```
; ****************************************************
; BASIC .ASM template file for AVR
; ****************************************************

.include "C:\VMLAB\include\m8def.inc"

.def    temp  =r16
.def    Dato = r18
.def    Gradiente = r19
.def    Cont    = r20
.def    Serial = r21
.def    Pasos = r22

.dseg
.org    $60
aux: .byte 1
.cseg
.org    $0
rjmp    Inicio

.org    $00b            ;salta a la rutina de interrupción
rjmp    rsi_USART       ;cuando llega la información serial
```

```
;*****************************************
;Configuro los puertos de entrada y salida
;*****************************************
Conf_Puertos:
        push    r16

        ldi     r16, $FF            ;configuro como salida el puerto B
        out      ddrb, r16
        ldi     r16, $FF
        out     ddrc, r16

        pop     r16
        ret

;*****************************************
;Subrutina que configura la comunicación
;serial mediante el USART:
;Velocidad normal asíncrona
;Interrupción habilitada por recepción Rx completa
;Recepción y transmisión activadas de 8 bits
;Sin paridad,1 bit de parada y velocidad de 9600
;*****************************************
Conf_USART:
        push    r16

        ldi     r16, high(12)       ;Taza de transmisión de 9600 baudios
        out     ubrrh, r16
        ldi     r16, low(12)
        out     ubrrl, r16
```

```
        ldi     r16, 0b00000010     ;Velocidad normal asíncrono sin multiprocesador
        out     ucsra, r16

        ldi     r16, 0b10011000     ;Interrupción por recepción completa activada
        out     ucsrb, r16          ;Recepción y Transmisión activadas y solo 8 bits

        ldi     r16, 0b10000110     ;Interrupción por recepción completa activada
        out     ucsrc, r16          ;Recepción y Transmisión activadas y solo 8 bits

        pop     r16
        ret

;*****************************************
;Subrutina que configura las interrupciones
;externas por flanco de subida.
;Periodo de la frecuencia
;*****************************************
Conf_Timer1:
        push    r16

        ldi     r16, high(750)      ;Se carga el Duty Cycle
        out     ocr1ah, r16

        ldi     r16, low(750)
        out     ocr1al, r16

        ldi     r16, high(20000)    ;Valor tope que define la frecuencia
        out     icr1h, r16
```

```
        ldi     r16, low(20000)
        out     icr1l, r16

        ldi     r16, 0b10000010     ;Modo PWM rapido no invertido.
        out     tccr1a, r16

        ldi     r16, 0b00011001     ;Sin preescalamiento
        out     tccr1b, r16

        pop     r16
        ret

;*****************************************
;Subrutina que incrementa el valor del Duty
;usando el valor de Gradiente predefinido.
;*****************************************
Mueve_parlante:
        push    r16
        push    r17

        in      r17, ocr1ah         ;Extraigo el valor del OCR1A
        in      r16, ocr1al
        add     r16, Gradiente      ;Incremento la parte baja
        sts     aux, Gradiente
        clr     Gradiente
        adc     r17, Gradiente      ;Incremento con carry

        out     ocr1ah, r17         ;Y lo cargo en OCR1A el nuevo valor
        out     ocr1al, r16
```

```
        lds     Gradiente, aux          ;Recupero el valor de Gradiente
        pop     r17
        pop     r16
        ret


;***********************************************************
;Subrutina de Interrupción que setea una bandera de indicador.
