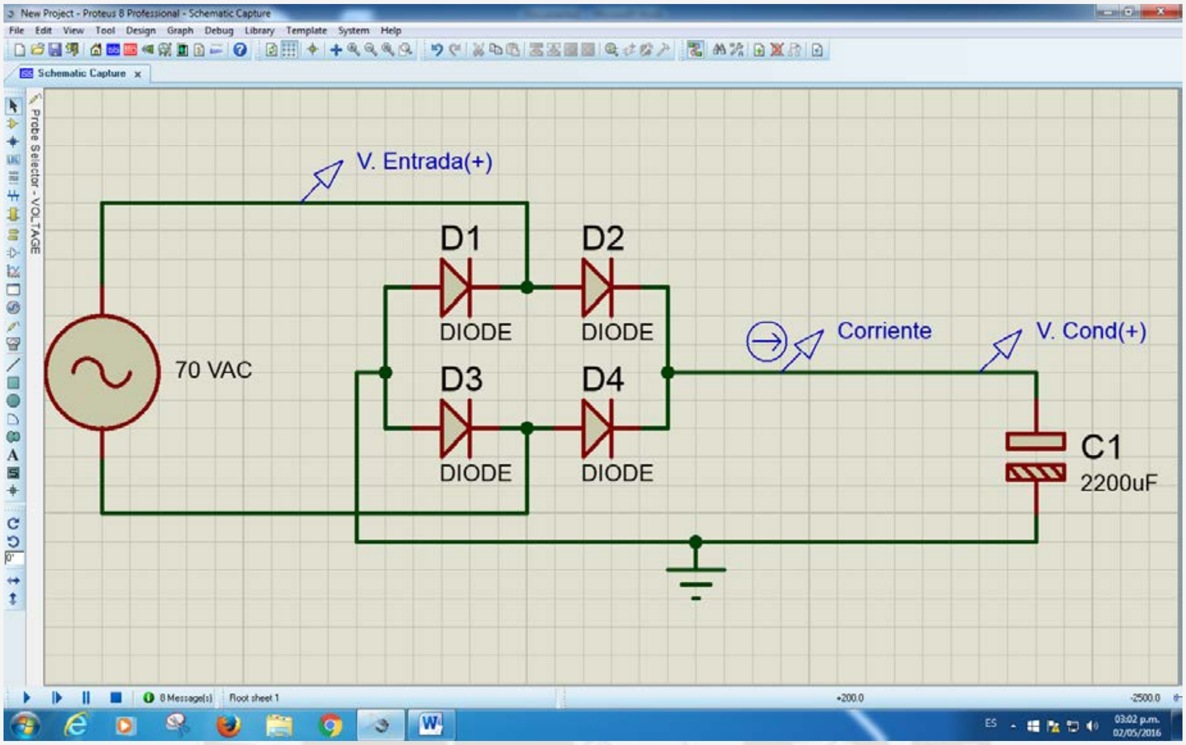
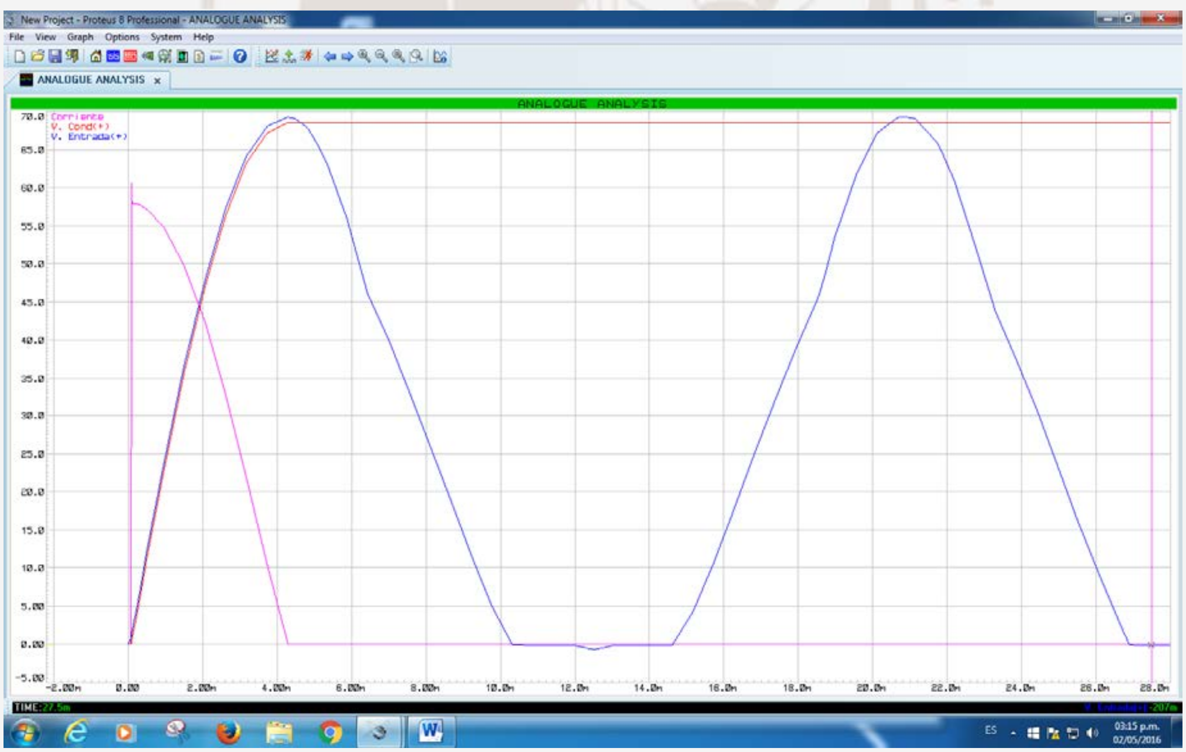
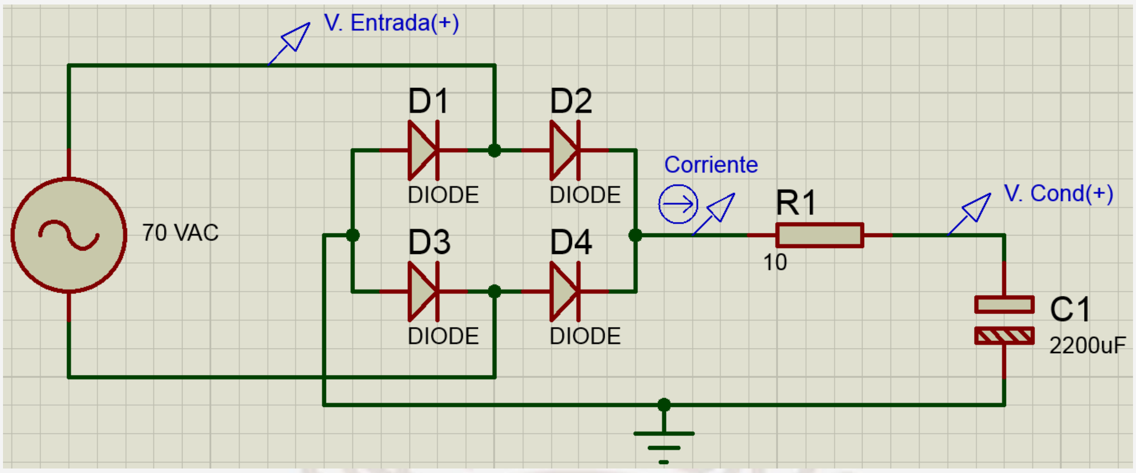
INDICE DE ANEXOS

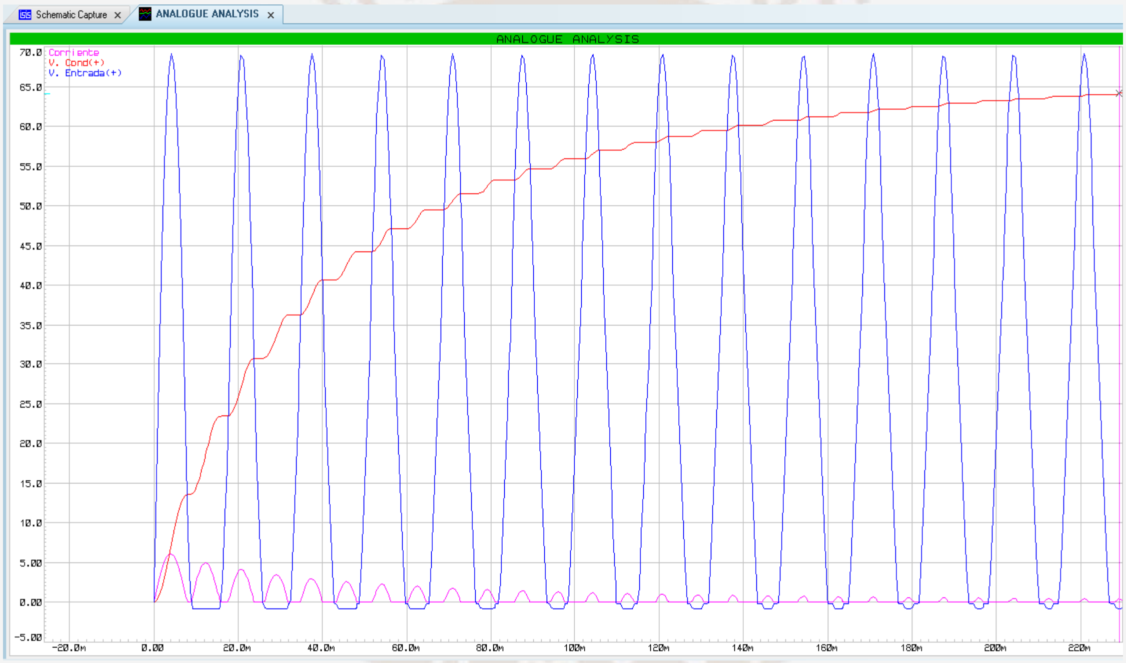
ANEXO1: SIMULACIÓN EN PROTEUS DE CIRCUITO DE RECTIFICACIÓN SIN RESISTENCIA MINIMIZADORA DE CORRIENTE INICIAL



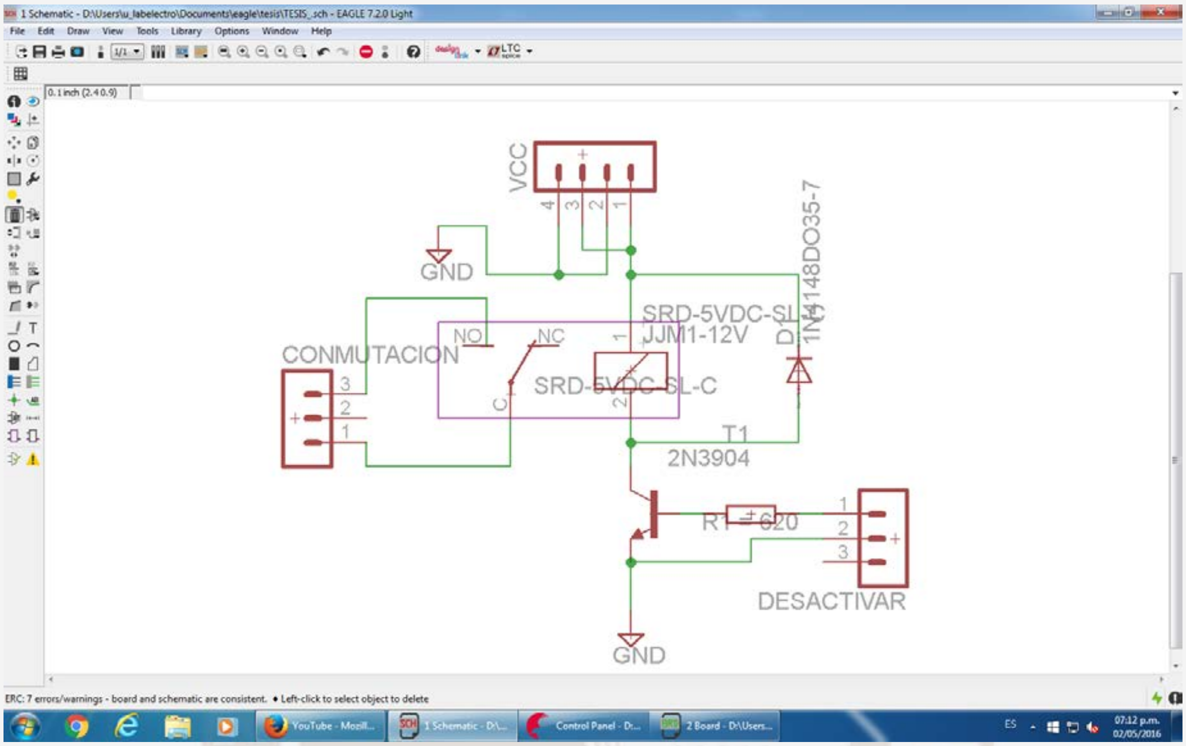


ANEXO 2: SIMULACIÓN EN PROTEUS DE CIRCUITO DE RECTIFICACIÓN CON LA RESISTENCIA MINIMIZADORA DE CORRIENTE INICIAL





ANEXO 3: DIAGRAMA ESQUEMÁTICO DEL CIRCUITO DE PROTECCIÓN CONTRA LA CORRIENTE DE IRRUPCIÓN

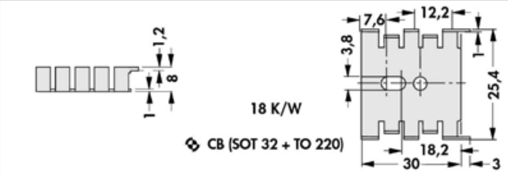


fischer elektronik

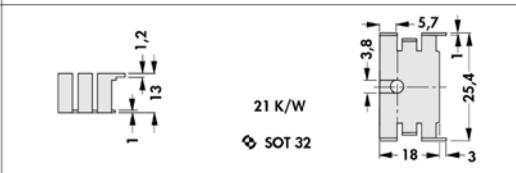
Heatsinks for transistors in plastic case

art. no. FK 211 32 ...

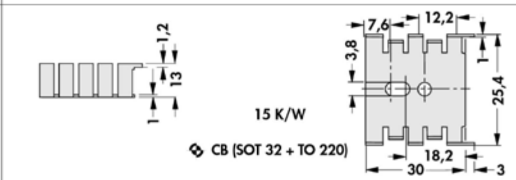
art. no. FK 212 CB ...



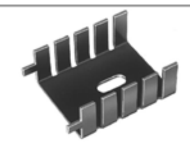
art. no. FK 215 32 ...

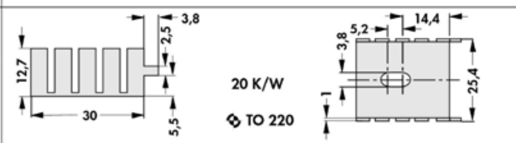


art. no. FK 216 SA

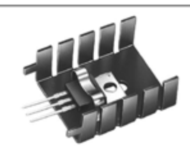


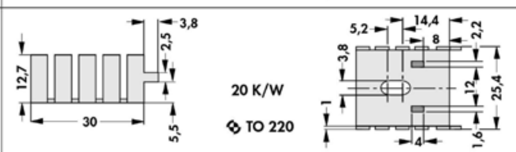
art. no. FK 222 ...





art. no. FK 222 THF ...





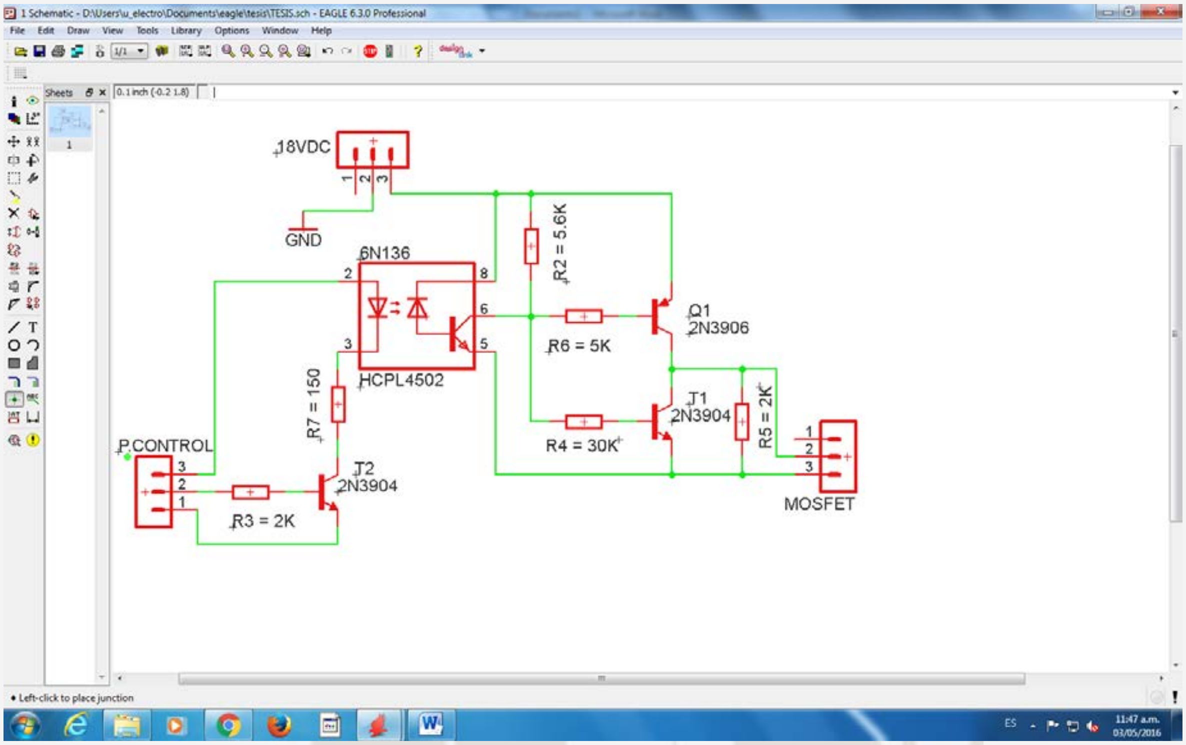
art. no. FK 247 220 ...

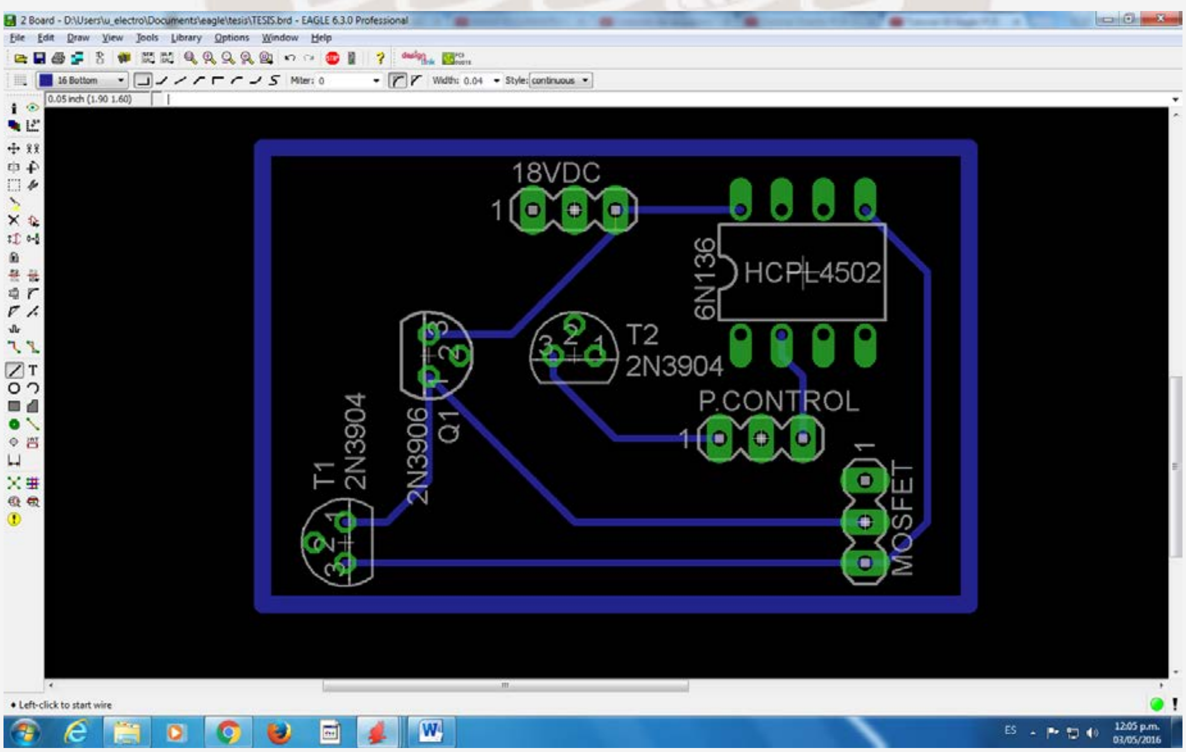
please indicate: ... **surface treatment**
SA=black anodised
MI=solderable

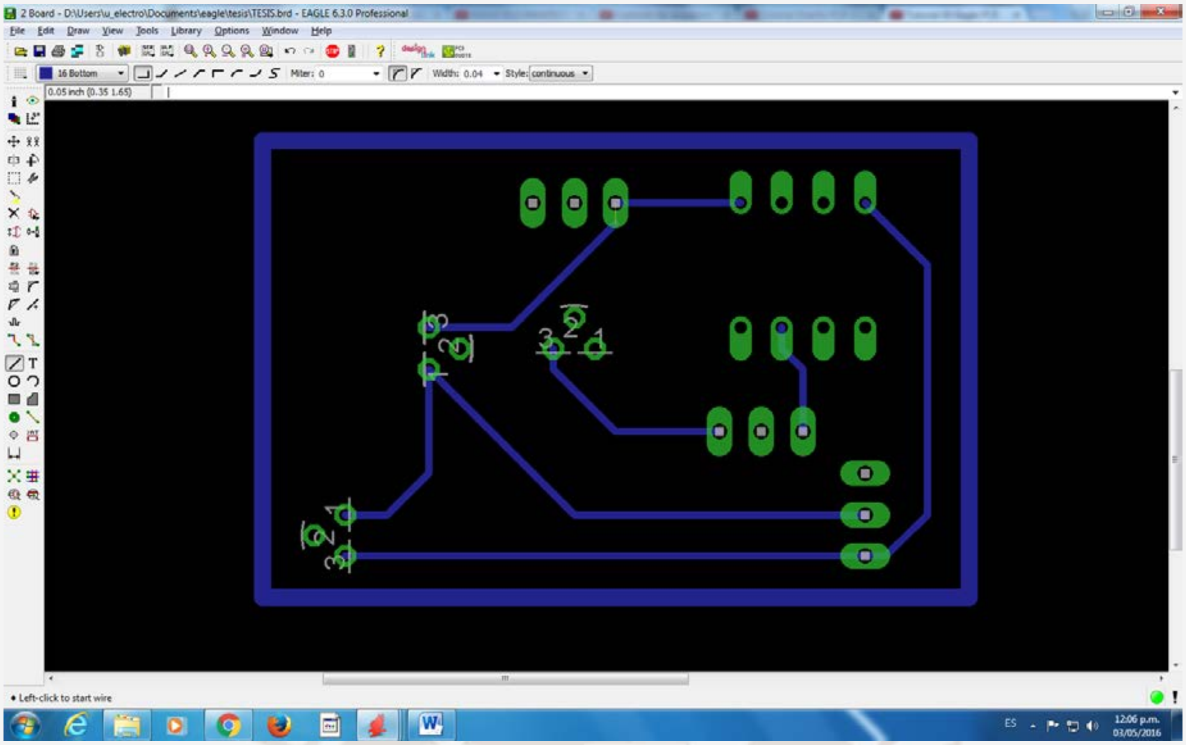
The heatsinks FK 211 ... 216 are available without hole pattern as well, e.g. FK 211 SA

Retaining springs for transistors → A 111 – 116
Heatsinks for D PAK → C 17
GEL thermal conductive foil → E 7
Insulator caps → E 40
Mounting material for semiconduct. → E 34 – 38
Mounting pads → E 36
Mounting pads for transistors → E 37
Mounting parts for heatsinks → E 39 – 40

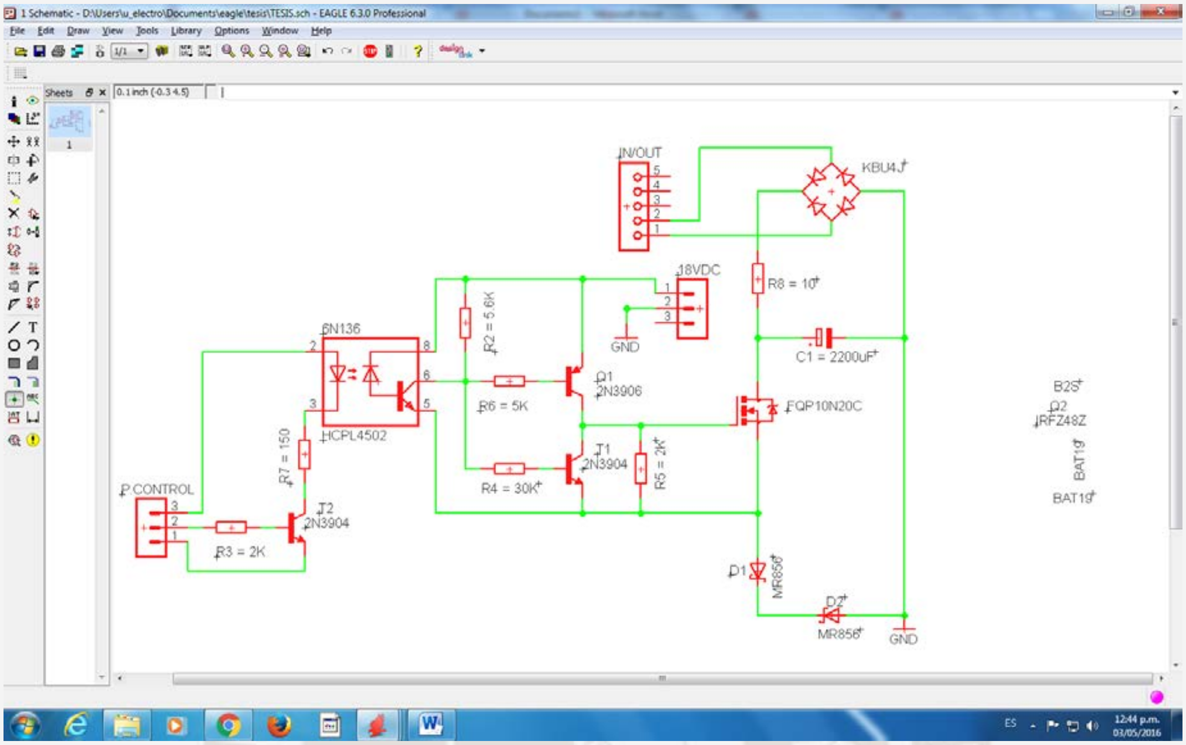
ANEXO 5: ESQUEMÁTICO DEL CIRCUITO DE MANDO DE COMPUERTA CON EL RESPECTIVO BOARD.

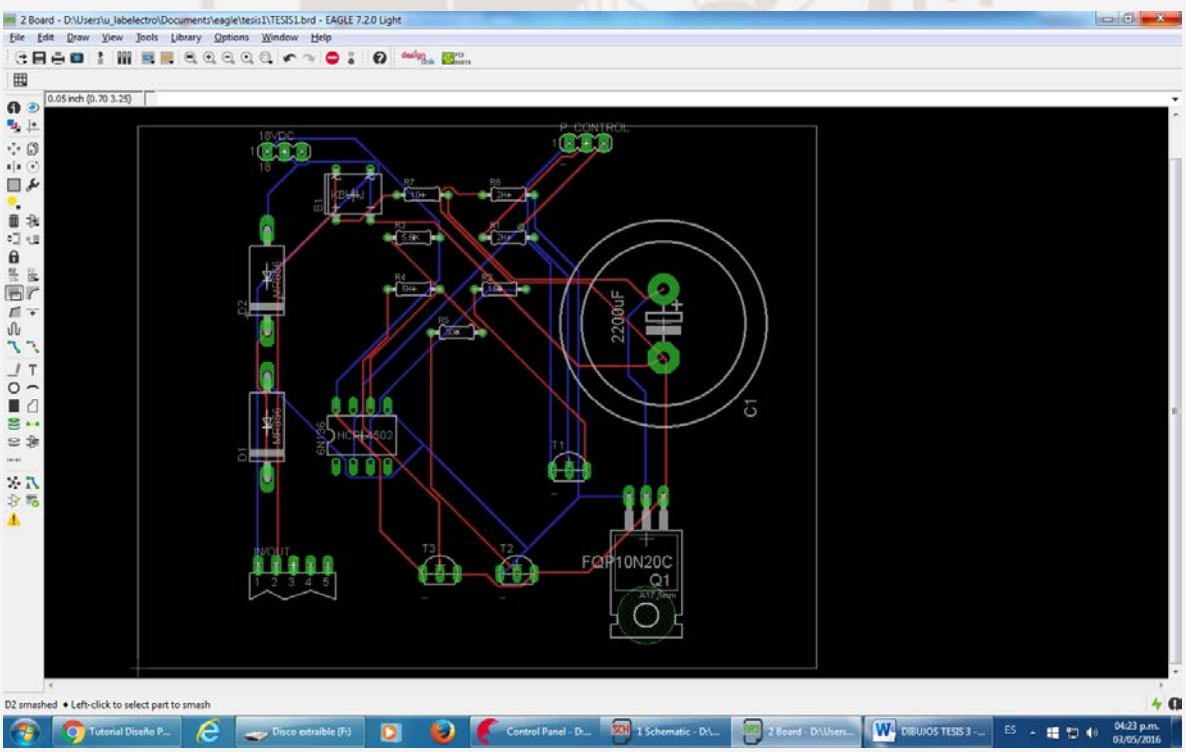




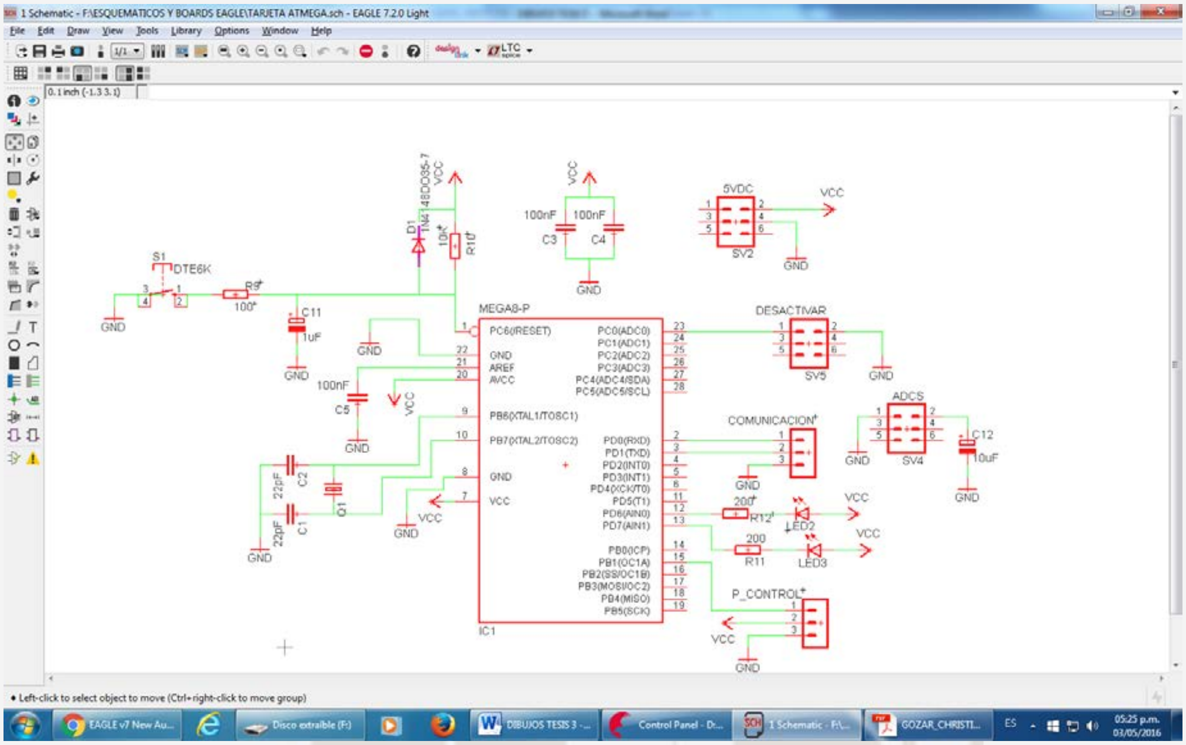

2 Board - D:\Users\u_electro\Documents\eagle\tesis\TESIS.brd - EAGLE 6.3.0 Professional
File Edit Draw View Tools Library Options Window Help
16 Bottom
Miter: 0
Width: 0.04
Style: continuous
0.05 inch (0.35 1.65)
Left-click to start wire
12:06 p.m.
03/05/2016

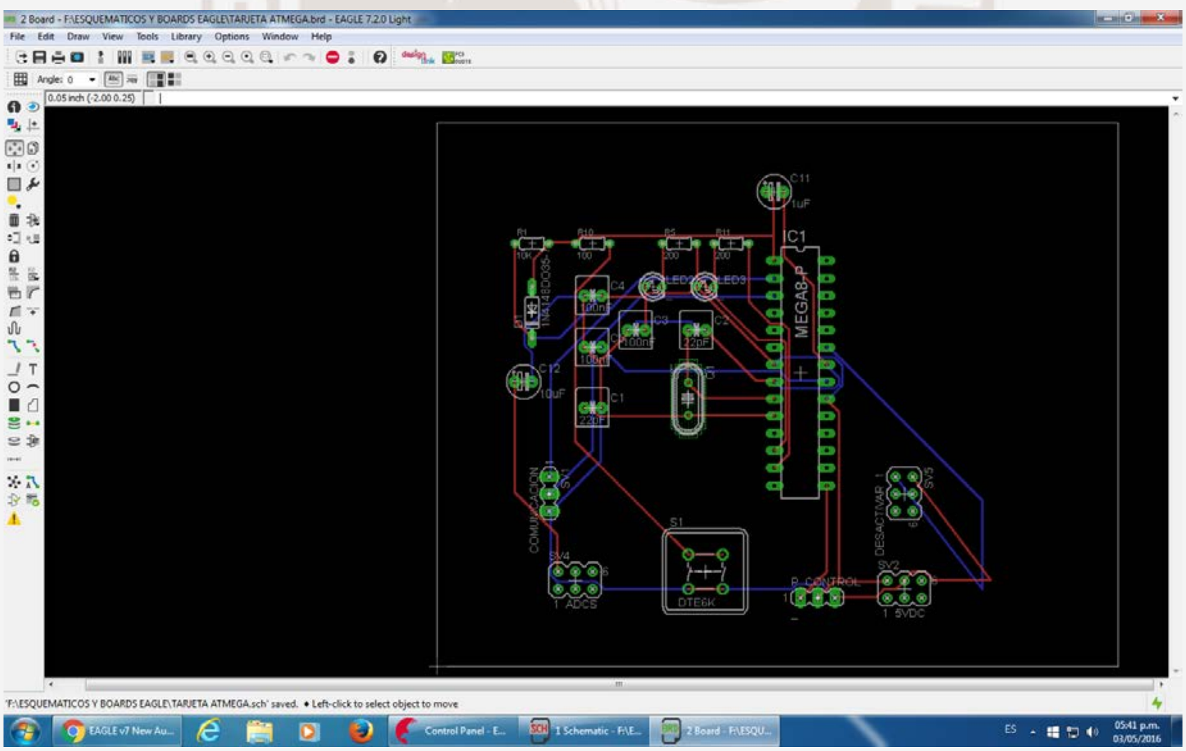
ANEXO 6: ESQUEMÁTICO DE LA FUENTE PARA EL CIRCUITO DE CAMPO DEL MOTOR CON EL RESPECTIVO BOARD.



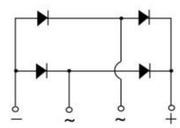


ANEXO 7: ESQUEMÁTICO DEL CIRCUITO DE CONTROL CON EL RESPECTIVO BOARD.





Single-Phase Bridge Rectifier



Case Style KBU

PRIMARY CHARACTERISTICS	
Package	KBU
$I_{F(AV)}$	4 A
V_{RRM}	50 V, 100, V, 200 V, 400 V, 600 V, 800 V, 1000 V
I_{FSM}	200 A
I_R	5 μA
V_F at I_F = 4 A	1.0 V
T_J max.	150 °C
Diode variations	In-Line

FEATURES

- UL recognition, file number E54214
- Ideal for printed circuit boards
- High surge current capability
- High case dielectric strength of 1500 V_{RMS}
- Solder dip 275 °C max. 10 s, per JESD 22-B106
- Material categorization: for definitions of compliance please see www.vishay.com/doc?99912

RoHS COMPLIANT

TYPICAL APPLICATIONS

General purpose use in AC/DC bridge full wave rectification for monitor, TV, printer, SMPS, adapter, audio equipment, and home appliances applications.

MECHANICAL DATA

Case: KBU

Molding compound meets UL 94 V-0 flammability rating Base P/N-E4 - RoHS-compliant, commercial grade

Terminals: Silver plated leads, solderable per J-STD-002 and JESD22-B102

Polarity: As marked on body

Mounting Torque: 10 cm-kg (8.8 inches-lbs) max.

Recommended Torque: 5.7 cm-kg (5 inches-lbs)

MAXIMUM RATINGS (T_A = 25 °C unless otherwise noted)

PARAMETER		SYMBOL	KBU4A	KBU4B	KBU4D	KBU4G	KBU4J	KBU4K	KBU4M	UNIT
Maximum repetitive peak reverse voltage		V_{RRM}	50	100	200	400	600	800	1000	V
Maximum RMS voltage		V_{RMS}	35	70	140	280	420	560	700	V
Maximum DC blocking voltage		V_{DC}	50	100	200	400	600	800	1000	V
Maximum average forward rectified output current at	T_C = 100 °C (1)	$I_{F(AV)}$	4.0							A
	T_A = 30 °C (2)		4.0							
Peak forward surge current single sine-wave superimposed on rated load		I_{FSM}	200							A
Operating junction and storage temperature range		T_J, T_{STG}	- 50 to + 150							°C

Notes

(1) Units mounted on a 2.0" x 1.6" x 0.3" thick (5 cm x 4 cm x 0.8 cm) aluminum plate

(2) Units mounted on PCB with 0.5" x 0.5" (12 mm x 12 mm) copper pads and 0.375" (9.5 mm) lead length

ELECTRICAL CHARACTERISTICS (T_A = 25 °C unless otherwise noted)

PARAMETER	TEST CONDITIONS	SYMBOL	KBU4A	KBU4B	KBU4D	KBU4G	KBU4J	KBU4K	KBU4M	UNIT
Maximum instantaneous forward drop per diode	I_F = 4.0 A	V_F	1.0							V
Maximum DC reverse current at rated DC blocking voltage per diode	T_A = 25 °C	I_R	5.0							μA
	T_A = 125 °C		1.0							mA

THERMAL CHARACTERISTICS (T_A = 25 °C unless otherwise noted)									
PARAMETER	**SYMBOL**	**KBU4A**	**KBU4B**	**KBU4D**	**KBU4G**	**KBU4J**	**KBU4K**	**KBU4M**	**UNIT**
Typical thermal resistance	$R_{\theta JA}$	19 [(2)]							°C/W
	$R_{\theta JL}$	4.0 [(1)]							

Notes

(1) Units mounted on a 2.0" x 1.6" x 0.3" thick (5 cm x 4 cm x 0.8 cm) aluminum plate

(2) Units mounted on PCB with 0.5" x 0.5" (12 mm x 12 mm) copper pads and 0.375" (9.5 mm) lead length

ORDERING INFORMATION (Example)				
PREFERRED P/N	**UNIT WEIGHT (g)**	**PREFERRED PACKAGE CODE**	**BASE QUANTITY**	**DELIVERY MODE**
KBU4J-E4/51	8.0	51	250	Anti-static PVC tray

RATINGS AND CHARACTERISTICS CURVES (T_A = 25 °C unless otherwise noted)

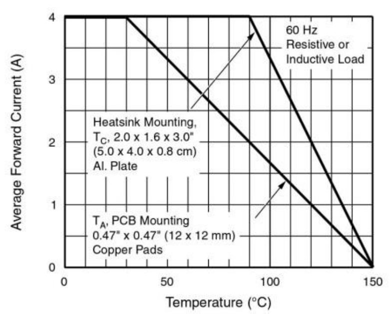


Fig. 1 - Derating Curve Output Rectified Current

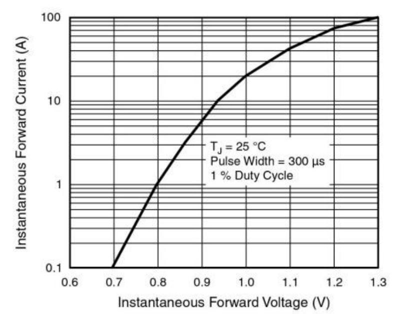


Fig. 3 - Typical Forward Characteristics Per Diode

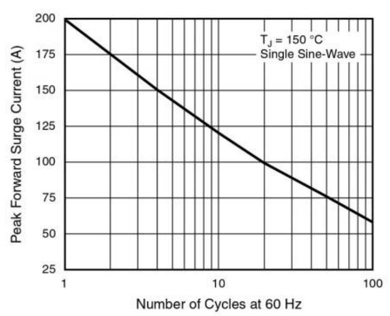


Fig. 2 - Maximum Non-Repetitive Peak Forward Surge Current Per Diode

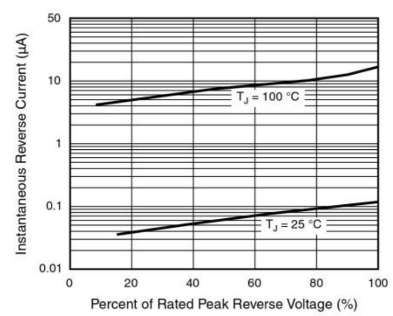


Fig. 4 - Typical Reverse Leakage Characteristics Per Diode

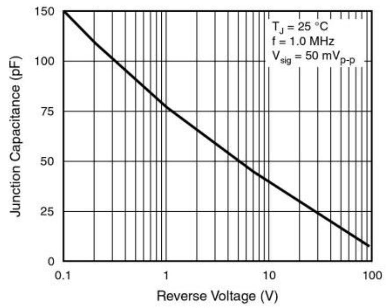
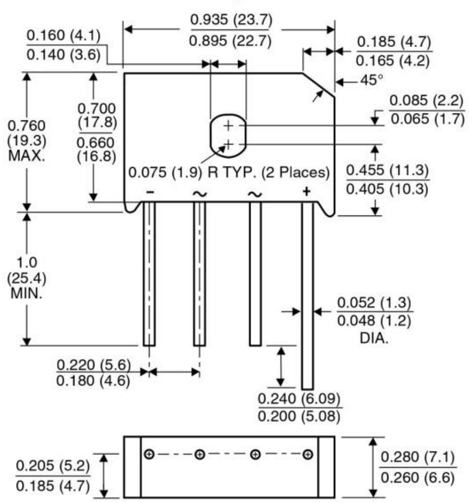


Fig. 5 - Typical Junction Capacitance Per Diode

PACKAGE OUTLINE DIMENSIONS in inches (millimeters)

Case Style KBU



Disclaimer

ALL PRODUCT, PRODUCT SPECIFICATIONS AND DATA ARE SUBJECT TO CHANGE WITHOUT NOTICE TO IMPROVE RELIABILITY, FUNCTION OR DESIGN OR OTHERWISE.

Vishay Intertechnology, Inc., its affiliates, agents, and employees, and all persons acting on its or their behalf (collectively, “Vishay”), disclaim any and all liability for any errors, inaccuracies or incompleteness contained in any datasheet or in any other disclosure relating to any product.

Vishay makes no warranty, representation or guarantee regarding the suitability of the products for any particular purpose or the continuing production of any product. To the maximum extent permitted by applicable law, Vishay disclaims (i) any and all liability arising out of the application or use of any product, (ii) any and all liability, including without limitation special, consequential or incidental damages, and (iii) any and all implied warranties, including warranties of fitness for particular purpose, non-infringement and merchantability.

Statements regarding the suitability of products for certain types of applications are based on Vishay’s knowledge of typical requirements that are often placed on Vishay products in generic applications. Such statements are not binding statements about the suitability of products for a particular application. It is the customer’s responsibility to validate that a particular product with the properties described in the product specification is suitable for use in a particular application. Parameters provided in datasheets and/or specifications may vary in different applications and performance may vary over time. All operating parameters, including typical parameters, must be validated for each customer application by the customer’s technical experts. Product specifications do not expand or otherwise modify Vishay’s terms and conditions of purchase, including but not limited to the warranty expressed therein.

Except as expressly indicated in writing, Vishay products are not designed for use in medical, life-saving, or life-sustaining applications or for any other application in which the failure of the Vishay product could result in personal injury or death. Customers using or selling Vishay products not expressly indicated for use in such applications do so at their own risk. Please contact authorized Vishay personnel to obtain written terms and conditions regarding products designed for such applications.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document or by any conduct of Vishay. Product names and markings noted herein may be trademarks of their respective owners.

Material Category Policy

Vishay Intertechnology, Inc. hereby certifies that all its products that are identified as RoHS-Compliant fulfill the definitions and restrictions defined under Directive 2011/65/EU of The European Parliament and of the Council of June 8, 2011 on the restriction of the use of certain hazardous substances in electrical and electronic equipment (EEE) - recast, unless otherwise specified as non-compliant.

Please note that some Vishay documentation may still make reference to RoHS Directive 2002/95/EC. We confirm that all the products identified as being compliant to Directive 2002/95/EC conform to Directive 2011/65/EU.

Vishay Intertechnology, Inc. hereby certifies that all its products that are identified as Halogen-Free follow Halogen-Free requirements as per JEDEC JS709A standards. Please note that some Vishay documentation may still make reference to the IEC 61249-2-21 definition. We confirm that all the products identified as being compliant to IEC 61249-2-21 conform to JEDEC JS709A standards.

FAIRCHILD SEMICONDUCTOR®

November 2013

FQP10N20C / FQPF10N20C

N-Channel QFET® MOSFET

200 V, 9.5 A, 360 mΩ

Features

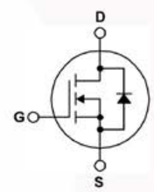
- 9.5 A, 200 V, $R_{DS(on)}$ = 360 mΩ (Max.) @ V_{GS} = 10 V, I_D = 4.75 A
- Low Gate Charge (Typ. 20 nC)
- Low Crss (Typ. 40.5 pF)
- 100% Avalanche Tested

Description

This N-Channel enhancement mode power MOSFET is produced using Fairchild Semiconductor's proprietary planar stripe and DMOS technology. This advanced MOSFET technology has been especially tailored to reduce on-state resistance, and to provide superior switching performance and high avalanche energy strength. These devices are suitable for switched mode power supplies, active power factor correction (PFC), and electronic lamp ballasts.

TO-220

TO-220F



MOSFET Maximum Ratings T_C = 25°C unless otherwise noted.

Symbol	Parameter			FQP10N20C	FQPF10N20C	Unit
V_{DSS}	Drain to Source Voltage			200		V
I_D	Drain Current	-Continuous (T_C = 25°C)		9.5	9.5 *	A
		-Continuous (T_C = 100°C)		6.0	6.0 *	A
I_{DM}	Drain Current	- Pulsed	(Note 1)	38	38 *	A
V_{GSS}	Gate to Source Voltage			± 30		V
E_{AS}	Single Pulsed Avalanche Energy		(Note 2)	210		mJ
I_{AR}	Avalanche Current		(Note 1)	9.5		A
E_{AR}	Repetitive Avalanche Energy		(Note 1)	7.2		mJ
dv/dt	Peak Diode Recovery dv/dt		(Note 3)	5.5		V/ns
P_D	Power Dissipation	(T_C = 25°C)		72	38	W
		- Derate above 25°C		0.57	0.3	W/°C
T_J, T_{STG}	Operating and Storage Temperature Range			-55 to +150		°C
T_L	Maximum Lead Temperature for Soldering Purpose, 1/8" from Case for 5 Seconds			300		°C

*Drain current limited by maximum junction temperature

Thermal Characteristics

Symbol	Parameter	FQP10N20C	FQPF10N20C	Unit
$R_{\theta JC}$	Thermal Resistance, Junction to Case, Max	1.74	3.33	°C/W
$R_{\theta JA}$	Thermal Resistance, Junction to Ambient, Max	62.5	62.5	°C/W

Package Marking and Ordering Information

Device Marking	Device	Package	Reel Size	Tape Width	Quantity
FQP10N20C	FQP10N20C	TO-220	Tube	N/A	50 units
FQPF10N20C	FQPF10N20C	TO-220F	Tube	N/A	50 units

Electrical Characteristics $T_C = 25^oC$ unless otherwise noted.

Symbol	Parameter	Test Conditions	Min	Typ	Max	Unit
Off Characteristics						
BV_{DSS}	Drain-Source Breakdown Voltage	$V_{GS} = 0\ V$, $I_D = 250\ \mu A$	200	--	--	V
$\Delta BV_{DSS} / \Delta T_J$	Breakdown Voltage Temperature Coefficient	$I_D = 250\ \mu A$, Referenced to 25°C	--	0.28	--	V/°C
I_{DSS}	Zero Gate Voltage Drain Current	$V_{DS} = 200\ V$, $V_{GS} = 0\ V$	--	--	10	µA
		$V_{DS} = 160\ V$, $T_C = 125°C$	--	--	100	µA
I_{GSSF}	Gate-Body Leakage Current, Forward	$V_{GS} = 30\ V$, $V_{DS} = 0\ V$	--	--	100	nA
I_{GSSR}	Gate-Body Leakage Current, Reverse	$V_{GS} = -30\ V$, $V_{DS} = 0\ V$	--	--	-100	nA
On Characteristics						
$V_{GS(th)}$	Gate Threshold Voltage	$V_{DS} = V_{GS}$, $I_D = 250\ \mu A$	2.0	--	4.0	V
$R_{DS(on)}$	Static Drain-Source On-Resistance	$V_{GS} = 10\ V$, $I_D = 4.75\ A$	--	0.29	0.36	Ω
g_{FS}	Forward Transconductance	$V_{DS} = 40\ V$, $I_D = 4.75\ A$	--	5.5	--	S
Dynamic Characteristics						
C_{iss}	Input Capacitance	$V_{DS} = 25\ V$, $V_{GS} = 0\ V$, f = 1.0 MHz	--	395	510	pF
C_{oss}	Output Capacitance		--	97	125	pF
C_{rss}	Reverse Transfer Capacitance		--	40.5	53	pF
Switching Characteristics						
$t_{d(on)}$	Turn-On Delay Time	$V_{DD} = 100\ V$, $I_D = 9.5\ A$, $R_G = 25\ \Omega$ (Note 4)	--	11	30	ns
t_r	Turn-On Rise Time		--	92	190	ns
$t_{d(off)}$	Turn-Off Delay Time		--	70	150	ns
t_f	Turn-Off Fall Time		--	72	160	ns
Q_g	Total Gate Charge	$V_{DS} = 160\ V$, $I_D = 9.5\ A$, $V_{GS} = 10\ V$ (Note 4)	--	20	26	nC
Q_{gs}	Gate-Source Charge		--	3.1	--	nC
Q_{gd}	Gate-Drain Charge		--	10.5	--	nC
Drain-Source Diode Characteristics and Maximum Ratings						
I_S	Maximum Continuous Drain-Source Diode Forward Current		--	--	9.5	A
I_{SM}	Maximum Pulsed Drain-Source Diode Forward Current		--	--	38	A
V_{SD}	Drain-Source Diode Forward Voltage	$V_{GS} = 0\ V$, $I_S = 9.5\ A$	--	--	1.5	V
t_{rr}	Reverse Recovery Time	$V_{GS} = 0\ V$, $I_S = 9.5\ A$, $dI_F / dt = 100\ A/\mu s$	--	158	--	ns
Q_{rr}	Reverse Recovery Charge		--	0.97	--	µC

Notes:

1. Repetitive Rating : Pulse width limited by maximum junction temperature.
2. L = 3.5 mH, $I_{AS} = 9.5\ A$, $V_{DD} = 50\ V$, $R_G = 25\ \Omega$, starting $T_J = 25°C$.
3. $I_{SD} \leq 9.5\ A$, $di/dt \leq 300\ A/\mu s$, $V_{DD} \leq BV_{DSS}$, starting $T_J = 25°C$.
4. Essentially independent of operating temperature.

Typical Characteristics

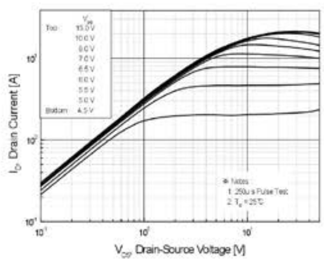


Figure 1. On-Region Characteristics

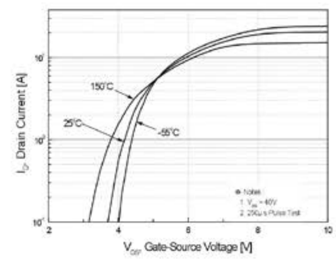


Figure 2. Transfer Characteristics

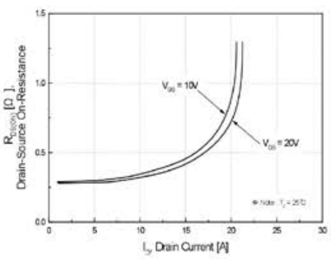


Figure 3. On-Resistance Variation vs Drain Current and Gate Voltage

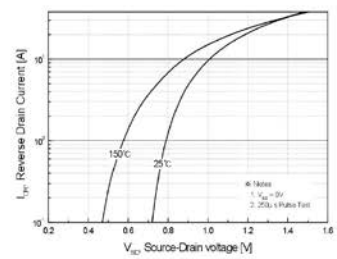


Figure 4. Body Diode Forward Voltage Variation with Source Current and Temperature

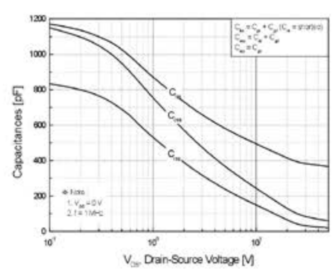


Figure 5. Capacitance Characteristics

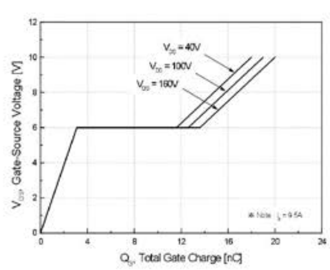


Figure 6. Gate Charge Characteristics

Typical Characteristics (Continued)

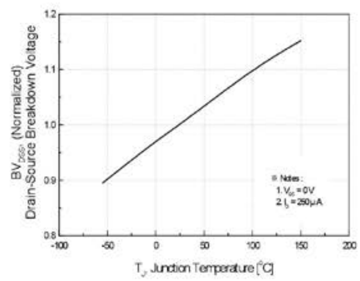


Figure 7. Breakdown Voltage Variation vs Temperature

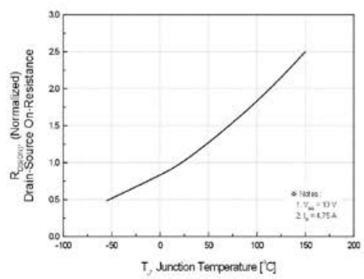


Figure 8. On-Resistance Variation vs Temperature

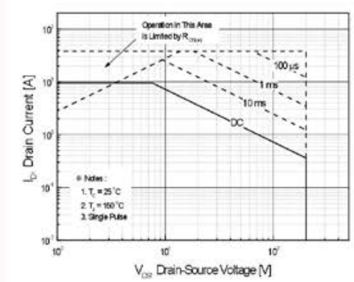


Figure 9-1. Maximum Safe Operating Area for FQP10N20C

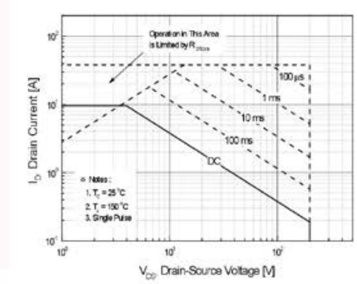


Figure 9-2. Maximum Safe Operating Area for FQPF10N20C

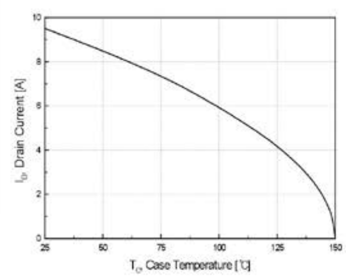


Figure 10. Maximum Drain Current vs Case Temperature

Typical Characteristics (Continued)

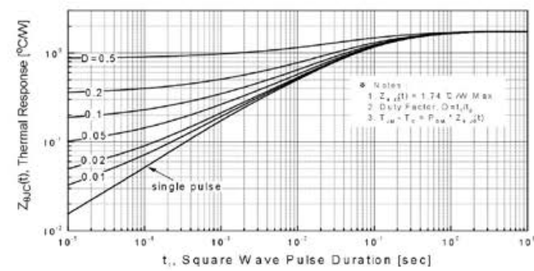


Figure 11-1. Transient Thermal Response Curve for FQP10N20C

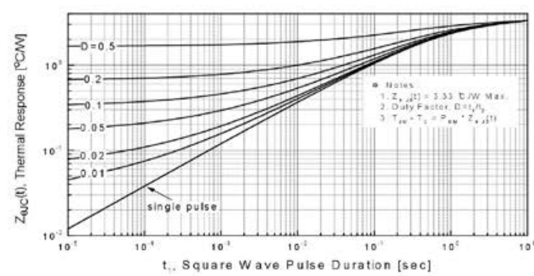


Figure 11-2. Transient Thermal Response Curve for FQPF10N20C

Figure 12. Gate Charge Test Circuit & Waveform

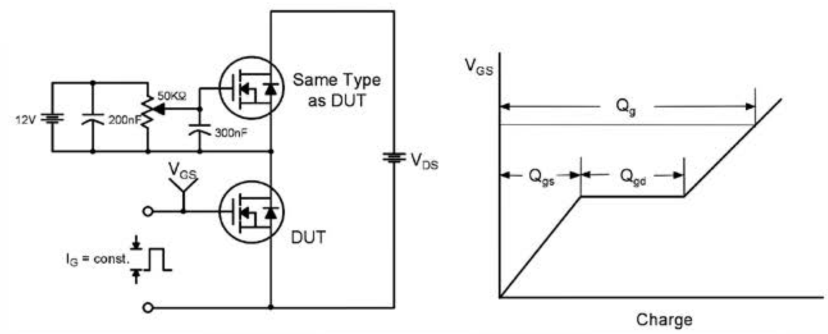


Figure 13. Resistive Switching Test Circuit & Waveforms

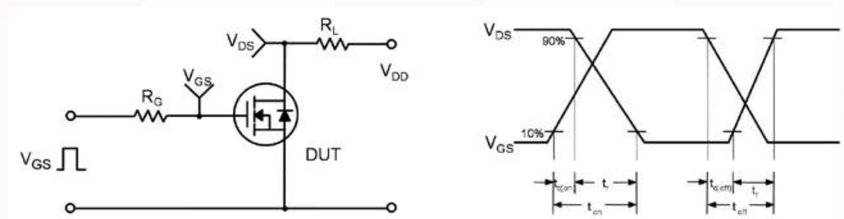


Figure 14. Unclamped Inductive Switching Test Circuit & Waveforms

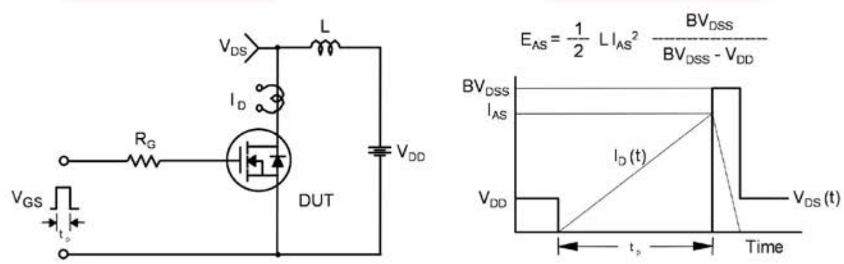
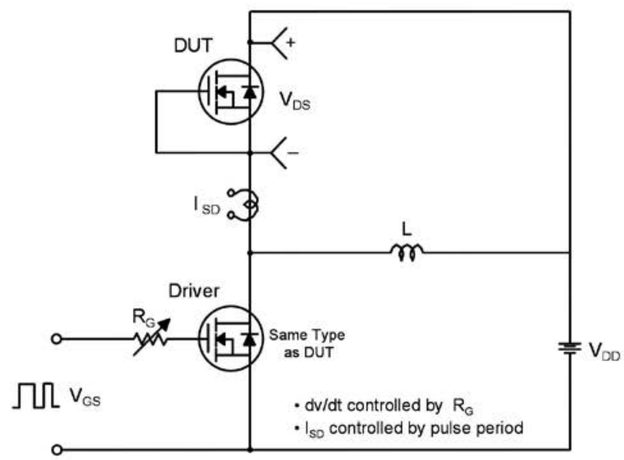
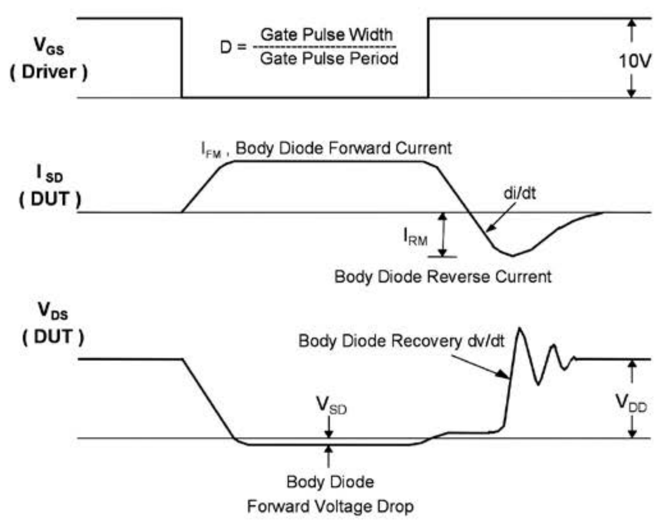


Figure 15. Peak Diode Recovery dv/dt Test Circuit & Waveforms





Mechanical Dimensions

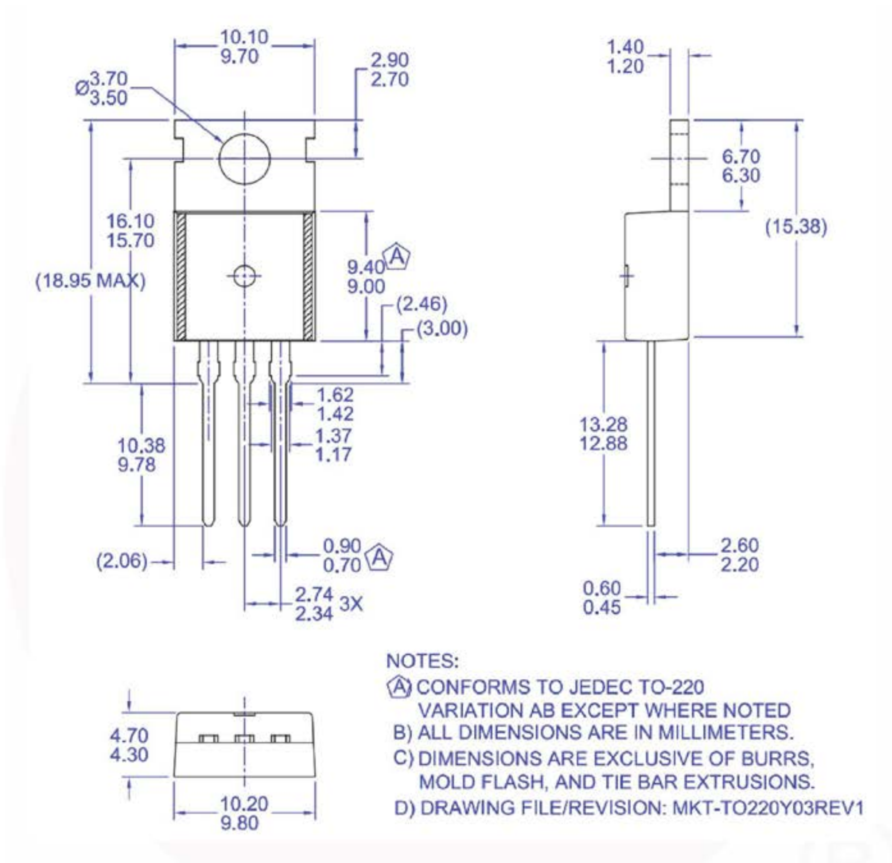


Figure 16. TO220, Molded, 3-Lead, Jedec Variation AB

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/package/packageDetails.html?id=PN_TO220-003

Mechanical Dimensions

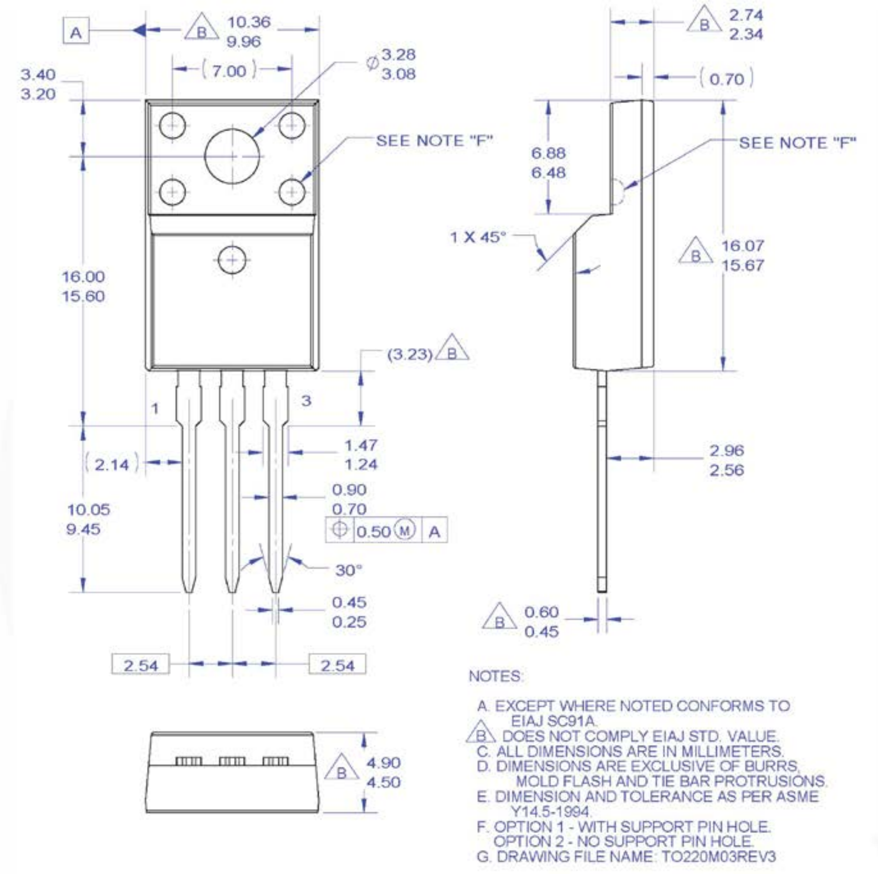


Figure 17. TO220, Molded, 3-Lead, Full Pack, EIAJ SC91, Straight Lead

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/package/packageDetails.html?id=PN_TF220-003

FAIRCHILD SEMICONDUCTOR®

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®*
BitSiC™
Build it Now™
CorePLUS™
CorePOWER™
CROSSVOLT™
CTL™
Current Transfer Logic™
DEUXPEED®
Dual Cool™
EcoSPARK®
EfficentMax™
ESBC™
F®
Fairchild®
Fairchild Semiconductor®
FACT Quiet Series™
FACT®
FAST®
FastvCore™
FETBench™
FPS™
F-PFS™
FRFET®
Global Power Resource℠
GreenBridge™
Green FPS™
Green FPS™ e-Series™
*G*max™
GTO™
IntelliMAX™
ISOPLANAR™
Marking Small Speakers Sound Louder and Better™
MegaBuck™
MICROCOUPLER™
MicroFET™
MicroPak™
MicroPak2™
MillerDrive™
MotionMax™
mWSaver®
OptoHiT™
OPTOLOGIC®
OPTOPLANAR®
®
PowerTrench®
PowerXS™
Programmable Active Droop™
QFET®
QS™
Quiet Series™
RapidConfigure™
™
Saving our world, 1mW/W/kW at a time™
SignalWise™
SmartMax™
SMART START™
Solutions for Your Success™
SPM®
STEALTH™
SuperFET®
SuperSOT™-3
SuperSOT™-6
SuperSOT™-8
SupreMOS®
SyncFET™
Sync-Lock™
SYSTEM GENERAL®*
TinyBoost®
TinyBuck®
TinyCalc™
TinyLogic®
TINYOPTO™
TinyPower™
TinyPWM™
TinyWire™
TranSiC™
TriFault Detect™
TRUECURRENT®*
μSerDes™
μSerDes™
UHC®
Ultra FRFET™
UniFET™
VCX™
VisualMax™
VoltagePlus™
XS™

*Trademarks of System General Corporation, used under license by Fairchild Semiconductor.

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. 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.

As used here in:

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 manufactures 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 application, 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 handing 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 and 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
Definition of Terms

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. I66

MR850 THRU MR856

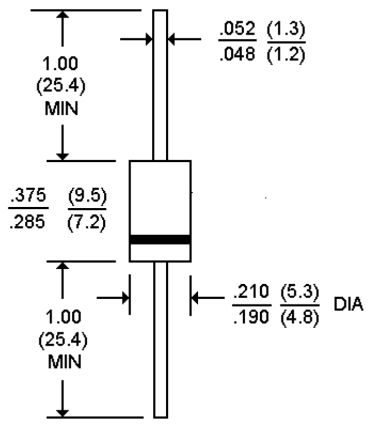
SOFT RECOVERU, FAST SWITCHING PLASTIC RECTIFIER

VOLTAGE - 50 to 600 Volts   CURRENT - 3.0 Amperes

FEATURES

- High surge current capability
- Plastic package has Underwriters Laboratory Flammability Classification 94V-O
- Void-free molded plastic package
- 3.0 ampere operation at T_A=50 ¢J with no thermal runaway
- Exceeds environmental standards of MIL-S-19500/228
- Fast switching for high efficiency

DO-201AD



Dimensions in inches and (millimeters)

MECHANICAL DATA

Case: JEDEC DO-201AD molded plastic

Terminals: Plated Axial leads, solderable per MIL-STD-750, Method 2026

Polarity: Color Band denotes end

Mounting Position: Any

Weight: 0.04 ounce, 1.1 gram

MAXIMUM RATINGS AND ELECTRICAL CHARACTERISTICS

Ratings at 25 ¢J ambient temperature unless otherwise specified.

Resistive or inductive load.

	SYMBOLS	MR850	MR851	MR852	MR854	MR856	UNITS
Maximum Recurrent Peak Reverse Voltage	V_{RRM}	50	100	200	400	600	Volts
Maximum RMS Voltage	V_{RMS}	35	70	140	280	480	Volts
Maximum DC Blocking Voltage	V_{DC}	50	100	200	400	600	Volts
Maximum Average Forward Rectified Current .375"(9.5mm) Lead Length at T_A=50 ¢J	$I_{(AV)}$			3.0			Amps
Peak Forward Surge Current 10ms single half sine-wave superimposed on rated load at T_A=25 ¢J	I_{FSM}			100.0			Amps
Maximum Repetitive Peak Forward Surge(Note1)	I_{FRM}			10.0			Amps
Maximum Instantaneous Forward Voltage at 3.0A	V_F			1.25			Volts
Maximum DC Reverse Current T_A=25 ¢J at Rated DC Blocking Voltage T_A=100 ¢J	I_R			10.0 500.0			£g A £g A
Maximum Reverse Recovery Time(Note 3) T_J=25 ¢J	T_{RR}			150			ns
Typical Junction capacitance (Note 2)	C_J			60			$_P$F
Typical Thermal Resistance (Note 4)	R £KJA			15.0			¢J/W
Operating Junction Temperature Range	T_J			-50 to +125			¢J
Storage Temperature Range	T_{STG}			-50 to +150			¢J

NOTES:

1. Repetitive Peak Forward Surge Current at f<15KHz
2. Measured at 1 MHz and applied reverse voltage of 4.0 Volts
3. Reverse Recovery Test Conditions: I_F=0.5A, I_R=1.0A, I_{rr}=0.25A
4. Thermal Resistance From Junction to Ambient at 0.375"(9.5mm) lead length with both leads to heat sink

RATING AND CHARACTERISTIC CURVES

MR850 THRU MR856

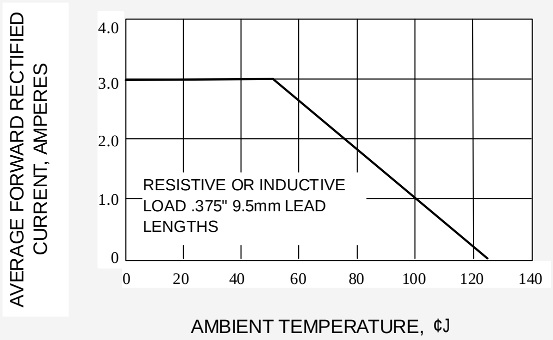


Fig. 1-FORWARD CURRENT DERATING CURVE

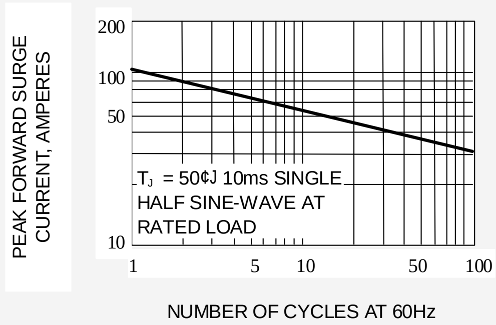


Fig. 2-MAXIMUM NON-REPETITIVE PEAK FORWARD SURGE CURRENT

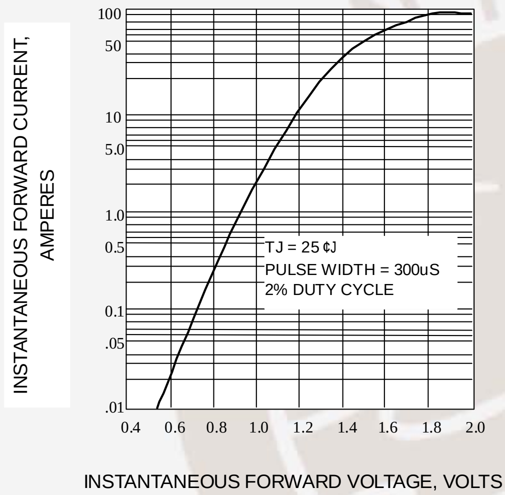


Fig. 3-TYPICAL INSTANTANEOUS FORWARD CHARACTERISTICS

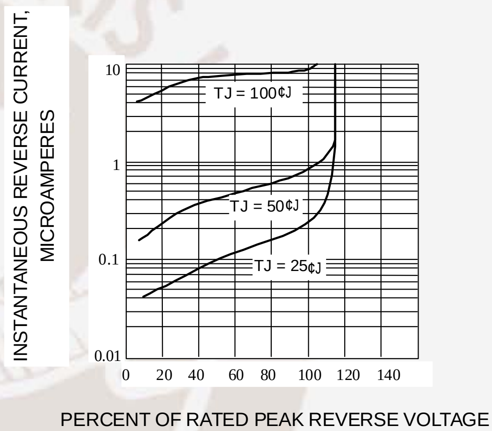


Fig. 4-TYPICAL REVERSE CHARACTERISTICS

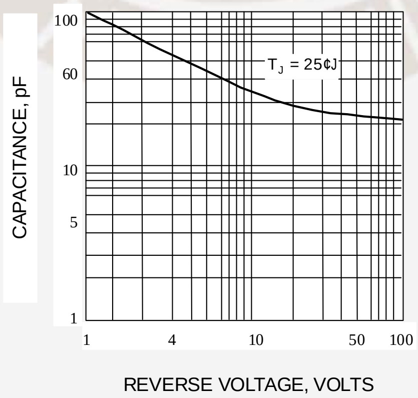
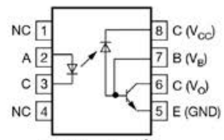


Fig. 5-TYPICAL JUNCTION CAPACITANCE

High Speed Optocoupler, 1 MBd, Photodiode with Transistor Output



DESCRIPTION

The 6N135 and 6N136 are optocouplers with a GaAIAs infrared emitting diode, optically coupled with an integrated photo detector which consists of a photo diode and a high-speed transistor in a DIP-8 plastic package.

Signals can be transmitted between two electrically separated circuits up to frequencies of 2 MHz. The potential difference between the circuits to be coupled should not exceed the maximum permissible reference voltages.

FEATURES

- Isolation test voltages: 5300 V_{RMS}
- TTL compatible
- High bit rates: 1 Mbit/s
- High common-mode interference immunity
- Bandwidth 2 MHz
- Open-collector output
- External base wiring possible
- Material categorization: for definitions of compliance please see www.vishay.com/doc?99912

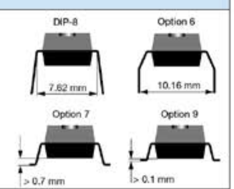
AGENCY APPROVALS

- UL1577 file no. E52744, double protection
- DIN EN 60747-5-5 (VDE0884-5) available with option 1
- cUL components acceptance service no. 5A
- CQC GB8898-2011, GB4943.1-2011

ORDERING INFORMATION

6 N 1 3 # - X 0 # # T

PART NUMBER — PACKAGE OPTION — TAPE AND REEL



AGENCY CERTIFIED / PACKAGE	CTR (%)	
UL, CSA	**≥ 7**	**≥ 19**
DIP-8	6N135	6N136
DIP-8, 400 mil, option 6	-	6N136-X006
SMD-8, option 7	6N135-X007T (1)	6N136-X007T (1)
SMD-8, option 9	-	6N136-X009T (1)
VDE, UL, CSA	**≥ 7**	**≥ 19**
DIP-8	-	6N136-X001
SMD-8, option 7	6N135-X017T (1)	6N136-X017T
SMD-8, option 9	-	6N136-X019T

Note

(1) Also available in tubes; do not add T to end

ABSOLUTE MAXIMUM RATINGS (T_{amb} = 25 °C, unless otherwise specified)				
PARAMETER	**CONDITION**	**SYMBOL**	**VALUE**	**UNIT**
INPUT				
Reverse voltage		V_R	5	V
Forward current		I_F	25	mA
Peak forward current	t = 1 ms, duty cycle 50 %	I_{FSM}	50	mA
Maximum surge forward current	t ≤ 1 μs, 300 pulses/s		1	A
Thermal resistance		R_{th}	700	K/W
Power dissipation	T_{amb} = 70 °C	P_{diss}	45	mW
OUTPUT				
Supply voltage		V_S	-0.5 to 15	V
Output voltage		V_O	-0.5 to 15	V
Emitter base voltage		V_{EBO}	5	V
Output current		I_O	8	mA
Maximum output current			16	mA
Base current		I_B	5	mA
Thermal resistance			300	K/W
Power dissipation	T_{amb} = 70 °C	P_{diss}	100	mW
COUPLER				
Storage temperature range		T_{stg}	-55 to +150	°C
Ambient temperature range		T_{amb}	-55 to +100	°C
Soldering temperature	max. ≤ 10 s, dip soldering ≥ 0.5 mm from case bottom	T_{sld}	260	°C

Note

- Stresses in excess of the absolute maximum ratings can cause permanent damage to the device. Functional operation of the device is not implied at these or any other conditions in excess of those given in the operational sections of this document. Exposure to absolute maximum ratings for extended periods of the time can adversely affect reliability.

ELECTRICAL CHARACTERISTICS (T_{amb} = 25 °C, unless otherwise specified)							
PARAMETER	**TEST CONDITION**	**PART**	**SYMBOL**	**MIN.**	**TYP.**	**MAX.**	**UNIT**
INPUT							
Forward voltage	I_F = 16 mA		V_F	-	1.33	1.9	V
Breakdown voltage	I_R = 10 μA		V_{BR}	5	-	-	V
Reverse current	V_R = 5 V		I_R	-	0.5	10	μA
Capacitance	V_R = 0 V, f = 1 MHz		C_O	-	30	-	pF
Temperature coefficient, forward voltage	I_F = 16 mA		$\Delta V_F/\Delta T_A$	-	-1.7	-	mV/°C
OUTPUT							
Logic low supply current	I_F = 16 mA, V_O = open, V_{CC} = 15 V		I_{CCL}	-	150	-	μA
Logic high supply current	I_F = 0 mA, V_O = open, V_{CC} = 15 V		I_{CCH}	-	0.01	1	μA
Output voltage, output low	I_F = 16 mA, I_O = 1.1 mA, V_{CC} = 4.5 V	6N135	V_{OL}	-	0.1	0.4	V
	I_F = 16 mA, I_O = 3.0 mA, V_{CC} = 4.5 V	6N136	V_{OL}	-	0.1	0.4	V
Output current, output high	I_F = 0 mA, V_O = V_{CC} = 5.5 V		I_{OH}	-	3	500	nA
	I_F = 0 mA, V_O = V_{CC} = 15 V		I_{OH}	-	0.01	1	μA
COUPLER							
Capacitance (input to output)	f = 1 MHz		C_{IO}	-	0.6	-	pF

Note

- Minimum and maximum values are testing requirements. Typical values are characteristics of the device and are the result of engineering evaluation. Typical values are for information only and are not part of the testing requirements.

CURRENT TRANSFER RATIO (T_{amb} = 25 °C, unless otherwise specified)							
PARAMETER	**TEST CONDITION**	**PART**	**SYMBOL**	**MIN.**	**TYP.**	**MAX.**	**UNIT**
Current transfer ratio	I_F = 16 mA, V_O = 0.4 V, V_{CC} = 4.5 V	6N135	CTR	7	16	-	%
		6N136	CTR	19	35	-	%
	I_F = 16 mA, V_O = 0.5 V, V_{CC} = 4.5 V	6N135	CTR	5	-	-	%
		6N136	CTR	15	-	-	%

SWITCHING CHARACTERISTICS (T_{amb} = 25 °C, unless otherwise specified)							
PARAMETER	**TEST CONDITION**	**PART**	**SYMBOL**	**MIN.**	**TYP.**	**MAX.**	**UNIT**
High to low	I_F = 16 mA, V_{CC} = 5 V, R_L = 4.1 kΩ	6N135	t_{PHL}	-	0.3	1.5	μs
	I_F = 16 mA, V_{CC} = 5 V, R_L = 1.9 kΩ	6N136	t_{PHL}	-	0.2	0.8	μs
Low to high	I_F = 16 mA, V_{CC} = 5 V, R_L = 4.1 kΩ	6N135	t_{PLH}	-	0.3	1.5	μs
	I_F = 16 mA, V_{CC} = 5 V, R_L = 1.9 kΩ	6N136	t_{PLH}	-	0.2	0.8	μs

COMMON MODE TRANSIENT IMMUNITY (T_{amb} = 25 °C, unless otherwise specified)							
PARAMETER	**TEST CONDITION**	**PART**	**SYMBOL**	**MIN.**	**TYP.**	**MAX.**	**UNIT**
High	I_F = 0 mA, V_{CM} = 10 V_{P-P}, V_{CC} = 5 V, R_L = 4.1 kΩ	6N135	$\|CM_H\|$	-	1000	-	V/μs
	I_F = 0 mA, V_{CM} = 10 V_{P-P}, V_{CC} = 5 V, R_L = 1.9 kΩ	6N136	$\|CM_H\|$	-	1000	-	V/μs
Low	I_F = 16 mA, V_{CM} = 10 V_{P-P}, V_{CC} = 5 V, R_L = 4.1 kΩ	6N135	$\|CM_L\|$	-	1000	-	V/μs
	I_F = 16 mA, V_{CM} = 10 V_{P-P}, V_{CC} = 5 V, R_L = 1.9 kΩ	6N136	$\|CM_L\|$	-	1000	-	V/μs

SAFETY AND INSULATION RATINGS				
PARAMETER	**TEST CONDITION**	**SYMBOL**	**VALUE**	**UNIT**
Climatic classification	According to IEC 68 part 1		55 / 100 / 21	
Pollution degree	According to DIN VDE 0109		2	
Comparative tracking index	Insulation group IIIa	CTI	175	
Maximum rated withstanding isolation voltage	According to UL1577, t = 1 min	V_{ISO}	5300	V_{RMS}
Maximum transient isolation voltage	According to DIN EN 60747-5-5	V_{IOTM}	8000	V_{peak}
Maximum repetitive peak isolation voltage	According to DIN EN 60747-5-5	V_{IORM}	890	V_{peak}
Isolation resistance	T_{amb} = 25 °C, V_{IO} = 500 V	R_{IO}	$\geq 10^{12}$	Ω
	T_{amb} = 100 °C, V_{IO} = 500 V	R_{IO}	$\geq 10^{11}$	Ω
Output safety power		P_{SO}	500	mW
Input safety current		I_{SI}	300	mA
Input safety temperature		T_S	175	°C
Creepage distance	DIP-8		≥ 7	mm
Clearance distance	DIP-8		≥ 7	mm
Creepage distance	DIP-8, 400 mil, option 6		≥ 8	mm
Clearance distance	DIP-8, 400 mil, option 6		≥ 8	mm
Creepage distance	SMD-8, option 7		≥ 8	mm
Clearance distance	SMD-8, option 7		≥ 8	mm
Creepage distance	SMD-8, option 9		≥ 8	mm
Clearance distance	SMD-8, option 9		≥ 8	mm
Insulation thickness		DTI	≥ 0.4	mm

Note

- As per IEC 60747-5-5, § 7.4.3.8.2, this optocoupler is suitable for "safe electrical insulation" only within the safety ratings. Compliance with the safety ratings shall be ensured by means of protective circuits.

TYPICAL CHARACTERISTICS (T_{amb} = 25 °C, unless otherwise specified)

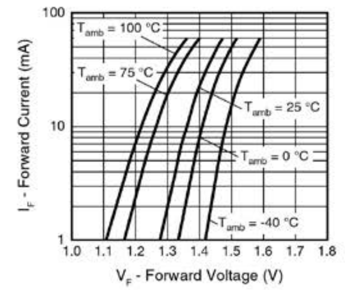


Fig. 1 - LED Forward Current vs. Forward Voltage

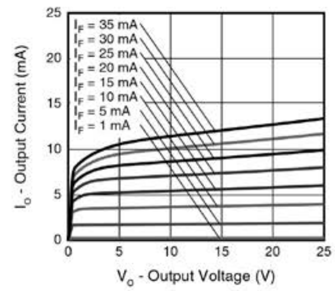


Fig. 4 - Output Current vs. Output Voltage

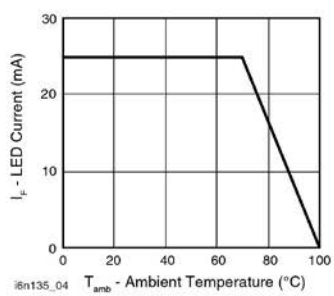


Fig. 2 - Permissible Forward LED Current vs. Temperature

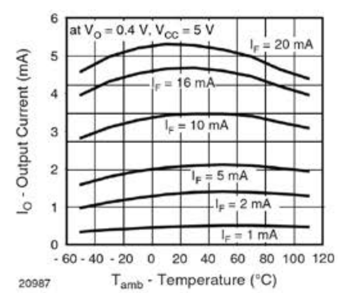


Fig. 5 - Output Current vs. Temperature

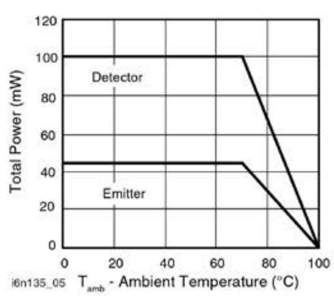


Fig. 3 - Permissible Power Dissipation vs. Temperature

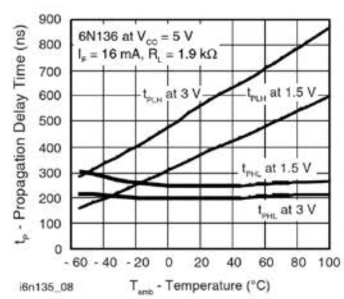


Fig. 6 - Propagation Delay vs. Ambient Temperature

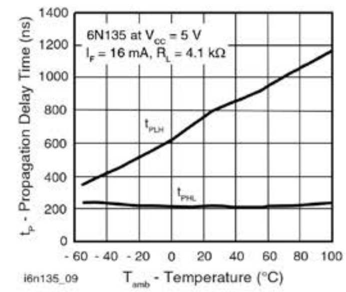


Fig. 7 - Propagation Delay vs. Ambient Temperature

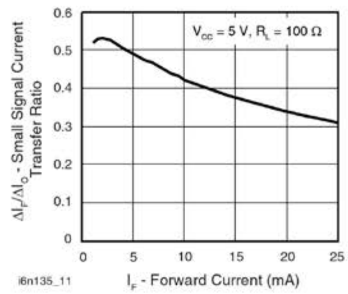


Fig. 9 - Small Signal Current Transfer Ratio vs. Quiescent Input Current

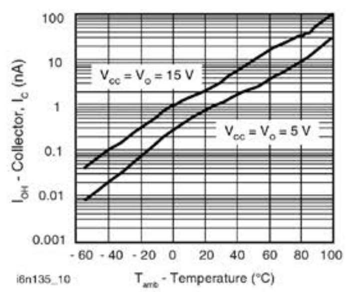


Fig. 8 - Logic High Output Current vs. Temperature

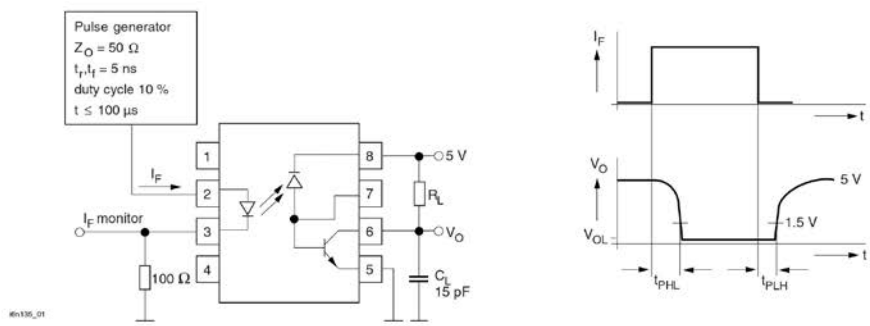


Fig. 10 - Switching Times

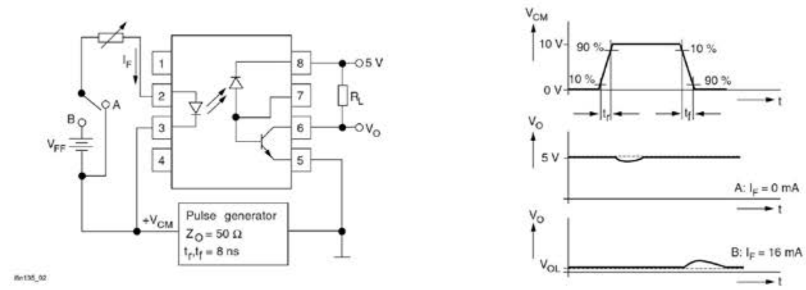
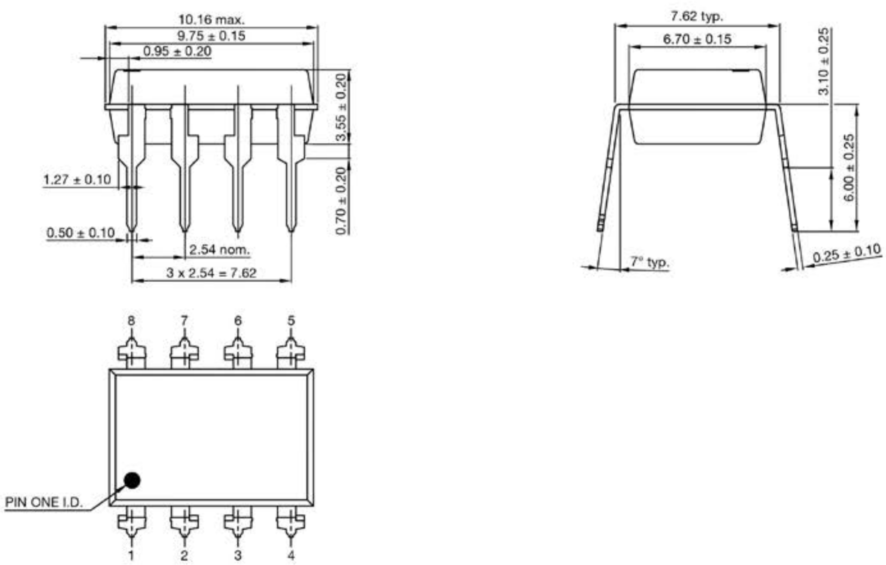


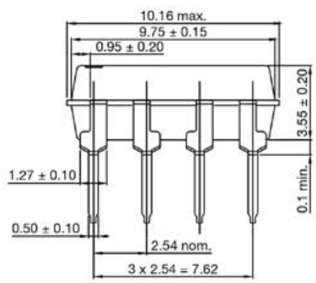
Fig. 11 - Common-Mode Interference Immunity

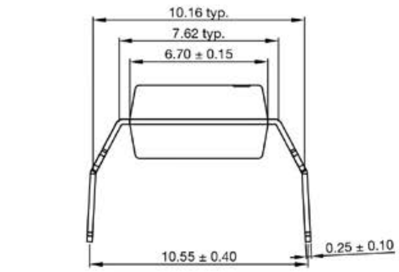
PACKAGE DIMENSIONS (in millimeters)

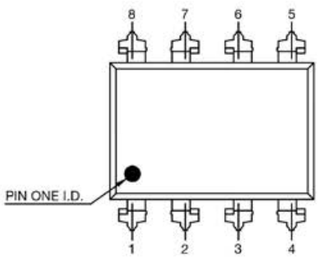
DIP-8, Standard



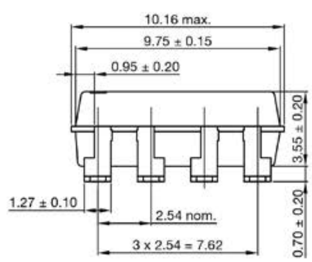
DIP-8, Option 6

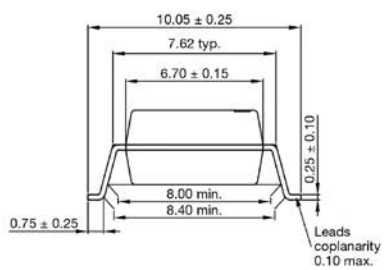


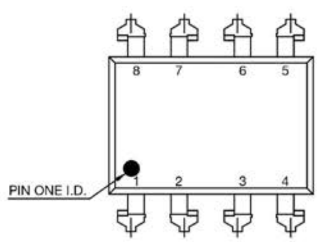


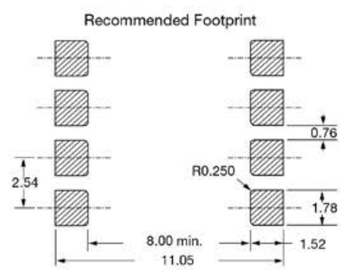


DIP-8, Option 7

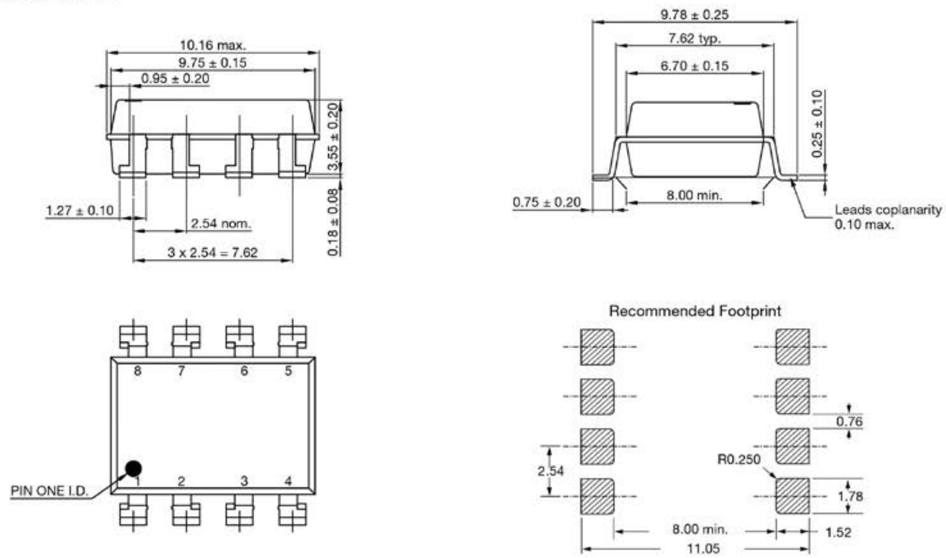








DIP-8, Option 9



PACKAGE MARKING

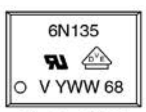


Fig. 12 - 6N135

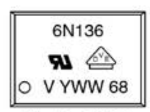


Fig. 13 - 6N136

Notes

- The VDE logo is only marked on option 1 parts.
- Tape and reel suffix (T) is not part of the package marking.

SOLDER PROFILES

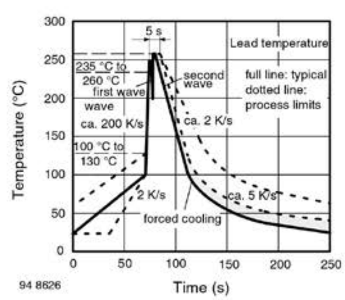


Fig. 14 - Wave Soldering Double Wave Profile According to J-STD-020 for DIP-8 Devices

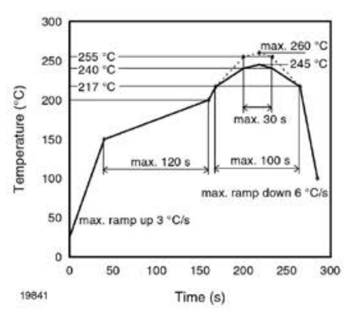


Fig. 15 - Lead (Pb)-free Reflow Solder Profile According to J-STD-020 for SMD-8 Devices

HANDLING AND STORAGE CONDITIONS

ESD level: HBM class 2

Floor life: unlimited

Conditions: T_{amb} < 30 °C, RH < 85 %

Moisture sensitivity level 1, according to J-STD-020

Disclaimer

ALL PRODUCT, PRODUCT SPECIFICATIONS AND DATA ARE SUBJECT TO CHANGE WITHOUT NOTICE TO IMPROVE RELIABILITY, FUNCTION OR DESIGN OR OTHERWISE.

Vishay Intertechnology, Inc., its affiliates, agents, and employees, and all persons acting on its or their behalf (collectively, "Vishay"), disclaim any and all liability for any errors, inaccuracies or incompleteness contained in any datasheet or in any other disclosure relating to any product.

Vishay makes no warranty, representation or guarantee regarding the suitability of the products for any particular purpose or the continuing production of any product. To the maximum extent permitted by applicable law, Vishay disclaims (i) any and all liability arising out of the application or use of any product, (ii) any and all liability, including without limitation special, consequential or incidental damages, and (iii) any and all implied warranties, including warranties of fitness for particular purpose, non-infringement and merchantability.

Statements regarding the suitability of products for certain types of applications are based on Vishay's knowledge of typical requirements that are often placed on Vishay products in generic applications. Such statements are not binding statements about the suitability of products for a particular application. It is the customer's responsibility to validate that a particular product with the properties described in the product specification is suitable for use in a particular application. Parameters provided in datasheets and/or specifications may vary in different applications and performance may vary over time. All operating parameters, including typical parameters, must be validated for each customer application by the customer's technical experts. Product specifications do not expand or otherwise modify Vishay's terms and conditions of purchase, including but not limited to the warranty expressed therein.

Except as expressly indicated in writing, Vishay products are not designed for use in medical, life-saving, or life-sustaining applications or for any other application in which the failure of the Vishay product could result in personal injury or death. Customers using or selling Vishay products not expressly indicated for use in such applications do so at their own risk. Please contact authorized Vishay personnel to obtain written terms and conditions regarding products designed for such applications.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document or by any conduct of Vishay. Product names and markings noted herein may be trademarks of their respective owners.

Material Category Policy

Vishay Intertechnology, Inc. hereby certifies that all its products that are identified as RoHS-Compliant fulfill the definitions and restrictions defined under Directive 2011/65/EU of The European Parliament and of the Council of June 8, 2011 on the restriction of the use of certain hazardous substances in electrical and electronic equipment (EEE) - recast, unless otherwise specified as non-compliant.

Please note that some Vishay documentation may still make reference to RoHS Directive 2002/95/EC. We confirm that all the products identified as being compliant to Directive 2002/95/EC conform to Directive 2011/65/EU.

Vishay Intertechnology, Inc. hereby certifies that all its products that are identified as Halogen-Free follow Halogen-Free requirements as per JEDEC JS709A standards. Please note that some Vishay documentation may still make reference to the IEC 61249-2-21 definition. We confirm that all the products identified as being compliant to IEC 61249-2-21 conform to JEDEC JS709A standards.

