ANEXO

1 Lazo.m

El siguiente fichero corresponde al programa realizado en Matlab para calcular la mitigación en un punto en función del campo magnético generado por una línea monofásica.

Calcula la amplitud en amperios (absIl) y desfase en grados (angIl) de la intensidad a inyectar al lazo activo

El desfase esta referido a la componente vertical del campo magnético (en µT) generado por la línea bifilar (Bfy) en (xp,yp), punto de medida del campo

Definición de intensidad positiva (por la linea o lazo): la que genera un campo magnético cuya dirección es hacia arriba.

IMPORTANTE: el punto de medida no tiene por que coincidir con el punto de mitigación

Punto de medida: (xp,yp)

Punto de mitigación: (xc,yc)

Parámetros fijos

```
hf=0.59; %altura conductores línea
hl=0.2; %altura conductores lazo
df=0.72; %separación conductores línea
dl=0.67; %separación conductores lazo
xp=-0.9;yp=0.53; %posición sensor de campo
xc=-0.5;yc=0.03; %posición punto de mitigación
```

Calculamos las posiciones de los conductores de línea y lazo

```
xf1=-df*0.5;
yf1=hf;
xf2=df*0.5;
yf2=hf;
xl1=-dl*0.5;
yl1=hl;
xl2=dl*0.5;
yl2=hl;
```

Suponemos una medida en el sensor de campo

```
Bfy=1.8; %amplitud en microteslas
```

Obtenemos la amplitud de la intensidad por la linea absIf que corresponde a dicha amplitud de campo

aux1=(xp-xf1)^2+(yp-yf1)^2;
aux2=(xp-xf2)^2+(yp-yf2)^2;
aux=(xp-xf1)/aux1-(xp-xf2)/aux2;
*absIf=abs(Bfy/(0.2*aux))*
*Bfx=0.2*absIf*((yf1-yp)/aux1-(yf2-yp)/aux2);*

Calculamos la amplitud de la corriente por el lazo

aux3=(xc-xl1)^2+(yc-yl1)^2;
aux4=(xc-xl2)^2+(yc-yl2)^2;
blx=0.2((yl1-yc)/aux3-(yl2-yc)/aux4);*
bly=0.2((xc-xl1)/aux3-(xc-xl2)/aux4);*
*absIl=abs((-blx*Bfx-bly*Bfy)/(blx^2+bly^2))*

Calculo factor de apantallado

*Bfix=Bfx+blx*absIl;*
*Bfiy=Bfy+bly*absIl;*
sf=sqrt(Bfx^2+Bfy^2)/sqrt(Bfix^2+Bfiy^2);

En este caso para el calculo factor de apantallado suponiendo un desfase distinto a 180º en la intensidad del lazo.

*dd=6*pi/180; %diferencia de desfase*
*Bfixd=Bfx+abs(blx)*absIl*(cos(pi-dd)+sin(pi-dd)*1i);*
*Bfiyd=Bfy+abs(bly)*absIl*(cos(pi-dd)+sin(pi-dd)*1i);*
sfd=sqrt(Bfx^2+Bfy^2)/sqrt(abs(Bfixd)^2+abs(Bfiyd)^2);
*error=abs((sf-sfd)/sf)*100*

