervals. If the trigger signal still is set when the conversion completes, a new conversion will not be started. If another positive edge occurs on the trigger signal during conversion, the edge will be ignored. Note that an Interrupt Flag will be set even if the specific interrupt is disabled or the global interrupt enable bit in SREG is cleared. A conversion can thus be triggered without causing an interrupt. However, the Interrupt Flag must be cleared in order to trigger a new conversion at the next interrupt event.

Figure 99. ADC Auto Trigger Logic



AIMEL

206

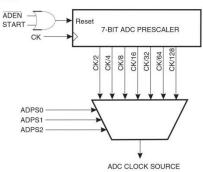


Using the ADC Interrupt Flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in Free Running mode, constantly sampling and updating the ADC Data Register. The first conversion must be started by writing a logical one to the ADSC bit in ADCSRA. In this mode the ADC will perform successive conversions independently of whether the ADC Interrupt Flag, ADIF is cleared or not.

If Auto Triggering is enabled, single conversions can be started by writing ADSC in ADCSRA to one. ADSC can also be used to determine if a conversion is in progress. The ADSC bit will be read as one during a conversion, independently of how the conversion was started.

Prescaling and Conversion Timing

Figure 100. ADC Prescaler



By default, the successive approximation circuitry requires an input clock frequency between 50 kHz and 200 kHz to get maximum resolution. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be higher than 200 kHz to get a higher sample rate.

The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100 kHz. The prescaling is set by the ADPS bits in ADCSRA. The prescaler starts counting from the moment the ADC is switched on by setting the ADEN bit in ADCSRA. The prescaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

When initiating a single ended conversion by setting the ADSC bit in ADCSRA, the conversion starts at the following rising edge of the ADC clock cycle. See "Differential Gain Channels" on page 209 for details on differential conversion timing.

A normal conversion takes 13 ADC clock cycles. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 25 ADC clock cycles in order to initialize the analog circuitry.

The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 13.5 ADC clock cycles after the start of a first conversion. When a conversion is complete, the result is written to the ADC Data Registers, and ADIF is set. In single conversion mode, ADSC is cleared simultaneously. The software may then set ADSC again, and a new conversion will be initiated on the first rising ADC clock edge.

When Auto Triggering is used, the prescaler is reset when the trigger event occurs. This assures a fixed delay from the trigger event to the start of conversion. In this mode, the sample-and-hold takes place 2 ADC clock cycles after the rising edge on the trigger source signal. Three additional CPU clock cycles are used for synchronization logic. When using Differential mode, along with Auto triggering from a source other than the ADC Conversion Complete, each conversion



207



will require 25 ADC clocks. This is because the ADC must be disabled and re-enabled after every conversion.

In Free Running mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. For a summary of conversion times, see Table 81.

Figure 101. ADC Timing Diagram, First Conversion (Single Conversion Mode)

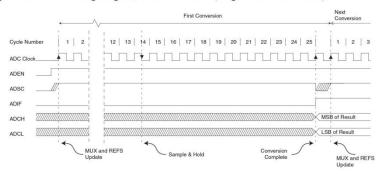
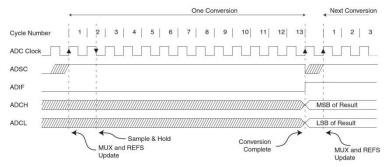


Figure 102. ADC Timing Diagram, Single Conversion





208



Figure 103. ADC Timing Diagram, Auto Triggered Conversion

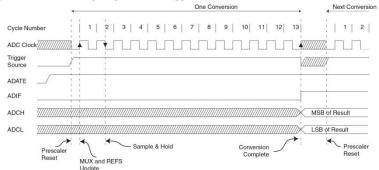


Figure 104. ADC Timing Diagram, Free Running Conversion

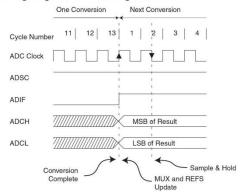


Table 81. ADC Conversion Time

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	13.5	25
Normal conversions, single ended	1.5	13
Auto Triggered conversions	2	13.5
Normal conversions, differential	1.5/2.5	13/14

Differential Gain Channels

When using differential gain channels, certain aspects of the conversion need to be taken into consideration.

Differential conversions are synchronized to the internal clock CK_{ADC2} equal to half the ADC clock. This synchronization is done automatically by the ADC interface in such a way that the sample-and-hold occurs at a specific phase of CK_{ADC2} . A conversion initiated by the user (that is,

ATMEL

209



all single conversions, and the first free running conversion) when CK_{ADC2} is low will take the same amount of time as a single ended conversion (13 ADC clock cycles from the next prescaled clock cycle). A conversion initiated by the user when CK_{ADC2} is high will take 14 ADC clock cycles due to the synchronization mechanism. In Free Running mode, a new conversion is initiated immediately after the previous conversion completes, and since CK_{ADC2} is high at this time, all automatically started (that is, all but the first) free running conversions will take 14 ADC clock cycles.

The gain stage is optimized for a bandwidth of 4 kHz at all gain settings. Higher frequencies may be subjected to non-linear amplification. An external low-pass filter should be used if the input signal contains higher frequency components than the gain stage bandwidth. Note that the ADC clock frequency is independent of the gain stage bandwidth limitation. For example, the ADC clock period may be 6 μ s, allowing a channel to be sampled at 12 kSPS, regardless of the bandwidth of this channel.

If differential gain channels are used and conversions are started by Auto Triggering, the ADC must be switched off between conversions. When Auto Triggering is used, the ADC prescaler is reset before the conversion is started. Since the gain stage is dependent of a stable ADC clock prior to the conversion, this conversion will not be valid. By disabling and then re-enabling the ADC between each conversion (writing ADEN in ADCSRA to "0" then to "1"), only extended conversions are performed. The result from the extended conversions will be valid. See "Prescaling and Conversion Timing" on page 207 for timing details.

Changing Channel or Reference Selection

The MUXn and REFS1:0 bits in the ADMUX Register are single buffered through a temporary register to which the CPU has random access. This ensures that the channels and reference selection only takes place at a safe point during the conversion. The channel and reference selection is continuously updated until a conversion is started. Once the conversion starts, the channel and reference selection is locked to ensure a sufficient sampling time for the ADC. Continuous updating resumes in the last ADC clock cycle before the conversion completes (ADIF in ADCSRA is set). Note that the conversion starts on the following rising ADC clock edge after ADSC is written. The user is thus advised not to write new channel or reference selection values to ADMUX until one ADC clock cycle after ADSC is written.

If Auto Triggering is used, the exact time of the triggering event can be indeterministic. Special care must be taken when updating the ADMUX Register, in order to control which conversion will be affected by the new settings.

If both ADATE and ADEN is written to one, an interrupt event can occur at any time. If the ADMUX Register is changed in this period, the user cannot tell if the next conversion is based on the old or the new settings. ADMUX can be safely updated in the following ways:

- 1. When ADATE or ADEN is cleared.
- 2. During conversion, minimum one ADC clock cycle after the trigger event.
- 3. After a conversion, before the Interrupt Flag used as trigger source is cleared.

When updating ADMUX in one of these conditions, the new settings will affect the next ADC conversion.

Special care should be taken when changing differential channels. Once a differential channel has been selected, the gain stage may take as much as 125 µs to stabilize to the new value. Thus conversions should not be started within the first 125 µs after selecting a new differential channel. Alternatively, conversion results obtained within this period should be discarded.

The same settling time should be observed for the first differential conversion after changing ADC reference (by changing the REFS1:0 bits in ADMUX).



210



ADC Input Channels

When changing channel selections, the user should observe the following guidelines to ensure that the correct channel is selected:

In Single Conversion mode, always select the channel before starting the conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the conversion to complete before changing the channel selection.

In Free Running mode, always select the channel before starting the first conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the first conversion to complete, and then change the channel selection. Since the next conversion has already started automatically, the next result will reflect the previous channel selection. Subsequent conversions will reflect the new channel selection.

When switching to a differential gain channel, the first conversion result may have a poor accuracy due to the required settling time for the automatic offset cancellation circuitry. The user should preferably disregard the first conversion result.

ADC Voltage Reference

The reference voltage for the ADC (V_{REF}) indicates the conversion range for the ADC. Single ended channels that exceed V_{REF} will result in codes close to 0x3FF. V_{REF} can be selected as either AVCC, internal 2.56V reference, or external AREF pin.

AVCC is connected to the ADC through a passive switch. The internal 2.56V reference is generated from the internal bandgap reference (V_{BG}) through an internal amplifier. In either case, the external AREF pin is directly connected to the ADC, and the reference voltage can be made more immune to noise by connecting a capacitor between the AREF pin and ground. V_{REF} can also be measured at the AREF pin with a high impedant voltmeter. Note that V_{REF} is a high impedant source, and only a capacitive load should be connected in a system.

If the user has a fixed voltage source connected to the AREF pin, the user may not use the other reference voltage options in the application, as they will be shorted to the external voltage. If no external voltage is applied to the AREF pin, the user may switch between AVCC and 2.56V as reference selection. The first ADC conversion result after switching reference voltage source may be inaccurate, and the user is advised to discard this result.

If differential channels are used, the selected reference should not be closer to AVCC than indicated in Table 122 on page 297.

ADC Noise Canceler

The ADC features a noise canceler that enables conversion during sleep mode to reduce noise induced from the CPU core and other I/O peripherals. The noise canceler can be used with ADC Noise Reduction and Idle mode. To make use of this feature, the following procedure should be used:

- Make sure that the ADC is enabled and is not busy converting. Single Conversion Mode must be selected and the ADC conversion complete interrupt must be enabled.
- 2. Enter ADC Noise Reduction mode (or Idle mode). The ADC will start a conversion once the CPU has been halted.
- 3. If no other interrupts occur before the ADC conversion completes, the ADC interrupt will wake up the CPU and execute the ADC Conversion Complete interrupt routine. If another interrupt wakes up the CPU before the ADC conversion is complete, that interrupt will be executed, and an ADC Conversion Complete interrupt request will be generated when the ADC conversion completes. The CPU will remain in active mode until a new sleep command is executed.

Note that the ADC will not be automatically turned off when entering other sleep modes than Idle mode and ADC Noise Reduction mode. The user is advised to write zero to ADEN before entering such sleep modes to avoid excessive power consumption. If the ADC is enabled in such



211



sleep modes and the user wants to perform differential conversions, the user is advised to switch the ADC off and on after waking up from sleep to prompt an extended conversion to get a valid result.

Analog Input Circuitry

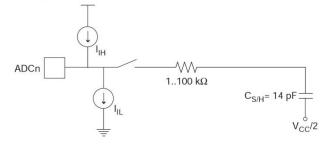
The Analog Input Circuitry for single ended channels is illustrated in Figure 105. An analog source applied to ADCn is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is selected as input for the ADC. When the channel is selected, the source must drive the S/H capacitor through the series resistance (combined resistance in the input path).

The ADC is optimized for analog signals with an output impedance of approximately 10 k Ω or less. If such a source is used, the sampling time will be negligible. If a source with higher impedance is used, the sampling time will depend on how long time the source needs to charge the S/H capacitor, with can vary widely. The user is recommended to only use low impedant sources with slowly varying signals, since this minimizes the required charge transfer to the S/H capacitor.

If differential gain channels are used, the input circuitry looks somewhat different, although source impedances of a few hundred $k\Omega$ or less is recommended.

Signal components higher than the Nyquist frequency $(f_{ADC}/2)$ should not be present for either kind of channels, to avoid distortion from unpredictable signal convolution. The user is advised to remove high frequency components with a low-pass filter before applying the signals as inputs to the ADC.

Figure 105. Analog Input Circuitry



Analog Noise Canceling Techniques

Digital circuitry inside and outside the device generates EMI which might affect the accuracy of analog measurements. If conversion accuracy is critical, the noise level can be reduced by applying the following techniques:

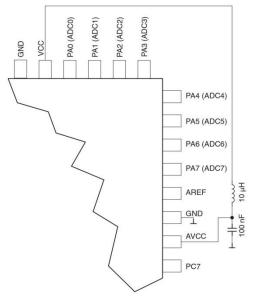
- Keep analog signal paths as short as possible. Keep them well away from highspeed switching digital tracks.
- The AVCC pin on the device should be connected to the digital V_{CC} supply voltage via an LC network as shown in Figure 106.
- 3. Use the ADC noise canceler function to reduce induced noise from the CPU.
- If any ADC port pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.



212



Figure 106. ADC Power Connections



Offset Compensation Schemes The gain stage has a built-in offset cancellation circuitry that nulls the offset of differential measurements as much as possible. The remaining offset in the analog path can be measured directly by selecting the same channel for both differential inputs. This offset residue can be then subtracted in software from the measurement results. Using this kind of software based offset correction, offset on any channel can be reduced below one LSB.

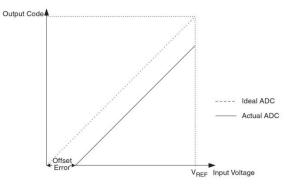
ADC Accuracy Definitions An n-bit single-ended ADC converts a voltage linearly between GND and V_{REF} in 2^n steps (LSBs). The lowest code is read as 0, and the highest code is read as 2^n -1.

Several parameters describe the deviation from the ideal behavior:

 Offset: The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB.

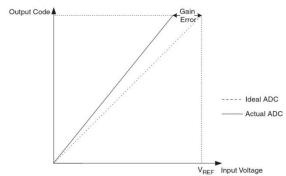


Figure 107. Offset Error



 Gain Error: After adjusting for offset, the Gain Error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB

Figure 108. Gain Error

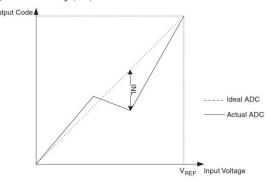


 Integral Non-linearity (INL): After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSB.



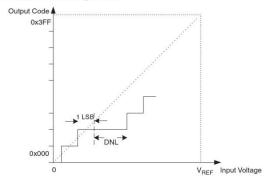


Figure 109. Integral Non-linearity (INL)



 Differential Non-linearity (DNL): The maximum deviation of the actual code width (the interval between two adjacent transitions) from the ideal code width (1 LSB). Ideal value: 0 LSB.

Figure 110. Differential Non-linearity (DNL)



- Quantization Error: Due to the quantization of the input voltage into a finite number of codes, a range of input voltages (1 LSB wide) will code to the same value. Always ±0.5 LSB.
- Absolute Accuracy: The maximum deviation of an actual (unadjusted) transition compared to an ideal transition for any code. This is the compound effect of Offset, Gain Error, Differential Error, Non-linearity, and Quantization Error. Ideal value: ±0.5 LSB.



215



ADC Conversion Result

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC Result Registers (ADCL, ADCH).

For single ended conversion, the result is

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

where V_{IN} is the voltage on the selected input pin and V_{REF} the selected voltage reference (see Table 83 on page 217 and Table 84 on page 218). 0x000 represents ground, and 0x3FF represents the selected reference voltage minus one LSB.

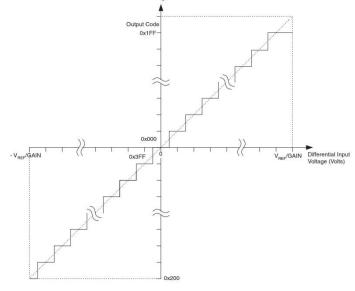
If differential channels are used, the result is

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot GAIN \cdot 512}{V_{REF}}$$

where V_{POS} is the voltage on the positive input pin, V_{NEG} the voltage on the negative input pin, GAIN the selected gain factor, and V_{REF} the selected voltage reference. The result is presented in two's complement form, from 0x200 (-512d) through 0x1FF (+511d). Note that if the user wants to perform a quick polarity check of the results, it is sufficient to read the MSB of the result (ADC9 in ADCH). If this bit is one, the result is negative, and if this bit is zero, the result is positive. Figure 111 shows the decoding of the differential input range.

Table 82 shows the resulting output codes if the differential input channel pair (ADCn - ADCm) is selected with a gain of GAIN and a reference voltage of $V_{\rm RFF}$.

Figure 111. Differential Measurement Range



216



Table 82. Correlation between Input Voltage and Output Codes

V _{ADCn}	Read code	Corresponding Decimal Value
V _{ADCm} + V _{REF} /GAIN	0x1FF	511
V _{ADCm} + 511/512 V _{REF} /GAIN	0x1FF	511
V _{ADCm} + 510/512 V _{REF} /GAIN	0x1FE	510
V _{ADCm} + 1/512 V _{REF} /GAIN	0x001	1
V _{ADCm}	0x000	0
V _{ADCm} - 1/512 V _{REF} /GAIN	0x3FF	-1
V _{ADCm} - 511/512 V _{REF} /GAIN	0x201	-511
V _{ADCm} - V _{REF} /GAIN	0x200	-512

Example:

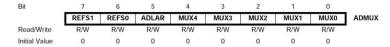
ADMUX = 0xED (ADC3 - ADC2, 10x gain, 2.56V reference, left adjusted result)

Voltage on ADC3 is 300 mV, voltage on ADC2 is 500 mV.

 $ADCR = 512 \times 10 \times (300 - 500) / 2560 = -400 = 0x270$

ADCL will thus read 0x00, and ADCH will read 0x9C. Writing zero to ADLAR right adjusts the result: ADCL = 0x70, ADCH = 0x02.

ADC Multiplexer Selection Register – ADMUX



· Bit 7:6 - REFS1:0: Reference Selection Bits

These bits select the voltage reference for the ADC, as shown in Table 83. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 83. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

· Bit 5 - ADLAR: ADC Left Adjust Result

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conver-



217



sions. For a complete description of this bit, see "The ADC Data Register – ADCL and ADCH" on page 220.

• Bits 4:0 - MUX4:0: Analog Channel and Gain Selection Bits

The value of these bits selects which combination of analog inputs are connected to the ADC. These bits also select the gain for the differential channels. See Table 84 for details. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set).

Table 84. Input Channel and Gain Selections

MUX40	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
00000	ADC0			
00001	ADC1	_		
00010	ADC2			
00011	ADC3	N/A		
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000		ADC0	ADC0	10x
01001		ADC1	ADC0	10x
01010		ADC0	ADC0	200x
01011		ADC1	ADC0	200>
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	10x
01110		ADC2	ADC2	200>
01111		ADC3	ADC2	200x
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	1x
10010	N/A	ADC2	ADC1	1x
10011		ADC3	ADC1	1x
10100		ADC4	ADC1	1x
10101		ADC5	ADC1	1x
10110		ADC6	ADC1	1x
10111		ADC7	ADC1	1x
11000		ADC0	ADC2	1x
11001		ADC1	ADC2	1x
11010		ADC2	ADC2	1x
11011		ADC3	ADC2	1x
11100		ADC4	ADC2	1x



218



Table 84. Input Channel and Gain Selections (Continued)

MUX40	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
11101		ADC5	ADC2	1x
11110	1.22V (V _{BG})	N/A	·	*
11111	0 V (GND)			

ADC Control and Status Register A – ADCSRA

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 - ADEN: ADC Enable

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

· Bit 6 - ADSC: ADC Start Conversion

In Single Conversion mode, write this bit to one to start each conversion. In Free Running Mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

• Bit 5 - ADATE: ADC Auto Trigger Enable

When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in SFIOR.

· Bit 4 - ADIF: ADC Interrupt Flag

This bit is set when an ADC conversion completes and the Data Registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

• Bit 3 - ADIE: ADC Interrupt Enable

When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

• Bits 2:0 - ADPS2:0: ADC Prescaler Select Bits

These bits determine the division factor between the XTAL frequency and the input clock to the ADC.



219



Table 85. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

The ADC Data Register – ADCL and ADCH

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
	-		-	_	12		ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	-	-	1-	-	-	-	ADCL
	7	6	5	4	3	2	1	0	5
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When an ADC conversion is complete, the result is found in these two registers. If differential channels are used, the result is presented in two's complement form.

When ADCL is read, the ADC Data Register is not updated until ADCH is read. Consequently, if the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH.

The ADLAR bit in ADMUX, and the MUXn bits in ADMUX affect the way the result is read from the registers. If ADLAR is set, the result is left adjusted. If ADLAR is cleared (default), the result is right adjusted.

· ADC9:0: ADC Conversion Result

These bits represent the result from the conversion, as detailed in "ADC Conversion Result" on page 216.

<u>AIMEL</u>

220



Special FunctionIO Register – SFIOR

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	-	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	Ţ.
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7:5 - ADTS2:0: ADC Auto Trigger Source

If ADATE in ADCSRA is written to one, the value of these bits selects which source will trigger an ADC conversion. If ADATE is cleared, the ADTS2:0 settings will have no effect. A conversion will be triggered by the rising edge of the selected Interrupt Flag. Note that switching from a trigger source that is cleared to a trigger source that is set, will generate a positive edge on the trigger signal. If ADEN in ADCSRA is set, this will start a conversion. Switching to Free Running mode (ADTS[2:0]=0) will not cause a trigger event, even if the ADC Interrupt Flag is set.

Table 86. ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

· Bit 4 - Res: Reserved Bit

This bit is reserved for future use. To ensure compatibility with future devices, this bit must be written to zero when SFIOR is written.





_____ ATmega16(L)

JTAG Interface and On-chip Debug System

Features

- · JTAG (IEEE std. 1149.1 Compliant) Interface
- Boundary-scan Capabilities According to the IEEE std. 1149.1 (JTAG) Standard
- · Debugger Access to:
 - All Internal Peripheral Units
 - Internal and External RAM
 - The Internal Register File
 - Program Counter
 - EEPROM and Flash Memories
 - Extensive On-chip Debug Support for Break Conditions, Including
 - AVR Break Instruction
 - Break on Change of Program Memory Flow
 - Single Step Break
 - Program Memory Breakpoints on Single Address or Address Range
 - Data Memory Breakpoints on Single Address or Address Range
- Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- On-chip Debugging Supported by AVR Studio[®]

Overview

The AVR IEEE std. 1149.1 compliant JTAG interface can be used for

- · Testing PCBs by using the JTAG Boundary-scan capability
- · Programming the non-volatile memories, Fuses and Lock bits
- · On-chip Debugging

A brief description is given in the following sections. Detailed descriptions for Programming via the JTAG interface, and using the Boundary-scan Chain can be found in the sections "Programming via the JTAG Interface" on page 278 and "IEEE 1149.1 (JTAG) Boundary-scan" on page 228, respectively. The On-chip Debug support is considered being private JTAG instructions, and distributed within ATMEL and to selected third party vendors only.

Figure 112 shows a block diagram of the JTAG interface and the On-chip Debug system. The TAP Controller is a state machine controlled by the TCK and TMS signals. The TAP Controller selects either the JTAG Instruction Register or one of several Data Registers as the scan chain (Shift Register) between the TDI input and TDO output. The Instruction Register holds JTAG instructions controlling the behavior of a Data Register.

The ID-Register, Bypass Register, and the Boundary-scan Chain are the Data Registers used for board-level testing. The JTAG Programming Interface (actually consisting of several physical and virtual Data Registers) is used for JTAG Serial Programming via the JTAG interface. The Internal Scan Chain and Break Point Scan Chain are used for On-chip Debugging only.

Test Access Port – TAP

The JTAG interface is accessed through four of the AVR's pins. In JTAG terminology, these pins constitute the Test Access Port – TAP. These pins are:

- TMS: Test Mode Select. This pin is used for navigating through the TAP-controller state machine.
- TCK: Test Clock. JTAG operation is synchronous to TCK.
- TDI: Test Data In. Serial input data to be shifted in to the Instruction Register or Data Register (Scan Chains).
- TDO: Test Data Out. Serial output data from Instruction register or Data Register.



222

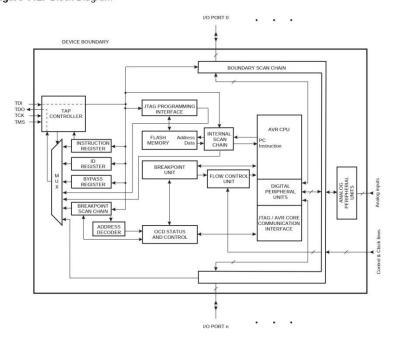


The IEEE std. 1149.1 also specifies an optional TAP signal; TRST – Test ReSeT – which is not provided.

When the JTAGEN fuse is unprogrammed, these four TAP pins are normal port pins and the TAP controller is in reset. When programmed and the JTD bit in MCUCSR is cleared, the TAP input signals are internally pulled high and the JTAG is enabled for Boundary-scan and programming. In this case, the TAP output pin (TDO) is left floating in states where the JTAG TAP controller is not shifting data, and must therefore be connected to a pull-up resistor or other hardware having pull-ups (for instance the TDI-input of the next device in the scan chain). The device is shipped with this fuse programmed.

For the On-chip Debug system, in addition to the JTAG interface pins, the $\overline{\text{RESET}}$ pin is monitored by the debugger to be able to detect external reset sources. The debugger can also pull the $\overline{\text{RESET}}$ pin low to reset the whole system, assuming only open collectors on the reset line are used in the application.

Figure 112. Block Diagram





223



Test-Logic-Reset 0 Run-Test/Idle Select-DR Scan Select-IR Scan 0 0 Capture-DR Capture-IR 0 0 Shift-DR 0 Shift-IR Exit1-DR Exit1-IR 0 0 Pause-DR Pause-IR Exit2-DR Exit2-IR 1 Update-DR Update-IR 0 0

Figure 113. TAP Controller State Diagram

TAP Controller

The TAP controller is a 16-state finite state machine that controls the operation of the Boundary-scan circuitry, JTAG programming circuitry, or On-chip Debug system. The state transitions depicted in Figure 113 depend on the signal present on TMS (shown adjacent to each state transition) at the time of the rising edge at TCK. The initial state after a Power-On Reset is Test-Logic-Reset.

As a definition in this document, the LSB is shifted in and out first for all Shift Registers.

Assuming Run-Test/Idle is the present state, a typical scenario for using the JTAG interface is:

 At the TMS input, apply the sequence 1, 1, 0, 0 at the rising edges of TCK to enter the Shift Instruction Register – Shift-IR state. While in this state, shift the four bits of the JTAG instructions into the JTAG Instruction Register from the TDI input at the rising edge of TCK. The TMS input must be held low during input of the 3 LSBs in order to remain in the Shift-IR state. The MSB of the instruction is shifted in when this state is left by setting TMS high. While the instruction is shifted in from the TDI pin, the captured IR-state 0x01 is shifted out

AIMEL

224



on the TDO pin. The JTAG Instruction selects a particular Data Register as path between TDI and TDO and controls the circuitry surrounding the selected Data Register.

- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. The instruction is latched onto the parallel output from the Shift Register path in the Update-IR state. The Exit-IR, Pause-IR, and Exit2-IR states are only used for navigating the state machine.
- At the TMS input, apply the sequence 1, 0, 0 at the rising edges of TCK to enter the Shift Data Register Shift-DR state. While in this state, upload the selected Data Register (selected by the present JTAG instruction in the JTAG Instruction Register) from the TDI input at the rising edge of TCK. In order to remain in the Shift-DR state, the TMS input must be held low during input of all bits except the MSB. The MSB of the data is shifted in when this state is left by setting TMS high. While the Data Register is shifted in from the TDI pin, the parallel inputs to the Data Register captured in the Capture-DR state is shifted out on the TDO pin.
- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. If the selected Data Register has a latched parallel-output, the latching takes place in the Update-DR state. The Exit-DR, Pause-DR, and Exit2-DR states are only used for navigating the state machine.

As shown in the state diagram, the Run-Test/Idle state need not be entered between selecting JTAG instruction and using Data Registers, and some JTAG instructions may select certain functions to be performed in the Run-Test/Idle, making it unsuitable as an Idle state.

Note: Independent of the initial state of the TAP Controller, the Test-Logic-Reset state can always be entered by holding TMS high for five TCK clock periods.

For detailed information on the JTAG specification, refer to the literature listed in "Bibliography" on page 227.

Using the Boundary-scan Chain

A complete description of the Boundary-scan capabilities are given in the section "IEEE 1149.1 (JTAG) Boundary-scan" on page 228.

Using the On-chip Debug System

As shown in Figure 112, the hardware support for On-chip Debugging consists mainly of:

- A scan chain on the interface between the internal AVR CPU and the internal peripheral units
- · Break Point unit
- Communication interface between the CPU and JTAG system

All read or modify/write operations needed for implementing the Debugger are done by applying AVR instructions via the internal AVR CPU Scan Chain. The CPU sends the result to an I/O memory mapped location which is part of the communication interface between the CPU and the JTAG system.

The Break Point Unit implements Break on Change of Program Flow, Single Step Break, 2 Program Memory Break Points, and 2 combined Break Points. Together, the 4 Break Points can be configured as either:

- 4 single Program Memory Break Points
- 3 Single Program Memory Break Point + 1 single Data Memory Break Point
- 2 single Program Memory Break Points + 2 single Data Memory Break Points
- 2 single Program Memory Break Points + 1 Program Memory Break Point with mask ("range Break Point")
- 2 single Program Memory Break Points + 1 Data Memory Break Point with mask ("range Break Point")



225



A debugger, like the AVR Studio, may however use one or more of these resources for its internal purpose, leaving less flexibility to the end-user.

A list of the On-chip Debug specific JTAG instructions is given in "On-chip Debug Specific JTAG Instructions" on page 226.

The JTAGEN Fuse must be programmed to enable the JTAG Test Access Port. In addition, the OCDEN Fuse must be programmed and no Lock bits must be set for the On-chip Debug system to work. As a security feature, the On-chip Debug system is disabled when *any* Lock bits are set. Otherwise, the On-chip Debug system would have provided a back-door into a secured device.

The AVR JTAG ICE from Atmel is a powerful development tool for On-chip Debugging of all AVR 8-bit RISC Microcontrollers with IEEE 1149.1 compliant JTAG interface. The JTAG ICE and the AVR Studio user interface give the user complete control of the internal resources of the microcontroller, helping to reduce development time by making debugging easier. The JTAG ICE performs real-time emulation of the microcontroller while it is running in a target system.

Please refer to the Support Tools section on the AVR pages on www.atmel.com for a full description of the AVR JTEG ICE. AVR Studio can be downloaded free from Software section on the same web site.

All necessary execution commands are available in AVR Studio, both on source level and on disassembly level. The user can execute the program, single step through the code either by tracing into or stepping over functions, step out of functions, place the cursor on a statement and execute until the statement is reached, stop the execution, and reset the execution target. In addition, the user can have an unlimited number of code breakpoints (using the BREAK instruction) and up to two data memory breakpoints, alternatively combined as a mask (range) Break Point.

On-chip Debug
Specific JTAG
Instructions

The On-chip Debug support is considered being private JTAG instructions, and distributed within ATMEL and to selected third party vendors only. Instruction opcodes are listed for reference.

PRIVATEO; \$8 Private JTAG instruction for accessing On-chip Debug system.

PRIVATE1; \$9 Private JTAG instruction for accessing On-chip Debug system.

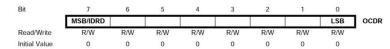
PRIVATE2; \$A Private JTAG instruction for accessing On-chip Debug system.

PRIVATE3; \$B Private JTAG instruction for accessing On-chip Debug system.



On-chip Debug Related Register in I/O Memory

On-chip Debug Register – OCDR



The OCDR Register provides a communication channel from the running program in the micro-controller to the debugger. The CPU can transfer a byte to the debugger by writing to this location. At the same time, an Internal Flag; I/O Debug Register Dirty – IDRD – is set to indicate to the debugger that the register has been written. When the CPU reads the OCDR Register the 7 LSB will be from the OCDR Register, while the MSB is the IDRD bit. The debugger clears the IDRD bit when it has read the information.

In some AVR devices, this register is shared with a standard I/O location. In this case, the OCDR Register can only be accessed if the OCDEN Fuse is programmed, and the debugger enables access to the OCDR Register. In all other cases, the standard I/O location is accessed.

Refer to the debugger documentation for further information on how to use this register.

Using the JTAG Programming Capabilities

Programming of AVR parts via JTAG is performed via the 4-pin JTAG port, TCK, TMS, TDI and TDO. These are the only pins that need to be controlled/observed to perform JTAG programming (in addition to power pins). It is not required to apply 12V externally. The JTAGEN Fuse must be programmed and the JTD bit in the MCUSR Register must be cleared to enable the JTAG Test Access Port.

The JTAG programming capability supports:

- · Flash programming and verifying
- · EEPROM programming and verifying
- Fuse programming and verifying
- · Lock bit programming and verifying

The Lock bit security is exactly as in Parallel Programming mode. If the Lock bits LB1 or LB2 are programmed, the OCDEN Fuse cannot be programmed unless first doing a chip erase. This is a security feature that ensures no back-door exists for reading out the content of a secured device.

The details on programming through the JTAG interface and programming specific JTAG instructions are given in the section "Programming via the JTAG Interface" on page 278.

Bibliography

For more information about general Boundary-scan, the following literature can be consulted:

- IEEE: IEEE Std. 1149.1-1990. IEEE Standard Test Access Port and Boundary-scan Architecture, IEEE, 1993
- Colin Maunder: The Board Designers Guide to Testable Logic Circuits, Addison-Wesley, 1992



227



IEEE 1149.1 (JTAG) Boundary-scan

Features

- · JTAG (IEEE std. 1149.1 Compliant) Interface
- Boundary-scan Capabilities According to the JTAG Standard
- · Full Scan of all Port Functions as well as Analog Circuitry having Off-chip Connections
- Supports the Optional IDCODE Instruction
- Additional Public AVR_RESET Instruction to Reset the AVR

System Overview

The Boundary-scan chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having Off-chip connections. At system level, all ICs having JTAG capabilities are connected serially by the TDI/TDO signals to form a long Shift Register. An external controller sets up the devices to drive values at their output pins, and observe the input values received from other devices. The controller compares the received data with the expected result. In this way, Boundary-scan provides a mechanism for testing interconnections and integrity of components on Printed Circuits Boards by using the four TAP signals only.

The four IEEE 1149.1 defined mandatory JTAG instructions IDCODE, BYPASS, SAMPLE/PRELOAD, and EXTEST, as well as the AVR specific public JTAG instruction AVR_RESET can be used for testing the Printed Circuit Board. Initial scanning of the Data Register path will show the ID-code of the device, since IDCODE is the default JTAG instruction. It may be desirable to have the AVR device in Reset during Test mode. If not reset, inputs to the device may be determined by the scan operations, and the internal software may be in an undetermined state when exiting the Test mode. Entering reset, the outputs of any Port Pin will instantly enter the high impedance state, making the HIGHZ instruction redundant. If needed, the BYPASS instruction can be issued to make the shortest possible scan chain through the device. The device can be set in the reset state either by pulling the external RESET pin low, or issuing the AVR_RESET instruction with appropriate setting of the Reset Data Register.

The EXTEST instruction is used for sampling external pins and loading output pins with data. The data from the output latch will be driven out on the pins as soon as the EXTEST instruction is loaded into the JTAG IR-Register. Therefore, the SAMPLE/PRELOAD should also be used for setting initial values to the scan ring, to avoid damaging the board when issuing the EXTEST instruction for the first time. SAMPLE/PRELOAD can also be used for taking a snapshot of the external pins during normal operation of the part.

The JTAGEN Fuse must be programmed and the JTD bit in the I/O Register MCUCSR must be cleared to enable the JTAG Test Access Port.

When using the JTAG interface for Boundary-scan, using a JTAG TCK clock frequency higher than the internal chip frequency is possible. The chip clock is not required to run.

Data Registers

The Data Registers relevant for Boundary-scan operations are:

- Bypass Register
- Device Identification Register
- · Reset Register
- · Boundary-scan Chain

Bypass Register

The Bypass Register consists of a single Shift Register stage. When the Bypass Register is selected as path between TDI and TDO, the register is reset to 0 when leaving the Capture-DR



228



controller state. The Bypass Register can be used to shorten the scan chain on a system when the other devices are to be tested.

Device Identification Register

Figure 114 shows the structure of the Device Identification Register.

Figure 114. The Format of the Device Identification Register

	MSB						LSB
Bit	31	28	27	12	11	1	0
Device ID	Vers	sion	Part N	umber	Manufa	cturer ID	1
	4 h	its	16	hits	11	hits	1 bit

Version

Version is a 4-bit number identifying the revision of the component. The JTAG version number follows the revision of the device. Revision A is 0x0, revision B is 0x1 and so on. However, some revisions deviate from this rule, and the relevant version number is shown in Table 87.

Table 87. JTAG Version Numbers

Version	JTAG Version Number (Hex)
ATmega16 revision G	0x6
ATmega16 revision H	0xE
ATmega16 revision I	0x8
ATmega16 revision J	0x9
ATmega16 revision K	0xA
ATmega16 revision L	0xB

Part Number

The part number is a 16-bit code identifying the component. The JTAG Part Number for ATmega16 is listed in Table 88.

Table 88. AVR JTAG Part Number

Part Number	JTAG Part Number (Hex)		
ATmega16	0x9403		

Manufacturer ID

The Manufacturer ID is a 11 bit code identifying the manufacturer. The JTAG manufacturer ID for ATMEL is listed in Table 89.

Table 89. Manufacturer ID

Manufacturer	JTAG Manufacturer ID (Hex)		
ATMEL	0x01F		

Reset Register

The Reset Register is a Test Data Register used to reset the part. Since the AVR tri-states Port Pins when reset, the Reset Register can also replace the function of the unimplemented optional JTAG instruction HIGHZ.

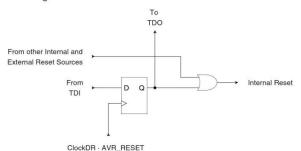
A high value in the Reset Register corresponds to pulling the External Reset low. The part is reset as long as there is a high value present in the Reset Register. Depending on the Fuse settings for the clock options, the part will remain reset for a Reset Time-Out Period (refer to "Clock Sources" on page 25) after releasing the Reset Register. The output from this Data Register is not latched, so the reset will take place immediately, as shown in Figure 115.

AIMEL

229



Figure 115. Reset Register



Boundary-scan Chain

The Boundary-scan Chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having Off-chip connections.

See "Boundary-scan Chain" on page 232 for a complete description.

Boundary-scan Specific JTAG Instructions

The instruction register is 4-bit wide, supporting up to 16 instructions. Listed below are the JTAG instructions useful for Boundary-scan operation. Note that the optional HIGHZ instruction is not implemented, but all outputs with tri-state capability can be set in high-impedant state by using the AVR_RESET instruction, since the initial state for all port pins is tri-state.

As a definition in this datasheet, the LSB is shifted in and out first for all Shift Registers.

The OPCODE for each instruction is shown behind the instruction name in hex format. The text describes which Data Register is selected as path between TDI and TDO for each instruction.

EXTEST; \$0

Mandatory JTAG instruction for selecting the Boundary-scan Chain as Data Register for testing circuitry external to the AVR package. For port-pins, Pull-up Disable, Output Control, Output Data, and Input Data are all accessible in the scan chain. For Analog circuits having Off-chip connections, the interface between the analog and the digital logic is in the scan chain. The contents of the latched outputs of the Boundary-scan chain is driven out as soon as the JTAG IR-register is loaded with the EXTEST instruction.

The active states are:

- Capture-DR: Data on the external pins are sampled into the Boundary-scan Chain.
- · Shift-DR: The Internal Scan Chain is shifted by the TCK input.
- · Update-DR: Data from the scan chain is applied to output pins.

IDCODE; \$1

Optional JTAG instruction selecting the 32-bit ID-register as Data Register. The ID-register consists of a version number, a device number and the manufacturer code chosen by JEDEC. This is the default instruction after power-up.

The active states are:

- · Capture-DR: Data in the IDCODE-register is sampled into the Boundary-scan Chain.
- · Shift-DR: The IDCODE scan chain is shifted by the TCK input.

SAMPLE_PRELOAD; \$2

Mandatory JTAG instruction for pre-loading the output latches and talking a snap-shot of the input/output pins without affecting the system operation. However, the output latches are not connected to the pins. The Boundary-scan Chain is selected as Data Register.



230



The active states are:

- · Capture-DR: Data on the external pins are sampled into the Boundary-scan Chain.
- · Shift-DR: The Boundary-scan Chain is shifted by the TCK input.
- Update-DR: Data from the Boundary-scan Chain is applied to the output latches. However, the output latches are not connected to the pins.

AVR_RESET; \$C

The AVR specific public JTAG instruction for forcing the AVR device into the Reset mode or releasing the JTAG Reset source. The TAP controller is not reset by this instruction. The one bit Reset Register is selected as Data Register. Note that the reset will be active as long as there is a logic 'one' in the Reset Chain. The output from this chain is not latched.

The active states are:

Shift-DR: The Reset Register is shifted by the TCK input.

BYPASS; \$F

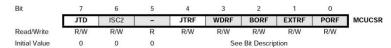
Mandatory JTAG instruction selecting the Bypass Register for Data Register.

The active states are:

- · Capture-DR: Loads a logic "0" into the Bypass Register.
- · Shift-DR: The Bypass Register cell between TDI and TDO is shifted.

Boundary-scan Related Register in I/O Memory

MCU Control and Status Register – MCUCSR The MCU Control and Status Register contains control bits for general MCU functions, and provides information on which reset source caused an MCU Reset.



• Bit 7 - JTD: JTAG Interface Disable

When this bit is zero, the JTAG interface is enabled if the JTAGEN Fuse is programmed. If this bit is one, the JTAG interface is disabled. In order to avoid unintentional disabling or enabling of the JTAG interface, a timed sequence must be followed when changing this bit: The application software must write this bit to the desired value twice within four cycles to change its value.

If the JTAG interface is left unconnected to other JTAG circuitry, the JTD bit should be set to one. The reason for this is to avoid static current at the TDO pin in the JTAG interface.

· Bit 4 - JTRF: JTAG Reset Flag

This bit is set if a reset is being caused by a logic one in the JTAG Reset Register selected by the JTAG instruction AVR_RESET. This bit is reset by a Power-on Reset, or by writing a logic zero to the flag.





Boundary-scan Chain

The Boundary-scan chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having Off-chip connection.

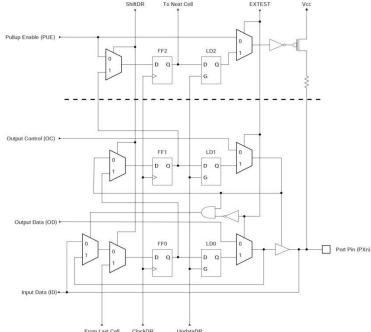
Scanning the Digital Port Pins Figure 116 shows the Boundary-scan Cell for a bi-directional port pin with pull-up function. The cell consists of a standard Boundary-scan cell for the Pull-up Enable – PUExn – function, and a bi-directional pin cell that combines the three signals Output Control – OCxn, Output Data – ODxn, and Input Data – IDxn, into only a two-stage Shift Register. The port and pin indexes are not used in the following description.

The Boundary-scan logic is not included in the figures in the datasheet. Figure 117 shows a simple digital Port Pin as described in the section "I/O Ports" on page 50. The Boundary-scan details from Figure 116 replaces the dashed box in Figure 117.

When no alternate port function is present, the Input Data – ID – corresponds to the PINxn Register value (but ID has no synchronizer), Output Data corresponds to the PORT Register, Output Control corresponds to the Data Direction – DD Register, and the Pull-up Enable – PUExn – corresponds to logic expression $\overline{\text{PUD}} \cdot \overline{\text{DDxn}} \cdot \text{PORTxn}$.

Digital alternate port functions are connected outside the dotted box in Figure 117 to make the scan chain read the actual pin value. For Analog function, there is a direct connection from the external pin to the analog circuit, and a scan chain is inserted on the interface between the digital logic and the analog circuitry.

Figure 116. Boundary-scan Cell for Bidirectional Port Pin with Pull-up Function.





232



PUD: PULLUP DISABLE PUEXI: PULLUP ENABLE for pin Pxn OCXn: OUTPUT CONTROL for pin Pxn WPx: READ DDRX WHITE PORTX REGISTER

Figure 117. General Port Pin Schematic Diagram⁽¹⁾

Note: 1. See Boundary-scan description for details.

Boundary-scan and the Two-wire Interface

The 2 Two-wire Interface pins SCL and SDA have one additional control signal in the scanchain; Two-wire Interface Enable – TWIEN. As shown in Figure 118, the TWIEN signal enables a tri-state buffer with slew-rate control in parallel with the ordinary digital port pins. A general scan cell as shown in Figure 122 is attached to the TWIEN signal.

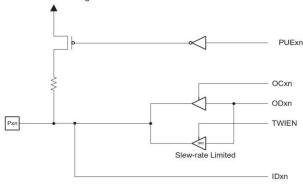
Notes: 1. A separate scan chain for the 50 ns spike filter on the input is not provided. The ordinary scan support for digital port pins suffice for connectivity tests. The only reason for having TWIEN in the scan path, is to be able to disconnect the slew-rate control buffer when doing boundary-scan.

Make sure the OC and TWIEN signals are not asserted simultaneously, as this will lead to drive contention.





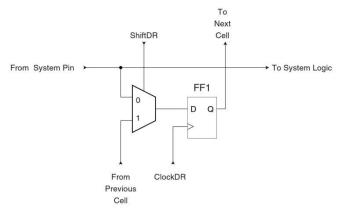
Figure 118. Additional Scan Signal for the Two-wire Interface



Scanning the RESET Pin

The RESET pin accepts 5V active low logic for standard reset operation, and 12V active high logic for High Voltage Parallel Programming. An observe-only cell as shown in Figure 119 is inserted both for the 5V reset signal; RSTT, and the 12V reset signal; RSTHV.

Figure 119. Observe-only Cell



Scanning the Clock Pins

The AVR devices have many clock options selectable by fuses. These are: Internal RC Oscillator, External RC, External Clock, (High Frequency) Crystal Oscillator, Low Frequency Crystal Oscillator, and Ceramic Resonator.

Figure 120 shows how each Oscillator with external connection is supported in the scan chain. The Enable signal is supported with a general boundary-scan cell, while the Oscillator/Clock output is attached to an observe-only cell. In addition to the main clock, the Timer Oscillator is scanned in the same way. The output from the internal RC Oscillator is not scanned, as this Oscillator does not have external connections.



234



Figure 120. Boundary-scan Cells for Oscillators and Clock Options

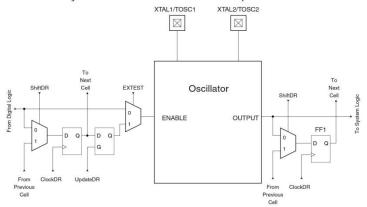


Table 90 summaries the scan registers for the external clock pin XTAL1, Oscillators with XTAL1/XTAL2 connections as well as 32 kHz Timer Oscillator.

Table 90. Scan Signals for the Oscillators (1)(2)(3)

Enable Signal	Scanned Clock Line	Clock Option	Scanned Clock Line when not Used
EXTCLKEN	EXTCLK (XTAL1)	External Clock	0
OSCON	OSCCK	External Crystal External Ceramic Resonator	0
RCOSCEN	RCCK	External RC	1
OSC32EN	OSC32CK	Low Freq. External Crystal	0
TOSKON	TOSCK	32 kHz Timer Oscillator	0

Notes: 1. Do not enable more than one clock source as main clock at a time.

- Scanning an Oscillator output gives unpredictable results as there is a frequency drift between the Internal Oscillator and the JTAG TCK clock. If possible, scanning an external clock is preferred.
- 3. The clock configuration is programmed by fuses. As a fuse is not changed run-time, the clock configuration is considered fixed for a given application. The user is advised to scan the same clock option as to be used in the final system. The enable signals are supported in the scan chain because the system logic can disable clock options in sleep modes, thereby disconnecting the Oscillator pins from the scan path if not provided. The INTCAP Fuses are not supported in the scan-chain, so the boundary scan chain can not make a XTAL Oscillator requiring internal capacitors to run unless the fuse is correctly programmed.

Scanning the Analog Comparator

The relevant Comparator signals regarding Boundary-scan are shown in Figure 121. The Boundary-scan cell from Figure 122 is attached to each of these signals. The signals are described in Table 91.

The Comparator need not be used for pure connectivity testing, since all analog inputs are shared with a digital port pin as well.



235



Figure 121. Analog Comparator

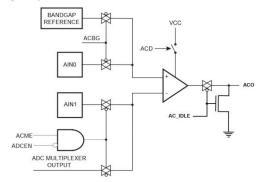


Figure 122. General Boundary-scan Cell used for Signals for Comparator and ADC

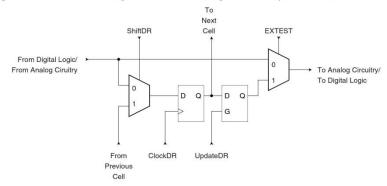






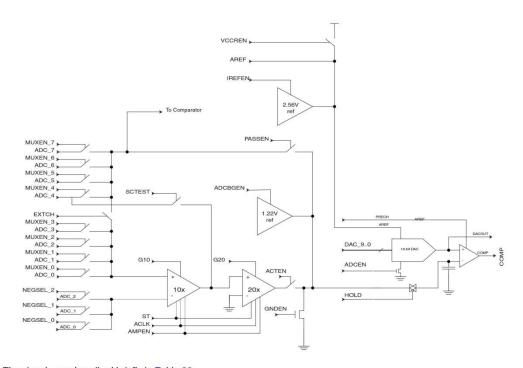
Table 91. Boundary-scan Signals for the Analog Comparator

Signal Name	Direction as Seen from the Comparator	Description	Recommended Input when Not in Use	Output Values when Recommended Inputs are Used
AC_IDLE	Input	Turns off Analog comparator when true	1	Depends upon μC code being executed
ACO	Output	Analog Comparator Output	Will become input to µC code being executed	0
ACME	Input	Uses output signal from ADC mux when true	0	Depends upon μC code being executed
ACBG	Input	Bandgap Reference enable	0	Depends upon μC code being executed

Scanning the ADC

Figure 123 shows a block diagram of the ADC with all relevant control and observe signals. The Boundary-scan cell from Figure 122 is attached to each of these signals. The ADC need not be used for pure connectivity testing, since all analog inputs are shared with a digital port pin as well.

Figure 123. Analog to Digital Converter



The signals are described briefly in Table 92.

AMEL

237



Table 92. Boundary-scan Signals for the ADC

Signal Name	Direction as Seen from the ADC Description		Recommended Input when Not in Use	Output Values when Recommended Inputs are used, and CPU is not Using the ADC
COMP	Output	Comparator Output	0	0
ACLK	Input	Clock signal to gain stages implemented as Switch-cap filters	0	0
ACTEN	Input	Enable path from gain stages to the comparator	0	0
ADCBGEN	Input	Enable Band-gap reference as negative input to comparator	0	0
ADCEN	Input	Power-on signal to the ADC	0	0
AMPEN	Input	Power-on signal to the gain stages	0	0
DAC_9	Input	Bit 9 of digital value to DAC	1	1
DAC_8	Input	Bit 8 of digital value to DAC	0	0
DAC_7	Input	Bit 7 of digital value to DAC	0	0
DAC_6	Input	Bit 6 of digital value to DAC	0	0
DAC_5	Input	Bit 5 of digital value to DAC	0	0
DAC_4	Input	Bit 4 of digital value to DAC	0	0
DAC_3	Input	Bit 3 of digital value to DAC	0	0
DAC_2	Input	Bit 2 of digital value to DAC	0	0
DAC_1	Input	Bit 1 of digital value to DAC	0	0
DAC_0	Input	Bit 0 of digital value to DAC	0	0
EXTCH	Input	Connect ADC channels 0 - 3 to by- pass path around gain stages	1	1
G10	Input	Enable 10x gain	0	0
G20	Input	Enable 20x gain	0	0
GNDEN	Input	Ground the negative input to comparator when true	0	0
HOLD	Input	Sample&Hold signal. Sample analog signal when low. Hold signal when high. If gain stages are used, this signal must go active when ACLK is high.	1	1
IREFEN	Input	Enables Band-gap reference as AREF signal to DAC	0	0
MUXEN_7	Input	Input Mux bit 7	0	0
MUXEN_6	Input	Input Mux bit 6	0	0
MUXEN_5	Input	Input Mux bit 5	0	0
MUXEN_4	Input	Input Mux bit 4	0	0
MUXEN_3	Input	Input Mux bit 3	0	0



238



Table 92. Boundary-scan Signals for the ADC (Continued)

Signal Name	Direction as Seen from the ADC	Description	Recommended Input when Not in Use	Output Values when Recommended Inputs are used, and CPU is not Using the ADC
MUXEN_2	Input	Input Mux bit 2	0	0
MUXEN_1	Input	Input Mux bit 1	0	0
MUXEN_0	Input	Input Mux bit 0	1	1
NEGSEL_2	Input	Input Mux for negative input for differential signal, bit 2	0	0
NEGSEL_1	Input	Input Mux for negative input for differential signal, bit 1	0	0
NEGSEL_0	Input	Input Mux for negative input for differential signal, bit 0	0	0
PASSEN	Input	Enable pass-gate of gain stages.	1	1
PRECH	Input	Precharge output latch of comparator. (Active low)	1	1
SCTEST	Input	Switch-cap TEST enable. Output from x10 gain stage send out to Port Pin having ADC_4	0	0
ST	Input	Output of gain stages will settle faster if this signal is high first two ACLK periods after AMPEN goes high.	0	0
VCCREN	Input	Selects Vcc as the ACC reference voltage.	0	0

Note: Incorrect setting of the switches in Figure 123 will make signal contention and may damage the part. There are several input choices to the S&H circuitry on the negative input of the output comparator in Figure 123. Make sure only one path is selected from either one ADC pin, Bandgap reference source, or Ground.

If the ADC is not to be used during scan, the recommended input values from Table 92 should be used. The user is recommended **not** to use the Differential Gain stages during scan. Switchcap based gain stages require fast operation and accurate timing which is difficult to obtain when used in a scan chain. Details concerning operations of the differential gain stage is therefore not provided.

The AVR ADC is based on the analog circuitry shown in Figure 123 with a successive approximation algorithm implemented in the digital logic. When used in Boundary-scan, the problem is usually to ensure that an applied analog voltage is measured within some limits. This can easily be done without running a successive approximation algorithm: apply the lower limit on the digital DAC[9:0] lines, make sure the output from the comparator is low, then apply the upper limit on the digital DAC[9:0] lines, and verify the output from the comparator to be high.

The ADC need not be used for pure connectivity testing, since all analog inputs are shared with a digital port pin as well.

When using the ADC, remember the following:

- The Port Pin for the ADC channel in use must be configured to be an input with pull-up disabled to avoid signal contention.
- In Normal mode, a dummy conversion (consisting of 10 comparisons) is performed when enabling the ADC. The user is advised to wait at least 200 ns after enabling the ADC before



239



controlling/observing any ADC signal, or perform a dummy conversion before using the first result.

 The DAC values must be stable at the midpoint value 0x200 when having the HOLD signal low (Sample mode).

As an example, consider the task of verifying a 1.5V $\pm 5\%$ input signal at ADC channel 3 when the power supply is 5.0V and AREF is externally connected to V_{CC} .

The recommended values from Table 92 are used unless other values are given in the algorithm in Table 93. Only the DAC and Port Pin values of the Scan-chain are shown. The column "Actions" describes what JTAG instruction to be used before filling the Boundary-scan Register with the succeeding columns. The verification should be done on the data scanned out when scanning in the data on the same row in the table.

Table 93. Algorithm for Using the ADC

Step	Actions	ADCEN	DAC	MUXEN	HOLD	PRECH	PA3. Data	PA3. Control	PA3. Pullup Enable
1	SAMPLE _PRELO AD	1	0x200	0x08	1	1	0	0	0
2	EXTEST	1	0x200	0x08	0	1	0	0	0
3		1	0x200	0x08	1	1	0	0	0
4		1	0x123	0x08	1	1	0	0	0
5		1	0x123	0x08	1	0	0	0	0
6	Verify the COMP bit scanned out to be 0	1	0x200	0x08	1	1	0	0	0
7		1	0x200	0x08	0	1	0	0	0
8		1	0x200	80x0	1	1	0	0	0
9		1	0x143	0x08	1	1	0	0	0
10		1	0x143	0x08	1	0	0	0	0
11	Verify the COMP bit scanned out to be 1	1	0x200	0x08	1	1	0	0	0

Using this algorithm, the timing constraint on the HOLD signal constrains the TCK clock frequency. As the algorithm keeps HOLD high for five steps, the TCK clock frequency has to be at least five times the number of scan bits divided by the maximum hold time, $t_{\text{hold,max}}$.





ATmega16 Boundary-scan Order Table 94 shows the scan order between TDI and TDO when the Boundary-scan chain is selected as data path. Bit 0 is the LSB; the first bit scanned in, and the first bit scanned out. The scan order follows the pin-out order as far as possible. Therefore, the bits of Port A is scanned in the opposite bit order of the other ports. Exceptions from the rules are the Scan chains for the analog circuits, which constitute the most significant bits of the scan chain regardless of which physical pin they are connected to. In Figure 116, PXn. Data corresponds to FF0, PXn. Control corresponds to FF1, and PXn. Pullup_enable corresponds to FF2. Bit 2, 3, 4, and 5 of Port C is not in the scan chain, since these pins constitute the TAP pins when the JTAG is enabled.

Table 94. ATmega16 Boundary-scan Order

Bit Number	Signal Name	Module
140	AC_IDLE	Comparator
139	ACO	
138	ACME	
137	ACBG	
136	COMP	ADC
135	PRIVATE_SIGNAL1(1)	
134	ACLK	
133	ACTEN	
132	PRIVATE_SIGNAL2(2)	
131	ADCBGEN	
130	ADCEN	
129	AMPEN	
128	DAC_9	
127	DAC_8	
126	DAC_7	
125	DAC_6	
124	DAC_5	
123	DAC_4	
122	DAC_3	
121	DAC_2	
120	DAC_1	
119	DAC_0	
118	EXTCH	
117	G10	
116	G20	
115	GNDEN	
114	HOLD	
113	IREFEN	
112	MUXEN_7	



241



Table 94. ATmega16 Boundary-scan Order (Continued)

Bit Number	Signal Name	Module
111	MUXEN_6	
110	MUXEN_5	
109	MUXEN_4	
108	MUXEN_3	
107	MUXEN_2	
106	MUXEN_1	
105	MUXEN_0	
104	NEGSEL_2	
103	NEGSEL_1	
102	NEGSEL_0	
101	PASSEN	
100	PRECH	
99	SCTEST	
98	ST	
97	VCCREN	
96	PB0.Data	Port B
95	PB0.Control	
94	PB0.Pullup_Enable	
93	PB1.Data	
92	PB1.Control	
91	PB1.Pullup_Enable	
90	PB2.Data	
89	PB2.Control	
88	PB2.Pullup_Enable	
87	PB3.Data	
86	PB3.Control	
85	PB3.Pullup_Enable	
84	PB4.Data	
83	PB4.Control	
82	PB4.Pullup_Enable	
81	PB5.Data	
80	PB5.Control	
79	PB5.Pullup_Enable	
78	PB6.Data	
77	PB6.Control	
76	PB6.Pullup_Enable	





Table 94. ATmega16 Boundary-scan Order (Continued)

Bit Number	Signal Name	Module
75	PB7.Data	
74	PB7.Control	
73	PB7.Pullup_Enable	
72	RSTT	Reset Logic
71	RSTHV	(Observe-Only)
70	EXTCLKEN	Enable signals for main clock/Oscillators
69	OSCON	
68	RCOSCEN	
67	OSC32EN	
66	EXTCLK (XTAL1)	Clock input and Oscillators for the main clock
65	OSCCK	(Observe-Only)
64	RCCK	
63	OSC32CK	
62	TWIEN	TWI
61	PD0.Data	Port D
60	PD0.Control	
59	PD0.Pullup_Enable	
58	PD1.Data	
57	PD1.Control	
56	PD1.Pullup_Enable	
55	PD2.Data	
54	PD2.Control	
53	PD2.Pullup_Enable	
52	PD3.Data	
51	PD3.Control	
50	PD3.Pullup_Enable	
49	PD4.Data	
48	PD4.Control	
47	PD4.Pullup_Enable	
46	PD5.Data	
45	PD5.Control	
44	PD5.Pullup_Enable	
43	PD6.Data	
42	PD6.Control	
41	PD6.Pullup_Enable	
40	PD7.Data	



243



Table 94. ATmega16 Boundary-scan Order (Continued)

Bit Number	Signal Name	Module
39	PD7.Control	
38	PD7.Pullup_Enable	
37	PC0.Data	Port C
36	PC0.Control	
35	PC0.Pullup_Enable	
34	PC1.Data	
33	PC1.Control	
32	PC1.Pullup_Enable	
31	PC6.Data	
30	PC6.Control	
29	PC6.Pullup_Enable	
28	PC7.Data	
27	PC7.Control	
26	PC7.Pullup_Enable	
25	TOSC	32 kHz Timer Oscillator
24	TOSCON	
23	PA7.Data	Port A
22	PA7.Control	
21	PA7.Pullup_Enable	
20	PA6.Data	
19	PA6.Control	
18	PA6.Pullup_Enable	
17	PA5.Data	
16	PA5.Control	
15	PA5.Pullup_Enable	
14	PA4.Data	
13	PA4.Control	
12	PA4.Pullup_Enable	
11	PA3.Data	
10	PA3.Control	
9	PA3.Pullup_Enable	
8	PA2.Data	
7	PA2.Control	
6	PA2.Pullup_Enable	
5	PA1.Data	



244



Table 94. ATmega16 Boundary-scan Order (Continued)

Bit Number	Signal Name	Module
4	PA1.Control	
3	PA1.Pullup_Enable	
2	PA0.Data	
1	PA0.Control	
0	PA0.Pullup_Enable	

Notes: 1. PRIVATE_SIGNAL1 should always be scanned in as zero.
2. PRIVATE:SIGNAL2 should always be scanned in as zero.

Boundary-scan Description Language Files Boundary-scan Description Language (BSDL) files describe Boundary-scan capable devices in a standard format used by automated test-generation software. The order and function of bits in the Boundary-scan Data Register are included in this description. A BSDL file for ATmega16 is available.





Boot Loader Support – Read-While-Write Self-Programming

The Boot Loader Support provides a real Read-While-Write Self-Programming mechanism for downloading and uploading program code by the MCU itself. This feature allows flexible application software updates controlled by the MCU using a Flash-resident Boot Loader program. The Boot Loader program can use any available data interface and associated protocol to read code and write (program) that code into the Flash memory, or read the code from the Program memory. The program code within the Boot Loader section has the capability to write into the entire Flash, including the Boot Loader memory. The Boot Loader can thus even modify itself, and it can also erase itself from the code if the feature is not needed anymore. The size of the Boot Loader memory is configurable with Fuses and the Boot Loader has two separate sets of Boot Lock bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

Features

- · Read-While-Write Self-Programming
- · Flexible Boot Memory size
- High Security (Separate Boot Lock Bits for a Flexible Protection)
- Separate Fuse to Select Reset Vector
- Optimized Page⁽¹⁾ Size
- · Code Efficient Algorithm
- Efficient Read-Modify-Write Support

Note: 1. A page is a section in the flash consisting of several bytes (see Table 107 on page 262) used during programming. The page organization does not affect normal operation.

Application and Boot Loader Flash Sections

The Flash memory is organized in two main sections, the Application section and the Boot Loader section (see Figure 125). The size of the different sections is configured by the BOOTSZ Fuses as shown in Table 100 on page 257 and Figure 125. These two sections can have different level of protection since they have different sets of Lock bits.

Application Section

The Application section is the section of the Flash that is used for storing the application code. The protection level for the application section can be selected by the Application Boot Lock bits (Boot Lock bits 0), see Table 96 on page 249. The Application section can never store any Boot Loader code since the SPM instruction is disabled when executed from the Application section.

BLS – Boot Loader Section

While the Application section is used for storing the application code, the The Boot Loader software must be located in the BLS since the SPM instruction can initiate a programming when executing from the BLS only. The SPM instruction can access the entire Flash, including the BLS itself. The protection level for the Boot Loader section can be selected by the Boot Loader Lock bits (Boot Lock bits 1), see Table 97 on page 249.

Read-While-Write and no Read-While-Write Flash Sections

Whether the CPU supports Read-While-Write or if the CPU is halted during a Boot Loader software update is dependent on which address that is being programmed. In addition to the two sections that are configurable by the BOOTSZ Fuses as described above, the Flash is also divided into two fixed sections, the Read-While-Write (RWW) section and the No Read-While-Write (NRWW) section. The limit between the RWW- and NRWW sections is given in Table 101 on page 257 and Figure 125 on page 248. The main difference between the two sections is:

- When erasing or writing a page located inside the RWW section, the NRWW section can be read during the operation.
- When erasing or writing a page located inside the NRWW section, the CPU is halted during the entire operation.

Note that the user software can never read any code that is located inside the RWW section during a Boot Loader software operation. The syntax "Read-While-Write section" refers to which section that is being programmed (erased or written), not which section that actually is being read during a Boot Loader software update.



246



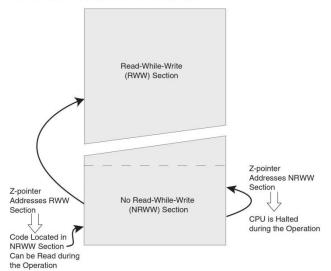
RWW – Read-While-Write Section If a Boot Loader software update is programming a page inside the RWW section, it is possible to read code from the Flash, but only code that is located in the NRWW section. During an ongoing programming, the software must ensure that the RWW section never is being read. If the user software is trying to read code that is located inside the RWW section (that is, by a call/jmp/lpm or an interrupt) during programming, the software might end up in an unknown state. To avoid this, the interrupts should either be disabled or moved to the Boot Loader section. The Boot Loader section. The Boot Loader section is always located in the NRWW section. The RWW Section Busy bit (RWWSB) in the Store Program Memory Control Register (SPMCR) will be read as logical one as long as the RWW section is blocked for reading. After a programming is completed, the RWWSB must be cleared by software before reading code located in the RWW section. See "Store Program Memory Control Register – SPMCR" on page 250. for details on how to clear RWWSB.

NRWW – No Read-While-Write Section The code located in the NRWW section can be read when the Boot Loader software is updating a page in the RWW section. When the Boot Loader code updates the NRWW section, the CPU is halted during the entire page erase or page write operation.

Table 95. Read-While-Write Features

Which Section does the Z- pointer Address during the Programming?	Which Section can be Read during Programming?	Is the CPU Halted?	Read-While- Write Supported?	
RWW section	NRWW section	No	Yes	
NRWW section	None	Yes	No	

Figure 124. Read-While-Write vs. No Read-While-Write

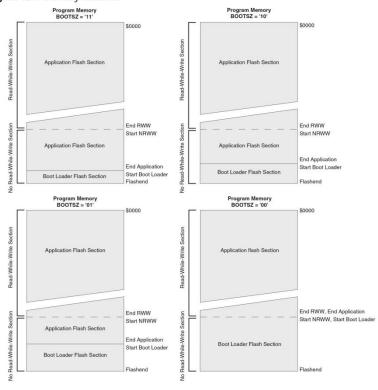




247



Figure 125. Memory Sections⁽¹⁾



Note: 1. The parameters in the figure above are given in Table 100 on page 257.

Boot Loader Lock Bits

If no Boot Loader capability is needed, the entire Flash is available for application code. The Boot Loader has two separate sets of Boot Lock bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

The user can select:

- · To protect the entire Flash from a software update by the MCU
- To protect only the Boot Loader Flash section from a software update by the MCU
- To protect only the Application Flash section from a software update by the MCU
- Allow software update in the entire Flash

See Table 96 and Table 97 for further details. The Boot Lock bits can be set in software and in Serial or Parallel Programming mode, but they can be cleared by a Chip Erase command only. The general Write Lock (Lock Bit mode 2) does not control the programming of the Flash memory by SPM instruction. Similarly, the general Read/Write Lock (Lock Bit mode 3) does not control reading nor writing by LPM/SPM, if it is attempted.

<u>AIMEL</u>

248



Table 96. Boot Lock BitO Protection Modes (Application Section)⁽¹⁾

BLB0 Mode	BLB02	BLB01	Protection
1	1	1	No restrictions for SPM or LPM accessing the Application section.
2	1	0	SPM is not allowed to write to the Application section.
3	0	0	SPM is not allowed to write to the Application section, and LPM executing from the Boot Loader section is not allowed to read from the Application section. If interrupt vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.
4	0	1	LPM executing from the Boot Loader section is not allowed to read from the Application section. If interrupt vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.

Note: 1. "1" means unprogrammed, "0" means programmed

Table 97. Boot Lock Bit1 Protection Modes (Boot Loader Section)⁽¹⁾

BLB1 mode	BLB12	BLB11	Protection				
1	1	1	No restrictions for SPM or LPM accessing the Boot Loader section.				
2	1	0	SPM is not allowed to write to the Boot Loader section.				
3	0	0	SPM is not allowed to write to the Boot Loader section, and LPM executing from the Application section is not allowed to read from the Boot Loader section. If interrupt vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.				
4	0	1	LPM executing from the Application section is not allowed to read from the Boot Loader section. If interrupt vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.				

Note: 1. "1" means unprogrammed, "0" means programmed

Entering the Boot Loader Program

Entering the Boot Loader takes place by a jump or call from the application program. This may be initiated by a trigger such as a command received via USART, or SPI interface. Alternatively, the Boot Reset Fuse can be programmed so that the Reset Vector is pointing to the Boot Flash start address after a reset. In this case, the Boot Loader is started after a reset. After the application code is loaded, the program can start executing the application code. Note that the fuses cannot be changed by the MCU itself. This means that once the Boot Reset Fuse is programmed, the Reset Vector will always point to the Boot Loader Reset and the fuse can only be changed through the serial or parallel programming interface.





Table 98. Boot Reset Fuse(1)

BOOTRST	Reset Address					
1	Reset Vector = Application reset (address \$0000)					
0	Reset Vector = Boot Loader reset (see Table 100 on page 257)					

Note: 1. "1" means unprogrammed, "0" means programmed

Store Program Memory Control Register – SPMCR The Store Program Memory Control Register contains the control bits needed to control the Boot Loader operations.

Bit	7	6	5	4	3	2	1	0	
	SPMIE	RWWSB	-	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	SPMCR
Read/Write	R/W	R	R	R/W	R/W	R/W	R/W	R/W	•
Initial value	0	0	0	0	0	0	0	0	

· Bit 7 - SPMIE: SPM Interrupt Enable

When the SPMIE bit is written to one, and the I-bit in the Status Register is set (one), the SPM ready interrupt will be enabled. The SPM ready Interrupt will be executed as long as the SPMEN bit in the SPMCR Register is cleared.

· Bit 6 - RWWSB: Read-While-Write Section Busy

When a self-programming (Page Erase or Page Write) operation to the RWW section is initiated, the RWWSB will be set (one) by hardware. When the RWWSB bit is set, the RWW section cannot be accessed. The RWWSB bit will be cleared if the RWWSRE bit is written to one after a Self-Programming operation is completed. Alternatively the RWWSB bit will automatically be cleared if a page load operation is initiated.

· Bit 5 - Res: Reserved Bit

This bit is a reserved bit in the ATmega16 and always read as zero.

· Bit 4 - RWWSRE: Read-While-Write Section Read Enable

When programming (Page Erase or Page Write) to the RWW section, the RWW section is blocked for reading (the RWWSB will be set by hardware). To re-enable the RWW section, the user software must wait until the programming is completed (SPMEN will be cleared). Then, if the RWWSRE bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles re-enables the RWW section. The RWW section cannot be re-enabled while the Flash is busy with a page erase or a page write (SPMEN is set). If the RWWSRE bit is written while the Flash is being loaded, the Flash load operation will abort and the data loaded will be lost.

· Bit 3 - BLBSET: Boot Lock Bit Set

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles sets Boot Lock bits, according to the data in R0. The data in R1 and the address in the Z-pointer are ignored. The BLBSET bit will automatically be cleared upon completion of the Lock bit set, or if no SPM instruction is executed within four clock cycles.

An LPM instruction within three cycles after BLBSET and SPMEN are set in the SPMCR Register, will read either the Lock bits or the Fuse bits (depending on Z0 in the Z-pointer) into the destination register. See "Reading the Fuse and Lock Bits from Software" on page 254 for details.



250



· Bit 2 - PGWRT: Page Write

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes Page Write, with the data stored in the temporary buffer. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGWRT bit will auto-clear upon completion of a page write, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire page write operation if the NRWW section is addressed.

· Bit 1 - PGERS: Page Erase

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes Page Erase. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGERS bit will auto-clear upon completion of a page erase, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire page write operation if the NRWW section is addressed.

Bit 0 – SPMEN: Store Program Memory Enable

This bit enables the SPM instruction for the next four clock cycles. If written to one together with either RWWSRE, BLBSET, PGWRT' or PGERS, the following SPM instruction will have a special meaning, see description above. If only SPMEN is written, the following SPM instruction will store the value in R1:R0 in the temporary page buffer addressed by the Z-pointer. The LSB of the Z-pointer is ignored. The SPMEN bit will auto-clear upon completion of an SPM instruction, or if no SPM instruction is executed within four clock cycles. During page erase and page write, the SPMEN bit remains high until the operation is completed.

Writing any other combination than "10001", "01001", "00101", "00001" or "00001" in the lower five bits will have no effect.

Addressing the Flash during Self-Programming

The Z-pointer is used to address the SPM commands.

Bit	15	14	13	12	11	10	9	8
ZH (R31)	Z15	Z14	Z13	Z12	Z11	Z10	Z9	Z8
ZL (R30)	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0
		6	- 5	1	2	2		0

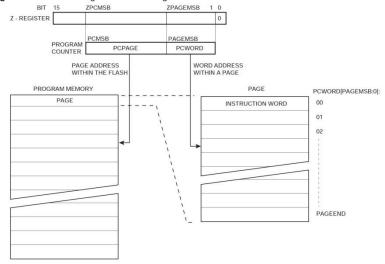
Since the Flash is organized in pages (see Table 107 on page 262), the Program Counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is shown in Figure 126. Note that the Page Erase and Page Write operations are addressed independently. Therefore it is of major importance that the Boot Loader software addresses the same page in both the Page Erase and Page Write operation. Once a programming operation is initiated, the address is latched and the Z-pointer can be used for other operations.

The only SPM operation that does not use the Z-pointer is Setting the Boot Loader Lock bits. The content of the Z-pointer is ignored and will have no effect on the operation. The LPM instruction does also use the Z pointer to store the address. Since this instruction addresses the Flash byte by byte, also the LSB (bit Z0) of the Z-pointer is used.





Figure 126. Addressing the Flash during SPM⁽¹⁾



Notes: 1. The different variables used in Figure 126 are listed in Table 102 on page 258.

2. PCPAGE and PCWORD are listed in Table 107 on page 262.

Self-Programming the Flash

The program memory is updated in a page by page fashion. Before programming a page with the data stored in the temporary page buffer, the page must be erased. The temporary page buffer is filled one word at a time using SPM and the buffer can be filled either before the page erase command or between a page erase and a page write operation:

Alternative 1, fill the buffer before a Page Erase

- Fill temporary page buffer
- Perform a Page Erase
- Perform a Page Write

Alternative 2, fill the buffer after Page Erase

- Perform a Page Erase
- · Fill temporary page buffer
- Perform a Page Write

If only a part of the page needs to be changed, the rest of the page must be stored (for example in the temporary page buffer) before the erase, and then be rewritten. When using alternative 1, the Boot Loader provides an effective Read-Modify-Write feature which allows the user software to first read the page, do the necessary changes, and then write back the modified data. If alternative 2 is used, it is not possible to read the old data while loading since the page is already erased. The temporary page buffer can be accessed in a random sequence. It is essential that the page address used in both the page erase and page write operation is addressing the same page. See "Simple Assembly Code Example for a Boot Loader" on page 256 for an assembly code example.



252



Performing Page Erase by SPM

To execute Page Erase, set up the address in the Z-pointer, write "X0000011" to SPMCR and execute SPM within four clock cycles after writing SPMCR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE in the Z-register. Other bits in the Z-pointer must be written zero during this operation.

- Page Erase to the RWW section: The NRWW section can be read during the page erase.
- · Page Erase to the NRWW section: The CPU is halted during the operation.

Filling the Temporary Buffer (Page Loading)

To write an instruction word, set up the address in the Z-pointer and data in R1:R0, write "00000001" to SPMCR and execute SPM within four clock cycles after writing SPMCR. The content of PCWORD in the Z-register is used to address the data in the temporary buffer. The temporary buffer will auto-erase after a page write operation or by writing the RWWSRE bit in SPMCR. It is also erased after a system reset. Note that it is not possible to write more than one time to each address without erasing the temporary buffer.

Note: If the EEPROM is written in the middle of an SPM Page Load operation, all data loaded will be lost.

Performing a Page Write

To execute Page Write, set up the address in the Z-pointer, write "X0000101" to SPMCR and execute SPM within four clock cycles after writing SPMCR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE. Other bits in the Z-pointer must be written zero during this operation.

- · Page Write to the RWW section: The NRWW section can be read during the Page Write.
- · Page Write to the NRWW section: The CPU is halted during the operation.

Using the SPM Interrupt

If the SPM interrupt is enabled, the SPM interrupt will generate a constant interrupt when the SPMEN bit in SPMCR is cleared. This means that the interrupt can be used instead of polling the SPMCR Register in software. When using the SPM interrupt, the Interrupt Vectors should be moved to the BLS section to avoid that an interrupt is accessing the RWW section when it is blocked for reading. How to move the interrupts is described in "Interrupts" on page 45.

Consideration while Updating BLS

Special care must be taken if the user allows the Boot Loader section to be updated by leaving Boot Lock bit11 unprogrammed. An accidental write to the Boot Loader itself can corrupt the entire Boot Loader, and further software updates might be impossible. If it is not necessary to change the Boot Loader software itself, it is recommended to program the Boot Lock bit11 to protect the Boot Loader software from any internal software changes.

Prevent Reading the RWW Section during Self-Programming

During Self-Programming (either Page Erase or Page Write), the RWW section is always blocked for reading. The user software itself must prevent that this section is addressed during the Self-Programming operation. The RWWSB in the SPMCR will be set as long as the RWW section is busy. During self-programming the Interrupt Vector table should be moved to the BLS as described in "Interrupts" on page 45, or the interrupts must be disabled. Before addressing the RWW section after the programming is completed, the user software must clear the RWWSB by writing the RWWSRE. See "Simple Assembly Code Example for a Boot Loader" on page 256 for an example.





Setting the Boot Loader Lock Bits by SPM To set the Boot Loader Lock bits, write the desired data to R0, write "X0001001" to SPMCR and execute SPM within four clock cycles after writing SPMCR. The only accessible Lock bits are the Boot Lock bits that may prevent the Application and Boot Loader section from any software update by the MCU.

Bit	7	6	5	4	3	2	1	0
R0	1	1	BLB12	BLB11	BLB02	BLB01	1	1

See Table 96 and Table 97 for how the different settings of the Boot Loader bits affect the Flash access.

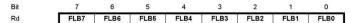
If bits 5..2 in R0 are cleared (zero), the corresponding Boot Lock bit will be programmed if an SPM instruction is executed within four cycles after BLBSET and SPMEN are set in SPMCR. The Z-pointer is don't care during this operation, but for future compatibility it is recommended to load the Z-pointer with \$0001 (same as used for reading the Lock bits). For future compatibility It is also recommended to set bits 7, 6, 1, and 0 in R0 to "1" when writing the Lock bits. When programming the Lock bits the entire Flash can be read during the operation.

EEPROM Write Prevents Writing to SPMCR Note that an EEPROM write operation will block all software programming to Flash. Reading the Fuses and Lock bits from software will also be prevented during the EEPROM write operation. It is recommended that the user checks the status bit (EEWE) in the EECR Register and verifies that the bit is cleared before writing to the SPMCR Register.

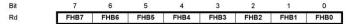
Reading the Fuse and Lock Bits from Software It is possible to read both the Fuse and Lock bits from software. To read the Lock bits, load the Z-pointer with \$0001 and set the BLBSET and SPMEN bits in SPMCR. When an LPM instruction is executed within three CPU cycles after the BLBSET and SPMEN bits are set in SPMCR, the value of the Lock bits will be loaded in the destination register. The BLBSET and SPMEN bits will auto-clear upon completion of reading the Lock bits or if no LPM instruction is executed within three CPU cycles or no SPM instruction is executed within four CPU cycles. When BLB-SET and SPMEN are cleared, LPM will work as described in the Instruction set Manual.

Bit	7	6	5	4	3	2	1	0
Rd	-	-	BLB12	BLB11	BLB02	BLB01	LB2	LB1

The algorithm for reading the Fuse Low bits is similar to the one described above for reading the Lock bits. To read the Fuse Low bits, load the Z-pointer with \$0000 and set the BLBSET and SPMEN bits in SPMCR. When an LPM instruction is executed within three cycles after the BLBSET and SPMEN bits are set in the SPMCR, the value of the Fuse Low bits (FLB) will be loaded in the destination register as shown below. Refer to Table 106 on page 261 for a detailed description and mapping of the Fuse Low bits.



Similarly, when reading the Fuse High bits, load \$0003 in the Z-pointer. When an LPM instruction is executed within three cycles after the BLBSET and SPMEN bits are set in the SPMCR, the value of the Fuse High bits (FHB) will be loaded in the destination register as shown below. Refer to Table 105 on page 260 for detailed description and mapping of the Fuse High bits.



Fuse and Lock bits that are programmed, will be read as zero. Fuse and Lock bits that are unprogrammed, will be read as one.

Preventing Flash Corruption During periods of low V_{CC} the Flash program can be corrupted because the supply voltage is too low for the CPU and the Flash to operate properly. These issues are the same as for board level systems using the Flash, and the same design solutions should be applied.

AIMEL

254



A Flash program corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the Flash requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage for executing instructions is too low.