Features

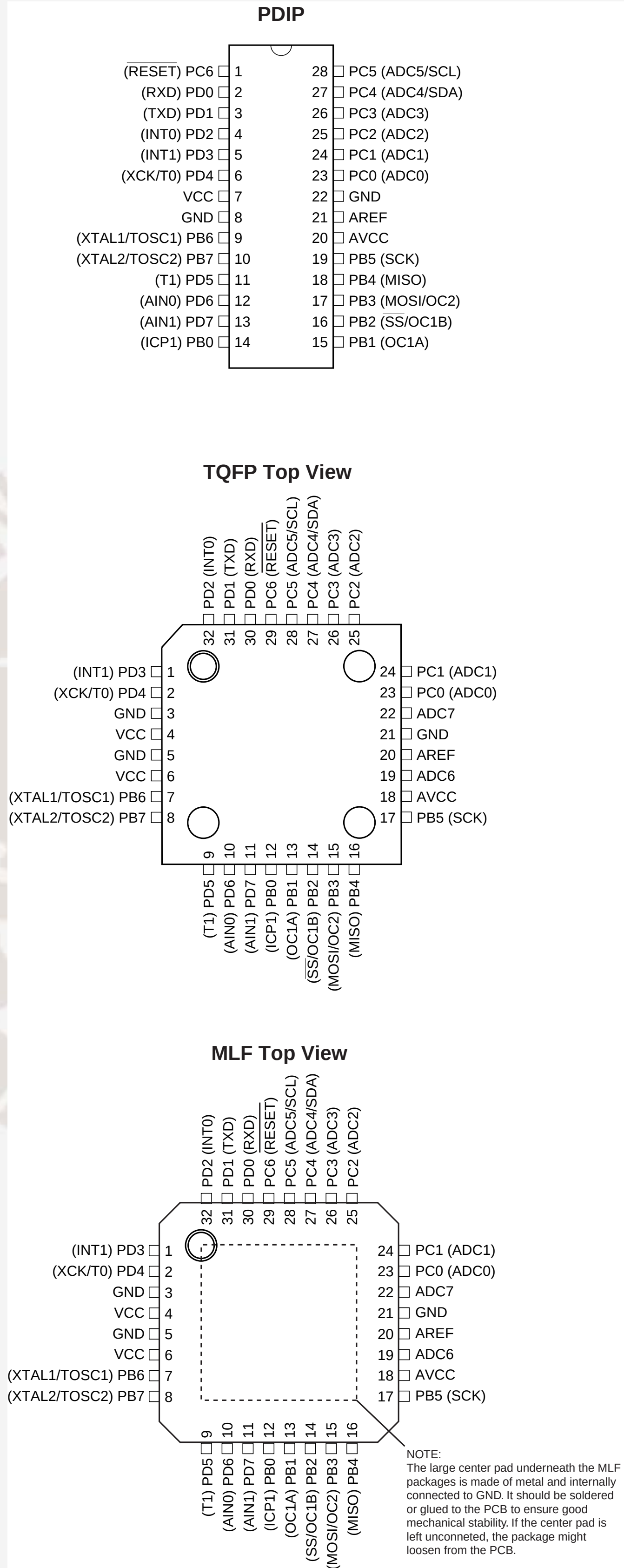
- High-performance, Low-power Atmel®AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 130 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 × 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16MIPS Throughput at 16MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 8Kbytes of In-System Self-programmable Flash program memory
  - 512Bytes EEPROM
  - 1Kbyte 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
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler, one Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Three PWM Channels
  - 8-channel ADC in TQFP and QFN/MLF package
    Eight Channels 10-bit Accuracy
  - 6-channel ADC in PDIP package
    Six Channels 10-bit Accuracy
  - 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
  - Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-lead PDIP, 32-lead TQFP, and 32-pad QFN/MLF
- Operating Voltages
  - 2.7V - 5.5V (ATmega8L)
  - 4.5V - 5.5V (ATmega8)
- Speed Grades
  - 0 - 8MHz (ATmega8L)
  - 0 - 16MHz (ATmega8)
- Power Consumption at 4Mhz, 3V, 25°C
  - Active: 3.6mA
  - Idle Mode: 1.0mA
  - Power-down Mode: 0.5µA

8-bit Atmel with 8KBytes In-System Programmable Flash

ATmega8
ATmega8L

Rev.2486AA–AVR–02/2013

Pin Configurations

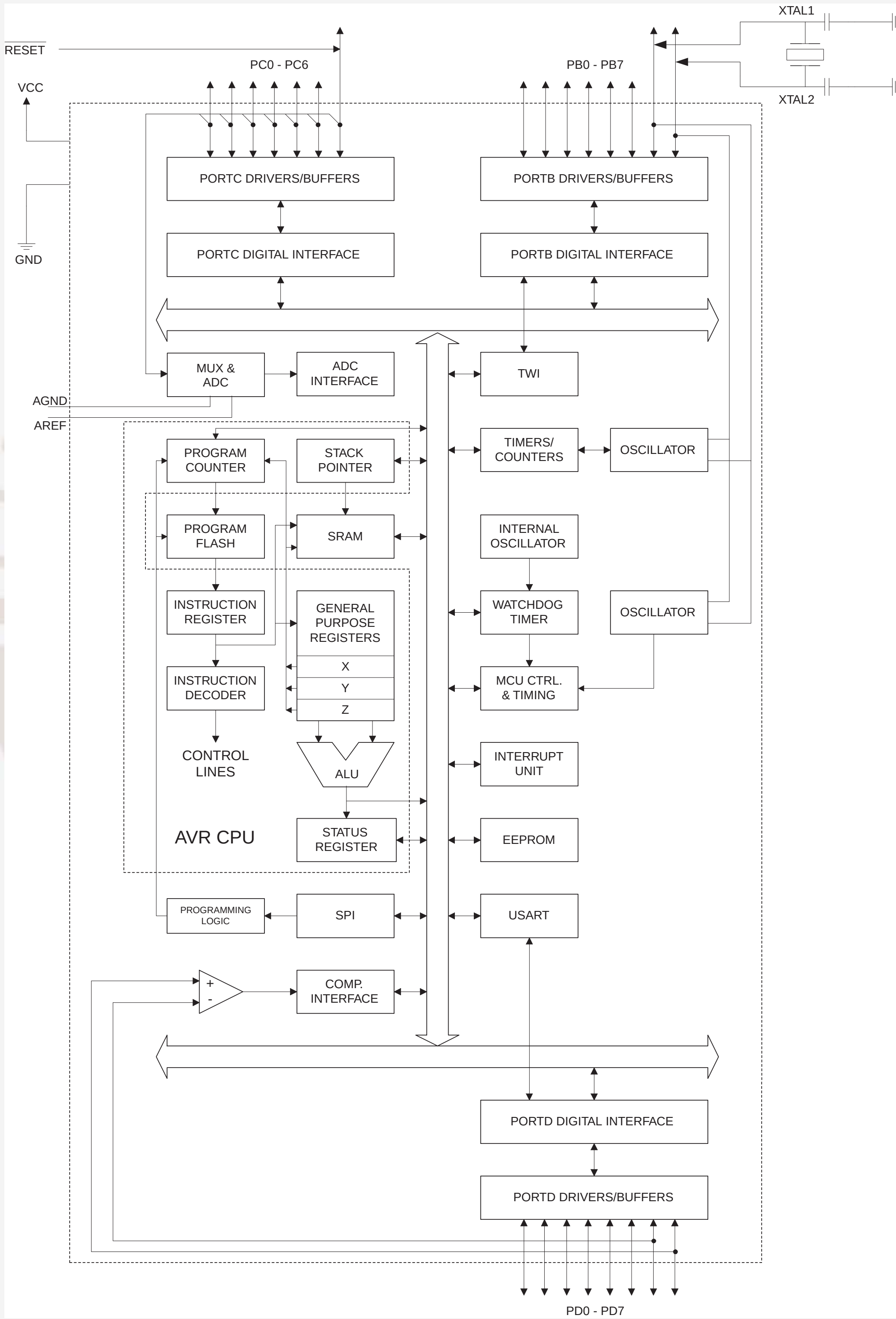


Overview

The Atmel®AVR® ATmega8 is a low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega8 achieves throughputs approaching 1MIPS per MHz, allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 1. Block Diagram



The Atmel®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 ATmega8 provides the following features: 8 Kbytes of In-System Programmable Flash with Read-While-Write capabilities, 512 bytes of EEPROM, 1 Kbyte of SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, a 6-channel ADC (eight channels in TQFP and QFN/MLF packages) with 10-bit accuracy, a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and five software selectable power saving modes. The Idle mode stops the CPU while allowing the 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 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.

The device is manufactured using Atmel's high density non-volatile memory technology. The Flash Program memory can be reprogrammed In-System through an SPI serial interface, by a conventional non-volatile 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 ATmega8 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The ATmega8 is supported with a full suite of program and system development tools, including C compilers, macro assemblers, program simulators, and evaluation kits.

Disclaimer

Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Minimum and Maximum values will be available after the device is characterized.

Pin Descriptions

VCC Digital supply voltage.

GND Ground.

Port B (PB7..PB0) XTAL1/XTAL2/TOSC1/ TOSC2

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.

Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier.

If the Internal Calibrated RC Oscillator is used as chip clock source, PB7..6 is used as TOSC2..1 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

The various special features of Port B are elaborated in “Alternate Functions of Port B” on page 58 and “System Clock and Clock Options” on page 25.

Port C (PC5..PC0)

Port C is an 7-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.

PC6/$\overline{\text{RESET}}$

If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C.

If the RSTDISBL Fuse is unprogrammed, PC6 is used as a 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.

The various special features of Port C are elaborated 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 ATmega8 as listed on page 63.

$\overline{\text{RESET}}$

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.

AV_{CC}

AV_{CC} is the supply voltage pin for the A/D Converter, Port C (3..0), and ADC (7..6). It should be externally connected 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. Note that Port C (5..4) use digital supply voltage, V_{CC}.

AREF

AREF is the analog reference pin for the A/D Converter.

ADC7..6 (TQFP and QFN/MLF Package Only)

In the TQFP and QFN/MLF package, ADC7..6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

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 datasheet 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.

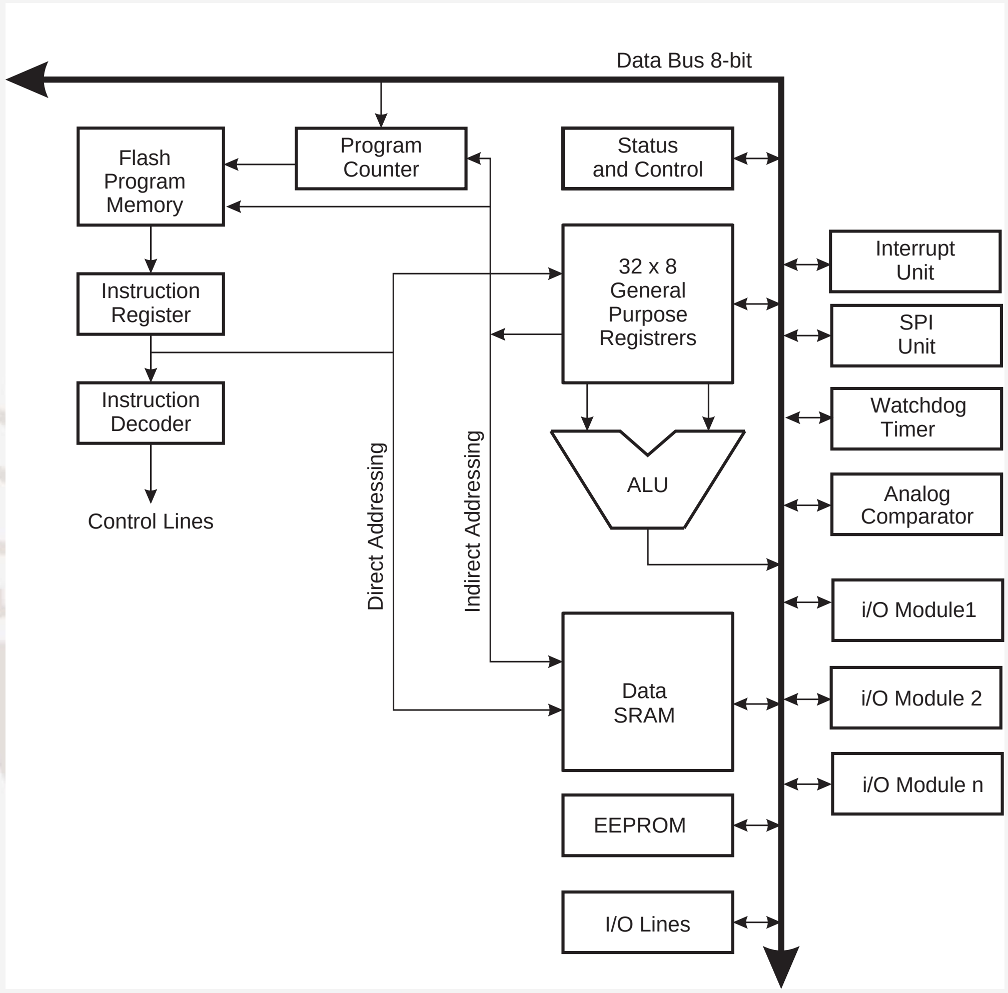
Atmel AVR CPU Core

Introduction

This section discusses the Atmel®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 2. 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.

The 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, 0x20 - 0x5F.

Arithmetic Logic Unit – ALU

The high-performance Atmel®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. For a detailed description, see “Instruction Set Summary” on page 311.

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	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
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 – 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.

- **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, $S = N \oplus V$**

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 3 shows the structure of the 32 general purpose working registers in the CPU.

Figure 3. AVR CPU General Purpose Working Registers

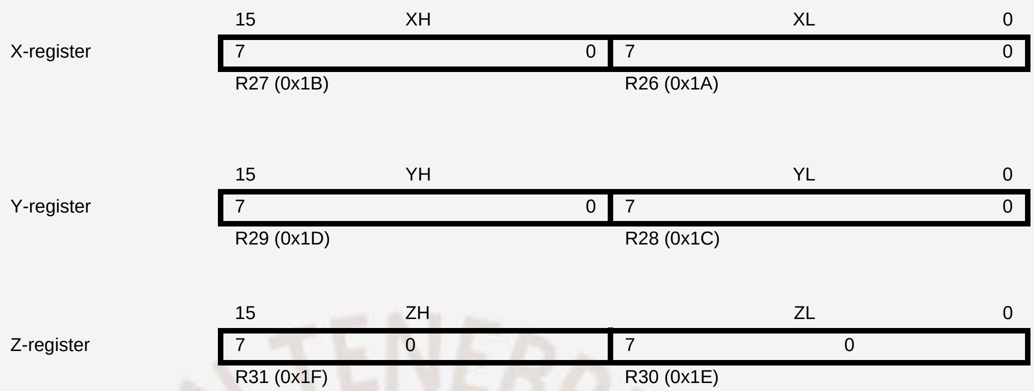
Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 3, 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-pointer, Y-pointer, and Z-pointer Registers can be set to index any register in the file.

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 4.

Figure 4. 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.

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 0x60. 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 address 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 Atmel®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 5 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 1MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Figure 5. The Parallel Instruction Fetches and Instruction Executions

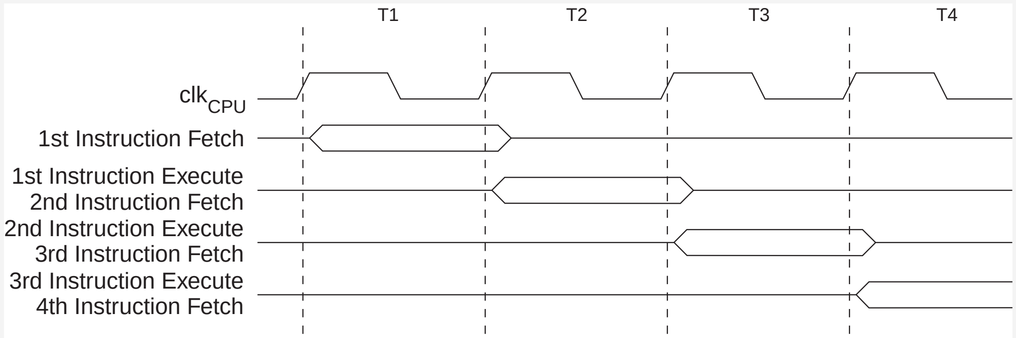
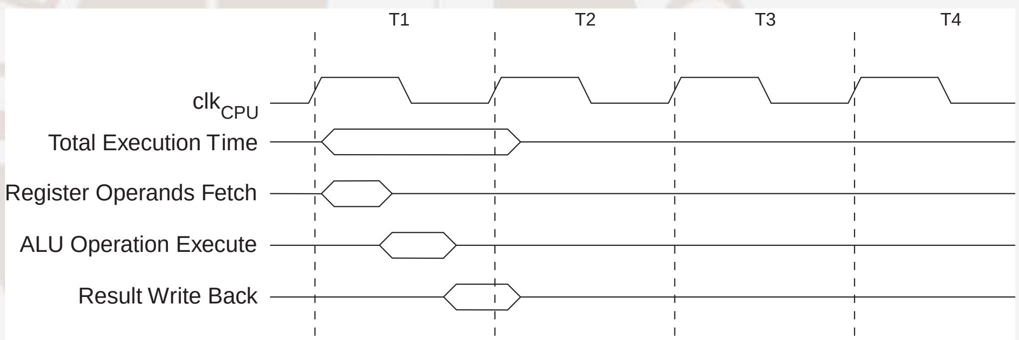


Figure 6 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 6. Single Cycle ALU Operation



Reset and Interrupt Handling

The Atmel®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 215 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 46. 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 INT0 – the External Interrupt Request 0. The Interrupt Vectors can be moved to the start of the boot Flash section by setting the Interrupt Vector Select (IVSEL) bit in the General Interrupt Control Register (GICR). Refer to "Interrupts" on page 46 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 202.

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 */
_CLI();
EECR |= (1<<EEMWE); /* start EEPROM write */
EECR |= (1<<EEWE);
SREG = cSREG; /* restore SREG value (I-bit) */
```

When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in the following 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 Atmel®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 4-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 (2 bytes) is popped back from the Stack, the Stack Pointer is incremented by 2, and the I-bit in SREG is set.

AVR ATmega8 Memories

This section describes the different memories in the Atmel®AVR® ATmega8. The AVR architecture has two main memory spaces, the Data memory and the Program Memory space. In addition, the ATmega8 features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

In-System Reprogrammable Flash Program Memory

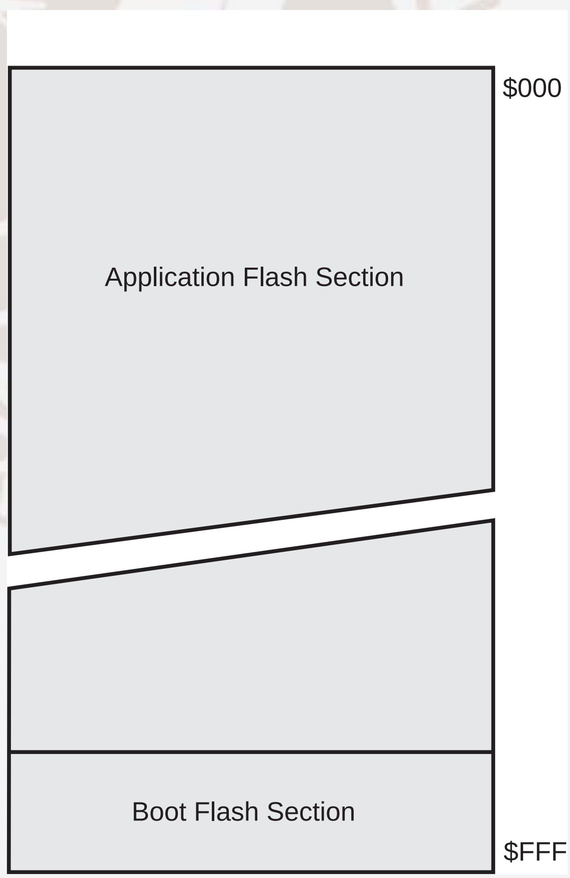
The ATmega8 contains 8Kbytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16-bits or 32-bits wide, the Flash is organized as 4K × 16 bits. For software security, the Flash Program memory space is divided into two sections, Boot Program section and Application Program section.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega8 Program Counter (PC) is 12 bits wide, thus addressing the 4K 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 202. "Memory Programming" on page 215 contains a detailed description on Flash Programming in SPI- or Parallel Programming mode.

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 7. Program Memory Map



SRAM Data Memory

Figure 8 shows how the Atmel®AVR® 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.

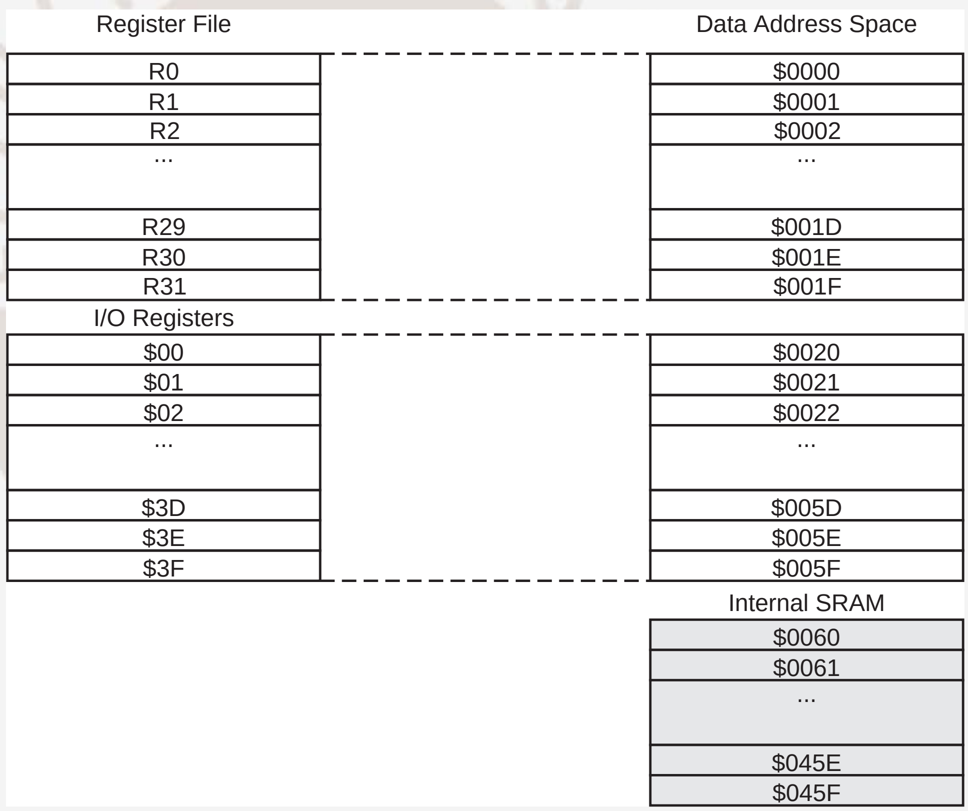
The direct addressing reaches the entire data space.

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 ATmega8 are all accessible through all these addressing modes. The Register File is described in "General Purpose Register File" on page 12.

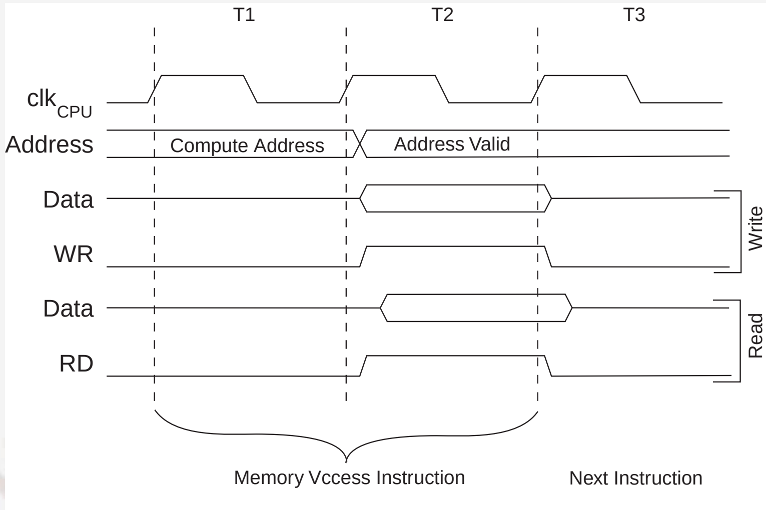
Figure 8. Data Memory Map



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 9.

Figure 9. On-chip Data SRAM Access Cycles



EEPROM Data Memory

The ATmega8 contains 512bytes 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 below, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

"Memory Programming" on page 215 contains a detailed description on EEPROM Programming in SPI- or Parallel Programming mode.

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 on page 21. 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 23. 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 EEPROM Control Register – EECR" on page 20 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.

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	R/W	R/W	R/W	R/W	R/W	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 ATmega8 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 512bytes 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

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	EEDR
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..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

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	EERIE	EEMWE	EEWE	EERE	EECR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	X	0	

- **Bits 7..4 – Res: Reserved Bits**

These bits are reserved bits in the Atmel®AVR® ATmega8 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 "Bit 1 – EEWE: EEPROM Write Enable" for an EEPROM write procedure.

- **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
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 202 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	8.5ms

Note: 1. Uses 1MHz clock, independent of CKSEL Fuse settings

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 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);
}
```

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. Second, 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_{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 ATmega8 is shown in "Register Summary" on page 309.

All Atmel®AVR® ATmega8 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 0x00 - 0x1F 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 Summary" on page 311 for more details. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 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 0x00 to 0x1F only.

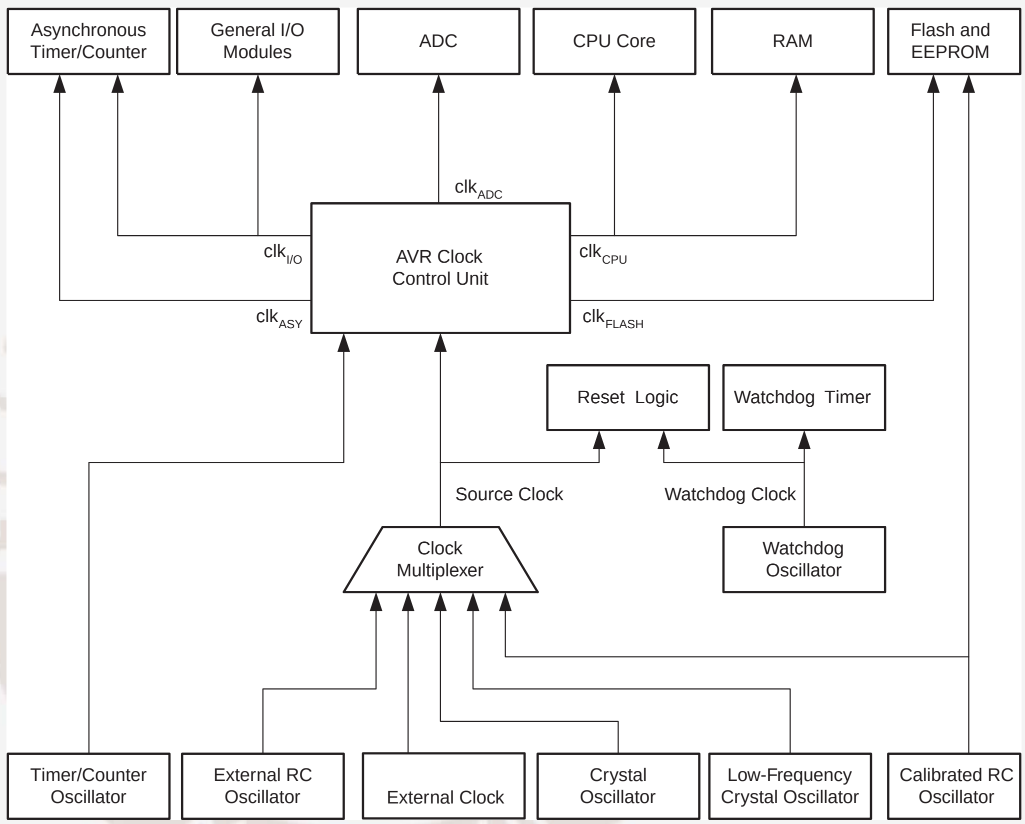
The I/O and Peripherals Control Registers are explained in later sections.

System Clock and Clock Options

Clock Systems and their Distribution

Figure 10 presents the principal clock systems in the Atmel®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 33. The clock systems are detailed Figure 10.

Figure 10. 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 32kHz clock crystal. The dedicated clock domain allows using this Timer/Counter as a real-time counter even when the device is in sleep mode. The Asynchronous Timer/Counter uses the same XTAL pins as the CPU main clock but requires a CPU main clock frequency of more than four times the Oscillator frequency. Thus, asynchronous operation is only available while the chip is clocked on the Internal Oscillator.

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	CKSEL3..0
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 “ATmega8 Typical Characteristics – TA = -40°C to 85°C”. The device is shipped with CKSEL = “0001” and SUT = “10” (1MHz Internal RC Oscillator, slowly rising power).

Table 3. Number of Watchdog Oscillator Cycles

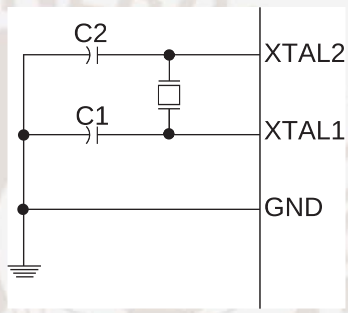
Typical Time-out (V_{CC} = 5.0V)	Typical Time-out (V_{CC} = 3.0V)	Number of Cycles
4.1ms	4.3ms	4K (4,096)
65ms	69ms	64K (65,536)

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 11. 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 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 cannot be used to drive other clock buffers.

For resonators, the maximum frequency is 8MHz with CKOPT unprogrammed and 16MHz 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 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 11. 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

CKOPT	CKSEL3..1	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 on page 28.

Table 5. Start-up Times for the Crystal Oscillator Clock Selection

CKSEL0	SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
0	00	258 CK[(1)]	4.1ms	Ceramic resonator, fast rising power
0	01	258 CK[(1)]	65ms	Ceramic resonator, slowly rising power
0	10	1K CK[(2)]	–	Ceramic resonator, BOD enabled
0	11	1K CK[(2)]	4.1ms	Ceramic resonator, fast rising power
1	00	1K CK[(2)]	65ms	Ceramic resonator, slowly rising power
1	01	16K CK	–	Crystal Oscillator, BOD enabled
1	10	16K CK	4.1ms	Crystal Oscillator, fast rising power
1	11	16K CK	65ms	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

2. 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 32.768kHz 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 11 on page 27. 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 36pF.

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

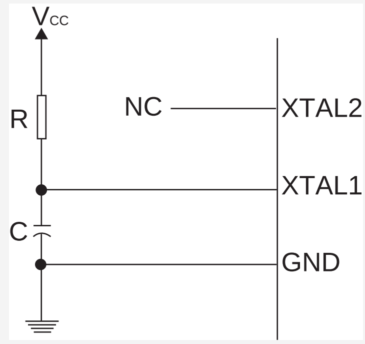
SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	1K CK[(1)]	4.1ms	Fast rising power or BOD enabled
01	1K CK[(1)]	65ms	Slowly rising power
10	32K CK	65ms	Stable frequency at start-up
11	Reserved		

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 12 on page 29 can be used. The frequency is roughly estimated by the equation f = 1/(3RC). C should be at least 22pF. By programming the CKOPT Fuse, the user can enable an internal 36pF capacitor between XTAL1 and GND, thereby removing the need for an external capacitor.

Figure 12. 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

CKSEL3..0	Frequency Range (MHz)
0101	0.1 - 0.9
0110	0.9 - 3.0
0111	3.0 - 8.0
1000	8.0 - 12.0

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

SUT1..0	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	4.1ms	Fast rising power
10	18 CK	65ms	Slowly rising power
11	6 CK(1)	4.1ms	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 1.0MHz, 2.0MHz, 4.0MHz, or 8.0MHz 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 unprogrammed when using this clock option. During reset, hardware loads the 1MHz calibration byte into the OSCCAL Register and thereby automatically calibrates the RC Oscillator. At 5V, 25°C and 1.0MHz Oscillator frequency selected, this calibration gives a frequency within ±3% of the nominal frequency. Using run-time calibration methods as described in application notes available at www.atmel.com/avr it is possible to achieve ±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 218.

Table 9. Internal Calibrated RC Oscillator Operating Modes

CKSEL3..0	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. PB6 (XTAL1/TOSC1) and PB7(XTAL2/TOSC2) can be used as either general I/O pins or Timer Oscillator pins..

Table 10. Start-up Times for the Internal Calibrated RC Oscillator Clock Selection

SUT1..0	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	4.1ms	Fast rising power
10[(1)]	6 CK	65ms	Slowly rising power
11	Reserved		

Note: 1. The device is shipped with this option selected

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	Device Specific Calibration Value								

- **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. During Reset, the 1MHz calibration value which is located in the signature row High byte (address 0x00) is automatically loaded into the OSCCAL Register. If the internal RC is used at other frequencies, the calibration values must be loaded manually. This can be done by first reading the signature row by a programmer, and then store the calibration values in the Flash or EEPROM. Then the value can be read by software and loaded into the OSCCAL Register. 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 0xFF 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 1.0MHz, 2.0MHz, 4.0MHz, or 8.0MHz. Tuning to other values is not guaranteed, as indicated in Table 11.

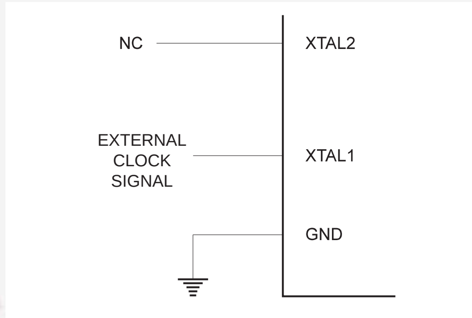
Table 11. Internal RC Oscillator Frequency Range

OSCCAL Value	Min Frequency in Percentage of Nominal Frequency (%)	Max Frequency in Percentage of Nominal Frequency (%)
0x00	50	100
0x7F	75	150
0xFF	100	200

External Clock

To drive the device from an external clock source, XTAL1 should be driven as shown in Figure 13. 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 36pF capacitor between XTAL1 and GND, and XTAL2 and GND.

Figure 13. 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

SUT1..0	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	4.1ms	Fast rising power
10	6 CK	65ms	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. By programming the CKOPT Fuse, the user can enable internal capacitors on XTAL1 and XTAL2, thereby removing the need for external capacitors. The Oscillator is optimized for use with a 32.768kHz 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 36pF

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 five 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, or 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.

Note that the Extended Standby mode present in many other AVR MCUs has been removed in the ATmega8, as the TOSC and XTAL inputs share the same physical pins.

Figure 10 on page 25 presents the different clock systems in the ATmega8, 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	
	SE	SM2	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	

- **Bit 7 – 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 programmer's purpose, it is recommended to set the Sleep Enable (SE) bit just before the execution of the SLEEP instruction.

- **Bits 6..4 – SM2..0: Sleep Mode Select Bits 2, 1, and 0**

These bits select between the five 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)]

Note: 1. Standby mode is only available with external crystals or resonators

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, or an external level interrupt on INT0 or INT1, 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, or an external level interrupt on INT0 or INT1, 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 66 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 26.

Power-save Mode

When the SM2..0 bits are written to 011, the SLEEP instruction makes the MCU enter Power-save 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/Counter 2 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 6 clock cycles.

Table 14. Active Clock Domains and Wake-up Sources in the Different Sleep Modes

	Active Clock Domains					Oscillators		Wake-up Sources					
Sleep Mode	**clk_{CPU}**	**clk_{FLASH}**	**clk_{IO}**	**clk_{ADC}**	**clk_{ASY}**	**Main Clock Source Enabled**	**Timer Osc. Enabled**	**INT1 INT0**	**TWI Address Match**	**Timer 2**	**SPM/ EEPROM Ready**	**ADC**	**Other I/O**
Idle			X	X	X	X	X[2]	X	X	X	X	X	X
ADC Noise Reduction				X	X	X	X[2]	X[3]	X	X	X	X	
Power Down								X[3]	X				
Power Save					X[2]		X[2]	X[3]	X	X[2]			
Standby[1]						X		X[3]	X				

Notes:
1. External Crystal or resonator selected as clock source
2. If AS2 bit in ASSR is set
3. Only 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 (ADC)

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 189 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 186 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 43 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 55 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.

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. 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 14 on page 38 shows the Reset Logic. Table 15 on page 38 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 26.

Reset Sources

The ATmega8 has four 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_{CC} is below the Brown-out Reset threshold (V_{BOT}) and the Brown-out Detector is enabled

Figure 14. Reset Logic

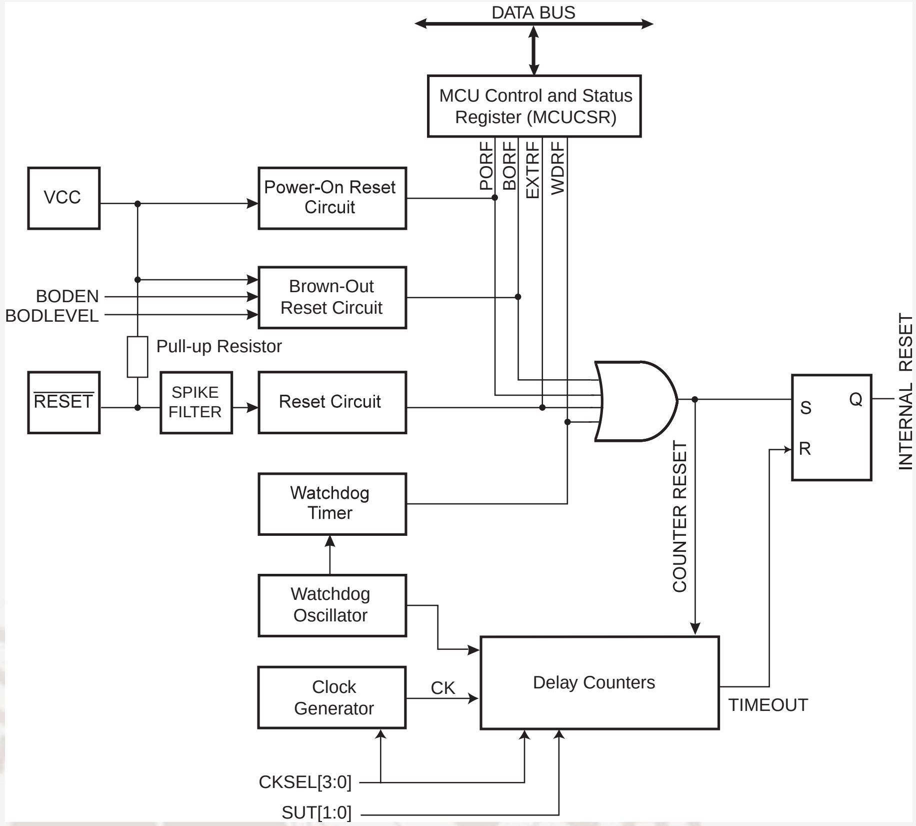


Table 15. Reset Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{POT}	Power-on Reset Threshold Voltage (rising)(1)			1.4	2.3	V
	Power-on Reset Threshold Voltage (falling)			1.3	2.3	
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold Voltage		0.2		0.9	V_{CC}
t_{RST}	Minimum pulse width on $\overline{\text{RESET}}$ Pin				1.5	µs
V_{BOT}	Brown-out Reset Threshold Voltage(2)	BODLEVEL = 1	2.4	2.6	2.9	V
		BODLEVEL = 0	3.7	4.0	4.5	
t_{BOD}	Minimum low voltage period for Brown-out Detection	BODLEVEL = 1		2		µs
		BODLEVEL = 0		2		
V_{HYST}	Brown-out Detector hysteresis			130		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 ATmega8L and BODLEVEL = 0 for ATmega8. BODLEVEL = 1 is not applicable for ATmega8

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 on page 38. 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 15. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}

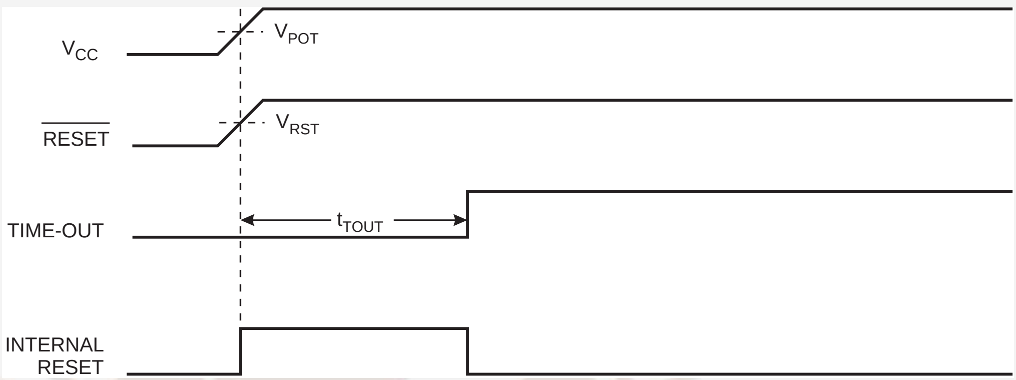
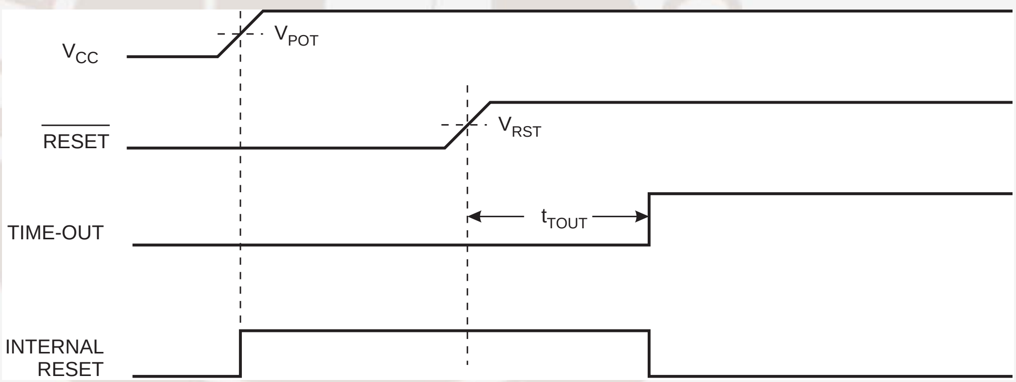


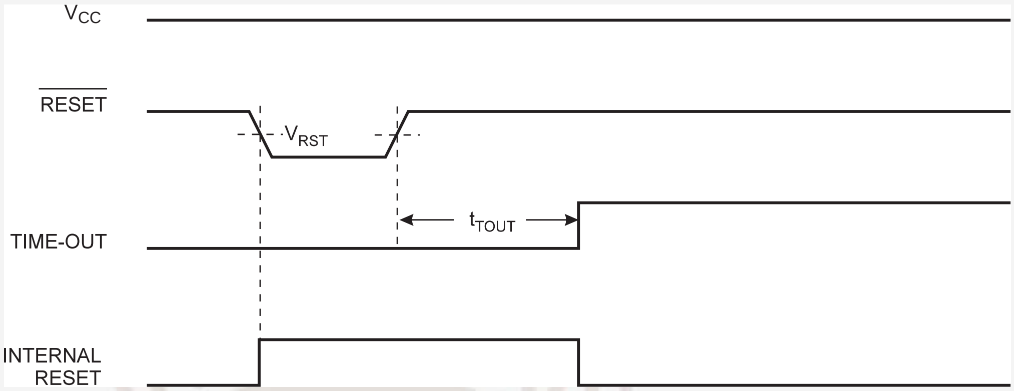
Figure 16. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally



External Reset

An External Reset is generated by a low level on the $\overline{RESET}$ pin. Reset pulses longer than the minimum pulse width (see Table 15 on page 38) 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 17. External Reset During Operation



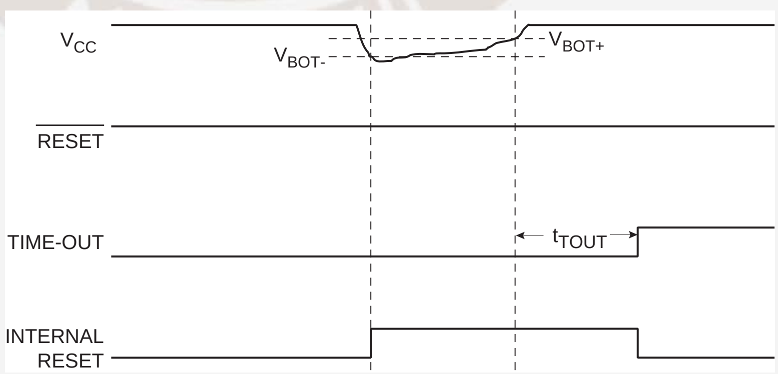
Brown-out Detection

ATmega8 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 18), the Brown-out Reset is immediately activated. When V_{CC} increases above the trigger level (V_{BOT+} in Figure 18), 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 on page 38.

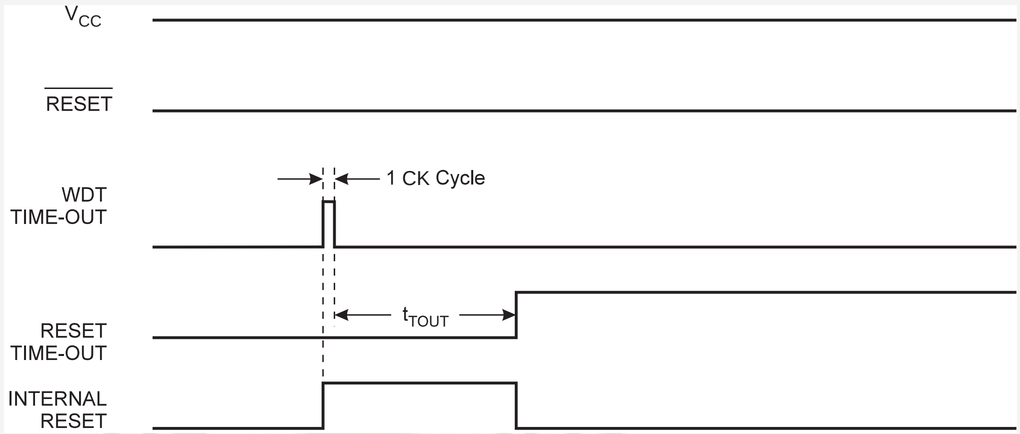
Figure 18. Brown-out Reset During Operation



Watchdog Reset

When the Watchdog times out, it will generate a short reset pulse of 1 CK cycle duration. On the falling edge of this pulse, the delay timer starts counting the time-out period t_{TOUT}. Refer to page 43 for details on operation of the Watchdog Timer.

Figure 19. 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	
	–	–	–	–	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	See Bit Description				

- **Bit 7..4 – Res: Reserved Bits**

These bits are reserved bits in the ATmega8 and always read as zero.

- **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

ATmega8 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)
2. 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

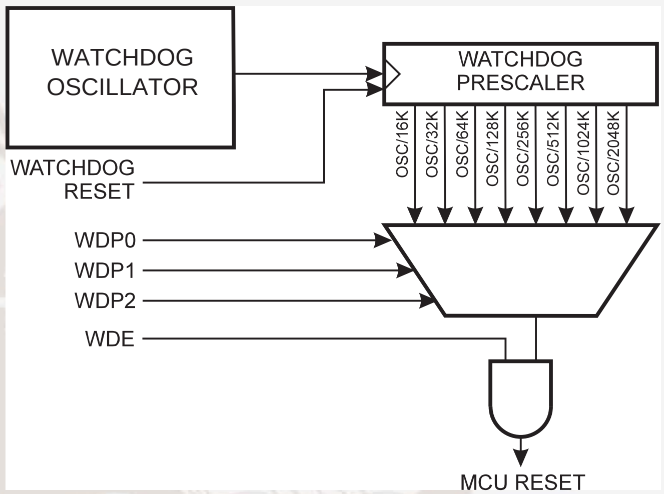
Symbol	Parameter	Min	Typ	Max	Units
V_{BG}	Bandgap reference voltage	1.15	1.30	1.40	V
t_{BG}	Bandgap reference start-up time		40	70	µs
I_{BG}	Bandgap reference current consumption		10		µA

Watchdog Timer

The Watchdog Timer is clocked from a separate On-chip Oscillator which runs at 1MHz. 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 44. 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 ATmega8 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 "Watchdog Timer Control Register – WDTCR" for details.

Figure 20. Watchdog Timer



Watchdog Timer Control Register – WDTCR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	WDCE	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 ATmega8 and will always read as zero.

- **Bit 4 – WDCE: Watchdog Change 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 "Bit 3 – WDE: Watchdog Enable" on page 44for a Watchdog disable procedure. In Safety Level 1 and 2, this bit must also be set when changing the prescaler bits. See the Code Examples on page 45.

- **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 WDCE 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 WDCE 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)	17.1ms	16.3ms
0	0	1	32K (32,768)	34.3ms	32.5ms
0	1	0	64K (65,536)	68.5ms	65ms
0	1	1	128K (131,072)	0.14s	0.13s
1	0	0	256K (262,144)	0.27s	0.26s
1	0	1	512K (524,288)	0.55s	0.52s
1	1	0	1,024K (1,048,576)	1.1s	1.0s
1	1	1	2,048K (2,097,152)	2.2s	2.1s

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.

Timed Sequences for Changing the Configuration of the Watchdog Timer

The sequence for changing the Watchdog Timer configuration differs slightly between the safety levels. Separate procedures are described for each level.

Assembly Code Example

```
WDT_off:
  ; reset WDT
  WDR
  ; Write logical one to WDCE and WDE
  in   r16, WDTCR
  ori  r16, (1<<WDCE)|(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();
  /* Write logical one to WDCE and WDE */
  WDTCR |= (1<<WDCE) | (1<<WDE);
  /* Turn off WDT */
  WDTCR = 0x00;
}
```

Safety Level 1 (WDTON Fuse Unprogrammed)

In this mode, the Watchdog Timer is initially disabled, but can be enabled by writing the WDE bit to 1 without any restriction. A timed sequence is needed when changing the Watchdog Time-out period or disabling an enabled Watchdog Timer. To disable an enabled Watchdog Timer and/or changing the Watchdog Time-out, the following procedure must be followed:

1. In the same operation, write a logic one to WDCE and WDE. A logic one must be written to WDE regardless of the previous value of the WDE bit
2. Within the next four clock cycles, in the same operation, write the WDE and WDP bits as desired, but with the WDCE bit cleared

Safety Level 2 (WDTON Fuse Programmed)

In this mode, the Watchdog Timer is always enabled, and the WDE bit will always read as one. A timed sequence is needed when changing the Watchdog Time-out period. To change the Watchdog Time-out, the following procedure must be followed:

1. In the same operation, write a logical one to WDCE and WDE. Even though the WDE always is set, the WDE must be written to one to start the timed sequence

Within the next four clock cycles, in the same operation, write the WDP bits as desired, but with the WDCE bit cleared. The value written to the WDE bit is irrelevant.

Interrupts

This section describes the specifics of the interrupt handling performed by the ATmega8. For a general explanation of the AVR interrupt handling, refer to "Reset and Interrupt Handling" on page 14.

Interrupt Vectors in ATmega8

Table 18. Reset and Interrupt Vectors

Vector No.	Program Address[2]	Source	Interrupt Definition
1	0x000[1]	RESET	External Pin, Power-on Reset, Brown-out Reset, and Watchdog Reset
2	0x001	INT0	External Interrupt Request 0
3	0x002	INT1	External Interrupt Request 1
4	0x003	TIMER2 COMP	Timer/Counter2 Compare Match
5	0x004	TIMER2 OVF	Timer/Counter2 Overflow
6	0x005	TIMER1 CAPT	Timer/Counter1 Capture Event
7	0x006	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	0x007	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	0x008	TIMER1 OVF	Timer/Counter1 Overflow
10	0x009	TIMER0 OVF	Timer/Counter0 Overflow
11	0x00A	SPI, STC	Serial Transfer Complete
12	0x00B	USART, RXC	USART, Rx Complete
13	0x00C	USART, UDRE	USART Data Register Empty
14	0x00D	USART, TXC	USART, Tx Complete
15	0x00E	ADC	ADC Conversion Complete
16	0x00F	EE_RDY	EEPROM Ready
17	0x010	ANA_COMP	Analog Comparator
18	0x011	TWI	Two-wire Serial Interface
19	0x012	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 202

2. 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 on page 47 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.

Table 19. Reset and Interrupt Vectors Placement

BOOTRST[(1)]	IVSEL	Reset Address	Interrupt Vectors Start Address
1	0	0x000	0x001
1	1	0x000	Boot Reset Address + 0x001
0	0	Boot Reset Address	0x001
0	1	Boot Reset Address	Boot Reset Address + 0x001

Note: 1. The Boot Reset Address is shown in Table 82 on page 213. 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 ATmega8 is:

```
addressLabels Code                Comments
$000          rjmp  RESET         ; Reset Handler
$001          rjmp  EXT_INT0      ; IRQ0 Handler
$002          rjmp  EXT_INT1      ; IRQ1 Handler
$003          rjmp  TIM2_COMP     ; Timer2 Compare Handler
$004          rjmp  TIM2_OVF      ; Timer2 Overflow Handler
$005          rjmp  TIM1_CAPT     ; Timer1 Capture Handler
$006          rjmp  TIM1_COMPA    ; Timer1 CompareA Handler
$007          rjmp  TIM1_COMPB    ; Timer1 CompareB Handler
$008          rjmp  TIM1_OVF      ; Timer1 Overflow Handler
$009          rjmp  TIM0_OVF      ; Timer0 Overflow Handler
$00a          rjmp  SPI_STC       ; SPI Transfer Complete Handler
$00b          rjmp  USART_RXC     ; USART RX Complete Handler
$00c          rjmp  USART_UDRE    ; UDR Empty Handler
$00d          rjmp  USART_TXC     ; USART TX Complete Handler
$00e          rjmp  ADC           ; ADC Conversion Complete Handler
$00f          rjmp  EE_RDY        ; EEPROM Ready Handler
$010          rjmp  ANA_COMP      ; Analog Comparator Handler
$011          rjmp  TWSI          ; Two-wire Serial Interface Handler
$012          rjmp  SPM_RDY       ; Store Program Memory Ready Handler
;
$013   RESET: ldi     r16,high(RAMEND); Main program start
$014          out   SPH,r16       ; Set Stack Pointer to top of RAM
$015          ldi   r16,low(RAMEND)
$016          out   SPL,r16
$017          sei                 ; Enable interrupts
$018          <instr>  xxx
...           ...      ...
```

When the BOOTRST Fuse is unprogrammed, the boot section size set to 2Kbytes 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:

```
AddressLabels Code                Comments
$000          rjmp  RESET          ; Reset handler
;
$001   RESET: ldi   r16,high(RAMEND); Main program start
$002          out   SPH,r16        ; Set Stack Pointer to top of RAM
$003          ldi   r16,low(RAMEND)
$004          out   SPL,r16
$005          sei                ; Enable interrupts
$006          <instr>  xxx
;
.org $c01
$c01          rjmp  EXT_INT0     ; IRQ0 Handler
$c02          rjmp  EXT_INT1     ; IRQ1 Handler
...           ...   ...;
$c12          rjmp  SPM_RDY      ; Store Program Memory Ready Handler
```

When the BOOTRST Fuse is programmed and the boot section size set to 2Kbytes, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
AddressLabels Code              Comments
.org $001
$001          rjmp  EXT_INT0     ; IRQ0 Handler
$002          rjmp  EXT_INT1     ; IRQ1 Handler
...           ...   ...          ;
$012          rjmp  SPM_RDY      ; Store Program Memory Ready Handler
;
.org $c00
$c00          rjmp  RESET        ; Reset handler
;
$c01   RESET: ldi   r16,high(RAMEND); Main program start
$c02          out   SPH,r16      ; Set Stack Pointer to top of RAM
$c03          ldi   r16,low(RAMEND)
$c04          out   SPL,r16
$c05          sei                ; Enable interrupts
$c06          <instr>  xxx
```

When the BOOTRST Fuse is programmed, the boot section size set to 2Kbytes, 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:

```
AddressLabels    Code              Comments
;
.org $c00
$c00          rjmp   RESET         ; Reset handler
$c01          rjmp   EXT_INT0      ; IRQ0 Handler
$c02          rjmp   EXT_INT1      ; IRQ1 Handler
...           ...    ...;
$c12          rjmp   SPM_RDY       ; Store Program Memory Ready Handler
$c13   RESET: ldi       r16,high(RAMEND); Main program start
$c14          out    SPH,r16       ; Set Stack Pointer to top of RAM
$c15          ldi    r16,low(RAMEND)
$c16          out    SPL,r16
$c17          sei                  ; Enable interrupts
$c18          <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	INT0	–	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R	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 202 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 202 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

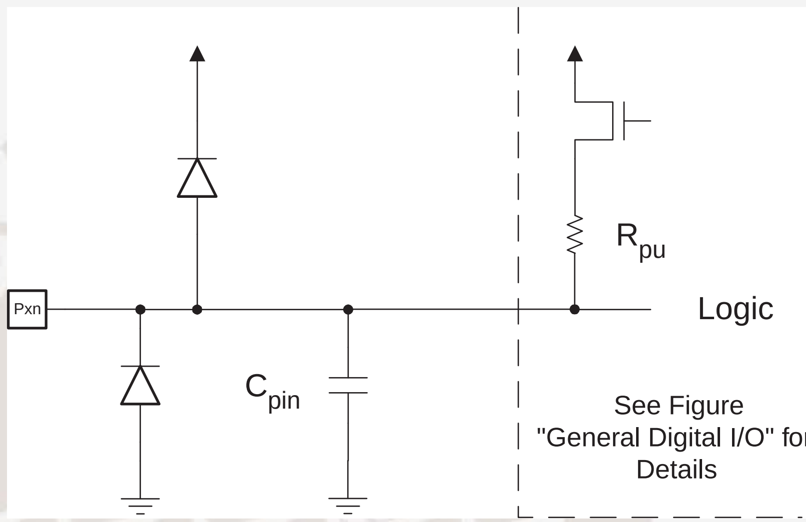
```
void Move_interrupts(void)
{
  /* Enable change of Interrupt Vectors */
  GICR = (1<<IVCE);
  /* Move interrupts to boot Flash section */
  GICR = (1<<IVSEL);
}
```

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_{CC} and Ground as indicated in Figure 21. Refer to "Electrical Characteristics – TA = -40°C to 85°C" on page 235 for a complete list of parameters.

Figure 21. 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 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 65.

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.

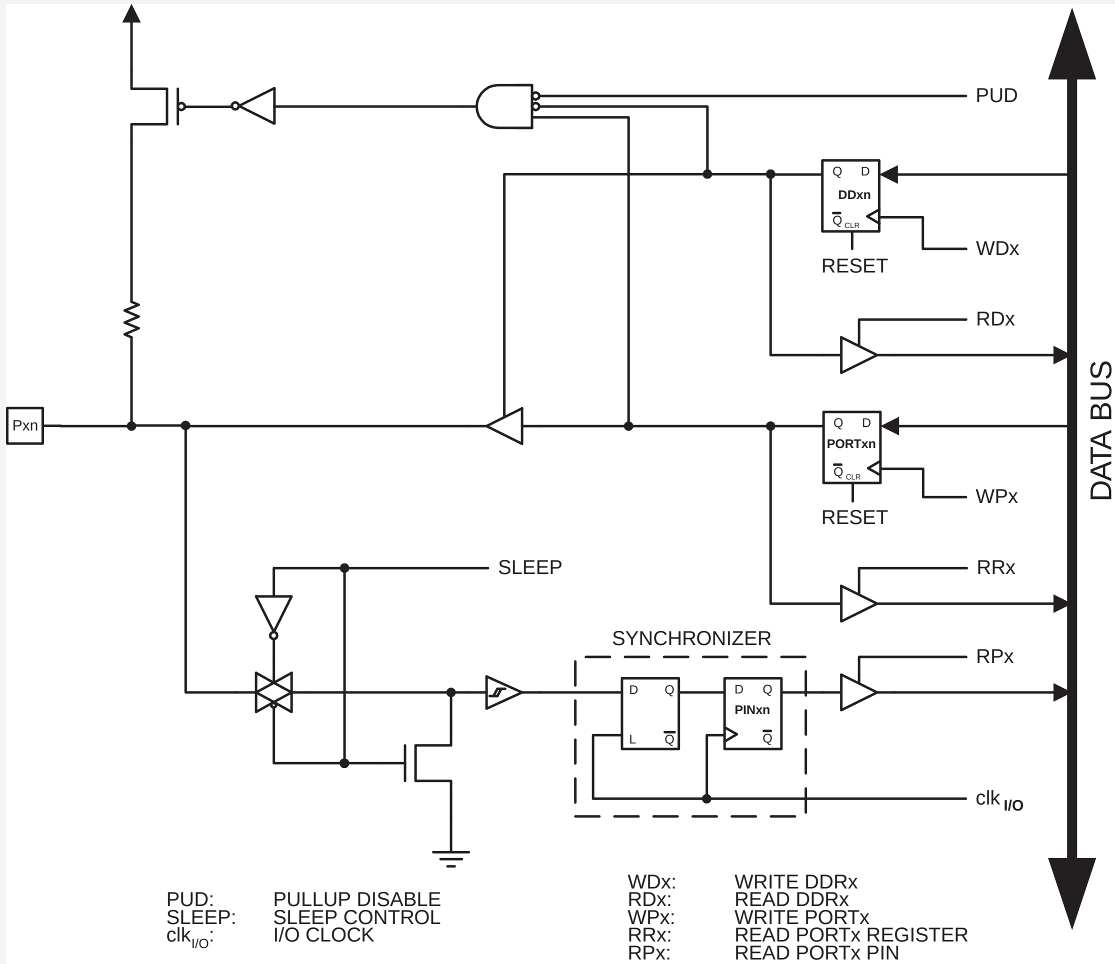
Using the I/O port as General Digital I/O is described in "Ports as General Digital I/O" . Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in "Alternate Port Functions" on page 56. 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 22 on page 52 shows a functional description of one I/O port pin, here generically called Pxn.

Figure 22. 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 3 Register bits: DDxn, PORTxn, and PINxn. As shown in "Register Description for I/O Ports" on page 65, 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.

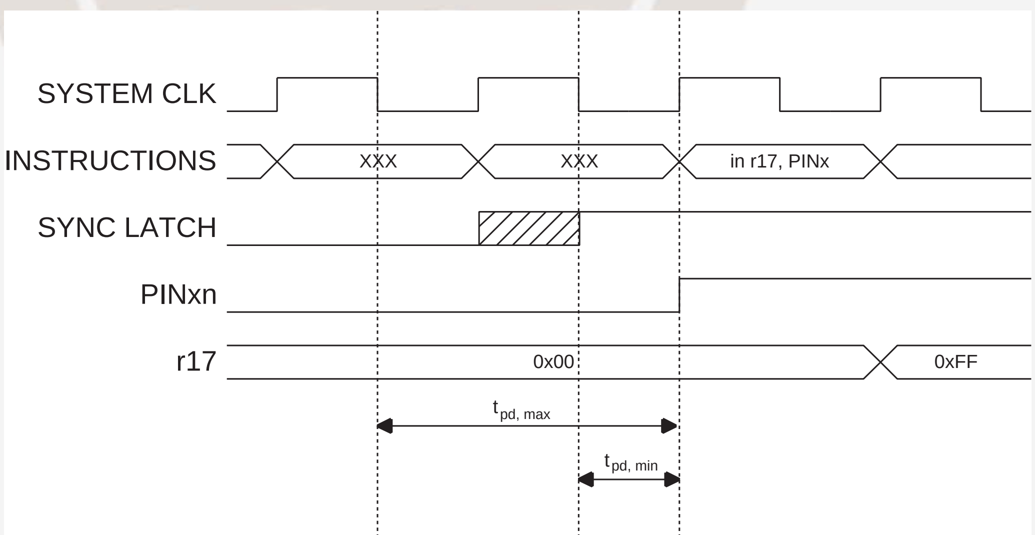
Table 20. Port Pin Configurations

DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if external pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	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 22 on page 52, 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 23 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{pd,max}$ and $t_{pd,min}$, respectively.

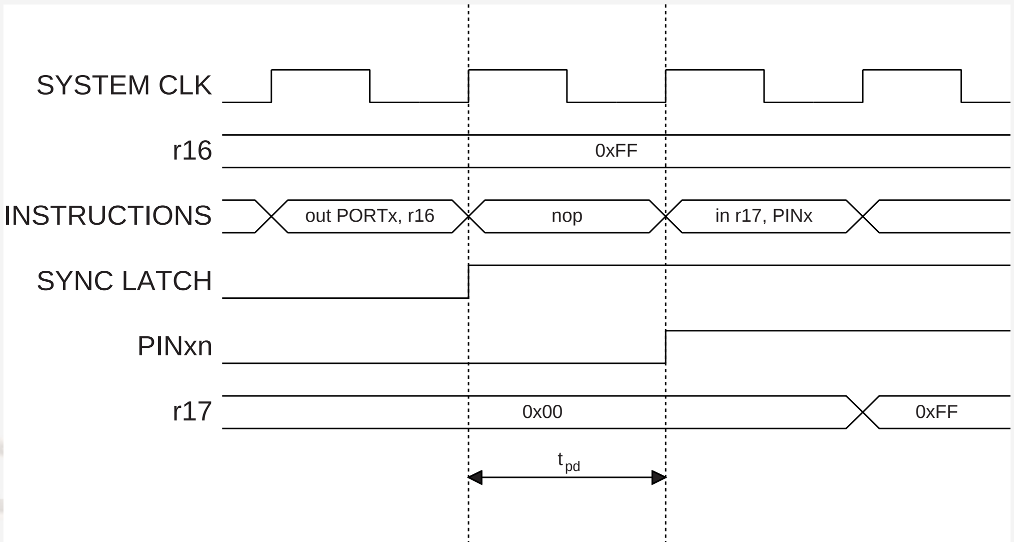
Figure 23. Synchronization when Reading an Externally Applied Pin Value



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 1-½ 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 24. 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 1 system clock period.

Figure 24. 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[(1)]

```
...
; 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[(1)]

```
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 pull-ups 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 22 on page 52, 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, and Standby mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{CC}/2$.

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 56.

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

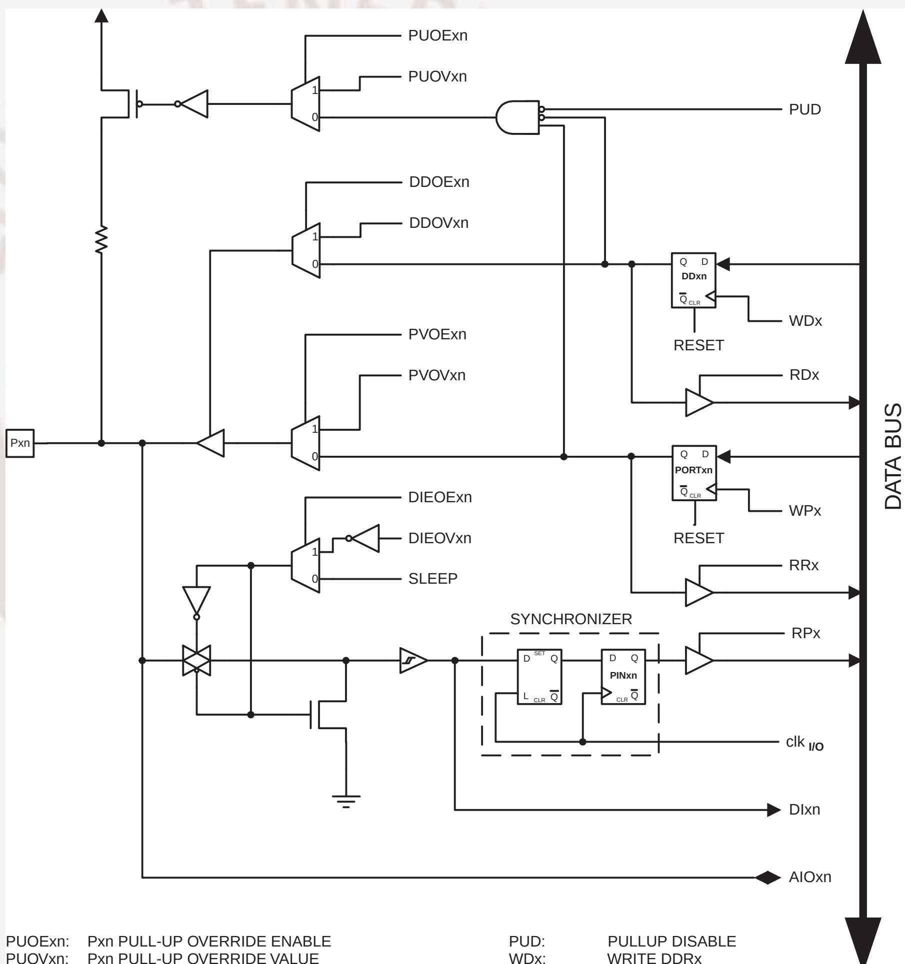
If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled (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_{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 25 shows how the port pin control signals from the simplified Figure 22 on page 52 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 25. Alternate Port Functions[(1)]



PUOExn:	Pxn PULL-UP OVERRIDE ENABLE	PUD:	PULLUP DISABLE
PUOVxn:	Pxn PULL-UP OVERRIDE VALUE	WDx:	WRITE DDRx
DDOExn:	Pxn DATA DIRECTION OVERRIDE ENABLE	RDx:	READ DDRx
DDOVxn:	Pxn DATA DIRECTION OVERRIDE VALUE	RRx:	READ PORTx REGISTER
PVOExn:	Pxn PORT VALUE OVERRIDE ENABLE	WPx:	WRITE PORTx
PVOVxn:	Pxn PORT VALUE OVERRIDE VALUE	RPx:	READ PORTx PIN
DIEOExn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE ENABLE	$clk_{I/O}$:	I/O CLOCK
DIEOVxn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE VALUE	DIxn:	DIGITAL INPUT PIN n ON PORTx
SLEEP:	SLEEP CONTROL	AIOxn:	ANALOG INPUT/OUTPUT PIN n ON PORTx

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

Table 21 summarizes the function of the overriding signals. The pin and port indexes from Figure 25 on page 56 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 IO Register – SFIOR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R	R	R	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 52 for more details about this feature.

Alternate Functions of Port B

The Port B pins with alternate functions are shown in Table 22.

Table 22. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB7	XTAL2 (Chip Clock Oscillator pin 2) TOSC2 (Timer Oscillator pin 2)
PB6	XTAL1 (Chip Clock Oscillator pin 1 or External clock input) TOSC1 (Timer Oscillator pin 1)
PB5	SCK (SPI Bus Master clock Input)
PB4	MISO (SPI Bus Master Input/Slave Output)
PB3	MOSI (SPI Bus Master Output/Slave Input) OC2 (Timer/Counter2 Output Compare Match Output)
PB2	$\overline{\text{SS}}$ (SPI Bus Master Slave select) OC1B (Timer/Counter1 Output Compare Match B Output)
PB1	OC1A (Timer/Counter1 Output Compare Match A Output)
PB0	ICP1 (Timer/Counter1 Input Capture Pin)

The alternate pin configuration is as follows:

- **XTAL2/TOSC2 – Port B, Bit 7**

XTAL2: Chip clock Oscillator pin 2. Used as clock pin for crystal Oscillator or Low-frequency crystal Oscillator. When used as a clock pin, the pin can not be used as an I/O pin.

TOSC2: Timer Oscillator pin 2. Used only if internal calibrated RC Oscillator is selected as chip clock source, and the asynchronous timer is enabled by the correct setting in ASSR. When the AS2 bit in ASSR is set (one) to enable asynchronous clocking of Timer/Counter2, pin PB7 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 cannot be used as an I/O pin.

If PB7 is used as a clock pin, DDB7, PORTB7 and PINB7 will all read 0.

- **XTAL1/TOSC1 – Port B, Bit 6**

XTAL1: Chip clock Oscillator pin 1. Used for all chip clock sources except internal calibrated RC Oscillator. When used as a clock pin, the pin can not be used as an I/O pin.

TOSC1: Timer Oscillator pin 1. Used only if internal calibrated RC Oscillator is selected as chip clock source, and the asynchronous timer is enabled by the correct setting in ASSR. When the AS2 bit in ASSR is set (one) to enable asynchronous clocking of Timer/Counter2, pin PB6 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.

If PB6 is used as a clock pin, DDB6, PORTB6 and PINB6 will all read 0.

- **SCK – Port B, Bit 5**

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 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.

- **MISO – Port B, Bit 4**

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 DDB4. When the SPI is enabled as a Slave, 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.

- **MOSI/OC2 – Port B, Bit 3**

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 DDB3. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB3. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB3 bit.

OC2, Output Compare Match Output: The PB3 pin can serve as an external output for the Timer/Counter2 Compare Match. The PB3 pin has to be configured as an output (DDB3 set (one)) to serve this function. The OC2 pin is also the output pin for the PWM mode timer function.

- **$\overline{SS}$/OC1B – Port B, Bit 2**

$\overline{SS}$: Slave Select input. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB2. 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 DDB2. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB2 bit.

OC1B, Output Compare Match output: The PB2 pin can serve as an external output for the Timer/Counter1 Compare Match B. The PB2 pin has to be configured as an output (DDB2 set (one)) to serve this function. The OC1B pin is also the output pin for the PWM mode timer function.

- **OC1A – Port B, Bit 1**

OC1A, Output Compare Match output: The PB1 pin can serve as an external output for the Timer/Counter1 Compare Match A. The PB1 pin has to be configured as an output (DDB1 set (one)) to serve this function. The OC1A pin is also the output pin for the PWM mode timer function.

- **ICP1 – Port B, Bit 0**

ICP1 – Input Capture Pin: The PB0 pin can act as an Input Capture Pin for Timer/Counter1.

Table 23 on page 60 and Table 24 on page 60 relate the alternate functions of Port B to the overriding signals shown in Figure 25 on page 56. SPI MSTR INPUT and SPI SLAVE OUTPUT constitute the MISO signal, while MOSI is divided into SPI MSTR OUTPUT and SPI SLAVE INPUT.

Table 23. Overriding Signals for Alternate Functions in PB7..PB4

Signal Name	PB7/XTAL2/ TOSC2[(1)(2)]	PB6/XTAL1/ TOSC1[(1)]	PB5/SCK	PB4/MISO
PUOE	$\overline{\text{EXT}}$ • ($\overline{\text{INTRC}}$ + AS2)	$\overline{\text{INTRC}}$ + AS2	SPE • $\overline{\text{MSTR}}$	SPE • MSTR
PUO	0	0	PORTB5 • $\overline{\text{PUD}}$	PORTB4 • $\overline{\text{PUD}}$
DDOE	$\overline{\text{EXT}}$ • ($\overline{\text{INTRC}}$ + AS2)	$\overline{\text{INTRC}}$ + AS2	SPE • $\overline{\text{MSTR}}$	SPE • MSTR
DDOV	0	0	0	0
PVOE	0	0	SPE • MSTR	SPE • $\overline{\text{MSTR}}$
PVOV	0	0	SCK OUTPUT	SPI SLAVE OUTPUT
DIEOE	$\overline{\text{EXT}}$ • ($\overline{\text{INTRC}}$ + AS2)	$\overline{\text{INTRC}}$ + AS2	0	0
DIEOV	0	0	0	0
DI	–	–	SCK INPUT	SPI MSTR INPUT
AIO	Oscillator Output	Oscillator/Clock Input	–	–

Notes:
1. INTRC means that the internal RC Oscillator is selected (by the CKSEL Fuse)
2. EXT means that the external RC Oscillator or an external clock is selected (by the CKSEL Fuse)

Table 24. Overriding Signals for Alternate Functions in PB3..PB0

Signal Name	PB3/MOSI/OC2	PB2/$\overline{\text{SS}}$/OC1B	PB1/OC1A	PB0/ICP1
PUOE	SPE • $\overline{\text{MSTR}}$	SPE • $\overline{\text{MSTR}}$	0	0
PUO	PORTB3 • $\overline{\text{PUD}}$	PORTB2 • $\overline{\text{PUD}}$	0	0
DDOE	SPE • $\overline{\text{MSTR}}$	SPE • $\overline{\text{MSTR}}$	0	0
DDOV	0	0	0	0
PVOE	SPE • MSTR + OC2 ENABLE	OC1B ENABLE	OC1A ENABLE	0
PVOV	SPI MSTR OUTPUT + OC2	OC1B	OC1A	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	SPI SLAVE INPUT	SPI $\overline{\text{SS}}$	–	ICP1 INPUT
AIO	–	–	–	–

Alternate Functions of Port C

The Port C pins with alternate functions are shown in Table 25.

Table 25. Port C Pins Alternate Functions

Port Pin	Alternate Function
PC6	$\overline{\text{RESET}}$ (Reset pin)
PC5	ADC5 (ADC Input Channel 5) SCL (Two-wire Serial Bus Clock Line)
PC4	ADC4 (ADC Input Channel 4) SDA (Two-wire Serial Bus Data Input/Output Line)
PC3	ADC3 (ADC Input Channel 3)
PC2	ADC2 (ADC Input Channel 2)
PC1	ADC1 (ADC Input Channel 1)
PC0	ADC0 (ADC Input Channel 0)

The alternate pin configuration is as follows:

- **$\overline{\text{RESET}}$ – Port C, Bit 6**

$\overline{\text{RESET}}$, Reset pin: When the RSTDISBL Fuse is programmed, this pin functions as a normal I/O pin, and the part will have to rely on Power-on Reset and Brown-out Reset as its reset sources. When the RSTDISBL Fuse is unprogrammed, the reset circuitry is connected to the pin, and the pin can not be used as an I/O pin.

If PC6 is used as a reset pin, DDC6, PORTC6 and PINC6 will all read 0.

- **SCL/ADC5 – Port C, Bit 5**

SCL, Two-wire Serial Interface Clock: When the TWEN bit in TWCR is set (one) to enable the Two-wire Serial Interface, pin PC5 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.

PC5 can also be used as ADC input Channel 5. Note that ADC input channel 5 uses digital power.

- **SDA/ADC4 – Port C, Bit 4**

SDA, Two-wire Serial Interface Data: When the TWEN bit in TWCR is set (one) to enable the Two-wire Serial Interface, pin PC4 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.

PC4 can also be used as ADC input Channel 4. Note that ADC input channel 4 uses digital power.

- **ADC3 – Port C, Bit 3**

PC3 can also be used as ADC input Channel 3. Note that ADC input channel 3 uses analog power.

- **ADC2 – Port C, Bit 2**

PC2 can also be used as ADC input Channel 2. Note that ADC input channel 2 uses analog power.

- **ADC1 – Port C, Bit 1**

PC1 can also be used as ADC input Channel 1. Note that ADC input channel 1 uses analog power.

- **ADC0 – Port C, Bit 0**

PC0 can also be used as ADC input Channel 0. Note that ADC input channel 0 uses analog power.

Table 26 and Table 27 relate the alternate functions of Port C to the overriding signals shown in Figure 25 on page 56.

Table 26. Overriding Signals for Alternate Functions in PC6..PC4

Signal Name	PC6/$\overline{\text{RESET}}$	PC5/SCL/ADC5	PC4/SDA/ADC4
PUOE	$\overline{\text{RSTDISBL}}$	TWEN	TWEN
PUOV	1	PORTC5 • $\overline{\text{PUD}}$	PORTC4 • $\overline{\text{PUD}}$
DDOE	$\overline{\text{RSTDISBL}}$	TWEN	TWEN
DDOV	0	SCL_OUT	SDA_OUT
PVOE	0	TWEN	TWEN
PVOV	0	0	0
DIEOE	$\overline{\text{RSTDISBL}}$	0	0
DIEOV	0	0	0
DI	–	–	–
AIO	RESET INPUT	ADC5 INPUT / SCL INPUT	ADC4 INPUT / SDA INPUT

Table 27. Overriding Signals for Alternate Functions in PC3..PC0[(1)]

Signal Name	PC3/ADC3	PC2/ADC2	PC1/ADC1	PC0/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

Note: 1. When enabled, the Two-wire Serial Interface enables slew-rate controls on the output pins PC4 and PC5. 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

The Port D pins with alternate functions are shown in Table 28.

Table 28. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD7	AIN1 (Analog Comparator Negative Input)
PD6	AIN0 (Analog Comparator Positive Input)
PD5	T1 (Timer/Counter 1 External Counter Input)
PD4	XCK (USART External Clock Input/Output) T0 (Timer/Counter 0 External Counter Input)
PD3	INT1 (External Interrupt 1 Input)
PD2	INT0 (External Interrupt 0 Input)
PD1	TXD (USART Output Pin)
PD0	RXD (USART Input Pin)

The alternate pin configuration is as follows:

- **AIN1 – Port D, Bit 7**

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.

- **AIN0 – Port D, Bit 6**

AIN0, 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.

- **T1 – Port D, Bit 5**

T1, Timer/Counter1 counter source.

- **XCK/T0 – Port D, Bit 4**

XCK, USART external clock.

T0, Timer/Counter0 counter source.

- **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**

INT0, 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 29 on page 64 and Table 30 on page 64 relate the alternate functions of Port D to the overriding signals shown in Figure 25 on page 56.

Table 29. Overriding Signals for Alternate Functions PD7..PD4

Signal Name	PD7/AIN1	PD6/AIN0	PD5/T1	PD4/XCK/T0
PUOE	0	0	0	0
PUO	0	0	0	0
OOE	0	0	0	0
OO	0	0	0	0
PVOE	0	0	0	UMSEL
PVO	0	0	0	XCK OUTPUT
DIEOE	0	0	0	0
DIEO	0	0	0	0
DI	–	–	T1 INPUT	XCK INPUT / T0 INPUT
AIO	AIN1 INPUT	AIN0 INPUT	–	–

Table 30. Overriding Signals for Alternate Functions in PD3..PD0

Signal Name	PD3/INT1	PD2/INT0	PD1/TXD	PD0/RXD
PUOE	0	0	TXEN	RXEN
PUO	0	0	0	PORTD0 • $\overline{\text{PUD}}$
OOE	0	0	TXEN	RXEN
OO	0	0	1	0
PVOE	0	0	TXEN	0
PVO	0	0	TXD	0
DIEOE	INT1 ENABLE	INT0 ENABLE	0	0
DIEO	1	1	0	0
DI	INT1 INPUT	INT0 INPUT	–	RXD
AIO	–	–	–	–

Register Description for I/O Ports

The 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	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The 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	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The 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	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

The Port C Data Register – PORTC

Bit	7	6	5	4	3	2	1	0	
	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Port C Data Direction Register – DDRC

Bit	7	6	5	4	3	2	1	0	
	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Port C Input Pins Address – PINC

Bit	7	6	5	4	3	2	1	0	
	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

The 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	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The 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	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The 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	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

External Interrupts

The external interrupts are triggered by the INT0, and INT1 pins. Observe that, if enabled, the interrupts will trigger even if the INT0..1 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. This is set up as indicated in the specification for the MCU Control Register – MCUCR. When the external interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INT0 and INT1 requires the presence of an I/O clock, described in "Clock Systems and their Distribution" on page 25. Low level interrupts on INT0/INT1 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 – TA = -40°C to 85°C" on page 235. 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 25. 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	
	SE	SM2	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	

- **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 31. 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 31. 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 32. 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 32. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request
0	1	Any logical change on INT0 generates an interrupt request
1	0	The falling edge of INT0 generates an interrupt request
1	1	The rising edge of INT0 generates an interrupt request

General Interrupt Control Register – GICR

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	–	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **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 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.

General Interrupt Flag Register – GIFR

Bit	7	6	5	4	3	2	1	0	
	INTF1	INTF0	–	–	–	–	–	–	GIFR
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – INTF1: External Interrupt Flag 1**

When an event 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 event on the INT0 pin triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the INT0 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 INT0 is configured as a level interrupt.

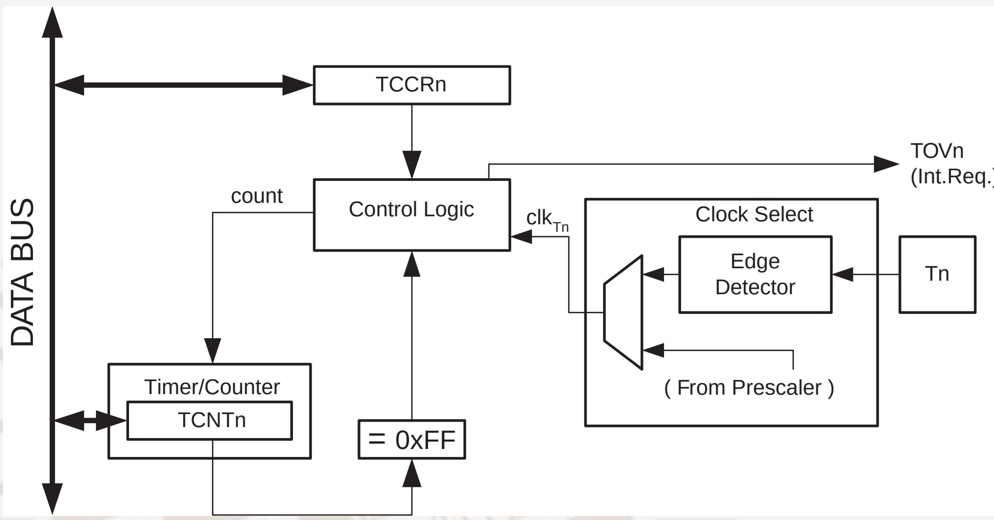
8-bit Timer/Counter0

Timer/Counter0 is a general purpose, single channel, 8-bit Timer/Counter module. The main features are:
- **Single Channel Counter**
- **Frequency Generator**
- **External Event Counter**
- **10-bit Clock Prescaler**

Overview

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 26. For the actual placement of I/O pins, refer to "Pin Configurations" 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 71.

Figure 26. 8-bit Timer/Counter Block Diagram



Registers

The Timer/Counter (TCNT0) is an 8-bit register. 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 or 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 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}).

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 33 are also used extensively throughout this datasheet.

Table 33. Definitions

BOTTOM	The counter reaches the BOTTOM when it becomes 0x00
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255)

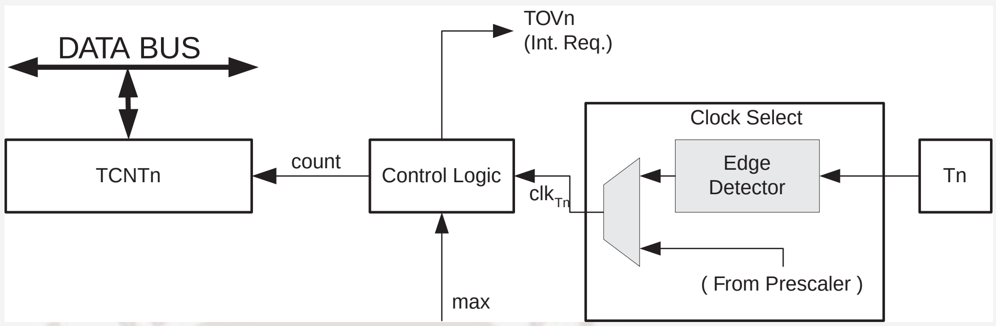
Timer/Counter Clock Sources

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 73.

Counter Unit

The main part of the 8-bit Timer/Counter is the programmable counter unit. Figure 27 shows a block diagram of the counter and its surroundings.

Figure 27. Counter Unit Block Diagram



Signal description (internal signals):

count Increment TCNT0 by 1

clk_{Tn} Timer/Counter clock, referred to as clk_{T0} in the following

max Signalize that TCNT0 has reached maximum value

The counter is incremented 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 clk_{T0} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

Operation

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 (MAX = 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. A new counter value can be written anytime.

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 28 on page 71 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value.

Figure 28. Timer/Counter Timing Diagram, No Prescaling

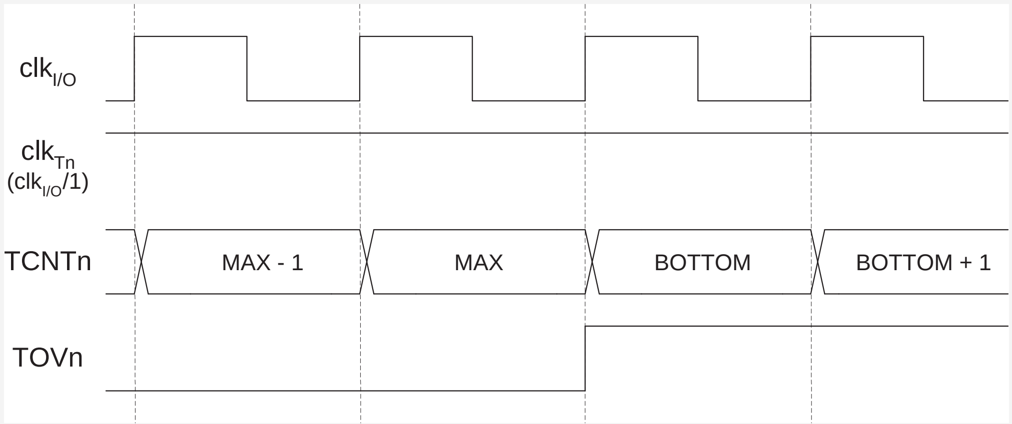
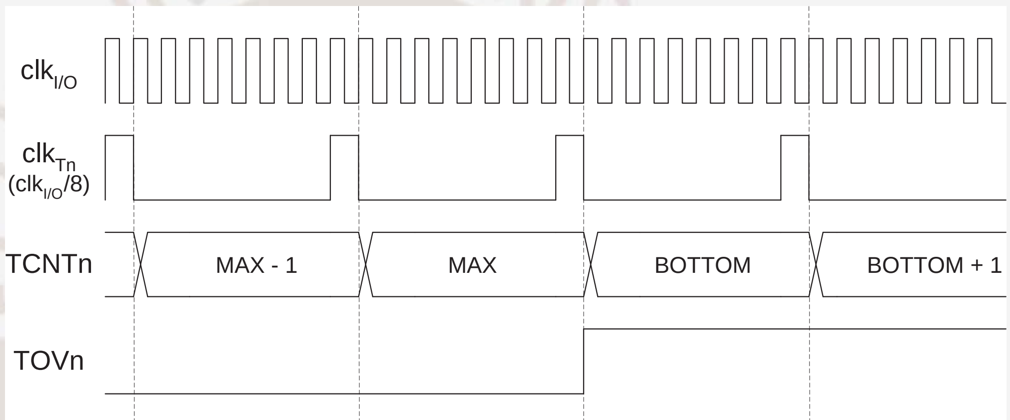


Figure 29 shows the same timing data, but with the prescaler enabled.

Figure 29. Timer/Counter Timing Diagram, 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	
	–	–	–	–	–	CS02	CS01	CS00	TCCR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 2:0 – CS02:0: Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter.

Table 34. 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	$clk_{I/O}$/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 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

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
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	

The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter.

Timer/Counter Interrupt Mask Register – TIMSK

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	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 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	–	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 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.

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).

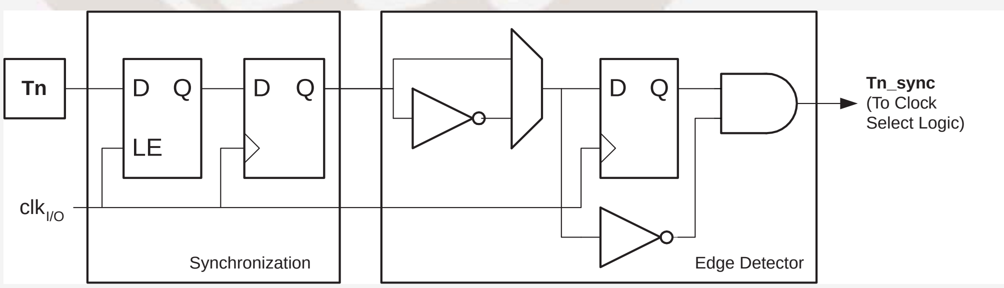
It is possible to use the prescaler reset for synchronizing the Timer/Counter to program execution. However, care must be taken if the other Timer/Counter that shares the same prescaler also uses prescaling. A prescaler reset will affect the prescaler period for all Timer/Counters it is connected to.

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 30 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 30. T1/T0 Pin Sampling



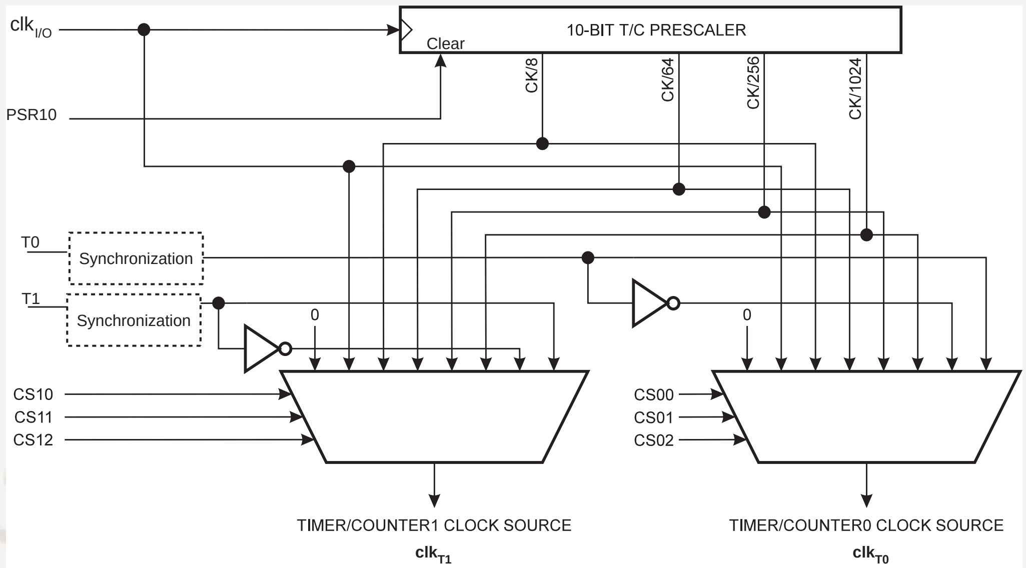
The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge has been applied to the T1/T0 pin to the counter is updated.

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 ($f_{ExtClk} < f_{clk\_I/O}/2$) given a 50/50% duty cycle. Since the edge detector uses 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 31. Prescaler for Timer/Counter0 and Timer/Counter1[(1)]



Note: 1. The synchronization logic on the input pins (T1/T0) is shown in Figure 30 on page 73

Special Function IO Register – SFIOR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **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:

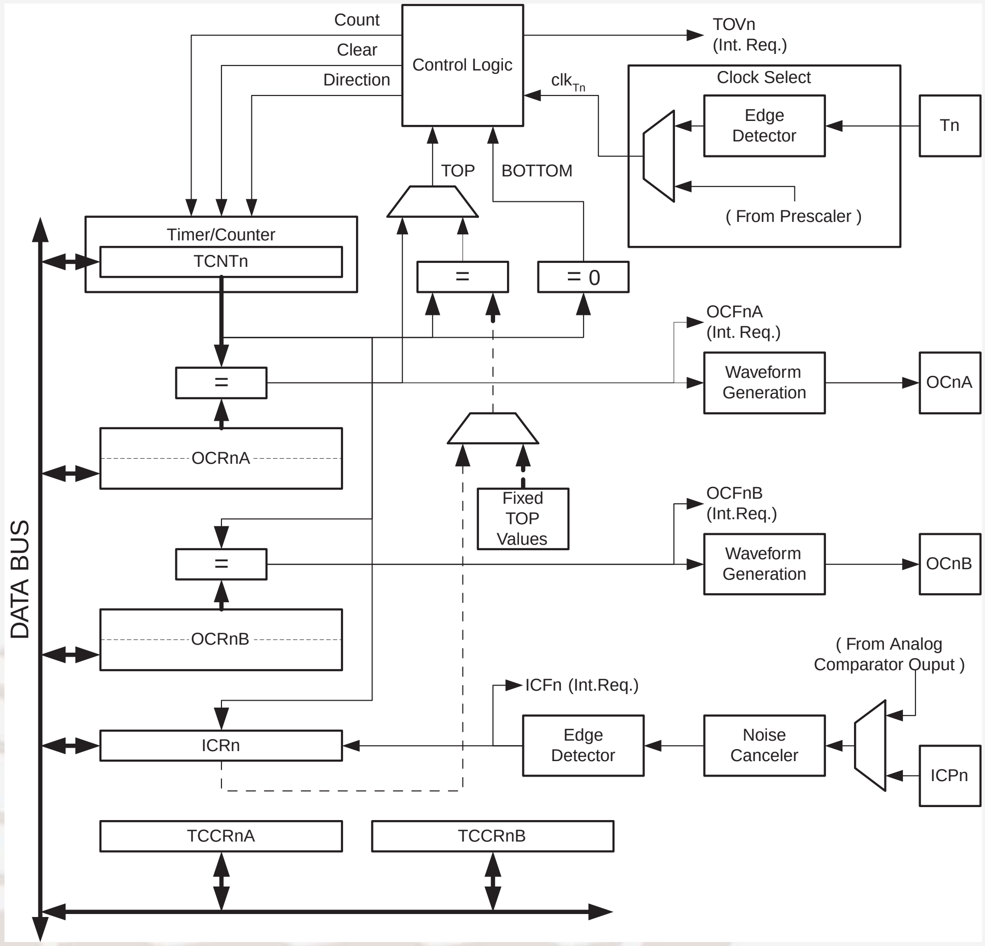
- **True 16-bit Design (that is, allows 16-bit PWM)**
- **Two Independent Output Compare Units**
- **Double Buffered Output Compare Registers**
- **One Input Capture Unit**
- **Input Capture Noise Canceler**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch-free, Phase Correct Pulse Width Modulator (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 channel. 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 32 on page 76. For the actual placement of I/O pins, refer to “Pin Configurations” 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 96.

Figure 32. 16-bit Timer/Counter Block Diagram[(1)]



Note: 1. Refer to "Pin Configurations" on page 2, Table 22 on page 58, and Table 28 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 77. 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 83. The Compare Match event will also set the Compare Match Flag (OCF1A/B) which can be used to generate an Output Compare interrupt request.