2 Seguidor Onda Ref

Este programa crea una copia de una señal de referencia introducida por el canal de entrada del DSP.

```
/*******************************************************/

#include "regs243.h"

/*****************  SETUP del registro OCRA  *************/
#define OCRA15                  0     /* 0:IOPB7   1:TCLKIN       */
#define OCRA14                  0     /* 0:IOPB6   1:TDIR   */
#define OCRA13                  0     /* 0:IOPB5   1:T2PWM        */
#define OCRA12                  0     /* 0:IOPB4   1:T1PWM        */
#define OCRA11                  0     /* 0:IOPB3   1:PWM6   */
#define OCRA10                  0     /* 0:IOPB2   1:PWM5   */
#define OCRA9                   0     /* 0:IOPB1   1:PWM4   */
#define OCRA8                   0     /* 0:IOPB0   1:PWM3   */
#define OCRA7                   0     /* 0:IOPA7   1:PWM2   */
```

```
#define OCRA6                   0      /* 0:IOPA6   1:PWM1   */
#define OCRA5                   0      /* 0:IOPA5   1:CAP3   */
#define OCRA4                   0      /* 0:IOPA4   1:CAP2/QEP2 */
#define OCRA3                   0      /* 0:IOPA3   1:CAP1/QEP1 */
#define OCRA2                   0      /* 0:IOPA2   1:XINT1        */
#define OCRA1                   0      /* 0:IOPA1   1:SCIRXD       */
#define OCRA0                   0      /* 0:IOPA0   1:SCITXD       */
/*********************************************************/

/*************** SETUP del registro OCRB  ***************/
#define OCRB9                   0      /* 0:IOPD1  1:XINT2/EXTSOC */
#define OCRB8                   1      /* 0:CKLKOUT 1:IOPD0        */
#define OCRB7                   0      /* 0:IOPC7   1:CANRX        */
#define OCRB6                   0      /* 0:IOPC6   1:CANTX        */
#define OCRB5                   0      /* 0:IOPC5   1:SPISTE       */
#define OCRB4                   0      /* 0:IOPC4   1:SPICLK       */
#define OCRB3                   0      /* 0:IOPC3   1:SPISOMI      */
#define OCRB2                   0      /* 0:IOPC2   1:SPISIMO      */
#define OCRB1                   1      /* 0:BIO           1:IOPC1        */
#define OCRB0                   0      /* 0:XF            1:IOPC0        */
/*********************************************************/

/******************    SETUP el WDCR    *****************/
#define WDDIS                   1      /* 0:Watchdog enabled 1:disabled*/
#define WDCHK2                  1      /* 0:System reset  1:Normal OP  */
#define WDCHK1                  0      /* 0:Normal Oper. 1: sys reset  */
#define WDCHK0                  1      /* 0:System reset 1:Normal OP   */
#define WDSP             7      /* Watchdog prescaler 7 : div 64*/
/*********************************************************/

/*************     SETUP del registro SCSR  **************/
#define CLKSRC                  0      /* 0:interno(20MHz)         */
#define LPM              0      /* 0:Modo bajo consume 0 si IDLE*/
#define ILLADR                  1      /* 1:borrar ILLADR          */
/*********************************************************/

/*************     SETUP de WSGR ************************/
#define BVIS       0  /* 10-9:00 Bus visibility OFF       */
#define ISWS       0  /* 8-6:000 0 estados de espera para I/O*/
#define DSWS       0  /* 5-3:000 0 estados de esp. para datos */
#define PSWS       0  /* 2-0:000 0 estados de esp. para código*/
/*********************************************************/

/*************   SETUP del registro GPTCON ***************/
#define GPTCON_T2TOADC    0
  /* 10-9  : T2TOADC = 00 : GPT2 no inicia ADC            */
#define GPTCON_T1TOADC    2
  /*  8-7  : T1TOADC = 10 : GPT1 inicia ACD por periodo */
#define GPTCON_TCOMPOE  0
   /*  6   : TCOMPOE = 0  : deshabilitadas las 2 salidas de comparación del
   GPT */
#define GPTCON_T2PIN      0
   /*  3-2  : T2PIN = 00 : Pol. De salida GPT2=forzada nivel bajo
       */
#define GPTCON_T1PIN      0
   /*  1-0  : T1PIN = 00 : Pol. De salida GPT1=forzada nivel bajo
       */
/*********************************************************/
```

```
/**************  SETUP de registro T1CON *****************/
#define T1CON_FREESOFT    0
   /* 15-14 FREE, SOFT : 00 stop en emulación JTAG suspendido*/
#define T1CON_TMODE       2
   /* 12-11 : TMODE1,0 : 10 Modo Continuous up en conteo */
#define T1CON_TPS         0
   /* 10-8  : TPS2-0   : 000 prescaler del reloj de entrada CPUCLK/1 */
#define T1CON_TENABLE     1
  /* 6     : TENABLE    : 1 Habilita GPT1                   */
#define T1CON_TCLKS       0
  /* 5-4   : TCLKS1,0   : 00 Fuente de reloj:interno        */
#define T1CON_TCLD        1
   /* 3-2   : TCLD1,0    : 01 Recarga el timer 1 al llegar a 0 o a T */
#define T1CON_TECMPR      0
   /* 1     :  TECMPR     : 0 deshabilita operación de comparación del timer
   */
/*********************************************************/

/*************     SETUP del registro EVIMRA *************/
#define T1OFINT     0  /* 10 : Timer 1 overflow interrupt         */
#define T1UFINT     0  /*  9 : Timer 1 underflow interrupt        */
#define T1CINT            0  /*  8 : Timer 1 compare interrupt         */
#define T1PINT            0  /*  7 : Timer 1 period interrupt    */
#define CMP3INT     0  /*  3 : Compare 3 interrupt                */
#define CMP2INT     0  /*  2 : Compare 2 interrupt                */
#define CMP1INT     0  /*  1 : Compare 1 interrupt                */
#define PDPINT            0  /*  0 : Power Drive Protect Interrupt*/
/*********************************************************/

/*************     SETUP del registro EVIMRB *************/
#define T2OFINT     0  /*  3 : Timer 2 overflow interrupt         */
#define T2UFINT     0  /*  2 : Timer 2 underflow interrupt  */
#define T2CINT            0  /*  1 : Timer 2 compare interrupt   */
#define T2PINT            0  /*  0 : Timer 2 period interrupt    */
/*********************************************************/

/*************     SETUP del registro EVIMRC *************/
#define CAP3INT     0  /*  2 : Capture Unit 3 interrupt    */
#define CAP2INT     0  /*  1 : Capture Unit 2 Interrupt    */
#define CAP1INT     0  /*  0 : Capture unit 1 interrupt    */
/*********************************************************/

/*************     SETUP del IMR    **********************/
#define INT6        1  /*  5 : Level INT6  no emascarada */
#define INT5        0  /*  4 : Level INT5  enmascarada   */
#define INT4        0  /*  3 : Level INT4  enmascarada   */