;***********************************************************
rsi_USART:
        push    r16
        in      r16, sreg
        push    r16

        ldi     Serial, 1

        pop     r16
        out     sreg, r16
        pop     r16
        reti

Envia_dato:
        sbis    ucsra, udre
        rjmp    Envia_dato
        out     udr, Dato
        ret
```

```
Recibe_dato:
        sbis    ucsra, rxc
        rjmp    Recibe_dato
        in      Dato, udr
        ret


;*****************************************************
;******************Programa Principal*****************
;*****************************************************
Inicio:
        ldi     r16, high(ramend)       ;Inicializo el stack pointer
        out     sph, r16
        ldi     r16, low(ramend)
        out     spl, r16

        rcall   Conf_Puertos            ;Configuro los puertos
        rcall   Conf_Timer1             ;Configuro el timer 1
        rcall   Conf_USART              ;Configuro el USART

        ldi     Dato, 0                 ;Inicializo las variables
        ldi     Cont, 0
        ldi     Serial, 0

        sbi     portc, 0
        sei                             ;Habilito las interrupciones
```

```
Conf_movimiento:
                                    ;Configuro la gradiente
        cpi     Serial, 1           ;Espero a que llegue el dato
        brne    Conf_movimiento

        ldi     Serial, 0
        rcall   Recibe_dato         ;Recibo el Dato

        cpi     dato, '1'           ;Si es 1, configuro con 5º
        breq    Cinco_grados

        cpi     dato, '2'           ;Si es 2, configuro con 10º
        breq    Diez_grados

        cpi     dato, '3'           ;Si es 3, configuro con 15º
        breq    Quince_grados

        cpi     dato, '4'           ;Si es 4, configuro con 20º
        breq    Veinte_grados

        rjmp    Fin                 ;Sino es error y se va al fin

Cinco_grados:
        ldi     r16, 50             ;Configuro el valor de la gradiente
        mov     Gradiente, r16

        ldi     r16, 36             ;Y la cantidad de pasos a dar
        mov     Pasos, r16
        rjmp    Recibe
```

```
Diez_grados:
        ldi     r16, 100                ;Configuro el valor de la gradiente
        mov     Gradiente, r16

        ldi     r16, 18                 ;Y la cantidad de pasos a dar
        mov     Pasos, r16
        rjmp    Recibe

Quince_grados:
        ldi     r16, 150                ;Configuro el valor de la gradiente
        mov     Gradiente, r16

        ldi     r16, 12                 ;Y la cantidad de pasos a dar
        mov     Pasos, r16
        rjmp    Recibe

Veinte_grados:
        ldi     r16, 200                ;Configuro el valor de la gradiente
        mov     Gradiente, r16

        ldi     r16, 9                  ;Y la cantidad de pasos a dar
        mov     Pasos, r16
        rjmp    Recibe

Recibe:
        cpi     Serial, 1               ;Espero a que llegue el dato
        brne    Recibe
        ldi     Serial, 0
        rcall   Recibe_dato             ;Recibo el Dato
```

```
        cpi     Dato, 'M'           ;Verifico que sea la letra M
        breq    Mueve               ;Si es así se va a mover el parlante

Responde:
        ldi     r16, 'N'            ;Sino, envía la letra N via serial
        mov     Dato, r16           ;y vuelve a esperar un caracter
        rcall   Envia_dato
        rjmp    Recibe

Mueve:
        rcall   Mueve_parlante      ;Aumentamos el Duty Cycle, movimiento parlante
        inc     Cont                ;Incrementamos el contador

        cp      Cont, Pasos         ;Verifico cantidad de pasos configurados
        brlo    Recibe              ;si aún no, regresa a esperar otro ingreso

        rcall   Mueve_parlante      ;Aumentamos el Duty Cycle movimiento parlante

        ldi     r16, 'Y'
        mov     Dato, r16
        rcall   Envia_dato          ;Envío confirmación de movimiento con la letra Y
        cli

        ldi     r16, 0b10000000     ;Interrupción por recepción completa activada
        out     ucsrb, r16
Fin:
        rjmp Fin
```

```
function varargout = Interfaz1_tesis(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Interfaz1_tesis_OpeningFcn, ...
                   'gui_OutputFcn',  @Interfaz1_tesis_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
end

function Interfaz1_tesis_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
end

function varargout = Interfaz1_tesis_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
end
```

```
%% --- Executes on button press in start_button.
function start_button_Callback(hObject, eventdata, handles)

%Genero la matriz donde se almacenaran las mediciones
 if handles.Pasos==36
    handles.medicion=zeros(801,73)
 elseif handles.Pasos==18
    handles.medicion=zeros(801,37)
 elseif handles.Pasos==12
    handles.medicion=zeros(801,25)
 else handles.medicion=zeros(801,19)
 guidata(hObject,handles);
 end

%Obtengo el analizador FFT
 handles.htemplates=get(handles.mo,'Templates','Working');
 handles.hsetup=get(handles.htemplates,'Setup');
 handles.mfft=get(handles.hsetup,'Instruments','FFT Analyzer');
 guidata(hObject,handles);

 set(handles.mfft,'Enabled','True');
 guidata(hObject,handles);

%Envio la orden para que inicie la medición