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 186). 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 35. Definitions

BOTTOM	The counter reaches the *BOTTOM* when it becomes 0x0000.
MAX	The counter reaches its *MAX*imum when it becomes 0xFFFF (decimal 65535).
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 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. The 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 the 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 TCNT1H,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 8

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[1]

```
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[1]

```
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 8

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[(1)]

```
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 8

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.

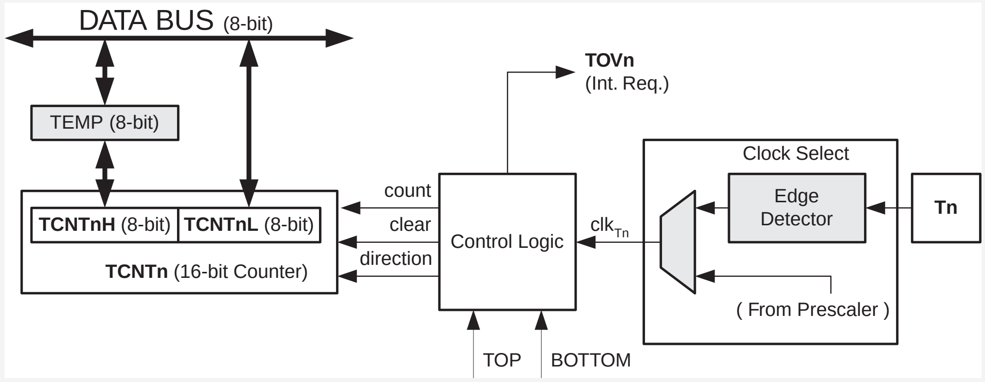
Timer/Counter Clock Sources

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 73.

Counter Unit

The main part of the 16-bit Timer/Counter is the programmable 16-bit bi-directional counter unit. Figure 33 on page 81 shows a block diagram of the counter and its surroundings.

Figure 33. Counter Unit Block Diagram



Signal description (internal signals):

count	Increment or decrement TCNT1 by 1
direction	Select between increment and decrement
clear	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 eight 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 87.

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.

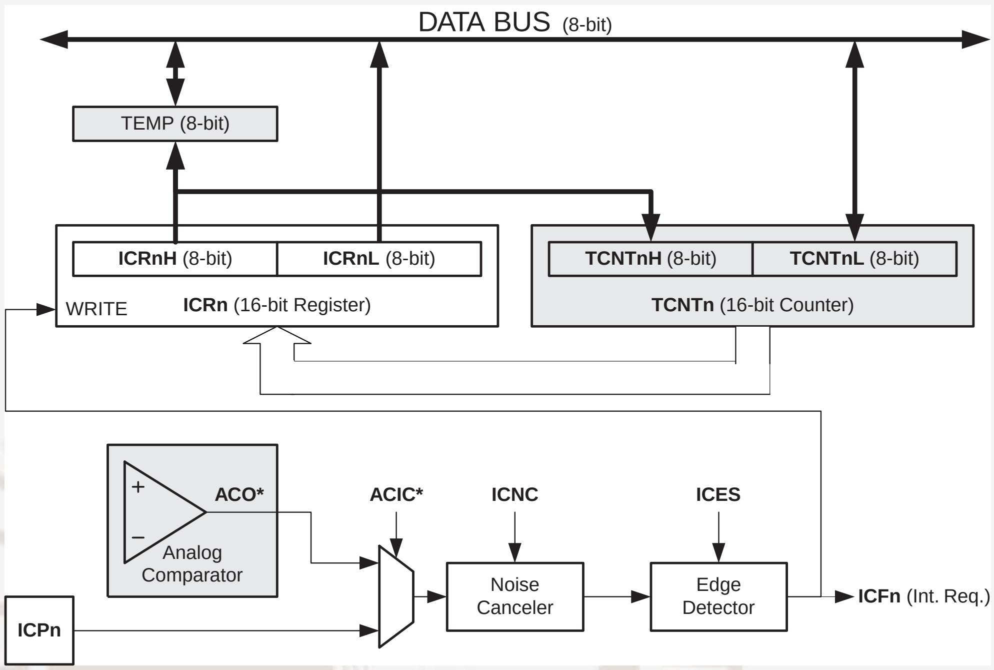
Input Capture Unit

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 34. 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 34. 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 77.

Input Capture Pin Source

The main trigger source for the Input Capture unit is the *Input Capture Pin* (ICP1). Timer/Counter 1 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 30 on page 73). 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).

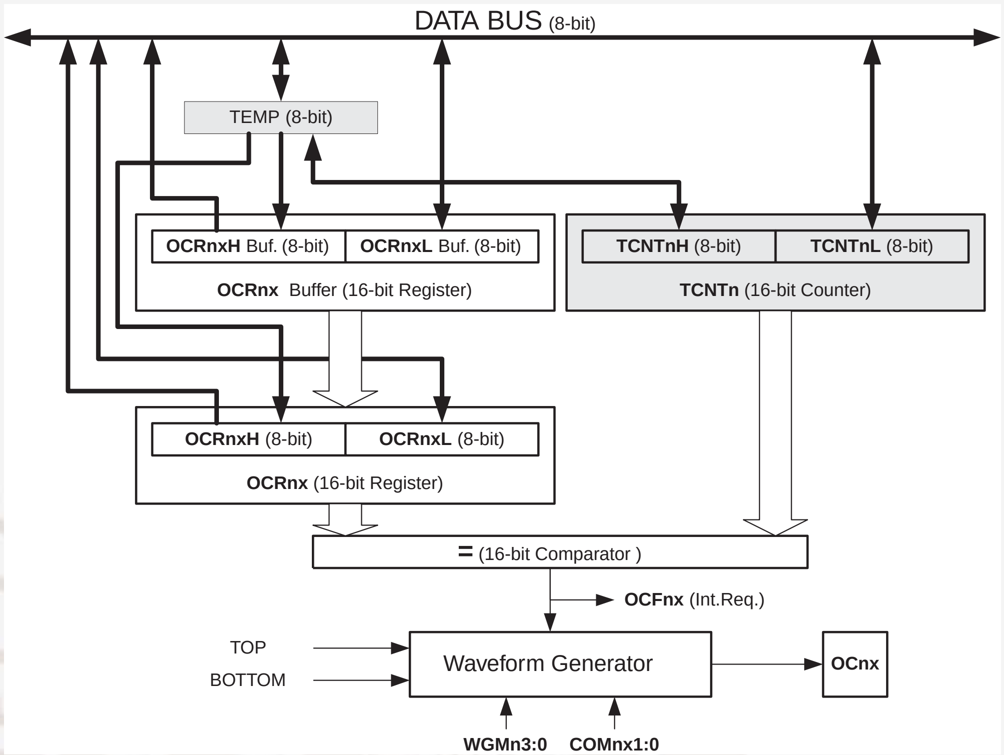
Output Compare Units

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 87).

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 35 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/Counter 1), 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 35. 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-bit 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 77.

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 channels, 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.

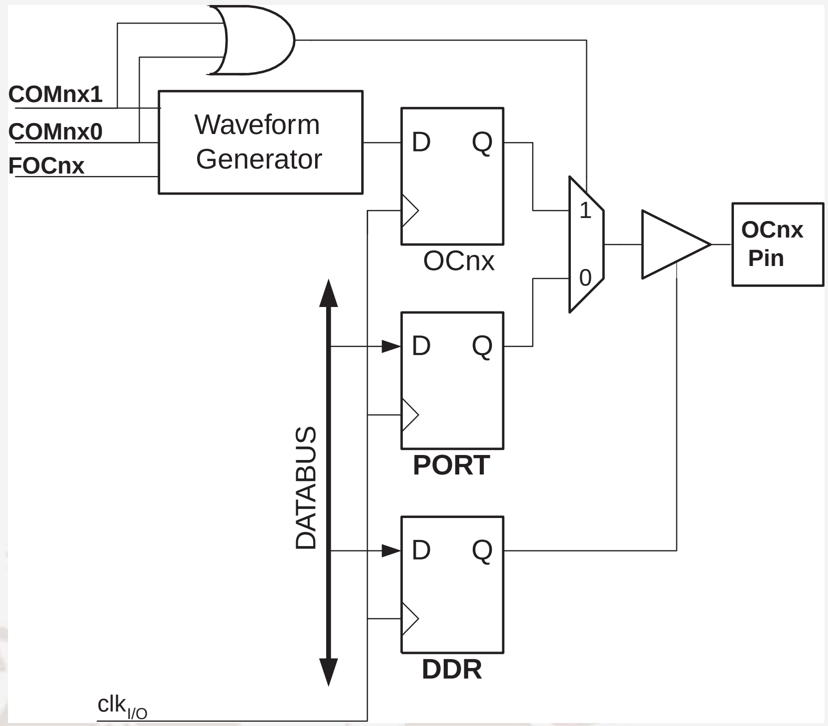
The setup of the OC1x should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC1x value is to use the Force Output Compare (FOC1x) strobe bits in Normal mode. The OC1x Register keeps its value even when changing between Waveform Generation 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 36 on page 86 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 36. 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 36 on page 96, Table 37 on page 96 and Table 38 on page 97 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 96.

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 36 on page 96. For fast PWM mode refer to Table 37 on page 96, and for phase correct and phase and frequency correct PWM refer to Table 38 on page 97.

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.

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 85.

For detailed timing information refer to "Timer/Counter Timing Diagrams" on page 94.

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.

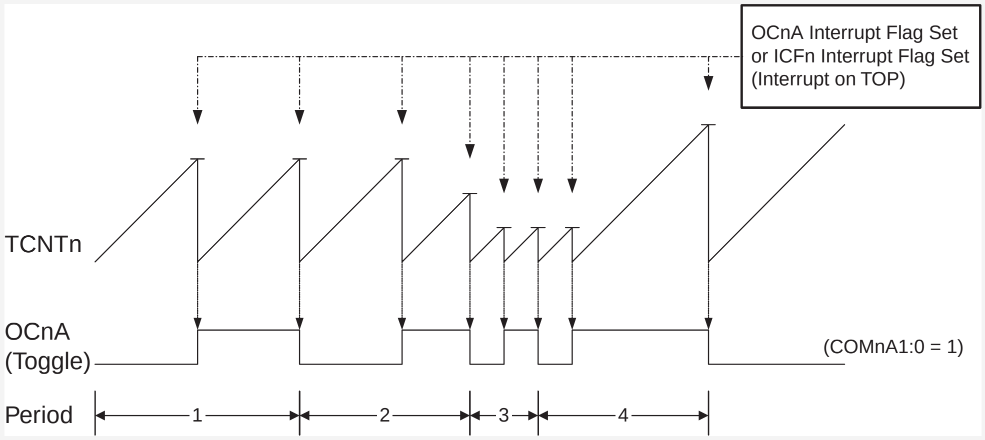
The Output Compare units can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

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 37 on page 88. The counter value (TCNT1) increases until a Compare Match occurs with either OCR1A or ICR1, and then counter (TCNT1) is cleared.

Figure 37. CTC Mode, Timing Diagram



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 $f_{OC1A} = f_{clk\_I/O}/2$ when OCR1A is set to zero (0x0000). The waveform frequency is defined by the following equation:

$$f_{OCnA} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$$

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

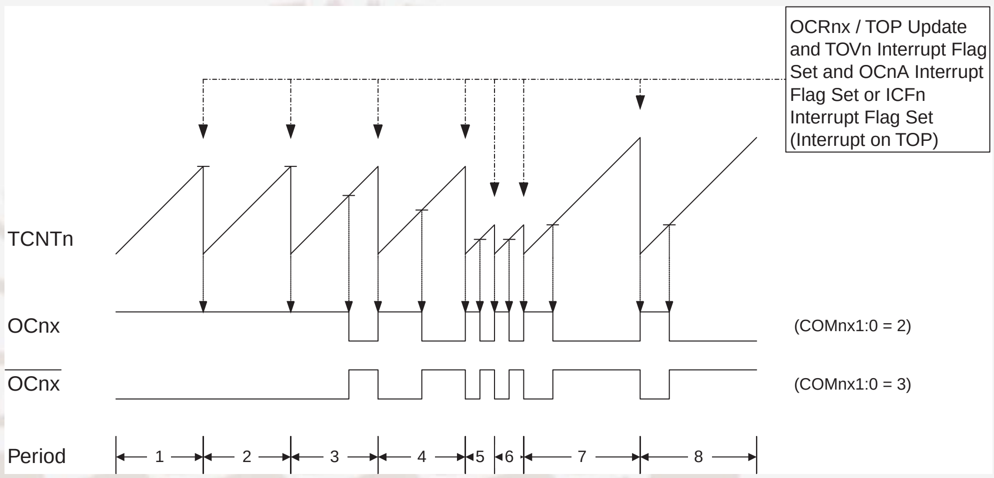
The *fast Pulse Width Modulation* or fast PWM mode (WGM13:0 = 5, 6, 7, 14, or 15) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM options by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. In non-inverting 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 38. 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 38. Fast PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV1) is set each time the counter reaches TOP. In addition the OCF1A 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 37 on page 96. 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 $f_{OC1A} = f_{clk\_I/O}/2$ 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

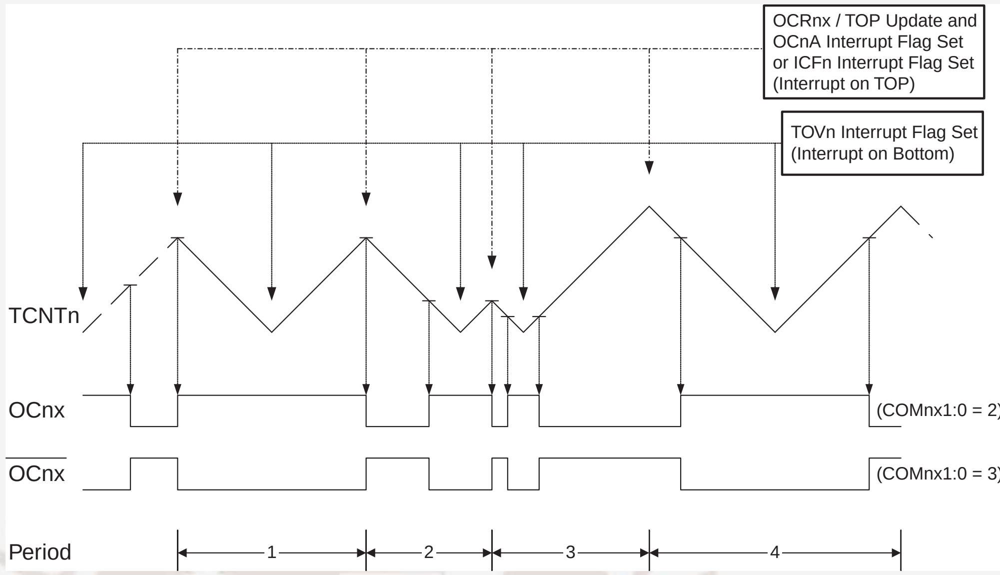
The *phase correct Pulse Width Modulation* or phase correct PWM mode (WGM13:0 = 1, 2, 3, 10, or 11) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is, like the phase and frequency correct PWM mode, based on a dual-slope operation. The counter counts 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 39 on page 91. 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 39. 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 39 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 38 on page 97. 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 (WMG13:0 = 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 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.

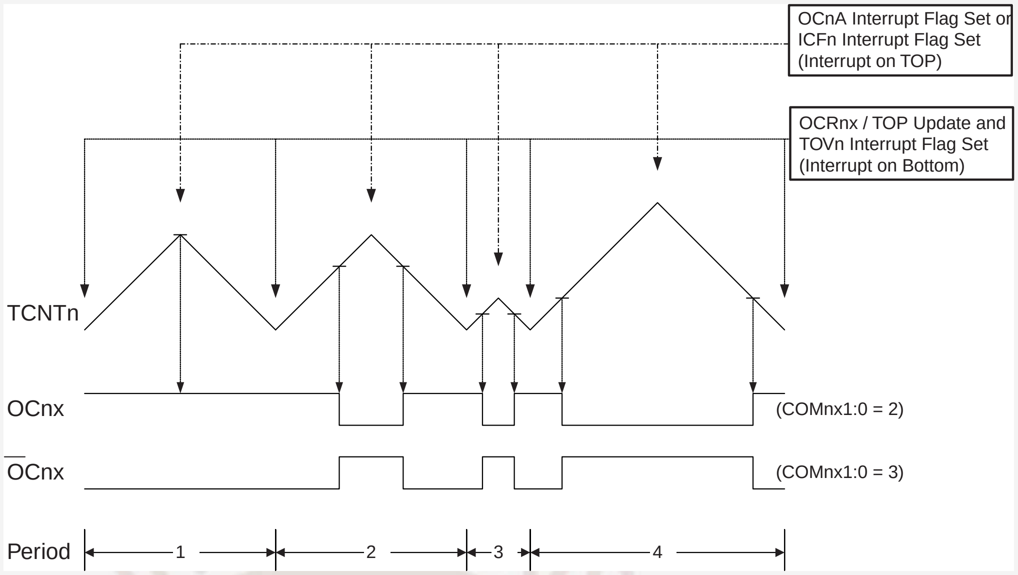
The main difference between the phase correct, and the phase and frequency correct PWM mode is the time the OCR1x Register is updated by the OCR1x Buffer Register, (see Figure 39 on page 91 and Figure 40 on page 93).

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 40 on page 93. 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 40. 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 40 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 38 on page 97. 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:

$$f_{OCnxPFCPWM} = \frac{f_{\text{clk\_I/O}}}{2 \cdot N \cdot TOP}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1x Register represents 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 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 41 shows a timing diagram for the setting of OCF1x.

Figure 41. Timer/Counter Timing Diagram, Setting of OCF1x, no Prescaling

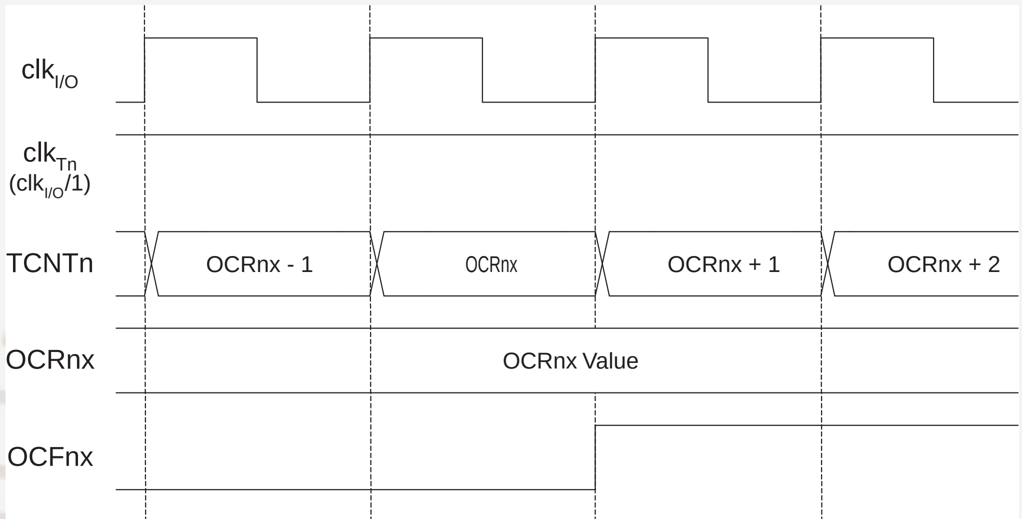


Figure 42 shows the same timing data, but with the prescaler enabled.

Figure 42. Timer/Counter Timing Diagram, Setting of OCF1x, with Prescaler ($f_{clk\_I/O}/8$)

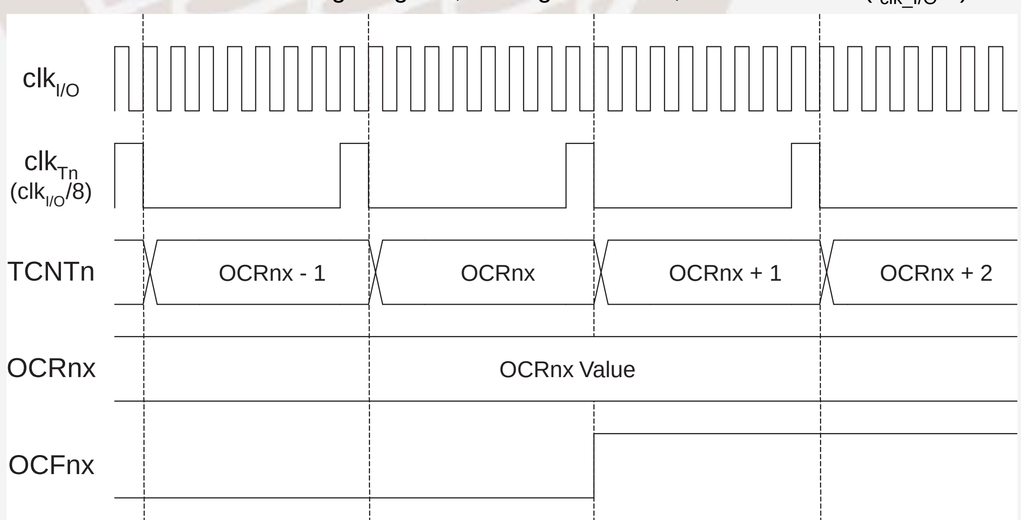


Figure 43 on page 95 shows the count sequence close to TOP in various modes. When using phase and frequency correct PWM mode the OCR1x Register is updated at BOTTOM. The timing diagrams will be the same, but TOP should be replaced by BOTTOM, TOP-1 by BOTTOM+1 and so on. The same renaming applies for modes that set the TOV1 Flag at BOTTOM.

Figure 43. Timer/Counter Timing Diagram, no Prescaling

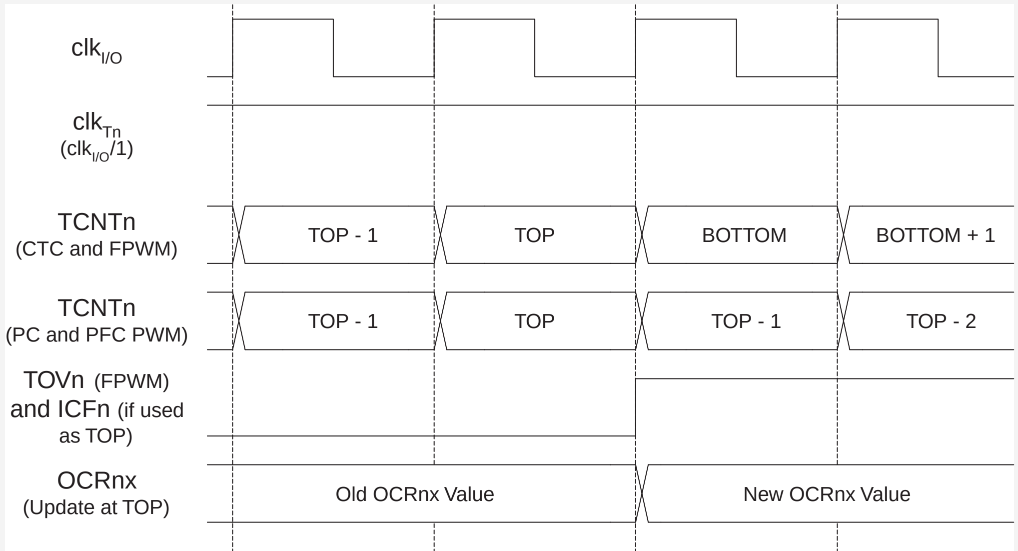
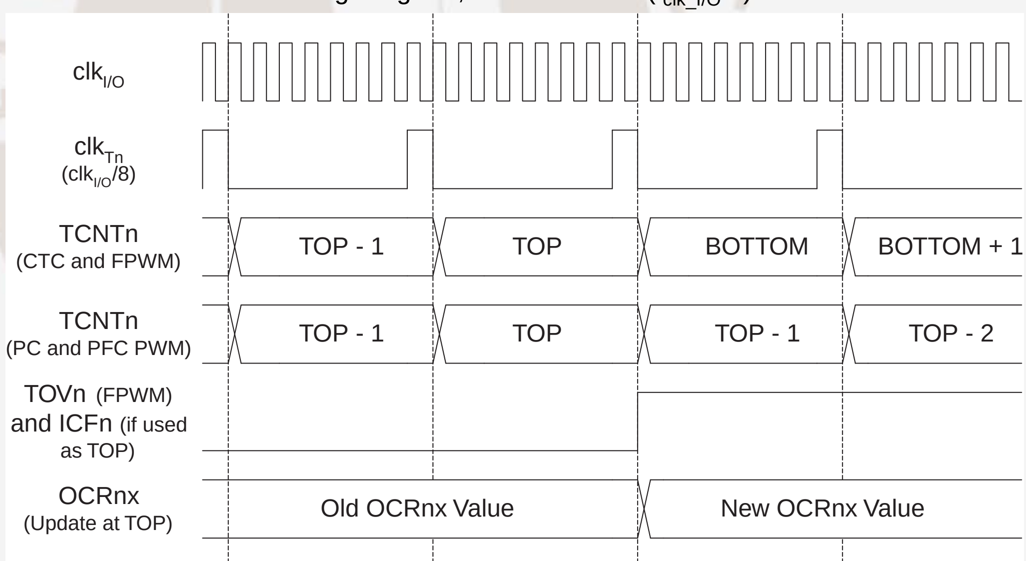


Figure 44 shows the same timing data, but with the prescaler enabled.

Figure 44. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)



16-bit Timer/Counter Register Description

Timer/Counter 1 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 36 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to a normal or a CTC mode (non-PWM).

Table 36. 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 37 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to the fast PWM mode.

Table 37. 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 WGM1 settings, normal port operation, OC1A/OC1B 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 88 for more details

Table 38 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 38. 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 OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM1 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 90 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 39. 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 87).

Table 39. Waveform Generation Mode Bit Description

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation[1]	TOP	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	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP

Table 39. Waveform Generation Mode Bit Description (Continued)

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation[1]	TOP	Update of OCR1x	TOV1 Flag Set on
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	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/Counter 1 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 41 on page 94 and Figure 42 on page 94.

Table 40. 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/Counter 1 – TCNT1H and TCNT1L

Bit	7	6	5	4	3	2	1	0	
	TCNT1[15:8]								TCNT1H
	TCNT1[7:0]								TCNT1L
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	

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 77.

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

Bit	7	6	5	4	3	2	1	0	
	OCR1A[15:8]								OCR1AH
	OCR1A[7:0]								OCR1AL
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	

Output Compare Register 1 B – OCR1BH and OCR1BL

Bit	7	6	5	4	3	2	1	0	
	OCR1B[15:8]								OCR1BH
	OCR1B[7:0]								OCR1BL
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	

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 77.

Input Capture Register 1 – ICR1H and ICR1L

Bit	7	6	5	4	3	2	1	0	
	ICR1[15:8]								ICR1H
	ICR1[7:0]								ICR1L
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	

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 77.

Timer/Counter Interrupt Mask Register – TIMSK[(1)]

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	TIMSK
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	

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 46) 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 46) 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 46) 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 46) is executed when the TOV1 Flag, located in TIFR, is set.

Timer/Counter Interrupt Flag Register – TIFR[1]

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	–	TOV0	TIFR
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	

Note: 1. 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

- **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 39 on page 97 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

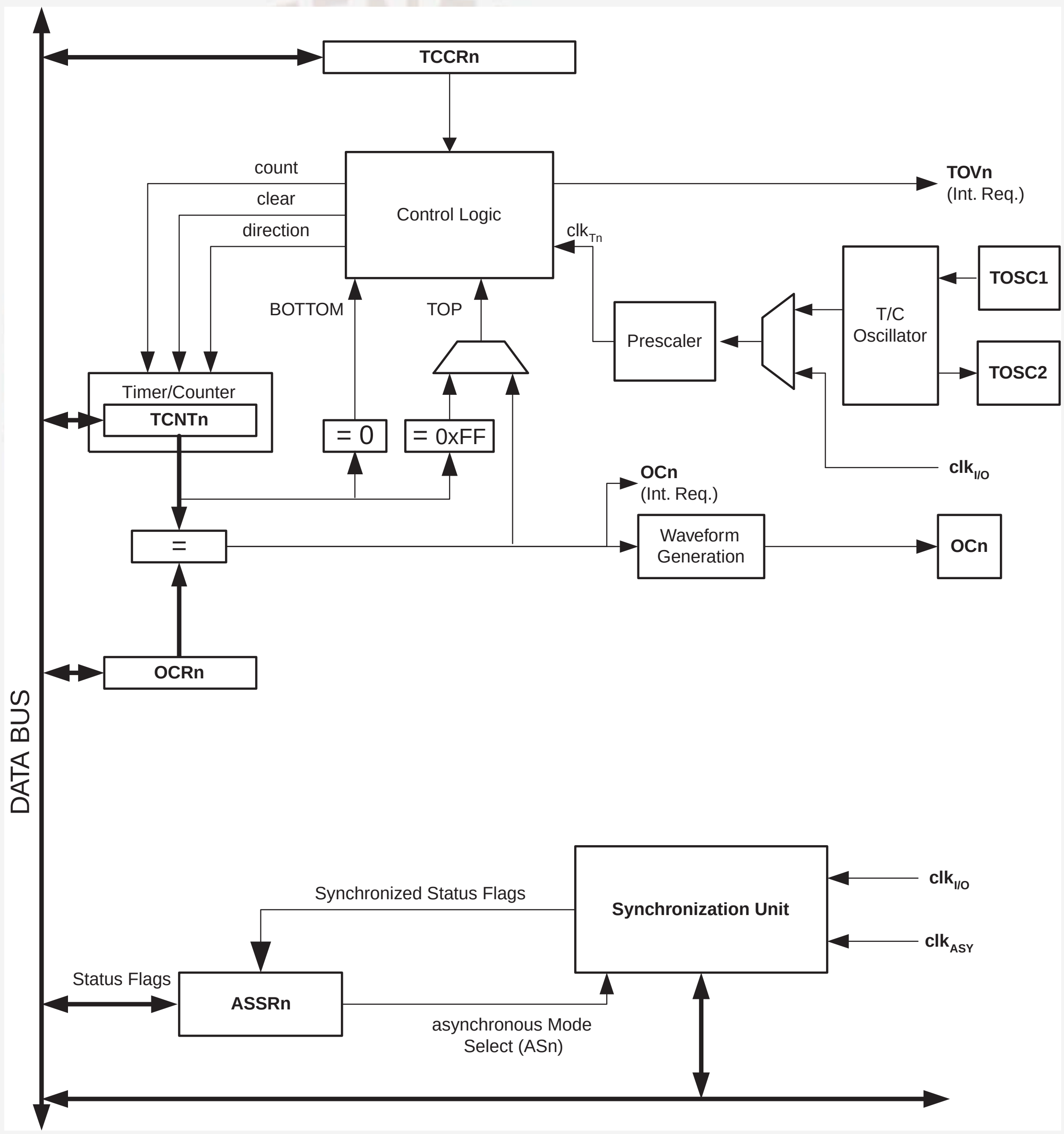
Timer/Counter2 is a general purpose, single channel, 8-bit Timer/Counter module. The main features are:

- **Single Channel 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 32kHz Watch Crystal Independent of the I/O Clock**

Overview

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 45. For the actual placement of I/O pins, refer to "Pin Configurations" 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 114.

Figure 45. 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). For details, see "Output Compare Unit" on page 105. 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 41 are also used extensively throughout the document.

Table 41. 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.

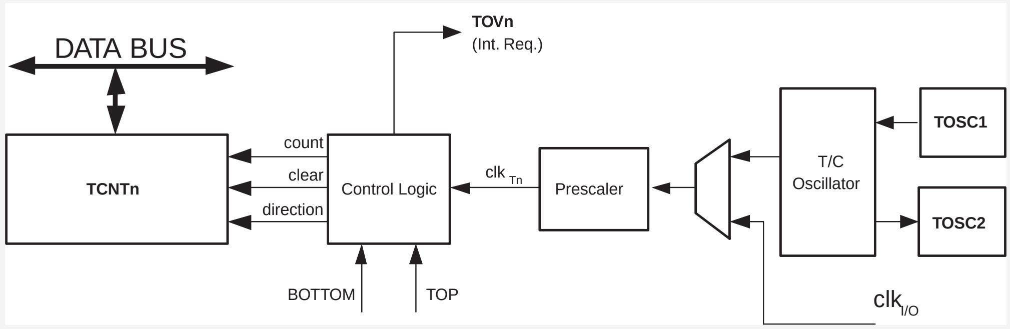
Timer/Counter Clock Sources

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 117. For details on clock sources and prescaler, see "Timer/Counter Prescaler" on page 120.

Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 46 shows a block diagram of the counter and its surrounding environment.

Figure 46. Counter Unit Block Diagram



Signal description (internal signals):

count	Increment or decrement TCNT2 by 1
direction	Selects between increment and decrement
clear	Clear TCNT2 (set all bits to zero)
clk_{T2}	Timer/Counter clock
TOP	Signalizes that TCNT2 has reached maximum value
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 108.

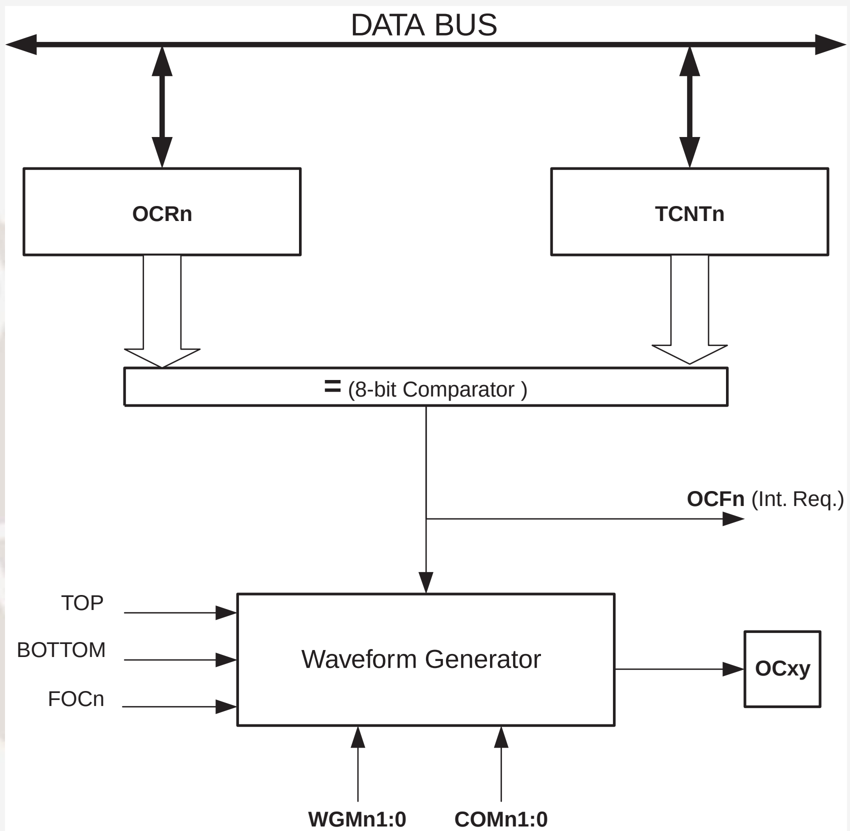
The Timer/Counter Overflow (TOV2) Flag is set according to the mode of operation selected by the WGM21:0 bits. TOV2 can be used for generating a CPU interrupt.

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 (see "Modes of Operation" on page 108).

Figure 47 shows a block diagram of the Output Compare unit.

Figure 47. 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 channel, 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.

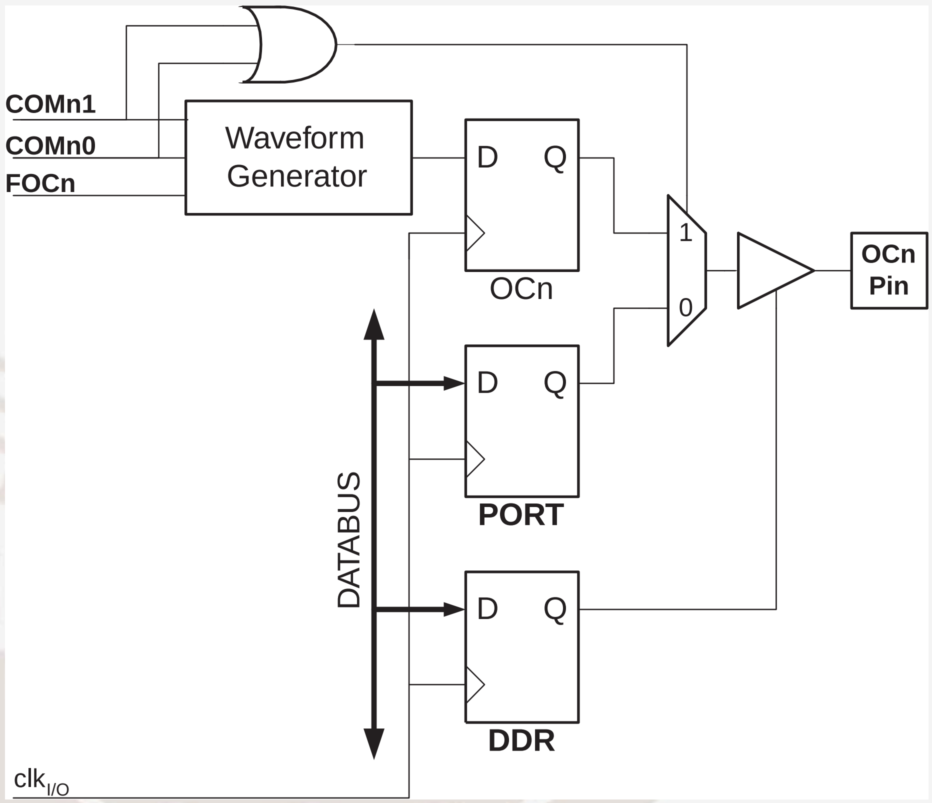
The setup of the OC2 should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC2 value is to use the Force Output Compare (FOC2) strobe bit in Normal mode. The OC2 Register keeps its value even when changing between waveform generation 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 48 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 48. 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 114.

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 43 on page 115. For fast PWM mode, refer to Table 44 on page 115, and for phase correct PWM refer to Table 45 on page 116.

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 107).

For detailed timing information refer to “Timer/Counter Timing Diagrams” on page 112.

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.

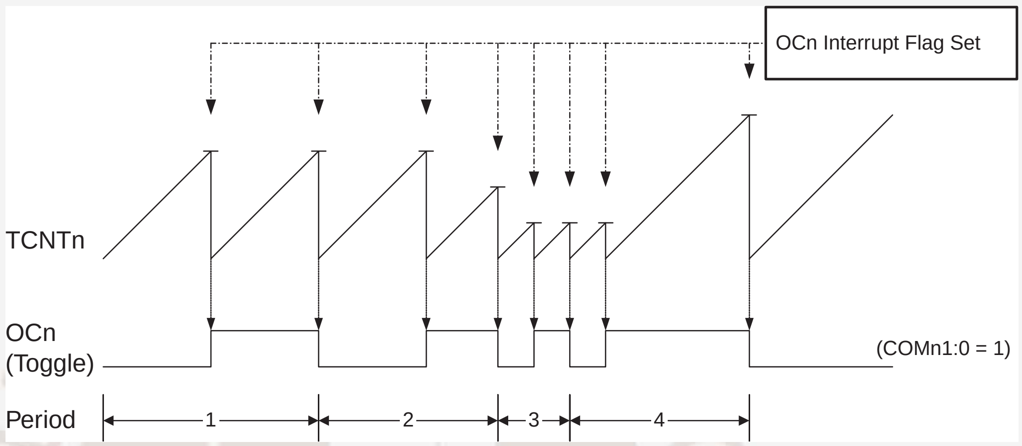
The Output Compare unit can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

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 49. The counter value (TCNT2) increases until a Compare Match occurs between TCNT2 and OCR2, and then counter (TCNT2) is cleared.

Figure 49. 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 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 $f_{OC2} = f_{clk\_I/O}/2$ 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).

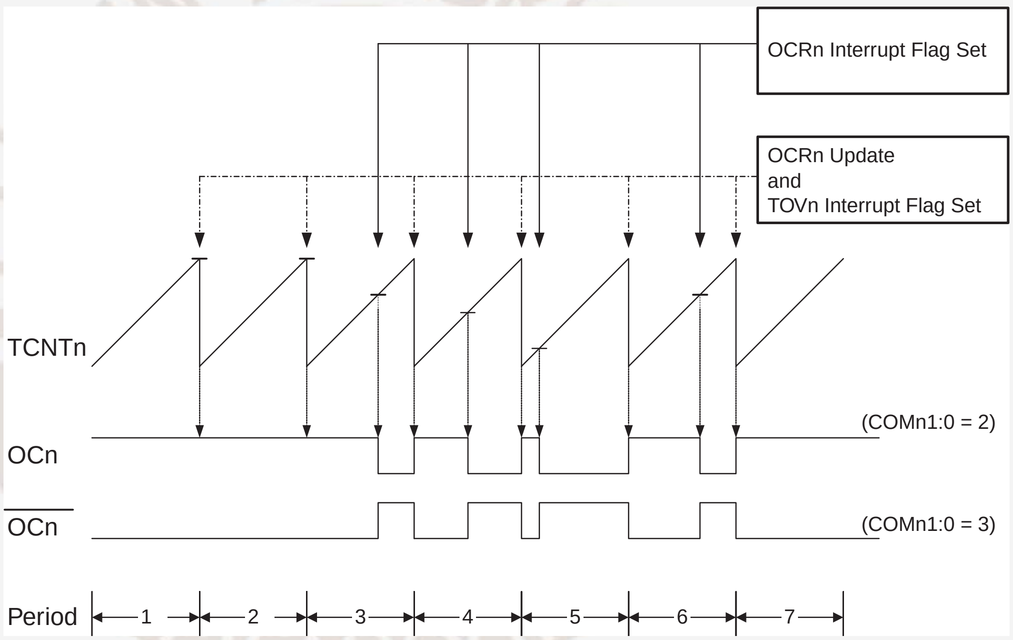
As for the Normal mode of operation, the TOV2 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

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 50. 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 50. 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.

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 44 on page 115). 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.)

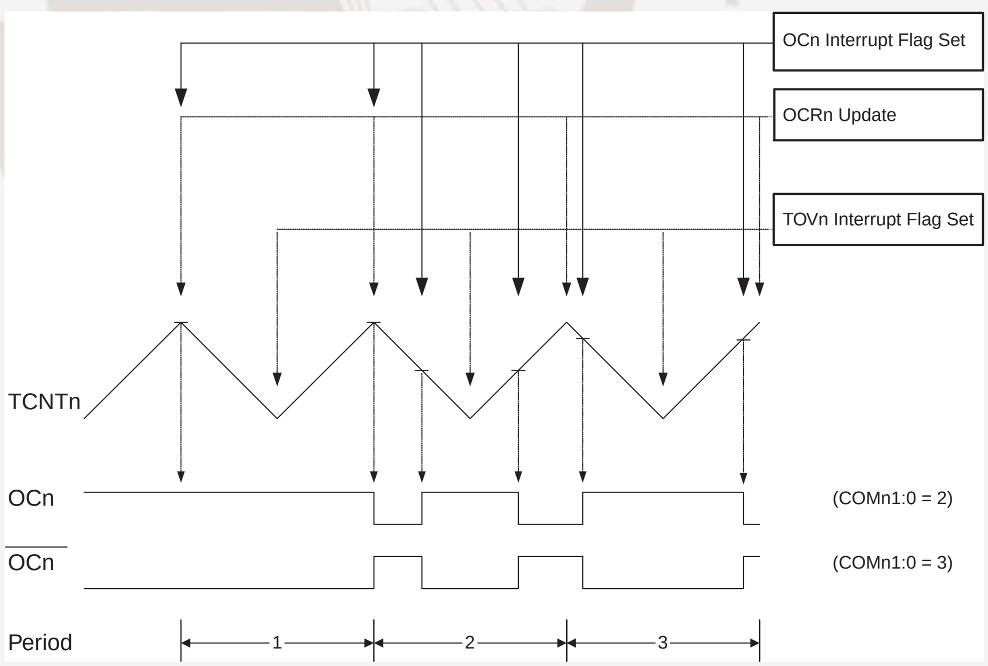
A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC2 to toggle its logical level on each Compare Match (COM21:0 = 1). The waveform generated will have a maximum frequency of $f_{oc2} = f_{clk\_I/O}/2$ when OCR2 is set to zero. This feature is similar to the 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 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 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 51. 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 51. 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 45 on page 116). 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 51 on page 111 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:

- OCR2A changes its value from MAX, like in Figure 51 on page 111. 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 correspond to 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 change 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 52 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 52. Timer/Counter Timing Diagram, no Prescaling

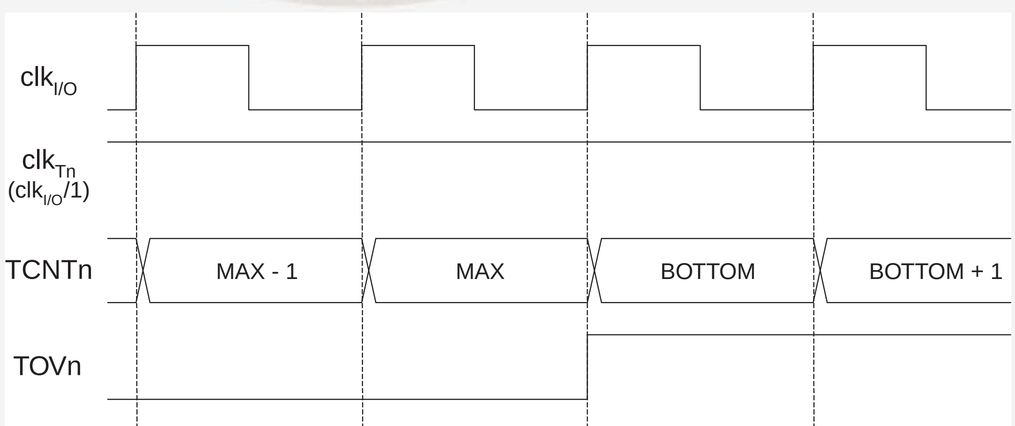


Figure 53 shows the same timing data, but with the prescaler enabled.

Figure 53. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)

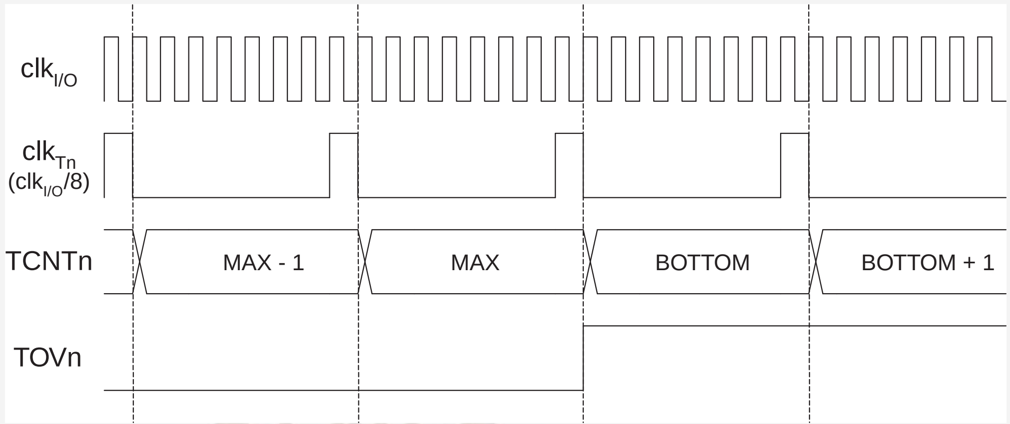


Figure 54 shows the setting of OCF2 in all modes except CTC mode.

Figure 54. Timer/Counter Timing Diagram, Setting of OCF2, with Prescaler ($f_{clk\_I/O}/8$)

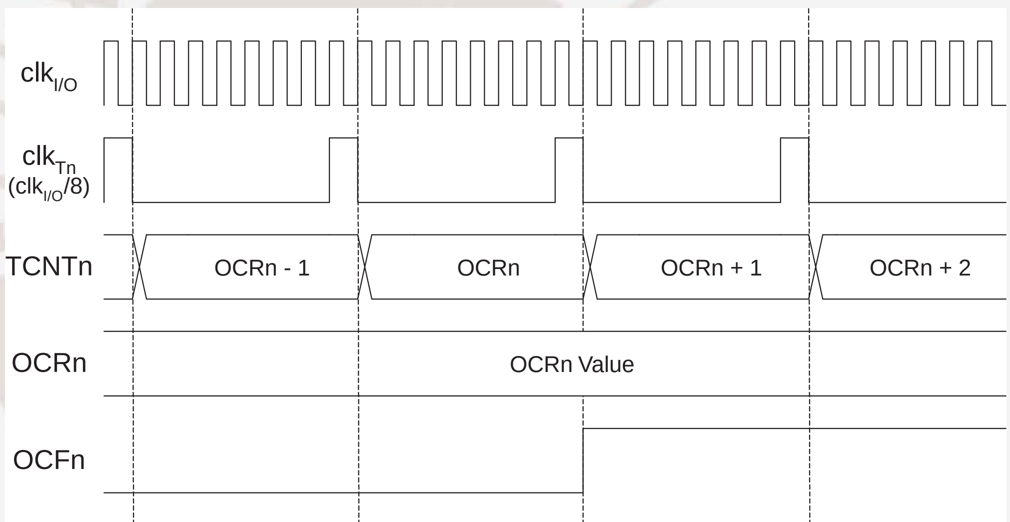
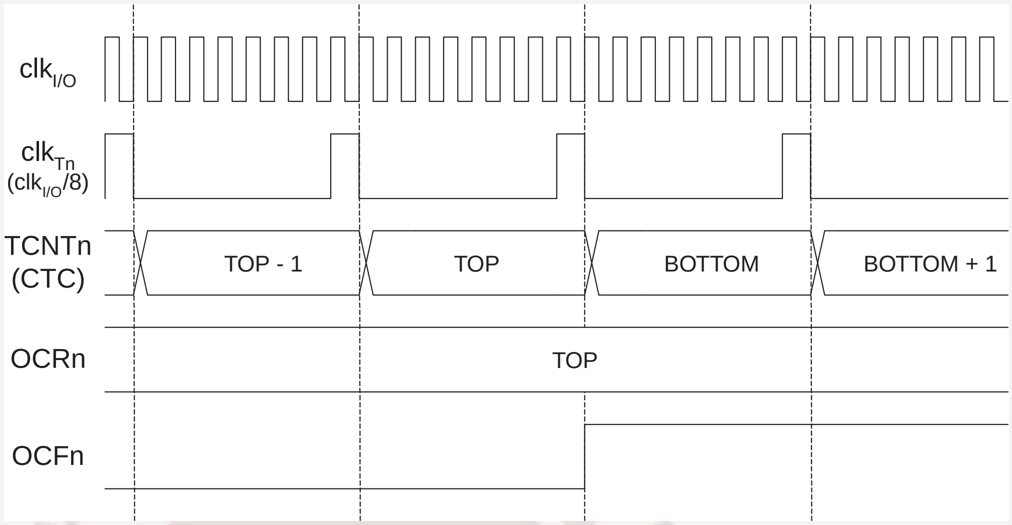


Figure 55 shows the setting of OCF2 and the clearing of TCNT2 in CTC mode.

Figure 55. 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 6:3 – 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 42 on page 115 and “Modes of Operation” on page 108.

Table 42. Waveform Generation Mode Bit Description

Mode	WGM21 (CTC2)	WGM20 (PWM2)	Timer/Counter Mode of Operation[(1)]	TOP	Update of OCR2	TOV2 Flag Set
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR2	Immediate	MAX
3	1	1	Fast PWM	0xFF	BOTTOM	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.

When OC2 is connected to the pin, the function of the COM21:0 bits depends on the WGM21:0 bit setting.

Table 43 shows the COM21:0 bit functionality when the WGM21:0 bits are set to a normal or CTC mode (non-PWM).

Table 43. 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 44 shows the COM21:0 bit functionality when the WGM21:0 bits are set to fast PWM mode.

Table 44. 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 110 for more details

Table 45 shows the COM21:0 bit functionality when the WGM21:0 bits are set to phase correct PWM mode.

Table 45. 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 111 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 46.

Table 46. 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

Bit	7	6	5	4	3	2	1	0	
	TCNT2[7:0]								TCNT2
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	

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

Bit	7	6	5	4	3	2	1	0	
	OCR2[7:0]								OCR2
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	

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	7	6	5	4	3	2	1	0	
	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	ASSR
Read/Write	R	R	R	R	R/W	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **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/Counter 2 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
- The Oscillator is optimized for use with a 32.768kHz 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 that, for example, 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 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 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 re-entering 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 Power-save 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 32.768kHZ 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	–	TOIE0	TIMSK
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 – 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	–	TOV0	TIFR
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 – OCF2: Output Compare Flag 2**

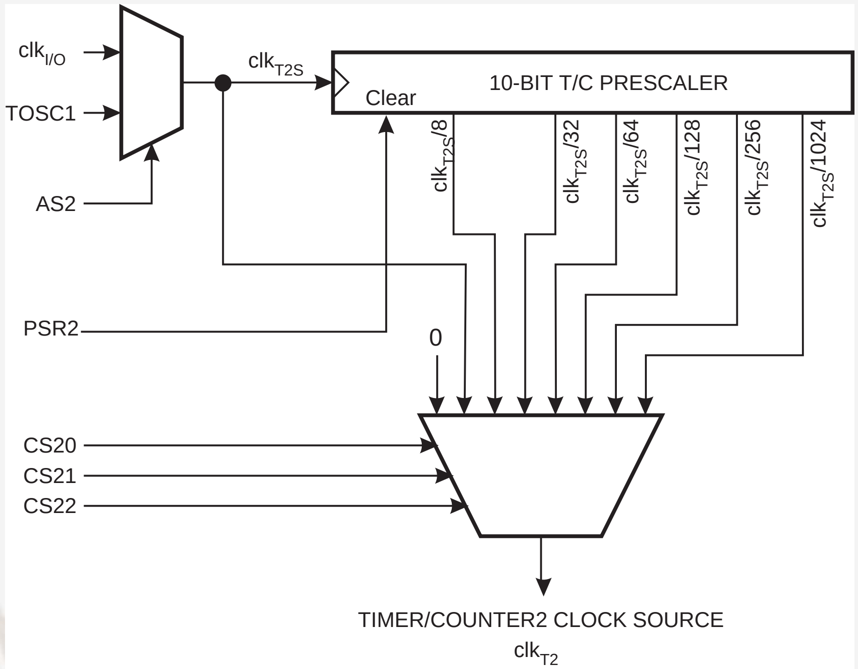
The OCF2 bit is set (one) when a Compare Match occurs between the Timer/Counter2 and the data in OCR2 – Output Compare Register2. OCF2 is cleared by hardware when executing the corresponding interrupt Handling Vector. Alternatively, OCF2 is cleared by writing a logic one to the flag. When the I-bit in 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 0x00.

Timer/Counter Prescaler

Figure 56. 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 $clk_{I/O}$. 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 B. 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 32.768kHz 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	7	6	5	4	3	2	1	0	
	–	–	–	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 1 – PSR2: Prescaler Reset Timer/Counter2**

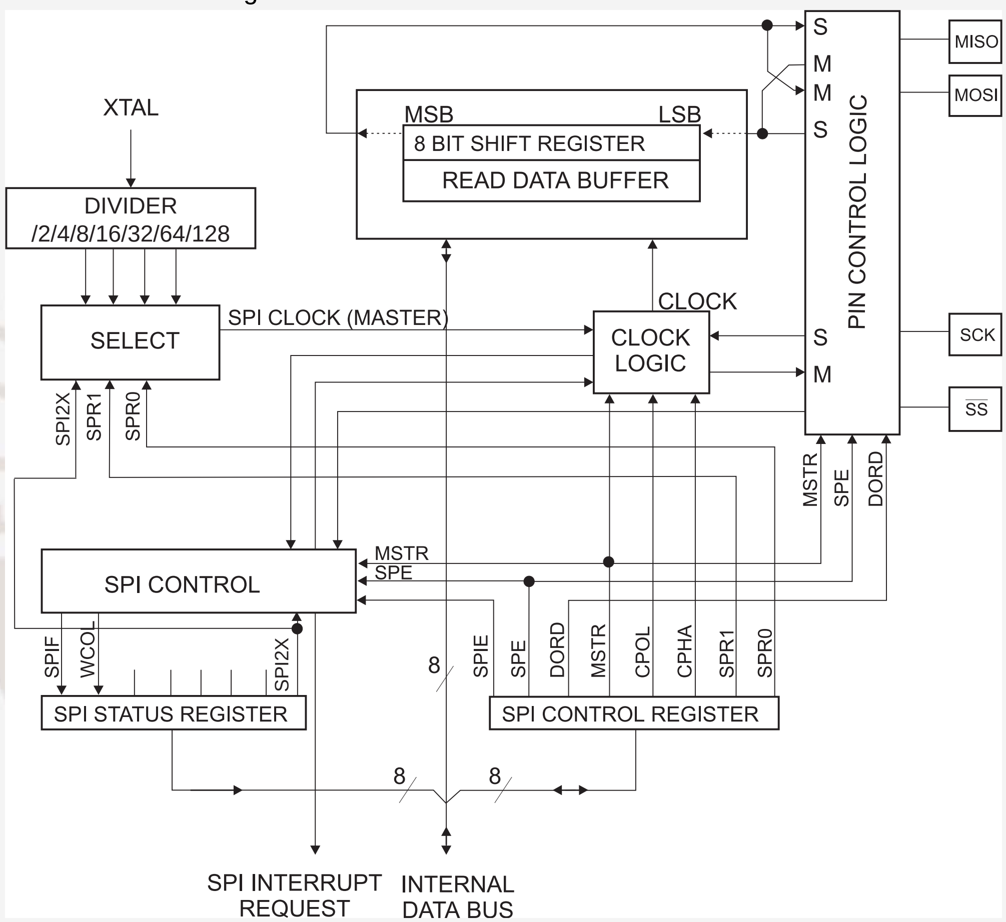
When this bit is written to one, the Timer/Counter2 prescaler will be reset. The bit will be cleared by hardware after the operation is performed. Writing a zero to this bit will have no effect. This bit will always be read as zero if Timer/Counter2 is clocked by the internal CPU clock. If this bit is 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 ATmega8 and peripheral devices or between several AVR devices. The ATmega8 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 57. SPI Block Diagram[(1)]



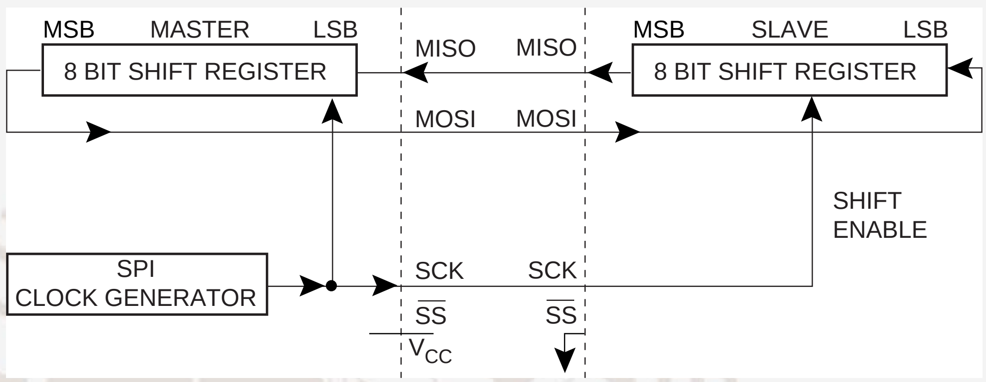
Note: 1. Refer to "Pin Configurations" on page 2, and Table 22 on page 58 for SPI pin placement

The interconnection between Master and Slave CPUs with SPI is shown in Figure 58 on page 122. 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 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, SS, line.

When configured as a Master, the SPI interface has no automatic control of the 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 58. 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 period: longer than 2 CPU clock cycles

High period: 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 47. For more details on automatic port overrides, refer to "Alternate Port Functions" on page 56.

Table 47. SPI Pin Overrides[1]

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
$\overline{SS}$	User Defined	Input

Note: 1. See "Port B Pins Alternate Functions" 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[(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[(1)]

```
void SPI_MasterInit(void)
{
  /* Set MOSI and SCK output, all others input */
  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 8

The following code examples show how to initialize the SPI as a Slave and how to perform a simple reception.

Assembly Code Example[(1)]

```
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[(1)]

```
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)))
    ;
  /* Return data register */
  return SPDR;
}
```

Note: 1. See "About Code Examples" on page 8

$\overline{\text{SS}}$ Pin Functionality

Slave Mode

When the SPI is configured as a Slave, the Slave Select ($\overline{\text{SS}}$) pin is always input. When $\overline{\text{SS}}$ is held low, the SPI is activated, and MISO becomes an output if configured so by the user. All other pins are inputs. When $\overline{\text{SS}}$ 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 $\overline{\text{SS}}$ pin is driven high.

The $\overline{\text{SS}}$ pin is useful for packet/byte synchronization to keep the Slave bit counter synchronous with the master clock generator. When the $\overline{\text{SS}}$ 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 $\overline{\text{SS}}$ pin.

If $\overline{\text{SS}}$ 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 $\overline{\text{SS}}$ pin of the SPI Slave.

If $\overline{\text{SS}}$ is configured as an input, it must be held high to ensure Master SPI operation. If the $\overline{\text{SS}}$ pin is driven low by peripheral circuitry when the SPI is configured as a Master with the $\overline{\text{SS}}$ 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:

1. 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 $\overline{\text{SS}}$ 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	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
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 – 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.

- **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 59 on page 128 and Figure 60 on page 128 for an example. The CPOL functionality is summarized below:

Table 48. 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 59 on page 128 and Figure 60 on page 128 for an example. The CPHA functionality is summarized below:

Table 49. CPHA Functionality

CPHA	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_{osc} is shown in the following table:

Table 50. 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	$f_{osc}/64$

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 ATmega8 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 50 on page 126). This means that the minimum SCK period will be 2 CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at $f_{osc}/4$ or lower.

The SPI interface on the ATmega8 is also used for Program memory and EEPROM downloading or uploading. See page 230 for Serial Programming and verification.

SPI Data Register – SPDR

Bit	7	6	5	4	3	2	1	0	
	MSB							**LSB**	**SPDR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

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 59 on page 128 and Figure 60 on page 128. 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 48 on page 126 and Table 49 on page 126, as done below:

Table 51. 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 59. SPI Transfer Format with CPHA = 0

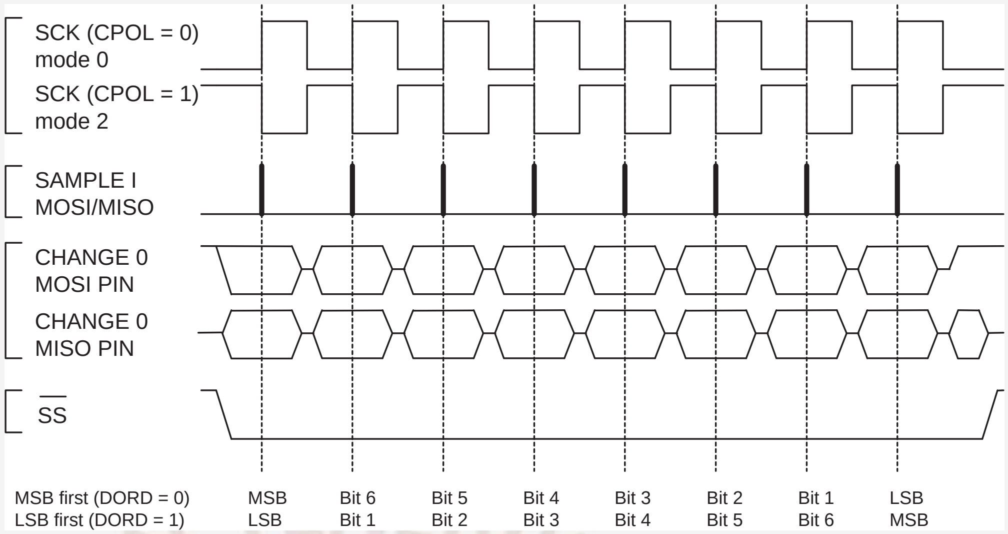
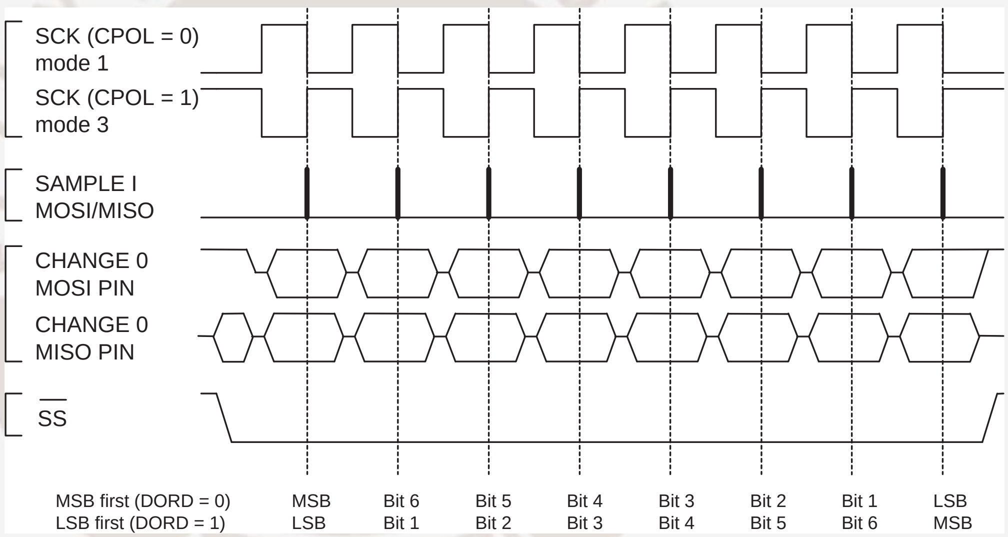


Figure 60. SPI Transfer Format with CPHA = 1



USART

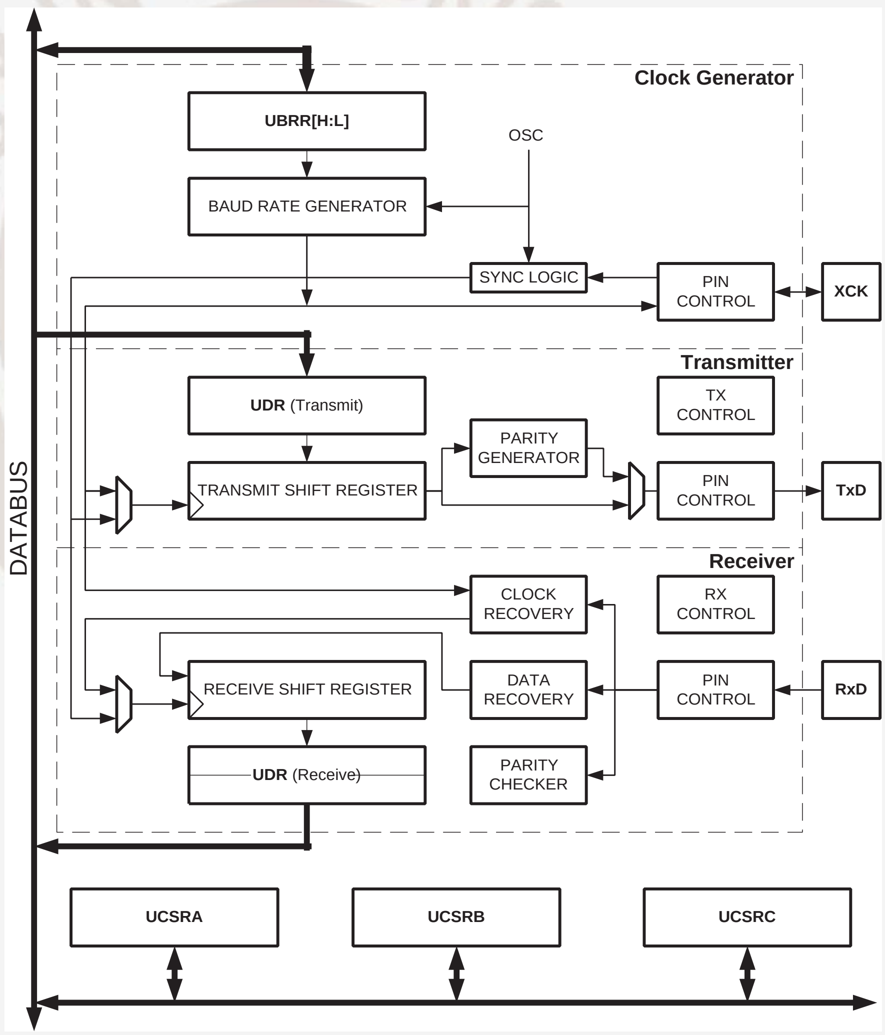
The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a highly-flexible serial communication device. The main features are:

- **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 Databits 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 61. CPU accessible I/O Registers and I/O pins are shown in bold.

Figure 61. USART Block Diagram[(1)]



Note: 1. Refer to "Pin Configurations" on page 2, Table 30 on page 64, and Table 29 on page 64 for USART pin placement

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 ninth 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 61 on page 129) 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:

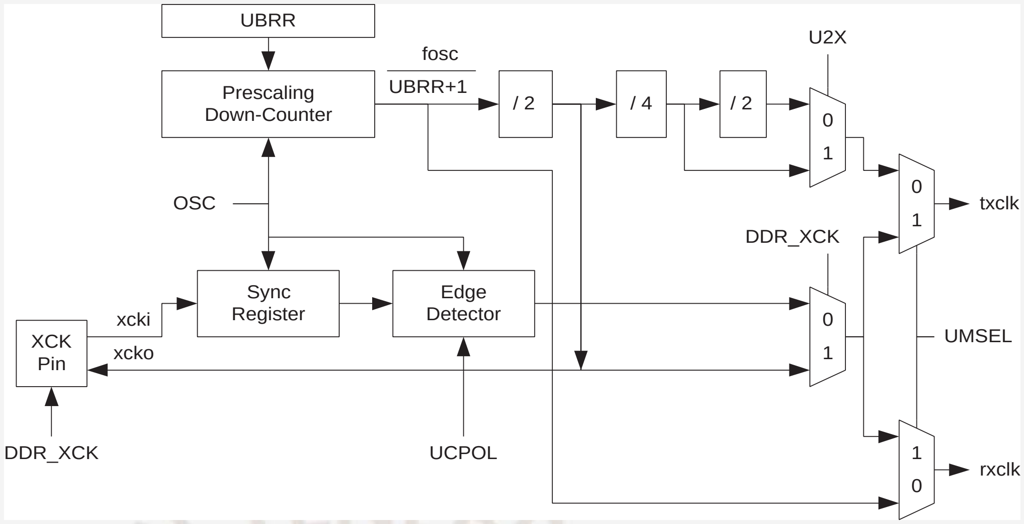
- CHR9 is changed to UCSZ2
- OR is changed to DOR

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 62 on page 131 shows a block diagram of the clock generation logic.

Figure 62. 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 62.

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 52 on page 132 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 52. 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)	$BAUD = \frac{f_{OSC}}{16(\mathbf{UBRR} + 1)}$	$\mathbf{UBRR} = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed Mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(\mathbf{UBRR} + 1)}$	$\mathbf{UBRR} = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master Mode	$BAUD = \frac{f_{OSC}}{2(\mathbf{UBRR} + 1)}$	$\mathbf{UBRR} = \frac{f_{OSC}}{2BAUD} - 1$

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)

fOSC 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 60 on page 153.

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 62 on page 131 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:

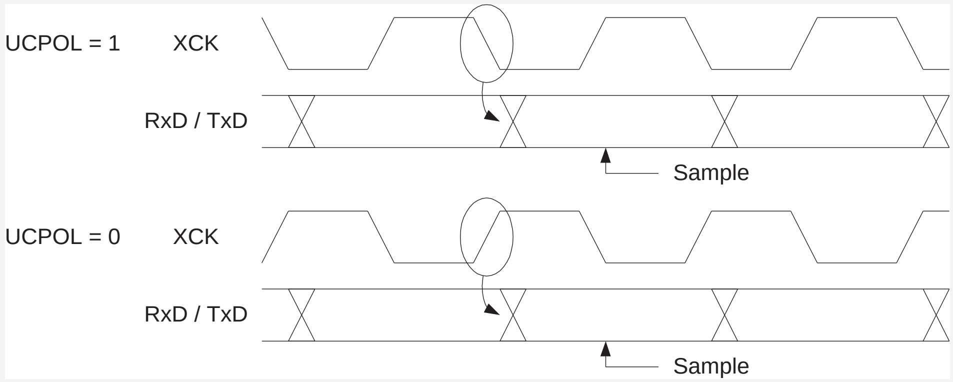
$$f_{XCK} < \frac{f_{OSC}}{4}$$

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 63. 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 63 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 64 illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

Figure 64. 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
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:

$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$
$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

P_{even} Parity bit using even parity

P_{odd} 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[1]

```
#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 8

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 Bits

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 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 8

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 Bits

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
  rjmp USART_Transmit
  ; Copy ninth 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[1]

```
void USART_Transmit( unsigned int data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE)) )
        ;
  /* Copy ninth 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.

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
  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 8

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 Data Bits

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[(1)]

```
USART_Receive:
  ; Wait for data to be received
  sbis UCSRA, RXC
  rjmp USART_Receive
  ; Get status and ninth 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 ninth bit, then return
  lsr  r17
  andi r17, 0x01
  ret
```

C Code Example[(1)]