#define INT3        0  /*  2 : Level INT3  enmascarada   */
#define INT2        0  /*  1 : Level INT2  enmascarada   */
#define INT1        0  /*  0 : Level INT1  enmascarada   */
/*********************************************************/

/*************        SETUP del registro ADCCTRL1 ************/
#define SOFTFREE    2  /* 15-14 : 10 completa conversión antes de parar */
#define ADCIMSTART  0  /* 13 :  no comienza conversión inmediatamente */
#define ADC2EN            1  /* 12 :  unidad ADC2 habilitada       */
#define ADC1EN            1  /* 11 :  unidad ADC1 habilitada       */
#define ADCCONRUN   0  /* 10 :  conversión en modo no continuo          */
#define ADCINTEN    1  /* 9  :  habilita interrupción del ADC    */
#define ADCINTFLAG  1  /* 8  :  borra interrupción del ADC */
```

```
#define ADC2CHSEL   0  /* 6-4:  000 canal 0 para la unidad ADC2  */
#define ADC1CHSEL   1  /* 3-1:  001 canal 1 para la unidad ADC1  */
#define ADCSOC            1  /* 0  :  Inicio de conversión    */
/*********************************************************/

/*************        SETUP del registro ADCCTRL2 ************/
#define ADCIM             1  /* 14 : flag de interrupción del ACD */
#define ADCINTPRI   1  /* 11 :prioridad de interrupción de ADC nivel 6  */
#define ADCEVSOC    1  /* 10 : habilitado inicio ADC por gestor de Eventos */
#define ADCEXTSOC   0  /*  9 :deshabilitado inicio externo del ADC */
#define ADCPSCALE   4  /* 2-0: 100 prescaler CLKOUT/12     */
/*********************************************************/

/*** Build options ***/

#define PERIOD 500 /* 500--->40Khz     T1 PERIOD = 50ns * 1 * 500 = 0.000025s
(40Khz)*/

unsigned int ADC0_result,ADC1_result;

signed int ref,io;
signed int err;
unsigned int y;

interrupt void ADC_ISR(void);

void c_dummy1(void);
extern _out_wsgr(); /*declaración de función externa para
       configurar WSGR */

void c_dummy1(void)
{
      while(1);          /*función para atrapar interrupciones
             espúreas */
}

interrupt void ADC_ISR(void) /*rutina de interrupción*/
{

 if((PIVR-0x0004)==0)     /*Verifica la interrupción ( 4 =
                  ADC )  */
      {
      ADCTRL1 |= 0x0100;
      ADC0_result=ADCFIFO2>>6;          /* Intensidad*/
 /*   asm (" LACL 7036h");     /* Vacío las FIFO */
 /*   asm (" LACL 7036h");  */
      ADC1_result=ADCFIFO1>>6;          /* Referencia */

 /*   asm (" LACL 7038h"); */
 /*   asm (" LACL 7038h");  */


      }
}

void main(void)
{
 asm ("  clrc XF");/*Pone 0 en la salida XF */
```

```
asm (" setc INTM"); /*Deshabilita todas las interrupciones
           */
asm (" clrc SXM");/*Borra el bit de extensión de signo*/
asm (" clrc OVM");/*Borra bit de modo de desbordamiento*/
asm (" clrc CNF");/*Configura B0 como memoria de datos */

WDCR=((WDDIS<<6)+(WDCHK2<<5)+(WDCHK1<<4)+(WDCHK0<<3)+WDSP);
                              /*Inicializa el registro WDCR    */

SCSR = ((CLKSRC<<14)+(LPM<<12)+ILLADR); /* Inicializa SCSR
     */

out_wsgr((BVIS<<9)+(ISWS<<6)+(DSWS<<3)+PSWS);
                 /* Función externa para configurar WSGR        */

OCRB = ((OCRB9<<9)+(OCRB8<<8)+
             (OCRB7<<7)+(OCRB6<<6)+(OCRB5<<5)+(OCRB4<<4)+
             (OCRB3<<3)+(OCRB2<<2)+(OCRB1<<1)+OCRB0);
                              /* Inicializa  OCRB       */


OCRA = ((OCRA15<<15)+(OCRA14<<14)+(OCRA13<<13)+(OCRA12<<12)+
         (OCRA11<<11)+(OCRA10<<10)+(OCRA9<<9)+(OCRA8<<8)+
         (OCRA7<<7)+(OCRA6<<6)+(OCRA5<<5)+(OCRA4<<4)+
         (OCRA3<<3)+(OCRA2<<2)+(OCRA1<<1)+OCRA0);
                       /* Inicializa OCRA        */

GPTCON=((GPTCON_T2TOADC<<9)+
         (GPTCON_T1TOADC<<7)+
         (GPTCON_TCOMPOE<<6)+
         (GPTCON_T2PIN<<2)+
         (GPTCON_T1PIN)); /*   Inicia el registro de control
                   del temporizador  */


PBDATDIR = 0x0000;    /*configure IOPB..B7 como entradas*/

T1PR = PERIOD;              /*configura el periodo del
     temporizador T1    */
T1CNT= 0x0000;              /* valor inicial del contador
     */

T1CON=((T1CON_FREESOFT<<14)+
          (T1CON_TMODE<<11)+
          (T1CON_TPS<<8)+
          (T1CON_TCLKS<<4)+
          (T1CON_TCLD<<2)+
          (T1CON_TECMPR<<1)); /*Configura el registro de
                        control del temporizador 1*/


ADCTRL1= ((SOFTFREE<<14)+(ADCIMSTART<<13)+(ADC2EN<<12)+
            (ADC1EN<<11) +(ADCCONRUN<<10) +(ADCINTEN<<9)+
            (ADCINTFLAG<<8)+(ADC2CHSEL<<4)+(ADC1CHSEL<<1));
                  /* Configura registro de control 1 de ADC*/

ADCTRL2= ((ADCIM<<14)+(ADCINTPRI<<11)+
          (ADCEVSOC<<10)+(ADCEXTSOC<<9)+ADCPSCALE);
                  /*Configura registro de control 2 de ADC*/
```

```
 asm (" LACL 7036h");     /* Vacío las FIFO */
 asm (" LACL 7036h");
 asm (" LACL 7038h");
 asm (" LACL 7038h");

 EVIFRA=0xFFFF;      /* Registro de flags del grupo A     de
                   EV*/

 EVIFRB=0xFFFF;      /* Registro de flags del grupo B     de
                   EV*/

 EVIFRC=0xFFFF;     /* Registro de flags del grupo C      de
                   EV*/


 EVIMRA=((T1OFINT<<10)+
            (T1UFINT<<9)+
            (T1CINT<<8)+
            (T1PINT<<7)+
            (CMP3INT<<3)+
            (CMP2INT<<2)+
            (CMP1INT<<1)+
            (PDPINT)); /*Registro de máscara del grupo A del
             EV*/

 EVIMRB=((T2OFINT<<3)+
          (T2UFINT<<2)+
          (T2CINT<<1)+
          (T2PINT));  /*Registro de máscara del grupo B del
             EV*/

 EVIMRC=((CAP3INT<<2)+
              (CAP2INT<<1)+
              (CAP1INT)); /*Registro de máscara del grupo C
                  del EV*/

 IFR=0xFFFF; /*Borra todas las interrupciones pendientes */

 IMR=((INT6<<5)+
       (INT5<<4)+
       (INT4<<3)+
       (INT3<<2)+
       (INT2<<1)+
       (INT1));      /*Registro de máscaras de interrupción
      */



 asm (" clrc INTM");  /*Habilita todas las interrupciones
             */

 T1CON=T1CON+(T1CON_TENABLE<<6); /*Habilita el GPT1 */

while(1){

       io= (ADC0_result-565)*12;
         ref= (ADC1_result-555)*49;
         err= (io - ref);
```

```
        if(err>100)
        {
          y=0;  /*bandera, no utilizada ahora*/
          asm ("  clrc XF"); /*XF es 0*/
                        /*PDDATDIR= 0xFF00;*/
        }

        else if(err<-100)
            {
              y=1;  /*bandera, no utilizada ahora*/
              asm ("   setc XF");     /*XF es 1*/
                                /*PDDATDIR= 0xFFFF;*/
            }
      }
}
```