Flash corruption can easily be avoided by following these design recommendations (one is sufficient):

- If there is no need for a Boot Loader update in the system, program the Boot Loader Lock bits to prevent any Boot Loader software updates.
- 2. Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD) if the operating voltage matches the detection level. If not, an external low V_{CC} Reset Protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.
- Keep the AVR core in Power-down Sleep mode during periods of low V_{CC}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the SPMCR Register and thus the Flash from unintentional writes.

Programming Time for Flash when using SPM

The Calibrated RC Oscillator is used to time Flash accesses. Table 99 shows the typical programming time for Flash accesses from the CPU.

Table 99. SPM Programming Time.

Symbol	Min Programming Time	Max Programming Time	
Flash write (Page Erase, Page Write, and write Lock bits by SPM)	3.7 ms	4.5 ms	





Simple Assembly Code Example for a **Boot Loader**

```
;-the routine writes one page of data from RAM to Flash ; the first data location in RAM is pointed to by the Y pointer ; the first data location in Flash is pointed to by the Z pointer
  ;-error handling is not included
  ;-the routine must be placed inside the boot space ; (at least the Do_spm sub routine). Only code inside NRWW section can
  ; be read during self-programming (page erase and page write).
  ;-registers used: r0, r1, temp1 (r16), temp2 (r17), looplo (r24), ; loophi (r25), spmcrval (r20)
  ; storing and restoring of registers is not included in the routine
  ; register usage can be optimized at the expense of code size
  ;-It is assumed that either the interrupt table is moved to the Boot
  ; loader section or that the interrupts are disabled.
.equ PAGESIZEB = PAGESIZE*2
                                        ; PAGESIZEB is page size in BYTES, not
                                         ; words
.org SMALLBOOTSTART
Write_page:
  ; page erase
  ldi spmcrval, (1<<PGERS) | (1<<SPMEN) call Do_spm
  ; re-enable the RWW section
 ldi spmcrval, (1<<RWWSRE) | (1<<SPMEN) call Do_spm
    transfer data from RAM to Flash page buffer
                                    ;init loop variable
  ldi looplo, low(PAGESIZEB)
ldi loophi, high(PAGESIZEB)
                                         ;not required for PAGESIZEB<=256
Wrloop:
 ld r0, Y+
ld r1, Y+
ldi spmcrval, (1<<SPMEN)
  call Do_spm
  adiw ZH:ZL, 2
  sbiw loophi:looplo, 2
                                       ;use subi for PAGESIZEB<=256
  brne Wrloop
  ; execute page write
  subi ZL, low(PAGESIZEB)
                                         ;restore pointer
                                          ;not required for PAGESIZEB<=256
  sbci ZH, high(PAGESIZEB)
      spmcrval, (1<<PGWRT) | (1<<SPMEN)
  call Do_spm
  ; re-enable the RWW section
  ldi spmcrval, (1<<RWWSRE) | (1<<SPMEN)
  call Do_spm
  ; read back and check, optional
 ldi looplo, low(PAGESIZEB)
ldi loophi, high(PAGESIZEB)
                                        ;init loop variable
                                         ;not required for PAGESIZEB<=256
  subi YL, low(PAGESIZEB)
sbci YH, high(PAGESIZEB)
                                         ;restore pointer
Rdloop:
 lpm r0, Z+
ld r1, Y+
  cpse r0, r1
  jmp Error
  sbiw loophi:looplo, 1
                                         ;use subi for PAGESIZEB<=256
  brne Rdloop
  ; return to RWW section
  ; verify that RWW section is safe to read
Return:
       temp1, SPMCR
```



```
sbrs temp1, RWWSB
                                    ; If RWWSB is set, the RWW section is not
                                    ; ready yet
  ret
  ; re-enable the RWW section
  ldi spmcrval, (1<<RWWSRE) | (1<<SPMEN)
  call Do_spm
  rjmp Return
Do_spm:
   check for previous SPM complete
Wait_spm:
 in temp1, SPMCR sbrc temp1, SPMEN
  rjmp Wait_spm
 ; input: spmcrval determines SPM action
; disable interrupts if enabled, store status
  in temp2, SREG
  cli
  ; check that no EEPROM write access is present
Wait_ee:
  sbic EECR, EEWE
  rjmp Wait_ee
  ; SPM timed sequence
  out SPMCR, spmcrval
  ; restore SREG (to enable interrupts if originally enabled)
  out SREG, temp2
  ret
```

ATmega16 Boot Loader Parameters

In Table 100 through Table 102, the parameters used in the description of the self programming are given.

Table 100. Boot Size Configuration⁽¹⁾

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Section	Boot Loader Flash Section	End Application section	Boot Reset Address (start Boot Loader Section)
1	1	128 words	2	\$0000 - \$1F7F	\$1F80 - \$1FFF	\$1F7F	\$1F80
1	0	256 words	4	\$0000 - \$1EFF	\$1F00 - \$1FFF	\$1EFF	\$1F00
0	1	512 words	8	\$0000 - \$1DFF	\$1E00 - \$1FFF	\$1DFF	\$1E00
0	0	1024 words	16	\$0000 - \$1BFF	\$1C00 - \$1FFF	\$1BFF	\$1C00

Note: 1. The different BOOTSZ Fuse configurations are shown in Figure 125

Table 101. Read-While-Write Limit (1)

Section	Pages	Address	
Read-While-Write section (RWW)	112	\$0000 - \$1BFF	
No Read-While-Write section (NRWW)	16	\$1C00 - \$1FFF	

Note: 1. For details about these two section, see "NRWW – No Read-While-Write Section" on page 247 and "RWW – Read-While-Write Section" on page 247



257



Table 102. Explanation of Different Variables used in Figure 126 and the Mapping to the Zpointer

Variable		Corresponding Z-value ⁽¹⁾	Description
PCMSB	12		Most significant bit in the Program Counter. (The Program Counter is 13 bits PC[12:0])
PAGEMSB	5		Most significant bit which is used to address the words within one page (64 words in a page requires 6 bits PC [5:0]).
ZPCMSB		Z13	Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1.
ZPAGEMSB		Z6	Bit in Z-register that is mapped to PAGEMSB. Because Z0 is not used, the ZPAGEMSB equals PAGEMSB + 1.
PCPAGE	PC[12:6]	Z13:Z7	Program Counter page address: Page select, for Page Erase and Page Write
PCWORD	PC[5:0]	Z6:Z1	Program Counter word address: Word select, for filling temporary buffer (must be zero during page write operation)

Z15:Z14: always ignored
 Z0: should be zero for all SPM commands, byte select for the LPM instruction.
 See "Addressing the Flash during Self-Programming" on page 251 for details about the use of Z-pointer during Self-Programming.





Memory Programming

Program And Data Memory Lock Bits The ATmega16 provides six Lock bits which can be left unprogrammed ("1") or can be programmed ("0") to obtain the additional features listed in Table 104. The Lock bits can only be erased to "1" with the Chip Erase command.

Table 103. Lock Bit Byte⁽¹⁾

Lock Bit Byte	Bit No.	Description	Default Value
	7	-	1 (unprogrammed)
	6		1 (unprogrammed)
BLB12	5	Boot Lock bit	1 (unprogrammed)
BLB11	4	Boot Lock bit	1 (unprogrammed)
BLB02	3	Boot Lock bit	1 (unprogrammed)
BLB01	2	Boot Lock bit	1 (unprogrammed)
LB2	1	Lock bit	1 (unprogrammed)
LB1	0	Lock bit	1 (unprogrammed)

Note: 1. "1" means unprogrammed, "0" means programmed

Table 104. Lock Bit Protection Modes

Memory	Lock Bit	s ⁽²⁾	Protection Type	
LB Mode	LB2	LB1		
1	1	1	No memory lock features enabled.	
2	1	0	Further programming of the Flash and EEPROM is disabled in Parallel and SPI/JTAG Serial Programming mode. The Fuse bits are locked in both Serial and Parallel Programming mode. ⁽¹⁾	
3	0	0	Further programming and verification of the Flash and EEPROM is disabled in Parallel and SPI/JTAG Serial Programming mode. The Fuse bits are locked in both Serial and Parallel Programming mode. ⁽¹⁾	
BLB0 Mode	BLB02	BLB01		
1	1	1	No restrictions for SPM or LPM accessing the Application section.	
2	1	0	SPM is not allowed to write to the Application section.	
3	0	0	SPM is not allowed to write to the Application section, and LPM executing from the Boot Loader section is not allowed to read from the Application section. If interrupt vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.	
4	0	1	LPM executing from the Boot Loader section is not allowed to read from the Application section. If interrupt vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.	



259



Table 104. Lock Bit Protection Modes (Continued)

Memory Lock Bits ⁽²⁾			Protection Type
BLB1 Mode	BLB12	BLB11	
1	1	1	No restrictions for SPM or LPM accessing the Boot Loader section.
2	1	0	SPM is not allowed to write to the Boot Loader section.
3	0	0	SPM is not allowed to write to the Boot Loader section, and LPM executing from the Application section is not allowed to read from the Boot Loader section. If interrupt vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.
4	0	1	LPM executing from the Application section is not allowed to read from the Boot Loader section. If interrupt vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.

Notes: 1. Program the Fuse bits before programming the Lock bits.

2. "1" means unprogrammed, "0" means programmed

Fuse Bits

The ATmega16 has two fuse bytes. Table 105 and Table 106 describe briefly the functionality of all the fuses and how they are mapped into the fuse bytes. Note that the fuses are read as logical zero, "0", if they are programmed.

Table 105. Fuse High Byte

Fuse High Bit No.		Description	Default Value
OCDEN ⁽⁴⁾	7	Enable OCD	1 (unprogrammed, OCD disabled)
JTAGEN ⁽⁵⁾	6	Enable JTAG	0 (programmed, JTAG enabled)
SPIEN ⁽¹⁾	5	Enable SPI Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
CKOPT ⁽²⁾	4	Oscillator options	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BOOTSZ1	2	Select Boot Size (see Table 100 for details)	0 (programmed) ⁽³⁾
BOOTSZ0	1	Select Boot Size (see Table 100 for details)	0 (programmed) ⁽³⁾
BOOTRST	0	Select reset vector	1 (unprogrammed)

Notes: 1. The SPIEN Fuse is not accessible in SPI Serial Programming mode.

- The CKOPT Fuse functionality depends on the setting of the CKSEL bits. See See "Clock Sources" on page 25. for details.
- 3. The default value of BOOTSZ1..0 results in maximum Boot Size. See Table 100 on page 257.
- 4. Never ship a product with the OCDEN Fuse programmed regardless of the setting of Lock bits and the JTAGEN Fuse. A programmed OCDEN Fuse enables some parts of the clock system to be running in all sleep modes. This may increase the power consumption.
- If the JTAG interface is left unconnected, the JTAGEN fuse should if possible be disabled. This to avoid static current at the TDO pin in the JTAG interface.



260



Table 106. Fuse Low Byte

Fuse Low Bit No. BODLEVEL 7		Description	Default Value	
		Brown-out Detector trigger level	1 (unprogrammed)	
BODEN	6	Brown-out Detector enable	1 (unprogrammed, BOD disabled)	
SUT1	5	Select start-up time	1 (unprogrammed) ⁽¹⁾	
SUT0	4	Select start-up time	0 (programmed) ⁽¹⁾	
CKSEL3	3	Select Clock source	ource 0 (programmed) ⁽²⁾	
CKSEL2	2	Select Clock source	0 (programmed) ⁽²⁾	
CKSEL1	1	Select Clock source	0 (programmed) ⁽²⁾	
CKSEL0	0	Select Clock source	1 (unprogrammed) ⁽²⁾	

Notes: 1. The default value of SUT1..0 results in maximum start-up time. SeeTable 10 on page 29 for details.

2. The default setting of CKSEL3..0 results in internal RC Oscillator @ 1 MHz. See Table 2 on page 25 for details.

The status of the Fuse bits is not affected by Chip Erase. Note that the Fuse bits are locked if Lock bit1 (LB1) is programmed. Program the Fuse bits before programming the Lock bits.

Latching of Fuses

The Fuse values are latched when the device enters programming mode and changes of the Fuse values will have no effect until the part leaves Programming mode. This does not apply to the EESAVE Fuse which will take effect once it is programmed. The fuses are also latched on Power-up in Normal mode.

Signature Bytes

All Atmel microcontrollers have a three-byte signature code which identifies the device. This code can be read in both serial and parallel mode, also when the device is locked. The three bytes reside in a separate address space.

For the ATmega16 the signature bytes are:

- 1. \$000: \$1E (indicates manufactured by Atmel)
- 2. \$001: \$94 (indicates 16 Kbytes Flash memory)
- 3. \$002: \$03 (indicates ATmega16 device when \$001 is \$94)

Calibration Byte

The ATmega16 stores four different calibration values for the internal RC Oscillator. These bytes resides in the signature row High Byte of the addresses 0x0000, 0x0001, 0x0002, and 0x0003 for 1 MHz, 2 MHz, 4 MHz, and 8 Mhz respectively. During Reset, the 1 MHz value is automatically loaded into the OSCCAL Register. If other frequencies are used, the calibration value has to be loaded manually, see "Oscillator Calibration Register – OSCCAL" on page 30 for details.





Page Size

Table 107. No. of Words in a Page and no. of Pages in the Flash

Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
8K words (16 Kbytes)	64 words	PC[5:0]	128	PC[12:6]	12

Table 108. No. of Words in a Page and no. of Pages in the EEPROM

EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
512 bytes	4 bytes	EEA[1:0]	128	EEA[8:2]	8

Parallel Programming Parameters, Pin Mapping, and Commands This section describes how to parallel program and verify Flash Program memory, EEPROM Data memory, Memory Lock bits, and Fuse bits in the ATmega16. Pulses are assumed to be at least 250 ns unless otherwise noted.

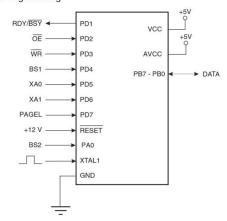
Signal Names

In this section, some pins of the ATmega16 are referenced by signal names describing their functionality during parallel programming, see Figure 127 and Table 109. Pins not described in the following table are referenced by pin names.

The XA1/XA0 pins determine the action executed when the XTAL1 pin is given a positive pulse. The bit coding is shown in Table 111.

When pulsing \overline{WR} or \overline{OE} , the command loaded determines the action executed. The different Commands are shown in Table 112.

Figure 127. Parallel Programming



<u>AIMEL</u>

262



Table 109. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
RDY/BSY	PD1	О	0: Device is busy programming, 1: Device is ready for new command
ŌĒ	PD2	1	Output Enable (Active low)
WR	PD3	1	Write Pulse (Active low)
BS1	PD4	1	Byte Select 1 ("0" selects Low byte, "1" selects High byte)
XA0	PD5	1	XTAL Action Bit 0
XA1	PD6	1	XTAL Action Bit 1
PAGEL	PD7	1	Program Memory and EEPROM data Page Load
BS2	PA0	1	Byte Select 2 ("0" selects Low byte, "1" selects 2'nd High byte)
DATA	PB7-0	I/O	Bidirectional Data bus (Output when OE is low)

Table 110. Pin Values used to Enter Programming Mode

Pin	Symbol	Value
PAGEL	Prog_enable[3]	0
XA1	Prog_enable[2]	0
XA0	Prog_enable[1]	0
BS1	Prog_enable[0]	0

Table 111. XA1 and XA0 Coding

XA1	XA0	Action when XTAL1 is Pulsed
0	0	Load Flash or EEPROM Address (High or low address byte determined by BS1)
0	1	Load Data (High or Low data byte for Flash determined by BS1)
1	0	Load Command
1	1	No Action, Idle

Table 112. Command Byte Bit Coding

Command Byte	Command Executed
1000 0000	Chip Erase
0100 0000	Write Fuse Bits
0010 0000	Write Lock Bits
0001 0000	Write Flash
0001 0001	Write EEPROM



263



Table 112. Command Byte Bit Coding

Command Byte	Command Executed	
0000 1000	Read Signature Bytes and Calibration byte	
0000 0100	Read Fuse and Lock bits	
0000 0010	Read Flash	
0000 0011	Read EEPROM	





Parallel Programming

Enter Programming Mode

The following algorithm puts the device in Parallel Programming mode:

- 1. Apply 4.5V 5.5V between V_{CC} and GND, and wait at least 100 $\mu s.$
- 2. Set RESET to "0" and toggle XTAL1 at least 6 times
- 3. Set the Prog_enable pins listed in Table 110 on page 263 to "0000" and wait at least 100 ns
- Apply 11.5V 12.5V to RESET. Any activity on Prog_enable pins within 100 ns after +12V has been applied to RESET, will cause the device to fail entering Programming mode.

Note, if External Crystal or External RC configuration is selected, it may not be possible to apply qualified XTAL1 pulses. In such cases, the following algorithm should be followed:

- 1. Set Prog_enable pins listed in Table 110 on page 263 to "0000".
- Apply 4.5V 5.5V between V_{CC} and GND simultaneously as 11.5V 12.5V is applied to RESET
- 3. Wait 100 µs.
- Re-program the fuses to ensure that External Clock is selected as clock source (CKSEL3:0 = 0b0000) If Lock bits are programmed, a Chip Erase command must be executed before changing the fuses.
- 5. Exit Programming mode by power the device down or by bringing RESET pin to 0b0.
- 6. Entering Programming mode with the original algorithm, as described above.

Considerations for Efficient Programming

The loaded command and address are retained in the device during programming. For efficient programming, the following should be considered.

- The command needs only be loaded once when writing or reading multiple memory locations.
- Skip writing the data value \$FF, that is the contents of the entire EEPROM (unless the EESAVE Fuse is programmed) and Flash after a Chip Erase.
- Address High byte needs only be loaded before programming or reading a new 256 word window in Flash or 256 byte EEPROM. This consideration also applies to Signature bytes reading.

Chip Erase

The Chip Erase will erase the Flash and EEPROM⁽¹⁾ memories plus Lock bits. The Lock bits are not reset until the program memory has been completely erased. The Fuse bits are not changed. A Chip Erase must be performed before the Flash and/or the EEPROM are reprogrammed.

Note: 1. The EEPRPOM memory is preserved during chip erase if the EESAVE Fuse is programmed.

Load Command "Chip Erase"

- 1. Set XA1, XA0 to "10". This enables command loading.
- 2. Set BS1 to "0".
- 3. Set DATA to "1000 0000". This is the command for Chip Erase.
- 4. Give XTAL1 a positive pulse. This loads the command.
- 5. Give $\overline{\text{WR}}$ a negative pulse. This starts the Chip Erase. RDY/ $\overline{\text{BSY}}$ goes low.
- 6. Wait until RDY/BSY goes high before loading a new command.



265



Programming the Flash

The Flash is organized in pages, see Table 107 on page 262. When programming the Flash, the program data is latched into a page buffer. This allows one page of program data to be programmed simultaneously. The following procedure describes how to program the entire Flash memory:

A. Load Command "Write Flash"

- 1. Set XA1, XA0 to "10". This enables command loading.
- 2. Set BS1 to "0".
- 3. Set DATA to "0001 0000". This is the command for Write Flash.
- 4. Give XTAL1 a positive pulse. This loads the command.
- B. Load Address Low byte
- 1. Set XA1, XA0 to "00". This enables address loading.
- 2. Set BS1 to "0". This selects low address.
- 3. Set DATA = Address Low byte (\$00 \$FF).
- 4. Give XTAL1 a positive pulse. This loads the address Low byte.
- C. Load Data Low Byte
- 1. Set XA1, XA0 to "01". This enables data loading.
- 2. Set DATA = Data Low byte (\$00 \$FF).
- 3. Give XTAL1 a positive pulse. This loads the data byte.
- D. Load Data High Byte
- 1. Set BS1 to "1". This selects high data byte.
- 2. Set XA1, XA0 to "01". This enables data loading.
- 3. Set DATA = Data High byte (\$00 \$FF).
- 4. Give XTAL1 a positive pulse. This loads the data byte.
- E. Latch Data
- 1. Set BS1 to "1". This selects high data byte.
- Give PAGEL a positive pulse. This latches the data bytes. (See Figure 129 for signal waveforms)
- F. Repeat B through E until the entire buffer is filled or until all data within the page is loaded.

While the lower bits in the address are mapped to words within the page, the higher bits address the pages within the FLASH. This is illustrated in Figure 128 on page 267. Note that if less than 8 bits are required to address words in the page (pagesize < 256), the most significant bit(s) in the address Low byte are used to address the page when performing a page write.

G. Load Address High byte

- 1. Set XA1, XA0 to "00". This enables address loading.
- 2. Set BS1 to "1". This selects high address.
- 3. Set DATA = Address High byte (\$00 \$FF).
- 4. Give XTAL1 a positive pulse. This loads the address High byte.
- H. Program Page
- 1. Set BS1 = "0"
- Give WR a negative pulse. This starts programming of the entire page of data. RDY/BSY goes low.
- 3. Wait until RDY/BSY goes high. (See Figure 129 for signal waveforms)

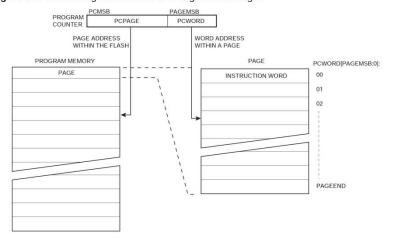


266



- I. Repeat B through H until the entire Flash is programmed or until all data has been programmed.
- J. End Page Programming
- 1. 1. Set XA1, XA0 to "10". This enables command loading.
- 2. Set DATA to "0000 0000". This is the command for No Operation.
- Give XTAL1 a positive pulse. This loads the command, and the internal write signals are reset

Figure 128. Addressing the Flash which is Organized in Pages



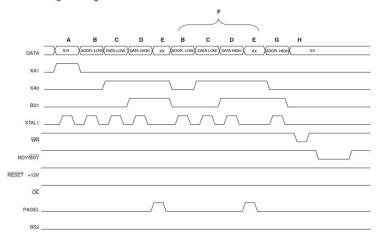
Note: 1. PCPAGE and PCWORD are listed in Table 107 on page 262.



267



Figure 129. Programming the Flash Waveforms⁽¹⁾



Note: 1. "XX" is don't care. The letters refer to the programming description above.

Programming the EEPROM

The EEPROM is organized in pages, see Table 108 on page 262. When programming the EEPROM, the program data is latched into a page buffer. This allows one page of data to be programmed simultaneously. The programming algorithm for the EEPROM data memory is as follows (refer to "Programming the Flash" on page 266 for details on Command, Address and Data loading):

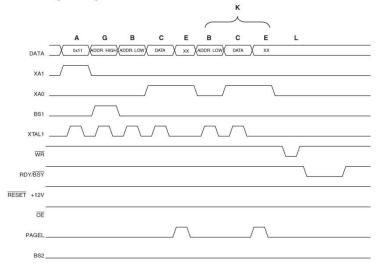
- 1. A: Load Command "0001 0001".
- 2. G: Load Address High Byte (\$00 \$FF)
- 3. B: Load Address Low Byte (\$00 \$FF)
- 4. C: Load Data (\$00 \$FF)
- 5. E: Latch data (give PAGEL a positive pulse)
- K: Repeat 3 through 5 until the entire buffer is filled
- L: Program EEPROM page
- 1. Set BS1 to "0".
- 2. Give $\overline{\text{WR}}$ a negative pulse. This starts programming of the EEPROM page. RDY/BSY goes low.
- 3. Wait until to RDY/BSY goes high before programming the next page. (See Figure 130 for signal waveforms)



268



Figure 130. Programming the EEPROM Waveforms



Reading the Flash

The algorithm for reading the Flash memory is as follows (refer to "Programming the Flash" on page 266 for details on Command and Address loading):

- 1. A: Load Command "0000 0010".
- 2. G: Load Address High Byte (\$00 \$FF)
- 3. B: Load Address Low Byte (\$00 \$FF)
- 4. Set $\overline{\text{OE}}$ to "0", and BS1 to "0". The Flash word Low byte can now be read at DATA.
- 5. Set BS1 to "1". The Flash word High byte can now be read at DATA.
- 6. Set OE to "1".

Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to "Programming the Flash" on page 266 for details on Command and Address loading):

- 1. A: Load Command "0000 0011".
- 2. G: Load Address High Byte (\$00 \$FF)
- 3. B: Load Address Low Byte (\$00 \$FF)
- 4. Set $\overline{\text{OE}}$ to "0", and BS1 to "0". The EEPROM Data byte can now be read at DATA.
- Set OE to "1".

Programming the Fuse Low Bits

The algorithm for programming the Fuse Low bits is as follows (refer to "Programming the Flash" on page 266 for details on Command and Data loading):

- 1. A: Load Command "0100 0000".
- 2. C: Load Data Low Byte. Bit n = "0" programs and bit n = "1" erases the Fuse bit.
- 3. Set BS1 to "0" and BS2 to "0".
- 4. Give WR a negative pulse and wait for RDY/BSY to go high.

<u>AIMEL</u>

269

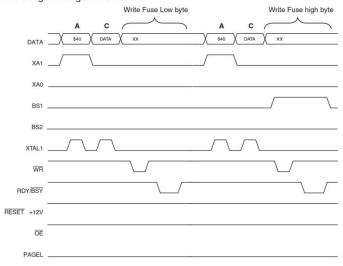


Programming the **Fuse High Bits**

The algorithm for programming the Fuse high bits is as follows (refer to "Programming the Flash" on page 266 for details on Command and Data loading):

- 1. A: Load Command "0100 0000".
- 2. C: Load Data Low Byte. Bit n = "0" programs and bit n = "1" erases the Fuse bit.
- 3. Set BS1 to "1" and BS2 to "0". This selects high data byte.
- 4. Give WR a negative pulse and wait for RDY/BSY to go high.
- Set BS1 to "0". This selects low data byte.

Figure 131. Programming the Fuses



Bits

Programming the Lock The algorithm for programming the Lock bits is as follows (refer to "Programming the Flash" on page 266 for details on Command and Data loading):

- 1. A: Load Command "0010 0000".
- 2. C: Load Data Low Byte. Bit n = "0" programs the Lock bit.
- 3. Give WR a negative pulse and wait for RDY/BSY to go high.

The Lock bits can only be cleared by executing Chip Erase.

Reading the Fuse and **Lock Bits**

The algorithm for reading the Fuse and Lock bits is as follows (refer to "Programming the Flash" on page 266 for details on Command loading):

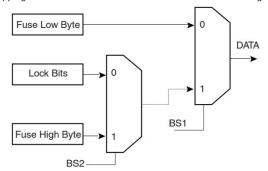
- 1. A: Load Command "0000 0100".
- Set $\overline{\text{OE}}$ to "0", BS2 to "0" and BS1 to "0". The status of the Fuse Low bits can now be read at DATA ("0" means programmed).
- Set OE to "0", BS2 to "1" and BS1 to "1". The status of the Fuse High bits can now be read at DATA ("0" means programmed).
- Set $\overline{\text{OE}}$ to "0", BS2 to "0" and BS1 to "1". The status of the Lock bits can now be read at DATA ("0" means programmed).
- 5. Set OE to "1".



270



Figure 132. Mapping between BS1, BS2 and the Fuse- and Lock Bits during Read



Bytes

Reading the Signature The algorithm for reading the Signature bytes is as follows (refer to "Programming the Flash" on page 266 for details on Command and Address loading):

- 1. A: Load Command "0000 1000".
- 2. B: Load Address Low Byte (\$00 \$02).
- 3. Set $\overline{\text{OE}}$ to "0", and BS1 to "0". The selected Signature byte can now be read at DATA.
- 4. Set OE to "1".

Reading the **Calibration Byte**

The algorithm for reading the Calibration byte is as follows (refer to "Programming the Flash" on page 266 for details on Command and Address loading):

- 1. A: Load Command "0000 1000".
- 2. B: Load Address Low Byte, \$00.
- 3. Set $\overline{\text{OE}}$ to "0", and BS1 to "1". The Calibration byte can now be read at DATA.
- Set OE to "1".

Parallel Programming Characteristics

Figure 133. Parallel Programming Timing, Including some General Timing Requirements

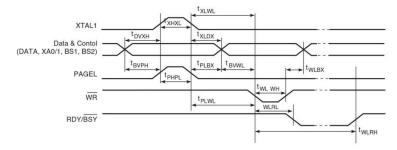
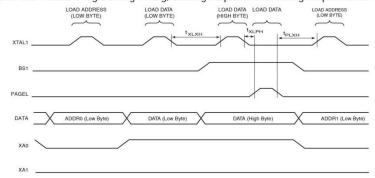


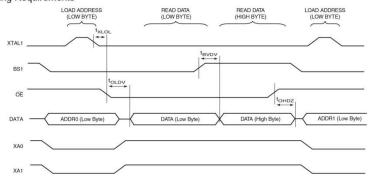


Figure 134. Parallel Programming Timing, Loading Sequence with Timing Requirements⁽¹⁾



Note: 1. The timing requirements shown in Figure 133 (that is, t_{DVXH} , t_{XHXL} , and t_{XLDX}) also apply to loading operation.

 $\begin{tabular}{ll} \textbf{Figure 135.} & Parallel Programming Timing, Reading Sequence (within the Same Page) with Timing Requirements $^{(1)}$ \\ \end{tabular}$



Note: 1. The timing requirements shown in Figure 133 (that is, t_{DVXH} , t_{XHXL} , and t_{XLDX}) also apply to reading operation.

Table 113. Parallel Programming Characteristics, $V_{CC} = 5V \pm 10\%$

Symbol	Parameter	Min	Тур	Max	Units
V _{PP}	Programming Enable Voltage	11.5		12.5	V
I _{PP}	Programming Enable Current			250	μА

AMEL

272



Table 113. Parallel Programming Characteristics, $V_{CC} = 5V \pm 10\%$ (Continued)

Symbol	Parameter	Min	Тур	Max	Units
t _{DVXH}	Data and Control Valid before XTAL1 High	67			
t _{XLXH}	XTAL1 Low to XTAL1 High	200			
t _{XHXL}	XTAL1 Pulse Width High	150			
t _{XLDX}	Data and Control Hold after XTAL1 Low	67			
t _{XLWL}	XTAL1 Low to WR Low	0			
t _{XLPH}	XTAL1 Low to PAGEL high	0			
t _{PLXH}	PAGEL low to XTAL1 high	150			
t _{BVPH}	BS1 Valid before PAGEL High	67			ns
t _{PHPL}	PAGEL Pulse Width High	150			
t _{PLBX}	BS1 Hold after PAGEL Low	67			
t _{WLBX}	BS2/1 Hold after WR Low	67			
t _{PLWL}	PAGEL Low to WR Low	67			
t _{BVWL}	BS1 Valid to WR Low	67			
t _{WLWH}	WR Pulse Width Low	150			
t _{WLRL}	WR Low to RDY/BSY Low	0		1	μs
t _{WLRH}	WR Low to RDY/BSY High ⁽¹⁾	3.7		4.5	
t _{WLRH_CE}	WR Low to RDY/BSY High for Chip Erase ⁽²⁾	7.5		9	ms
t _{XLOL}	XTAL1 Low to OE Low	0			
t _{BVDV}	BS1 Valid to DATA valid	0		250	
t _{OLDV}				250	ns
t _{OHDZ}	OE High to DATA Tri-stated			250	

Notes: 1. t_{WLRH} is valid for the Write Flash, Write EEPROM, Write Fuse bits and Write Lock bits commands.

2. t_{WLRH_CE} is valid for the Chip Erase command.

Serial Downloading

Both the Flash and EEPROM memory arrays can be programmed using the serial SPI bus while RESET is pulled to GND. The serial interface consists of pins SCK, MOSI (input), and MISO (output). After RESET is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed. NOTE, in Table 114 on page 273, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

SPI Serial Programming Pin Mapping

Table 114. Pin Mapping SPI Serial Programming

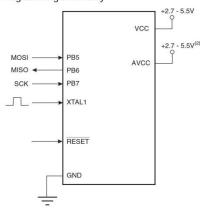
Symbol	Pins	I/O	Description	
MOSI	PB5	1	Serial Data in	
MISO	PB6	0	Serial Data out	
SCK	PB7	1	Serial Clock	

AIMEL

273



Figure 136. SPI Serial Programming and Verify(1)



Notes: 1. If the device is clocked by the Internal Oscillator, it is no need to connect a clock source to the XTAL1 pin.

2. V_{CC} -0.3V < AVCC < V_{CC} +0.3V, however, AVCC should always be within 2.7V - 5.5V

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the serial mode ONLY) and there is no need to first execute the Chip Erase instruction. The Chip Erase operation turns the content of every memory location in both the Program and EEPROM arrays into \$FF.

Depending on CKSEL Fuses, a valid clock must be present. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 2 CPU clock cycles for f_{ck} < 12 MHz, 3 CPU clock cycles for $f_{ck} \geq$ 12 MHz

High: > 2 CPU clock cycles for f_{ck} < 12 MHz, 3 CPU clock cycles for f_{ck} \geq 12 MHz

SPI Serial Programming Algorithm When writing serial data to the ATmega16, data is clocked on the rising edge of SCK.

When reading data from the ATmega16, data is clocked on the falling edge of SCK. See Figure 138 for timing details.

To program and verify the ATmega16 in the SPI Serial Programming mode, the following sequence is recommended (See four byte instruction formats in Figure 116 on page 276):

- 1. Power-up sequence:
 - Apply power between V_{CC} and GND while \overline{RESET} and SCK are set to "0". In some systems, the programmer can not guarantee that SCK is held low during power-up. In this case, \overline{RESET} must be given a positive pulse of at least two CPU clock cycles duration after SCK has been set to "0".
- Wait for at least 20 ms and enable SPI Serial Programming by sending the Programming Enable serial instruction to pin MOSI.
- 3. The SPI Serial Programming instructions will not work if the communication is out of synchronization. When in sync. the second byte (\$53), will echo back when issuing the third byte of the Programming Enable instruction. Whether the echo is correct or not, all four bytes of the instruction must be transmitted. If the \$53 did not echo back, give RESET a positive pulse and issue a new Programming Enable command.



274



——— ATmega16(L)

- 4. The Flash is programmed one page at a time. The page size is found in Table 107 on page 262. The memory page is loaded one byte at a time by supplying the 6 LSB of the address and data together with the Load Program Memory Page instruction. To ensure correct loading of the page, the data Low byte must be loaded before data High byte is applied for a given address. The Program Memory Page is stored by loading the Write Program Memory Page instruction with the 7 MSB of the address. If polling is not used, the user must wait at least t_{WD_FLASH} before issuing the next page. (See Table 115). Accessing the SPI Serial Programming interface before the Flash write operation completes can result in incorrect programming.
- 5. The EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. An EEPROM memory location is first automatically erased before new data is written. If polling is not used, the user must wait at least t_{WD_EEPROM} before issuing the next byte. (See Table 115). In a chip erased device, no \$FFs in the data file(s) need to be programmed.
- Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO.
- At the end of the programming session, RESET can be set high to commence normal operation.
- Power-off sequence (if needed): Set RESET to "1". Turn V_{CC} power off.

Data Polling Flash

When a page is being programmed into the Flash, reading an address location within the page being programmed will give the value \$FF. At the time the device is ready for a new page, the programmed value will read correctly. This is used to determine when the next page can be written. Note that the entire page is written simultaneously and any address within the page can be used for polling. Data polling of the Flash will not work for the value \$FF, so when programming this value, the user will have to wait for at least t_{WD_FLASH} before programming the next page. As a chip erased device contains \$FF in all locations, programming of addresses that are meant to contain \$FF, can be skipped. See Table 115 for t_{WD_FLASH} value

Data Polling EEPROM

When a new byte has been written and is being programmed into EEPROM, reading the address location being programmed will give the value \$FF. At the time the device is ready for a new byte, the programmed value will read correctly. This is used to determine when the next byte can be written. This will not work for the value \$FF, but the user should have the following in mind: As a chip erased device contains \$FF in all locations, programming of addresses that are meant to contain \$FF, can be skipped. This does not apply if the EEPROM is re-programmed without chip erasing the device. In this case, data polling cannot be used for the value \$FF, and the user will have to wait at least $t_{\rm WD_EEPROM}$ before programming the next byte. See Table 115 for $t_{\rm WD_EEPROM}$ value.





Table 115. Minimum Wait Delay before Writing the Next Flash or EEPROM Location

Symbol	Minimum Wait Delay
t _{WD_FUSE}	4.5 ms
t _{WD_FLASH}	4.5 ms
t _{WD_EEPROM}	9.0 ms
t _{WD_ERASE}	9.0 ms

Serial Programming Instruction set

Table 116 on page 276 and Figure 137 on page 277 describes the Instruction set.

Table 116. Serial Programming Instruction Set (Hexadecimal values)

	Instruction Format						
Instruction ⁽¹⁾ /Operation	Byte 1	Byte 2	Byte 3	Byte 4			
Programming Enable	\$AC	\$53	\$00	\$00			
Chip Erase (Program Memory/EEPROM)	\$AC	\$80	\$00	\$00			
Poll RDY/BSY	\$FO	\$00	\$00	data byte out			
Load Instructions							
Load Extended Address byte ⁽¹⁾	\$4D	\$00	Extended adr	\$00			
Load Program Memory Page, High byte	\$48	adr MSB	adr LSB	high data byte in			
Load Program Memory Page, Low byte	\$40	adr MSB	adr LSB	low data byte in			
Load EEPROM Memory Page (page access) ⁽¹⁾	\$C1	\$00	adr LSB	data byte in			
Read Instructions				'			
Read Program Memory, High byte	\$28	adr MSB	adr LSB	high data byte out			
Read Program Memory, Low byte	\$20	adr MSB	adr LSB	low data byte out			
Read EEPROM Memory	\$A0	adr MSB	adr LSB	data byte out			
Read Lock bits	\$58	\$00	\$00	data byte out			
Read Signature Byte	\$30	\$00	0000 000aa	data byte out			
Read Fuse bits	\$50	\$00	\$00	data byte out			
Read Fuse High bits	\$58	\$08	\$00	data byte out			
Read Extended Fuse Bits	\$50	\$08	\$00	data byte out			
Read Calibration Byte	\$38	\$00	\$0b00 000bb	data byte out			
Write Instructions				1			
Write Program Memory Page	\$4C	000a aaaa	aa00 0000	\$00			
Write EEPROM Memory	\$C0	adr MSB	adr LSB	data byte in			
Write EEPROM Memory Page (page access)(1)	\$C2	adr MSB	adr LSB	\$00			
Write Lock bits	\$AC	\$E0	\$00	data byte in			
Write Fuse bits	\$AC	\$A0	\$00	data byte in			
Write Fuse High bits	\$AC	\$A8	\$00	data byte in			
Write Extended Fuse Bits	\$AC	\$A4	\$00	data byte in			



276



Notes: 1. Not all instructions are applicable for all parts.

- 2. a = address
- 3. Bits are programmed '0', unprogrammed '1'.
- 4. To ensure future compatibility, unused Fuses and Lock bits should be unprogrammed ('1') .
- Refer to the correspondig section for Fuse and Lock bits, Calibration and Signature bytes and Page size.
- 6. See htt://www.atmel.com/avr for Application Notes regarding programming and programmers.

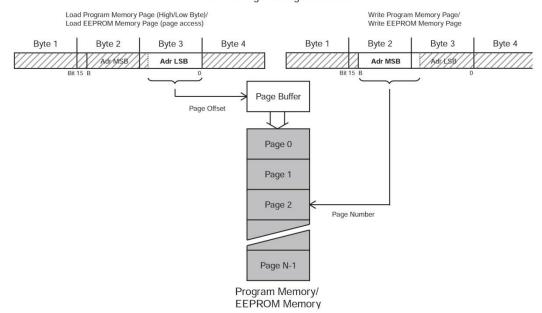
If the LSB in RDY/BSY data byte out is '1', a programming operation is still pending. Wait until this bit returns '0' before the next instruction is carried out.

Within the same page, the low data byte must be loaded prior to the high data byte.

After data is loaded to the page buffer, program the EEPROM page, see Figure 137 on page 277.

Figure 137. Serial Programming Instruction example

Serial Programming Instruction



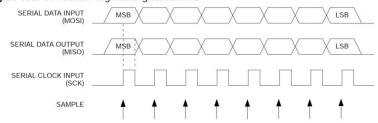
AIMEL

277



SPI Serial Programming Characteristics For characteristics of the SPI module, see "SPI Timing Characteristics" on page 295.

Figure 138. SPI Serial Programming Waveforms



Programming via the JTAG Interface

Programming through the JTAG interface requires control of the four JTAG specific pins: TCK, TMS, TDI and TDO. Control of the reset and clock pins is not required.

To be able to use the JTAG interface, the JTAGEN Fuse must be programmed. The device is default shipped with the fuse programmed. In addition, the JTD bit in MCUCSR must be cleared. Alternatively, if the JTD bit is set, the External Reset can be forced low. Then, the JTD bit will be cleared after two chip clocks, and the JTAG pins are available for programming. This provides a means of using the JTAG pins as normal port pins in running mode while still allowing In-System Programming via the JTAG interface. Note that this technique can not be used when using the JTAG pins for Boundary-scan or On-chip Debug. In these cases the JTAG pins must be dedicated for this purpose.

As a definition in this datasheet, the LSB is shifted in and out first of all Shift Registers.

Programming Specific JTAG Instructions

The instruction register is 4-bit wide, supporting up to 16 instructions. The JTAG instructions useful for Programming are listed below.

The OPCODE for each instruction is shown behind the instruction name in hex format. The text describes which Data Register is selected as path between TDI and TDO for each instruction.

The Run-Test/Idle state of the TAP controller is used to generate internal clocks. It can also be used as an idle state between JTAG sequences. The state machine sequence for changing the instruction word is shown in Figure 139.





Test-Logic-Reset 0 Run-Test/Idle Select-DR Scan Select-IR Scan 0 0 Capture-DR Capture-IR 0 0 Shift-IR Shift-DR 0 Exit1-DR Exit1-IR 0 0 Pause-DR Pause-IR Exit2-DR Exit2-IR Update-DR Update-IR 0 0

Figure 139. State Machine Sequence for Changing the Instruction Word

AVR_RESET (\$C)

The AVR specific public JTAG instruction for setting the AVR device in the Reset mode or taking the device out from the Reset Mode. The TAP controller is not reset by this instruction. The one bit Reset Register is selected as Data Register. Note that the Reset will be active as long as there is a logic "one" in the Reset Chain. The output from this chain is not latched.

The active states are:

· Shift-DR: The Reset Register is shifted by the TCK input.

PROG_ENABLE (\$4)

The AVR specific public JTAG instruction for enabling programming via the JTAG port. The 16-bit Programming Enable Register is selected as Data Register. The active states are the following:

- · Shift-DR: The programming enable signature is shifted into the Data Register.
- Update-DR: The programming enable signature is compared to the correct value, and Programming mode is entered if the signature is valid.

<u>AIMEL</u>

279



PROG_COMMANDS (\$5)

The AVR specific public JTAG instruction for entering programming commands via the JTAG port. The 15-bit Programming Command Register is selected as Data Register. The active states are the following:

- · Capture-DR: The result of the previous command is loaded into the Data Register.
- Shift-DR: The Data Register is shifted by the TCK input, shifting out the result of the previous command and shifting in the new command.
- · Update-DR: The programming command is applied to the Flash inputs
- Run-Test/Idle: One clock cycle is generated, executing the applied command (not always required, see Table 117 below).

PROG_PAGELOAD (\$6)

The AVR specific public JTAG instruction to directly load the Flash data page via the JTAG port. The 1024 bit Virtual Flash Page Load Register is selected as Data Register. This is a virtual scan chain with length equal to the number of bits in one Flash page. Internally the Shift Register is 8-bit. Unlike most JTAG instructions, the Update-DR state is not used to transfer data from the Shift Register. The data are automatically transferred to the Flash page buffer byte by byte in the Shift-DR state by an internal state machine. This is the only active state:

 Shift-DR: Flash page data are shifted in from TDI by the TCK input, and automatically loaded into the Flash page one byte at a time.

Note: The JTAG instruction PROG_PAGELOAD can only be used if the AVR device is the first device in JTAG scan chain. If the AVR cannot be the first device in the scan chain, the byte-wise programming algorithm must be used.

PROG_PAGEREAD (\$7)

The AVR specific public JTAG instruction to read one full Flash data page via the JTAG port. The 1032 bit Virtual Flash Page Read Register is selected as Data Register. This is a virtual scan chain with length equal to the number of bits in one Flash page plus 8. Internally the Shift Register is 8-bit. Unlike most JTAG instructions, the Capture-DR state is not used to transfer data to the Shift Register. The data are automatically transferred from the Flash page buffer byte by byte in the Shift-DR state by an internal state machine. This is the only active state:

 Shift-DR: Flash data are automatically read one byte at a time and shifted out on TDO by the TCK input. The TDI input is ignored.

Note: The JTAG instruction PROG_PAGEREAD can only be used if the AVR device is the first device in JTAG scan chain. If the AVR cannot be the first device in the scan chain, the byte-wise programming algorithm must be used.

Data Registers

The Data Registers are selected by the JTAG Instruction Registers described in section "Programming Specific JTAG Instructions" on page 278. The Data Registers relevant for programming operations are:

- Reset Register
- · Programming Enable Register
- · Programming Command Register
- Virtual Flash Page Load Register
- Virtual Flash Page Read Register





Reset Register

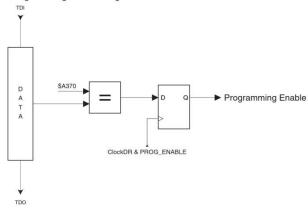
The Reset Register is a Test Data Register used to reset the part during programming. It is required to reset the part before entering programming mode.

A high value in the Reset Register corresponds to pulling the external Reset low. The part is reset as long as there is a high value present in the Reset Register. Depending on the Fuse settings for the clock options, the part will remain reset for a Reset Time-out Period (refer to "Clock Sources" on page 25) after releasing the Reset Register. The output from this Data Register is not latched, so the reset will take place immediately, as shown in Figure 115 on page 230.

Programming Enable Register

The Programming Enable Register is a 16-bit register. The contents of this register is compared to the programming enable signature, binary code 1010_0011_0111_0000. When the contents of the register is equal to the programming enable signature, programming via the JTAG port is enabled. The register is reset to 0 on Power-on Reset, and should always be reset when leaving Programming mode.

Figure 140. Programming Enable Register



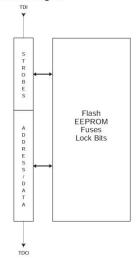


281



Programming Command Register The Programming Command Register is a 15-bit register. This register is used to serially shift in programming commands, and to serially shift out the result of the previous command, if any. The JTAG Programming Instruction Set is shown in Table 117. The state sequence when shifting in the programming commands is illustrated in Figure 142.

Figure 141. Programming Command Register





282



Table 117.JTAG Programming Instruction Set \mathbf{a} = address high bits, \mathbf{b} = address low bits, \mathbf{H} = 0 – Low byte, 1 – High Byte, \mathbf{o} = data out, \mathbf{i} = data in, \mathbf{x} = don't care

Instruction	TDI sequence	TDO sequence	Notes
1a. Chip erase	0100011_10000000 0110001_10000000 0110011_10000000 0110011_10000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxx	
1b. Poll for chip erase complete	0110011_10000000	xxxxx o x_xxxxxxx	(2)
2a. Enter Flash Write	0100011_00010000	xxxxxxx_xxxxxxxx	
2b. Load Address High Byte	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	(9)
2c. Load Address Low Byte	0000011_ bbbbbbb	xxxxxxx_xxxxxxxx	
2d. Load Data Low Byte	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	
2e. Load Data High Byte	0010111_iiiiiiii	xxxxxxx_xxxxxxxx	
2f. Latch Data	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
2g. Write Flash Page	0110111_00000000 0110101_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxx	(1)
2h. Poll for Page Write complete	0110111_00000000	xxxxx o x_xxxxxxxx	(2)
3a. Enter Flash Read	0100011_00000010	xxxxxxx_xxxxxxxx	
3b. Load Address High Byte	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	(9)
3c. Load Address Low Byte	0000011_ bbbbbbb	xxxxxxx_xxxxxxxx	
3d. Read Data Low and High Byte	0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_00000000 xxxxxxxx	Low byte High byte
4a. Enter EEPROM Write	0100011_00010001	xxxxxxx_xxxxxxxx	
4b. Load Address High Byte	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	(9)
4c. Load Address Low Byte	0000011_ bbbbbbb	xxxxxxx_xxxxxxxx	
4d. Load Data Byte	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	
4e. Latch Data	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
4f. Write EEPROM Page	0110011_00000000 0110001_00000000 0110011_00000000	XXXXXXX_XXXXXXXX XXXXXXX_XXXXXXXX XXXXXX	(1)
4g. Poll for Page Write complete	0110011_00000000	xxxxx o x_xxxxxxx	(2)
5a. Enter EEPROM Read	0100011_00000011	xxxxxxx_xxxxxxxx	
5b. Load Address High Byte	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	(9)
5c. Load Address Low Byte	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	



283



Table 117.JTAG Programming Instruction Set (Continued) \mathbf{a} = address high bits, \mathbf{b} = address low bits, \mathbf{H} = 0 – Low byte, 1 – High Byte, \mathbf{o} = data out, \mathbf{i} = data in, \mathbf{x} = don't care

Instruction	TDI sequence	TDO sequence	Notes
5d. Read Data Byte	0110011_ bbbbbbbb	xxxxxxx_xxxxxxxx	
	0110010_00000000	XXXXXXX_XXXXXXXX	
	0110011_00000000	XXXXXXX_00000000	
6a. Enter Fuse Write	0100011_01000000	xxxxxxx_xxxxxxxx	
6b. Load Data Low Byte ⁽⁶⁾	0010011 _iiiiiiii	xxxxxxx_xxxxxxxx	(3)
6c. Write Fuse High byte	0110111_00000000	xxxxxxx_xxxxxxxx	(1)
	0110101_00000000	xxxxxxx_xxxxxxxx	200 200
	0110111_00000000	xxxxxxx_xxxxxxxx	
	0110111_00000000	XXXXXXX_XXXXXXX	
6d. Poll for Fuse Write complete	0110111_00000000	xxxxx o x_xxxxxxx	(2)
6e. Load Data Low Byte ⁽⁷⁾	0010011 _iiiiiiii	xxxxxxx_xxxxxxxx	(3)
6f. Write Fuse Low byte	0110011_00000000	xxxxxxx_xxxxxxxx	(1)
	0110001_00000000	xxxxxxx_xxxxxxxx	
	0110011_00000000	xxxxxxx_xxxxxxxx	
	0110011_00000000	xxxxxxx_xxxxxxx	
6g. Poll for Fuse Write complete	0110011_00000000	xxxxx o x_xxxxxxx	(2)
7a. Enter Lock Bit Write	0100011_00100000	xxxxxxx_xxxxxxxx	
7b. Load Data Byte ⁽⁸⁾	0010011_11 iiiiii	xxxxxxx_xxxxxxxx	(4)
7c. Write Lock Bits	0110011_00000000	xxxxxxx_xxxxxxxx	(1)
	0110001_00000000	xxxxxxx_xxxxxxxx	100.00
	0110011_00000000	xxxxxxx_xxxxxxxx	
	0110011_00000000	XXXXXXX_XXXXXXXX	
7d. Poll for Lock Bit Write complete	0110011_00000000	xxxxx o x_xxxxxxx	(2)
8a. Enter Fuse/Lock Bit Read	0100011_00000100	xxxxxxx_xxxxxxxx	
8b. Read Fuse High Byte ⁽⁶⁾	0111110_00000000	xxxxxxx_xxxxxxxx	
1000 - 04P	0111111_00000000	xxxxxxx_oooooooo	
8c. Read Fuse Low Byte ⁽⁷⁾	0110010_00000000	xxxxxxx_xxxxxxxx	
	0110011_00000000	XXXXXXX_00000000	
8d. Read Lock Bits ⁽⁸⁾	0110110_00000000	xxxxxxx_xxxxxxxx	(5)
	0110111_00000000	xxxxxxx_xxooooo	ARTON
8e. Read Fuses and Lock Bits	0111110_00000000	xxxxxxx_xxxxxxxx	(5)
	0110010_00000000	xxxxxxx_oooooooo	Fuse High Byte
	0110110_00000000	xxxxxxx_oooooooo	Fuse Low Byte
	0110111_00000000	xxxxxxx_oooooooo	Lock bits
9a. Enter Signature Byte Read	0100011_00001000	xxxxxxx_xxxxxxxx	
9b. Load Address Byte	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
9c. Read Signature Byte	0110010_00000000	xxxxxxx_xxxxxxxx	
	0110011_00000000	xxxxxxx_oooooooo	
10a. Enter Calibration Byte Read	0100011_00001000	xxxxxxx_xxxxxxxx	



284



Table 117. JTAG Programming Instruction Set (Continued)

a = address high bits, b = address low bits, H = 0 - Low byte, I - High Byte, I - Byte, I

Instruction	TDI sequence	TDO sequence	Notes
10b. Load Address Byte	0000011_ bbbbbbb	xxxxxxx_xxxxxxxx	
10c. Read Calibration Byte	0110110_0000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_ ooooooo	
11a. Load No Operation Command	0100011_00000000 0110011_00000000	xxxxxxx_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	

Notes: 1. This command sequence is not required if the seven MSB are correctly set by the previous command sequence (which is normally the case).

2. Repeat until **o** = "1".

- Set bits to "0" to program the corresponding fuse, "1" to unprogram the fuse.
 Set bits to "0" to program the corresponding lock bit, "1" to leave the lock bit unchanged.
 "0" = programmed, "1" = unprogrammed.
 The bit mapping for Fuses High byte is listed in Table 105 on page 260

- 7. The bit mapping for Fuses Low byte is listed in Table 106 on page 261
- 8. The bit mapping for Lock bits byte is listed in Table 103 on page 259
- 9. Address bits exceeding PCMSB and EEAMSB (Table 107 and Table 108) are don't care





Test-Logic-Reset ◀ 0 Run-Test/Idle Select-DR Scan Select-IR Scan 0 0 Capture-DR Capture-IR 0 0 Shift-DR Shift-IR Exit1-DR Exit1-IR 0 0 Pause-DR Pause-IR 1 Exit2-IR Exit2-DR 1 Update-DR Update-IR 0 0

Figure 142. State Machine Sequence for Changing/Reading the Data Word

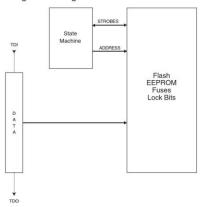
Virtual Flash Page Load Register The Virtual Flash Page Load Register is a virtual scan chain with length equal to the number of bits in one Flash page. Internally the Shift Register is 8-bit, and the data are automatically transferred to the Flash page buffer byte by byte. Shift in all instruction words in the page, starting with the LSB of the first instruction in the page and ending with the MSB of the last instruction in the page. This provides an efficient way to load the entire Flash page buffer before executing Page Write.

AIMEL

286

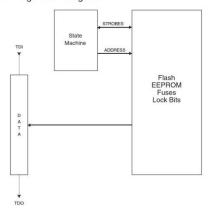


Figure 143. Virtual Flash Page Load Register



Virtual Flash Page Read Register The Virtual Flash Page Read Register is a virtual scan chain with length equal to the number of bits in one Flash page plus 8. Internally the Shift Register is 8-bit, and the data are automatically transferred from the Flash data page byte by byte. The first 8 cycles are used to transfer the first byte to the internal Shift Register, and the bits that are shifted out during these 8 cycles should be ignored. Following this initialization, data are shifted out starting with the LSB of the first instruction in the page and ending with the MSB of the last instruction in the page. This provides an efficient way to read one full Flash page to verify programming.

Figure 144. Virtual Flash Page Read Register





287



_____ ATmega16(L)

Programming Algorithm

All references below of type "1a", "1b", and so on, refer to Table 117.

Entering Programming Mode

- 1. Enter JTAG instruction AVR RESET and shift 1 in the Reset Register.
- Enter instruction PROG_ENABLE and shift 1010_0011_0111_0000 in the Programming Enable Register.

Leaving Programming Mode

- 1. Enter JTAG instruction PROG_COMMANDS.
- 2. Disable all programming instructions by using no operation instruction 11a.
- Enter instruction PROG_ENABLE and shift 0000_0000_0000 in the programming Enable Register.
- 4. Enter JTAG instruction AVR_RESET and shift 0 in the Reset Register.

- Performing Chip Erase 1. Enter JTAG instruction PROG_COMMANDS.
 - 2. Start chip erase using programming instruction 1a.
 - 3. Poll for Chip Erase complete using programming instruction 1b, or wait for tweether (refer to Table 113 on page 272).

Programming the Flash

Before programming the Flash a Chip Erase must be performed. See "Performing Chip Erase" on page 288.

- 1. Enter JTAG instruction PROG_COMMANDS.
- 2. Enable Flash write using programming instruction 2a.
- 3. Load address High byte using programming instruction 2b.
- 4. Load address Low byte using programming instruction 2c.
- 5. Load data using programming instructions 2d, 2e and 2f.
- 6. Repeat steps 4 and 5 for all instruction words in the page.
- 7. Write the page using programming instruction 2g.
- 8. Poll for Flash write complete using programming instruction 2h, or wait for twelf (refer to Table 113 on page 272).
- 9. Repeat steps 3 to 7 until all data have been programmed.

A more efficient data transfer can be achieved using the PROG_PAGELOAD instruction:

- 1. Enter JTAG instruction PROG_COMMANDS.
- 2. Enable Flash write using programming instruction 2a.
- Load the page address using programming instructions 2b and 2c. PCWORD (refer to Table 107 on page 262) is used to address within one page and must be written as 0.
- 4. Enter JTAG instruction PROG_PAGELOAD.
- Load the entire page by shifting in all instruction words in the page, starting with the LSB of the first instruction in the page and ending with the MSB of the last instruction in the
- 6. Enter JTAG instruction PROG_COMMANDS.
- 7. Write the page using programming instruction 2g.
- Poll for Flash write complete using programming instruction 2h, or wait for twi RH (refer to Table 113 on page 272).
- 9. Repeat steps 3 to 8 until all data have been programmed.



288



Reading the Flash

- Enter JTAG instruction PROG_COMMANDS.
- 2. Enable Flash read using programming instruction 3a.
- 3. Load address using programming instructions 3b and 3c.
- 4. Read data using programming instruction 3d.
- 5. Repeat steps 3 and 4 until all data have been read.

A more efficient data transfer can be achieved using the PROG_PAGEREAD instruction:

- 1. Enter JTAG instruction PROG_COMMANDS.
- 2. Enable Flash read using programming instruction 3a.
- Load the page address using programming instructions 3b and 3c. PCWORD (refer to Table 107 on page 262) is used to address within one page and must be written as 0.
- 4. Enter JTAG instruction PROG_PAGEREAD.
- Read the entire page by shifting out all instruction words in the page, starting with the LSB of the first instruction in the page and ending with the MSB of the last instruction in the page. Remember that the first 8 bits shifted out should be ignored.
- 6. Enter JTAG instruction PROG_COMMANDS.
- 7. Repeat steps 3 to 6 until all data have been read.

Programming the EEPROM

Before programming the EEPROM a Chip Erase must be performed. See "Performing Chip Erase" on page 288.

- 1. Enter JTAG instruction PROG_COMMANDS.
- 2. Enable EEPROM write using programming instruction 4a.
- 3. Load address High byte using programming instruction 4b.
- 4. Load address Low byte using programming instruction 4c.
- 5. Load data using programming instructions 4d and 4e.
- Repeat steps 4 and 5 for all data bytes in the page.
- 7. Write the data using programming instruction 4f.
- 8. Poll for EEPROM write complete using programming instruction 4g, or wait for t_{WLRH} (refer to Table 113 on page 272).
- 9. Repeat steps 3 to 8 until all data have been programmed.

Note that the PROG_PAGELOAD instruction can not be used when programming the EEPROM

Reading the EEPROM

- 1. Enter JTAG instruction PROG_COMMANDS.
- 2. Enable EEPROM read using programming instruction 5a.
- 3. Load address using programming instructions 5b and 5c.
- 4. Read data using programming instruction 5d.
- 5. Repeat steps 3 and 4 until all data have been read.

Note that the PROG_PAGEREAD instruction can not be used when reading the EEPROM



289



Programming the **Fuses**

- 1. Enter JTAG instruction PROG_COMMANDS.
- 2. Enable Fuse write using programming instruction 6a.
- 3. Load data High byte using programming instructions 6b. A bit value of "0" will program the corresponding fuse, a "1" will unprogram the fuse.
- 4. Write Fuse High byte using programming instruction 6c.
- 5. Poll for Fuse write complete using programming instruction 6d, or wait for t_{WLRH} (refer to Table 113 on page 272).
- 6. Load data Low byte using programming instructions 6e. A "0" will program the fuse, a "1" will unprogram the fuse.
- 7. Write Fuse Low byte using programming instruction 6f.
- Poll for Fuse write complete using programming instruction 6g, or wait for t_{WLRH} (refer to Table 113 on page 272).

- Programming the Lock 1. Enter JTAG instruction PROG_COMMANDS.
 - Enable Lock bit write using programming instruction 7a.
 - Load data using programming instructions 7b. A bit value of "0" will program the corresponding Lock bit, a "1" will leave the Lock bit unchanged.
 - 4. Write Lock bits using programming instruction 7c.
 - Poll for Lock bit write complete using programming instruction 7d, or wait for t_{WLRH} (refer to Table 113 on page 272).

Reading the Fuses and Lock Bits

- 1. Enter JTAG instruction PROG_COMMANDS.
- 2. Enable Fuse/Lock bit read using programming instruction 8a.
- To read all Fuses and Lock bits, use programming instruction 8e. To only read Fuse High byte, use programming instruction 8b. To only read Fuse Low byte, use programming instruction 8c. To only read Lock bits, use programming instruction 8d.

Bytes

- Reading the Signature 1. Enter JTAG instruction PROG_COMMANDS.
 - Enable Signature byte read using programming instruction 9a.
 - 3. Load address \$00 using programming instruction 9b.
 - Read first signature byte using programming instruction 9c.
 - Repeat steps 3 and 4 with address \$01 and address \$02 to read the second and third signature bytes, respectively.

Reading the **Calibration Byte**

- 1. Enter JTAG instruction PROG COMMANDS.
- 2. Enable Calibration byte read using programming instruction 10a.
- 3. Load address \$00 using programming instruction 10b.
- 4. Read the calibration byte using programming instruction 10c.



290



Electrical Characteristics

Absolute Maximum Ratings*

Operating Temperature55°C to +125°C
Storage Temperature65°C to +150°C
Voltage on any Pin except RESET with respect to Ground0.5V to V _{CC} +0.5V
Voltage on RESET with respect to Ground0.5V to +13.0V
Maximum Operating Voltage 6.0V
DC Current per I/O Pin
DC Current V_{CC} and GND Pins200.0 mA PDIP and
400.0 mA TQFP/MLF

*NOTICE

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

 T_A = -40°C to 85°C, V_{CC} = 2.7V to 5.5V (Unless Otherwise Noted)

Symbol	Parameter	Condition	Min	Тур	Max	Units
V _{IL}	Input Low Voltage except XTAL1 and RESET pins	V _{CC} = 2.7 - 5.5	-0.5		0.2 V _{CC} ⁽¹⁾	
V _{IH}	Input High Voltage except XTAL1 and RESET pins	V _{CC} = 2.7 - 5.5	0.6 V _{CC} ⁽²⁾		V _{CC} +0.5	V
V _{IH1}	Input High Voltage XTAL1 pin	V _{CC} = 2.7 - 5.5	0.7 V _{CC} ⁽²⁾		V _{CC} +0.5	
V _{IL1}	Input Low Voltage XTAL1 pin	V _{CC} = 2.7 - 5.5	-0.5		0.1 V _{CC} ⁽¹⁾	V
V _{IH2}	Input High Voltage RESET pin	V _{CC} = 2.7 - 5.5	0.9 V _{CC} ⁽²⁾		V _{CC} +0.5	
V _{IL2}	Input Low Voltage RESET pin	V _{CC} = 2.7 - 5.5	-0.5		0.2 V _{CC}	
V _{OL}	Output Low Voltage ⁽³⁾ (Ports A,B,C,D)	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 3V			0.7 0.5	V V
V _{OH}	Output High Voltage ⁽⁴⁾ (Ports A,B,C,D)	I_{OH} = -20 mA, V_{CC} = 5V I_{OH} = -10 mA, V_{CC} = 3V	4.2 2.2			V V
I _{IL}	Input Leakage Current I/O Pin	Vcc = 5.5V, pin low (absolute value)			1	^
l _{iH}	Input Leakage Current I/O Pin	Vcc = 5.5V, pin high (absolute value)			1	μА
R _{RST}	Reset Pull-up Resistor		30		60	ko
R _{pu}	I/O Pin Pull-up Resistor		20		50	kΩ





 $T_A = -40$ °C to 85 °C, $V_{CC} = 2.7$ V to 5.5V (Unless Otherwise Noted) (Continued)

Symbol	Parameter	Condition	Min	Тур	Max	Units	
			Active 1 MHz, V _{CC} = 3V (ATmega16L)		1.1		
		Active 4 MHz, V _{CC} = 3V (ATmega16L)		3.8	5		
	Dower Supply Current	Active 8 MHz, V _{CC} = 5V (ATmega16)		12	15		
lcc	Power Supply Current	Idle 1 MHz, V _{CC} = 3V (ATmega16L)		0.35		mA	
		Idle 4 MHz, V _{CC} = 3V (ATmega16L)		1.2	2		
		Idle 8 MHz, V _{CC} = 5V (ATmega16)		5.5	7		
	Power-down Mode ⁽⁵⁾	WDT enabled, V _{CC} = 3V		<8	15		
	Power-down Mode	WDT disabled, V _{CC} = 3V		< 1	4	μΑ	
V _{ACIO}	Analog Comparator Input Offset Voltage	$V_{CC} = 5V$ $V_{in} = V_{CC}/2$			40	mV	
I _{ACLK}	Analog Comparator Input Leakage Current	V _{CC} = 5V V _{in} = V _{CC} /2	-50		50	nA	
t _{ACPD}	Analog Comparator Propagation Delay	V _{CC} = 2.7V V _{CC} = 4.0V		750 500		ns	

- Notes: 1. "Max" means the highest value where the pin is guaranteed to be read as low
 - "Min" means the lowest value where the pin is guaranteed to be read as high
 - 3. Although each I/O port can sink more than the test conditions (20 mA at Vcc = 5V, 10 mA at Vcc = 3V) under steady state conditions (non-transient), the following must be observed: PDIP Package:
 - 1] The sum of all IOL, for all ports, should not exceed 200 mA.

 - 2] The sum of all IOL, for port A0 A7, should not exceed 100 mA.

 3] The sum of all IOL, for ports B0 B7,C0 C7, D0 D7 and XTAL2, should not exceed 100 mA.
 - TQFP and QFN/MLF Package:
 - 1] The sum of all IOL, for all ports, should not exceed 400 mA.

 - 2] The sum of all IOL, for ports AO A7, should not exceed 100 mA. 3] The sum of all IOL, for ports BO B4, should not exceed 100 mA.
 - 4] The sum of all IOL, for ports B3 B7, XTAL2, D0 D2, should not exceed 100 mA. 5] The sum of all IOL, for ports D3 D7, should not exceed 100 mA. 6] The sum of all IOL, for ports C0 C7, should not exceed 100 mA.
 - If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
 - 4. Although each I/O port can source more than the test conditions (20 mA at Vcc = 5V, 10 mA at Vcc = 3V) under steady state conditions (non-transient), the following must be observed: PDIP Package:

 - 1] The sum of all IOH, for all ports, should not exceed 200 mA. 2] The sum of all IOH, for port A0 A7, should not exceed 100 mA.
 - 3] The sum of all IOH, for ports B0 B7,C0 C7, D0 D7 and XTAL2, should not exceed 100 mA.
 - TQFP and QFN/MLF Package:
 - 1] The sum of all IOH, for all ports, should not exceed 400 mA.

 - 2] The sum of all IOH, for ports A0 A7, should not exceed 100 mA.
 3] The sum of all IOH, for ports B0 B4, should not exceed 100 mA.
 4] The sum of all IOH, for ports B3 B7, XTAL2, D0 D2, should not exceed 100 mA.



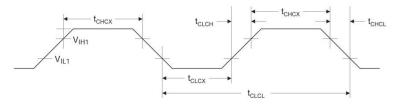
292



- 5] The sum of all IOH, for ports D3 D7, should not exceed 100 mA.
- 6] The sum of all IOH, for ports C0 C7, should not exceed 100 mA.If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
- 5. Minimum V_{CC} for Power-down is 2.5V.

External Clock Drive Waveforms

Figure 145. External Clock Drive Waveforms



External Clock Drive

Table 118. External Clock Drive⁽¹⁾

		$V_{CC} = 2.7V \text{ to } 5.5V$		$V_{CC} = 4.5V \text{ to } 5.5V$			
Symbol	Parameter	Min	Max	Min	Max	Units	
1/t _{CLCL}	Oscillator Frequency	0	8	0	16	MHz	
t _{CLCL}	Clock Period	125		62.5			
t _{CHCX}	High Time	50		25		ns	
t _{CLCX}	Low Time	50		25			
t _{CLCH}	Rise Time		1.6		0.5		
t _{CHCL}	Fall Time		1.6		0.5	μs	
Δt_{CLCL}	Change in period from one clock cycle to the next		2		2	%	

Note: 1. Refer to "External Clock" on page 31 for details.

Table 119. External RC Oscillator, Typical Frequencies ($V_{CC} = 5$)

$R [k\Omega]^{(1)}$	C [pF]	f ⁽²⁾
33	22	650 kHz
10	22	2.0 MHz

Notes: 1. R should be in the range 3 k Ω - 100 k Ω , and C should be at least 20 pF.

2. The frequency will vary with package type and board layout.





Two-wire Serial Interface Characteristics

Table 120 describes the requirements for devices connected to the Two-wire Serial Bus. The ATmega16 Two-wire Serial Interface meets or exceeds these requirements under the noted conditions. Timing symbols refer to Figure 146.

Table 120. Two-wire Serial Bus Requirements

Symbol	Parameter	Condition	Min	Max	Units	
$V_{\rm IL}$	Input Low-voltage		-0.5	0.3 V _{CC}		
V_{IH}	Input High-voltage		0.7 V _{CC}	$V_{CC} + 0.5$	V	
V _{hys} ⁽¹⁾	Hysteresis of Schmitt Trigger Inputs		0.05 V _{CC} ⁽²⁾	=	V	
V _{OL} ⁽¹⁾	Output Low-voltage	3 mA sink current	0	0.4		
t _r ⁽¹⁾	Rise Time for both SDA and SCL		20 + 0.1C _b ⁽³⁾⁽²⁾	300		
t _{of} ⁽¹⁾	Output Fall Time from V _{IHmin} to V _{ILmax}	10 pF < C _b < 400 pF ⁽³⁾	20 + 0.1C _b ⁽³⁾⁽²⁾	250	ns	
t _{SP} ⁽¹⁾	Spikes Suppressed by Input Filter		0	50 ⁽²⁾		
l _i	Input Current each I/O Pin	$0.1V_{CC} < V_i < 0.9V_{CC}$	-10	10	μA	
C _i ⁽¹⁾	Capacitance for each I/O Pin		-	10	pF	
f _{scL}	SCL Clock Frequency	$f_{CK}^{(4)} > max(16f_{SCL}, 250kHz)^{(5)}$	0	400	kHz	
Rp	Value of Pull-up resistor	f _{SCL} ≤ 100 kHz	$\frac{V_{CC} - 0.4V}{3 \text{ mA}}$	$\frac{1000 \text{ns}}{C_b}$		
		f _{SCL} > 100 kHz	$\frac{V_{CC} - 0.4V}{3 \text{ mA}}$	$\frac{300 \text{ns}}{C_b}$	Ω	
t _{HD;STA}	Hold Time (repeated) START Condition	f _{SCL} ≤ 100 kHz	4.0	_		
HD;STA	The first (speaker, e.f. it is contained.)	f _{SCL} > 100 kHz	0.6	_		
t _{LOW}	Low Period of the SCL Clock	f _{SCL} ≤ 100 kHz	4.7	-		
LOW		f _{SCL} > 100 kHz	1.3	-		
t _{HIGH}	High period of the SCL clock	f _{SCL} ≤ 100 kHz	4.0	_		
HIGH		f _{SCL} > 100 kHz	0.6	_	μs	
t _{SU:STA}		f _{SCL} ≤ 100 kHz	4.7	-		
-30,31A	Set-up time for a repeated START condition	f _{SCL} > 100 kHz	0.6	_		
t _{HD:DAT}	Data hold time	f _{SCL} ≤ 100 kHz	0	3.45		
*HD;DAT		f _{SCL} > 100 kHz	0	0.9		
t _{SU:DAT}	Data setup time	f _{SCL} ≤ 100 kHz	250	_		
-SU;DAI		f _{SCL} > 100 kHz	100	-	ns	
t _{su;sto}	Setup time for STOP condition	f _{SCL} ≤ 100 kHz	4.0	-		
30,310		f _{SCL} > 100 kHz	0.6	-		
t _{BUF}	Bus free time between a STOP and START	f _{SCL} ≤ 100 kHz	4.7	_	μs	
DUF	condition	f _{SCL} > 100 kHz	1.3	-		

Notes: 1. In ATmega16, this parameter is characterized and not 100% tested.

- Required only for f_{SCL} > 100 kHz.
 C_b = capacitance of one bus line in pF.
 f_{CK} = CPU clock frequency

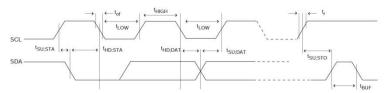


294



This requirement applies to all ATmega16 Two-wire Serial Interface operation. Other devices connected to the Two-wire Serial Bus need only obey the general f_{SCL} requirement.

Figure 146. Two-wire Serial Bus Timing



SPI Timing Characteristics

See Figure 147 and Figure 148 for details.

Table 121. SPI Timing Parameters

	Description	Mode	Min	Тур	Max	
1	SCK period	Master		See Table 58		
2	SCK high/low	Master		50% duty cycle		
3	Rise/Fall time	Master		3.6		
4	Setup	Master		10		
5	Hold	Master		10		
6	Out to SCK	Master		0.5 • t _{SCK}		ns
7	SCK to out	Master		10		
8	SCK to out high	Master		10		
9	SS low to out	Slave		15		
10	SCK period	Slave	4 • t _{SCK}			
11	SCK high/low	Slave	2 • t _{SCK}			
12	Rise/Fall time	Slave			1.6	μs
13	Setup	Slave	10			
14	Hold	Slave	10			
15	SCK to out	Slave		15		nc
16	SCK to SS high	Slave	20			ns
17	SS high to tri-state	Slave		10		
18	SS low to SCK	Slave	2 • t _{SCK}			





Figure 147. SPI Interface Timing Requirements (Master Mode)

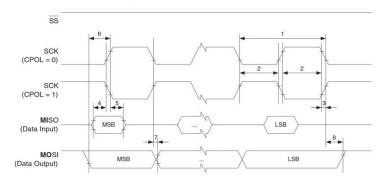
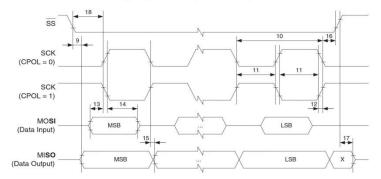


Figure 148. SPI Interface Timing Requirements (Slave Mode)







ADC Characteristics

Table 122. ADC Characteristics

Symbol	Parameter	Condition	Min ⁽¹⁾	Typ ⁽¹⁾	Max ⁽¹⁾	Units	
		Single Ended Conversion		10			
	Resolution	Differential Conversion Gain = 1x or 10x		8		Bits	
		Differential Conversion Gain = 200x		7			
		Single Ended Conversion V _{REF} = 4V, V _{CC} = 4V ADC clock = 200 kHz		1.5	2.5	LSB	
		Single Ended Conversion V _{REF} = 4V, V _{CC} = 4V ADC clock = 1 MHz		3	4		
	Absolute Accuracy (Including INL, DNL, Quantization Error, Gain, and Offset Error).	Single Ended Conversion $V_{REF} = 4V$, $V_{CC} = 4V$ ADC clock = 200 kHz Noise Reduction mode		1.5			
		Single Ended Conversion $V_{REF} = 4V$, $V_{CC} = 4V$ ADC clock = 1 MHz Noise Reduction mode		3			
	Integral Non-linearity (INL)	Single Ended Conversion V _{REF} = 4V, V _{CC} = 4V ADC clock = 200 kHz		1			
	Differential Non-linearity (DNL)	Single Ended Conversion V _{REF} = 4V, V _{CC} = 4V ADC clock = 200 kHz		0.5			
	Gain Error	Single Ended Conversion V _{REF} = 4V, V _{CC} = 4V ADC clock = 200 kHz		1			
	Offset Error	Single Ended Conversion V _{REF} = 4V, V _{CC} = 4V ADC clock = 200 kHz					
	Conversion Time	Free Running Conversion	13		260	μs	
	Clock Frequency		50		1000	kHz	
AVCC	Analog Supply Voltage		V _{CC} - 0.3 ⁽²⁾		$V_{CC} + 0.3^{(3)}$	V	
V_{REF}	Reference Voltage	Single Ended Conversion	2.0		AVCC		
* REF	Reference voltage	Differential Conversion	2.0		AVCC - 0.2		
	Input voltage	Single ended channels	GND		V _{REF}		
V _{IN}		Differential channels	0		V _{REF}		
V IN	Input bandwidth	Single ended channels		38.5		kHz	
		Differential channels		4		KIIZ	



297



Table 122. ADC Characteristics (Continued)

Symbol	Parameter	Condition	Min ⁽¹⁾	Typ ⁽¹⁾	Max ⁽¹⁾	Units
V_{INT}	Internal Voltage Reference		2.3	2.6	2.9	V
R_{REF}	Reference Input Resistance			32		kΩ
R _{AIN}	Analog Input Resistance			100		MΩ

- Notes: 1. Values are guidelines only.
 2. Minimum for AVCC is 2.7V.
 3. Maximum for AVCC is 5.5V.





ATmega16 **Typical** Characteristics

The following charts show typical behavior. These figures are not tested during manufacturing. All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled. A sine wave generator with rail-to-rail output is used as clock

The power consumption in Power-down mode is independent of clock selection.

The current consumption is a function of several factors such as: operating voltage, operating frequency, loading of I/O pins, switching rate of I/O pins, code executed and ambient temperature. The dominating factors are operating voltage and frequency.

The current drawn from capacitive loaded pins may be estimated (for one pin) as C_L*V_{cc}*f where C_L = load capacitance, V_{CC} = operating voltage and f = average switching frequency of I/O pin.

The parts are characterized at frequencies higher than test limits. Parts are not guaranteed to function properly at frequencies higher than the ordering code indicates.

The difference between current consumption in Power-down mode with Watchdog Timer enabled and Power-down mode with Watchdog Timer disabled represents the differential current drawn by the Watchdog Timer.