```
unsigned int USART_Receive( void )
{
  unsigned char status, resh, resl;
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) )
        ;
  /* Get status and ninth bit, then data */
  /* from buffer */
  status = UCSRA;
  resh = UCSRB;
  resl = UDR;
  /* If error, return -1 */
  if ( status & (1<<FE)|(1<<DOR)|(1<<PE) )
    return -1;
  /* Filter the ninth bit, then return */
  resh = (resh >> 1) & 0x01;
  return ((resh << 8) | resl);
}
```

Note: 1. See "About Code Examples" on page 8

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 134 and "Parity Checker" .

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.

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 8

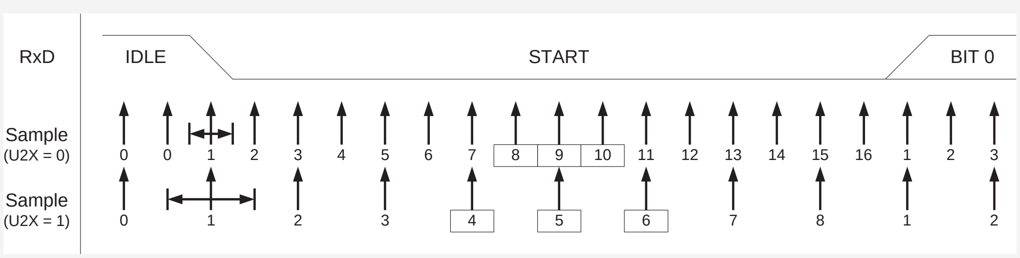
Asynchronous Data Reception

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 65 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 eight 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 65. Start Bit Sampling

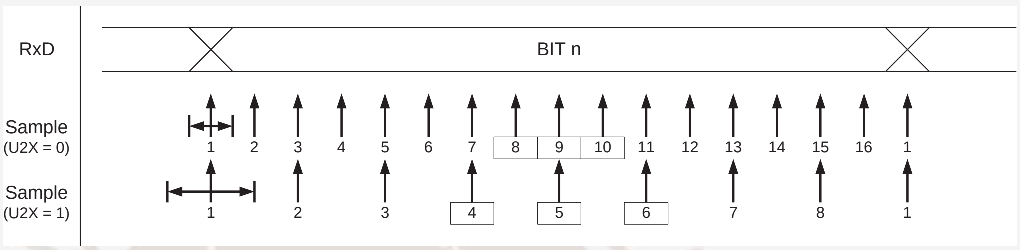


When the clock recovery logic detects a high (idle) to low (start) transition on the RxD line, the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample as shown in the figure. The clock recovery logic then uses samples 8, 9 and 10 for Normal mode, and samples 4, 5 and 6 for Double Speed mode (indicated with sample numbers inside boxes on the 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 eight states for each bit in Double Speed mode. Figure 66 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.

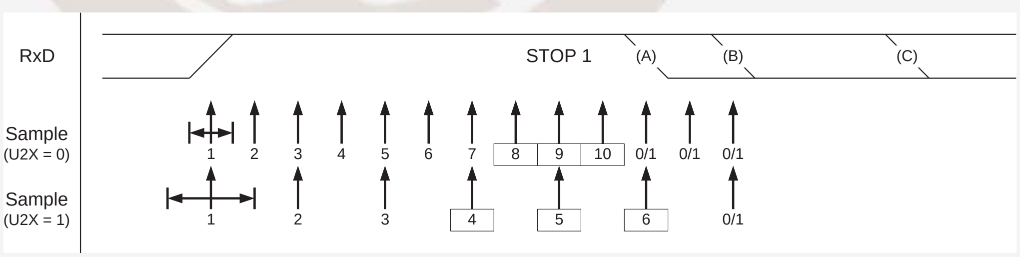
Figure 66. Sampling of Data and Parity Bit



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 67 shows the sampling of the stop bit and the earliest possible beginning of the start bit of the next frame.

Figure 67. 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 67. 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.

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 53) 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.

$$R_{slow} = \frac{(D+1)S}{S-1+D\cdot S+S_F} \qquad R_{fast} = \frac{(D+2)S}{(D+1)S+S_M}$$

D Sum of character size and parity size (D = 5-bit 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 53 and Table 54 list the maximum Receiver baud rate error that can be tolerated. Note that Normal Speed mode has higher toleration of baud rate variations.

Table 53. 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	±2.0
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 54. 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
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)
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
3. 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
4. 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
5. 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 Examples[(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 Examples[(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 8

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 8

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

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDR (Read)
	TXB[7:0]								UDR (Write)
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	

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.

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 “Bit 7 – RXCIE: RX Complete Interrupt Enable” on page 149).

- **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 “Bit 6 – TXCIE: TX Complete Interrupt Enable” on page 149).

- **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 “Bit 5 – UDRIE: USART Data Register Empty Interrupt Enable” on page 149).

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 145.

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	
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 146 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 55. 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 56. 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 57. 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 58. 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 59. 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 UBRRHs

Bit	15	14	13	12	11	10	9	8	
	URSEL	–	–	–	UBRR[11:8]				UBRRH
	UBRR[7:0]								UBRRL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R	R	R	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	

The UBRRH Register shares the same I/O location as the UCSRC Register. See the "Accessing UBRRH/UCSRC Registers" on page 146 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.

- **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 eight 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 60. 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 144). The error values are calculated using the following equation:

$$\text{Error}[\%] = \left(\frac{\text{BaudRate}_{\text{Closest Match}}}{\text{BaudRate}} - 1\right) \bullet 100\%$$

Table 60. Examples of UBRR Settings for Commonly Used Oscillator Frequencies

Baud Rate (bps)	f_{osc} = 1.0000MHz				f_{osc} = 1.8432MHz				f_{osc} = 2.0000MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	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	–	–	–	–	–	–	**0**	**0.0%**	–	-	–	–
250k	–	–	–	–	–	–	–	–	–	–	**0**	**0.0%**
Max [(1)]	62.5kbps		125kbps		115.2kbps		230.4kbps		125kbps		250kbps	

1. UBRR = 0, Error = 0.0%

Table 61. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	f_{osc} = 3.6864MHz				f_{osc} = 4.0000MHz				f_{osc} = 7.3728MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	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]	230.4kbps		460.8kbps		250kbps		0.5Mbps		460.8kbps		921.6kbps	

1. UBRR = 0, Error = 0.0%

Table 62. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	f_{osc} = 8.0000MHz				f_{osc} = 11.0592MHz				f_{osc} = 14.7456MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	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)]	0.5Mbps		1Mbps		691.2kbps		1.3824Mbps		921.6kbps		1.8432Mbps	

1. UBRR = 0, Error = 0.0%

Table 63. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	f_{osc} = 16.0000MHz				f_{osc} = 18.4320MHz				f_{osc} = 20.0000MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	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%	–	–	–	–	–	–	–	–
Max [1]	1Mbps		2Mbps		1.152Mbps		2.304Mbps		1.25Mbps		2.5Mbps	

1. UBRR = 0, Error = 0.0%

Two-wire Serial Interface

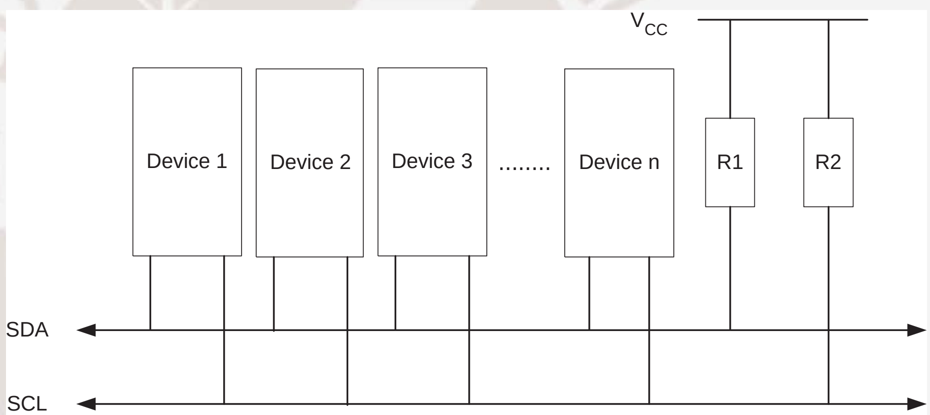
Features

- **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 400kHz 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 68. TWI Bus Interconnection



TWI Terminology

The following definitions are frequently encountered in this section.

Table 64. 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 68 on page 157, 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.

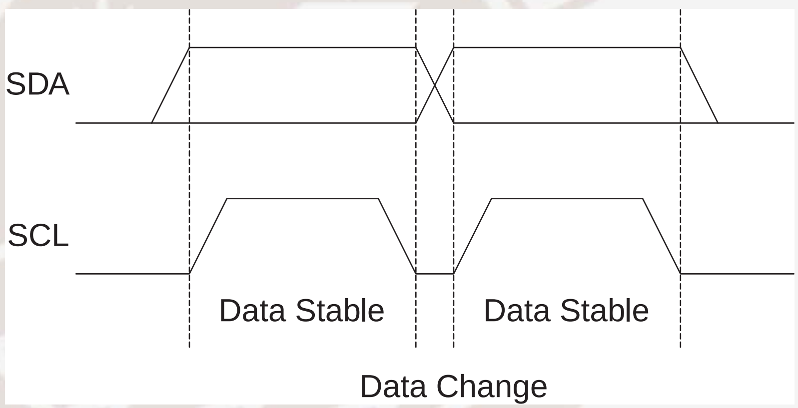
The number of devices that can be connected to the bus is only limited by the bus capacitance limit of 400pF and the 7-bit slave address space. A detailed specification of the electrical characteristics of the TWI is given in "Two-wire Serial Interface Characteristics" on page 238. Two different sets of specifications are presented there, one relevant for bus speeds below 100kHz, and one valid for bus speeds up to 400kHz.

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.

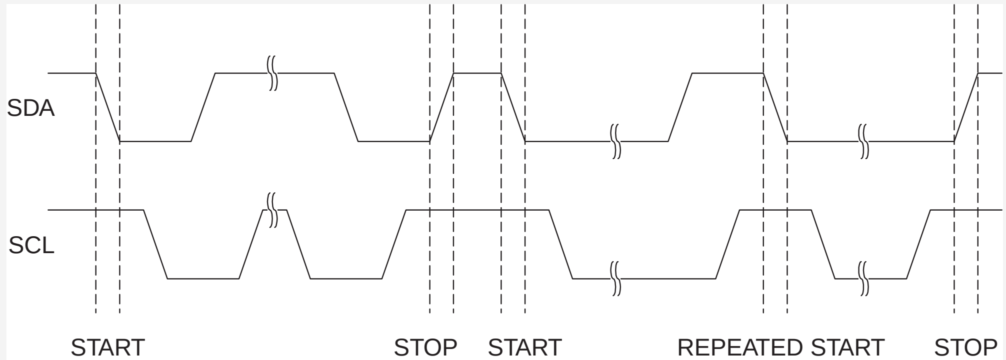
Figure 69. Data Validity



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 relinquishing 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 70. START, REPEATED START and STOP conditions



Address Packet Format

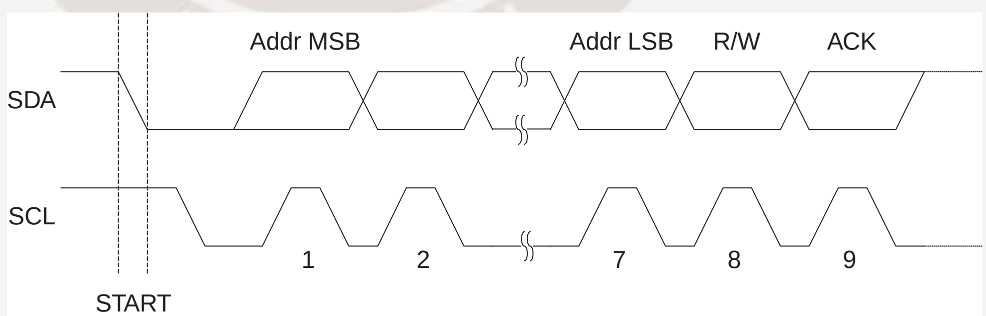
All address packets transmitted on the TWI bus are 9 bits long, consisting of 7 address bits, one READ/WRITE control bit and an acknowledge bit. If the READ/WRITE bit is set, a read operation is to be performed, otherwise a write operation should be performed. When a Slave recognizes that it is being 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.

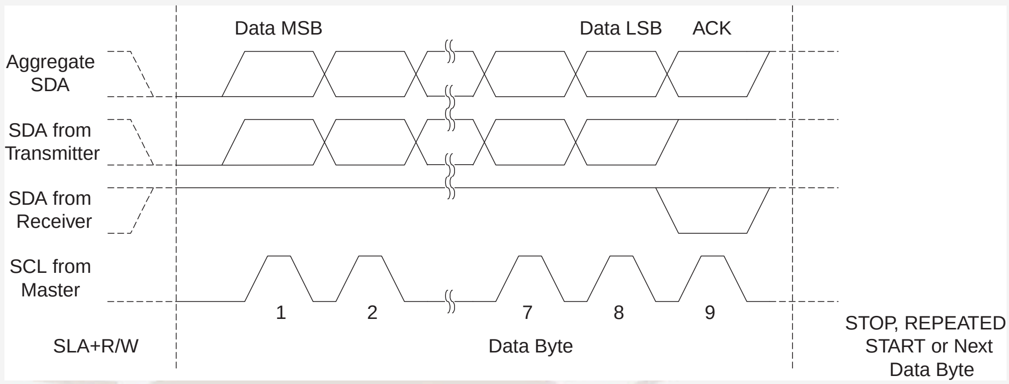
Figure 71. Address Packet Format



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 72. Data Packet Format

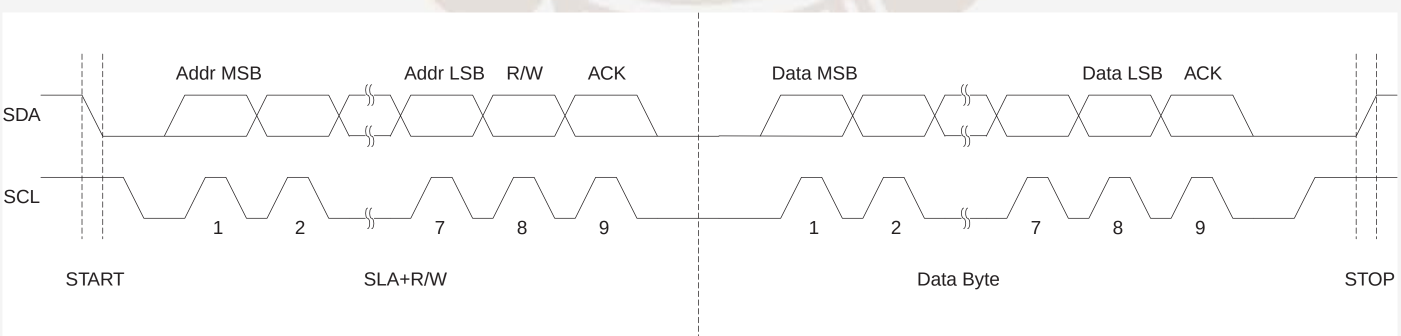


Combining Address and Data Packets into a Transmission

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 73 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 73. Typical Data Transmission



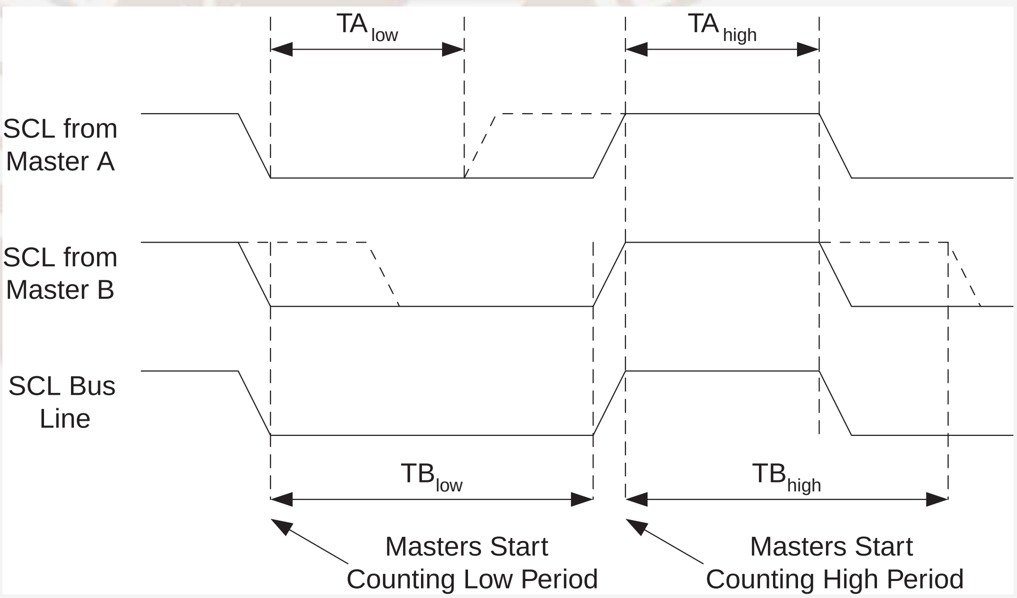
Multi-master Bus Systems, Arbitration and Synchronization

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

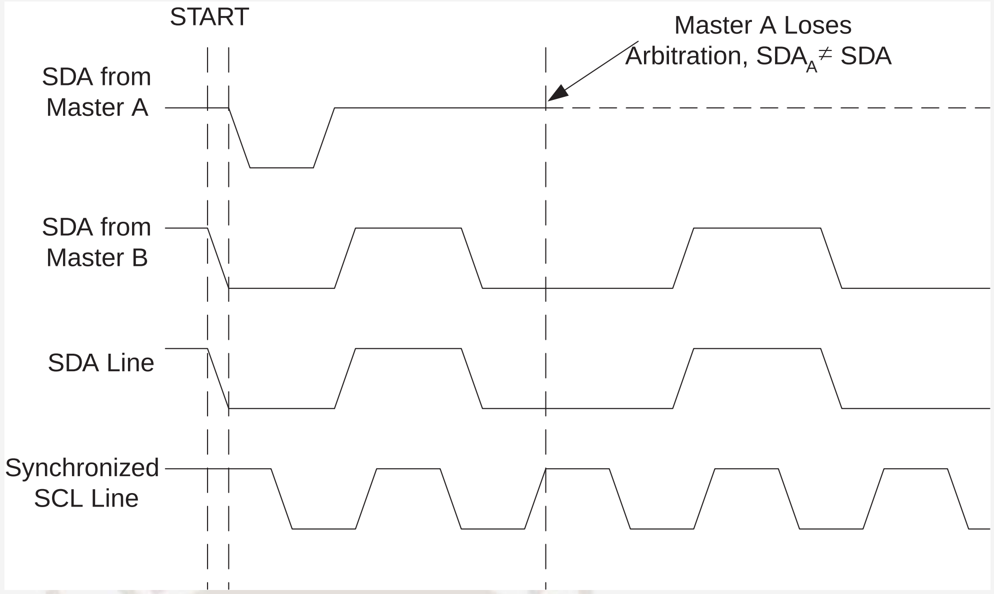
The wired-ANDing of the bus lines is used to solve both these problems. The serial clocks from all masters will be wired-ANDed, yielding a combined clock with a high period equal to the one from the Master with the shortest high period. The low period of the combined clock is equal to the low period 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 74. 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 75. Arbitration Between Two Masters



Note that arbitration is not allowed between:

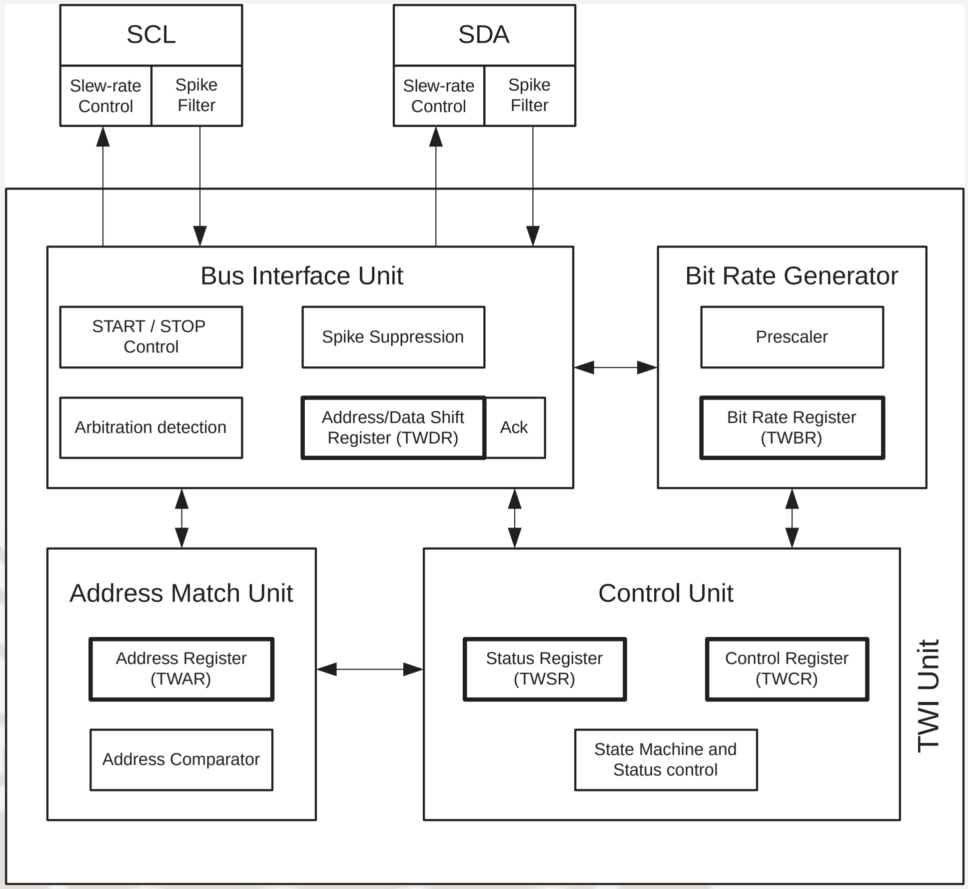
- A REPEATED START condition and a data bit
- A STOP condition and a data bit
- A REPEATED START and a STOP condition

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 76. All registers drawn in a thick line are accessible through the AVR data bus.

Figure 76. 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 50ns. 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:

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot 4^{TWPS}}$$

- TWBR = Value of the TWI Bit Rate Register
- TWPS = Value of the prescaler bits in the TWI Status Register

Note: Pull-up resistor values should be selected according to the SCL frequency and the capacitive bus line load. See Table 101 on page 238 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, 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 seven-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. If another interrupt (for example, INT0) occurs during TWI Power-down address match and wakes up the CPU, the TWI aborts operation and return to it’s idle state. If this cause any problems, ensure that TWI Address Match is the only enabled interrupt when entering Power-down.

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	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..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 164 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 Two-wire 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 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 5 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 65. 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 164. 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	R/W	R/W	R/W	R/W	R/W	R/W	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 constitute 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	R/W	R/W	R/W	R/W	R/W	R/W	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, and not needed in the Master modes. In multimaster 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 (0x00). 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.

- **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.

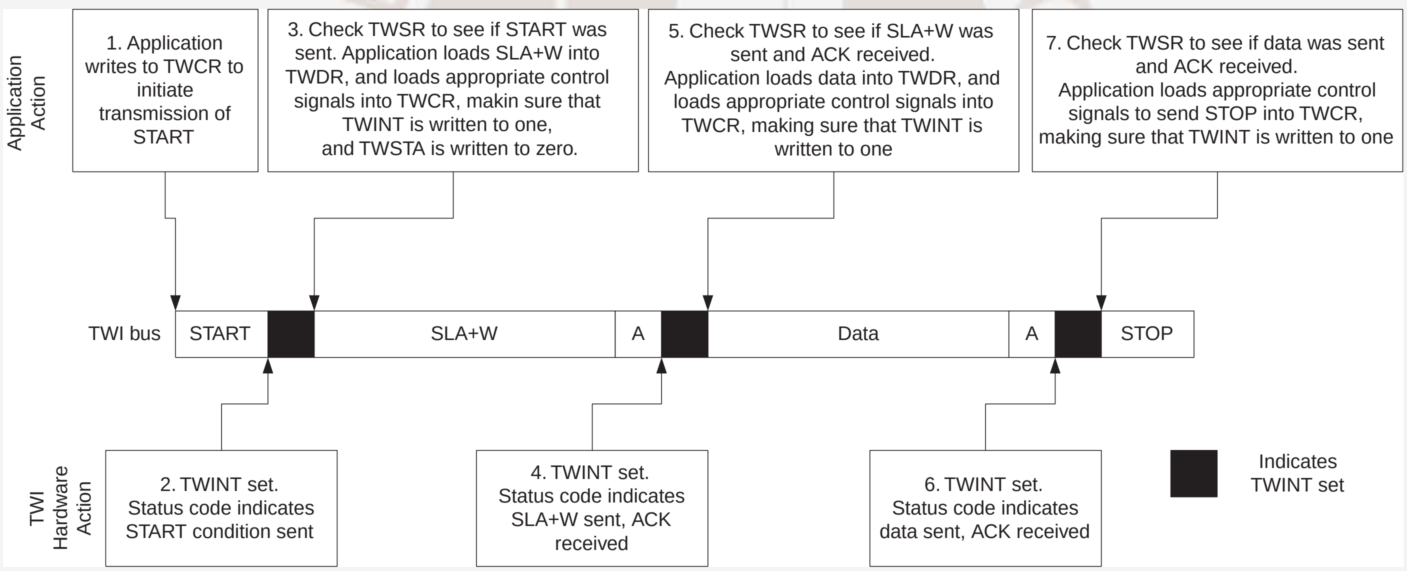
Using the TWI

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 77 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 77. 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
2. 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 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.

<table>
<tr><th></th><th>Assembly Code Example</th><th>C Example</th><th>Comments</th></tr>
<tr><td>1</td><td>ldi r16, (1<<TWINT)|(1<<TWSTA)|
(1<<TWEN)
out TWCR, r16</td><td>TWCR = (1<<TWINT)|(1<<TWSTA)|
(1<<TWEN)</td><td>Send START condition</td></tr>
<tr><td>2</td><td>wait1:
in r16,TWCR
sbrs r16,TWINT
rjmp wait1</td><td>while (!(TWCR & (1<<TWINT)))
;</td><td>Wait for TWINT Flag set. This indicates that the START condition has been transmitted</td></tr>
<tr><td rowspan="2">3</td><td>in r16,TWSR
andi r16, 0xF8
cpi r16, START
brne ERROR</td><td>if ((TWSR & 0xF8) != START)
ERROR();</td><td>Check value of TWI Status Register. Mask prescaler bits. If status different from START go to ERROR</td></tr>
<tr><td>ldi r16, SLA_W
out TWDR, r16
ldi r16, (1<<TWINT) | (1<<TWEN)
out TWCR, r16</td><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></tr>
<tr><td>4</td><td>wait2:
in r16,TWCR
sbrs r16,TWINT
rjmp wait2</td><td>while (!(TWCR & (1<<TWINT)))
;</td><td>Wait for TWINT Flag set. This indicates that the SLA+W has been transmitted, and ACK/NACK has been received.</td></tr>
<tr><td rowspan="2">5</td><td>in r16,TWSR
andi r16, 0xF8
cpi r16, MT_SLA_ACK
brne ERROR</td><td>if ((TWSR & 0xF8) !=
MT_SLA_ACK)
ERROR();</td><td>Check value of TWI Status Register. Mask prescaler bits. If status different from MT_SLA_ACK go to ERROR</td></tr>
<tr><td>ldi r16, DATA
out TWDR, r16
ldi r16, (1<<TWINT) | (1<<TWEN)
out TWCR, r16</td><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></tr>
<tr><td>6</td><td>wait3:
in r16,TWCR
sbrs r16,TWINT
rjmp wait3</td><td>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></tr>
<tr><td rowspan="2">7</td><td>in r16,TWSR
andi r16, 0xF8
cpi r16, MT_DATA_ACK
brne ERROR</td><td>if ((TWSR & 0xF8) !=
MT_DATA_ACK)
ERROR();</td><td>Check value of TWI Status Register. Mask prescaler bits. If status different from MT_DATA_ACK go to ERROR</td></tr>
<tr><td>ldi r16, (1<<TWINT)|(1<<TWEN)|
(1<<TWSTO)
out TWCR, r16</td><td>TWCR = (1<<TWINT)|(1<<TWEN)|
(1<<TWSTO);</td><td>Transmit STOP condition</td></tr>
</table>

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)

$\overline{\text{A}}$: Not acknowledge bit (high level at SDA)

Data: 8-bit data byte

P: STOP condition

SLA: Slave Address

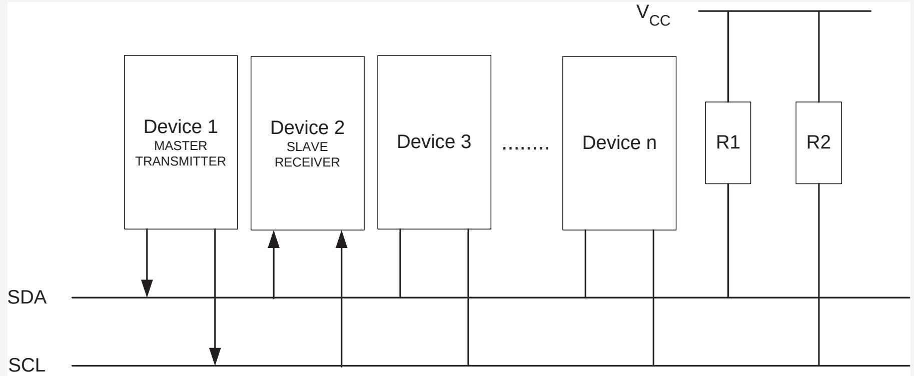
In Figure 79 on page 174 to Figure 85 on page 183, circles are used to indicate that the TWINT Flag is set. The numbers in the circles show the status code held in TWSR, with the prescaler bits masked to zero. At these points, actions must be taken by the application to continue or complete the TWI transfer. 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 66 on page 173 to Table 69 on page 182. 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 78 on page 172). 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 78. Data Transfer in Master Transmitter Mode



A START condition is sent by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	1	0	X	1	0	X

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 0x08 (see Table 66 on page 173). 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	X	0	0	X	1	0	X

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 0x18, 0x20, or 0x38. The appropriate action to be taken for each of these status codes is detailed in Table 66 on page 173.

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	X	0	0	X	1	0	X

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	X

A REPEATED START condition is generated by writing the following value to TWCR:

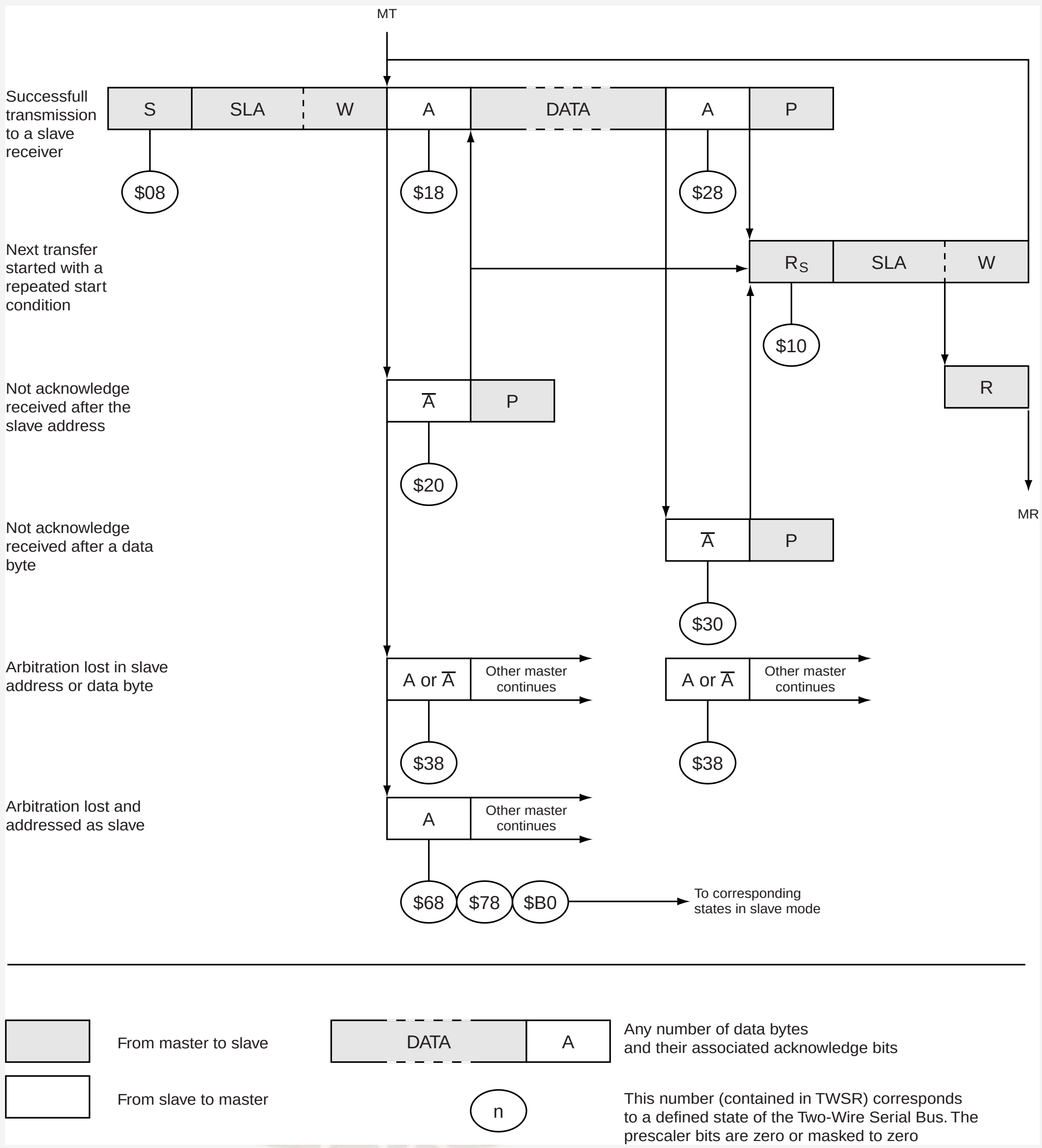
TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	1	0	X	1	0	X

After a repeated START condition (state 0x10) 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 66. Status codes for Master Transmitter Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x08	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
0x10	A repeated START condition has been transmitted	Load SLA+W or	0	0	1	X	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
0x18	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	X	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
0x20	SLA+W 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
		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	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x28	Data byte 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	X	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
0x30	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
		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	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x38	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	X	A START condition will be transmitted when the bus becomes free

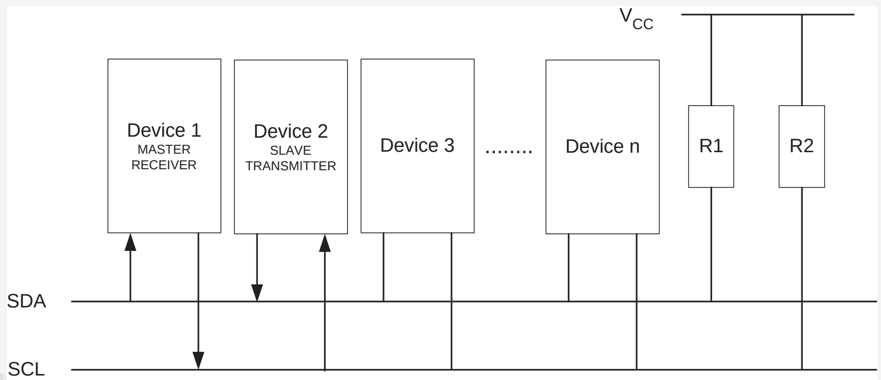
Figure 79. Formats and States in the Master Transmitter Mode



Master Receiver Mode

In the Master Receiver mode, a number of data bytes are received from a Slave Transmitter (see Figure 80). 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 80. 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	X	1	0	X	1	0	X

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 0x08 (see Table 66 on page 173). 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	X	0	0	X	1	0	X

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 0x38, 0x40, or 0x48. The appropriate action to be taken for each of these status codes is detailed in Table 67 on page 176. 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	X	0	1	X	1	0	X

A REPEATED START condition is generated by writing the following value to TWCR:

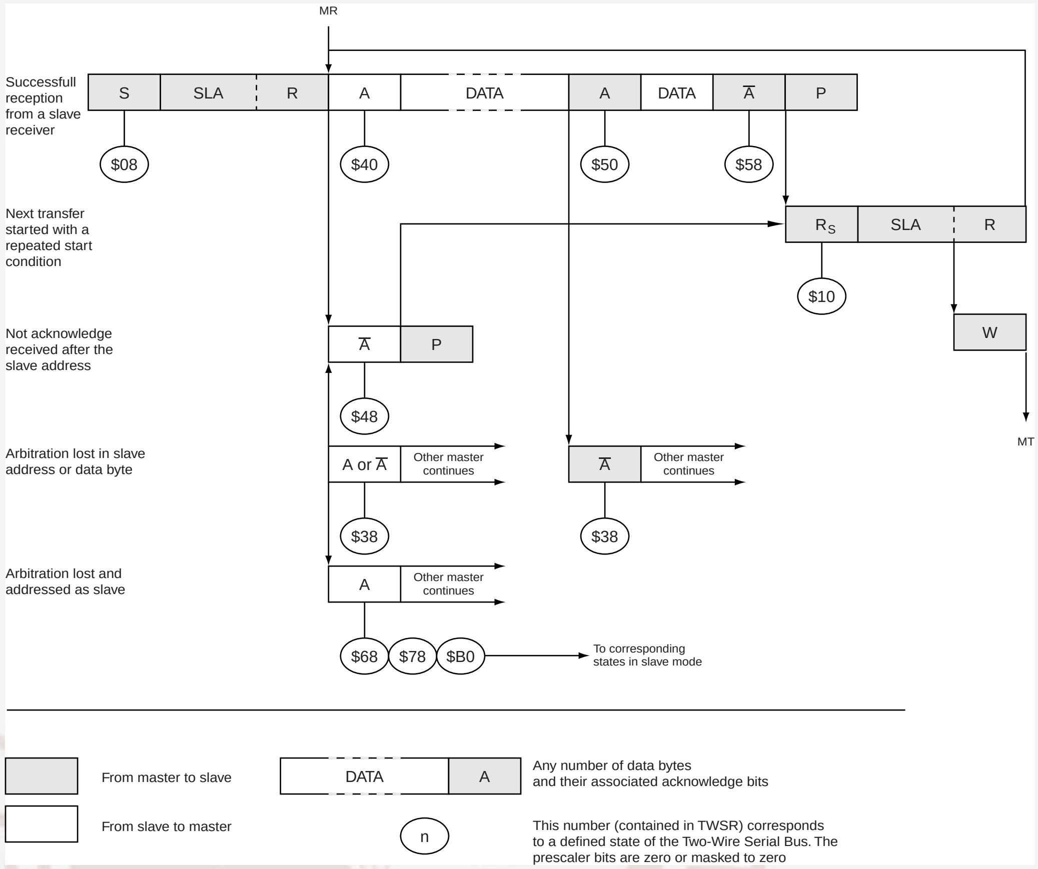
TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	1	0	X	1	0	X

After a repeated START condition (state 0x10) 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 67. Status codes for Master Receiver Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x08	A START condition has been transmitted	Load SLA+R	0	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+R or Load SLA+W	0 0	0 0	1 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
0x38	Arbitration lost in SLA+R or NOT ACK bit	No TWDR action or No TWDR action	0 1	0 0	1 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
0x40	SLA+R has been transmitted; ACK has been received	No TWDR action or No TWDR action	0 0	0 0	1 1	0 1	Data byte will be received and NOT ACK will be returned Data byte will be received and ACK will be returned
0x48	SLA+R has been transmitted; NOT ACK has been received	No TWDR action or No TWDR action or No TWDR action	1 0 1	0 1 1	1 1 1	X 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
0x50	Data byte has been received; ACK has been returned	Read data byte or Read data byte	0 0	0 0	1 1	0 1	Data byte will be received and NOT ACK will be returned Data byte will be received and ACK will be returned
0x58	Data byte has been received; NOT ACK has been returned	Read data byte or Read data byte or Read data byte	1 0 1	0 1 1	1 1 1	X 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

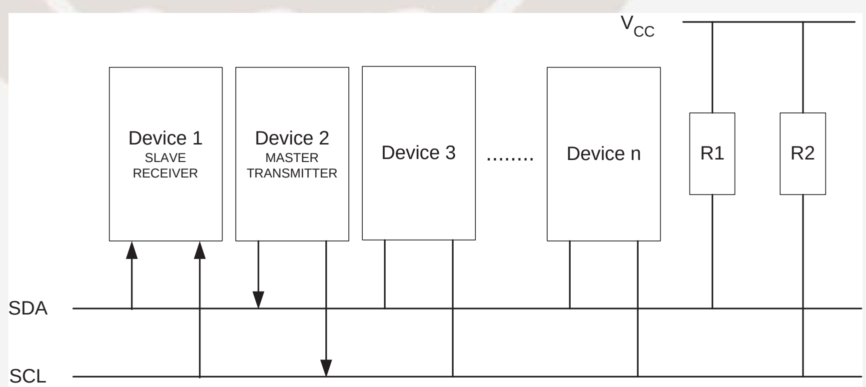
Figure 81. Formats and States in the Master Receiver Mode



Slave Receiver Mode

In the Slave Receiver mode, a number of data bytes are received from a Master Transmitter (see Figure 82). All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 82. 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 Own Slave Address							

The upper 7 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 (0x00), otherwise it will ignore the general call address.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	0	1	0	0	0	1	0	X

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 68 on page 179. The Slave Receiver mode may also be entered if arbitration is lost while the TWI is in the Master mode (see states 0x68 and 0x78).

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.

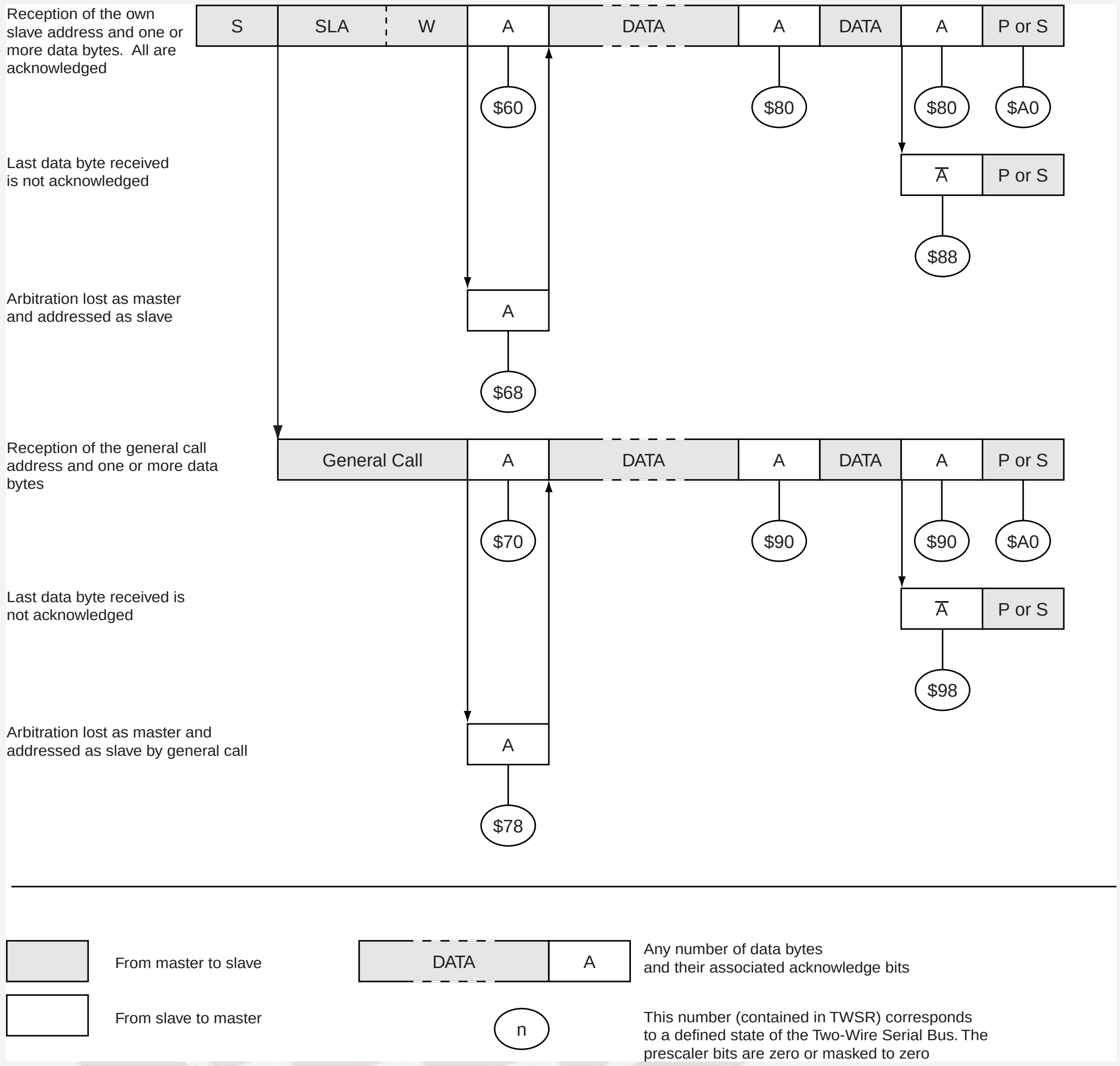
In all sleep modes other than Idle mode, the clock system to the TWI is turned off. If the TWEA bit is set, the interface can still acknowledge its own slave address or the general call address by using the Two-wire Serial Bus clock as a clock source. The part will then wake up from sleep and the TWI 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 68. Status Codes for Slave Receiver Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x60	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
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x68	Arbitration lost in SLA+R/W as Master; 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
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x70	General call address 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
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x78	Arbitration lost in SLA+R/W as Master; General call address 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
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x80	Previously addressed with own SLA+W; data has been received; ACK has been returned	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
0x88	Previously addressed with own SLA+W; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		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
0x90	Previously addressed with general call; data has been received; ACK has been returned	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
0x98	Previously addressed with general call; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		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
0xA0	A STOP condition or repeated START condition has been received while still addressed as Slave	No action	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
			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

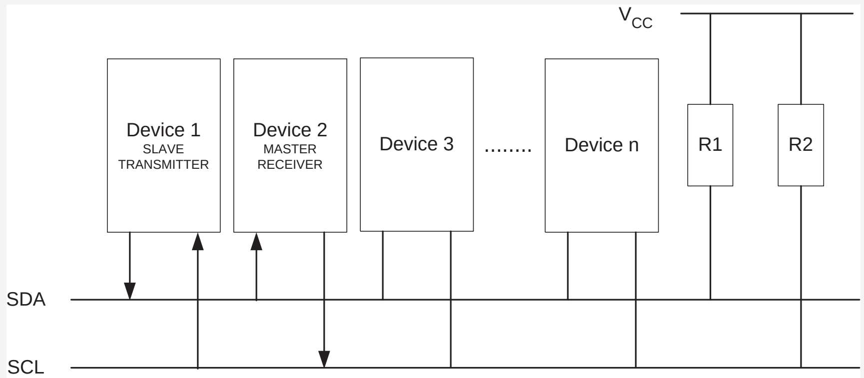
Figure 83. Formats and States in the Slave Receiver Mode



Slave Transmitter Mode

In the Slave Transmitter mode, a number of data bytes are transmitted to a Master Receiver (see Figure 84). All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 84. 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							

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 (0x00), otherwise it will ignore the general call address.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	0	1	0	0	0	1	0	X

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 69 on page 182. The Slave Transmitter mode may also be entered if arbitration is lost while the TWI is in the Master mode (see state 0xB0).

If the TWEA bit is written to zero during a transfer, the TWI will transmit the last byte of the transfer. State 0xC0 or state 0xC8 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 0xC8 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 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.

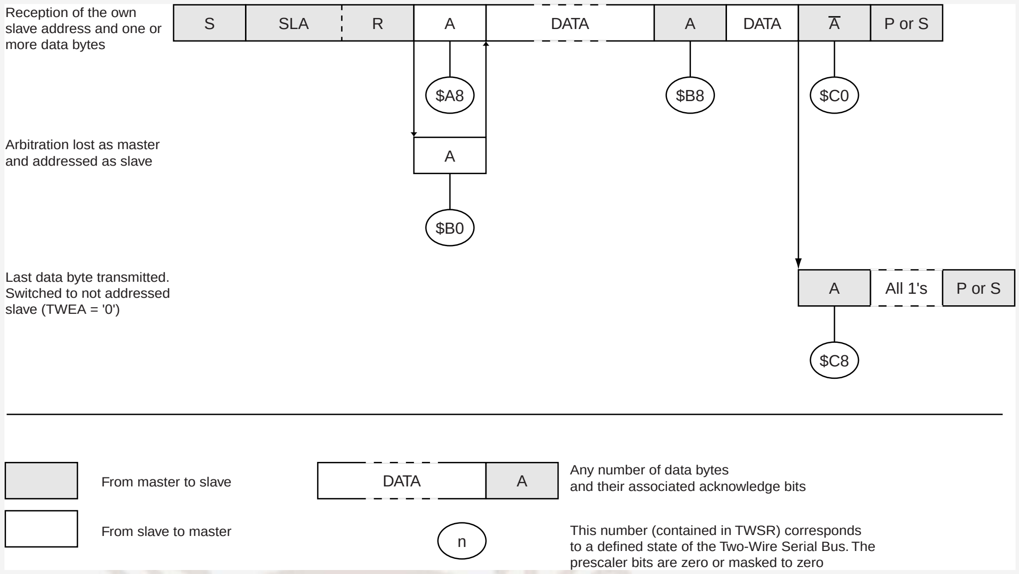
In all sleep modes other than Idle mode, the clock system to the TWI is turned off. If the TWEA bit is set, the interface can still acknowledge its own slave address or the general call address by using the Two-wire Serial Bus clock as a clock source. The part will then wake up from sleep and the TWI will hold the SCL clock 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 69. Status Codes for Slave Transmitter Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0xA8	Own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
0xB0	Arbitration lost in SLA+R/W as Master; own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
0xB8	Data byte in TWDR has been transmitted; ACK has been received	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
0xC0	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
0xC8	Last data byte in TWDR has been transmitted (TWEA = “0”); 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

Figure 85. 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 70.

Status 0xF8 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 0x00 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 70. Miscellaneous States

Status Code (TWSR) Prescaler Bits are 0	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application Software Response						Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR					
			STA	STO	TWINT	TWEA		
0xF8	No relevant state information available; TWINT = “0”	No TWDR action	No TWCR action					Wait or proceed current transfer
0x00	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.

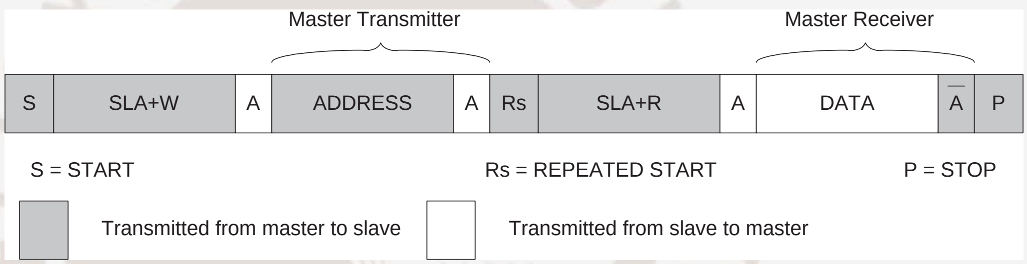
Combining Several TWI Modes

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 multimaster 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.

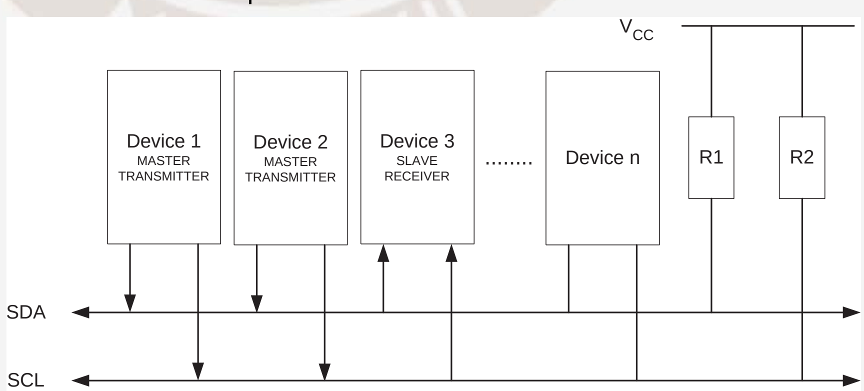
Figure 86. Combining Several TWI Modes to Access a Serial EEPROM



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 87. An Arbitration Example



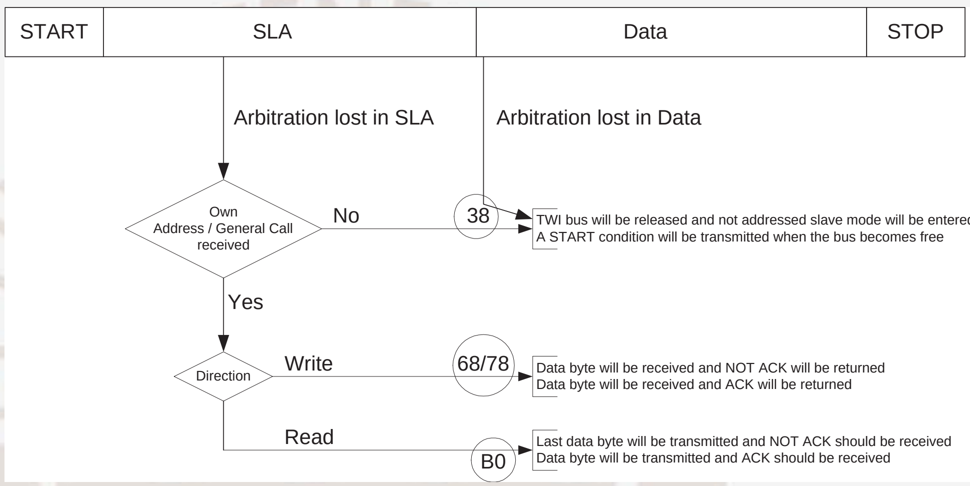
Several different scenarios may arise during arbitration, as described below:

- 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 88. Possible status values are given in circles.

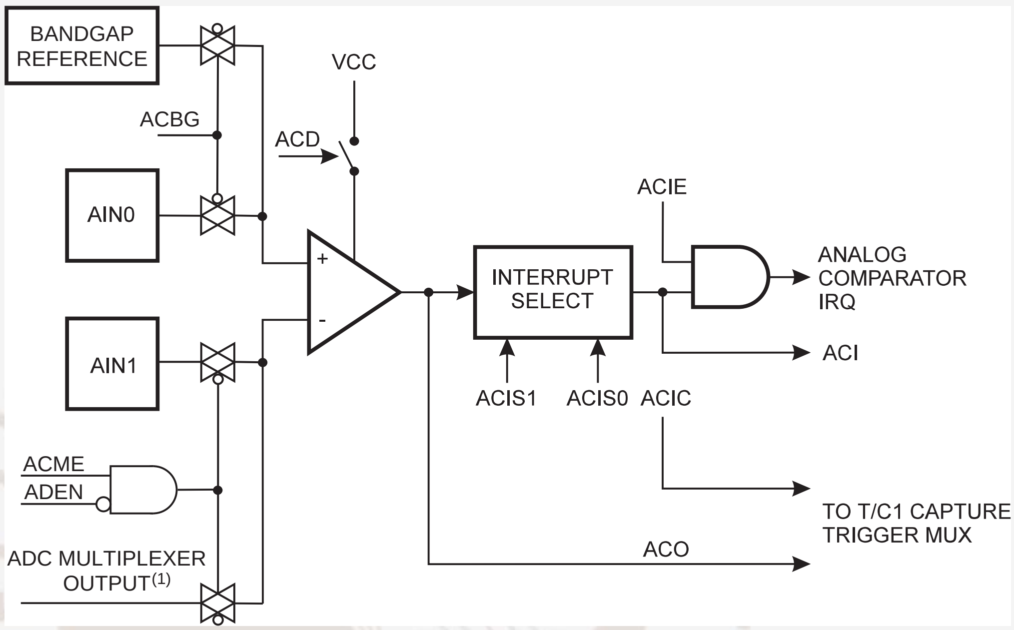
Figure 88. Possible Status Codes Caused by Arbitration



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 89.

Figure 89. Analog Comparator Block Diagram(2)



Notes: 1. See Table 72 on page 188
2. Refer to "Pin Configurations" on page 2 and Table 28 on page 63 for Analog Comparator pin placement

Special Function IO Register – SFIOR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **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 188.

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 71.

Table 71. 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[(1)] 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 72. If ACME is cleared or ADEN is set, AIN1 is applied to the negative input to the Analog Comparator.

Table 72. Analog Comparator Multiplexed Input[(1)]

ACME	ADEN	MUX2..0	Analog Comparator Negative Input
0	x	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

Note: 1. ADC7..6 are only available in TQFP and QFN/MLF Package

Analog-to-Digital Converter

Features

- **10-bit Resolution**
- **0.5 LSB Integral Non-linearity**
- **±2 LSB Absolute Accuracy**
- **13µs - 260µs Conversion Time**
- **Up to 15 kSPS at Maximum Resolution**
- **6 Multiplexed Single Ended Input Channels**
- **2 Additional Multiplexed Single Ended Input Channels (TQFP and QFN/MLF Package only)**
- **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**
- **Interrupt on ADC Conversion Complete**
- **Sleep Mode Noise Canceler**

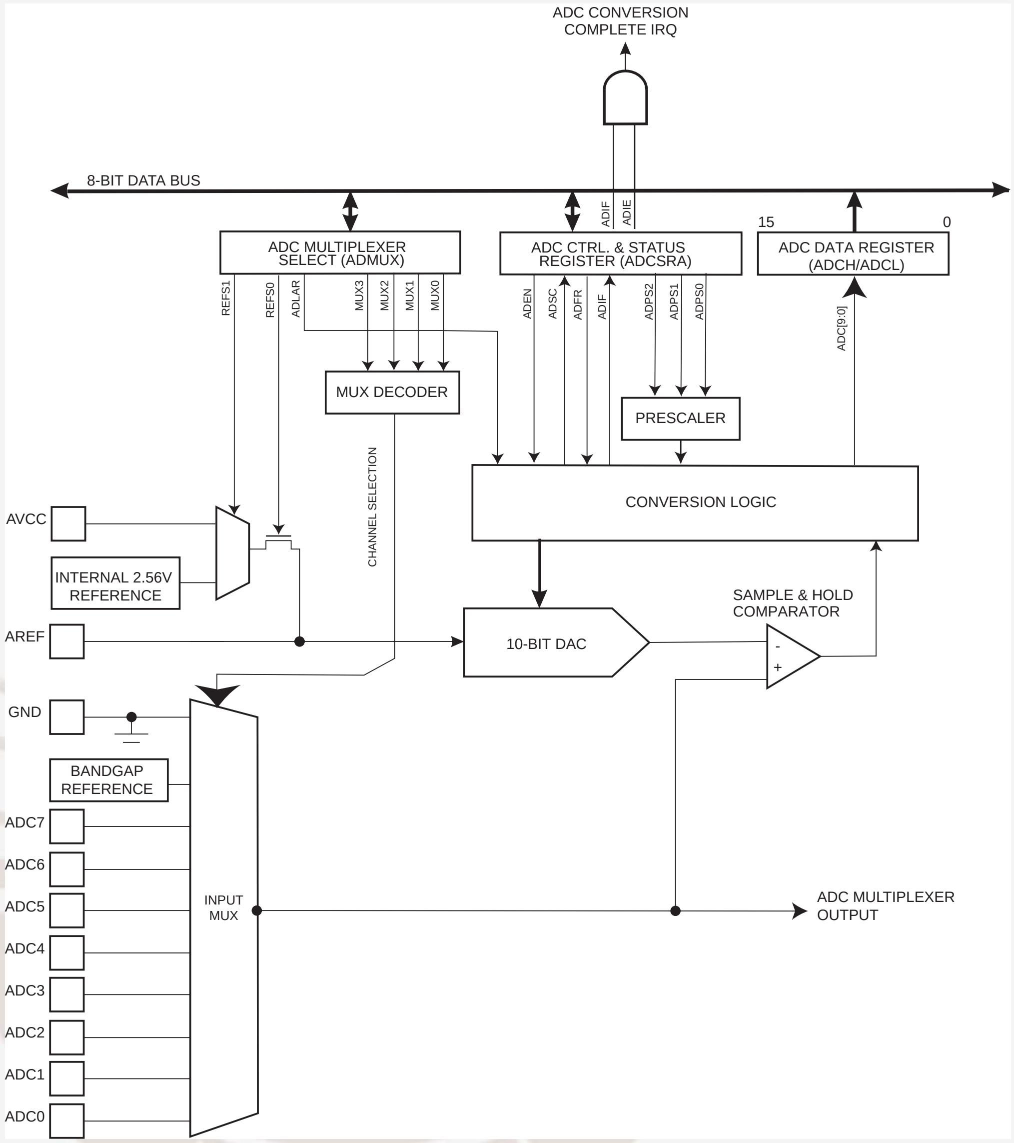
The ATmega8 features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer which allows eight single-ended voltage inputs constructed from the pins of Port C. The single-ended voltage inputs refer to 0V (GND).

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 90 on page 190.

The ADC has a separate analog supply voltage pin, AV_{CC}. AV_{CC} must not differ more than ±0.3V from V_{CC}. See the paragraph "ADC Noise Canceler" on page 195 on how to connect this pin.

Internal reference voltages of nominally 2.56V or AV_{CC} are provided On-chip. The voltage reference may be externally decoupled at the AREF pin by a capacitor for better noise performance.

Figure 90. 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, AV_{CC} 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 is 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. 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.

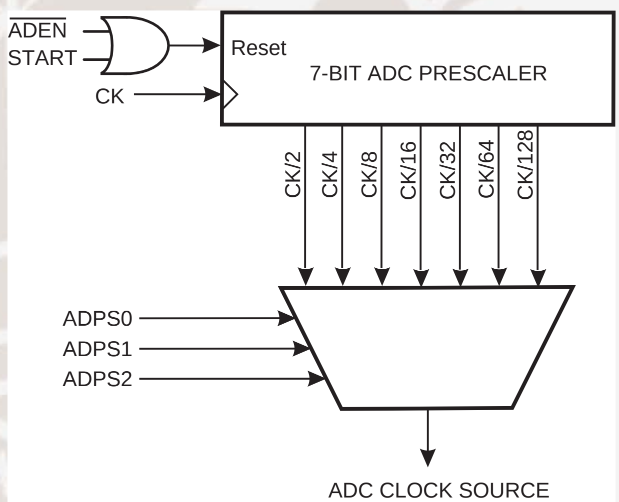
Starting a Conversion

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.

In Free Running mode, the ADC is constantly sampling and updating the ADC Data Register. Free Running mode is selected by writing the ADFR bit in ADCSRA to one. 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.

Prescaling and Conversion Timing

Figure 91. ADC Prescaler



By default, the successive approximation circuitry requires an input clock frequency between 50kHz and 200kHz to get maximum resolution. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be higher than 200kHz to get a higher sample rate.

The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100kHz. 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. 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 an 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.

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 73 on page 193.

Figure 92. ADC Timing Diagram, First Conversion (Single Conversion Mode)

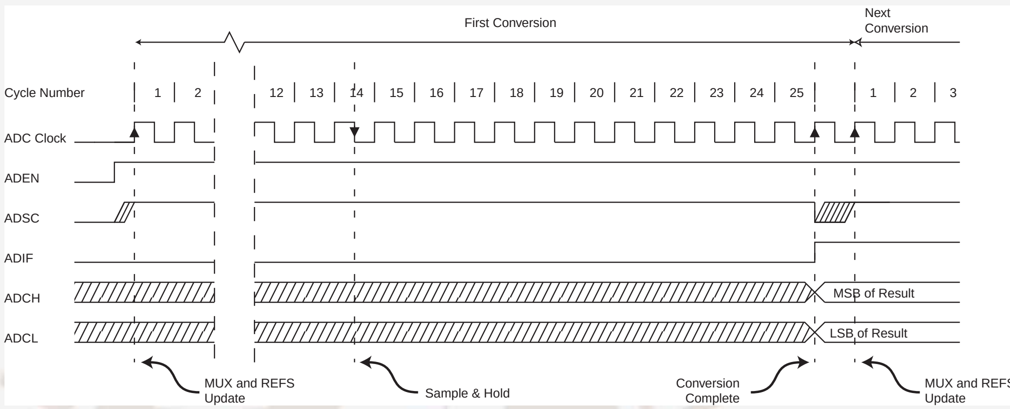


Figure 93. ADC Timing Diagram, Single Conversion

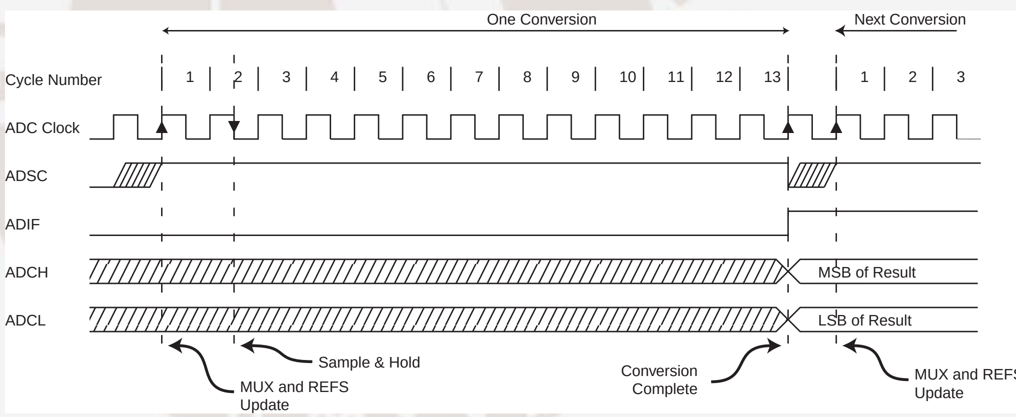


Figure 94. ADC Timing Diagram, Free Running Conversion

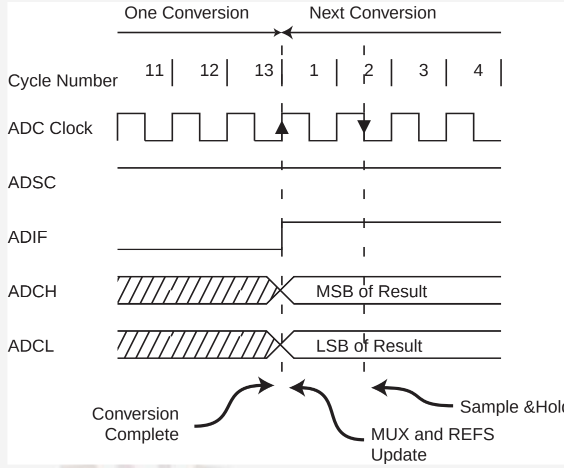


Table 73. ADC Conversion Time

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
Extended conversion	13.5	25
Normal conversions, single ended	1.5	13

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 both ADFR 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 ADFR 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.

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.

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 AV_{CC}, internal 2.56V reference, or external AREF pin.

AV_{CC} 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 AV_{CC} 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.

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:

1. 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.

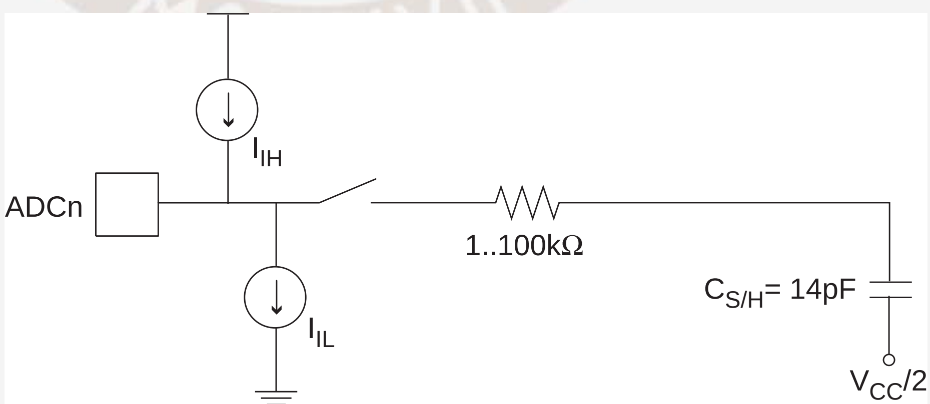
Analog Input Circuitry

The analog input circuitry for single ended channels is illustrated in Figure 95. 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 kΩ 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.

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 95. Analog Input Circuitry

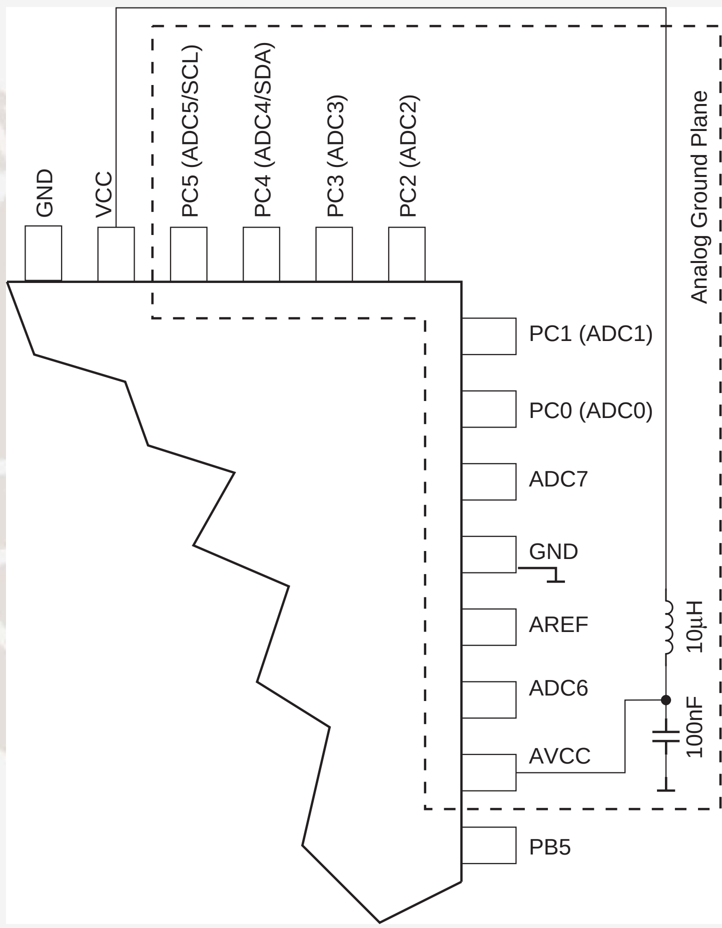


Analog Noise Canceling Techniques

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:

1. Keep analog signal paths as short as possible. Make sure analog tracks run over the ground plane, and keep them well away from high-speed switching digital tracks.
2. The AV_{CC} pin on the device should be connected to the digital V_{CC} supply voltage via an LC network as shown in Figure 96.
3. Use the ADC noise canceler function to reduce induced noise from the CPU.
4. If any ADC [3..0] port pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress. However, using the Two-wire Interface (ADC4 and ADC5) will only affect the conversion on ADC4 and ADC5 and not the other ADC channels.

Figure 96. ADC Power Connections



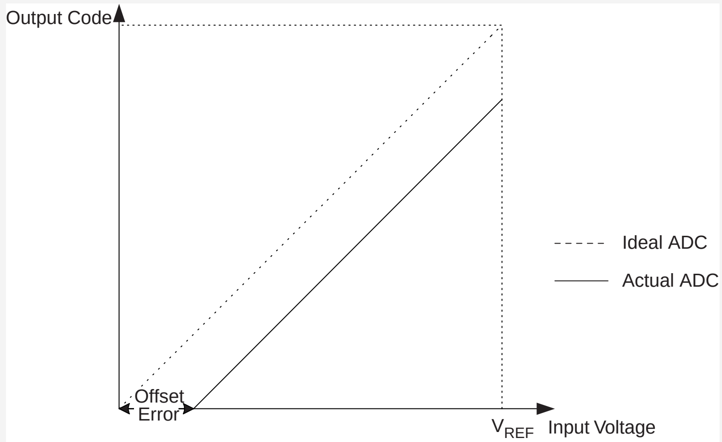
ADC Accuracy Definitions

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:

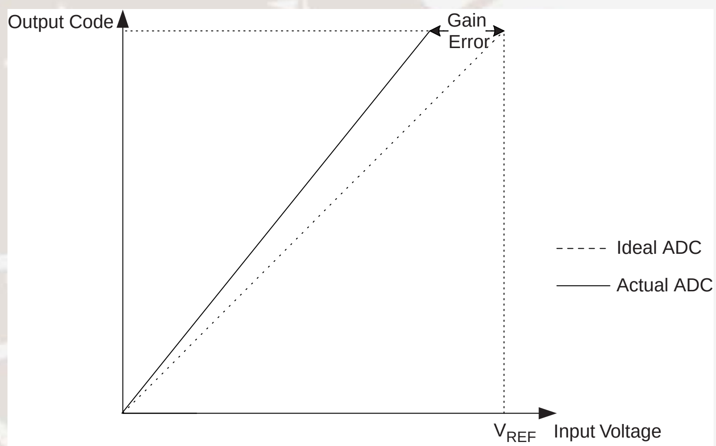
- Offset: The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB

Figure 97. Offset Error



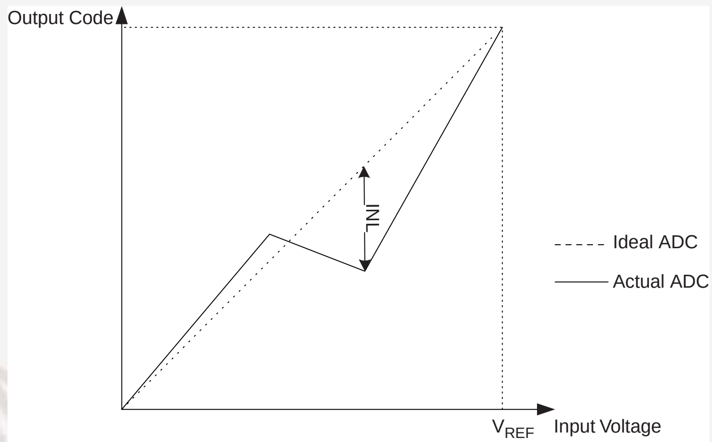
- Gain error: After adjusting for offset, the gain error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB

Figure 98. Gain Error



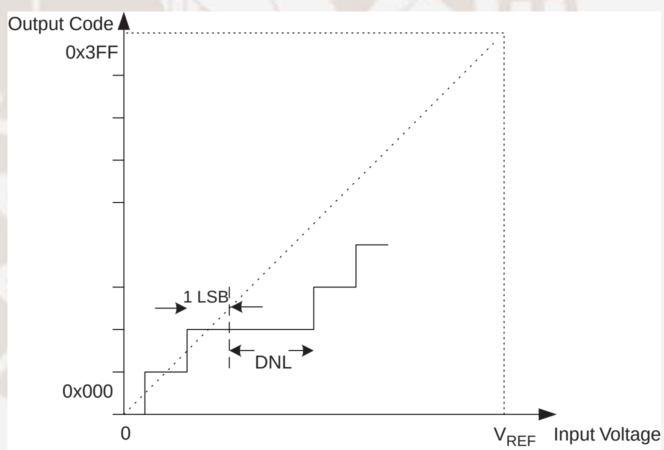
- Integral Non-linearity (INL): After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSB

Figure 99. 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 100. 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 74 and Table 75). 0x000 represents ground, and 0x3FF represents the selected reference voltage minus one LSB.

ADC Multiplexer Selection Register – ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
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 7:6 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 74. 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 74. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{ref} turned off
0	1	AV_{CC} 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 conversions. For a complete description of this bit, see "The ADC Data Register – ADCL and ADCH" on page 201.

- **Bits 3:0 – MUX3:0: Analog Channel Selection Bits**

The value of these bits selects which analog inputs are connected to the ADC. See Table 75 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 75. Input Channel Selections

MUX3..0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5

Table 75. Input Channel Selections (Continued)

MUX3..0	Single Ended Input
0110	ADC6
0111	ADC7
1000	
1001	
1010	
1011	
1100	
1101	
1110	1.30V (V_{BG})
1111	0V (GND)

ADC Control and Status Register A – ADCSRA

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADFR	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 – ADFR: ADC Free Running Select**

When this bit is set (one) the ADC operates in Free Running mode. In this mode, the ADC samples and updates the Data Registers continuously. Clearing this bit (zero) will terminate Free Running mode.

- **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.

Table 76. 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.

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 199.

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.

Boot Loader 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 89 on page 218) 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 102 on page 204). The size of the different sections is configured by the BOOTSZ Fuses as shown in Table 82 on page 213 and Figure 102 on page 204. 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 78 on page 205. 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 79 on page 205.

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 83 on page 214 and Figure 102 on page 204. 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 on-going 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/rjmp/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 206. 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 77. 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 101. Read-While-Write vs. No Read-While-Write

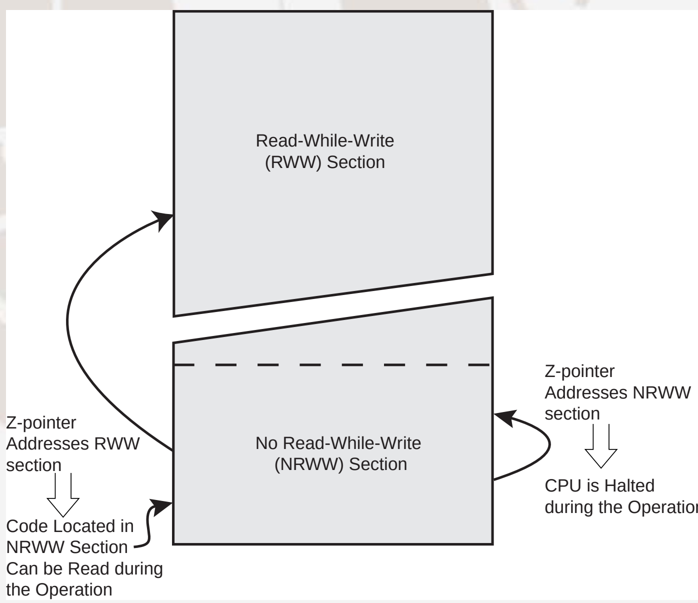
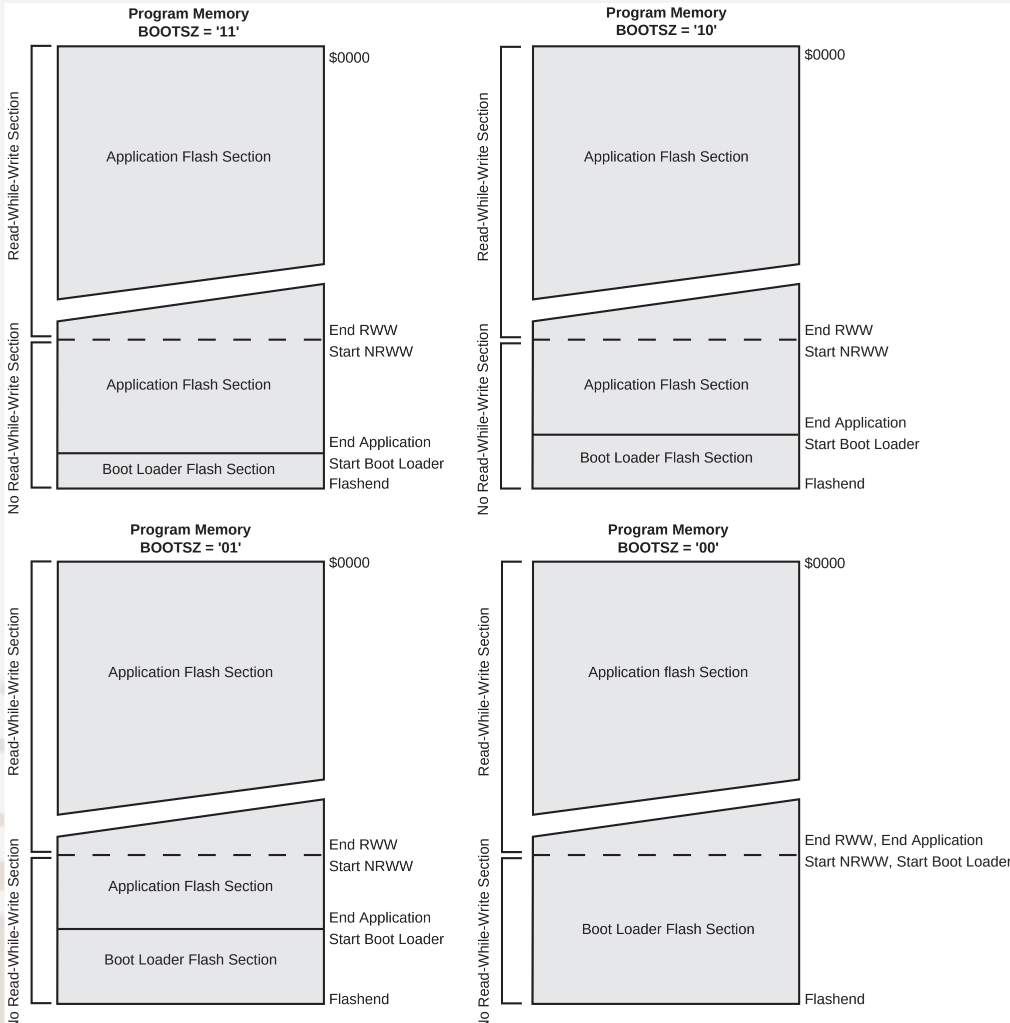


Figure 102. Memory Sections[(1)]



Note: 1. The parameters in the figure are given in Table 82 on page 213

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 78 on page 205 and Table 79 on page 205 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.

Table 78. Boot Lock Bit0 Protection Modes (Application Section)[1]

BLB0 Mode	BLB02 Mode	BLB01 Mode	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 79. Boot Lock Bit1 Protection Modes (Boot Loader Section)[1]

BLB1 Mode	BLB12 Mode	BLB11 Mode	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 80. Boot Reset Fuse[1]

BOOTRST	Reset Address
1	Reset Vector = Application Reset (address 0x0000)
0	Reset Vector = Boot Loader Reset (see Table 82 on page 213)

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 ATmega8 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 (The page buffer will be cleared when the Read-While-Write section is re-enabled).

- **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 210 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”, “00011” or “00001” in the lower five bits will have no effect.

Addressing the Flash During Self-Programming

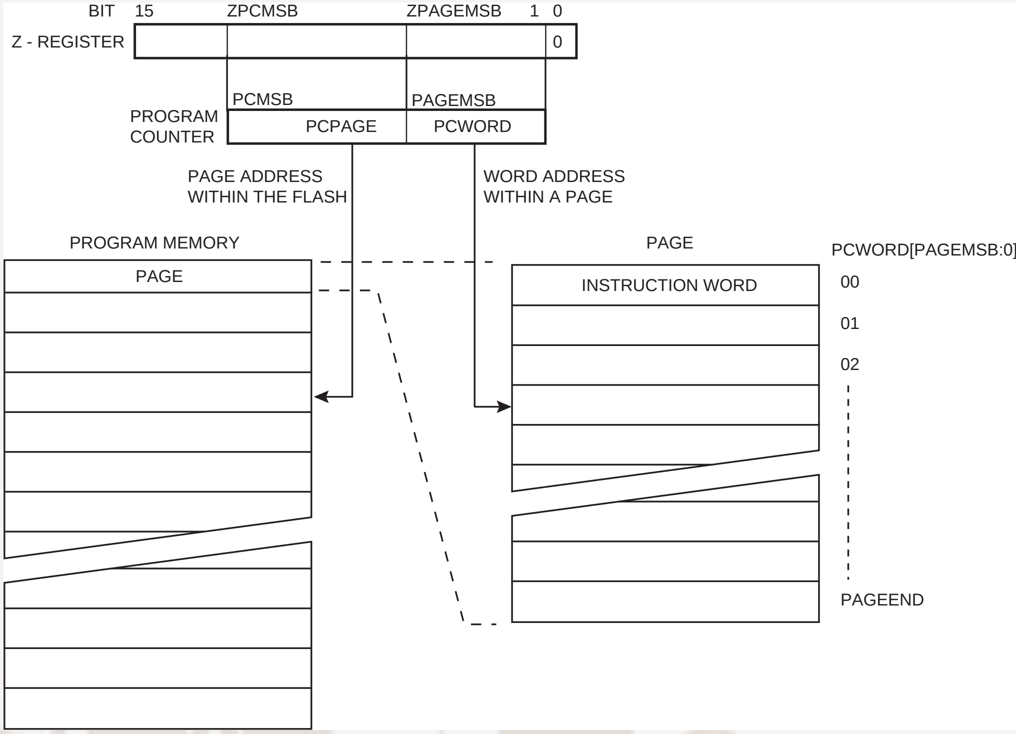
The Z-pointer is used to address the SPM commands.

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	5	4	3	2	1	0

Since the Flash is organized in pages (see Table 89 on page 218), 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 103 on page 208. 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 103. Addressing the Flash during SPM(1)



Notes: 1. The different variables used in the figure are listed in Table 84 on page 214
2. PCPAGE and PCWORD are listed in Table 89 on page 218

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 212 for an assembly code example.

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 will be ignored 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

Note: If an interrupt occurs in the timed sequence, the four cycle access cannot be guaranteed. In order to ensure atomic operation, disable interrupts before writing to SPMCSR.

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 page 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 to 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 46.

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 46, 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 212 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 78 on page 205 and Table 79 on page 205 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 0x0001 (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 0x0001 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 BLBSET 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 0x0000 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 88 on page 217 for a detailed description and mapping of the fuse low bits.

Bit	7	6	5	4	3	2	1	0
Rd	FLB7	FLB6	FLB5	FLB4	FLB3	FLB2	FLB1	FLB0

Similarly, when reading the Fuse High bits, load 0x0003 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 87 on page 216 for detailed description and mapping of the fuse high bits.

Bit	7	6	5	4	3	2	1	0
Rd	FHB7	FHB6	FHB5	FHB4	FHB3	FHB2	FHB1	FHB0

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.

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):

1. 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
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 81 shows the typical programming time for Flash accesses from the CPU.

Table 81. SPM Programming Time[1]

Symbol	Min Programming Time	Max Programming Time
Flash write (page erase, page write, and write Lock Bits by SPM)	3.7ms	4.5ms

Note: 1. Minimum and maximum programming time is per individual operation

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)
  rcallDo_spm

  ; re-enable the RWW section
  ldi  spmcrval, (1<<RWWSRE) | (1<<SPMEN)
  rcallDo_spm

  ; transfer data from RAM to Flash page buffer
  ldi  looplo, low(PAGESIZEB)   ;init loop variable
  ldi  loophi, high(PAGESIZEB)  ;not required for PAGESIZEB<=256
Wrloop:
  ld   r0, Y+
  ld   r1, Y+
  ldi  spmcrval, (1<<SPMEN)
  rcallDo_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)
  rcallDo_spm

  ; re-enable the RWW section
  ldi  spmcrval, (1<<RWWSRE) | (1<<SPMEN)
  rcallDo_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)       ;restore pointer
  sbci YH, high(PAGESIZEB)
Rdloop:
  lpm  r0, Z+
  ld   r1, Y+
  cpse r0, r1
  rjmp 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:
  in   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)
  rcallDo_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
```

ATmega8 Boot Loader Parameters

In Table 82 through Table 84 on page 214, the parameters used in the description of the self programming are given.

Table 82. Boot Size Configuration

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	4	0x000 - 0xF7F	0xF80 - 0xFFF	0xF7F	0xF80
1	0	256 words	8	0x000 - 0xEFF	0xF00 - 0xFFF	0xEFF	0xF00
0	1	512 words	16	0x000 - 0xDFF	0xE00 - 0xFFF	0xDFF	0xE00
0	0	1024 words	32	0x000 - 0xBFF	0xC00 - 0xFFF	0xBFF	0xC00

Note: The different BOOTSZ Fuse configurations are shown in Figure 102 on page 204

Table 83. Read-While-Write Limit

Section	Pages	Address
Read-While-Write section (RWW)	96	0x000 - 0xBFF
No Read-While-Write section (NRWW)	32	0xC00 - 0xFFF

For details about these two section, see "NRWW – No Read-While-Write Section" on page 203 and "RWW – Read-While-Write Section" on page 203.

Table 84. Explanation of Different Variables used in Figure 103 on page 208 and the Mapping to the Z-pointer

Variable		Corresponding Z-value[(1)]	Description
PCMSB	11		Most significant bit in the Program Counter. (The Program Counter is 12 bits PC[11:0])
PAGEMSB	4		Most significant bit which is used to address the words within one page (32 words in a page requires 5 bits PC [4:0])
ZPCMSB		Z12	Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1
ZPAGEMSB		Z5	Bit in Z-register that is mapped to PAGEMSB. Because Z0 is not used, the ZPAGEMSB equals PAGEMSB + 1
PCPAGE	PC[11:5]	Z12:Z6	Program counter page address: Page select, for page erase and page write
PCWORD	PC[4:0]	Z5:Z1	Program counter word address: Word select, for filling temporary buffer (must be zero during page write operation)

Note: 1. Z15:Z13: 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 207 for details about the use of Z-pointer during Self-Programming

Memory Programming

Program And Data Memory Lock Bits

The ATmega8 provides six Lock Bits which can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional features listed in Table 86. The Lock Bits can only be erased to “1” with the Chip Erase command.

Table 85. Lock Bit Byte

Lock Bit Byte	Bit No.	Description	Default Value[1]
	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 86. Lock Bit Protection Modes[2]

Memory Lock Bits			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 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 Serial Programming mode. The Fuse Bits are locked in both Serial and Parallel Programming modes [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 86. Lock Bit Protection Modes[2] (Continued)

Memory Lock Bits			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 ATmega8 has two fuse bytes. Table 87 and Table 88 on page 217 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 87. Fuse High Byte

Fuse High Byte	Bit No.	Description	Default Value
RSTDISBL[4]	7	Select if PC6 is I/O pin or RESET pin	1 (unprogrammed, PC6 is RESET-pin)
WDTON	6	WDT always on	1 (unprogrammed, WDT enabled by WDTCR)
SPIEN[1]	5	Enable 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 82 on page 213 for details)	0 (programmed)[3]
BOOTSZ0	1	Select Boot Size (see Table 82 on page 213 for details)	0 (programmed)[3]
BOOTRST	0	Select Reset Vector	1 (unprogrammed)

Notes: 1. The SPIEN Fuse is not accessible in Serial Programming mode
2. The CKOPT Fuse functionality depends on the setting of the CKSEL bits, see "Clock Sources" on page 26 for details
3. The default value of BOOTSZ1..0 results in maximum Boot Size. See Table 82 on page 213
4. When programming the RSTDISBL Fuse Parallel Programming has to be used to change fuses or perform further programming

Table 88. 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. See Table 10 on page 30 for details

2. The default setting of CKSEL3..0 results in internal RC Oscillator @ 1MHz. See Table 2 on page 26 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 3-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 ATmega8 the signature bytes are:

1. 0x000: 0x1E (indicates manufactured by Atmel)
2. 0x001: 0x93 (indicates 8KB Flash memory)
3. 0x002: 0x07 (indicates ATmega8 device)

Calibration Byte

The ATmega8 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 1MHz, 2MHz, 4MHz, and 8Mhz respectively. During Reset, the 1MHz 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 31 for details.

Page Size

Table 89. No. of Words in a Page and no. of Pages in the Flash

Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
4K words (8 Kbytes)	32 words	PC[4:0]	128	PC[11:5]	11

Table 90. 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 ATmega8. Pulses are assumed to be at least 250ns unless otherwise noted.

Signal Names

In this section, some pins of the ATmega8 are referenced by signal names describing their functionality during parallel programming, see Figure 104 and Table 91. 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 93 on page 220.

When pulsing WR or OE, the command loaded determines the action executed. The different Commands are shown in Table 94 on page 220.

Figure 104. Parallel Programming

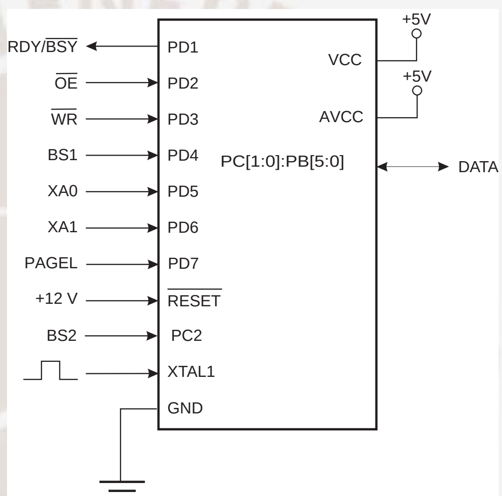


Table 91. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
RDY/BSY	PD1	O	0: Device is busy programming, 1: Device is ready for new command
OE	PD2	I	Output Enable (Active low)
WR	PD3	I	Write Pulse (Active low)
BS1	PD4	I	Byte Select 1 (“0” selects Low byte, “1” selects High byte)
XA0	PD5	I	XTAL Action Bit 0
XA1	PD6	I	XTAL Action Bit 1

Table 91. Pin Name Mapping (Continued)

Signal Name in Programming Mode	Pin Name	I/O	Function
PAGEL	PD7	I	Program memory and EEPROM Data Page Load
BS2	PC2	I	Byte Select 2 (“0” selects Low byte, “1” selects 2’nd High byte)
DATA	{PC[1:0]: PB[5:0]}	I/O	Bi-directional Data bus (Output when $\overline{OE}$ is low)

Table 92. 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 93. 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 94. 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
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 100µs
2. Set $\overline{\text{RESET}}$ to “0” and toggle XTAL1 at least 6 times
3. Set the Prog_enable pins listed in Table 92 on page 220 to “0000” and wait at least 100ns
4. Apply 11.5V - 12.5V to $\overline{\text{RESET}}$. Any activity on Prog_enable pins within 100ns after +12V has been applied to $\overline{\text{RESET}}$, will cause the device to fail entering Programming mode

Note, if the $\overline{\text{RESET}}$ pin is disabled by programming the RSTDISBL Fuse, it may not be possible to follow the proposed algorithm above. The same may apply when External Crystal or External RC configuration is selected because it is not possible to apply qualified XTAL1 pulses. In such cases, the following algorithm should be followed:

1. Set Prog_enable pins listed in Table 92 on page 220 to “0000”
2. Apply 4.5V - 5.5V between V_{CC} and GND simultaneously as 11.5V - 12.5V is applied to $\overline{\text{RESET}}$
3. Wait 100ns
4. Re-program the fuses to ensure that External Clock is selected as clock source (CKSEL3:0 = 0’b0000) and $\overline{\text{RESET}}$ pin is activated (RSTDISBL unprogrammed). 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 $\overline{\text{RESET}}$ pin to 0’b0
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 0xFF, 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/$\overline{\text{BSY}}$ goes high before loading a new command

Programming the Flash

The Flash is organized in pages, see Table 89 on page 218. 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 (0x00 - 0xFF)
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 (0x00 - 0xFF)
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 (0x00 - 0xFF)
4. Give XTAL1 a positive pulse. This loads the data byte

E. Latch Data

1. Set BS1 to "1". This selects high data byte
2. Give PAGEL a positive pulse. This latches the data bytes (see Figure 106 on page 224 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 105 on page 223. Note that if less than eight 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 (0x00 - 0xFF)
4. Give XTAL1 a positive pulse. This loads the address High byte

H. Program Page

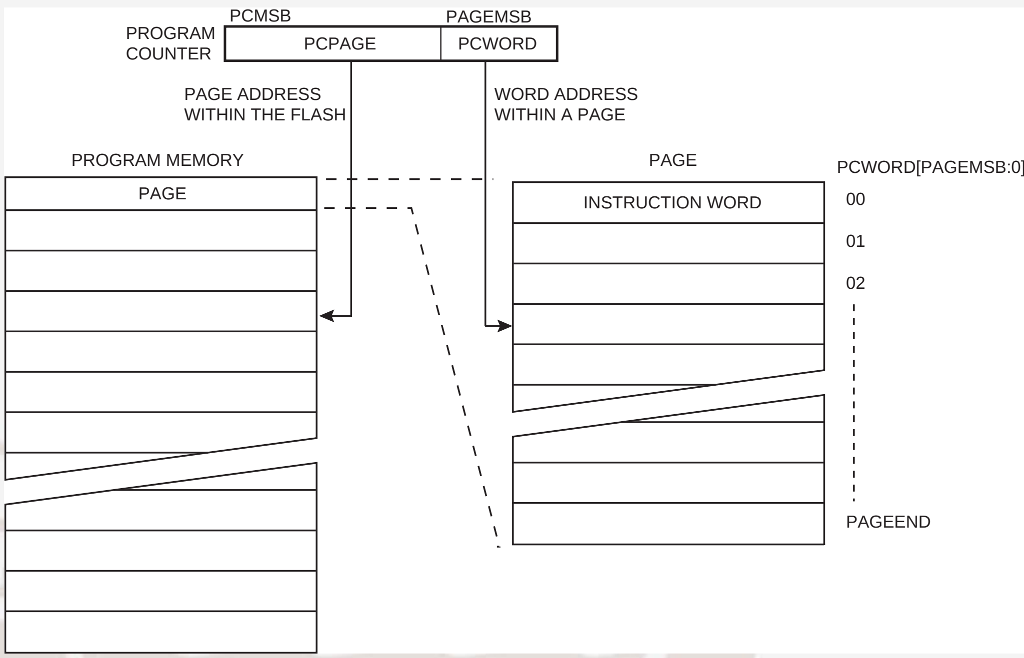
1. Set BS1 = "0"
2. Give $\overline{\text{WR}}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{\text{BSY}}$ goes low
3. Wait until RDY/$\overline{\text{BSY}}$ goes high. (See Figure 106 on page 224 for signal waveforms)

I. Repeat B through H until the entire Flash is programmed or until all data has been programmed.

J. End Page Programming

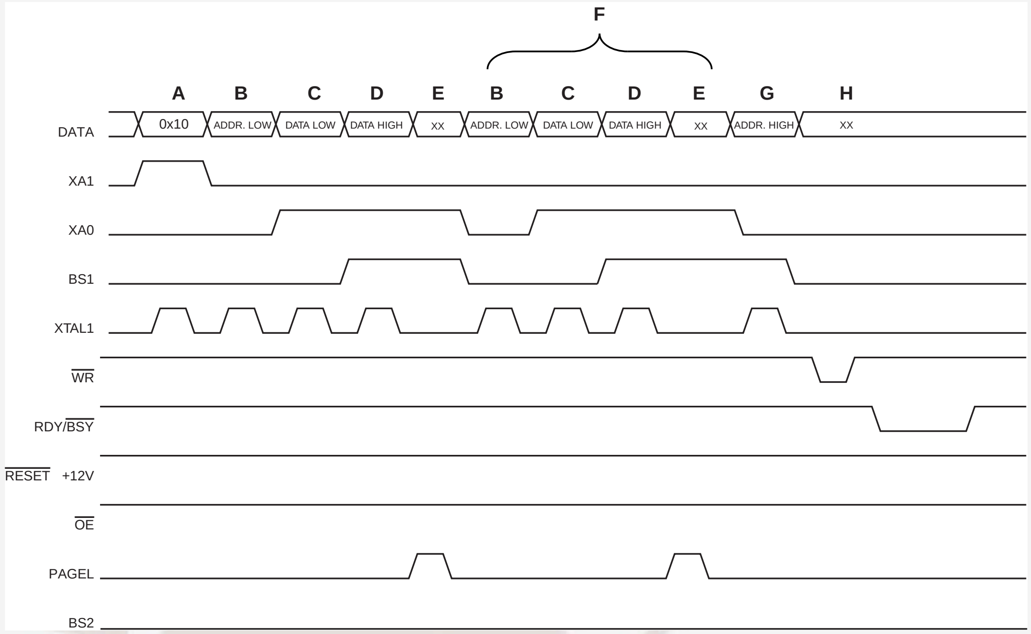
1. Set XA1, XA0 to “10”. This enables command loading
2. Set DATA to “0000 0000”. This is the command for No Operation
3. Give XTAL1 a positive pulse. This loads the command, and the internal write signals are reset

Figure 105. Addressing the Flash which is Organized in Pages[(1)]



Note: 1. PCPAGE and PCWORD are listed in Table 89 on page 218

Figure 106. 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 90 on page 218. 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 222 for details on Command, Address and Data loading):

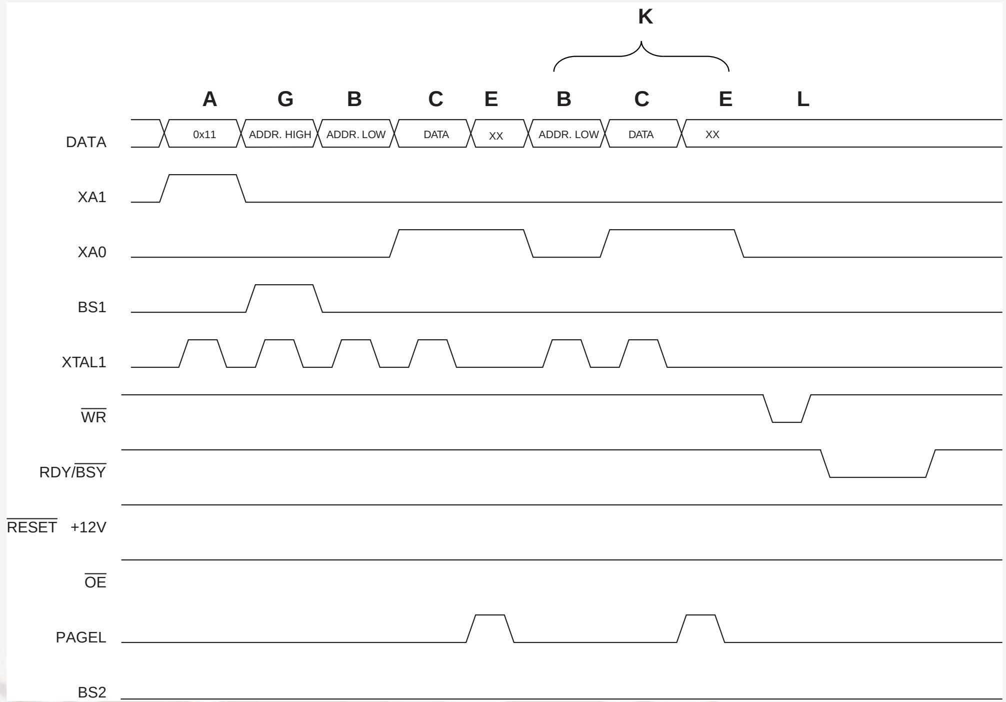
1. A: Load Command "0001 0001"
2. G: Load Address High byte (0x00 - 0xFF)
3. B: Load Address Low byte (0x00 - 0xFF)
4. C: Load Data (0x00 - 0xFF)
5. E: Latch data (give PAGEL a positive pulse)

K: Repeat 3 through 5 until the entire buffer is filled

L: Program EEPROM page

1. Set BS1 to "0"
2. Give $\overline{WR}$ a negative pulse. This starts programming of the EEPROM page. RDY/$\overline{BSY}$ goes low
3. Wait until to RDY/$\overline{BSY}$ goes high before programming the next page (see Figure 107 on page 225 for signal waveforms)

Figure 107. Programming the EEPROM Waveforms



Reading the Flash

The algorithm for reading the Flash memory is as follows (refer to "Programming the Flash" on page 222 for details on Command and Address loading):

1. A: Load Command "0000 0010"
2. G: Load Address High byte (0x00 - 0xFF)
3. B: Load Address Low byte (0x00 - 0xFF)
4. Set $\overline{OE}$ 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 $\overline{OE}$ to "1"

Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to "Programming the Flash" on page 222 for details on Command and Address loading):

1. A: Load Command "0000 0011"
2. G: Load Address High byte (0x00 - 0xFF)
3. B: Load Address Low byte (0x00 - 0xFF)
4. Set $\overline{OE}$ to "0", and BS1 to "0". The EEPROM Data byte can now be read at DATA
5. Set $\overline{OE}$ 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 222 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 and BS2 to "0"
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high

Programming the Fuse High Bits

The algorithm for programming the Fuse high bits is as follows (refer to "Programming the Flash" on page 222 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 $\overline{\text{WR}}$ a negative pulse and wait for RDY/$\overline{\text{BSY}}$ to go high
5. Set BS1 to "0". This selects low data byte

Programming the Lock Bits

The algorithm for programming the Lock Bits is as follows (refer to "Programming the Flash" on page 222 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 $\overline{\text{WR}}$ a negative pulse and wait for RDY/$\overline{\text{BSY}}$ to go high

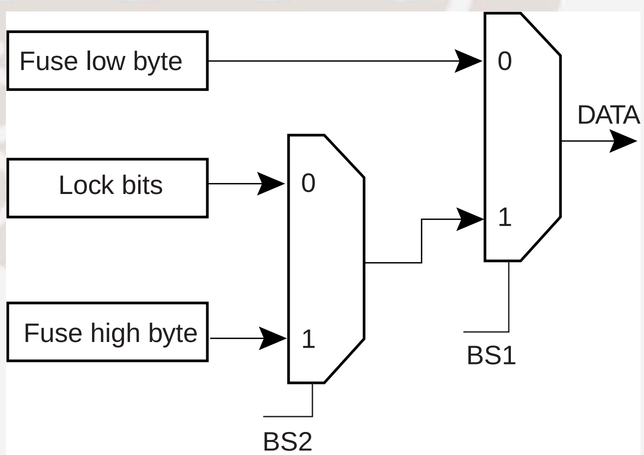
The Lock Bits can only be cleared by executing Chip Erase.

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 222 for details on Command loading):

1. A: Load Command "0000 0100"
2. 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)
3. Set $\overline{\text{OE}}$ 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)
4. 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 $\overline{\text{OE}}$ to "1"

Figure 108. 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 222 for details on Command and Address loading):

1. A: Load Command “0000 1000”
2. B: Load Address Low byte (0x00 - 0x02)
3. Set $\overline{\text{OE}}$ to “0”, and BS1 to “0”. The selected Signature byte can now be read at DATA.
4. Set $\overline{\text{OE}}$ to “1”

Reading the Calibration Byte

The algorithm for reading the Calibration bytes is as follows (refer to “Programming the Flash” on page 222 for details on Command and Address loading):

1. A: Load Command “0000 1000”
2. B: Load Address Low byte, (0x00 - 0x03)
3. Set $\overline{\text{OE}}$ to “0”, and BS1 to “1”. The Calibration byte can now be read at DATA
4. Set $\overline{\text{OE}}$ to “1”

Parallel Programming Characteristics

Figure 109. Parallel Programming Timing, Including some General Timing Requirements

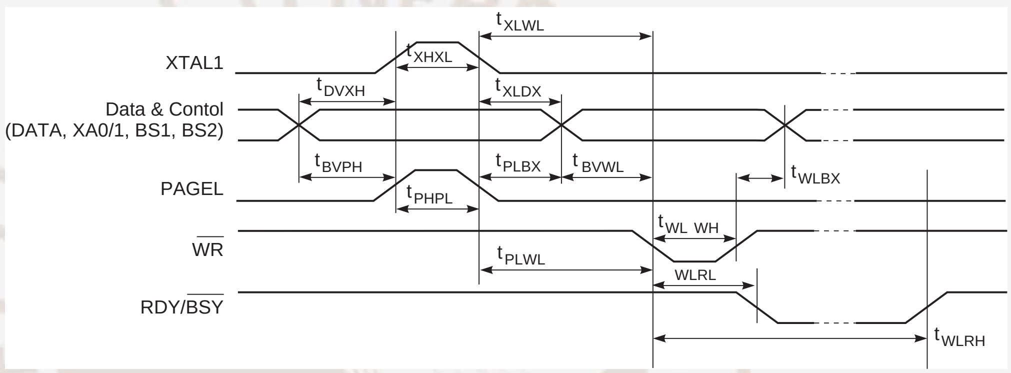
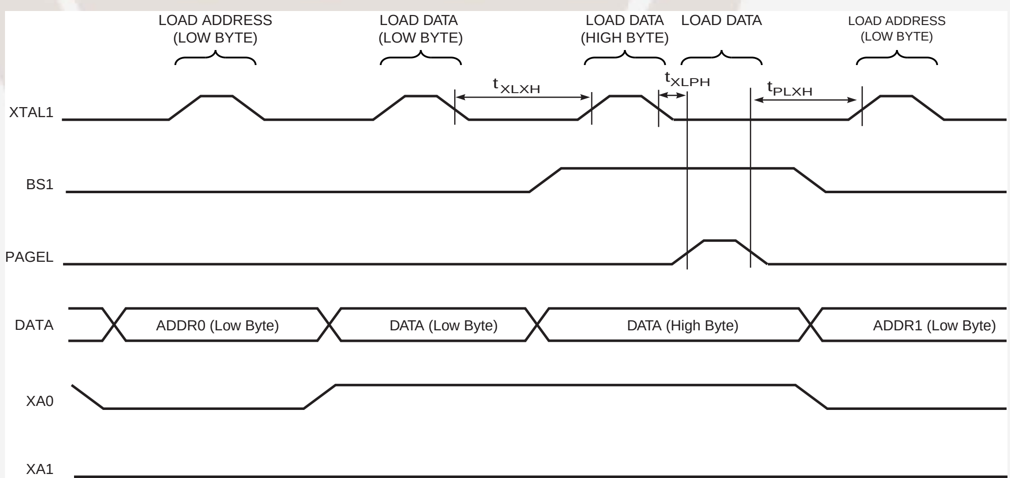
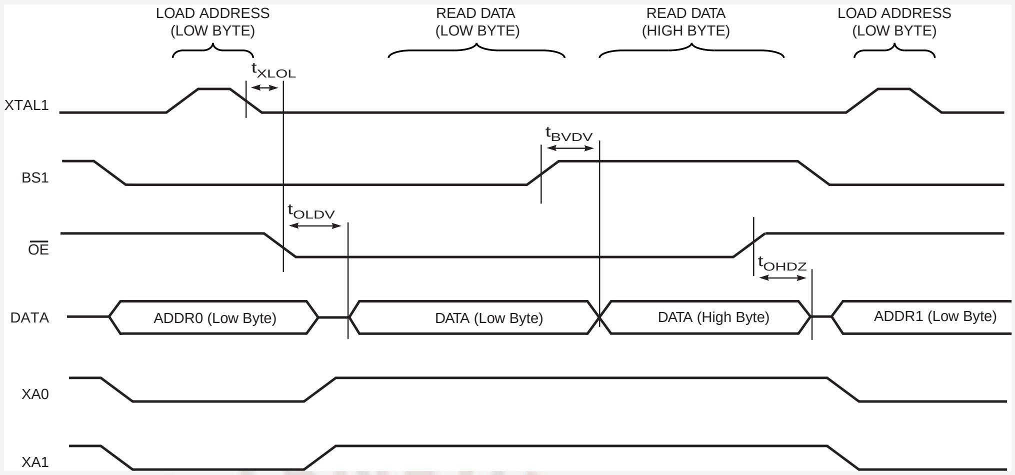


Figure 110. Parallel Programming Timing, Loading Sequence with Timing Requirements[1]



Note: 1. The timing requirements shown in Figure 109 (that is, t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to loading operation

Figure 111. Parallel Programming Timing, Reading Sequence (within the same Page) with Timing Requirements(1)



Note: 1. The timing requirements shown in Figure 109 on page 227 (that is, t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to reading operation

Table 95. Parallel Programming Characteristics, V_{CC} = 5V ±10%

Symbol	Parameter	Min	Typ	Max	Units
V_{PP}	Programming Enable Voltage	11.5		12.5	V
I_{PP}	Programming Enable Current			250	µA
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 $\overline{WR}$ Low	0			
t_{XLPH}	XTAL1 Low to PAGEL high	0			
t_{PLXH}	PAGEL low to XTAL1 high	150			ns
t_{BVPH}	BS1 Valid before PAGEL High	67			
t_{PHPL}	PAGEL Pulse Width High	150			
t_{PLBX}	BS1 Hold after PAGEL Low	67			
t_{WLBX}	BS2/1 Hold after $\overline{WR}$ Low	67			
t_{PLWL}	PAGEL Low to $\overline{WR}$ Low	67			
t_{BVWL}	BS1 Valid to $\overline{WR}$ Low	67			
t_{WLWH}	$\overline{WR}$ Pulse Width Low	150			
t_{WLRL}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ Low	0		1	µs
t_{WLRH}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High(1)	3.7		4.5	ms
t_{WLRH_CE}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High for Chip Erase(2)	7.5		9	

Table 95. Parallel Programming Characteristics, V_{CC} = 5V ±10% (Continued)

Symbol	Parameter	Min	Typ	Max	Units
t_{XLOL}	XTAL1 Low to $\overline{OE}$ Low	0			ns
t_{BVDV}	BS1 Valid to DATA valid	0		250	
t_{OLDV}	$\overline{OE}$ Low to DATA Valid			250	
t_{OHDZ}	$\overline{OE}$ 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

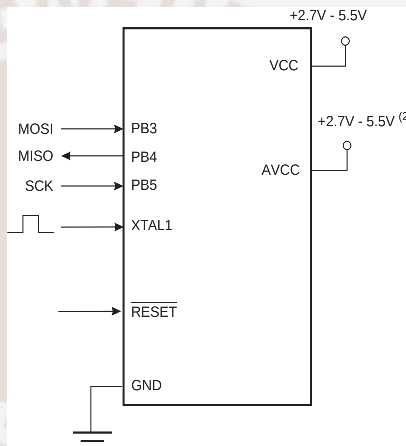
Both the Flash and EEPROM memory arrays can be programmed using the serial SPI bus while RESET is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After RESET is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed. NOTE, in Table 96, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

Serial Programming Pin Mapping

Table 96. Pin Mapping Serial Programming

Symbol	Pins	I/O	Description
MOSI	PB3	I	Serial data in
MISO	PB4	O	Serial data out
SCK	PB5	I	Serial clock

Figure 112. 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} - 0.3 < AV_{CC} < V_{CC} + 0.3, however, AV_{CC} 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 0xFF.

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} <12MHz, 3 CPU clock cycles for f_{ck} >=12MHz

High:> 2 CPU clock cycles for f_{ck} <12MHz, 3 CPU clock cycles for f_{ck} >=12MHz

Serial Programming Algorithm

When writing serial data to the ATmega8, data is clocked on the rising edge of SCK.

When reading data from the ATmega8, data is clocked on the falling edge of SCK. See Figure 113 on page 232 for timing details.

To program and verify the ATmega8 in the Serial Programming mode, the following sequence is recommended (see four byte instruction formats in Table 98 on page 233):

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"
2. Wait for at least 20ms and enable Serial Programming by sending the Programming Enable serial instruction to pin MOSI
3. The Serial Programming instructions will not work if the communication is out of synchronization. When in sync. the second byte (0x53), 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 0x53 did not echo back, give $\overline{RESET}$ 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 89 on page 218. The memory page is loaded one byte at a time by supplying the 5 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 7MSB of the address. If polling is not used, the user must wait at least t_{WD_FLASH} before issuing the next page (see Table 97 on page 232).

   Note: If other commands than polling (read) are applied before any write operation (FLASH, EEPROM, Lock Bits, Fuses) is completed, it may 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 97 on page 232). In a chip erased device, no 0xFFs in the data file(s) need to be programmed
6. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO
7. At the end of the programming session, $\overline{RESET}$ can be set high to commence normal operation
8. Power-off sequence (if needed):
   Set $\overline{RESET}$ 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 0xFF. 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 0xFF, 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 0xFF in all locations, programming of addresses that are meant to contain 0xFF, can be skipped. See Table 97 on page 232 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 0xFF. 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 0xFF, but the user should have the following in mind: As a chip-erased device contains 0xFF in all locations, programming of addresses that are meant to contain 0xFF, 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 0xFF, and the user will have to wait at least t_{WD_EEPROM} before programming the next byte. See Table 97 for t_{WD_EEPROM} value.

Table 97. Minimum Wait Delay Before Writing the Next Flash or EEPROM Location

Symbol	Minimum Wait Delay
t_{WD_FUSE}	4.5ms
t_{WD_FLASH}	4.5ms
t_{WD_EEPROM}	9.0ms
t_{WD_ERASE}	9.0ms

Figure 113. Serial Programming Waveforms

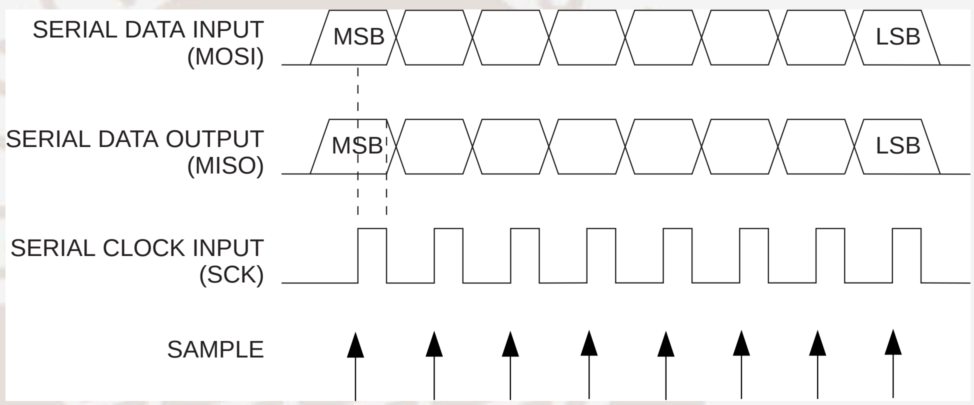


Table 98. Serial Programming Instruction Set

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte 4	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable Serial Programming after RESET goes low
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip Erase EEPROM and Flash
Read Program Memory	0010 **H**000	0000 **aaaa**	**bbbb bbbb**	**oooo oooo**	Read **H** (high or low) data **o** from Program memory at word address **a**:**b**
Load Program Memory Page	0100 **H**000	0000 xxxx	xxx**b bbbb**	**iiii iiii**	Write **H** (high or low) data **i** to Program memory page at word address **b**. Data Low byte must be loaded before Data High byte is applied within the same address
Write Program Memory Page	0100 1100	0000 **aaaa**	**bbb**x xxxx	xxxx xxxx	Write Program memory Page at address **a**:**b**
Read EEPROM Memory	1010 0000	00xx xxx**a**	**bbbb bbbb**	**oooo oooo**	Read data **o** from EEPROM memory at address **a**:**b**
Write EEPROM Memory	1100 0000	00xx xxx**a**	**bbbb bbbb**	**iiii iiii**	Write data **i** to EEPROM memory at address **a**:**b**
Read Lock Bits	0101 1000	0000 0000	xxxx xxxx	xx**oo oooo**	Read Lock Bits. "0" = programmed, "1" = unprogrammed. See Table 85 on page 215 for details
Write Lock Bits	1010 1100	111x xxxx	xxxx xxxx	11**ii iiii**	Write Lock Bits. Set bits = "0" to program Lock Bits. See Table 85 on page 215 for details
Read Signature Byte	0011 0000	00xx xxxx	xxxx xx**bb**	**oooo oooo**	Read Signature Byte **o** at address **b**
Write Fuse Bits	1010 1100	1010 0000	xxxx xxxx	**iiii iiii**	Set bits = "0" to program, "1" to unprogram. See Table 88 on page 217 for details
Write Fuse High Bits	1010 1100	1010 1000	xxxx xxxx	**iiii iiii**	Set bits = "0" to program, "1" to unprogram. See Table 87 on page 216 for details
Read Fuse Bits	0101 0000	0000 0000	xxxx xxxx	**oooo oooo**	Read Fuse Bits. "0" = programmed, "1" = unprogrammed. See Table 88 on page 217 for details
Read Fuse High Bits	0101 1000	0000 1000	xxxx xxxx	**oooo oooo**	Read Fuse high bits. "0" = programmed, "1" = unprogrammed. See Table 87 on page 216 for details
Read Calibration Byte	0011 1000	00xx xxxx	0000 00**bb**	**oooo oooo**	Read Calibration Byte

Note: **a** = address high bits
b = address low bits
H = 0 – Low byte, 1 – High byte
o = data out
i = data in
x = don't care

SPI Serial Programming Characteristics

For characteristics of the SPI module, see “SPI Timing Characteristics” on page 239.

Electrical Characteristics – TA = -40°C to 85°C

Note: 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.

Absolute Maximum Ratings*

Operating Temperature................................. -55°C to +125°C

Storage Temperature .... -65°C to +150°C

Voltage on any Pin except $\overline{RESET}$ with respect to Ground ....-0.5V to V_{CC}+0.5V

Voltage on $\overline{RESET}$ with respect to Ground......-0.5V to +13.0V

Maximum Operating Voltage 6.0V

DC Current per I/O Pin 40.0mA

DC Current V_{CC} and GND Pins................................ 300.0mA

*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	Typ	Max	Units
V_{IL}	Input Low Voltage except XTAL1 and $\overline{RESET}$ pins	V_{CC} = 2.7V - 5.5V	-0.5		0.2 V_{CC}(1)	V
V_{IH}	Input High Voltage except XTAL1 and $\overline{RESET}$ pins	V_{CC} = 2.7V - 5.5V	0.6V_{CC}(2)		V_{CC} + 0.5	
V_{IL1}	Input Low Voltage XTAL1 pin	V_{CC} = 2.7V - 5.5V	-0.5		0.1V_{CC}(1)	
V_{IH1}	Input High Voltage XTAL 1 pin	V_{CC} = 2.7V - 5.5V	0.8V_{CC}(2)		V_{CC} + 0.5	
V_{IL2}	Input Low Voltage $\overline{RESET}$ pin	V_{CC} = 2.7V - 5.5V	-0.5		0.2 V_{CC}	
V_{IH2}	Input High Voltage $\overline{RESET}$ pin	V_{CC} = 2.7V - 5.5V	0.9V_{CC}(2)		V_{CC} + 0.5	
V_{IL3}	Input Low Voltage $\overline{RESET}$ pin as I/O	V_{CC} = 2.7V - 5.5V	-0.5		0.2V_{CC}	
V_{IH3}	Input High Voltage $\overline{RESET}$ pin as I/O	V_{CC} = 2.7V - 5.5V	0.6V_{CC}(2) 0.7V_{CC}(2)		V_{CC} + 0.5	
V_{OL}	Output Low Voltage(3) (Ports B,C,D)	I_{OL} = 20mA, V_{CC} = 5V I_{OL} = 10mA, V_{CC} = 3V			0.9 0.6	
V_{OH}	Output High Voltage(4) (Ports B,C,D)	I_{OH} = -20mA, V_{CC} = 5V I_{OH} = -10mA, V_{CC} = 3V	4.2 2.2			
I_{IL}	Input Leakage Current I/O Pin	Vcc = 5.5V, pin low (absolute value)			1	µA
I_{IH}	Input Leakage Current I/O Pin	Vcc = 5.5V, pin high (absolute value)			1	
R_{RST}	Reset Pull-up Resistor		30		80	kΩ

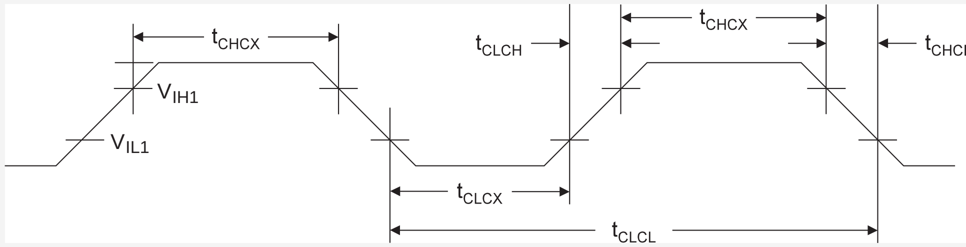
T_A = -40°C to +85°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted) (Continued)

Symbol	Parameter	Condition	Min	Typ	Max	Units
R_{pu}	I/O Pin Pull-up Resistor		20		50	kΩ
I_{CC}	Power Supply Current	Active 4MHz, V_{CC} = 3V (ATmega8L)		3	5	mA
		Active 8MHz, V_{CC} = 5V (ATmega8)		11	15	
		Idle 4MHz, V_{CC} = 3V (ATmega8L)		1	2	
		Idle 8MHz, V_{CC} = 5V (ATmega8)		4.5	7	
	Power-down mode(5)	WDT enabled, V_{CC} = 3V		< 22	28	µA
		WDT disabled, V_{CC} = 3V		< 1	3	
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} = 5.0V		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 (20mA at Vcc = 5V, 10mA at Vcc = 3V) under steady state conditions (non-transient), the following must be observed:
   PDIP, TQFP, and QFN/MLF Package:
   1] The sum of all IOL, for all ports, should not exceed 300mA.
   2] The sum of all IOL, for ports C0 - C5 should not exceed 100mA.
   3] The sum of all IOL, for ports B0 - B7, C6, D0 - D7 and XTAL2, should not exceed 200mA.
   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 (20mA at Vcc = 5V, 10mA at Vcc = 3V) under steady state conditions (non-transient), the following must be observed:
   PDIP, TQFP, and QFN/MLF Package:
   1] The sum of all IOH, for all ports, should not exceed 300mA.
   2] The sum of all IOH, for port C0 - C5, should not exceed 100mA.
   3] The sum of all IOH, for ports B0 - B7, C6, D0 - D7 and XTAL2, should not exceed 200mA.
   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 114. External Clock Drive Waveforms



External Clock Drive

Table 99. External Clock Drive

Symbol	Parameter	V_{CC} = 2.7V to 5.5V		V_{CC} = 4.5V to 5.5V		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	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	μs
t_{CHCL}	Fall Time		1.6		0.5	
Δt_{CLCL}	Change in period from one clock cycle to the next		2		2	%

Table 100. External RC Oscillator, Typical Frequencies

R [kΩ][(1)]	C [pF]	f[(2)]
33	22	650kHz
10	22	2.0MHz

Notes: 1. R should be in the range 3kΩ - 100kΩ, and C should be at least 20pF. The C values given in the table includes pin capacitance. This will vary with package type

2. The frequency will vary with package type and board layout

Two-wire Serial Interface Characteristics

Table 101 describes the requirements for devices connected to the Two-wire Serial Bus. The ATmega8 Two-wire Serial Interface meets or exceeds these requirements under the noted conditions.

Timing symbols refer to Figure 115 on page 239.

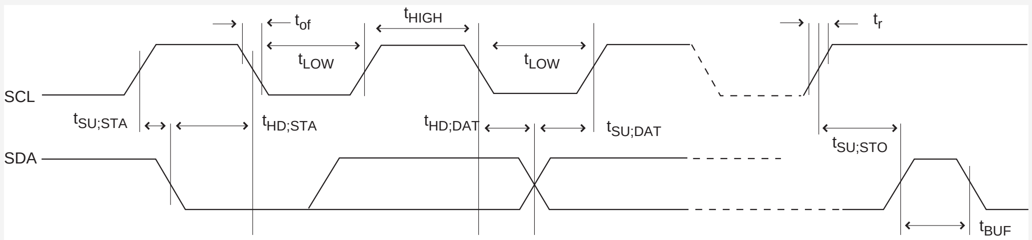
Table 101. Two-wire Serial Bus Requirements

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low-voltage		-0.5	$0.3V_{CC}$	V
V_{IH}	Input High-voltage		$0.7V_{CC}$	$V_{CC} + 0.5$	
V_{hys}(1)	Hysteresis of Schmitt Trigger Inputs		$0.05V_{CC}$(2)	–	
V_{OL}(1)	Output Low-voltage	3mA sink Current	0	0.4	
t_r(1)	Rise Time for both SDA and SCL		$20 + 0.1C_b$(3)(2)	300	ns
t_{of}(1)	Output Fall Time from V_{IHmin} to V_{ILmax}	$10pF < C_b < 400pF$(3)	$20 + 0.1C_b$(3)(2)	250	
t_{SP}(1)	Spikes Suppressed by Input Filter		0	50(2)	
I_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	$f_{SCL} \leq 100kHz$	$\frac{V_{CC} - 0.4V}{3mA}$	$\frac{1000ns}{C_b}$	Ω
		$f_{SCL} > 100kHz$	$\frac{V_{CC} - 0.4V}{3mA}$	$\frac{300ns}{C_b}$	
$t_{HD;STA}$	Hold Time (repeated) START Condition	$f_{SCL} \leq 100kHz$	4.0	–	µs
		$f_{SCL} > 100kHz$	0.6	–	
t_{LOW}	Low Period of the SCL Clock	$f_{SCL} \leq 100kHz$(6)	4.7	–	
		$f_{SCL} > 100kHz$(7)	1.3	–	
t_{HIGH}	High period of the SCL clock	$f_{SCL} \leq 100kHz$	4.0	–	
		$f_{SCL} > 100kHz$	0.6	–	
$t_{SU;STA}$	Set-up time for a repeated START condition	$f_{SCL} \leq 100kHz$	4.7	–	
		$f_{SCL} > 100kHz$	0.6	–	
$t_{HD;DAT}$	Data hold time	$f_{SCL} \leq 100kHz$	0	3.45	
		$f_{SCL} > 100kHz$	0	0.9	
$t_{SU;DAT}$	Data setup time	$f_{SCL} \leq 100kHz$	250	–	ns
		$f_{SCL} > 100kHz$	100	–	
$t_{SU;STO}$	Setup time for STOP condition	$f_{SCL} \leq 100kHz$	4.0	–	µs
		$f_{SCL} > 100kHz$	0.6	–	
t_{BUF}	Bus free time between a STOP and START condition	$f_{SCL} \leq 100kHz$	4.7	–	
		$f_{SCL} > 100kHz$	1.3	–	

Notes:
1. In ATmega8, this parameter is characterized and not 100% tested
2. Required only for $f_{SCL} > 100kHz$
3. C_b = capacitance of one bus line in pF
4. f_{CK} = CPU clock frequency

5. This requirement applies to all ATmega8 Two-wire Serial Interface operation. Other devices connected to the Two-wire Serial Bus need only obey the general f_{SCL} requirement
6. The actual low period generated by the ATmega8 Two-wire Serial Interface is $(1/f_{SCL} - 2/f_{CK})$, thus f_{CK} must be greater than 6MHz for the low time requirement to be strictly met at f_{SCL} = 100kHz
7. The actual low period generated by the ATmega8 Two-wire Serial Interface is $(1/f_{SCL} - 2/f_{CK})$, thus the low time requirement will not be strictly met for f_{SCL} > 308kHz when f_{CK} = 8MHz. Still, ATmega8 devices connected to the bus may communicate at full speed (400kHz) with other ATmega8 devices, as well as any other device with a proper t_{LOW} acceptance margin

Figure 115. Two-wire Serial Bus Timing



SPI Timing Characteristics

See Figure 116 on page 240 and Figure 117 on page 240 for details.

Table 102. SPI Timing Parameters

	Description	Mode	Min	Typ	Max	
1	SCK period	Master		See Table 50 on page 126		ns
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		$0.5 \cdot t_{SCK}$		
7	SCK to out	Master		10		
8	SCK to out high	Master		10		
9	$\overline{SS}$ low to out	Slave		15		
10	SCK period	Slave	$4 \cdot t_{ck}$			
11	SCK high/low[(1)]	Slave	$2 \cdot t_{ck}$			
12	Rise/Fall time	Slave			1600	
13	Setup	Slave	10			
14	Hold	Slave	10			
15	SCK to out	Slave		15		
16	SCK to $\overline{SS}$ high	Slave	20			
17	$\overline{SS}$ high to tri-state	Slave		10		
18	$\overline{SS}$ low to SCK	Salve	$2 \cdot t_{ck}$			

Note: 1. In SPI Programming mode the minimum SCK high/low period is:
- $2t_{CLCL}$ for f_{CK} < 12MHz
- $3t_{CLCL}$ for f_{CK} > 12MHz

Figure 116. SPI interface timing requirements (Master Mode)

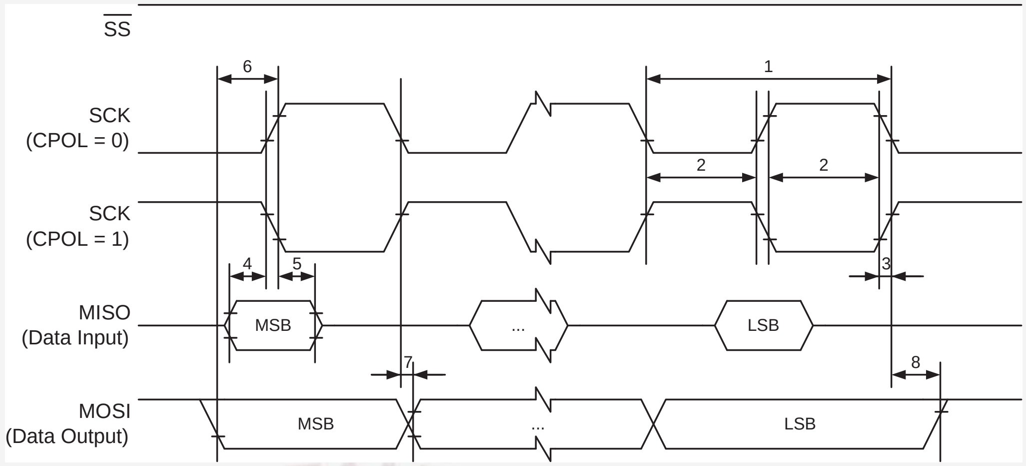
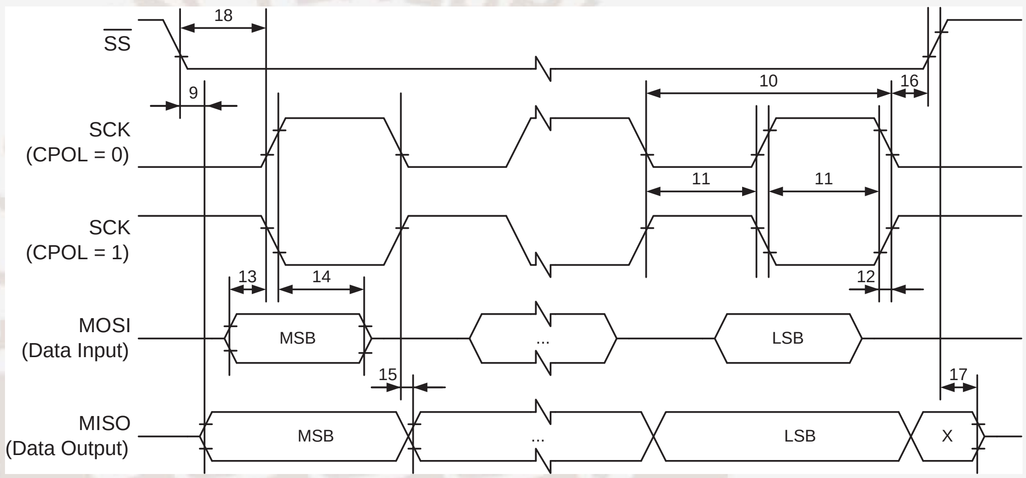


Figure 117. SPI interface timing requirements (Slave Mode)



ADC Characteristics

Table 103. ADC Characteristics

Symbol	Parameter	Condition	Min[1]	Typ[1]	Max[1]	Units
	Resolution	Single Ended Conversion		10		Bits
	Absolute accuracy (including INL, DNL, Quantization Error, Gain, and Offset Error)	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 200kHz		1.75		
		Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 1MHz		3		
	Integral Non-linearity (INL)	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 200kHz		0.75		LSB
	Differential Non-linearity (DNL)	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 200kHz		0.5		
	Gain Error	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 200kHz		1		
	Offset Error	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 200kHz		1		
	Conversion Time[4]	Free Running Conversion	13		260	µs
	Clock Frequency		50		1000	kHz
AV_{CC}	Analog Supply Voltage		V_{CC} - 0.3[2]		V_{CC} + 0.3[3]	
V_{REF}	Reference Voltage		2.0		AV_{CC}	V
V_{IN}	Input voltage		GND		V_{REF}	
	Input bandwidth			38.5		kHz
V_{INT}	Internal Voltage Reference		2.3	2.56	2.9	V
R_{REF}	Reference Input Resistance			32		kΩ
R_{AIN}	Analog Input Resistance		55	100		MΩ

Notes:
1. Values are guidelines only
2. Minimum for AV_{CC} is 2.7V
3. Maximum for AV_{CC} is 5.5V
4. Maximum conversion time is 1/50kHz × 25 = 0.5ms

Electrical Characteristics – TA = -40°C to 105°C

Note: Typical values contained in this data sheet 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.

Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{RESET}$ with respect to Ground	-0.5V to V_{CC}+0.5V
Voltage on $\overline{RESET}$ with respect to Ground	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0 mA
DC Current V_{CC} and GND Pins	200.0 mA

*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 105°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{IL}	Input Low Voltage	Except XTAL1 pin	-0.5		0.2 V_{CC}[(1)]	V
V_{IL1}	Input Low Voltage	XTAL1 pin, External Clock Selected	-0.5		0.1 V_{CC}[(1)]	V
V_{IH}	Input High Voltage	Except XTAL1 and $\overline{RESET}$ pins	0.6 V_{CC}[(2)]		V_{CC} + 0.5	V
V_{IH1}	Input High Voltage	XTAL1 pin, External Clock Selected	0.8 V_{CC}[(2)]		V_{CC} + 0.5	V
V_{IH2}	Input High Voltage	$\overline{RESET}$ pin	0.9 V_{CC}[(2)]		V_{CC} + 0.5	V
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			0.8 0.6	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	4.0 2.2			V V
I_{IL}	Input Leakage Current I/O Pin	Vcc = 5.5V, pin low (absolute value)			3	µA
I_{IH}	Input Leakage Current I/O Pin	Vcc = 5.5V, pin high (absolute value)			3	µA
R_{RST}	Reset Pull-up Resistor		30		80	kΩ
R_{pu}	I/O Pin Pull-up Resistor		20		50	kΩ

DC Characteristics

T_A = -40°C to 105°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted) (Continued)

Symbol	Parameter	Condition	Min	Typ	Max	Units
I_{CC}	Power Supply Current	Active 4 MHz, V_{CC} = 3V (ATmega8L)			6	mA
		Active 8 MHz, V_{CC} = 5V (ATmega8)			15	mA
		Idle 4 MHz, V_{CC} = 3V (ATmega8L)			3	mA
		Idle 8 MHz, V_{CC} = 5V (ATmega8)			8	mA
	Power-down mode(5)	WDT enabled, V_{CC} = 3V			35	µA
		WDT disabled, V_{CC} = 3V			6	µA
V_{ACIO}	Analog Comparator Input Offset Voltage	V_{CC} = 5V V_{in} = V_{CC}/2			20	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} = 5.0V		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 (20mA at Vcc = 5V, 10mA 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 400 mA.
   2] The sum of all IOL, for ports C0 - C5 should not exceed 200 mA.
   3] The sum of all IOL, for ports B0 - B7, C6, D0 - D7 and XTAL2, should not exceed 100 mA.
   TQFP and MLF Package:
   1] The sum of all IOL, for all ports, should not exceed 400 mA.
   2] The sum of all IOL, for ports C0 - C5, should not exceed 200 mA.
   3] The sum of all IOL, for ports C6, D0 - D4, should not exceed 300 mA.
   4] The sum of all IOL, for ports B0 - B7, D5 - D7, should not exceed 300 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 (20mA at Vcc = 5V, 10mA 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 400 mA.
   2] The sum of all IOH, for port C0 - C5, should not exceed 100 mA.
   3] The sum of all IOH, for ports B0 - B7, C6, D0 - D7 and XTAL2, should not exceed 100 mA.
   TQFP and MLF Package:
   1] The sum of all IOH, for all ports, should not exceed 400 mA.
   2] The sum of all IOH, for ports C0 - C5, should not exceed 200 mA.
   3] The sum of all IOH, for ports C6, D0 - D4, should not exceed 300 mA.
   4] The sum of all IOH, for ports B0 - B7, D5 - D7, should not exceed 300 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.