 invoke(handles.htemplates,'ActivateTemplate');
 guidata(hObject,handles);
 pause(5)
 invoke(handles.Servidor,'Start')
 pause(15)
invoke(handles.Servidor,'Stop')
 pause(5)
```

```
handles.fg=get(handles.fo,'FunctionGroups','FFT Analyzer');
handles.output=get(handles.fg,'Functions','Frequency Response H1(micro,Generador) -
Input');
pause(2)
guidata(hObject,handles);

handles.spectro=get(handles.output,'FunctionData');
guidata(hObject,handles);

%Obtengo el vector de magnitudes
handles.imaginario=get(handles.spectro,'GetAllValues',0); %Parte imaginaria
guidata(hObject,handles);

handles.real=get(handles.spectro,'GetAllValues',1); %Parte real
guidata(hObject,handles);

%Calculamos el modulo en Pa
handles.magnitud=abs(complex(handles.real,handles.imaginario));

%Calculamos su nivel RMS
handles.magnitud=handles.magnitud*0.707;

%Calculamos el nivel de presion sonora Lp(Decibeles)
handles.magnitud=20*log((handles.magnitud*1000000)/20);
guidata(hObject,handles);

%Almaceno en la primera columna de la matriz de medicion la matriz de magnitudes
handles.medicion(:,1)=handles.magnitud;
handles.medicion(:,(2*handles.Pasos)+1)=handles.magnitud;
guidata(hObject,handles);
```

```
%Giramos y medimos el parlante tantas veces como nos lo indique la
%variable pasos
 contador=0
 pasos=handles.Pasos;

%****************************************************************
%****************************************************************
 while contador~=pasos

%Realizo el movimiento del parlante
    datos='M'
    s=handles.S;
    senddata(datos,s);
    pause(5)

%Realizo medicion con el Pulse
    invoke(handles.htemplates,'ActivateTemplate');
    pause(5)
    invoke(handles.Servidor,'Start')
    pause(15)
    invoke(handles.Servidor,'Stop')

    pause(5)
%Obtengo la magnitud de la data y la almaceno en la siguiente columna
    handles.fg=get(handles.fo,'FunctionGroups','FFT Analyzer');
    handles.output=get(handles.fg,'Functions','Frequency Response H1(micro,Generador)
- Input');
    pause(2)
    guidata(hObject,handles);

    handles.spectro=get(handles.output,'FunctionData');
    guidata(hObject,handles);
```

```
    handles.real=get(handles.spectro,'GetAllValues',1); %Parte real
    guidata(hObject,handles);

    handles.imaginario=get(handles.spectro,'GetAllValues',0); %Parte imaginaria
    guidata(hObject,handles);

%Calculamos el modulo y lo normalizamos a su valor RMS
    handles.magnitud=abs(complex(handles.real,handles.imaginario));
    handles.magnitud=handles.magnitud*0.707;
    handles.magnitud=20*log((handles.magnitud*1000000)/20);
    handles.medicion(:,contador+2)=handles.magnitud;
    handles.medicion(:,(handles.Pasos*2)-contador)=handles.magnitud;
    guidata(hObject,handles);

%Limpiamos las variables
    clear handles.magnitud
    clear handles.real
    clear handles.imaginario
    guidata(hObject,handles);
%Incrementa el contador y retoma el bucle
    contador=contador+1
 end
%****************************************************************
%****************************************************************

%Creamos el vector 'Angulos'
if handles.Pasos==36
   handles.angulos=[-179*pi/180:5*pi/180:181*pi/180]
elseif handles.Pasos==18
   handles.angulos=[-177*pi/180:10*pi/180:183*pi/180]
elseif handles.Pasos==12
   handles.angulos=[-175*pi/180:15*pi/180:185*pi/180]
else handles.angulos=[-172*pi/180:20*pi/180:188*pi/180]
end
```

```
medidas=handles.medicion;
save medidas;
guidata(hObject,handles);

end

%% --- Executes on button press in pause_button.
function pause_button_Callback(hObject, eventdata, handles)
handles.pausa=handles.pausa+1;

while handles.pausa~=2
end

handles.pausa=0;

guidata(hObject,handles);
end

%% --- Executes on button press in stop_button.
function stop_button_Callback(hObject, eventdata, handles)
opc=questdlg('¿Desea cancelar la medición?','Salir','Si','No','No')
if strcmp(opc,'No')
    return;
end

%Cerramos el Pulse
invoke(handles.Servidor,'Stop')
invoke(handles.Servidor,'CloseProject');
invoke(handles.Servidor,'Exit');
end
```

```
%% --- Executes on button press in exit_button.
function exit_button_Callback(hObject, eventdata, handles)
opc=questdlg('¿Desea salir del programa?','Salir','Si','No','No')
if strcmp(opc,'No')
    return;
end

%Cerramos el Pulse
invoke(handles.Servidor,'CloseProject');
invoke(handles.Servidor,'Exit');

%Cerramos interfaz
close(Interfaz1_tesis)
close all
clear all
end

%% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)

%Obtengo el indice de la frecuencia
hertz=get(handles.popupmenu1,'value');
hertz=hertz-1;
hertz=hertz*32;
hertz=hertz+1;
axes(handles.axes3);

%Obtengo el vector de radios
handles.radios=handles.medicion(hertz,:);
handles.maximo=max(handles.radios);
handles.radios=handles.radios/handles.maximo;
guidata(hObject,handles);
```

```
%Grafico el patron de radiacion a la frecuencia solicitada
polar(handles.angulos,handles.radios);
end

%% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

%% --- Executes on button press in cinco_grados.
function cinco_grados_Callback(hObject, eventdata, handles)

%Deshabilito los demas radio buttons
set(handles.diez_grados,'value',0)
set(handles.quince_grados,'value',0)
set(handles.veinte_grados,'value',0)
guidata(hObject,handles);

%Cargo 'datos' con '1' y envio al uC para confirgurar pasos.