Active Supply Current Figure 149. Active Supply Current vs. Frequency (0.1 MHz - 1.0 MHz

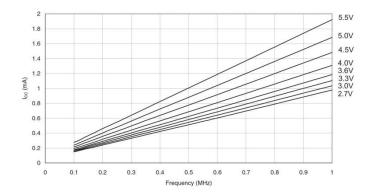






Figure 150. Active Supply Current vs. Frequency (1 MHz - 20 MHz)

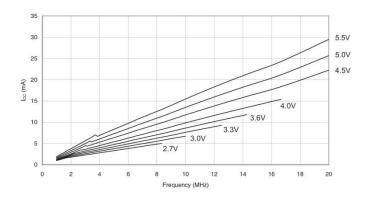
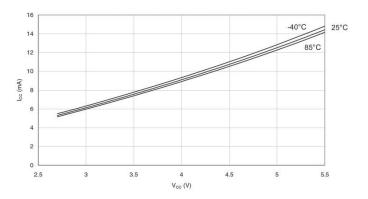


Figure 151. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)



<u>AIMEL</u>

300



Figure 152. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 4 MHz)

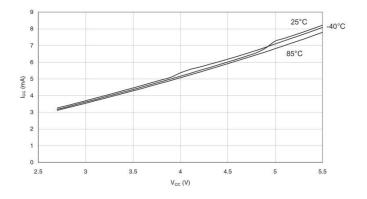
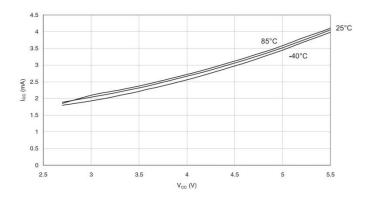


Figure 153. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 2 MHz)



ATMEL



Figure 154. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 1 MHz)

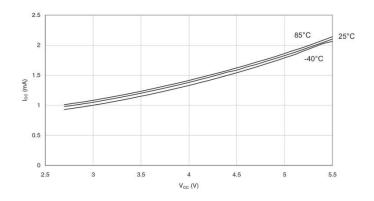
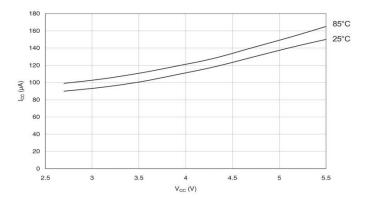


Figure 155. Active Supply Current vs. V_{CC} (32 kHz External Oscillator)



AMEL

302



Idle Supply Current

Figure 156. Idle Supply Current vs. Frequency (0.1 MHz - 1.0 MHz)

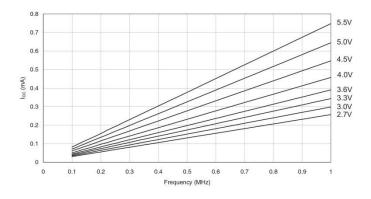
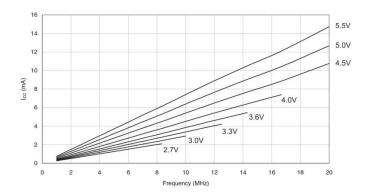


Figure 157. Idle Supply Current vs. Frequency (1 MHz - 20 MHz)



AIMEL

303



Figure 158. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)

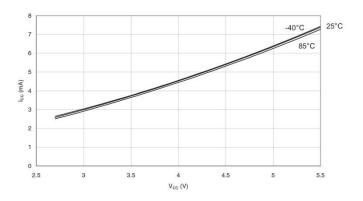
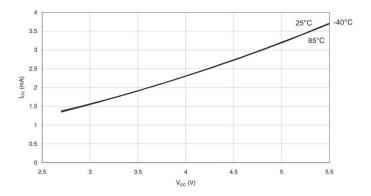


Figure 159. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 4 MHz)



AIMEL

304



Figure 160. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 2 MHz)

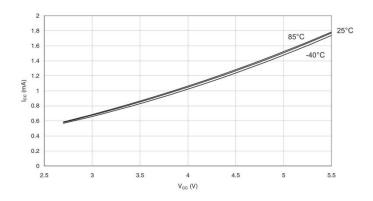
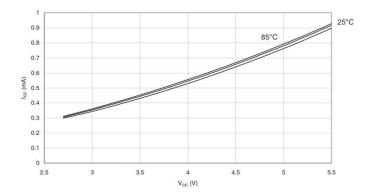


Figure 161. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 1 MHz)

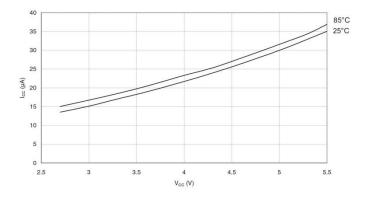


<u>AIMEL</u>

305



Figure 162. Idle Supply Current vs. V_{CC} (32 kHz External Oscillator)



Power-Down Supply Current

Figure 163. Power-Down Supply Current vs. V_{CC} (Watchdog Timer Disabled)

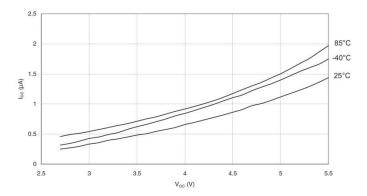
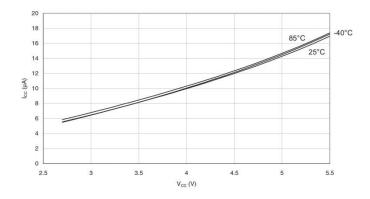




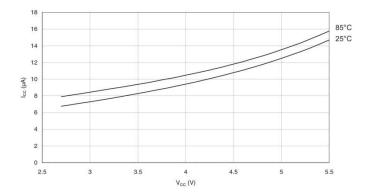


Figure 164. Power-Down Supply Current vs. V_{CC} (Watchdog Timer Enabled)



Power-Save Supply Current

Figure 165. Power-Save Supply Current vs. V_{CC} (Watchdog Timer Disabled)



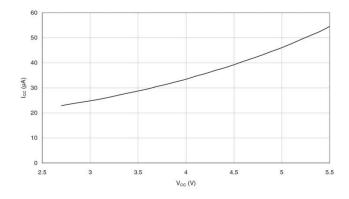
<u>AIMEL</u>

307

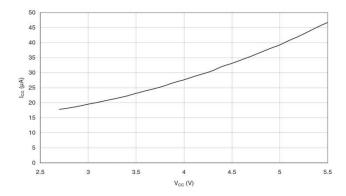


Standby Supply Current

 $\textbf{Figure 166.} \ \ \text{Standby Supply Current vs. V}_{\text{CC}} \ (\text{455 kHz Resonator, Watchdog Timer Disabled})$



 $\textbf{Figure 167.} \ \ \textbf{Standby Supply Current vs.} \ \ \textbf{V}_{\text{CC}} \ \ \textbf{(1 MHz Resonator, Watchdog Timer Disabled)}$



AIMEL



 $\textbf{Figure 168.} \ \ \textbf{Standby Supply Current vs. V}_{\text{CC}} \ (2 \ \text{MHz Resonator, Watchdog Timer Disabled})$

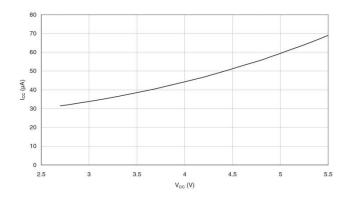
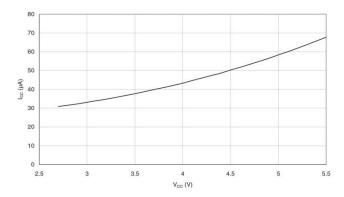


Figure 169. Standby Supply Current vs. V_{CC} (2 MHz Xtal, Watchdog Timer Disabled)



309



 $\textbf{Figure 170.} \ \ \textbf{Standby Supply Current vs. V}_{\text{CC}} \ (4 \ \text{MHz Resonator, Watchdog Timer Disabled})$

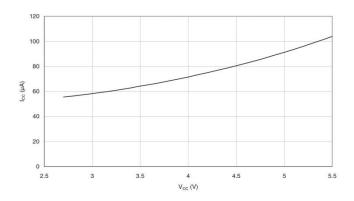
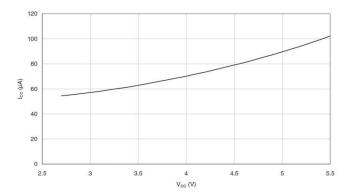


Figure 171. Standby Supply Current vs. V_{CC} (4 MHz Xtal, Watchdog Timer Disabled)



AIMEL



 $\textbf{Figure 172.} \ \ \textbf{Standby Supply Current vs. V}_{\text{CC}} \ (6 \ \text{MHz Resonator, Watchdog Timer Disabled})$

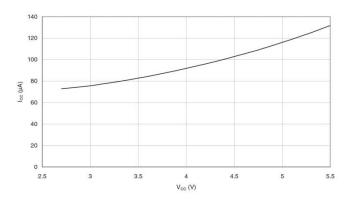
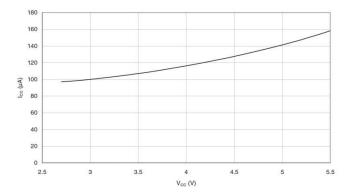


Figure 173. Standby Supply Current vs. V_{CC} (6 MHz Xtal, Watchdog Timer Disabled)



AIMEL



Pin Pullup

Figure 174. I/O Pin Pull-Up Resistor Current vs. Input Voltage ($V_{CC} = 5V$)

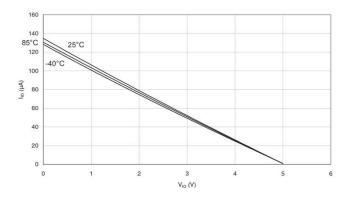
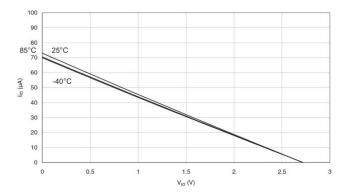


Figure 175. I/O Pin Pull-Up Resistor Current vs. Input Voltage ($V_{CC} = 2.7V$)



AIMEL



Figure 176. Reset Pull-Up Resistor Current vs. Reset Pin Voltage (V_{CC} = 5V)

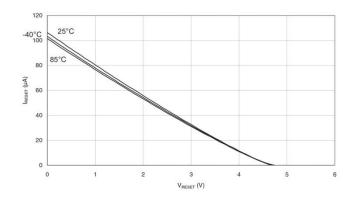
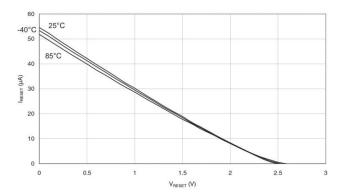


Figure 177. Reset Pull-Up Resistor Current vs. Reset Pin Voltage ($V_{CC} = 2.7V$)



AIMEL



Pin Driver Strength Figure 178. I/O Pin Source Current vs. Output Voltage (V_{CC} = 5V)

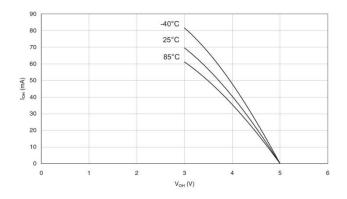
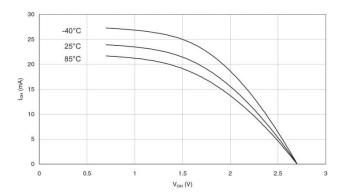


Figure 179. I/O Pin Source Current vs. Output Voltage ($V_{CC} = 2.7V$)



AIMEL



Figure 180. I/O Pin Sink Current vs. Output Voltage ($V_{CC} = 5V$)

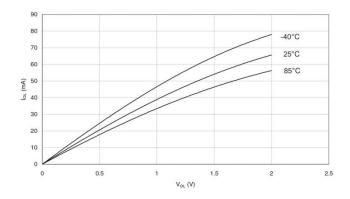
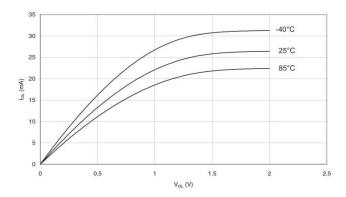


Figure 181. I/O Pin Sink Current vs. Output Voltage ($V_{CC} = 2.7V$)



AIMEL



Pin Thresholds And Hysteresis

Figure 182. I/O Pin Input Threshold Voltage vs. V_{CC} (V_{IH} , I/O Pin Read As '1')

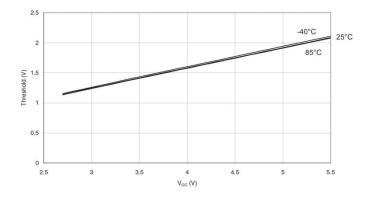
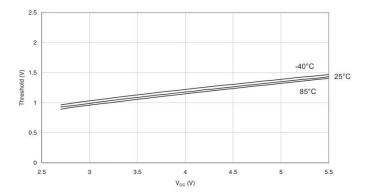


Figure 183. I/O Pin Input Threshold Voltage vs. V_{CC} (V_{IL} , I/O Pin Read As '0')



AIMEL



Figure 184. I/O Pin Input Hysteresis vs. $V_{\rm CC}$

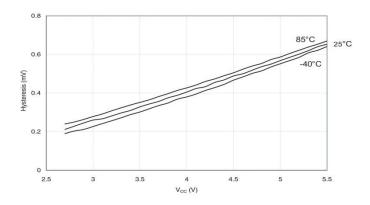
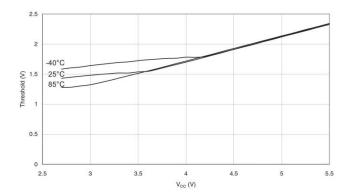


Figure 185. Reset Input Threshold Voltage vs. V_{CC} ($V_{IH^{\prime}}$ Reset Pin Read As '1')



AIMEL



Figure 186. Reset Input Threshold Voltage vs. V_{CC} (V_{IL} , Reset Pin Read As '0')

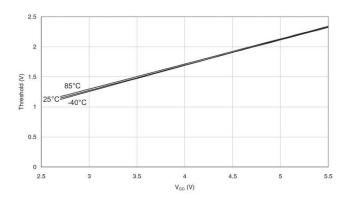
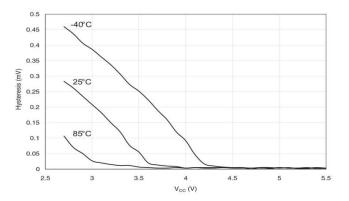


Figure 187. Reset Input Pin Hysteresis vs. V_{CC}



ATMEL



Bod Thresholds And Analog Comparator Offset

Figure 188. Bod Thresholds vs. Temperature (Bodlevel is 4.0V)

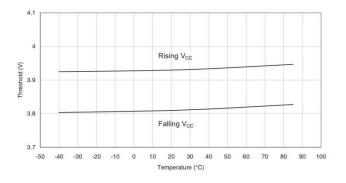


Figure 189. Bod Thresholds vs. Temperature (Bodlevel is 2.7V)

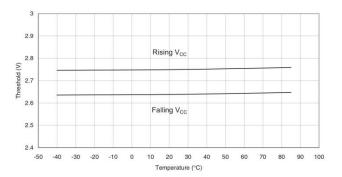




Figure 190. Bandgap Voltage vs. V_{CC}

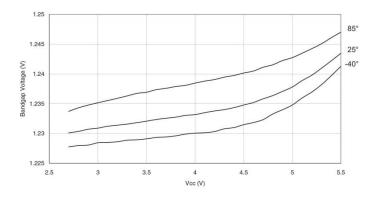
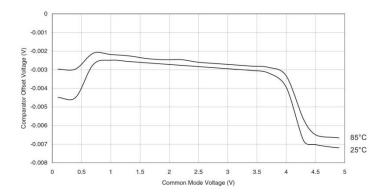


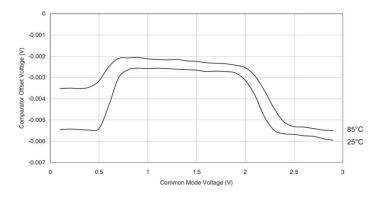
Figure 191. Analog Comparator Offset Voltage vs. Common Mode Voltage (V_{CC} = 5V)



AIMEL



Figure 192. Analog Comparator Offset Voltage vs. Common Mode Voltage ($V_{CC} = 3V$)



Internal Oscillator Speed

Figure 193. Watchdog Oscillator Frequency vs. $V_{\rm CC}$

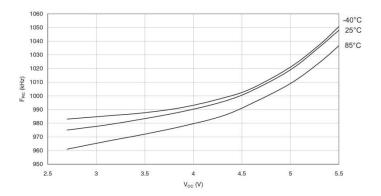






Figure 194. Calibrated 8 MHz RC Oscillator Frequency vs. Temperature

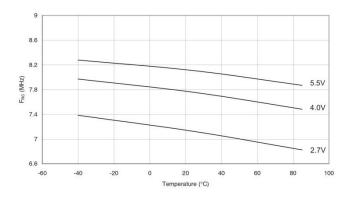
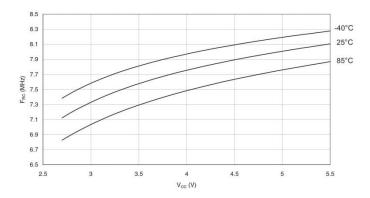


Figure 195. Calibrated 8 MHz RC Oscillator Frequency vs. $V_{\rm CC}$



AIMEL



Figure 196. Calibrated 8 MHz RC Oscillator Frequency vs. Osccal Value

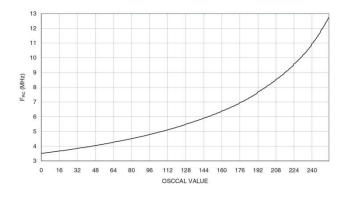
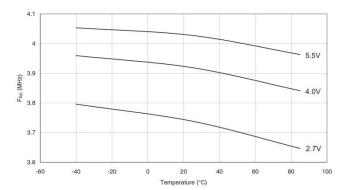


Figure 197. Calibrated 4 MHz RC Oscillator Frequency vs. Temperature



AMEL



Figure 198. Calibrated 4 MHz RC Oscillator Frequency vs. $V_{\rm CC}$

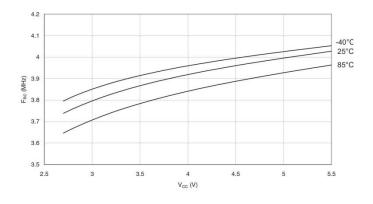
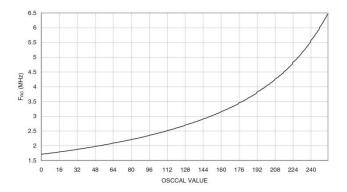


Figure 199. Calibrated 4 MHz RC Oscillator Frequency vs. Osccal Value



AIMEL



Figure 200. Calibrated 2 MHz RC Oscillator Frequency vs. Temperature

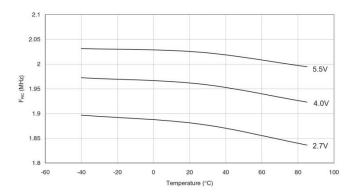
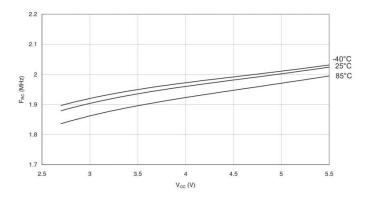


Figure 201. Calibrated 2 MHz RC Oscillator Frequency vs. $V_{\rm CC}$



AIMEL



Figure 202. Calibrated 2 MHz RC Oscillator Frequency vs. Osccal Value

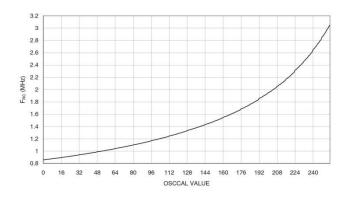
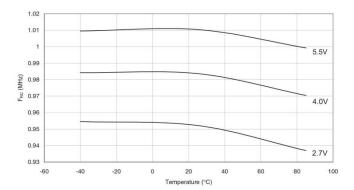


Figure 203. Calibrated 1 MHz RC Oscillator Frequency vs. Temperature



ATMEL



Figure 204. Calibrated 1 MHz RC Oscillator Frequency vs. $V_{\rm CC}$

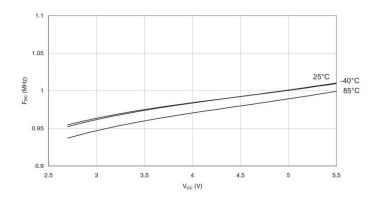
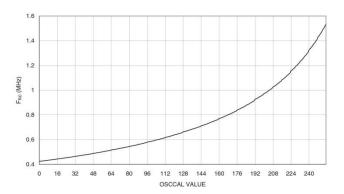


Figure 205. Calibrated 1 MHz RC Oscillator Frequency vs. Osccal Value



<u>AIMEL</u>



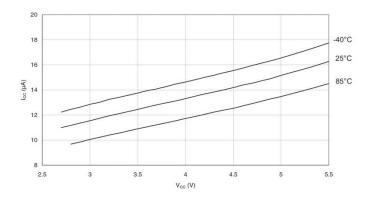


Figure 207. ADC Current vs. V_{CC}(Aref = AVCC)

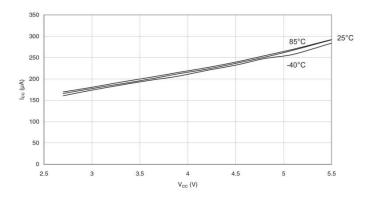




Figure 208. Aref External Reference Current vs. V_{CC}

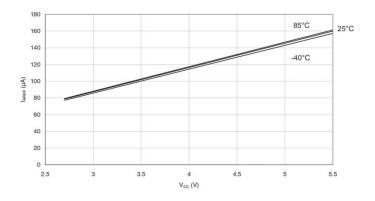
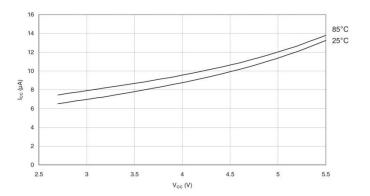


Figure 209. 32khz Tosc Current vs. $V_{\rm CC}$ (Watchdog Timer Disabled)



ATMEL



Figure 210. Watchdog Timer Current vs. V_{CC}

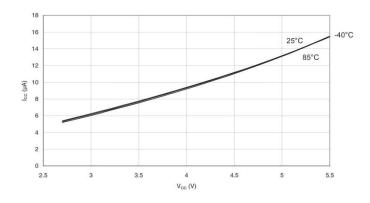
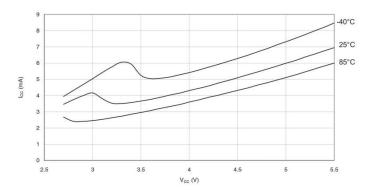


Figure 211. Programming Current vs. V_{CC}



<u>AIMEL</u>



Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
\$3F (\$5F)	SREG	1	Т	н	s	V	N	Z	С	9
\$3E (\$5E)	SPH	-	-	-	-	-	SP10	SP9	SP8	12
\$3D (\$5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	12
\$3C (\$5C)	OCR0	Timer/Counter	0 Output Compar	e Register						85
\$3B (\$5B)	GICR	INT1	INTO	INT2	-	-	-	IVSEL	IVCE	48, 69
\$3A (\$5A)	GIFR	INTF1	INTF0	INTF2	-	-	-	_	-	70
\$39 (\$59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	85, 115, 133
\$38 (\$58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	86, 115, 133
\$37 (\$57)	SPMCR	SPMIE	RWWSB	-	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	250
\$36 (\$56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWE	180
\$35 (\$55)	MCUCR MCUCSR	SM2 JTD	SE ISC2	SM1	SM0 JTRF	ISC11 WDRF	ISC10 BORF	ISC01	ISC00 PORF	32, 68 41, 69, 231
\$34 (\$54) \$33 (\$53)	TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	83
\$32 (\$52)	TCNTO	Timer/Counter		COMO	CONIDO	VVGIVIOT	0302	C301	L 0300	85
6000 6000	OSCCAL		bration Register							30
\$31 ⁽¹⁾ (\$51) ⁽¹⁾	OCDR	On-Chip Debu								227
\$30 (\$50)	SFIOR	ADTS2	ADTS1	ADTS0	_	ACME	PUD	PSR2	PSR10	57,88,134,201,221
\$2F (\$4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	110
\$2E (\$4E)	TCCR1B	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	113
\$2D (\$4D)	TCNT1H		1 – Counter Regi	ster High Byte						114
\$2C (\$4C)	TCNT1L	Timer/Counter	1 – Counter Regi	ster Low Byte						114
\$2B (\$4B)	OCR1AH	Timer/Counter	1 - Output Comp	are Register A Hi	gh Byte					114
\$2A (\$4A)	OCR1AL	Timer/Counter	1 – Output Comp	are Register A Lo	w Byte					114
\$29 (\$49)	OCR1BH	Timer/Counter	1 - Output Comp	are Register B Hi	gh Byte					114
\$28 (\$48)	OCR1BL			are Register B Lo						114
\$27 (\$47)	ICR1H	Timer/Counter	1 - Input Capture	Register High By	te					114
\$26 (\$46)	ICR1L			Register Low By						114
\$25 (\$45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	128
\$24 (\$44)	TCNT2	Timer/Counter								130
\$23 (\$43)	OCR2	Timer/Counter	2 Output Compar	e Register			T 200000			130
\$22 (\$42)	ASSR WDTCR	-	-	-	WDT0E	AS2 WDE	TCN2UB WDP2	OCR2UB WDP1	TCR2UB WDP0	131 43
\$21 (\$41)	UBRRH	URSEL	-	-	- WDTOE	WDE		R[11:8]	VVDPU	167
\$20(2) (\$40)(2)	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	166
\$1F (\$3F)	EEARH	UKSEL	OWISEL	OF WIT	OF WIO	0080	- 00321	-	EEAR8	19
\$1E (\$3E)	EEARL	FEPROM Add	ress Register Lov	v Byte					LLFITO	19
\$1D (\$3D)	EEDR	EEPROM Data		, byte						19
\$1C (\$3C)	EECR	-	-	_	_	EERIE	EEMWE	EEWE	EERE	19
\$1B (\$3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	66
\$1A (\$3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	66
\$19 (\$39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	66
\$18 (\$38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	66
\$17 (\$37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	66
\$16 (\$36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	66
\$15 (\$35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	67
\$14 (\$34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	67
\$13 (\$33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	67
\$12 (\$32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	67
\$11 (\$31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	67
\$10 (\$30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	67
\$0F (\$2F)	SPDR	SPI Data Reg							CDIOY	142
\$0E (\$2E)	SPSR	SPIF	WCOL	-	-	-	-	-	SPI2X	142
\$0D (\$2D)	SPCR	SPIE USART I/O D	SPE eta Bagietar	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	140 163
\$0C (\$2C) \$0B (\$2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	164
\$0A (\$2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	165
\$09 (\$29)	UBRRL		Rate Register Lo		INALIA	IALI	00022	INADO	1700	167
\$08 (\$28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	202
\$07 (\$27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUXO	217
\$06 (\$26)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	219
\$05 (\$25)	ADCH		gister High Byte							220
	ADGL		gister Low Byte							220
\$04 (\$24)										
\$04 (\$24) \$03 (\$23)	TWDR		al Interface Data F	Register						182



331



Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
\$01 (\$21)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	#1	TWPS1	TWPS0	181
\$00 (\$20)	TWBR	Two-wire Serial Interface Bit Rate Register							180	

Notes: 1. When the OCDEN Fuse is unprogrammed, the OSCCAL Register is always accessed on this address. Refer to the debugger specific documentation for details on how to use the OCDR Register.

- 2. Refer to the USART description for details on how to access UBRRH and UCSRC.
- 3. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
- 4. Some of the Status Flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O Register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers \$00 to \$1F only.





Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND	LOGIC INSTRUCTIO	NS			
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	$Rd \leftarrow Rd + Rr + C$	Z,C,N,V,H	1
ADIW	RdI,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	RdI,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	$Rd \leftarrow Rd \bullet Rr$	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	$Rd \leftarrow Rd \oplus Rr$	Z,N,V	1
COM	Rd	One's Complement	Rd ← \$FF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← \$00 - Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • (\$FF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd − 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER			Rd ← \$FF		1
	Rd	Set Register		None	
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \le 1$	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \leq 1$	Z,C	2
BRANCH INSTRUC	CTIONS				
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
JMP	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
CALL	k	Direct Subroutine Call	PC ← k	None	4
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	1	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P. b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC \leftarrow PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Carry Cleared Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Same of Higher Branch if Lower	if (C = 0) then PC ← PC + k + 1 if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k		if (N = 1) then PC ← PC + k + 1 if (N = 1) then PC ← PC + k + 1		1/2
		Branch if Minus		None	
BRPL	k	Branch if Creater as Equal Signed	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
	. k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVS BRVC	k	Branen ii Overnow i lag io oct	if (V = 0) then PC ← PC + k + 1		1/2



333



Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2
DATA TRANSFER					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X Rd, Y	Load Indirect and Pre-Dec. Load Indirect	$X \leftarrow X - 1$, $Rd \leftarrow (X)$ $Rd \leftarrow (Y)$	None None	2 2
LD	Rd, Y+	Load Indirect and Post-Inc.	$Rd \leftarrow (Y), Y \leftarrow Y + 1$	None	2
LD	Rd, 14	Load Indirect and Pre-Dec.	$Y \leftarrow Y - 1$, $Rd \leftarrow (Y)$	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	$Rd \leftarrow (Y + q)$	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	$Z \leftarrow Z - 1$, $Rd \leftarrow (Z)$	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	$Rd \leftarrow (Z + q)$	None	2
LDS	Rd, k	Load Direct from SRAM	$Rd \leftarrow (k)$	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	$X \leftarrow X - 1$, $(X) \leftarrow Rr$	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	$Y \leftarrow Y - 1$, $(Y) \leftarrow Rr$	None	2
STD	Y+q,Rr	Store Indirect with Displacement	$(Y + q) \leftarrow Rr$	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
STD	-Z, Rr	Store Indirect and Pre-Dec. Store Indirect with Displacement	Z ← Z - 1, (Z) ← Rr	None None	2 2
STS	Z+q,Rr k, Rr	Store Direct to SRAM	$(Z + q) \leftarrow Rr$ $(k) \leftarrow Rr$	None	2
LPM	K, KI	Load Program Memory	(K) ← K1 R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	$Rd \leftarrow (Z), Z \leftarrow Z+1$	None	3
SPM	110, 2	Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
BIT AND BIT-TEST	INSTRUCTIONS		W.		
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	$Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0$	Z,C,N,V	1
LSR	Rd	Logical Shift Right	$Rd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0$	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	$Rd(0)\leftarrow C_1Rd(n+1)\leftarrow Rd(n)_1C\leftarrow Rd(7)$	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	$Rd(7)\leftarrow C, Rd(n)\leftarrow Rd(n+1), C\leftarrow Rd(0)$	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=06	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(30)←Rd(74),Rd(74)←Rd(30)	None SPEC(e)	1 1
BCLR	5 S	Flag Set Flag Clear	$SREG(s) \leftarrow 1$ $SREG(s) \leftarrow 0$	SREG(s) SREG(s)	1
BST	Rr, b	Bit Store from Register to T	$T \leftarrow Rr(b)$	T SREG(S)	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C←0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	1←1	1	1
CLI		Global Interrupt Disable	1←0	1	1
SES		Set Signed Test Flag	S←1	S	1
CLS	1	Clear Signed Test Flag	S ← 0	S	1
SEV	-	Set Twos Complement Overflow.	V ← 1	V	1
CLV	+	Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	Ţ	1
CLT		Clear T in SREG	T ← 0	T	1 1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1 1



334



Mnemonics	Operands	Description	Operation	Flags	#Clocks
CLH		Clear Half Carry Flag in SREG	H ← 0	Н	1
MCU CONTROL	INSTRUCTIONS	n .	0		
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-Chip Debug Only	None	N/A





Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range	
8	2.7V - 5.5V	ATmega16L-8AU ⁽¹⁾ ATmega16L-8PU ⁽¹⁾ ATmega16L-8MU ⁽¹⁾	44A 40P6 44M1	Industrial (-40°C to 85°C)	
16	4.5V - 5.5V	ATmega16-16AU ⁽¹⁾ ATmega16-16PU ⁽¹⁾ ATmega16-16MU ⁽¹⁾	44A 40P6 44M1	Industrial (-40°C to 85°C)	

Note: 1. Pb-free packaging complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.

	Package Type					
44A	44-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flat Package (TQFP)					
40P6	40-pin, 0.600" Wide, Plastic Dual Inline Package (PDIP)					
44M1	44-pad, 7 × 7 × 1.0 mm body, lead pitch 0.50 mm, Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)					

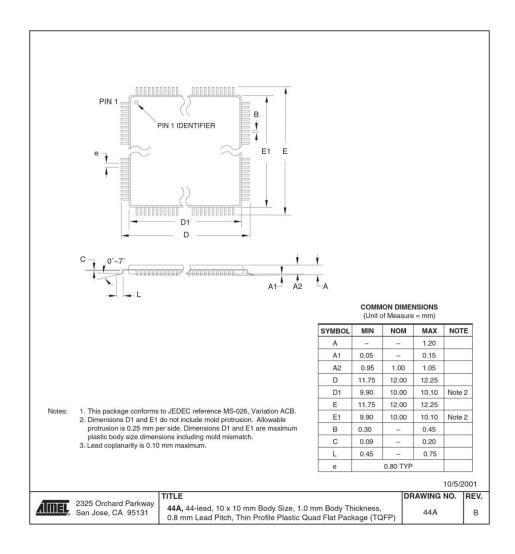


336



Packaging Information

44A

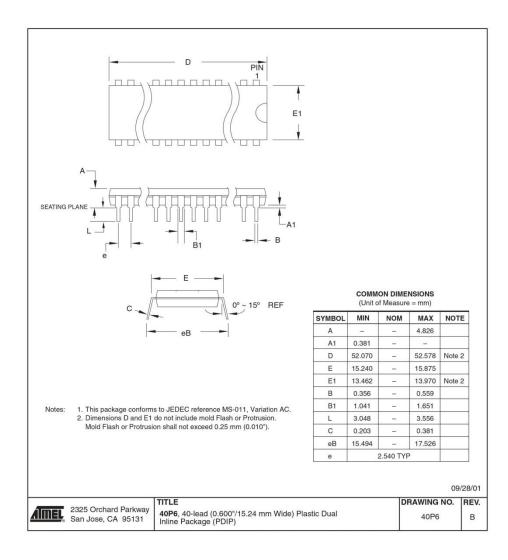




337



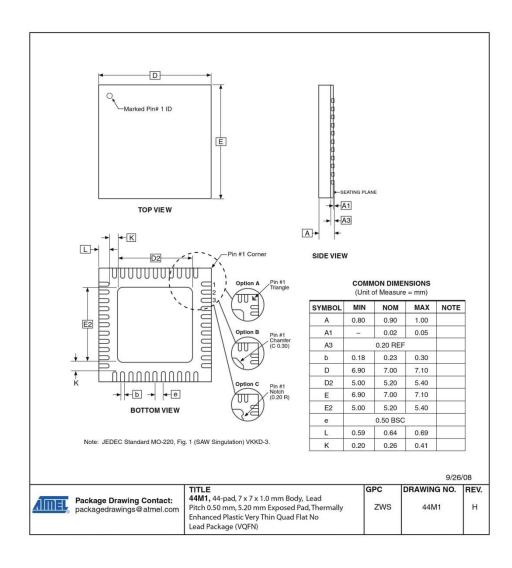
40P6



<u>AIMEL</u>



44M1







Errata

The revision letter in this section refers to the revision of the ATmega16 device.

ATmega16(L) Rev.

- · First Analog Comparator conversion may be delayed
- · Interrupts may be lost when writing the timer registers in the asynchronous timer
- · IDCODE masks data from TDI input
- · Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request

1. First Analog Comparator conversion may be delayed

If the device is powered by a slow rising V_{CC} , the first Analog Comparator conversion will take longer than expected on some devices.

Problem Fix/Workaround

When the device has been powered or reset, disable then enable the Analog Comparator before the first conversion.

2. Interrupts may be lost when writing the timer registers in the asynchronous timer

The interrupt will be lost if a timer register that is synchronized to the asynchronous timer clock is written when the asynchronous Timer/Counter register(TCNTx) is 0x00.

Problem Fix / Workaround

Always check that the asynchronous Timer/Counter register neither have the value 0xFF nor 0x00 before writing to the asynchronous Timer Control Register(TCCRx), asynchronous Timer Counter Register(TCNTx), or asynchronous Output Compare Register(OCRx).

3. IDCODE masks data from TDI input

The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

Problem Fix / Workaround

- If ATmega16 is the only device in the scan chain, the problem is not visible.
- Select the Device ID Register of the ATmega16 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega16 while reading the Device ID Registers of preceding devices of the boundary scan chain.
- If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega16 must be the fist device in the chain.

Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request.

Reading EEPROM by using the ST or STS command to set the EERE bit in the EECR register triggers an unexpected EEPROM interrupt request.

Problem Fix / Workaround

Always use OUT or SBI to set EERE in EECR.

ATmega16(L) Rev.

- First Analog Comparator conversion may be delayed
- Interrupts may be lost when writing the timer registers in the asynchronous timer
- IDCODE masks data from TDI input
- · Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request

1. First Analog Comparator conversion may be delayed

If the device is powered by a slow rising V_{CC} , the first Analog Comparator conversion will take longer than expected on some devices.

<u>AIMEL</u>

340



——— ATmega16(L)

Problem Fix/Workaround

When the device has been powered or reset, disable then enable the Analog Comparator before the first conversion.

2. Interrupts may be lost when writing the timer registers in the asynchronous timer

The interrupt will be lost if a timer register that is synchronized to the asynchronous timer clock is written when the asynchronous Timer/Counter register(TCNTx) is 0x00.

Problem Fix / Workaround

Always check that the asynchronous Timer/Counter register neither have the value 0xFF nor 0x00 before writing to the asynchronous Timer Control Register(TCCRx), asynchronous Timer Counter Register(TCNTx), or asynchronous Output Compare Register(OCRx).

3. IDCODE masks data from TDI input

The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

Problem Fix / Workaround

- If ATmega16 is the only device in the scan chain, the problem is not visible.
- Select the Device ID Register of the ATmega16 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega16 while reading the Device ID Registers of preceding devices of the boundary scan chain.
- If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega16 must be the fist device in the chain.

4. Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request.

Reading EEPROM by using the ST or STS command to set the EERE bit in the EECR register triggers an unexpected EEPROM interrupt request.

Problem Fix / Workaround

Always use OUT or SBI to set EERE in EECR.

ATmega16(L) Rev.

- · First Analog Comparator conversion may be delayed
- · Interrupts may be lost when writing the timer registers in the asynchronous timer
- IDCODE masks data from TDI input
- · Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request

1. First Analog Comparator conversion may be delayed

If the device is powered by a slow rising V_{CC} , the first Analog Comparator conversion will take longer than expected on some devices.

Problem Fix/Workaround

When the device has been powered or reset, disable then enable the Analog Comparator before the first conversion.

2. Interrupts may be lost when writing the timer registers in the asynchronous timer

The interrupt will be lost if a timer register that is synchronized to the asynchronous timer clock is written when the asynchronous Timer/Counter register(TCNTx) is 0x00.



341



Problem Fix / Workaround

Always check that the asynchronous Timer/Counter register neither have the value 0xFF nor 0x00 before writing to the asynchronous Timer Control Register(TCCRx), asynchronous Timer Counter Register(TCNTx), or asynchronous Output Compare Register(OCRx).

3. IDCODE masks data from TDI input

The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

Problem Fix / Workaround

- If ATmega16 is the only device in the scan chain, the problem is not visible.
- Select the Device ID Register of the ATmega16 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega16 while reading the Device ID Registers of preceding devices of the boundary scan chain.
- If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega16 must be the fist device in the chain.

Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request.

Reading EEPROM by using the ST or STS command to set the EERE bit in the EECR register triggers an unexpected EEPROM interrupt request.

Problem Fix / Workaround

Always use OUT or SBI to set EERE in EECR.

ATmega16(L) Rev.

- First Analog Comparator conversion may be delayed
- Interrupts may be lost when writing the timer registers in the asynchronous timer
- IDCODE masks data from TDI input
- Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request

1. First Analog Comparator conversion may be delayed

If the device is powered by a slow rising V_{CC} , the first Analog Comparator conversion will take longer than expected on some devices.

Problem Fix/Workaround

When the device has been powered or reset, disable then enable the Analog Comparator before the first conversion.

2. Interrupts may be lost when writing the timer registers in the asynchronous timer

The interrupt will be lost if a timer register that is synchronized to the asynchronous timer clock is written when the asynchronous Timer/Counter register(TCNTx) is 0x00.

Problem Fix / Workaround

Always check that the asynchronous Timer/Counter register neither have the value 0xFF nor 0x00 before writing to the asynchronous Timer Control Register(TCCRx), asynchronous Timer Counter Register(TCNTx), or asynchronous Output Compare Register(OCRx).

3. IDCODE masks data from TDI input

The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

Problem Fix / Workaround



342



- If ATmega16 is the only device in the scan chain, the problem is not visible.
- Select the Device ID Register of the ATmega16 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega16 while reading the Device ID Registers of preceding devices of the boundary scan chain.
- If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega16 must be the fist device in the chain.

Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request.

Reading EEPROM by using the ST or STS command to set the EERE bit in the EECR register triggers an unexpected EEPROM interrupt request.

Problem Fix / Workaround

Always use OUT or SBI to set EERE in EECR.

ATmega16(L) Rev.

- · First Analog Comparator conversion may be delayed
- Interrupts may be lost when writing the timer registers in the asynchronous timer
- IDCODE masks data from TDI input
- · Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request

1. First Analog Comparator conversion may be delayed

If the device is powered by a slow rising V_{CC} , the first Analog Comparator conversion will take longer than expected on some devices.

Problem Fix/Workaround

When the device has been powered or reset, disable then enable the Analog Comparator before the first conversion.

2. Interrupts may be lost when writing the timer registers in the asynchronous timer

The interrupt will be lost if a timer register that is synchronized to the asynchronous timer clock is written when the asynchronous Timer/Counter register(TCNTx) is 0x00.

Problem Fix / Workaround

Always check that the asynchronous Timer/Counter register neither have the value 0xFF nor 0x00 before writing to the asynchronous Timer Control Register(TCCRx), asynchronous Timer Counter Register(TCNTx), or asynchronous Output Compare Register(OCRx).

3. IDCODE masks data from TDI input

The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

Problem Fix / Workaround

- If ATmega16 is the only device in the scan chain, the problem is not visible.
- Select the Device ID Register of the ATmega16 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega16 while reading the Device ID Registers of preceding devices of the boundary scan chain.
- If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega16 must be the fist device in the chain.



343



Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request.

Reading EEPROM by using the ST or STS command to set the EERE bit in the EECR register triggers an unexpected EEPROM interrupt request.

Problem Fix / Workaround

Always use OUT or SBI to set EERE in EECR.

ATmega16(L) Rev.

Н

- First Analog Comparator conversion may be delayed
- Interrupts may be lost when writing the timer registers in the asynchronous timer
- IDCODE masks data from TDI input
- · Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request

1. First Analog Comparator conversion may be delayed

If the device is powered by a slow rising V_{CC} , the first Analog Comparator conversion will take longer than expected on some devices.

Problem Fix/Workaround

When the device has been powered or reset, disable then enable the Analog Comparator before the first conversion.

2. Interrupts may be lost when writing the timer registers in the asynchronous timer

The interrupt will be lost if a timer register that is synchronized to the asynchronous timer clock is written when the asynchronous Timer/Counter register(TCNTx) is 0x00.

Problem Fix / Workaround

Always check that the asynchronous Timer/Counter register neither have the value 0xFF nor 0x00 before writing to the asynchronous Timer Control Register(TCCRx), asynchronous Timer Counter Register(TCNTx), or asynchronous Output Compare Register(OCRx).

3. IDCODE masks data from TDI input

The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

Problem Fix / Workaround

- If ATmega16 is the only device in the scan chain, the problem is not visible.
- Select the Device ID Register of the ATmega16 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega16 while reading the Device ID Registers of preceding devices of the boundary scan chain.
- If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega16 must be the fist device in the chain.

Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request.

Reading EEPROM by using the ST or STS command to set the EERE bit in the EECR register triggers an unexpected EEPROM interrupt request.

Problem Fix / Workaround

Always use OUT or SBI to set EERE in EECR.



344



Datasheet Revision History

Please note that the referring page numbers in this section are referred to this document. The referring revision in this section are referring to the document revision.

Rev. 2466T-07/10

- Corrected use of comma in formula Rp in Table 120, "Two-wire Serial Bus Requirements," on page 294.
- 2. Updated document according to Atmel's Technical Terminology
- 3. Note 6 and Note 7 under Table 120, "Two-wire Serial Bus Requirements," on page 294 have been removed.
- Rev. 2466S-05/09
- 1. Updated "Errata" on page 340.
- 2. Updated the last page with Atmel's new adresses.
- Rev. 2466R-06/08
- 1. Added "Not recommended for new designs" note in Figure on page 1.

Rev. 2466Q-05/08

- Updated "Fast PWM Mode" on page 77 in "8-bit Timer/Counter0 with PWM" on page 71:
 - Removed the last section describing how to achieve a frequency with 50% duty cycle waveform output in fast PWM mode.
- 2. Removed note from Feature list in "Analog to Digital Converter" on page 204.
- 3. Removed note from Table 84 on page 218.
- 4. Updated "Ordering Information" on page 336:
 - Commercial ordering codes removed.
 - Non Pb-free package option removed.

Rev. 2466P-08/07

- 1. Updated "Features" on page 1.
- 2. Added "Data Retention" on page 6.
- 3. Updated "Errata" on page 340.
- 4. Updated "Slave Mode" on page 140.

Rev. 2466O-03/07

- 1. Updated "Calibrated Internal RC Oscillator" on page 29.
- 2. Updated C code example in "USART Initialization" on page 149.
- 3. Updated "ATmega16 Boundary-scan Order" on page 241.
- 4. Removed "premilinary" from "ADC Characteristics" on page 297.
- 5. Updated from V to mV in "I/O Pin Input Hysteresis vs. VCC" on page 317.
- 6. Updated from V to mV in "Reset Input Pin Hysteresis vs. VCC" on page 318.



345



Rev. 2466N-10/06

- 1. Updated "Timer/Counter Oscillator" on page 31.
- 2. Updated "Fast PWM Mode" on page 102.
- 3. Updated Table 38 on page 83, Table 40 on page 84, Table 45 on page 111, Table 47 on page 112, Table 50 on page 128 and Table 52 on page 129.
- 4. Updated C code example in "USART Initialization" on page 149.
- 5. Updated "Errata" on page 340.

Rev. 2466M-04/06

- 1. Updated typos.
- 2. Updated "Serial Peripheral Interface SPI" on page 135.
- Updated Table 86 on page 221, Table 116 on page 276, Table 121 on page 295 and Table 122 on page 297.

Rev. 2466L-06/05

- 1. Updated note in "Bit Rate Generator Unit" on page 178.
- 2. Updated values for V_{INT} in "ADC Characteristics" on page 297.
- 3. Updated "Serial Programming Instruction set" on page 276.
- 4. Updated USART init C-code example in "USART" on page 144.

Rev. 2466K-04/05

- 1. Updated "Ordering Information" on page 336.
- MLF-package alternative changed to "Quad Flat No-Lead/Micro Lead Frame Package QFN/MLF".
- 3. Updated "Electrical Characteristics" on page 291.

Rev. 2466J-10/04

1. Updated "Ordering Information" on page 336.

Rev. 2466I-10/04

- 1. Removed references to analog ground.
- Updated Table 7 on page 28, Table 15 on page 38, Table 16 on page 42, Table 81 on page 209, Table 116 on page 276, and Table 119 on page 293.
- 3. Updated "Pinout ATmega16" on page 2.
- 4. Updated features in "Analog to Digital Converter" on page 204.
- 5. Updated "Version" on page 229.
- 6. Updated "Calibration Byte" on page 261.
- 7. Added "Page Size" on page 262.

Rev. 2466H-12/03

1. Updated "Calibrated Internal RC Oscillator" on page 29.



346



Rev. 2466G-10/03

- 1. Removed "Preliminary" from the datasheet.
- 2. Changed ICP to ICP1 in the datasheet.
- 3. Updated "JTAG Interface and On-chip Debug System" on page 36.
- Updated assembly and C code examples in "Watchdog Timer Control Register WDTCR" on page 43.
- 5. Updated Figure 46 on page 103.
- 6. Updated Table 15 on page 38, Table 82 on page 217 and Table 115 on page 276.
- 7. Updated "Test Access Port TAP" on page 222 regarding JTAGEN.
- 8. Updated description for the JTD bit on page 231.
- 9. Added note 2 to Figure 126 on page 252.
- 10. Added a note regarding JTAGEN fuse to Table 105 on page 260.
- Updated Absolute Maximum Ratings* and DC Characteristics in "Electrical Characteristics" on page 291.
- 12. Updated "ATmega16 Typical Characteristics" on page 299.
- 13. Fixed typo for 16 MHz QFN/MLF package in "Ordering Information" on page 336.
- 14. Added a proposal for solving problems regarding the JTAG instruction IDCODE in "Errata" on page 340.

Rev. 2466F-02/03

- 1. Added note about masking out unused bits when reading the Program Counter in "Stack Pointer" on page 12.
- Added Chip Erase as a first step in "Programming the Flash" on page 288 and "Programming the EEPROM" on page 289.
- 3. Added the section "Unconnected pins" on page 55.
- Added tips on how to disable the OCD system in "On-chip Debug System" on page 34.
- Removed reference to the "Multi-purpose Oscillator" application note and "32 kHz Crystal Oscillator" application note, which do not exist.
- 6. Added information about PWM symmetry for Timer0 and Timer2.
- Added note in "Filling the Temporary Buffer (Page Loading)" on page 253 about writing to the EEPROM during an SPM Page Load.
- 8. Removed ADHSM completely.



347



- Added Table 73, "TWI Bit Rate Prescaler," on page 182 to describe the TWPS bits in the "TWI Status Register – TWSR" on page 181.
- 10. Added section "Default Clock Source" on page 25.
- Added note about frequency variation when using an external clock. Note added in "External Clock" on page 31. An extra row and a note added in Table 118 on page 293.
- 12. Various minor TWI corrections.
- 13. Added "Power Consumption" data in "Features" on page 1.
- 14. Added section "EEPROM Write During Power-down Sleep Mode" on page 22.
- 15. Added note about Differential Mode with Auto Triggering in "Prescaling and Conversion Timing" on page 207.
- 16. Added updated "Packaging Information" on page 337.

Rev. 2466E-10/02

1. Updated "DC Characteristics" on page 291.

Rev. 2466D-09/02

- 1. Changed all Flash write/erase cycles from 1,000 to 10,000.
- Updated the following tables: Table 4 on page 26, Table 15 on page 38, Table 42 on page 85, Table 45 on page 111, Table 46 on page 111, Table 59 on page 143, Table 67 on page 167, Table 90 on page 235, Table 102 on page 258, "DC Characteristics" on page 291, Table 119 on page 293, Table 121 on page 295, and Table 122 on page 297.
- 3. Updated "Errata" on page 340.

Rev. 2466C-03/02

- 1. Updated typical EEPROM programming time, Table 1 on page 20.
- 2. Updated typical start-up time in the following tables:

Table 3 on page 25, Table 5 on page 27, Table 6 on page 28, Table 8 on page 29, Table 9 on page 29, and Table 10 on page 29.

- 3. Updated Table 17 on page 43 with typical WDT Time-out.
- 4. Added Some Preliminary Test Limits and Characterization Data.

Removed some of the TBD's in the following tables and pages:

Table 15 on page 38, Table 16 on page 42, Table 116 on page 272 (table removed in document review #D), "Electrical Characteristics" on page 291, Table 119 on page 293, Table 121 on page 295, and Table 122 on page 297.

5. Updated TWI Chapter.

Added the note at the end of the "Bit Rate Generator Unit" on page 178.

- Corrected description of ADSC bit in "ADC Control and Status Register A ADCSRA" on page 219.
- Improved description on how to do a polarity check of the ADC doff results in "ADC Conversion Result" on page 216.



348



- 8. Added JTAG version number for rev. H in Table 87 on page 229.
- 9. Added not regarding OCDEN Fuse below Table 105 on page 260.
- 10. Updated Programming Figures:

Figure 127 on page 262 and Figure 136 on page 274 are updated to also reflect that AVCC must be connected during Programming mode. Figure 131 on page 270 added to illustrate how to program the fuses.

- 11. Added a note regarding usage of the "PROG_PAGELOAD (\$6)" on page 280 and "PROG_PAGEREAD (\$7)" on page 280.
- **12. Removed alternative algortihm for leaving JTAG Programming mode.** See "Leaving Programming Mode" on page 288.
- 13. Added Calibrated RC Oscillator characterization curves in section "ATmega16 Typical Characteristics" on page 299.
- 14. Corrected ordering code for QFN/MLF package (16 MHz) in "Ordering Information" on page 336.
- 15. Corrected Table 90, "Scan Signals for the Oscillators(1)(2)(3)," on page 235.





Table of Contents

Features 1

Pin Configurations 2

Disclaimer 2

Overview 3

Block Diagram 3 Pin Descriptions 4

Resources 6

Data Retention 6

About Code Examples 7

AVR CPU Core 8

Introduction 8
Architectural Overview 8
ALU – Arithmetic Logic Unit 9
Status Register 9
General Purpose Register File 11
Stack Pointer 12
Instruction Execution Timing 13
Reset and Interrupt Handling 13

AVR ATmega16 Memories 16

In-System Reprogrammable Flash Program Memory 16 SRAM Data Memory 17 EEPROM Data Memory 18 I/O Memory 23

System Clock and Clock Options 24

Clock Systems and their Distribution 24
Clock Sources 25
Default Clock Source 25
Crystal Oscillator 25
Low-frequency Crystal Oscillator 28
External RC Oscillator 28
Calibrated Internal RC Oscillator 29
External Clock 31
Timer/Counter Oscillator 31

Power Management and Sleep Modes 32

AMEL





Idle Mode 33
ADC Noise Reduction Mode 33
Power-down Mode 33
Power-save Mode 33
Standby Mode 34
Extended Standby Mode 34
Minimizing Power Consumption 35

System Control and Reset 37

Internal Voltage Reference 42 Watchdog Timer 42

Interrupts 45

Interrupt Vectors in ATmega16 45

I/O Ports 50

Introduction 50 Ports as General Digital I/O 50 Alternate Port Functions 55 Register Description for I/O Ports 66

External Interrupts 68

8-bit Timer/Counter0 with PWM 71

Overview 71
Timer/Counter Clock Sources 72
Counter Unit 72
Output Compare Unit 73
Compare Match Output Unit 74
Modes of Operation 76
Timer/Counter Timing Diagrams 81
8-bit Timer/Counter Register Description 83

Timer/Counter0 and Timer/Counter1 Prescalers 87

16-bit Timer/Counter1 89

Overview 89
Accessing 16-bit Registers 92
Timer/Counter Clock Sources 94
Counter Unit 95
Input Capture Unit 96
Output Compare Units 98
Compare Match Output Unit 100
Modes of Operation 101
Timer/Counter Timing Diagrams 108
16-bit Timer/Counter Register Description 110

ATmega16(L)

2466T-AVR-07/10

ii



8-bit Timer/Counter2 with PWM and Asynchronous Operation 117

Overview 117

Timer/Counter Clock Sources 118

Counter Unit 118

Output Compare Unit 119

Compare Match Output Unit 121

Modes of Operation 122

Timer/Counter Timing Diagrams 126

8-bit Timer/Counter Register Description 128

Asynchronous Operation of the Timer/Counter 131

Timer/Counter Prescaler 134

Serial Peripheral Interface - SPI 135

SS Pin Functionality 140

Data Modes 143

USART 144

Overview 144

Clock Generation 145

Frame Formats 148

USART Initialization 149

Data Reception - The USART Receiver 154

Asynchronous Data Reception 157

Multi-processor Communication Mode 161

Accessing UBRRH/ UCSRC Registers 162

USART Register Description 163

Examples of Baud Rate Setting 168

Two-wire Serial Interface 172

Features 172

Two-wire Serial Interface Bus Definition 172

Data Transfer and Frame Format 173

Multi-master Bus Systems, Arbitration and Synchronization 176

Overview of the TWI Module 178

TWI Register Description 180

Using the TWI 183

Transmission Modes 186

Multi-master Systems and Arbitration 199

Analog Comparator 201

Analog Comparator Multiplexed Input 203

Analog to Digital Converter 204

Features 204

Operation 205

Starting a Conversion 206

AMEL

iii





Prescaling and Conversion Timing 207 Changing Channel or Reference Selection 210 ADC Noise Canceler 211 ADC Conversion Result 216

Features 222

JTAG Interface and On-chip Debug System 222

Overview 222
Test Access Port – TAP 222
TAP Controller 224
Using the Boundary-scan Chain 225
Using the On-chip Debug System 225
On-chip Debug Specific JTAG Instructions 226
On-chip Debug Related Register in I/O Memory 227
Using the JTAG Programming Capabilities 227
Bibliography 227

IEEE 1149.1 (JTAG) Boundary-scan 228

Features 228
System Overview 228
Data Registers 228
Boundary-scan Specific JTAG Instructions 230
Boundary-scan Chain 232
ATmega16 Boundary-scan Order 241
Boundary-scan Description Language Files 245

Boot Loader Support - Read-While-Write Self-Programming 246

Features 246
Application and Boot Loader Flash Sections 246
Read-While-Write and no Read-While-Write Flash Sections 246
Boot Loader Lock Bits 248
Entering the Boot Loader Program 249
Addressing the Flash during Self-Programming 251
Self-Programming the Flash 252

Memory Programming 259

Frogram And Data Memory Lock Bits 259
Fuse Bits 260
Signature Bytes 261
Calibration Byte 261
Page Size 262
Parallel Programming Parameters, Pin Mapping, and Commands 262
Parallel Programming 265
Serial Downloading 273
Programming via the JTAG Interface 278

ATmega16(L)



Electrical Characteristics 291

Absolute Maximum Ratings* 291
DC Characteristics 291
External Clock Drive Waveforms 293
External Clock Drive 293
Two-wire Serial Interface Characteristics 294
SPI Timing Characteristics 295
ADC Characteristics 297

ATmega16 Typical Characteristics 299

Register Summary 331

Instruction Set Summary 333

Ordering Information 336

Packaging Information 337

44A 337 40P6 338 44M1 339

Errata 340

ATmega16(L) Rev. M 340 ATmega16(L) Rev. L 340 ATmega16(L) Rev. K 341 ATmega16(L) Rev. J 342 ATmega16(L) Rev. I 343 ATmega16(L) Rev. H 344

Datasheet Revision History 345

Rev. 2466T-07/10 345
Rev. 2466S-05/09 345
Rev. 2466R-06/08 345
Rev. 2466Q-05/08 345
Rev. 2466P-08/07 345
Rev. 2466O-03/07 345
Rev. 2466N-10/06 346
Rev. 2466M-04/06 346
Rev. 2466K-04/05 346
Rev. 2466K-04/05 346
Rev. 2466I-10/04 346
Rev. 2466H-12/03 346
Rev. 2466G-10/03 347
Rev. 2466F-02/03 347

<u>AIMEL</u>





Rev. 2466E-10/02 348 Rev. 2466D-09/02 348 Rev. 2466C-03/02 348

Table of Contents i

vi ATmega16(L)





vii





Headquarters

Atmel Corporation

2325 Orchard Parkway San Jose, CA 95131

Tel: 1(408) 441-0311 Fax: 1(408) 487-2600

International

Atmel Asia

Unit 1-5 & 16, 19/F BEA Tower, Millennium City 5 418 Kwun Tong Road Kwun Tong, Kowloon Hong Kong Tel: (852) 2245-6100 Fax: (852) 2722-1369

France Tel: (33) 1-30-60-70-00 Fax: (33) 1-30-60-71-11

78054 Saint-Quentin-en-

8, Rue Jean-Pierre Timbaud

Atmel Europe

Yvelines Cedex

Le Krebs

BP 309

Atmel Japan 9F, Tonetsu Shinkawa Bldg. 1-24-8 Shinkawa Chuo-ku, Tokyo 104-0033 Japan Tel: (81) 3-3523-3551 Fax: (81) 3-3523-7581

Product Contact

www.atmel.com

Technical Support avr@atmel.com

Sales Contact

www.atmel.com/contacts

Literature Requests www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, ITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION), DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2010 Atmel Corporation. All rights reserved. Atmel®, Atmel logo and combinations thereof, AVR® and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others



HEF40106B

Hex inverting Schmitt trigger

Rev. 7 — 21 November 2011

Product data sheet

1. General description

The HEF40106B provides six inverting buffers. Each input has a Schmitt trigger circuit. The inverting buffer switches at different points for positive-going and negative-going signals. The difference between the positive voltage (V_{T+}) and the negative voltage (V_{T-}) is defined as hysteresis voltage (V_H) .

The HEF40106B may be used for enhanced noise immunity or to "square up" slowly changing waveforms.

It operates over a recommended V_{DD} power supply range of 3 V to 15 V referenced to V_{SS} (usually ground). Unused inputs must be connected to V_{DD} , V_{SS} , or another input.

2. Features and benefits

- Schmitt trigger input discrimination
- Fully static operation
- 5 V, 10 V, and 15 V parametric ratings
- Standardized symmetrical output characteristics
- Specified from -40 °C to +125 °C
- Complies with JEDEC standard JESD 13-B

3. Applications

- Wave and pulse shapers
- Astable multivibrators
- Monostable multivibrators

4. Ordering information

Table 1. Ordering information

All types operate from -40 °C to +125 °C

Type number	Package					
	Name Description					
HEF40106BP	DIP14	plastic dual in-line package; 14 leads (300 mil)	SOT27-1			
HEF40106BT	SO14	plastic small outline package; 14 leads; body width 3.9 mm	SOT108-1			
HEF40106BTT	TSSOP14	plastic thin shrink small outline package; 14 leads; body width 4.4 mm	SOT402-1			

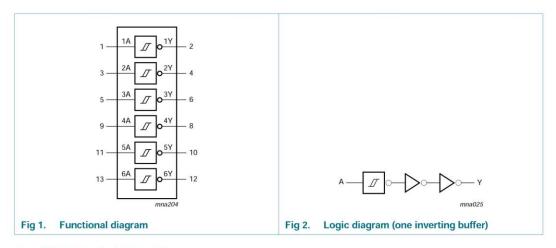




HEF40106B

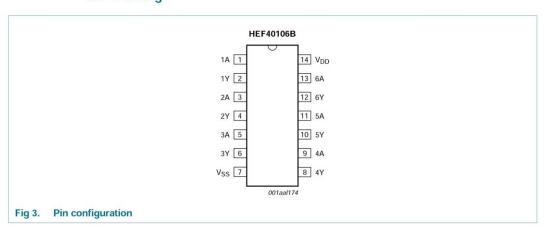
Hex inverting Schmitt trigger

5. Functional diagram



6. Pinning information

6.1 Pinning



HEF40106B

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2011. All rights reserved.



HEF40106B

Hex inverting Schmitt trigger

6.2 Pin description

Table 2. Pin description

Symbol	Pin	Description
1A to 6A	1, 3, 5, 9, 11, 13	input
1Y to 6Y	2, 4, 6, 8, 10, 12	output
V_{DD}	14	supply voltage
V_{SS}	7	ground (0 V)

7. Functional description

Table 3. Function table[1]

Input	Output
nA	nY
L	Н
Н	L

^[1] H = HIGH voltage level; L = LOW voltage level.

8. Limiting values

Table 4. Limiting values

In accordance with the Absolute Maximum Rating System (IEC 60134). Voltages are referenced to $V_{SS} = 0 \ V$ (ground).

Symbol	Parameter	Conditions	Min	Max	Unit
V_{DD}	supply voltage		-0.5	+18	V
I _{IK}	input clamping current	$V_I < -0.5 \text{ V or } V_I > V_{DD} + 0.5 \text{ V}$	-	±10	mA
VI	input voltage		-0.5	$V_{DD} + 0.5$	V
I _{OK}	output clamping current	$V_O < -0.5 \text{ V or } V_O > V_{DD} + 0.5 \text{ V}$	-	±10	mA
I _{I/O}	input/output current			±10	mA
I_{DD}	supply current		-	50	mA
T_{stg}	storage temperature		-65	+150	°C
T _{amb}	ambient temperature		-40	+125	°C
P_{tot}	total power dissipation	$T_{amb} = -40 ^{\circ}\text{C} \text{ to } +125 ^{\circ}\text{C}$			
		DIP14	[1] -	750	mW
			[2] -	500	mW
		TSSOP14	[3]	500	mW
P	power dissipation	per output	-	100	mW

^[1] For DIP14 packages: above T_{amb} = 70 °C, P_{tot} derates linearly with 12 mW/K.

HEF40106

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2011. All rights reserved.

Product data sheet

Rev. 7 — 21 November 2011

3 of 16

^[2] For SO14 packages: above T_{amb} = 70 °C, P_{tot} derates linearly with 8 mW/K.

^[3] For TSSOP14 packages: above T_{amb} = 60 °C, P_{tot} derates linearly with 5.5 mW/K.



HEF40106B

Hex inverting Schmitt trigger

9. Recommended operating conditions

Table 5. Recommended operating conditions

	. 3				
Symbol	Parameter	Conditions	Min	Max	Unit
V_{DD}	supply voltage		3	15	V
VI	input voltage		0	V_{DD}	V
T _{amb}	ambient temperature	in free air	-40	+125	°C

10. Static characteristics

Table 6. Static characteristics

 V_{SS} = 0 V; V_{I} = V_{SS} or V_{DD} ; unless otherwise specified.

Symbol	Parameter	Conditions	V_{DD}	T _{amb} =	-40 °C	T _{amb} =	+25 °C	T _{amb} =	+85 °C	T _{amb} = +125 °C		Unit
				Min	Max	Min	Max	Min	Max	Min	Max	
V _{OH}	HIGH-level	$ I_{O} < 1 \mu A$	5 V	4.95	-	4.95	-	4.95	-	4.95	-	V
	output voltage		10 V	9.95	-	9.95	-	9.95	-	9.95	-	V
			15 V	14.95	-	14.95	-	14.95	ū.	14.95	-	V
V _{OL}	LOW-level	$ I_O < 1 \mu A$	5 V	9	0.05	-	0.05	-	0.05	4	0.05	V
	output voltage		10 V	-	0.05		0.05	-	0.05	8	0.05	V
			15 V	2	0.05		0.05		0.05	-	0.05	V
I _{OH}	HIGH-level output current	$V_{O} = 2.5 \text{ V}$	5 V	Ħ	-1.7		-1.4	•	-1.1	-	-1.1	mA
		$V_{O} = 4.6 \text{ V}$	5 V	-	-0.64		-0.5	-	-0.36	-	-0.36	mA
		$V_{O} = 9.5 V$	10 V	-	-1.6		-1.3	-	-0.9	-	-0.9	mA
		$V_{O} = 13.5 \text{ V}$	15 V	-	-4.2	-	-3.4	141	-2.4	-	-2.4	mA
I _{OL}	LOW-level	$V_{O} = 0.4 \ V$	5 V	0.64	-	0.5	-	0.36	2	0.36	1341	mA
	output current	$V_{O} = 0.5 \ V$	10 V	1.6	-	1.3	- 12	0.9	2	0.9	12:	mA
		$V_{O} = 1.5 V$	15 V	4.2	-	3.4	-	2.4	-	2.4	-	mA
I _I	input leakage current		15 V	-	±0.1	-	±0.1	•	±1.0		±1.0	μА
I _{DD}	supply current	all valid input	5 V	-	0.25	-	0.25	-	7.5	-	7.5	μΑ
		combinations;	10 V	-	0.5		0.5	-	15.0	-	15.0	μΑ
		$I_O = 0 A$	15 V		1.0		1.0	-	30.0		30.0	μΑ
CI	input capacitance			-	-	-	7.5	-	-	-	-	pF



HEF40106B

Hex inverting Schmitt trigger

11. Dynamic characteristics

Table 7. Dynamic characteristics

 $T_{amb} = 25 \,^{\circ}$ C; $C_L = 50 \, pF$; $t_f = t_f \le 20 \, ns$; wave forms see Figure 4; test circuit see Figure 5; unless otherwise specified.

Cirric								
Symbol	Parameter	Conditions	V_{DD}	Extrapolation formula[1]	Min	Тур	Max	Unit
t _{PHL}	HIGH to LOW	nA or nB to nY	5 V	63 ns + (0.55 ns/pF)C _L	-	90	180	ns
	propagation delay		10 V	29 ns + (0.23 ns/pF)C _L	-	35	70	ns
			15 V	22 ns + (0.16 ns/pF)C _L	-	30	60	ns
t _{PLH}	LOW to HIGH	nA or nB to nY	5 V	58 ns + (0.55 ns/pF)C _L	-	75	150	ns
	propagation delay		10 V	29 ns + (0.23 ns/pF)C _L	<u> (10</u>)	35	70	ns
			15 V	22 ns + (0.16 ns/pF)C _L	-	30	60	ns
t _{THL}	HIGH to LOW output transition time	nY to LOW	5 V	10 ns + (1.00 ns/pF)C _L	150	60	120	ns
			10 V	9 ns + (0.42 ns/pF)C _L		30	60	ns
			15 V	6 ns + (0.28 ns/pF)C _L	-	20	40	ns
t _{TLH}	LOW to HIGH output	nA or nB to	5 V	10 ns + (1.00 ns/pF)C _L		60	120	ns
	transition time	HIGH	10 V	9 ns + $(0.42 \text{ ns/pF})C_L$		30	60	ns
			15 V	6 ns + (0.28 ns/pF)C _L		20	40	ns

^[1] Typical value of the propagation delay and output transition time can be calculated with the extrapolation formula (C_L in pF).

Table 8. Dynamic power dissipation

 $V_{SS} = 0 \ V; \ t_r = t_f \le 20 \ ns; \ T_{amb} = 25 \ ^{\circ}C.$

Symbol	Parameter	V_{DD}	Typical formula	where:			
	dynamic power	5 V	$P_D = 2300 \times f_i + \Sigma (f_o \times C_L) \times V_{DD}^2 (\mu W)$	f _i = input frequency in MHz;			
	dissipation	10 V	$P_D = 9000 \times f_i + \Sigma (f_o \times C_L) \times V_{DD}^2 (\mu W)$	fo = output frequency in MHz;			
		15 V	$P_D = 20000 \times f_i + \Sigma (f_o \times C_L) \times V_{DD}^2 (\mu W)$	C_L = output load capacitance in pF; $\Sigma(f_0 \times C_L)$ = sum of the outputs; V_{DD} = supply voltage in V.			

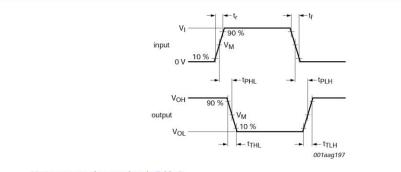
Product data sheet



HEF40106B

Hex inverting Schmitt trigger

12. Waveforms



Measurement points are given in Table 9.

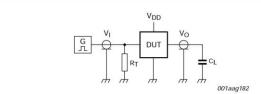
Logic levels: V_{OL} and V_{OH} are typical output voltage levels that occur with the output load.

 t_r , t_f = input rise and fall times.

Fig 4. Propagation delay and output transition time

Table 9. Measurement points

Supply voltage	Input	Output	
V _{DD}	V _M	V _M	
5 V to 15 V	0.5V _{DD}	0.5V _{DD}	



Test data given in Table 10.

Definitions for test circuit:

DUT = Device Under Test.

 $\ensuremath{C_L}$ = load capacitance including jig and probe capacitance.

 R_T = termination resistance should be equal to the output impedance Z_o of the pulse generator.

Fig 5. Test circuit

Table 10. Test data

Supply voltage	Input	Load	
V _{DD}	VI	t _r , t _f	CL
5 V to 15 V	V _{SS} or V _{DD}	≤ 20 ns	50 pF

HEF40106

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2011. All rights reserved

Product data sheet

Rev. 7 — 21 November 2011

6 of 16



HEF40106B

Hex inverting Schmitt trigger

13. Transfer characteristics

Table 11. Transfer characteristics $V_{SS} = 0 V$; see <u>Figure 6</u> and <u>Figure 7</u>.

Symbol	Parameter	Conditions	V _{DD}	T _{amb} =	T _{amb} = to +1	Unit			
				Min	Typ[1]	Max	Min	Max	
V_{T+}	positive-going threshold voltage		5 V	2.0	3.0	3.5	2.0	3.5	V
			10 V	3.7	5.8	7.0	3.7	7.0	V
			15 V	4.9	8.3	11.0	4.9	11.0	V
V_{T-}	negative-going threshold voltage		5 V	1.5	2.2	3.0	1.5	3.0	V
			10 V	3.0	4.5	6.3	3.0	6.3	V
			15 V	4.0	6.5	10.1	4.0	3.0 6.3 10.1	V
V _H	hysteresis voltage		5 V	0.5	0.8	-	0.5	-	V
			10 V	0.7	1.3	-	0.7	1-0	V
			15 V	0.9	1.8	-	0.9	-	V

[1] All typical values are at T_{amb} = 25 °C.

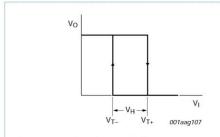


Fig 6. Transfer characteristic

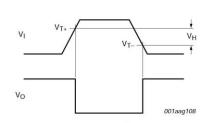


Fig 7. Waveforms showing definition of V_{T+} and V_{T-} (between limits at 30 % and 70 %) and V_H

Product data sheet

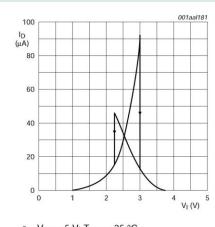
7 of 16

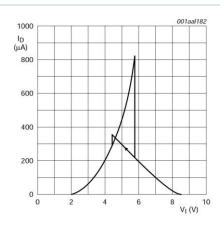
No olvide citar esta tesis



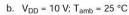
HEF40106B

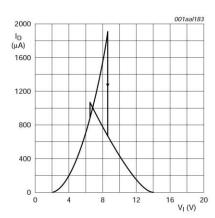
Hex inverting Schmitt trigger





a. $V_{DD} = 5 \text{ V}$; $T_{amb} = 25 ^{\circ}\text{C}$





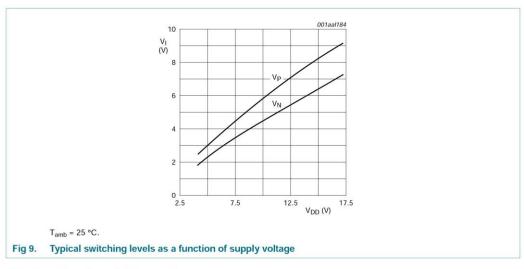
c. $V_{DD} = 15 \text{ V}$; $T_{amb} = 25 \text{ °C}$

Fig 8. Typical drain current as a function of input



HEF40106B

Hex inverting Schmitt trigger



14. Application information

Some examples of applications for the HEF40106B are:

- · Wave and pulse shapers
- · Astable multivibrators
- · Monostable multivibrators



If a Schmitt trigger is driven via a high-impedance (R > 1 k Ω), then it is necessary to incorporate a capacitor C with a value of $\frac{c}{C_P} > \frac{V_{DD} - V_{SS}}{V_H}$; otherwise oscillation can occur on the edges of a pulse.

 $\boldsymbol{C}_{\boldsymbol{p}}$ is the external parasitic capacitance between inputs and output; the value depends on the circuit board layout.

HEF40106B

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2011. All rights reserved.

Product data sheet

Rev. 7 — 21 November 2011

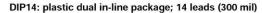
9 of 16



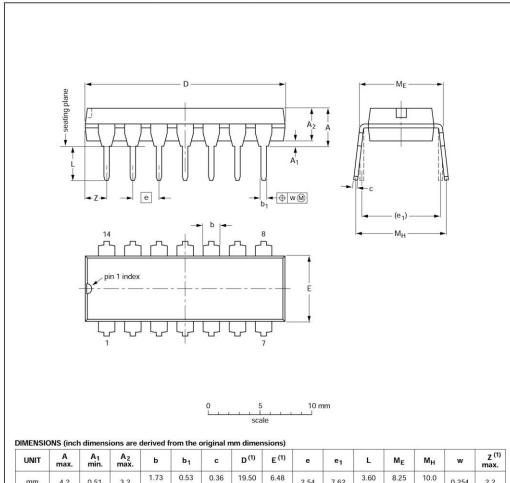
HEF40106B

Hex inverting Schmitt trigger

15. Package outline



SOT27-1



UNIT	A max.	A ₁ min.	A ₂ max.	b	b ₁	С	D (1)	E ⁽¹⁾	е	e ₁	L	ME	Мн	w	Z ⁽¹⁾ max.
mm	4.2	0.51	3.2	1.73 1.13	0.53 0.38	0.36 0.23	19.50 18.55	6.48 6.20	2.54	7.62	3.60 3.05	8.25 7.80	10.0 8.3	0.254	2.2
inches	0.17	0.02	0.13	0.068 0.044	0.021 0.015	0.014 0.009	0.77 0.73	0.26 0.24	0.1	0.3	0.14 0.12	0.32 0.31	0.39 0.33	0.01	0.087

Note

1. Plastic or metal protrusions of 0.25 mm (0.01 inch) maximum per side are not included.

OUTLINE		REFE	RENCES	EUROPEAN	ISSUE DATE
VERSION	IEC	JEDEC	JEITA	PROJECTION	
SOT27-1	050G04	MO-001	SC-501-14	€ ⊕	99-12-27 03-02-13

Fig 12. Package outline SOT27-1 (DIP14)

HEF40106B All information provided in this document is subject to legal disclaimers. ONXP B.V. 2011. All rights reserved.

Product data sheet Rev. 7 — 21 November 2011 10 of 16



HEF40106B

Hex inverting Schmitt trigger

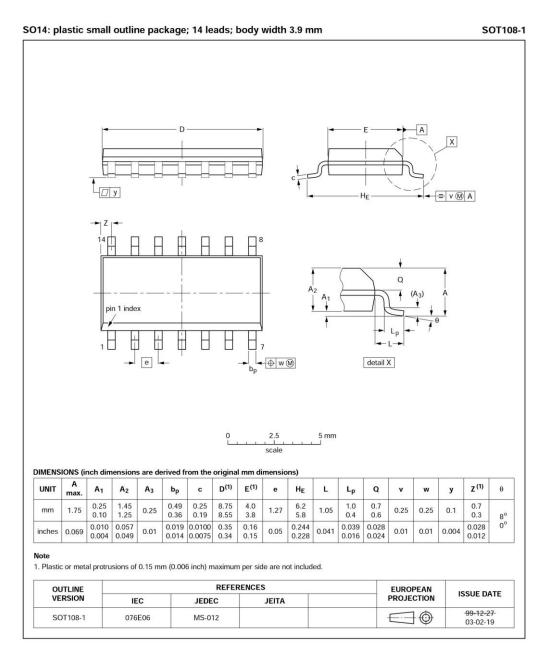


Fig 13. Package outline SOT108-1 (SO14)

HEF40106B All information provided in this document is subject to legal disclaimers.

• NXP B.V. 2011. All rights reserved.

Product data sheet

Rev. 7 — 21 November 2011

11 of 16



HEF40106B

Hex inverting Schmitt trigger

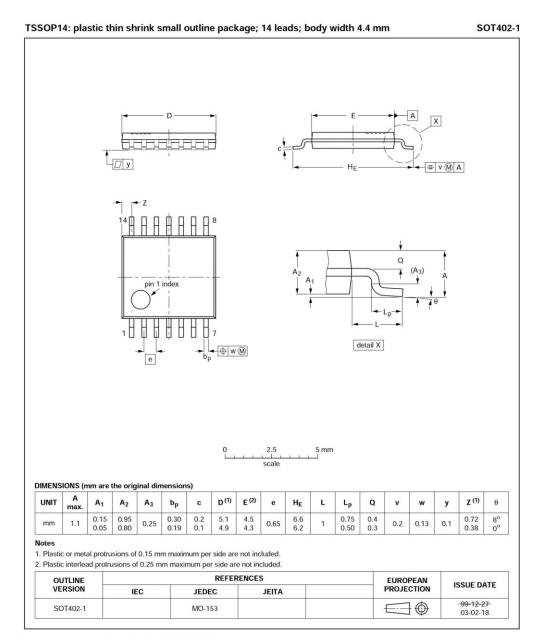


Fig 14. Package outline SOT402-1 (TSSOP14)

HEF401068 All information provided in this document is subject to legal disclaimers.

• NXP B.V. 2011. All rights reserved.

Product data sheet Rev. 7 — 21 November 2011 12 of 16



HEF40106B

Hex inverting Schmitt trigger

16. Revision history

Table 12. Revision history

Document ID	Release date	Data sheet status	Change notice	Supersedes
HEF40106B v.7	20111121	Product data sheet		HEF40106B v.6
Modifications:	 Legal pages 	s updated.		
	 Changes in 	"General description" and "F	eatures and benefits".	
HEF40106B v.6	20110823	Product data sheet		HEF40106B v.5
HEF40106B v.5	20110511	Product data sheet	:#	HEF40106B v.4
HEF40106B v.4	20101115	Product data sheet	192	HEF40106B_CNV v.3
HEF40106B_CNV v.3	19950101	Product specification	TV .	HEF40106B_CNV v.2
HEF40106B_CNV v.2	19950101	Product specification	-	-



HEF40106B

Hex inverting Schmitt trigger

17. Legal information

17.1 Data sheet status

Document status[1][2]	Product status[3]	Definition
Objective [short] data sheet	Development	This document contains data from the objective specification for product development.
Preliminary [short] data sheet	Qualification	This document contains data from the preliminary specification.
Product [short] data sheet	Production	This document contains the product specification.

- [1] Please consult the most recently issued document before initiating or completing a design.
- [2] The term 'short data sheet' is explained in section "Definitions"
- [3] The product status of device(s) described in this document may have changed since this document was published and may differ in case of multiple devices. The latest product status information is available on the Internet at URL https://www.nxp.com.

17.2 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

Short data sheet — A short data sheet is an extract from a full data sheet with the same product type number(s) and title. A short data sheet is intended for quick reference only and should not be relied upon to contain detailed and full information. For detailed and full information see the relevant full data sheet, which is available on request via the local NXP Semiconductors sales office. In case of any inconsistency or conflict with the short data sheet, the full data sheet shall prevail.

Product specification — The information and data provided in a Product data sheet shall define the specification of the product as agreed between NXP Semiconductors and its customer, unless NXP Semiconductors and customer have explicitly agreed otherwise in writing. In no event however, shall an agreement be valid in which the NXP Semiconductors product is deemed to offer functions and qualities beyond those described in the Product data sheet.

17.3 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or

malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Limiting values — Stress above one or more limiting values (as defined in the Absolute Maximum Ratings System of IEC 60134) will cause permanent damage to the device. Limiting values are stress ratings only and (proper) operation of the device at these or any other conditions above those given in the Recommended operating conditions section (if present) or the Characteristics sections of this document is not warranted. Constant or repeated exposure to limiting values will permanently and irreversibly affect the quality and reliability of the device.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

No offer to sell or license — Nothing in this document may be interpreted or construed as an offer to sell products that is open for acceptance or the grant, conveyance or implication of any license under any copyrights, patents or other industrial or intellectual property rights.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

HEF40106B

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2011. All rights reserved

Product data sheet

Rev. 7 — 21 November 2011



HEF40106B

Hex inverting Schmitt trigger

Non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond

NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

17.4 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

18. Contact information

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com



NXP Semiconductors

HEF40106B

Hex inverting Schmitt trigger

19. Contents

1	General description
2	Features and benefits
3	Applications
4	Ordering information 1
5	Functional diagram
6	Pinning information 2
6.1	Pinning
6.2	Pin description
7	Functional description 3
8	Limiting values 3
9	Recommended operating conditions 4
10	Static characteristics 4
11	Dynamic characteristics 5
12	Waveforms
13	Transfer characteristics
14	Application information 9
15	Package outline
16	Revision history
17	Legal information14
17.1	Data sheet status
17.2	Definitions 14
17.3	Disclaimers
17.4	Trademarks
18	Contact information
19	Contents

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© NXP B.V. 2011.

All rights reserved.