3 OndaDSP

```
/*********************************************************************/
/* CONSTRUCCION DE UNA SEÑAL SENOIDAL A PARTIR DEL TEMPORIZADOR DEL    */
/* DSP
                                    */
/*********************************************************************/
/* PERIODO DE LA ONDA f=50 Hz; Divisiones de la onda 1000; T=20us
      */
/* Ajustal el preescaler del temporizador a 8-> significa 50 señales   */
/* por cada division de la onda, o sea 0.36 grados de onda.
      */
/*********************************************************************/

#include "regs243.h"
#include "math.h"

/*************     SETUP for the OCRA - Register   **************/
#define OCRA15                  0     /* 0 : IOPB7 1 : TCLKIN         */
#define OCRA14                  0     /* 0 : IOPB6 1 : TDIR           */
#define OCRA13                  0     /* 0 : IOPB5 1 : T2PWM          */
#define OCRA12                  0     /* 0 : IOPB4 1 : T1PWM          */
#define OCRA11                  0     /* 0 : IOPB3 1 : PWM6           */
#define OCRA10                  0     /* 0 : IOPB2 1 : PWM5           */
#define OCRA9                   0     /* 0 : IOPB1 1 : PWM4           */
#define OCRA8                   0   /* 0 : IOPB0   1 : PWM3           */
#define OCRA7                   0     /* 0 : IOPA7 1 : PWM2           */
#define OCRA6                   0     /* 0 : IOPA6 1 :   PWM1         */
#define OCRA5                   0     /* 0 : IOPA5 1 : CAP3           */
#define OCRA4                   0     /* 0 : IOPA4 1 :   CAP2/QEP2    */
#define OCRA3                   0     /* 0 : IOPA3 1 : CAP1/QEP1      */
#define OCRA2                   0     /* 0 : IOPA2 1 :   XINT1        */
#define OCRA1                   0     /* 0 : IOPA1 1 :   SCIRXD       */
#define OCRA0                   0     /* 0 : IOPA0 1 : SCITXD         */
/*************************************************************/
/*************     SETUP for the OCRB - Register   **************/
#define OCRB9                   0     /* 0 : IOPD1 1 : XINT2/EXTSOC   */
#define OCRB8                   1     /* 0 : CKLKOUT  1 : IOPD0       */
#define OCRB7                   0     /* 0 : IOPC7 1 : CANRX          */
#define OCRB6                   0     /* 0 : IOPC6 1 : CANTX          */
```

```
#define OCRB5                   0     /* 0 : IOPC5 1 : SPISTE         */
#define OCRB4                   0     /* 0 : IOPC4 1 : SPICLK         */
#define OCRB3                   0     /* 0 : IOPC3 1 : SPISOMI        */
#define OCRB2                   0     /* 0 : IOPC2 1 : SPISIMO        */
#define OCRB1                   1     /* 0 : BIO         1 : IOPC1    */
#define OCRB0                   0     /* 0 : XF          1 : IOPC0    */
/*****************************************************************/
/*************     SETUP for the WDCR - Register   **************/
#define WDDIS                   1     /* 0 : Watchdog enabled 1: disabled */
#define WDCHK2                  1     /* 0 : System reset   1: Normal OP*/
#define WDCHK1                  0     /* 0 : Normal Oper.  1: sys reset*/
#define WDCHK0                  1     /* 0 : System reset   1: Normal OP*/
#define WDSP              7     /* Watchdog prescaler 7 : div 64*/
/*****************************************************************/
/*************     SETUP for the SCSR - Register   **************/
#define CLKSRC                  0     /* 0 : intern(20MHz)            */
#define LPM                     0     /* 0 : Low power mode 0 if idle */
#define ILLADR                  1     /* 1 : clear Illegal Address Flag*/
/*****************************************************************/
/*************     SETUP for the WSGR - Register   **************/
#define BVIS              0     /* 10-9 : 00 Bus visibility OFF       */
#define ISWS              0     /* 8 -6 : 000 0 Waitstates for IO     */
#define DSWS              0     /* 5 -3 : 000 0 Waitstates  data*/
#define PSWS              0     /* 2 -0 : 000 0 Waitstaes code        */
/*****************************************************************/
/*************     SETUP for the GPTCON - Register  *************/
#define GPTCON_T2TOADC    0
  /* 10-9  : T2TOADC = 00 : no ADC-Start by any GPT2-Event            */
#define GPTCON_T1TOADC    2
  /*  8-7  : T1TOADC = 10 : ADC-Start by GPT1-Period-Event            */
#define GPTCON_TCOMPOE  0
  /*   6   : TCOMPOE = 0  : disable all 2 GPT compare outputs    */
#define GPTCON_T2PIN      0
  /*  3-2  : T2PIN = 01 : Pol. of GPT2 comp out=forced low            */
#define GPTCON_T1PIN      0
  /*  1-0  : T1PIN = 00 : Pol. of GPT1 comp out=forced low            */
/*****************************************************************/
/**************     SETUP for the T1CON - Register  *************/
#define T1CON_FREESOFT    0
  /* 15-14 FREE, SOFT : 00 stop on JTAG-emulation suspend       */
#define T1CON_TMODE       2
  /* 12-11 : TMODE1,0   : 10
                    Count mode selection: Continuous up counting mode  */
#define T1CON_TPS         0
  /* 10-8  : TPS2-0     : 000
                    Input clock prescaler CPUCLK/1                        */
#define T1CON_TENABLE     1
  /* 6     : TENABLE    : 1
                    enable GPT1,  first instruction: disable GPT1 !!   */
#define T1CON_TCLKS       0
  /* 5-4   : TCLKS1,0   : 00
                    Clock source select : internal                    */
#define T1CON_TCLD        1
  /* 3-2   : TCLD1,0    : 01
                    Timer compare(active) register reload condition
                    when counter value is 0 or equal to period register*/
#define T1CON_TECMPR      0
  /* 1     :  TECMPR        : 0
                    disable timer compare operation                   */
```

```
/************************************************************/
/**************     SETUP for the T2CON - Register  *************/
#define T2CON_FREESOFT    0
  /* 15-14 FREE, SOFT : 00 stop on JTAG-emulation suspend       */
#define T2CON_TMODE       2
  /* 12-11 : TMODE1,0   : 10
                    Count mode selection: Continuous up counting mode  */
#define T2CON_TPS         7
  /* 10-8  : TPS2-0     : 111
                    Input clock prescaler CPUCLK/128                    */
#define T2CON_TENABLE     1
  /* 6     : TENABLE    : 1
                    enable GPT1,  first instruction: disable GPT1 !!   */
#define T2CON_TCLKS       00
  /* 5-4   : TCLKS1,0   : 00
                    Clock source select : internal
      */
#define T2CON_TCLD        1
  /* 3-2   : TCLD1,0    : 01
                    Timer compare(active) register reload condition
                    when counter value is 0 or equal to period register*/
#define T2CON_TECMPR      1
  /* 1     :  TECMPR        : 0
                    disable timer compare operation
      */
#define T2CON_SELT1PR 0
        /* 0 : SELT1PR :0 use su propio registro de periodo */
/************************************************************/
/*************     SETUP for the EVIMRA - Register  *************/
#define T1OFINT                0  /* 10 : Timer 1 overflow interrupt  */
#define T1UFINT                0  /*  9 : Timer 1 underflow interrupt */
#define T1CINT                 0  /*  8 : Timer 1 compare interrupt   */
#define T1PINT                 0  /*  7 : Timer 1 period interrupt    */
#define      CMP3INT                  0  /*  3 : Compare 3 interrupt  */
#define CMP2INT                0  /*  2 : Compare 2 interrupt         */
#define CMP1INT                0  /*  1 : Compare 1 interrupt         */
#define PDPINT                 0  /*  0 : Power Drive Protect Interrupt*/
/************************************************************/
/*************     SETUP for the EVIMRB - Register  *************/
#define T2OFINT                0  /*  3 : Timer 2 overflow interrupt  */
#define T2UFINT                0  /*  2 : Timer 2 underflow interrupt  */
#define T2CINT                 0  /*  1 : Timer 2 compare interrupt   */
#define T2PINT                 1  /*  0 : Timer 2 period interrupt     */
/************************************************************/
/*************     SETUP for the EVIMRC- Register  *************/
#define CAP3INT                0  /*  2 : Capture Unit 3 interrupt    */
#define CAP2INT                0  /*  1 : Capture Unit 2 Interrupt    */
#define CAP1INT                0  /*  0 : Capture unit 1 interrupt       */
/************************************************************/
/*************     SETUP for the IMR - Register    *************/
#define INT6              1  /*  5 : Level INT6  is unmasked           */
#define INT5              0  /*  4 : Level INT5  is masked       */
#define INT4              0  /*  3 : Level INT4  is masked       */
#define INT3              1  /*  2 : Level INT3  is masked             */
#define INT2              0  /*  1 : Level INT2  is masked       */
#define INT1              0  /*  0 : Level INT1  is masked          */