datos='1';
s=handles.S;
senddata(datos,s);
```

```
%Cargo el numero de pasos correspondientes
handles.Pasos=36;
guidata(hObject,handles);
end

%% --- Executes on button press in diez_grados.
function diez_grados_Callback(hObject, eventdata, handles)

%Deshabilito los demas radio buttons
set(handles.cinco_grados,'value',0)
set(handles.quince_grados,'value',0)
set(handles.veinte_grados,'value',0)

%Cargo 'datos' con '2' y envio al uC para confirgurar pasos.
datos='2';
s=handles.S;
senddata(datos,s);

%Cargo el numero de pasos correspondientes
handles.Pasos=18;
guidata(hObject,handles);
end

%% --- Executes on button press in quince_grados.
function quince_grados_Callback(hObject, eventdata, handles)

%Deshabilito los demas radio buttons
set(handles.cinco_grados,'value',0)
set(handles.diez_grados,'value',0)
set(handles.veinte_grados,'value',0)
```

```
%Cargo 'datos' con '3' y envio al uC para confirgurar pasos.
datos='3';
s=handles.S;
senddata(datos,s);

%Cargo el numero de pasos correspondientes
handles.Pasos=12;
guidata(hObject,handles);
end

%% --- Executes on button press in veinte_grados.
function veinte_grados_Callback(hObject, eventdata, handles)

%Deshabilito los demas radio buttons
set(handles.cinco_grados,'value',0)
set(handles.diez_grados,'value',0)
set(handles.quince_grados,'value',0)

%Cargo 'datos' con '4' y envio al uC para confirgurar pasos.
datos='4';
s=handles.S;
senddata(datos,s);

%Cargo el numero de pasos correspondientes
handles.Pasos=9;
guidata(hObject,handles);
end
```

```
%% --- Executes on button press in conect_button.
function conect_button_Callback(hObject, eventdata, handles)

%*****************************************************************
% Defino el puerto de entrada para la comunicacion con el circuito
%*****************************************************************
puerto=get(handles.port,'String');
handles.S=serial(['COM',puerto]);
guidata(hObject,handles);

%*****************************************************************
% Definición de Pulse como objeto servidor y Matlab como cliente
%*****************************************************************
handles.Servidor=actxserver('PULSE.LabShop.Application')
set(handles.Servidor,'visible',1);
guidata(hObject,handles);

% Accedo al proyecto de pulse previamente creado
strfile='C:\Users\angelo.velarde\Desktop\Tesis Samuel\Tesis Samuel.pls'
myproject=invoke(handles.Servidor,'OpenProject',strfile,0,0)
pause(10)
guidata(hObject,handles);

%Obtengo el Organizador de Medida y el Funciones
handles.mo=get(myproject,'MeasurementOrganiser')
handles.fo=get(myproject,'FunctionOrganiser')
guidata(hObject,handles);
end
```

```
%% --- Executes during object creation, after setting all properties.
function axes2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: place code in OpeningFcn to populate axes2
end

%%
function port_Callback(hObject, eventdata, handles)
% hObject    handle to port (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of port as text
%        str2double(get(hObject,'String')) returns contents of port as a double
end

%% --- Executes during object creation, after setting all properties.
function port_CreateFcn(hObject, eventdata, handles)
% hObject    handle to port (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end
```

```
%% --- Executes during object creation, after setting all properties.
function axes3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes3
end

%% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
opc=questdlg('¿Desea salir del programa?','Salir','Si','No','No')
if strcmp(opc,'No')
    return;
end
delete(hObject);
end

%% --- Executes on selection change in popupmenu3.
function popupmenu3_Callback(hObject, eventdata, handles)

%Obtengo el indice del angulo
pos=get(handles.popupmenu3,'value');
pos=pos-1;
axes(handles.axes2);

if handles.Pasos==36
   pos=pos*2+(pos-2);
% elseif handles.Pasos==18
%    handles.medicion=zeros(801,36)
elseif handles.Pasos==12
   pos=pos;
end
```

```
%Obtengo el vector de magnitudes
handles.magnitud=handles.medicion(:,pos);
guidata(hObject,handles);

%Graficamos en el espacio correspondiente
frecuencia=xlsread('I:\Tesis\frecuencias.xls');
axes(handles.axes2);
grid on;
semilogx(frecuencia,handles.magnitud)
end

%% --- Executes during object creation, after setting all properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end


function senddata(datos,s)
datos = num2str(datos);
trama=[datos];
t=whos('trama');
set(s,'outputBufferSize',t.bytes);
set(s,'BaudRate',9600);
fopen(s);
fprintf(s,'%s',trama)
fclose(s);
```