For more information, please visit: http://www.nxp.com For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 21 November 2011 Document identifier: HEF40106B





N - CHANNEL 400V - 0.48 W - 10 A - TO-220 PowerMESHÔ MOSFET

TYPE	V _{DSS}	R _{DS(on)}	I _D	
IRF740	400 V	< 0.55 W	10 A	

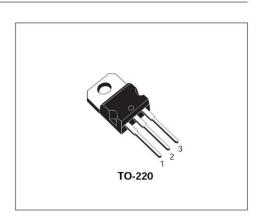
- n TYPICAL R_{DS(on)} = 0.48 W
- EXTREMELY HIGH dv/dt CAPABILITY
- 100% AVALANCHE TESTED
- VERY LOW INTRINSIC CAPACITANCES
- , GATE CHARGE MINIMIZED

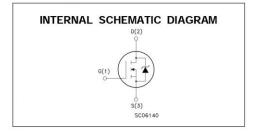
DESCRIPTION

This power MOSFET is designed using the company's consolidated strip layout-based MESH OVERLAYO process. This technology matches and improves the performances compared with standard parts from various sources.

APPLICATIONS

- " HIGH CURRENT SWITCHING
- UNINTERRUPTIBLE POWER SUPPLY (UPS)
- DC/DC COVERTERS FOR TELECOM, INDUSTRIAL, AND LIGHTING EQUIPMENT.





ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V _{DS}	Drain-source Voltage (V _{GS} = 0)	400	V
V_{DGR}	Drain- gate Voltage (R _{GS} = 20 kW)	400	V
V _{GS}	Gate-source Voltage	± 20	V
ID	Drain Current (continuous) at T _c = 25 °C	10	А
ID	Drain Current (continuous) at T _c = 100 °C	6.3	А
I _{DM} (⋅)	Drain Current (pulsed)	40	А
P _{tot}	Total Dissipation at T _c = 25 °C	125	W
	Derating Factor	1.0	W/°C
dv/dt(1)	Peak Diode Recovery voltage slope	4.0	V/ns
T _{stg}	Storage Temperature	-65 to 150	°C
Tj	Max. Operating Junction Temperature	150	°C

(.) Pulse width limited by safe operating area (1) I_{SD} £10 Å, di/dt £120 Å/ns, V_{DD} £ V_{(BR)DSS}, Tj £ T_{JMAX} First Digit of the Datecode Being Z or K Identifies Silicon Characterized in this Datasheet

October 1998 1/8



THERMAL DATA

R _{thj-case}	Thermal Resistance Junction-case	Max	1.0	°C/W
Rthj-amb	Thermal Resistance Junction-ambient	Max	62.5	oC/W
R _{thc-sink}	Thermal Resistance Case-sink	Тур	0.5	°C/W
T ₁	Maximum Lead Temperature For Soldering F	Purpose	300	°C

AVALANCHE CHARACTERISTICS

Symbol	Parameter	Max Value	Unit
I _{AR}	Avalanche Current, Repetitive or Not-Repetitive (pulse width limited by T_j max)	10	Α
E _{AS}	Single Pulse Avalanche Energy (starting $T_j = 25$ °C, $I_D = I_{AR}$, $V_{DD} = 50$ V)	520	mJ

ELECTRICAL CHARACTERISTICS (T_{case} = 25 °C unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Тур.	Max.	Unit
V _{(BR)DSS}	Drain-source Breakdown Voltage	I _D = 250 mA V _{GS} = 0	400			V
I _{DSS}	Zero Gate Voltage Drain Current (V _{GS} = 0)	V_{DS} = Max Rating V_{DS} = Max Rating T_c = 125 o C			1 50	mA mA
I _{GSS}	Gate-body Leakage Current (V _{DS} = 0)	V _{GS} = ± 20 V			± 100	nA

ON (*)

Symbol	Parameter	Test Conditions	Min.	Тур.	Max.	Unit
$V_{GS(th)}$	Gate Threshold Voltage	$V_{DS} = V_{GS}$ $I_D = 250 \text{ mA}$	2	3	4	V
R _{DS(on)}	Static Drain-source On Resistance	$V_{GS} = 10V I_D = 5.3 A$		0.48	0.55	W
I _{D(on)}	On State Drain Current	$V_{DS} > I_{D(on)} \times R_{DS(on)max}$ $V_{GS} = 10 \text{ V}$	10			А

DYNAMIC

Symbol	Parameter	Test Conditions	Min.	Тур.	Max.	Unit
g _{fs} (*)	Forward Transconductance	$V_{DS} > I_{D(on)} \times R_{DS(on)max}$ $I_D = 6 A$	5.8			S
C _{iss} C _{oss} C _{rss}	Input Capacitance Output Capacitance Reverse Transfer Capacitance	V _{DS} = 25 V f = 1 MHz V _{GS} = 0		1400 220 27		pF pF pF



ELECTRICAL CHARACTERISTICS (continued)

SWITCHING ON

Symbol	Parameter	Test Conditions	Min.	Тур.	Max.	Unit
t _{d(on)} t _r	Turn-on Time Rise Time	V_{DD} = 200 V I_D = 5 A R _G = 4.7 W V_{GS} = 10 V (see test circuit, figure 3)		17 10		ns ns
$egin{array}{c} Q_g \ Q_{gs} \ Q_{gd} \end{array}$	Total Gate Charge Gate-Source Charge Gate-Drain Charge	$V_{DD} = 320 \text{ V}$ $I_D = 10.7 \text{ A}$ $V_{GS} = 10 \text{ V}$		35 11 12	43	nC nC nC

SWITCHING OFF

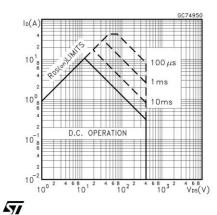
Symbol	Parameter	Test Conditions	Min.	Тур.	Max.	Unit
t _{r(Voff)}	Off-voltage Rise Time	V _{DD} = 320 V I _D = 10 A		10		ns
t_f	Fall Time	$R_G = 4.7 \text{ W } V_{GS} = 10 \text{ V}$		10		ns
t_c	Cross-over Time	(see test circuit, figure 5)		17		ns

SOURCE DRAIN DIODE

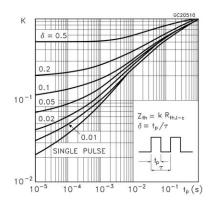
Symbol	Parameter	Test Conditions	Min.	Тур.	Max.	Unit
I _{SD} I _{SDM} (·)	Source-drain Current Source-drain Current (pulsed)				10 40	A A
V _{SD} (*)	Forward On Voltage	I _{SD} = 10 A V _{GS} = 0			1.6	V
t _{rr}	Reverse Recovery Time	I _{SD} = 10 A di/dt = 100 A/ms V _{DD} = 100 V T _i = 150 °C		370		ns
Q_{rr}	Reverse Recovery Charge	(see test circuit, figure 5)		3.2		mС
I_{RRM}	Reverse Recovery Current			17		Α

^(*) Pulsed: Pulse duration = 300 ms, duty cycle 1.5 %
(·) Pulse width limited by safe operating area

Safe Operating Area

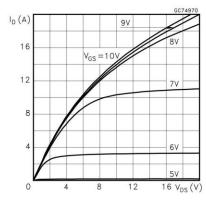


Thermal Impedance

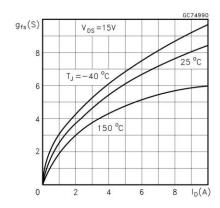




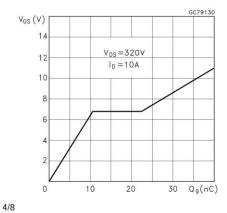
Output Characteristics



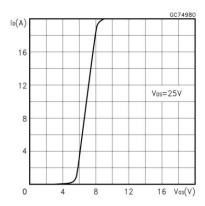
Transconductance



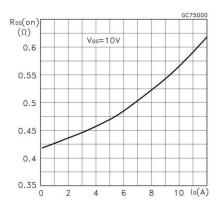
Gate Charge vs Gate-source Voltage



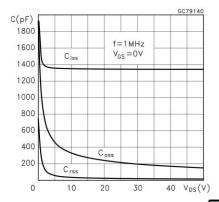
Transfer Characteristics



Static Drain-source On Resistance

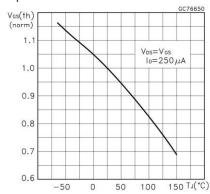


Capacitance Variations

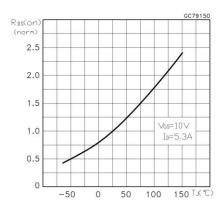




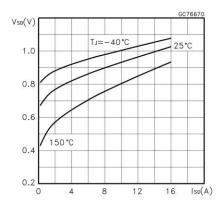
Normalized Gate Threshold Voltage vs Temperature



Normalized On Resistance vs Temperature



Source-drain Diode Forward Characteristics



477



Fig. 1: Unclamped Inductive Load Test Circuit

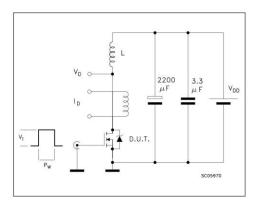


Fig. 3: Switching Times Test Circuits For Resistive Load

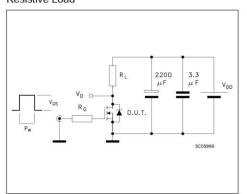


Fig. 5: Test Circuit For Inductive Load Switching And Diode Recovery Times

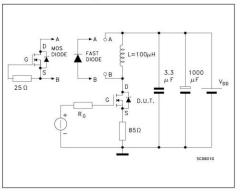


Fig. 1: Unclamped Inductive Waveform

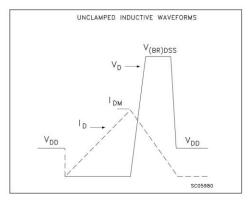
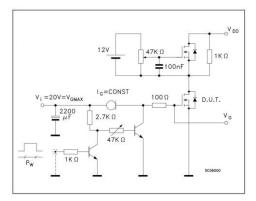


Fig. 4: Gate Charge test Circuit

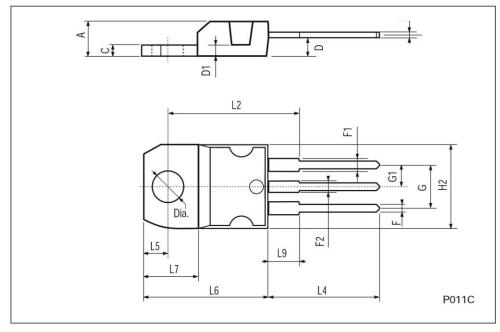


47/



TO-220 MECHANICAL DATA

DIM.		mm			inch			
DIIVI.	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.		
Α	4.40		4.60	0.173		0.181		
С	1.23		1.32	0.048		0.051		
D	2.40		2.72	0.094		0.107		
D1		1.27			0.050			
E	0.49		0.70	0.019		0.027		
F	0.61		0.88	0.024		0.034		
F1	1.14		1.70	0.044		0.067		
F2	1.14		1.70	0.044		0.067		
G	4.95		5.15	0.194		0.203		
G1	2.4		2.7	0.094		0.106		
H2	10.0		10.40	0.393		0.409		
L2		16.4			0.645			
L4	13.0		14.0	0.511		0.551		
L5	2.65		2.95	0.104		0.116		
L6	15.25		15.75	0.600		0.620		
L7	6.2		6.6	0.244		0.260		
L9	3.5		3.93	0.137		0.154		
DIA.	3.75		3.85	0.147		0.151		



477 7/8



Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specification mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of STMicroelectronics.

The ST logo is a registered trademark of STMicroelectronics

© 1998 STMicroelectronics – Printed in Italy – All Rights Reserved STMicroelectronics GROUP OF COMPANIES

Australia - Brazil - Canada - China - France - Germany - Italy - Japan - Korea - Malaysia - Malta - Mexico - Morocco - The Netherlands - Singapore - Spain - Sweden - Switzerland - Taiwan - Thailand - United Kingdom - U.S.A.

http://www.st.com





This datasheet has been download from:

www.datasheetcatalog.com

Datasheets for electronics components.



LM139, LM139A, LM239, LM239A, LM339, LM339A, LM2901, LM2901V QUAD DIFFERENTIAL COMPARATORS

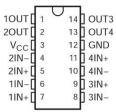
SLCS006L - OCTOBER 1979 - REVISED JUNE 2004

- Single Supply or Dual Supplies
- Wide Range of Supply Voltage:
 - Max Rating . . . 2 V to 36 V
 - Tested to 30 V . . . Non-V Devices
 - Tested to 32 V . . . V-Suffix Devices
- Low Supply-Current Drain Independent of Supply Voltage . . . 0.8 mA Typ
- Low Input Bias Current . . . 25 nA Typ
- Low Input Offset Current . . . 3 nA Typ (LM139)
- Low Input Offset Voltage . . . 2 mV Typ
- Common-Mode Input Voltage Range Includes Ground
- Differential Input Voltage Range Equal to Maximum-Rated Supply Voltage . . . ±36 V
- Low Output Saturation Voltage
- Output Compatible With TTL, MOS, and CMOS

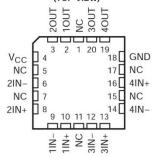
description/ordering information

These devices consist of four independent voltage comparators that are designed to operate from a single power supply over a wide range of voltages. Operation from dual supplies also is possible as long as the difference between the two supplies is 2 V to 36 V, and V_{CC} is at least 1.5 V more positive than the input common-mode voltage. Current drain is independent of the supply voltage. The outputs can be connected to other open-collector outputs to achieve wired-AND relationships.

LM139, LM139A . . . D, J, OR W PACKAGE
LM239 . . . D, N, OR PW PACKAGE
LM239A . . . D PACKAGE
LM339, LM339A . . . D, DB, N, NS, OR PW PACKAGE
LM2901 . . . D, N, NS, OR PW PACKAGE
(TOP VIEW)



LM139, LM139A . . . FK PACKAGE (TOP VIEW)



NC - No internal connection

The LM139 and LM139A are characterized for operation over the full military temperature range of -55°C to 125°C. The LM239 and LM239A are characterized for operation from -25°C to 125°C. The LM339 and LM339A are characterized for operation from 0°C to 70°C. The LM2901 is characterized for operation from -40°C to 125°C.



Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include the control of the company of the control o



Copyright © 2004, Texas Instruments Incorporated On products compliant to MIL-PRF-38535, all parameters are tested unless otherwise noted. On all other products, production processing does not necessarily include testing of all parameters.

1



description/ordering information (continued)

ORDERING INFORMATION

TA	V _{IO} max AT 25°C	MAX V _{CC}	PACK	AGE†	ORDERABLE PART NUMBER	TOP-SIDE MARKING
			PDIP (N)	Tube of 25	LM339N	LM339N
			0.010 (0)	Tube of 50	LM339D	
			SOIC (D)	Reel of 2500	LM339DR	LM339
	5 mV	30 V	SOP (NS)	Reel of 2000	LM339NSR	LM339
			SSOP (DB)	Reel of 2000	LM339DBR	LM339
			T0000 (DIA)	Tube of 90	LM339PW	
			TSSOP (PW)	Reel of 2000	LM339PWR	L339
0°C to 70°C			PDIP (N)	Tube of 25	LM339AN	LM339AN
			0010 (D)	Tube of 50	LM339AD	
			SOIC (D)	Reel of 2500	LM339ADR	LM339A
	2 mV	30 V	SOP (NS)	Reel of 2000	LM339ANSR	LM339A
			SSOP (DB)	Reel of 2000	LM339ADBR	L339A
				Tube of 90	LM339APW	
			TSSOP (PW)	Reel of 2000	LM339APWR	L339A
		30 V	PDIP (N)	Tube of 25	LM239N	LM239N
	5 mV		0010 (D)	Tube of 50	LM239D	
			SOIC (D)	Reel of 2500	LM239DR	LM239
-25°C to 85°C			T0000 (D110)	Tube of 90	LM239PW	
			TSSOP (PW)	Reel of 2000	LM239PWR	L239
		2011	0010 (D)	Tube of 50	LM239AD	
	2 mV	30 V	SOIC (D)	Reel of 2500	LM239ADR	LM239A
			PDIP (N)	Tube of 25	LM2901N	LM2901N
			0.010 (0)	Tube of 50	LM2901D	
			SOIC (D)	Reel of 2500	LM2901DR	LM2901
	7 mV	30 V	SOP (NS)	Reel of 2000	LM2901NSR	LM2901
4000 to 40500				Tube of 90	LM2901PW	
-40°C to 125°C			TSSOP (PW)	Reel of 2000	LM2901PWR	L2901
	7	2011	SOIC (D)	Reel of 2500	LM2901VQDR	L2901V
	7 mV	32 V	TSSOP (PW)	Reel of 2000	LM2901VQPWR	L2901V
	2 m)/	22.1/	SOIC (D)	Reel of 2500	LM2901AVQDR	L2901AV
	2 mV	32 V	TSSOP (PW)	Reel of 2000	LM2901AVQPWR	L2901AV

[†] Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.





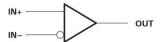
description/ordering information (continued)

ORDERING INFORMATION

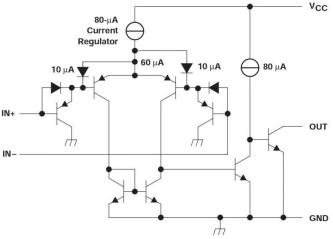
TA	V _{IO} max AT 25°C	MAX V _{CC}	PACKA	GE [†]	ORDERABLE PART NUMBER	TOP-SIDE MARKING
			CFP (W)	Tube of 25	LM139W	LM139W
			CDIP (J)	Tube of 25	LM139J	LM139J
	5 mV	30 V	LCCC (FK)	Tube of 55	LM139FK	LM139FK
			0010 (D)	Tube of 50	LM139D	
-55°C to 125°C			SOIC (D)	Reel of 2500	LM139DR	LM139D
-55°C to 125°C			CFP (W)	Tube of 25	LM139AW	LM139AW
			CDIP (J)	Tube of 25	LM139AJ	LM139AJ
	2 mV	30 V	LCCC (FK)	Tube of 55	LM139AFK	LM139AFK
			SOIC (D)	Tube of 50	LM139AD	LM139AD
			SOIC (D)	Reel of 2500	LM139ADR	LIVITS9AD

[†] Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

symbol (each comparator)



schematic (each comparator)



All current values shown are nominal.





LM139, LM139A, LM239, LM239A, LM339, LM339A, LM2901, LM2901V **QUAD DIFFERENTIAL COMPARATORS**

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage, V_{CC} (see Note 1)	± 0.3 V to	36 V 36 V
Output voltage, V _O		
Output current, I _O		
Duration of output short circuit to ground (see Note 3)	Unlin	nited
Package thermal impedance, θ _{JA} (see Notes 4 and 5):	D package 86°	C/W
	DB package 96°	C/W
	N package 80°	C/W
	NS package 76°	
	PW package 113°	C/W
Package thermal impedance, θ _{JC} (see Notes 6 and 7):	FK package 5.61°	C/W
	J package 15.05°	
	W package 14.65°	C/W
Operating virtual junction temperature, T _J		50°C
Case temperature for 60 seconds: FK package		
Lead temperature 1,6 mm (1/16 inch) from case for 60	seconds: J package 30	00°C
Storage temperature range, T _{stg}	–65°C to 15	50°C

[†] Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. All voltage values, except differential voltages, are with respect to network ground.

- 2. Differential voltages are at IN+ with respect to IN-.
- 3. Short circuits from outputs to V_{CC} can cause excessive heating and eventual destruction.
- 4. Maximum power dissipation is a function of T_J(max), θ_JA, and T_A. The maximum allowable power dissipation at any allowable ambient temperature is $P_D = (T_J(max) - T_A)/\theta_{JA}$. Operating at the absolute maximum T_J of 150°C can affect reliability. 5. The package thermal impedance is calculated in accordance with JESD 51-7.
- 6. Maximum power dissipation is a function of $T_J(max)$, θ_{JC} , and T_C . The maximum allowable power dissipation at any allowable case temperature is $P_D = (T_J(max) T_C)/\theta_{JC}$. Operating at the absolute maximum T_J of 150°C can affect reliability.
- 7. The package thermal impedance is calculated in accordance with MIL-STD-883.





electrical characteristics at specified free-air temperature, V_{CC} = 5 V (unless otherwise noted)

				_ +	L	M139		LN	1139A		LIBUT	
	PARAMETER	TEST CO	NDITIONS [†]	TA [‡]	MIN	TYP	MAX	MIN	TYP	MAX	UNIT	
		V _{CC} = 5 V to		25°C		2	5		1	2		
VIO	Input offset voltage	$V_{IC} = V_{ICR}(min),$ $V_{O} = 1.4 V$		Full range			9			4	mV	
			2000			3	25		3	25		
liO	Input offset current	V _O = 1.4 V		Full range			100			100	nA	
	to a filtre and a	V 14V	V _O = 1.4 V			-25	-100		-25	-100	4	
IIB	Input bias current	V _O = 1.4 V					-300			-300	nA	
	Common-mode			25°C	0 to V _{CC} -1.5			0 to V _{CC} -1.5			V	
VICR	input-voltage range			Full range	0 to V _{CC} -2			0 to V _{CC} -2			V	
A_{VD}	Large-signal differential-voltage amplification	$V_{CC} \pm = \pm 7.5$ $V_{O} = -5 \text{ V to}$		25°C		200		50	200		V/mV	
	High-level output	V 1V	V _{OH} = 5 V	25°C		0.1			0.1		nA	
ІОН	current	V _{ID} = 1 V	V _{OH} = 30 V	Full range			1			1	μΑ	
	Low-level output			25°C		150	400		150	400		
VOL	voltage	$V_{ID} = -1 V$,	IOL = 4 mA	Full range			700			700	mV	
loL	Low-level output current	V _{ID} = -1 V,	V _{OL} = 1.5 V	25°C	6	16		6	16		mA	
Icc	Supply current (four comparators)	V _O = 2.5 V,	No load	25°C		0.8	2		0.8	2	mA	

[†] All characteristics are measured with zero common-mode input voltage, unless otherwise specified.

switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER	TEST CO	L	UNIT			
		MIN	TYP	MAX		
Deenenee time	R _I connected to 5 V through 5.1 kΩ,	100-mV input step with 5-mV overdrive		1.3		
Response time	C _L = 15 pF [§] , See Note 8	TTL-level input step	0.3			μS

[§] C_L includes probe and jig capacitance.

NOTE 8: The response time specified is the interval between the input step function and the instant when the output crosses 1.4 V.



[‡] Full range (MIN to MAX) for LM139 and LM139A is -55°C to 125°C. All characteristics are measured with zero common-mode input voltage, unless otherwise specified.



electrical characteristics at specified free-air temperature, V_{CC} = 5 V (unless otherwise noted)

	PARAMETER	TEST CO	NDITIONS†	TA [‡]		M239 M339			1239A 1339A		UNIT
					MIN	TYP	MAX	MIN	TYP	MAX	
X2224	INTELLEGISTRA	00	$V_{CC} = 5 \text{ V to } 30 \text{ V},$ $V_{IC} = V_{ICR}(\text{min}),$ $V_{O} = 1.4 \text{ V}$			2	5		1	3	
VIO	Input offset voltage						9			4	mV
1	t -#t	V 14V		25°C		5	50		5	50	0
lio	Input offset current	V _O = 1.4 V		Full range			150			150	nA
		V 4.1V		25°C		-25	-250		-25	-250	0
IB	Input bias current	V _O = 1.4 V		Full range			-400			-400	nA
.,	Common-mode			25°C	0 to V _{CC} -1.5			0 to V _{CC} -1.5			
VICR	input-voltage range			Full range	0 to V _{CC} -2			0 to V _{CC} -2			V
AVD	Large-signal differential-voltage amplification	$V_{CC} = 15 \text{ V},$ $V_{O} = 1.4 \text{ V to}$ $R_{L} \ge 15 \text{ k}\Omega \text{ to}$		25°C	50	200		50	200		V/mV
	High-level output		V _{OH} = 5 V	25°C		0.1	50		0.1	50	nA
ІОН	current	V _{ID} = 1 V	V _{OH} = 30 V	Full range			1			1	μА
	Low-level output			25°C		150	400		150	400	
VOL	voltage	$V_{ID} = -1 V$,	$I_{OL} = 4 \text{ mA}$	Full range			700			700	mV
I _{OL}	Low-level output current	V _{ID} = −1 V,	V _{OL} = 1.5 V	25°C	6	16		6	16		mA
Icc	Supply current (four comparators)	V _O = 2.5 V,	No load	25°C		0.8	2		0.8	2	mA

[†] All characteristics are measured with zero common-mode input voltage, unless otherwise specified.

switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER	TEST CO	NDITIONS	LM23 LM33	UNIT		
			MIN	TYP	MAX	
Despense time	R _L connected to 5 V through 5.1 k Ω ,	100-mV input step with 5-mV overdrive		1.3		6
Response time	C _L = 15 pF [§] , See Note 8	TTL-level input step		0.3		μS

§ CL includes probe and jig capacitance.

NOTE 8: The response time specified is the interval between the input step function and the instant when the output crosses 1.4 V.



[‡] Full range (MIN to MAX) for LM239 and LM239A is -25°C to 85°C, for LM339 and LM339A is 0°C to 70°C. All characteristics are measured with zero common-mode input voltage, unless otherwise specified.



electrical characteristics at specified free-air temperature, V_{CC} = 5 V (unless otherwise noted)

	24244555		t	- +	LN	/12901		
	PARAMETER	TEST CO	NDITIONS†	TA [‡]	MIN	TYP	MAX	UNIT
			No. A desired	25°C		2	7	
,,		$V_{IC} = V_{ICR}(min),$	Non-A devices	Full range			15	
VIO	Input offset voltage	$V_O = 1.4 \text{ V},$ $V_{CC} = 5 \text{ V to MAX}$	A 60: I I I	25°C		1	2	mV
			A-suffix devices	Full range			4	
			25°C		5	50	- 0	
lio	Input offset current	V _O = 1.4 V		Full range			200	nA
Losson			25°C		-25	-250	^	
IB	Input bias current	V _O = 1.4 V	Full range			-500	nA	
. Common-mode input-voltage			25°C	0 to V _{CC} -1.5			v	
VICR	range		Full range	0 to V _{CC} -2			V	
A _{VD}	Large-signal differential-voltage amplification	$V_{CC} = 15 \text{ V},$ $V_{O} = 1.4 \text{ V to } 11.4 \text{ V}$ $R_{L} \ge 15 \text{ k}\Omega \text{ to } V_{CC}$		25°C	25	100		V/mV
	I Colored and a second	V 4 V	V _{OH} = 5 V	25°C		0.1	50	nA
ІОН	High-level output current	V _{ID} = 1 V	V _{OH} = V _{CC} MAX§	Full range			1	μА
		1015 (U-1010)	Non-V devices	2500		150	500	
VOL	Low-level output voltage	$V_{ID} = -1 V$, $I_{OL} = 4 \text{ mA}$	V-suffix devices	25°C		150	400	mV
		All devices		Full range			700	
loL	Low-level output current	$V_{ID} = -1 \text{ V}, \qquad V_{OL} = 1.5 \text{ V}$		25°C	6	16		mA
loo	Supply current $V_O = 2.5 \text{ V}$, $V_{CC} = 5 \text{ V}$ (four comparators) $V_O = 2.5 \text{ V}$, No load $V_{CC} = MAX^S$		V _{CC} = 5 V	25°C		0.8	2	mA
Icc			V _{CC} = MAX§	25.0		1	2.5	IIIA

[†] All characteristics are measured with zero common-mode input voltage, unless otherwise specified.

switching characteristics, V_{CC} = 5 V, T_A = 25°C

DADAMETER		L	UNIT				
PARAMETER	TEST CC	TEST CONDITIONS					
	R _I connected to 5 V through 5.1 kΩ,	100-mV input step with 5-mV overdrive		1.3			
Response time	C _L = 15 pF [¶] , See Note 8	TTL-level input step		0.3		μS	

¶ C_L includes probe and jig capacitance.

NOTE 8: The response time specified is the interval between the input step function and the instant when the output crosses 1.4 V.



Full range (MIN to MAX) for LM2901 is -40°C to 125°C. All characteristics are measured with zero common-mode input voltage, unless otherwise specified.

 $[\]S$ V_{CC} MAX = 30 V for non-V devices, and 32 V for V-suffix devices.





PACKAGE OPTION ADDENDUM

18-Feb-2005

PACKAGING INFORMATION

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish	n MSL Peak Temp (3)
5962-7700801VCA	ACTIVE	CDIP	J	14	1	None	A42 SNPB	Level-NC-NC-NC
5962-87739012A	ACTIVE	LCCC	FK	20	1	None	POST-PLATE	Level-NC-NC-NC
5962-8773901CA	ACTIVE	CDIP	J	14	1	None	A42 SNPB	Level-NC-NC-NC
5962-8773901DA	ACTIVE	CFP	W	14	1	None	A42 SNPB	Level-NC-NC-NC
77008012A	ACTIVE	LCCC	FK	20	1	None	POST-PLATE	Level-NC-NC-NC
7700801CA	ACTIVE	CDIP	J	14	1	None	A42 SNPB	Level-NC-NC-NC
7700801DA	ACTIVE	CFP	W	14	1	None	A42 SNPB	Level-NC-NC-NC
JM38510/11201BCA	ACTIVE	CDIP	J	14	1	None	A42 SNPB	Level-NC-NC-NC
LM139AD	ACTIVE	SOIC	D	14	50	None	CU NIPDAU	Level-3-245C-168 HR
LM139ADR	ACTIVE	SOIC	D	14	2500	Pb-Free (RoHS)	CU NIPDAU	Level-2-250C-1 YEAR/ Level-1-235C-UNLIM
LM139AFKB	ACTIVE	LCCC	FK	20	1	None	POST-PLATE	Level-NC-NC-NC
LM139AJ	ACTIVE	CDIP	J	14	1	None	A42 SNPB	Level-NC-NC-NC
LM139AJB	ACTIVE	CDIP	J	14	1	None	A42 SNPB	Level-NC-NC-NC
LM139AN	OBSOLETE	PDIP	N	14		None	Call TI	Call TI
LM139AW	ACTIVE	CFP	W	14	1	None	A42 SNPB	Level-NC-NC-NC
LM139AWB	ACTIVE	CFP	W	14	1	None	A42 SNPB	Level-NC-NC-NC
LM139D	ACTIVE	SOIC	D	14	50	None	CU NIPDAU	Level-1-220C-UNLIM
LM139DR	ACTIVE	SOIC	D	14	2500	None	CU NIPDAU	Level-1-220C-UNLIM
LM139FKB	ACTIVE	LCCC	FK	20	1	None	POST-PLATE	Level-NC-NC-NC
LM139J	ACTIVE	CDIP	J	14	1	None	A42 SNPB	Level-NC-NC-NC
LM139JB	ACTIVE	CDIP	J	14	1	None	A42 SNPB	Level-NC-NC-NC
LM139N	OBSOLETE	PDIP	N	14		None	Call TI	Call TI
LM139W	ACTIVE	CFP	W	14	1	None	A42 SNPB	Level-NC-NC-NC
LM139WB	ACTIVE	CFP	W	14	1	None	A42 SNPB	Level-NC-NC-NC
LM239AD	ACTIVE	SOIC	D	14	50	Pb-Free (RoHS)	CU NIPDAU	Level-2-260C-1 YEAR/ Level-1-235C-UNLIM
LM239ADR	ACTIVE	SOIC	D	14	2500	Pb-Free (RoHS)	CU NIPDAU	Level-2-260C-1 YEAR/ Level-1-235C-UNLIM
LM239AN	OBSOLETE	PDIP	N	14		None	Call TI	Call TI
LM239D	ACTIVE	SOIC	D	14	50	Pb-Free (RoHS)	CU NIPDAU	Level-2-260C-1 YEAR/ Level-1-235C-UNLIM
LM239DR	ACTIVE	SOIC	D	14	2500	Pb-Free (RoHS)	CU NIPDAU	Level-2-260C-1 YEAR/ Level-1-235C-UNLIM
LM239N	ACTIVE	PDIP	N	14	25	Pb-Free (RoHS)	CU NIPDAU	Level-NC-NC-NC
LM239PW	ACTIVE	TSSOP	PW	14	90	Pb-Free (RoHS)	CU NIPDAU	Level-1-250C-UNLIM
LM239PWR	ACTIVE	TSSOP	PW	14	2000	Pb-Free (RoHS)	CU NIPDAU	Level-1-250C-UNLIM
LM2901AVQDR	ACTIVE	SOIC	D	14	2500	Pb-Free (RoHS)	CU NIPDAU	Level-2-250C-1 YEAR/ Level-1-235C-UNLIM
LM2901AVQPWR	ACTIVE	TSSOP	PW	14	2000	None	CU NIPDAU	Level-1-250C-UNLIM
LM2901D	ACTIVE	SOIC	D	14	50	Pb-Free (RoHS)	CU NIPDAU	Level-2-260C-1 YEAR/ Level-1-235C-UNLIM

Addendum-Page 1





PACKAGE OPTION ADDENDUM

18-Feb-2005

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish	MSL Peak Temp (3)
LM2901DR	ACTIVE	SOIC	D	14	2500	Pb-Free (RoHS)		Level-2-260C-1 YEAR Level-1-235C-UNLIM
LM2901N	ACTIVE	PDIP	N	14	25	Pb-Free (RoHS)	CU NIPDAU	Level-NC-NC-NC
LM2901NSR	ACTIVE	SO	NS	14	2000	Pb-Free (RoHS)		Level-2-260C-1 YEAR Level-1-235C-UNLIM
LM2901PW	ACTIVE	TSSOP	PW	14	90	Pb-Free (RoHS)	CU NIPDAU	Level-1-250C-UNLIM
LM2901PWLE	OBSOLETE	TSSOP	PW	14		None	Call TI	Call TI
LM2901PWR	ACTIVE	TSSOP	PW	14	2000	Pb-Free (RoHS)	CU NIPDAU	Level-1-250C-UNLIM
LM2901QD	OBSOLETE	SOIC	D	14		None	Call TI	Call TI
LM2901QN	OBSOLETE	PDIP	N	14		None	Call TI	Call TI
LM2901VQDR	ACTIVE	SOIC	D	14	2500	Pb-Free (RoHS)		Level-2-250C-1 YEAR Level-1-235C-UNLIM
LM2901VQPWR	ACTIVE	TSSOP	PW	14	2000	None	CU NIPDAU	Level-1-250C-UNLIM
LM339AD	ACTIVE	SOIC	D	14	50	Pb-Free (RoHS)		Level-2-260C-1 YEAR Level-1-235C-UNLIM
LM339ADBR	ACTIVE	SSOP	DB	14	2000	Pb-Free (RoHS)		Level-2-260C-1 YEAR Level-1-235C-UNLIM
LM339ADR	ACTIVE	SOIC	D	14	2500	Pb-Free (RoHS)		Level-2-260C-1 YEAR Level-1-235C-UNLIM
LM339AN	ACTIVE	PDIP	N	14	25	Pb-Free (RoHS)	CU NIPDAU	Level-NC-NC-NC
LM339ANSR	ACTIVE	SO	NS	14	2000	Pb-Free (RoHS)		Level-2-260C-1 YEAR Level-1-235C-UNLIM
LM339APW	ACTIVE	TSSOP	PW	14	90	Pb-Free (RoHS)	CU NIPDAU	Level-1-250C-UNLIM
LM339APWR	ACTIVE	TSSOP	PW	14	2000	Pb-Free (RoHS)	CU NIPDAU	Level-1-250C-UNLIM
LM339D	ACTIVE	SOIC	D	14	50	Pb-Free (RoHS)		Level-2-260C-1 YEAR Level-1-235C-UNLIM
LM339DBLE	OBSOLETE	SSOP	DB	14		None	Call TI	Call TI
LM339DBR	ACTIVE	SSOP	DB	14	2000	Pb-Free (RoHS)		Level-2-260C-1 YEAR Level-1-235C-UNLIM
LM339DR	ACTIVE	SOIC	D	14	2500	Pb-Free (RoHS)		Level-2-260C-1 YEAR Level-1-235C-UNLIM
LM339N	ACTIVE	PDIP	N	14	25	Pb-Free (RoHS)	CU NIPDAU	Level-NC-NC-NC
LM339NSLE	OBSOLETE	so	NS	14		None	Call TI	Call TI
LM339NSR	ACTIVE	SO	NS	14	2000	Pb-Free (RoHS)		Level-2-260C-1 YEAR Level-1-235C-UNLIM
LM339PW	ACTIVE	TSSOP	PW	14	90	Pb-Free (RoHS)	CU NIPDAU	Level-1-250C-UNLIM
LM339PWLE	OBSOLETE	TSSOP	PW	14		None	Call TI	Call TI
LM339PWR	ACTIVE	TSSOP	PW	14	2000	Pb-Free (RoHS)	CU NIPDAU	Level-1-250C-UNLIM
LM339Y	OBSOLETE			0		None	Call TI	Call TI

 $[\]ensuremath{^{(1)}}$ The marketing status values are defined as follows:

Addendum-Page 2





PACKAGE OPTION ADDENDUM

18-Feb-2005

ACTIVE: Product device recommended for new designs.

LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.

NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.

PREVIEW: Device has been announced but is not in production. Samples may or may not be available.

OBSOLETE: TI has discontinued the production of the device.

(2) Eco Plan - May not be currently available - please check http://www.ti.com/productcontent for the latest availability information and additional product content details.

None: Not yet available Lead (Pb-Free).

Pb-Free (RoHS): Tl's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.

Green (RoHS & no Sb/Br): TI defines "Green" to mean "Pb-Free" and in addition, uses package materials that do not contain halogens, including bromine (Br) or antimony (Sb) above 0.1% of total product weight.

(3) MSL, Peak Temp. -- The Moisture Sensitivity Level rating according to the JEDECindustry standard classifications, and peak solder temperature.

Important Information and Disclaimer: The information provided on this page represents TI's knowledge and belief as of the date that it is provided. TI bases its knowledge and belief on information provided by third parties, and makes no representation or warranty as to the accuracy of such information. Efforts are underway to better integrate information from third parties. TI has taken and continues to take reasonable steps to provide representative and accurate information but may not have conducted destructive testing or chemical analysis on incoming materials and chemicals. TI and TI suppliers consider certain information to be proprietary, and thus CAS numbers and other limited information may not be available for release.

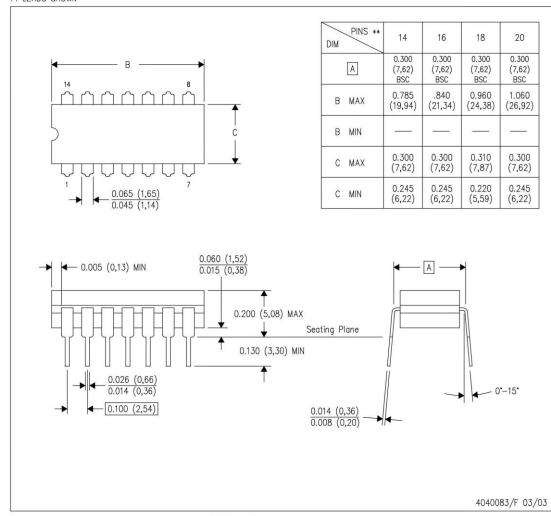
In no event shall TI's liability arising out of such information exceed the total purchase price of the TI part(s) at issue in this document sold by TI to Customer on an annual basis.

Addendum-Page 3





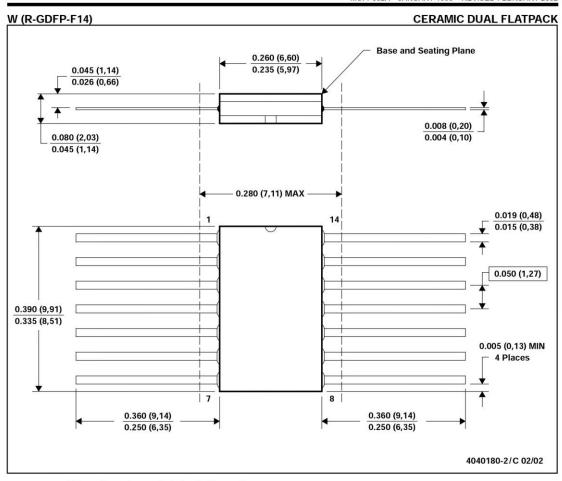
CERAMIC DUAL IN-LINE PACKAGE



- A. All linear dimensions are in inches (millimeters).
- B. This drawing is subject to change without notice.
- C. This package is hermetically sealed with a ceramic lid using glass frit.
- D. Index point is provided on cap for terminal identification only on press ceramic glass frit seal only.
- E. Falls within MIL STD 1835 GDIP1-T14, GDIP1-T16, GDIP1-T18 and GDIP1-T20.



MCFP002A - JANUARY 1995 - REVISED FEBRUARY 2002



NOTES: A. All linear dimensions are in inches (millimeters).

- B. This drawing is subject to change without notice.
- C. This package can be hermetically sealed with a ceramic lid using glass frit.
- D. Index point is provided on cap for terminal identification only.

 E. Falls within MIL STD 1835 GDFP1-F14 and JEDEC MO-092AB



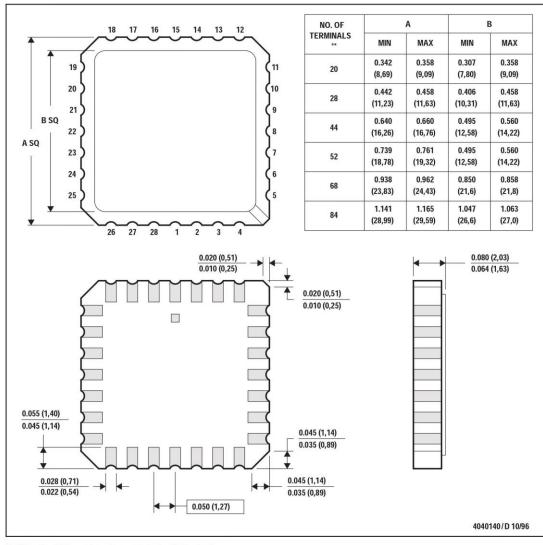


MLCC006B - OCTOBER 1996

FK (S-CQCC-N**)

28 TERMINAL SHOWN

LEADLESS CERAMIC CHIP CARRIER



NOTES: A. All linear dimensions are in inches (millimeters).

- B. This drawing is subject to change without notice.
- C. This package can be hermetically sealed with a metal lid.
- D. The terminals are gold plated.
- E. Falls within JEDEC MS-004

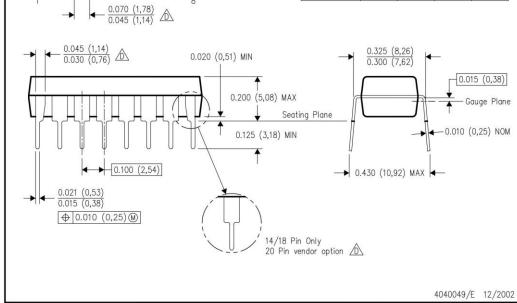




N (R-PDIP-T**) 16 PINS SHOWN

PLASTIC DUAL-IN-LINE PACKAGE

PINS ** 14 16 18 20 DIM 9 0.775 0.775 0.920 1.060 A MAX (19,69)(19,69)(23,37)(26,92) 0.260 (6,60) 0.240 (6,10) 0.745 0.745 0.850 0.940 A MIN (18,92)(18,92)(21,59)(23,88)MS-001 BB AC AD AA VARIATION 8



- A. All linear dimensions are in inches (millimeters).

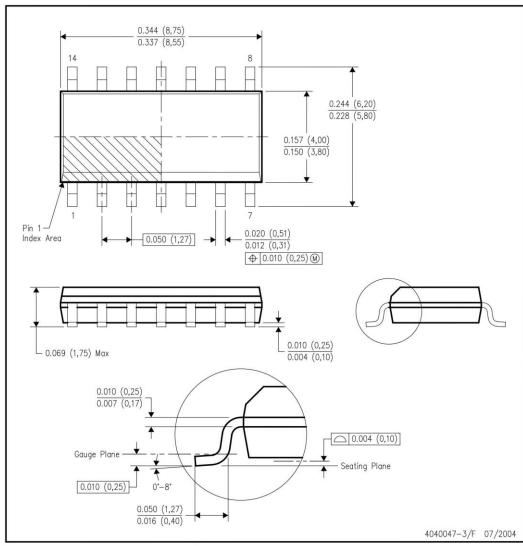
 B. This drawing is subject to change without notice.
- Falls within JEDEC MS-001, except 18 and 20 pin minimum body length (Dim A).
- The 20 pin end lead shoulder width is a vendor option, either half or full width.





D (R-PDSO-G14)

PLASTIC SMALL-OUTLINE PACKAGE



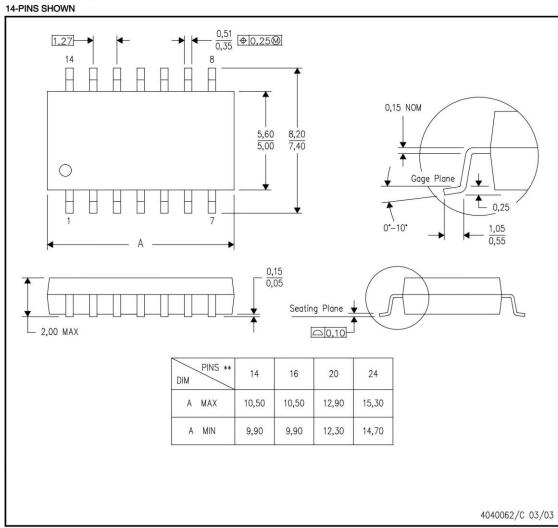
- A. All linear dimensions are in inches (millimeters).
 B. This drawing is subject to change without notice.
 C. Body dimensions do not include mold flash or protrusion not to exceed 0.006 (0,15).
- D. Falls within JEDEC MS-012 variation AB.





NS (R-PDSO-G**)

PLASTIC SMALL-OUTLINE PACKAGE



- A. All linear dimensions are in millimeters.
 B. This drawing is subject to change without notice.
 C. Body dimensions do not include mold flash or protrusion, not to exceed 0,15.



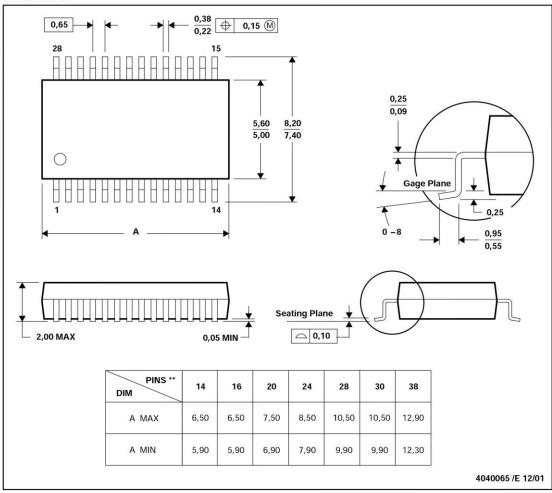


MSSO002E - JANUARY 1995 - REVISED DECEMBER 2001

DB (R-PDSO-G**)

PLASTIC SMALL-OUTLINE

28 PINS SHOWN



NOTES: A. All linear dimensions are in millimeters.

B. This drawing is subject to change without notice.

C. Body dimensions do not include mold flash or protrusion not to exceed 0,15.

D. Falls within JEDEC MO-150



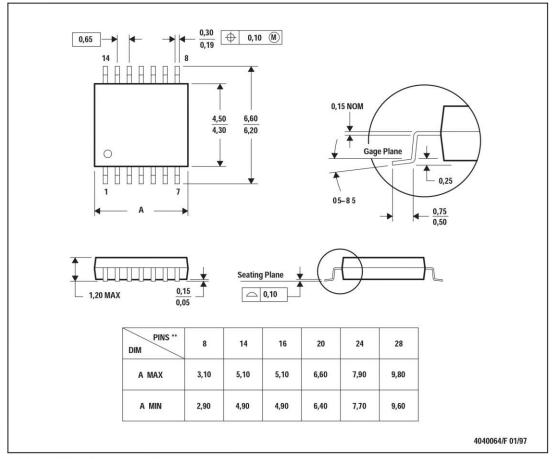


MTSS001C - JANUARY 1995 - REVISED FEBRUARY 1999

PW (R-PDSO-G**)

14 PINS SHOWN

PLASTIC SMALL-OUTLINE PACKAGE



NOTES: A. All linear dimensions are in millimeters.

- B. This drawing is subject to change without notice.
 C. Body dimensions do not include mold flash or protrusion not to exceed 0,15.
 D. Falls within JEDEC MO-153





IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		Applications	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments

Post Office Box 655303 Dallas, Texas 75265

Copyright © 2005, Texas Instruments Incorporated



This datasheet has been download from:

www.datasheetcatalog.com

Datasheets for electronics components.



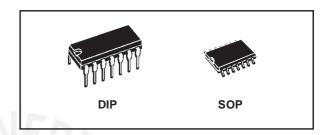


QUAD 2 INPUT AND GATE

- MEDIUM SPEED OPERATION : t_{PD} = 60ns (Typ.) at 10V
- QUIESCENT CURRENT SPECIFIED UP TO 20V
- 5V, 10V AND 15V PARAMETRIC RATINGS
- INPUT LEAKAGE CURRENT I_I = 100nA (MAX) AT V_{DD} = 18V T_A = 25°C
- 100% TESTED FOR QUIESCENT CURRENT
- MEETS ALL REQUIREMENTS OF JEDEC JESD13B "STANDARD SPECIFICATIONS FOR DESCRIPTION OF B SERIES CMOS DEVICES"

DESCRIPTION

The HCF4081B is a monolithic integrated circuit fabricated in Metal Oxide Semiconductor technology available in DIP and SOP packages. The HCF4081B QUAD 2 INPUT AND GATE provide the system designer with direct

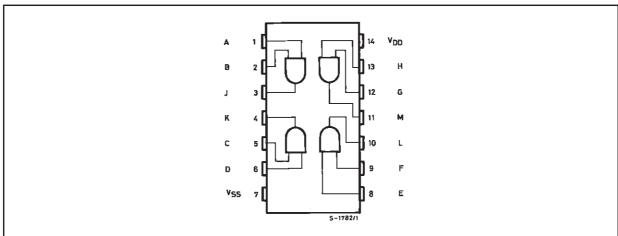


ORDER CODES

PACKAGE	TUBE	T & R
DIP	HCF4081BEY	
SOP	HCF4081BM1	HCF4081M013TR

implementation of the AND function and supplement the existing family of CMOS gates.

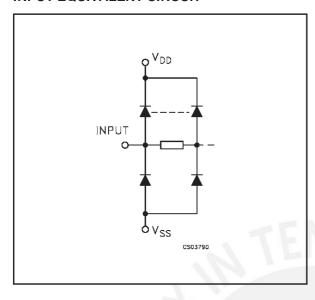
PIN CONNECTION



September 2001 1/7



INPUT EQUIVALENT CIRCUIT



PIN DESCRIPTION

PIN No SYMBOL		NAME AND FUNCTION
1, 5, 8, 12	A, C, E, G	Data Inputs
2, 6, 9, 13	B, D, F, H	Data Inputs
3, 4, 10, 11	J, K, L, M	Data Outputs
7	V_{SS}	Negative Supply Voltage
14	V_{DD}	Positive Supply Voltage

TRUTH TABLE

INP	UTS	OUTPUTS
A, C, E, G	B, D, F,H	J, K, L, M
P L	L	L
h D	Н	L
H	/ L	L
Н	Н	Н

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V _{DD}	Supply Voltage	-0.5 to +22	V
VI	DC Input Voltage	-0.5 to V _{DD} + 0.5	V
II	DC Input Current	± 10	mA
P _D	Power Dissipation per Package	200	mW
	Power Dissipation per Output Transistor	100	mW
T _{op}	Operating Temperature	-55 to +125	°C
T _{stg}	Storage Temperature	-65 to +150	°C

Absolute Maximum Ratings are those values beyond which damage to the device may occur. Functional operation under these conditions is not implied.

All voltage values are referred to V_{SS} pin voltage.

RECOMMENDED OPERATING CONDITIONS

Symbol	Parameter	Value	Unit
V_{DD}	Supply Voltage	3 to 20	V
VI	Input Voltage	0 to V _{DD}	V
T _{op}	Operating Temperature	-55 to 125	°C

DC SPECIFICATIONS

			Test Con	dition					Value				
Symbol	Parameter	Vı	v _o	I _O	V _{DD}	Т	T _A = 25°C		-40 to 85°C		-55 to 125°C		Unit
		(V)	(V)	(μ A)	(V)	Min.	Тур.	Max.	Min.	Max.	Min.	Max.	
ΙL	Quiescent Current	0/5			5		0.01	0.25		7.5		7.5	
		0/10			10		0.01	0.5		15		15	μΑ
		0/15			15		0.01	1		30		30	μΛ
		0/20			20		0.02	5		150		150	
V _{OH}	High Level Output	0/5		<1	5	4.95			4.95		4.95		
Voltage	0/10		<1	10	9.95			9.95		9.95		V	
		0/15		<1	15	14.95			14.95		14.95		
V _{OL}	Low Level Output	5/0		<1	5		0.05			0.05		0.05	
Voltage	10/0		<1	10		0.05			0.05		0.05	V	
		15/0	1	<1	15	1	0.05			0.05		0.05	
V _{IH}	High Level Input	6.	0.5/4.5	<1	5	3.5	JA	1	3.5		3.5		V
	Voltage	. 1	1/9	<1	10	7	1	/ (7		7		
			1.5/13.5	<1	15	11			11		11		
V _{IL}	Low Level Input	M	4.5/0.5	<1	5			1.5		1.5		1.5	
	Voltage	3 1	9/1	<1	10		- 1	3	N. Committee	3		3	V
		7/	13.5/1.5	<1	15			4		4		4	
I _{OH}	Output Drive	0/5	2.5	<1	5	-1.36	-3.2		-1.15		-1.1		
	Current	0/5	4.6	<1	5	-0.44	-1		-0.36		-0.36		mA
		0/10	9.5	<1	10	-1.1	-2.6		-0.9		-0.9		IIIA
	1 1 1 1	0/15	13.5	<1	15	-3.0	-6.8		-2.4		-2.4		
I _{OL}	Output Sink	0/5	0.4	<1	5	0.44	11	_/	0.36		0.36		
Current	Current	0/10	0.5	<1	10	1.1	2.6	10	0.9		0.9		mΑ
		0/15	1.5	<1	15	3.0	6.8		2.4		2.4		
I _I	Input Leakage Current	0/18	Any In	put	18		±10 ⁻⁵	±0.1		±1		±1	μΑ
Cl	Input Capacitance		Any In	put	1		5	7.5					рF

The Noise Margin for both "1" and "0" level is: 1V min. with V_{DD} =5V, 2V min. with V_{DD} =10V, 2.5V min. with V_{DD} =15V

$\textbf{DYNAMIC ELECTRICAL CHARACTERISTICS} \; (T_{amb} = 25^{\circ}C, \;\; C_{L} = 50 pF, \; R_{L} = 200 \text{K}\Omega, \;\; t_{f} = t_{f} = 20 \; \text{ns})$

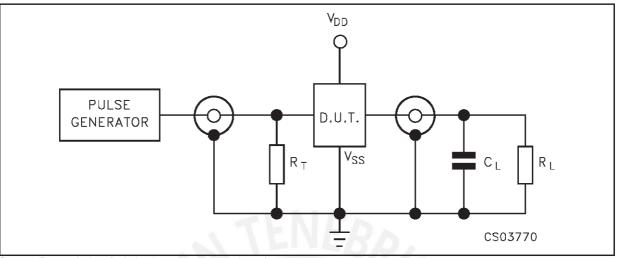
Symbol	Doromotor	Test Condition			Value (*)		
	Parameter	V _{DD} (V)		Min.	Тур.	Max.	
t _{PLH} t _{PHL}	Propagation Delay Time	5			125	250	
		10			60	125	ns
		15			45	90	
t _{TLH} t _{THL}	Output Transition Time	5			100	200	
		10			60	100	ns
		15			40	80	

^(*) Typical temperature coefficient for all V_{DD} value is 0.3 %/°C.



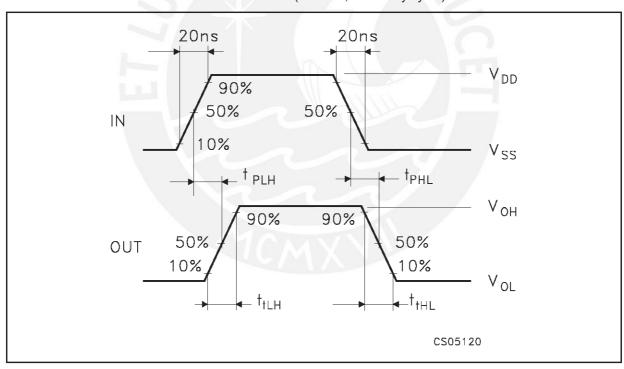


TEST CIRCUIT



 C_L = 50pF or equivalent (includes jig and probe capacitance) R_L = 200KΩ R_T = Z_{OUT} of pulse generator (typically 50Ω)

WAVEFORM: PROPAGATION DELAY TIMES (f=1MHz; 50% duty cycle)

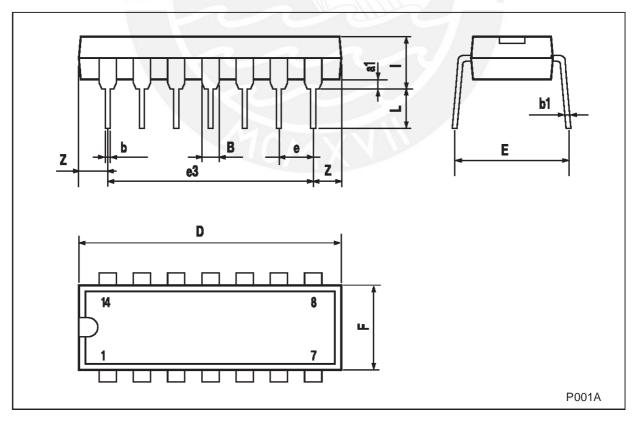


57



Plastic DIP-14 MECHANICAL DATA

DIM.		mm.			inch			
DINI.	MIN.	TYP	MAX.	MIN.	TYP.	MAX.		
a1	0.51			0.020				
В	1.39		1.65	0.055		0.065		
b		0.5			0.020			
b1		0.25			0.010			
D			20			0.787		
E		8.5	ENE	RA.	0.335			
е		2.54		1//0	0.100			
e3		15.24			0.600			
F			7.1			0.280		
I	V		5.1		5	0.201		
L		3.3			0.130			
Z	1.27		2.54	0.050		0.100		

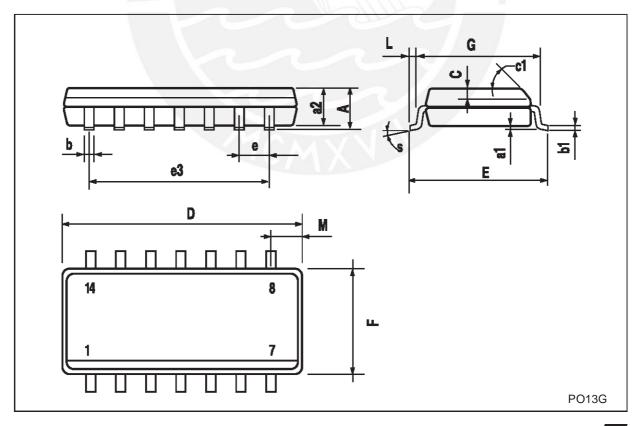


577



SO-14 MECHANICAL DATA

DIM.		mm.			inch				
DIIVI.	MIN.	TYP	MAX.	MIN.	TYP.	MAX.			
А			1.75			0.068			
a1	0.1		0.2	0.003		0.007			
a2			1.65			0.064			
b	0.35		0.46	0.013		0.018			
b1	0.19		0.25	0.007		0.010			
С		0.5			0.019				
c1			45°	(typ.)					
D	8.55	1 7	8.75	0.336		0.344			
Е	5.8	1 101 1	6.2	0.228		0.244			
е	1	1.27		4.0	0.050				
еЗ		7.62			0.300				
F	3.8		4.0	0.149		0.157			
G	4.6		5.3	0.181		0.208			
L	0.5		1.27	0.019	()	0.050			
М			0.68		1 m m	0.026			
S			8° (1	max.)					





HCF4081B



Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of STMicroelectronics.

© The ST logo is a registered trademark of STMicroelectronics

© 2001 STMicroelectronics - Printed in Italy - All Rights Reserved STMicroelectronics GROUP OF COMPANIES

Australia - Brazil - China - Finland - France - Germany - Hong Kong - India - Italy - Japan - Malaysia - Malta - Morocco Singapore - Spain - Sweden - Switzerland - United Kingdom © http://www.st.com





This datasheet has been download from:

 $\underline{www.datasheet catalog.com}$

Datasheets for electronics components.





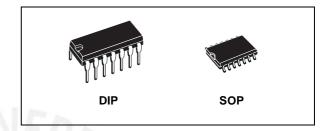


QUAD 2 INPUT NAND SCHMITT TRIGGER

- SCHMITT TRIGGER ACTION ON EACH INPUT WITH NO EXTERNAL COMPONENTS
- HYSTERESIS VOLTAGE TYPICALLY 0.9V at $V_{DD} = 5V$ AND 2.3V at $V_{DD} = 10V$
- NOISE IMMUNITY GREATER THAN 50%OF V_{DD} (Typ.)
- NO LIMIT ON INPUT RISE AND FALL TIMES
- QUIESCENT CURRENT SPECIFIED UP TO
- STANDARDIZED SYMMETRICAL OUTPUT CHARACTERISTICS
- 5V, 10V AND 15V PARAMETRIC RATINGS
- INPUT LEAKAGE CURRENT I_I = 100nA (MAX) AT V_{DD} = 18V T_A = 25°C
- 100% TESTED FOR QUIESCENT CURRENT
- MEETS ALL REQUIREMENTS OF JEDEC JESD13B " STANDARD SPECIFICATIONS FOR DESCRIPTION OF B SERIES CMOS DEVICES"

DESCRIPTION

The HCF4093B is a monolithic integrated circuit fabricated in Metal Oxide Semiconductor technology available in DIP and SOP packages.

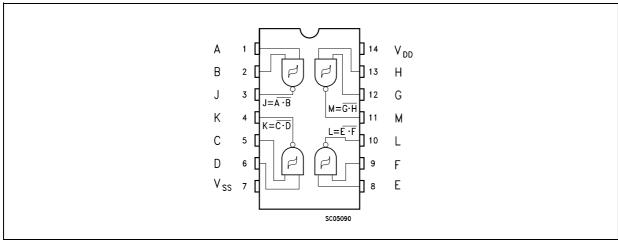


ORDER CODES

PACKAGE	TUBE	T&R
DIP	HCF4093BEY	
SOP	HCF4093BM1	HCF4093M013TR

The HCF4093B type consists of four schmitt trigger circuits. Each circuit functions as a two input NAND gate with schmitt trigger action on both inputs. The gate switches at different points for positive and negative going signals. The difference between the positive voltage (V_P) and the negative voltage (V_N) is defined as hysteresis voltage (V_H) .

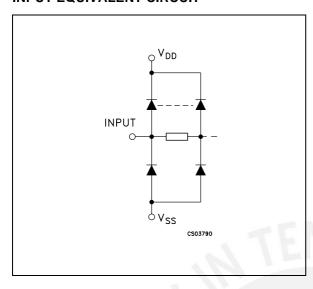
PIN CONNECTION



September 2001 1/7



INPUT EQUIVALENT CIRCUIT



PIN DESCRIPTION

PIN No	SYMBOL	NAME AND FUNCTION
1, 2, 5, 6, 8, 9, 12, 13	A, B, C, D, E, F, G, H	Data Inputs
3, 4, 10, 11	J, K, L, M	Data Outputs
7	V _{SS}	Negative Supply Voltage
14	V_{DD}	Positive Supply Voltage

TRUTH TABLE

INP	OUTPUTS	
A, C, E, G	B, D, F, H	J, K, L, M
L	L	Н
	Н	Н
C/SH/O	L	Н
H	Н	L

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V _{DD}	Supply Voltage	-0.5 to +22	V
V _I	DC Input Voltage	-0.5 to V _{DD} + 0.5	V
I _I	DC Input Current	± 10	mA
P _D	Power Dissipation per Package	200	mW
	Power Dissipation per Output Transistor	100	mW
T _{op}	Operating Temperature	-55 to +125	°C
T _{stg}	Storage Temperature	-65 to +150	°C

Absolute Maximum Ratings are those values beyond which damage to the device may occur. Functional operation under these conditions is not implied.

All voltage values are referred to V_{SS} pin voltage.

RECOMMENDED OPERATING CONDITIONS

Symbol	Parameter	Value	Unit
V _{DD}	Supply Voltage	3 to 20	V
V _I	Input Voltage	0 to V _{DD}	V
T _{op}	Operating Temperature	-55 to 125	°C



DC SPECIFICATIONS

		Test Condition			Value								
Symbol	Parameter	Vı	٧o	اماا	I _O V _{DD} (V)	Т	A = 25°	С	-40 to	85°C	-55 to	125°C	Unit
		(V)	(V)			Min.	Тур.	Max.	Min.	Max.	Min.	Max.	
ΙL	Quiescent Current	0/5			5		0.02	1		30		30	
		0/10			10		0.02	2		60		60	
		0/15			15		0.02	4		120		120	μΑ
		0/20			20		0.04	20		600		600	
V _{OH}	High Level Output	0/5		<1	5	4.95			4.95		4.95		
	Voltage	0/10		<1	10	9.95			9.95		9.95		V
		0/15		<1	15	14.95			14.95		14.95		
V _{OL}	Low Level Output	5/0		<1	5		0.05			0.05		0.05	
	Voltage	10/0		<1	10		0.05			0.05		0.05	V
		15/0		<1	15		0.05			0.05		0.05	
V _P	Positive Trigger	а	To T	N B	5	2.2	2.9	3.6	2.2	3.6	2.2	3.6	
	Threshold Voltage	а	110		10	4.6	5.9	7.1	4.6	7.1	4.6	7.1	
		а	7		15	6.8	8.8	10.8	6.8	10.8	6.8	10.8	V
		b	/ 1		5	2.6	3.3	4.0	2.6	4	2.6	4	
	b			10	5.6	7	8.2	5.6	8.2	5.6	8.2		
	b	- 7A		15	6.3	9.4	12.7	6.3	12.7	6.3	12.7		
V _N	Negative Trigger	а			5	0.9	1.9	2.8	0.9	2.8	0.9	2.8	V
	Threshold Voltage	а			10	2.5	3.9	5.2	2.5	5.2	2.5	5.2	
		а			15	4	5.8	7.4	4	7.4	4	7.4	
	1 1 1	b			5	1.4	2.3	3.2	1.4	3.2	1.4	3.2	
		b			10	3.4	5.1	6.6	3.4	6.6	3.4	6.6	
		b	1		15	4.8	7.3	9.6	4.8	9.6	4.8	9.6	
V _H	Hysteresis Voltage	а			5	0.3	0.9	1.6	0.3	1.6	0.3	1.6	
••		а	<i>a</i>		10	1.2	2.3	3.4	1.2	3.4	1.2	3.4	
		а			15	1.6	3.5	5	1.6	5	1.6	5	
		b			5	0.3	0.9	1.6	0.3	1.6	0.3	1.6	V
		b	10		10	1.2	2.3	3.4	1.2	3.4	1.2	3.4	
		b	76		15	1.6	3.5	5	1.6	5	1.6	5	
I _{OH}	Output Drive	0/5	2.5	<1	5	-1.36	-3.2		-1.15		-1.1		
	Current	0/5	4.6	<1	5	-0.44	-1		-0.36		-0.36		^
		0/10	9.5	<1	10	-1.1	-2.6		-0.9		-0.9		mA
		0/15	13.5	<1	15	-3.0	-6.8		-2.4		-2.4		
I _{OL}	Output Sink	0/5	0.4	<1	5	0.44	1		0.36		0.36		
-	Current	0/10	0.5	<1	10	1.1	2.6		0.9		0.9		mΑ
		0/15	1.5	<1	15	3.0	6.8		2.4		2.4		
I _I	Input Leakage Current	0/18	Any Ir	put	18		±10 ⁻⁵	±0.1		±1		±1	μΑ
C _I	Input Capacitance		Any In	put			5	7.5					pF

The Noise Margin for both "1" and "0" level is: 1V min. with V_{DD} =5V, 2V min. with V_{DD} =10V, 2.5V min. with V_{DD} =15V a: Input on terminals 1, 5, 8, 12 or 2, 6, 9, 13; other inputs to V_{DD} . b: Input on terminals 1 and 2, 5 and 6, 8 and 9, or 12 and 13; other inputs to V_{DD} .



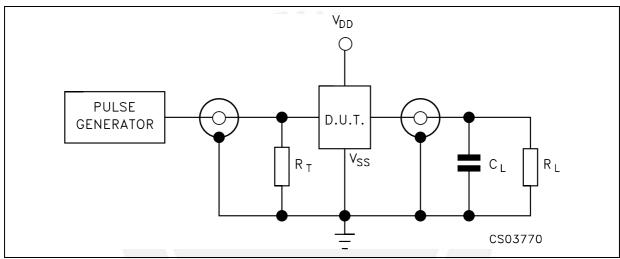


$\textbf{DYNAMIC ELECTRICAL CHARACTERISTICS} \; (T_{amb} = 25^{\circ}\text{C}, \;\; C_{L} = 50 \text{pF}, \; R_{L} = 200 \text{K}\Omega, \;\; t_{f} = t_{f} = 20 \; \text{ns})$

0	B		Test Condition		Value (*)		
Symbol	Symbol Parameter			Min.	Тур.	Max.	
t _{PLH} t _{PHL}	Propagation Delay Time	5			190	380	
		10			90	180	ns
		15			65	130	
t _{TLH} t _{THL}	Output Transition Time	5			100	200	
		10			50	100	ns
		15			40	80	

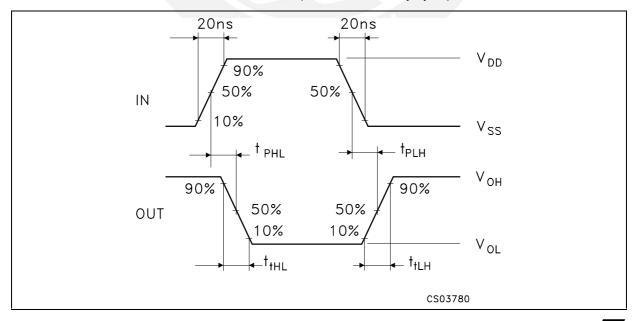
^(*) Typical temperature coefficient for all V_{DD} value is 0.3 %/°C.

TEST CIRCUIT



 C_L = 50pF or equivalent (includes jig and probe capacitance) R_L = 200KΩ R_T = Z_{OUT} of pulse generator (typically 50Ω)

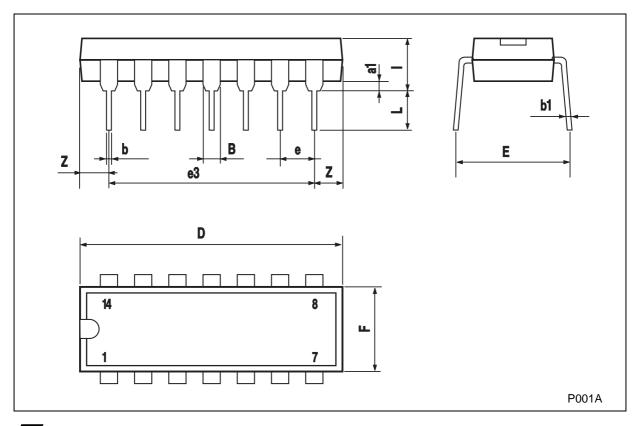
WAVEFORM: PROPAGATION DELAY TIMES (f=1MHz; 50% duty cycle)





Plastic DIP-14 MECHANICAL DATA

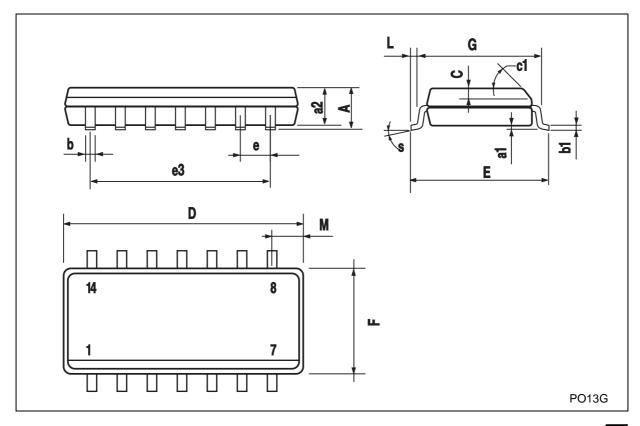
DIM		mm.			inch		
DIM.	MIN.	TYP	MAX.	MIN.	TYP.	MAX.	
a1	0.51			0.020			
В	1.39		1.65	0.055		0.065	
b		0.5			0.020		
b1		0.25			0.010		
D			20			0.787	
E		8.5			0.335		
е		2.54			0.100		
e3		15.24			0.600		
F			7.1			0.280	
I			5.1			0.201	
L		3.3			0.130		
Z	1.27		2.54	0.050		0.100	





SO-14 MECHANICAL DATA

DIM		mm.			inch	
DIM.	MIN.	TYP	MAX.	MIN.	TYP.	MAX.
А			1.75			0.068
a1	0.1		0.2	0.003		0.007
a2			1.65			0.064
b	0.35		0.46	0.013		0.018
b1	0.19		0.25	0.007		0.010
С		0.5			0.019	
c1			45°	(typ.)		•
D	8.55		8.75	0.336		0.344
Е	5.8		6.2	0.228		0.244
е		1.27			0.050	
e3		7.62			0.300	
F	3.8		4.0	0.149		0.157
G	4.6		5.3	0.181		0.208
L	0.5		1.27	0.019		0.050
М			0.68			0.026
S			8° (ı	max.)		





Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of STMicroelectronics.

© The ST logo is a registered trademark of STMicroelectronics

© 2001 STMicroelectronics - Printed in Italy - All Rights Reserved STMicroelectronics GROUP OF COMPANIES

Australia - Brazil - China - Finland - France - Germany - Hong Kong - India - Italy - Japan - Malaysia - Malta - Morocco Singapore - Spain - Sweden - Switzerland - United Kingdom © http://www.st.com





This datasheet has been download from:

 $\underline{www.datasheet catalog.com}$

Datasheets for electronics components.





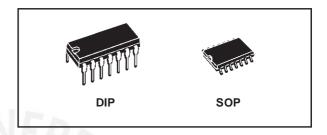


HEX SCHMITT TRIGGER

- SCHMITT TRIGGER ACTION WITH NO EXTERNAL COMPONENTS
- HYSTERESIS VOLTAGE (Typ.):
 0.9V at V_{DD} = 5V
 2.3V at V_{DD} = 10V
 3.5V at V_{DD} = 15V
- NOISE IMMUNITY GREATER THAN 50%
- NO LIMIT ON INPUT RISE AND FALL TIME
- LOW V_{DD} TO V_{SS} CURRENT DURING SLOW INPUT RAMP
- STANDARDIZED SYMMETRICAL OUTPUT CHARACTERISTICS
- QUIESCENT CURRENT SPECIFIED UP TO 20V
- 5V, 10V AND 15V PARAMETRIC RATINGS
- INPUT LEAKAGE CURRENT I_I = 100nA (MAX) AT V_{DD} = 18V T_A = 25°C
- 100% TESTED FOR QUIESCENT CURRENT
- MEETS ALL REQUIREMENTS OF JEDEC JESD13B " STANDARD SPECIFICATIONS FOR DESCRIPTION OF B SERIES CMOS DEVICES"

DESCRIPTION

The HCF40106B is a monolithic integrated circuit fabricated in Metal Oxide Semiconductor technology available in DIP and SOP packages.

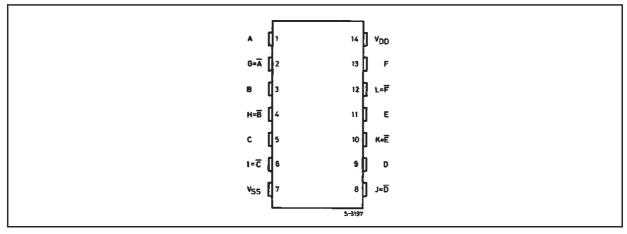


ORDER CODES

PACKAGE	TUBE	T&R
DIP	HCF40106BEY	
SOP	HCF40106BM1	HCF40106M013TR

The HCF40106B consist of six Schmitt trigger circuits. Each circuit functions as an inverter with Schmitt trigger action on the input. The trigger switches at different points for positive and negative going signals. The difference between the positive going voltage (V_P) and the negative going voltage (V_N) is defined as hysteresis voltage (V_N) .

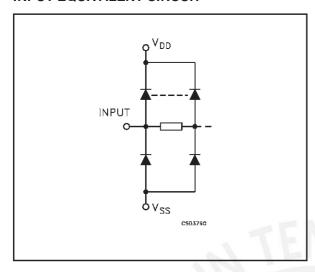
PIN CONNECTION



September 2001 1/9



INPUT EQUIVALENT CIRCUIT



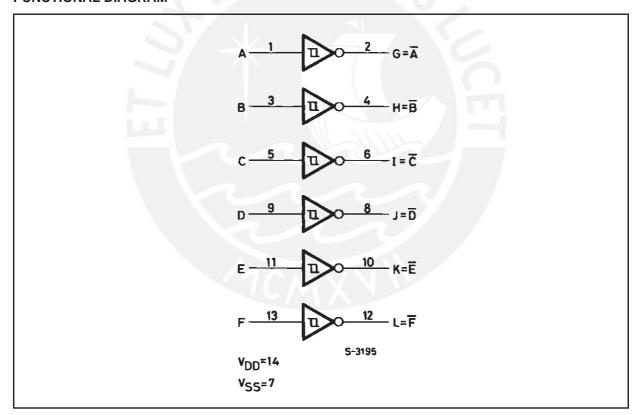
PIN DESCRIPTION

PIN No	SYMBOL	NAME AND FUNCTION
13	A, B, C, D, E, F	
2, 4, 6, 8, 10, 12	G, H, I, J, K, L	Data Outputs
7	V_{SS}	Negative Supply Voltage
14	V_{DD}	Positive Supply Voltage

TRUTH TABLE

INPUTS	OUTPUTS
A to F	G to L
	Н
H	L

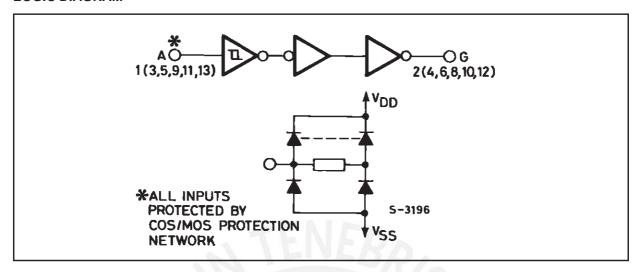
FUNCTIONAL DIAGRAM



57



LOGIC DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V _{DD}	Supply Voltage	-0.5 to +22	V
VI	DC Input Voltage	-0.5 to V _{DD} + 0.5	V
I _I	DC Input Current	± 10	mA
P _D	Power Dissipation per Package	200	mW
	Power Dissipation per Output Transistor	100	mW
T _{op}	Operating Temperature	-55 to +125	°C
T _{stg}	Storage Temperature	-65 to +150	°C

Absolute Maximum Ratings are those values beyond which damage to the device may occur. Functional operation under these conditions is not implied.

All voltage values are referred to $V_{\mbox{SS}}$ pin voltage.

RECOMMENDED OPERATING CONDITIONS

Symbol	Parameter	Value	Unit
V _{DD}	Supply Voltage	3 to 20	V
V _I	Input Voltage	0 to V _{DD}	V
T _{op}	Operating Temperature	-55 to 125	°C





DC SPECIFICATIONS

			Test Co	ndition	1				Value				
Symbol	Parameter	Vı	٧o	I _O	V _{DD}	Т	A = 25°	С	-40 to	85°C	-55 to	125°C	Unit
		(V)	(V)	(μ A)	(V)	Min.	Тур.	Max.	Min.	Max.	Min.	Max.	
IL	Quiescent Current	0/5			5		0.02	1		30		30	
		0/10			10		0.02	2		60		60	μΑ
		0/15			15		0.02	4		120		120	μΑ
		0/20			20		0.04	20		600		600	
V _{OH}	High Level Output	0/5		<1	5	4.95			4.95		4.95		
	Voltage	0/10		<1	10	9.95			9.95		9.95		V
		0/15		<1	15	14.95			14.95		14.95		
V _{OL}	Low Level Output	5/0		<1	5		0.05			0.05		0.05	
	Voltage	10/0		<1	10		0.05			0.05		0.05	V
		15/0	1	<1	15		0.05			0.05		0.05	
V _P	Positive Trigger	- 4		N.	5	2.2	2.9	3.6	2.2	3.6	2.2	3.6	
	Threshold Voltage		110		10	4.6	5.9	7.1	4.6	7.1	4.6	7.1	V
					15	6.8	8.8	10.8	6.8	10.8	6.8	10.8	
V _N	Negative Trigger	V			5	0.9	1.9	2.8	0.9	2.8	0.9	2.8	
	Threshold Voltage	5 A		1	10	2.5	3.9	5.2	2.5	5.2	2.5	5.2	\ \
					15	4	5.8	7.4	4	7.4	4	7.4	
V_{H}	Hysteresis Voltage	1	7.0		5	0.3	0.9	1.6	0.3	1.6	0.3	1.6	
		A			10	1.2	2.3	3.4	1.2	3.4	1.2	3.4	V
					15	1.6	3.5	5	1.6	5	1.6	5	
I _{OH}	Output Drive	0/5	2.5	<1	5	-1.36	-3.2		-1.15		-1.1		
	Current	0/5	4.6	<1	5	-0.44	-1	_/	-0.36	- 4	-0.36		mA
		0/10	9.5	<1	10	-1.1	-2.6	10	-0.9		-0.9		''''
		0/15	13.5	<1	15	-3.0	-6.8		-2.4		-2.4		
I _{OL}	Output Sink	0/5	0.4	<1	5	0.44	1		0.36		0.36		
	Current	0/10	0.5	<1	10	1.1	2.6		0.9	1	0.9		mΑ
		0/15	1.5	<1	15	3.0	6.8	////	2.4	7	2.4		
I _I	Input Leakage Current	0/18	Any	Input	18		±10 ⁻⁵	±0.1		±1		±1	μΑ
C _I	Input Capacitance		Any	Input	To the	676	5	7.5					pF

The Noise Margin for both "1" and "0" level is: 1V min. with V_{DD} =5V, 2V min. with V_{DD} =10V, 2.5V min. with V_{DD} =15V

$\textbf{DYNAMIC ELECTRICAL CHARACTERISTICS} \ (T_{amb} = 25^{\circ}C, \ \ C_{L} = 50 pF, \ R_{L} = 200 K\Omega, \ \ t_{f} = t_{f} = 20 \ ns)$

Comple	B		Test Condition	١	Unit		
Symbol	Parameter	V _{DD} (V)		Min.	Тур.	Max.	
t _{PLH} t _{PHL}	Propagation Delay Time	5			140	280	
		10			70	140	ns
		15			60	120	
t _{TLH} t _{THL}	Output Transition Time	5			100	200	
		10			50	100	ns
		15			40	80	

^(*) Typical temperature coefficient for all $\rm V_{DD}$ value is 0.3 %/°C.