ATmega8 Typical Characteristics – TA = -40°C to 85°C

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:

$C_L \times V_{CC} \times f$

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 118. Active Supply Current vs. Frequency (0.1MHz - 1.0MHz)

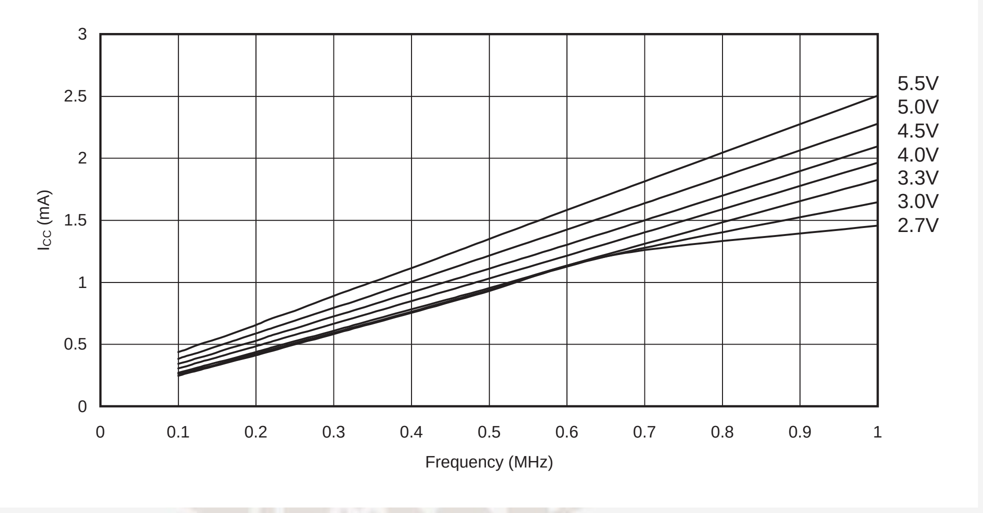


Figure 119. Active Supply Current vs. Frequency (1MHz - 20MHz)

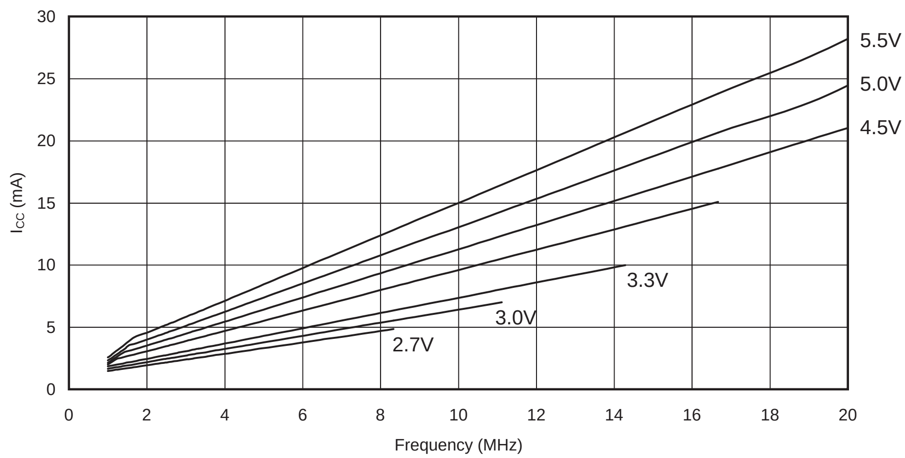


Figure 120. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 8MHz)

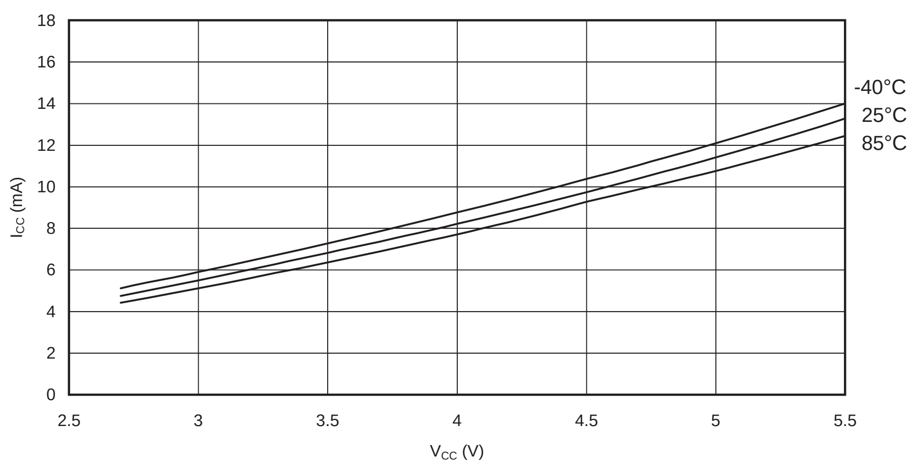


Figure 121. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 4MHz)

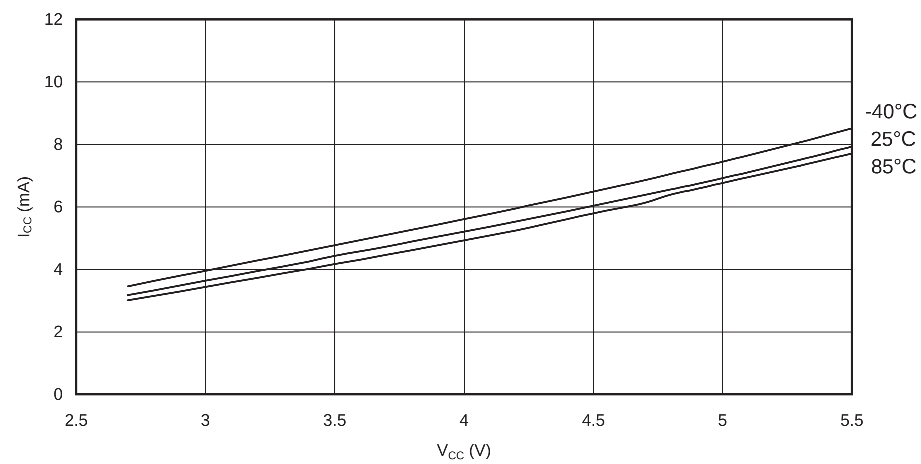


Figure 122. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 2MHz)

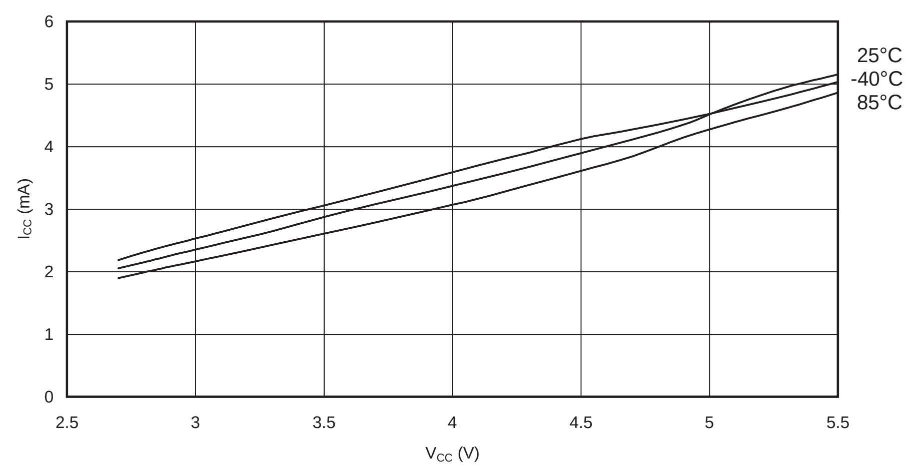


Figure 123. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 1MHz)

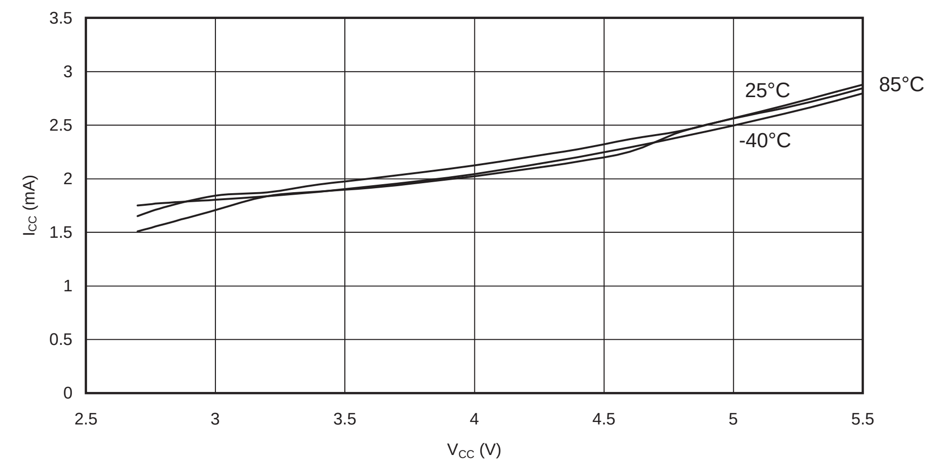
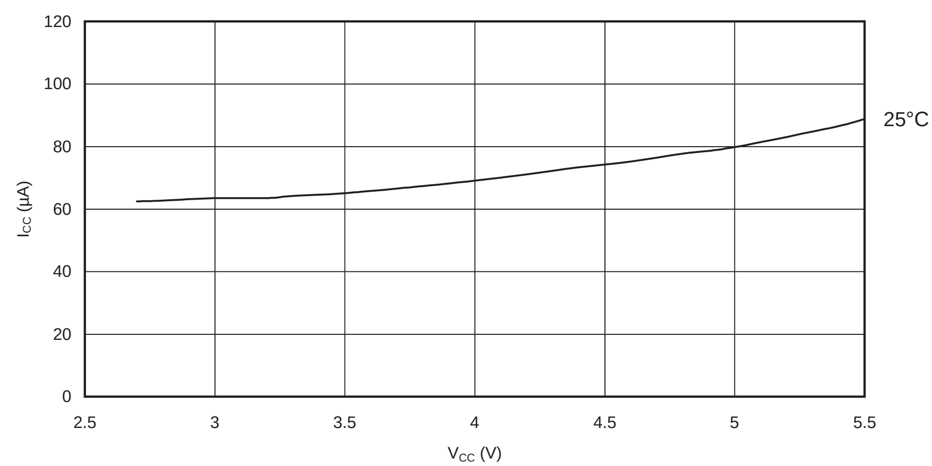


Figure 124. Active Supply Current vs. V_{CC} (32kHz External Oscillator)



Idle Supply Current

Figure 125. Idle Supply Current vs. Frequency (0.1MHz - 1.0MHz)

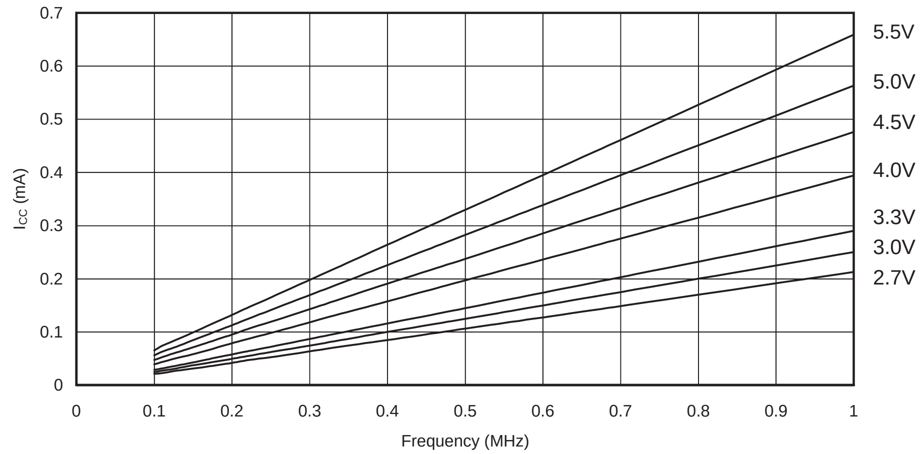


Figure 126. Idle Supply Current vs. Frequency (1MHz - 20MHz)

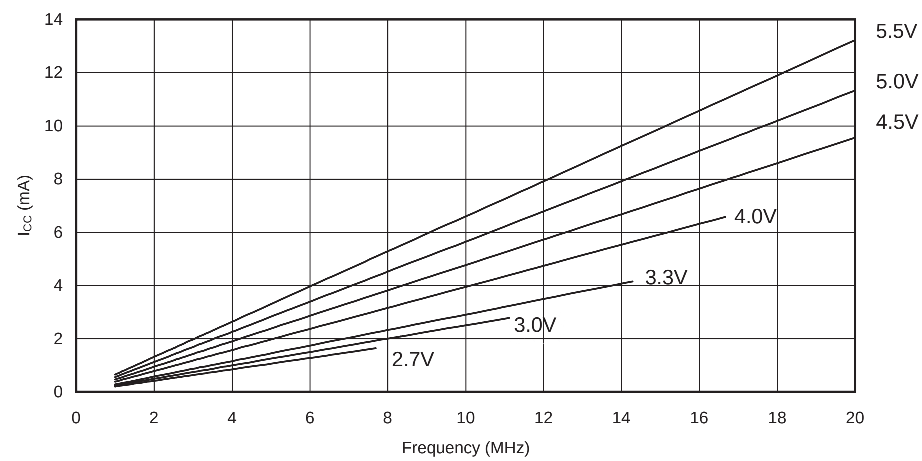


Figure 127. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 8MHz)

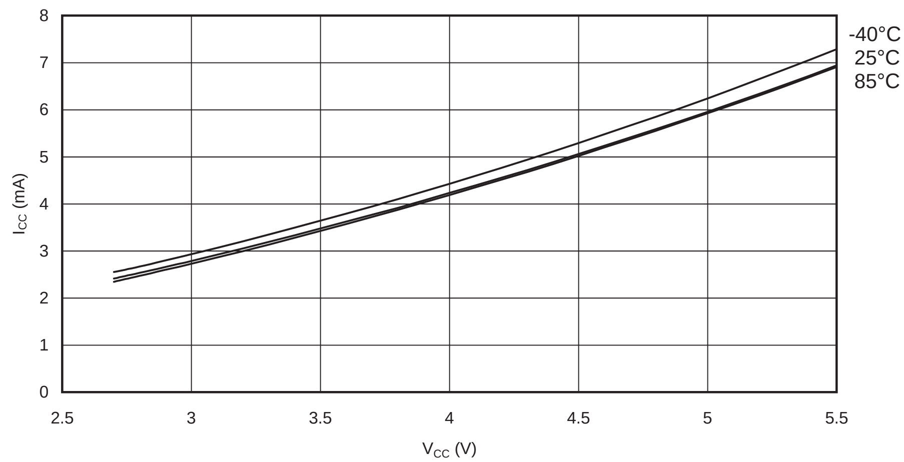


Figure 128. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 4MHz)

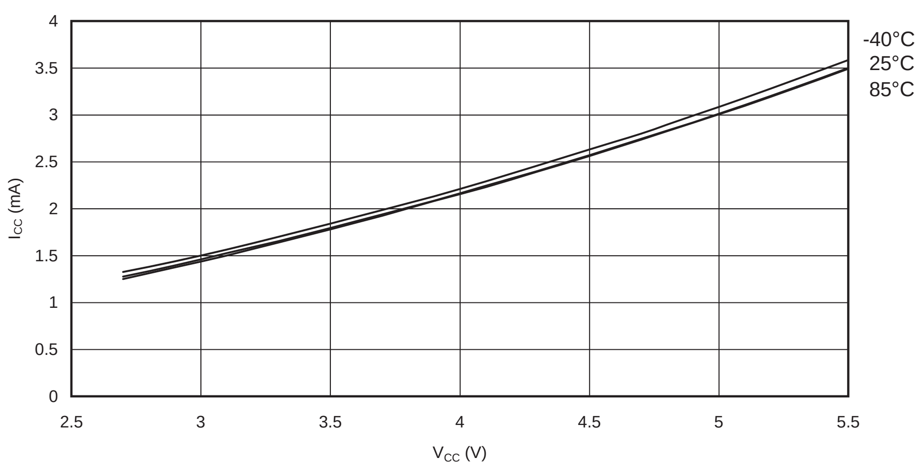


Figure 129. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 2MHz)

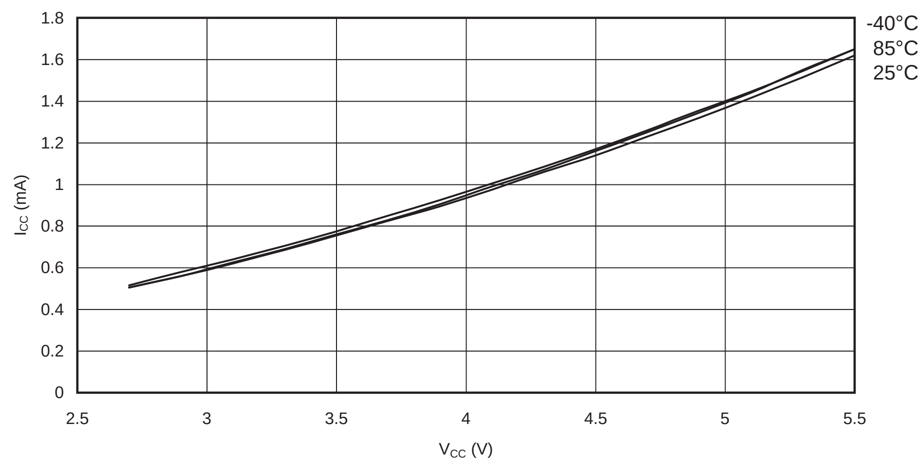


Figure 130. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 1MHz)

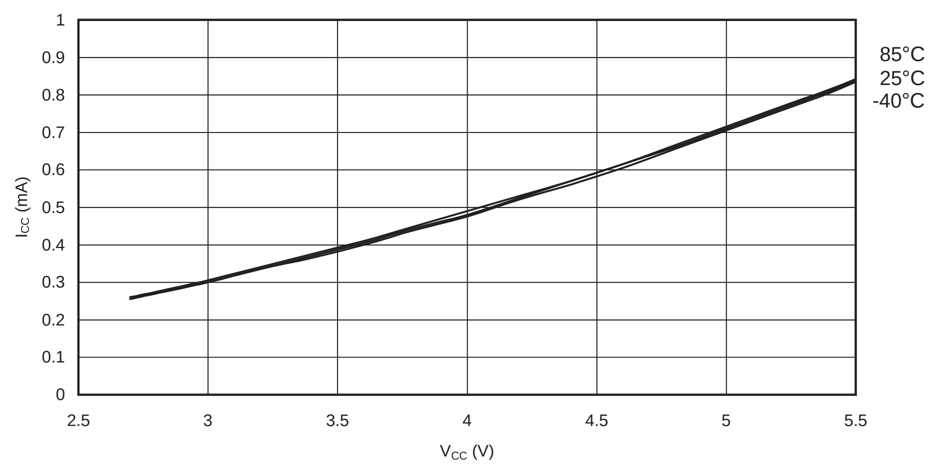
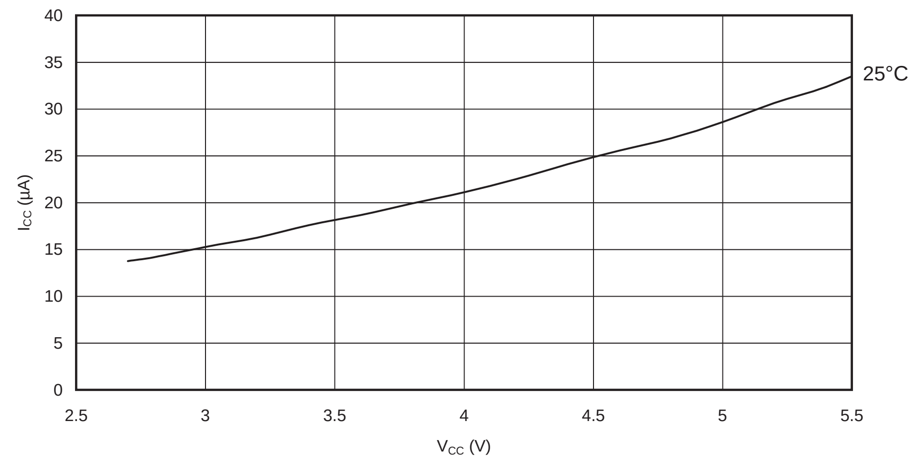


Figure 131. Idle Supply Current vs. V_{CC} (32kHz External Oscillator)



Power-down Supply Current

Figure 132. Power-down Supply Current vs. V_{CC} (Watchdog Timer Disabled)

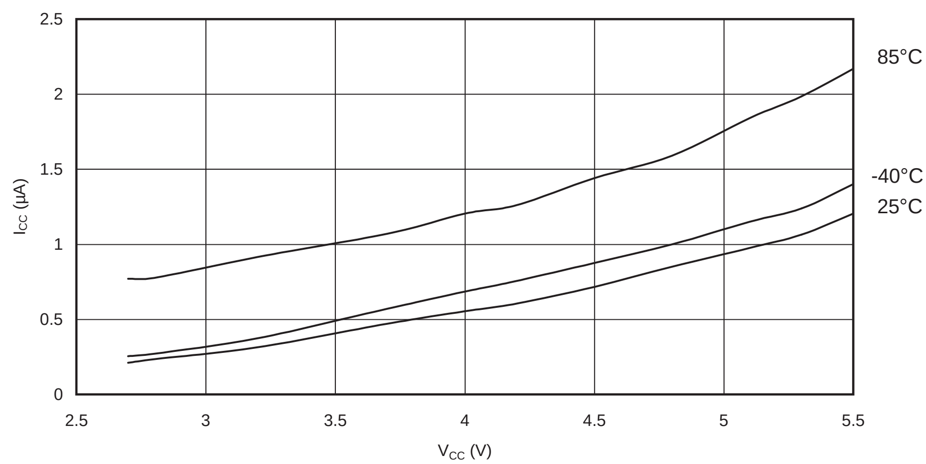
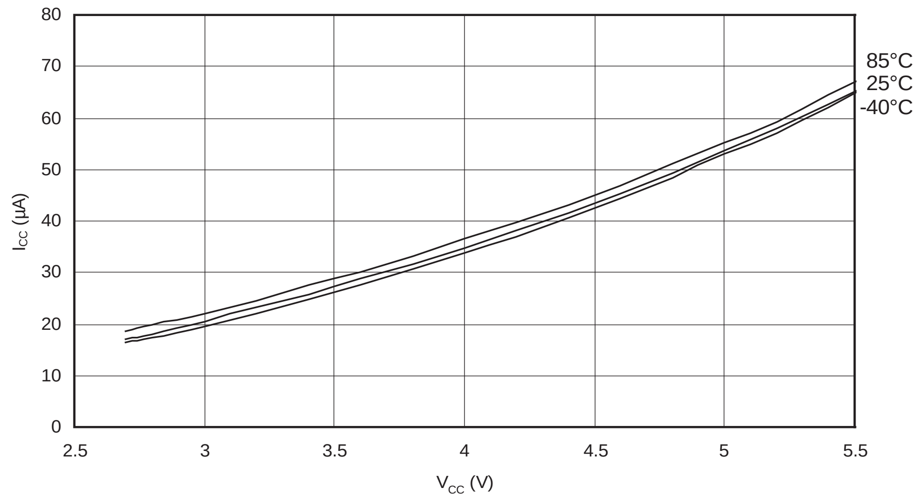
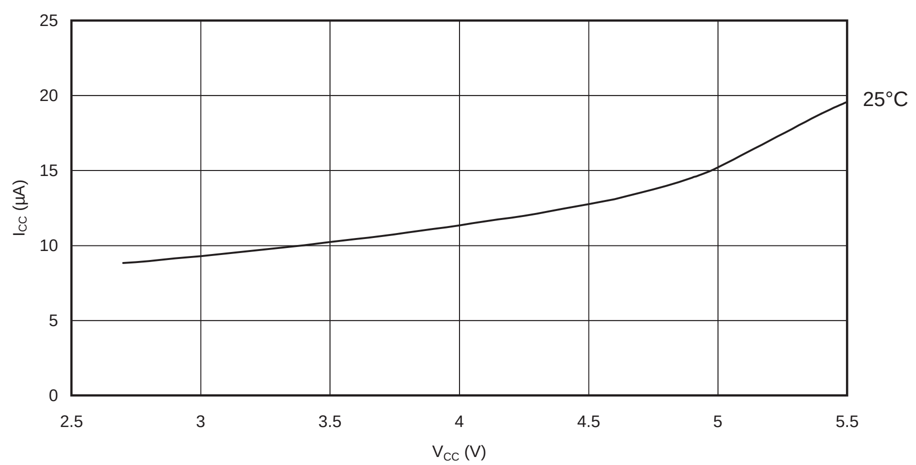


Figure 133. Power-down Supply Current vs. V_{CC} (Watchdog Timer Enabled)



Power-save Supply Current

Figure 134. Power-save Supply Current vs. V_{CC} (Watchdog Timer Disabled)



Standby Supply Current

Figure 135. Standby Supply Current vs. V_{CC} (455kHz Resonator, Watchdog Timer Disabled)

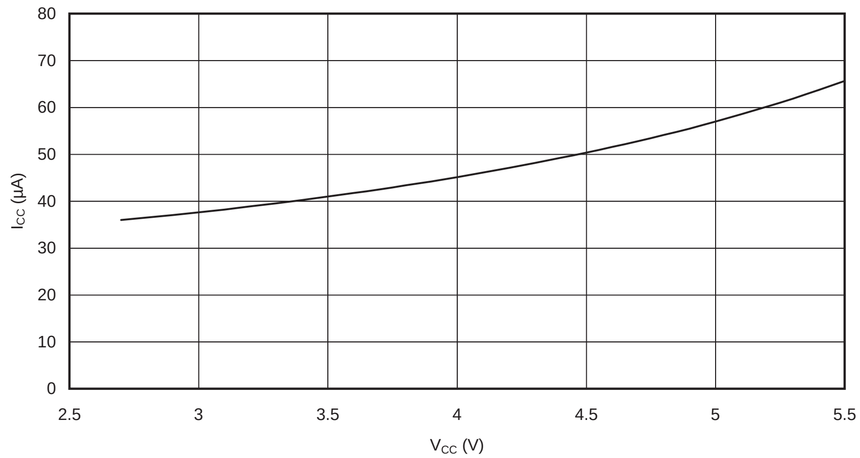


Figure 136. Standby Supply Current vs. V_{CC} (1MHz Resonator, Watchdog Timer Disabled)

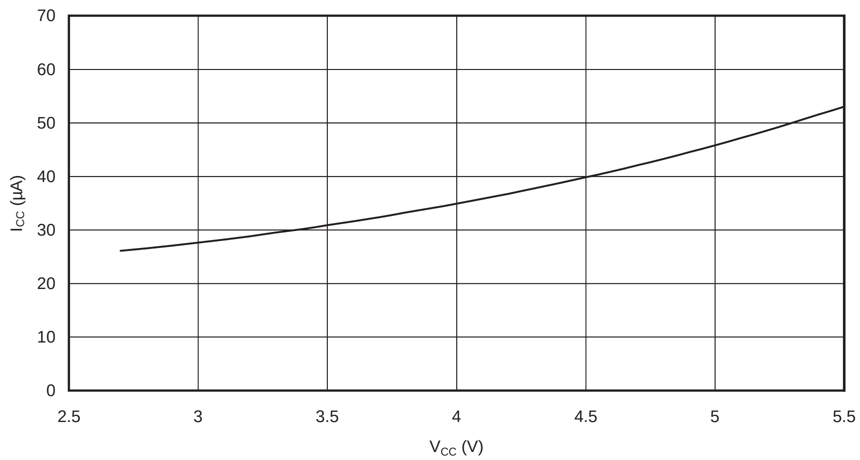


Figure 137. Standby Supply Current vs. V_{CC} (2MHz Resonator, Watchdog Timer Disabled)

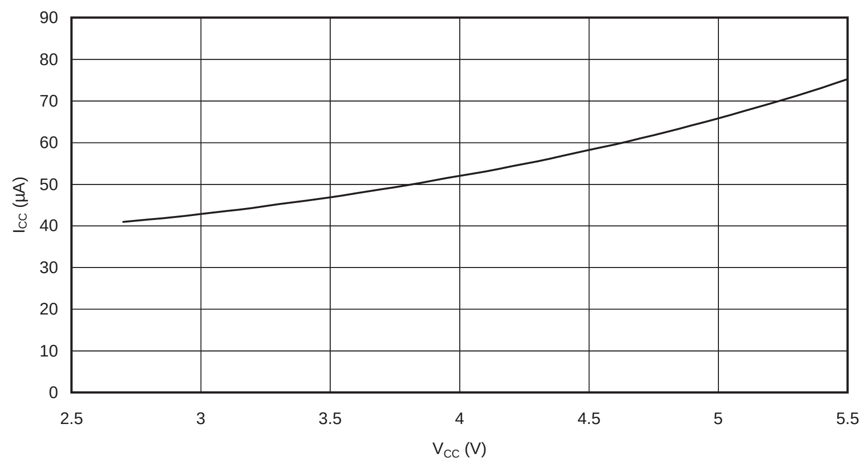


Figure 138. Standby Supply Current vs. V_{CC} (2MHz Xtal, Watchdog Timer Disabled)

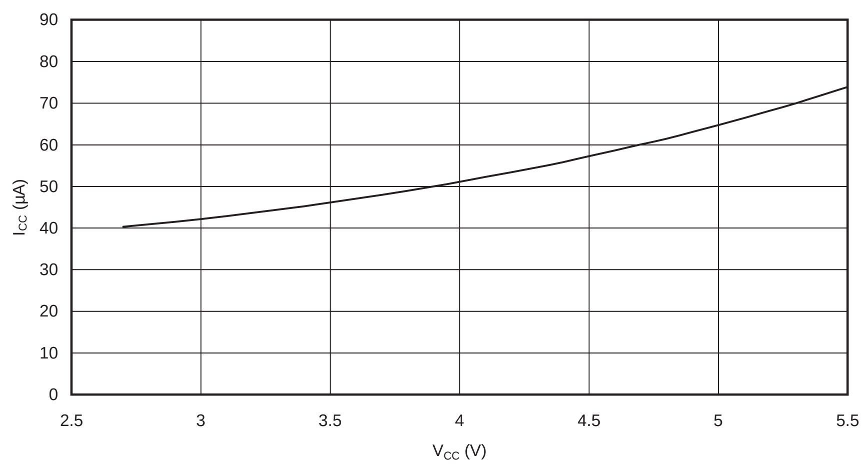


Figure 139. Standby Supply Current vs. V_{CC} (4MHz Resonator, Watchdog Timer Disabled)

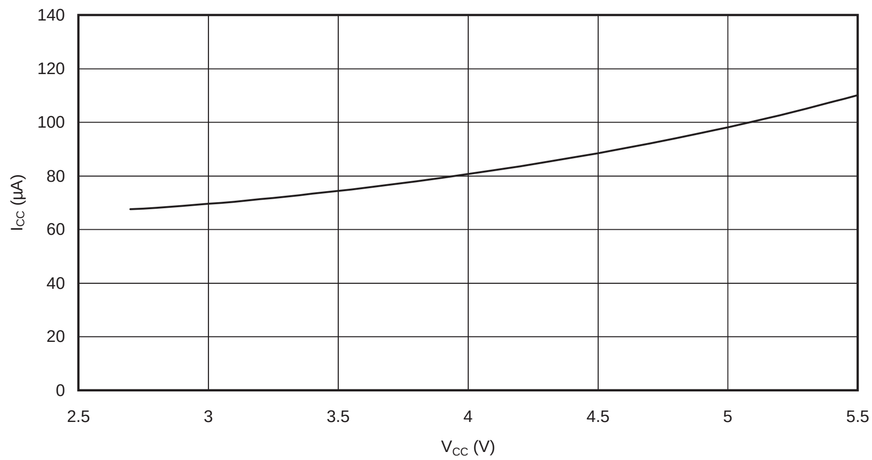


Figure 140. Standby Supply Current vs. V_{CC} (4MHz Xtal, Watchdog Timer Disabled)

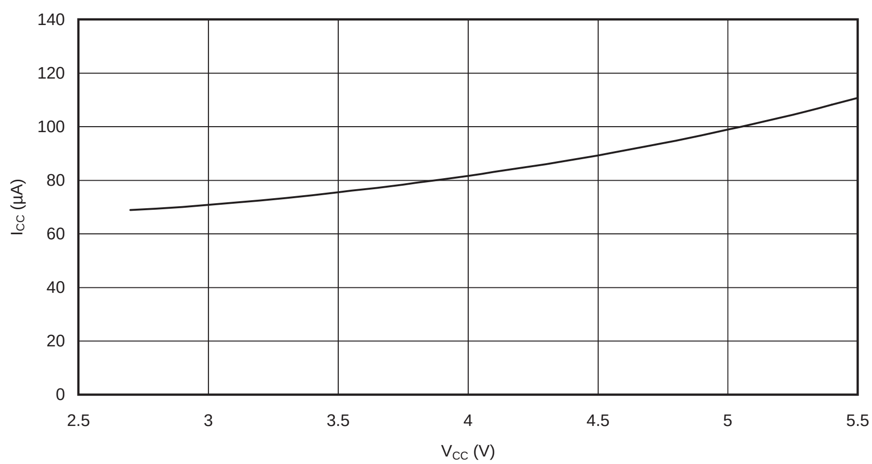


Figure 141. Standby Supply Current vs. V_{CC} (6MHz Resonator, Watchdog Timer Disabled)

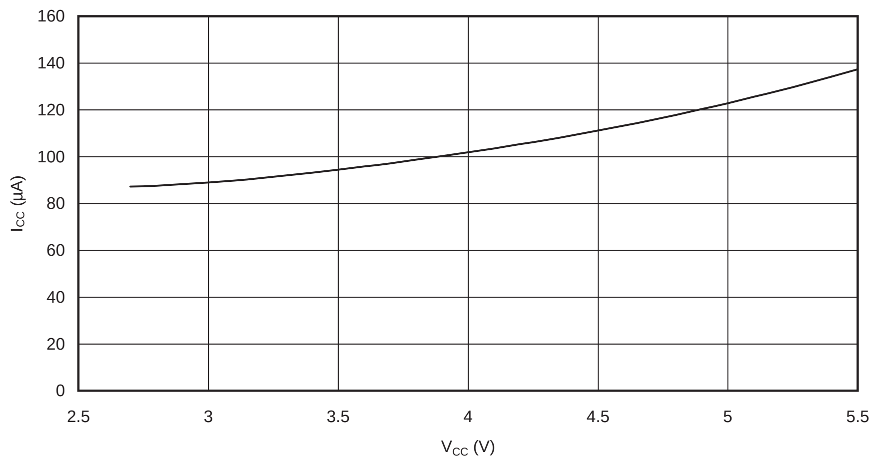
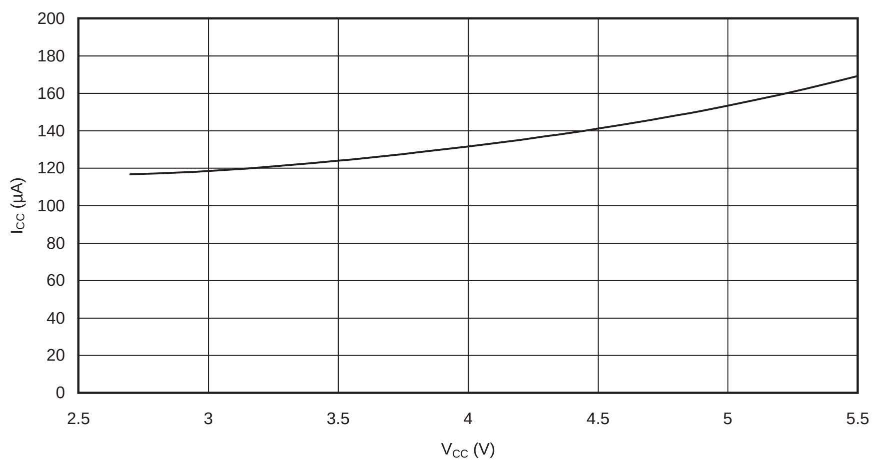


Figure 142. Standby Supply Current vs. V_{CC} (6MHz Xtal, Watchdog Timer Disabled)



Pin Pull-up

Figure 143. I/O Pin Pull-up Resistor Current vs. Input Voltage (V_{CC} = 5V)

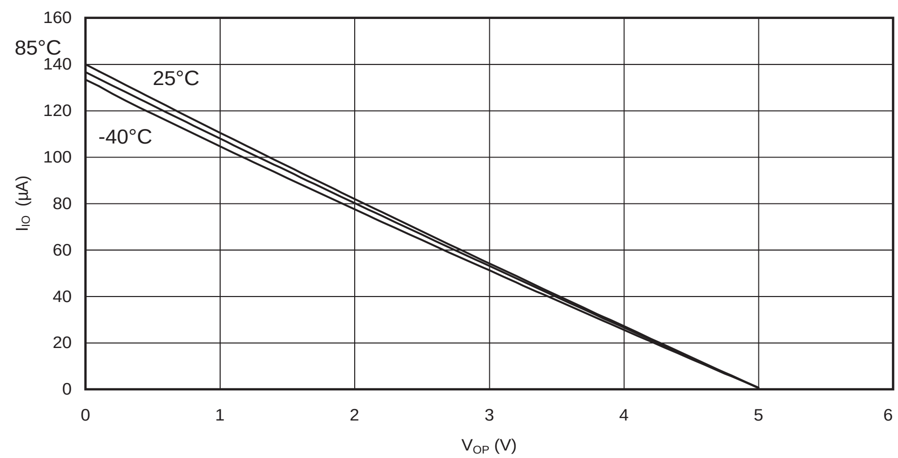


Figure 144. I/O Pin Pull-up Resistor Current vs. Input Voltage (V_{CC} = 2.7V)

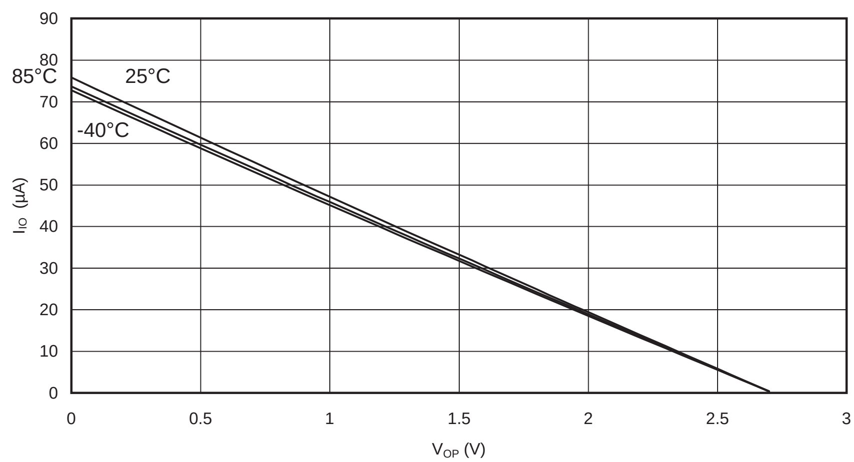


Figure 145. Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 5V)

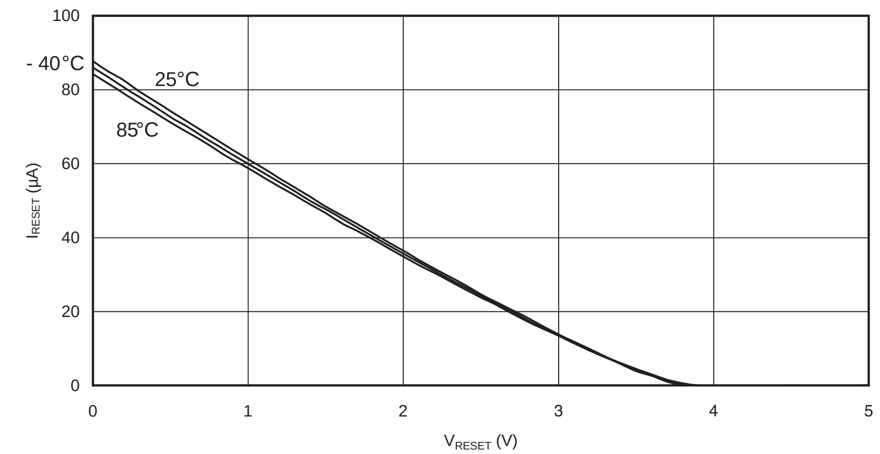
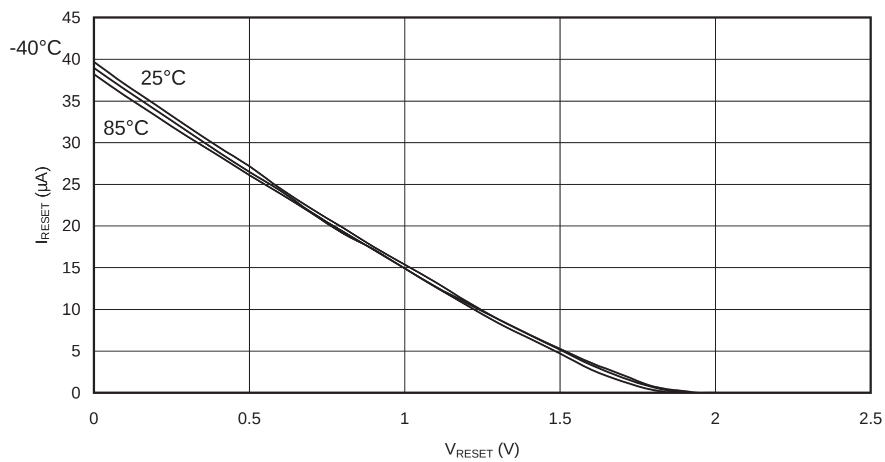


Figure 146. Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 2.7V)



Pin Driver Strength

Figure 147. I/O Pin Source Current vs. Output Voltage (V_{CC} = 5V)

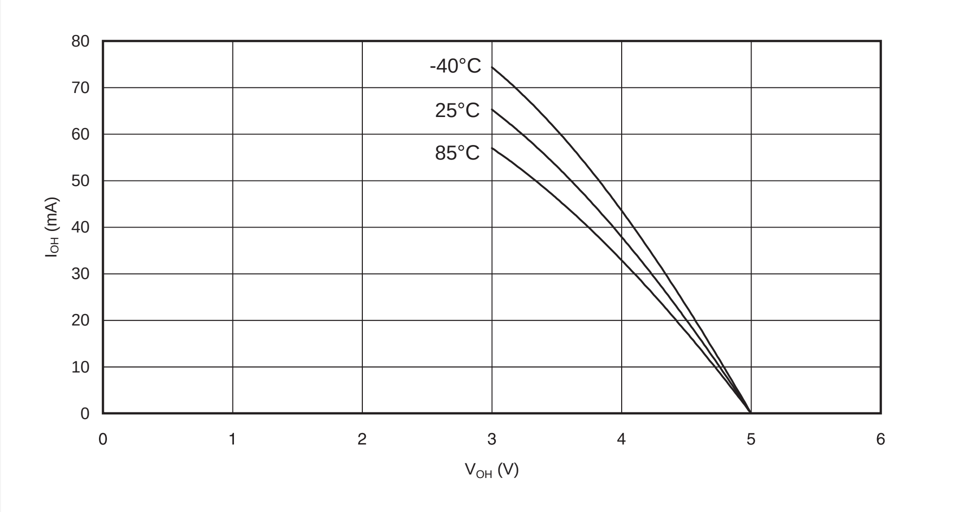


Figure 148. I/O Pin Source Current vs. Output Voltage (V_{CC} = 2.7V)

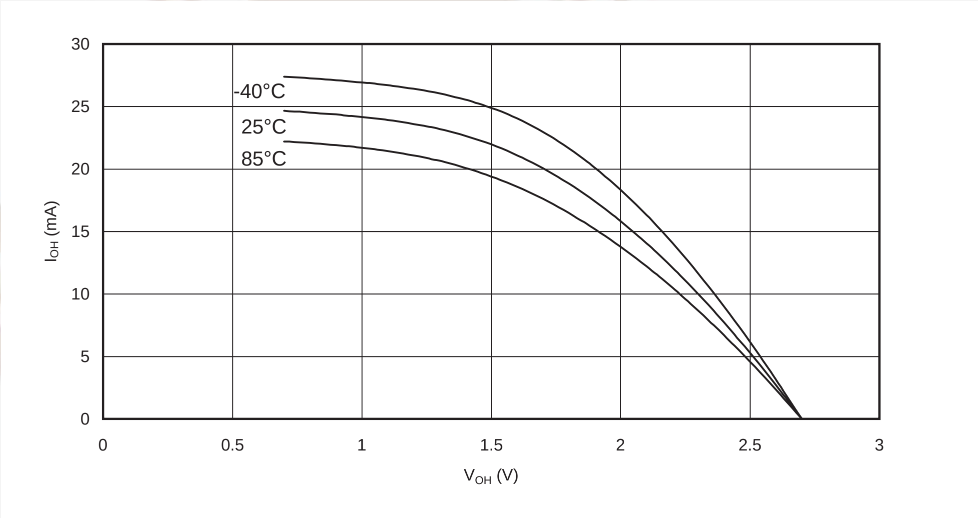


Figure 149. I/O Pin Sink Current vs. Output Voltage (V_{CC} = 5V)

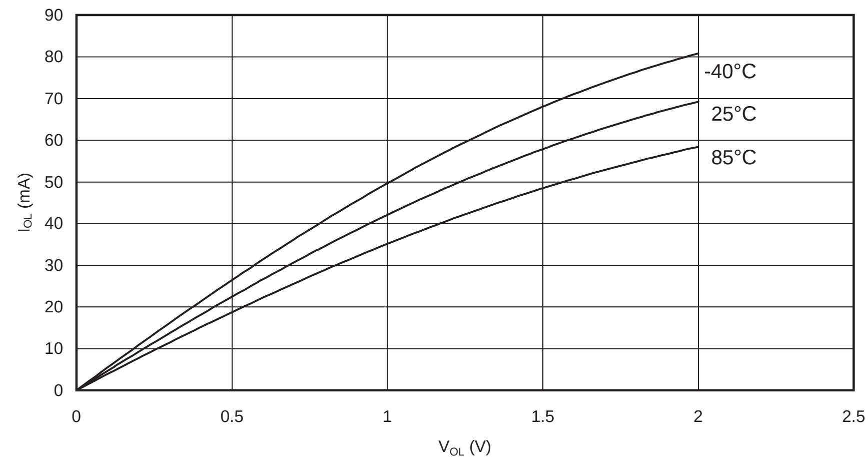


Figure 150. I/O Pin Sink Current vs. Output Voltage (V_{CC} = 2.7V)

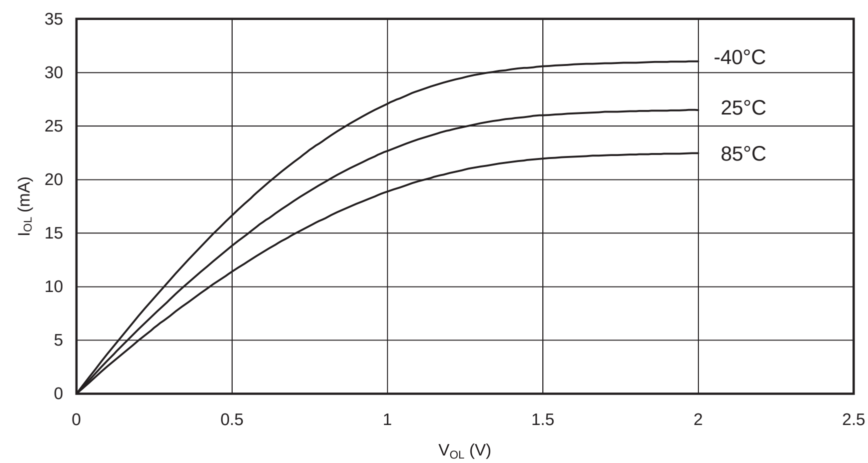


Figure 151. Reset Pin as I/O – Pin Source Current vs. Output Voltage (V_{CC} = 5V)

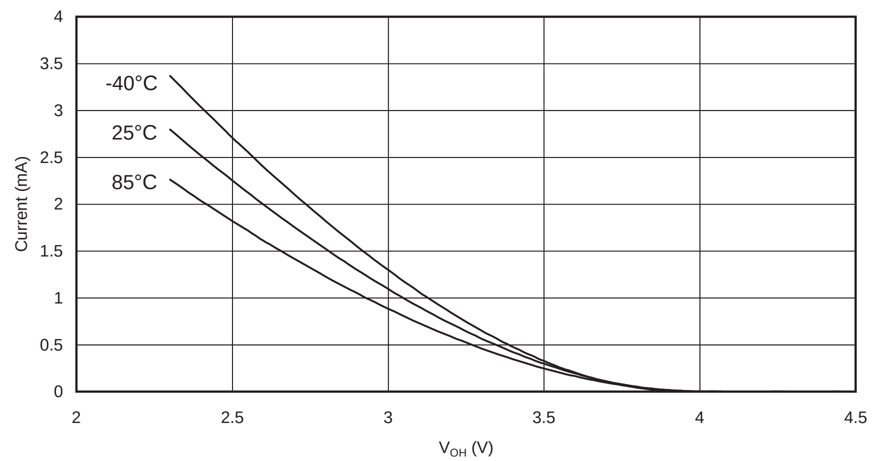


Figure 152. Reset Pin as I/O – Pin Source Current vs. Output Voltage (V_{CC} = 2.7V)

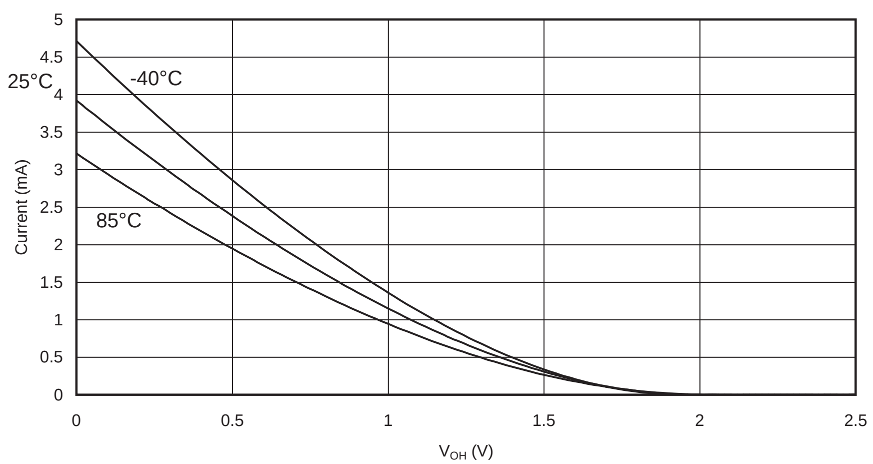


Figure 153. Reset Pin as I/O – Pin Sink Current vs. Output Voltage (V_{CC} = 5V)

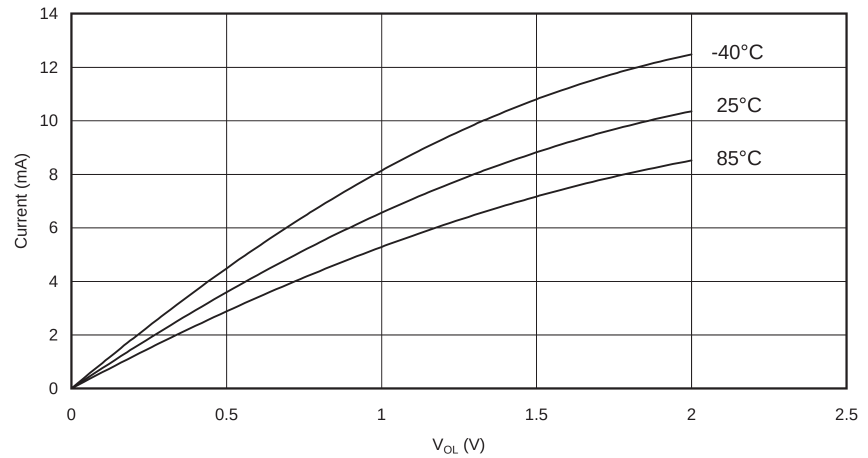
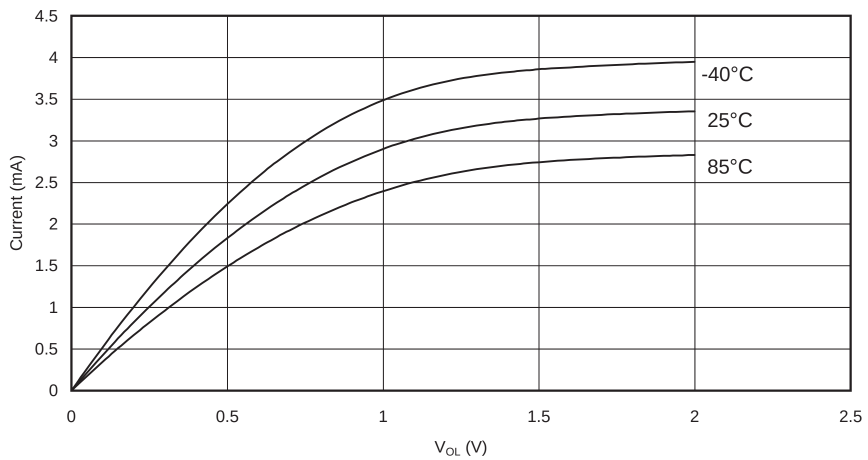


Figure 154. Reset Pin as I/O – Pin Sink Current vs. Output Voltage (V_{CC} = 2.7V)



Pin Thresholds and Hysteresis

Figure 155. I/O Pin Input Threshold Voltage vs. V_{CC} (V_{IH}, I/O Pin Read as “1”)

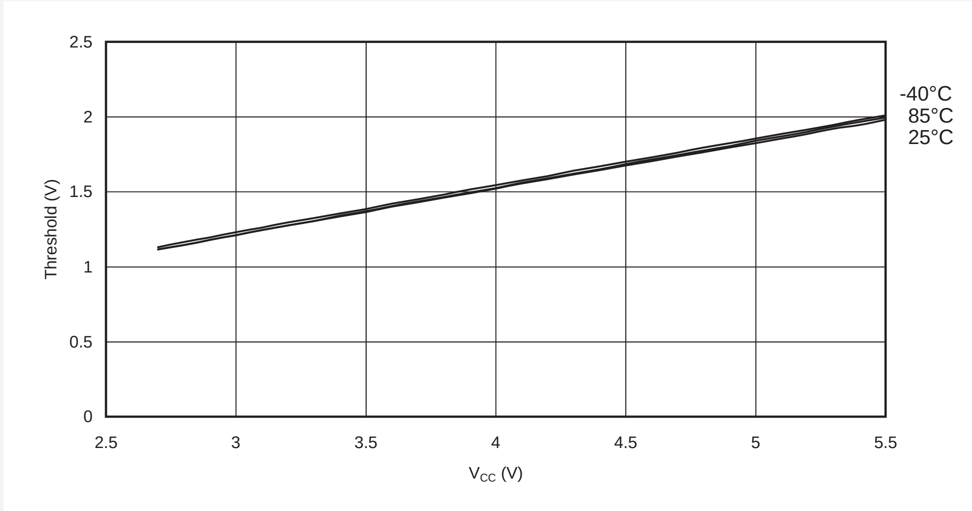


Figure 156. I/O Pin Input Threshold Voltage vs. V_{CC} (V_{IL}, I/O Pin Read as “0”)

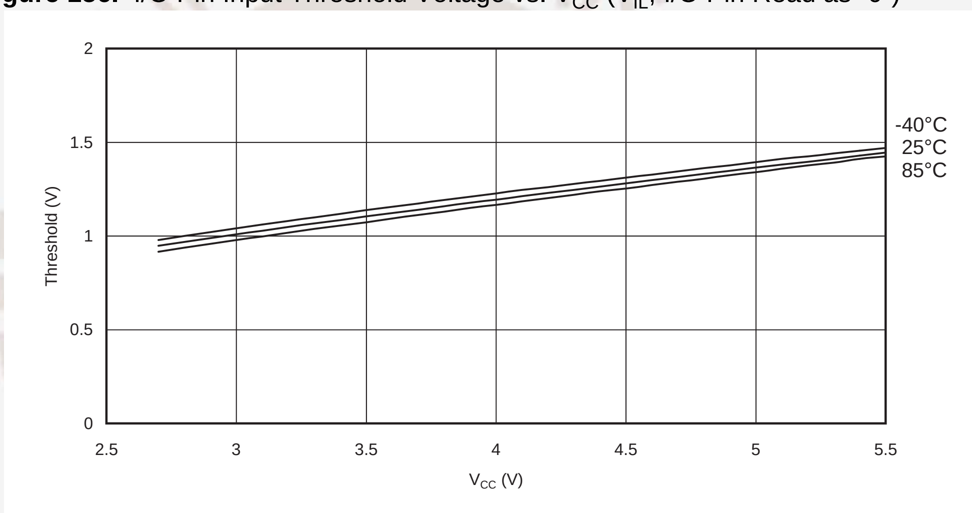


Figure 157. I/O Pin Input Hysteresis vs. V_{CC}

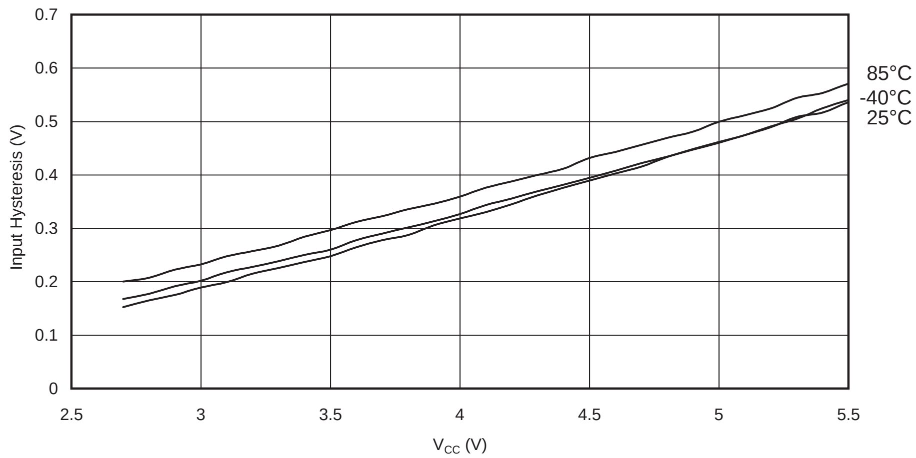


Figure 158. Reset Pin as I/O – Input Threshold Voltage vs. V_{CC} (V_{IH}, I/O Pin Read as "1")

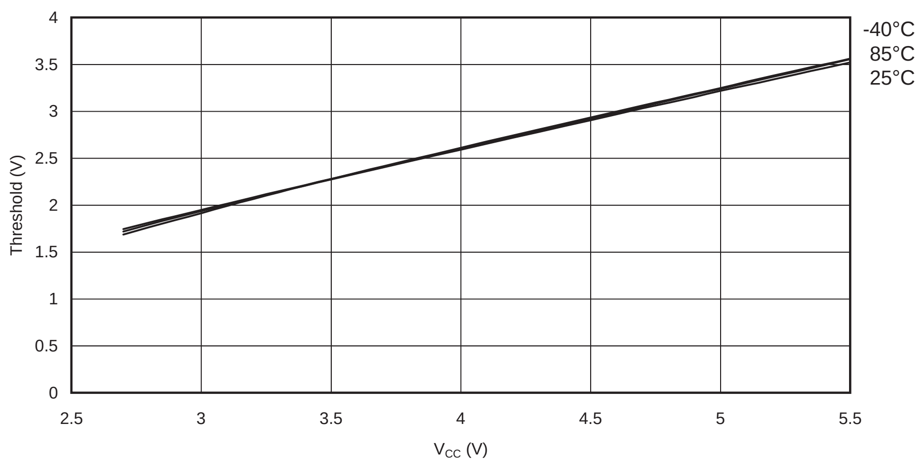


Figure 159. Reset Pin as I/O – Input Threshold Voltage vs. V_{CC} (V_{IL}, I/O Pin Read as "0")

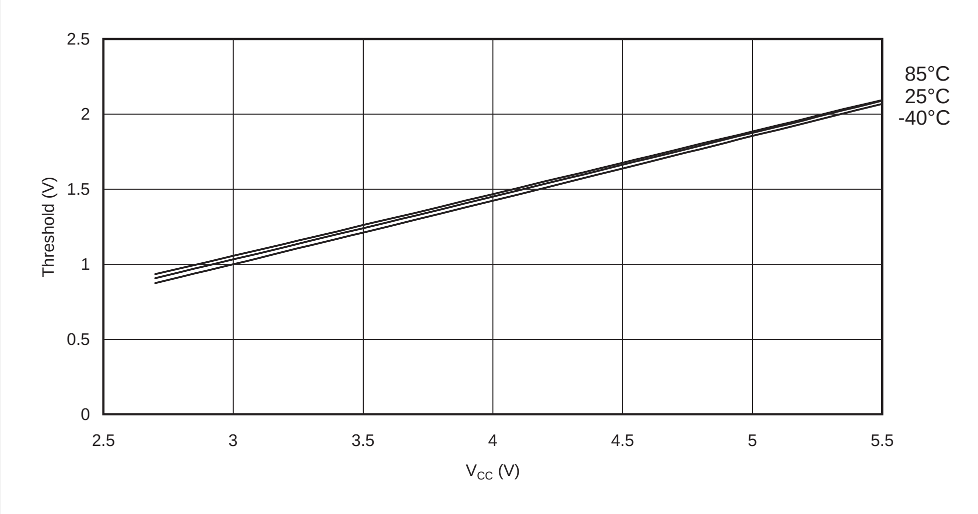


Figure 160. Reset Pin as I/O – Pin Hysteresis vs. V_{CC}

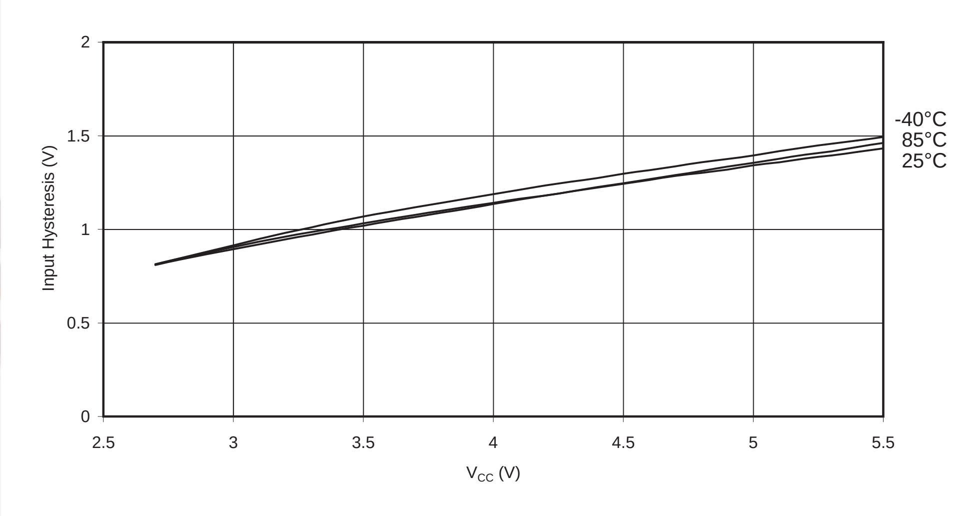


Figure 161. Reset Input Threshold Voltage vs. V_{CC} (V_{IH}, Reset Pin Read as "1")

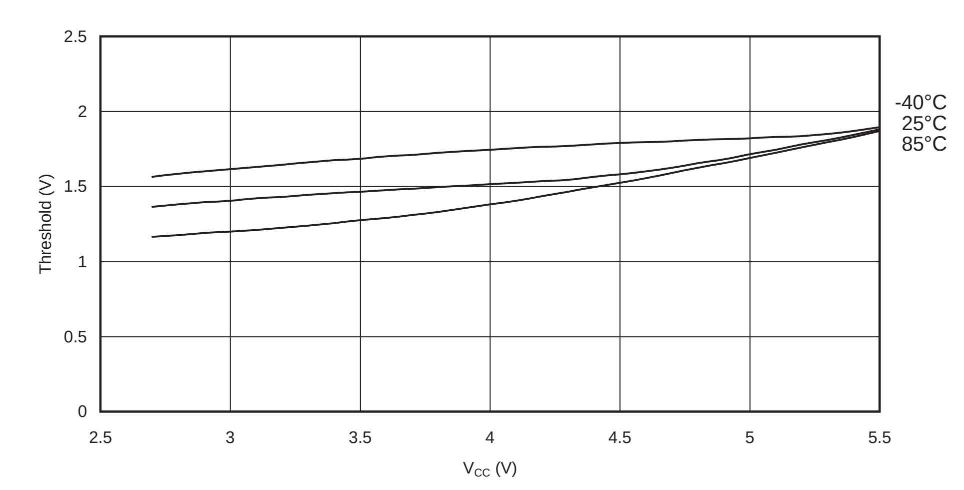


Figure 162. Reset Input Threshold Voltage vs. V_{CC} (V_{IL}, Reset Pin Read as "0")

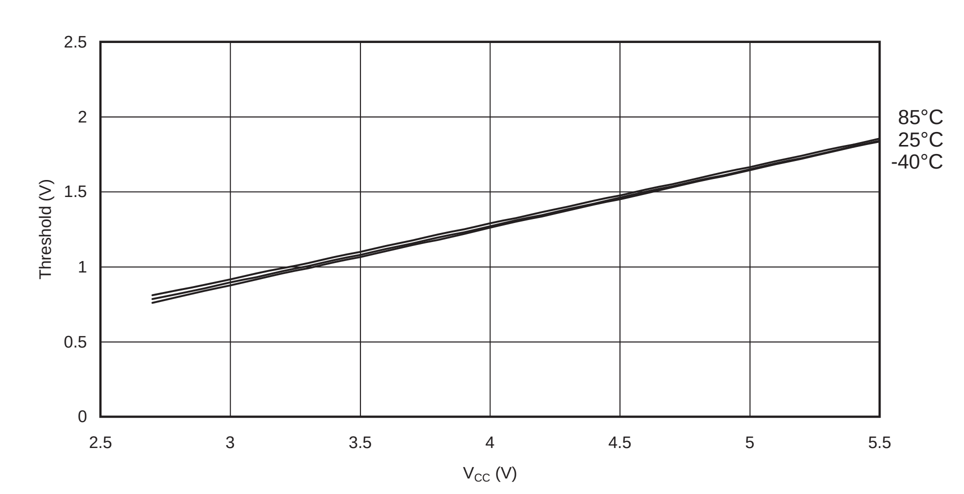
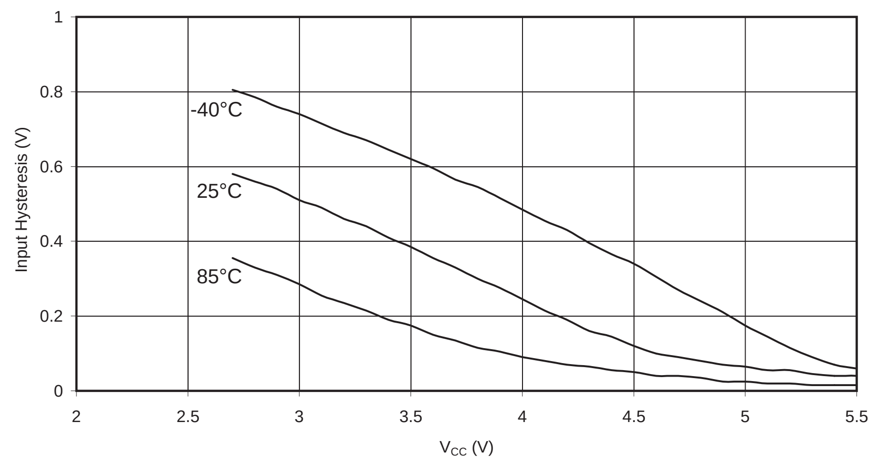


Figure 163. Reset Input Pin Hysteresis vs. V_{CC}



Bod Thresholds and Analog Comparator Offset

Figure 164. BOD Thresholds vs. Temperature (BOD Level is 4.0V)

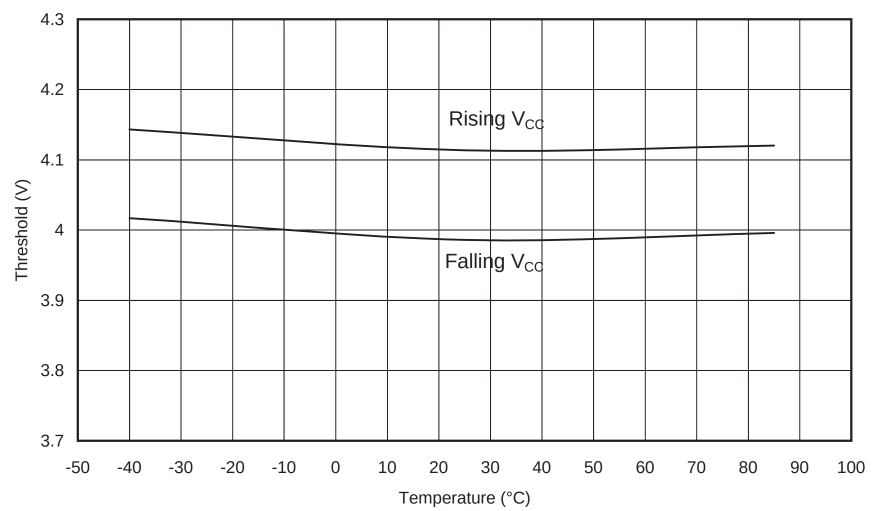


Figure 165. BOD Thresholds vs. Temperature (BOD Level is 2.7V)

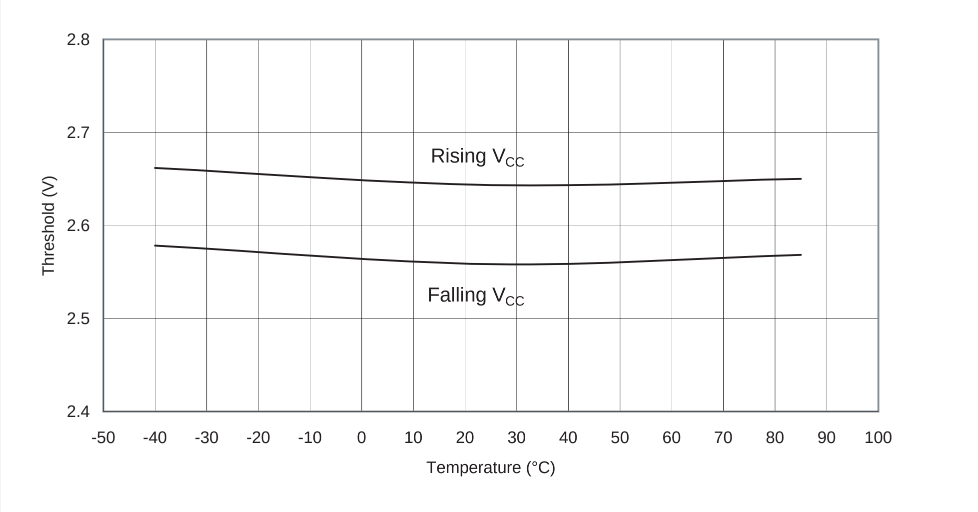


Figure 166. Bandgap Voltage vs. V_{CC}

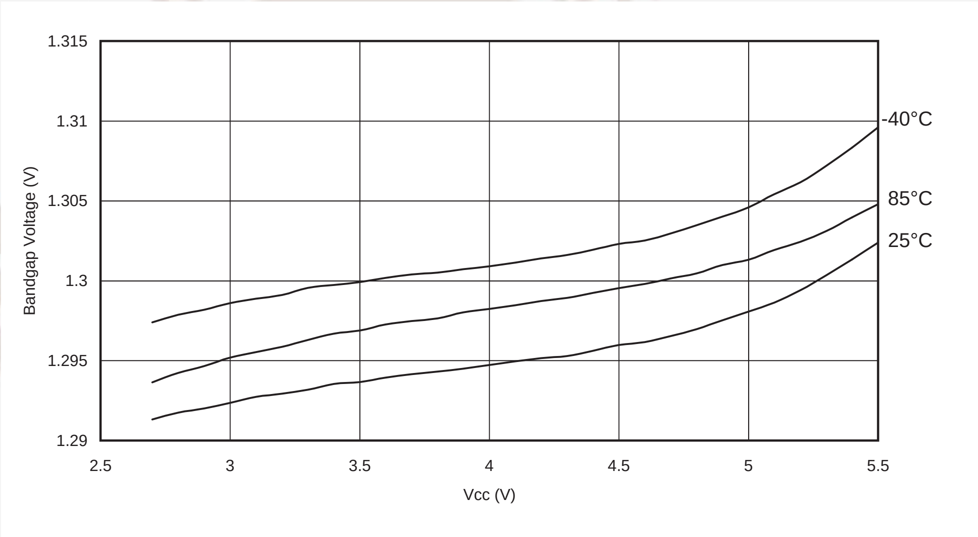


Figure 167. Analog Comparator Offset Voltage vs. Common Mode Voltage (V_{CC} = 5V)

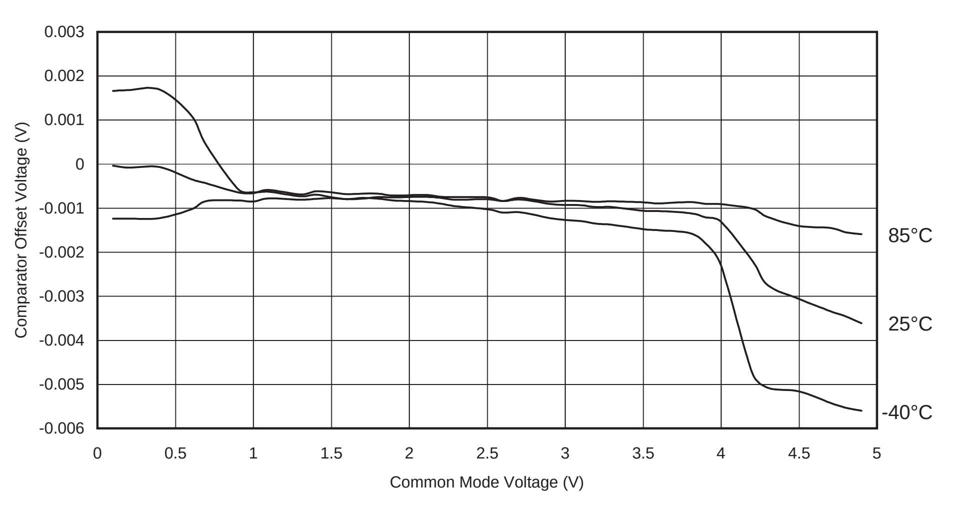
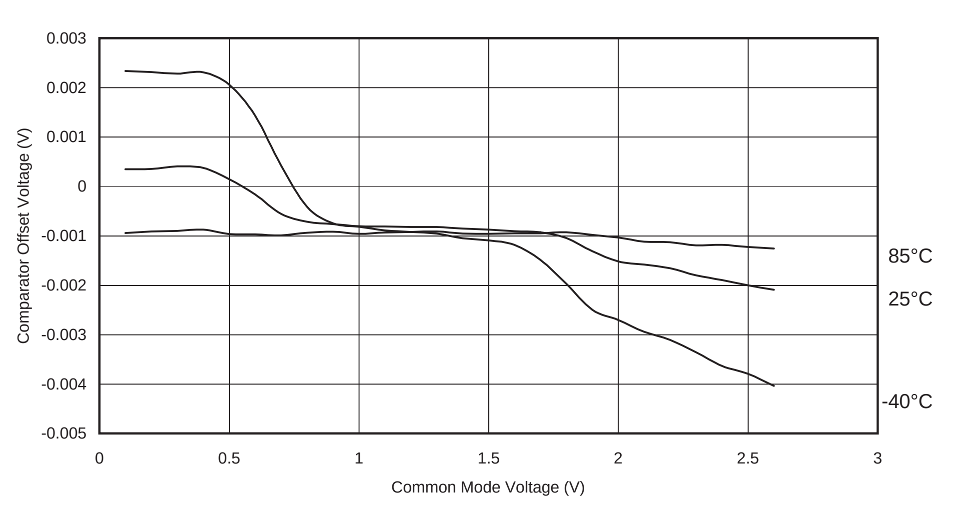


Figure 168. Analog Comparator Offset Voltage vs. Common Mode Voltage (V_{CC} = 2.7V)



Internal Oscillator Speed

Figure 169. Watchdog Oscillator Frequency vs. V_{CC}

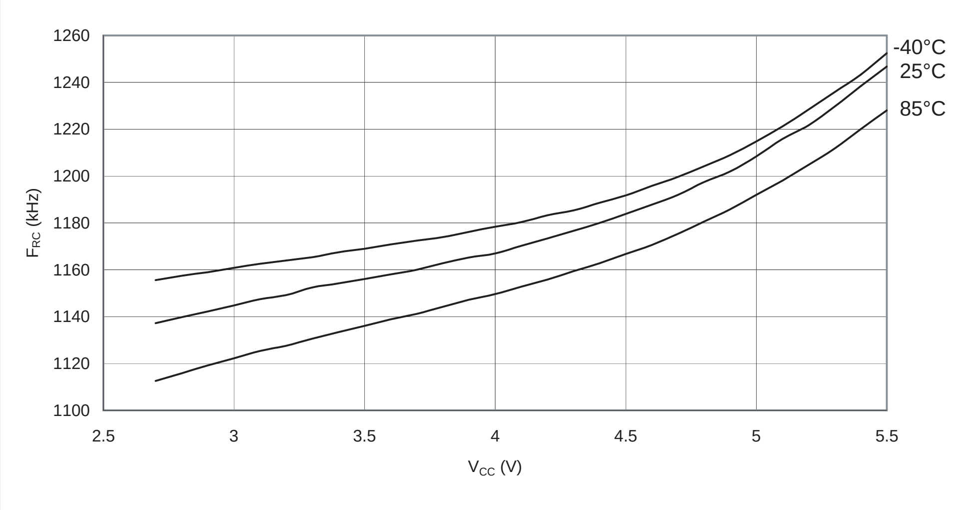


Figure 170. Calibrated 8MHz RC Oscillator Frequency vs. Temperature

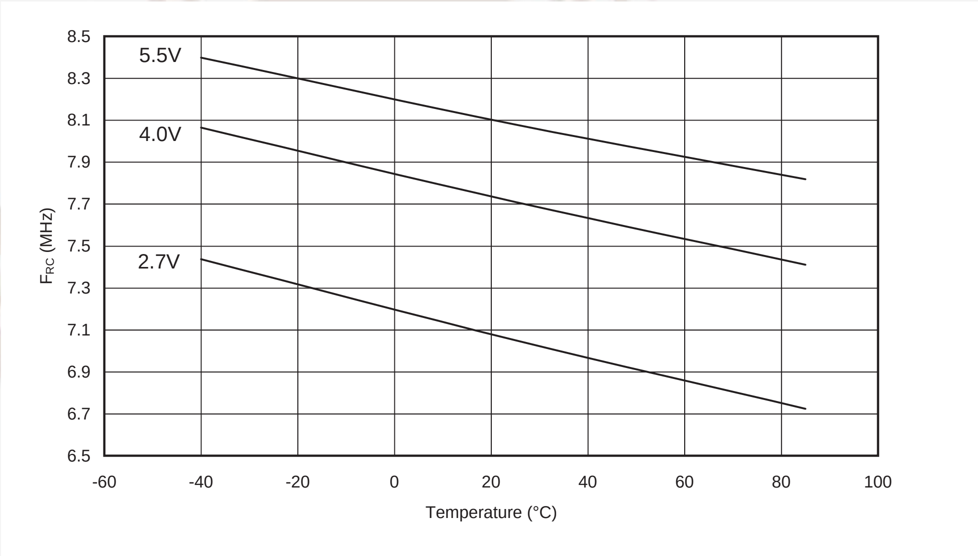


Figure 171. Calibrated 8MHz RC Oscillator Frequency vs. V_{CC}

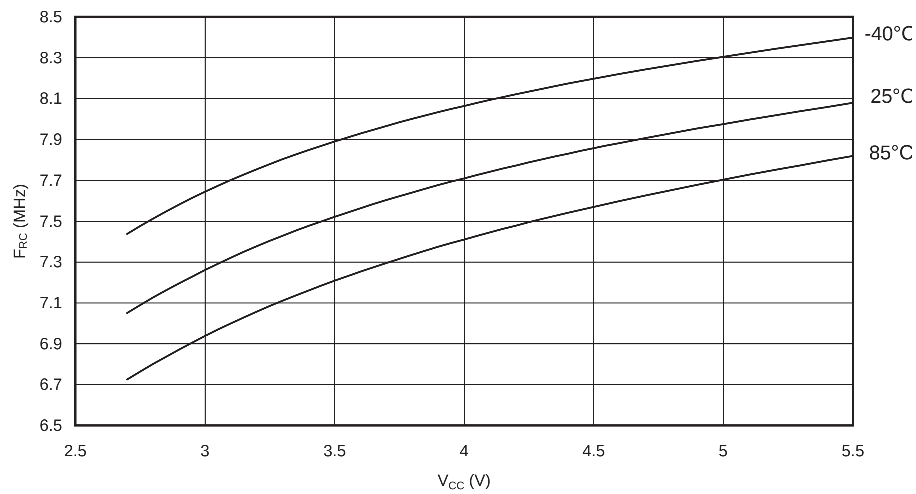


Figure 172. Calibrated 8MHz RC Oscillator Frequency vs. Osccal Value

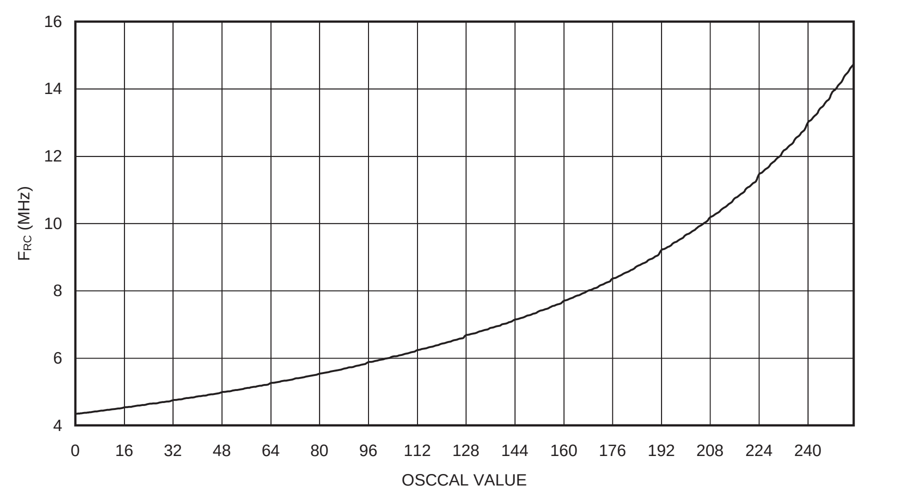


Figure 173. Calibrated 4MHz RC Oscillator Frequency vs. Temperature

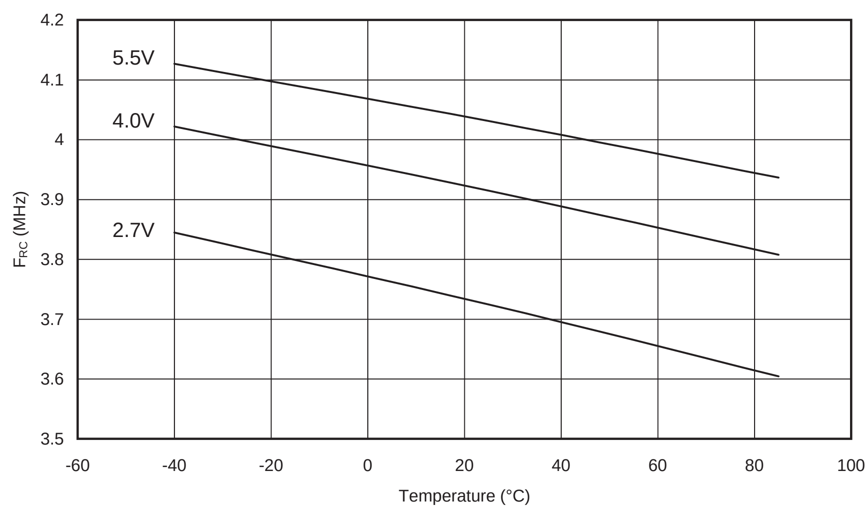


Figure 174. Calibrated 4MHz RC Oscillator Frequency vs. V_{CC}

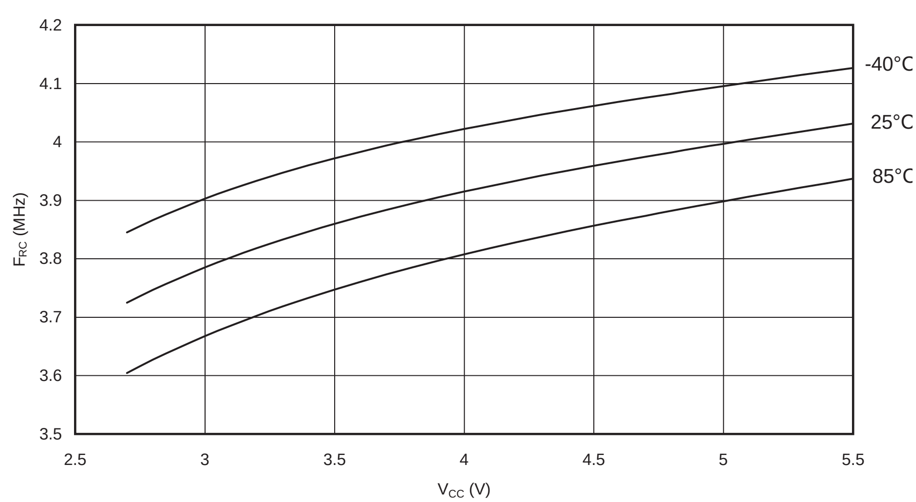


Figure 175. Calibrated 4MHz RC Oscillator Frequency vs. Osccal Value

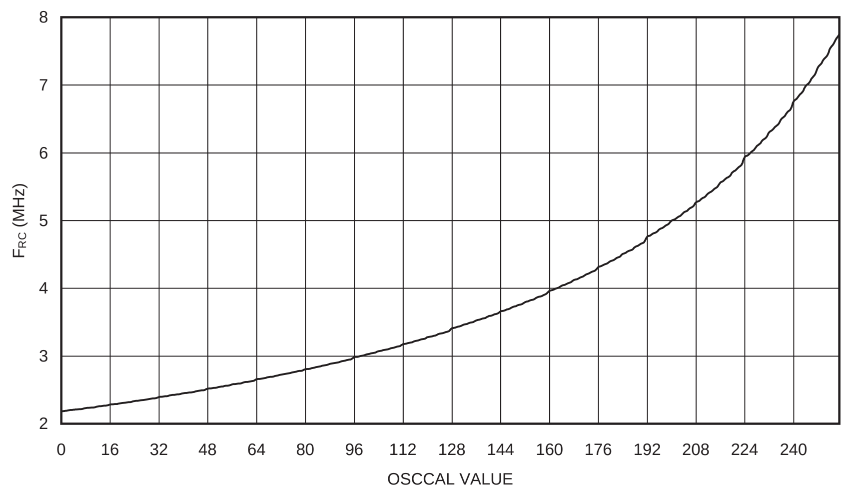


Figure 176. Calibrated 2MHz RC Oscillator Frequency vs. Temperature

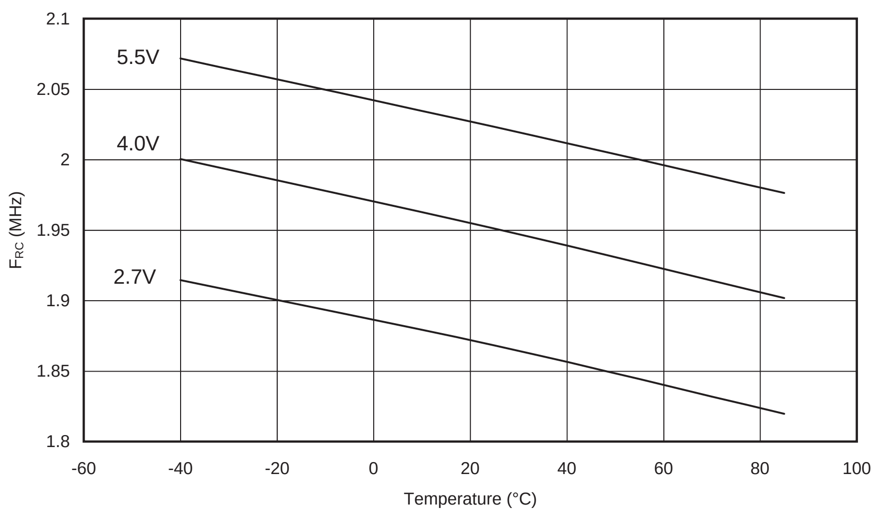


Figure 177. Calibrated 2MHz RC Oscillator Frequency vs. V_{CC}

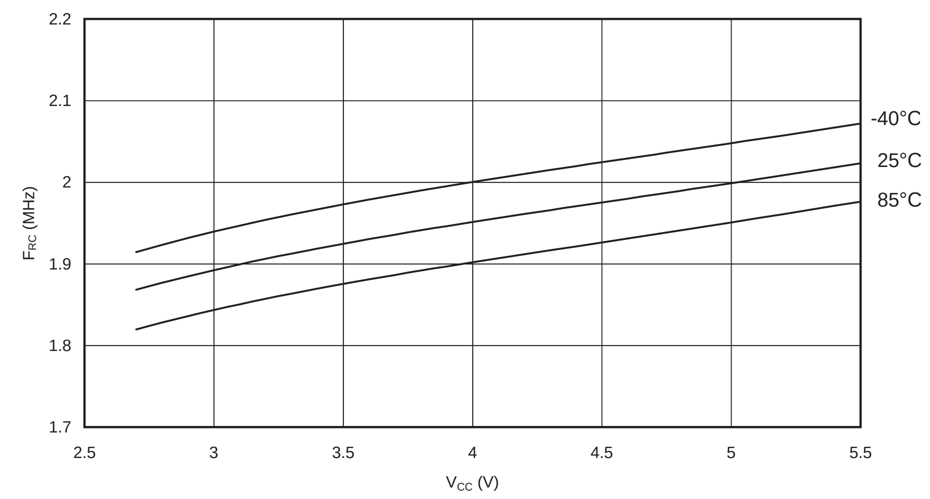


Figure 178. Calibrated 2MHz RC Oscillator Frequency vs. Osccal Value

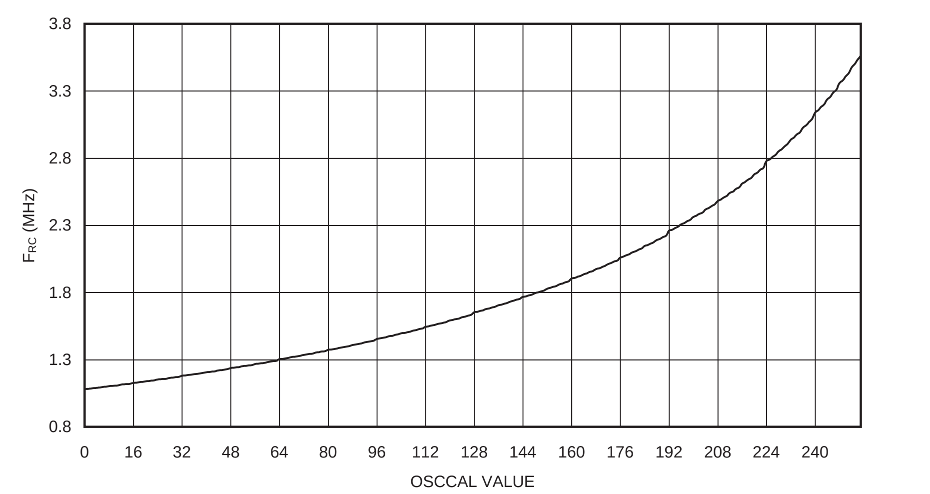


Figure 179. Calibrated 1MHz RC Oscillator Frequency vs. Temperature

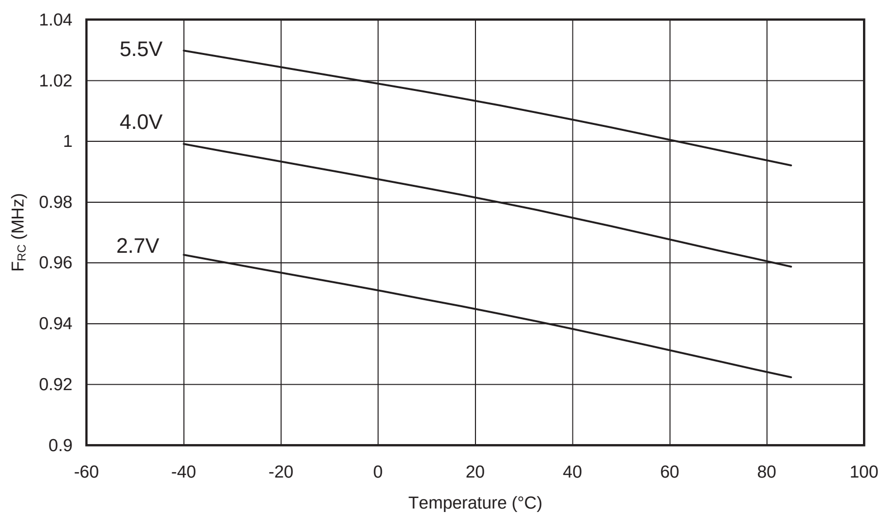


Figure 180. Calibrated 1MHz RC Oscillator Frequency vs. V_{CC}

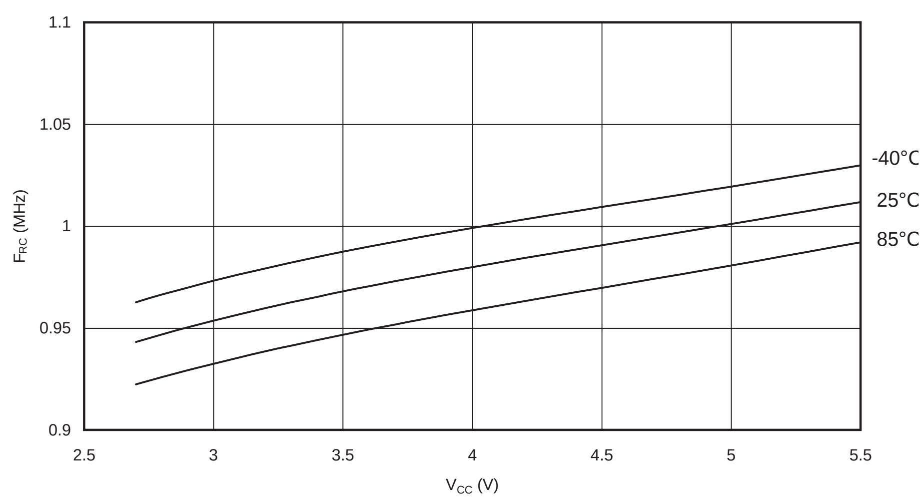
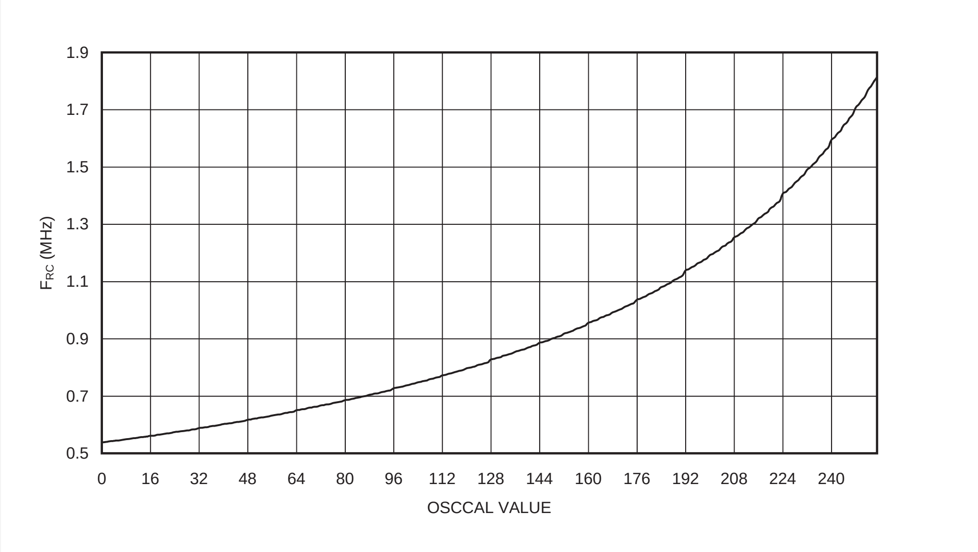


Figure 181. Calibrated 1MHz RC Oscillator Frequency vs. Osccal Value



Current Consumption of Peripheral Units

Figure 182. Brown-out Detector Current vs. V_{CC}

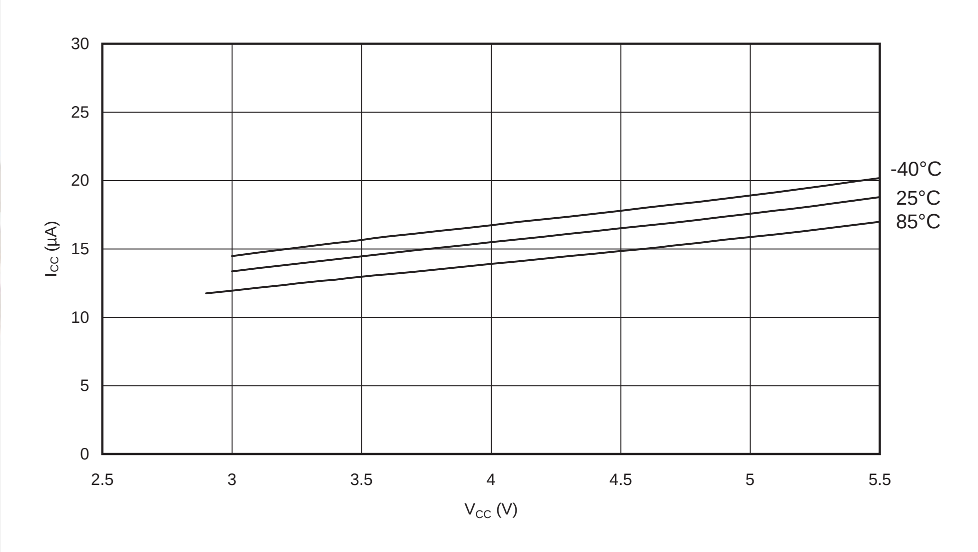


Figure 183. ADC Current vs. V_{CC} (AREF = AV_{CC})

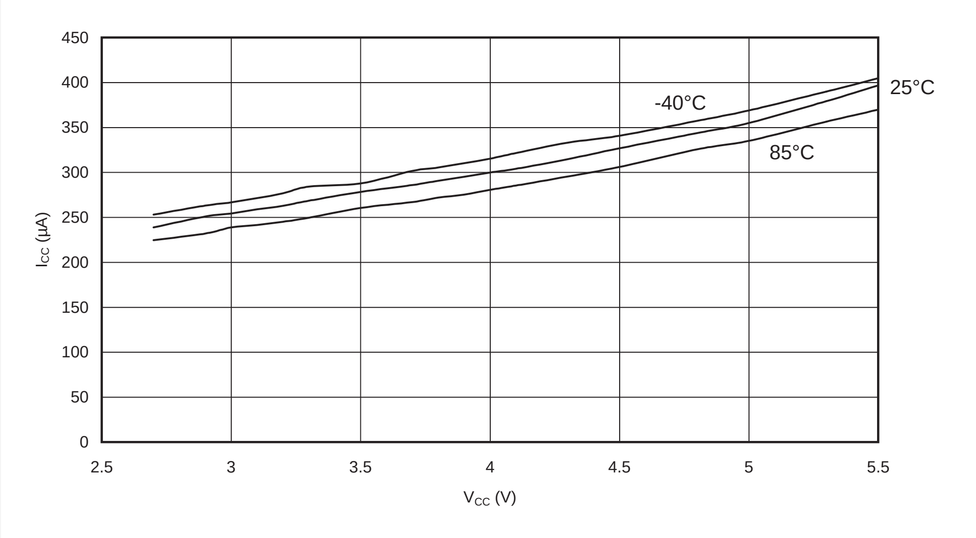


Figure 184. AREF External Reference Current vs. V_{CC}

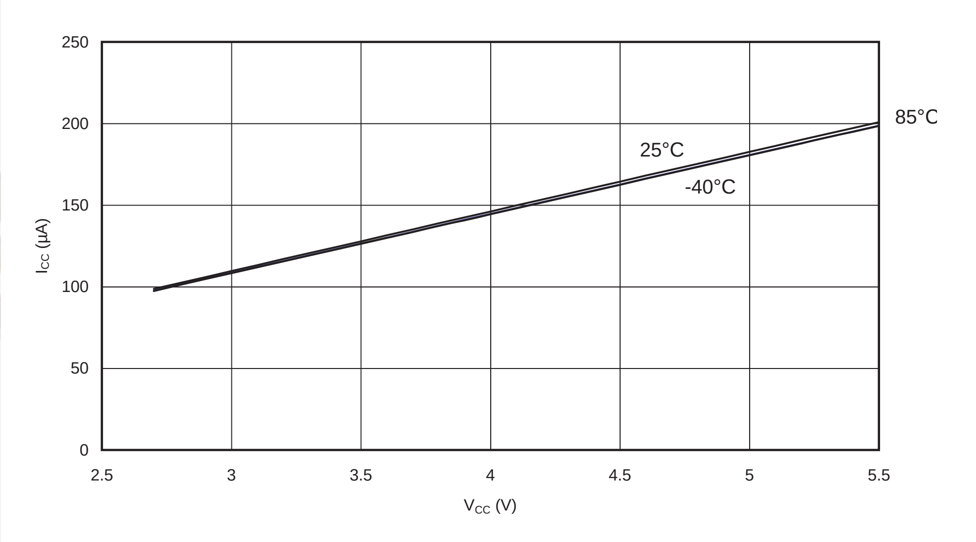


Figure 185. 32kHz TOSC Current vs. V_{CC} (Watchdog Timer Disabled)

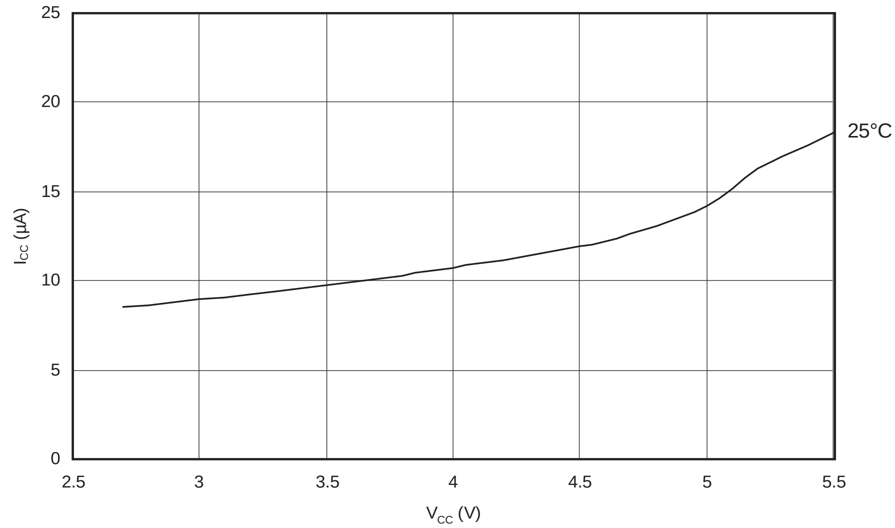


Figure 186. Watchdog Timer Current vs. V_{CC}

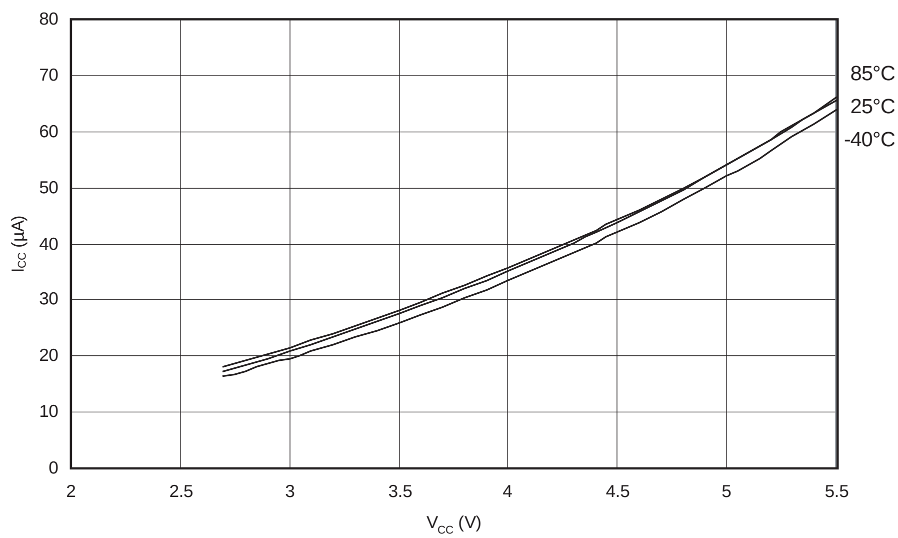


Figure 187. Analog Comparator Current vs. V_{CC}

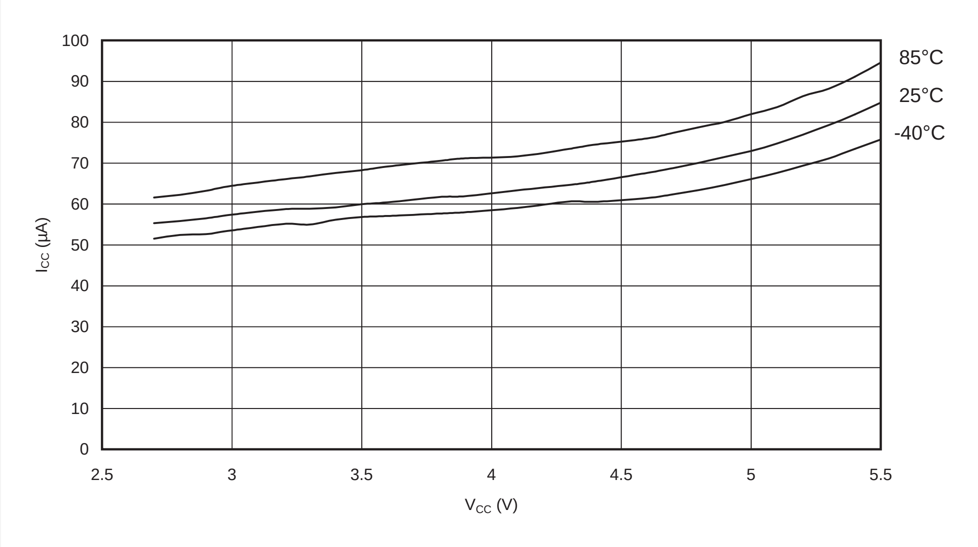
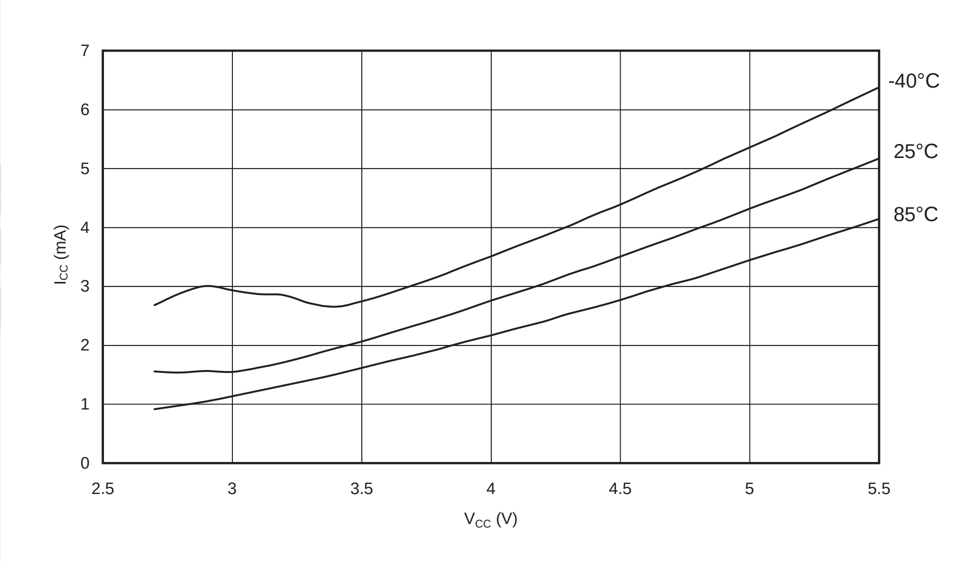