/************************************************************/
/*************        SETUP for the ADCCTRL1 - Register ************/
#define SOFTFREE          2  /* 15-14 : 10 complete ADC before halt*/
```

```
#define ADCIMSTART        0  /* 13 :  no ADC immediately start   */
#define ADC2EN                1  /* 12 :  ADC-unit 2 enabled        */
#define ADC1EN                1  /* 11 :  ADC-unit 1 enabled        */
#define ADCCONRUN         0  /* 10 :  no continous mode             */
#define ADCINTEN          1  /* 9  :  enable ADC-interrupt      */
#define ADCINTFLAG        1  /* 8  :  clear previous interrupts  */
#define ADC2CHSEL         0  /* 6-4:  001 ADC-Channel 1             */
#define ADC1CHSEL         1  /* 3-1:   000    ADC-Channel 0         */
#define ADCSOC                1  /* 0  :  Start of conversion       */
/*************        SETUP for the ADCCTRL2 - Register ************/
#define ADCIM                 1  /* 14 : ADC interrupt flag when 1 result*/
#define ADCINTPRI         1  /* 11 : ADC interrupt on level 6       */
#define ADCEVSOC          1  /* 10 : enable ADC-start by EV-signal*/
#define ADCEXTSOC         0  /*  9 : disable external start         */
#define ADCPSCALE         4  /* 2-0: 000 prescaler CLKOUT/12        */
/*************************************************************/

/*** Build options ***/

/* Periodo de muestreo del conversor AD  T1CON*/

#define PERIOD1 500 /* 500--->40Khz    T1 PERIOD = 50ns * 1 * 640 = 0.000032s
(31Khz)*/
/* Periodo de formacion de la onda senoidal */
#define PERIOD2 3125  /* 50ns*128*3125=.002s o sea 50Hz. Si 360/3125 =
0.1125º */
#define DESFASE 40  /*valor en grados*/
/* definimos las variables para el seno */
float seno1,seno2;

float pi=3.14159;

signed int refin,refret;  /*señales de referencia internas y ref int
retrasada*/

unsigned int ADC0_result,ADC1_result;


signed int ref,io;
signed int err;
unsigned int y;
unsigned int
t1=0,t2=0,terror=0,contador1=0,contador2=0,contador11=0,contador22=0;


interrupt void ADC_ISR(void);
interrupt void ONDA_SEN(void);

void c_dummy1(void);
extern _out_wsgr();


void c_dummy1(void)
{
      while(1);          /*Dummy ISR used to trap spurious interrupts*/
}
```

```
interrupt void ADC_ISR(void)
{

 if((PIVR-0x0004)==0)     /*Verify type of interrupt ( 4 = ADC ) */
       {

       ADC0_result=ADCFIFO2>>6;          /* Intensidad*/
 /*asm (" LACL 7036h");     /* Vacío las FIFO */
 /*asm (" LACL 7036h");              */
       ADC1_result=ADCFIFO1>>6;          /* Referencia */

 /*asm (" LACL 7038h");
 asm (" LACL 7038h"); */
       io= (ADC0_result-565)*12;

/*FFFF corresponde a  65535, el conversor nos da FFC0 65472, al desplazar
tenemos
 3FF 1023, al hacer la diferencia tenemos intervalo (458 , -565) que al
multiplicar
 por 12 tenemos (5496,6780) que son los valores de io */
/* lectura de la señal de intensidad */
/* los 10 bits mas representativos los pasamos a los 10 bits menos
representativos */



/*       ref= (ADC1_result-555)*49; /*lectura referencia ext */

; /*Almacena el valor del timer2 para posterior manipulación */
/* este valor esta comprendido entre 0 y 3125  */

t1=T2CNT;
       seno1=sin(2*pi*T2CNT/PERIOD2);
       refin=seno1*2000;
       err=(io-refin);



t2=T2CNT;                /*programa para medir el tiempo de ejecuión */
terror=t2-t1;

if(terror<10)
{      contador1++;
       if(contador1==60000)
              contador11++;
              }
else
{      contador2++;
       if(contador2==60000)
              contador22++;
       }

/*onda desfasada*/
/*pues como nuestra onda se mueve entre 0 y 2*pi o sea entre 0 y 360 grados
si
queremos retrasar nuestra onda x grados nuestra onda se tiene que mover entre
0-x y 360-x grados, en radianes es 0-x/360*2*pi y 2pi-x/360*2*pi. La ecuacion
es
Y-x/360*2*pi, esto es-> 2*pi*(T2CNT/PERIOD2-x/360); DESFASE=x */
```

```
      seno2=sin(2*pi*(T2CNT/PERIOD2-DESFASE/360));
      refret=seno2*2000; /* refret es referencia retrasada*/


    ADCTRL1 |= 0x0100;     /*Volver a activar el conversor AD */
      }
}
interrupt void ONDA_SEN(void)   /* Para construir la señal senoidal*/
{
 if((PIVR-0x002B)==0) /*verifica interrupción Compara Timer2 */
        EVIFRB=T2PINT;     /*habilita la interrupción escribiendo un 1*/
}

/*****************************************************************/
/* INICIO PROGRAMA PRINCIPAL*/
/*****************************************************************/
void main(void)
{
 asm ("  clrc XF");
 asm (" setc INTM");/*Disable all interrupts                          */
 asm (" clrc SXM");/*Clear Sign Extension Mode bit                    */
 asm (" clrc OVM");/*Reset Overflow Mode bit*/
 asm (" clrc CNF");/*Configure block B0 to data mem.                  */

 WDCR=((WDDIS<<6)+(WDCHK2<<5)+(WDCHK1<<4)+(WDCHK0<<3)+WDSP);
                              /* Initialize Watchdog-timer
      */

 SCSR = ((CLKSRC<<14)+(LPM<<12)+ILLADR); /* Initialize SCSR
      */

 out_wsgr((BVIS<<9)+(ISWS<<6)+(DSWS<<3)+PSWS);
                              /* external Function for access  WSGR     */

 OCRB = ((OCRB9<<9)+(OCRB8<<8)+
              (OCRB7<<7)+(OCRB6<<6)+(OCRB5<<5)+(OCRB4<<4)+
              (OCRB3<<3)+(OCRB2<<2)+(OCRB1<<1)+OCRB0);
                              /* Initialize output control register B*/


 OCRA = ((OCRA15<<15)+(OCRA14<<14)+(OCRA13<<13)+(OCRA12<<12)+
         (OCRA11<<11)+(OCRA10<<10)+(OCRA9<<9)+(OCRA8<<8)+
         (OCRA7<<7)+(OCRA6<<6)+(OCRA5<<5)+(OCRA4<<4)+
         (OCRA3<<3)+(OCRA2<<2)+(OCRA1<<1)+OCRA0);
                        /* Initialize output control register A      */

 GPTCON=((GPTCON_T2TOADC<<9)+
         (GPTCON_T1TOADC<<7)+
         (GPTCON_TCOMPOE<<6)+
         (GPTCON_T2PIN<<2)+
         (GPTCON_T1PIN)); /*   Initialize GP Timer Control           */


 PBDATDIR = 0x0000;    /* use IOPB0..B7 as inputs                */

 /*Temporizador 1 */
 T1PR = PERIOD1;             /* initialize T1-period                */
 T1CNT= 0x0000;              /* start value for T1-counter       */
```

```
T1CON=((T1CON_FREESOFT<<14)+
          (T1CON_TMODE<<11)+
          (T1CON_TPS<<8)+
          (T1CON_TCLKS<<4)+
          (T1CON_TCLD<<2)+
          (T1CON_TECMPR<<1));

/* Temporizador 2 */
T2PR = PERIOD2;             /* initialize T2-period                    */
T2CNT= 0x0000;              /* start value for T2-counter      */

T2CON=((T2CON_FREESOFT<<14)+
          (T2CON_TMODE<<11)+
          (T2CON_TPS<<8)+
          (T2CON_TCLKS<<4)+
          (T2CON_TCLD<<2)+
          (T2CON_TECMPR<<1)+
     T2CON_SELT1PR);

ADCTRL1= ((SOFTFREE<<14)+(ADCIMSTART<<13)+(ADC2EN<<12)+
                  (ADC1EN<<11) +(ADCCONRUN<<10) +(ADCINTEN<<9)+
                  (ADCINTFLAG<<8)+(ADC2CHSEL<<4)+(ADC1CHSEL<<1));
                              /*  Initialize ADC control register 1*/

ADCTRL2= ((ADCIM<<14)+(ADCINTPRI<<11)+
          (ADCEVSOC<<10)+(ADCEXTSOC<<9)+ADCPSCALE);

asm (" LACL 7036h");     /* Vacío las FIFO */
asm (" LACL 7036h");
asm (" LACL 7038h");
asm (" LACL 7038h");

EVIFRA=0xFFFF;      /* EV Interrupt Flag Register Group A      */

EVIFRB=0xFFFF;      /* EV Interrupt Flag Register Group B */

EVIFRC=0xFFFF;     /* EV Interupt Flag Register Group C  */

EVIMRA=((T1OFINT<<10)+
           (T1UFINT<<9)+
           (T1CINT<<8)+
           (T1PINT<<7)+
           (CMP3INT<<3)+
           (CMP2INT<<2)+
           (CMP1INT<<1)+
           (PDPINT)); /* EV Interrupt Mask Register Group A  */

EVIMRB=((T2OFINT<<3)+
         (T2UFINT<<2)+
         (T2CINT<<1)+
         (T2PINT));  /* EV Interrupt Mask Register Group B */


EVIMRC=((CAP3INT<<2)+
             (CAP2INT<<1)+
             (CAP1INT)); /* EV Interrupt Mask Register Group C */

IFR=0xFFFF; /* Interrupt Flag Register , address 0x0006  */
```

```
                  /* Reset  all core interrupts
      */

/* Interrupciones:
                  INT6 interrupción del ACD
                       INT3 interrupcióm del Timer2
                       */
 IMR=((INT6<<5)+
       (INT5<<4)+
       (INT4<<3)+
       (INT3<<2)+
       (INT2<<1)+
       (INT1));      /* Interrupt Mask Register                        */


 asm (" clrc INTM");  /* Enable all unmasked interrupts         */

 T1CON=T1CON+(T1CON_TENABLE<<6); /* enable GPT1 now       */
                            /* Initialize GPT1                        */
 T2CON=T2CON+(T2CON_TENABLE<<6); /*timer 2 funcionando */

 PCDATDIR = 0xFF02;   /* Clear Port D , all LED's off           */


 while(1){

      /*io= (ADC0_result-565)*12;    (5496,6780) valores maximos
        ref= (ADC1_result-545)*49;   (23422,-26705) si io es 4 veces mayor
que
        ref tenemos que ref se mueve entre (5855,6676) valores maximos
        err= (io - ref);  */
        if(err>684)
        {
          y=0;
          asm ("  clrc XF");
          /*PDDATDIR= 0xFF00;*/
        }

        else if(err<-684)
            {
              y=1;
              asm ("   setc XF");
              /*PDDATDIR= 0xFFFF;*/
            }
      }
}
```