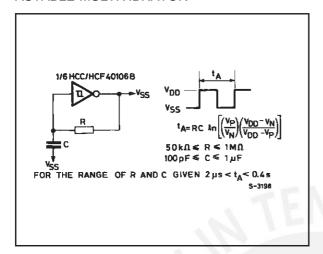
577



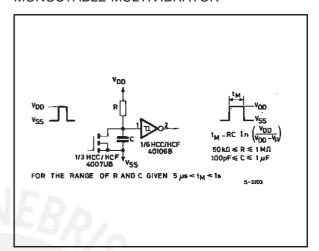


TYPICAL APPLICATIONS

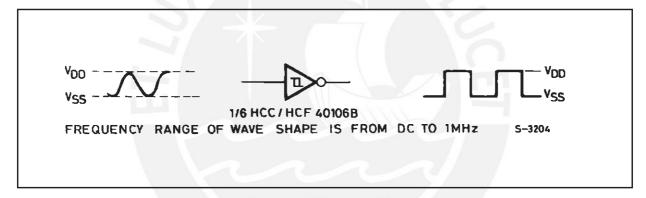
ASTABLE MULTIVIBRATOR



MONOSTABLE MULTIVIBRATOR

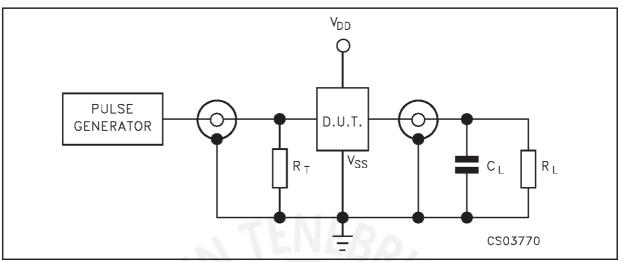


WAVE SHAPER



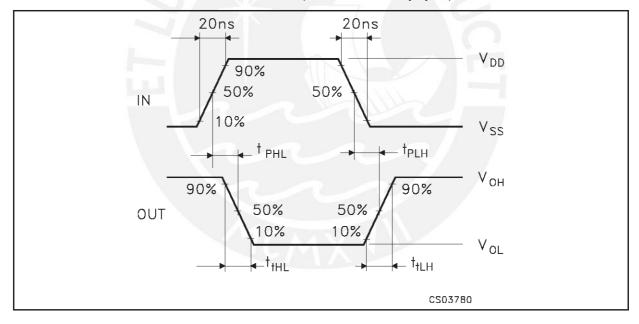


TEST CIRCUIT



 C_L = 50pF or equivalent (includes jig and probe capacitance) R_L = 200K Ω R_T = Z_{OUT} of pulse generator (typically 50 Ω)

WAVEFORM: PROPAGATION DELAY TIMES (f=1MHz; 50% duty cycle)

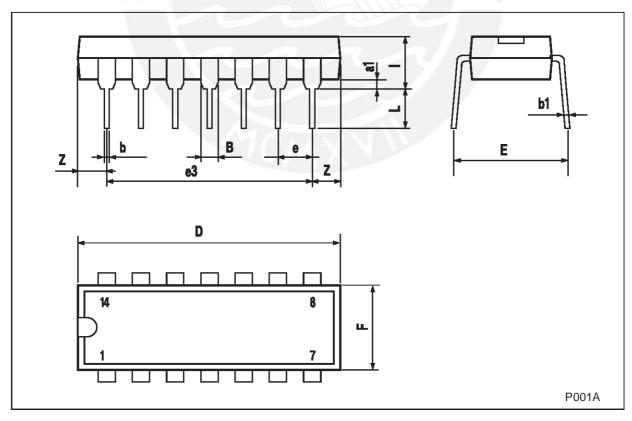


57



Plastic DIP-14 MECHANICAL DATA

DIM		mm.		inch					
DIM.	MIN.	TYP	MAX.	MIN.	TYP.	MAX.			
a1	0.51			0.020					
В	1.39		1.65	0.055		0.065			
b		0.5			0.020				
b1		0.25			0.010				
D			20			0.787			
E		8.5	LIVE	RA.	0.335				
е		2.54		21/0	0.100				
e3	- 4	15.24		No.	0.600				
F			7.1			0.280			
I	V		5.1		6	0.201			
L		3.3			0.130				
Z	1.27		2.54	0.050		0.100			

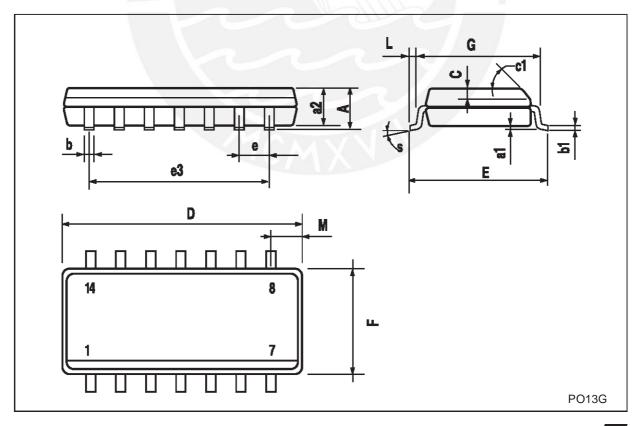


577



SO-14 MECHANICAL DATA

DIM.		mm.		inch					
DIIVI.	MIN.	TYP	MAX.	MIN.	TYP.	MAX.			
А			1.75			0.068			
a1	0.1		0.2	0.003		0.007			
a2			1.65			0.064			
b	0.35		0.46	0.013		0.018			
b1	0.19		0.25	0.007		0.010			
С		0.5			0.019				
c1			45°	(typ.)					
D	8.55	1 7	8.75	0.336		0.344			
E	5.8	1 101 1	6.2	0.228		0.244			
е	1	1.27		4.0	0.050				
e3		7.62			0.300				
F	3.8		4.0	0.149		0.157			
G	4.6		5.3	0.181		0.208			
L	0.5		1.27	0.019	()	0.050			
M			0.68			0.026			
S		•	8° (ı	max.)		1			







Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of STMicroelectronics.

© The ST logo is a registered trademark of STMicroelectronics

© 2001 STMicroelectronics - Printed in Italy - All Rights Reserved STMicroelectronics GROUP OF COMPANIES

Australia - Brazil - China - Finland - France - Germany - Hong Kong - India - Italy - Japan - Malaysia - Malta - Morocco Singapore - Spain - Sweden - Switzerland - United Kingdom © http://www.st.com





This datasheet has been download from:

 $\underline{www.datasheet catalog.com}$

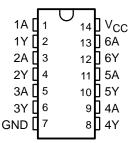
Datasheets for electronics components.





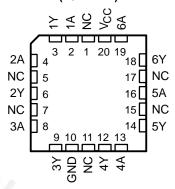
- Wide Operating Voltage Range of 2 V to 6 V
- Outputs Can Drive Up To 10 LSTTL Loads
- Low Power Consumption, 20-μA Max I_{CC}

SN54HC04...J OR W PACKAGE SN74HC04...D, N, NS, OR PW PACKAGE (TOP VIEW)



- Typical t_{pd} = 8 ns
- ±4-mA Output Drive at 5 V
- Low Input Current of 1 μA Max

SN54HC04 . . . FK PACKAGE (TOP VIEW)



NC - No internal connection

description/ordering information

The 'HC04 devices contain six independent inverters. They perform the Boolean function $Y = \overline{A}$ in positive logic.

ORDERING INFORMATION

TA	PACK	AGET	ORDERABLE PART NUMBER	TOP-SIDE MARKING
	PDIP – N	Tube of25	SN74HC04N	SN74HC04N
		Tube of 50	SN74HC04D	
4000 +- 0500	SOIC - D	Reel of 2500	SN74HC04DR	HC04
		Reel of 250	SN74HC04DT	
–40°C to 85°C	SOP - NS	Reel of 2000	SN74HC04NSR	HC04
		Tube of 90	SN74HC04PW	
	TSSOP - PW	Reel of 2000	SN74HC04PWR	HC04
		Reel of 250	SN74HC04PWT	
	CDIP – J	Tube of 25	SNJ54HC04J	SNJ54HC04J
–55°C to 125°C	CFP – W	Tube of 150	SNJ54HC04W	SNJ54HC04W
	LCCC – FK	Tube of 55	SNJ54HC04FK	SNJ54HC04FK

[†] Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

FUNCTION TABLE (each inverter)

•	
INPUT	OUTPUT
Α	Υ
Н	L
L	Н



Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.



Copyright © 2003, Texas Instruments Incorporated On products compliant to MIL-PRF-38535, all parameters are tested unless otherwise noted. On all other products, production processing these not product of the product



logic diagram (positive logic)



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage range, V _{CC}		–0.5 V to 7 V
Input clamp current, I_{IK} ($V_I < 0$ or $V_I > V_{CC}$) (see		
Output clamp current, I_{OK} ($V_O < 0$ or $V_O > V_{Cl}$		
Continuous output current, I_O ($V_O = 0$ to V_{CC})		
Continuous current through V _{CC} or GND		
Package thermal impedance, θ _{JA} (see Note 2)	: D package	
-	N package	80°C/W
	NS package .	
	PW package .	113°C/W
Storage temperature range, T _{stq}		–65°C to 150°C

[†] Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. The input and output voltage ratings may be exceeded if the input and output current ratings are observed.

recommended operating conditions (see Note 3)

			S	N54HC0	4	S	N74HC0	4	
			MIN	NOM	MAX	MIN	NOM	MAX	UNIT
Vcc	Supply voltage	· · · · · · · · · · · · · · · · · · ·	2	5	6	2	5	6	V
		V _{CC} = 2 V	1.5			1.5			
\vee_{IH}	High-level input voltage	V _{CC} = 4.5 V	3.15		7/	3.15			V
		V _{CC} = 6 V	4.2	1		4.2			
	V _{IL} Low-level input voltage	V _{CC} = 2 V			0.5	/		0.5	V
\vee_{IL}		V _{CC} = 4.5 V		///	1.35			1.35	
		V _{CC} = 6 V			1.8			1.8	
٧ı	Input voltage		0		Vcc	0		VCC	V
۷o	Output voltage	MATEM	0		Vcc	0		VCC	V
		V _{CC} = 2 V	A		1000			1000	
$\Delta t/\Delta v$	Input transition rise/fall time	V _{CC} = 4.5 V			500		-	500	ns
		V _{CC} = 6 V			400			400	
TA	Operating free-air temperature	•	-55		125	-40		85	°C

NOTE 3: All unused inputs of the device must be held at V_{CC} or GND to ensure proper device operation. Refer to the TI application report, Implications of Slow or Floating CMOS Inputs, literature number SCBA004.

^{2.} The package thermal impedance is calculated in accordance with JESD 51-7.

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS		vcc	Т	A = 25°C	;	SN54I	HC04	SN74HC04		UNIT
PARAMETER	1251 66	1201 GONDING			TYP	MAX	MIN	MAX	MIN	MAX	UNIT
			2 V	1.9	1.998		1.9		1.9		
Voн		I _{OH} = -20 μA	4.5 V	4.4	4.499		4.4		4.4		
	VI = VIH or VIL		6 V	5.9	5.999		5.9		5.9		V
		$I_{OH} = -4 \text{ mA}$	4.5 V	3.98	4.3		3.7		3.84		
		$I_{OH} = -5.2 \text{ mA}$	6 V	5.48	5.8		5.2		5.34		
			2 V		0.002	0.1		0.1		0.1	
		I _{OL} = 20 μA	4.5 V		0.001	0.1		0.1		0.1	
V _{OL}	$V_I = V_{IH}$ or V_{IL}		6 V		0.001	0.1		0.1		0.1	V
		I _{OL} = 4 mA	4.5 V		0.17	0.26		0.4		0.33	
		I _{OL} = 5.2 mA	6 V		0.15	0.26		0.4		0.33	
lį	$V_I = V_{CC}$ or 0	. 7	6 V	h D	±0.1	±100		±1000		±1000	nA
Icc	$V_I = V_{CC}$ or 0,	I _O = 0	6 V	-0	NI	2		40		20	μΑ
Ci			2 V to 6 V		3	10		10		10	pF

switching characteristics over recommended operating free-air temperature range, C_L = 50 pF (unless otherwise noted) (see Figure 1)

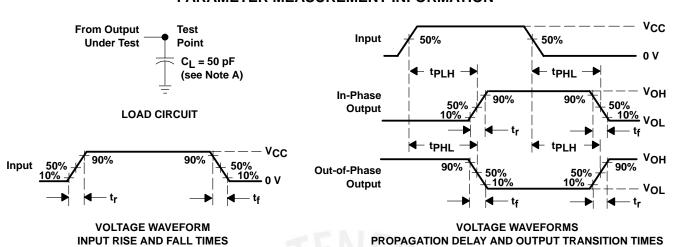
PARAMETER	FROM	FROM	то	Vaa	T,	4 = 25°C		SN54I	HC04	SN74H	HC04	UNIT
PARAMETER	(INPUT)	(OUTPUT)	Vcc	MIN	TYP	MAX	MIN	MAX	MIN	MAX	UNIT	
			2 V		45	95		145		120		
t _{pd}	Α	Y	4.5 V	111	9	19		29		24	ns	
			6 V	2///	8	16		25		20		
			2 V		38	75		110		95		
t _t		Y	4.5 V		8	15	y e	22		19	ns	
·			6 V		6	13	1	19		16		

operating characteristics, $T_A = 25^{\circ}C$

	PARAMETER	TEST CONDITIONS	TYP	UNIT
C _{pd}	Power dissipation capacitance per inverter	No load	20	pF



PARAMETER MEASUREMENT INFORMATION



NOTES: A. C_L includes probe and test-fixture capacitance.

- B. Phase relationships between waveforms were chosen arbitrarily. All input pulses are supplied by generators having the following characteristics: PRR \leq 1 MHz, Z_O = 50 Ω , t_f = 6 ns, t_f = 6 ns.
- C. The outputs are measured one at a time with one input transition per measurement.
- D. tpLH and tpHL are the same as tpd.

Figure 1. Load Circuit and Voltage Waveforms





PACKAGING INFORMATION

Orderable Device	Status ⁽¹⁾	Package Type	Package Drawing	Pins	Package Qty	e Eco Plan ⁽²⁾	Lead/Ball Finish	MSL Peak Temp ⁽³⁾
5962-8409801VCA	ACTIVE	CDIP	J	14	1	None	Call TI	Level-NC-NC-NC
5962-8409801VDA	ACTIVE	CFP	W	14	1	None	Call TI	Level-NC-NC-NC
84098012A	ACTIVE	LCCC	FK	20	1	None	Call TI	Level-NC-NC-NC
8409801CA	ACTIVE	CDIP	J	14	1	None	Call TI	Level-NC-NC-NC
8409801DA	ACTIVE	CFP	W	14	1	None	Call TI	Level-NC-NC-NC
JM38510/65701B2A	ACTIVE	LCCC	FK	20	1	None	Call TI	Level-NC-NC-NC
JM38510/65701BCA	ACTIVE	CDIP	J	14	1	None	Call TI	Level-NC-NC-NC
JM38510/65701BDA	ACTIVE	CFP	W	14	1	None	Call TI	Level-NC-NC-NC
SN54HC04J	ACTIVE	CDIP	J	14	1	None	Call TI	Level-NC-NC-NC
SN74HC04D	ACTIVE	SOIC	D	14	50	Pb-Free (RoHS)	CU NIPDAU	Level-2-260C-1 YEAR Level-1-235C-UNLIM
SN74HC04DBR	ACTIVE	SSOP	DB	14	2000	Pb-Free (RoHS)	CU NIPDAU	Level-2-260C-1 YEAR
SN74HC04DR	ACTIVE	SOIC	D	14	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN74HC04DT	ACTIVE	SOIC	D	14	250	Pb-Free (RoHS)	CU NIPDAU	Level-2-260C-1 YEAR Level-1-235C-UNLIM
SN74HC04N	ACTIVE	PDIP	N	14	25	Pb-Free (RoHS)	CU NIPDAU	Level-NC-NC-NC
SN74HC04N3	OBSOLETE	PDIP	N	14		None	Call TI	Call TI
SN74HC04NSR	ACTIVE	SO	NS	14	2000	Pb-Free (RoHS)	CU NIPDAU	Level-2-260C-1 YEAR Level-1-235C-UNLIM
SN74HC04PW	ACTIVE	TSSOP	PW	14	90	Pb-Free (RoHS)	CU NIPDAU	Level-1-250C-UNLIM
SN74HC04PWLE	OBSOLETE	TSSOP	PW	14		None	Call TI	Call TI
SN74HC04PWR	ACTIVE	TSSOP	PW	14	2000	Pb-Free (RoHS)	CU NIPDAU	Level-1-250C-UNLIM
SN74HC04PWT	ACTIVE	TSSOP	PW	14	250	Pb-Free (RoHS)	CU NIPDAU	Level-1-250C-UNLIM
SNJ54HC04FK	ACTIVE	LCCC	FK	20	1	None	Call TI	Level-NC-NC-NC
SNJ54HC04J	ACTIVE	CDIP	J	14	1	None	Call TI	Level-NC-NC-NC
SNJ54HC04W	ACTIVE	CFP	W	14	1	None	Call TI	Level-NC-NC-NC
SNV54HC04J	ACTIVE	CDIP	J	14		None	Call TI	Call TI
SNV54HC04W	ACTIVE	CFP	W	14		None	Call TI	Call TI

⁽¹⁾ The marketing status values are defined as follows:

ACTIVE: Product device recommended for new designs.

LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.

NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.

PREVIEW: Device has been announced but is not in production. Samples may or may not be available.

OBSOLETE: TI has discontinued the production of the device.

None: Not yet available Lead (Pb-Free).

Pb-Free (RoHS): TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.

⁽²⁾ Eco Plan - May not be currently available - please check http://www.ti.com/productcontent for the latest availability information and additional product content details.







Green (RoHS & no Sb/Br): TI defines "Green" to mean "Pb-Free" and in addition, uses package materials that do not contain halogens, including bromine (Br) or antimony (Sb) above 0.1% of total product weight.

(3) MSL, Peak Temp. -- The Moisture Sensitivity Level rating according to the JEDECindustry standard classifications, and peak solder temperature.

Important Information and Disclaimer: The information provided on this page represents TI's knowledge and belief as of the date that it is provided. TI bases its knowledge and belief on information provided by third parties, and makes no representation or warranty as to the accuracy of such information. Efforts are underway to better integrate information from third parties. TI has taken and continues to take reasonable steps to provide representative and accurate information but may not have conducted destructive testing or chemical analysis on incoming materials and chemicals. TI and TI suppliers consider certain information to be proprietary, and thus CAS numbers and other limited information may not be available for release.

In no event shall TI's liability arising out of such information exceed the total purchase price of the TI part(s) at issue in this document sold by TI to Customer on an annual basis.

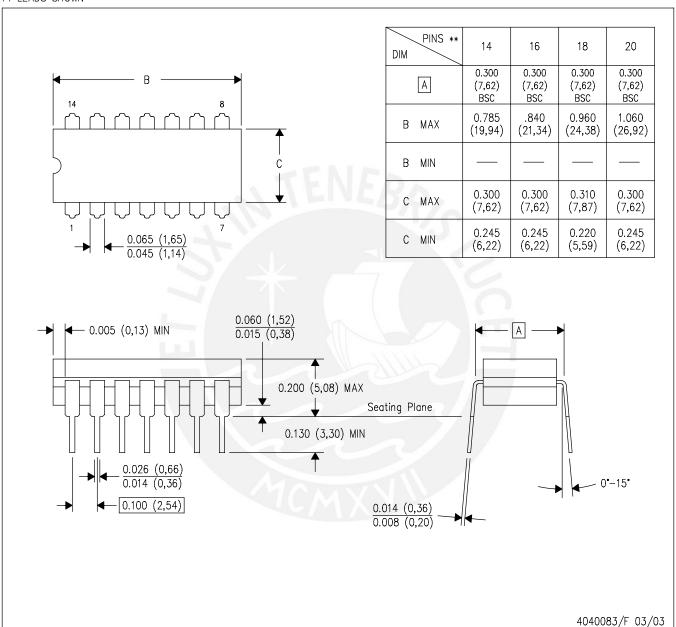




J (R-GDIP-T**)

CERAMIC DUAL IN-LINE PACKAGE

14 LEADS SHOWN

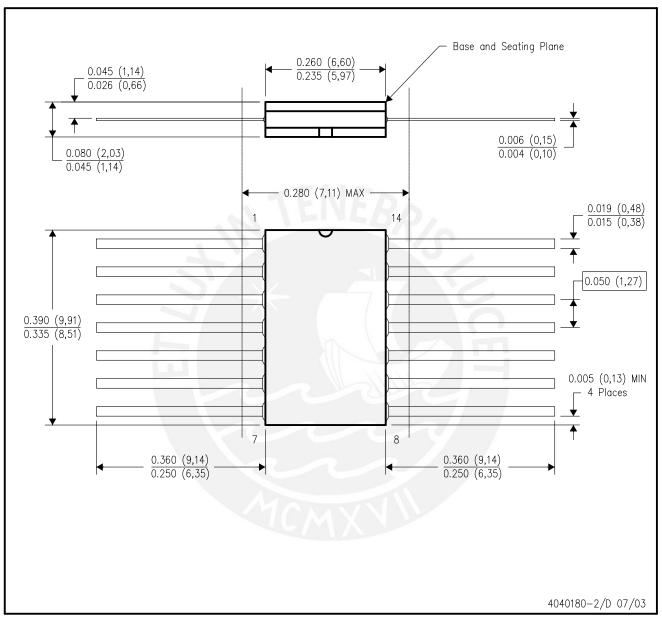


- A. All linear dimensions are in inches (millimeters).
- B. This drawing is subject to change without notice.
- C. This package is hermetically sealed with a ceramic lid using glass frit.
- D. Index point is provided on cap for terminal identification only on press ceramic glass frit seal only.
- E. Falls within MIL STD 1835 GDIP1-T14, GDIP1-T16, GDIP1-T18 and GDIP1-T20.



W (R-GDFP-F14)

CERAMIC DUAL FLATPACK



- A. All linear dimensions are in inches (millimeters).
- B. This drawing is subject to change without notice.
- C. This package can be hermetically sealed with a ceramic lid using glass frit.
- D. Index point is provided on cap for terminal identification only.
- E. Falls within MIL STD 1835 GDFP1-F14 and JEDEC MO-092AB

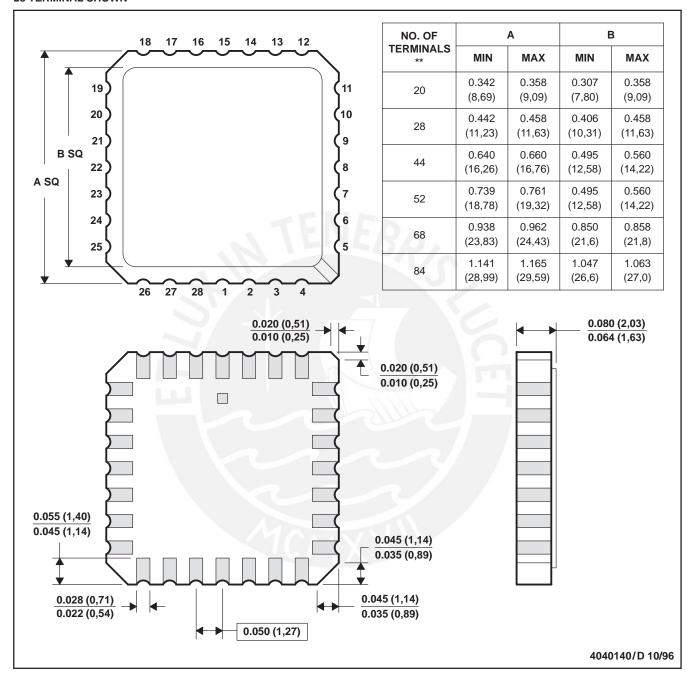




FK (S-CQCC-N**)

28 TERMINAL SHOWN

LEADLESS CERAMIC CHIP CARRIER



- NOTES: A. All linear dimensions are in inches (millimeters).
 - B. This drawing is subject to change without notice.
 - C. This package can be hermetically sealed with a metal lid.
 - D. The terminals are gold plated.
 - E. Falls within JEDEC MS-004

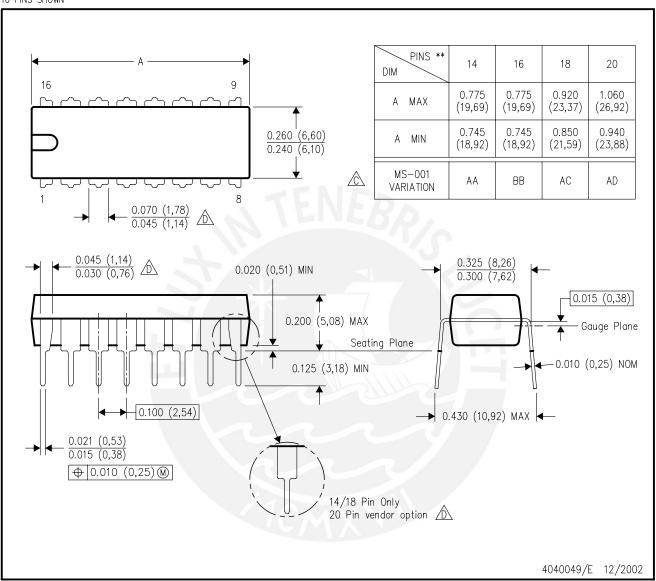




N (R-PDIP-T**)

PLASTIC DUAL-IN-LINE PACKAGE

16 PINS SHOWN



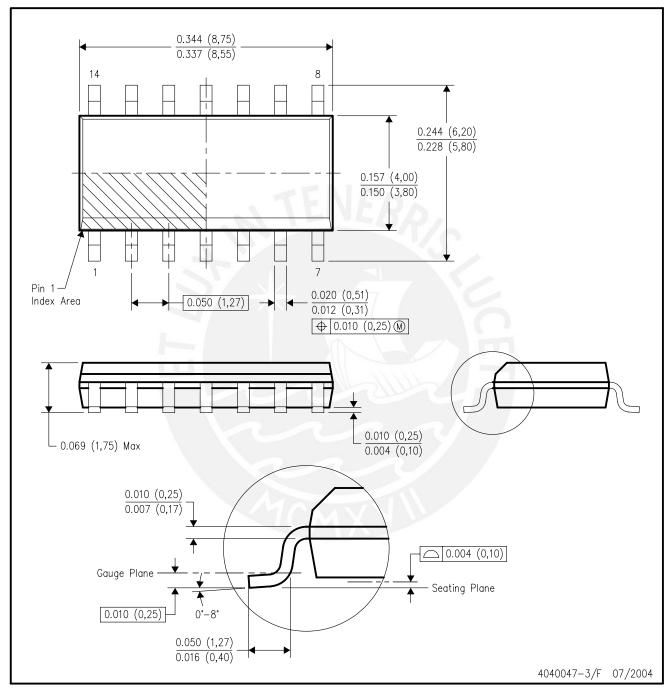
- A. All linear dimensions are in inches (millimeters).
- B. This drawing is subject to change without notice.
- Falls within JEDEC MS-001, except 18 and 20 pin minimum body length (Dim A).
- The 20 pin end lead shoulder width is a vendor option, either half or full width.





D (R-PDSO-G14)

PLASTIC SMALL-OUTLINE PACKAGE



- A. All linear dimensions are in inches (millimeters).
- B. This drawing is subject to change without notice.
- C. Body dimensions do not include mold flash or protrusion not to exceed 0.006 (0,15).
- D. Falls within JEDEC MS-012 variation AB.



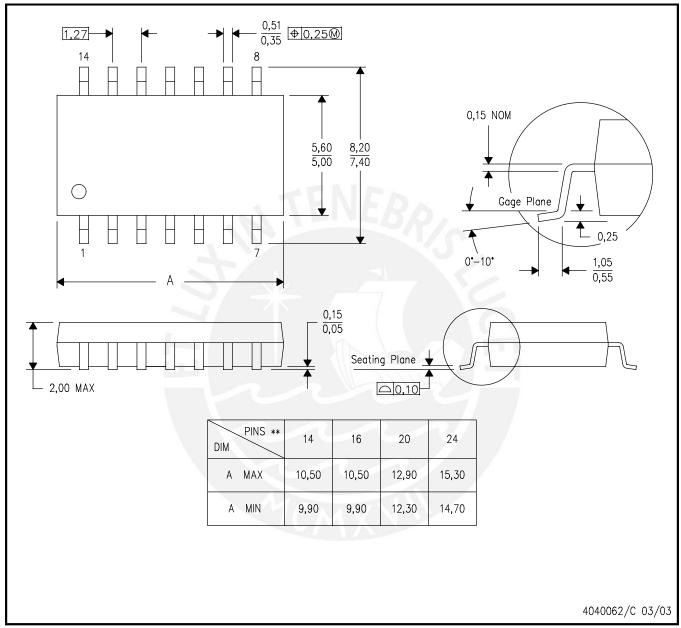


MECHANICAL DATA

NS (R-PDSO-G**)

14-PINS SHOWN

PLASTIC SMALL-OUTLINE PACKAGE



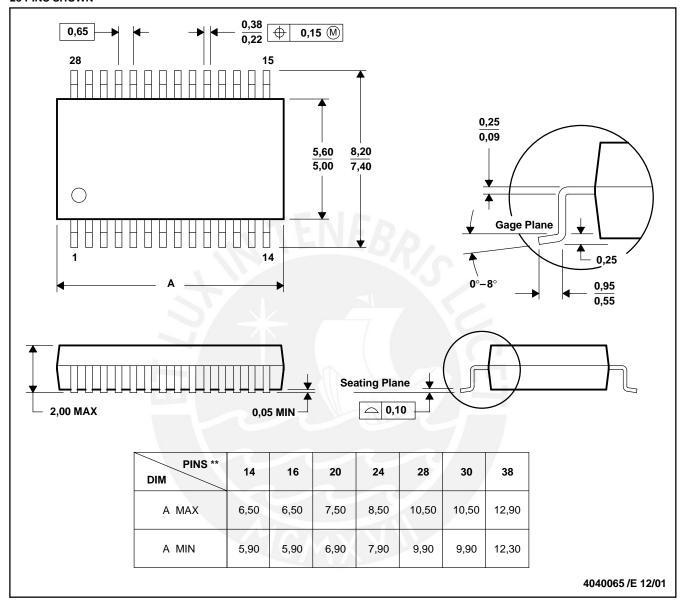
- a. All linear dimensions are in millimeters.
- B. This drawing is subject to change without notice.
- C. Body dimensions do not include mold flash or protrusion, not to exceed 0,15.



DB (R-PDSO-G**)

PLASTIC SMALL-OUTLINE

28 PINS SHOWN



NOTES: A. All linear dimensions are in millimeters.

B. This drawing is subject to change without notice.

C. Body dimensions do not include mold flash or protrusion not to exceed 0,15.

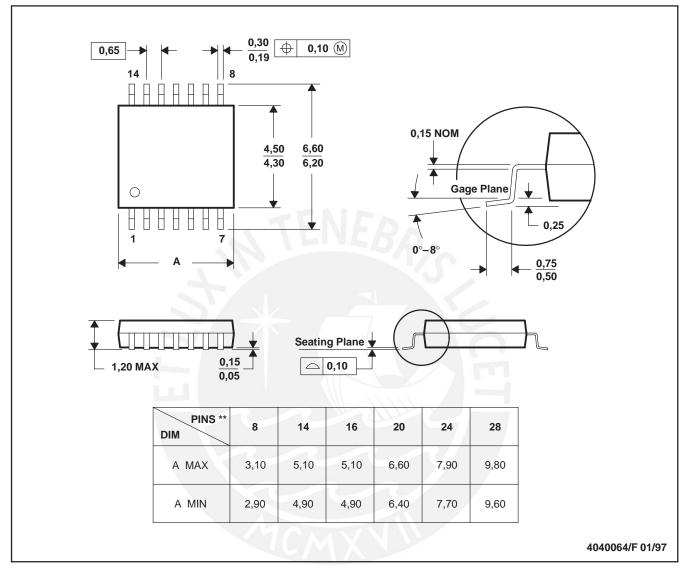
D. Falls within JEDEC MO-150



PW (R-PDSO-G**)

14 PINS SHOWN

PLASTIC SMALL-OUTLINE PACKAGE



NOTES: A. All linear dimensions are in millimeters.

B. This drawing is subject to change without notice.

C. Body dimensions do not include mold flash or protrusion not to exceed 0,15.

D. Falls within JEDEC MO-153



IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		Applications	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments

Post Office Box 655303 Dallas, Texas 75265

Copyright © 2005, Texas Instruments Incorporated





LM358, LM258, LM2904, LM2904V

Dual Low Power Operational Amplifiers

Utilizing the circuit designs perfected for recently introduced Quad Operational Amplifiers, these dual operational amplifiers feature 1) low power drain, 2) a common mode input voltage range extending to ground/VEE, 3) single supply or split supply operation and 4) pinouts compatible with the popular MC1558 dual operational amplifier. The LM158 series is equivalent to one–half of an LM124.

These amplifiers have several distinct advantages over standard operational amplifier types in single supply applications. They can operate at supply voltages as low as 3.0 V or as high as 32 V, with quiescent currents about one–fifth of those associated with the MC1741 (on a per amplifier basis). The common mode input range includes the negative supply, thereby eliminating the necessity for external biasing components in many applications. The output voltage range also includes the negative power supply voltage.

- Short Circuit Protected Outputs
- True Differential Input Stage
- Single Supply Operation: 3.0 V to 32 V
- Low Input Bias Currents
- Internally Compensated
- Common Mode Range Extends to Negative Supply
- Single and Split Supply Operation
- Similar Performance to the Popular MC1558
- ESD Clamps on the Inputs Increase Ruggedness of the Device without Affecting Operation

MAXIMUM RATINGS (T_A = +25°C, unless otherwise noted.)

Rating	Symbol	LM258 LM358	LM2904 LM2904V	Unit
Power Supply Voltages Single Supply Split Supplies	V _{CC}	32 ±16	26 ±13	Vdc
Input Differential Voltage Range (Note 1)	VIDR	±32	±26	Vdc
Input Common Mode Voltage Range (Note 2)	VICR	-0.3 to 32	-0.3 to 26	Vdc
Output Short Circuit Duration	tsc	Continuous		
Junction Temperature	TJ	150		°C
Storage Temperature Range	T _{stg}	-55 to +125		°C
Operating Ambient Temperature Range	TA			°C
LM258		-25 to +85	-	
LM358		0 to +70	- 40.5	
LM2904 LM2904V		_ _	-40 to +105 -40 to +125	

NOTES: 1. Split Power Supplies.

For Supply Voltages less than 32 V for the LM258/358 and 26 V for the LM2904, the absolute maximum input voltage is equal to the supply voltage.

DUAL DIFFERENTIAL INPUT OPERATIONAL AMPLIFIERS

SEMICONDUCTOR TECHNICAL DATA

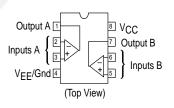


N SUFFIX
PLASTIC PACKAGE
CASE 626



D SUFFIX
PLASTIC PACKAGE
CASE 751
(SO-8)

PIN CONNECTIONS



ORDERING INFORMATION

Device	Operating Temperature Range	Package
LM2904D	T _A = -40° to +105°C	SO-8
LM2904N	1A = 40 10 + 103 C	Plastic DIP
LM2904VD	$T_A = -40^{\circ} \text{ to } +125^{\circ}\text{C}$	SO-8
LM2904VN	1A = -40 10 +123 C	Plastic DIP
LM258D	$T_{A} = -25^{\circ} \text{ to } +85^{\circ}\text{C}$	SO-8
LM258N	1A = 20 to 100 0	Plastic DIP
LM358D	T. 00 to 1700C	SO-8
LM358N	$T_A = 0^{\circ} \text{ to } +70^{\circ}\text{C}$	Plastic DIP

TESIS PUCP

LM358, LM258, LM2904, LM2904V



ELECTRICAL CHARACTERISTICS (VCC.

ELECTRICAL CHARACTERISTIC	- (00		LM258		^ -	LM358			LM2904		L	M2904	V	
Characteristic	Symbol	Min	Тур	Max	Min	Тур	Max	Min	Тур	Max	Min	Тур	Max	Unit
Input Offset Voltage $\begin{array}{l} \text{V}_{CC} = 5.0 \text{ V to } 30 \text{ V } (26 \text{ V for} \\ \text{LM2904, V), V}_{IC} = 0 \text{ V to V}_{CC} -1.7 \text{ V,} \\ \text{V}_{O} \simeq 1.4 \text{ V, R}_{S} = 0 \Omega \end{array}$	VIO													mV
$T_A = 25^{\circ}C$ $T_A = T_{high}$ (Note 1) $T_A = T_{low}$ (Note 1)		- - -	2.0	5.0 7.0 2.0	-	2.0 - -	7.0 9.0 9.0	- - -	2.0	7.0 10 10	- - -	- - -	- 13 10	
Average Temperature Coefficient of Input Offset Voltage $T_{A} = T_{high} \text{ to } T_{low} \text{ (Note 1)}$	ΔV _{IO} /ΔΤ	-	7.0	-	-	7.0	-	-	7.0	-	-	7.0	_	μV/°0
Input Offset Current TA = Thigh to Tlow (Note 1) Input Bias Current The Transit to Transit (Note 1)	I _{IO}	- - -	3.0 - -45 -50	30 100 –150 –300		5.0 - -45 -50	50 150 –250 –500		5.0 45 –45 –50	50 200 –250 –500		5.0 45 –45 –50	50 200 –250 –500	nA
T _A = T _{high} to T _{low} (Note 1) Average Temperature Coefficient of Input Offset Current T _A = T _{high} to T _{low} (Note 1)	ΔΙ _{ΙΟ} /ΔΤ	-	10	-	-	10	-	-	10	-	-	10	-	pA/°(
Input Common Mode Voltage Range (Note 2),V _{CC} = 30 V (26 V for LM2904, V) V _{CC} = 30 V (26 V for LM2904, V), TA = Thigh to Tlow	VICR	0	1-7	28.3 28	0	8	28.3 28	0	- -	24.3 24	0	- -	24.3 24	V
Differential Input Voltage Range	V _{IDR}	- 1	-	Vcc	-		Vcc) -	_	Vcc	-	-	Vcc	V
Large Signal Open Loop Voltage Gain $R_L = 2.0 \text{ k}\Omega$, $V_{CC} = 15 \text{ V}$, For Large V_O Swing,	Avol	50	100	-	25	100		25	100	-	25	100	-	V/m\
$T_A = T_{high}$ to T_{low} (Note 1)		25	-	/-	15	-	_	15	_	-	15	-	-	
Channel Separation $1.0 \text{ kHz} \le \text{f} \le 20 \text{ kHz}$, Input Referenced	CS	-	-120	-	- (-120	-	1	-120	-	_	-120	-	dB
Common Mode Rejection $R_{S} \leq 10 \; k\Omega$	CMR	70	85		65	70		50	70	-	50	70	_	dB
Power Supply Rejection	PSR	65	100	-	65	100	-	50	100	j -	50	100	-	dB
	VOH	3.3 26 27	3.5 - 28		3.3 26 27	3.5 –	7	3.3 22 23	3.5 - 24	- - -	3.3 22 23	3.5 - 24	- - -	V
Output Voltage–Low Limit $V_{CC} = 5.0 \text{ V}, R_L = 10 \text{ k}\Omega, T_A = T_{high} \text{ to } T_{low} \text{ (Note 1)}$	VOL	1	5.0	20	Ϋ́	5.0	20	-	5.0	20	-	5.0	20	mV
Output Source Current V _{ID} = +1.0 V, V _{CC} = 15 V	I _O +	20	40	_	20	40	-	20	40	-	20	40	-	mA
Output Sink Current $V_{ID} = -1.0 \text{ V}, V_{CC} = 15 \text{ V}$ $V_{ID} = -1.0 \text{ V}, V_{O} = 200 \text{ mV}$	IO-	10 12	20 50	- -	10 12	20 50	- -	10 –	20 -	-	10 –	20 -	- -	mA μA
Output Short Circuit to Ground (Note 3)	Isc	-	40	60	-	40	60	-	40	60	_	40	60	mA
Power Supply Current ($T_A = T_{high}$ to T_{low}) (Note 1) $V_{CC} = 30 \text{ V } (26 \text{ V for LM2904, V}),$ $V_{O} = 0 \text{ V, } R_I = \infty$	ICC	_	1.5	3.0	-	1.5	3.0	_	1.5	3.0	-	1.5	3.0	mA
$V_{CC} = 5 \text{ V}, \ V_{C} = 0 \text{ V}, \ R_{L} = \infty$		_	0.7	1.2	_	0.7	1.2	_	0.7	1.2	-	0.7	1.2	

= -25°C for LM258

= +125°C for LM2904V = +85°C for LM258

= 0°C for LM358

= +70°C for LM358

^{2.} The input common mode voltage or either input signal voltage should not be allowed to go negative by more than 0.3 V. The upper end of the common mode voltage range is V_{CC} –1.7 V.

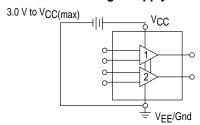
3. Short circuits from the output to V_{CC} can cause excessive heating and eventual destruction. Destructive dissipation can result from simultaneous shorts

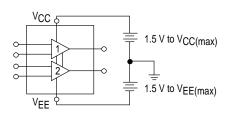
on all amplifiers.



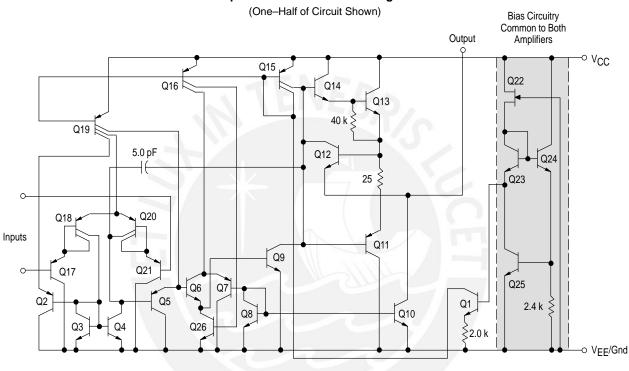
Single Supply

Split Supplies





Representative Schematic Diagram



CIRCUIT DESCRIPTION

The LM258 series is made using two internally compensated, two-stage operational amplifiers. The first stage of each consists of differential input devices Q20 and Q18 with input buffer transistors Q21 and Q17 and the differential to single ended converter Q3 and Q4. The first stage performs not only the first stage gain function but also performs the level shifting and transconductance reduction functions. By reducing the transconductance, a smaller compensation capacitor (only 5.0 pF) can be employed, thus saving chip area. The transconductance reduction is accomplished by splitting the collectors of Q20 and Q18. Another feature of this input stage is that the input common mode range can include the negative supply or ground, in single supply operation, without saturating either the input devices or the differential to single-ended converter. The second stage consists of a standard current source load amplifier stage.

Each amplifier is biased from an internal-voltage regulator which has a low temperature coefficient thus giving each amplifier good temperature characteristics as well as excellent power supply rejection.

Large Signal Voltage Follower Response

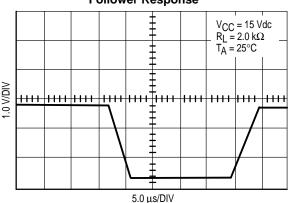




Figure 1. Input Voltage Range 20 18 16 V_I, INPUT VOLTAGE (V) 14 12 10 Negative 8.0 Positive 6.0 4.0 2.0 2.0 16 18 20 0 4.0 6.0 8.0 10 12 14 VCC/VEE, POWER SUPPLY VOLTAGES (V)

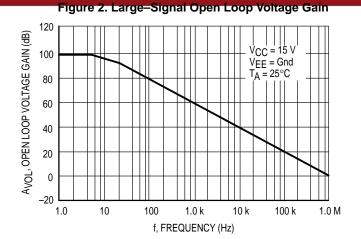
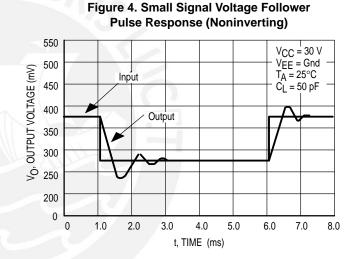
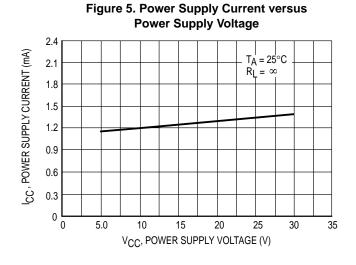


Figure 3. Large-Signal Frequency Response 14 VOR, OUTPUT VOLTAGE RANGE (Vpp) $R_L = 2.0 \text{ k}\Omega$ 12 $V_{CC} = 15 V$ VEE = Gnd 10 Gain = -100 $R_I = 1.0 \text{ k}\Omega$ 8.0 $R_F = 100 \text{ k}\Omega$ 6.0 4.0 2.0 0 1.0 1000 f, FREQUENCY (kHz)





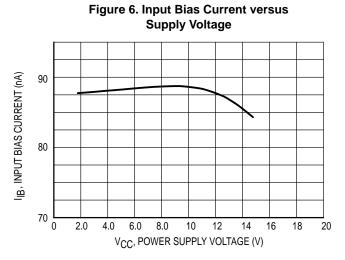




Figure 7. Voltage Reference

Figure 8. Wien Bridge Oscillator

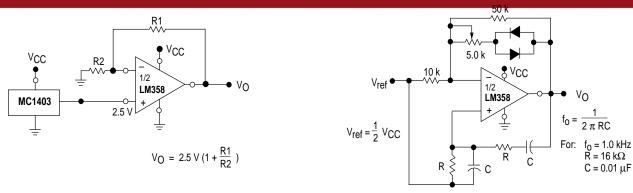


Figure 9. High Impedance Differential Amplifier

Figure 10. Comparator with Hysteresis

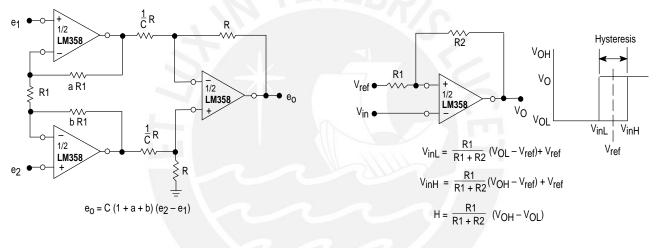


Figure 11. Bi-Quad Filter

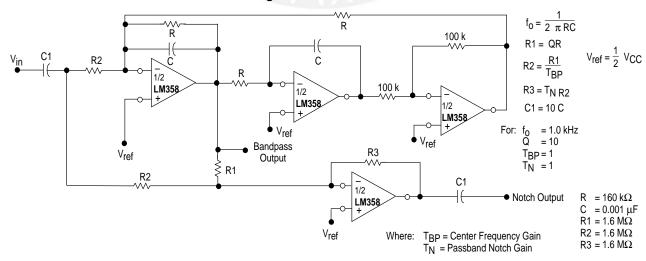
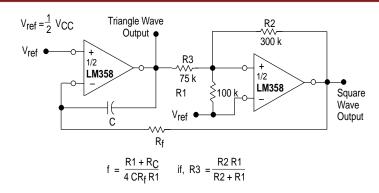
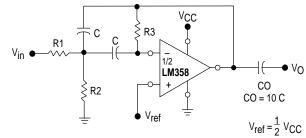




Figure 12. Function Generator

Figure 13. Multiple Feedback Bandpass Filter





Given: f_0 = center frequency $A(f_0)$ = gain at center frequency

Choose value f_0 , C

Then: R3 =
$$\frac{Q}{\pi f_0 C}$$

R1 = $\frac{R3}{2 A(f_0)}$
R2 = $\frac{R1 R3}{40^2 R1 - R}$

For less than 10% error from operational amplifier. $\frac{Q_0 f_0}{BW} < 0.1$

Where fo and BW are expressed in Hz.

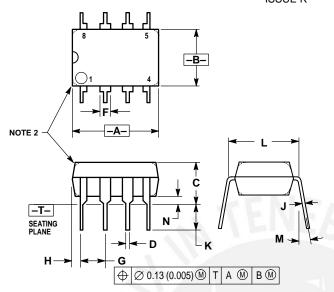
If source impedance varies, filter may be preceded with voltage follower buffer to stabilize filter parameters.



OUTLINE DIMENSIONS

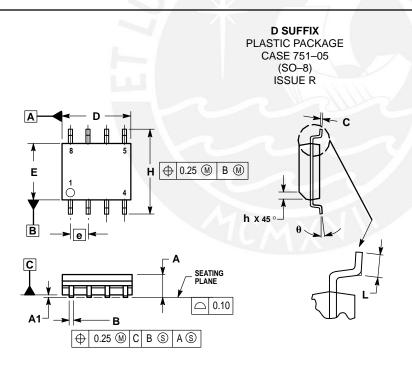
N SUFFIX

PLASTIC PACKAGE CASE 626-05 ISSUE K



- 1. DIMENSION L TO CENTER OF LEAD WHEN FORMED PARALLEL.
- PACKAGE CONTOUR OPTIONAL (ROUND OR SQUARE CORNERS).
- DIMENSIONING AND TOLERANCING PER ANSI
 Y14.5M, 1982.

	MILLIN	IETERS	INCHES			
DIM	MIN	MAX	MIN	MAX		
Α	9.40	10.16	0.370	0.400		
В	6.10	6.60	0.240	0.260		
С	3.94	4.45	0.155	0.175		
D	0.38	0.51	0.015	0.020		
F	1.02	1.78	0.040	0.070		
G	2.54	BSC	0.100 BSC			
Н	0.76	1.27	0.030	0.050		
J	0.20	0.30	0.008	0.012		
K	2.92	3.43	0.115	0.135		
L	7.62	BSC	0.300	BSC		
M		10°		10°		
N	0.76	1.01	0.030	0.040		



- VOIES:

 1. DIMENSIONING AND TOLERANCING PER ASME Y14.5M, 1994.

 2. DIMENSIONS ARE IN MILLIMETERS.

 3. DIMENSION D AND E DO NOT INCLUDE MOLD PROTRUSION.
- MAXIMUM MOLD PROTRUSION 0.15 PER SIDE.
 DIMENSION B DOES NOT INCLUDE MOLD
 PROTRUSION. ALLOWABLE DAMBAR
 PROTRUSION SHALL BE 0.127 TOTAL IN EXCESS
 OF THE B DIMENSION AT MAXIMUM MATERIAL CONDITION.

	MILLIN	IETERS
DIM	MIN	MAX
Α	1.35	1.75
A1	0.10	0.25
В	0.35	0.49
C	0.18	0.25
D	4.80	5.00
Е	3.80	4.00
е	1.27	BSC
Η	5.80	6.20
h	0.25	0.50
۲	0.40	1.25
θ	0 °	7°
		<u> </u>





Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

How to reach us:

USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036. 1–800–441–2447 or 602–303–5454

MFAX: RMFAX0@email.sps.mot.com – TOUCHTONE 602–244–6609 INTERNET: http://Design-NET.com

JAPAN: Nippon Motorola Ltd.; Tatsumi–SPD–JLDC, 6F Seibu–Butsuryu–Center, 3–14–2 Tatsumi Koto–Ku, Tokyo 135, Japan. 03–81–3521–8315

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852–26629298







This datasheet has been downloaded from:

www. Data sheet Catalog.com

Datasheets for electronic components.





National Semiconductor

LM35

Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±1/4°C at room temperature and $\pm 3/4$ °C over a full -55 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a -55° to +150°C temperature range, while the LM35C is rated for a -40° to +110°C range (-10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteeable (at +25°C)
- Rated for full -55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 µA current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only ±½°C typical
- \blacksquare Low impedance output, 0.1 Ω for 1 mA load

Typical Applications

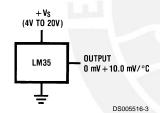
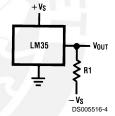


FIGURE 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)



Choose R₁ = $-V_S/50 \mu A$ V _{OUT}=+1,500 mV at +150°C = +250 mV at +25°C = -550 mV at -55°C

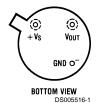
FIGURE 2. Full-Range Centigrade Temperature Sensor



Connection Diagrams



TO-46 Metal Can Package*



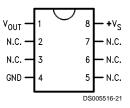
*Case is connected to negative pin (GND)

Order Number LM35H, LM35AH, LM35CH, LM35CAH or LM35DH
See NS Package Number H03H

TO-92 Plastic Package



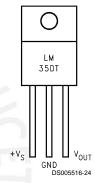
Order Number LM35CZ, LM35CAZ or LM35DZ See NS Package Number Z03A SO-8 Small Outline Molded Package



N.C. = No Connection

Top View Order Number LM35DM See NS Package Number M08A

> TO-220 Plastic Package*



*Tab is connected to the negative pin (GND).

Note: The LM35DT pinout is different than the discontinued LM35DP.

Order Number LM35DT See NS Package Number TA03F



Absolute Maximum Ratings (Note 10)

TO-92 and TO-220 Package (Soldering 10 seconds)

please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage +35 V to -0.2 VOutput Voltage +6 V to -1.0 VOutput Current 10 mA

Storage Temp.;

TO-46 Package, -60°C to +180°C
TO-92 Package, -60°C to +150°C
SO-8 Package, -65°C to +150°C
TO-220 Package, -65°C to +150°C

Lead Temp.:

TO-46 Package,

(Soldering, 10 seconds)

SO Package (Note 12)

Vapor Phase (60 seconds) 215°C Infrared (15 seconds) 220°C ESD Susceptibility (Note 11) 2500V

Specified Operating Temperature Range: $\rm T_{MIN}$ to T $_{\rm MAX}$ (Note 2)

LM35D 0° C to +100 $^{\circ}$ C

Electrical Characteristics

(Notes 1, 6)

			LM35A	ILV	A			
Parameter	Conditions	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Units (Max.)
Accuracy	T _A =+25°C	±0.2	±0.5	1997	±0.2	±0.5		°C
(Note 7)	T _A =-10°C	±0.3			±0.3		±1.0	°C
	T _A =T _{MAX}	±0.4	±1.0		±0.4	±1.0		°C
	T _A =T _{MIN}	±0.4	±1.0		±0.4		±1.5	°C
Nonlinearity	T _{MIN} ST _A ST _{MAX}	±0.18	1	±0.35	±0.15		±0.3	°C
(Note 8)						1 1 1		
Sensor Gain	T _{MIN} ST _A ST _{MAX}	+10.0	+9.9,	1///	+10.0		+9.9,	mV/°C
(Average Slope)			+10.1	5111			+10.1	
Load Regulation	T _A =+25°C	±0.4	±1.0		±0.4	±1.0		mV/mA
(Note 3) $0 \le I_L \le 1 \text{ mA}$	$T_{MIN} \leq T_A \leq T_{MAX}$	±0.5		±3.0	±0.5		±3.0	mV/mA
Line Regulation	T _A =+25°C	±0.01	±0.05		±0.01	±0.05		mV/V
(Note 3)	4V≤V _S ≤30V	±0.02		±0.1	±0.02		±0.1	mV/V
Quiescent Current	V _S =+5V, +25°C	56	67		56	67		μΑ
(Note 9)	V _S =+5V	105		131	91		114	μΑ
	V _S =+30V, +25°C	56.2	68	$\vee \vee$	56.2	68		μΑ
	V _S =+30V	105.5	MI	133	91.5		116	μΑ
Change of	4V≤V _S ≤30V, +25°C	0.2	1.0		0.2	1.0		μΑ
Quiescent Current	4V≤V _S ≤30V	0.5		2.0	0.5		2.0	μΑ
(Note 3)								
Temperature		+0.39		+0.5	+0.39		+0.5	μΑ/°C
Coefficient of								
Quiescent Current								
Minimum Temperature	In circuit of	+1.5		+2.0	+1.5		+2.0	°C
for Rated Accuracy	Figure 1, I _L =0							
Long Term Stability	T _J =T _{MAX} , for 1000 hours	±0.08			±0.08			°C

300°C

www.national.com

Electrical Characteristics



(Notes 1, 6)

			LM35		L	LM35C, LM35D		
Parameter	Conditions		Tested	Design		Tested	Design	Units
		Typical	Limit	Limit	Typical	Limit	Limit	(Max.)
			(Note 4)	(Note 5)		(Note 4)	(Note 5)	
Accuracy,	T _A =+25°C	±0.4	±1.0		±0.4	±1.0		°C
LM35, LM35C	T _A =-10°C	±0.5			±0.5		±1.5	°C
(Note 7)	T A=TMAX	±0.8	±1.5		±0.8		±1.5	°C
	T _A =T _{MIN}	±0.8		±1.5	±0.8		±2.0	°C
Accuracy, LM35D	T _A =+25°C				±0.6	±1.5		°C
(Note 7)	$T_A = T_{MAX}$				±0.9		±2.0	°C
	$T_A = T_{MIN}$				±0.9		±2.0	°C
Nonlinearity	$T_{MIN} \le T_A \le T_{MAX}$	±0.3		±0.5	±0.2		±0.5	°C
(Note 8)								
Sensor Gain	$T_{MIN} \le T_A \le T_{MAX}$	+10.0	+9.8,		+10.0		+9.8,	mV/°C
(Average Slope)			+10.2				+10.2	
Load Regulation	T _A =+25°C	±0.4	±2.0		±0.4	±2.0		mV/mA
(Note 3) 0≤I _L ≤1 mA	$T_{MIN} \le T_A \le T_{MAX}$	±0.5	150	±5.0	±0.5		±5.0	mV/mA
Line Regulation	T _A =+25°C	±0.01	±0.1	TV I	±0.01	±0.1		mV/V
(Note 3)	4V≤V _S ≤30V	±0.02	10.00	±0.2	±0.02		±0.2	mV/V
Quiescent Current	V _S =+5V, +25°C	56	80		56	80		μA
(Note 9)	V _S =+5V	105	7	158	91		138	μA
	V _S =+30V, +25°C	56.2	82	A	56.2	82		μA
	V _S =+30V	105.5		161	91.5		141	μA
Change of	4V≤V _S ≤30V, +25°C	0.2	2.0		0.2	2.0		μA
Quiescent Current	4V≤V _S ≤30V	0.5		3.0	0.5		3.0	μA
(Note 3)								
Temperature		+0.39		+0.7	+0.39		+0.7	μΑ/°C
Coefficient of								
Quiescent Current				-19/				
Minimum Temperature	In circuit of	+1.5	4	+2.0	+1.5		+2.0	°C
for Rated Accuracy	Figure 1, I _L =0							
Long Term Stability	$T_{J}=T_{MAX}$, for	±0.08			±0.08			°C
	1000 hours	AL	111					

Note 1: Unless otherwise noted, these specifications apply: $-55^{\circ}C \le T_{J} \le +150^{\circ}C$ for the LM35 and LM35A; $-40^{\circ} \le T_{J} \le +110^{\circ}C$ for the LM35C and LM35CA; and $0^{\circ} \le T_{J} \le +100^{\circ}C$ for the LM35D. $V_{S} = +5^{\circ}V$ dc and $I_{LOAD} = 50$ μ A, in the circuit of *Figure 2*. These specifications also apply from $+2^{\circ}C$ to T_{MAX} in the circuit of *Figure 1*. Specifications in **boldface** apply over the full rated temperature range.

Note 2: Thermal resistance of the TO-46 package is 400°C/W, junction to ambient, and 24°C/W junction to case. Thermal resistance of the TO-92 package is 180°C/W junction to ambient. Thermal resistance of the small outline molded package is 220°C/W junction to ambient. Thermal resistance of the TO-220 package is 90°C/W junction to ambient. For additional thermal resistance information see table in the Applications section.

Note 3: Regulation is measured at constant junction temperature, using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.

Note 4: Tested Limits are guaranteed and 100% tested in production.

Note 5: Design Limits are guaranteed (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.

Note 6: Specifications in boldface apply over the full rated temperature range.

Note 7: Accuracy is defined as the error between the output voltage and 10mv/°C times the device's case temperature, at specified conditions of voltage, current, and temperature (expressed in °C).

Note 8: Nonlinearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line, over the device's rated temperature range.

Note 9: Quiescent current is defined in the circuit of Figure 1.

Note 10: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions. See Note 1.

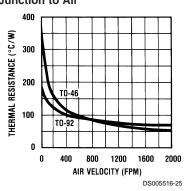
Note 11: Human body model, 100 pF discharged through a 1.5 $k\Omega$ resistor.

Note 12: See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" or the section titled "Surface Mount" found in a current National Semiconductor Linear Data Book for other methods of soldering surface mount devices.

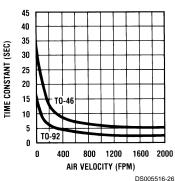


Typical Performance Characteristics

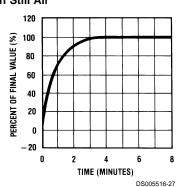
Thermal Resistance Junction to Air



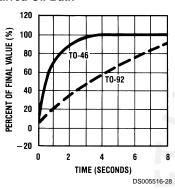
Thermal Time Constant



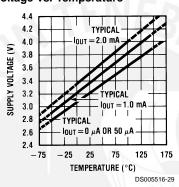
Thermal Response in Still Air



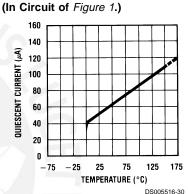
Thermal Response in Stirred Oil Bath



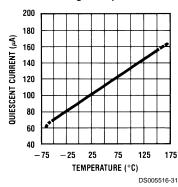
Minimum Supply Voltage vs. Temperature



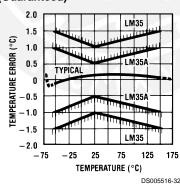
Quiescent Current vs. Temperature



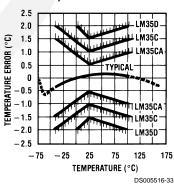
Quiescent Current vs. Temperature (In Circuit of Figure 2.)



Accuracy vs. Temperature (Guaranteed)



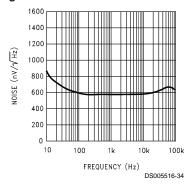
Accuracy vs. Temperature (Guaranteed)



Typical Performance Characteristics (Continued)



Noise Voltage



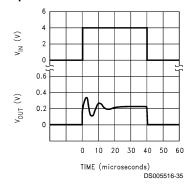
Applications

The LM35 can be applied easily in the same way as other integrated-circuit temperature sensors. It can be glued or cemented to a surface and its temperature will be within about 0.01°C of the surface temperature.

This presumes that the ambient air temperature is almost the same as the surface temperature; if the air temperature were much higher or lower than the surface temperature, the actual temperature of the LM35 die would be at an intermediate temperature between the surface temperature and the air temperature. This is expecially true for the TO-92 plastic package, where the copper leads are the principal thermal path to carry heat into the device, so its temperature might be closer to the air temperature than to the surface temperature.

To minimize this problem, be sure that the wiring to the LM35, as it leaves the device, is held at the same temperature as the surface of interest. The easiest way to do this is to cover up these wires with a bead of epoxy which will insure that the leads and wires are all at the same temperature as the surface, and that the LM35 die's temperature will not be affected by the air temperature.

Start-Up Response



The TO-46 metal package can also be soldered to a metal surface or pipe without damage. Of course, in that case the V- terminal of the circuit will be grounded to that metal. Alternatively, the LM35 can be mounted inside a sealed-end metal tube, and can then be dipped into a bath or screwed into a threaded hole in a tank. As with any IC, the LM35 and accompanying wiring and circuits must be kept insulated and dry, to avoid leakage and corrosion. This is especially true if the circuit may operate at cold temperatures where condensation can occur. Printed-circuit coatings and varnishes such as Humiseal and epoxy paints or dips are often used to insure that moisture cannot corrode the LM35 or its connections.

These devices are sometimes soldered to a small light-weight heat fin, to decrease the thermal time constant and speed up the response in slowly-moving air. On the other hand, a small thermal mass may be added to the sensor, to give the steadiest reading despite small deviations in the air temperature.

Temperature Rise of LM35 Due To Self-heating (Thermal Resistance, θ_{JA})

	TO-46,	TO-46*,	TO-92,	TO-92**,	SO-8	SO-8**	TO-220
	no heat sink	small heat fin	no heat sink	small heat fin	no heat sink	small heat fin	no heat sink
Still air	400°C/W	100°C/W	180°C/W	140°C/W	220°C/W	110°C/W	90°C/W
Moving air	100°C/W	40°C/W	90°C/W	70°C/W	105°C/W	90°C/W	26°C/W
Still oil	100°C/W	40°C/W	90°C/W	70°C/W			
Stirred oil	50°C/W	30°C/W	45°C/W	40°C/W			
(Clamped to metal,							
Infinite heat sink)	(2	4°C/W)			(5	5°C/W)	

^{*}Wakefield type 201, or 1" disc of 0.020" sheet brass, soldered to case, or similar.

^{**}TO-92 and SO-8 packages glued and leads soldered to 1" square of 1/16" printed circuit board with 2 oz. foil or similar.

Typical Applications



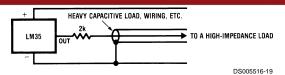


FIGURE 3. LM35 with Decoupling from Capacitive Load

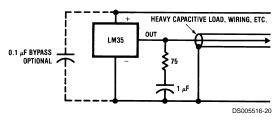


FIGURE 4. LM35 with R-C Damper

CAPACITIVE LOADS

Like most micropower circuits, the LM35 has a limited ability to drive heavy capacitive loads. The LM35 by itself is able to drive 50 pf without special precautions. If heavier loads are anticipated, it is easy to isolate or decouple the load with a resistor; see *Figure 3*. Or you can improve the tolerance of capacitance with a series R-C damper from output to ground; see *Figure 4*.

When the LM35 is applied with a 200Ω load resistor as shown in Figure 5, Figure 6 or Figure 8 it is relatively immune to wiring capacitance because the capacitance forms a bypass from ground to input, not on the output. However, as with any linear circuit connected to wires in a hostile environment, its performance can be affected adversely by intense electromagnetic sources such as relays, radio transmitters, motors with arcing brushes, SCR transients, etc, as its wiring can act as a receiving antenna and its internal junctions can act as rectifiers. For best results in such cases, a bypass capacitor from $V_{\rm IN}$ to ground and a series R-C damper such as 75Ω in series with 0.2 or 1 µF from output to ground are often useful. These are shown in Figure 13, Figure 14, and Figure 16.

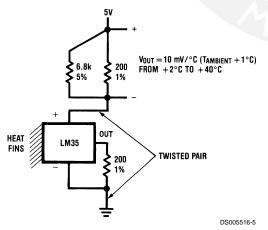


FIGURE 5. Two-Wire Remote Temperature Sensor (Grounded Sensor)

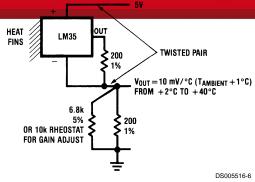


FIGURE 6. Two-Wire Remote Temperature Sensor (Output Referred to Ground)

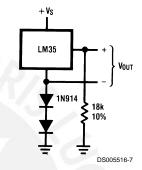


FIGURE 7. Temperature Sensor, Single Supply, -55° to +150°C

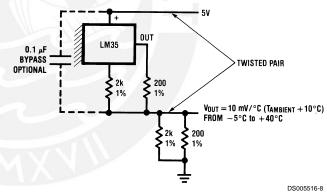


FIGURE 8. Two-Wire Remote Temperature Sensor (Output Referred to Ground)

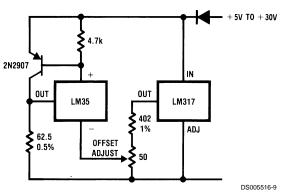


FIGURE 9. 4-To-20 mA Current Source (0°C to +100°C)

Typical Applications (Continued)



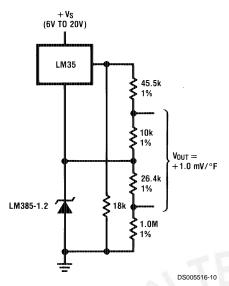


FIGURE 10. Fahrenheit Thermometer

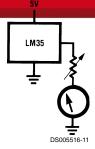


FIGURE 11. Centigrade Thermometer (Analog Meter)

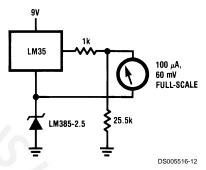


FIGURE 12. Fahrenheit ThermometerExpanded Scale
Thermometer
(50° to 80° Fahrenheit, for Example Shown)

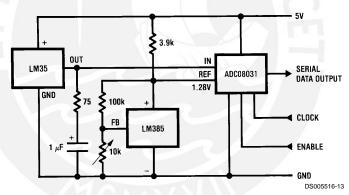


FIGURE 13. Temperature To Digital Converter (Serial Output) (+128°C Full Scale)

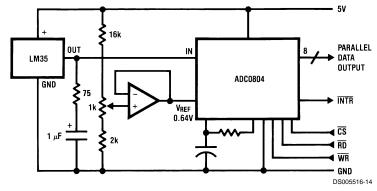
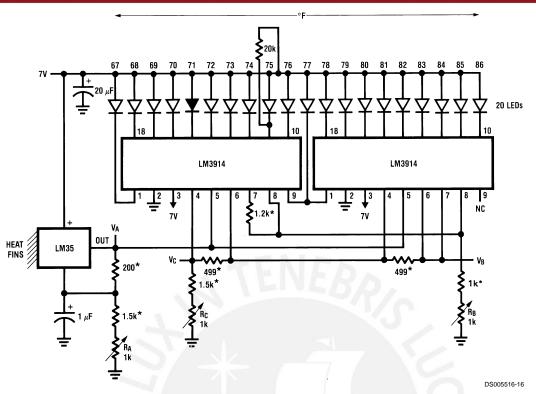


FIGURE 14. Temperature To Digital Converter (Parallel TRI-STATE™ Outputs for Standard Data Bus to µP Interface) (128°C Full Scale)

www.national.com 8







*=1% or 2% film resistor Trim R_B for V_B=3.075V Trim R_C for V_C=1.955V Trim R_A for V_A=0.075V + 100mV/ $^{\circ}$ C x T_{ambient} Example, V_A=2.275V at 22 $^{\circ}$ C

FIGURE 15. Bar-Graph Temperature Display (Dot Mode)

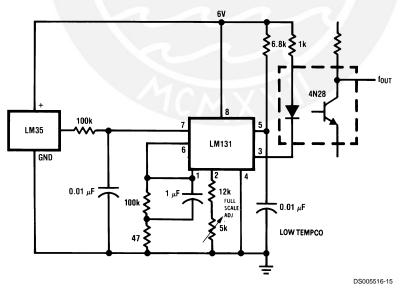
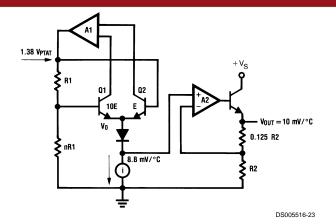


FIGURE 16. LM35 With Voltage-To-Frequency Converter And Isolated Output (2°C to +150°C; 20 Hz to 1500 Hz)

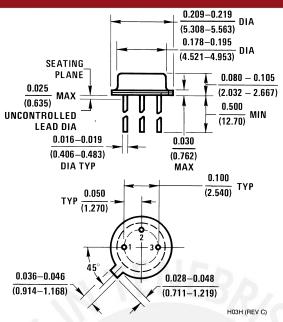
Block Diagram



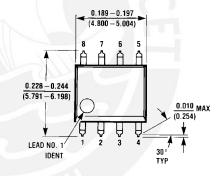


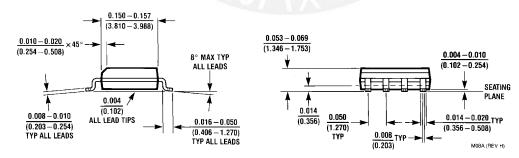






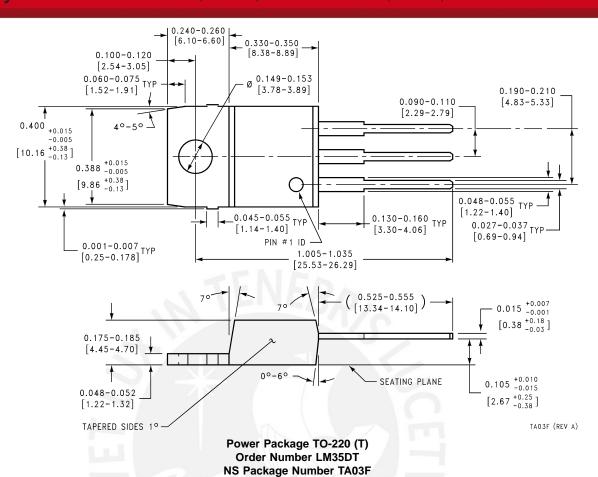
TO-46 Metal Can Package (H)
Order Number LM35H, LM35AH, LM35CH,
LM35CAH, or LM35DH
NS Package Number H03H

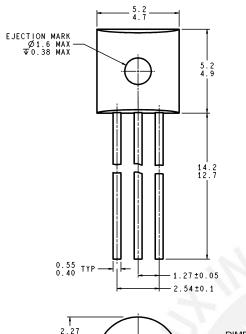


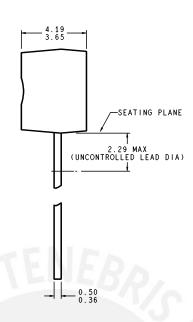


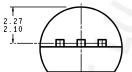
SO-8 Molded Small Outline Package (M) Order Number LM35DM NS Package Number M08A

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)









DIMENSIONS ARE IN MILLIMETERS

ZO3A (Rev G)

TO-92 Plastic Package (Z) Order Number LM35CZ, LM35CAZ or LM35DZ NS Package Number Z03A

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

- 1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
- 2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.



National Semiconductor Corporation

Americas Tel: 1-800-272-9959 Fax: 1-800-737-7018 Email: support@nsc.com

www.national.com

National Semiconductor Europe

Fax: +49 (0) 180-530 85 86 Email: europe.support@nsc.com Deutsch Tel: +49 (0) 69 9508 6208 English Tel: +44 (0) 870 24 0 2171

Français Tel: +33 (0) 1 41 91 8790

National Semiconductor Asia Pacific Customer Response Group Tel: 65-2544466

Fax: 65-2504466 Email: ap.support@nsc.com **National Semiconductor**

Tel: 81-3-5639-7560 Fax: 81-3-5639-7507





NOT RECOMMENDED FOR NEW DESIGNS I KECUNINIENUEU FUK NEW DESIGI POSSIBLE SUBSTITUTE PRODUCT

IRF540, RF1S540SM

January 2002

28A, 100V, 0.077 Ohm, N-Channel Power **MOSFETS**

These are N-Channel enhancement mode silicon gate power field effect transistors. They are advanced power MOSFETs designed, tested, and guaranteed to withstand a specified level of energy in the breakdown avalanche mode of operation. All of these power MOSFETs are designed for applications such as switching regulators, switching convertors, motor drivers, relay drivers, and drivers for high power bipolar switching transistors requiring high speed and low gate drive power. These types can be operated directly from integrated circuits.

Formerly developmental type TA17421.

Ordering Information

PART NUMBER	PACKAGE	BRAND
IRF540	TO-220AB	IRF540
RF1S540SM	TO-263AB	RF1S540SM

NOTE: When ordering, use the entire part number. Add the suffix 9A to obtain the TO-263AB variant in the tape and reel, i.e., RF1S540SM9A.

Features

- · 28A, 100V
- $r_{DS(ON)} = 0.077$
- · Single Pulse Avalanche Energy Rated
- · Nanosecond Switching Speeds
- · Linear Transfer Characteristics
- · High Input Impedance
- · Related Literature
 - TB334 "Guidelines for Soldering Surface Mount Components to PC Boards"

Symbol



Packaging

JEDEC TO-220AB



JEDEC TO-263AB



©2002 Fairchild Semiconductor Corporation



Absolute Maximum Ratings $T_C = 25^{\circ}C$, Unless Otherwise Specified

3	IRF540, RF1S540SM	UNITS
Drain to Source Breakdown Voltage (Note 1)	100	V
Drain to Gate Voltage (R _{GS} = 20k) (Note 1)	100	V
Continuous Drain Current	28	Α
$T_{C} = 100^{\circ}C$	20	Α
Pulsed Drain Current (Note 3)	110	Α
Gate to Source Voltage	±20	V
Maximum Power Dissipation	120	W
Dissipation Derating Factor	0.8	W/oC
Single Pulse Avalanche Energy Rating (Note 4)	230	mJ
Operating and Storage Temperature	-55 to 175	°C
Maximum Temperature for Soldering		
Leads at 0.063in (1.6mm) from Case for 10s	300	°C
Package Body for 10s, See Techbrief 334	260	°C

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

NOTE:

1. $T_J = 25^{\circ}C$ to $T_J = 150^{\circ}C$.

Electrical Specifications T_C = 25^oC, Unless Otherwise Specified

PARAMETER	SYMBOL	TEST CO	NDITIONS	MIN	TYP	MAX	UNITS
Drain to Source Breakdown Voltage	BV _{DSS}	I _D = 250μA, V _{GS} = 0V (Fig	ure 10)	100	-	12	V
Gate to Threshold Voltage	V _{GS(TH)}	$V_{GS} = V_{DS}, I_{D} = 250 \mu A$		2	-	4	V
Zero Gate Voltage Drain Current	I _{DSS}	V _{DS} = 95V, V _{GS} = 0V		-	-	25	μA
	339930000	V _{DS} = 0.8 x Rated BV _{DSS}	$V_{GS} = 0V, T_{J} = 150^{\circ}C$	-	-	250	μA
On-State Drain Current (Note 2)	I _{D(ON)}	V _{DS} > I _{D(ON)} x r _{DS(ON)} MA	AX, V _{GS} = 10V (Figure 7)	28	-	3.5	Α
Gate to Source Leakage Current	I _{GSS}	V _{GS} = ±20V			(5)	±100	nA
Drain to Source On Resistance (Note 2)	r _{DS(ON)}	I _D = 17A, V _{GS} = 10V (Figures 8, 9)			0.060	0.077	
Forward Transconductance (Note 2)	9fs	V _{DS} 50V, I _D = 17A (Figu	re 12)	8.7	13	0.50	S
Turn-On Delay Time	t _{d(ON)}	V _{DD} = 50V, I _D 28A, R _G 9.1 , R _L = 1.7 MOSFET Switching Times are Essentially Independent of Operating			15	23	ns
Rise Time	t _r				70	110	ns
Turn-Off Delay Time	t _d (OFF)	Temperature	-	40	60	ns	
Fall Time	t _f		-	50	83	ns	
Total Gate Charge (Gate to Source + Gate to Drain)	Q _{g(TOT)}	V_{GS} = 10V, I_{D} = 28A, V_{DS} = 0.8 x Rated BV _{DSS} , $I_{g(REF)}$ = 1.5mA (Figure 14) Gate Charge is Essentially			38	59	nC
Gate to Source Charge	Q _{qs}	Independent of Operating	-	8		nC	
Gate to Drain "Miller" Charge	Q _{gd}			-	21		nC
Input Capacitance	C _{ISS}	V _{DS} = 25V, V _{GS} = 0V, f =	1MHz	-	1450	-	pF
Output Capacitance	Coss	(Figure 11)		-	550	-	pF
Reverse Transfer Capacitance	C _{RSS}	-		-	100	7/40	pF
Internal Drain Inductance	L _D	Measured From the Contact Screw on Tab To Center of Die	Modified MOSFET Symbol Showing the Internal Devices Inductances	-	3.5	-	nH
		Measured From the Drain Lead, 6mm (0.25in) from Package to Center of Die	ξι _D	-	4.5	7.00	nH
Internal Source Inductance	LS	Measured From the Source Lead, 6mm (0.25in) From Header to Source Bonding Pad	Go ELS	-	7.5	-	nH
Thermal Resistance Junction to Case	R _{JC}			-	-	1.25	°C/W
Thermal Resistance Junction to Ambient	R JA	Free Air Operation		-		80	°C/W
	R _{JA}	RF1S540SM Mounted on FR-4 Board with Minimum Mounting Pad			-	62	°C/W

©2002 Fairchild Semiconductor Corporation



Source to Drain Diode Specifications

PARAMETER	SYMBOL	TEST CONDITIONS	MIN	TYP	MAX	UNITS
Continuous Source to Drain Current	I _{SD}	Modified MOSFET Symbol		-	28	А
Pulse Source to Drain Current (Note 3)	I _{SDM}	Reverse P-N Junction Diode	S S	-	110	А
Source to Drain Diode Voltage (Note 2)	V _{SD}	$T_J = 25^{\circ}$ C, $I_{SD} = 27$ A, $V_{GS} = 0$ V (Figure 1	3) -	183	2.5	V
Reverse Recovery Time	t _{rr}	$T_J = 25^{\circ}C$, $I_{SD} = 28A$, $dI_{SD}/dt = 100A/\mu s$	70	150	300	ns
Reverse Recovery Charge	Q _{RR}	$T_J = 25^{\circ}C$, $I_{SD} = 28A$, $dI_{SD}/dt = 100A/\mu s$	0.2	1.0	1.9	μC

NOTES:

- 2. Pulse test: pulse width 300µs, duty cycle 2%.
- 3. Repetitive rating: pulse width limited by maximum junction temperature. See Transient Thermal Impedance curve (Figure 3).
- 4. V_{DD} = 25V, starting T_J = 25°C, L = 440 μ H, R_G = 25 , peak I_{AS} = 28A.