Figure 188. Programming Current vs. V_{CC}



Current Consumption in Reset and Reset Pulsewidth

Figure 189. Reset Supply Current vs. V_{CC} (0.1MHz - 1.0MHz, Excluding Current Through The Reset Pull-up)

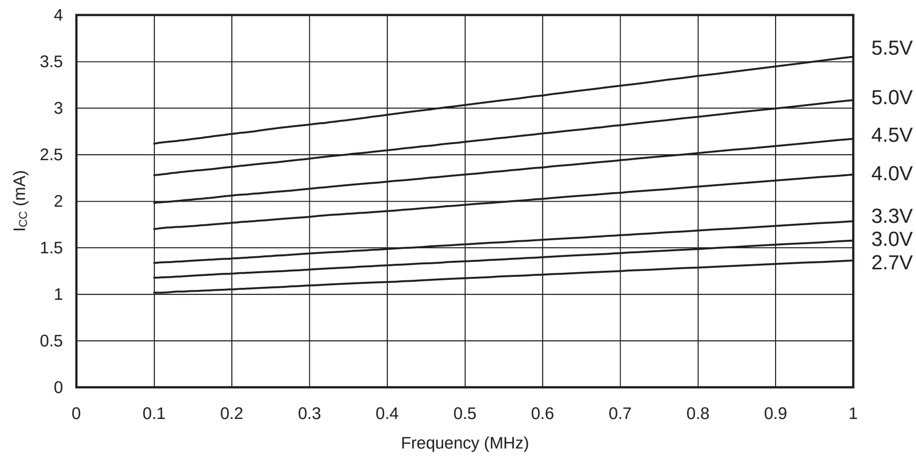


Figure 190. Reset Supply Current vs. V_{CC} (1MHz - 20MHz, Excluding Current Through The Reset Pull-up)

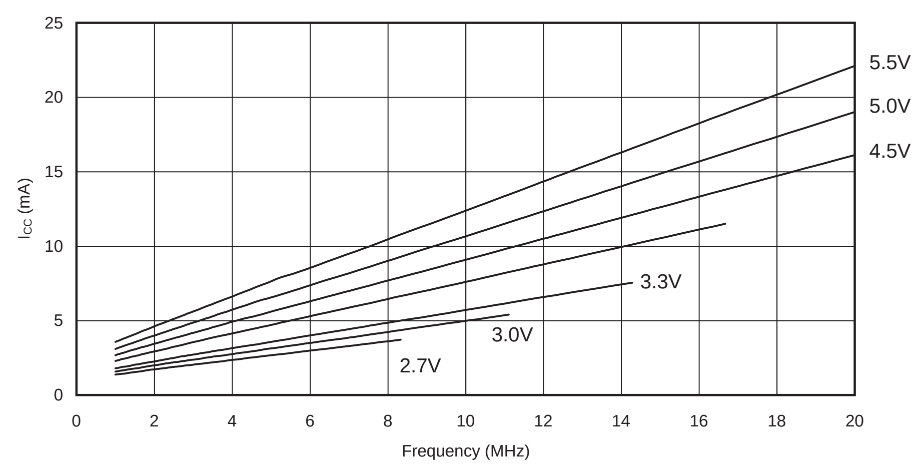
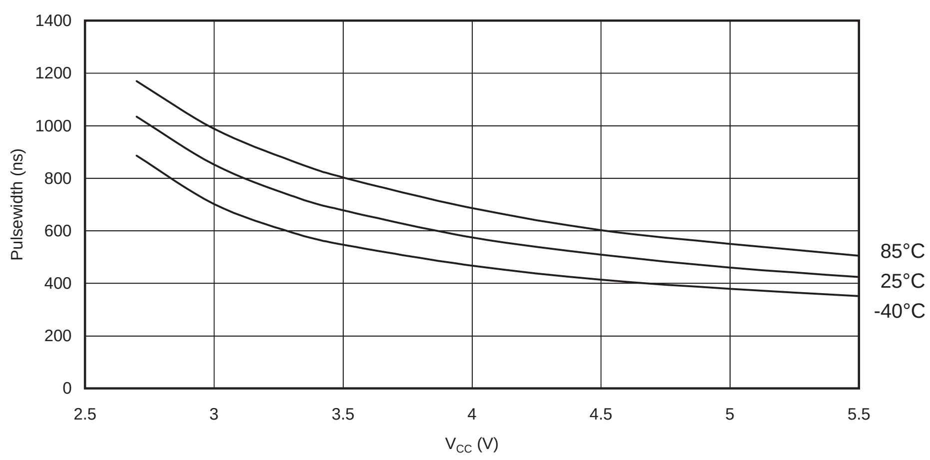


Figure 191. Reset Pulse Width vs. V_{CC}



ATmega8 Typical Characteristics – TA = -40°C to 105°C

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 0-1. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)

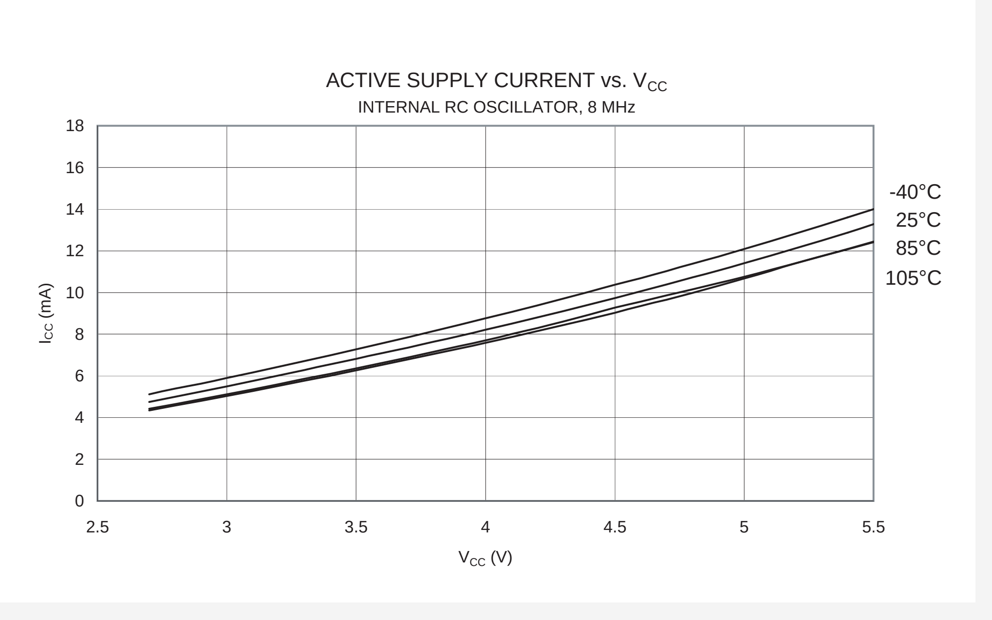


Figure 0-2. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 4 MHz)

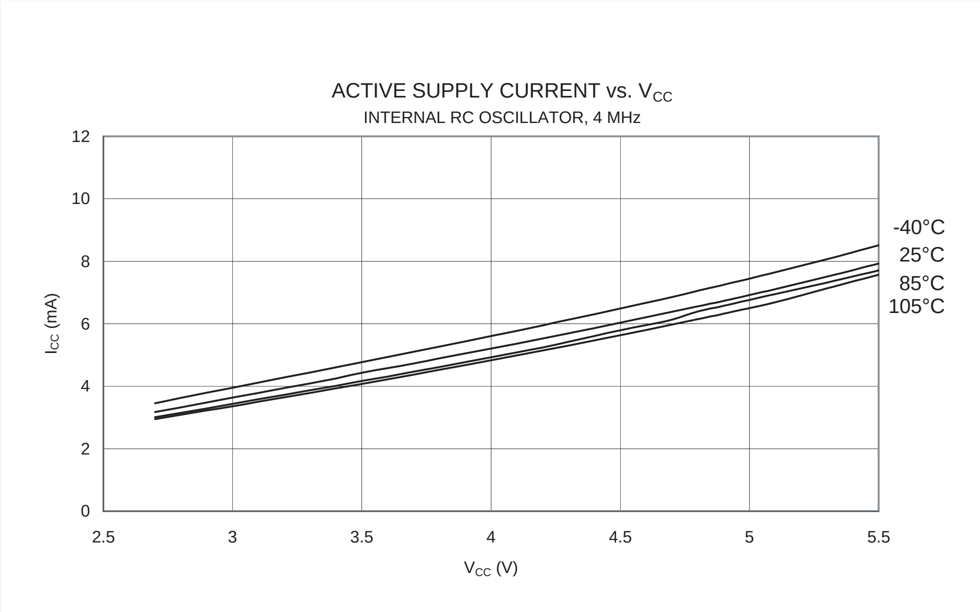


Figure 0-3. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 2 MHz)

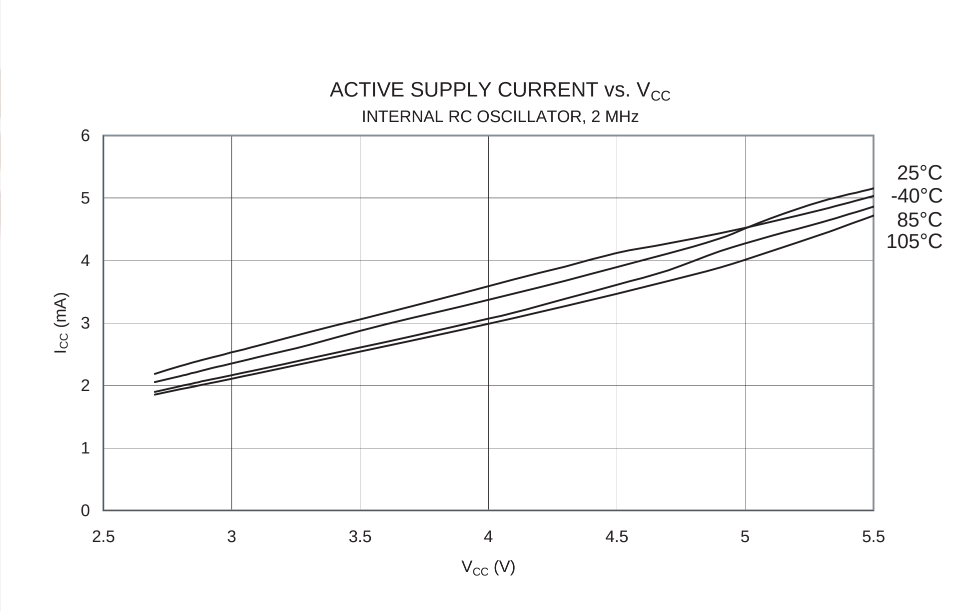
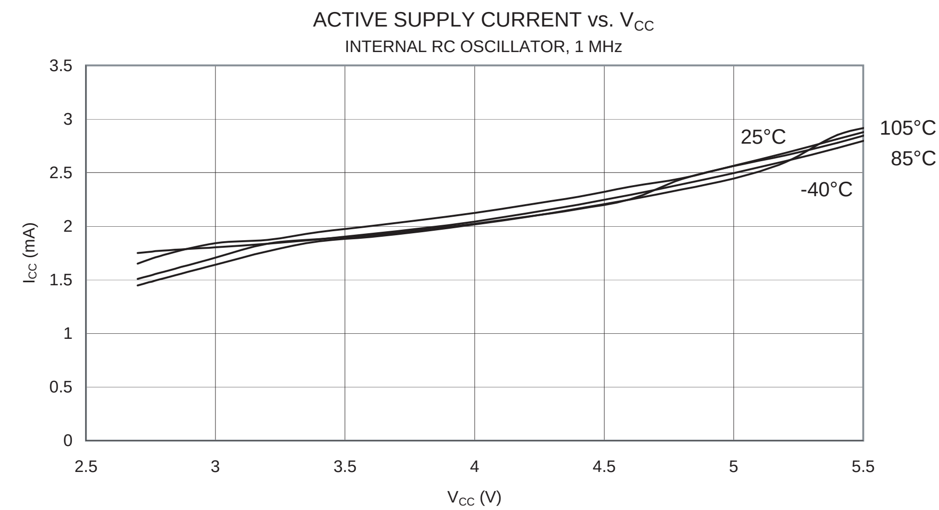


Figure 0-4. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 1 MHz)



Idle Supply Current

Figure 0-5. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)

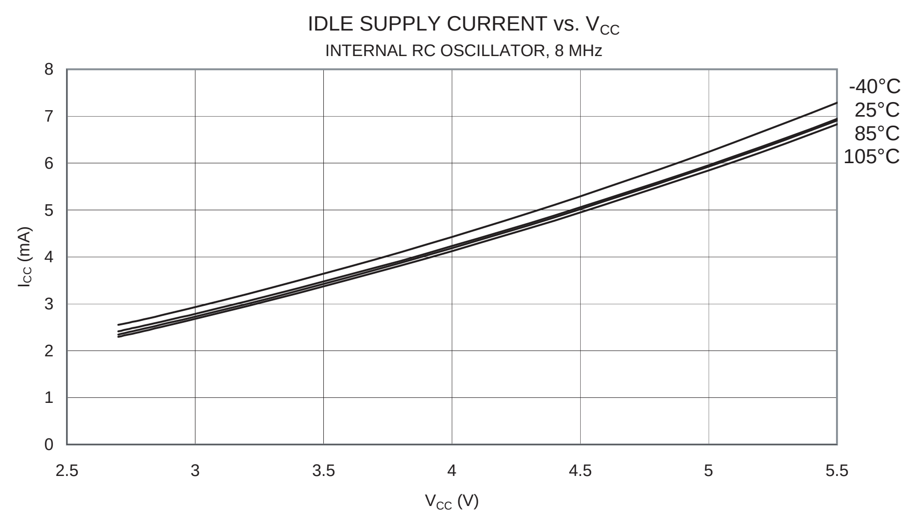


Figure 0-6. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 4 MHz)

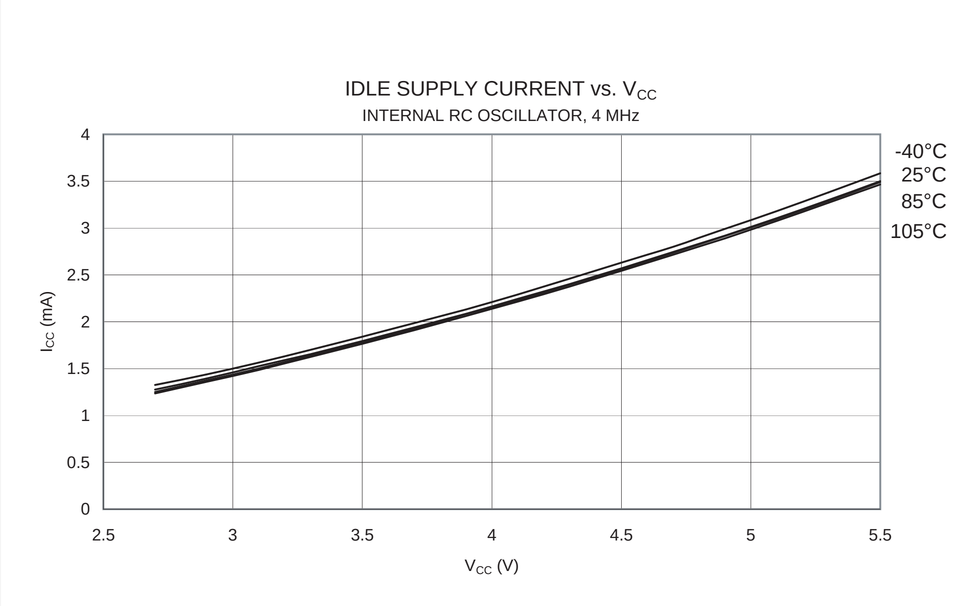


Figure 0-7. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 2 MHz)

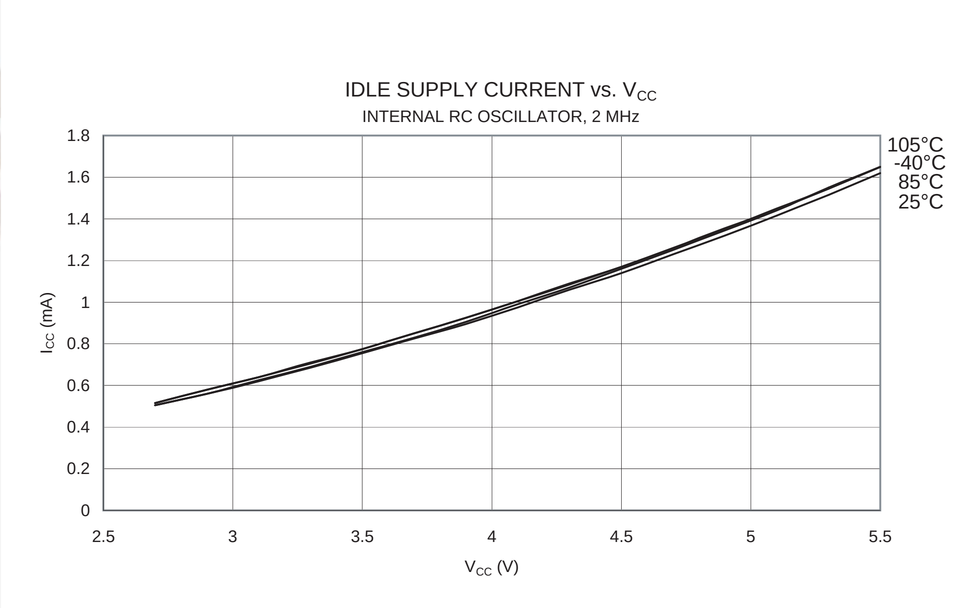
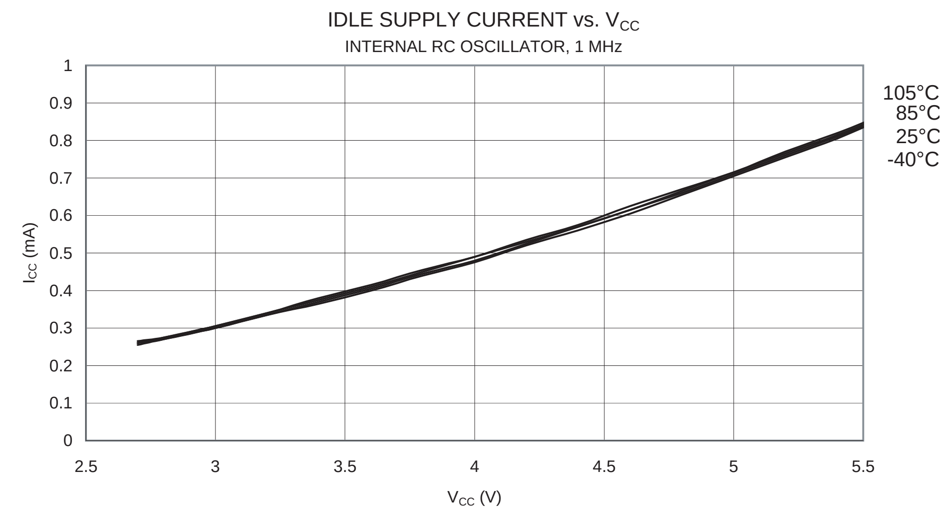


Figure 0-8. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 1 MHz)



Power-down Supply Current

Figure 0-9. Power-down Supply Current vs. V_{CC} (Watchdog Timer Disabled)

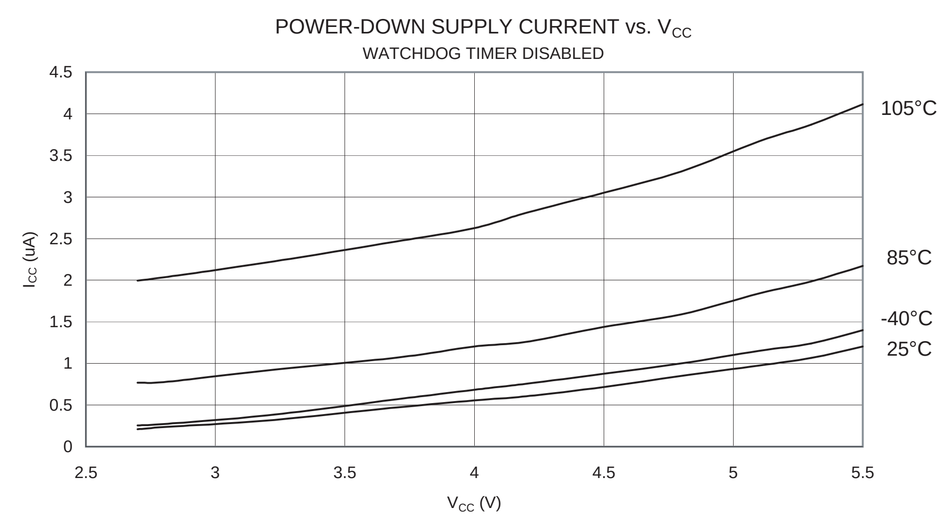
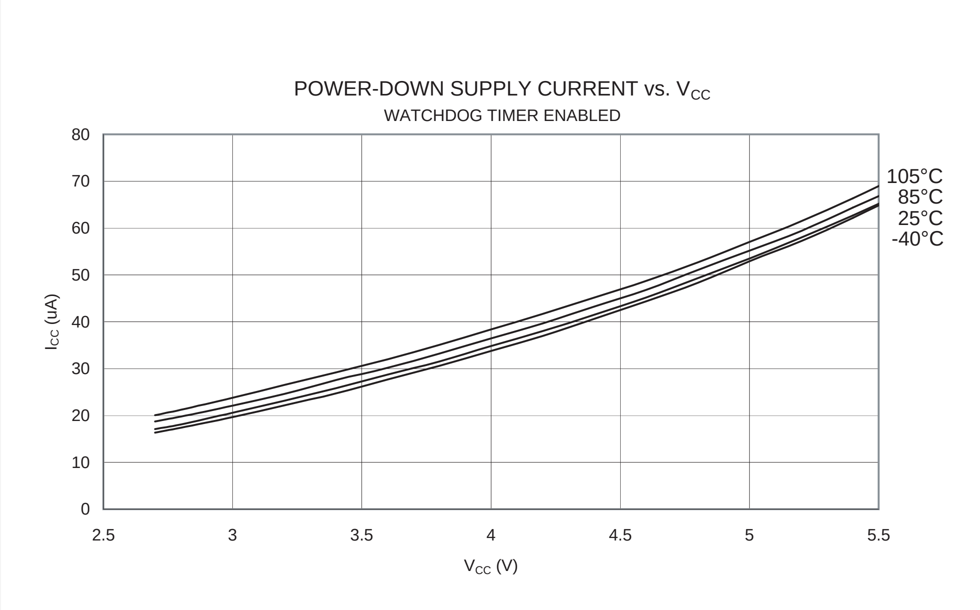


Figure 0-10. Power-down Supply Current vs. V_{CC} (Watchdog Timer Enabled)



Pin Pull-up

Figure 0-11. I/O Pin Pull-up Resistor Current vs. Input Voltage (V_{CC} = 5V)

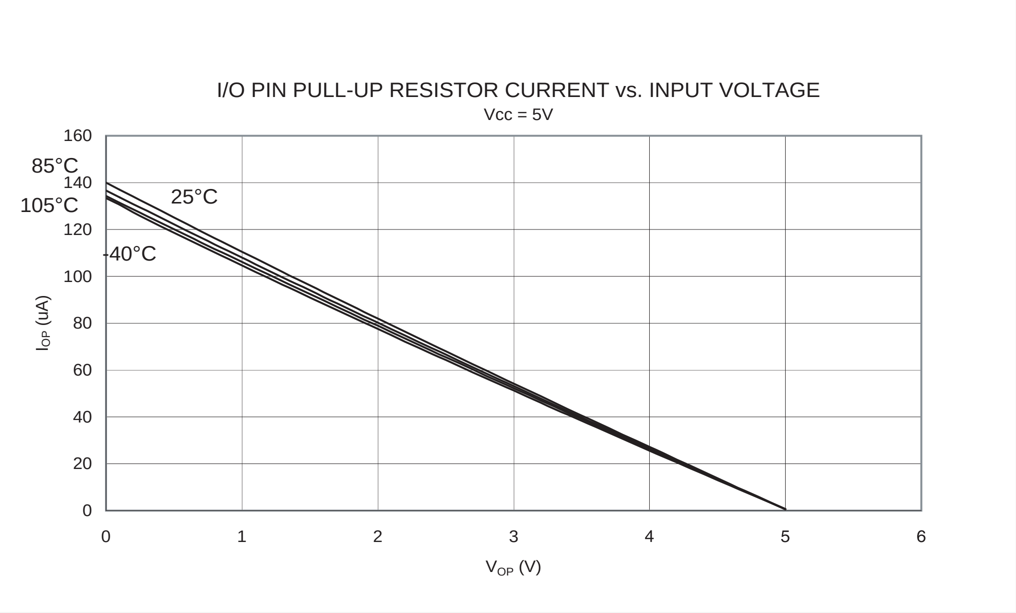


Figure 0-12. I/O Pin Pull-up Resistor Current vs. Input Voltage (V_{CC} = 2.7V)

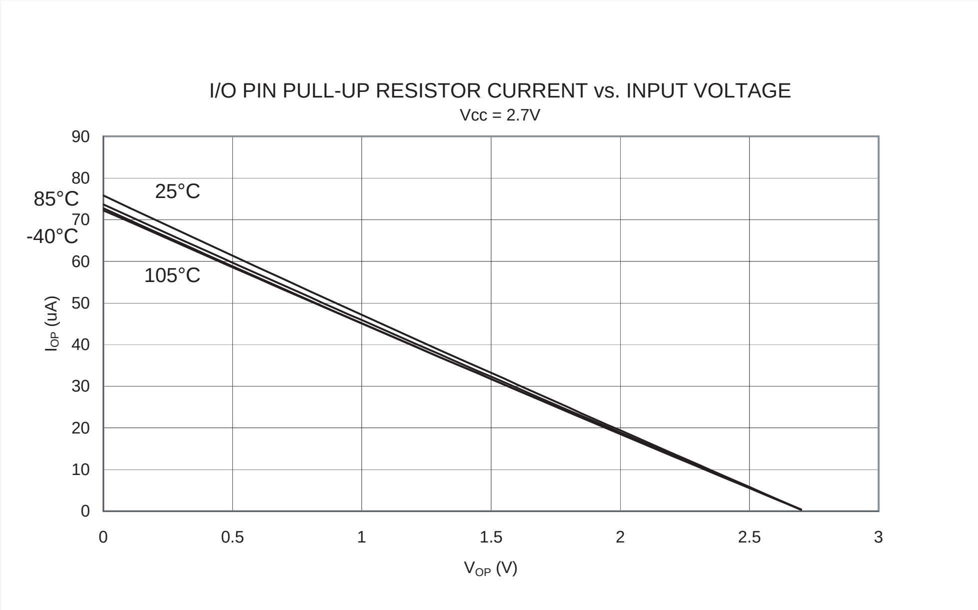
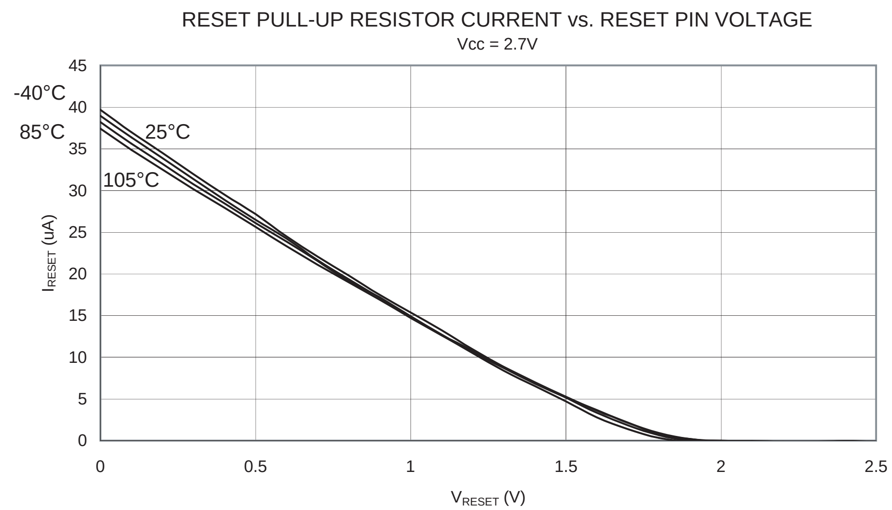


Figure 0-13. Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 5V)

Figure 0-14. Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 2.7V)



Pin Driver Strength

Figure 0-15. I/O Pin Source Current vs. Output Voltage (V_{CC} = 5V)

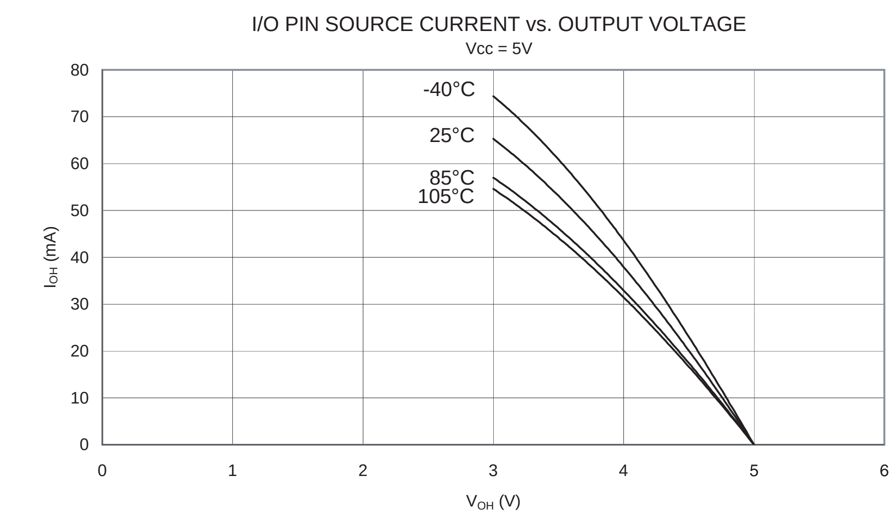


Figure 0-16. I/O Pin Source Current vs. Output Voltage (V_{CC} = 2.7V)

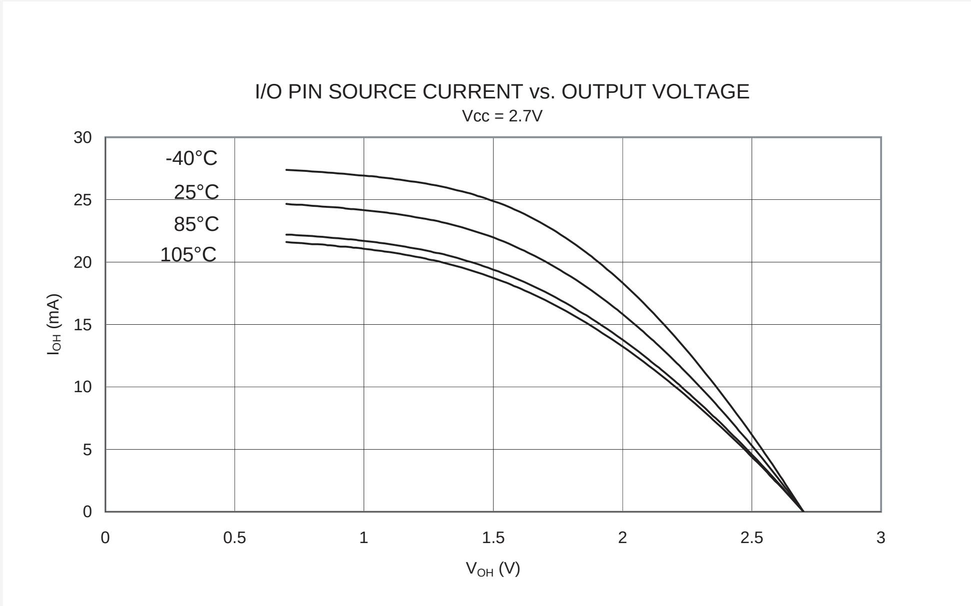


Figure 0-17. I/O Pin Sink Current vs. Output Voltage (V_{CC} = 5V)

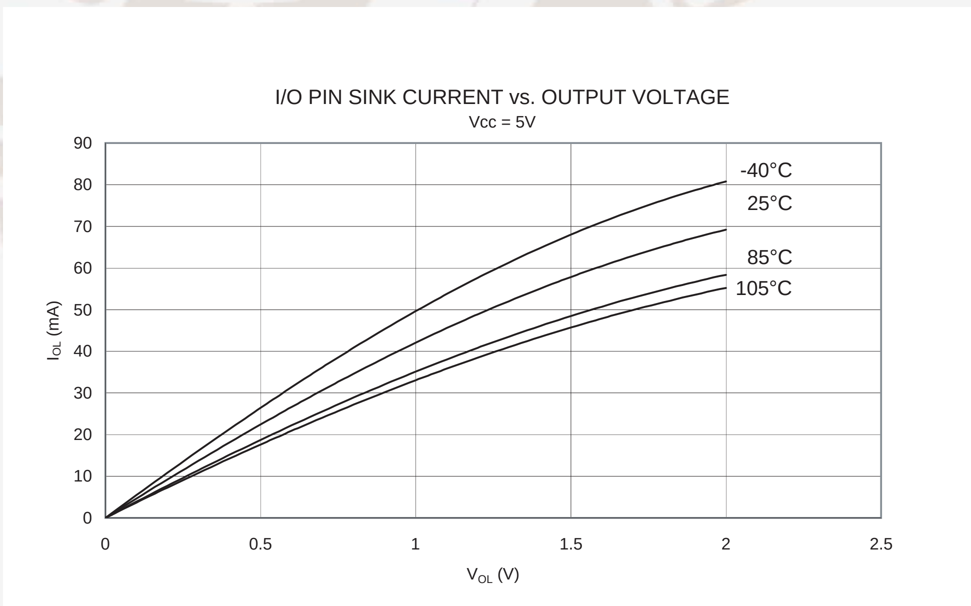


Figure 0-18. I/O Pin Sink Current vs. Output Voltage (V_{CC} = 2.7V)

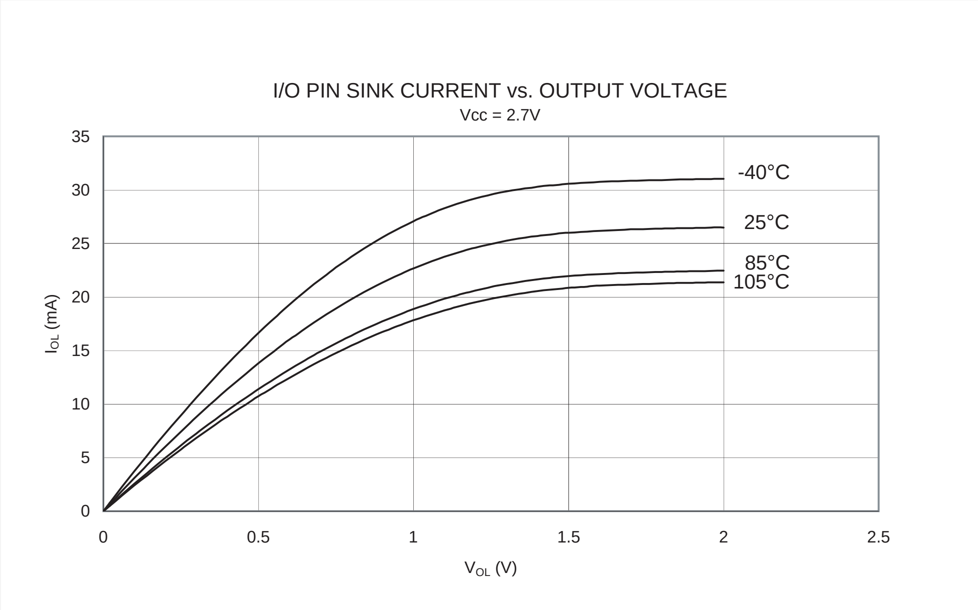


Figure 0-19. Reset Pin as I/O – Pin Source Current vs. Output Voltage (V_{CC} = 5V)

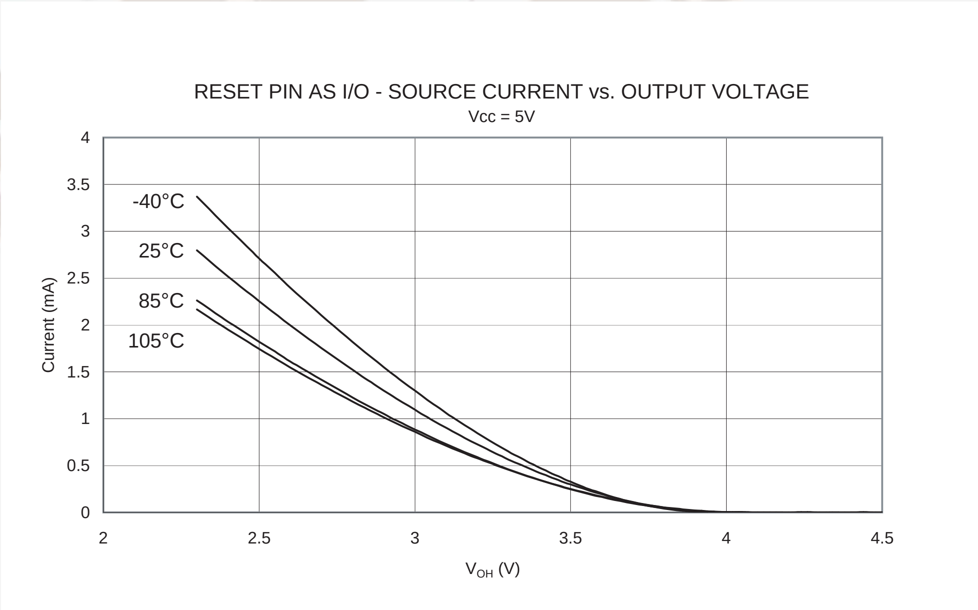


Figure 0-20. Reset Pin as I/O – Pin Source Current vs. Output Voltage (V_{CC} = 2.7V)

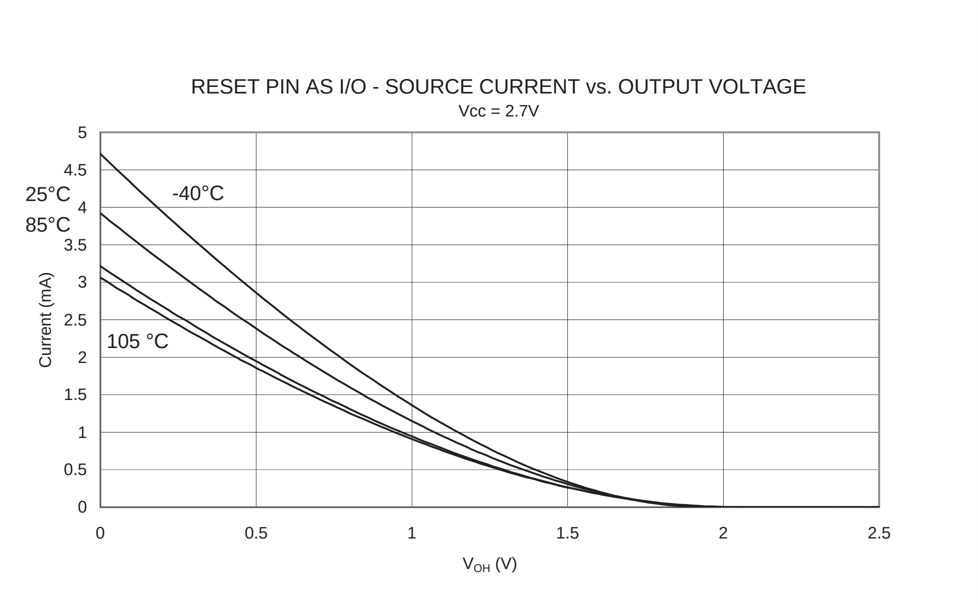


Figure 0-21. Reset Pin as I/O – Pin Sink Current vs. Output Voltage (V_{CC} = 5V)

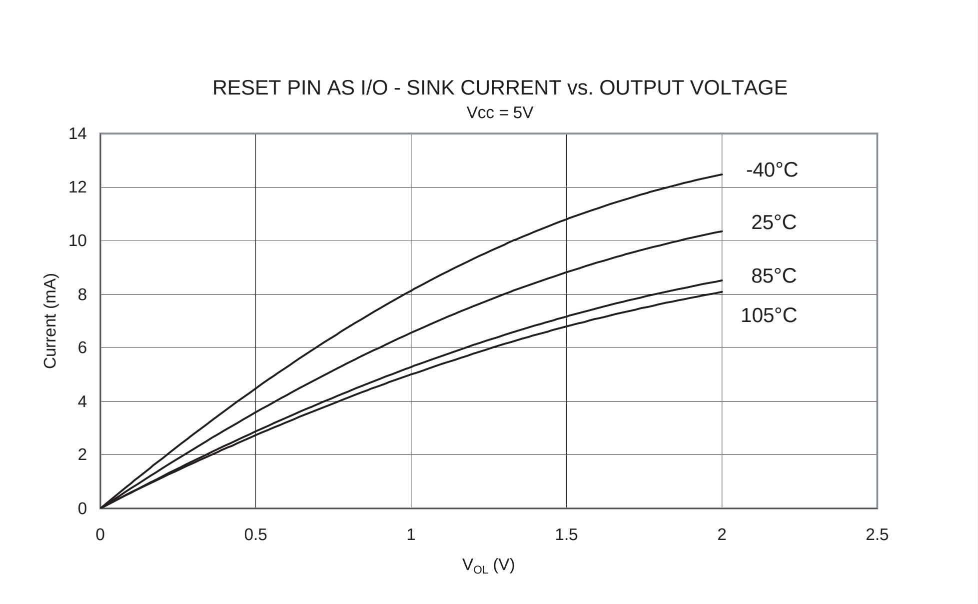
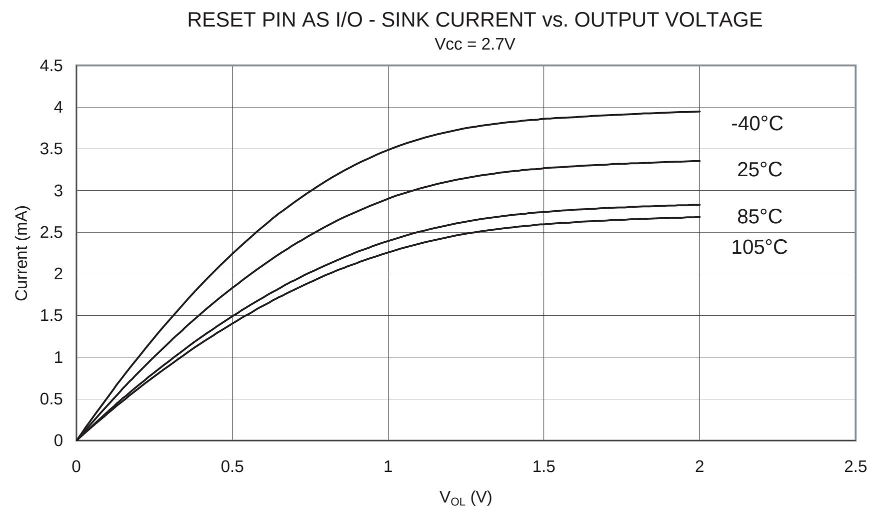


Figure 0-22. Reset Pin as I/O – Pin Sink Current vs. Output Voltage (V_{CC} = 2.7V)



Pin Thresholds and Hysteresis

Figure 0-23. I/O Pin Input Threshold Voltage vs. V_{CC} (V_{IH}, I/O Pin Read as "1")

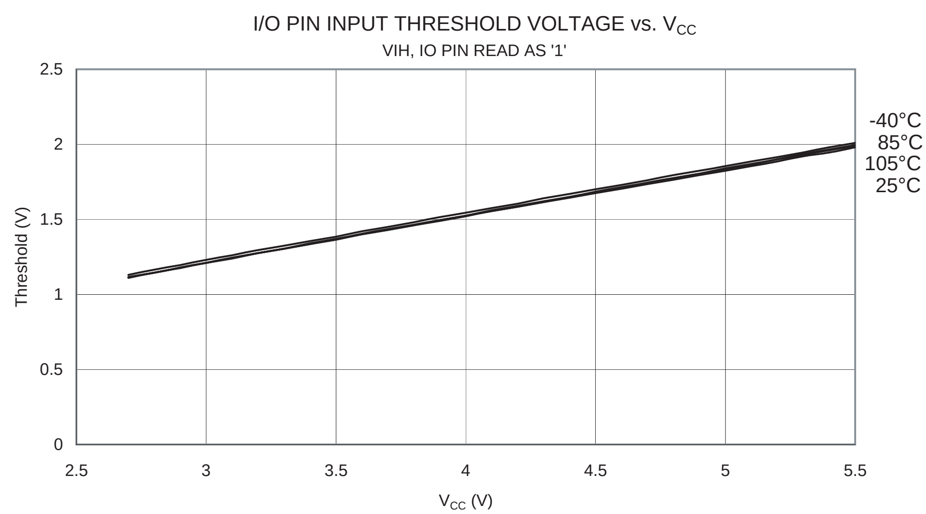


Figure 0-24. I/O Pin Input Threshold Voltage vs. V_{CC} (V_{IL}, I/O Pin Read as "0")

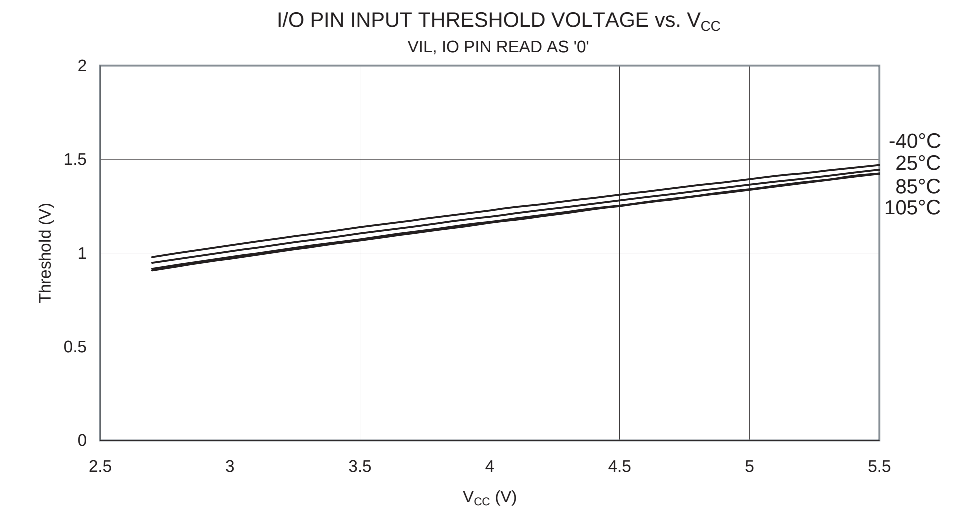


Figure 0-25. I/O Pin Input Hysteresis vs. V_{CC}

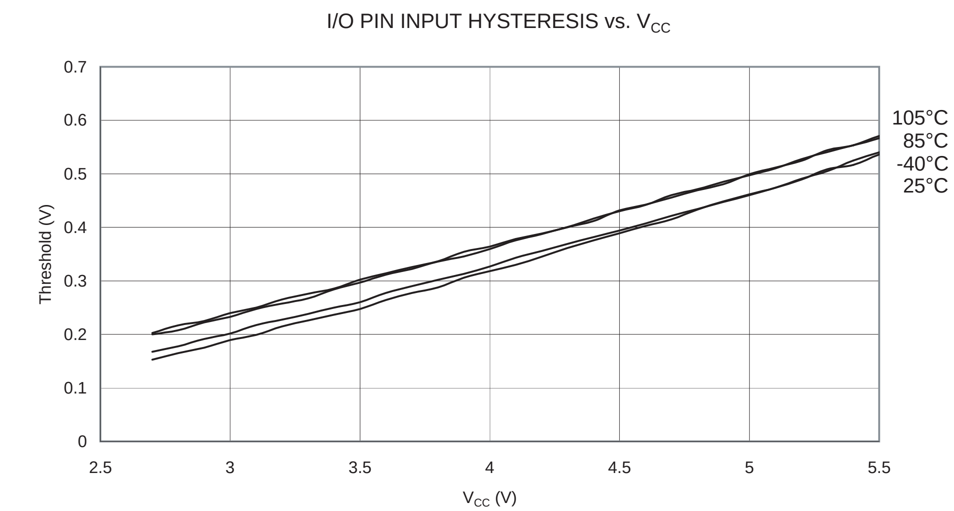


Figure 0-26. Reset Pin as I/O – Input Threshold Voltage vs. V_{CC} (V_{IH},I/O Pin Read as "1")

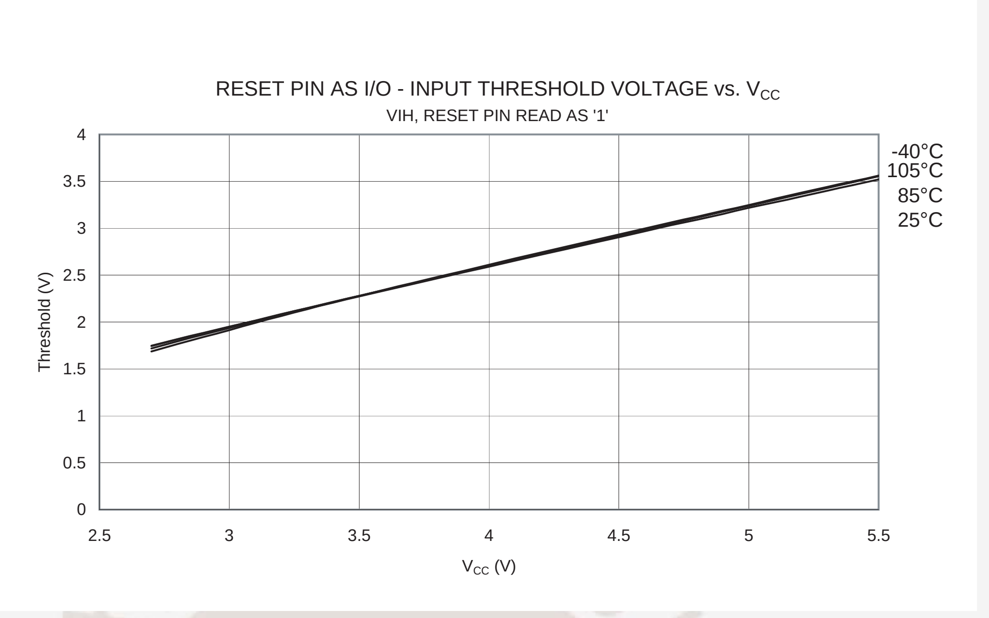


Figure 0-27. Reset Pin as I/O – Input Threshold Voltage vs. V_{CC} (V_{IL}, I/O Pin Read as "0")

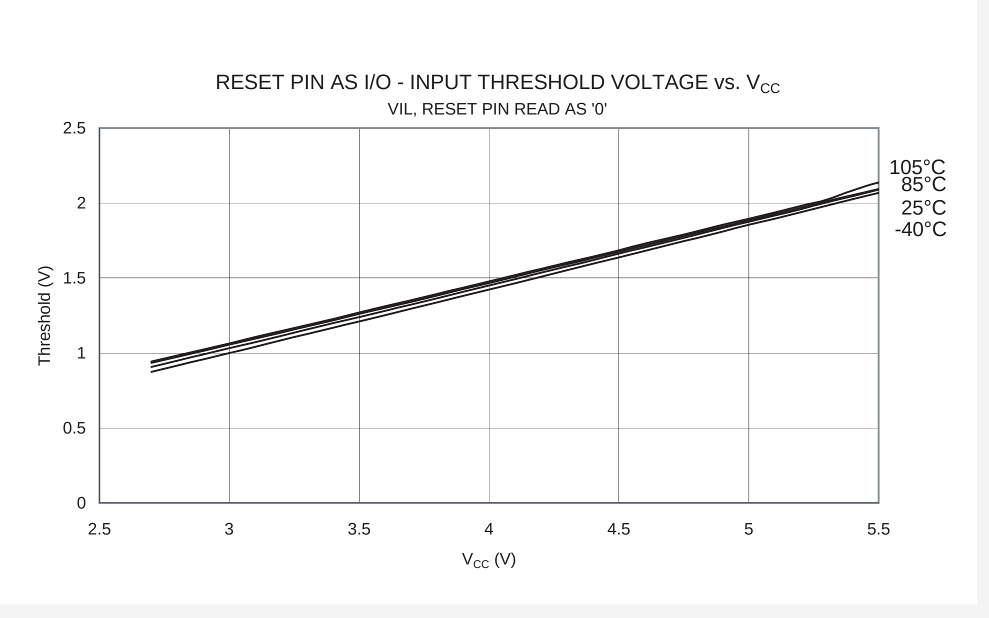


Figure 0-28. Reset Pin as I/O – Pin Hysteresis vs. V_{CC}

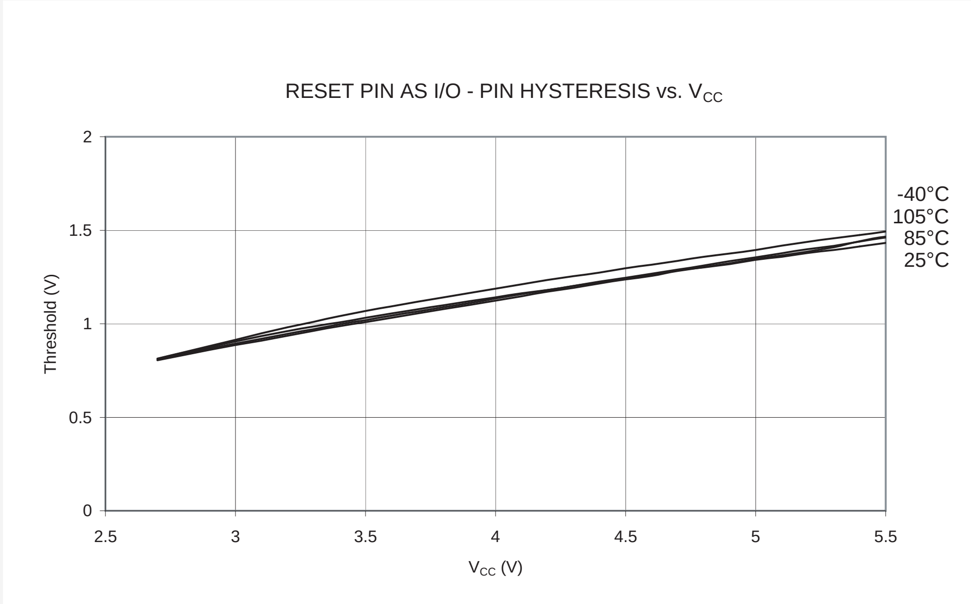


Figure 0-29. Reset Input Threshold Voltage vs. V_{CC} (V_{IH}, Reset Pin Read as “1”)

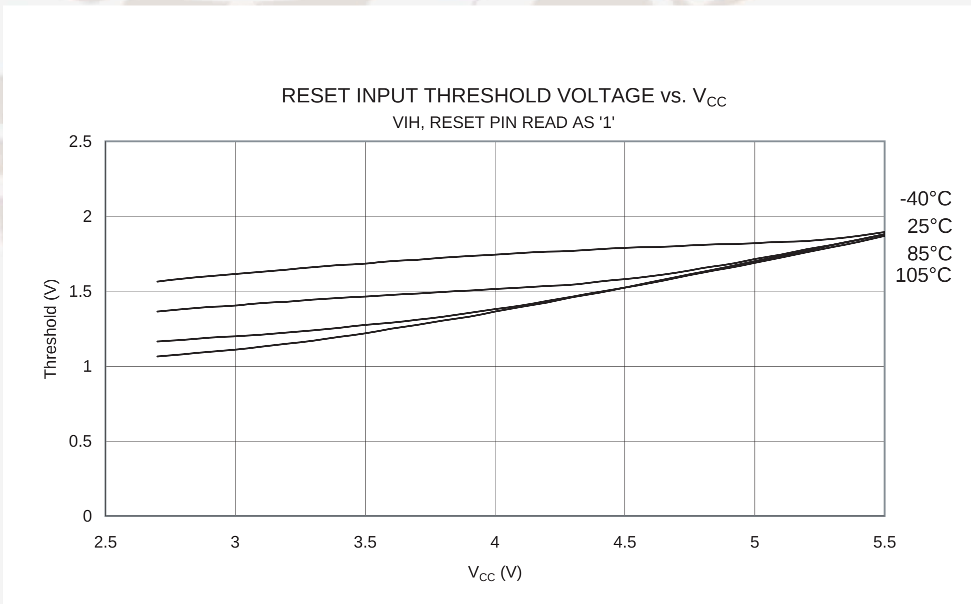


Figure 0-30. Reset Input Threshold Voltage vs. V_{CC} (V_{IL}, Reset Pin Read as "0")

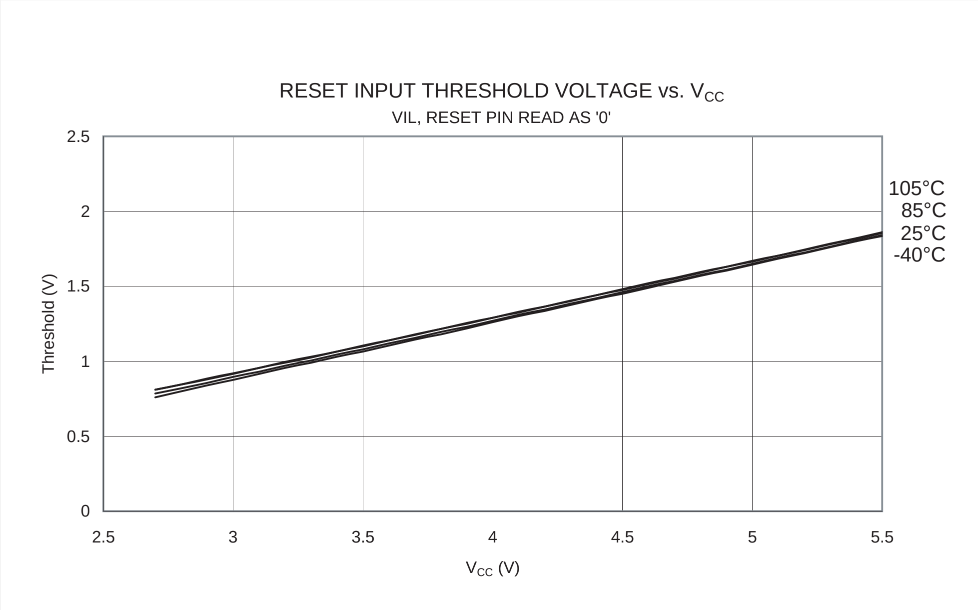
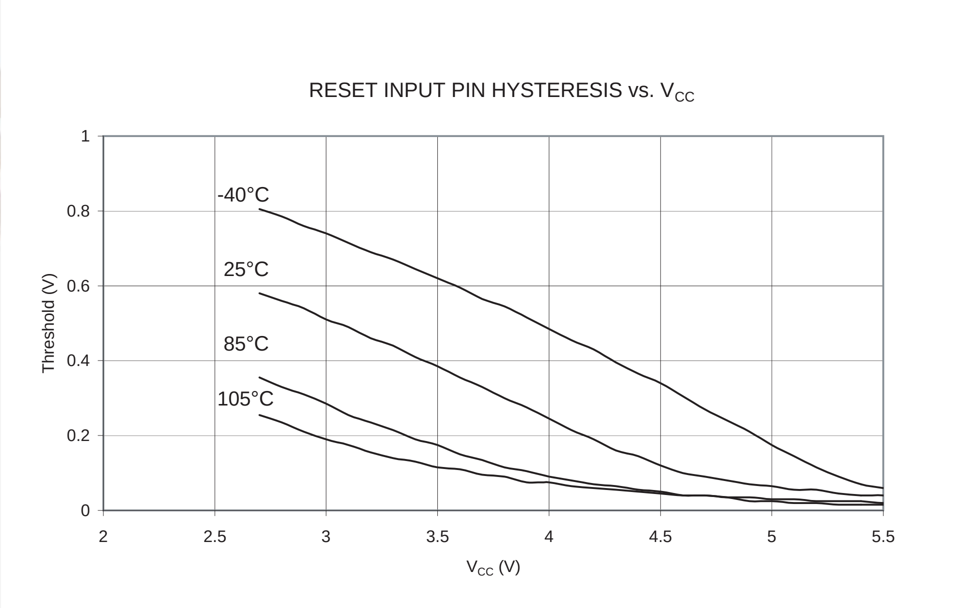


Figure 0-31. Reset Input Pin Hysteresis vs. V_{CC}



Bod Thresholds and Analog Comparator Offset

Figure 0-32. BOD Thresholds vs. Temperature (BOD Level is 4.0V)

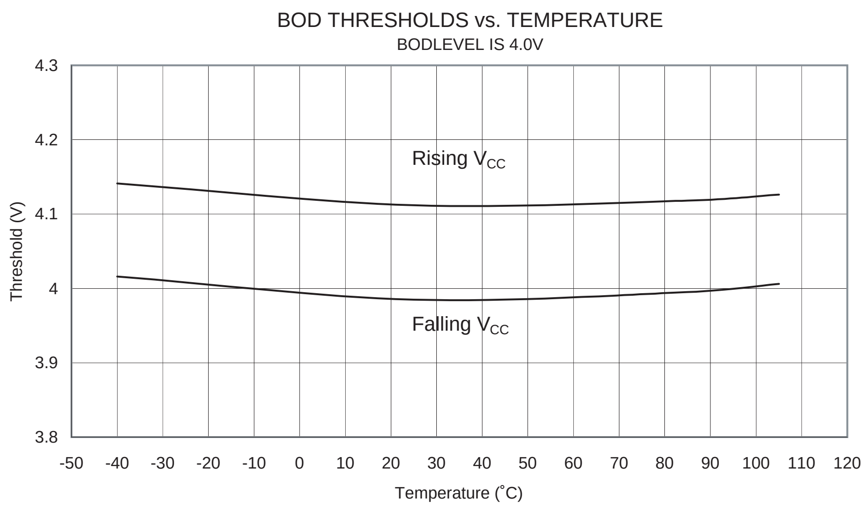


Figure 0-33. BOD Thresholds vs. Temperature (BOD Level is 2.7V)

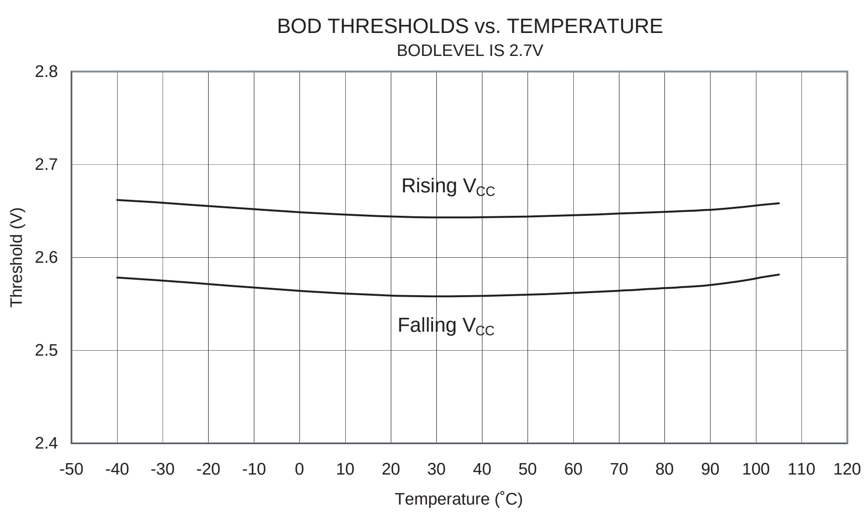


Figure 0-34. Bandgap Voltage vs. V_{CC}

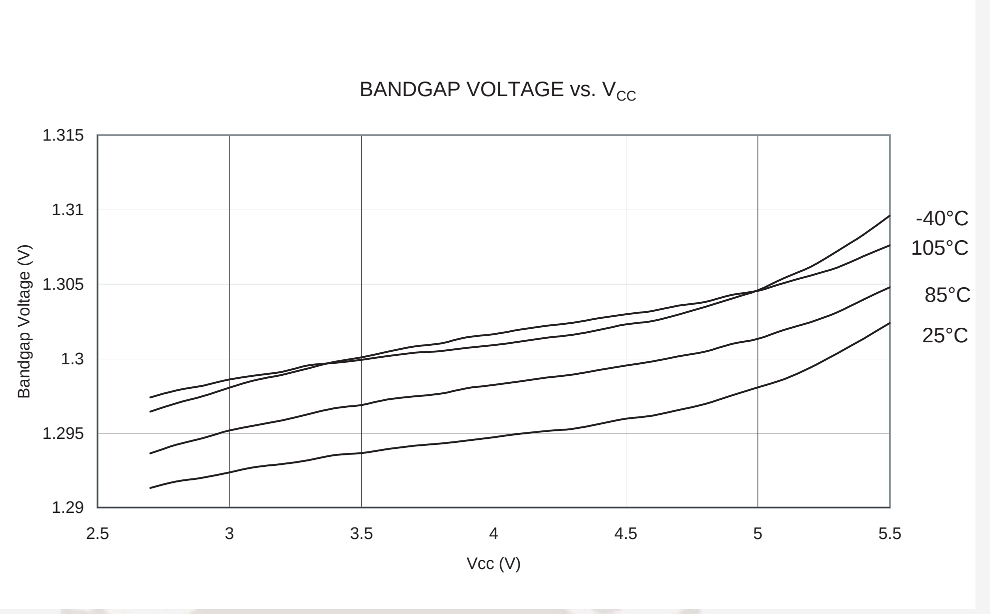


Figure 0-35. Analog Comparator Offset Voltage vs. Common Mode Voltage (V_{CC} = 5V)

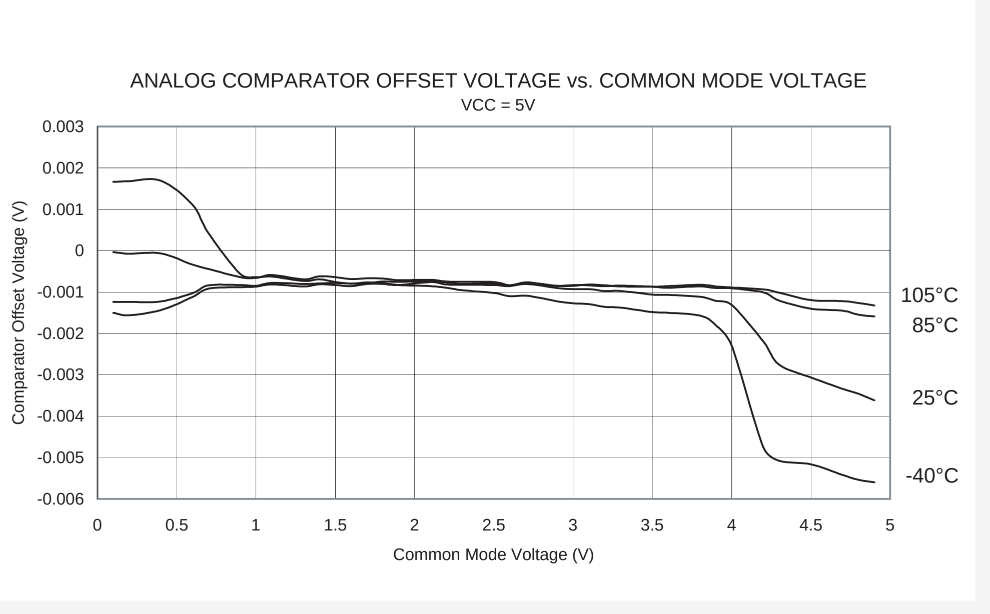
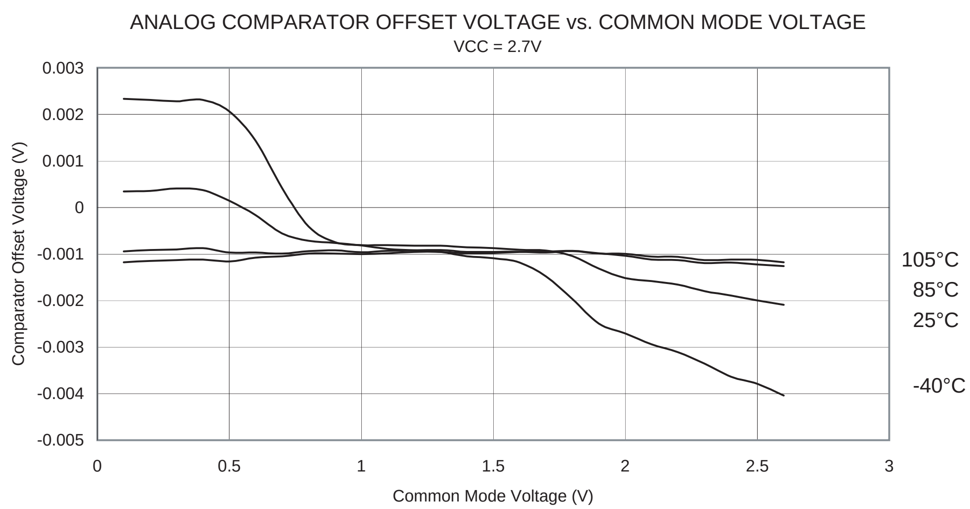


Figure 0-36. Analog Comparator Offset Voltage vs. Common Mode Voltage (V_{CC} = 2.7V)



Internal Oscillator Speed

Figure 0-37. Watchdog Oscillator Frequency vs. V_{CC}

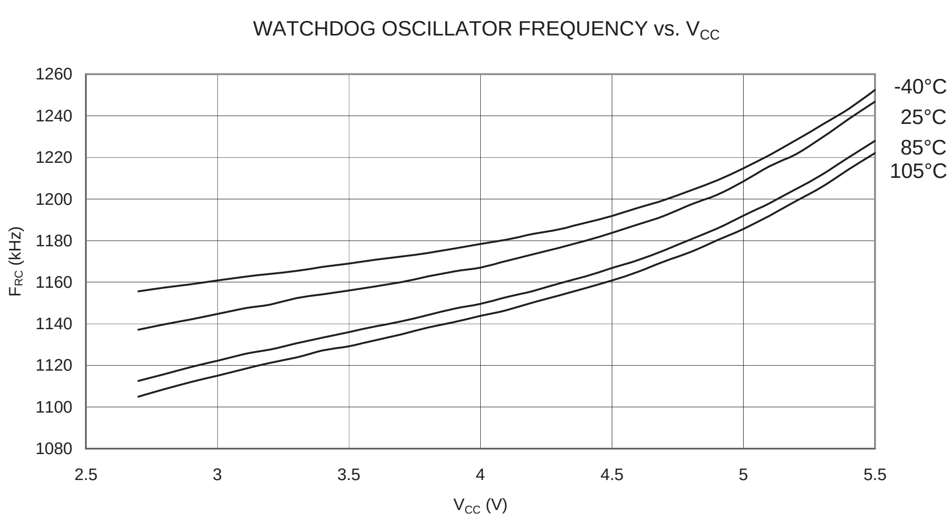


Figure 0-38. Calibrated 8 MHz RC Oscillator Frequency vs. Temperature

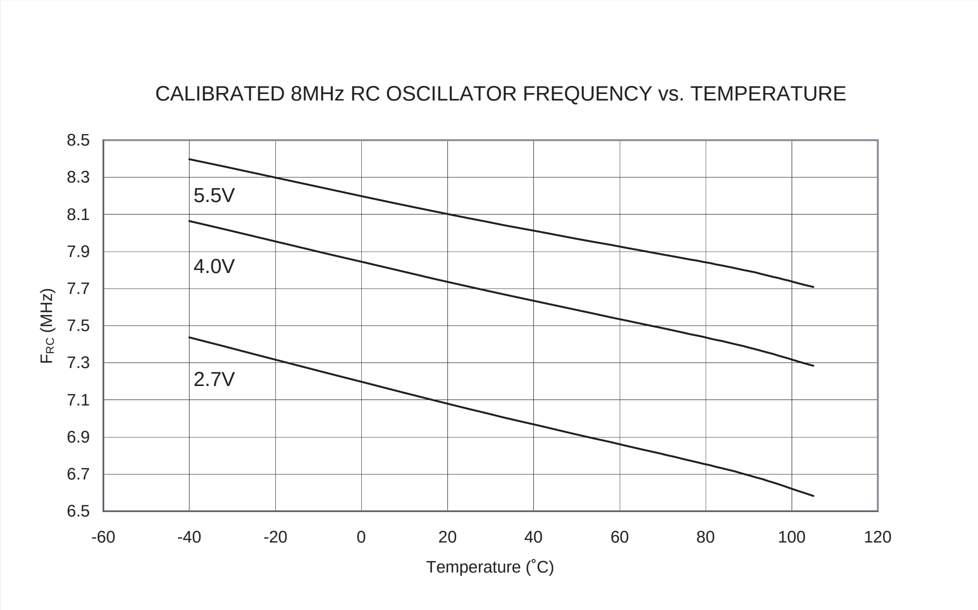


Figure 0-39. Calibrated 8 MHz RC Oscillator Frequency vs. V_{CC}

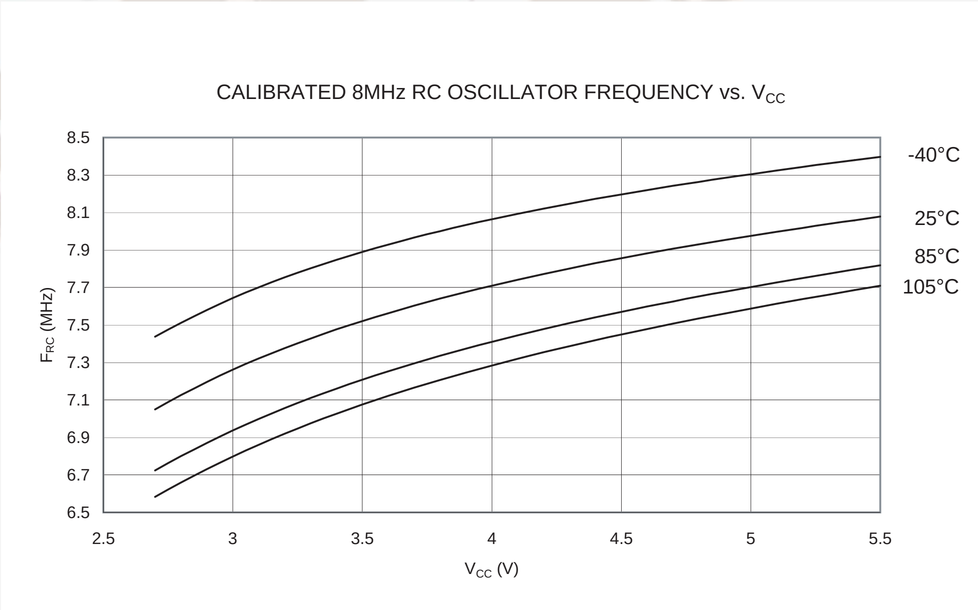


Figure 0-40. Calibrated 4 MHz RC Oscillator Frequency vs. Temperature

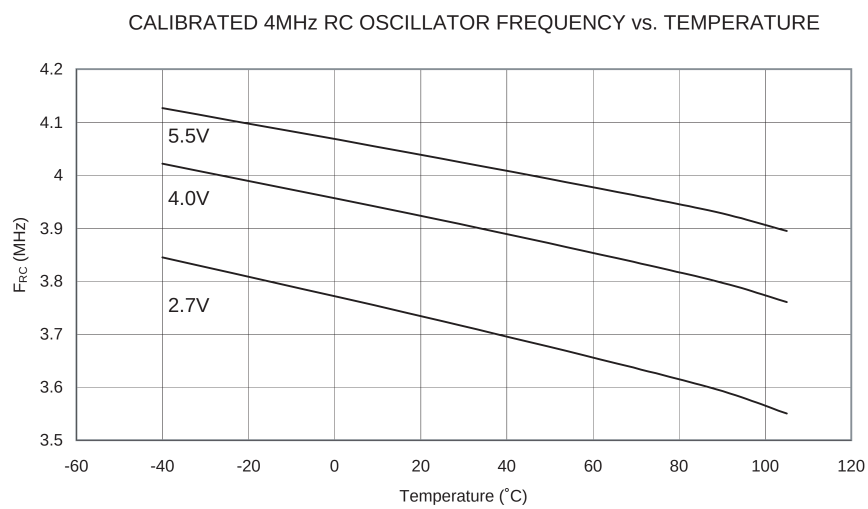


Figure 0-41. Calibrated 4 MHz RC Oscillator Frequency vs. V_{CC}

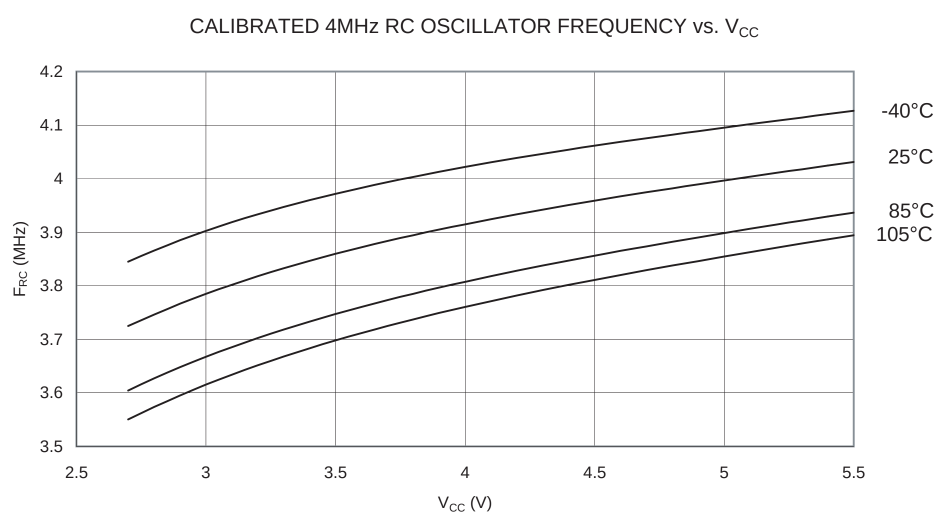


Figure 0-42. Calibrated 2 MHz RC Oscillator Frequency vs. Temperature

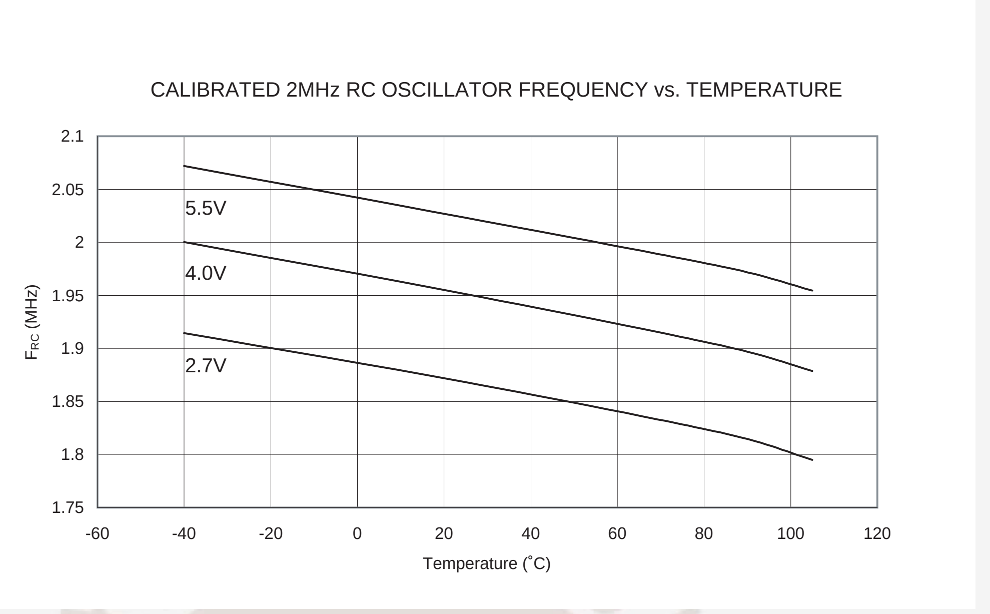


Figure 0-43. Calibrated 2 MHz RC Oscillator Frequency vs. V_{CC}

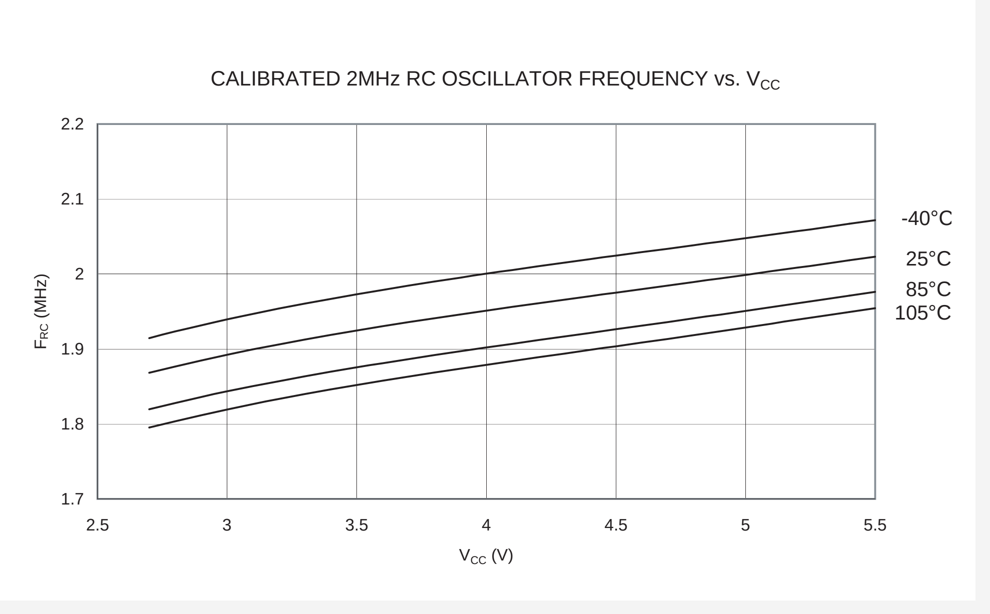


Figure 0-44. Calibrated 1 MHz RC Oscillator Frequency vs. Temperature

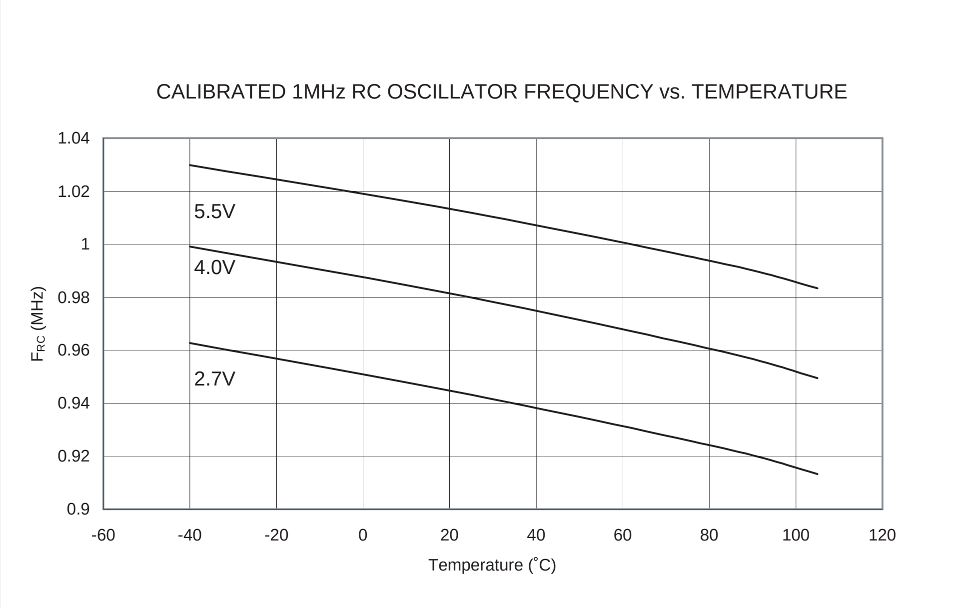
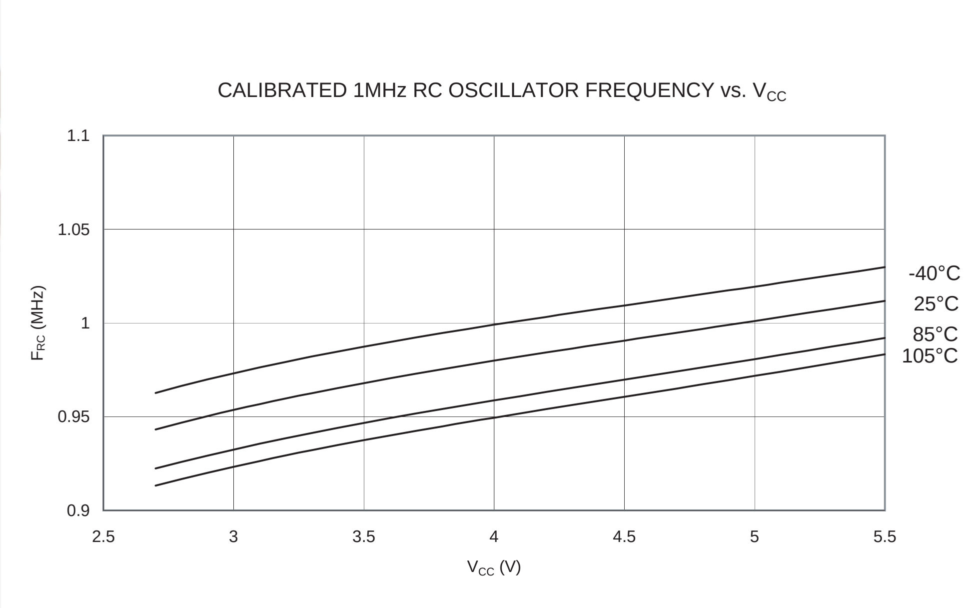


Figure 0-45. Calibrated 1 MHz RC Oscillator Frequency vs. V_{CC}



Current Consumption of Peripheral Units

Figure 0-46. Brown-out Detector Current vs. V_{CC}

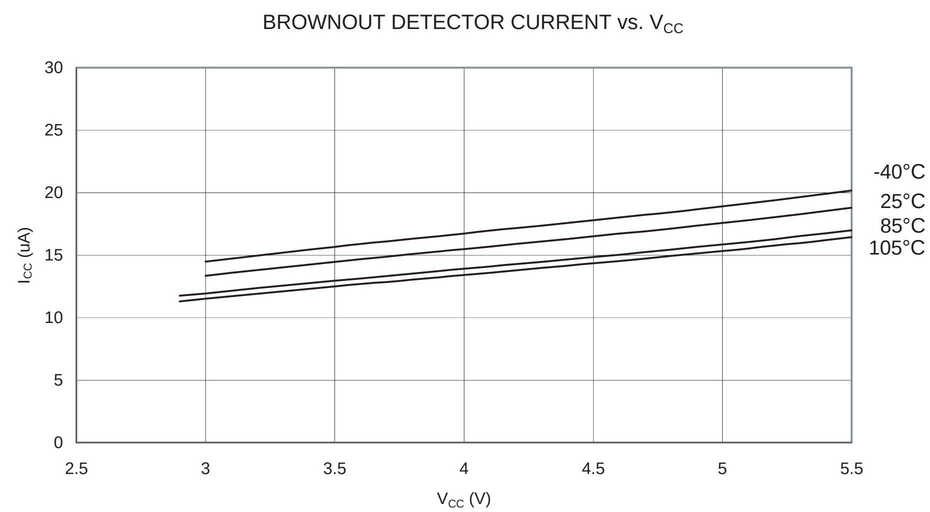


Figure 0-47. ADC Current vs. V_{CC} (AREF = AVCC)

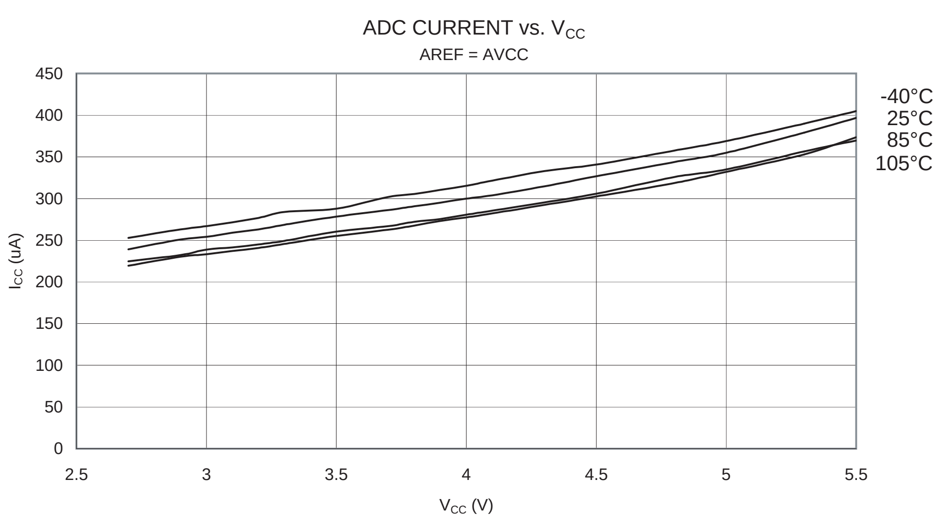


Figure 0-48. AREF External Reference Current vs. V_{CC}

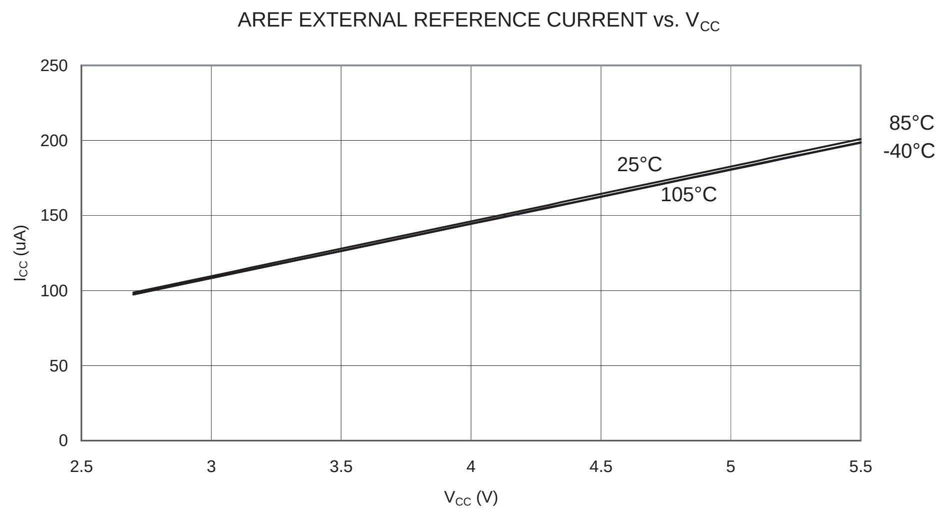


Figure 0-49. Watchdog Timer Current vs. V_{CC}

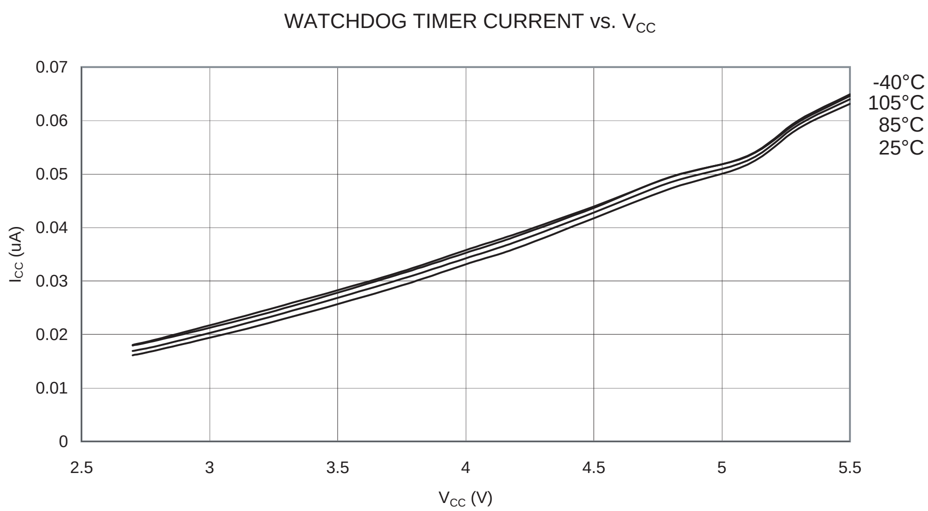


Figure 0-50. Analog Comparator Current vs. V_{CC}

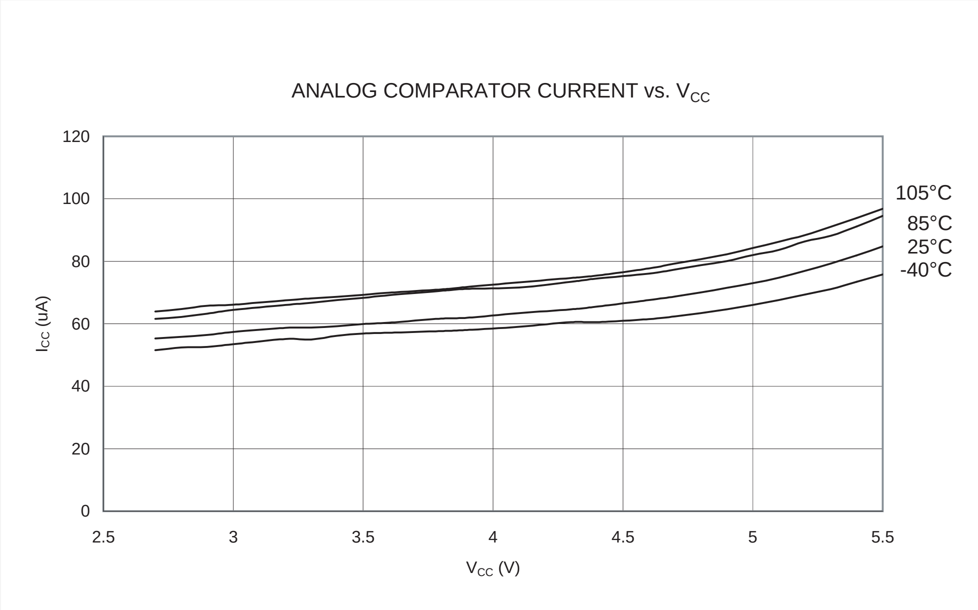
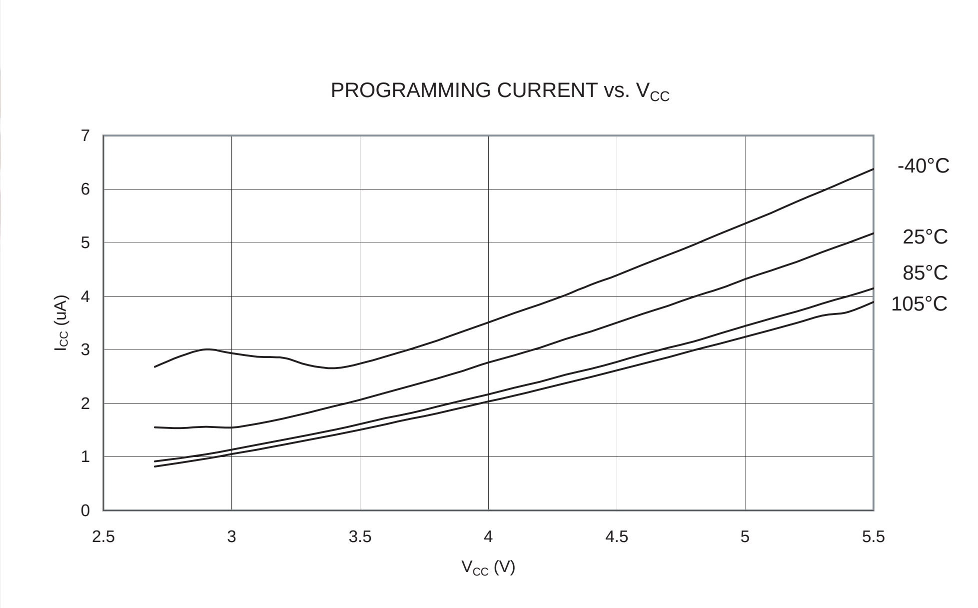
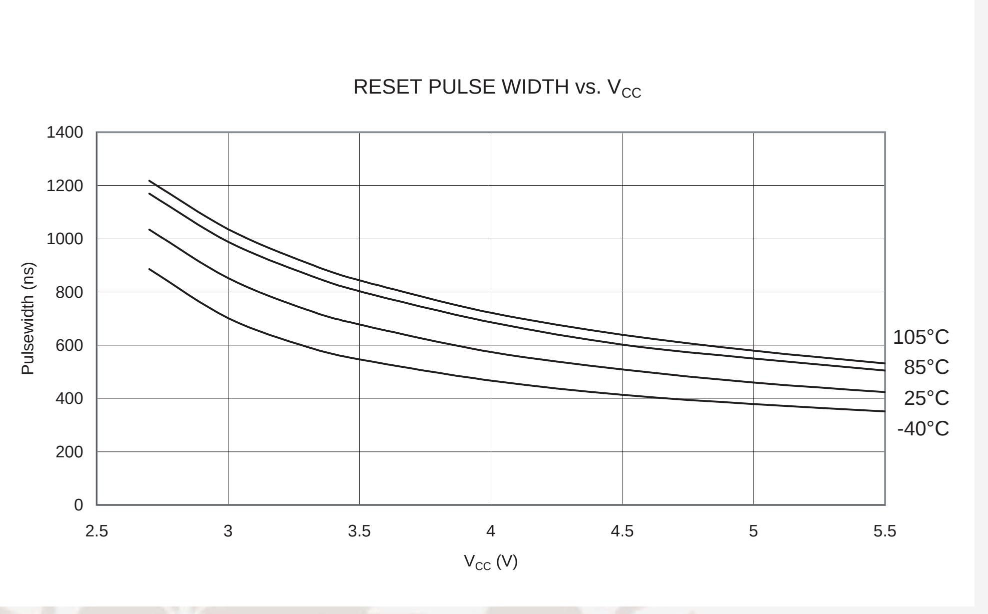


Figure 0-51. Programming Current vs. V_{CC}



Current Consumption in Reset and Reset Pulsewidth

Figure 0-52. Reset Pulse Width vs. V_{CC}



Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	11
0x3E (0x5E)	SPH	–	–	–	–	–	SP10	SP9	SP8	13
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	13
0x3C (0x5C)	Reserved									
0x3B (0x5B)	GICR	INT1	INT0	–	–	–	–	IVSEL	IVCE	49, 67
0x3A (0x5A)	GIFR	INTF1	INTF0	–	–	–	–	–	–	67
0x39 (0x59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	72, 100, 119
0x38 (0x58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	–	TOV0	72, 101, 119
0x37 (0x57)	SPMCR	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	206
0x36 (0x56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	165
0x35 (0x55)	MCUCR	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	33, 66
0x34 (0x54)	MCUCSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	41
0x33 (0x53)	TCCR0	–	–	–	–	–	CS02	CS01	CS00	71
0x32 (0x52)	TCNT0	Timer/Counter0 (8 Bits)								72
0x31 (0x51)	OSCCAL	Oscillator Calibration Register								31
0x30 (0x50)	SFIOR	–	–	–	–	ACME	PUD	PSR2	PSR10	58, 74, 120, 186
0x2F (0x4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	96
0x2E (0x4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	98
0x2D (0x4D)	TCNT1H	Timer/Counter1 – Counter Register High byte								99
0x2C (0x4C)	TCNT1L	Timer/Counter1 – Counter Register Low byte								99
0x2B (0x4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High byte								99
0x2A (0x4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low byte								99
0x29 (0x49)	OCR1BH	Timer/Counter1 – Output Compare Register B High byte								99
0x28 (0x48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low byte								99
0x27 (0x47)	ICR1H	Timer/Counter1 – Input Capture Register High byte								100
0x26 (0x46)	ICR1L	Timer/Counter1 – Input Capture Register Low byte								100
0x25 (0x45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	114
0x24 (0x44)	TCNT2	Timer/Counter2 (8 Bits)								116
0x23 (0x43)	OCR2	Timer/Counter2 Output Compare Register								116
0x22 (0x42)	ASSR	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	117
0x21 (0x41)	WDTCR	–	–	–	WDCE	WDE	WDP2	WDP1	WDP0	43
0x20[1] (0x40)[1]	UBRRH	URSEL	–	–	–	UBRR[11:8]				152
	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	150
0x1F (0x3F)	EEARH	–	–	–	–	–	–	–	EEAR8	20
0x1E (0x3E)	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	20
0x1D (0x3D)	EEDR	EEPROM Data Register								20
0x1C (0x3C)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE	20
0x1B (0x3B)	Reserved									
0x1A (0x3A)	Reserved									
0x19 (0x39)	Reserved									
0x18 (0x38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	65
0x17 (0x37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	65
0x16 (0x36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	65
0x15 (0x35)	PORTC	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	65
0x14 (0x34)	DDRC	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	65
0x13 (0x33)	PINC	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	65
0x12 (0x32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	65
0x11 (0x31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	65
0x10 (0x30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	65
0x0F (0x2F)	SPDR	SPI Data Register								127
0x0E (0x2E)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	126
0x0D (0x2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	125
0x0C (0x2C)	UDR	USART I/O Data Register								148
0x0B (0x2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	148
0x0A (0x2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	149
0x09 (0x29)	UBRRL	USART Baud Rate Register Low byte								152
0x08 (0x28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	186
0x07 (0x27)	ADMUX	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	199
0x06 (0x26)	ADCSRA	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	200
0x05 (0x25)	ADCH	ADC Data Register High byte								201
0x04 (0x24)	ADCL	ADC Data Register Low byte								201
0x03 (0x23)	TWDR	Two-wire Serial Interface Data Register								167
0x02 (0x22)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	167

Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x01 (0x21)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	166
0x00 (0x20)	TWBR	Two-wire Serial Interface Bit Rate Register								165

Notes: 1. Refer to the USART description ("USART" on page 129) for details on how to access UBRRH and UCSRC ("Accessing UBRRH/UCSRC Registers" on page 146)

2. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written

3. 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 0x00 to 0x1F only

Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z, C, N, V, H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z, C, N, V, H	1
ADIW	Rdl,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	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z, C, N, V, S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • 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 ← Rd ⊕ Rr	Z, N, V	1
COM	Rd	One's Complement	Rd ← 0xFF − Rd	Z, C, N, V	1
NEG	Rd	Two's Complement	Rd ← 0x00 − 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 • (0xFF - 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	Set Register	Rd ← 0xFF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z, C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x 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 ← (Rd x Rr) << 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 ← (Rd x Rr) << 1	Z, C	2
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	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 ← 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= 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

Instruction Set Summary (Continued)

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 ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (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 ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (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 ← Y - 1, (Y) ← 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 ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← 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 ← (Z), Z ← 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	Rd	Pop Register from Stack	Rd ← STACK	None	2
BIT AND BIT-TEST INSTRUCTIONS					
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) ← Rd(n), Rd(0) ← 0	Z, C, N, V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z, C, N, V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z, C, N, V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z, C, N, V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z, C, N, V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	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	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	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

Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	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

Ordering Information

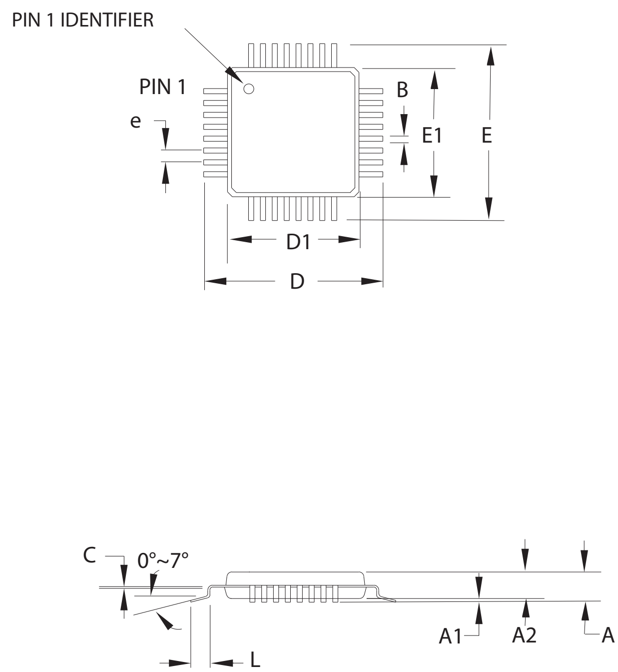
Speed (MHz)	Power Supply (V)	Ordering Code[2]	Package[1]	Operation Range
8	2.7 - 5.5	ATmega8L-8AU ATmega8L-8AUR[3] ATmega8L-8PU ATmega8L-8MU ATmega8L-8MUR[3]	32A 32A 28P3 32M1-A 32M1-A	Industrial (-40°C to 85°C)
16	4.5 - 5.5	ATmega8-16AU ATmega8-16AUR[3] ATmega8-16PU ATmega8-16MU ATmega8-16MUR[3]	32A 32A 28P3 32M1-A 32M1-A	
8	2.7 - 5.5	ATmega8L-8AN ATmega8L-8ANR[3] ATmega8L-8PN ATmega8L-8MN ATmega8L-8MUR[3]	32A 32A 28P3 32M1-A 32M1-A	Industrial (-40°C to 105°C)
16	4.5 - 5.5	ATmega8-16AN ATmega8-16ANR[3] ATmega8-16PN ATmega8-16MN ATmega8-16MUR[3]	32A 32A 28P3 32M1-A 32M1-A	

Notes:
1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities
2. Pb-free packaging complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green
3. Tape & Reel
4. See characterization specification at 105°C

Package Type	
32A	32-lead, Thin (1.0mm) Plastic Quad Flat Package (TQFP)
28P3	28-lead, 0.300” Wide, Plastic Dual Inline Package (PDIP)
32M1-A	32-pad, 5 × 5 × 1.0 body, Lead Pitch 0.50mm Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)

Packaging Information

32A



COMMON DIMENSIONS
(Unit of measure = mm)

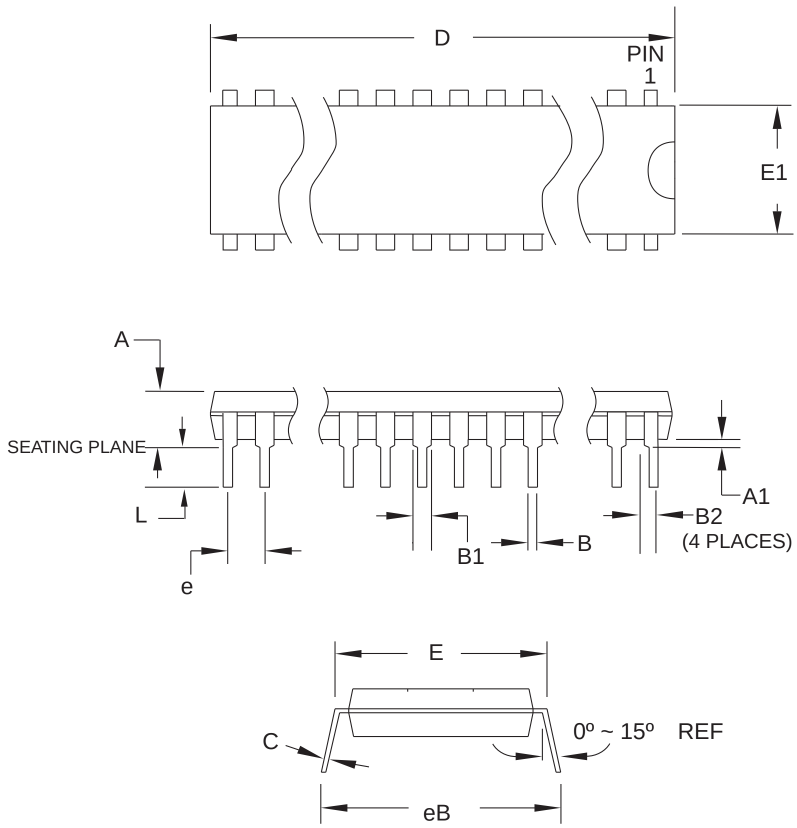
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.20	
A1	0.05	–	0.15	
A2	0.95	1.00	1.05	
D	8.75	9.00	9.25	
D1	6.90	7.00	7.10	Note 2
E	8.75	9.00	9.25	
E1	6.90	7.00	7.10	Note 2
B	0.30	–	0.45	
C	0.09	–	0.20	
L	0.45	–	0.75	
e		0.80 TYP		

Notes:
1. This package conforms to JEDEC reference MS-026, Variation ABA.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
3. Lead coplanarity is 0.10mm maximum.