4 DSPSENO

El programa comentado en el tema 6

```
/*******************************************************/

#include "regs243.h"
```

```
#include "math.h"

/*****************  SETUP del registro OCRA  *************/
#define OCRA15                  0      /* 0:IOPB7   1:TCLKIN       */
#define OCRA14                  0      /* 0:IOPB6   1:TDIR   */
#define OCRA13                  0      /* 0:IOPB5   1:T2PWM        */
#define OCRA12                  0      /* 0:IOPB4   1:T1PWM        */
#define OCRA11                  0      /* 0:IOPB3   1:PWM6   */
#define OCRA10                  0      /* 0:IOPB2   1:PWM5   */
#define OCRA9                   0      /* 0:IOPB1   1:PWM4   */
#define OCRA8                   0      /* 0:IOPB0   1:PWM3   */
#define OCRA7                   0      /* 0:IOPA7   1:PWM2   */
#define OCRA6                   0      /* 0:IOPA6   1:PWM1   */
#define OCRA5                   0      /* 0:IOPA5   1:CAP3   */
#define OCRA4                   0      /* 0:IOPA4   1:CAP2/QEP2 */
#define OCRA3                   0      /* 0:IOPA3   1:CAP1/QEP1 */
#define OCRA2                   0      /* 0:IOPA2   1:XINT1        */
#define OCRA1                   0      /* 0:IOPA1   1:SCIRXD       */
#define OCRA0                   0      /* 0:IOPA0   1:SCITXD       */
/*********************************************************/

/**************** SETUP del registro OCRB  ***************/
#define OCRB9                   0      /* 0:IOPD1  1:XINT2/EXTSOC */
#define OCRB8                   1      /* 0:CKLKOUT 1:IOPD0        */
#define OCRB7                   0      /* 0:IOPC7   1:CANRX        */
#define OCRB6                   0      /* 0:IOPC6   1:CANTX        */
#define OCRB5                   0      /* 0:IOPC5   1:SPISTE       */
#define OCRB4                   0      /* 0:IOPC4   1:SPICLK       */
#define OCRB3                   0      /* 0:IOPC3   1:SPISOMI      */
#define OCRB2                   0      /* 0:IOPC2   1:SPISIMO      */
#define OCRB1                   1      /* 0:BIO           1:IOPC1        */
#define OCRB0                   0      /* 0:XF            1:IOPC0        */
/*********************************************************/

/*******************    SETUP el WDCR    *****************/
#define WDDIS                   1      /* 0:Watchdog enabled 1:disabled*/
#define WDCHK2                  1      /* 0:System reset  1:Normal OP  */
#define WDCHK1                  0      /* 0:Normal Oper. 1: sys reset  */
#define WDCHK0                  1      /* 0:System reset 1:Normal OP   */
#define WDSP              7     /* Watchdog prescaler 7 : div 64*/
/*********************************************************/

/*************     SETUP del registro SCSR  **************/
#define CLKSRC                  0      /* 0:interno(20MHz)         */
#define LPM               0     /* 0:Modo bajo consume 0 si IDLE*/
#define ILLADR                  1      /* 1:borrar ILLADR          */
/*********************************************************/

/*************     SETUP de WSGR ************************/
#define BVIS       0  /* 10-9:00 Bus visibility OFF       */
#define ISWS       0  /* 8-6:000 0 estados de espera para I/O*/
#define DSWS       0  /* 5-3:000 0 estados de esp. para datos */
#define PSWS       0  /* 2-0:000 0 estados de esp. para código*/
/*********************************************************/

/*************   SETUP del registro GPTCON **************/
#define GPTCON_T2TOADC    0
  /* 10-9  : T2TOADC = 00 : GPT2 no inicia ADC            */
#define GPTCON_T1TOADC    2
```

```
  /*  8-7  : T1TOADC = 10 : GPT1 inicia ACD por periodo */
#define GPTCON_TCOMPOE  0
   /*  6   : TCOMPOE = 0  : deshabilitadas las 2 salidas de comparación del
   GPT */
#define GPTCON_T2PIN      0
   /*  3-2  : T2PIN = 00 : Pol. De salida GPT2=forzada nivel bajo
       */
#define GPTCON_T1PIN      0
   /*  1-0  : T1PIN = 00 : Pol. De salida GPT1=forzada nivel bajo
       */
/*******************************************************/

/**************  SETUP de registro T1CON *****************/
#define T1CON_FREESOFT    0
   /* 15-14 FREE, SOFT : 00 stop en emulación JTAG suspendido*/
#define T1CON_TMODE       2
   /* 12-11 : TMODE1,0 : 10 Modo Continuous up en conteo */
#define T1CON_TPS         0
   /* 10-8  : TPS2-0   : 000 prescaler del reloj de entrada CPUCLK/1 */
#define T1CON_TENABLE     1
  /* 6     : TENABLE    : 1 Habilita GPT1                   */
#define T1CON_TCLKS       0
  /* 5-4   : TCLKS1,0   : 00 Fuente de reloj:interno        */
#define T1CON_TCLD        1
   /* 3-2   : TCLD1,0    : 01 Recarga el timer 1 al llegar a 0 o a T */
#define T1CON_TECMPR      0
   /* 1     :  TECMPR     : 0 deshabilita operación de comparación del timer
   */
/*******************************************************/

/**************     SETUP de registro T2CON **************/
#define T2CON_FREESOFT    0
   /* 15-14 FREE, SOFT : 00 stop en emulación JTAG suspendido*/
#define T2CON_TMODE       2
  /* 12-11 : TMODE1,0 : 10 Modo Continuous up en conteo */
#define T2CON_TPS         7
  /* 10-8  : TPS2-0   : 111 prescaler del reloj de entrada
                              CPUCLK/128 */
#define T2CON_TENABLE     1
  /* 6     : TENABLE    : 1     Habilita GPT1                    */
#define T2CON_TCLKS       00
  /* 5-4   : TCLKS1,0   : 00 Fuente de reloj:interno        */
#define T2CON_TCLD        1
   /* 3-2   : TCLD1,0    : 01 Recarga el timer 2 al llegar a 0 o a T */
#define T2CON_TECMPR      1
   /* 1     :  TECMPR     : 1 habilita operación de comparación del timer */
#define T2CON_SELT1PR 0
  /* 0 : SELT1PR :0 use su propio registro de periodo */
/*******************************************************/

/*************     SETUP del registro EVIMRA *************/
#define T1OFINT     0  /* 10 : Timer 1 overflow interrupt         */
#define T1UFINT     0  /*  9 : Timer 1 underflow interrupt        */
#define T1CINT            0  /*  8 : Timer 1 compare interrupt          */
#define T1PINT            0  /*  7 : Timer 1 period interrupt    */
#define CMP3INT     0  /*  3 : Compare 3 interrupt                */
#define CMP2INT     0  /*  2 : Compare 2 interrupt                */
#define CMP1INT     0  /*  1 : Compare 1 interrupt                */
#define PDPINT            0  /*  0 : Power Drive Protect Interrupt*/
```

```
/*******************************************************/

/*************     SETUP del registro EVIMRB *************/
#define T2OFINT     0  /*  3 : Timer 2 overflow interrupt        */
#define T2UFINT     0  /*  2 : Timer 2 underflow interrupt  */
#define T2CINT            0  /*  1 : Timer 2 compare interrupt   */
#define T2PINT            1  /*  0 : Timer 2 period interrupt    */
/*******************************************************/

/*************     SETUP del registro EVIMRC *************/
#define CAP3INT     0  /*  2 : Capture Unit 3 interrupt    */
#define CAP2INT     0  /*  1 : Capture Unit 2 Interrupt    */
#define CAP1INT     0  /*  0 : Capture unit 1 interrupt    */
/*******************************************************/

/*************     SETUP del IMR    **********************/
#define INT6        1  /*  5 : Level INT6  no emascarada */
#define INT5        0  /*  4 : Level INT5  enmascarada   */
#define INT4        0  /*  3 : Level INT4  enmascarada   */
#define INT3        1  /*  2 : Level INT3  enmascarada   */
#define INT2        0  /*  1 : Level INT2  enmascarada   */
#define INT1        0  /*  0 : Level INT1  enmascarada   */
/*******************************************************/

/*************        SETUP del registro ADCCTRL1 ************/
#define SOFTFREE    2  /* 15-14 : 10 completa conversión antes de parar */
#define ADCIMSTART  0  /* 13 :  no comienza conversión inmediatamente */
#define ADC2EN            1  /* 12 :  unidad ADC2 habilitada       */
#define ADC1EN            1  /* 11 :  unidad ADC1 habilitada       */
#define ADCCONRUN   0  /* 10 :  conversión en modo no continuo          */
#define ADCINTEN    1  /* 9  :  habilita interrupción del ADC    */
#define ADCINTFLAG  1  /* 8  :  borra interrupción del ADC */
#define ADC2CHSEL   0  /* 6-4:  000 canal 0 para la unidad ADC2  */
#define ADC1CHSEL   1  /* 3-1:  001 canal 1 para la unidad ADC1  */
#define ADCSOC            1  /* 0  :  Inicio de conversión    */
/*******************************************************/

/*************        SETUP del registro ADCCTRL2 ************/
#define ADCIM             1  /* 14 : flag de interrupción del ACD */
#define ADCINTPRI   1  /* 11 :prioridad de interrupción de ADC nivel 6  */
#define ADCEVSOC    1  /* 10 : habilitado inicio ADC por gestor de Eventos */
#define ADCEXTSOC   0  /*  9 :deshabilitado inicio externo del ADC */
#define ADCPSCALE   4  /* 2-0: 100 prescaler CLKOUT/12     */
/*******************************************************/
#define PERIOD 500 /* 500--->40Khz     T1 PERIOD = 50ns * 1 * 640 = 0.000032s
(31Khz)*/

/* Periodo de formación de la onda senoidal */
#define PERIOD2 3125 /* 50ns*128*3125=.002s o sea 50Hz. Si 360/3125 = 0.1125º
*/

#define DESFASE 40  /*valor en grados del desfase que se le quiere introducir
a la señal*/

/* definimos las variables para el seno */

float pi=3.14159;

signed int seno[360],senodesf[360]; /*tablas del seno y seno desfasado*/
```

```
signed int refin,refret;  /*señales de referencia internas y referencia
internas desfasada*/

unsigned int tref; /*tiempo de referencia*/

unsigned int desf=DESFASE; /*desfase*/

unsigned int ADC0_result;


signed int ref,io,err;

unsigned int y,n;  /*parametros de interacion*/
```