Typical Performance Curves Unless Otherwise Specified

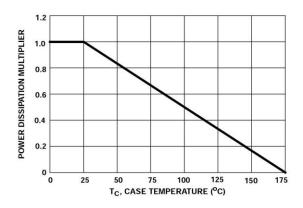


FIGURE 1. NORMALIZED POWER DISSIPATION vs CASE TEMPERATURE

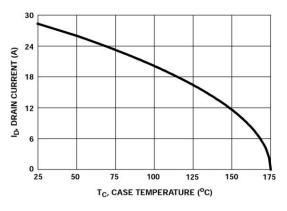


FIGURE 2. MAXIMUM CONTINUOUS DRAIN CURRENT vs CASE TEMPERATURE

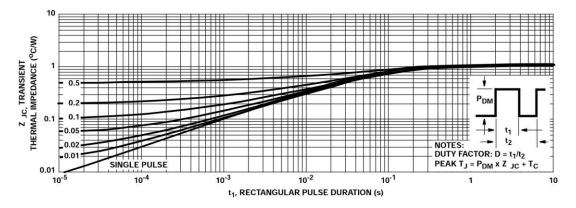


FIGURE 3. MAXIMUM TRANSIENT THERMAL IMPEDANCE

©2002 Fairchild Semiconductor Corporation



Typical Performance Curves Unless Otherwise Specified (Continued)

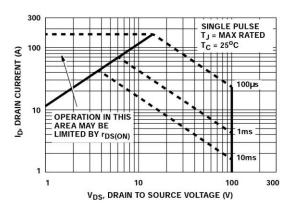


FIGURE 4. FORWARD BIAS SAFE OPERATING AREA

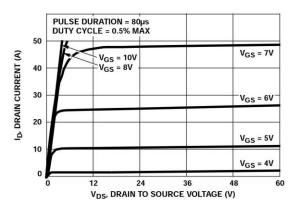


FIGURE 5. OUTPUT CHARACTERISTICS

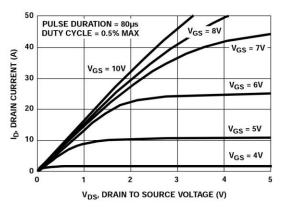


FIGURE 6. SATURATION CHARACTERISTICS

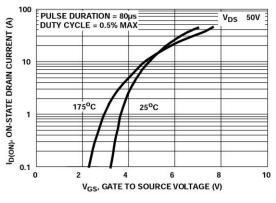


FIGURE 7. TRANSFER CHARACTERISTICS

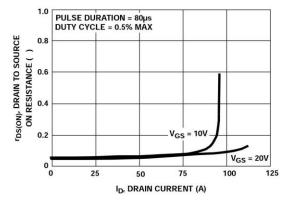


FIGURE 8. DRAIN TO SOURCE ON RESISTANCE VS GATE VOLTAGE AND DRAIN CURRENT

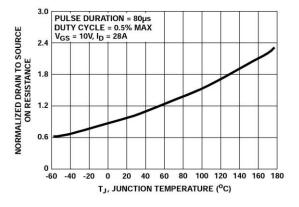


FIGURE 9. NORMALIZED DRAIN TO SOURCE ON RESISTANCE vs JUNCTION TEMPERATURE

©2002 Fairchild Semiconductor Corporation



Typical Performance Curves Unless Otherwise Specified (Continued)

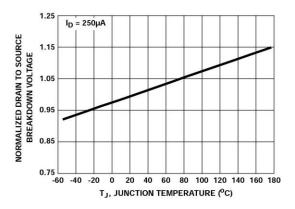


FIGURE 10. NORMALIZED DRAIN TO SOURCE BREAKDOWN VOLTAGE vs JUNCTION TEMPERATURE

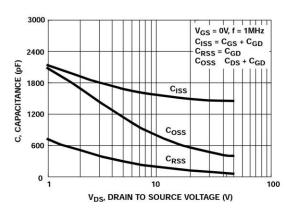


FIGURE 11. CAPACITANCE vs DRAIN TO SOURCE VOLTAGE

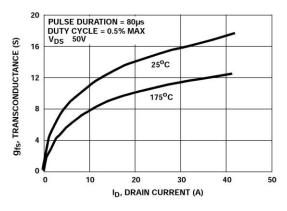


FIGURE 12. TRANSCONDUCTANCE vs DRAIN CURRENT

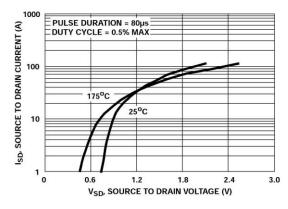


FIGURE 13. SOURCE TO DRAIN DIODE VOLTAGE

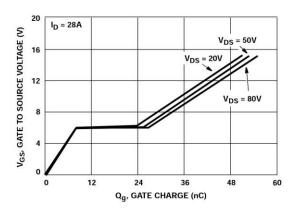


FIGURE 14. GATE TO SOURCE VOLTAGE vs GATE CHARGE

©2002 Fairchild Semiconductor Corporation



Test Circuits and Waveforms

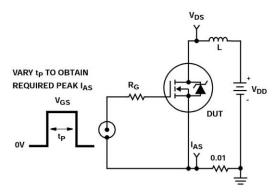


FIGURE 15. UNCLAMPED ENERGY TEST CIRCUIT

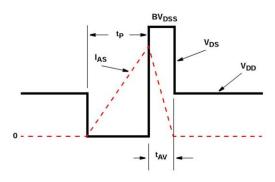


FIGURE 16. UNCLAMPED ENERGY WAVEFORMS

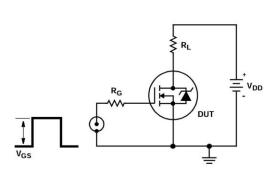


FIGURE 17. SWITCHING TIME TEST CIRCUIT

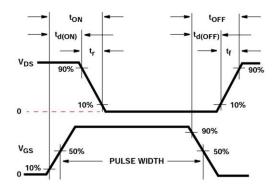


FIGURE 18. RESISTIVE SWITCHING WAVEFORMS

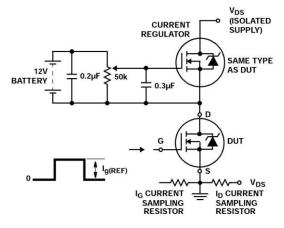


FIGURE 19. GATE CHARGE TEST CIRCUIT

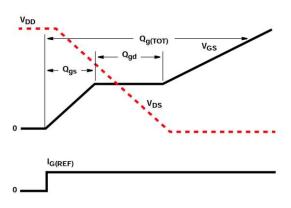


FIGURE 20. GATE CHARGE WAVEFORMS

©2002 Fairchild Semiconductor Corporation





LB9051

Switching Type Hall IC

Overview

The LB9051 is a Hall IC that is operated in the presence of an alternating magnetic field and produces a digital output. The LB9051 contains a silicon Hall generator, an amplifier, a Schmitt trigger circuit on chip and especially suited for detection of magnetism (ex. detection of the rotation of a small magnet-used substance).

Applications

- Detection of magnetism.
- · Contactless switch.
- Detection of the rotation, position of a magnetic substance.

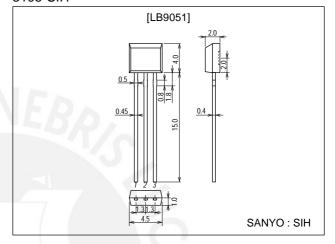
Features

- Operated in the presence of an alternating magnetic field.
- Wide operating voltage range (3.6 to 16V).
- Output capable of direct driving a TTL, MOS IC.
- High sensitivity (sensitive to low magnetism).

Package Dimensions

unit:mm

3105-SIH



Specifications

Absolute Maximum Ratings at Ta = 25°C

Parameter	Symbol	Conditions	Ratings	Unit
Maximum supply voltage	V _{CC} max		18	V
Maximum supply current	I _{CC} max		8	mA
Maximum output current	I _O max		20	mA
Allowable power dissipation	Pd max	Ta=80°C	100	mW
Operating temperature	Topr		-40 to +85	°C
Storage temperature	Tstg		-55 to +125	°C

Electrical Characteristics at Ta = 25°C

Parameter	Symbol	Conditions		Ratings			
Falametei	Symbol			typ	max	Unit	
Release point	B _{LH}	V_{CC} =12V, $V_{O}: L \rightarrow H$	-300			Gauss	
Operate point	B _{HL}	V_{CC} =12V, V_{O} : H \rightarrow L			300	Gauss	
Output low-level voltage	V _{OL1}	V _{CC} =16V, I _O =12mA, B=300Gauss			0.4	V	
Output low-level voltage	V _{OL2}	V _{CC} =3.6V, I _O =12mA, B=300Gauss			0.4	V	
Output high-level voltage	V _{OH1}	V _{CC} =16V, I _O =-30μA, B=-300Gauss	14.6			V	
Output High-level voltage	V _{OH2}	V _{CC} =3.6V, I _O =-30μA, B=-300Gauss	2.2			V	

Continued on next page.

- Any and all SANYO products described or contained herein do not have specifications that can handle applications that require extremely high levels of reliability, such as life-support systems, aircraft's control systems, or other applications whose failure can be reasonably expected to result in serious physical and/or material damage. Consult with your SANYO representative nearest you before using any SANYO products described or contained herein in such applications.
- SANYO assumes no responsibility for equipment failures that result from using products at values that exceed, even momentarily, rated values (such as maximum ratings, operating condition ranges,or other parameters) listed in products specifications of any and all SANYO products described or contained herein.

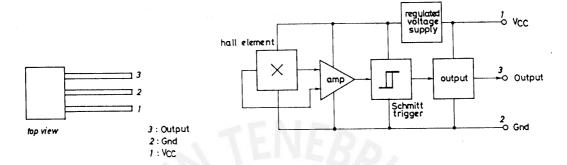
LB9051



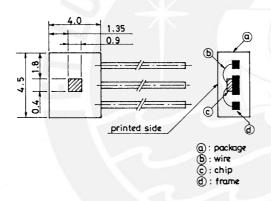
Continued from preceding page.

Parameter	Symbol	Conditions	Ratings			Unit
r alametei			min	typ	max	Offic
Output short current	-los	V _{CC} =16V, V _O =0V, B=-300Gauss	0.4		0.9	mA
Supply current	I _{CC1}	V _{CC} =16V			6	mA
Supply current	I _{CC2}	V _{CC} =3.6V			5.5	mA

Pin Assignment and Block Diagram

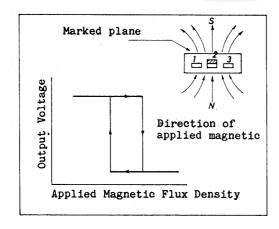


Location of the Hall Generator and Cross-sectional View of the Hall IC



The Hall generator is located in the dashed area.

Magnetic Flux to Electric Voltage Transduce Characteristic







- Specifications of any and all SANYO products described or contained herein stipulate the performance, characteristics, and functions of the described products in the independent state, and are not guarantees of the performance, characteristics, and functions of the described products as mounted in the customer's products or equipment. To verify symptoms and states that cannot be evaluated in an independent device, the customer should always evaluate and test devices mounted in the customer's products or equipment.
- SANYO Electric Co., Ltd. strives to supply high-quality high-reliability products. However, any and all semiconductor products fail with some probability. It is possible that these probabilistic failures could give rise to accidents or events that could endanger human lives, that could give rise to smoke or fire, or that could cause damage to other property. When designing equipment, adopt safety measures so that these kinds of accidents or events cannot occur. Such measures include but are not limited to protective circuits and error prevention circuits for safe design, redundant design, and structural design.
- In the event that any or all SANYO products(including technical data, services) described or contained herein are controlled under any of applicable local export control laws and regulations, such products must not be exported without obtaining the export license from the authorities concerned in accordance with the above law.
- No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or any information storage or retrieval system, or otherwise, without the prior written permission of SANYO Electric Co., Ltd.
- Any and all information described or contained herein are subject to change without notice due to product/technology improvement, etc. When designing equipment, refer to the "Delivery Specification" for the SANYO product that you intend to use.
- Information (including circuit diagrams and circuit parameters) herein is for example only; it is not guaranteed for volume production. SANYO believes information herein is accurate and reliable, but no guarantees are made or implied regarding its use or any infringements of intellectual property rights or other rights of third parties.

This catalog provides information as of February, 2001. Specifications and information herein are subject to change without notice.



This datasheet has been download from:

 $\underline{www.datasheet catalog.com}$

Datasheets for electronics components.





Tiva[™] C Series TM4C123G LaunchPad Evaluation Board

User's Guide



Literature Number: SPMU296 April 2013





Contents

1	Boar	d Overview4
	1.1	Kit Contents5
	1.2	Using the Tiva C Series LaunchPad5
	1.3	Features
	1.4	BoosterPacks
	1.5	Specifications6
2	Hard	lware Description
	2.1	Functional Description
		2.1.1 Microcontroller
		2.1.2 USB Connectivity
		2.1.3 Motion Control
		2.1.4 User Switches and RGB User LED
		2.1.5 Headers and BoosterPacks9
	2.2	Power Management11
		2.2.1 Power Supplies
		2.2.2 Hibernate
		2.2.3 Clocking
		2.2.4 Reset
	2.3	In-Circuit Debug Interface (ICDI)
		2.3.1 Virtual COM Port
3	Soft	ware Development13
	3.1	Software Description
	3.2	Source Code
	3.3	Tool Options
	3.4	Programming the Tiva C Series LaunchPad Evaluation Board
4	Refe	rences, PCB Layout, and Bill of Materials15
	4.1	References
	4.2	Component Locations
	4.3	Bill of Materials (BOM)
Δ	Sche	ematics 19



www.ti.com



List of Figures

1-1.	Tiva C Series TM4C123G LaunchPad Evaluation Board	4
2-1.	Tiva C Series LaunchPad Evaluation Board Block Diagram	7
4-1.	Tiva C Series LaunchPad Component Locations (Top View)	
4-2.	Tiva C Series LaunchPad Dimensions	17
	List of Tables	
1-1.	EK-TM4C123GXL Specifications	6
2-1.	USB Device Signals	8
2-2.	User Switches and RGB LED Signals	9
2-3.	J1 Connector	9
2-4.	J2 Connector	10
2-5.	J3 Connector	10
2-6.	J4 Connector	11
2-7.	In-Circuit Debug Interface (ICDI) Signals	12
2-8.	Virtual COM Port Signals	
4-1.	EK-TM4C123GXL Bill of Materials	17





Chapter 1
SPMU296-April 2013

Board Overview

The Tiva™ C Series TM4C123G LaunchPad Evaluation Board (EK-TM4C123GXL) is a low-cost evaluation platform for ARM® Cortex™-M4F-based microcontrollers. The Tiva C Series LaunchPad design highlights the TM4C123GH6PMI microcontroller USB 2.0 device interface, hibernation module, and motion control pulse-width modulator (MC PWM) module. The Tiva C Series LaunchPad also features programmable user buttons and an RGB LED for custom applications. The stackable headers of the Tiva C Series TM4C123G LaunchPad BoosterPack XL interface demonstrate how easy it is to expand the functionality of the Tiva C Series LaunchPad when interfacing to other peripherals on many existing BoosterPack add-on boards as well as future products. Figure 1-1 shows a photo of the Tiva C Series LaunchPad.

Power Select **USB** Connector Green Power LED Switch (Power/ICDI) TM4C123GH6PMI Microcontroller USB Micro-A/-B Reset Switch Connector (Device) **RGB User LED** Tiva C Series LaunchPad BoosterPack XL Interface (J1, J2, J3, Tiva C Series and J4 Connectors) LaunchPad BoosterPack XL Interface (J1, J2, J3, TM4C123GH6PMI and J4 Connectors) Microcontroller MSP430 EXAS INSTRUMENT LaunchPad-Compatible LaunchPad-Compatible BoosterPack Interface BoosterPack Interface Tiva™ C Series .aunchPad User Switch 1 User Switch 2

Figure 1-1. Tiva C Series TM4C123G LaunchPad Evaluation Board

Tiva, MSP430, Code Composer Studio are trademarks of Texas Instruments. Cortex is a trademark of ARM Limited.

ARM, RealView are registered trademarks of ARM Limited.

Microsoft, Windows are registered trademarks of Microsoft Corporation.

All other trademarks are the property of their respective owners.



www.ti.com Kit Contents

1.1 Kit Contents

The Tiva C Series TM4C123G LaunchPad Evaluation Kit contains the following items:

- Tiva C Series LaunchPad Evaluation Board (EK-TM4C123GXL)
- On-board In-Circuit Debug Interface (ICDI)
- USB micro-B plug to USB-A plug cable
- README First document

1.2 Using the Tiva C Series LaunchPad

The recommended steps for using the Tiva C Series TM4C123G LaunchPad Evaluation Kit are:

- 1. **Follow the README First document included in the kit.** The README First document will help you get the Tiva C Series LaunchPad up and running in minutes. See the <u>Tiva C Series LaunchPad web page</u> for additional information to help you get started.
- 2. **Experiment with LaunchPad BoosterPacks.** A selection of Tiva C Series BoosterPacks and compatible MSP430™ BoosterPacks can be found at the TI MCU LaunchPad web page.
- 3. Take your first step toward developing an application with Project 0 using your preferred ARM tool-chain and the Tiva C Series TivaWare Peripheral Driver Library. Software applications are loaded using the on-board In-Circuit Debug Interface (ICDI). See Chapter 3, Software Development, for the programming procedure. The TivaWare for C Series Peripheral Driver Library Software Reference Manual contains specific information on software structure and function. For more information on Project 0, go to the Tiva C Series LaunchPad wiki page.
- 4. **Customize and integrate the hardware to suit an end application.** This user's manual is an important reference for understanding circuit operation and completing hardware modification.

You can also view and download almost six hours of training material on configuring and using the LaunchPad. Visit the Tiva C Series LaunchPad Workshop for more information and tutorials.

1.3 Features

Your Tiva C Series LaunchPad includes the following features:

- Tiva TM4C123GH6PMI microcontroller
- Motion control PWM
- USB micro-A and micro-B connector for USB device, host, and on-the-go (OTG) connectivity
- RGB user LED
- Two user switches (application/wake)
- Available I/O brought out to headers on a 0.1-in (2.54-mm) grid
- On-board ICDI
- Switch-selectable power sources:
 - ICDI
 - USB device
- Reset switch
- Preloaded RGB quickstart application
- Supported by TivaWare for C Series software including the USB library and the peripheral driver library
- Tiva C Series TM4C123G LaunchPad BoosterPack XL Interface, which features stackable headers to expand the capabilities of the Tiva C Series LaunchPad development platform
 - For a complete list of available BoosterPacks that can be used with the Tiva C Series LaunchPad, see the <u>LaunchPad web page</u>.

BoosterPacks www.ti.com

1.4 BoosterPacks

The Tiva C Series LaunchPad provides an easy and inexpensive way to develop applications with the TM4C123GH6PM microcontroller. Tiva C Series BoosterPacks and MSP430 BoosterPacks expand the available peripherals and potential applications of the Tiva C Series LaunchPad. BoosterPacks can be used with the Tiva C Series LaunchPad or you can simply use the on-board TM4C123GH6PM microcontroller as its processor. See Chapter 2 for more information.

Build your own BoosterPack and take advantage of <u>Texas Instruments' website</u> to help promote it! From sharing a new idea or project, to designing, manufacturing, and selling your own BoosterPack kit, TI offers a variety of avenues for you to reach potential customers with your solutions.

1.5 Specifications

Table 1-1 summarizes the specifications for the Tiva C Series LaunchPad.

Table 1-1. EK-TM4C123GXL Specifications

Parameter	Value
Board supply voltage	 4.75 V_{DC} to 5.25 V_{DC} from one of the following sources: Debugger (ICDI) USB Micro-B cable (connected to a PC) USB Device Micro-B cable (connected to a PC)
Dimensions	2.0 in x 2.25 in x 0.425 in (5.0 cm x 5.715 cm x 10.795 mm) (L x W x H)
Break-out power output	 3.3 V_{DC} (300 mA max) 5.0 V_{DC} (depends on 3.3 V_{DC} usage, 23 mA to 323 mA)
RoHS status	Compliant





Chapter 2 SPMU296-April 2013

Hardware Description

The Tiva C Series LaunchPad includes a TM4C123GH6PM microcontroller and an integrated ICDI as well as a range of useful peripheral features (as the block diagram in Figure 2-1 shows). This chapter describes how these peripherals operate and interface to the microcontroller.

Debug Breakout Pads 00000 JTAG/SWD ICDI UART0 **GPIO** USB Debug TM4C123GH6PMI Connector **GPIO USB** Device USB Connector Device **GPIO** Power Select **RGB LED** Switch VDD HIB WAKE **GPIO** User Switches Power Management **Breakout Pads**

Figure 2-1. Tiva C Series LaunchPad Evaluation Board Block Diagram

2.1 Functional Description

2.1.1 Microcontroller

The TM4C123GH6PM is a 32-bit ARM Cortex-M4-based microcontroller with 256-kB Flash memory, 32-kB SRAM, and 80-MHz operation; USB host, device, and OTG connectivity; a Hibernation module and PWM; and a wide range of other peripherals. See the TM4C123GH6PM microcontroller data sheet (literature number SPMS376) for complete device details.

Functional Description www.ti.com

Most of the microcontroller signals are routed to 0.1-in (2.54-mm) pitch headers. An internal multiplexer allows different peripheral functions to be assigned to each of these GPIO pads. When adding external circuitry, consider the additional load on the evaluation board power rails.

The TM4C123GH6PM microcontroller is factory-programmed with a quickstart demo program. The quickstart program resides in on-chip Flash memory and runs each time power is applied, unless the quickstart application has been replaced with a user program.

2.1.2 USB Connectivity

The EK-TM4C123GXL is designed and functions as a USB device without hardware modification. The USB device signals are dedicated to USB functionality and are not shared with the BoosterPack headers. The USB device signals are listed in Table 2-1.

Table 2-1. USB Device Signals

GPIO Pin	Pin Function	USB Device				
PD4	USB0DM	D-				
PD5	USB0DP	D+				

The TM4C123GH6PM target device is also capable of USB embedded host and on-the-go (OTG) functions. OTG functionality can be enabled by populating R25 and R29 with $0-\Omega$ resistors. These resistors connect the USB ID and USB V_{BUS} signals to PB0 and PB1. When these resistors are populated, PB0 and PB1 must remain in the respective USB pin mode configurations to prevent device damage. PB0 and PB1 are also present on the J1 BoosterPack header. Therefore, if R25 or R29 are populated, care must be taken not to conflict these signals with BoosterPack signals.

USB embedded host operation can be enabled in the same way for USB devices that are self-powered. Providing power when acting as a USB host requires a BoosterPack with power switching and appropriate connectors. All USB host signals are available on the BoosterPack interface except D+ and D-, which are only available on the USB micro-A/-B connector and the two adjacent test points.

When connected as a USB device, the evaluation board can be powered from either the ICDI or the USB Device connectors. The user can select the power source by moving the POWER SELECT switch (SW3) to the Device position. See the *Power Management* schematic (appended to this document).

2.1.3 Motion Control

The EK-TM4C123GXL includes the Tiva C-Series Motion Control PWM technology, featuring two PWM modules capable of generating 16 PWM outputs. Each PWM module provides a great deal of flexibility and can generate simple PWM signals—for example, those required by a simple charge pump—as well as paired PWM signals with dead-band delays, such as those required by a half-H bridge driver. Three generator blocks can also generate the full six channels of gate controls required by a 3-phase inverter bridge.

Two quadrature encoder interfaces (QEI) are also available to provide motion control feedback. See the Headers and BoosterPacks section of this document for details about the availability of these signals on the BoosterPack interface.

Functional Description

2.1.4 User Switches and RGB User LED

The Tiva C Series LaunchPad comes with an RGB LED. This LED is used in the preloaded RGB quickstart application and can be configured for use in custom applications.

Two user buttons are included on the board. The user buttons are both used in the preloaded quickstart application to adjust the light spectrum of the RGB LED as well as go into and out of hibernation. The user buttons can be used for other purposes in the user's custom application.

The evaluation board also has a green power LED. Table 2-2 shows how these features are connected to the pins on the microcontroller.

GPIO Pin Pin Function **USB Device** PF4 **GPIO** SW1 **GPIO** PF0 SW₂ **GPIO** PF1 RGB LED (Red) PF2 **GPIO** RGB LED (Blue) PF3 **GPIO** RGD LED (Green)

Table 2-2. User Switches and RGB LED Signals

2.1.5 Headers and BoosterPacks

The two double rows of stackable headers are mapped to most of the GPIO pins of the TM4C123GH6PM microcontroller. These rows are labeled as connectors J1, J2, J3, and J4. Connectors J3 and J4 are located 0.1 in (2.54 mm) inside of the J1 and J2 connectors. All 40 header pins of the J1, J2, J3, and J4 connectors make up the Tiva C Series TM4C123G LaunchPad BoosterPack XL Interface. Table 2-3 through Table 2-6 show how these header pins are connected to the microcontroller pins and which GPIO functions can be selected.

NOTE: To configure the device peripherals easily and intuitively using a graphical user interface (GUI), see the Tiva C Series Pinmux Utility found at www.ti.com/tool/lm4f_pinmux. This easyto-use interface makes setting up alternate functions for GPIOs simple and error-free.

Analog Function Tiva C **GPIOPCTL** Register Setting On-J4 Pin GPIO board MCU GPIO Function 1 2 3 4 5 6 7 8 9 14 15 AMSEL 3.3 V 1.01 T1CCP1 1 02 PB5 AIN11 57 SSI2Fss MOPWM3 CAN0Tx 1.03 PB0 USB0ID 45 U1Rx T2CCP0 USB0VBUS U1Tx T2CCP1 1.04 PB1 46 PE4 AIN9 U5Rx I2C2SCL M0PWM4 M1PWM2 CAN0Rx 1.05 59 _ I2C2SDA M0PWM5 M1PWM3 CAN0Tx 1.06 PE5 AIN8 60 U5Tx CAN0Rx 1.07 PB4 AIN10 58 SSI2CIk M0PWM2 T1CCP0 SSI0Tx 1.08 PA5 22 1.09 PA6 23 I2C1SCL M1PWM2 _ M1PWM3 1.10 PA7 24 I2C1SDA

Table 2-3, J1 Connector(1)

Shaded cells indicate configuration for compatibility with the MSP430 LaunchPad.

Functional Description www.ti.com

Table 2-4. J2 Connector⁽¹⁾

J4	GPIO	Analog Function	On-board	Tiva C Series				GPIOPCTL Register Setting									
Pin	GPIO	GPIO AMSEL	Function	MCU Pin	1	2	3	4	5	6	7	8	9	14	15		
2.01							GND					<u> </u>	<u> </u>				
2.02	PB2	-	-	47	-	-	I2C0SCL	-	-	-	T3CCP0	_	-	-	-		
2.03	PE0	AIN3	-	9	U7Rx	-	-	-	-	-	-	-	-	-	-		
2.04	PF0	-	USR_SW2/ WAKE (R1)	28	U1RTS	SSI1Rx	CAN0Rx	-	M1PWM4	PhA0	T0CCP0	NMI	C0o	_	_		
2.05							RESET	Ī									
	PB7	-	-	4	-	SSI2Tx	_	M0PWM1	-	-	T0CCP1	-	_	_	-		
2.06	PD1	AIN6	Connected for MSP430 Compatibility (R10)	62	SSI3Fss	SSI1Fss	I2C3SDA	M0PWM7	M1PWM1	-	WT2CCP1	-	-	-	_		
	PB6	-	-	1	ı	SSI2Rx	-	M0PWM0	-	-	T0CCP0	-	-	-	-		
2.07	PD0	AIN7	Connected for MSP430 Compatibility (R9)	61	SSI3Clk	SSI1Clk	I2C3SCL	MOPWM6	M1PWM0	-	WT2CCP0	-	-	-	-		
2.08	PA4	-	-	21	1 - 1	SSI0Rx	U. I J. A			-	-	_	_	_	-		
2.09	PA3	-	-	20	-	SSI0Fss	-	T441.		-	-	-	-	-	-		
2.10	PA2	-	- 1	19	-	SSI0Clk	-		- (L)	_	-	-	-	-	-		

⁽¹⁾ Shaded cells indicate configuration for compatibility with the MSP430 LaunchPad.

Table 2-5. J3 Connector⁽¹⁾

J4	GPIO	Analog Function	On-board	Tiva C Series				GPIC	PCTL Regist	er Sett	ing				
Pin	GFIO	GPIO AMSEL	Function	MCU Pin	1	2	3	4	5	6	7	8	9	14	15
3.01							5.0) V							
3.02							GI	ND							
	PD0	AIN7	\\ -	61	SSI3CIk	SSI1Clk	I2C3SCL	M0PWM6	M1PWM0	_	WT2CCP0	-	_	_	-
3.03	PB6	-	Connected for MSP430 Compatibilit y (R9)	1	2	SSI2Rx	_	M0PWM0	3/	-	T0CCP0	-	ı	-	-
	PD1	AIN6	- \	92	SSI3Fss	SSI1Fss	I2C3SDA	M0PWM7	M1PWM1	-	WT2CCP1	-	_	-	-
3.04	PB7	-	Connected for MSP430 Compatibilit y (R10)	4		SSI2Tx	-	M0PWM1	-	/-	T0CCP1	-	-	-	-
3.05	PD2	AIN5		63	SSI3Rx	SSI1Rx	$\Lambda^{-}X$	M0FAULT0	-	-	WT3CCP0	USB0EPE N			
3.06	PD3	AIN4	-	64	SSI3Tx	SSI1Tx	<u>_</u>	-	=	_	WT3CCP1	USB0PFLT	_	-	-
3.07	PE1	AIN2	-	8	U7Tx	_	-	-	-	-		-	-	-	-
3.08	PE2	AIN1	-	7	1	-	-	-	-	_	-	-	-	-	-
3.09	PE3	AIN0	-	6	1	-	-	-	-	_	-	-	-	-	-
3.10	PF1	-	-	29	U1CTS	SSI1Tx	-	ı	M1PWM5	-	T0CCP1	-	C1o	TRD1	-

 $^{^{\}rm (1)}$ Shaded cells indicate configuration for compatibility with the MSP430 LaunchPad.



www.ti.com Power Manageme.



J4	GPIO	Analog Function	On- board	Tiva C Series				GI	PIOPCTL Reg	ister Setti	ng				
Pin	GFIO	GPIO AMSEL	Function	MCU Pin			3	4	5	6	7	8	9	14	15
4.01	PF2	=	Blue LED (R11)	30	-	SSI1Clk	=	M0FAULT0	M1PWM6	-	T1CCP0	=	-	_	TRD0
4.02	PF3	-	Green LED (R12)	31	-	SSI1Fss	CAN0Tx	-	M1PWM7	-	T1CCP1	-	-	_	TRCLK
4.03	PB3	-	-	48	-	-	I2C0SDA	-	-	-	T3CCP1	-	-	-	-
4.04	PC4	C1-	-	16	U4Rx	U1Rx	-	M0PWM6	-	IDX1	WT0CCP0	U1RTS	-	-	-
4.05	PC5	C1+	-	15	U4Tx	U1Tx	-	M0PWM7	-	PhA1	WT0CCP1	U1CTS	-	-	-
4.06	PC6	C0+	-	14	U3Rx	-	=	=	-	PhB1	WT1CCP0	USB0EPE N	-	_	-
4.07	PC7	C0-	-	13	U3Tx	_	-	-	-	1	WT1CCP1	USB0PFLT	-	-	_
4.08	PD6	-	-	53	U2Rx	-	-	-	-	PhA0	WT5CCP0	-	-	-	-
4.09	PD7	-	-	10	U2Tx	-	-	-	-	PhB0	WT5CCP1	NMI	-	-	-
4.10	PF4	-	USR_SW 1 (R13)	5	-	TE	N-I	Ch	M1FAULT0	IDX0	T2CCP0	USB0EPE N	-	-	-

Connectors J1 and J2 of the Tiva C Series TM4C123G LaunchPad BoosterPack XL Interface provide compatibility with MSP430 LaunchPad BoosterPacks. Highlighted functions (shaded cells) in Table 2-3 through Table 2-5 indicate configuration for compatibility with the MSP430 LaunchPad.

A complete list of Tiva C Series BoosterPacks and Tiva C Series LaunchPad-compatible MSP430 BoosterPacks is available at www.ti.com/tm4c123g-launchpad.

2.2 Power Management

2.2.1 Power Supplies

The Tiva C Series LaunchPad can be powered from one of two power sources:

- On-board ICDI USB cable (Debug, Default)
- USB device cable (Device)

The POWER SELECT switch (SW3) is used to select one of the two power sources. Select only one source at a time.

2.2.2 Hibernate

The Tiva C Series LaunchPad provides an external 32.768-kHz crystal (Y1) as the clock source for the TM4C123GH6PM Hibernation module clock source. The current draw while in Hibernate mode can be measured by making some minor adjustments to the Tiva C Series LaunchPad. This procedure is explained in more detail later in this section.

The conditions that can generate a wake signal to the Hibernate module on the Tiva C Series LaunchPad are waking on a Real-time Clock (RTC) match and/or waking on assertion of the \overline{WAKE} pin. (1) The second user switch (SW2) is connected to the \overline{WAKE} pin on the microcontroller. The \overline{WAKE} pin, as well as the V_{DD} and \overline{HIB} pins, are easily accessible through breakout pads on the Tiva C Series LaunchPad. See the appended schematics for details.

⁽¹⁾ If the board does not turn on when you connect it to a power source, the microcontroller might be in Hibernate mode (depending on the programmed application). You must satisfy one of the programmed wake conditions and connect the power to bring the microcontroller out of Hibernate mode and turn on the board.

www.ti.con

There is no external battery source on the Tiva C Series LaunchPad Hibernation module, which means the VDD3ON power control mechanism should be used. This mechanism uses internal switches to remove power from the Cortex-M4 processor as well as to most analog and digital functions while retaining I/O pin power.

To measure the Hibernation mode current or the Run mode current, the VDD jumper that connects the 3.3 V pin and the MCU_PWR pin must be removed. See the complete schematics (appended to this document) for details on these pins and component locations. An ammeter should then be placed between the 3.3 V pin and the MCU_PWR pin to measure I_{DD} (or I_{HIB_VDD3ON}). The TM4C123GH6PM microcontroller uses V_{DD} as its power source during V_{DD3ON} Hibernation mode, so I_{DD} is the Hibernation mode (VDD3ON mode) current. This measurement can also be taken during Run mode, which measures I_{DD} the microcontroller running current.

2.2.3 Clocking

The Tiva C Series LaunchPad uses a 16.0-MHz crystal (Y2) to complete the TM4C123GH6PM microcontroller main internal clock circuit. An internal PLL, configured in software, multiples this clock to higher frequencies for core and peripheral timing.

The Hibernation module is clocked from an external 32.768-KHz crystal (Y1).

2.2.4 Reset

The RESET signal into the TM4C123GH6PM microcontroller connects to the RESET switch and to the ICDI circuit for a debugger-controlled reset.

External reset is asserted (active low) under any of three conditions:

- Power-on reset (filtered by an R-C network)
- · RESET switch held down
- By the ICDI circuit when instructed by the debugger (this capability is optional, and may not be supported by all debuggers)

2.3 In-Circuit Debug Interface (ICDI)

The Tiva C Series LaunchPad evaluation board comes with an on-board In-Circuit Debug Interface (ICDI). The ICDI allows for the programming and debug of the TM4C123GH6PM using the LM Flash Programmer and/or any of the supported tool chains. Note that the ICDI supports only JTAG debugging. An external debug interface can be connected for Serial Wire Debug (SWD) and SWO (trace).

Table 2-7 shows the pins used for JTAG and SWD. These signals are also mapped out to easily accessible breakout pads and headers on the board.

Table 2-7. In-Circuit Debug Interface (ICDI) Signals

GPIO Pin	Pin Function
PC0	TCK/SWCLK
PC1	TMS/SWDIO
PC2	TDI
PC3	TDO/SWO

2.3.1 Virtual COM Port

When plugged in to a PC, the device enumerates as a debugger and a virtual COM port. Table 2-8 shows the connections for the COM port to the pins on the microcontroller.

Table 2-8. Virtual COM Port Signals

GPIO Pin	Pin Function
PA0	U0RX
PA1	U0TX





Chapter 3
SPMU296-April 2013

Software Development

This chapter provides general information on software development as well as instructions for Flash memory programming.

3.1 Software Description

The TivaWare software provided with the Tiva C Series LaunchPad provides access to all of the peripheral devices supplied in the design. The Tiva C Series Peripheral Driver Library is used to operate the on-chip peripherals as part of TivaWare.

TivaWare includes a set of example applications that use the TivaWare Peripheral Driver Library. These applications demonstrate the capabilities of the TM4C123GH6PM microcontroller, as well as provide a starting point for the development of the final application for use on the Tiva C Series LaunchPad evaluation board.

3.2 Source Code

The complete source code including the source code installation instructions are provided at www.ti.com/tm4c123g-launchpad. The source code and binary files are installed in the DriverLib tree.

3.3 Tool Options

The source code installation includes directories containing projects and/or makefiles for the following toolchains:

- Keil ARM RealView® Microcontroller Development System
- · IAR Embedded Workbench for ARM
- Sourcery CodeBench
- Texas Instruments' Code Composer Studio™ IDE

Download evaluation versions of these tools from the TI website. Due to code size restrictions, the evaluation tools may not build all example programs. A full license is necessary to re-build or debug all examples.

Instructions on installing and using each of the evaluation tools can be found in the Quickstart guides (for example, Quickstart-Keil, Quickstart-IAR) which are available for download from the evaluation kit section of the TI website at www.ti.com/tiva-c.

For detailed information on using the tools, see the documentation included in the tool chain installation or visit the respective web site of the tool supplier.

www.ti.con

3.4 Programming the Tiva C Series LaunchPad Evaluation Board

The Tiva C Series LaunchPad software package includes pre-built binaries for each of the example applications. If you have installed TivaWare to the default installation path of C:\ti\TivaWare_C_Series_<version>, you can find the example applications in C:\ti\TivaWare_C_Series_<version>\examples\boards\ek-tm4c123gxl . The on-board ICDI is used with the LM Flash Programmer tool to program applications on the Tiva C Series LaunchPad.

Follow these steps to program example applications into the Tiva C Series LaunchPad evaluation board using the ICDI:

- 1. Install LM Flash Programmer on a PC running Microsoft® Windows®.
- 2. Switch the **POWER SELECT** switch to the right for Debug mode.
- 3. Connect the USB-A cable plug to an available port on the PC and the Micro-B plug to the **Debug** USB port on the board.
- 4. Verify that the POWER LED D4 on the board is lit.
- 5. Run the LM Flash Programmer.
- 6. In the Configuration tab, use the Quick Set control to select the EK-TM4C123GXL evaluation board.
- 7. Move to the Program tab and click the **Browse** button. Navigate to the example applications directory (the default location is *C:\ti\TivaWare_C_Series_*<*version>**examples\boards\ek-tm4c123gxl*).
- 8. Each example application has its own directory. Navigate to the example directory that you want to load and then into the directory which contains the binary (*.bin) files. Select the binary file and click **Open**.
- Set the Erase Method to Erase Necessary Pages, check the Verify After Program box, and check Reset MCU After Program.

Program execution starts once the Verify process is complete.





Chapter 4
SPMU296-April 2013

References, PCB Layout, and Bill of Materials

4.1 References

In addition to this document, the following references are available for download at www.ti.com:

- Tiva C Series TM4C123GH6PM Microcontroller Data Sheet (literature number <u>SPMS376</u>).
- LM Flash Programmer tool. Available for download at www.ti.com/tool/lmflashprogrammer.
- TivaWare for C Series Driver Library. Available for download at www.ti.com/tool/sw-tm4c-drl.
- TivaWare for C Series Driver Library User's Manual (literature number SPMU298).
- TPS73633 Low-Dropout Regulator with Reverse Current Protection Data Sheet (literature number SBVS038)
- Texas Instruments' Code Composer Studio IDE website: www.ti.com/ccs

Additional support:

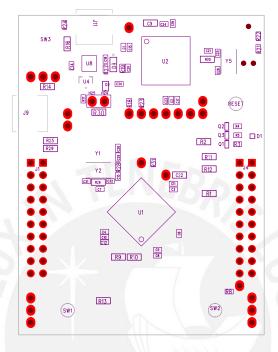
- RealView MDK (www.keil.com/arm/rvmdkkit.asp)
- IAR Embedded Workbench (www.iar.com).
- Sourcery CodeBench development tools (<u>www.codesourcery.com/gnu_toolchains/arm</u>).



4.2 Component Locations

Plots of the top-side component locations are shown in Figure 4-1 and the board dimensions are shown in Figure 4-2.

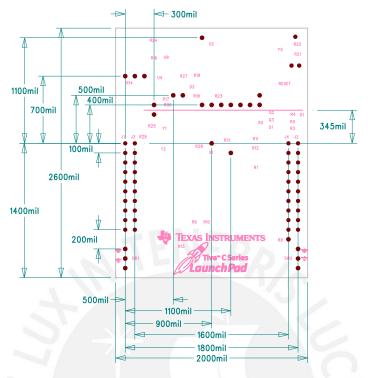
Figure 4-1. Tiva C Series LaunchPad Component Locations (Top View)





www.ti.com Bill of Materials (BOM)

Figure 4-2. Tiva C Series LaunchPad Dimensions



NOTE: Units are in mils (one thousandth of an inch): 1 mil = 0.001 inch (0.0254 mm).

4.3 Bill of Materials (BOM)

Table 4-1 shows the bill of materials for the EK-TM4C123GXL evaluation board.

Table 4-1. EK-TM4C123GXL Bill of Materials

Item	Ref Des	Qty	Description	Manufacturer	Manufacturer Part No
1	C1-2, C7, C12, C14	5	Capacitor, 0402, X5R, 10 V, Low ESR	Johanson Dielectrics Inc	100R07X105KV4T
2	C25-26, C31-32	4	Capacitor, 10 pF, 50 V, 5%, Murata GRM1555C1H100JZ NPO/COG, 0402		GRM1555C1H100JZ01D
3	C28-29	2	Capacitor, 24 pF, 50 V, 5%, NPO/COG, 0402 TDK C1005C0G1H240J		
4	C3, C5, C8, C15, C18-19, C21	7	Capacitor, 0.01 µF 25 V, 10% 0402 X7R	Taiyo Yuden	TMK105B7103KV-F
5	C4, C6, C10-11, C17, C20, C23-24	8	Capacitor, 0.1 μF 16 V, 10% 0402 X7R	Taiyo Yuden	EMK105B7104KV-F
6	C9, C22	2	Capacitor, 2.2 μF, 16 V, 10%, 0603, X5R	Murata	GRM188R61C225KE15D
7	D1	1	LED, Tri-Color RGB, 0404 SMD Common Anode	Everlight	18-038/RSGHBHC1-S02/2T
8	D4	1	LED, Green 565 nm, Clear 0805 SMD	Lite-On	LTST-C171GKT
9	H24	1	Header, 1x2, 0.100, T-Hole,	3M	961102-6404-AR
			Vertical Unshrouded, 0.220 Mate	FCI	68001-102HLF
10	H25	1	Jumper, 0.100, Gold, Black, Closed	Sullins	SPC02SYAN
11	J1, J3	2	Header, 2x10, T-Hole Vertical unshrouded stacking	Samtec	SSW-110-23-S-D



Table 4-1. EK-TM4C123GXL Bill of Materials (continued)

Item	Ref Des	Qty	Description	Manufacturer	Manufacturer Part No	
12	J11	1	USB Connector, Micro B Recept RA SMT BTTM MNT	Hirose	ZX62-B-5PA	
13	J2, J4	2	Header, 1x2, 0.100, SMT,	Samtec	TSM-110-01-S-DH-A-P-TR	
			Horizontal Unshrouded, 0.230 Mate	4UCON	10995	
			Wate	Major League Electronics	TSHSM-110-D-02-T-H-AP- TR-P-LF	
14	J9	1	USB Connector, Micro A/B Receptacle SMD	Hirose	ZX62-AB-5PA	
15	Q1-3	3	NPN SC70 pre-biased	Diodes Inc	DTC114EET1G	
16	R1-2, R9-16, R20, R26	12	Resistor, 0 Ω 1/10W 0603 SMD	Panasonic	ERJ-3GEY0R00V	
17	R18-19, R21-23, R28	6	Resistor, 10 k Ω , 1/10W, 5%, 0402 Thick Film	Yageo	RC0402FR-0710KL	
18	R3-5, R8, R27	5	Resistor, 330 Ω, 1/10W, 5%, 0402	Yageo	RC0402FR-07330RL	
19	R31	1	Resistor, 1 MΩ 1/10W, 5%, 0402	Rohm	MCR01MRTF1004	
20	RESET SW1, SW2	3	Switch, Tact 6 mm SMT, 160gf	Omron	B3S-1000	
21	SW3	1	Switch, DPDT, SMT 300 mA × 2 at 6 V	C K Components	JS202011SCQN	
22	U1, U2	2	Tiva C Series MCU TM4C123GH6PM	Texas Instruments	TM4C123GH6PMI	
23	U8	_/1/	Regulator, 3.3 V, 400 mA, LDO	Texas Instruments	TPS73633DRBT	
24	Y1	1	Crystal, 32.768 kHz Radial Can	Abracon	AB26TRB-32.768KHZ- T	
25	Y2, Y5	2	Crystal, 16.00 MHz 5.0x3.2mm	NDK	NX5032GA-16.000000 MHz	
			SMT	Abracon	ABM3-16.000 MHz-B2- T	
			PCB Do Not Populate List (Shown for information only)			
26	C31, C34	2	Capacitor, 0.1 µF 16 V, 10% 0402 X7R	Taiyo Yuden	EMK105B7104KV-F	
27	D2	1	Diode, Dual Schottky, SC70, BAS70 Common Cathode	Diodes Inc	BAS70W-05-7-F	
28	R17	1	Resistor, 10 k Ω 1/10W 5%, 0402 Thick Film	Yageo	RC0402FR-0710KL	
29	R24	1	Resistor, 330 Ω, 1/10W, 5%, 0402	Yageo	RC0402FR-07330RL	
30	R25, R29-30	3	Resistor, 0 Ω, 1/10W 0603	Panasonic	ERJ-3GEY0R00V	
31	U4	1	IC, Single Voltage Supervisor, 5V, DBV	Texas Instruments	TLV803MDBZR	





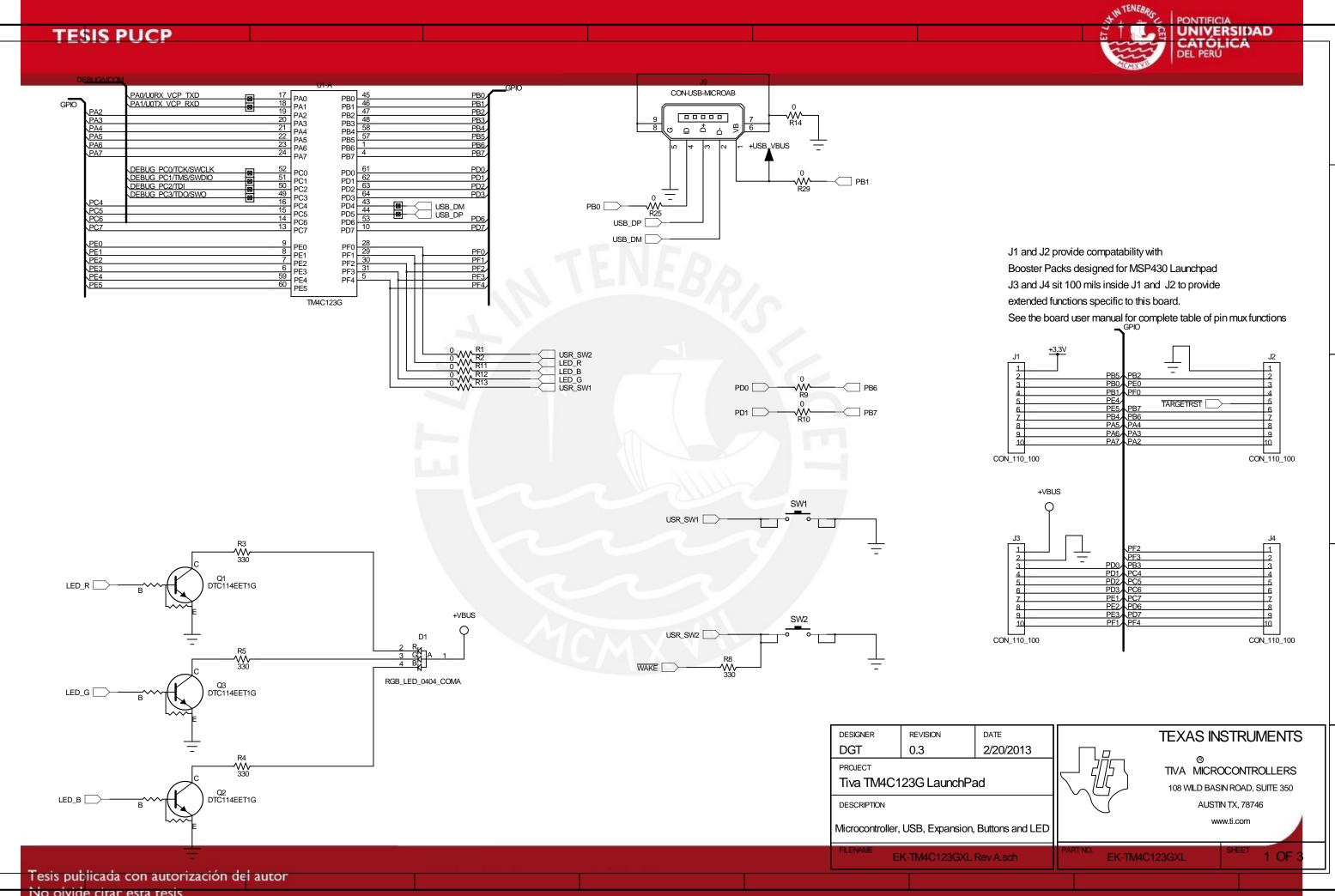
Appendix A SPMU296–April 2013

Schematics

This section contains the complete schematics for the Tiva C Series LaunchPad board.

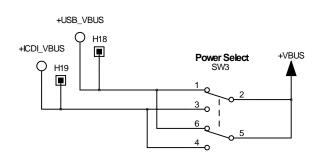
- Microcontroller, USB, Expansion, Buttons, and LED
- Power Management
- In-Circuit Debug Interface

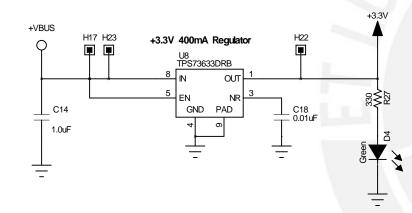


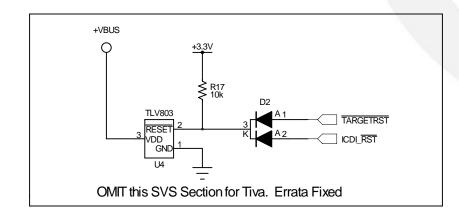


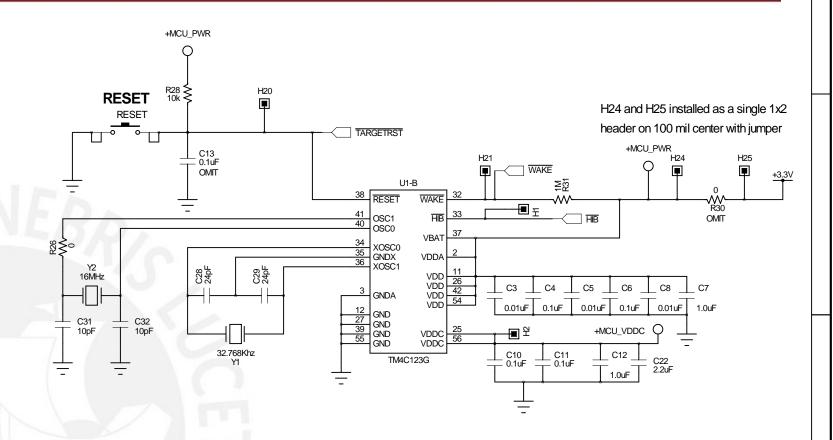
No oivide citar esta tesis

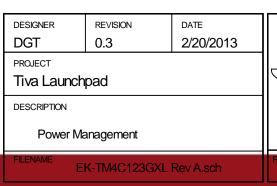














TEXAS INSTRUMENTS

TIVA MICROCONTROLLERS

108 WILD BASIN ROAD, SUITE 350

AUSTIN TX, 78746

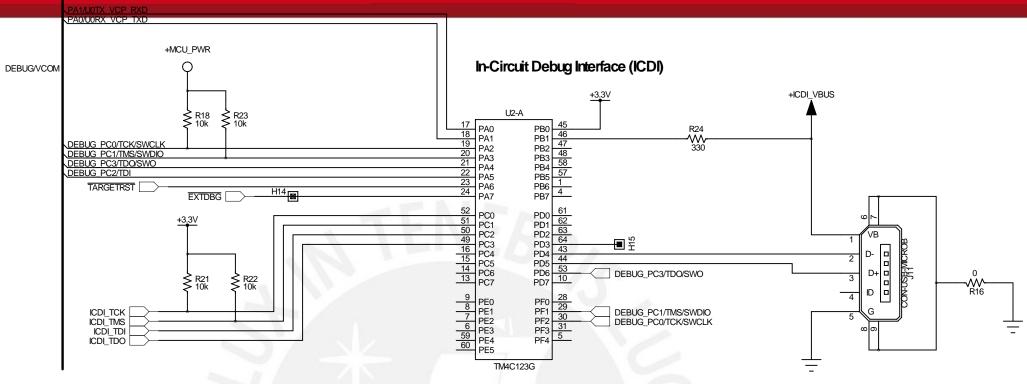
www.ti.com

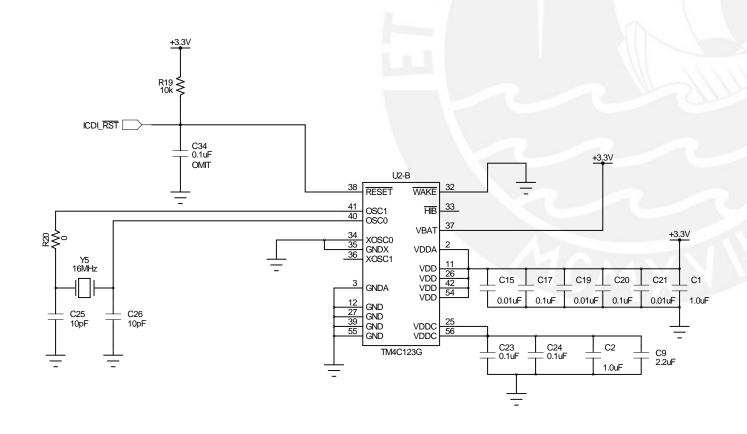
TNO. EK-TM4C123GXL

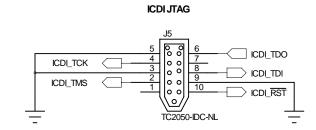
H13

SHEET 2 OF

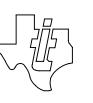












TEXAS INSTRUMENTS

TIVA MICROCONTROLLERS 108 WILD BASIN ROAD, SUITE 350 AUSTIN TX, 78746

www.ti.com

EK-TM4C123GXL

SHEET 3 OF



EVALUATION BOARD/KIT/MODULE (EVM) ADDITIONAL TERMS

Texas Instruments (TI) provides the enclosed Evaluation Board/Kit/Module (EVM) under the following conditions:

The user assumes all responsibility and liability for proper and safe handling of the goods. Further, the user indemnifies TI from all claims arising from the handling or use of the goods.

Should this evaluation board/kit not meet the specifications indicated in the User's Guide, the board/kit may be returned within 30 days from the date of delivery for a full refund. THE FOREGOING LIMITED WARRANTY IS THE EXCLUSIVE WARRANTY MADE BY SELLER TO BUYER AND IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED, IMPLIED, OR STATUTORY, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. EXCEPT TO THE EXTENT OF THE INDEMNITY SET FORTH ABOVE, NEITHER PARTY SHALL BE LIABLE TO THE OTHER FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES.

Please read the User's Guide and, specifically, the Warnings and Restrictions notice in the User's Guide prior to handling the product. This notice contains important safety information about temperatures and voltages. For additional information on TI's environmental and/or safety programs, please visit www.ti.com/esh or contact TI.

No license is granted under any patent right or other intellectual property right of TI covering or relating to any machine, process, or combination in which such TI products or services might be or are used. TI currently deals with a variety of customers for products, and therefore our arrangement with the user is not exclusive. TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein.

REGULATORY COMPLIANCE INFORMATION

As noted in the EVM User's Guide and/or EVM itself, this EVM and/or accompanying hardware may or may not be subject to the Federal Communications Commission (FCC) and Industry Canada (IC) rules.

For EVMs **not** subject to the above rules, this evaluation board/kit/module is intended for use for ENGINEERING DEVELOPMENT, DEMONSTRATION OR EVALUATION PURPOSES ONLY and is not considered by TI to be a finished end product fit for general consumer use. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to part 15 of FCC or ICES-003 rules, which are designed to provide reasonable protection against radio frequency interference. Operation of the equipment may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

General Statement for EVMs including a radio

User Power/Frequency Use Obligations: This radio is intended for development/professional use only in legally allocated frequency and power limits. Any use of radio frequencies and/or power availability of this EVM and its development application(s) must comply with local laws governing radio spectrum allocation and power limits for this evaluation module. It is the user's sole responsibility to only operate this radio in legally acceptable frequency space and within legally mandated power limitations. Any exceptions to this are strictly prohibited and unauthorized by Texas Instruments unless user has obtained appropriate experimental/development licenses from local regulatory authorities, which is responsibility of user including its acceptable authorization.

For EVMs annotated as FCC - FEDERAL COMMUNICATIONS COMMISSION Part 15 Compliant

Caution

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

FCC Interference Statement for Class A EVM devices

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.



FCC Interference Statement for Class B EVM devices

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- · Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- · Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

For EVMs annotated as IC - INDUSTRY CANADA Compliant

This Class A or B digital apparatus complies with Canadian ICES-003.

Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

Concerning EVMs including radio transmitters

This device complies with Industry Canada licence-exempt RSS standard(s). Operation is subject to the following two conditions: (1) this device may not cause interference, and (2) this device must accept any interference, including interference that may cause undesired operation of the device.

Concerning EVMs including detachable antennas

Under Industry Canada regulations, this radio transmitter may only operate using an antenna of a type and maximum (or lesser) gain approved for the transmitter by Industry Canada. To reduce potential radio interference to other users, the antenna type and its gain should be so chosen that the equivalent isotropically radiated power (e.i.r.p.) is not more than that necessary for successful communication.

This radio transmitter has been approved by Industry Canada to operate with the antenna types listed in the user guide with the maximum permissible gain and required antenna impedance for each antenna type indicated. Antenna types not included in this list, having a gain greater than the maximum gain indicated for that type, are strictly prohibited for use with this device.

Cet appareil numérique de la classe A ou B est conforme à la norme NMB-003 du Canada.

Les changements ou les modifications pas expressément approuvés par la partie responsable de la conformité ont pu vider l'autorité de l'utilisateur pour actionner l'équipement.

Concernant les EVMs avec appareils radio

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes : (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

Concernant les EVMs avec antennes détachables

Conformément à la réglementation d'Industrie Canada, le présent émetteur radio peut fonctionner avec une antenne d'un type et d'un gain maximal (ou inférieur) approuvé pour l'émetteur par Industrie Canada. Dans le but de réduire les risques de brouillage radioélectrique à l'intention des autres utilisateurs, il faut choisir le type d'antenne et son gain de sorte que la puissance isotrope rayonnée équivalente (p.i.r.e.) ne dépasse pas l'intensité nécessaire à l'établissement d'une communication satisfaisante.

Le présent émetteur radio a été approuvé par Industrie Canada pour fonctionner avec les types d'antenne énumérés dans le manuel d'usage et ayant un gain admissible maximal et l'impédance requise pour chaque type d'antenne. Les types d'antenne non inclus dans cette liste, ou dont le gain est supérieur au gain maximal indiqué, sont strictement interdits pour l'exploitation de l'émetteur.



[Important Notice for Users of this Product in Japan]

This development kit is NOT certified as Confirming to Technical Regulations of Radio Law of Japan

If you use this product in Japan, you are required by Radio Law of Japan to follow the instructions below with respect to this product:

- Use this product in a shielded room or any other test facility as defined in the notification #173 issued by Ministry of Internal Affairs and Communications on March 28, 2006, based on Sub-section 1.1 of Article 6 of the Ministry's Rule for Enforcement of Radio Law of Japan,
- 2. Use this product only after you obtained the license of Test Radio Station as provided in Radio Law of Japan with respect to this product, or
- 3. Use of this product only after you obtained the Technical Regulations Conformity Certification as provided in Radio Law of Japan with respect to this product. Also, please do not transfer this product, unless you give the same notice above to the transferee. Please note that if you could not follow the instructions above, you will be subject to penalties of Radio Law of Japan.

Texas Instruments Japan Limited (address) 24-1, Nishi-Shinjuku 6 chome, Shinjuku-ku, Tokyo, Japan

http://www.tij.co.jp

【ご使用にあたっての注】

本開発キットは技術基準適合証明を受けておりません。

本製品のご使用に際しては、電波法遵守のため、以下のいずれかの措置を取っていただく必要がありますのでご注意ください。

- 1. 電波法施行規則第6条第1項第1号に基づく平成18年3月28日総務省告示第173号で定められた電波暗室等の試験設備でご使用いただく。
- 2. 実験局の免許を取得後ご使用いただく。
- 3. 技術基準適合証明を取得後ご使用いただく。

なお、本製品は、上記の「ご使用にあたっての注意」を譲渡先、移転先に通知しない限り、譲渡、移転できないものとします。

上記を遵守頂けない場合は、電波法の罰則が適用される可能性があることをご留意ください。

日本テキサス・インスツルメンツ株式会社 東京都新宿区西新宿6丁目24番1号 西新宿三井ビル

http://www.tij.co.jp



EVALUATION BOARD/KIT/MODULE (EVM) WARNINGS, RESTRICTIONS AND DISCLAIMERS

For Feasibility Evaluation Only, in Laboratory/Development Environments. Unless otherwise indicated, this EVM is not a finished electrical equipment and not intended for consumer use. It is intended solely for use for preliminary feasibility evaluation in laboratory/development environments by technically qualified electronics experts who are familiar with the dangers and application risks associated with handling electrical mechanical components, systems and subsystems. It should not be used as all or part of a finished end product.

Your Sole Responsibility and Risk. You acknowledge, represent and agree that:

- 1. You have unique knowledge concerning Federal, State and local regulatory requirements (including but not limited to Food and Drug Administration regulations, if applicable) which relate to your products and which relate to your use (and/or that of your employees, affiliates, contractors or designees) of the EVM for evaluation, testing and other purposes.
- 2. You have full and exclusive responsibility to assure the safety and compliance of your products with all such laws and other applicable regulatory requirements, and also to assure the safety of any activities to be conducted by you and/or your employees, affiliates, contractors or designees, using the EVM. Further, you are responsible to assure that any interfaces (electronic and/or mechanical) between the EVM and any human body are designed with suitable isolation and means to safely limit accessible leakage currents to minimize the risk of electrical shock hazard.
- 3. You will employ reasonable safeguards to ensure that your use of the EVM will not result in any property damage, injury or death, even if the EVM should fail to perform as described or expected.
- 4. You will take care of proper disposal and recycling of the EVM's electronic components and packing materials.

Certain Instructions. It is important to operate this EVM within TI's recommended specifications and environmental considerations per the user guidelines. Exceeding the specified EVM ratings (including but not limited to input and output voltage, current, power, and environmental ranges) may cause property damage, personal injury or death. If there are questions concerning these ratings please contact a TI field representative prior to connecting interface electronics including input power and intended loads. Any loads applied outside of the specified output range may result in unintended and/or inaccurate operation and/or possible permanent damage to the EVM and/or interface electronics. Please consult the EVM User's Guide prior to connecting any load to the EVM output. If there is uncertainty as to the load specification, please contact a TI field representative. During normal operation, some circuit components may have case temperatures greater than 60°C as long as the input and output are maintained at a normal ambient operating temperature. These components include but are not limited to linear regulators, switching transistors, pass transistors, and current sense resistors which can be identified using the EVM schematic located in the EVM User's Guide. When placing measurement probes near these devices during normal operation, please be aware that these devices may be very warm to the touch. As with all electronic evaluation tools, only qualified personnel knowledgeable in electronic measurement and diagnostics normally found in development environments should use these EVMs.

Agreement to Defend, Indemnify and Hold Harmless. You agree to defend, indemnify and hold TI, its licensors and their representatives harmless from and against any and all claims, damages, losses, expenses, costs and liabilities (collectively, "Claims") arising out of or in connection with any use of the EVM that is not in accordance with the terms of the agreement. This obligation shall apply whether Claims arise under law of tort or contract or any other legal theory, and even if the EVM fails to perform as described or expected.

Safety-Critical or Life-Critical Applications. If you intend to evaluate the components for possible use in safety critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, such as devices which are classified as FDA Class III or similar classification, then you must specifically notify TI of such intent and enter into a separate Assurance and Indemnity Agreement.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265 Copyright © 2013, Texas Instruments Incorporated



IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have *not* been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products Applications

Audio www.ti.com/audio Automotive and Transportation www.ti.com/automotive Amplifiers amplifier.ti.com Communications and Telecom www.ti.com/communications **Data Converters** dataconverter.ti.com Computers and Peripherals www.ti.com/computers **DLP® Products** www.dlp.com Consumer Electronics www.ti.com/consumer-apps

DSP dsp.ti.com **Energy and Lighting** www.ti.com/energy Clocks and Timers www.ti.com/clocks Industrial www.ti.com/industrial Interface interface.ti.com Medical www.ti.com/medical www.ti.com/security Logic logic.ti.com Security

Power Mgmt power.ti.com Space, Avionics and Defense www.ti.com/space-avionics-defense

Microcontrollers <u>microcontroller.ti.com</u> Video and Imaging <u>www.ti.com/video</u>

RFID <u>www.ti-rfid.com</u>

OMAP Applications Processors <u>www.ti.com/omap</u> TI E2E Community <u>e2e.ti.com</u>

Wireless Connectivity <u>www.ti.com/wirelessconnectivity</u>

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265 Copyright © 2013, Texas Instruments Incorporated

TESIS PUCP Mouser Electronics



Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Texas Instruments:

EK-TM4C123GXL









SERIE NP/NPH/NPX SELLADAS Y RECARGABLES BATERIAS DE PLOMO-ACIDO



			Capacidad					Altura	Total				
Tipo	Tipo FR*	Volts	Nominal	Larg		Ancho		Inc. Term		Pes		Layout	Terminales
NIDUO 40	NPH2-12FR		(10 hr rate - Ah) 2.0	mm.	(in.)	mm	(in.)	mm.	(in.)	kgs. 0.84	(lbs.)	2	A
NPH2-12	I S MARK STOLEN STOLEN	10	77.75	68.0	2.68	51.0	2.01	88.0	3.46	17707021	1.85	2	0.54
NPH3.2-12	NPH3.2-12FR	12	3.2	134.0	5,28	67.0	2.64	64.0	2.52	1.40	3.09	3	A
NPH5-12	NPH5-12FR		5.0	90.0	3.54	70.0	2.75	106.0	4.17	2.00	4.41	1	D
	_		Regimen 20hr										
SERIE N	IP	_	(Ah)										
NP4.2-4H		4	4.2	48.0	1.89	35.5	1.40	119.0	4.68	0.56	1.23	6	-
NP1.2-6	NP1.2-6FR		1.2	97.0	3.82	25.0	0.98	54.5	2.15	0.30	0.66	1	Α
NP3-6	-		3.0	134.0	5.28	34.0	1.33	64.0	2.52	0.65	1.43	1	Α
NP4-6	-	6	4.0	70.0	2.76	47.0	1.85	105.5	4.15	0.85	1.87	5	А
NP7-6	NP7-6FR		7.0	151.0	5.95	64.0	1.33	97.5	3.84	1.35	2.98	1	A/D
NP10-6	NP10-6FR		10.0	151.0	5.95	50.0	1.97	97.5	3.84	2.00	4.41	1	A/D
NP0.8-12	NP0.8-12FR**		0.8	96.0	3.78	25.0	0.98	61.5	2.42	0.35	0.77	7	1
NP1.2-12	NP1.2-12FR		1.2	97.0	3.82	48.0	1.89	54.5	2.15	0.57	1.25	3	А
NP2-12	-		2.0	150.0	5.91	20.0	0.79	89.0	3.50	0.70	1.54	8	В
NP2.3-12	NP2.3-12FR		2.3	178.0	7.01	34.0	1.34	64.0	2.52	0.94	2.07	1	А
NP2.6-12	NP2.6-12FR]	2.6	134.0	5.28	67.0	2.64	64.0	2.52	1.12	2.47	3	Α
NP4-12	NP4-12FR		4.0	90.0	3.54	70.0	2.76	106.0	4.17	1.70	3.74	1	A/D
NP7-12	NP7-12FR	1	7.0	151.0	5.94	65.0	2.56	97.5	3.84	2.65	6.17	4	A/D
NP12-12	NP12-12FR	12	12.0	151.0	5.94	98.0	3.86	97.5	3.84	4.00	8.82	4	D
NP18-12B	NP18-12BFR	1	17.2	181.0	7.13	76.2	2.99	167.0	6.57	6.20	13.64	2	Е
NP24-12	NP24-12FR	1	24.0	166.0	6.54	175.0	6.89	125.0	4.92	8.65	19.05	2	С
NP24-12B	NP24-12BFR	1	24.0	166.0	6.54	175.0	6.89	125.0	4.92	8.65	19.05	2	Е
-	NP26-12B	1	26.0	166.0	6.54	125.0	4.92	175.0	6.89	9.30	20.50	2	J
-	NP26-12R	1	26.0	166.0	6.54	125.0	4.92	175.0	6.89	9.30	20.50	2	К
	NP38-12B	1	38.0	197.0	7.74	165.0	6.50	175.0	6.89	13.80	30.40	2	F
	NP38-12R	1	38.0	197.0	7.74	165.0	6.50	175.0	6.89	13.80	30.40	2	К
NP65-12	NP65-12FR	1	65.0	350.0	13.78	166.0	6.54	174.0	6.85	22.80	50.20	2	G
CEDIE I	IDV		W/Celda A 1.67 V Final										
SERIE N		1.0	(Regimen 15 min.)	151.0	F 05	50.0	1.07	07.5	201	2.00	4.40	1	A/D
NPX-50	NPX-50FR	6	50W/Cell	151.0	5.95	50.0	1.97	97.5	3.84	2.00	4.41	1	A/D
NPX-25	NPX-25FR	-	23W/Cell	90.0	3.54	70.0	2.75	106.0	4.17	2.00	4.41	1	D
7	NPX-100B		95W/Cell	166.0	6.54	125.0	4.92	175.0	6.89	9.30	20.80	2	J
-	NPX-100R	12	95W/Cell	166.0	6.54	125.0	4.92	175.0	6.89	9.30	20.80	2	K
*	NPX-150B		150W/Cell	197.0	7.76	165.0	6.50	175.0	6.89	15.50	34.10	2	J

NOTAS:

- * FR: UL94-VO, Caja y Tapa Antiflama (Indice de Oxigeno; 30)
- ** FR: UL94-V2, Caja y Tapa Antiflama (Indice de Oxigeno; 30)
- † Reconocido por UL, File No. MH 12970
- Hecho en Los EEUU (Hays, KS) UL File No. MH16464

Toda información está sujeta a cambio sin previo aviso

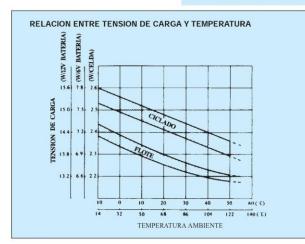


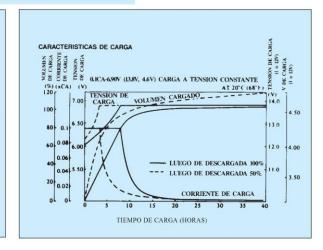
La marca Yuasa de baterías selladas, ha establecido desde su introducción las normas de calidad y excelencia en el ámbito de la tecnologia para baterias de plomo-ácido recargables y selladas. Este estándar ha sido utilizado como único en aplicaciones, tales como; seguridad, sistemas de energía ininterrumpida (UPS), telecomunicaciones, iluminación de emergencia y equipamiento médico. En cualquier, lugar, donde se requiera energía para back-up en forma eficiente y confiable, Yuasa ocupa el espacio.

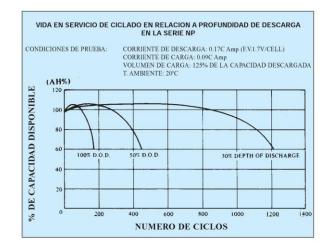
La línea de productos NP de marca Yuasa, cubre todo el espectro del tamaño de baterías, desde 0.8 Ah hasta 65 Ah en 4,6 y 12 volts. Yuasa también ofrece una linea completa de baterias antiflama (UL94-V0, LOI 30). Designadas como 'FR', estas baterías cumplen con las especificaciones, UL1778 para retardo de llamas en sistemas UPS y con los requerimientos de Bellcore para telecomunicaciones.

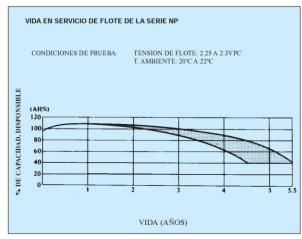
Yuasa también ofrece las series de baterías NPX y NPH, diseñadas para aplicaciones con descargas de alto régimen. Estas baterías se utilizan fundamentalmente cuando se requiere gran potencia en un corto tiempo. Con un 50% más de potencia disponible y un 30% de menos tamaño que las baterías convencionales, las series NPX y NPH ofrecen un valor superior, especialmente en las aplicaciones para sistemas UPS.

SERIE NPX - W	ATTS POR	CELDA H	ASTA 1.67	V FINAL
	5 MIN	10 MIN	15 MIN	20 MIN
NPX-25	47	31	23	18
NPX-50	94	60	50	38
NPX-35	66	45	35	29
NPX-80	140	104	80	65
NPX-100	185	125	95	75
NPX-150	285	200	150	120

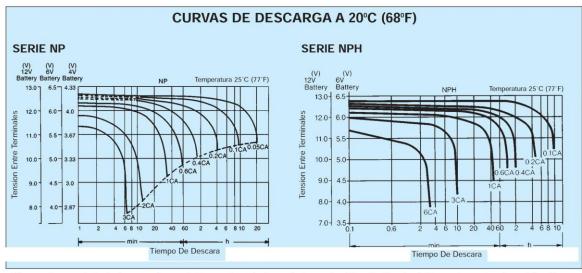












· Si se requiere corriente de descarga de mas de 3 veces la corriente nominal consulte al Servicio Tecnico de Enersystem antes de su utilización.

Notas importantes para prolongar la vida de la bateria:

Carga

- q Uso standby: Aplicar v constante a 2.275 v por celda (0 2.25-2.30 vpc).
- q Uso ciclado: Aplicar v constante a 2.40-2.50 v por celda La corriente inicial de carga debe ser menor a 0.25 CA.
- q Almacenaje de 6 meses: cargar a una tension constante de 2.40 volts por celda. La corriente inicial de carga debe ser menor a 0.1 CA por 15 a 20 horas.

n Descarga

- **q** Interrumpir la descarga al llegar a la tension minima permitida recargar inmediatamente.
- q No operar a mas de 6 veces la corriente nominal.

n Almacenaje

- **q** Almacenar siempre la bateria totalmente cargada.
- q Si la bateria debe ser almacenada por un largo periodo de tiempo, recargar cada 6 meses.

n Temperatura

- q Mantener a t ambiente entre 15°C y + 50°C tanto para la carga como para la descarga.
- q Almacenar las baterias en un lugar calido y seco.

n Incorporacion de la bateria en equipos

- q Colocar la bateria en un lugar bien ventilado.
- q Evite instalar la bateria cerca de unidades que liberen calor, como ser un transformador.
- q Coloque la bateria en la parte mas baja del equipo para evitar el calentamiento.

n Otros

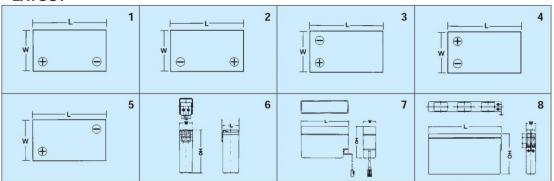
- q Evite los cortocircuitos o fuegos cercanos.
- q No exponer a llama.
- q Evite el contacto de la bateria con gasolina, solventes, resinas sinteticas, thinner, aceites, etc.

TENSION FINAL DE DESCARGA

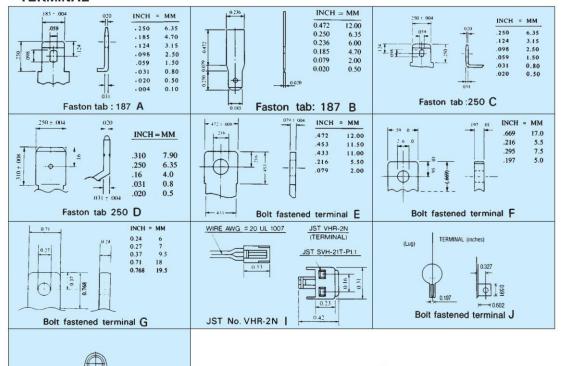
Corriente de descarga	Voltaje final de descarga (V/celda)		
0.1 C o menor o descargas intermitentes	1.7		
0.1 7C o corriente cercana a esta	1.70		
0. 6C o corriente cercana a esta	1.67		
0.6C o corriente cercana a esta	1.60		
Desde 0.6C a	1. 0		
Corriente en e ceso (> C)	1. 0		



LAYOUT



TERMINAL









UNF#10-32 Threaded Receptacle K



Tiva[™] C Series ARM[®] Cortex[™]-M Microcontrollers

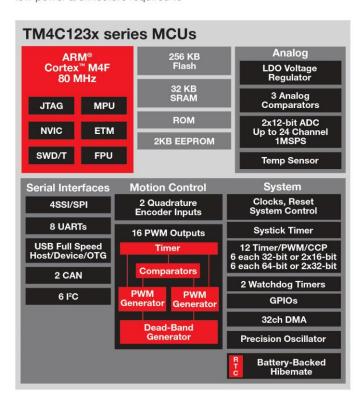


Product Bulletin

Overview

The Tiva C Series microcontrollers provide a broad portfolio of floating-point enabled ARM Cortex-M4F MCUs. Designers who migrate to the Tiva C Series MCUs benefit from a balance between the performance needed to create highly responsive mixed-signal applications and the low power architecture required to

address increasingly aggressive power budgets. Tiva C Series MCUs are supported by TivaWare[™] for C Series, designed specifically for those customers who want to get started easily, write production-ready code quickly, and minimize their overall cost of software ownership.



Key Features

Performance:

 ARM Cortex-M4F core up to 80 MHz

Power:

- Active power as low as 375 µA/MHz
- Low-power modes down to 1.6 μA

Integration:

- Up to 40 PWM outputs, 2 QEIs
- Up to 24 timers
- 12-bit ADCs up to 1 MSPS
- Up to 20 serial ports-UART, I²C, SPI/SSI
- CAN A/B, USB 2.0 OTG/H/D

Applications

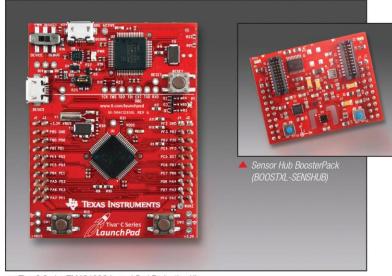
- · Security and access control
- Home and building automation
- · Industrial automation
- · Human machine interface
- · Lighting control
- · System management

www.ti.com/tiva



Tiva EK-TM4C123GXL LaunchPad Evaluation Kit

The Tiva EK-TM4C123GXL LaunchPad provides everything a customer needs to get started evaluating a Tiva C Series microcontroller in just 10 minutes or less. With an affordable price point, visit www.ti.com/launchpad and purchase this speedy and highly-integrated microcontroller evaluation tool. Don't forget to tell your friends!