2010-10-20

	TITLE	DRAWING NO.	REV.
Atmel	**32A,** 32-lead, 7 x 7mm body size, 1.0mm body thickness, 0.8mm lead pitch, thin profile plastic quad flat package (TQFP)	32A	C

28P3



Note: 1. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25mm (0.010").

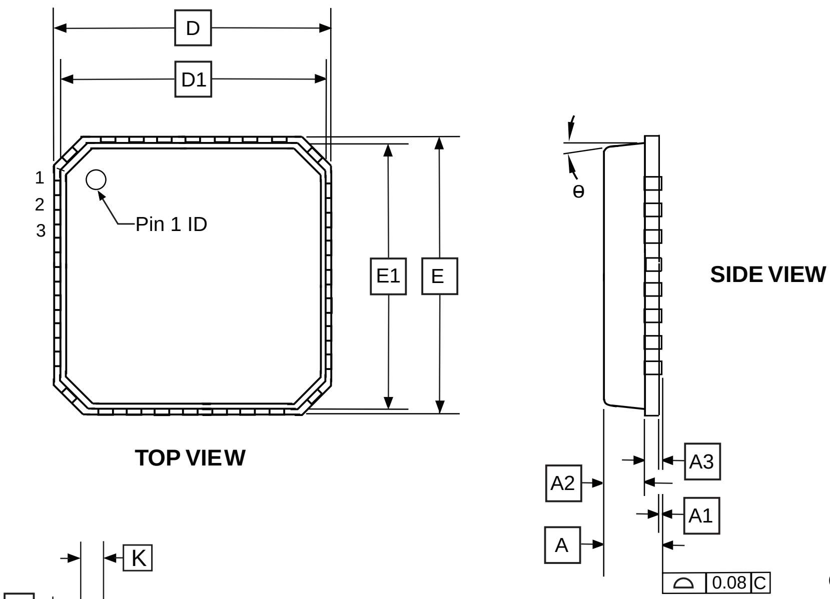
COMMON DIMENSIONS
(Unit of Measure = mm)

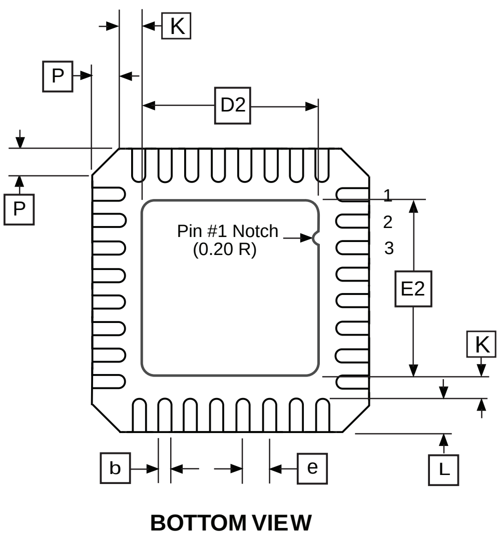
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	4.5724	
A1	0.508	–	–	
D	34.544	–	34.798	Note 1
E	7.620	–	8.255	
E1	7.112	–	7.493	Note 1
B	0.381	–	0.533	
B1	1.143	–	1.397	
B2	0.762	–	1.143	
L	3.175	–	3.429	
C	0.203	–	0.356	
eB	–	–	10.160	
e	2.540 TYP			

09/28/01

	TITLE	DRAWING NO.	REV.
ATMEL 2325 Orchard Parkway San Jose, CA 95131	**28P3**, 28-lead (0.300"/7.62mm Wide) Plastic Dual Inline Package (PDIP)	28P3	B

32M1-A





COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.80	0.90	1.00	
A1	–	0.02	0.05	
A2	–	0.65	1.00	
A3		0.20 REF		
b	0.18	0.23	0.30	
D	4.90	5.00	5.10	
D1	4.70	4.75	4.80	
D2	2.95	3.10	3.25	
E	4.90	5.00	5.10	
E1	4.70	4.75	4.80	
E2	2.95	3.10	3.25	
e		0.50 BSC		
L	0.30	0.40	0.50	
P	–	–	0.60	
θ	–	–	12^{o}	
K	0.20	–	–	

Note: JEDEC Standard MO-220, Fig. 2 (Anvil Singulation), VHHD-2.

5/25/06

	TITLE	DRAWING NO.	REV.
ATMEL 2325 Orchard Parkway San Jose, CA 95131	**32M1-A**, 32-pad, 5 x 5 x 1.0mm Body, Lead Pitch 0.50mm, 3.10mm Exposed Pad, Micro Lead Frame Package (MLF)	32M1-A	E

Errata

The revision letter in this section refers to the revision of the ATmega8 device.

ATmega8 Rev. D to I, M

- **First Analog Comparator conversion may be delayed**
- **Interrupts may be lost when writing the timer registers in the asynchronous timer**
- **Signature may be Erased in Serial Programming Mode**
- **CKOPT Does not Enable Internal Capacitors on XTALn/TOSCn Pins when 32KHz Oscillator is Used to Clock the Asynchronous Timer/Counter2**
- **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. **Signature may be Erased in Serial Programming Mode**

   If the signature bytes are read before a chiperase command is completed, the signature may be erased causing the device ID and calibration bytes to disappear. This is critical, especially, if the part is running on internal RC oscillator.

   Problem Fix / Workaround:

   Ensure that the chiperase command has exceeded before applying the next command.

4. **CKOPT Does not Enable Internal Capacitors on XTALn/TOSCn Pins when 32KHz Oscillator is Used to Clock the Asynchronous Timer/Counter2**

   When the internal RC Oscillator is used as the main clock source, it is possible to run the Timer/Counter2 asynchronously by connecting a 32KHz Oscillator between XTAL1/TOSC1 and XTAL2/TOSC2. But when the internal RC Oscillator is selected as the main clock source, the CKOPT Fuse does not control the internal capacitors on XTAL1/TOSC1 and XTAL2/TOSC2. As long as there are no capacitors connected to XTAL1/TOSC1 and XTAL2/TOSC2, safe operation of the Oscillator is not guaranteed.

   Problem Fix / Workaround

   Use external capacitors in the range of 20pF - 36pF on XTAL1/TOSC1 and XTAL2/TOSC2. This will be fixed in ATmega8 Rev. G where the CKOPT Fuse will control internal capacitors also when internal RC Oscillator is selected as main clock source. For ATmega8 Rev. G, CKOPT = 0 (programmed) will enable the internal capacitors on XTAL1 and XTAL2. Customers who want compatibility between Rev. G and older revisions, must ensure that CKOPT is unprogrammed (CKOPT = 1).

5. 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.

Changes from Rev. 2486Z- 02/11 to Rev. 2486AA- 02/2013

1. Updated the datasheet according to the Atmel new Brand Style Guide.
2. Removed the reference to "On-chip debugging" from the content.
3. Added "Electrical Characteristics – TA = -40°C to 105°C" on page 242.
4. Added "ATmega8 Typical Characteristics – TA = -40°C to 105°C" on page 282.
5. Updated "Ordering Information" on page 314.

Changes from Rev. 2486Y- 10/10 to Rev. 2486Z- 02/11

1. Updated the datasheet according to the Atmel new Brand Style Guide.
2. Updated "Ordering Information" on page 314. Added Ordering Information for "Tape & Reel" devices

Changes from Rev. 2486X- 06/10 to Rev. 2486Y- 10/10

1. Max Rise/Fall time in Table 102 on page 239 has been corrected from 1.6ns to 1600ns.
2. Note is added to "Performing Page Erase by SPM" on page 209.
3. Updated/corrected several short-cuts and added some new ones.
4. Updated last page according to new standard.

Changes from Rev. 2486W- 02/10 to Rev. 2486X- 06/10

1. Updated "DC Characteristics" on page 235 with new V_{OL} maximum value (0.9V and 0.6V).

Changes from Rev. 2486V- 05/09 to Rev. 2486W- 02/10

1. Updated "ADC Characteristics" on page 241 with V_{INT} maximum value (2.9V).

Changes from Rev. 2486U- 08/08 to Rev. 2486V- 05/09

1. Updated "Errata" on page 318.
2. Updated the last page with Atmel's new adresses.

Changes from Rev. 2486T- 05/08 to Rev. 2486U- 08/08

1. Updated "DC Characteristics" on page 235 with I_{CC} typical values.

Changes from Rev. 2486S- 08/07 to Rev. 2486T- 05/08

1. Updated Table 98 on page 233.
2. Updated "Ordering Information" on page 314.
   - Commercial Ordering Code removed.
   - No Pb-free packaging option removed.

Changes from Rev. 2486R- 07/07 to Rev. 2486S- 08/07

1. Updated "Features" on page 1.
2. Added "Data Retention" on page 7.
3. Updated "Errata" on page 318.
4. Updated "Slave Mode" on page 125.

Changes from Rev. 2486Q- 10/06 to Rev. 2486R- 07/07

1. Added text to Table 81 on page 211.
2. Fixed typo in "Peripheral Features" on page 1.
3. Updated Table 16 on page 42.
4. Updated Table 75 on page 199.
5. Removed redundancy and updated typo in Notes section of "DC Characteristics" on page 235.

Changes from Rev. 2486P- 02/06 to Rev. 2486Q- 10/06

1. Updated "Timer/Counter Oscillator" on page 32.
2. Updated "Fast PWM Mode" on page 88.
3. Updated code example in "USART Initialization" on page 134.
4. Updated Table 37 on page 96, Table 39 on page 97, Table 42 on page 115, Table 44 on page 115, and Table 98 on page 233.
5. Updated "Errata" on page 318.

Changes from Rev. 2486O-10/04 to Rev. 2486P- 02/06

1. Added "Resources" on page 7.
2. Updated "External Clock" on page 32.
3. Updated "Serial Peripheral Interface – SPI" on page 121.
4. Updated Code Example in "USART Initialization" on page 134.
5. Updated Note in "Bit Rate Generator Unit" on page 164.
6. Updated Table 98 on page 233.
7. Updated Note in Table 103 on page 241.

8. Updated “Errata” on page 318.

Changes from Rev. 2486N-09/04 to Rev. 2486O-10/04

1. Removed to instances of “analog ground”. Replaced by “ground”.
2. Updated Table 7 on page 29, Table 15 on page 38, and Table 100 on page 237.
3. Updated “Calibrated Internal RC Oscillator” on page 30 with the 1MHz default value.
4. Table 89 on page 218 and Table 90 on page 218 moved to new section “Page Size” on page 218.
5. Updated descripton for bit 4 in “Store Program Memory Control Register – SPMCR” on page 206.
6. Updated “Ordering Information” on page 314.

Changes from Rev. 2486M-12/03 to Rev. 2486N-09/04

1. Added note to MLF package in “Pin Configurations” on page 2.
2. Updated “Internal Voltage Reference Characteristics” on page 42.
3. Updated “DC Characteristics” on page 235.
4. ADC4 and ADC5 support 10-bit accuracy. Document updated to reflect this.
   Updated features in “Analog-to-Digital Converter” on page 189.
   Updated “ADC Characteristics” on page 241.
5. Removed reference to “External RC Oscillator application note” from “External RC Oscillator” on page 28.

Changes from Rev. 2486L-10/03 to Rev. 2486M-12/03

1. Updated “Calibrated Internal RC Oscillator” on page 30.

Changes from Rev. 2486K-08/03 to Rev. 2486L-10/03

1. Removed “Preliminary” and TBDs from the datasheet.
2. Renamed ICP to ICP1 in the datasheet.
3. Removed instructions CALL and JMP from the datasheet.
4. Updated t_{RST} in Table 15 on page 38, V_{BG} in Table 16 on page 42, Table 100 on page 237 and Table 102 on page 239.
5. Replaced text “XTAL1 and XTAL2 should be left unconnected (NC)” after Table 9 in “Calibrated Internal RC Oscillator” on page 30. Added text regarding XTAL1/XTAL2 and CKOPT Fuse in “Timer/Counter Oscillator” on page 32.
6. Updated Watchdog Timer code examples in “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 45.
7. Removed bit 4, ADHSM, from “Special Function IO Register – SFIOR” on page 58.
8. Added note 2 to Figure 103 on page 208.

9. Updated item 4 in the "Serial Programming Algorithm" on page 231.

10. Added t_{WD_FUSE} to Table 97 on page 232 and updated Read Calibration Byte, Byte 3, in Table 98 on page 233.

11. Updated Absolute Maximum Ratings* and DC Characteristics in "Electrical Characteristics – TA = -40°C to 85°C" on page 235.

Changes from Rev. 2486J-02/03 to Rev. 2486K-08/03

1. Updated V_{BOT} values in Table 15 on page 38.

2. Updated "ADC Characteristics" on page 241.

3. Updated "ATmega8 Typical Characteristics – TA = -40°C to 85°C" on page 244.

4. Updated "Errata" on page 318.

Changes from Rev. 2486I-12/02 to Rev. 2486J-02/03

1. Improved the description of "Asynchronous Timer Clock – clkASY" on page 26.

2. Removed reference to the "Multipurpose Oscillator" application note and the "32kHz Crystal Oscillator" application note, which do not exist.

3. Corrected OCn waveforms in Figure 38 on page 89.

4. Various minor Timer 1 corrections.

5. Various minor TWI corrections.

6. Added note under "Filling the Temporary Buffer (Page Loading)" on page 209 about writing to the EEPROM during an SPM Page load.

7. Removed ADHSM completely.

8. Added section "EEPROM Write during Power-down Sleep Mode" on page 23.

9. Removed XTAL1 and XTAL2 description on page 5 because they were already described as part of "Port B (PB7..PB0) XTAL1/XTAL2/TOSC1/TOSC2" on page 5.

10. Improved the table under "SPI Timing Characteristics" on page 239 and removed the table under "SPI Serial Programming Characteristics" on page 234.

11. Corrected PC6 in "Alternate Functions of Port C" on page 61.

12. Corrected PB6 and PB7 in "Alternate Functions of Port B" on page 58.

13. Corrected 230.4 Mbps to 230.4 kbps under "Examples of Baud Rate Setting" on page 153.

14. Added information about PWM symmetry for Timer 2 in "Phase Correct PWM Mode" on page 111.

15. Added thick lines around accessible registers in Figure 76 on page 163.

16. Changed "will be ignored" to "must be written to zero" for unused Z-pointer bits under "Performing a Page Write" on page 209.

17. Added note for RSTDISBL Fuse in Table 87 on page 216.

18. Updated drawings in "Packaging Information" on page 315.

Changes from Rev. 2486H-09/02 to Rev. 2486I-12/02

1. Added errata for Rev D, E, and F on page 318.

Changes from Rev. 2486G-09/02 to Rev. 2486H-09/02

1. Changed the Endurance on the Flash to 10,000 Write/Erase Cycles.

Changes from Rev. 2486F-07/02 to Rev. 2486G-09/02

1. Updated Table 103, "ADC Characteristics," on page 241.

Changes from Rev. 2486E-06/02 to Rev. 2486F-07/02

1. Changes in "Digital Input Enable and Sleep Modes" on page 55.

2. Addition of OCS2 in "MOSI/OC2 – Port B, Bit 3" on page 59.

3. The following tables have been updated:

   Table 51, "CPOL and CPHA Functionality," on page 127, Table 59, "UCPOL Bit Settings," on page 152, Table 72, "Analog Comparator Multiplexed Input(1)," on page 188, Table 73, "ADC Conversion Time," on page 193, Table 75, "Input Channel Selections," on page 199, and Table 84, "Explanation of Different Variables used in Figure 103 on page 208 and the Mapping to the Z-pointer," on page 214.

4. Changes in "Reading the Calibration Byte" on page 227.

5. Corrected Errors in Cross References.

Changes from Rev. 2486D-03/02 to Rev. 2486E-06/02

1. Updated Some Preliminary Test Limits and Characterization Data

   The following tables have been updated:

   Table 15, "Reset Characteristics," on page 38, Table 16, "Internal Voltage Reference Characteristics," on page 42, DC Characteristics on page 235, Table , "ADC Characteristics," on page 241.

2. Changes in External Clock Frequency

   Added the description at the end of "External Clock" on page 32.

   Added period changing data in Table 99, "External Clock Drive," on page 237.

3. Updated TWI Chapter

   More details regarding use of the TWI bit rate prescaler and a Table 65, "TWI Bit Rate Prescaler," on page 167.

Changes from Rev. 2486C-03/02 to Rev. 2486D-03/02

1. **Updated Typical Start-up Times.**

   The following tables has been updated:

   Table 5, "Start-up Times for the Crystal Oscillator Clock Selection," on page 28, Table 6, "Start-up Times for the Low-frequency Crystal Oscillator Clock Selection," on page 28, Table 8, "Start-up Times for the External RC Oscillator Clock Selection," on page 29, and Table 12, "Start-up Times for the External Clock Selection," on page 32.

2. **Added "ATmega8 Typical Characteristics – TA = -40°C to 85°C" on page 244.**

Changes from Rev. 2486B-12/01 to Rev. 2486C-03/02

1. **Updated TWI Chapter.**

   More details regarding use of the TWI Power-down operation and using the TWI as Master with low TWBRR values are added into the datasheet.

   Added the note at the end of the "Bit Rate Generator Unit" on page 164.

   Added the description at the end of "Address Match Unit" on page 164.

2. **Updated Description of OSCCAL Calibration Byte.**

   In the datasheet, it was not explained how to take advantage of the calibration bytes for 2, 4, and 8MHz Oscillator selections. This is now added in the following sections:

   Improved description of "Oscillator Calibration Register – OSCCAL" on page 31 and "Calibration Byte" on page 218.

3. **Added Some Preliminary Test Limits and Characterization Data.**

   Removed some of the TBD's in the following tables and pages:

   Table 3 on page 26, Table 15 on page 38, Table 16 on page 42, Table 17 on page 44, "TA = -40°C to +85°C, VCC = 2.7V to 5.5V (unless otherwise noted)" on page 235, Table 99 on page 237, and Table 102 on page 239.

4. **Updated Programming Figures.**

   Figure 104 on page 219 and Figure 112 on page 230 are updated to also reflect that AV_{CC} must be connected during Programming mode.

5. **Added a Description on how to Enter Parallel Programming Mode if RESET Pin is Disabled or if External Oscillators are Selected.**

   Added a note in section "Enter Programming Mode" on page 221.

Table of Contents

Table of Contents 1

Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1) (408) 441-0311
Fax: (+1) (408) 487-2600
www.atmel.com

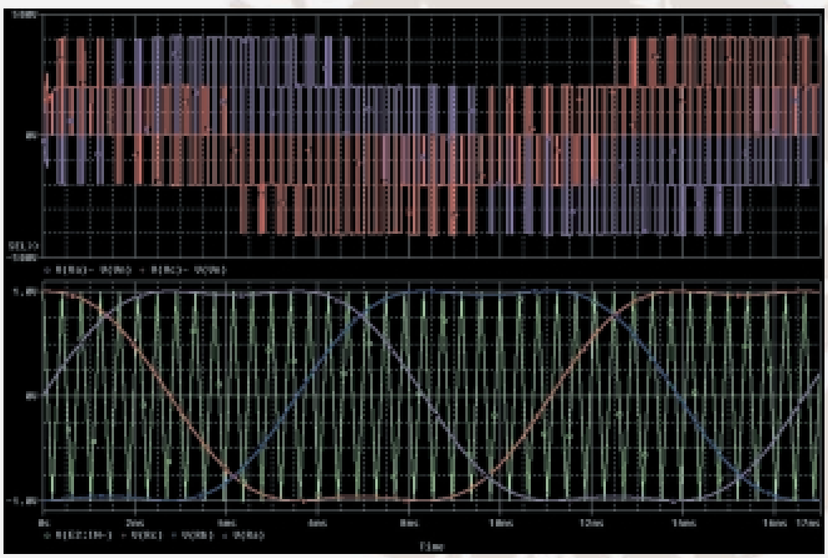
Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Roa
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81) (3) 6417-0300
Fax: (+81) (3) 6417-0370

Modulación por ancho de pulso (PWM) y modulación vectorial (SVM). Una introducción a las técnicas de modulación

Johnny Posada Contreras*



Resumen

Los conversores DC/AC tienen por objetivo la transformación de tensión DC a tensión AC de amplitud y/o frecuencia variable dependiendo de la aplicación. El proceso de conversión de voltaje se logra mediante la implementación de técnicas de modulación, las cuales actúan sobre un puente inversor monofásico o trifásico. Según las características de estas técnicas, las propiedades de eficiencia en la conversión, contenido armónico de la señal de salida y pérdidas en el puente inversor cambian. En el presente artículo se da un repaso de diferentes técnicas de modulación escalares (PWM) y vectoriales (SVM), enfatizando en las vectoriales por ser las más utilizadas actualmente en los sistemas drive para motores de inducción y en

* Universidad Autónoma de Occidente. Departamento de Automática y Electrónica. Grupo de Investigación en Energías (GIEN). Línea de investigación en Máquinas Eléctricas y Electrónica de Potencia (LIMEEP). e-mail: jposada@uao.edu.co

Fecha de recepción: 04/13/05, Fecha de aprobación: 06/24/05

sistemas de alimentación trifásica, a la vez que presenta las mejores características de desempeño que las técnicas de modulación escalares o PWM (Modulación por Ancho de Pulso).

Palabras clave: Hexágono de tensiones, modulación vectorial, puente inversor trifásico.

Abstract

Converters DC/AC haven by objective the transformation of DC voltage to AC voltage of variable amplitude and/or variable frequency depending of the application. The process of voltage conversion is obtained by means of the implementation of modulation techniques which act on a single-phase or three-phase inverter bridge. According to the characteristics of these techniques, the properties of efficiency in the conversion, harmonic content of the output signal and losses in the inverter bridge change. In the present paper a review of different modulation techniques scalar (PWM) and vectorial (SVM) is made, in special the last techniques, since it is the most used at the moment in the systems drives for induction motors and in systems of three-phase feeding, simultaneously that presents the best characteristics of performance than the modulation techniques you will scale or PWM (Pulse Wide Modulation).

Key words: Voltage hexagon, space vector modulation, three phase inverter.

1. Introducción

Los circuitos de conversión DC/AC tienen amplia aplicación en la industria. Son utilizados en variadores de velocidad, sistemas de alimentación ininterrumpida, filtros activos, etc. Los conversores DC/AC se clasifican como inversores con fuente de voltaje (VSI) e inversores con fuente de corriente (CSI).[1,2] Los CSI se usan en sistemas de alta potencia, los VSI se reservan para aplicaciones en baja y mediana potencia. Dentro de esta clasificación existen varias configuraciones de conversores DC/AC que dependen de la aplicación final y el nivel de voltaje o corriente de su salida. En el caso de los drive para motores de baja y mediana potencia, la topología típica es el medio puente inversor trifásico con fuente de voltaje (Figura 1), formado por seis elementos de conmutación Mosfet's, Transistores Bipolares de Compuerta Aislada (IGBT), Tiristores desactivados por Compuerta (GTO) o Tiristores Controlados por MOS (MCT).[3,5]

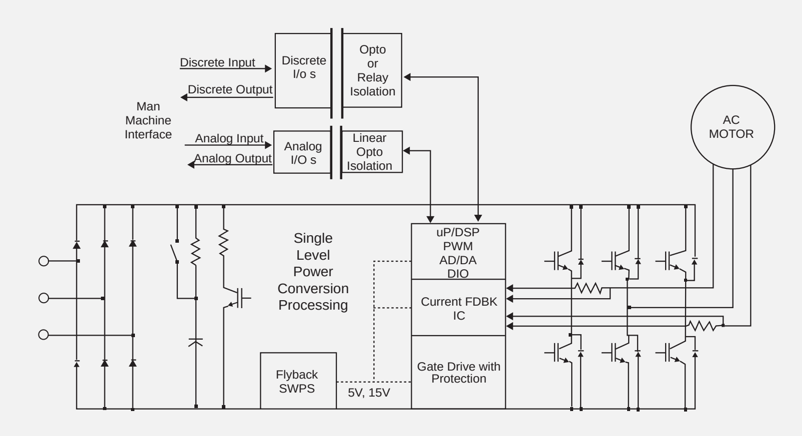


Figura 1. Medio puente inversor trifásico con circuito intermedio de DC.

Adicionalmente, se debe considerar la técnica de modulación que activará los elementos de conmutación. En el inversor con fuente de voltaje la técnica de modulación se encarga de la forma de onda de la señal de salida AC, su nivel de tensión y su frecuencia. Las técnicas de modulación se pueden clasificar en escalares o PWM («Pulse Width Modulation»),[6,9] y vectorialeso SVM («Space Vector Modulation»).[10,13] Entre las técnicas escalares se encuentran la técnica de modulación de onda cuadrada (*six-step*), técnica de modulación sinusoidal, técnica de modulación sinusoidal con tercer armónico, entre otras; divisibles a la vez en técnicas de modulación basadas en portadora triangular (*carrier based*) y técnicas programadas.[7] La técnica de modulación SVM se presenta en los años ochentas la cual maneja el puente inversor trifásico como una unidad y se basa en la representación vectorial del voltaje trifásico para el manejo del puente inversor, disminuye las pérdidas por conmutación en el mismo y minimiza el contenido armónico de la señal de salida.[11,14,16]

2. Técnicas de modulación escalares o PWM

Se usa en inversores DC/AC monofásicos y trifásicos. Se basan en la comparación de una señal de referencia a modular y una señal portadora de forma triangular o diente de sierra (Figura 2); la comparación generará un tren de pulsos de ancho específico que se utilizan en la conmutación del puente inversor. La relación entre la amplitud de la señal portadora y la señal de referencia se llama **«índice de modulación»** y se representa por **«m_a»** (1), donde Ar es la amplitud de la señal de referencia y Ac es la amplitud de la señal portadora. El índice de modulación permite obtener tensión variable a la salida del inversor.

$$m_a = \frac{A_r}{A_c} \quad (1)$$

$$m_f = \frac{F_r}{F_c} \quad (2)$$

La relación entre la frecuencia de la señal portadora y la frecuencia de referencia se denomina **«índice de frecuencia»** y se representa por «m_f» (2), idealmente m_f debe ser mayor a 21 y la frecuencia de la portadora múltiplo de la frecuencia de la señal de referencia.[15] El índice de frecuencia determina la distorsión armónica de la señal de salida la cual es una medida de su contenido armónico. La variación de la señal de referencia y la secuencia de conmutación dan como resultado diferentes técnicas de modulación PWM, cada una modifica la eficiencia de la conversión, las pérdidas por conmutación en el puente inversor y la pureza de la señal de salida.

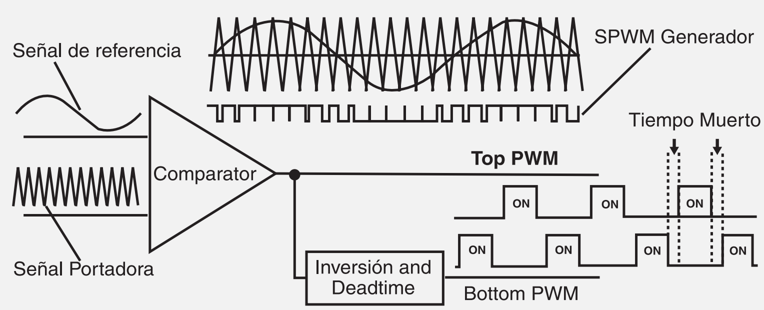


Figura 2. Circuito generador escalar PWM.

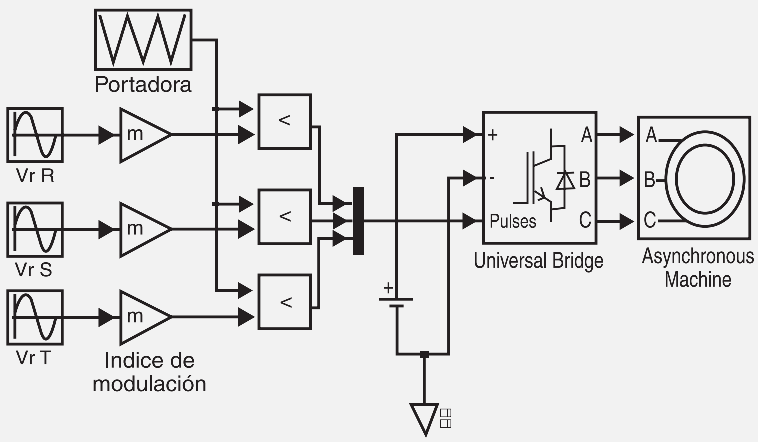


Figura 3. Diagrama en bloques del generador CB-SPWM. Las referencias están desfasadas 120° entre sí.

2.1 Modulación SPWM Bipolar (SPWM o CB-SPWM): genera la inversión de voltaje utilizando un tren de pulsos cuyo ancho depende del tiempo y del nivel de tensión deseado en la salida. La integración en el tiempo de este tren de pulsos representa una señal sinusoidal. Se conoce también como *triangular carrier-based sinusoidal PWM (CB-SPWM)* o método de sub-oscilación y fue propuesto en la década de los años sesenta.[17] El tren de pulsos se forma por la comparación de una portadora triangular a una frecuencia específica fc con tres señales de referencia sinusoidales; *Ua, Ub y Uc* desfasadas 120° entre sí (Figura 3). La tensión de salida en AC contendrá armónicos a múltiplos de la frecuencia de la señal portadora.[15] El máximo valor de tensión, ***Mmax,*** alcanzado por la componente fundamental en la técnica de modulación CB-SPWM, es del orden de π/4 o 0.785 para un índice de modulación «$\mathbf{m_a}$» igual a uno.[1,7,10] Cuando el índice de modulación de amplitud «$\mathbf{m_a}$» sobrepasa la unidad, el inversor trabaja en la zona no lineal y gradualmente alcanzará la amplitud máxima (operación *six-step*).

2.2 Modulación SPWM con tercera armónica: en inversores trifásicos con neutro de carga flotante (Figura 4) la corriente de la carga depende de la tensión de línea y es posible sumar a la señal de referencia señales de secuencia cero o ZSS. Cuando a la señal sinusoidal se suma su tercer armónico, esta no produce distorsión de su voltaje de fase U_{aN}, U_{bN}, U_{cN} y tampoco se afecta la corriente promedio de la carga; pero se puede obtener mayor nivel de tensión con índices de modulación menores mejorando la eficiencia en la conversión DC/AC. La adición del tercer armónico de la señal sinusoidal aumenta en 15.5% la eficiencia de conversión.[10] En la Figura 5 se observan las formas de onda de salida de este tipo de modulación. La técnica de modulación sinusoidal con tercer armónico se implementa de forma similar a la CD-SPWM (Figura 6) y se presenta en 1975 por G. Buja y G. Indri. En general, es posible la adición de cualquier señal ZSS siempre que su frecuencia sea múltiplos triples de la frecuencia a generar. En[18,19 y 35] se expone una diversidad de técnicas basadas en ZSS. El M_{max} alcanzado por esta técnica de modulación es del orden de $U_{dc}/\sqrt{3}$ y corresponde a 0.907 de la fundamental. La forma de onda resultante de la suma de la ZSS se aproxima por (3) y su magnitud no debe sobrepasar la unidad.

$$U_r(x)_{3\text{-}Armonico} = \frac{2}{\sqrt{3}}sin(x) + \frac{1}{3\sqrt{3}}sin(3x) \tag{3}$$

2.3 Modulación PWM a 60°: la técnica de modulación PWM a 60° se basa en la adición de ZSS. El objetivo es «achatar» la forma de onda de voltaje de salida desde los 60° hasta los 120° y desde 240° a 300°. Los dispositivos del puente inversor se mantienen encendidos durante un tercio de ciclo, se presentan menos pérdidas por conmutación. La técnica de modulación PWM a 60° aprovecha mejor la tensión del bus de DC, alcanzando una tensión de fase igual a 0.57735Vdc. La componente de secuencia cero añadida a la señal sinusoidal de referencia se puede aproximar por (4).

$$U_r(x)_{60^\circ} = \frac{2}{\sqrt{3}}sin(x) + \frac{1}{2\pi}sin(3x) + \frac{1}{60\pi}sin(9x) + \frac{1}{280\pi}sin(15x) + ... \tag{4}$$

2.4 Modulación por eliminación de armónicas: pretende una generación de pulsos estratégica para la eliminación de «algunos armónicos»; es similar a la técnica anterior y disminuye las pérdidas por conmutación. Pertenece al grupo de técnicas de modulación programada. Un amplio estudio de estas técnicas de modulación se muestra en [20,21].

3. Técnica de modulación vectorial o SVM

En la técnica SVM el puente inversor es manejado por ocho estados de conmutación. Se considera la mejor alternativa de modulación para inversores ya que maximiza el uso de la tensión DC, su contenido armónico es bajo y minimiza pérdidas por conmutación. Sin embargo, algunos autores indican que su única ventaja es su representación compleja.[43,44] La técnica SVM se propone en 1982 por Pfaff, Weschta y Wick[22] y se desarrolla en 1988 por Broeck, Skudelny y Stanke[11] gracias a los sistemas microprocesadores. Diversos estudios e implementaciones alrededor de esta técnica se presentan en[23,28] La generación de voltaje con la técnica SVM se logra seleccionando adecuadamente y por un tiempo determinado los estados de los interruptores del puente inversor en cada período de conmutación.[29,30]

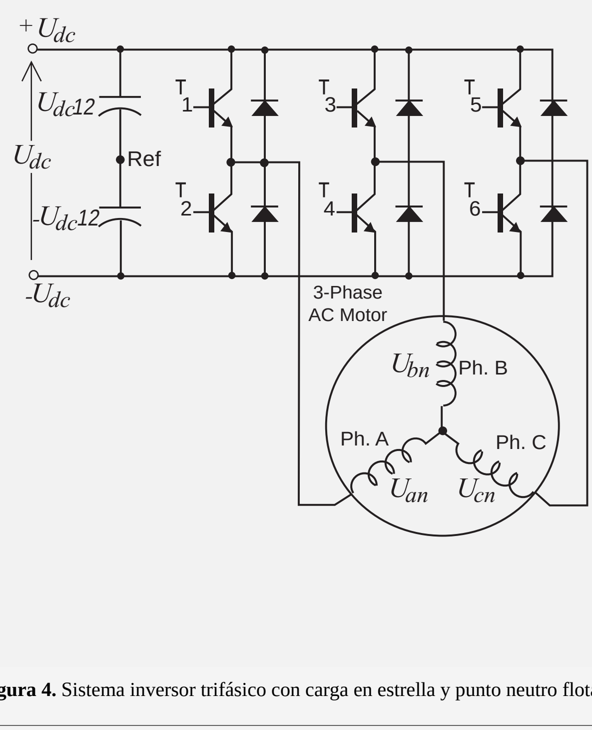
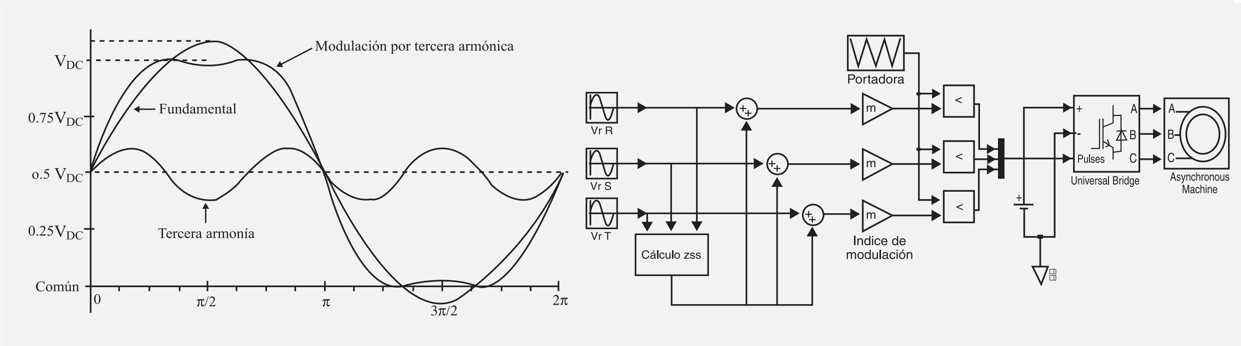


Figura 4. Sistema inversor trifásico con carga en estrella y punto neutro flotante.



(a) (b)

Figura 5. Sistema generador SPWM+ZSS. (a) Formas de onda SPWM + 3er Armónica. (b) Diagrama de bloques generador SPWM + 3er Armónico

3.1 El Hexágono de Tensiones: en un sistema trifásico los voltajes de fase Ua, Ub y Uc se representan por un vector rotatorio «U» de amplitud constante que gira en el plano complejo con frecuencia angular w (frecuencia de la señal de salida) (5).

$$U = \frac{1}{c}\left(U_a + aU_b + a^2 U_c\right) \qquad (5)$$

donde:

$$a = e^{j2\pi/3}, \quad a^2 = e^{j4\pi/3}$$

$$U_a = U_M \sin(\omega t) = U_M e^{j\omega t}$$

$$U_b = U_M \sin(\omega t - 2\pi/3) = U_M e^{j\omega t - 2\pi/3}$$

$$U_c = U_M \sin(\omega t - 4\pi/3) = U_M e^{j\omega t - 4\pi/3}$$

(6)

El coeficiente C puede seleccionarse entre $\sqrt{3/2}$ para la conservación de potencia o 3/2 para mantener la proyección ortogonal sobre la base (conservación de la magnitud de voltaje). Reemplazando (6) en (5) se obtiene el vector rotatorio *U* (7).

$$U = U_M e^{j\omega t} \qquad (7)$$

De forma equivalente, cada uno de los vectores de voltaje se puede obtener de la transformación de Clarke,[29,31], la cual permite pasar de un sistema trifásico a uno bifásico ortogonal, como se indica en (8).

$$\begin{bmatrix} U_\alpha \\ U_\beta \end{bmatrix} = \begin{bmatrix} \cos(\omega t) & -\sin(\omega t) \\ \sin(\omega t) & \cos(\omega t) \end{bmatrix} \begin{bmatrix} U_x \\ U_y \end{bmatrix}$$

donde (8)

$$\begin{bmatrix} U_x \\ U_y \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \end{bmatrix} \begin{bmatrix} U_a \\ U_b \\ U_c \end{bmatrix}$$

Dependiendo del estado de conducción de los transistores se generan los vectores U_0, U_1,...,U_6, los cuales se encuentran espaciados 60° (π/3) entre sí (Figura 6). El vector U_1 se obtiene al hacer conducir los transistores T_1, T_4 y T_6, con lo cual la fase 'a' se conecta a potencial positivo (+U_{dc}) y las fases 'b' y 'c' a negativo (GND). Esto se representa mediante (+--), los restantes vectores de tensión se muestran en la Tabla 1.[31] Las combinaciones (+++) y (---) no producen tensión resultante sobre la carga, por lo que se denominan vectores nulos. El hexágono que forman estos vectores en el plano complejo (αβ), representa la región máxima alcanzable usando un bus de DC a un voltaje U_{dc} determinado (Figura 6a).[11] La circunferencia dentro del hexágono corresponde a una operación sinusoidal lineal, y como resultado los voltajes en la carga son voltajes sinusoidales, el sentido de rotación del vector de voltaje determina la secuencia de fase en la salida del inversor.[10]

Tabla 1
Valores de los vectores de voltaje que forman el hexágono de tensiones

Conmutación	Vector de Voltaje	Conmutación	Vector de Voltaje
(- - -)	$U_0 = 0$	(- + +)	$U_4 = -\frac{2}{3}U_{dc}$
(+ - -)	$U_1 = \frac{2}{3}U_{dc}$	(- - +)	$U_5 = -\frac{1}{3}U_{dc} - j\frac{\sqrt{3}}{3}U_{dc}$
(+ + -)	$U_2 = \frac{1}{3}U_{dc} + j\frac{\sqrt{3}}{3}U_{dc}$	(+ - +)	$U_6 = \frac{1}{3}U_{dc} - j\frac{\sqrt{3}}{3}U_{dc}$
(- + -)	$U_3 = -\frac{1}{3}U_{dc} + j\frac{\sqrt{3}}{3}U_{dc}$	(+ + +)	$U_7 = 0$

3.2 Forma de trabajo de la modulación vectorial: Para un voltaje determinado a la salida del inversor se tendrá un vector de magnitud ***U*** y ángulo ϕ en el mapa de estados, el cual se desplazará por cada uno de los sectores en que se divide el mapa. La generación del vector referencia **U** se obtiene mediante la aplicación sucesiva de dos vectores adyacentes y un vector nulo durante un período **Ts** (Figura 7), en este intervalo se considera que el vector de referencia se encuentra constante y estacionario. La expresión que relaciona los voltajes de fase-neutro con $\mathbf{T_s}$ se presenta en (9). En general ϕ será el ángulo entre ***U*** y $\boldsymbol{U_n}$, donde ***n*** será el sector activo. El máximo voltaje de fase-neutro sinusoidal se obtiene con la modulación vectorial cuando ϕ es igual a 30° y está dado por (10), el cual corresponde a una tensión 15.5% mayor al máximo obtenible con técnicas PWM convencionales.[10] Los tiempos de conmutación de cada uno de los vectores de tensión se encuentran según.[10,11]

En (9) ΔT_1 es el tiempo durante el cual se aplica a la carga el vector de voltaje U_1, ΔT_2 el tiempo durante el cual se aplica a la carga el vector de voltaje U_2 y ΔT_0 el tiempo durante el cual se aplica a la carga el vector de voltaje U_0, este último corresponde al vector nulo con la combinación (+ + +) o (- - -).

$$U = \frac{(U_1\Delta T_1 + U_2\Delta T_2 + U_0\Delta T_0)}{T_s} \quad \text{donde} \quad T_s = \Delta T_1 + \Delta T_2 + \Delta T_0 \qquad (9)$$

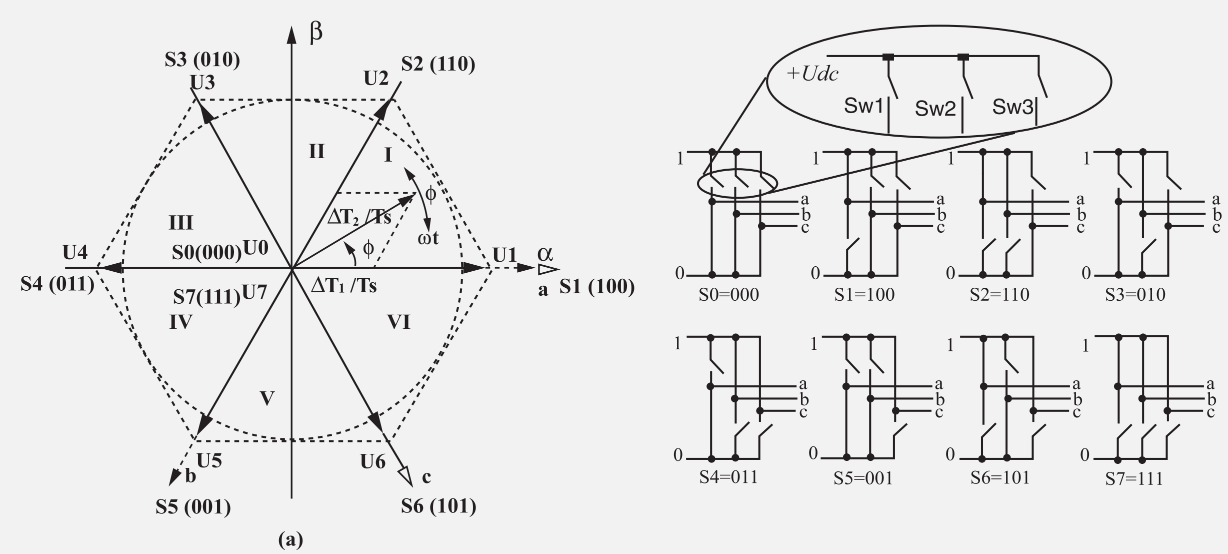


Figura 6. a) Combinaciones de conmutación del puente inversor. b) Hexágono de tensiones o campo de estados.

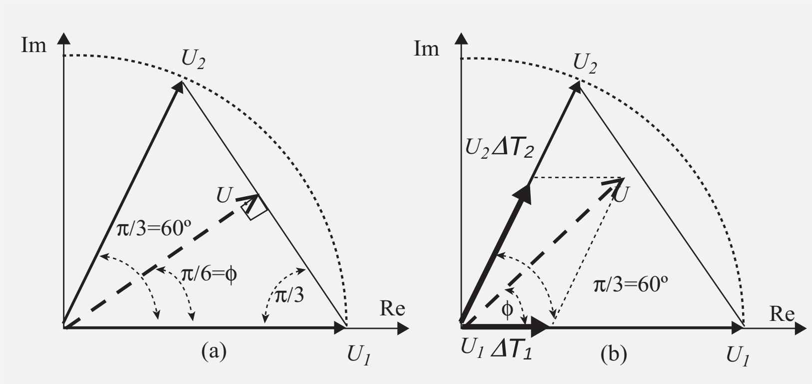


Figura 7. (a) Representación de vector referencia, (flecha punteada), en el sector 1). (b) Descomposición del vector de referencia sobre los vectores U_1 y U_2

$$U_{max} = U_{dc} / \sqrt{3} = 0.57735 U_{dc} \quad (10)$$

$$\Delta T_1 = T_S \times m_a \times sin(60 - \phi)$$
$$\Delta T_2 = T_S \times m_a \times sin(\phi) \quad (11)$$
$$\Delta T_0 = T_S - \Delta T_1 - \Delta T_2$$

3.3 Secuencias de conmutación: En la SVM es posible pasar de un vector a otro conmutando una rama del inversor; de este modo se minimizan el número de conmutaciones y las pérdidas en los semiconductores. La técnica SVM tiene variaciones basadas en el cambio del vector nulo y el orden de conmutación de los vectores activos, ofreciendo diferentes desempeños en conmutación, pérdidas por armónicos, rizo de corriente, etc. La secuencia seleccionada debe asegurar que los voltajes fase-neutro tengan simetría de cuarto de onda para reducir las armónicas impares es sus espectros.[10] La secuencia de conmutación más popular es la *«Alternating zero vector sequence»*, prioriza la reducción en las pérdidas por conmutación seleccionando el vector nulo apropiado, las cuales se minimizan si el vector nulo es el último en cada intervalo y entre dos intervalos consecutivos se alternan U_7 a U_0. En la Figura 8 se muestra una secuencia de conmutación y en[32] una implementación simple de esta técnica.

Con el ánimo de aprovechar los módulos PWM integrados en sistemas procesadores y DSP's, se ha desarrollado la técnica *«Symmetrical placement of zero vectors (SVPWM)»*, que consiste en dividir el ΔT_0 por dos y así durante dos períodos de conmutación iniciar y terminar con un mismo valor de vector cero; sea este S7 o S0. Esta secuencia de conmutación se implementa en,[11] logrando frecuencias de conmutación hasta 2KHz en procesadores 8086. En (12) se muestra el cálculo de los respectivos tiempos y en la Figura 9a los patrones de pulsos de disparo para las «gates» de los interruptores del puente inversor.

$$\Delta T_{0[S0]} = \Delta T_{0[S7]} = (T_S - \Delta T_1 - \Delta T_2)/2 \quad (12)$$

La secuencia de conmutación *«Symmetric Sequence»* se presenta en[27] y [28], ésta mantiene una secuencia de conmutación fija en cada sector. Su funcionamiento es muy similar a la anterior pero posee mayores pérdidas por conmutación. Cada período de conmutación inicia y finaliza con un vector cero y los vectores activos se intercambian. La Figura 9b muestra los patrones de disparo para las «gates» del puente inversor, posee la misma cantidad de conmutaciones que la técnica *«Alternating zero vector sequence»*, pero posee menos distorsión armónica dada la simetría de los pulsos de disparo de la señal de salida. En[32] se presenta otra secuencia de conmutación para la modulación vectorial denominada *«The bus campled sequence»*. En esta secuencia de modulación una rama del inversor se mantiene a un potencial positivo o negativo por unos períodos de conmutación, mientas las otras dos se conmutan. Como resultado de esto la frecuencia de conmutación en cada dispositivo se reduce a dos tercios de la que presenta la técnica anterior y las pérdidas por conmutación en los dispositivos semiconductores del puente inversor se reducen enormemente. Se propuso inicialmente en[33] y se ha desarrollado en muchos trabajos.[18,34] Los patrones de las señales de disparo se presentan en la Figura 9c.

3.4 Operación en sobre modulación y six-step*:* Cuando el índice de modulación «m_a» supera la unidad, el inversor trabaja en la zona no lineal y la forma de onda de salida es no sinusoidal. En las técnicas de modulación escalares, la maximización del voltaje DC se obtiene con la inserción de señales ZSS. Otro enfoque sugiere el uso de un índice de modulación compensado presentado por Kaura y Blasco;[36] este método se aplica a técnicas de modulación sinusoidales y cuando m_a = 1 aplica un índice de modulación compensado M_{icp} como se muestra en (13) al inversor, logrando que el voltaje de salida U_o varíe de forma lineal en todo el rango de sobre modulación.

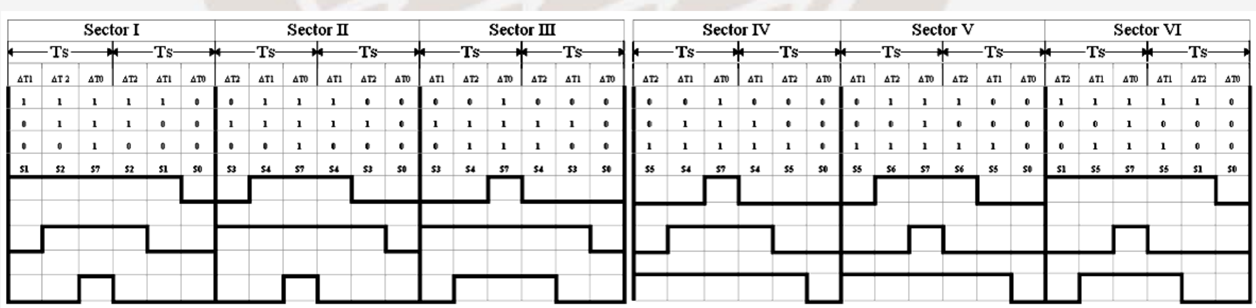


Figura 8. Secuencia de conmutación en el modo Alt-Rev para minimizar conmutaciones en el puente inversor. y con vector nulo V_7 sector 1. El estado de los interruptores se lee de arriba hacia abajo, S1 = 100 significa $S\omega_1$ en ON, SW_2 y SW_3 en OFF.

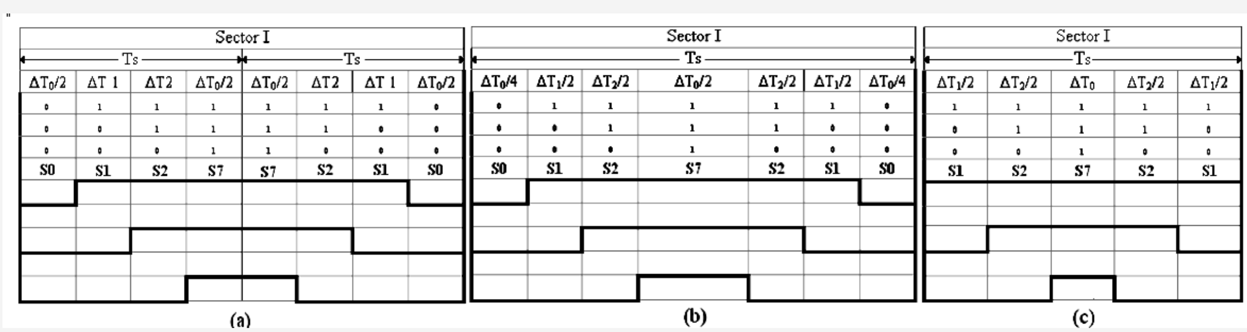


Figura 9. a) *«Symmetrical placement of zero vectors (SVPWM)»*. b) *«Symmetric Sequence»*, c) *«The bus campled sequence»*.

$$U_o = \frac{2}{\pi}\left[sin^{-1}\left(\frac{1}{M_{icp}}\right) + \frac{1}{M_{icp}}\sqrt{\left(1 - \frac{1}{M_{icp}^2}\right)} \right] M_{icp} \tag{13}$$

En la técnica SVM, la sobremodulación se da cuando el vector de referencia sigue una trayectoria circular que amplía los límites del hexágono. Holtz presenta[37] un método para calcular los tiempos de conmutación en sobremodulación y expone dos modos de operación. En estado estable, la trayectoria del vector de voltaje de referencia es circular, a medida que el índice de modulación es mayor a la unidad la circunferencia se extiende más allá del hexágono y el tiempo ΔT_0 es negativo, la máxima tensión de salida será $0.907U_{dc}$ cuando ΔT_0 es cero, el cual se aumenta hasta $0.952U_{dc}$ cambiando la magnitud de la referencia a medida que dicha referencia se acerca a cualquiera de los ocho vectores activos (Ver Figura 10).

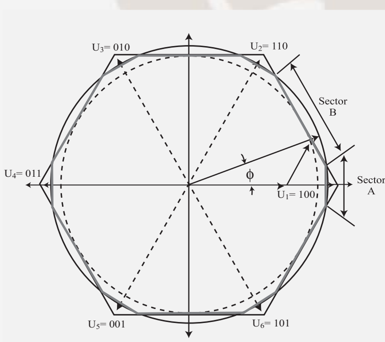


Figura 10. Límites del hexágono y nueva circunferencia trabajando en sobre modulación.

En la sección limitada por el hexágono, sector B, solamente los dos vectores activos del sector de trabajo son conmutados alternadamente y el cálculo de ΔT_1 y ΔT_2 se da por (14). Las partes del círculo dentro del hexágono usan las mimas ecuaciones SVM para determinar los tiempos de estado ΔT_0, ΔT_1, y ΔT_2. Cuando la tensión de salida alcanza valor de $0.952U_{dc}$, la tensión de salida se puede variar gradualmente desde $0.952U_{dc}$ hasta el máximo correspondiente a la operación Six Step; este proceso se conoce como *Over-modulation Mode II* y se basa en variar el recorrido del vector de referencia de voltaje en ángulos constantes iguales a $k\pi/6$, donde k toma valores de 0, 1, ..., 5. El vector de referencia se mantiene en una posición determinada por un sexto del período de la fundamental.

$$T_1 = \frac{T_S}{3} \times \frac{\sqrt{3}\,cos\phi - sin\phi}{\sqrt{3}\,cos\phi + sin\phi} \approx \frac{T_s\phi}{\pi} \tag{14}$$

$$T_2 = \frac{T_S}{3} - T_1$$

Aunque la sobremodulación permite más utilización del voltaje de entrada DC, da como resultado voltajes de salida no sinusoidales con alto grado de distorsión, en especial a baja frecuencia de salida, y no linealidad entre la tensión de salida y el índice de modulación. Otros autores como D-C. Lee y G-M. Lee[38] agregan mejoras en la linealidad de la respuesta del voltaje de salida en función del índice de modulación, así como también reducción en el contenido armónico de la misma. Otros estudios se presentan también.[39,42]

4. Armónicos, tiempo muerto y pérdidas por conmutación

Según la técnica de modulación, los armónicos triples desaparecen del espectro de frecuencia quedando presentes sólo las bandas laterales de dichos armónicos ubicadas en *(fc±2fr)* y *(fc±4fr)* para el primer armónico, *(2fc±fr)*,*(2fc±5fc)* y *(2fc±7fc)* para el segundo armónico, *(3fc±2fr)*,*(3fc±4fc)* y *(3fc±8fc)* para el tercer armónico, etc. El aporte de las componentes residuales debidas a los armónicos mayores al cuarto orden es menor al 5%, pequeño comparado con los tres primeros armónicos. En la Figura 11, se muestra un resumen de tres técnicas de modulación implementadas en simulación, con su respectivo espectro armónico, en donde se muestran las magnitudes normalizadas de cada uno.

En los inversores conmutados las pérdidas de potencia pueden ser por conducción o por conmutación. Las primeras son las mismas en todas las técnicas de modulación y son mucho menores que las pérdidas por conmutación. Se realiza una medición de las pérdidas para diferentes técnicas de modulación, y se presenta un método para su cálculo asumiendo una dependencia lineal entre la pérdida de energía dada la conmutación y la corriente que se conmuta. Las pérdidas por conmutación son directamente proporcionales a la frecuencia de conmutación de la modulación PWM. Valtine[10] presenta un estudio sobre el fenómeno de las pérdidas debidas a la conmutación y asocia las mismas a la *radio-frequency interference* (RFI) producida en cargas RL. Transiciones rápidas entre estados de conducción y no conducción minimizan las pérdidas por conducción en los dispositivos semiconductores, pero a su vez producen RFI cuando dichos cambios son menores a 10us. Transiciones lentas de conducción a no conducción y viceversa en las señales PWM minimizan la cantidad de RFI producido. Sin embargo, estas transiciones lentas incrementan las pérdidas por conmutación en los dispositivos del puente inversor. Por lo tanto, para minimizar las pérdidas por conmutación son deseables transiciones rápidas entre estados de conducción a no conducción. En la Figura 12 se muestra la relación entre los cambios de estados de conducción y no conducción y las perdidas por conmutación de un dispositivo semiconductor de potencia típico. Aunque las pérdidas por conmutación son un problema de estudio continuo, éste se mitiga estableciendo la secuencia de conmutación que menos conmutaciones realice en el inversor para generar un vector de referencia *U* determinado; un ejemplo de esta se presenta en[46].

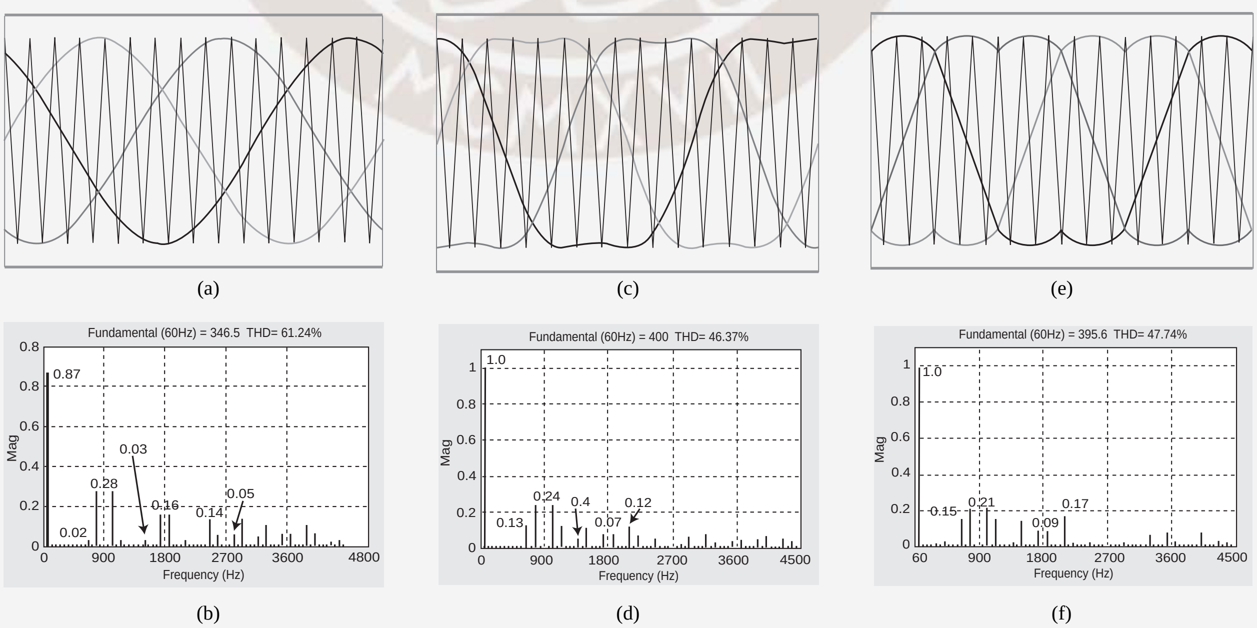


Figura 11. Referencias de técnicas de modulación simuladas en Matlab-Simulink y espectro de frecuencias. (a), (b). CB-SPWM. (c), (d). CB-SPWM+3er Armónico. (e), (f) SVM.

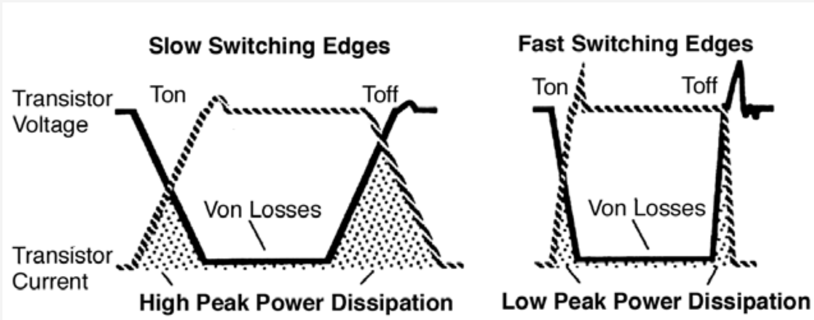


Figura 12. Pérdidas por conmutación en dispositivos semiconductores asociadas a la velocidad de cambio de los flancos de las señales PWM.[10]

Otro problema presente en el inversor es el *tiempo muerto*, que es necesario incluir para evitar la destrucción de dispositivos en una misma rama del puente inversor. Este tiempo muerto produce distorsión en la forma de onda de voltaje y corriente de salida cuando se manejan cargas inductivas y se produce por la desigualdad en la cantidad de corriente que fluye en los dispositivos semiconductores en sus estados ON y OFF.[13,49] Dicha distorsión es corregida cuando se utilizan sensores de corriente en las fases de la carga para implementar lazos de control de corriente, los cuales son típicos en el control vectorial de motores.[47-48] Además de esto existen soluciones hardware basadas en realimentación de corriente; ver ejemplos de éstas donde se sugiere la redistribución del tiempo muerto para compensar la distorsión en el voltaje y la corriente de salida.[10,51]

5. Conclusiones y comentarios finales

En el documento se dio una introducción a las diferentes técnicas de modulación. Se presentan las ventajas de diversas técnicas escalares y vectoriales aplicables a sistemas trifásicos, permitiendo así tener un panorama de qué técnica de modulación escoger al momento de priorizar pérdidas, contenido armónico o eficiencia de conversión. Aunque las diferentes técnicas de modulación sirven para el diseño de «drivers» de AC a frecuencia y amplitud variable, las técnicas SVM han logrado llamar la atención de muchos debido a su gran aprovechamiento del bus de DC, lo que garantiza una eficiencia mayor en el proceso de doble conversión. El hecho de ser una técnica que basa su funcionamiento en la teoría de los vectores espaciales hace que su implementación sea adecuada para procesos de control de máquinas de AC. Sin embargo, esto sólo se logra gracias a la aparición de procesadores de alta velocidad ya que la complejidad de los cálculos requiere mucho tiempo de cómputo. Entre otras de las bondades de la modulación vectorial se encuentra el bajo nivel de ruido generado (armónicos) y la disminución por pérdidas de conmutación, debido a la versatilidad que se tiene al momento de generar los vectores de conmutación. Aunque este último aspecto es obtenible con técnicas de modulación CB-SPWM con ZSS, los controladores de motores de AC manejan notación compleja por lo que la modulación SVM es directamente compatible. Por otro lado, el proceso de optimización del bus de DC siempre es prioridad cuando se diseña o se escoge una técnica de modulación. El principal interés es obtener una relación lineal entre el índice de modulación $\boldsymbol{m_a}$ y la tensión máxima obtenible en la salida del inversor. Aunque muchos métodos se han sugerido, el más popular es el propuesto por Holtz, dada su simple implementación. Aunque se trató de revisar diferentes técnicas de modulación queda abierta la discusión para tocar otras técnicas de modulación no menos importantes como lo son la random PWM, técnicas de modulación adaptativas y técnicas de modulación realimentadas para el control de corriente.

Por último, existen dos vías para minimizar las pérdidas en los semiconductores de potencia: mejorando la construcción de los semiconductores de potencia y encontrando la técnica de modulación que menos conmutaciones necesite para representar un nivel de tensión deseado. En este punto es crítica la frecuencia de conmutación que se utilice, por lo que bajas frecuencias de conmutación con transiciones rápidas entre estados ON y OFF son deseables, pero sujetas a incrementos en el nivel de armónicos. Al final se debe llegar a un balance entre nivel de armónicas, pérdidas por conmutación y frecuencia de conmutación, balance qué depende de la carga a manejar mediante el puente inversor. ✪

Referencias

1. Ned Mohan, Tore M. Undeland, Willian P. Robbins: *Power Electronics. Converters, applications and design.* Second Edition, John Wiley & Sons Inc. 1997.
2. B. K. Bose: *Power Electronics and AC Drives. Prentice-Hall,* 1986.
3. Michael S. Adler, et. all: *The Evolution of Power Devices Technology.* IEEE Transactions on Electronics Devices. vol. ED-31, no. 11, pp 1570-1591, Nov. 1984.
4. B. R. Pelly: *Power Mosfets - A Status Review.* International Rectifier. Pow. Electron. Conf. Vol 1, pp 19-32, March. 1983.
5. Victor A. K. Temple: *Mos-Controlled Thyristors. A New Class of Power Devices.* IEEE Transactions on Electronics Devices. vol. ED-33, no. 10, pp 1609-1619, Oct. 1986.
6. Ahmet M. Hava, et. all: *Carrier-Based PWM-VSI Overmodulation Strategies: Analysis, Comparison, and Design.* IEEE Transactions on Power Electronics. Vol 13, No 4 July 1998.
7. Michael A. Boost, Phoivos P. Ziogas: *State-of-the-Art Carrier PWM Techniques: A Critical Evaluation.* IEEE Trans. Ind. Applicat. vol. 24, no. 2, pp. 271-280., March./April. 1988.
8. Joan Berga, et. all: *Del PWM a la Modulación Vectorial.* Automática e Instrumentación., no. 296, pp. 75-79. Mayo. 1999.
9. J. Holtz: *Pulsewidth Modulation - A Survey.* IEEE Trans. Ind. Elect. vol. 39, no. 5, pp. 410 - 420., December. 1992.
10. Richard Valentine: *AC Induction Motor Control.* Motor Control Electronics Hand Book, McGraw-Hill. 1999.
11. H. W. Van Der Broeck, H Skudelny, G. V. Stanke: *Analysis and realization of a Pulsewidth Modulator Based on Voltage Space Vectors.* IEEE Transactions on Industry Applications .vol. 24, no. 1, pp. 142-150., Jan./Feb. 1988.
12. Keliang Zhou, Danwei Wang: *Relationship Between Space Vector Modulation and Three Phase Carrier Based PWM. A Comprehensive Analysis.* IEEE Transactions on Industrial Electronics. vol. 49, no. 1, Feb. 2002.
13. Hadiouche, D.; Baghli, L.; Rezzoug, A.; *Space vector PWM techniques for dual three-phase AC machine: analysis, performance evaluation and DSP implementation.* 38th IAS Annual Meeting. Vol 1. p.p. 648 - 655. Oct. 2003.
14. H. Pinheiro, F. Botterón, C. Rech, L. Schuch, R. F. Camargo, H. L. Hey, H. A. Gründling, J. R. Pinheiro: *Space Vector Modulation for Voltage-Source Inverters: A Unified Approach.* GEPOC. Federal University of Santa Maria - UFSM - Brazil.
15. Muhammad H. Rashid: *Electrónica de Potencia. Circuitos, dispositivos y aplicaciones.* 3a. Edición, Prentice-Hall, 2004.
16. Frede Blaabjerg. Ulrik Jaeger. Stig Munk-Nielsen. John K. Pedersen: *Power Losses in PWM-VSI Inverter Using NPT or PT IGBT Devices.* IEEE Trans. On Pow. Elect., Vol 10, No 3, p.p 358-367. May 1995.
17. A. Schönung, H. Stemmler: *Static Frequncy Changers With Subharmonic Control in Conjunction With Reversible Variable Speed a.c. Drives.* Brown Boweri Rev. 51, 555-577 (1964).
18. V. Blasco: *Analysis of a hybrid PWM based on modified space-vector and triangle-comparison methods.* IEEE Transactions on Industry Application. Vol 33, No 3 May/June 1997.
19. A. M. Hava, R. J. Kerkman, T. A. Lipo: *A high performance generalized discontinuos PWM algorithm.* IEEE Transactions on Industry Application. Vol 34, No 5 September/October 1998.
20. P. N. Enjeti, P. D. Ziogas, J. F. Lindsay: *Programmed PWM Techniques to Eliminate Harmonics: A Critical Evaluation.* IEEE Transactions on Industry Application. Vol 26, No 2 March/April 1990.
21. J. N. Chiasson, L. M. Tolbert, K. J. McKenzie, Z. Du: *A Complete Solution to the Harmonic Elimination Problem.* IEEE Transactions on Power Electronics. Vol 19, No 2 March 2004.
22. G. Pfaff, A. Weschta, A. Wick: *Desing and Experimental Result of a Brushless AC Servo Drive.* IEEE Transactions on Industry Application. Vol IA-22, pp 814-821, 1984.
23. S. Ogasawara, H. Akagi, A. Nabae: *A Novel PWM Scheme of Voltage Source Inverters Based on Space Vector Theory.* European Power Electronics Conference, Aachen, Germany, pp 1197-1202, 1989.
24. L. Zhang, C. Wathanasarn, F. Hardan: *An Efficient Microprocessor-Based Pulse-Width Modulator Using Space Vector Modulation Strategy.* IEEE 1994.
25. Ying-YU Tzou, Hau-Jean Hsu: *FPGA Realization of Space-Vector PWM Control IC for Three-Phase Inverters.* IEEE Transactions on Power Electroncis. Vol 12, No 6, pp 953-963, November 1997.
26. Berto, S.; Bolognani, S.; Ceschia, M.; Paccagnella, A.; Zigliotto, M.; *FPGA-based random PWM with real-time dead time compensation.* IEEE 34th PESC ’03. 2003. Vol 2, p.p 513 - 518. June 2003.
27. S. Senini, P. Wolfs: A Novel Implementation Using Integer Microcontrollers. Central Queensland University. 2000.
28. M. Copeland: Space Vector Modulation and Overmodulation with an 8-bit Microcontroller. Infineon Technologies, AP0836.
29. S.A.Nasar, I Boldea: Máquinas Eléctricas (Dinámica y Control). Editorial CECSA. 1a. Edición en Español México 1995.
30. Iwaji, Y, Fukuda, S: *A pulse frequency modulated PWM inverter for induction motor drives.* IEEE Transactions on Power Electronics. Volume 7. Issue 2. p.p 404-410. April 1992.
31. Kwon. B. H., Min. B.: *A fully software-controlled PWM rectifier with current link.* IEEE Tran. On Ind. Electron. Vol 40. No 3. p.p. 355-360. June 1993.
32. J. Arenas.: *Técnicas de Modulación Avanzadas. Aplicaciones de Convertidores Estáticos.* Cap 6. Universidad de la Plata.
33. H. Van der Broeck: *Analysis of the harmonic in voltae fed invertir drives caused by PWM schemes with discontinuous switching operation.* Proc, EPE Conf. p.p. 261-266, 1991.
34. W. Sangchai, T. Wiangtong, A. Hongyapanun, P. Wardkean.: *Desing and Implementation of FPGA-Based control IC for 3-phase PWM inverter with*

optimized SVM schemes. IEEE APCCAS 2000. p.p 144-147. Dic/2000.

35. J. A. Houldsworth, D. A. Grant.: *The use of harmonic distorion to increase the output voltage of a three-phase PWM inverters.* IEEE Trans. On Ind. Appl. Vol 20, No 5. p.p. 1224-1228. 1984.

36. V. Kaura, V. Blasko.: *A new method to extend linearity of a sinusoidal PWM in the overmodulation region.* IEEE Trans. On Ind. Appl. Vol 32, No 5. p.p. 1115-1121. 1996.

37. J. Holtz: Pulse Width Modulation for Electronic Power Conversion. Chapter 4. on, Power Electronics and Variable Frecuency Drives. Edit by, Bilman K. Bose., IEEE Press. 1997.

38. Dong-Choon Lee. G-Myoung Lee: *A novel over-modulation technique for space vector PWM inverters.* IEEE Transactions on Power Electronics. Vol 13. Issue 6, p.p 1144 - 1151. Nov. 1998.

39. F. Blaabjerg, S. Freysson, H. Hansen, S. Hansen: *A new optimized Space-Vector Modulation Strategy for a Component-Minimized Voltae Source Inverter.* IEEE Transactions on Power Electronics. Vol 12. No 4, p.p 704 - 714. July / 1998.

40. J. Holtz, P. Lammert, W. Lotzkat: *High-speed drive system with ultrasonic MOSFET-PWM-inverter and single-chip microprocessor control.* IEEE Trans. On Ind. Appl. Vol IA-23 , p.p. 1010-1015. 1987.

41. Bolognani S., Zigliotto M.: *Space vector Fourier analysis of SVM inverters in the over modulation range.* Proc. of the 1996 Int. Conf. on Pow. Elec., Drives and Systems for Industrial Growth. Vol 1. p.p 319-324. Jan/1996.

42. Narayanan G. Ranganathan V.T.: *Extension of operation of space vector PWM strategies with low switching frequencies using different overmodulation algorithms.* IEEE Trans. On Pow. Elect. Vol 17, No 5, p.p. 788-798. Sep / 2002.

43. A. Kwasinski, P. T. Krein, P. L. Chapman.: *Time Domain Comparison of Pulse-Width Modulation Schemes.* IEEE Pow. Elect. Letters. Vol 1, No 3, p.p. 64-68. Sep / 2003.

44. Marian P. Kazmierkowski, R. Krishnan, Frede Blaabjerg.: *Control in Power Electronics.* Academic Press. 2002.

45. J. W. Kolar, H. Ertl, F. C. Zach: *Influence of the modulation method on the conduction and switching losses of a PWM converter system.* IEEE Transactions on Industrial Applications, 27, 1063-1075, 1991.

46. A. M. Trzynadlowski, S. Legowski: *Minimum Loss Vector PWM Strategy for Three Phase Inverters,* IEEE Transactions on Power Electronics, Vol 9, No 1, pp 26-34, January 1994.

47. D. M. Brod, D. W. Novotny: *Current Control of VSI-PWM Inverters.* IEEE Transactions on Industrial Applications, IA-21, No 3. p.p 562-570, May/June 1985.

48. A. Nabae, S. Ogasawara, H. Akagi, *A Novel Control Echeme for PWM Controlled Inverters.* IEEE Ind. App. Annual Meeting, 1985 p.p 473-478.

49. R. D. Klug.: *Effects and Correction of Switching Dead-Times in 3-Phase PWM inverter Drives.* EPE Europ. Conf. Power Electronics and Appl. Aachen. p.p 1261-1266. 1989.

50. Seung-Gi Jeong, Min-Ho Park: *The analysis and Compensation of Dead-Time Effects in PWM inverter.* IEEE Transactions on Power Electronics. Vol 38, No 2, p.p. 108 - 114. April 1991.

ANEXO 14: Programa en C para arranque de motor síncrono

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

unsigned volatile char Onda_5hz_detectada;
unsigned volatile int input_capture;
unsigned char valor_adc;

//Interrupción que relaciona al INPUT CAPTURE
ISR (TIMER1_CAPT_vect)
{
//se lee el registro del input capture
input_capture = ICR1L;
input_capture = ICR1H*256 + input_capture;

//se considera una toleancia de 2%
//el valor debe estar entre 5Hz +- 0.2%
if ( (input_capture > 6125 ) && (input_capture < 6375) )
{
	Onda_5hz_detectada = 1;
	//se enciende el timer 2, pwm 50%
	TCCR2=0b01101010;
	//OCR2=0x80;
}


// se reinicia el contador del timer 1
TCNT1H=0x00;
TCNT1L=0x00;
}

//Función que devuelve el valor de conversión del ADC
//acd_input representa el canal del ADC que se desea leer
unsigned char lee_adc(unsigned char adc_input)
{
ADMUX=adc_input | (0x60 & 0xff);
_delay_us(10);
ADCSRA|=0x40;
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}

//Función que determina la proporción de ciclo de trabajo
//para el timer2 (en función al valor de ADC)

unsigned char determina_proporcion_pwm(unsigned char adc)
{
```

```
return (float)(0.5*adc+127);
}

void setup(void)
{

//configuraciones
//puertos

DDRB = 0x08;
PORTB = 0x00; //OC2 como salida

DDRC = 0x00;
PORTC = 0x00;

DDRD = 0xFF;
PORTD = 0x00;

//timer 1
TCCR1A=0x00;
TCCR1B=0xC4;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

//timer 2
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x80;

TIMSK=0x20;

ACSR=0x80;
SFIOR=0x00;

//adc
ADMUX=0x20;
ADCSRA=0x85;

OSCCAL = 0xB4;

Onda_5hz_detectada = 0;
asm("sei");
```

```
}

int main(void) {

	setup();
	while(1)
	{
		_delay_ms(100);
		//se lee ADC 0
		valor_adc = lee_adc(0);
		//si se detectó previamente la onda de 5hz
		//se modifica el ciclo de trabajo de forma proporcional
		// 5V --> 100%
		// 0V --> 50%
		if (Onda_5hz_detectada)   OCR2 = 
determina_proporcion_pwm(valor_adc);
	}