Se define la función de interrupción interrupt void ADC_ISR(void), interrupt void ONDA_SEN(void)y la función para atrapar interrupciones void c_dummy1(void).

```
interrupt void ADC_ISR(void);
interrupt void ONDA_SEN(void);

void c_dummy1(void);
extern _out_wsgr(); /*declaración de función externa para
       configurar WSGR */


void c_dummy1(void)
{
      while(1);          /*función para atrapar interrupciones
             espúreas */
}
interrupt void ADC_ISR(void) /*rutina de interrupción*/
{

 if((PIVR-0x0004)==0)     /*Verifica la interrupción ( 4 =
                   ADC )  */
      {
   /*lo primero que hacemos es leer el tiempo de la onda que generamos*/
      tref=T2CNT; /*valor entre 0 y 3125*/
      ADC0_result=ADCFIFO2>>6;          /* Intensidad*/
 /*   asm (" LACL 7036h");     /* Vacío las FIFO */
 /*   asm (" LACL 7036h");  */

      io= (ADC0_result-565)*12;

      ADCTRL1 |= 0x0100;   /*activar conversor*/
      }
}
interrupt void ONDA_SEN(void)   /* Para construir la señal      senoidal*/
{
      if((PIVR-0x002B)==0) /*verifica interrupción Compara Timer2 */
          EVIFRB=T2PINT;     /*habilita la interrupción escribiendo un 1*/
}
void main(void)
{
 asm ("  clrc XF");/*Pone 0 en la salida XF */
```

```
asm (" setc INTM"); /*Deshabilita todas las interrupciones
           */
asm (" clrc SXM");/*Borra el bit de extensión de signo*/
asm (" clrc OVM");/*Borra bit de modo de desbordamiento*/
asm (" clrc CNF");/*Configura B0 como memoria de datos */

WDCR=((WDDIS<<6)+(WDCHK2<<5)+(WDCHK1<<4)+(WDCHK0<<3)+WDSP);
                              /*Inicializa el registro WDCR    */

SCSR = ((CLKSRC<<14)+(LPM<<12)+ILLADR); /* Inicializa SCSR
     */

out_wsgr((BVIS<<9)+(ISWS<<6)+(DSWS<<3)+PSWS);
                 /* Función externa para configurar WSGR        */

OCRB = ((OCRB9<<9)+(OCRB8<<8)+
             (OCRB7<<7)+(OCRB6<<6)+(OCRB5<<5)+(OCRB4<<4)+
             (OCRB3<<3)+(OCRB2<<2)+(OCRB1<<1)+OCRB0);
                              /* Inicializa  OCRB       */


OCRA = ((OCRA15<<15)+(OCRA14<<14)+(OCRA13<<13)+(OCRA12<<12)+
         (OCRA11<<11)+(OCRA10<<10)+(OCRA9<<9)+(OCRA8<<8)+
         (OCRA7<<7)+(OCRA6<<6)+(OCRA5<<5)+(OCRA4<<4)+
         (OCRA3<<3)+(OCRA2<<2)+(OCRA1<<1)+OCRA0);
                        /* Inicializa OCRA        */

GPTCON=((GPTCON_T2TOADC<<9)+
         (GPTCON_T1TOADC<<7)+
         (GPTCON_TCOMPOE<<6)+
         (GPTCON_T2PIN<<2)+
         (GPTCON_T1PIN)); /*   Inicia el registro de control
                   del temporizador  */


PBDATDIR = 0x0000;    /*configure IOPB..B7 como entradas*/
```

Aquí está el temporizador 1, donde se define el tiempo de muestreo y del temporizador 2 donde se define el tiempo de formación de la señal senoidal que será la referencia de la señal que se va a construir en el inversor.

```
T1PR = PERIOD;              /*configura el periodo del
     temporizador T1    */
T1CNT= 0x0000;              /* valor inicial del contador
     */

T1CON=((T1CON_FREESOFT<<14)+
          (T1CON_TMODE<<11)+
          (T1CON_TPS<<8)+
          (T1CON_TCLKS<<4)+
          (T1CON_TCLD<<2)+
          (T1CON_TECMPR<<1)); /*Configura el registro de
                        control del temporizador 1*/


T2PR = PERIOD2;             /*configure el periodo del
            temporizador T2    */
T2CNT= 0x0000;              /*valor inicial del contador
            */
```

```
T2CON=((T2CON_FREESOFT<<14)+
          (T2CON_TMODE<<11)+
          (T2CON_TPS<<8)+
          (T2CON_TCLKS<<4)+
          (T2CON_TCLD<<2)+
          (T2CON_TECMPR<<1)+
         T2CON_SELT1PR);      /*configuración del
                       registro timer 2*/




ADCTRL1= ((SOFTFREE<<14)+(ADCIMSTART<<13)+(ADC2EN<<12)+
           (ADC1EN<<11) +(ADCCONRUN<<10) +(ADCINTEN<<9)+
           (ADCINTFLAG<<8)+(ADC2CHSEL<<4)+(ADC1CHSEL<<1));
                 /* Configura registro de control 1 de ADC*/

ADCTRL2= ((ADCIM<<14)+(ADCINTPRI<<11)+
          (ADCEVSOC<<10)+(ADCEXTSOC<<9)+ADCPSCALE);
                 /*Configura registro de control 2 de ADC*/

asm (" LACL 7036h");     /* Vacío las FIFO */
asm (" LACL 7036h");
asm (" LACL 7038h");
asm (" LACL 7038h");

EVIFRA=0xFFFF;      /* Registro de flags del grupo A     de
                 EV*/

EVIFRB=0xFFFF;      /* Registro de flags del grupo B     de
                 EV*/

EVIFRC=0xFFFF;     /* Registro de flags del grupo C      de
                 EV*/


EVIMRA=((T1OFINT<<10)+
           (T1UFINT<<9)+
           (T1CINT<<8)+
           (T1PINT<<7)+
           (CMP3INT<<3)+
           (CMP2INT<<2)+
           (CMP1INT<<1)+
           (PDPINT)); /*Registro de máscara del grupo A del
            EV*/

EVIMRB=((T2OFINT<<3)+
         (T2UFINT<<2)+
         (T2CINT<<1)+
         (T2PINT));  /*Registro de máscara del grupo B del
            EV*/

EVIMRC=((CAP3INT<<2)+
             (CAP2INT<<1)+
             (CAP1INT)); /*Registro de máscara del grupo C
                 del EV*/
```

```
 IFR=0xFFFF; /*Borra todas las interrupciones pendientes */

 IMR=((INT6<<5)+
       (INT5<<4)+
       (INT4<<3)+
       (INT3<<2)+
       (INT2<<1)+
       (INT1));      /*Registro de máscaras de interrupción
       */
/*******************************************************/
 /*Programa generación tabla del seno y desfase*/


 for(n=0;n<180;n++)
  {
      seno[n]=1023*sin(pi*n/180);       /*el valor de lo almacenado va desde
0 hasta n-1*/
    seno[180+n]=-seno[n];             /*1023, factor de escala*/
      }

  for(n=0;n<360;n++)    /*creación onda desfasada*/
  {
      if(n+desf<360)
            {
            senodesf[n+desf]=seno[n];     /*angulo comprendido entre desf y
360*/
            }
      else
      {
            senodesf[(n+desf)-360]=seno[n];    /*comprendido entre 0 y
desf*/
            }
  }

 /*******************************************************/


 asm (" clrc INTM");  /*Habilita todas las interrupciones
            */

 T1CON=T1CON+(T1CON_TENABLE<<6); /*Habilita el GPT1 */
T2CON=T2CON+(T2CON_TENABLE<<6); /*timer 2 funcionando */

 while(1){

        n=(tref*360)/PERIOD2; /*valor comprendido entre 0 y
359 grados*/

        ref=5*seno[n]; /*amplificamos la señal por 5*/
        err=io-ref;  /*error de seguimiento*/



        if(err>100)
        {
          y=0;  /*bandera, no utilizada ahora*/
          asm ("clrc XF"); /*XF es 0*/
                        /*PDDATDIR= 0xFF00;*/
        }
```

```
        else if(err<-100)
            {
              y=1;  /*bandera, no utilizada ahora*/
              asm ("setc XF");  /*XF es 1*/
                                /*PDDATDIR= 0xFFFF;*/
            }
      }
}
```