▲ Tiva C Series TM4C123G LaunchPad Evaluation Kit

Part Number Reference

Old Part Number	New Part Number	Old Part Number	New Part Number	Old Part Number	New Part Number
LM4F110B2QR	TM4C1231C3PM	LM4F121C4QR	TM4C1232D5PM	LM4F210E5QR	TM4C123BE6PM
LM4F110C4QR	TM4C1231D5PM	LM4F121E5QR	TM4C1232E6PM	LM4F210H5QR	TM4C123BH6PM
LM4F110E5QR	TM4C1231E6PM	LM4F121H5QR	TM4C1232H6PM	LM4F211E5QR	TM4C123AE6PM
LM4F110H5QR	TM4C1231H6PM	LM4F122C4QC	TM4C1233D5PZ	LM4F211H5QR	TM4C123AH6PM
LM4F111B2QR	TM4C1230C3PM	LM4F122E5QC	TM4C1233E6PZ	LM4F212E5QC	TM4C123BE6PZ
LM4F111C4QR	TM4C1230D5PM	LM4F122H5QC	TM4C1233H6PZ	LM4F212H5BB	TM4C123BH6ZRB
LM4F111E5QR	TM4C1230E6PM	LM4F122H5QD	TM4C1233H6PGE	LM4F212H5QC	TM4C123BH6PZ
LM4F111H5QR	TM4C1230H6PM	LM4F130C4QR	TM4C1237D5PM	LM4F212H5QD	TM4C123BH6PGE
LM4F112C4QC	TM4C1231D5PZ	LM4F130E5QR	TM4C1237E6PM	LM4F230E5QR	TM4C123GE6PM
LM4F112E5QC	TM4C1231E6PZ	LM4F130H5QR	TM4C1237H6PM	LM4F230H5QR	TM4C123GH6PM
LM4F112H5QC	TM4C1231H6PZ	LM4F131C4QR	TM4C1236D5PM	LM4F231E5QR	TM4C123FE6PM
LM4F112H5QD	TM4C1231H6PGE	LM4F131E5QR	TM4C1236E6PM	LM4F231H5QR	TM4C123FH6PM
LM4F120B2QR	TM4C1233C3PM	LM4F131H5QR	TM4C1236H6PM	LM4F232E5QC	TM4C123GE6PZ
LM4F120C4QR	TM4C1233D5PM	LM4F132C4QC	TM4C1237D5PZ	LM4F232H5BB	TM4C123GH6ZRB
LM4F120E5QR	TM4C1233E6PM	LM4F132E5QC	TM4C1237E6PZ	LM4F232H5QC	TM4C123GH6PZ
LM4F120H5QR	TM4C1233H6PM	LM4F132H5QC	TM4C1237H6PZ	LM4F232H5QD	TM4C123GH6PGE
LM4F121B2QR	TM4C1232C3PM	LM4F132H5QD	TM4C1237H6PGE		

Important Notice: The products and services of Texas Instruments Incorporated and its subsidiaries described herein are sold subject to Ti's standard terms and conditions of sale. Customers are advised to obtain the most current and complete information about Ti products and services before placing orders. Ti assumes no liability for applications assistance, customer's applications or product designs, software performance, or infringement of patents. The publication of information regarding any other company's products or services does not constitute Ti's approval, warranty or endorsement thereof.

B090712

The platform bar, Tiva and TivaWare are trademarks of Texas Instruments. All other trademarks are the property of their respective owners.



SPMT284

© 2013 Texas Instruments Incorporated



IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have not been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Applications

Products Audio Automotive and Transportation www.ti.com/automotive www.ti.com/audio **Amplifiers** amplifier.ti.com Communications and Telecom www.ti.com/communications Data Converters Computers and Peripherals dataconverter.ti.com www.ti.com/computers DLP® Products www.dlp.com Consumer Electronics www.ti.com/consumer-apps DSP dsp.ti.com Energy and Lighting www.ti.com/energy Clocks and Timers www.ti.com/clocks Industrial www.ti.com/industrial Interface interface.ti.com Medical www.ti.com/medical www.ti.com/security Logic logic.ti.com Security Power Mgmt Space, Avionics and Defense www.ti.com/space-avionics-defense power.ti.com

microcontroller.ti.com Microcontrollers Video and Imaging www.ti.com/video

OMAP Applications Processors www.ti.com/omap TI E2E Community e2e.ti.com

www.ti-rfid.com

Wireless Connectivity www.ti.com/wirelessconnectivity

> Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265 Copyright © 2013, Texas Instruments Incorporated



Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Texas Instruments:

BOOSTXL-SENSHUB



Cortex®-M4

Revision r0p1

Technical Reference Manual





Cortex-M4

Technical Reference Manual

Copyright © 2009, 2010 ARM Limited. All rights reserved.

Release Information

The following changes have been made to this book.

Change History

Date	Issue	Confidentiality	Change
22 December 2009	A	Non-Confidential, Restricted Access	First release for r0p0
02 March 2010	В	Non-Confidential	Second release for r0p0
29 June 2010	С	Non-Confidential	Fiirst release for r0p1

Proprietary Notice

Words and logos marked with * or ™ are registered trademarks or trademarks of ARM* Limited in the EU and other countries, except as otherwise stated in this proprietary notice. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM Limited in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM Limited shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

Where the term ARM is used it means "ARM or any of its subsidiaries as appropriate".

Some material in this document is based on IEEE 754-2008 IEEE Standard for Binary Floating-Point Arithmetic. The IEEE disclaims any responsibility or liability resulting from the placement and use in the described manner.

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to

Unrestricted Access is an ARM internal classification.

Product Status

The information in this document is Final (information on a developed product).

Web Address

http://www.arm.com



Contents

Cortex-M4 Technical Reference Manual

1 1014			
	About this book	i)	
	Feedback	xi	
Intro	duction		
1.1	About the processor	1-2	
1.2			
1.3			
1.4	Configurable options	1-5	
1.5			
1.6			
Functional Description			
2.1		2-2	
2.2			
Prog	rammers Model		
3.1		3-2	
3.2			
3.3			
3.4			
3.5			
3.6			
3.7			
	<u> </u>		
3.8	Processor core register summary		
	1.1 1.2 1.3 1.4 1.5 1.6 Func 2.1 2.2 Prog 3.1 3.2 3.3 3.4 3.5 3.6	1.2 Features 1.3 Interfaces 1.4 Configurable options 1.5 Product documentation 1.6 Product revisions Functional Description 2.1 About the functions 2.2 Interfaces Programmers Model 3.1 About the programmers model 3.2 Modes of operation and execution 3.3 Instruction set summary 3.4 System address map 3.5 Write buffer 3.6 Exclusive monitor	



Chapter 4	System Control			
•	4.1 About system control	4-2		
	4.2 Register summary			
	4.3 Register descriptions			
Chapter 5	Memory Protection Unit			
	5.1 About the MPU	5-2		
	5.2 MPU functional description	5-3		
	5.3 MPU programmers model	5-4		
Chapter 6	Nested Vectored Interrupt Controller			
-	6.1 About the NVIC	6-2		
	6.2 NVIC functional description	6-3		
	6.3 NVIC programmers model			
Chapter 7	Floating Point Unit			
-	7.1 About the FPU	7-2		
	7.2 FPU Functional Description	7-3		
	7.3 FPU Programmers Model	7-9		
Chapter 8	Debug			
	8.1 About debug	8-2		
	8.2 About the AHB-AP	8-6		
	8.3 About the Flash Patch and Breakpoint Unit (FPB)	8-9		
Chapter 9	Data Watchpoint and Trace Unit			
•	9.1 About the DWT	9-2		
	9.2 DWT functional description	9-3		
	9.3 DWT Programmers Model			
Chapter 10	Instrumentation Trace Macrocell Unit			
•	10.1 About the ITM	10-2		
	10.2 ITM functional description	10-3		
	10.3 ITM programmers model	10-4		
Chapter 11	Trace Port Interface Unit			
	11.1 About the Cortex-M4 TPIU	11-2		
	11.2 TPIU functional description	11-3		
	11.3 TPIU programmers model	11-5		
Appendix A	Revisions			
	Glossary			



List of Tables Cortex-M4 Technical Reference Manual

	Change History	i
Table 1-1	Optional implementation components	
Table 3-1	Cortex-M4 instruction set summary	
Table 3-2	Cortex-M4 DSP instruction set summary	
Table 3-3	Memory regions	
Table 4-1	System control registers	
Table 4-2	ACTLR bit assignments	
Table 4-3	CPUID bit assignments	4-6
Table 4-4	AFSR bit assignments	
Table 5-1	MPU registers	5-4
Table 6-1	NVIC registers	6-4
Table 6-2	ICTR bit assignments	6-5
Table 7-1	FPU instruction set	7-4
Table 7-2	Default NaN values	7-6
Table 7-3	QNaN and SNaN handling	7-7
Table 7-4	Cortex-M4F Floating Point system registers	7-9
Table 8-1	Cortex-M4 ROM table identification values	8-3
Table 8-2	Cortex-M4 ROM table components	8-4
Table 8-3	SCS identification values	8-5
Table 8-4	Debug registers	8-5
Table 8-5	AHB-AP register summary	8-6
Table 8-6	CSW bit assignments	8-7
Table 8-7	FPB register summary	8-10
Table 9-1	DWT register summary	9-4
Table 10-1	ITM register summary	10-4
Table 10-2	ITM_TPR bit assignments	10-5
Table 11-1	TPIU registers	11-5
Table 11-2	TPIU_ACPR bit assignments	11-6
Table 11-3	TPIU_FFSR bit assignments	11-7
Table 11-4	TPIU_FFCR bit assignments	11-7

TESIS PUCP



Table 11-5	TRIGGER bit assignments	11-8
Table 11-6	Integration ETM Data bit assignments	
Table 11-7	ITATBCTR2 bit assignments	11-10
Table 11-8	Integration ITM Data bit assignments	
Table 11-9	ITATBCTR0 bit assignments	11-11
Table 11-10	TPIU ITCTRL bit assignments	
Table 11-11	TPIU_DEVID bit assignments	11-12
Table A-1	Issue A	A-1
Table A-2	Differences between issue A and issue B	A-1
Table A-3	Differences between issue B and issue C	A-1





List of Figures Cortex-M4 Technical Reference Manual

Cortex-M4 block diagram	2-2
System address map	3-14
Bit-band mapping	3-20
Processor register set	3-21
ACTLR bit assignments	4-5
CPUID bit assignments	4-6
AFSR bit assignments	4-6
ICTR bit assignments	6-4
FPU register bank	7-3
CoreSight discovery	8-2
CSW bit assignments	8-7
ITM_TPR bit assignments	10-5
TPIU block diagram	11-3
TPIU_ACPR bit assignments	11-6
TPIU_FFSR bit assignments	11-6
TPIU_FFCR bit assignments	11-7
TRIGGER bit assignments	11-8
Integration ETM Data bit assignments	11-9
ITATBCTR2 bit assignments	11-9
Integration ITM Data bit assignments	11-10
ITATBCTR0 bit assignments	11-11
TPIU_ITCTRL bit assignments	11-11
TPIU_DEVID bit assignments	11-12
TPIU_DEVTYPE bit assignments	11-13
	Bit-band mapping Processor register set ACTLR bit assignments CPUID bit assignments AFSR bit assignments ICTR bit assignments FPU register bank CoreSight discovery CSW bit assignments ITM_TPR bit assignments TPIU block diagram TPIU_ACPR bit assignments TPIU_FFSR bit assignments TPIU_FFCR bit assignments TRIGGER bit assignments ITATBCTR2 bit assignments Integration ITM Data bit assignments Integration ITM Data bit assignments ITATBCTR0 bit assignments ITATBCTR0 bit assignments TPIU_ITCTRL bit assignments TPIU_ITCTRL bit assignments TPIU_DEVID bit assignments



Preface

This preface introduces the *Cortex-M4 Technical Reference Manual* (TRM). It contains the following sections:

- About this book on page ix
- Feedback on page xii.



This book is for the Cortex-M4 processor.

Product revision status

The rnpn identifier indicates the revision status of the product described in this manual, where:

Identifies the major revision of the product. rn

Identifies the minor revision or modification status of the product. рn

Intended audience

This manual is written to help system designers, system integrators, verification engineers, and software programmers who are implementing a System-on-Chip (SoC) device based on the Cortex-M4 processor.

Using this book

This book is organized into the following chapters:

Chapter 1 Introduction

Read this for a description of the components of the processor, and of the product documentation.

Chapter 2 Functional Description

Read this for a description of the functionality of the processor.

Chapter 3 Programmers Model

Read this for a description of the processor register set, modes of operation, and other information for programming the processor.

Chapter 4 System Control

Read this for a description of the registers and programmers model for system control.

Chapter 5 Memory Protection Unit

Read this for a description of the Memory Protection Unit (MPU).

Chapter 6 Nested Vectored Interrupt Controller

Read this for a description of the interrupt processing and control.

Chapter 7 Floating Point Unit

Read this for a description of the Floating Point Unit (FPU)

Chapter 8 Debug

Read this for information about debugging and testing the processor core.

Chapter 9 Data Watchpoint and Trace Unit

Read this for a description of the Data Watchpoint and Trace (DWT) unit.

Chapter 10 Instrumentation Trace Macrocell Unit

Read this for a description of the Instrumentation Trace Macrocell (ITM) unit.

Chapter 11 Trace Port Interface Unit

Read this for a description of the Trace Port Interface Unit (TPIU).



Appendix A Revisions

Read this for a description of the technical changes between released issues of this book.

Glossary Read this for definitions of terms used in this book.

Conventions

Conventions that this book can use are described in:

Typographical

Typographical

The typographical conventions are:

italic Highlights important notes, introduces special terminology, denotes

internal cross-references, and citations.

bold Highlights interface elements, such as menu names. Denotes signal

names. Also used for terms in descriptive lists, where appropriate.

monospace Denotes text that you can enter at the keyboard, such as commands, file

and program names, and source code.

monospace Denotes a permitted abbreviation for a command or option. You can enter

the underlined text instead of the full command or option name.

monospace italic Denotes arguments to monospace text where the argument is to be

replaced by a specific value.

monospace Denotes language keywords when used outside example code.

< and > Enclose replaceable terms for assembler syntax where they appear in code

or code fragments. For example:

ADD Rd, Rn, <op2>

Additional reading

This section lists publications by ARM and by third parties.

See Infocenter, http://infocenter.arm.com, for access to ARM documentation.

ARM publications

This book contains information that is specific to this product. See the following documents for other relevant information:

- *ARMv7-M Architecture Reference Manual* (ARM DDI 0403)
- ARM Cortex-M4 Integration and Implementation Manual (ARM DII 0239)
- ARM ETM-M4 Technical Reference Manual (ARM DDI 0440)
- ARM AMBA® 3 AHB-Lite Protocol (v1.0) (ARM IHI 0033)
- *ARM AMBA*[™] 3 *APB Protocol Specification* (ARM IHI 0024)
- ARM CoreSight™ Components Technical Reference Manual (ARM DDI 0314)
- *ARM Debug Interface v5 Architecture Specification* (ARM IHI 0031).



Other publications

This section lists relevant documents published by third parties:

- IEEE Standard Test Access Port and Boundary-Scan Architecture 1149.1-2001 (JTAG)
- IEEE Standard IEEE Standard for Binary Floating-Point Arithmetic 754-2008.





Feedback

ARM welcomes feedback on this product and its documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on this manual

If you have comments on content then send e-mail to errata@arm.com. Give:

- the title
- the number, ARM DDI 0439C
- the page number(s) to which your comments refer
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.



Chapter 1 **Introduction**

This chapter introduces the processor and instruction set. It contains the following sections:

- *About the processor* on page 1-2
- Features on page 1-3
- Interfaces on page 1-4
- Configurable options on page 1-5
- Product documentation on page 1-6.



1.1 About the processor

The Cortex-M4 processor is a low-power processor that features low gate count, low interrupt latency, and low-cost debug. The Cortex-M4F is a processor with the same capability as the Cortex-M4 processor, and includes floating point arithmetic functionality (see Chapter 7 *Floating Point Unit*). Both processors are intended for deeply embedded applications that require fast interrupt response features.

Throughout this document the name Cortex-M4 refers to both Cortex-M4 and Cortex-M4F processors unless otherwise indicated.





1.2 Features

The Cortex-M4 processor incorporates:

- a processor core
- a Nested Vectored Interrupt Controller (NVIC) closely integrated with the processor core to achieve low latency interrupt processing
- multiple high-performance bus interfaces
- a low-cost debug solution with the optional ability to:
 - implement breakpoints and code patches
 - implement watchpoints, tracing, and system profiling
 - support printf() style debugging.
 - bridge to a *Trace Port Analyzer* (TPA).
- an optional *Memory Protection Unit* (MPU)
- a Floating Point Unit (FPU) unit, in the Cortex-M4F processor.





1.3 Interfaces

The processor has the following external interfaces:

- multiple memory and device bus interfaces
- ETM interface
- trace port interface
- debug port interface.





1.4 Configurable options

You can configure your Cortex-M4 implementation to include the following optional components as Table 1-1 shows:

Table 1-1 Optional implementation components

Component	Description
MPU	See Chapter 5 Memory Protection Unit
FPB	See Chapter 8 Debug
DWT	See Chapter 9 Data Watchpoint and Trace Unit
ITM	See Chapter 10 Instrumentation Trace Macrocell Unit
ETM	See the ETM-M4 Technical Reference Manual
AHB-AP	See Chapter 8 Debug
HTM interface	See AHB Trace Macrocell interface on page 2-7
TPIU	See Chapter 11 Trace Port Interface Unit
WIC	See Low power modes on page 6-3
Debug Port	See Debug Port AHB-AP interface on page 2-7
FPU	See Chapter 7 Floating Point Unit
Bit-banding	See Bit-banding on page 3-19
Constant AHB control	See Bus interfaces on page 2-5

- Note ----

You can only configure trace functionality in the following combinations:

- no trace functionality
- ITM and DWT
- ITM, DWT, and ETM
- ITM, DWT, ETM, and HTM.

You can configure the features provided in the DWT independently.



1.5 Product documentation

This section describes the processor books, how they relate to the design flow, and the relevant architectural standards and protocols.

See *Additional reading* on page x for more information about the books described in this section.

1.5.1 Documentation

The Cortex-M4 documentation is as follows:

Technical Reference Manual

The *Technical Reference Manual* (TRM) describes the functionality and the effects of functional options on the behavior of the Cortex-M4 processor. It is required at all stages of the design flow. Some behavior described in the TRM might not be relevant because of the way that the Cortex-M4 processor is implemented and integrated. If you are programming the Cortex-M4 processor then contact:

- the implementer to determine:
 - the build configuration of the implementation
 - what integration, if any, was performed before implementing the processor.
- the integrator to determine the pin configuration of the SoC that you are using.

Integration and Implementation Manual

The Integration and Implementation Manual (IIM) describes:

- The available build configuration options and related issues in selecting them.
- How to configure the *Register Transfer Level* (RTL) with the build configuration options.
- How to integrate the processor into a SoC. This includes a description of the integration kit and describes the pins that the integrator must tie off to configure the macrocell for the required integration.
- How to implement the processor into your design. This includes floorplanning guidelines, Design for Test (DFT) information, and how to perform netlist dynamic verification on the processor.
- The processes to sign off the integration and implementation of the design.

The ARM product deliverables include reference scripts and information about using them to implement your design.

Reference methodology documentation from your EDA tools vendor complements the IIM.

The IIM is a confidential book that is only available to licensees.

ETM-M4 Technical Reference Manual

The ETM-M4 *Technical Reference Manual* (TRM) describes the functionality and behavior of the Cortex-M4 Embedded Trace Macrocell. It is required at all stages of the design flow. Typically the ETM-M4 is integrated with the Cortex-M4 processor prior to implementation as a single macrocell.



Cortex-M4 User Guide Reference Material

This document provides reference material that ARM partners can configure and include in a User Guide for an ARM Cortex-M4 processor. Typically:

- each chapter in this reference material might correspond to a section in the User Guide
- each top-level section in this reference material might correspond to a chapter in the User Guide.

However, you can organize this material in any way, subject to the conditions of the licence agreement under which ARM supplied the material.

1.5.2 Design Flow

The processor is delivered as synthesizable RTL. Before it can be used in a product, it must go through the following process:

Implementation

The implementer configures and synthesizes the RTL.

Integration The integrator connects the implemented design into a SoC. This includes connecting it to a memory system and peripherals.

Programming

The system programmer develops the software required to configure and initialize the processor, and tests the required application software.

Each stage in the process can be performed by a different party. Implementation and integration choices affect the behavior and features of the processor.

For MCUs, often a single design team integrates the processor before synthesizing the complete design. Alternatively, the team can synthesise the processor on its own or partially integrated, to produce a macrocell that is then integrated, possibly by a separate team.

The operation of the final device depends on:

Build configuration

The implementer chooses the options that affect how the RTL source files are pre-processed. These options usually include or exclude logic that affects one or more of the area, maximum frequency, and features of the resulting macrocell.

Configuration inputs

The integrator configures some features of the processor by tying inputs to specific values. These configurations affect the start-up behavior before any software configuration is made. They can also limit the options available to the software.

Software configuration

Note -

The programmer configures the processor by programming particular values into registers. This affects the behavior of the processor.

11000
This manual refers to implementation-defined features that are applicable to build configuration
options. Reference to a feature that is included means that the appropriate build and pin
configuration options are selected. Reference to an enabled feature means one that has also been
configured by software.



1.5.3 Architecture and protocol information

The processor complies with, or implements, the specifications described in:

- ARM architecture
- Bus architecture
- Debug
- Embedded Trace Macrocell.

This book complements architecture reference manuals, architecture specifications, protocol specifications, and relevant external standards. It does not duplicate information from these sources.

ARM architecture

The processor implements the ARMv7-M architecture profile. See the *ARMv7-M Architecture Reference Manual*.

Bus architecture

The processor provides three primary bus interfaces implementing a variant of the AMBA 3 AHB-Lite protocol. The processor implements an interface for CoreSight and other debug components using the AMBA 3 APB protocol. See:

- the ARM AMBA 3 AHB-Lite Protocol (v1.0)
- the ARM AMBA 3 APB Protocol Specification.

Debug

The debug features of the processor implement the ARM debug interface architecture. See the ARM Debug Interface v5 Architecture Specification.

Embedded Trace Macrocell

The trace features of the processor implement the ARM Embedded Trace Macrocell architecture. See the *ARM Embedded Trace Macrocell Architecture Specification*.

Floating Point Unit

The Cortex-M4F processor implements single precision floating-point data processing as defined by the FPv4-SP architecture, that is part of the ARMv7-M architecture. It provides floating-point computation functionality that is compliant with the *ANSI/IEEE Std 754-2008*, *IEEE Standard for Binary Floating-Point Arithmetic*. See the *ARMv7M Architecture Reference Manual* and Chapter 7 *Floating Point Unit*.



1.6 Product revisions

This section describes the differences in functionality between product revisions:

• Differences in functionality between r0p0 and r0p1.

1.6.1 Differences in functionality between r0p0 and r0p1

In summary, the differences in functionality include:

• New implementation option to ensure constant AHB control during wait-stated transfers.





Chapter 2 Functional Description

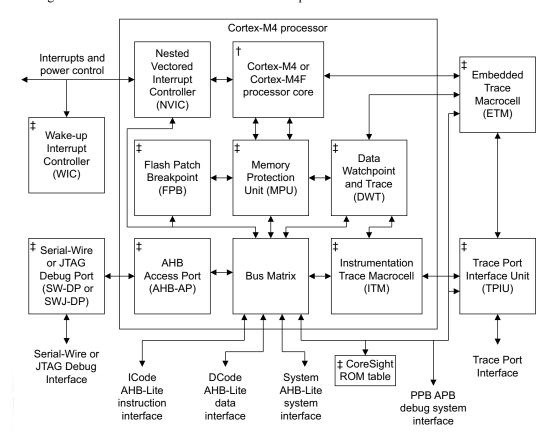
This chapter introduces the processor and its external interfaces. It contains the following sections:

- About the functions on page 2-2
- *Interfaces* on page 2-5.



2.1 About the functions

Figure 2-1 shows the structure of the Cortex-M4 processor.



- † For the Cortex-M4F processor, the core includes a Floating Point Unit (FPU)
- ‡ Optional component

Figure 2-1 Cortex-M4 block diagram

The Cortex-M4 processor features:

- A low gate count processor core, with low latency interrupt processing that has:
 - A subset of the Thumb instruction set, defined in the *ARMv7-M Architecture Reference Manual*.
 - Banked Stack Pointer (SP).
 - Hardware integer divide instructions, SDIV and UDIV.
 - Handler and Thread modes.
 - Thumb and Debug states.
 - Support for interruptible-continued instructions LDM, STM, PUSH, and POP for low interrupt latency.
 - Automatic processor state saving and restoration for low latency *Interrupt Service Routine* (ISR) entry and exit.
 - Support for ARMv6 big-endian byte-invariant or little-endian accesses.
 - Support for ARMv6 unaligned accesses.



- Floating Point Unit (FPU) in the Cortex-M4F processor providing
 - 32-bit instructions for single-precision (C float) data-processing operations.
 - Combined Multiply and Accumulate instructions for increased precision (Fused MAC).
 - Hardware support for conversion, addition, subtraction, multiplication with optional accumulate, division, and square-root.
 - Hardware support for denormals and all IEEE rounding modes.
 - 32 dedicated 32-bit single precision registers, also addressable as 16 double-word registers.
 - Decoupled three stage pipeline.
- Nested Vectored Interrupt Controller (NVIC) closely integrated with the processor core to achieve low latency interrupt processing. Features include:
 - External interrupts, configurable from 1 to 240.
 - Bits of priority, configurable from 3 to 8.
 - Dynamic reprioritization of interrupts.
 - Priority grouping. This enables selection of preempting interrupt levels and non preempting interrupt levels.
 - Support for tail-chaining and late arrival of interrupts. This enables back-to-back interrupt processing without the overhead of state saving and restoration between interrupts.
 - Processor state automatically saved on interrupt entry, and restored on interrupt exit, with no instruction overhead.
 - Optional Wake-up Interrupt Controller (WIC), providing ultra-low power sleep mode support.
- *Memory Protection Unit* (MPU). An optional MPU for memory protection, including:
 - Eight memory regions.
 - Sub Region Disable (SRD), enabling efficient use of memory regions.
 - The ability to enable a background region that implements the default memory map attributes.
- Bus interfaces:
 - Three Advanced High-performance Bus-Lite (AHB-Lite) interfaces: ICode, DCode, and System bus interfaces.
 - Private Peripheral Bus (PPB) based on Advanced Peripheral Bus (APB) interface.
 - Bit-band support that includes atomic bit-band write and read operations.
 - Memory access alignment.
 - Write buffer for buffering of write data.
 - Exclusive access transfers for multiprocessor systems.
- Low-cost debug solution that features:
 - Debug access to all memory and registers in the system, including access to
 memory mapped devices, access to internal core registers when the core is halted,
 and access to debug control registers even while SYSRESETn is asserted.
 - Serial Wire Debug Port (SW-DP) or Serial Wire JTAG Debug Port (SWJ-DP) debug access.



Optional *Flash Patch and Breakpoint* (FPB) unit for implementing breakpoints and code patches.

- Optional Data Watchpoint and Trace (DWT) unit for implementing watchpoints, data tracing, and system profiling.
- Optional *Instrumentation Trace Macrocell* (ITM) for support of printf() style debugging.
- Optional Trace Port Interface Unit (TPIU) for bridging to a Trace Port Analyzer (TPA), including Single Wire Output (SWO) mode.
- Optional *Embedded Trace Macrocell* (ETM) for instruction trace.





2.2 Interfaces

The processor contains the following external interfaces:

- Bus interfaces
- ETM interface on page 2-7
- *AHB Trace Macrocell interface* on page 2-7
- *Debug Port AHB-AP interface* on page 2-7.

2.2.1 Bus interfaces

The processor contains three external *Advanced High-performance Bus* (AHB)-Lite bus interfaces and one *Advanced Peripheral Bus* (APB) interface:

- *ICode memory interface*
- DCode memory interface on page 2-6
- System interface on page 2-6
- Private Peripheral Bus (PPB) on page 2-6.

The processor matches the AMBA 3 specification except for maintaining control information during waited transfers. The AMBA 3 AHB-Lite Protocol states that when the slave is requesting wait states the master must not change the transfer type, except for the following cases:

- On an IDLE transfer, the master can change the transfer type from IDLE to NONSEQ.
- On a BUSY transfer with a fixed length burst, the master can change the transfer type from BUSY to SEQ.
- On a BUSY transfer with an undefined length burst, the master can change the transfer type from BUSY to any other transfer type.

The processor does not match this definition because it might change the access type from SEQ or NONSEQ to IDLE during a waited transfer. The processor might also change the address or other control information and therefore request an access to a new location. The original address that was retracted might not be requested again. This cancels the outstanding transfer that has not occurred because the previous access is wait-stated and awaiting completion. This is done so that the processor can have a lower interrupt latency and higher performance in wait-stated systems by retracting accesses that are no longer required.

To achieve complete compliance with the AMBA 3 specification you can implement the design with the AHB_CONST_CTRL parameter set to 1. This ensures that once transfers are issued during a wait-stated response they are never retracted or modified and the original transfer is honoured. The consequence of setting this parameter is that the performance of the core might decrease for wait-stated systems as a result of the interrupt and branch latency increasing.

ICode memory interface

Instruction fetches from Code memory space, 0x00000000 to 0x1FFFFFC, are performed over this 32-bit AHB-Lite bus.

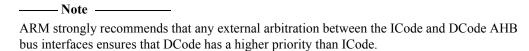
The Debugger cannot access this interface. All fetches are word-wide. The number of instructions fetched per word depends on the code running and the alignment of the code in memory.



DCode memory interface

Data and debug accesses to Code memory space, 0x00000000 to 0x1FFFFFFF, are performed over this 32-bit AHB-Lite bus. Core data accesses have a higher priority than debug accesses on this bus. This means that debug accesses are waited until core accesses have completed when there are simultaneous core and debug access to this bus.

Control logic in this interface converts unaligned data and debug accesses into two or three aligned accesses, depending on the size and alignment of the unaligned access. This stalls any subsequent data or debug access until the unaligned access has completed.



System interface

Instruction fetches, and data and debug accesses, to address ranges 0x20000000 to 0xDFFFFFFF and 0xE0100000 to 0xFFFFFFFF are performed over this 32-bit AHB-Lite bus.

For simultaneous accesses to this bus, the arbitration order in decreasing priority is:

- data accesses
- instruction and vector fetches
- debug.

The system bus interface contains control logic to handle unaligned accesses, FPB remapped accesses, bit-band accesses, and pipelined instruction fetches.

Private Peripheral Bus (PPB)

Data and debug accesses to external PPB space, 0xE0040000 to 0xE00FFFFF, are performed over this 32-bit *Advanced Peripheral Bus* (APB) bus. The *Trace Port Interface Unit* (TPIU) and vendor specific peripherals are on this bus.

Core data accesses have higher priority than debug accesses, so debug accesses are waited until core accesses have completed when there are simultaneous core and debug access to this bus. Only the address bits necessary to decode the External PPB space are supported on this interface.

The External PPB (EPPB) space, 0xE0040000 up to 0xE0100000, is intended for CoreSight-compatible debug and trace components, and has a number of irregular limitations which make it less useful for regular system peripherals. ARM recommends that system peripherals are placed in suitable Device type areas of the System bus address space, with use of an AHB2APB protocol converter for APB-based devices.

Limitations of the EPPB space are:

- it is accessible in privileged mode only
- it is accessed in little-endian fashion irrespective of the data endianness setting of the processor
- accesses behave as Strongly Ordered
- unaligned accesses have Unpredictable results
- only 32-bit data accesses are supported



it is accessible from the Debug Port and the local processor, but not from any other processor in the system.

2.2.2 ETM interface

The ETM interface enables simple connection of the ETM-M4 to the processor. It provides a channel for instruction trace to the ETM. See the *ARM Embedded Trace Macrocell Architecture Specification*.

2.2.3 AHB Trace Macrocell interface

The *AHB Trace Macrocell* (HTM) interface enables a simple connection of the AHB trace macrocell to the processor. It provides a channel for the data trace to the HTM.

Your implementation must include this interface to use the HTM interface. You must set TRCENA to 1 in the Debug Exception and Monitor Control Register (DEMCR) before you enable the HTM port to supply trace data. See the *ARMv7-M Architecture Reference Manual*.

2.2.4 Debug Port AHB-AP interface

The processor contains an *Advanced High-performance Bus Access Port* (AHB-AP) interface for debug accesses. An external *Debug Port* (DP) component accesses this interface. The Cortex-M4 system supports three possible DP implementations:

- The Serial Wire JTAG Debug Port (SWJ-DP). The SWJ-DP is a standard CoreSight debug port that combines JTAG-DP and Serial Wire Debug Port (SW-DP).
- The SW-DP. This provides a two-pin interface to the AHB-AP port.
- No DP present. If no debug functionality is present within the processor, a DP is not required.

The two DP implementations provide different mechanisms for debug access to the processor. Your implementation must contain only one of these components.

Note
Your implementation might contain an alternative implementer-specific DP instead of SW-DP
or SWJ-DP. See your implementer for details.

For more detailed information on the DP components, see the *CoreSight Components Technical Reference manual*.

For more information on the AHB-AP, see Chapter 8 Debug.

The DP and AP together are referred to as the *Debug Access Port* (DAP).

For more detailed information on the debug interface, see the *ARM Debug Interface v5 Architecture Specification*.



Chapter 3 **Programmers Model**

This chapter describes the processor programmers model. It contains the following sections:

- *About the programmers model* on page 3-2
- *Modes of operation and execution* on page 3-3
- Instruction set summary on page 3-4
- System address map on page 3-14
- Write buffer on page 3-17
- Bit-banding on page 3-19
- Processor core register summary on page 3-21
- *Exceptions* on page 3-23.



3.1 About the programmers model

The *ARMv7-M Architecture Reference Manual* provides a complete description of the programmers model. This chapter gives an overview of the Cortex-M4 processor programmers model that describes the implementation-defined options. It also contains the ARMv7-M Thumb instructions the model uses, and their cycle counts for the processor. In addition:

- Chapter 4 summarizes the system control features of the programmers model
- Chapter 5 summarizes the MPU features of the programmers model
- Chapter 6 summarizes the NVIC features of the programmers model
- Chapter 7 summarizes the FPU features of the programmers model
- Chapter 8 summarizes the Debug features of the programmers model
- Chapter 9 summarizes the DWT features of the programmers model
- Chapter 10 summarizes the ITM features of the programmers model
- Chapter 11 summarizes the TPIU features of the programmers model.





3.2 Modes of operation and execution

This section briefly describes the modes of operation and execution of the Cortex-M4 processor. See the *ARMv7-M Architecture Reference Manual* for more information.

3.2.1 Operating modes

The processor supports two modes of operation, Thread mode and Handler mode:

- The processor enters Thread mode on Reset, or as a result of an exception return. Privileged and Unprivileged code can run in Thread mode.
- The processor enters Handler mode as a result of an exception. All code is privileged in Handler mode.

3.2.2 Operating states

The processor can operate in one of two operating states:

- Thumb state. This is normal execution running 16-bit and 32-bit halfword aligned Thumb instructions.
- Debug State. This is the state when the processor is in halting debug.

3.2.3 Privileged access and user access

Code can execute as privileged or unprivileged. Unprivileged execution limits or excludes access to some resources. Privileged execution has access to all resources. Handler mode is always privileged. Thread mode can be privileged or unprivileged.



3.3 Instruction set summary

This section provides information on:

- Cortex-M4 instructions
- Load/store timings on page 3-11
- *Binary compatibility with other Cortex processors* on page 3-12.

3.3.1 Cortex-M4 instructions

The processor implements the ARMv7-M Thumb instruction set. Table 3-1 shows the Cortex-M4 instructions and their cycle counts. The cycle counts are based on a system with zero wait states.

Within the assembler syntax, depending on the operation, the <op2> field can be replaced with one of the following options:

- a simple register specifier, for example Rm
- an immediate shifted register, for example Rm, LSL #4
- a register shifted register, for example Rm, LSL Rs
- an immediate value, for example #0xE000E000.

For brevity, not all load and store addressing modes are shown. See the *ARMv7-M Architecture Reference Manual* for more information.

Table 3-1 uses the following abbreviations in the Cycles column:

- P The number of cycles required for a pipeline refill. This ranges from 1 to 3 depending on the alignment and width of the target instruction, and whether the processor manages to speculate the address early.
- B The number of cycles required to perform the barrier operation. For DSB and DMB, the minimum number of cycles is zero. For ISB, the minimum number of cycles is equivalent to the number required for a pipeline refill.
- N The number of registers in the register list to be loaded or stored, including PC or LR.
- W The number of cycles spent waiting for an appropriate event.

Table 3-1 Cortex-M4 instruction set summary

Operation	Description	Assembler	Cycles
Move	Register	MOV Rd, <op2></op2>	1
	16-bit immediate	MOVW Rd, # <imm></imm>	1
	Immediate into top	MOVT Rd, # <imm></imm>	1
	To PC	MOV PC, Rm	1 + P
Add	Add	ADD Rd, Rn, <op2></op2>	1
	Add to PC	ADD PC, PC, Rm	1 + P
	Add with carry	ADC Rd, Rn, <op2></op2>	1
	Form address	ADR Rd, <label></label>	1



Table 3-1 Cortex-M4 instruction set summary (continued)

	Table 3-1 Cortex-	M4 instruction set summar	y (continue
Operation	Description	Assembler	Cycles
Subtract	Subtract	SUB Rd, Rn, <op2></op2>	1
	Subtract with borrow	SBC Rd, Rn, <op2></op2>	1
	Reverse	RSB Rd, Rn, <op2></op2>	1
Multiply	Multiply	MUL Rd, Rn, Rm	1
	Multiply accumulate	MLA Rd, Rn, Rm	1
	Multiply subtract	MLS Rd, Rn, Rm	1
	Long signed	SMULL RdLo, RdHi, Rn, Rm	1
	Long unsigned	UMULL RdLo, RdHi, Rn, Rm	1
	Long signed accumulate	SMLAL RdLo, RdHi, Rn, Rm	1
	Long unsigned accumulate	UMLAL RdLo, RdHi, Rn, Rm	1
Divide	Signed	SDIV Rd, Rn, Rm	2 to 12a
	Unsigned	UDIV Rd, Rn, Rm	2 to 12a
Saturate	Signed	SSAT Rd, # <imm>, <op2></op2></imm>	1
	Unsigned	USAT Rd, # <imm>, <op2></op2></imm>	1
Compare	Compare	CMP Rn, <op2></op2>	1
	Negative	CMN Rn, <op2></op2>	1
Logical	AND	AND Rd, Rn, <op2></op2>	1
	Exclusive OR	EOR Rd, Rn, <op2></op2>	1
	OR	ORR Rd, Rn, <op2></op2>	1
	OR NOT	ORN Rd, Rn, <op2></op2>	1
	Bit clear	BIC Rd, Rn, <op2></op2>	1
	Move NOT	MVN Rd, <op2></op2>	1
	AND test	TST Rn, <op2></op2>	1
	Exclusive OR test	TEQ Rn, <op1></op1>	
Shift	Logical shift left	LSL Rd, Rn, # <imm></imm>	1
	Logical shift left	LSL Rd, Rn, Rs	1
	Logical shift right	LSR Rd, Rn, # <imm></imm>	1
	Logical shift right	LSR Rd, Rn, Rs	1
	Arithmetic shift right	ASR Rd, Rn, # <imm></imm>	1
	Arithmetic shift right	ASR Rd, Rn, Rs	1
Rotate	Rotate right	ROR Rd, Rn, # <imm></imm>	1
	Rotate right	ROR Rd, Rn, Rs	1
	With extension	RRX Rd, Rn	1



Table 3-1 Cortex-M4 instruction set summary (continued)

	lable 3-1 Cortex	c-M4 instruction set summar	y (continueu)
Operation	Description	Assembler	Cycles
Count	Leading zeroes	CLZ Rd, Rn	1
Load	Word	LDR Rd, [Rn, <op2>]</op2>	2 ^b
	To PC	LDR PC, [Rn, <op2>]</op2>	$2^b + P$
	Halfword	LDRH Rd, [Rn, <op2>]</op2>	2 ^b
	Byte	LDRB Rd, [Rn, <op2>]</op2>	2 ^b
	Signed halfword	LDRSH Rd, [Rn, <op2>]</op2>	2 ^b
	Signed byte	LDRSB Rd, [Rn, <op2>]</op2>	2 ^b
	User word	LDRT Rd, [Rn, # <imm>]</imm>	2 ^b
	User halfword	LDRHT Rd, [Rn, # <imm>]</imm>	2 ^b
	User byte	LDRBT Rd, [Rn, # <imm>]</imm>	2 ^b
	User signed halfword	LDRSHT Rd, [Rn, # <imm>]</imm>	2 ^b
	User signed byte	LDRSBT Rd, [Rn, # <imm>]</imm>	2 ^b
	PC relative	LDR Rd,[PC, # <imm>]</imm>	2 ^b
	Doubleword	LDRD Rd, Rd, [Rn, # <imm>]</imm>	1 + N
	Multiple	LDM Rn, { <reglist>}</reglist>	1 + N
	Multiple including PC	LDM Rn, { <reglist>, PC}</reglist>	1 + N + P
Store	Word	STR Rd, [Rn, <op2>]</op2>	2 ^b
	Halfword	STRH Rd, [Rn, <op2>]</op2>	2 ^b
	Byte	STRB Rd, [Rn, <op2>]</op2>	2 ^b
	Signed halfword	STRSH Rd, [Rn, <op2>]</op2>	2 ^b
	Signed byte	STRSB Rd, [Rn, <op2>]</op2>	2 ^b
	User word	STRT Rd, [Rn, # <imm>]</imm>	2 ^b
	User halfword	STRHT Rd, [Rn, # <imm>]</imm>	2 ^b
	User byte	STRBT Rd, [Rn, # <imm>]</imm>	2 ^b
	User signed halfword	STRSHT Rd, [Rn, # <imm>]</imm>	2 ^b
	User signed byte	STRSBT Rd, [Rn, # <imm>]</imm>	2 ^b
	Doubleword	STRD Rd, Rd, [Rn, # <imm>]</imm>	1 + N
	Multiple	STM Rn, { <reglist>}</reglist>	1 + N
Push	Push	PUSH { <reglist>}</reglist>	1 + N
	Push with link register	PUSH { <reglist>, LR}</reglist>	1 + N
Pop	Pop	POP { <reglist>}</reglist>	1 + N
	Pop and return	POP { <reglist>, PC}</reglist>	1 + N + P



Table 3-1 Cortex-M4 instruction set summary (continued)

	10010 0 1 001107 1	vi4 instruction set summary	
Operation	Description	Assembler	Cycles
Semaphore	Load exclusive	LDREX Rd, [Rn, # <imm>]</imm>	2
	Load exclusive half	LDREXH Rd, [Rn]	2
	Load exclusive byte	LDREXB Rd, [Rn]	2
	Store exclusive	STREX Rd, Rt, [Rn, # <imm>]</imm>	2
	Store exclusive half	STREXH Rd, Rt, [Rn]	2
	Store exclusive byte	STREXB Rd, Rt, [Rn]	2
	Clear exclusive monitor	CLREX	1
Branch	Conditional	B <cc> <label></label></cc>	1 or 1 + Pc
	Unconditional	B <label></label>	1 + P
	With link	BL <label></label>	1 + P
	With exchange	BX Rm	1 + P
	With link and exchange	BLX Rm	1 + P
	Branch if zero	CBZ Rn, <1abe1>	1 or 1 + Pc
	Branch if non-zero	CBNZ Rn, <1abe1>	1 or 1 + Pc
	Byte table branch	TBB [Rn, Rm]	2 + P
	Halfword table branch	TBH [Rn, Rm, LSL#1]	2 + P
State change	Supervisor call	SVC # <imm></imm>	-
	If-then-else	IT <cond></cond>	1 ^d
	Disable interrupts	CPSID <flags></flags>	1 or 2
	Enable interrupts	CPSIE <flags></flags>	1 or 2
	Read special register	MRS Rd, <specreg></specreg>	1 or 2
	Write special register	MSR <specreg>, Rn</specreg>	1 or 2
	Breakpoint	BKPT # <imm></imm>	-
Extend	Signed halfword to word	SXTH Rd, <op2></op2>	1
	Signed byte to word	SXTB Rd, <op2></op2>	1
	Unsigned halfword	UXTH Rd, <op2></op2>	1
	Unsigned byte	UXTB Rd, <op2></op2>	1
Bit field	Extract unsigned	UBFX Rd, Rn, # <imm>, #<imm></imm></imm>	1
	Extract signed	SBFX Rd, Rn, # <imm>, #<imm></imm></imm>	1
	Clear	BFC Rd, Rn, # <imm>, #<imm></imm></imm>	1
	Insert	BFI Rd, Rn, # <imm>, #<imm></imm></imm>	1



Table 3-1 Cortex-M4 instruction set summary (continued)

Operation	Description	Assembler	Cycles
Reverse	Bytes in word	REV Rd, Rm	1
	Bytes in both halfwords	REV16 Rd, Rm	1
	Signed bottom halfword	REVSH Rd, Rm	1
	Bits in word	RBIT Rd, Rm	1
Hint	Send event	SEV	1
	Wait for event	WFE	1 + W
	Wait for interrupt	WFI	1 + W
	No operation	NOP	1
Barriers	Instruction synchronization	ISB	1 + B
	Data memory	DMB	1 + B
	Data synchronization	DSB <flags></flags>	1 + B

- a. Division operations use early termination to minimize the number of cycles required based on the number of leading ones and zeroes in the input operands.
- b. Neighboring load and store single instructions can pipeline their address and data phases.

 This enables these instructions to complete in a single execution cycle.
- c. Conditional branch completes in a single cycle if the branch is not taken.
- d. An IT instruction can be folded onto a preceding 16-bit Thumb instruction, enabling execution in zero cycles.

Table 3-2 shows the DSP instructions that the Cortex-M4 processor implements.

Table 3-2 Cortex-M4 DSP instruction set summary

Operation	Description	Assembler	Cycles
Multiply	32-bit multiply with 32-most-significant-bit accumulate	SMMLA	1
	32-bit multiply with 32-most-significant-bit subtract	SMMLS	1
	32-bit multiply returning 32-most-significant-bits	SMMUL	1
	32-bit multiply with rounded 32-most-significant-bit accumulate	SMMLAR	1
	32-bit multiply with rounded 32-most-significant-bit subtract	SMMLSR	1
	32-bit multiply returning rounded 32-most-significant-bits	SMMULR	1



Table 3-2 Cortex-M4 DSP instruction set summary (continued

Operation	Description	Assembler	Cycles
Signed Multiply	Q setting 16-bit signed multiply with 32-bit accumulate, bottom by bottom	SMLABB	1
	Q setting 16-bit signed multiply with 32-bit accumulate, bottom by top	SMLABT	1
	16-bit signed multiply with 64-bit accumulate, bottom by bottom	SMLALBB	1
	16-bit signed multiply with 64-bit accumulate, bottom by top	SMLALBT	1
	Dual 16-bit signed multiply with single 64-bit accumulator	SMLALD{X}	1
	16-bit signed multiply with 64-bit accumulate, top by bottom	SMLALTB	1
	16-bit signed multiply with 64-bit accumulate, top by top	SMLALTT	1
	16-bit signed multiply yielding 32-bit result, bottom by bottom	SMULBB	1
	16-bit signed multiply yielding 32-bit result, bottom by top	SMULBT	1
	16-bit signed multiply yielding 32-bit result, top by bottom	SMULTB	1
	16-bit signed multiply yielding 32-bit result, top by bottom	SMULTT	1
	16-bit by 32-bit signed multiply returning 32-most-significant-bits, bottom	SMULWB	1
	16-bit by 32-bit signed multiply returning 32-most-significant-bits, top	SMULWT	1
	Dual 16-bit signed multiply returning difference	SMUSD{X}	1
	Q setting 16-bit signed multiply with 32-bit accumulate, top by bottom	SMLATB	1
	Q setting 16-bit signed multiply with 32-bit accumulate, top by top	SMLATT	1
	Q setting dual 16-bit signed multiply with single 32-bit accumulator	SMLAD{X}	1
	Q setting 16-bit by 32-bit signed multiply with 32-bit accumulate, bottom	SMLAWB	1
	Q setting 16-bit by 32-bit signed multiply with 32-bit accumulate, top	SMLAWT	1
	Q setting dual 16-bit signed multiply subtract with 32-bit accumulate	SMLSD{X}	1
	Q setting dual 16-bit signed multiply subtract with 64-bit accumulate	SMLSLD{X}	1
	Q setting sum of dual 16-bit signed multiply	SMUAD{X}	1
Unsigned Multiply	32-bit unsigned multiply with double 32-bit accumulation yielding 64-bit result	UMAAL	1
Saturate	Q setting dual 16-bit saturate	SSAT16	1
	Q setting dual 16-bit unsigned saturate	USAT16	1



Table 3-2 Cortex-M4 DSP instruction set summary (continued

Operation	Description	Assembler	Cycles
Packing and Unpacking	Pack half word top with shifted bottom	PKHTB	1
	Pack half word bottom with shifted top	PKHBT	1
	Extract 8-bits and sign extend to 32-bits	SXTB	1
	Dual extract 8-bits and sign extend each to 16-bits	SXTB16	1
	Extract 16-bits and sign extend to 32-bits	SXTH	1
	Extract 8-bits and zero-extend to 32-bits	UXTB	1
	Dual extract 8-bits and zero-extend to 16-bits	UXTB16	1
	Extract 16-bits and zero-extend to 32-bits	UXTH	1
	Extract 8-bit to 32-bit unsigned addition	UXTAB	1
	Dual extracted 8-bit to 16-bit unsigned addition	UXTAB16	1
	Extracted 16-bit to 32-bit unsigned addition	UXTAH	1
	Extracted 8-bit to 32-bit signed addition	SXTAB	1
	Dual extracted 8-bit to 16-bit signed addition	SXTAB16	1
	Extracted 16-bit to 32-bit signed addition	SXTAH	1
Miscellaneous	Select bytes based on GE bits	SEL	1
Data Processing	Unsigned sum of quad 8-bit unsigned absolute difference	USAD8	1
	Unsigned sum of quad 8-bit unsigned absolute difference with 32-bit accumulate	USADA8	1
Addition	Dual 16-bit unsigned saturating addition	UQADD16	1
	Quad 8-bit unsigned saturating addition	UQADD8	1
	Q setting saturating add	QADD	1
	Q setting dual 16-bit saturating add	QADD16	1
	Q setting quad 8-bit saturating add	QADD8	1
	Q setting saturating double and add	QDADD	1
	GE setting quad 8-bit signed addition	SADD8	1
	GE setting dual 16-bit signed addition	SADD16	1
	Dual 16-bit signed addition with halved results	SHADD16	1
	Quad 8-bit signed addition with halved results	SHADD8	1
	GE setting dual 16-bit unsigned addition	UADD16	1
	GE setting quad 8-bit unsigned addition	UADD8	1
	Dual 16-bit unsigned addition with halved results	UHADD16	1
	Quad 8-bit unsigned addition with halved results	UHADD8	1



Table 3-2 Cortex-M4 DSP instruction set summary (continued

Operation	Description	Assembler	Cycles
Subtraction	Q setting saturating double and subtract	QDSUB	1
	Dual 16-bit unsigned saturating subtraction	UQSUB16	1
	Quad 8-bit unsigned saturating subtraction	UQSUB8	1
	Q setting saturating subtract	QSUB	1
	Q setting dual 16-bit saturating subtract	QSUB16	1
	Q setting quad 8-bit saturating subtract	QSUB8	1
	Dual 16-bit signed subtraction with halved results	SHSUB16	1
	Quad 8-bit signed subtraction with halved results	SHSUB8	1
	GE setting dual 16-bit signed subtraction	SSUB16	1
	GE setting quad 8-bit signed subtraction	SSUB8	1
	Dual 16-bit unsigned subtraction with halved results	UHSUB16	1
	Quad 8-bit unsigned subtraction with halved results	UHSUB8	1
	GE setting dual 16-bit unsigned subtract	USUB16	1
	GE setting quad 8-bit unsigned subtract	USUB8	1
Parallel Addition and Subtraction	Dual 16-bit unsigned saturating addition and subtraction with exchange	UQASX	1
	Dual 16-bit unsigned saturating subtraction and addition with exchange	UQSAX	1
	GE setting dual 16-bit addition and subtraction with exchange	SASX	1
	Q setting dual 16-bit add and subtract with exchange	QASX	1
	Q setting dual 16-bit subtract and add with exchange	QSAX	1
	Dual 16-bit signed addition and subtraction with halved results	SHASX	1
	Dual 16-bit signed subtraction and addition with halved results	SHSAX	1
	GE setting dual 16-bit signed subtraction and addition with exchange	SSAX	1
	GE setting dual 16-bit unsigned addition and subtraction with exchange	UASX	1
	Dual 16-bit unsigned addition and subtraction with halved results and exchange	UHASX	1
	Dual 16-bit unsigned subtraction and addition with halved results and exchange	UHSAX	1
	GE setting dual 16-bit unsigned subtract and add with exchange	USAX	1

3.3.2 Load/store timings

This section describes how best to pair instructions to achieve more reductions in timing.

• STR Rx, [Ry, #imm] is always one cycle. This is because the address generation is performed in the initial cycle, and the data store is performed at the same time as the next instruction is executing. If the store is to the write buffer, and the write buffer is full or not enabled, the next instruction is delayed until the store can complete. If the store is not to the write buffer, for example to the Code segment, and that transaction stalls, the impact on timing is only felt if another load or store operation is executed before completion.



- LDR PC, [any] is always a blocking operation. This means at least two cycles for the load, and three cycles for the pipeline reload. So this operation takes at least five cycles, or more if stalled on the load or the fetch.
- Any load or store that generates an address dependent on the result of a preceding data processing operation will stall the pipeline for an additional cycle whilst the register bank is updated. There is no forwarding path for this scenario.
- LDR Rx, [PC, #imm] might add a cycle because of contention with the fetch unit.
- TBB and TBH are also blocking operations. These are at least two cycles for the load, one cycle for the add, and three cycles for the pipeline reload. This means at least six cycles, or more if stalled on the load or the fetch.
- LDR [any] are pipelined when possible. This means that if the next instruction is an LDR or STR, and the destination of the first LDR is not used to compute the address for the next instruction, then one cycle is removed from the cost of the next instruction. So, an LDR might be followed by an STR, so that the STR writes out what the LDR loaded. More multiple LDRs can be pipelined together. Some optimized examples are:
 - LDR R0, [R1]; LDR R1, [R2] normally three cycles total
 - LDR R0,[R1,R2]; STR R0,[R3,#20] normally three cycles total
 - LDR R0,[R1,R2]; STR R1,[R3,R2] normally three cycles total
 - LDR R0, [R1,R5]; LDR R1, [R2]; LDR R2, [R3,#4] normally four cycles total.
- Other instructions cannot be pipelined after STR with register offset. STR can only be pipelined when it follows an LDR, but nothing can be pipelined after the store. Even a stalled STR normally only takes two cycles, because of the write buffer.
- LDREX and STREX can be pipelined exactly as LDR. Because STREX is treated more like an LDR, it can be pipelined as explained for LDR. Equally LDREX is treated exactly as an LDR and so can be pipelined.
- LDRD and STRD cannot be pipelined with preceding or following instructions. However, the two words are pipelined together. So, this operation requires three cycles when not stalled.
- LDM and STM cannot be pipelined with preceding or following instructions. However, all elements after the first are pipelined together. So, a three element LDM takes 2+1+1 or 5 cycles when not stalled. Similarly, an eight element store takes nine cycles when not stalled. When interrupted, LDM and STM instructions continue from where they left off when returned to. The continue operation adds one or two cycles to the first element when started.
- Unaligned word or halfword loads or stores add penalty cycles. A byte aligned halfword load or store adds one extra cycle to perform the operation as two bytes. A halfword aligned word load or store adds one extra cycle to perform the operation as two halfwords. A byte-aligned word load or store adds two extra cycles to perform the operation as a byte, a halfword, and a byte. These numbers increase if the memory stalls. A STR or STRH cannot delay the processor because of the write buffer.

3.3.3 Binary compatibility with other Cortex processors

The processor implements a binary compatible subset of the instruction set and features provided by other Cortex-M profile processors. You can move software, including system level software, from the Cortex-M4 processor to other Cortex-M profile processors.



To ensure a smooth transition, ARM recommends that code designed to operate on other Cortex-M profile processor architectures obey the following rules and configure the *Configuration and Control Register* (CCR) appropriately:

- use word transfers only to access registers in the NVIC and System Control Space (SCS).
- treat all unused SCS registers and register fields on the processor as Do-Not-Modify.
- configure the following fields in the CCR:
 - STKALIGN bit to 1
 - UNALIGN_TRP bit to 1
 - Leave all other bits in the CCR register as their original value.





3.4 System address map

The processor contains a bus matrix that arbitrates the processor core and optional *Debug Access Port* (DAP) memory accesses to both the external memory system and to the internal *System Control Space* (SCS) and debug components.

Priority is always given to the processor to ensure that any debug accesses are as non-intrusive as possible. For a zero wait state system, all debug accesses to system memory, SCS, and debug resources are completely non-intrusive.

Figure 3-1 shows the system address map.

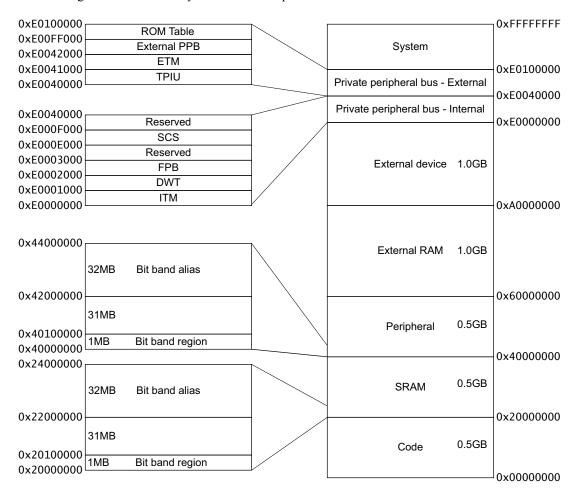


Figure 3-1 System address map

Table 3-3 shows the processor interfaces that are addressed by the different memory map regions.

Table 3-3 Memory regions

Memory Map	Region
Code	Instruction fetches are performed over the ICode bus. Data accesses are performed over the DCode bus.
SRAM	Instruction fetches and data accesses are performed over the system bus.
SRAM bit-band	Alias region. Data accesses are aliases. Instruction accesses are not aliases.

Table 3-3 Memory regions (continued

Memory Map	Region
Peripheral	Instruction fetches and data accesses are performed over the system bus.
Peripheral bit-band	Alias region. Data accesses are aliases. Instruction accesses are not aliases.
External RAM	Instruction fetches and data accesses are performed over the system bus.
External Device	Instruction fetches and data accesses are performed over the system bus.
Private Peripheral Bus	External and internal <i>Private Peripheral Bus</i> (PPB) interfaces. See <i>Private peripheral bus</i> . This memory region is <i>Execute Never</i> (XN), and so instruction fetches are prohibited. An MPU, if present, cannot change this.
System	System segment for vendor system peripherals. This memory region is XN, and so instruction fetches are prohibited. An MPU, if present, cannot change this.

See the ARMv7-M Architecture Reference Manual for more information about the memory model.

3.4.1 Private peripheral bus

The internal Private Peripheral Bus (PPB) interface provides access to:

- the *Instrumentation Trace Macrocell* (ITM)
- the Data Watchpoint and Trace (DWT)
- the Flashpatch and Breakpoint (FPB)
- the *System Control Space* (SCS), including the Memory Protection Unit (MPU) and the Nested Vectored Interrupt Controller (NVIC).

The external PPB interface provides access to:

- the Trace Point Interface Unit (TPIU)
- the *Embedded Trace Macrocell* (ETM)
- the ROM table.
- implementation-specific areas of the PPB memory map.

3.4.2 Unaligned accesses that cross regions

The Cortex-M4 processor supports ARMv7 unaligned accesses, and performs all accesses as single, unaligned accesses. They are converted into two or more aligned accesses by the DCode and System bus interfaces.

—— Note –				
All Cortex-M4	external	accesses	are	aligned

Unaligned support is only available for load/store singles (LDR, LDRH, STR, STRH). Load/store double already supports word aligned accesses, but does not permit other unaligned accesses, and generates a fault if this is attempted.



Unaligned accesses that cross memory map boundaries are architecturally Unpredictable. The processor behavior is boundary dependent, as follows:

- DCode accesses wrap within the region. For example, an unaligned halfword access to the last byte of Code space (0x1FFFFFFF) is converted by the DCode interface into a byte access to 0x1FFFFFFF followed by a byte access to 0x00000000.
- System accesses that cross into PPB space do not wrap within System space. For example, an unaligned halfword access to the last byte of System space (0xDFFFFFF) is converted by the System interface into a byte access to 0xDFFFFFFF followed by a byte access to 0xE0000000. 0xE0000000 is not a valid address on the System bus.
- Unaligned accesses are not supported to PPB space, and so there are no boundary crossing cases for PPB accesses.

Unaligned accesses that cross into the bit-band alias regions are also architecturally Unpredictable. The processor performs the access to the bit-band alias address, but this does not result in a bit-band operation. For example, an unaligned halfword access to 0x21FFFFFF is performed as a byte access to 0x21FFFFFF followed by a byte access to 0x22000000 (the first byte of the bit-band alias).

Unaligned loads that match against a literal comparator in the FPB are not remapped. FPB only remaps aligned addresses.



3.5 Write buffer

To prevent bus wait cycles from stalling the processor during data stores, buffered stores to the DCode and System buses go through a one-entry write buffer. If the write buffer is full, subsequent accesses to the bus stall until the write buffer has drained. The write buffer is only used if the bus waits the data phase of the buffered store, otherwise the transaction completes on the bus.

DMB and DSB instructions wait for the write buffer to drain before completing. If an interrupt comes in while DMB or DSB is waiting for the write buffer to drain, the processor returns to the instruction following the DMB or DSB after the interrupt completes. This is because interrupt processing acts as a memory barrier operation.





3.6 Exclusive monitor

The Cortex-M4 processor implements a local exclusive monitor. For more information about semaphores and the local exclusive monitor see the *ARMv7M ARM Architecture Reference Manual*.

The local monitor within the processor has been constructed so that it does not hold any physical address, but instead treats any access as matching the address of the previous LDREX. This means that the implemented exclusives reservation granule is the entire memory address range.

The Cortex-M4 processor does not support exclusive accesses to bit-band regions.





3.7 Bit-banding

Bit-banding is an optional feature of the Cortex-M4 processor. Bit-banding maps a complete word of memory onto a single bit in the bit-band region. For example, writing to one of the alias words sets or clears the corresponding bit in the bit-band region. This enables every individual bit in the bit-banding region to be directly accessible from a word-aligned address using a single LDR instruction. It also enables individual bits to be toggled without performing a read-modify-write sequence of instructions.

The processor memory map includes two bit-band regions. These occupy the lowest 1MB of the SRAM and Peripheral memory regions respectively. These bit-band regions map each word in an alias region of memory to a bit in a bit-band region of memory.

The System bus interface contains logic that controls bit-band accesses as follows:

- It remaps bit-band alias addresses to the bit-band region.
- For reads, it extracts the requested bit from the read byte, and returns this in the *Least Significant Bit* (LSB) of the read data returned to the core.
- For writes, it converts the write to an atomic read-modify-write operation.
- The processor does not stall during bit-band operations unless it attempts to access the System bus while the bit-band operation is being carried out.

The memory map has two 32-MB alias regions that map to two 1-MB bit-band regions:

- Accesses to the 32-MB SRAM alias region map to the 1-MB SRAM bit-band region.
- Accesses to the 32-MB peripheral alias region map to the 1-MB peripheral bit-band region.

A mapping formula shows how to reference each word in the alias region to a corresponding bit, or target bit, in the bit-band region. The mapping formula is:

```
bit\_word\_offset = (byte\_offset x 32) + (bit\_number \times 4)
```

bit_word_addr = bit_band_base + bit_word_offset

where:

- bit_word_offset is the position of the target bit in the bit-band memory region.
- bit_word_addr is the address of the word in the alias memory region that maps to the targeted bit.
- bit_band_base is the starting address of the alias region.
- byte_offset is the number of the byte in the bit-band region that contains the targeted bit.
- bit_number is the bit position, 0 to 7, of the targeted bit.

Figure 3-2 on page 3-20 shows examples of bit-band mapping between the SRAM bit-band alias region and the SRAM bit-band region:

- The alias word at 0x23FFFFE0 maps to bit [0] of the bit-band byte at 0x200FFFFF: 0x23FFFFE0 = 0x22000000 + (0xFFFFF*32) + 0*4.
- The alias word at 0x23FFFFFC maps to bit [7] of the bit-band byte at 0x200FFFFF: 0x23FFFFFC = 0x22000000 + (0xFFFFF*32) + 7*4.
- The alias word at 0x22000000 maps to bit [0] of the bit-band byte at 0x20000000: 0x22000000 = 0x22000000 + (0*32) + 0*4.



The alias word at 0x2200001C maps to bit [7] of the bit-band byte at 0x20000000: 0x2200001C = 0x22000000 + (0*32) + 7*4.

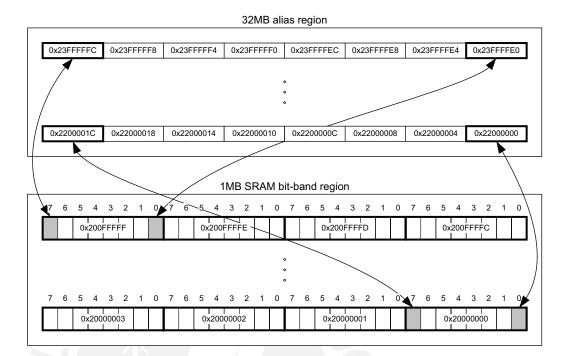


Figure 3-2 Bit-band mapping

3.7.1 Directly accessing an alias region

Writing to a word in the alias region has the same effect as a read-modify-write operation on the targeted bit in the bit-band region.

Bit [0] of the value written to a word in the alias region determines the value written to the targeted bit in the bit-band region. Writing a value with bit [0] set writes a 1 to the bit-band bit, and writing a value with bit [0] cleared writes a 0 to the bit-band bit.

Bits [31:1] of the alias word have no effect on the bit-band bit. Writing 0x01 has the same effect as writing 0x0F. Writing 0x00 has the same effect as writing 0x0E.

Reading a word in the alias region returns either 0x01 or 0x00. A value of 0x01 indicates that the targeted bit in the bit-band region is set. A value of 0x00 indicates that the targeted bit is clear. Bits [31:1] are zero.

3.7.2 Directly accessing a bit-band region

You can directly access the bit-band region with normal reads and writes to that region.



3.8 Processor core register summary

The processor has the following 32-bit registers:

- 13 general-purpose registers, R0-R12
- Stack Pointer (SP), R13 alias of banked registers, SP_process and SP_main
- Link Register (LR), R14
- Program Counter (PC), R15
- Special-purpose *Program Status Registers*, (xPSR).

Figure 3-3 shows the processor register set.

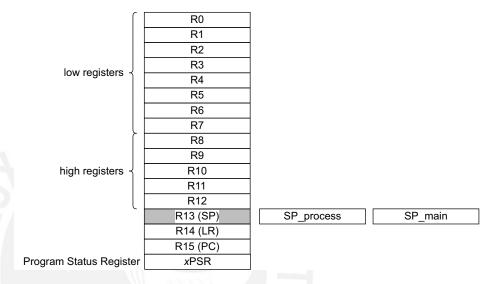


Figure 3-3 Processor register set

The general-purpose registers R0-R12 have no special architecturally-defined uses. Most instructions that can specify a general-purpose register can specify R0-R12.

Low registers Registers R0-R7 are accessible by all instructions that specify a

general-purpose register.

High registers Registers R8-R12 are accessible by all 32-bit instructions that specify a

general-purpose register.

Registers R8-R12 are not accessible by any 16-bit instructions.

Registers R13, R14, and R15 have the following special functions:

Stack pointer Register R13 is used as the *Stack Pointer* (SP). Because the SP ignores

writes to bits [1:0], it is autoaligned to a word, four-byte boundary.

Handler mode always uses SP_main, but you can configure Thread mode

to use either SP_main or SP_process.

Link register Register R14 is the subroutine *Link Register* (LR).

The LR receives the return address from PC when a *Branch and Link* (BL)

or Branch and Link with Exchange (BLX) instruction is executed.

The LR is also used for exception return.

At all other times, you can treat R14 as a general-purpose register.

Program counter Register R15 is the *Program Counter* (PC).



Bit [0] is always 0, so instructions are always aligned to word or halfword boundaries.

See the ARMv7-M Architecture Reference Manual for more information.





3.9 Exceptions

The processor and the *Nested Vectored Interrupt Controller* (NVIC) prioritize and handle all exceptions. When handling exceptions:

- All exceptions are handled in Handler mode.
- Processor state is automatically stored to the stack on an exception, and automatically restored from the stack at the end of the *Interrupt Service Routine* (ISR).
- The vector is fetched in parallel to the state saving, enabling efficient interrupt entry.

The processor supports tail-chaining that enables back-to-back interrupts without the overhead of state saving and restoration.

You configure the number of interrupts, and bits of interrupt priority, during implementation. Software can choose only to enable a subset of the configured number of interrupts, and can choose how many bits of the configured priorities to use.

Note -	
--------	--

Vector table entries are compatible with interworking between ARM and Thumb instructions. This causes bit [0] of the vector value to load into the *Execution Program Status Register* (EPSR) T-bit on exception entry. All populated vectors in the vector table entries must have bit [0] set. Creating a table entry with bit [0] clear generates an INVSTATE fault on the first instruction of the handler corresponding to this vector.

3.9.1 Exception handling

The processor implements advanced exception and interrupt handling, as described in the *ARMv7-M Architecture Reference Manual*.

To reduce interrupt latency, the processor implements both interrupt late-arrival and interrupt tail-chaining mechanisms, as defined by the ARMv7-M architecture:

- There is a maximum of a twelve cycle latency from asserting the interrupt to execution of the first instruction of the ISR when the memory being accessed has no wait states being applied. The first instruction to be executed is fetched in parallel to the stack push.
- Returns from interrupts similarly take twelve cycles where the instruction being returned to is fetched in parallel to the stack pop.
- Tail chaining requires six cycles when using zero wait state memory. No stack pushes or pops are performed and only the instruction for the next ISR is fetched.

The processor exception model has the following implementation-defined behavior in addition to the architecturally defined behavior:

- exceptions on stacking from HardFault to NMI lockup at NMI priority
- exceptions on unstacking from NMI to HardFault lockup at HardFault priority.

To minimize interrupt latency, the processor abandons any divide instruction to take any pending interrupt. On return from the interrupt handler, the processor restarts the divide instruction from the beginning The processor implements the Interruptible-continuable Instruction field. Load multiple (LDM) operations and store multiple (STM) operations are interruptible. The EPSR holds the information required to continue the load or store multiple from the point where the interrupt occurred.



This means that software must not use load-multiple or store-multiple instructions to access a device or access a memory region that is read-sensitive or sensitive to repeated writes. The software must not use these instructions in any case where repeated reads or writes might cause inconsistent results or unwanted side-effects.

Base register update in LDM and STM operations

There are cases when an LDM or STM updates the base register:

- When the instruction specifies base register write-back, the base register changes to the updated address. An abort restores the original base value.
- When the base register is in the register list of an LDM, and is not the last register in the list, the base register changes to the loaded value.

An LDM or STM is restarted rather than continued if:

- the instruction faults
- the instruction is inside an IT.

If an LDM has completed a base load, it is continued from before the base load.





Chapter 4 **System Control**

This chapter describes the registers that program the processor. It contains the following sections:

- About system control on page 4-2
- Register summary on page 4-3
- Register descriptions on page 4-5.



4.1 About system control

This chapter describes the registers that control the operation of the processor.





4.2 Register summary

Table 4-1 shows the system control registers. Registers not described in this chapter are described in the *ARMv7-M Architecture Reference Manual*

Table 4-1 System control registers

Address	Name	Type	Reset	Description
0xE000E008	ACTLR	RW	0x00000000	Auxiliary Control Register, ACTLR on page 4-5
0xE000E010	STCSR	RW	0×00000000	SysTick Control and Status Register
0xE000E014	STRVR	RW	Unknown	SysTick Reload Value Register
0xE000E018	STCVR	RW clear	Unknown	SysTick Current Value Register
0xE000E01C	STCR	RO	STCALIB	SysTick Calibration Value Register
0xE000ED00	CPUID	RO	0x410FC241	CPUID Base Register, CPUID on page 4-5
0xE000ED04	ICSR	RW or RO	0×00000000	Interrupt Control and State Register
0xE000ED08	VTOR	RW	0x00000000	Vector Table Offset Register
0xE000ED0C	AIRCR	RW	0x00000000a	Application Interrupt and Reset Control Register
0xE000ED10	SCR	RW	0×00000000	System Control Register
0xE000ED14	CCR	RW	0x00000200	Configuration and Control Register.
0xE000ED18	SHPR1	RW	0×00000000	System Handler Priority Register 1
0xE000ED1C	SHPR2	RW	0×00000000	System Handler Priority Register 2
0xE000ED20	SHPR3	RW	0x00000000	System Handler Priority Register 3
0xE000ED24	SHCSR	RW	0×00000000	System Handler Control and State Register
0xE000ED28	CFSR	RW	0×00000000	Configurable Fault Status Registers
0xE000ED2C	HFSR	RW	0x00000000	HardFault Status Register
0xE000ED30	DFSR	RW	0×00000000	Debug Fault Status Register
0xE000ED34	MMFAR	RW	Unknown	MemManage Fault Address Register ^b
0xE000ED38	BFAR	RW	Unknown	BusFault Address Register ^b
0xE000ED3C	AFSR	RW	0×00000000	Auxiliary Fault Status Register, AFSR on page 4-6
0xE000ED40	ID_PFR0	RO	0x00000030	Processor Feature Register 0
0xE000ED44	ID_PFR1	RO	0x00000200	Processor Feature Register 1
0xE000ED48	ID_DFR0	RO	0x00100000	Debug Features Register 0 ^c
0xE000ED4C	ID_AFR0	RO	0×00000000	Auxiliary Features Register 0
0xE000ED50	ID_MMFR0	RO	0x00100030	Memory Model Feature Register 0
0xE000ED54	ID_MMFR1	RO	0x00000000	Memory Model Feature Register 1
0xE000ED58	ID_MMFR2	RO	0x01000000	Memory Model Feature Register 2
0xE000ED5C	ID_MMFR3	RO	0x00000000	Memory Model Feature Register 3
0xE000ED60	ID_ISAR0	RO	0x01141110	Instruction Set Attributes Register 0



Address	Name	Туре	Reset	Description
0xE000ED64	ID_ISAR1	RO	0x02112000	Instruction Set Attributes Register 1
0xE000ED68	ID_ISAR2	RO	0x21232231	Instruction Set Attributes Register 2
0xE000ED6C	ID_ISAR3	RO	0x01111131	Instruction Set Attributes Register 3
0xE000ED70	ID_ISAR4	RO	0x01310132	Instruction Set Attributes Register 4
0xE000ED88	CPACR	RW	-	Coprocessor Access Control Register
0xE000EF00	STIR	WO	0x00000000	Software Triggered Interrupt Register

- a. Bits [10:8] are reset to zero. The ENDIANNESS bit, bit [15], can reset to either state, depending on the implementation.

 BFAR and MMFAR are the same physical register. Because of this, the BFARVALID and MMFARVALID bits are
- mutually exclusive.
- c. ID_DFR0 will read as 0 if no debug support is implemented.





4.3 Register descriptions

This section describes the system control registers whose implementation is specific to this processor.

4.3.1 Auxiliary Control Register, ACTLR

The ACTLR characteristics are:

Purpose Disables certain aspects of functionality within the processor.

Usage Constraints There are no usage constraints.

Configurations This register is available in all processor configurations.

Attributes See the register summary in Table 4-1 on page 4-3.

Figure 4-1 shows the ACTLR bit assignments.

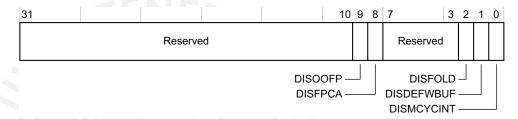


Figure 4-1 ACTLR bit assignments

Table 4-2 shows the ACTLR bit assignments.

Table 4-2 ACTLR bit assignments

Bits	Name	Function
[31:10]	-	Reserved
[9]	DISOOFP	Disables floating point instructions completing out of order with respect to integer instructions.
[8]	DISFPCA	Disable automatic update of CONTROL.FPCA. See <i>Exceptions</i> on page 7-8 for more information.
[7:3]	-	Reserved
[2]	DISFOLD	Disables folding of IT instructions.
[1]	DISDEFWBUF	Disables write buffer use during default memory map accesses. This causes all bus faults to be precise, but decreases the performance of the processor because stores to memory must complete before the next instruction can be executed.
[0]	DISMCYCINT	Disables interruption of multi-cycle instructions. This increases the interrupt latency of the processor because load/store and multiply/divide operations complete before interrupt stacking occurs.

4.3.2 CPUID Base Register, CPUID

The CPUID characteristics are:

Purpose Specifies:

• the ID number of the processor core



the version number of the processor core

• the implementation details of the processor core.

Usage Constraints There are no usage constraints.

Configurations This register is available in all processor configurations.

Attributes See the register summary in Table 4-1 on page 4-3.

Figure 4-2 shows the CPUID bit assignments.

31	24	23 20	19 16	15	4	1 3 0
	IMPLEMENTER	VARIANT	(Constant)	PAR	ΓΝΟ	REVISION

Figure 4-2 CPUID bit assignments

Table 4-3 shows the CPUID bit assignments.

Table 4-3 CPUID bit assignments

Bits NAME		Function			
[31:24]	IMPLEMENTER	Indicates implementer: 0x41 = ARM			
[23:20]	VARIANT	Indicates processor revision: 0x0 = Revision 0			
[19:16]	(Constant)	Reads as 0xF			
[15:4]	PARTNO	Indicates part number: 0xC24 = Cortex-M4			
[3:0]	REVISION	Indicates patch release: 0x1= Patch 1.			

4.3.3 Auxiliary Fault Status Register, AFSR

The AFSR characteristics are:

Purpose Specifies additional system fault information to software.

Usage Constraints The AFSR flags map directly onto the AUXFAULT inputs of the

processor, and a single-cycle high level on an external pin causes the corresponding AFSR bit to become latched as one. The bit can only be

cleared by writing a one to the corresponding AFSR bit.

When an AFSR bit is written or latched as one, an exception does not occur. To make use of AUXFAULT input signals, software must poll the

AFSR.

Configurations This register is available in all processor configurations.

Attributes See the register summary in Table 4-1 on page 4-3.

Figure 4-3 shows the AFSR bit assignments.



Figure 4-3 AFSR bit assignments



Table 4-4 shows the AFSR bit assignments.

Table 4-4 AFSR bit assignments

Bits	Name	Function
[31:0]	AUXFAULT	Latched version of the AUXFAULT inputs.





Chapter 5 **Memory Protection Unit**

This chapter describes the processor *Memory Protection Unit* (MPU). It contains the following sections:

- About the MPU on page 5-2
- MPU functional description on page 5-3
- *MPU programmers model* on page 5-4.



5.1 About the MPU

The MPU is an optional component for memory protection. The processor supports the standard ARMv7 *Protected Memory System Architecture* model. The MPU provides full support for:

- protection regions
- overlapping protection regions, with ascending region priority:
 - 7 = highest priority
 - -- 0 = lowest priority.
- · access permissions
- exporting memory attributes to the system.

MPU mismatches and permission violations invoke the programmable-priority MemManage fault handler. See the *ARMv7-M Architecture Reference Manual* for more information.

You can use the MPU to:

- enforce privilege rules
- separate processes
- enforce access rules.





5.2 MPU functional description

The access permission bits, TEX, C, B, AP, and XN, of the Region Access Control Register control access to the corresponding memory region. If an access is made to an area of memory without the required permissions, a permission fault is raised. For more information, see the *ARMv7-M Architecture Reference Manual*.





5.3 MPU programmers model

Table 5-5 shows the MPU registers. These registers are described in the *ARMv7-M Architecture Reference Manual*.

Table 5-1 MPU registers

Address	Name	Type	Reset	Description
0xE000ED90	MPU_TYPE	RO	0x00000800	MPU Type Register
0xE000ED94	MPU_CTRL	RW	0x00000000	MPU Control Register
0xE000ED98	MPU_RNR	RW	0x00000000	MPU Region Number Register
0xE000ED9C	MPU_RBAR	RW	0x00000000	MPU Region Base Address Register
0xE000EDA0	MPU_RASR	RW	0x00000000	MPU Region Attribute and Size Register
0xE000EDA4	MPU_RBAR_A1	NICA	0x00000000	MPU alias registers
0xE000EDA8	MPU_RASR_A1	NER	0x00000000	-
0xE000EDAC	MPU_RBAR_A2		0×00000000	-
0xE000EDB0	MPU_RASR_A2	-	0x00000000	
0xE000EDB4	MPU_RBAR_A3		0x00000000	
0xE000EDB8	MPU_RASR_A3		0×00000000	



Chapter 6 Nested Vectored Interrupt Controller

This chapter describes the *Nested Vectored Interrupt Controller* (NVIC). It contains the following sections:

- *About the NVIC* on page 6-2
- NVIC functional description on page 6-3
- *NVIC programmers model* on page 6-4.



6.1 About the NVIC

The NVIC provides configurable interrupt handling abilities to the processor. It:

- facilitates low-latency exception and interrupt handling
- controls power management.