}
```


SYNCHRONIZER
E-M
The ABC's of Synchronous Motors
200-SYN-42

SYNCHRONOUS

The ABC's of Synchronous Motors

The Greeks had a word for it — SYNCHRONOUS— "syn" a prefix meaning "with", and "chronos" a word denoting time. A synchronous motor literally operates "in time with" or "in sync with" the power supply.

The modern synchronous motor is a double-duty motor. It is a highly efficient means of converting alternating current electrical energy into mechanical power. Also it can operate at either leading, unity, or in rare cases at lagging power factor, providing the function of a synchronous condenser for power factor correction.

In the synchronous motor, the basic magnetic field is obtained by direct current excitation rather than through the air-gap from the armature, as is the case with induction motors. Comparatively large air-gaps are used, making practicable the manufacture, even in relatively low horsepower ratings of low speed synchronous motors. In all low speed ratings and in large high speed ratings, synchronous motors are physically smaller and less costly to build than squirrel-cage induction motors of equivalent horsepower.

A synchronous motor can be applied to any load which can be successfully driven by a NEMA Design B squirrel-cage motor. However, there are certain types of loads for which the synchronous motor is especially well suited. The correct application of synchronous motors results in substantial savings in several ways:

Synchronous motor 1500 HP at 720 RPM with Paper Mill Dripproof™ enclosure driving a refiner in a paper mill.

1. High efficiency

Synchronous motors have a unique and merited position as the most efficient electrical drive in industry. They perform with great economy while converting electrical power to mechanical power and can be built with unique physical features. Synchronous motors can be designed to effectively operate over a wide range of speeds to provide the best drive for a wide variety of loads.

2. Power factor correction

Because they can operate at leading power factors, synchronous motors can help reduce your power cost and improve efficiency of the power system by correcting low power factor. In a few years, savings in power bills may equal the original investment in the motor.

3. Reduced maintenance

Synchronous motors with brushless exciters practically eliminate the need for motor maintenance other than routine inspection and cleaning.

4. Space savings

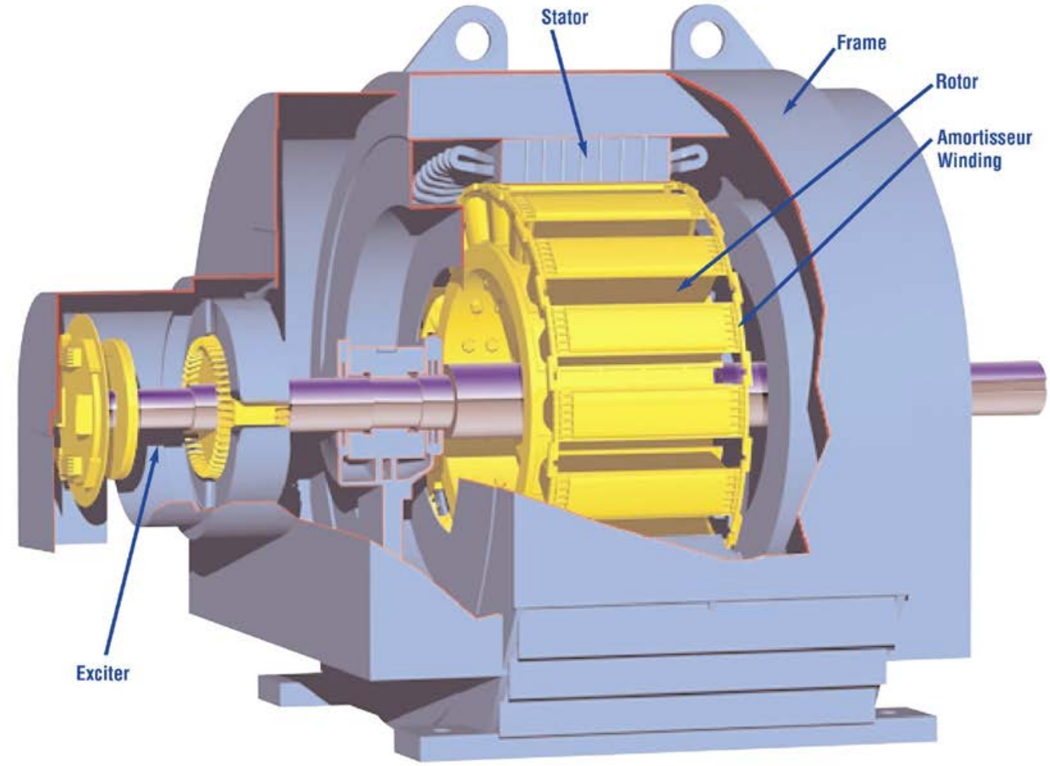
Engine-type synchronous motors can be connected directly to the shaft and bearings of the driven equipment. This approach saves on floor space, extra building costs, and makes for a simple installation.

5. Constant speed means better quality products

Synchronous motor speed is unaffected by line or load conditions. In certain applications, e.g. paper mill pulp machines, constant speed operation results in superior uniformity and quality of the product being produced.

6. Adjustable speed means lower power costs

In many cases it is much more economical to operate driven equipment at reduced speeds. This can be accomplished using synchronous motors in conjunction with magnetic drives, or through electronic methods, which can be used to supply an adjustable frequency to the machine and control its operating speed. Typical applications for adjustable speeds include pumps and fans. Either approach can reduce power costs and provide a more cost-effective process.



EM synchronous motors are designed and built to industry standards

ANSI, IEC, IEEE and NEMA standards help both manufacturer and user by providing definitions and suggested values. They are continually updated to incorporate the latest trends in machine use. Although these standards are gradually coming closer together, this is a long process. ANSI and/or NEMA provide the standards for synchronous motors in the United States, whereas IEC standards provide the guidelines for European applications. Standards are a great help in understanding the application of motors. Please refer to NEMA MGI part 21, Synchronous Motors, or ANSI C50.

The parts of a synchronous motor

FRAME ... supports and protects the motor; it is built in horizontal and vertical types and in various protective constructions for indoor or outdoor service.

STATOR ... consists of the stationary magnetic parts, including the core and winding which operate off the alternating current power supply to provide a rotating magnetic field.

ROTOR ... consists of the rotating active parts, and includes the spider, the field-pole winding and the amortisseur winding. Field poles are magnetized by direct current from the exciter or directly through collector rings and brushes; they interlock magnetically through the air-gap and revolve in synchronism with the rotating magnetic poles of the stator.

EXCITER ... supplies magnetizing current to the field winding. Modem exciters are of the rotating brushless type with no collector rings or brushes.

AMORTISSEUR WINDING ... makes a synchronous motor self-starting like a squirrel-cage induction motor. Depending on the type of loads and the torque required, a variety of bar materials and arrangements can be used to provide adequate starting torque.

GEARED

Synchronous motors are "geared" to the power supply

During operation, the stator of a polyphase alternating current motor has a rotating magnetic field. This is developed by the direction and amount of current flow in the stator coils.

Figure 1 represents a two-pole, three-phase stator, having one slot per pole per phase in which there are six stator coils. These coils are connected to a three phase power supply having phase rotation A, B, C.

At the instant shown, the current in phase A in maximum positive enters the motor at A+ and leaves at neutral where the three conductors meet. The "A" arrows in Figure 1 represent the magnetic flux (lines of force) developed by this current. At this instant the current in phases B and C is negative, each equal to one-half the current in A. The magnetic flux developed by B is shown by the "B" arrows, each equal to half the "A" arrows. The flux developed by C is represented by the "C" arrow, each equal to half of A. B and C also have horizontal components but, as shown, they cancel out.

The result is a vertically polarized stator, with a North pole at the top and a South pole at the bottom. The arrows show the flux flowing downward through the rotor.

One-twelfth revolution later (Figure 2), the positive current in A has dropped in value and is equal to the negative value in C, which has increased in value. The current in B is zero. Equal field strengths are developed by A and C and, canceling out the opposing components, we find the total flux in Figure 2a to be represented by the two "A" and two "C" arrows, 30° from the former position.

At the 60° position, shown in Figure 3, the current in phase B and the corresponding stator coils are positive and equal to phase A. The current in phase C is maximum negative. After canceling out the opposing components, we find a resultant field, as shown in Figure 3a composed principally of current in phase C, supplemented by smaller equal amounts in phases A and B.

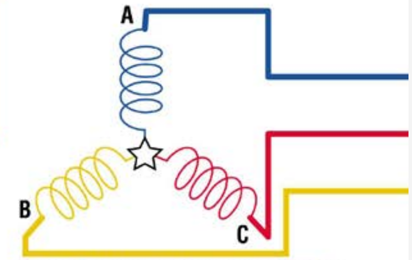


Figure 1 *See top left of page 5 for balance of this diagram, plus its caption.*

Figures 4 and 4a complete a one-quarter turn or 90° of a two-pole motor. Continuation of this completes one revolution.

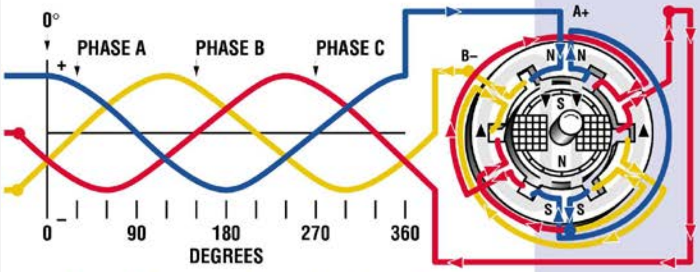
The rotor shown in the above figures is that of a two-pole synchronous motor having definite North and South poles developed by direct current flowing in the field winding. The current flows from the observer in the left hand portion of the winding, towards the observer in the right hand portion. Using the "right-hand rule" this develops a North pole at the bottom of the rotor and a South pole at the top.

The magnetic poles developed in the stator are exactly opposite. However, unlike poles attract, so the North pole of the rotor lines up under the South pole of the stator.

As the North pole of the stator shifts 30° clockwise, the rotor is pulled along with it, and (assuming no load on the rotor shaft and no losses) the rotor and stator poles line up exactly as shown. Each rotor pole lags behind its corresponding stator pole by 20 to 30 electrical degrees at full load. This is called the "load angle," and it is a function of motor load, where increased torque requirements result in greater magnetic pull and a greater displacement from magnetic center.

A normal two pole stator differs in many respects from that shown in Figures 1 through 4. Instead of one slot per phase there are six or seven. The coil pitch is about 120° instead of 60° and will overlap the coils of the various phases. The configuration shown results in rotation by a series of jerks. The use of more closely spaced coils will result in a smoother, more uniform turning action.

Figures 1 through 4 *AC Generator. Three-phase power supply for Synchronous Motor. Generator voltage is assumed equal and in phase with the current; there is no load torque for friction so rotor and stator poles are free to line up magnetically.*



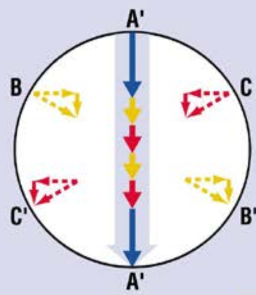
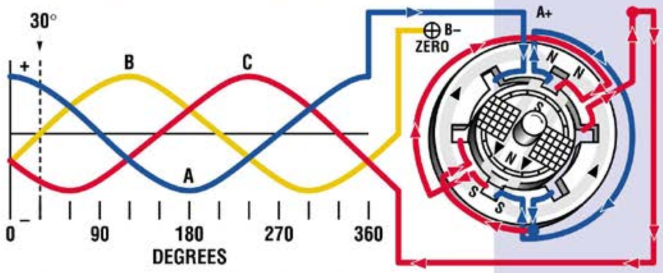


Figure 1 *Current relationship, and magnetic resultant (Figure 1a) of a two-pole synchronous motor with rotating magnetic field at 0°.*
SEE TOP LEFT OF PAGE 4 FOR BALANCE OF THIS DIAGRAM.

Figure 1a



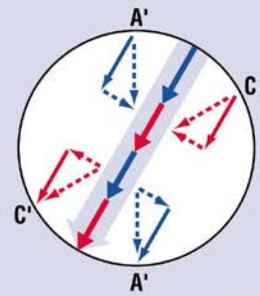
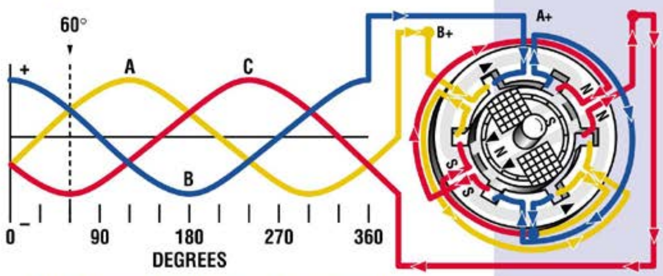


Figure 2 *Current relationship, and magnetic resultant (Figure 2a) of a two-pole synchronous motor with rotating magnetic field at 30°.*

Figure 2a



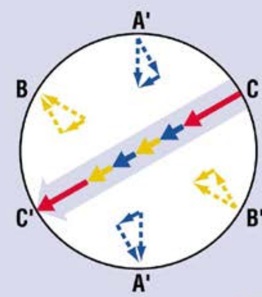
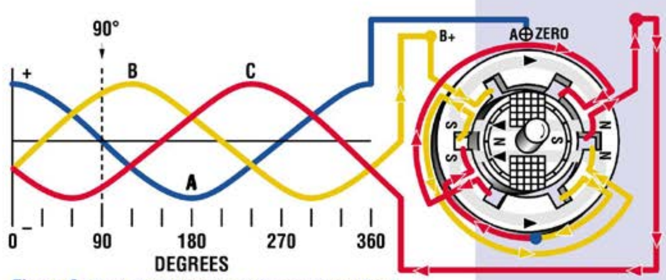


Figure 3 *Current relationship, and magnetic resultant (Figure 3a) of a two-pole synchronous motor with rotating magnetic field at 60°.*

Figure 3a



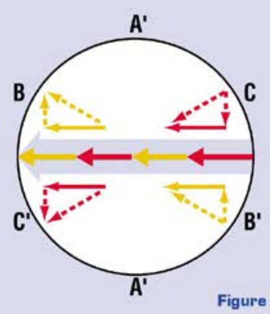


Figure 4 *Current relationship, and magnetic resultant (Figure 4a) of a two-pole synchronous motor with rotating magnetic field at 90°.*

Figure 4a

How synchronous motors control power factor

Power factor is the factor by which apparent power, or kVA, is multiplied to obtain actual power, or kW, in an ac system. It is the ratio of the in-phase component of current to total current. It is also the cosine of the angle by which the current lags (or leads) the voltage.

The conversion of electrical energy to mechanical energy in a motor is accomplished by magnetic fields. In the previous section, it was explained how magnetic poles are formed when current flows in coils placed around the stator gap bore. As explained, these poles rotate around the stator. When voltage is applied to a motor, an armature current flows to provide the necessary magnetic push (mmf or ampere turns) to produce a flux which, in turn, produces a voltage (back emf) that opposes the applied voltage. This mmf is a magnetizing current. It is loss-less, except for the i^2r in the winding and any core loss due to the changing flux in the iron. The magnetic energy is transferred from the line to the motor and back again each half cycle. The net power is zero, and the power factor is zero.

The power factor of a synchronous motor is controllable within its design and load limits. It may operate at unity, leading, or in rare cases, lagging power factor and may be used to modify the power factor of the system to which it is connected. A simplified explanation of phasor relationships will describe what takes place under various load and excitation conditions.

Phasors are vectors that represent the relationship between voltages or currents. The phasor length represents the magnitude and, as it is allowed to rotate counterclockwise about the origin, the projection on the x axis is the instantaneous value.

Figures 5, 6 and 7 show the instantaneous voltage, current and power (V times I) for unity, leading and lagging power factors. The magnitude of the current and voltages are held constant for all cases. Also shown is the corresponding phasor diagram for each in Figure 5a, 6a and 7a. Note that for lagging power factor, the current phasor lags the voltage phasor and the instantaneous current passes through zero after the voltage. The power for unity power factor is always positive while both leading and lagging power factors have some negative power values, and the peak is less than for unity. Thus, the average value of power is less for either leading or lagging power factor for the same current and voltage.

Once the synchronous motor is synchronized, the field poles on the rotor are in line with the rotating magnetic poles of the stator. If dc is applied to the rotor pole windings, the rotor can supply the necessary ampere turns to generate the flux which produces the internal motor voltage. Thus, the field current can replace part or all of the magnetizing current. In fact, if more dc field current is supplied, the increased flux will try to increase the line voltage. To increase the line voltage, the motor will supply ac magnetizing current to all "magnets" on the system to increase their magnetic flux. This is leading power factor.

The synchronous torque developed is roughly proportional to the angle of lag (load angle) of the motor rotor with respect to the terminal voltage, which, at full load, is in the area of 20 to 30 electrical degrees. If a restraining force is applied to the motor shaft, it will momentarily slow down until a torque is developed equal to the applied restraint. The motor will then continue to operate at synchronous speed.

Figure 8 shows a somewhat simplified phasor diagram for a unity power factor motor connected to a large system such that the terminal voltage is fixed. Since synchronous machines have poles and interpole spaces, a system of phasors represent each of the "direct" (in line with the pole) and "quadrature" axes (interpole).

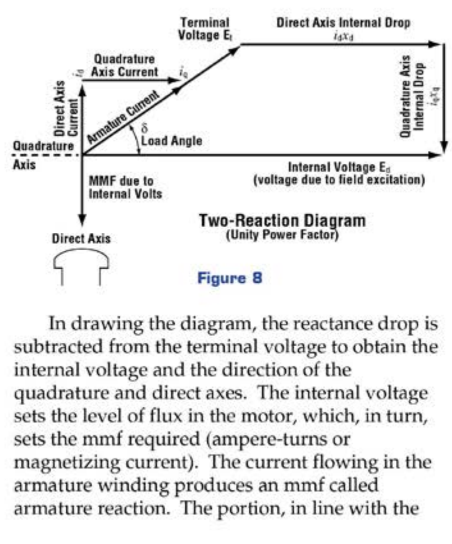


Figure 8

In drawing the diagram, the reactance drop is subtracted from the terminal voltage to obtain the internal voltage and the direction of the quadrature and direct axes. The internal voltage sets the level of flux in the motor, which, in turn, sets the mmf required (ampere-turns or magnetizing current). The current flowing in the armature winding produces an mmf called armature reaction. The portion, in line with the direct axis, must be added (or subtracted, for lagging P.F.) to the mmf, producing the internal voltage. Since the field winding on the rotor poles is in line with the direct axis, dc in this winding can supply the required mmf.

If the dc is increased, the internal voltage will increase, and since the terminal voltage is fixed by the connected system, the internal drop must increase. The result of this is shown in Figure 9

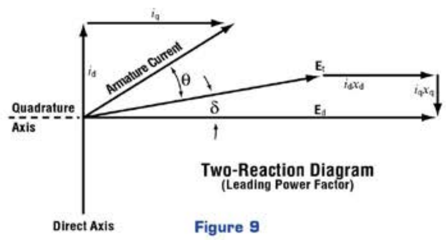
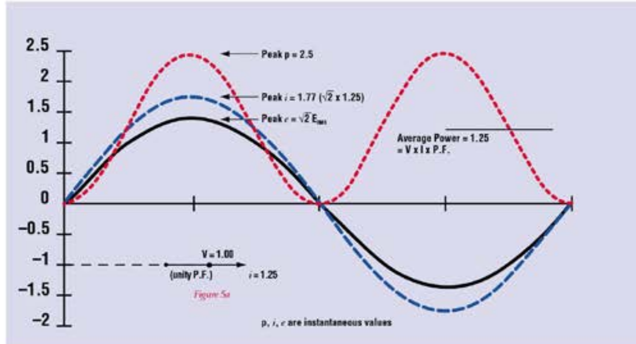


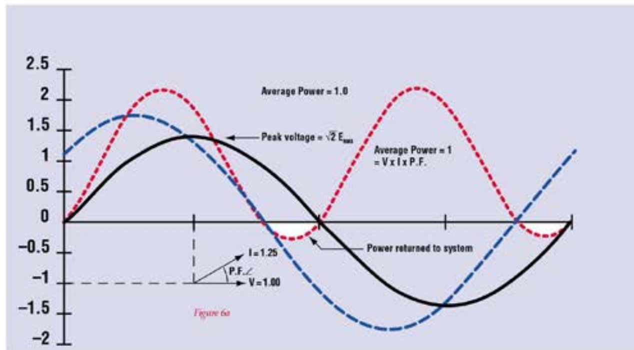
Figure 9

where the line current is increased and repositioned to a leading power factor. Note that the current in phase with the voltage represents the load and is unchanged, but a magnetizing component has been added. This component flows to the line to try to raise the system voltage. The motor has a leading power factor.

If, instead, the dc is decreased, the internal voltage will try to reduce the system voltage, resulting in a magnetizing current drawn from the system. The motor then has a lagging power factor.

The fact that the field poles of a synchronous motor are magnetically in line with the stator flux allows dc current on the field to replace ac magnetizing current on the stator. Since power factor is determined by the magnetizing current, the power factor is adjustable by changing the field current.





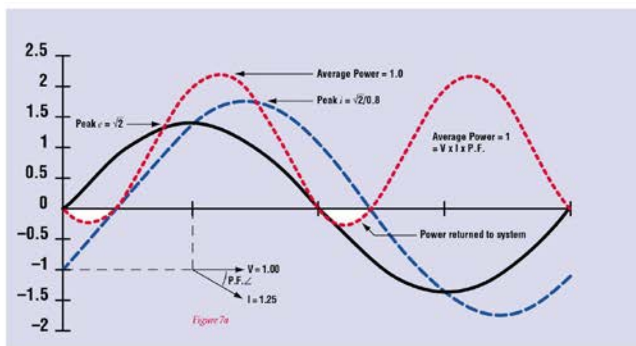


Figure 5 (top) *0.8 PF Motor at unity PF.*
Figure 6 (middle) *0.8 PF Leading.*
Figure 7 (bottom) *0.8 PF Lagging.*

PF∠ stands for Power Factor Angle.

Figure 10
For any given load the power factor at some given excitation value will be the ratio of minimum line amperes at that load, to the line amperes at the excitation value under consideration. Assume a steady load condition in which the line amperes, at various excitation values, lie along line A-O-B; with point O representing minimum ac input and point B representing the input at the desired excitation value. The power factor at that excitation is CO ÷ DB.

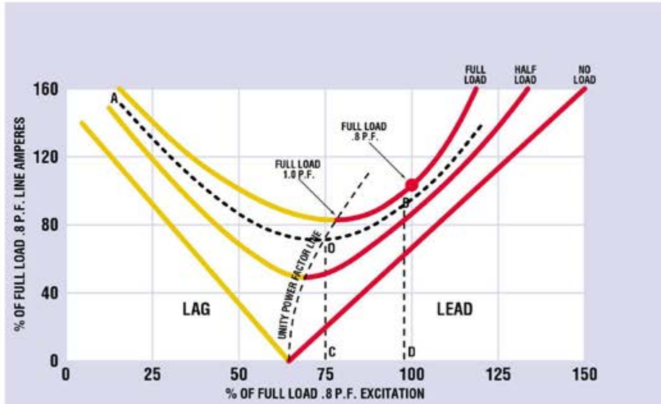


Figure 11
Plant loading and power factor.

Plant loading and power factor (induction motor proposed)

	Monthly Averages			Calculated		Transformer		
Load	MWH	P.F.	Hrs (Est)	kW	kWR	kVA	Load	Rating
T1 Office	77	0.92	194	400	-170	435	435	T1=1000
T2 Pumps	587	0.72	335	1750	-1687	2431	2431	T2=2000
T3 Air	235	0.82	335	700	-489	854	854	T3=1500
T4 Mill	1347	0.88	464	2900	-1565	3295		
T4 Proposed	335	0.81	335	1000	-724	1235	4522	T4=4000
Overall		0.824		6750	-4635	8188		

Plant Loading and Power factor (synchronous motor proposed)

	Monthly Averages			Calculated		Transformer		
Load	MWH	P.F.	Hrs (Est)	kW	kWR	kVA	Load	Rating
T1 Office	77	0.92	194	400	-170	435	435	T1=1000
T2 Pumps	587	0.72	335	1750	-1687	2431	2431	T2=2000
T3 Air	235	0.82	335	700	-489	854	854	T3=1500
T4 Mill	1347	0.88	464	2900	-1565	3295		
T4 Proposed	335	-.8	335	1000	750	1250	4522	T4=4000
Overall		0.906		6750	-3161	7453		

"V" Curves

It is generally assumed that the line voltage will be substantially constant, and it is apparent from the preceding discussion that load, excitation, line amperes, and power factor are closely related. This relationship is readily expressed by a family of characteristic curves known from their shape as "V" curves. These are represented in Figure 10. Note that for each curve the power is constant and the excitation is varied to give different magnetizing currents.

The minimum value of line amperes for each load condition is at 1.0 or unity power factor. As excitation is decreased, the line current will increase and the motor will operate at lagging power factor. If excitation is increased from the 1.0 power factor condition, the line current will again increase, but now the motor will operate under a leading power factor condition.

For any load, the power factor will be the ratio of minimum line amperes at that load to the line amperes at the excitation value under consideration. To operate at 1.0 power factor for maximum efficiency, set the excitation for minimum ac line current.

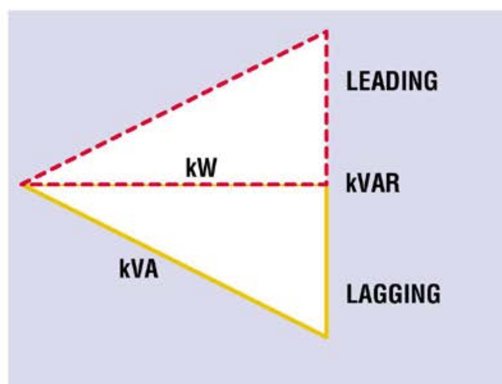


Figure 12
For leading power factor synchronous motors, leading kVARs indicates kVARs fed to the system, where as induction motors always absorb kVARs from the system.

System Power Factor Correction

The power usage and the power factor of a plant should be periodically reviewed to avoid surprises. A typical plant power system is shown in the form of a spreadsheet in Figure 11. Here it is assumed that you have monthly records for the various load centers. A proposed plant expansion employing either an induction or a synchronous motor is under consideration. T1, T2, etc. represent transformers or load centers.

From the MWH and the estimated operating hours, the average kW is calculated. Next, calculate the kVAR and kVA using the power factor. Tabulation of results is shown in Figure 13.

Note that the terms "leading" and "lagging" refer to the current leading or lagging the voltage. The current to induction motors will lag, and the motor will draw magnetizing current from the system. Leading P.F. synchronous motors feed magnetizing current to the system. As shown in Figure 12, lagging currents are treated as negative, leading as positive.

$$kVAR = \sqrt{kVA^2 - kW^2}$$

$$kVAR = kW \times Tan\ (\theta)$$

(where θ is the angle in degrees, by which the current differs from the voltage.)

$$kW = kWH/H$$

$$\text{Motor rated kVA} = \frac{0.746 \times HP}{Eff. \times P.F.}$$

If the proposed expansion is an induction motor, the overall power factor will be less than 0.9. This will result in additional charges by the power company. Note that transformer T4 in Figure 11 will be overloaded.

If the addition is a 0.8 leading P.F. synchronous motor, the total kW will be the same, but the kVAR will be less, and the overall power factor will be over 0.9. The overload on T4 is eliminated. In calculating the load on T4, it is necessary to first combine the kWs and kVARs separately and then calculate the kVA as the square root of the sum of the squares.

Figure 14 shows the reactive capability of typical synchronous motors. These curves plot the percent kVAR vs. percent load. To use the curve, determine the kVAR based on the percent load and rated power factor of the motor. The reactive kVAR is then the rated horsepower multiplied by the percent kVAR determined. For example, a 250 horsepower, 0.8 power factor motor operating at 100% load gives a 0.6 factor. The kVAR is then 250 X 0.6 = 150 kVAR supplied to the system. Excitation is maintained at the full load value.

Power Factor	Ratio kVAR/kW	Power Factor	Ratio kVAR/kW	Power Factor	Ratio kVAR/kW
1.00	.000	.80	.750	.60	1.333
.99	.143	.79	.776	.59	1.369
.98	.203	.78	.802	.58	1.405
.97	.251	.77	.829	.57	1.442
.96	.292	.76	.855	.56	1.480
.95	.329	.75	.882	.55	1.518
.94	.363	.74	.909	.54	1.559
.93	.395	.73	.936	.53	1.600
.92	.426	.72	.964	.52	1.643
.91	.456	.71	.992	.51	1.687
.90	.484	.70	1.020	.50	1.732
.89	.512	.69	1.049	.49	1.779
.88	.540	.68	1.078	.48	1.828
.87	.567	.67	1.108	.47	1.878
.86	.593	.66	1.138	.46	1.930
.85	.620	.65	1.169	.45	1.985
.84	.646	.64	1.201	.44	2.041
.83	.672	.63	1.233	.43	2.100
.82	.698	.62	1.266	.42	2.161
.81	.724	.61	1.299	.41	2.225

Figure 13 *Power Factor Table*

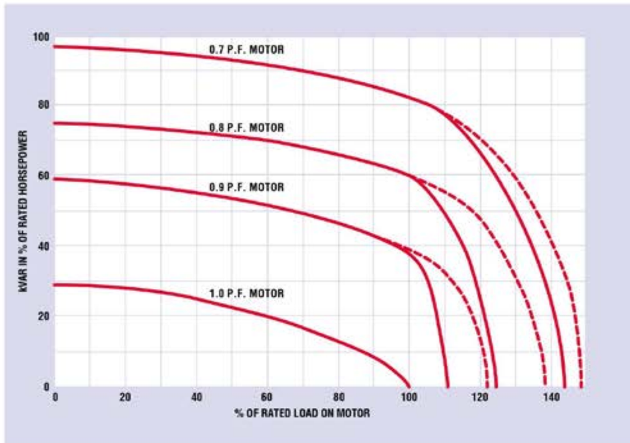


Figure 14 *Leading reactive kVA in percent of motor horsepower for synchronous motors at part load and at various power factor ratings. Solid lines are based on reduction in excitation, at overload, to maintain rated full-load amperes; dotted lines represent values with rated excitation to maintain rated pull-out torque.*

TORQUE

Torque makes the wheels go 'round

Torque may represent either the turning effort developed by a motor or the resistance to turning effort exerted by the driven load. It is defined as tangential pull at a radius of one unit from the center of rotation. In the United States it is measured in foot-pounds or inch-pounds. For a given condition:

$$\text{Torque} = \frac{\text{HP X 5250}}{\text{rpm}}$$

Torque = Tangential effort in pounds at one foot radius
HP = Horsepower developed
rpm = Revolutions per minute

In some cases motor torques are expressed in foot-pounds; however, they are usually expressed as a percentage of full load torque.

A synchronous motor has many important torques that determine the ability of the motor to start, accelerate, pull its connected load into step, and operate it through anticipated peak loads within its design limits.

These torques may be described as:

1. *Starting torque or breakaway torque.*
   This is the torque developed at the instant of starting at zero speed.

2. *Accelerating torque.*
   This is the motor torque developed from stand-still to pull-in speed minus the load torque.

3. *Pull-up torque.*
   This is the minimum torque developed between stand-still and the pull-in.

4. *Pull-in torque.*
   This is the torque developed during the transition from slip speed to synchronous speed and generally is defined at 95% speed.

5. *Synchronous torque.*
   This is the steady state torque developed during operation. It is load dependent.

6. *Pull-out torque.*
   This is defined as the maximum steady state torque developed by the motor, for one minute, before it pulls out of step due to overload.

Starting and accelerating torques

The characteristics of these torques are closely allied and will be discussed together. Five individual torques contribute to the turning effort of a synchronous motor during starting and accelerating. They are produced by:

1. Amortisseur windings
2. Field windings
3. Hysteresis in pole faces
4. Eddy currents in pole faces
5. Variation in magnetic reluctance

Of the five torques, only the first two are significant during starting and accelerating, and only their characteristics will be discussed here.

The amortisseur windings

These windings develop most of the starting and accelerating torque in motors of this type. This winding consists of a partially distributed cage winding, with bars imbedded in the pole faces and short circuited at each end by end rings. Varying the number, location and resistance of these bars will have a substantial effect on the torque and on the kVA input. In some cases a double cage, consisting of one row of shallow bars and one row of deepset bars, is used. These may be separate, or they may have common end rings as circumstances will dictate. Typical amortisseur windings are illustrated in Figure 15.

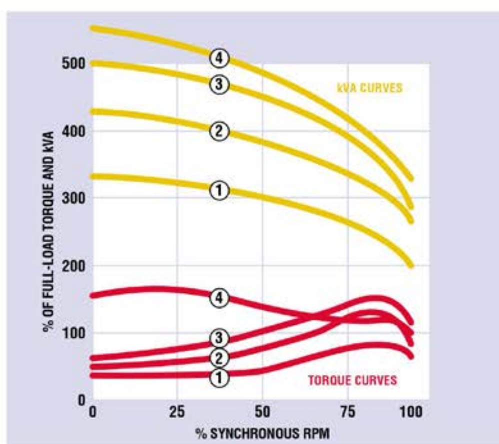


Figure 15 *Modern amortisseur windings use a variety of bar materials and arrangements to secure the desired resistance and reactance to produce the required torques.*

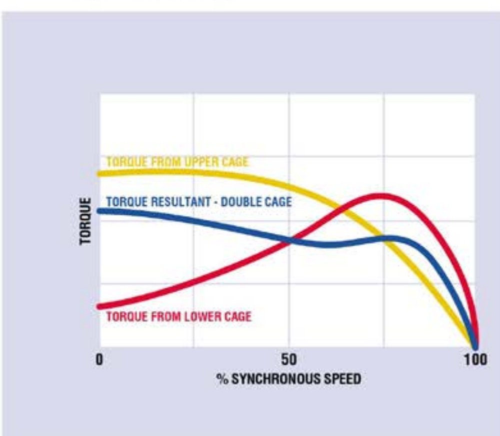
Amortisseur windings and end ring

As discussed previously, the application of polyphase power to the stator winding results in the development of a rotating magnetic field. Magnetic lines of force developed by this rotating magnetic field cut across the cage bars and generate voltages, causing currents at slip frequency to flow in them. The interaction between these currents and the rotating magnetic field develops torque, tending to turn the rotor in the same direction as the rotating magnetic field.

Recommended starting torques range from 40% to 200% of full load torque. See NEMA recommendations on torques for various synchronous motor applications.

Torque comes from the action of the flux produced by the stator. Getting more torque requires more flux and, therefore, more iron. More torque also results in more inrush. For any given motor, the effective resistance and reactance of the cage winding are the principal factors in determining motor torques. Typical starting and accelerating torques and their associated kVA values are shown in Figure 16.

The double cage motor was developed to give essentially constant pull-up torque. The shallow cage has a high resistance and low reactance, while the deep cage has low resistance and high reactance. At standstill, slip frequency equals line frequency, and the high reactance of the deep cage forces most of the induced current to flow through the shallow cage, high resistance bars, resulting in high starting torque.



As the rotor comes up to speed, the slip frequency decreases and, with it, the impedance of the deep cage. This permits more of the induced current to flow through the deep-set, low-resistance bars, resulting in high torques at speeds near pull-in. Figure 17 illustrates the resultant torques using this type of construction.

Ideal applications for double cage windings are those where the load is uniformly high during the entire accelerating period. Disadvantages include saturation, due to the number of cage bars in the pole head, and stress, due to the differential thermal expansion in bars that are close together. Double cage motors are relatively uncommon.

Figure 16 (left) *Curves show starting torque and starting kVA of synchronous motors using various types of amortisseur winding constructions. For any given motor the effective resistance and reactance of the cage winding are the principal factors in determining the shape of the curves, and the ratio of torque to kVA.*

Figure 17 (right) *Curves show double-cage torque resulting from high resistance upper cage winding, and low resistance, high reactance lower cage winding.*

Field Windings

The amortisseur winding imparts squirrel-cage motor starting and accelerating characteristics to the synchronous motor. The field winding develops a torque similar to that of a wound rotor motor with a single-phase secondary circuit.

The single-phase characteristic results in two torque components, one rotating in the same direction as the rotor and one rotating in the opposite direction. The second component is positive in value to 50 percent speed and negative from that point to pull-in. Figure 18 illustrates the torque components and the resultant torque derived from the field windings. If a motor was built with a relatively weak cage, the dip in torque due to the effect of the negative rotating component could result in the motor not being able to accelerate much past half-speed.

During starting and accelerating, the field winding is short-circuited through a field discharge resistor to limit the induced field voltage. The ratio of this resistance to the field resistance has a significant effect on the starting torque, the torque at pull-in and, to a lesser degree, on starting kVA. Figure 19 shows the effect of varying the field discharge resistance on one specific motor. It will affect the negative rotating component of the field winding torque, and a high value of resistance may reduce it considerably. However, a high value of field discharge resistance results in a high induced voltage across the field winding. In extreme cases, as in the case of open-circuit field starting (infinite resistance) the induced field voltage may exceed 100 times the normal excitation voltage.

Pull-up Torque

Pull-up torque is the minimum torque developed from stand-still to the pull-in point. The pull-up torque must exceed the load torque (torque required by the driven machine) by enough margin to maintain a satisfactory rate of acceleration from stand-still to pull-in under the minimum expected voltage condition.

Net Accelerating Torque

Net accelerating torque is the margin by which the motor torque exceeds the load torque from stand-still to the pull-in point.

In the case of high inertia loads, it is important that the starting time be determined so proper relaying action of overcurrent relays and amortisseur winding protective relays, etc., can be obtained.

Accelerating time from standstill to the pull-in point can be approximated by applying the following relationship:

$$\text{Accelerating time } t = \frac{WK^2 \times \Delta rpm}{308T} \quad \text{(seconds)}$$

- WK^2 = total inertia of the load and the motor in lb. ft.^{2}
- Δrpm = the change in speed (rpm_2 - rpm_1)
- 308 = a constant
- T = net accelerating torque, lb. ft. from rpm_1 to rpm_2
- t = the time increment to accelerate from rpm_1 to rpm_2

Note the motor torque as a function of speed, the load torque as a function of speed, and the

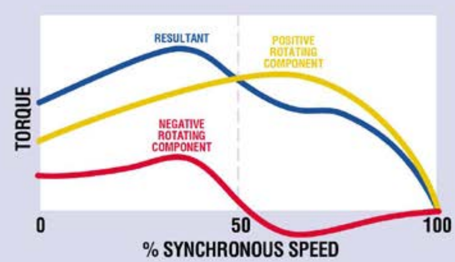


Figure 18 *Torque components and resultant torque derived from the field winding of a synchronous motor during starting.*

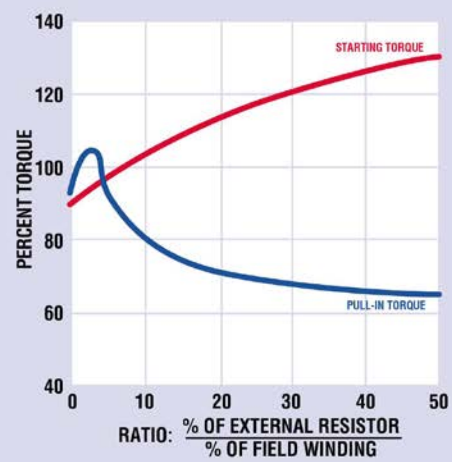


Figure 19 *Effect of ratio of field discharge resistance to field resistance on the starting and pull-in characteristics of the motor.*

motor and load inertias. The latter are obtained from the manufacturers as constants; the former are most frequently illustrated as speed-torque curves.

Refer to Figure 20, which shows curves for a typical 600 hp. 900 rpm motor and a throttled centrifugal fan load. The total motor and load WK^2 (inertia) is given as 29770 lb.ft^2.

Net accelerating torque at any speed is determined by subtracting the load torque from the motor torques at that speed. By taking values at 10% intervals, starting at 5% speed and continuing to 95% speed for example, a reasonable approximation of acceleration time from standstill to pull-in speed can be calculated. Refer to Figure 21 for a tabulation from this example.

During the starting period, the amortisseur winding must absorb energy. In addition to the energy required to overcome the load torque, energy must also be absorbed to accelerate the inertia of the rotating system. Since high temperatures that affect material strength can be realized in a short period of time, it is important that a means be provided for transmitting as much heat as possible into the pole body and also that the heat absorbed will not overheat the amortisseur winding. During starting the temperature rise of the amortisseur winding may be as high as 150°C. Quantitatively, the energy absorbed in accelerating a rotating mass is covered by this formula:

$$H = \frac{0.231 \times WK^2 \times (rpm)^2}{(1000)^2}$$

H = Energy in kW seconds
WK^2 = Total WK^2
rpm = Speed in rpm
0.231 = a constant

In the case of the motor and fan referred to above, the energy absorbed in the rotor, due to inertia, in bringing the unit to 95% of synchronous speed is:

$$H = \frac{0.231 \times 29{,}770 \times (855)^2}{(1{,}000)^2} = 5020 \text{ kW seconds}$$

H is expressed in per unit by dividing the above by rated kW (rated horsepower x 0.746). Interestingly, two times H in per unit is the accelerating time if the accelerating torque is equal to rated torque. Applying this to our example: H = 11.2 and the average accelerating torque is 76%. Accelerating time (t) = 11.2/0.76 x 2 = 29.5 seconds.

Some types of loads, such as ball mills, have very low inertia values, but their load torque may approach that developed by the motor if the rate of acceleration and the accelerating torque is low. Large fans may have both high inertia values and appreciable torque requirements. In such cases, it is desirable to determine the total energy input to the rotor, both as a result of accelerating the inertia, and of overcoming the load torque. Since all of the torque developed by the motor must either accelerate inertia or overcome load torque, and since slip loss is a function of this torque and of slip, it is readily possible to determine this energy input.

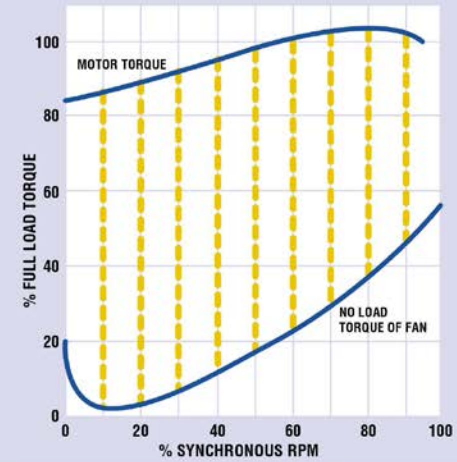


Figure 20 *Motor and load torques for centrifugal fan, illustrating method for determining accelerating time of the motor.*

% RPM	% Motor Torque	% Fan Torque	% Net Torque	Ft.Lb. Torque	Time in Seconds
5	86	5	81	2840	1.46
15	88	3	85	2980	2.78
25	91	6	85	2980	2.78
35	93	10	83	2920	2.84
45	97	14	83	2920	2.84
55	99	19	80	2800	2.96
65	102	25	77	2700	3.08
75	104	32	72	2520	3.30
85	104	40	64	2240	3.80
95	100	50	50	1750	4.75
					30.59

Figure 21

Referring again to the motor and load characteristics covered by Figure 20 and Figure 21, we can proceed as follows:

Slip loss = 0.746 x TM x S x HP x t, in kilowatt seconds
TM = % motor torque
S = % slip
t = Time at that torque

Note that slip loss is proportional to average slip and to average total motor torque (not net accelerating torque) for each 10% speed interval considered. As shown in Figure 21, total loss in the rotor is 6400 kW seconds.

Since it was previously determined that the loss due to accelerating the inertia was 5020 kW seconds, we now determine the additional loss due to the load is 6400 - 5020 = 1380 kW seconds.

In some cases the speed of the driven unit will differ from that of the driving motor, because of belting or gearing arrangements. All WK_2 values used in determining accelerating time or energy absorption must be effective values referring to the motor speed.

$$\text{Effective } WK_1^2 = WK_2^2 \times \frac{(rpm_2)^2}{(rpm_1)^2}$$

Effective WK_1^2 = WK^2 referred to the motor shaft rpm
WK_2^2 = Inertia of driven machine in lb.ft.^{2} @ rpm_2
rpm_2 = Speed of load shaft in revolutions per minute
rpm_1 = Motor shaft speed in revolutions per minute

Pull-in Torque

The pull-in point (when a synchronous motor changes from induction to synchronous characteristics and operation) is usually the most critical period in the starting of a synchronous motor. The torques developed by the amortisseur and field windings become zero at synchronous speed. They cannot pull the motor into step, so, the reluctance torque and the synchronizing torque (resulting from exciting the field windings with direct current) take over.

Reluctance Torque

Any magnetic object tends to align itself in a magnetic field so that the magnetic reluctance is at a minimum. There are as many positions of a salient pole rotor within the rotating stator magnetic field as there are poles. Below synchronous speed, a pulsating torque called reluctance torque, is developed. It has a net average value of zero and a frequency equal to twice the slip frequency. This torque may lock a lightly loaded rotor into step and develop approximately 30% pull-out torque.

Synchronizing Torque

When excitation is applied, definite polarity is developed in the rotor poles. If the slip is low enough, the attraction between unlike poles in the rotor and the stator will lock the rotor into step with the rotating stator magnetic field. Neglecting the influence of reluctance torque, the value of slip from which the motor will pull into step may be expressed as follows:

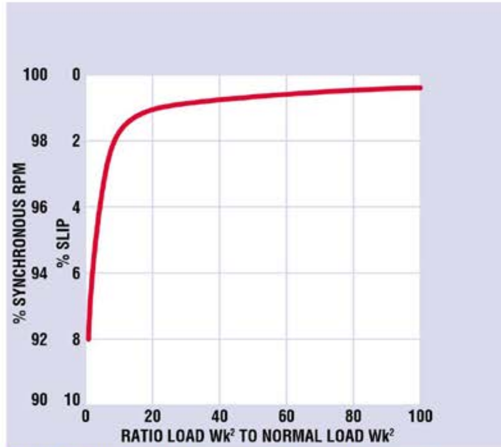


Figure 22 *Relationship between load inertia and minimum slip for pulling into step of a specific synchronous motor.*

$$S = \frac{C}{rpm} \times \frac{HP}{WK^2}$$

S = Slip in % of synchronous speed
WK^2 = Inertia of motor rotor plus load inertia
C = Constant taking into account such factor as efficiency, power factor, etc.

The relationship between slip and inertia in a given motor is shown by the curve in Figure 22. The larger the inertia the higher the speed required to achieve synchronization.

Load Inertia

Since load inertia is so important in determining the speed to which the motor must accelerate before pull-in is possible, NEMA has established normal load WK^2 values. These are covered by the formula:

$$\text{Normal load } WK^2 = \frac{3.75 \times HP^{1.15}}{(rpm/1000)^2}$$

Figure 23 illustrates the relationship between rated horsepower, speed and normal load WK^2.

We may designate as "inertia factor" the ratio of actual load WK^2 to normal load WK^2. This varies over a wide range. Typical values are shown in Figure 24.

NEMA defines pull-in torque as the maximum constant torque under which the motor will pull its connected inertia load into synchronism at rated voltage and frequency, when excitation is applied. This is sometimes referred to as "load" pull-in torque.

From Figure 24, we note that the inertia factor of the connected load may vary from 1 to 100. For the specific motor covered by Figure 22, pull-in can be affected at 8% slip with normal load WK^2. However, with an inertia factor of 100, the motor must reach 99.2% of synchronous speed on the cage winding before the motor will pull into step on application of excitation. A motor will therefore have a much higher "load" pull-in torque when driving a load having a low inertia factor than with a load having a high inertia factor.

It is rarely possible for the motor manufacturer to shop test the pull-in torque of a motor under actual load conditions. As a practical substitute, the motor torque developed at 95% speed ("nominal" pull-in torque) is used as a design and test point to predetermine the pull-in capability of the motor. Since the torque at 95% speed is a steady state condition, and actual pull-in is transient, it is necessary to take into account inertia, motor reactance and other factors.

To investigate the relationship of "load" pull-in torque and "nominal" pull-in torque, assume an application of a centrifugal fan having an inertia factor of 20, and connected to a 1000 HP, 1200 rpm motor. The fan load at synchronizing speed, with closed discharge, is 50% of full load torque. Assume it is determined that the motor slip must not exceed 1.8% if the motor is to pull into step on application of excitation. In this case, the effect of reluctance torque is neglected and it is assumed excitation may be applied at the worst phase angle for synchronizing.

In Figure 25, point A is the speed which must be attained on the amortisseur winding to permit synchronizing. Assume that the torque developed by the amortisseur winding is directly proportional to slip from zero slip (synchronous speed) to 5% slip (95% speed). We can then extend line 0A, as a straight line to B, at 95% speed. Point B will then determine the torque required at the 95% speed point (in this case 135%). This value is the "nominal" pull-in torque.

If, instead of the centrifugal fan discussed above, a motor of this rating were connected to a centrifugal pump, again having a load a synchronizing speed of 50% but having an inertia factor of 1, it would pull into step, on application of excitation, from a speed of 96% or at a slip of 4%. This is indicated by point C. The corresponding "nominal" pull-in torque at point D is 63%.

This illustrates two applications, each having load pull-in torques of 50%, one requiring 135% "nominal" pull-in torque and the other 63% "nominal." The difference is due to the connected inertia load.

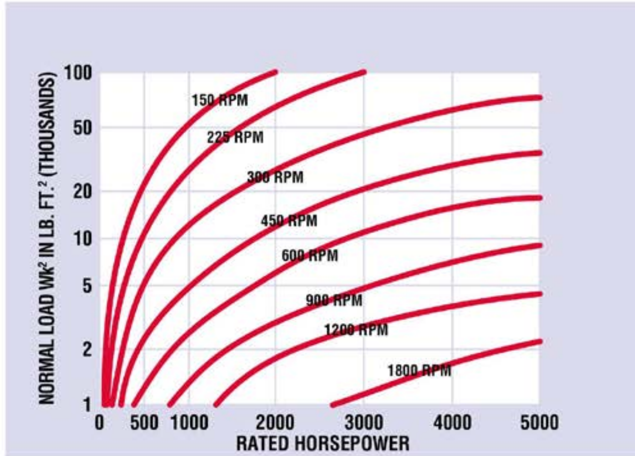


Figure 23 *Curves show the relationship between rated horsepower, speed and normal load WK^2 for synchronous motors.*

Type of Load	(Inertia Factor) Load WK^2/Normal Load^2
Ball Mill	3
Band Saw	100
Centrifugal Fan	12-60
Chipper	30-100
Pump, Centrifugal	1
Air Compressor	10
Hammer Mill	25

Figure 24

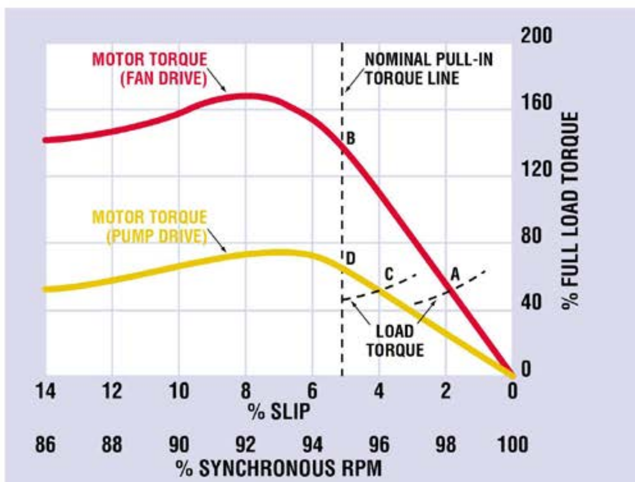


Figure 25 *Relation of load and "nominal" pull-in torque, which is the torque required at 95% speed of pulling into step.*

Figure 26

1	2	3	4	5	6	7
Ball Mill	1000	180	4	4.3	100	115
Fan	1000	1200	20	1.8	50	138
Pump	1000	1200	1	4.0	50	63
Centrifugal Compressor	2500	900	15	1.8	50	138
Chipper	1500	277	50	1.0	10	50

1. Type of load
2. Horsepower
3. RPM
4. Assumed inertia factor
5. Maximum slip from which motor will pull-in, in percent
6. Load torque required at pull-in
7. Nominal pull-in torque

Figure 26 shows various common loads having typical horsepower and speed values, typical inertia factors and corresponding values of maximum permissible slip for synchronizing, with typical load pull-in torque values, and corresponding nominal pull-in torques.

The load pull-in torque value (indicated in column 6) for chippers is merely friction and windage as chippers must be started, accelerated, and synchronized unloaded.

In some cases, in order to obtain an adequate rate of acceleration, it will be desirable to design for higher nominal pull-in torque values than those shown in column 7 of Figure 26.

Frequently, where it has been difficult for the motor to pull into step on application of excitation, raising the applied excitation voltage will facilitate pull-in.

During the starting interval, the field winding is short-circuited through the field discharge resistor. As shown in Figure 27, the frequency of the field discharge current is line frequency (60 Hertz on a 60 Hertz system) at standstill, declining to 3 Hertz at 95% speed. The time constant of the field winding is quite long. The excitation current builds up relatively slowly when excitation voltage is applied. As shown in Figure 28, if the excitation is applied at the proper interval of the field discharge current wave, it will readily build up to a maximum value, being assisted by the field discharge current, thus pulling the rotor into step promptly and with a minimum of line disturbance. Improved synchronization can be obtained by proper timing of field application.

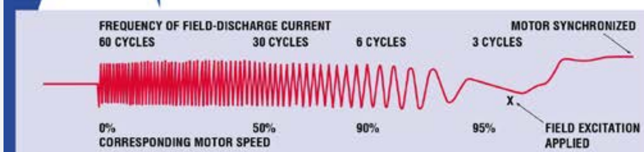


Figure 27 *Frequency of the current induced in the field winding of a synchronous motor declines as the motor comes up to speed.*

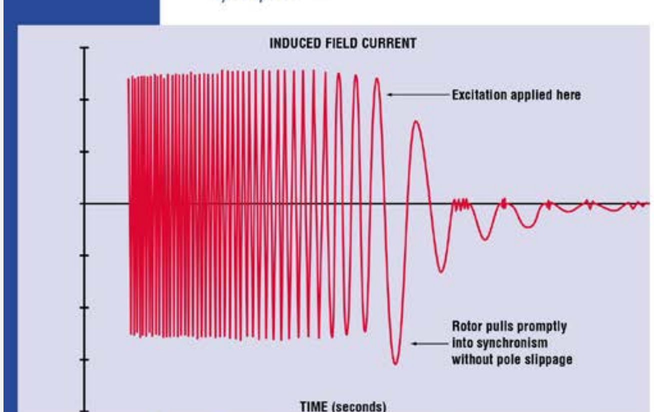


Figure 28a *Oscillogram shows synchronizing of the motor by application of dc excitation. If the point of application anticipates the interval before synchronizing as shown, the excitation will readily build up to pull the rotor into step smoothly.*

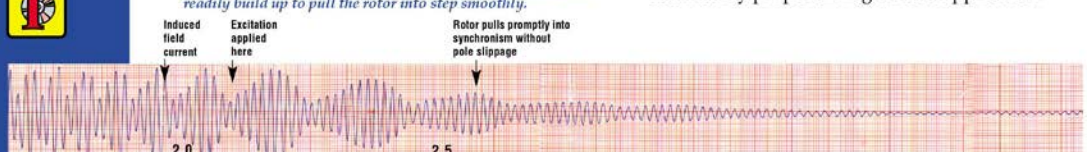


Figure 28b *Stator Current. Rotor pulls into synchronism with minimum stator current. (Note: Pulsations of current after synchronism are oscillations due to very large flywheel effect of the load.)*

Synchronous Torque

The unlike poles of the stator rotating magnetic field and of the rotor lock together and operate simultaneously. Figure 29 represents conditions of a typical motor at no load, rotating in a counter-clockwise direction. The South pole, S, of the rotor is directly opposite the North pole, N, of the stator. At that point all torques are zero. However, at any displacement from that position, as with the rotor pole dropping back to the position S', a resultant torque along curve C will be developed. This is the resultant of torque curves A and B.

The torque curve A is due to magnetic reluctance. It becomes zero at 90° displacement and is negative from that point to 180°. Torque curve B is that due to definite polarity from the excited poles. It becomes a maximum at 90° and zero at 180°.

The resultant C curve is the synchronous torque. It reaches a maximum at approximately 70° lag of the rotor behind the rotating stator magnetic field and is unstable from that point to 180°. If the load suddenly increases, the rotor will pull back and oscillate at its natural frequency about the next load point. There will be a slight variance in the speed for a moment.

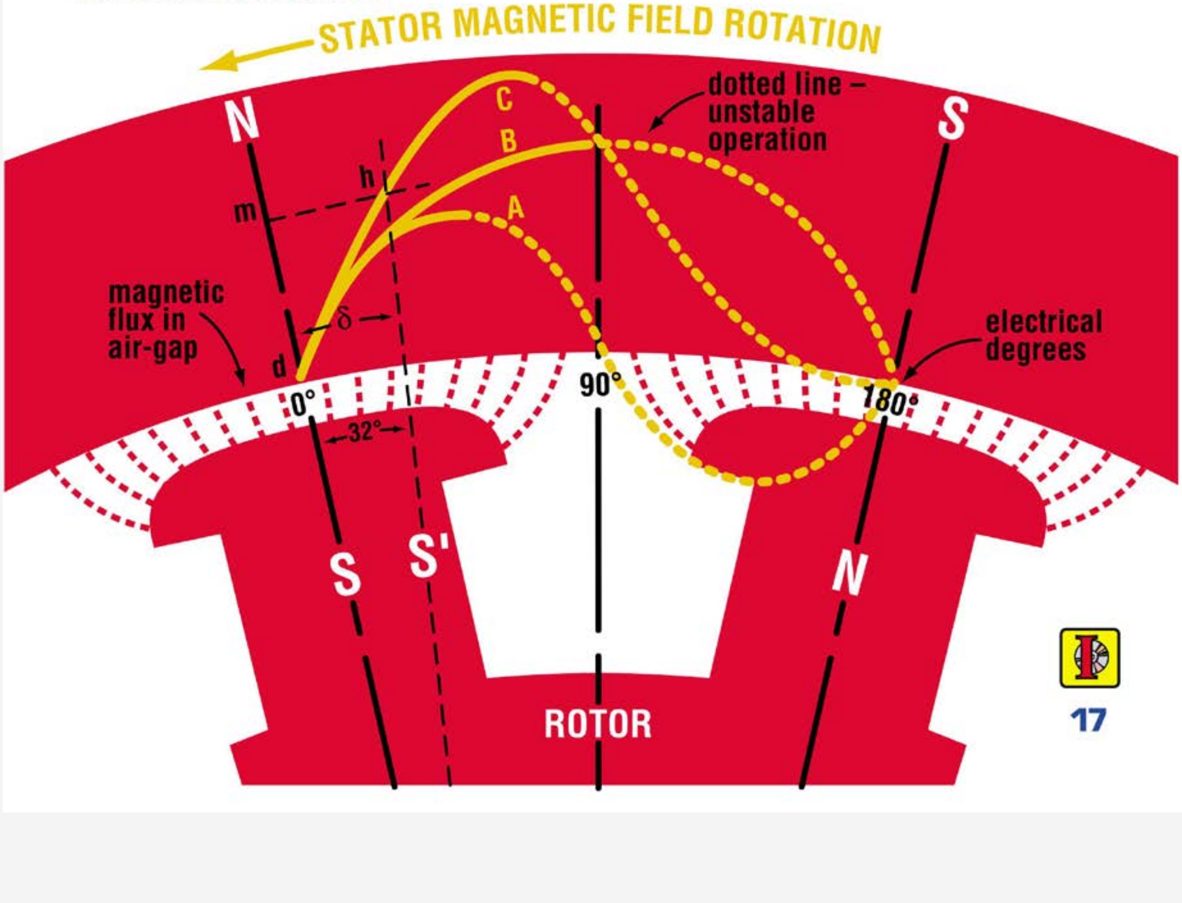
Synchronizing power is measured in kilowatts along the line d-m. The angle of lag is the angular distance between S and S'. In this case we assume it to be 0.56 radian (32°) at full load. Then synchronizing power Pr is defined as:

$$Pr = \frac{kW}{d}$$

Pr = Synchronizing power in kilowatts per radian at full load displacement
kW = Power measured at motor shaft
d = Displacement in electrical radians

Pr is important because it is a factor of the stiffness of magnetic coupling between the stator and the rotor. It affects the pull-out torque and also the neutral frequency. Pr is approximately HP x 1.35 for unity power factor, and HP x 1.8 for 0.8 leading power factor motors.

Figure 29 *Diagram shows conditions in a synchronous motor when operating in synchronism and at no load. When the motor is loaded the rotor will drop back along the red curve C, the curve of synchronous torque, sufficiently to develop the load torque. C is the resultant of magnetic reluctance torque, A, and the definite polarity torque, B. The maximum synchronous torque is reached at about 70 electrical degrees lag of the rotor.*



Pull-out Torque

NEMA defines pull-out torque of a synchronous motor as the maximum sustained torque which the motor will develop at synchronous speed for one minute at rated frequency and normal excitation. Normal pull-out torque is usually 150% of full load torque for unity power factor motors and 200% to 225% for 0.8 leading power factor motors. It can be increased by increasing the field strength, which generally means increasing the air-gap and flux density. This usually means increasing the physical size of the motor.

As shown in Figure 29, any sudden increase in load is accompanied by an increased angle position of the rotor and stator and a corresponding increase in line current. Due to transformer action, there is a transient increase in excitation and pull-out torque. This increased value disappears in a few cycles so it can be effective only on instantaneous peaks. However, the change in excitation can be used to "trigger" an increase in excitation voltage, thus tending to maintain the value of excitation established by transformer action.

The reactance of the motor windings will determine the amount of increase in excitation current and pull-out torque for a typical motor. The effect is shown in Figure 30.

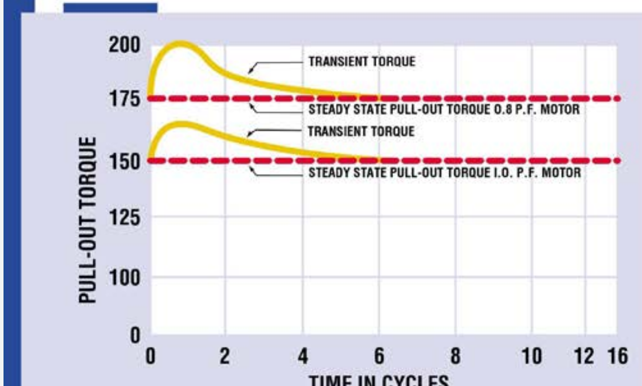


Figure 30 *Transient pull-out torque of a synchronous motor. This torque is influenced by reactance of the motor windings.*

Pulsating Torque

Since the rotor of a salient pole synchronous motor has poles and interpole spaces, its magnetic circuit varies from point to point around the rotor. The instantaneous starting or accelerating torque is greater when the flux set up by the stator passes a rotor position where the poles provide a good flux path. Less torque is developed when passing between poles. It is common to show a torque curve as a smooth curve from zero to 95% speed; this is really the average torque. At a given speed, the torque varies from maximum to minimum at twice slip frequency. At standstill, the torque pulses at 120 Hz. As shown in Figure 31, the torque at 95% speed has a pulsating component at 2 x 60 x (1.00-0.95) = 6 Hz; one cycle every 0.167 second. At 80% speed this is 2 x 60 x (1.00-0.80) = 24 Hz, or 0.042 second per cycle. Figure 32 demonstrates torque pulsation at 80% speed.

Synchronous motor 1750 HP at 240 RPM with Paper Mill Dripproof™ enclosure and double shaft extensions driving vacuum pumps in a paper mill.

The pulsating component of torque needs to be carefully evaluated as to its effect on the combined motor driven equipment. A typical pulsating torque is approximately 20% of rated torque. Typically, we have the mass of the motor, the gear and the load (i.e. compressor) all connected by the shaft. Each individual part has a natural frequency as well as the combinations. The responsibility to avoid a torsional natural frequency that is near running speed is that of the system designer. Further, special consideration has to be given to the pulsating torque in certain applications. For example, large 6 pole motors driving compressors frequently have a resonance between 60 and 80% speed. Fortunately, it is only present during starting. If the start-up is fast, the resonance has little time to build up. The motor manufacturer provides curves of the average and the pulsating torque so that an overall system analysis can be made. Figure 33 is a representation of a torque vs. speed curve. Shaft sizes may be adjusted or the coupling changed, but the pulsating torque will still be there.

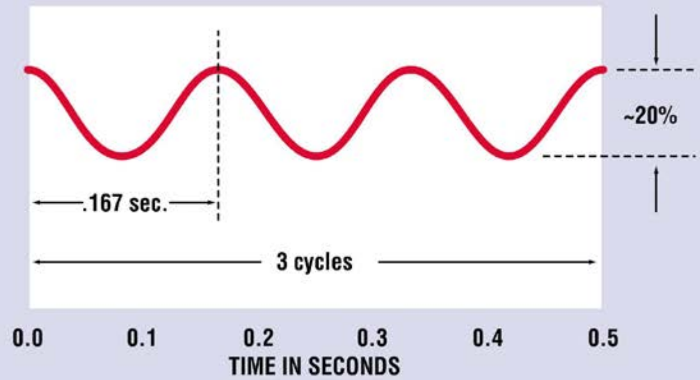


Figure 31
Pulsations in total torque due to reluctance variations in magnetic circuit at 95% speed; 6 Hz. or .167 sec./cycle.

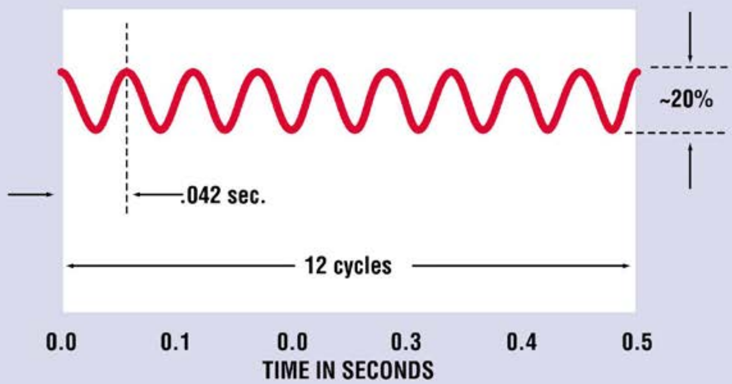


Figure 32
Pulsations in total torque due to reluctance variations in magnetic circuit at 80% speed; 24 Hz. or .042 sec./cycle.

Figure 33
Typical speed-torque curve showing average torque.

Starting & excitation methods for synchronous motors

Starting

As will be noted from the application section, the starting (zero speed) torques required of salient-pole industrial synchronous motors range from 40% to 200%.

The required pull-in torque depends upon the load torque and the total inertia. As covered in the previous section, the nominal pull-in torque is the actual torque the motor develops at 95% speed.

The starting kVA depends on both the starting torque and the nominal pull-in torque. For a given starting torque, the starting kVA of a synchronous motor in percent of the full-load kVA will increase with an increase in the ratio of nominal pull-in torque to starting torque. Figure 34 shows some typical values.

% Starting Torque	% Nominal Pull-in Torque	% Starting Kva
80	60	355
80	80	410
80	100	470

Figure 34

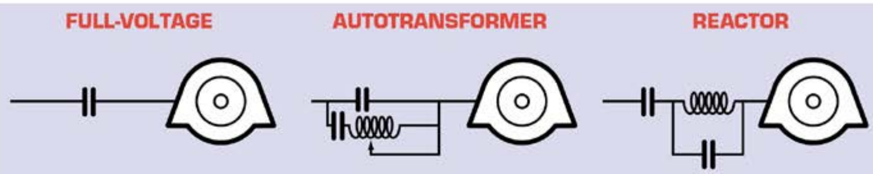
The starting kVA of synchronous motors often taxes the capacity of the generating or distributing system to which it is connected. Extreme care must then be taken in analysis of the system's capacity, torque requirements, starting kVA, and starting method.

Full-voltage starting, due to its simplicity, should be used wherever possible. Most systems can stand dips of up to 20%. System capacities are usually expressed as MVA short circuit capacity. A quick approximation of the voltage dip during starting can be made by converting the motor inrush to MVA as follows:

SCMVA = System short circuit capacity in MVA
MMVA = (% motor inrush/100) x (motor kVA/1000)
MMVA = motor starting kVA
Motor kVA = HP x 0.746/(P.F. x Eff.)
% dip = MMVA x 100/(SCMVA + MMVA)

In some cases a reduction in starting kVA is necessary to limit voltage dip. Line diagrams for various reduced kVA starting methods are illustrated in Figure 35.

Figure 36 shows motor voltage, line kVA and motor torque relationships for various methods of



Full-voltage starting should always be used unless (1) limited capacity of power system makes reduced kVA starting necessary, or (2) torque increments are required in starting.

Line kVA is reduced approximately as the square of the motor terminal voltage, as is the motor torque. Closed transition auto-transformer starters avoid the line surges experienced with earlier open-transition switching.

Line kVA is reduced approximately directly as the motor terminal voltage, but torque is reduced as the square of the voltage. The starter has the advantage of simplicity and closed transition switching.

Figure 35 *Common starting methods for synchronous motors.*

Type of Starting	Motor Voltage in % of Full Voltage	Line kVA in % of Full Voltage kVA	Motor Torque in % of Full Voltage Torque
Full Voltage	100	100	100
Reduced Voltage	78	60	60
Reactor	60	60	36

Figure 36

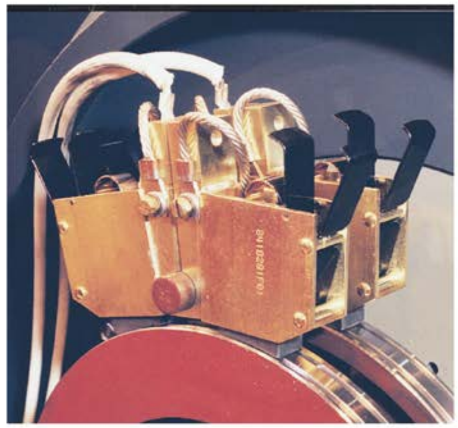
starting. Figure 37 shows full voltage starting kVAs for 1.0 P.F. motors at high and low speeds.

It is important that sufficient torque be developed to start and accelerate the load properly. In some applications, incremental starting which employs part winding, reactor, resistor, or autotransformer is used. The motor need not start on the first step, but additional torque increments are successively applied until the motor starts and accelerates. This method eases the mechanical shock on driven equipment and, on regulated systems, permits restoration of normal voltage before application of the next torque and kVA step.

The motor pulls into step on application of excitation to the rotor field winding. The starting method applies excitation at the proper speed and phase angle to assure proper synchronizing.

Excitation

All synchronous motors need a source of direct current for their field winding. Historically, this was supplied by an exciter, a specialized dc generator which may have been direct-connected to the shaft of the motor so that if the motor is running, dc is available. In the past, conventional



Collector ring and brushes are not required or used with brushless excitation.

exciters were small dc generators with commutators to convert the generated voltage to dc. Brushes allow the current to come from the exciter to the stationary control center. The current is then routed back through slip-rings and brushes to the motor field. This system works very well except for a minor nuisance, the brushes. Brushes can last as long as a year, but there is the carbon dust to clean up and continual operator visual inspection is required. Occasionally conditions will change and brushes will wear rapidly. The selection of the proper grade of brush depends on current density, ring material, ring speed, humidity, brush pressure, and many other factors. It may take a long time to find the right brush. Put this together with the brush and ring maintenance and there's an opportunity for a better method.

APPROXIMATE FULL-VOLTAGE STARTING kVA OF UNITY POWER FACTOR* SYNCHRONOUS MOTORS
(In % of full-load kVA of motor; for various starting, pull-in and pull-out torques; for 50 and 60 Hertz motors)

High-Speed Motors (500 to 1800 rpm)				Low-Speed Motors (450 rpm and Lower)			
% Torques			% Starting kVA	% Torques			% Starting kVA
Starting	Pull-in**	Pull-out	at Full Voltage	Starting	Pull-in**	Pull-out	at Full Voltage
50	50	150-175	375	40	40	150	250-325
50	75	150-175	400	50	75	150	350-375
75	75	150-175	400	75	110	150	500
75	110	150-175	500	100	50	150	375-400
110	110	150-175	500	160	110	150	575
125	110	200-250	550-600	100	110	200-250	600-650
150	110	150-175	550	125	110	200-250	600-650
175	110	200-250	600-650	125	125	200-250	600-650

*For 0.8 power factor motors the percent starting kVA will be approximately 80% of the values shown.
** The above pull-in torques are based on normal load WK^2.
NOTE: The percent starting kVA in the tables above are approximate for estimating purposes only.
Specific values for particular motors can be furnished.

Figure 37

Brushless excitation is more reliable with no brush or collector ring maintenance

By the 1960s, solid state diodes and thyristors had advanced to where they could carry the current and block the voltages as required. It was then that Electric Machinery Manufacturing Company developed the brushless exciter (see Figure 38). The exciter is physically direct connected as before. The rotor has a three phase ac armature winding. The stationary field winding is on poles on the stator and is connected to a variac and rectifier or equivalent source of variable dc. The generated ac current is directly connected along the shaft to a rotating diode wheel, where it is rectified to dc before going to the motor field. The magnitude of the motor field current is adjusted by changing the current to the stationary exciter field.

The most amazing part of this design was the mounting of the control on the rotor. Starting a synchronous motor requires shorting the field with a discharge resistor and blocking the dc current until the rotor is near full speed. The dc current is then applied and the discharge resistor is removed. For the brush type machines this was in a control cabinet the size of a four drawer file cabinet. Now it is a small package on the rotor.

To show what is required of the control, consider the starting of a synchronous motor. The ac breaker closes, applying three phase voltage to the stator winding. The stator winding has been wound to form a number of magnetic poles depending on the rated speed of the motor. These poles on the stator cause a magnetic field to rotate at rated speed. The rotor has not started to move, so the field winding and cage are swept by the rotating magnetic field. This will induce a very high voltage (thousands of volts) in the field winding. To avoid this, the field is shorted by a discharge resistor. In addition, this resistor is designed to give additional pull-in torque. As the rotor accelerates, the frequency of the discharge current is proportional to the difference in speed between the stator flux and the rotor winding (slip). At synchronous speed there is no slip, so no voltage is induced and no torque is produced by the cage. Since there is some load torque, the speed never reaches synchronous speed but reaches equilibrium where the decreasing cage torque matches the load torque. At this point, dc is applied to the field, creating strong magnetic North and South poles. These are attracted to the opposite magnetic poles on the stator. If the attraction is great enough, the rotor with its inertia will be pulled up to synchronous speed. The rotor is thereafter locked to the rotating stator magnetic field.

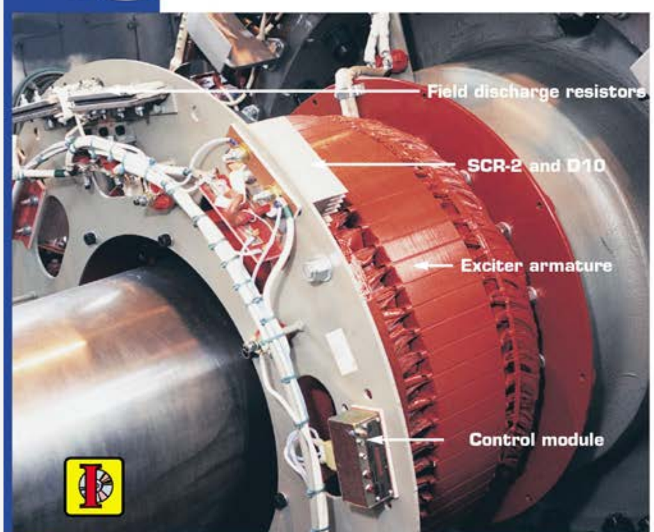


Figure 38 *Shaft mounted brushless exciter.*

Field Application System

Functions of Field Application System:

1. Provide a discharge path for the current induced in the field of the motor during starting, and open this circuit when excitation is applied.
2. Apply field excitation positively when the motor reaches an adequate speed. This excitation should be applied with such polarity that maximum torque will be obtained at the time of pull-in.
3. Remove excitation and reapply the field discharge resistor immediately if the motor pulls out of step.

The circuit for accomplishing this is shown in Figure 39. The field discharge resistor protects the motor field winding from the high voltage induced in starting and provides the voltage source for the control circuit. The ac output of the exciter is converted to dc by the *rotating* rectifier diodes. This output is switched on or off to the motor field winding by silicon controlled rectifier SCR-1, which is gated by the control circuit.

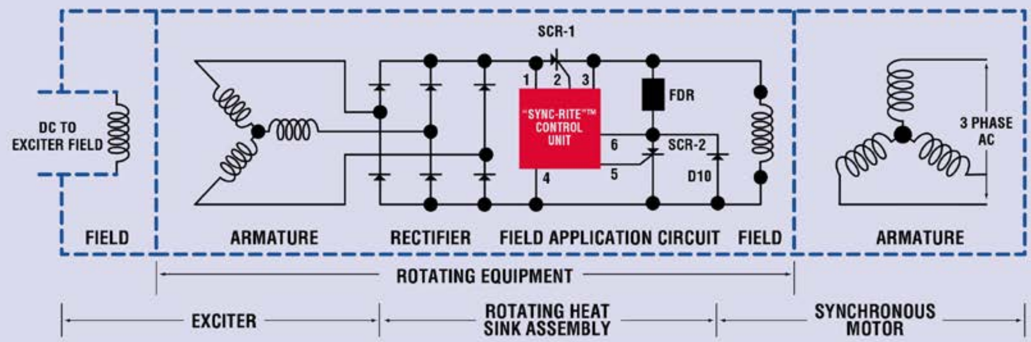


Figure 39

Control circuit

The control circuit keeps the SCR-1 from firing until the induced field current frequency is very low, representing a close approach to synchronous speed, and then fires the rectifier SCR-1 at the proper time and applies excitation to the synchronous motor field. At the same time, the field discharge resistor is removed from the circuit. This is done by the inherent operating characteristics of silicon controlled rectifier SCR-2. This frequency sensitive part of the control circuit assures that field excitation is applied at the proper pull-in speed for successful synchronizing and at the proper polarity to give maximum pull-in torque with minimum line disturbance.

The control circuit operates to remove excitation should the motor pull out of step due to a voltage step or excessive mechanical load. On the first half cycle after pull-out, the induced field voltage causes the net field current to pass through zero, turning SCR-1 off, automatically removing excitation. SCR-2 operates to connect the field discharge resistor back in to the circuit. During this time the motor operates as an induction motor. When conditions permit, field is then re-applied as during starting.

In Figure 39, the voltage from the exciter-rectifier is blocked by SCR-1 until the point of synchronization. The field has an alternating voltage causing current to flow first through SCR-2 and the discharge resistor. On the next half cycle, current flows through the diode and discharge resistor. The control circuit waits until the frequency drops to the preset value, indicating the rotor is at an adequate speed. Then, after a North pole on the stator is in the right position to be attracted to what will be a South pole on the rotor, it triggers SCR-1 to apply excitation. If the rotor does not synchronize, it will slip a pole, the induced field voltage will oppose the exciter voltage causing the current to go to zero, turning SCR-1 off. SCR-2 is turned on only at a voltage higher than the exciter voltage so it will not be on when SCR-1 is on.

Occasionally a lightly loaded motor will synchronize without excitation being applied. This is due to the reluctance torque. Reluctance torque results from the magnetic circuit having less reluctance when the poles line up with the stator flux. The EM design includes a "zero slip" circuit to apply excitation in these situations.

While this may seem complex, it has proven to be a highly reliable system.

Features of the brushless exciter

1. No brushes, no carbon dust problems, no brush maintenance.
2. No commutator or slip rings to resurface.
3. Completely automatic field application at the best angle for sure synchronization.
4. Removal of excitation and application of field discharge resistor in the event of out of step operation and automatic resynchronization.
5. No sparking. Can be used in hazardous areas.
6. No field cubicle with FDR and field contactor to maintain.

APPLICATIONS

Where synchronous motors should be applied

Synchronous motors can handle any load which can be driven by a NEMA design B squirrel-cage motor. Whether they should be used for any specific application is a matter of some investigation. A rough rule of thumb is that synchronous motors are less expensive than squirrel-cage motors if the rating exceeds 1 HP per rpm. However, that considers only initial cost and does not take into account:

1. Higher efficiency of the synchronous motor.
2. Power factor improvement of the synchronous motor.

These two factors become important at speeds below 500 rpm where induction motor characteristics leave much to be desired. On the other side of the balance are these:

3. Necessity for excitation source and field control means for the synchronous motors.
4. Relatively low torque/kVA efficiency in starting.
5. Slightly greater maintenance cost, especially with motors with slip rings and brushes.

Synchronous motor 2500 HP at 327 RPM with WPII enclosure driving a reciprocating compressor in a refinery.

A few general rules may be established as a preliminary guide in selection of synchronous vs. induction motors, but specific local conditions must govern:

1. If especially low starting kVA, controllable torque or adjustable speed are desired, a synchronous motor and slip coupling may be used.
2. At 3600 rpm, synchronous motors are sometimes used from 10,000 - 20,000 HP. Above that range they are the only choice.
3. At 1800 rpm, motors show an advantage above 1000 HP. From 2000 to 10,000 HP, synchronous motors are used if power factor improvement is especially important. Above 15,000 HP, they are the only choice.
4. Induction motors for operation below 500 rpm have lower efficiency and a lower power factor. Synchronous motors are built at unity or leading power factor and with good efficiencies at speeds as low as 72 rpm. For direct connection in ratings above 1000 HP and in speeds below 500 rpm, the synchronous motor should be the first choice for compressors, grinders, mixers, chippers, etc.

Figure 40 is a general sizing chart for synchronous and induction motors.

Leading power factor motors are slightly less efficient than unity power factor motors, but they do double duty. Their application depends on the necessity for power factor correction.

Induction motors **require** from 0.3 to 0.6 reactive magnetizing kVA per HP of operating load. 0.8 leading power factor synchronous motors will **deliver** from 0.4 to 0.6 corrective magnetizing kVA per HP depending on the mechanical load carried. Equal amounts of connected HP in induction motors and 0.8 leading power factor synchronous motors will result in approximate unity power factor.

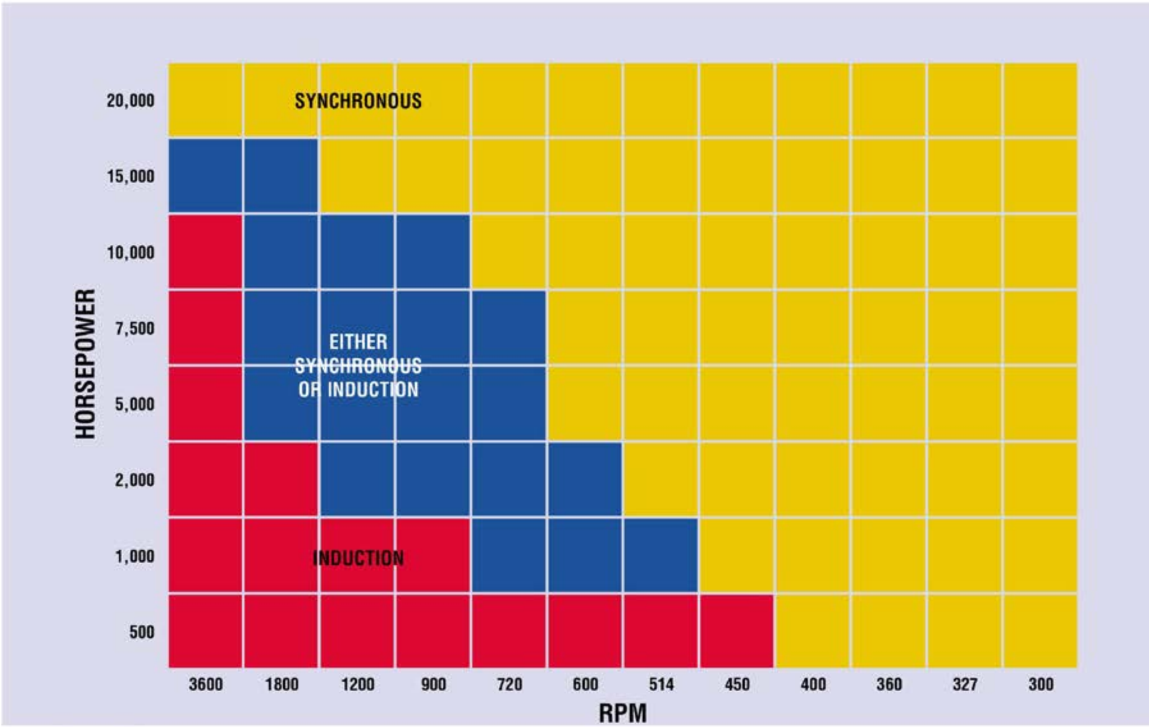


Figure 40 *General areas of application of synchronous and induction motors.*

Synchronous motors for air and gas compressors and vacuum pumps

Air and gas compressors and vacuum pumps may be reciprocating, rotary or centrifugal. Synchronous motors may be used with any of these types. Detailed application information follows:

Reciprocating compressors and vacuum pumps

Many more synchronous motors are direct connected to reciprocating compressors than are applied to all other types of loads combined. Factors contributing to this are:

1. Low starting and pull-in torque requirement of unloaded reciprocating compressors.
2. Elimination of belt, chain and gear drives.
3. High efficiency of low speed synchronous motors direct connected to compressors.
4. Power factor of low speed synchronous motors direct connected to compressors.
5. Minimum floor space requirement.
6. Low maintenance cost.

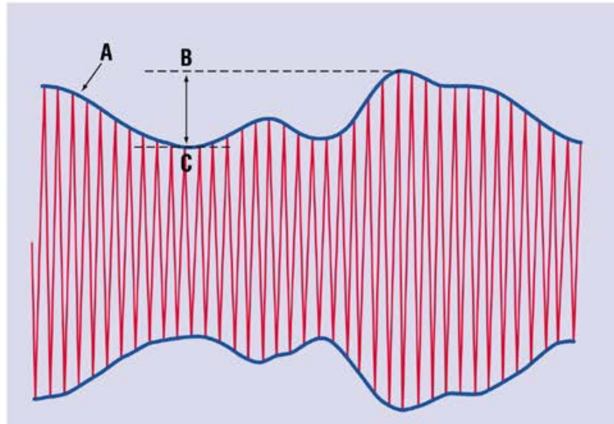
Motors designed for direct connection to reciprocating compressors are usually engine type or single bearing construction. The stator is mounted on soleplates set in the foundation. A special case is the flange-mounted motor in which the stator frame is mounted on a flange on the compressor frame. In either case, torque requirements are usually within the range of normal design, low speed motors. See Figure 41.

The torque values may be exceeded, for certain compressors and vacuum pumps, the manufacturer should be checked for specific values.

NEMA Recommended Torques for Reciprocating Compressors	Torques in % of Full-load Torque		
	Starting	Pull-in	Pull-out
Air or Gas – Starting Unloaded	40	30	150
Vacuum Pumps – Starting Unloaded	40	60	150

Figure 41

Figure 42
Oscillogram of synchronous motor driving a reciprocating compressor. Line A is envelope of current wave, B-C is current pulsation. B-C divided by rated full-load motor current is percent pulsation = 55.2%.



Current pulsation

A reciprocating compressor has a varying torque requirement per revolution, depending on number of cylinders, crank angle, etc. As discussed previously, the angle of rotor lag and the amount of stator current vary with torque. There will be a cyclic pulsation of the load current during each revolution. This in itself is not too objectionable, but excessive current pulsation may result in appreciable voltage variation and thus affect lights or other devices sensitive to voltage change.

Present standards limit current pulsation to 66% of rated full-load current, corresponding to an angular deviation of approximately 5% from a uniform rotative speed. The current pulsation is the difference between maximum and minimum values expressed in percent of full-load current. Figure 42 shows an actual oscillogram of current input to a synchronous motor driving an ammonia compressor.

In some cases, step unloading of compressors involving successive unloading of various cylinder ends is used. This introduces additional irregularity into the crank effort and usually increases the current pulsation.

Natural frequency

Any system having mass in equilibrium and having a force tending to return this mass to its initial position, if displaced, will tend to have a natural period of oscillation. The common pendulum is a good example.

The natural frequency of this oscillation in a synchronous motor is determined by the formula:

$$NF = \frac{35200}{N}\sqrt{\frac{Pr \times Hz}{WK^2}}$$

NF = Natural frequency in oscillations per minute
N = Synchronous speed in revolutions per minute
Pr = Synchronizing power
Hz = Line frequency (CPS)
WK^2 = Flywheel effect in foot pounds squared

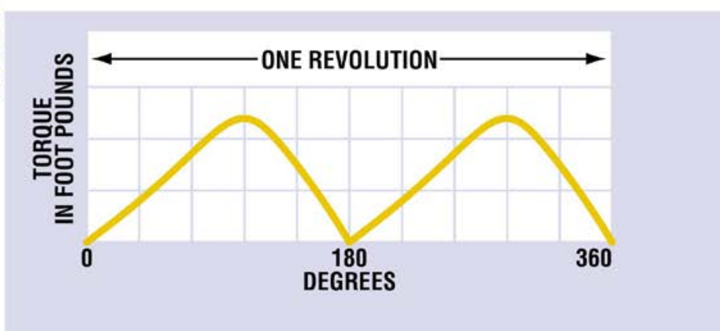
It is assumed there is no effective damping and the motor is connected to an infinite system. This natural frequency should differ from any forcing frequency by at least 20%

Compressor factor

Neglecting inertia and damping forces, a single cylinder, double acting or a two cylinder, single acting compressor will have a crank effort diagram similar to Figure 43. A two cylinder, double acting or a four cylinder, single acting crank effort diagram is illustrated in Figure 44.

Various factors may tend to disturb the symmetry of the torque effort diagrams illustrated. For instance, on the two cylinder,

Figure 43
Crank effort diagram for a one cylinder, double-acting compressor, or two cylinder, single-acting compressor.



double acting compressor with 90° cranks, a "galloping" effect due to acceleration of the unbalanced reciprocating parts is introduced once per revolution.

Two-stage compressors usually do not have exactly equal loading on all cylinder ends. In some cases, recipriciation compressors operate on two or more suction systems operating at different temperatures and at different pressures. Various arrangements to secure part-load operation such as lifting suction valves, opening clearance pockets, cutting in partial by-pass systems, etc., may introduce various degrees of unbalance. Most types of compressors are covered by a NEMA table in which each common application is assigned an application number and "Compressor Factor." Figure 45 illustrates percent current pulsation for NEMA application No. 5 covering a single stage, two cylinder, double acting compressor with 90° cranks, plotted against "Compressor Factor" "X" as the abscissa. This curve shows that with "X" values of 2.0 to 6.0 or above 12.0 the current pulsation will not exceed 66%.

Flywheel effect

The required flywheel effect to limit current pulsation is proportional to the compressor factor "X" shown in this formula:

$WK^2 = X \times Hz \times Pr \times G$

WK^2 = Total flywheel effect in foot pounds squared

X = Compressor factor

Hz = Line frequency in CPS

Pr = Synchronizing power

$$G = 1.34 \times \frac{(100)^4}{rpm^4}$$

Flywheel effect values calculated by the above formula and using compressor factors established by NEMA will satisfy the condition that the natural and forcing frequencies differ sufficiently to meet allowable pulsation.

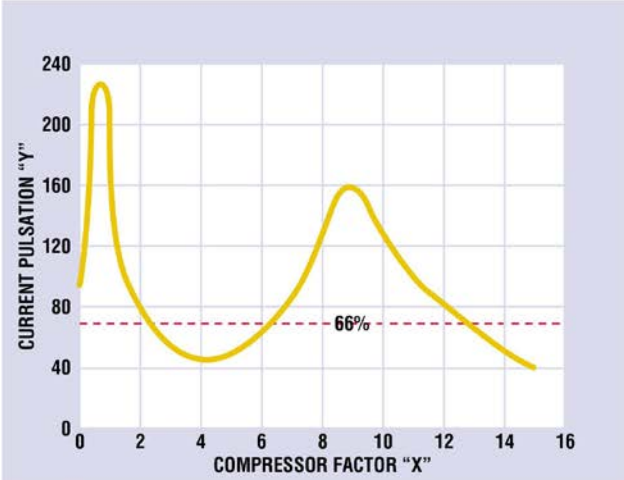


Figure 45 *X-Y curve for determining flywheel requirements.*

Centrifugal compressors and vacuum pumps

Various types of centrifugal, positive displacement compressors are used to compress air or various gas or to serve as vacuum pumps. The Hytor, or water seal type, is frequently used as a vacuum pump in paper mills, and NEMA recommends special torque consideration. See Figure 46.

NEMA Recommended Torques for Compressors	Torques in % of Full-load Torque		
	Starting	Pull-in	Pull-out
Compressors, centrifugal – starting with:			
a. Inlet or discharge valve closed	30	40-60	150
b. Inlet or discharge valve open	30	100	150
Compressors, Fuller Company			
a. Starting unloaded (by-pass open)	60	60	150
b. Starting loaded (by-pass closed)	60	100	150
Compressors, Nash-Hytor – starting unloaded	40	60	150
Compressors, reciprocating – starting unloaded			
a. Air and gas	30	25	150
b. Ammonia (discharge pressure 100-250 psi)	30	25	150
c. Freon	30	40	150

Figure 46

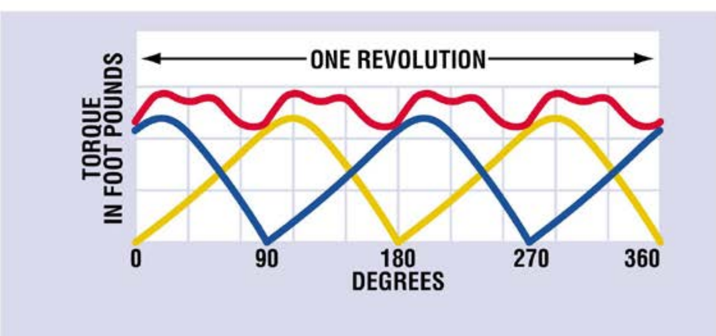


Figure 44
Crank effort diagram for a two cylinder, double-acting compressor, or four cylinder, single-acting compressor.

Centrifugal fans, blowers, compressors, and exhausters

Centrifugal fans, blowers and compressors differ from reciprocating units in that the pressure developed is a function of the density of the air or gas handled, the speed of the impeller, and of restriction to flow. The gas is accelerated in passing through the rapidly rotating impeller, this velocity being converted into pressure in the volute casing surrounding the impeller.

Units operating at pressures above 30 psi are called compressors. For pressures above 10 psi, multi-stage compression is commonly used. These units are especially suited for high volume at low pressure. NEMA recommended torques are shown in Figure 47.

Many of these applications have substantially more than normal load WK^2. The actual value should be taken into account in applying and designing motors for such loads. Care should also be taken in applying motors to fans normally handling hot gases. The load, when handling cold air, as in starting, will be substantially higher.

NEMA Recommended Torques	Torques in % of Full-load Torque		
	Starting	Pull-in	Pull-out
Blower, Centrifugal –			
Inlet or discharge valve closed	30	40-60	150
Inlet and discharge valve open	30	100	150
Compressor, Centrifugal			
Inlet or discharge valve closed	40	40-60	150
Inlet and discharge valve open	40	100	150
Fans, Centrifugal (Except Sintering)			
Inlet and discharge valve open	30	40-60	150
Inlet and discharge valve open	30	100	150
Fan, Sintering – Inlet gates open or closed	40	100	150
Fan Propeller – Discharge open	30	100	150

Figure 47

Type	Specific Speed	% Torque at Shut-off
Radial-flow	500	45
	1000	50
	2000	60
	3000	70
Mixed-flow	5000	120
Axial-flow	10000	220

Figure 48

Synchronous motors for centrifugal pumps

The term "centrifugal" may be applied to radial-flow, mixed-flow or axial-flow pumps. In all cases, energy is added to the flowing liquid by means of pressure differences created by vanes of a rotating impeller or propeller. The radial-flow pump discharges the liquid at right angles to the shaft, the mixed-flow discharges it at an angle, and the axial-flow propels the liquid in an axial direction.

Hydraulically, the difference is fundamentally one of the "specific speed" used in the design. Mathematically, it is shown by this formula:

$$Ns = \sqrt{\frac{NQ}{H^{3/4}}}$$

Ns = Specific speed
N = rpm
Q = gpm
H = Total head in feet where Q and H correspond to conditions at which maximum efficiency is obtained

These differences are important in the application of synchronous motors, as can be seen in Figure 48.

The important factor is the torque required at full speed and closed discharge. In the radial-flow centrifugal pump this is usually about 55% of full load torque. In the axial-flow it may reach 220%. Obviously, no attempt should be made to synchronize a motor driving a mixed-flow or axial-flow pump under shut-off conditions. However, the inertia of the water column alone simulates a partially closed discharge in case of attempted rapid acceleration, as in the case of pulling into step. See Figure 49.

The pull-in torques given above are inadequate where mixed-flow or axial-flow pumps are concerned. In some cases it is possible to start these pumps with partially empty casings, thus reducing pull-in requirements.

On vertical pump motors, additional thrust capacity is usually required where rigid couplings or hollow shafts are used, and the motor thrust bearing must support all rotating parts plus hydraulic load. Where thrust loads are unusually high, the break-away or starting torque required may be considerably in excess of values in Figure 49. In such cases hydraulic lift bearings should be considered.

Synchronous motors for crushers, grinders, and mills

Crushing, grinding or pulverizing is a necessary step in separation of metal from ore, preparation of crushed rock for the construction industry, preparation of agricultural limestone, and manufacture of portland cement.

Primary crushers, usually jaw, gyratory or hammermill type, are fed the rock or ore directly after blasting. The material next goes to the secondary crusher where reduction to about 1/2" may be accomplished. Secondary mills may be gyratory, cone, hammermill, or roll type.

Rod mills may also be used as secondary grinders. A rod mill consists of a steel shell lined with wear-resistant material and rotated about a horizontal cylinder axis. The mill uses steel rods 2 to 5 inches in diameter running the length of the mill. As the mill rotates, the tumbling action of the rods causes grinding.

Final grinding, possibly down to a material which will pass through a 200 mesh screen or finer, is usually done in a ball mill. This is similar to a rod mill except that steel balls tumble over the material to be crushed. Autogenous mills use the crushed rock to grind itself to powder. See Figure 50 for NEMA recommended torques for the various types of mills.

Synchronous motors for pulp and paper mills

One of the largest power consuming industries is that of processing wood and wood products in the manufacture of paper and pulp.

Although alternative fibers may be used in relatively small quantities in the manufacture of pulp and paper, by far the largest source is wood. Paper may be made either by the groundwood or chemical process. The application table and recommended NEMA torques shown in Figure 51 cover the usual synchronous motor applications in paper mills, most of which are common to both processes.

Paper mills are large users of synchronous motors for pumps, refiners, chippers and other equipment.

NEMA Recommended Torques for Pumps	Torques in % of Full-load Torque		
	Starting	Pull-in	Pull-out
Pumps, axial flow, adjustable blade – starting with:			
a. Casing dry	5-40	15	150
b. Casing filled, blades feathered	5-40	40	150
Pumps, axial flow, fixed blade – starting with:			
a. Casing dry	5-40	15	150
b. Casing filled, discharge closed	5-40	175-250	150
c. Casing filled, discharge open	5-40	100	150
Pumps, centrifugal, Francis impeller – starting with:			
a. Casing dry	5-40	15	150
b. Casing filled, discharge closed	5-40	60-80	150
c. Casing filled, discharge open	5-40	100	150
Pumps, centrifugal, radial impeller – starting with:			
a. Casing dry	5-40	15	150
b. Casing filled, discharge closed	5-40	40-60	150
c. Casing filled, discharge open	5-40	100	150
Pumps, mixed flow – starting with:			
a. Casing dry	5-40	15	150
b. Casing filled, discharge closed	5-40	82-125	150
c. Casing filled, discharge open	5-40	100	150
Pumps, reciprocating – starting with:			
a. Cylinders dry	40	30	150
b. By-pass open	40	40	150
c. No by-pass (three cylinder)	150	100	150

Figure 49

	Torques in % of Full-load Torque		
Application	Starting	Pull-in	Pull-out
Gyratory (unloaded)	100	100	250
Cone (unloaded)	100	100	250
Hammer Mill (unloaded)	100	80	250
Roll Crusher (unloaded)	150	100	250
Rod Mill – Ore	160	120	175
Ball Mill – Ore	150	110	175
Ball Mill – Rock or Coal	150	110	175

Figure 50

NEMA Recommended Torques	Torques in % of Full-load Torque		
	Starting	Pull-in	Pull-out
Refiners (unloaded)	50	50-100	150
Conical (Jordan) disc	50	50	150
Chippers – Empty (1)	60	50	250
Grinders (unloaded)			
Magazine	50	40	150
Pocket (unloaded)	40	30	150
Vacuum Pumps (Hytor)	60	100	150

(1) These are high-inertia loads and motor torque requirements can not be determined from load values alone. WK^2 values must be known to permit proper motor application.

Figure 51

Adjustable Speed

Some applications have varying requirements. For example, a process may at times require more water or air. The pump or compressor and motor are sized for the maximum, and a throttle is used to reduce the flow. This does not reduce the power used substantially, since the reduced quantity is offset by the greater head caused by the throttle. See Figure 52.

To overcome this, a slip clutch (eddy current drive) is sometimes used. Electric Machinery manufactures a magnetic drive for this use. Here the motor runs at rated speed and the magnetic drive slips, allowing the compressor to run at a lower speed. At the lower speed the output and the torque required are reduced. Although the torque is reduced, the motor still runs at rated speed, so the HP is down in proportion to the torque. There is some slip loss in the slip clutch. This type of variable speed drive is declining in use.

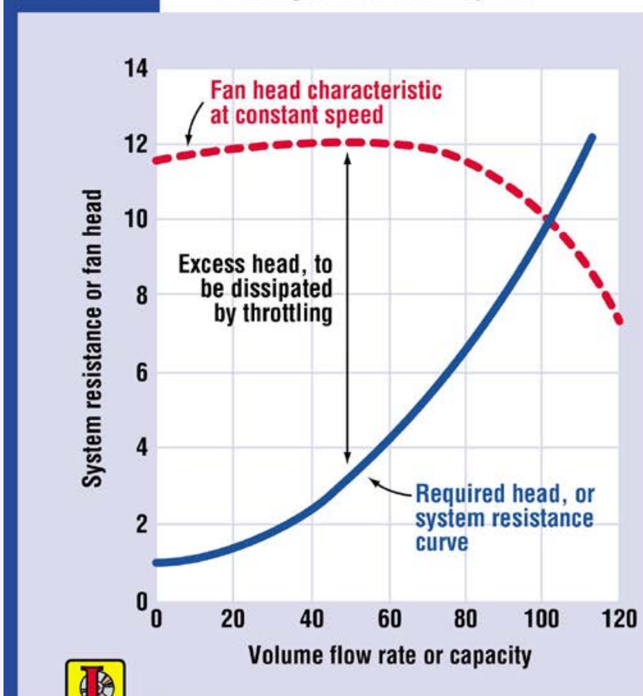


Figure 52 *System resistance and fan head characteristics.*

Adjustable speed with synchronous motors

When one thinks of synchronous motors, the idea of a motor running at an absolutely constant speed comes to mind. While this mental picture is correct, it is so only because the power line frequency is a constant, and synchronous motors are "locked in" to the line frequency.

As technology has advanced over the latter third of the twentieth century, the ability to change the frequency of the incoming line power - and thus, to control of the speed of "constant speed" motors - has become a reality. This has led to adjustable speed drives for pumps, fans, compressors and other loads, yielding higher operational efficiencies, lower wear rates, softer starting, lower starting currents, and even quieter operation as machines run at reduced speeds. See Figure 54 and compare it with Figure 53 to see potential efficiency improvements with adjustable frequency vs. eddy current drive.

A "tool" which has come out of development efforts and is finding application is the Load Commutated Inverter, or LCI. The LCI is reminiscent of the "incher," an array of six contactors used to switch direct current in sequence, first positive polarity and then negative, to the three phases of a normally excited synchronous motor, thus enabling it to slowly "inch" its load around to position it for maintenance reasons. The difference is that the LCI can do this job at a wide variety of frequencies, including frequencies higher than line frequency, thus even enabling supersynchronous (> synchronous speed) operation. The incher was limited to one slow speed.

The LCI drive (also commonly referred to as "VFD," or variable frequency drive) consists of an incoming section which supplies direct current via a current-smoothing choke to an inverter section. The inverter supplies dc power to the synchronous motor by switching it between phases at the desired frequency corresponding to speed. Because of the functions being accomplished, the inverter and motor together are sometimes referred to as a brushless dc motor. Refer to Figure 55.

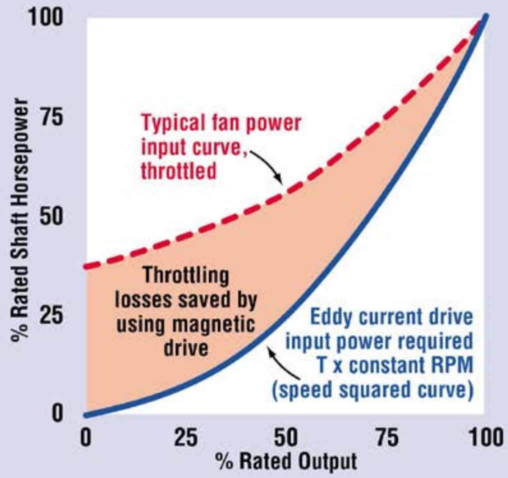


Figure 53 *Power into eddy current drive is product of torque required and constant input rpm. With an unthrottled fan or pump, the torque requirement is nearly a direct function of speed squared, thus, input power varies as the output speed squared.*

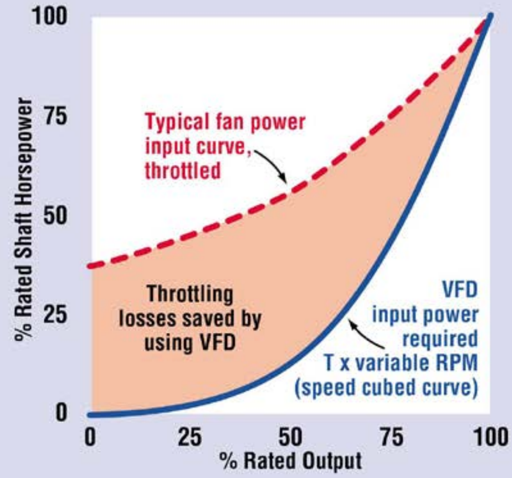


Figure 54 *Power into variable frequency drive is the product of the torque required and the output rpm. With a load as in figure 53, the input varies as the speed cubed.*

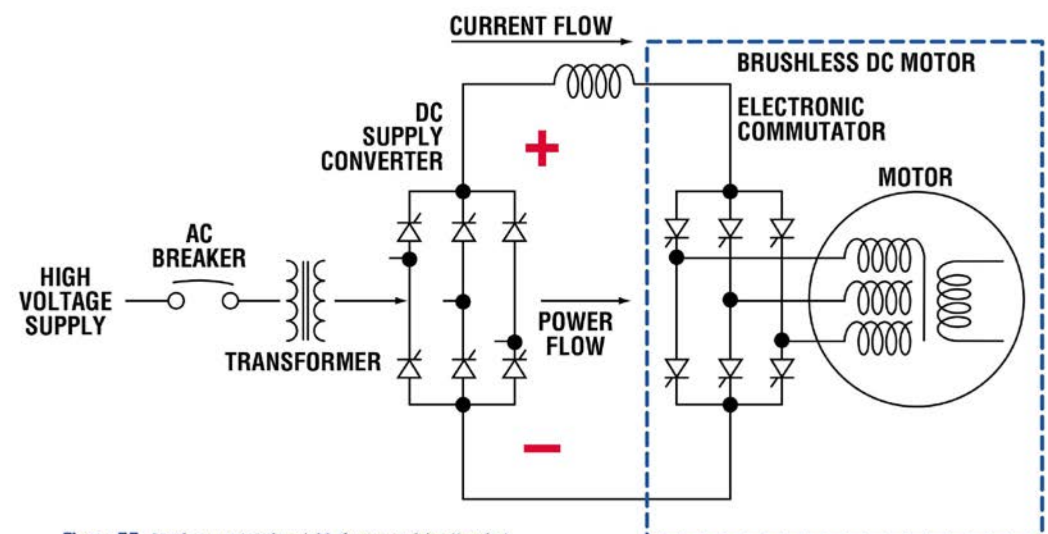


Figure 55 *Load commutated variable frequency drive (6-pulse)*

As with any thyristor circuit, once a thyristor is turned on, the current must be reduced to zero before the thyristor returns to the non-conducting state. With an excited synchronous motor as the load, the commutation is accomplished using the open circuited winding to generate opposing voltages in the machine. (Figures 56 and 57.) To provide reliable operation at low speeds, encoders are sometimes used to indicate the instantaneous position of the rotor to the control so that succeeding thyristors can be turned on at the proper instant. Variations in technologies also exist which use no encoders, but work by sensing the back voltages from the machine. Whichever method is used, a rotating magnetic field is established.

The simplest type of LCI uses a six-step approach. Refer to Figure 59 and note the progression of the resultant vector as dc is applied in sequence and at alternating polarity to each phase of the machine. The vector, representing the magnetic field, progresses around the stator in one complete cycle.

The six-step approach, while the simplest, is not the smoothest. The pulsations associated with the finite progression of magnetic fields around the stator are reflected in torque pulsations and magnetic noise. Care must be exercised to avoid critical frequencies in the motor shaft and other elements which could respond to pulsations set up by the inverter action. In some cases, special couplings must be applied to avoid premature failures. The six-step approach is commonly used for horsepower in the 500-2500 range, sometimes up to 6000HP.

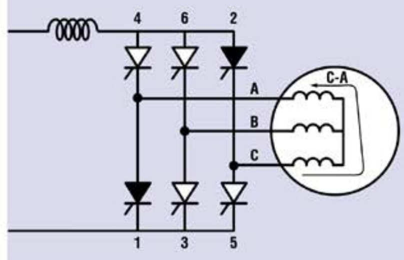


Figure 56 *Load Current Path (C-A)*

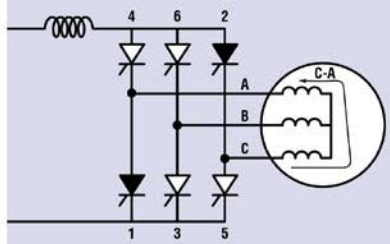
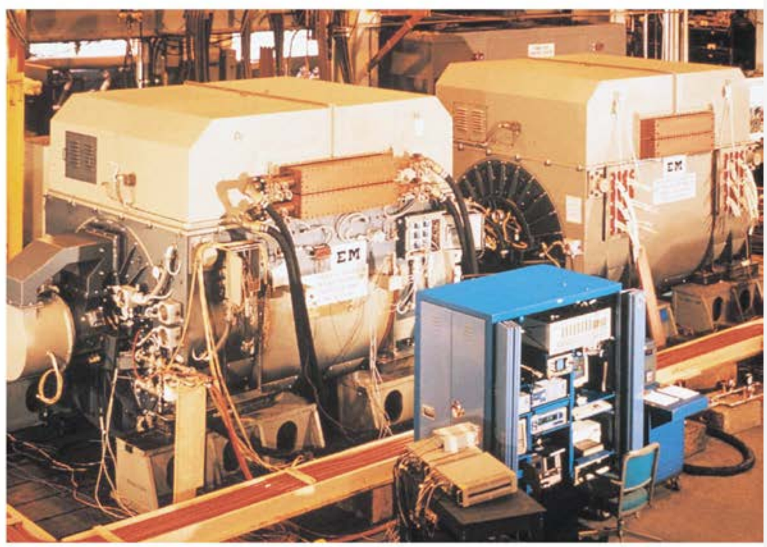


Figure 57 *Load Current Path (C-B)*

A smoother-acting system can be assembled by taking advantage of the 30° difference between the secondary voltages in wye and delta-connected transformers. Using this motor and transformer combination, a twelve-step inverter can be applied.



Figures 58 *Back to back test of 2-14200 HP 3200 RPM 2 pole synchronous motors for VFD application. The shaft extensions are coupled together, and with their respective VFD's one operates as a motor and the other operates as a generator, achieving essentially full load operation of each machine. In this type of test the local utility must only make up the losses of the machines and VFD's.*

Refer to Figure 60. In Figure 61, the progression of the vector can be followed as above, but the steps now are at 30° intervals rather than at 60°, as in the six-step inverter. Another significant advantage is a reduction in harmonics imposed on the incoming line as well as torque harmonics to the load. Figures 62 and 63 further illustrate the differences between six and twelve-step inverter actions. The fifth, seventh, seventeenth, nineteenth, and many higher order harmonics cancel in the wye and delta secondaries; they do not appear in the incoming power to the twelve-step converter. The result is smoother operation, fewer concerns about torsional stresses, quieter running, lower harmonics on the incoming line, and a reduced output requirement from each thyristor bridge.

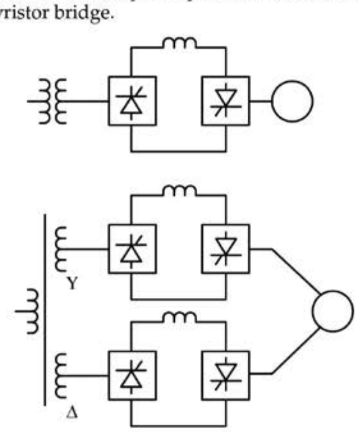


Figure 60 *Top: 6 Pulse Single Winding. Bottom: 12 Pulse Dual Winding.*

While two transformers could be used to feed the twelve-step converter, a three-winding transformer is frequently used (note in Figure 60). Although this could be done with transformers between the inverter and the motor, the motor, rather, is frequently wound with a dual winding to isolate the outputs of the two inverter sections. The two separate windings are shifted 30 electrical degrees. Each part of the motor stator actually sees a six-step progression, but the rotor sees the twelve-step sum of the two inputs to the machine. The twelve step approach is typically used for horsepowers of 2000 and larger, even up to 50,000.

Current-limiting functions are often employed during starting, keeping currents to values considerably under normal locked rotor values. The current limit is adjustable and is set as a function of the amount of torque required. Available starting torque, when operating with an LCI, is more like the maximum operating torque of the motor, since it is

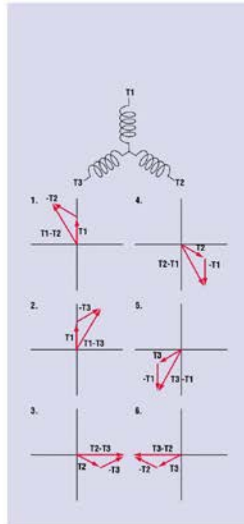


Figure 59 *Single Winding 6 Pulse.*

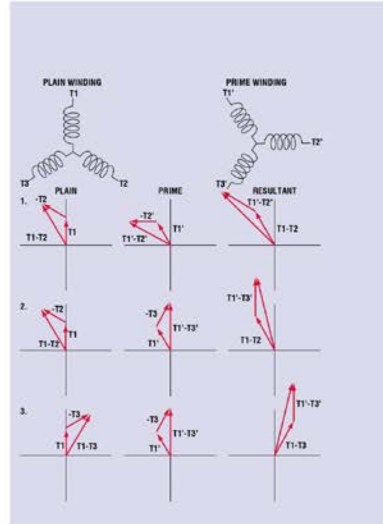
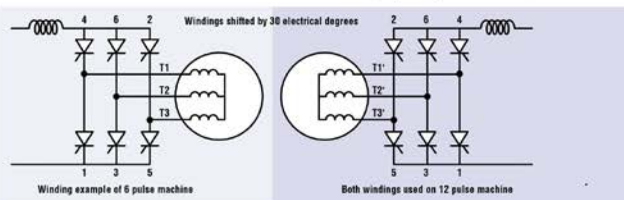
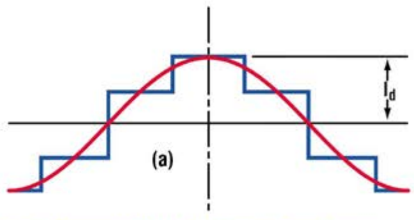


Figure 61 *Dual Winding 12 Pulse. (Only 3 of the 12 pulses are shown)*



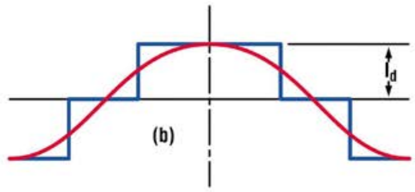
	Degrees	SCR 1	SCR 2	SCR 3	SCR 4	SCR 5	SCR 6	Current In	Current Out	SCR' 1	SCR' 2	SCR' 3	SCR' 4	SCR' 5	SCR' 6	Current In	Current Out	Comment
		Single Winding																
6 Step Single Winding	60			X	X			T1	T2									
	120				X	X		T1	T3									SCR's
	180					X	X	T2	T3									Switch
	240	X					X	T2	T1									Every
	300	X	X					T3	T1									60°
	360		X	X				T3	T2									
12 Pulse Dual Winding	30			X	X			T1	T2			X	X			T1'	T2'	
	60			X	X			T1	T2				X	X		T1'	T3'	
	90				X	X		T1	T3				X	X		T1'	T3'	
	120				X	X		T1	T3					X	X	T2'	T3'	
	150					X	X	T2	T3					X	X	T2'	T3'	SCR's
	180					X	X	T2	T3	X					X	T2'	T1'	Switch
	210	X					X	T2	T1	X					X	T2'	T1'	Every
	240	X					X	T2	T1	X	X					T3'	T1'	30°
	270	X	X					T3	T1	X	X					T3'	T1'	
	300	X	X					T3	T1		X	X				T3'	T2'	
	330		X	X				T3	T2		X	X				T3'	T2'	
	360		X	X				T3	T2			X	X			T1'	T2'	

Figure 62 *SCR firing sequence for 6 pulse and 12 pulse drive.*



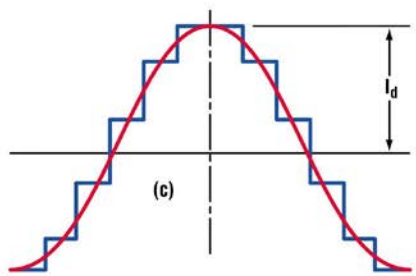
6-Pulse Converter With Delta/WYE Transformer
Fourier Series:

$$i = \frac{2\sqrt{3}}{\pi} I_d \left(\cos\theta + \tfrac{1}{5}\cos 5\theta - \tfrac{1}{7}\cos 7\theta - \tfrac{1}{11}\cos 11\theta + \tfrac{1}{13}\cos 13\theta + \tfrac{1}{17}\cos 17\theta - \tfrac{1}{19}\cos 19\theta \text{ —}\right)$$



6-Pulse Converter With Delta/Delta Transformer
Fourier Series:

$$i = \frac{2\sqrt{3}}{\pi} I_d \left(\cos\theta - \tfrac{1}{5}\cos 5\theta + \tfrac{1}{7}\cos 7\theta - \tfrac{1}{11}\cos 11\theta + \tfrac{1}{13}\cos 13\theta - \tfrac{1}{17}\cos 17\theta + \tfrac{1}{19}\cos 19\theta \text{ —}\right)$$



Current in Delta/Delta and Delta/WYE Transformers
Fourier Series:

$$I_{total} = \frac{2\sqrt{3}}{\pi} I_d \left(\cos\theta - \tfrac{1}{11}\cos 11\theta + \tfrac{1}{13}\cos 13\theta - \tfrac{1}{23}\cos 23\theta + \tfrac{1}{25}\cos 25\theta \text{ —}\right)$$

Figure 63 *Sine wave simulation of VFD output.*

synchronized with the very low frequency source. Greater starting torques may be available at the same time that lower starting currents are demanded when compared with across-the-line starting. Reduced speed operation is controlled at a constant volts per hertz basis so the magnetic structure is not saturated.

Occasionally, a by-pass contactor or circuit breaker will be supplied to connect the line directly to the machine. This will affect the cost of the installation, but it may also be desired by the user as a backup system. It is also possible to start or operate multiple machines while using only one LCI. For example, there may be two pumps; one may be operating via the LCI, but its capacity has been exceeded by the demand. This machine can be transferred to across-the-line operation via the by-pass contactor and the LCI transferred to the second machine, which it can bring up to operating speed, increasing the total capacity in a very smooth and bumpless manner. Refer to Figure 64.

Occasionally, it will be desired that voltage will be increased to rated level at speeds over 80% (of line frequency rating). For this condition, the motor is built with enough iron so that magnetic saturation does not become problematic. The result is that the machine can run without a power factor penalty. Low voltage means delayed firing angles in the front (input) end of the converter, and that translates into a low power factor. Since it is common for large machines (forced draft and induced draft fans) to run at speeds between 80% and 100%, it is desirable to operate at 100% voltage and realize the benefit of the improved power factor (as seen by the line) available from 80 to 100% speed.

While unity or leading power factors are generally associated with synchronous motors, this feature does not hold true when the motor is operated through an LCI. The motor will be operating at a leading power factor, but the line does not see the motor; it sees the input section of the LCI, and that is always a lagging power factor load. The lower the speed (voltage) the more lagging the power factor, and the best power factor is still only about 90% when output voltage to the motor is at maximum.

Cooling the machine may be by use of an integral fan if the load is variable torque and is not expected to operate at speeds below about 50% for extended periods. If a constant torque load is being driven, however, and particularly if the speed will be less than 50% rated, external cooling fans must be employed. Often, the motor will be oversized to lessen the effects of reduced cooling at lower speeds.

Sometimes present in the output supplied to the motor from the LCI are common mode

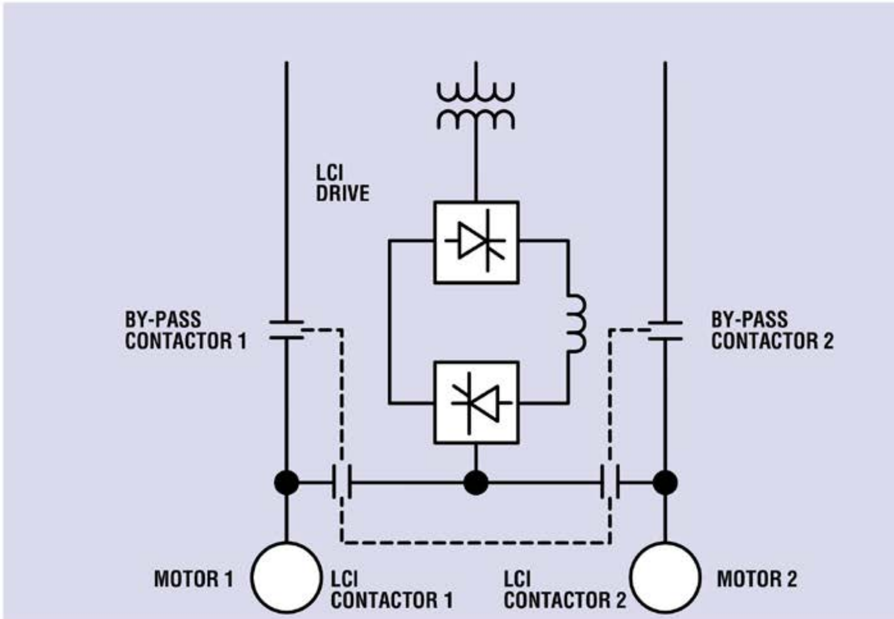


Figure 64
By-Pass and LCI Contractor are interlocked to be mutually exclusive.

voltages which may impose higher-than-usual stresses to ground from the machine winding. Accordingly, it is necessary to provide groundwall insulation with 1.3 to 1.8 times the normal dielectric strength compared to ordinary machines when the unit is to be operated from an inverter.

The exciter design must differ from conventional designs in that conventional exciters extract energy from the rotating shaft, whereas, that source would be such a variable factor – even approaching zero at very low speeds – the energy must be transmitted to the rotor by another means. Exciter designs for these motors use three-phase ac excitation on the stator, applied so the rotating field is counter-rotating to the rotor. In a very real sense, it is a rotating transformer with an air gap in the magnetic circuit, and all of the power developed in the rotor comes from the excitation source. The exciter may or may not have the same number of poles as the motor; however, the frequency in the rotor circuit is not critical, since it is rectified anyhow.

An advantage of the LCI drive which has not been discussed is the ability to run at supersynchronous speed (higher than the synchronous speed of a machine with the same number of poles when connected to the line). If this kind of operation is planned, it must be kept in mind that centrifugal forces are a function of the square of the speed and loads for centrifugal pumps and fans are related to the cube of the speed. Realizing this, it soon becomes apparent that a small increase in speed results in much larger changes in centrifugal stresses and in loading; thus, these expectations must be known by the machine designer so that proper capabilities can be incorporated into the design. Critical frequencies which, conceivably, might otherwise occur at speeds only slightly above normal synchronous speed could also be avoided.

We have seen that the motors used with inverters are much the same as those which are applied only at line voltages and frequencies; the major differences being in two-winding stators for the twelve-step inverter applications, possible heavier ground-wall insulation, cooling, and three-phase exciter stator windings, plus torsional considerations and, possibly, provisions for supersynchronous operation which affect the mechanical facets of the design as well as the electrical. However, these are but refinements to the machine to adapt it to a special application.

Advances in technology, bringing about such equipment as the Load Commutated Inverter, continue to open new doors of opportunity for application of that old, reliable workhorse of industry, the Synchronous Motor.

Synchronous motors fill an important place in industry. They are usually associated with loads of major importance. A corresponding degree of care in their application is well advised and worth the extra effort.

AUTHORS

About the authors

Gerry Oscarson

Now deceased, Gerry Oscarson was the original author of this special issue of the EM Synchronizer. He held Electrical Engineering degrees from the University of Minnesota. Specializing in electric power apparatus, Mr. Oscarson was directly associated with the industrial application of motors, generators, and controls. For many years, he was Chief Application Engineer of the Electric Machinery Mfg. Company. The ABC series of EM Synchronizers are Gerry's work. These remain as standard references for industrial plants around the world, and are often used as texts by engineering schools.

Jack Imbertson

Jack Imberton holds an Electrical Engineering degree from the University of Minnesota. While at EM, he held the positions of Design Engineer, Programmer, Manager of Synchronous Machines, and Manager of Turbo Generators. He has been active in IEEE and was a past chairman of the IAS section. He also served as a member of the ANSI C50 standards committee on synchronous machines.

Since retirement, he has taught electro-mechanics at the University of Minnesota and does some consulting.

Ben Imbertson

Ben Imbertson holds a Bachelors Degree in Electrical Engineering from the University of Minnesota and is a registered Professional Engineer. While at EM, he worked as a Control Design Engineer, Control Engineering Section Head, and Principal Engineer-Control Products.

His EM experience included applications in synchronous and induction motor control, generator control, as well as adjustable-speed drives. His post EM experience has been in similar areas.

Ben's writing has appeared in several EM publications including the Synchronizer, "The ABC's of Motor Control."

Steve Moll

Steve Moll is presently Senior Marketing Representative with Electric Machinery. He holds a Bachelor of Electrical Engineering degree from the University of Minnesota.

His experience at EM includes Senior Design Engineer where he worked with motor and generator controls, as well as high voltage switchgear and controls for gas turbine driven generators. In his present position, his area of specialty is application and marketing of 2-pole turbine driven generators. He has significant experience with synchronous and induction motors as well.

Several of Steve's articles have appeared in trade as well as company publications.

Electric Machinery Company
800 Central Avenue; Minneapolis, MN 55413
Business: (612) 378-8000 Fax: (612) 378-8054
www.electricmachinery.com

LLOYD'S REGISTER QUALITY COMPANY ISO 9001

200-SYN-42