6.2 NVIC functional description

The NVIC supports up to 240 interrupts each with up to 256 levels of priority. You can change the priority of an interrupt dynamically. The NVIC and the processor core interface are closely coupled, to enable low latency interrupt processing and efficient processing of late arriving interrupts. The NVIC maintains knowledge of the stacked, or nested, interrupts to enable tail-chaining of interrupts.

You can only fully access the NVIC from privileged mode, but you can cause interrupts to enter a pending state in user mode if you enable the Configuration and Control Register. Any other user mode access causes a bus fault.

You can access all NVIC registers using byte, halfword, and word accesses unless otherwise stated. NVIC registers are located within the SCS.

All NVIC registers and system debug registers are little-endian regardless of the endianness state of the processor.

Processor exception handling is described in *Exceptions* on page 3-23.

6.2.1 Low power modes

Your implementation can include a *Wake-up Interrupt Controller* (WIC). This enables the processor and NVIC to be put into a very low-power sleep mode leaving the WIC to identify and prioritize interrupts.

The processor fully implements the *Wait For Interrupt* (WFI), *Wait For Event* (WFE) and the *Send Event* (SEV) instructions. In addition, the processor also supports the use of SLEEPONEXIT, that causes the processor core to enter sleep mode when it returns from an exception handler to Thread mode. See the *ARMv7-M Architecture Reference Manual* for more information.

6.2.2 Level versus pulse interrupts

The processor supports both level and pulse interrupts. A level interrupt is held asserted until it is cleared by the ISR accessing the device. A pulse interrupt is a variant of an edge model. You must ensure that the pulse is sampled on the rising edge of the Cortex-M4 clock, **FCLK**, instead of being asynchronous.

For level interrupts, if the signal is not deasserted before the return from the interrupt routine, the interrupt again enters the pending state and re-activates. This is particularly useful for FIFO and buffer-based devices because it ensures that they drain either by a single ISR or by repeated invocations, with no extra work. This means that the device holds the signal in assert until the device is empty.

A pulse interrupt can be reasserted during the ISR so that the interrupt can be in the pending state and active at the same time. If another pulse arrives while the interrupt is still pending, the interrupt will remain pending and the ISR will only run once.

Pulse interrupts are mostly used for external signals and for rate or repeat signals.



6.3 NVIC programmers model

Table 6-1 shows the NVIC registers.

Table 6-1 NVIC registers

Address	Name	Туре	Reset	Description
0xE000E004	ICTR	RO	-	Interrupt Controller Type Register, ICTR
0xE000E100 - 0xE000E11C	NVIC_ISER0 - NVIC_ISER7	RW	0x00000000	Interrupt Set-Enable Registers
0xE000E180 - 0E000xE19C	NVIC_ICER0 - NVIC_ICER7	RW	0x00000000	Interrupt Clear-Enable Registers
0xE000E200 - 0xE000E21C	NVIC_ISPR0 - NVIC_ISPR7	RW	0x00000000	Interrupt Set-Pending Registers
0xE000E280 - 0xE000E29C	NVIC_ICPR0 - NVIC_ICPR7	RW	0x00000000	Interrupt Clear-Pending Registers
0xE000E300 - 0xE000E31C	NVIC_IABR0- NVIC_IABR7	RO	0x00000000	Interrupt Active Bit Register
0xE000E400 - 0xE000E4EC	NVIC_IPR0 - NVIC_IPR59	RW	0×00000000	Interrupt Priority Register

The following sections describe the NVIC registers whose implementation is specific to this processor. Other registers are described in the *ARMv7M Architecture Reference Manual*.

6.3.1 Interrupt Controller Type Register, ICTR

The ICTR characteristics are:

Purpose Shows the number of interrupt lines that the NVIC supports.

Usage Constraints There are no usage constraints.

Configurations This register is available in all processor configurations.

Attributes See the register summary in Table 6-1.

Figure 6-1 shows the ICTR bit assignments.

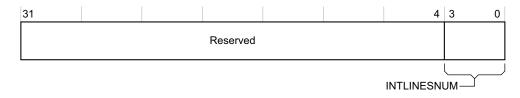


Figure 6-1 ICTR bit assignments



Table 6-2 shows the ICTR bit assignments.

Table 6-2 ICTR bit assignments

Bits	Name	Function
[31:4]	-	Reserved.
[3:0]	INTLINESNUM	Total number of interrupt lines in groups of 32: b0000 = 032 b0001 = 3364 b0010 = 6596 b0011 = 97128 b0100 = 129160 b0101 = 161192 b0110 = 193224 b0111 = 225256a

a. The processor supports a maximum of 240 external interrupts.





Chapter 7 Floating Point Unit

This chapter describes the programmers model of the *Floating Point Unit* (FPU). It contains the following sections:

- *About the FPU* on page 7-2
- FPU Functional Description on page 7-3
- FPU Programmers Model on page 7-9.

The Cortex-M4F processor is a Cortex-M4 processor that includes the optional FPU. In this chapter, the generic term *processor* means only the Cortex-M4F processor.



7.1 About the FPU

The Cortex-M4 FPU is an implementation of the single precision variant of the ARMv7-M Floating-Point Extension (FPv4-SP). It provides floating-point computation functionality that is compliant with the *ANSI/IEEE Std 754-2008, IEEE Standard for Binary Floating-Point Arithmetic*, referred to as the IEEE 754 standard. The FPU supports all single-precision data-processing instructions and data types described in the *ARM Architecture Reference Manual*.





7.2 FPU Functional Description

The FPU fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions.

The FPU functional description includes the following topics:

- FPU views of the register bank
- Modes of operation
- FPU instruction set on page 7-4
- Compliance with the IEEE 754 standard on page 7-6
- Complete implementation of the IEEE 754 standard on page 7-6
- *IEEE 754 standard implementation choices* on page 7-6
- Exceptions on page 7-8
- Enabling the FPU on page 7-9.

7.2.1 FPU views of the register bank

The FPU provides an extension register file containing 32 single-precision registers. These can be viewed as:

- sixteen 64-bit doubleword registers, D0-D15
- thirty-two 32-bit single-word registers, \$0-\$31
- a combination of registers from the above views:

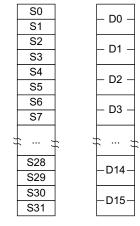


Figure 7-1 FPU register bank

The mapping between the registers is as follows:

- S<2n> maps to the least significant half of D<n>
- S<2n+1> maps to the most significant half of D<n>.

For example, you can access the least significant half of the value in D6 by accessing S12, and the most significant half of the elements by accessing S13.

7.2.2 Modes of operation

The FPU provides three modes of operation to accommodate a variety of applications:

- Full-compliance mode on page 7-4
- Flush-to-zero mode on page 7-4
- Default NaN mode on page 7-4.



Full-compliance mode

In full-compliance mode, the FPU processes all operations according to the IEEE 754 standard in hardware.

Flush-to-zero mode

Setting the FZ bit of the *Floating -point Staus and Control Register* FPSCR[24], enables flush-to-zero mode. In this mode, the FPU treats all subnormal input operands of arithmetic CDP operations as zeros in the operation. Exceptions that result from a zero operand are signaled appropriately. VABS, VNEG, and VMOV are not considered arithmetic CDP operations and are not affected by flush-to-zero mode. A result that is *tiny*, as described in the IEEE 754 standard, where the destination precision is smaller in magnitude than the minimum normal value *before rounding*, is replaced with a zero. The IDC flag, FPSCR[7], indicates when an input flush occurs. The UFC flag, FPSCR[3], indicates when a result flush occurs.

Default NaN mode

Setting the DN bit, FPSCR[25], enables default NaN mode. In this mode, the result of any arithmetic data processing operation that involves an input NaN, or that generates a NaN result, returns the default NaN. Propagation of the fraction bits is maintained only by VABS, VNEG, and VMOV operations. All other CDP operations ignore any information in the fraction bits of an input NaN.

7.2.3 FPU instruction set

Table 7-1 shows the instruction set of the FPU.

Table 7-1 FPU instruction set

Operation	Description	Assembler	Cycles	
Absolute value	of float	VABS.F32	1	
Addition	floating point	VADD.F32	1	
Compare	float with register or zero	VCMP.F32	1	
	float with register or zero	VCMPE.F32	1	
Convert	between integer, fixed-point, half-precision and float	VCVT.F32	1	
Divide	Floating-point	VDIV.F32	14	
Load	multiple doubles	VLDM.64	1+2*N, where N is the number of doubles.	
	multiple floats	VLDM.32	1+N, where N is the number of floats.	
	single double	VLDR.64	3	
	single float	VLDR.32	2	



Table 7-1 FPU instruction set (continued

Operation	Description	Assembler	Cycles	
Move	top/bottom half of double to/from core register	VMOV	1	
	immediate/float to float-register	VMOV	1	
	two floats/one double to/from two core registers or one float to/from one core register	VMOV	2	
	floating-point control/status to core register	VMRS	1	
	core register to floating-point control/status	VMSR	1	
Multiply	float	VMUL.F32	1	
	then accumulate float	VMLA.F32	3	
	then subtract float	VMLS.F32	3	
	then accumulate then negate float	VNMLA.F32	3	
	then subtract then negate float	VNMLS.F32	3	
Multiply	then accumulate float	VFMA.F32	3	
(fused)	then subtract float	VFMS.F32	3	
	then accumulate then negate float	VFNMA.F32	3	
	then subtract then negate float	VFNMS.F32	3	
Negate	float	VNEG.F32	1	
	and multiply float	VNMUL.F32	1	
Pop	double registers from stack	VPOP.64	1+2*N, where N is the number of double registers.	
	float registers from stack	VPOP.32	1+N where N is the number of registers.	
Push	double registers to stack	VPUSH.64	1+2*N, where N is the number of double registers.	
	float registers to stack	VPUSH.32	1+N, where N is the number of registers.	
Square-root	of float	VSQRT.F32	14	
Store	multiple double registers	VSTM.64	1+2*N, where N is the number of doubles.	
	multiple float registers	VSTM.32	1+N, where N is the number of floats.	
	single double register	VSTR.64	3	
	single float registers	VSTR.32	2	
Subtract	float	VSUB.F32	1	



– Note –

- Integer-only instructions following VDIVR or VSQRT instructions complete out-of-order.
 VDIV and VSQRT instructions take one cycle if no further floating-point instructions are executed.
- Floating-point arithmetic data processing instructions, such as add, subtract, multiply, divide, square-root, all forms of multiply with accumulate, as well as conversions of all types take one cycle longer if their result is consumed by the following instruction.
- Both fused and chained multiply with accumulate instructions consume their addend one cycle later, so the result of an arithmetic instruction that is followed by a multiply with accumulate instruction is consumed as the addend of the MAC instruction.

7.2.4 Compliance with the IEEE 754 standard

When *Default NaN* (DN) and *Flush-to-Zero* (FZ) modes are disabled, FPv4 functionality is compliant with the IEEE 754 standard in hardware. No support code is required to achieve this compliance.

See the *ARM Architecture Reference Manual* for information about FP architecture compliance with the IEEE 754 standard.

7.2.5 Complete implementation of the IEEE 754 standard

The Cortex-M4F floating point instruction set does not support all operations defined in the IEEE 754-2008 standard. Unsupported operations include, but are not limited to the following:

- remainder
- round floating-point number to integer-valued floating-point number
- binary-to-decimal conversions
- decimal-to-binary conversions
- direct comparison of single-precision and double-precision values.

The Cortex-M4 FPU supports fused MAC operations as described in the IEEE standard. For complete implementation of the IEEE 754-2008 standard, floating-point functionality must be augmented with library functions.

7.2.6 IEEE 754 standard implementation choices

Some of the implementation choices permitted by the IEEE 754-2008 standard and used in the FPv4 architecture are described in the *ARM Architecture Reference Manual*.

NaN handling

All single-precision values with the maximum exponent field value and a nonzero fraction field are valid NaNs. A most significant fraction bit of zero indicates a *Signaling NaN* (SNaN). A one indicates a *Quiet NaN* (QNaN). Two NaN values are treated as different NaNs if they differ in any bit. Table 7-2 shows the default NaN values.

Table 7-2 Default NaN values

Sign	Fraction	Fraction	
0	00xFF	bit [22] = 1, bits [21:0] are all zeros	



Processing of input NaNs for ARM floating-point functionality and libraries is defined a follows:

- In full-compliance mode, NaNs are handled as described in the *ARM Architecture Reference Manual*. The hardware processes the NaNs directly for arithmetic CDP instructions. For data transfer operations, NaNs are transferred without raising the Invalid Operation exception. For the non-arithmetic CDP instructions, VABS, VNEG, and VMOV, NaNs are copied, with a change of sign if specified in the instructions, without causing the Invalid Operation exception.
- In default NaN mode, arithmetic CDP instructions involving NaN operands return the default NaN regardless of the fractions of any NaN operands. SNaNs in an arithmetic CDP operation set the IOC flag, FPSCR[0]. NaN handling by data transfer and non-arithmetic CDP instructions is the same as in full-compliance mode.

Table 7-3 summarizes the effects of NaN operands on instruction execution.

Table 7-3 QNaN and SNaN handling

Instruction Default type NaN mode		With QNaN operand	With SNaN operand	
Arithmetic CDP	Off	The QNaN or one of the QNaN operands, if there is more than one, is returned according to the rules given in the ARM Architecture Reference Manual.	IOC ^a set. The SNaN is quieted and the result NaN is determined by the rules given in the <i>ARM Architecture</i> Reference Manual.	
	On	Default NaN returns.	IOCa set. Default NaN returns.	
Non-arithmetic	Off	NaN passes to destination with sign changed as appropriate.		
CDP	On			
FCMP(Z)		Unordered compare.	IOC set. Unordered compare.	
FCMPE(Z)	. _	IOC set. Unordered compare. IOC set. Unordered com		
Load/store	Off	All NaNa tua	and formed	
Load/store	On	- All NaNs transferred.		

a. IOC is the Invalid Operation exception flag, FPSCR[0].

Comparisons

Comparison results modify the flags in the FPSCR. You can use the MVRS APSR_nzcv instruction (formerly FMSTAT) to transfer the current flags from the FPSCR to the APSR. See the *ARM Architecture Reference Manual* for mapping of IEEE 754-2008 standard predicates to ARM conditions. The flags used are chosen so that subsequent conditional execution of ARM instructions can test the predicates defined in the IEEE standard.

Underflow

The Cortex-M4F FPU uses the *before rounding* form of *tininess* and the *inexact result* form of *loss of accuracy* as described in the IEEE 754-2008 standard to generate Underflow exceptions.

In flush-to-zero mode, results that are tiny before rounding, as described in the IEEE standard, are flushed to a zero, and the UFC flag, FPSCR[3], is set. See the *ARM Architecture Reference Manual* for information on flush-to-zero mode.



When the FPU is not in flush-to-zero mode, operations are performed on subnormal operands. If the operation does not produce a tiny result, it returns the computed result, and the UFC flag, FPSCR[3], is not set. The IXC flag, FPSCR[4], is set if the operation is inexact. If the operation produces a tiny result, the result is a subnormal or zero value, and the UFC flag, FPSCR[3], is set if the result was also inexact.

7.2.7 Exceptions

The FPU sets the cumulative exception status flag in the FPSCR register as required for each instruction, in accordance with the FPv4 architecture. The FPU does not support user-mode traps. The exception enable bits in the FPSCR read-as-zero, and writes are ignored. The processor also has six output pins, **FPIXC**, **FPUFC**, **FPOFC**, **FPDZC**, **FPIDC**, and **FPIOC**, that each reflect the status of one of the cumulative exception flags. See the *Cortex-M4 Integration and Implementation Manual* for a description of these outputs.

The processor can reduce the exception latency by using lazy stacking. See *Auxiliary Control Register*, *ACTLR* on page 4-5. This means that the processor reserves space on the stack for the FP state, but does not save that state information to the stack. See the *ARMv7-M Architecture Reference Manual* for more information.





7.3 FPU Programmers Model

Table 7-4 shows the FP system registers in the Cortex-M4F FPU.

Table 7-4 Cortex-M4F Floating Point system registers

Address	Name	Туре	Reset	Description
0xE000EF34	FPCCR	RW	0xC0000000	FP Context Control Register
0xE000EF38	FPCAR	RW	-	FP Context Address Register
0xE000EF3C	FPDSCR	RW	0x00000000	FP Default Status Control Register
0xE000EF40	MVFR0	RO	0x10110021	Media and VFP Feature Register 0, MVFR0
0xE000EF44	MVFR1	RO	0x11000011	Media and VFP Feature Register 1, MVFR1

All Cortex-M4F FPU registers are described in the ARMv7-M Architecture Reference Manual.

7.3.1 Enabling the FPU

Example 7-1 shows an example code sequence for enabling the FPU in both privileged and user modes. The processor must be in privileged mode to read from and write to the CPACR.

Example 7-1 Enabling the FPU

```
; CPACR is located at address 0xE000ED88 LDR.W R0, =0xE000ED88
```

[;] Read CPACR

LDR R1, [R0]

[;] Set bits $20\mbox{-}23$ to enable CP10 and CP11 coprocessors

ORR R1, R1, #(0xF << 20)

[;] Write back the modified value to the CPACR

STR R1, [R0]



Chapter 8 **Debug**

This chapter describes how to debug and test software running on the processor. It contains the following sections:

- About debug on page 8-2
- About the AHB-AP on page 8-6
- About the Flash Patch and Breakpoint Unit (FPB) on page 8-9.



8.1 About debug

The processor implementation determines the debug configuration, including whether debug is implemented. If the processor does not implement debug, no ROM table is present and the halt, breakpoint, and watchpoint functionality is not present.

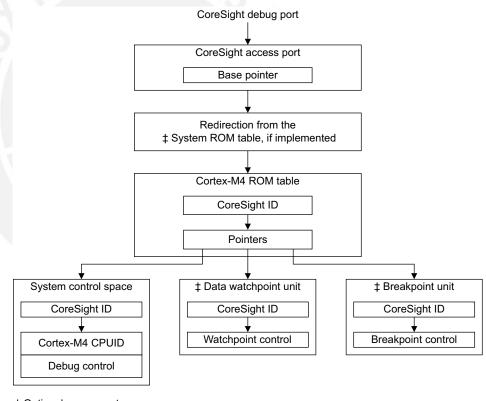
Basic debug functionality includes processor halt, single-step, processor core register access, Vector Catch, unlimited software breakpoints, and full system memory access. See the *ARMv7-M Architectural Reference Manual* for more information.

The debug option might include:

- a breakpoint unit supporting two literal comparators and six instruction comparators, or only two instruction comparators
- a watchpoint unit supporting one or four watchpoints.

For processors that implement debug, ARM recommends that a debugger identify and connect to the debug components using the CoreSight debug infrastructure.

Figure 8-1 shows the recommended flow that a debugger can follow to discover the components in the CoreSight debug infrastructure. In this case a debugger reads the peripheral and component ID registers for each CoreSight component in the CoreSight system.



‡ Optional component

Figure 8-1 CoreSight discovery

To identify the Cortex-M4 processor within the CoreSight system, ARM recommends that a debugger perform the following actions:

1. Locate and identify the Cortex-M4 ROM table using its CoreSight identification. See Table 8-1 on page 8-3 for more information.



2. Follow the pointers in that Cortex-M4 ROM table

- a. System Control Space (SCS)
- b. Breakpoint unit (BPU)
- c. Data watchpoint unit (DWT).

See Table 8-2 on page 8-4 for more information.

When a debugger identifies the SCS from its CoreSight identification, it can identify the processor and its revision number from the CPUID register in the SCS at address 0xE000ED00.

A debugger cannot rely on the Cortex-M4 ROM table being the first ROM table encountered. One or more system ROM tables are required between the access port and the Cortex-M4 ROM table if other CoreSight components are in the system. If a system ROM table is present, this can include a unique identifier for the implementation.

8.1.1 Cortex-M4 ROM table identification and entries

Table 8-1 shows the ROM table identification registers and values for debugger detection. This permits debuggers to identify the processor and its debug capabilities.

Table 8-1 Cortex-M4 ROM table identification values

Address	Register	Value	Description
0xE00FFFD0	Peripheral ID4	0x00000004	Component and Peripheral ID register formats in the
0xE00FFFD4	Peripheral ID5	0x00000000	ARMv7-M Architectural Reference Manual
0xE00FFFD8	Peripheral ID6	0x00000000	
0xE00FFFDC	Peripheral ID7	0x00000000	
0xE00FFFE0	Peripheral ID0	0x000000C4	
0xE00FFFE4	Peripheral ID1	0x000000B4	
0xE00FFFE8	Peripheral ID2	0x0000000B	
0xE00FFFEC	Peripheral ID3	0x00000000	
0xE00FFFF0	Component ID0	0x0000000D	
0xE00FFFF4	Component ID1	0x00000010	
0xE00FFFF8	Component ID2	0x00000005	-
0xE00FFFFC	Component ID3	0x000000B1	-

These values for the Peripheral ID registers identify this as a generic ROM table for the Cortex-M4 processor. Your implementation might use these registers to identify the manufacturer and part number for the device.

The Component ID registers identify this as a CoreSight ROM table.

——Note ———
The Cortex-M4 ROM table only supports word size transactions.



Table 8-2 shows the CoreSight components that the Cortex-M4 ROM table points to. The values depend on the implemented debug configuration.

Table 8-2 Cortex-M4 ROM table components

Address	Component	Value	Description
0xE00FF000	SCS	0xFFF0F003	See System Control Space
0xE00FF004	DWT	0xFFF02003a	See Table 9-1 on page 9-4
0xE00FF008	FPB	0xFFF03003 ^b	See Table 8-7 on page 8-10
0xE00FF00C	ITM	0xFFF01003c	See Table 10-1 on page 10-4
0xE00FF010	TPIU	0xFFF41003 ^d	See Table 11-1 on page 11-5.
0xE00FF014	ETM	0xFFF42003e	See the ETM-M4 Technical Reference Manual.
0xE00FF018	End marker	0x00000000	See DAP accessible ROM table in the ARMv7-M
0xE00FFFCC	SYSTEM ACCESS	0x00000001	Architectural Reference Manual.

- a. Reads as 0xFFF02002 if no watchpoints are implemented.
- b. Reads as 0xFFF03002 if no breakpoints are implemented.
- c. Reads as 0xFFF01002 if no ITM is implemented.
- d. Reads as 0xFFF41002 if no TPIU is implemented.
- e. Reads as 0xFFF42002 if no ETM is implemented.

The ROM table entries point to the debug components of the processor. The offset for each entry is the offset of that component from the ROM table base address, 0xE00FF000.

See the *ARMv7-M Architectural Reference Manual* and the *ARM CoreSight Components Technical Reference Manual* for more information about the ROM table ID and component registers, and their addresses and access types.

8.1.2 System Control Space

If debug is implemented, the processor provides debug through registers in the SCS. See:

- Debug register summary on page 8-5
- System address map on page 3-14.



SCS CoreSight identification

Table 8-3 shows the SCS CoreSight identification registers and values for debugger detection. Final debugger identification of the Cortex-M4 processor is through the CPUID register in the SCS. See *CPUID Base Register*, *CPUID* on page 4-5.

Table 8-3 SCS identification values

Address	Register	Value	Description
0xE000EFD0	Peripheral ID4	0x00000004	Component and Peripheral ID register formats in
0xE000EFE0	Peripheral ID0	0x0000000C	the ARMv7-M Architectural Reference Manual.
0xE000EFE4	Peripheral ID1	0x000000B0	-
0xE000EFE8	Peripheral ID2	0x0000000B	_
0xE000EFEC	Peripheral ID3	0×00000000	_
0xE000EFF0	Component ID0	0x0000000D	
0xE000EFF4	Component ID1	0×000000E0	/_
0xE000EFF8	Component ID2	0x00000005	
0xE000EFFC	Component ID3	0x000000B1	

See the *ARMv7-M Architectural Reference Manual* and the *ARM CoreSight Components Technical Reference Manual* for more information about the SCS CoreSight identification registers, and their addresses and access types.

8.1.3 Debug register summary

Table 8-4 shows the debug registers. Each of these registers is 32 bits wide and is described in the *ARMv7-M Architectural Reference Manual*.

Table 8-4 Debug registers

Address	Name	Туре	Reset	Description
0xE000ED30	DFSR	RW	0x00000000a	Debug Fault Status Register
0xE000EDF0	DHCSR	RW	0x00000000	Debug Halting Control and Status Register
0xE000EDF4	DCRSR	WO	-	Debug Core Register Selector Register
0xE000EDF8	DCRDR	RW	-	Debug Core Register Data Register
0xE000EDFC	DEMCR	RW	0×00000000	Debug Exception and Monitor Control Register

a. Power-on reset only

Core debug is an optional component. If core debug is removed then halt mode debugging is not supported, and there is no halt, stepping, or register transfer functionality. Debug monitor mode is still supported.



8.2 About the AHB-AF

The AHB-AP is a *Memory Access Port* (MEM-AP) as defined in the *ARM Debug Interface v5 Architecture Specification*. The AHB-AP is an optional debug access port into the Cortex-M4 system, and provides access to all memory and registers in the system, including processor registers through the SCS. System access is independent of the processor status. Either SW-DP or SWJ-DP is used to access the AHB-AP.

The AHB-AP is a master into the Bus Matrix. Transactions are made using the AHB-AP programmers model, which generates AHB-Lite transactions into the Bus Matrix.

8.2.1 AHB-AP transaction types

The AHB-AP does not perform back-to-back transactions on the bus, and so all transactions are non-sequential. The AHB-AP can perform unaligned and bit-band transactions. The Bus Matrix handles these. The AHB-AP transactions are not subject to MPU lookups. AHB-AP transactions bypass the FPB, and so the FPB cannot remap AHB-AP transactions.

AHB-AP transactions are little-endian.

8.2.2 AHB-AP programmers model

Table 8-5 shows the AHB-AP registers. If the AHB-AP is not present, these registers read as zero. Any register that is not specified in this table reads as zero.

Table 8-5 AHB-AP register summary

Offseta	Name	Туре	Reset	Description
0x00	CSW	RW	See register	AHB-AP Control and Status Word Register, CSW
0x04	TAR	RW	SIL	AHB-AP Transfer Address Register
0x0C	DRW	RW		AHB-AP Data Read/Write Register
0x10	BD0	RW		AHB-AP Banked Data Register0
0x14	BD1	RW		AHB-AP Banked Data Register1
0x18	BD2	RW	1117	AHB-AP Banked Data Register2
0x1C	BD3	RW	- 1	AHB-AP Banked Data Register3
0xF8	DBGDRAR	RO	0xE00FF003	AHB-AP ROM Address Register
0xFC	IDR	RO	0x24770011	AHB-AP Identification Register

a. The offset given in this table is relative to the location of the AHB-AP in the DAP memory space. This space is only visible from the access port. It is not part of the processor memory map.

The following sections describe the AHB-AP registers whose implementation is specific to this processor. Other registers are described in the *CoreSight Components Technical Reference Manual*.

AHB-AP Control and Status Word Register, CSW

The CSW characteristics are:

Purpose Configures and controls transfers through the AHB interface.

Usage constraints There are no usage constraints.



Configurations This register is available in all processor configurations

Attributes See the register summary in Table 8-5 on page 8-6.

Figure 8-2 shows the CSW bit assignments.

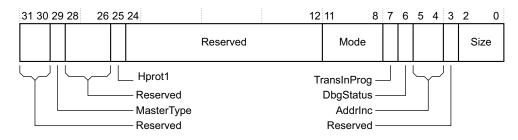


Figure 8-2 CSW bit assignments

Table 8-6 shows the CSW bit assignments.

Table 8-6 CSW bit assignments

Bits	Name	Function
[31:30]	- 5	Reserved. Read as b00.
[29]	MasterTypea	0 = core.
		1 = debug.
		This bit must not be changed if a transaction is outstanding. A debugger must first check bit [7], TransInProg.
		Reset value = $b1$.
		An implementation can configure this bit to be read only with a value of 1. In that case, transactions are always indicated as debug.
[28:26]	-	Reserved, b000.
[25]	Hprot1	User and Privilege control - HPROT[1].
		Reset value = $b1$.
[24]	-	Reserved, b1.
[23:12]	-	Reserved, 0x000.
[11:8]	Mode	Mode of operation bits:
		b0000 = normal download and upload mode
		b0001-b1111 are reserved.
		Reset value = $b0000$.
[7]	TransInProg	Transfer in progress. This field indicates if a transfer is in progress on the APB master port.
[6]	DbgStatus	Indicates the status of the DAPEN port.
		1 = AHB transfers permitted.
		0 = AHB transfers not permitted.



Table 8-6 CSW bit assignments (continued

Bits	Name	Function					
[5:4]	AddrInc	Auto address increment and pack mode on Read or Write data access. Only increments if the current transaction completes with no error.					
		Auto address incrementing and packed transfers are not performed on access to Banked Data registers 0x10 - 0x1C. The status of these bits is ignored in these cases.					
		Increments and wraps within a 4-KB address boundary, for example from 0x1000 to 0x1FFC. If the start is at 0x14A0, the counter increments to 0x1FFC, wraps to 0x1000, then continues incrementing to 0x149C.					
		b00 = auto increment off.					
		b01 = increment single. Single transfer from corresponding byte lane.					
		b10 = increment packed.b					
		b11 = reserved. No transfer.					
		Size of address increment is defined by the Size field [2:0].					
		Reset value: b00.					
[3]	-	Reserved.					
[2:0]	Size	Size of access field:					
		b000 = 8 bits					
		b001 = 16 bits					
		b010 = 32 bits					
		b011-111 are reserved.					
		Reset value: b000.					

a. When clear, this bit prevents the debugger from setting the C_DEBUGEN bit in the Debug Halting Control and Status Register, and so prevents the debugger from being able to halt the processor.

b. See the definition of packed transfers in the ARM Debug Interface v5 Architecture Specification.



8.3 About the Flash Patch and Breakpoint Unit (FPB)

The FPB:

- implements hardware breakpoints
- patches code and data from code space to system space.

A full FPB unit contains:

- Two literal comparators for matching against literal loads from Code space, and remapping to a corresponding area in System space.
- Six instruction comparators for matching against instruction fetches from Code space, and
 remapping to a corresponding area in System space. Alternatively, you can configure the
 comparators individually to return a *Breakpoint Instruction* (BKPT) to the processor core
 on a match, to provide hardware breakpoint capability.

A reduced FPB unit contains:

Two instruction comparators. You can configure each comparator individually to return a
Breakpoint Instruction to the processor on a match, to provide hardware breakpoint
capability.

8.3.1 FPB functional description

The FPB contains both a global enable and individual enables for the eight comparators. If the comparison for an entry matches, the address is either:

- remapped to the address set in the remap register plus an offset corresponding to the comparator that matched
- remapped to a BKPT instruction if that feature is enabled.

The comparison happens dynamically, but the result of the comparison occurs too late to stop the original instruction fetch or literal load taking place from the Code space. The processor ignores this transaction however, and only the remapped transaction is used.

If an MPU is present, the MPU lookups are performed for the original address, not the remapped address.

You can remove the FPB if no debug is required, or you can reduce the number of breakpoints it supports to two. If the FPB supports only two breakpoints then only comparators 0 and 1 are used, and the FPB does not support flash patching.

_____Note _____

- Unaligned literal accesses are not remapped. The original access to the DCode bus takes place in this case.
- Load exclusive accesses can be remapped. However, it is Unpredictable whether they are performed as exclusive accesses or not.
- Setting the flash patch remap location to a bit-band alias is not supported and results in Unpredictable behavior.



8.3.2 FPB programmers mode

Table 8-7 shows the FPB registers. Depending on the implementation of your processor, some of these registers might not be present. Any register that is configured as not present reads as zero.

Table 8-7 FPB register summary

Address	Name	Туре	Reset	Description
0xE0002000	FP_CTRL	RW	0x260	FlashPatch Control Register
0xE0002004	FP_REMAP	RW	-	FlashPatch Remap Register
0xE0002008	FP_COMP0	RW	b0a	FlashPatch Comparator Register0
0xE000200C	FP_COMP1	RW	b0	FlashPatch Comparator Register1
0xE0002010	FP_COMP2	RW	b0	FlashPatch Comparator Register2
0xE0002014	FP_COMP3	RW	b0	FlashPatch Comparator Register3
0xE0002018	FP_COMP4	RW	b0	FlashPatch Comparator Register4
0xE000201C	FP_COMP5	RW	b0	FlashPatch Comparator Register5
0xE0002020	FP_COMP6	RW	b0	FlashPatch Comparator Register6
0xE0002024	FP_COMP7	RW	b0	FlashPatch Comparator Register7
0xE0002FD0	PID4	RO	0x04	Peripheral identification registers
0xE0002FD4	PID5	RO	0x00	
0xE0002FD8	PID6	RO	0x00	
0xE0002FDC	PID7	RO	0x00	
0xE0002FE0	PID0	RO	0x03	
0xE0002FE4	PID1	RO	0xB0	-
0xE0002FE8	PID2	RO	0x2B	-
0xE0002FEC	PID3	RO	0x00	-
0xE0002FF0	CID0	RO	0x0D	Component identification registers
0xE0002FF4	CID1	RO	0xE0	-
0xE0002FF8	CID2	RO	0x05	-
0xE0002FFC	CID3	RO	0xB1	-

a. For FP_COMP0 to FP_COMP7, bit 0 is reset to 0. Other bits in these registers are

All FPB registers are described in the ARMv7-M Architecture Reference Manual.



Chapter 9 **Data Watchpoint and Trace Unit**

This chapter describes the *Data Watchpoint and Trace* (DWT) unit. It contains the following sections:

- *About the DWT* on page 9-2
- DWT functional description on page 9-3
- DWT Programmers Model on page 9-4.



9.1 About the DW

The DWT is an optional debug unit that provides watchpoints, data tracing, and system profiling for the processor.





9.2 DWT functional description

A full DWT contains four comparators that you can configure as

- a hardware watchpoint
- an ETM trigger
- a PC sampler event trigger
- a data address sampler event trigger.

The first comparator, DWT_COMP0, can also compare against the clock cycle counter, CYCCNT. You can also use the second comparator, DWT_COMP1, as a data comparator.

A reduced DWT contains one comparator that you can use as a watchpoint or as a trigger. It does not support data matching.

The DWT if present contains counters for:

- clock cycles (CYCCNT)
- folded instructions
- Load Store Unit (LSU) operations
- sleep cycles
- CPI, that is all instruction cycles except for the first cycle
- interrupt overhead.

Note
An event is generated each time a counter overflows.

You can configure the DWT to generate PC samples at defined intervals, and to generate interrupt event information.

The DWT provides periodic requests for protocol synchronization to the ITM and the TPIU, if the your implementation includes the Cortex-M4 TPIU.



9.3 DWT Programmers Model

Table 9-1 lists the DWT registers. Depending on the implementation of your processor, some of these registers might not be present. Any register that is configured as not present reads as zero.

Table 9-1 DWT register summary

Address	Name	Туре	Reset	Description
0xE0001000	DWT_CTRL	RW	See a	Control Register
0xE0001004	DWT_CYCCNT	RW	0x00000000	Cycle Count Register
0xE0001008	DWT_CPICNT	RW	-	CPI Count Register
0xE000100C	DWT_EXCCNT	RW	-	Exception Overhead Count Register
0xE0001010	DWT_SLEEPCNT	RW	-	Sleep Count Register
0xE0001014	DWT_LSUCNT	RW	-	LSU Count Register
0xE0001018	DWT_FOLDCNT	RW	5	Folded-instruction Count Register
0xE000101C	DWT_PCSR	RO	30	Program Counter Sample Register
0xE0001020	DWT_COMP0	RW	-	Comparator Register0
0xE0001024	DWT_MASK0	RW	- 1	Mask Register0
0xE0001028	DWT_FUNCTION0	RW	0x00000000	Function Register0
0xE0001030	DWT_COMP1	RW		Comparator Register1
0xE0001034	DWT_MASK1	RW	-/	Mask Register1
0xE0001038	DWT_FUNCTION1	RW	0x00000000	Function Register1
0xE0001040	DWT_COMP2	RW	5//	Comparator Register2
0xE0001044	DWT_MASK2	RW	-7/	Mask Register2
0xE0001048	DWT_FUNCTION2	RW	0x00000000	Function Register2
0xE0001050	DWT_COMP3	RW	- 3-2-	Comparator Register3
0xE0001054	DWT_MASK3	RW	-	Mask Register3
0xE0001058	DWT_FUNCTION3	RW	0x00000000	Function Register3
0xE0001FD0	PID4	RO	0x04	Peripheral identification registers
0xE0001FD4	PID5	RO	0x00	-
0xE0001FD8	PID6	RO	0x00	-
0xE0001FDC	PID7	RO	0x00	-
0xE0001FE0	PID0	RO	0x02	-
0xE0001FE4	PID1	RO	0xB0	-
0xE0001FE8	PID2	RO	0x3B	-
0xE0001FEC	PID3	RO	0x00	-



Table 9-1 DWT register summary (continued

Address	Name	Type	Reset	Description
0xE0001FF0	CID0	RO	0x0D	Component identification registers
0xE0001FF4	CID1	RO	0xE0	_
0xE0001FF8	CID2	RO	0x05	_
0xE0001FFC	CID3	RO	0xB1	_

a. Possible reset values are:

0x40000000 if four comparators for watchpoints and triggers are present

0x4F000000 if four comparators for watchpoints only are present

0x10000000 if only one comparator is present

0x1F000000 if one comparator for watchpoints and not triggers is present

0x00000000 if DWT is not present.

DWT registers are described in the *ARMv7M Architecture Reference Manual*. Peripheral Identification. Component Identification registers are described in the *ARM CoreSight Components Technical Reference Manual*.

____Note _____

- Cycle matching functionality is only available in comparator 0.
- Data matching functionality is only available in comparator 1.
- Data value is only sampled for accesses that do not produce an MPU or bus fault. The PC is sampled irrespective of any faults. The PC is only sampled for the first address of a burst.
- The FUNCTION field in the DWT_FUNCTION1 register is overridden for comparators given by DATAVADDR0 and DATAVADDR1 if DATAVMATCH is also set in DWT_FUNCTION1. The comparators given by DATAVADDR0 and DATAVADDR1 can then only perform address comparator matches for comparator 1 data matches.
- If the data matching functionality is not included during implementation it is not possible to set DATAVADDR0, DATAVADDR1, or DATAVMATCH in DWT_FUNCTION1. This means that the data matching functionality is not available in the implementation. Test the availability of data matching by writing and reading the DATAVMATCH bit in DWT_FUNCTION1. If this bit cannot be set then data matching is unavailable.
- PC match is not recommended for watchpoints because it stops after the instruction. It mainly guards and triggers the ETM.



Chapter 10 **Instrumentation Trace Macrocell Unit**

This chapter describes the *Instrumentation Trace Macrocell* (ITM) unit. It contains the following sections:

- *About the ITM* on page 10-2
- ITM functional description on page 10-3
- *ITM programmers model* on page 10-4.



10.1 About the ITM

The ITM is a an optional application-driven trace source that supports printf style debugging to trace operating system and application events, and generates diagnostic system information.





10.2 ITM functional description

The ITM generates trace information as packets. There are four sources that can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The four sources in decreasing order of priority are:

- Software trace. Software can write directly to ITM stimulus registers to generate packets.
- Hardware trace. The DWT generates these packets, and the ITM outputs them.
- Time stamping. Timestamps are generated relative to packets. The ITM contains a 21-bit counter to generate the timestamp. The Cortex-M4 clock or the bitclock rate of the *Serial Wire Viewer* (SWV) output clocks the counter.
- Global system timestamping. Timestamps can optionally be generated using a system-wide 48-bit count value. The same count value can be used to insert timestamps in the ETM trace stream, allowing coarse-grain correlation.





10.3 ITM programmers model

Table 10-1 shows the ITM registers. Depending on the implementation of your processor, the ITM registers might not be present. Any register that is configured as not present reads as zero.

— Note —

- You must enable TRCENA of the Debug Exception and Monitor Control Register before you program or use the ITM.
- If the ITM stream requires synchronization packets, you must configure the synchronization packet rate in the DWT.

Table 10-1 ITM register summary

Address	Name	Type	Reset	Description
0xE0000000- 0xE000007C	ITM_STIM0- ITM_STIM31	RW	VEB,	Stimulus Port Registers 0-31
0xE0000E00	ITM_TER	RW	0x00000000	Trace Enable Register
0xE0000E40	ITM_TPR	RW	0x00000000	ITM Trace Privilege Register, ITM_TPR on page 10-5
0xE0000E80	ITM_TCR	RW	0x00000000	Trace Control Register
0xE0000FD0	PID4	RO	0x00000004	Peripheral Identification registers
0xE0000FD4	PID5	RO	0x00000000	
0xE0000FD8	PID6	RO	0x00000000	
0xE0000FDC	PID7	RO	0x00000000	
0xE0000FE0	PID0	RO	0x00000001	
0xE0000FE4	PID1	RO	0x000000B0	
0xE0000FE8	PID2	RO	0x0000003B	
0xE0000FEC	PID3	RO	0x00000000	
0xE0000FF0	CID0	RO	0×0000000D	Component Identification registers
0xE0000FF4	CID1	RO	0x000000E0	_
0xE0000FF8	CID2	RO	0x00000005	_
0xE0000FFC	CID3	RO	0x000000B1	_

——Note ————
ITM registers are fully accessible in privileged mode. In user mode, all registers can be read,

but only the Stimulus Registers and Trace Enable Registers can be written, and only when the corresponding Trace Privilege Register bit is set. Invalid user mode writes to the ITM registers

are discarded.

The following sections describes the ITM registers whose implementation is specific to this processor. Other registers are described in the *ARMv7-M Architectural Reference Manual*.



10.3.1 ITM Trace Privilege Register, ITM TPR

The ITM_TPR characteristics are:

Purpose Enables an operating system to control the stimulus ports that are

accessible by user code.

Usage constraints You can only write to this register in privileged mode.

Configurations This register is available if the ITM is configured in your implementation.

Attributes See Table 10-1 on page 10-4.

Figure 10-1 shows the ITM_TPR bit assignments.



Figure 10-1 ITM_TPR bit assignments

Table 10-2 shows the ITM_TPR bit assignments.

Table 10-2 ITM_TPR bit assignments

Bits	Name	Function	
[31:4]	-	Reserved.	
[3:0]	PRIVMASK	Bit mask to enable tracing on ITM stimulus ports: bit [0] = stimulus ports [7:0] bit [1] = stimulus ports [15:8] bit [2] = stimulus ports [23:16] bit [3] = stimulus ports [31:24].	



Chapter 11 Trace Port Interface Unit

This chapter describes the Cortex-M4 TPIU, the Trace Port Interface Unit that is specific to the Cortex-M4 processor. It contains the following sections:

- About the Cortex-M4 TPIU on page 11-2
- TPIU functional description on page 11-3
- TPIU programmers model on page 11-5.



11.1 About the Cortex-M4 TPIL

The Cortex-M4 TPIU is an optional component that acts as a bridge between the on-chip trace data from the *Embedded Trace Macrocell* (ETM) and the *Instrumentation Trace Macrocell* (ITM), with separate IDs, to a data stream. The TPIU encapsulates IDs where required, and the data stream is then captured by a *Trace Port Analyzer* (TPA).

The Cortex-M4 TPIU is specially designed for low-cost debug. It is a special version of the CoreSight TPIU. Your implementation can replace the Cortex-M4 TPIU with other CoreSight components if your implementation requires the additional features of the CoreSight TPIU.

In this chapter, the term TPIU refers to the Cortex-M4 TPIU. For information about the CoreSight TPIU, see the *ARM CoreSight Components Technical Reference Manual*.



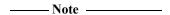


11.2 TPIU functional description

There are two configurations of the TPIU:

- A configuration that supports ITM debug trace.
- A configuration that supports both ITM and ETM debug trace.

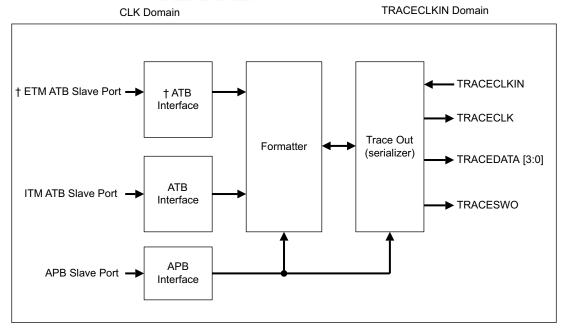
If your implementation requires no trace support then the TPIU might not be present.



If your Cortex-M4 system uses the optional ETM component, the TPIU configuration supports both ITM and ETM debug trace. See the *ETM-M4 Technical Reference Manual*.

11.2.1 TPIU block diagrams

Figure 11-1 shows the component layout of the TPIU for both configurations.



† Optional component

Figure 11-1 TPIU block diagram

11.2.2 TPIU Formatter

The formatter inserts source ID signals into the data packet stream so that trace data can be re-associated with its trace source. The formatter is always active when the Trace Port Mode is active.

The formatting protocol is described in the *CoreSight Architecture Specification*. You must enable synchronization packets in the DWT to provide synchronization for the formatter.

When the formatter is enabled, half-sync packets may be inserted if there is no data to output after a frame has been started. Synchronization, caused by the distributed synchronization from the DWT, will ensure that any partial frame is completed, and at least one full synchronization packet will be generated.



11.2.3 Serial Wire Output format

The TPIU can output trace data in a Serial Wire Output (SWO) format:

- TPIU_DEVID specifies the formats that are supported. See *TPIU_DEVID* on page 11-12.
- TPIU_SPPR specifies the SWO format in use. See the *ARMv7-M Architecture Reference Manual*.

When one of the two SWO modes is selected, you can enable the TPIU to bypass the formatter for trace output. If the formatter is bypassed, only the ITM and DWT trace source passes through. The TPIU accepts and discards data from the ETM. This function can be used to connect a device containing an ETM to a trace capture device that is only able to capture SWO data.





11.3 TPIU programmers model

Table 11-1 provides a summary of the TPIU registers. Depending on the implementation of your processor, the TPIU registers might not be present, or the CoreSight TPIU might be present instead. Any register that is configured as not present reads as zero.

Table 11-1 TPIU registers

Address	Name	Туре	Reset	Description
0xE0040000	TPIU_SSPSR	RO	0x0xx	Supported Parallel Port Size Register
0xE0040004	TPIU_CSPSR	RW	0x01	Current Parallel Port Size Register
0xE0040010	TPIU_ACPR	RW	0x0000	Asynchronous Clock Prescaler Register, TPIU_ACPR on page 11-6
0xE00400F0	TPIU_SPPR	RW	0x01	Selected Pin Protocol Register
0xE0040300	TPIU_FFSR	RO	0x08	Formatter and Flush Status Register, TPIU_FFSR on page 11-6
0xE0040304	TPIU_FFCR	RW	0x102	Formatter and Flush Control Register, TPIU_FFCR on page 11-7
0xE0040308	TPIU_FSCR	RO	0x00	Formatter Synchronization Counter Register
0xE0040EE8	TRIGGER	RO	0x0	TRIGGER on page 11-8
0xE0040EEC	FIFO data 0	RO	0x000000	Integration ETM Data on page 11-8
0xE0040EF0	ITATBCTR2	RO	0x0	ITATBCTR2 on page 11-9
0xE0040EFC	FIFO data 1	RO	0x000000	Integration ITM Data on page 11-10
0xE0040EF8	ITATBCTR0	RO	0x0	ITATBCTR0 on page 11-11
0xE0040F00	ITCTRL	RW	0x0	Integration Mode Control, TPIU_ITCTRL on page 11-11
0xE0040FA0	CLAIMSET	RW	0xF	Claim tag set
0xE0040FA4	CLAIMCLR	RW	0x0	Claim tag clear
0xE0040FC8	DEVID	RO	0xCA0/0xCA1	TPIU_DEVID on page 11-12
0xE0040FCC	DEVTYPE	RO	0x11	TPIU_DEVTYPE on page 11-13
0xE0040FD0	PID4	RO	0x04	Peripheral identification registers
0xE0040FD4	PID5	RO	0x00	A '-
0xE0040FD8	PID6	RO	0x00	
0xE0040FDC	PID7	RO	0x00	
0xE0040FE0	PID0	RO	0xA1	
0xE0040FE4	PID1	RO	0xB9	
0xE0040FE8	PID2	RO	0x0B	
0xE0040FEC	PID3	RO	0x00	
0xE0040FF0	CID0	RO	0x0D	Component identification registers
0xE0040FF4	CID1	RO	0x90	
0xE0040FF8	CID2	RO	0x05	
0xE0040FFC	CID3	RO	0xB1	



The following sections describe the TPIU registers whose implementation is specific to thi

processor. The Formatter, Integration Mode Control, and Claim Tag registers are described in the *CoreSight Components Technical Reference Manual*. Other registers are described in the *ARMv7-M Architecture Reference Manual*.

11.3.1 Asynchronous Clock Prescaler Register, TPIU_ACPR

The TPIU_ACPR characteristics are:

Purpose Scales the baud rate of the asynchronous output.

Usage constraints There are no usage constraints.

Configurations This register is available in all processor configurations.

Attributes See Table 11-1 on page 11-5.

Figure 11-2 shows the TPIU_ACPR bit assignments.



Figure 11-2 TPIU_ACPR bit assignments

Table 11-2 shows the TPIU ACPR bit assignments.

Table 11-2 TPIU_ACPR bit assignments

Bits	Name	Function
[31:13]		Reserved. RAZ/SBZP.
[12:0]	PRESCALER	Divisor for TRACECLKIN is Prescaler + 1.

11.3.2 Formatter and Flush Status Register, TPIU_FFSR

The TPIU_FFSR characteristics are:

Purpose Indicates the status of the TPIU formatter.

Usage constraints There are no usage constraints.

Configurations This register is available in all processor configurations.

Attributes See Table 11-1 on page 11-5.

Figure 11-3 shows the TPIU FFSR bit assignments.

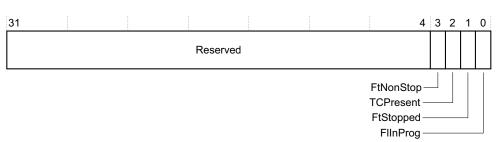


Figure 11-3 TPIU_FFSR bit assignments



Table 11-3 shows the TPIU FFSR bit assignments.

Table 11-3 TPIU_FFSR bit assignments

Bits	Name	Function
[31:4]	-	Reserved
[3]	FtNonStop	Formatter cannot be stopped
[2]	TCPresent	This bit always reads zero
[1]	FtStopped	This bit always reads zero
[0]	FlInProg	This bit always reads zero

11.3.3 Formatter and Flush Control Register, TPIU_FFCR

The TPIU_FFCR characteristics are:

Purpose Controls the TPIU formatter.

Usage constraints There are no usage constraints.

Configurations This register is available in all processor configurations.

Attributes See Table 11-1 on page 11-5.

Figure 11-4 shows the TPIU_FFCR bit assignments.

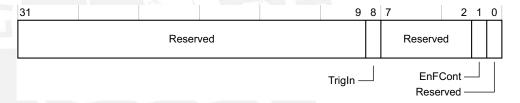


Figure 11-4 TPIU_FFCR bit assignments

Table 11-4 shows the TPIU_FFCR bit assignments.

Table 11-4 TPIU_FFCR bit assignments

Bits	Name	Function
[31:9]	-	Reserved.
[8]	TrigIn	This bit Reads-As-One (RAO), specifying that triggers are inserted when a trigger pin is asserted.
[7:2]	-	Reserved.
[1]	EnFCont	Enable continuous formatting. Value can be: 0 = Continuous formatting disabled. 1 = Continuous formatting enabled.
[0]	-	Reserved.

The TPIU can output trace data in a *Serial Wire Output* (SWO) format. See *Serial Wire Output format* on page 11-4.



When one of the two SWO modes is selected, bit [1] of TPIU_FFCR enables the formatter to

be bypassed. If the formatter is bypassed, only the ITM and DWT trace source passes through. The TPIU accepts and discards data from the ETM. This function is can be used to connect a device containing an ETM to a trace capture device that is only able to capture SWO data. Enabling or disabling the formatter causes momentary data corruption.



If TPIU_SPPR is set to select Trace Port Mode, the formatter is automatically enabled. If you then select one of the SWO modes, TPIU_FFCR reverts to its previously programmed value.

11.3.4 TRIGGER

The TRIGGER characteristics are:

Purpose Integration test of the TRIGGER input.

Usage constraints There are no usage constraints.

Configurations This register is available in all processor configurations.

Attributes See Table 11-1 on page 11-5.

Figure 11-5 shows the TRIGGER bit assignments.



Figure 11-5 TRIGGER bit assignments

Table 11-5 shows the TRIGGER bit assignments.

Table 11-5 TRIGGER bit assignments

Bits	Name	Function
[31:1]	MVVI	Reserved
[0]	TRIGGER input value	When read, this bit returns the TRIGGER input.

11.3.5 Integration ETM Data

The Integration ETM Data characteristics are:

Purpose Trace data integration testing.

Usage constraints You must set bit [1] of TPIU_ITCTRL to use this register. See *Integration*

Mode Control, TPIU ITCTRL on page 11-11.

Configurations This register is available in all processor configurations.

Attributes See Table 11-1 on page 11-5

Figure 11-6 on page 11-9 shows the Integration ETM Data bit assignments.



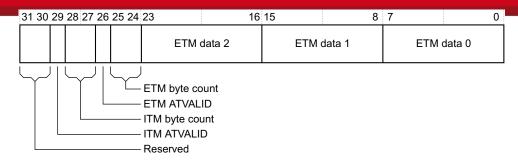


Figure 11-6 Integration ETM Data bit assignments

Table 11-6 shows the Integration ETM Data bit assignments.

Table 11-6 Integration ETM Data bit assignments

Bits	Name	Function
[31:30]	- 3	Reserved
[29]	ITM ATVALID input	Returns the value of the ITM ATVALID signal.
[28:27]	ITM byte count	Number of bytes of ITM trace data since last read of Integration ITM Data Register.
[26]	ETM ATVALID input	Returns the value of the ETM ATVALID signal.
[25:24]	ETM byte count	Number of bytes of ETM trace data since last read of Integration ETM Data Register.
[23:16]	ETM data 2	ETM trace data. The TPIU discards this data when the register is read.
[15:8]	ETM data 1	
[7:0]	ETM data 0	

11.3.6 ITATBCTR2

The ITATBCTR2 characteristics are:

Purpose Integration test.

Usage constraints You must set bit [0] of TPIU_ITCTRL to use this register. See *Integration*

Mode Control, TPIU_ITCTRL on page 11-11.

Configurations This register is available in all processor configurations.

Attributes See Table 11-1 on page 11-5.

Figure 11-7 shows the ITATBCTR2 bit assignments.

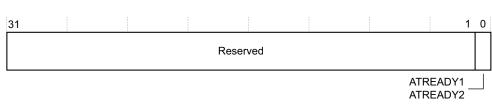


Figure 11-7 ITATBCTR2 bit assignments



Table 11-7 shows the ITATBCTR2 bit assignments.

Table 11-7 ITATBCTR2 bit assignments

Bits	Name	Function
[31:1]	-	Reserved
[0]	ATREADY1, ATREADY2	This bit sets the value of both the ETM and ITM ATREADY outputs, if the TPIU is in integration test mode.

11.3.7 Integration ITM Data

The Integration ITM Data characteristics are:

Purpose Trace data integration testing.

Usage constraints You must set bit [1] of TPIU_ITCTRL to use this register. See *Integration*

Mode Control, TPIU_ITCTRL on page 11-11.

Configurations This register is available in all processor configurations.

Attributes See Table 11-1 on page 11-5

Figure 11-8 shows the Integration ITM Data bit assignments.

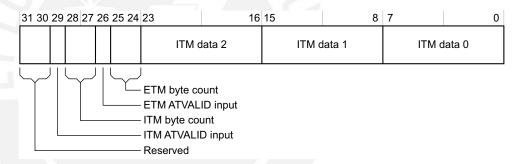


Figure 11-8 Integration ITM Data bit assignments

Table 11-8 shows the Integration ITM Data bit assignments.

Table 11-8 Integration ITM Data bit assignments

Bits	Name	Function
[31:30]	-	Reserved
[29]	ITM ATVALID input	Returns the value of the ITM ATVALID signal.
[28:27]	ITM byte count	Number of bytes of ITM trace data since last read of Integration ITM Data Register.
[26]	ETM ATVALID input	Returns the value of the ETM ATVALID signal.
[25:24]	ETM byte count	Number of bytes of ETM trace data since last read of Integration ETM Data Register.
[23:16]	ITM data 2	ITM trace data. The TPIU discards this data when the register is read.
[15:8]	ITM data 1	<u>-</u>
[7:0]	ITM data 0	-



11 3 8 ITATECTE

The ITATBCTR0 characteristics are:

Purpose Integration test.

Usage constraints There are no usage constraints.

Configurations This register is available in all processor configurations.

Attributes See Table 11-1 on page 11-5.

Figure 11-9 shows the ITATBCTR0 bit assignments.

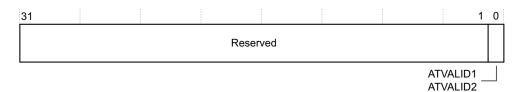


Figure 11-9 ITATBCTR0 bit assignments

Table 11-9 shows the ITATBCTR0 bit assignments.

Table 11-9 ITATBCTR0 bit assignments

Bits	Name	Function
[31:1]	1 -	Reserved
[0]	ATVALID1, ATVALID2	A read of this bit returns the value of ATVALIDS1 OR-ed with ATVALIDS2.

11.3.9 Integration Mode Control, TPIU_ITCTRL

The TPIU_ITCTRL characteristics are:

Purpose Specifies normal or integration mode for the TPIU.

Usage constraints There are no usage constraints.

Configurations This register is available in all processor configurations.

Attributes See Table 11-1 on page 11-5.

Figure 11-10 shows the TPIU_ITCTRL bit assignments.

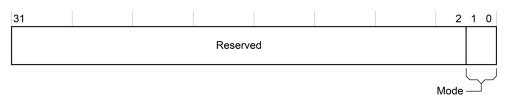


Figure 11-10 TPIU_ITCTRL bit assignments



Table 11-10 shows the TPIU ITCTRL bit assignments

Table 11-10 TPIU_ITCTRL bit assignments

Bits	Name	Function	
[31:2]	-	Reserved.	
[1:0]	Mode	Reserved. Specifies the current mode for the TPIU: b00 normal mode b01 integration test mode b10 integration data test mode b11 Reserved. In integration data test mode, the trace output is disabled, and data can be read directly from each input port using the integration data registers.	

11.3.10 TPIU_DEVID

The TPIU_DEVID characteristics are:

Purpose Indicates the functions provided by the TPIU for use in topology

detection.

Usage constraints There are no usage constraints.

Configurations This register is available in all processor configurations.

Attributes See Table 11-1 on page 11-5.

Figure 11-11 shows the TPIU_DEVID bit assignments.

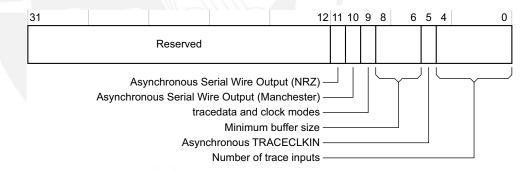


Figure 11-11 TPIU_DEVID bit assignments

Table 11-11 shows the TPIU_DEVID bit assignments.

Table 11-11 TPIU_DEVID bit assignments

Bits	Name	Function
[31:12]	-	Reserved
[11]	Asynchronous Serial Wire Output (NRZ)	This bit Reads-As-One (RAO), indicating that the output is supported.
[10]	Asynchronous Serial Wire Output (Manchester)	This bit Reads-As-One (RAO), indicating that the output is supported.
[9]	Tracedata and clock modes	This bit Reads-As-Zero (RAZ), indicating that tracedata and clock modes are supported



Table 11-11 TPIU DEVID bit assignments (continued

Bits	Name	Function
[8:6]	Minimum buffer size	Specifies the minimum TPIU buffer size: b010 = 4 bytes
[5]	Asynchronous TRACECLKIN	Specifies whether TRACECLKIN can be asynchronous to CLK $b0 = \textbf{TRACECLKIN} \text{ must be synchronous to } \textbf{CLK}$ $b1 = \textbf{TRACECLKIN} \text{ can be asynchronous to } \textbf{CLK}$
[4:0]	Number of trace inputs	Specifies the number of trace inputs: $b000000 = 1 \text{ input}$ $b000001 = 2 \text{ inputs}$ If your implementation includes an ETM, the value of this field is b0000001.

11.3.11 TPIU_DEVTYPE

The Device Type Identifier Register is read-only. It provides a debugger with information about the component when the Part Number field is not recognized. The debugger can then report this information.

The TPIU DEVTYPE characteristics are:

Purpose Indicates the type of functionality the component supports.

Usage Constraints There are no usage constraints.

Configurations This register is available in all processor configurations.

Attributes The Device Type reads as 0x11 and indicates this device is a trace sink and

specifically a TPIU

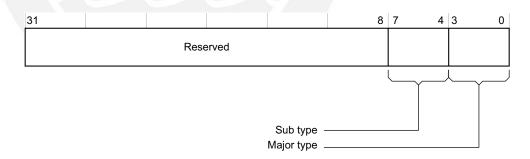


Figure 11-12 TPIU_DEVTYPE bit assignments



Appendix A **Revisions**

This appendix describes the technical changes between released issues of this book.

Table A-1 Issue A

Change	Location	Affects
First release	-	-

Table A-2 Differences between issue A and issue B

Change	Location	Affects
No technical changes	-	-

Table A-3 Differences between issue B and issue C

Change	Location	Affects
Additional information on bus interfaces.	Bus interfaces on page 2-5	All
Additional information on Private Peripheral Bus.	Private Peripheral Bus (PPB) on page 2-6	All
Updated the Cortex-M4 instruction set cycle times.	Table 3-1 on page 3-4	All
Updated assembler of the signed multiply instructions for DSP instructions.	Table 3-2 on page 3-8	All
Updated information on Load/store timings.	Load/store timings on page 3-11	All
Added information on local exclusive monitor.	Exclusive monitor on page 3-18	All



Table A-3 Differences between issue B and issue C (continued

Change	Location	Affects
Reset values updated.	Table 5-1 on page 5-4	All
Updated bit order for Auxiliary Control Register.	Figure 4-1 on page 4-5	All
Updated bit order for Auxiliary Control Register.	Table 4-2 on page 4-5	All
Updated information for Auxiliary Fault Status Register	Auxiliary Fault Status Register, AFSR on page 4-6	All
Changed address range for NVIC_IPR registers.	Table 6-1 on page 6-4	All
Added Peripheral IDs 5-7.	Table 8-1 on page 8-3	All
Updated reset value.	Table 8-7 on page 8-10	All
Added names of TPIU registers. Reset values updated and added TPIU_DEVTYPE.	Table 11-1 on page 11-5	All
Added TPIU_DEVTYPE bit assignments.	TPIU_DEVTYPE on page 11-13	All





Glossary

This glossary describes some of the terms used in technical documents from ARM.

Abort

A mechanism that indicates to a core that the attempted memory access is invalid or not allowed or that the data returned by the memory access is invalid. An abort can be caused by the external or internal memory system as a result of attempting to access invalid or protected instruction or data memory.

See also Data Abort, External Abort and Prefetch Abort.

Addressing modes

Various mechanisms, shared by many different instructions, for generating values used by the instructions.

Advanced High-performance Bus (AHB)

A bus protocol with a fixed pipeline between address/control and data phases. It only supports a subset of the functionality provided by the AMBA AXI protocol. The full AMBA AHB protocol specification includes a number of features that are not commonly required for master and slave IP developments and ARM recommends only a subset of the protocol is usually used. This subset is defined as the AMBA AHB-Lite protocol.

See also Advanced Microcontroller Bus Architecture and AHB-Lite.

Advanced Microcontroller Bus Architecture (AMBA)

A family of protocol specifications that describe a strategy for the interconnect. AMBA is the ARM open standard for on-chip buses. It is an on-chip bus specification that details a strategy for the interconnection and management of functional blocks that make up a *System-on-Chip* (SoC). It aids in the development of embedded processors with one or more CPUs or signal processors and multiple peripherals. AMBA complements a reusable design methodology by defining a common backbone for SoC modules.



Advanced Peripheral Bus (APB)

A simpler bus protocol than AXI and AHB. It is designed for use with ancillary or general-purpose peripherals such as timers, interrupt controllers, UARTs, and I/O ports. Connection to the main system bus is through a system-to-peripheral bus bridge that helps to

reduce system power consumption.

AHB See Advanced High-performance Bus.

AHB Access Port (AHB-AP)

An optional component of the DAP that provides an AHB interface to a SoC.

AHB-AP See AHB Access Port.

AHB-Lite A subset of the full AMBA AHB protocol specification. It provides all of the basic functions

required by the majority of AMBA AHB slave and master designs, particularly when used with a multi-layer AMBA interconnect. In most cases, the extra facilities provided by a full AMBA AHB interface are implemented more efficiently by using an AMBA AXI protocol interface.

AHB Trace Macrocell

A hardware macrocell that, when connected to a processor core, outputs data trace information

on a trace port.

Aligned A data item stored at an address that is divisible by the number of bytes that defines the data size

is said to be aligned. Aligned words and halfwords have addresses that are divisible by four and two respectively. The terms word-aligned and halfword-aligned therefore stipulate addresses

that are divisible by four and two respectively.

AMBA See Advanced Microcontroller Bus Architecture.

Advanced Trace Bus (ATB)

A bus used by trace devices to share CoreSight capture resources.

APB See Advanced Peripheral Bus.

Application Specific Integrated Circuit (ASIC)

An integrated circuit that has been designed to perform a specific application function. It can be

custom-built or mass-produced.

Architecture The organization of hardware and/or software that characterizes a processor and its attached

components, and enables devices with similar characteristics to be grouped together when describing their behavior, for example, Harvard architecture, instruction set architecture,

ARMv7-M architecture.

ARM instruction An instruction of the ARM Instruction Set Architecture (ISA). These cannot be executed by the

Cortex-M4 processor.

ARM state The processor state in which the processor executes the instructions of the ARM ISA. The

processor only operates in Thumb state, never in ARM state.

ASIC See Application Specific Integrated Circuit.

ATB See Advanced Trace Bus.

ATB bridge A synchronous ATB bridge provides a register slice to facilitate timing closure through the

addition of a pipeline stage. It also provides a unidirectional link between two synchronous ATB

domains.

An asynchronous ATB bridge provides a unidirectional link between two ATB domains with asynchronous clocks. It is intended to support connection of components with ATB ports

residing in different clock domains.



Base register A register specified by a load or store instruction that is used to hold the base value of the address

calculation for the instruction. Depending on the instruction and its addressing mode, an offset can be added to or subtracted from the base register value to form the address that is sent to

memory.

Base register write-back

Updating the contents of the base register used in an instruction target address calculation so that the modified address is changed to the next higher or lower sequential address in memory. This means that it is not necessary to fetch the target address for successive instruction transfers and

enables faster burst accesses to sequential memory.

Beat Alternative word for an individual data transfer within a burst. For example, an INCR4 burst

comprises four beats.

BE-8 Big-endian view of memory in a byte-invariant system.

See also BE-32, LE, Byte-invariant and Word-invariant.

BE-32 Big-endian view of memory in a word-invariant system.

See also BE-8, LE, Byte-invariant and Word-invariant.

Big-endian Byte ordering scheme in which bytes of decreasing significance in a data word are stored at

increasing addresses in memory.

See also Little-endian and Endianness.

Big-endian memory Memory in which:

 a byte or halfword at a word-aligned address is the most significant byte or halfword within the word at that address

• a byte at a halfword-aligned address is the most significant byte within the halfword at that

address.

See also Little-endian memory.

Boundary scan chain

A boundary scan chain is made up of serially-connected devices that implement boundary scan technology using a standard JTAG TAP interface. Each device contains at least one TAP controller containing shift registers that form the chain connected between **TDI** and **TDO**, through which test data is shifted. Processors can contain several shift registers to enable you to access selected parts of the device.

access selected parts of the devic

Branch folding Branch folding is a technique where the branch instruction is completely removed from the

instruction stream presented to the execution pipeline.

Breakpoint A breakpoint is a mechanism provided by debuggers to identify an instruction at which program

execution is to be halted. Breakpoints are inserted by the programmer to enable inspection of register contents, memory locations, variable values at fixed points in the program execution to test that the program is operating correctly. Breakpoints are removed after the program is

successfully tested.

See also Watchpoint.

Burst A group of transfers to consecutive addresses. Because the addresses are consecutive, there is

no requirement to supply an address for any of the transfers after the first one. This increases the speed at which the group of transfers can occur. Bursts over AMBA are controlled using signals

to indicate the length of the burst and how the addresses are incremented.

See also Beat.



Byte An 8-bit data item.

Byte-invariant In a byte-invariant system, the address of each byte of memory remains unchanged when

switching between little-endian and big-endian operation. When a data item larger than a byte is loaded from or stored to memory, the bytes making up that data item are arranged into the correct order depending on the endianness of the memory access. The ARM architecture supports byte-invariant systems in ARMv6 and later versions. When byte-invariant support is selected, unaligned halfword and word memory accesses are also supported. Multi-word

accesses are expected to be word-aligned.

See also Word-invariant.

Clock gating Gating a clock signal for a macrocell with a control signal and using the modified clock that

results to control the operating state of the macrocell.

Clocks Per Instruction (CPI)

See Cycles Per Instruction (CPI).

Cold reset Also known as power-on reset.

See also Warm reset.

Context The environment that each process operates in for a multitasking operating system.

See also Fast context switch.

Core A core is that part of a processor that contains the ALU, the datapath, the general-purpose

registers, the Program Counter, and the instruction decode and control circuitry.

Core reset See Warm reset.

CoreSight The infrastructure for monitoring, tracing, and debugging a complete system on chip.

CPI See Cycles per instruction.

Cycles Per instruction (CPI)

Cycles per instruction (or clocks per instruction) is a measure of the number of computer instructions that can be performed in one clock cycle. This figure of merit can be used to compare the performance of different CPUs that implement the same instruction set against each

other. The lower the value, the better the performance.

Data Abort An indication from a memory system to the core of an attempt to access an illegal data memory

location. An exception must be taken if the processor attempts to use the data that caused the

abort.

See also Abort.

DCode Memory Memory space at 0x00000000 to 0x1FFFFFFF.

Debug Access Port (DAP)

A TAP block that acts as an AMBA, AHB or AHB-Lite, master for access to a system bus. The DAP is the term used to encompass a set of modular blocks that support system wide debug. The DAP is a modular component, intended to be extendable to support optional access to multiple systems such as memory mapped AHB and CoreSight APB through a single debug interface.

Debugger A debugging system that includes a program, used to detect, locate, and correct software faults,

together with custom hardware that supports software debugging.

Embedded Trace Buffer

The ETB provides on-chip storage of trace data using a configurable sized RAM.



Embedded Trace Macrocell (ETM

A hardware macrocell that, when connected to a processor core, outputs instruction trace

information on a trace port.

Endianness Byte ordering. The scheme that determines the order that successive bytes of a data word are

stored in memory. An aspect of the systems memory mapping.

See also Little-endian and Big-endian

ETB See Embedded Trace Buffer.

ETM See *Embedded Trace Macrocell*.

Exception An error or event which can cause the processor to suspend the currently executing instruction

stream and execute a specific exception handler or interrupt service routine. The exception could be an external interrupt or NMI, or it could be a fault or error event that is considered serious enough to require that program execution is interrupted. Examples include attempting to perform an invalid memory access, external interrupts, and undefined instructions. When an exception occurs, normal program flow is interrupted and execution is resumed at the corresponding exception vector. This contains the first instruction of the interrupt service

routine to deal with the exception.

Exception handler

See Interrupt service routine.

Exception vector See Interrupt vector.

External PPB PPB memory space at 0xE0040000 to 0xE00FFFFF.

Flash Patch and Breakpoint unit (FPB)

A set of address matching tags, that reroute accesses into flash to a special part of SRAM. This

permits patching flash locations for breakpointing and quick fixes or changes.

Formatter The formatter is an internal input block in the ETB and TPIU that embeds the trace source ID

within the data to create a single trace stream.

Halfword A 16-bit data item.

Halt mode One of two mutually exclusive debug modes. In halt mode all processor execution halts when a

breakpoint or watchpoint is encountered. All processor state, coprocessor state, memory and

input/output locations can be examined and altered by the JTAG interface.

See also Monitor debug-mode.

Host A computer that provides data and other services to another computer. Especially, a computer

providing debugging services to a target being debugged.

HTM See AHB Trace Macrocell.

ICode Memory Space at 0x00000000 to 0x1FFFFFFF.

Illegal instruction An instruction that is architecturally Undefined.

Implementation-defined

The behavior is not architecturally defined, but is defined and documented by individual

implementations.

Implementation-specific

The behavior is not architecturally defined, and does not have to be documented by individual

implementations. Used when there are a number of implementation options available and the

option chosen does not affect software compatibility.



Instruction cycle coun

The number of cycles for which an instruction occupies the Execute stage of the pipeline.

Instrumentation trace

A component for debugging real-time systems through a simple memory-mapped trace interface, providing printf style debugging.

Intelligent Energy Management (IEM)

A technology that enables dynamic voltage scaling and clock frequency variation to be used to reduce power consumption in a device.

Internal PPB PPB memory space at 0xE0000000 to 0xE003FFFF.

Interrupt service

routine

A program that control of the processor is passed to when an interrupt occurs.

Interrupt vector One of a number of fixed addresses in low memory that contains the first instruction of the

corresponding interrupt service routine.

Joint Test Action Group (JTAG)

The name of the organization that developed standard IEEE 1149.1. This standard defines a boundary-scan architecture used for in-circuit testing of integrated circuit devices. It is commonly known by the initials JTAG.

JTAG See Joint Test Action Group.

JTAG Debug Port (JTAG-DP)

An optional external interface for the DAP that provides a standard JTAG interface for debug

access.

JTAG-DP See JTAG Debug Port.

Little-endian view of memory in both byte-invariant and word-invariant systems. See also

Byte-invariant, Word-invariant.

Little-endian Byte ordering scheme in which bytes of increasing significance in a data word are stored at

increasing addresses in memory.

See also Big-endian and Endianness.

Little-endian memory

Memory in which:

• a byte or halfword at a word-aligned address is the least significant byte or halfword

within the word at that address

• a byte at a halfword-aligned address is the least significant byte within the halfword at that

address.

See also Big-endian memory.

Load/store architecture

A processor architecture where data-processing operations only operate on register contents, not

directly on memory contents.

Load Store Unit (LSU)

The part of a processor that handles load and store transfers.

LSU See Load Store Unit.

Macrocell A complex logic block with a defined interface and behavior. A typical VLSI system comprises

several macrocells (such as a processor, an ETM, and a memory block) plus application-specific

logic.



most recently written to that location. Memory coherency is made difficult when there are multiple possible physical locations that are involved, such as a system that has main memory,

a write buffer and a cache.

Memory Protection Unit (MPU)

Hardware that controls access permissions to blocks of memory. Unlike an MMU, an MPU does not modify addresses.

Microprocessor See Processor.

Monitor debug-mode

One of two mutually exclusive debug modes. In Monitor debug-mode the processor enables a software abort handler provided by the debug monitor or operating system debug task. When a breakpoint or watchpoint is encountered, this enables vital system interrupts to continue to be serviced while normal program execution is suspended.

See also Halt mode.

MPU See Memory Protection Unit.

An interconnect scheme similar to a cross-bar switch. Each master on the interconnect has a Multi-layer

> direct link to each slave. The link is not shared with other masters. This enables each master to process transfers in parallel with other masters. Contention only occurs in a multi-layer

interconnect at a payload destination, typically the slave.

Nested Vectored Interrupt Controller (NVIC)

Provides the processor with configurable interrupt handling abilities.

NMI See Non-maskable interrupt

Non-maskable interrupt

A Non-Maskable Interrupt (NMI) can be signalled by a peripheral or triggered by software. This is the highest priority exception other than reset. It is permanently enabled and has a fixed priority of -2. NMIs cannot be:

masked or prevented from activation by any other exception

preempted by any exception other than Reset.

NVIC See Nested Vectored Interrupt Controller.

Penalty The number of cycles in which no useful Execute stage pipeline activity can occur because an

instruction flow is different from that assumed or predicted.

PFU See Prefetch Unit.

PMU See Power Management Unit.

Power Management Unit (PMU)

Provides the processor with power management capability.

Power-on reset See Cold reset.

PPB See Private Peripheral Bus.

Prefetching In pipelined processors, the process of fetching instructions from memory to fill up the pipeline

before the preceding instructions have finished executing. Prefetching an instruction does not

mean that the instruction has to be executed.



Prefetch Abort An indication from a memory system to the core that an instruction has been fetched from an

illegal memory location. An exception must be taken if the processor attempts to execute the instruction. A Prefetch Abort can be caused by the external or internal memory system as a

result of attempting to access invalid instruction memory.

See also Data Abort, Abort.

Prefetch Unit (PFU) The PFU fetches instructions from the memory system that can supply one word each cycle. The

PFU buffers up to three word fetches in its FIFO, which means that it can buffer up to three

32-bit Thumb instructions or six 16-bit Thumb instructions.

Private Peripheral Bus

Memory space at 0xE0000000 to 0xE00FFFFF.

Processor A processor is the circuitry in a computer system required to process data using the computer

instructions. It is an abbreviation of microprocessor. A clock source, power supplies, and main

memory are also required to create a minimum complete working computer system.

RW1C Register bits marked RW1C can be read normally and support write-one-to-clear. A read then

write of the result back to the register will clear all bits set. RW1C protects against

read-modify-write errors occurring on bits set between reading the register and writing the value

back (since they are written as zero, they will not be cleared).

RealView ICE A system for debugging embedded processor cores using a JTAG interface.

Reserved A field in a control register or instruction format is reserved if the field is to be defined by the

implementation, or produces Unpredictable results if the contents of the field are not zero. These

fields are reserved for use in future extensions of the architecture or are

implementation-specific. All reserved bits not used by the implementation must be written as 0

and read as 0.

Scan chain A scan chain is made up of serially-connected devices that implement boundary scan technology

using a standard JTAG TAP interface. Each device contains at least one TAP controller

containing shift registers that form the chain connected between **TDI** and **TDO**, through which test data is shifted. Processors can contain several shift registers to enable you to access selected

parts of the device.

Serial-Wire Debug Port

An optional external interface for the DAP that provides a serial-wire bidirectional debug

interface.

Serial-Wire JTAG

Debug Port

A standard debug port that combines JTAG-DP and SW-DP.

SW-DP *See* Serial-Wire Debug Port.

SWJ-DP *See* Serial-Wire JTAG Debug Port.

Synchronization primitive

The memory synchronization primitive instructions are those instructions that are used to ensure

memory synchronization. That is, the LDREX and STREX instructions.

System memory Memory space at 0x20000000 to 0xFFFFFFF, excluding PPB space at 0xE0000000 to 0xE00FFFFF.

TAP See Test access port.

Test Access Port (TAP)

The collection of four mandatory and one optional terminals that form the input/output and control interface to a JTAG boundary-scan architecture. The mandatory terminals are **TDI**, **TDO**, **TMS**, and **TCK**. The optional terminal is **TRST**. This signal is mandatory in ARM cores

because it is used to reset the debug logic.



Thread Control Block A data structure used by an operating system kernel to maintain information specific to a single

thread of execution.

Thumb instruction A halfword that specifies an operation for an ARM processor in Thumb state to perform. Thumb

instructions must be halfword-aligned.

Thumb state A processor that is executing Thumb (16-bit) halfword aligned instructions is operating in

Thumb state.

TPA See *Trace Port Analyzer*.

TPIU See Trace Port Interface Unit.

Trace Port Analyzer (TPA)

A hardware device that captures trace information output on a trace port. This can be a low-cost

product designed specifically for trace acquisition, or a logic analyzer.

Trace Port Interface Unit (TPIU)

Drains trace data and acts as a bridge between the on-chip trace data and the data stream

captured by a TPA.

Unaligned A data item stored at an address that is not divisible by the number of bytes that defines the data

size is said to be unaligned. For example, a word stored at an address that is not divisible by four.

Wake-up Interrupt Controller (WIC)

The Wake-up Interrupt Controller provides significantly reduced gate count interrupt detection

and prioritization logic.

Warm reset Also known as a core reset. Initializes the majority of the processor excluding the debug

controller and debug logic. This type of reset is useful if you are using the debugging features

of a processor.

Watchpoint A watchpoint is a mechanism provided by debuggers to halt program execution when the data

contained by a particular memory address is changed. Watchpoints are inserted by the programmer to enable inspection of register contents, memory locations, and variable values when memory is written to test that the program is operating correctly. Watchpoints are removed

after the program is successfully tested. See also Breakpoint.

WIC See Wake-up Interrupt Controller.

Word A 32-bit data item.

Word-invariant In a word-invariant system, the address of each byte of memory changes when switching

between little-endian and big-endian operation, in such a way that the byte with address A in one endianness has address A EOR 3 in the other endianness. As a result, each aligned word of memory always consists of the same four bytes of memory in the same order, regardless of endianness. The change of endianness occurs because of the change to the byte addresses, not

because the bytes are rearranged.

The ARM architecture supports word-invariant systems in ARMv3 and later versions. When word-invariant support is selected, the behavior of load or store instructions that are given unaligned addresses is instruction-specific, and is in general not the expected behavior for an unaligned access. It is recommended that word-invariant systems use the endianness that produces the required byte addresses at all times, apart possibly from very early in their reset handlers before they have set up the endianness, and that this early part of the reset handler must

use only aligned word memory accesses.

See also Byte-invariant.

Write buffer A pipeline stage for buffering write data to prevent bus stalls from stalling the processor.