5 f243acd2.c

Este es el programa principal que se utiliza en la programación del DSP para la mitigación del campo magnético. A su vez se hace llamadas a funciones que se encuentra en el fichero calculos.c. Este programa está ampliamente comentado en el capítulo 8 del documento.

```
#include "regs243.h"
#include "math.h"

/*****************  SETUP del registro OCRA  *************/
#define OCRA15                  0     /* 0:IOPB7   1:TCLKIN       */
#define OCRA14                  0     /* 0:IOPB6   1:TDIR   */
#define OCRA13                  0     /* 0:IOPB5   1:T2PWM        */
#define OCRA12                  0     /* 0:IOPB4   1:T1PWM        */
#define OCRA11                  0     /* 0:IOPB3   1:PWM6   */
#define OCRA10                  0     /* 0:IOPB2   1:PWM5   */
#define OCRA9                   0     /* 0:IOPB1   1:PWM4   */
#define OCRA8                   0     /* 0:IOPB0   1:PWM3   */
#define OCRA7                   0     /* 0:IOPA7   1:PWM2   */
#define OCRA6                   0     /* 0:IOPA6   1:PWM1   */
#define OCRA5                   0     /* 0:IOPA5   1:CAP3   */
#define OCRA4                   0     /* 0:IOPA4   1:CAP2/QEP2 */
#define OCRA3                   0     /* 0:IOPA3   1:CAP1/QEP1 */
#define OCRA2                   0     /* 0:IOPA2   1:XINT1        */
#define OCRA1                   0     /* 0:IOPA1   1:SCIRXD       */
#define OCRA0                   0     /* 0:IOPA0   1:SCITXD       */
/*********************************************************/

/***************** SETUP del registro OCRB  ***************/
#define OCRB9                   0     /* 0:IOPD1  1:XINT2/EXTSOC */
#define OCRB8                   1     /* 0:CKLKOUT 1:IOPD0        */
#define OCRB7                   0     /* 0:IOPC7   1:CANRX        */
#define OCRB6                   0     /* 0:IOPC6   1:CANTX        */
#define OCRB5                   0     /* 0:IOPC5   1:SPISTE       */
#define OCRB4                   0     /* 0:IOPC4   1:SPICLK       */
#define OCRB3                   0     /* 0:IOPC3   1:SPISOMI      */
#define OCRB2                   0     /* 0:IOPC2   1:SPISIMO      */
#define OCRB1                   1     /* 0:BIO           1:IOPC1        */
#define OCRB0                   0     /* 0:XF            1:IOPC0        */
/*********************************************************/

/*******************    SETUP el WDCR    *****************/
#define WDDIS                   1     /* 0:Watchdog enabled 1:disabled*/
#define WDCHK2                  1     /* 0:System reset  1:Normal OP  */
#define WDCHK1                  0     /* 0:Normal Oper. 1: sys reset  */
#define WDCHK0                  1     /* 0:System reset 1:Normal OP   */
#define WDSP              7     /* Watchdog prescaler 7 : div 64*/
/*********************************************************/

/*************     SETUP del registro SCSR  **************/
#define CLKSRC                  0     /* 0:interno(20MHz)         */
#define LPM               0     /* 0:Modo bajo consume 0 si IDLE*/
#define ILLADR                  1     /* 1:borrar ILLADR          */
/*********************************************************/
```

```
/*************     SETUP de WSGR ************************/
#define BVIS       0  /* 10-9:00 Bus visibility OFF       */
#define ISWS       0  /* 8-6:000 0 estados de espera para I/O*/
#define DSWS       0  /* 5-3:000 0 estados de esp. para datos */
#define PSWS       0  /* 2-0:000 0 estados de esp. para código*/
/*****************************************************/

/*************   SETUP del registro GPTCON ***************/
#define GPTCON_T2TOADC    0
  /* 10-9  : T2TOADC = 00 : GPT2 no inicia ADC            */
#define GPTCON_T1TOADC    2
  /*  8-7  : T1TOADC = 10 : GPT1 inicia ACD por periodo */
#define GPTCON_TCOMPOE  0
   /*  6   : TCOMPOE = 0  : deshabilitadas las 2 salidas de comparación del
   GPT */
#define GPTCON_T2PIN      0
   /*  3-2  : T2PIN = 00 : Pol. De salida GPT2=forzada nivel bajo
       */
#define GPTCON_T1PIN      0
   /*  1-0  : T1PIN = 00 : Pol. De salida GPT1=forzada nivel bajo
       */
/*****************************************************/

/**************  SETUP de registro T1CON *****************/
#define T1CON_FREESOFT    0
   /* 15-14 FREE, SOFT : 00 stop en emulación JTAG suspendido*/
#define T1CON_TMODE       2
   /* 12-11 : TMODE1,0 : 10 Modo Continuous up en conteo */
#define T1CON_TPS         0
   /* 10-8  : TPS2-0   : 000 prescaler del reloj de entrada CPUCLK/1 */
#define T1CON_TENABLE     1
  /* 6     : TENABLE    : 1 Habilita GPT1                  */
#define T1CON_TCLKS       0
  /* 5-4   : TCLKS1,0   : 00 Fuente de reloj:interno       */
#define T1CON_TCLD        1
   /* 3-2   : TCLD1,0    : 01 Recarga el timer 1 al llegar a 0 o a T */
#define T1CON_TECMPR      0
   /* 1     :  TECMPR     : 0 deshabilita operación de comparación del timer
   */
/*****************************************************/

/**************     SETUP de registro T2CON **************/
#define T2CON_FREESOFT    0
   /* 15-14 FREE, SOFT : 00 stop en emulación JTAG suspendido*/
#define T2CON_TMODE       2
  /* 12-11 : TMODE1,0 : 10 Modo Continuous up en conteo */
#define T2CON_TPS         7
  /* 10-8  : TPS2-0   : 111 prescaler del reloj de entrada
                               CPUCLK/128 */
#define T2CON_TENABLE     1
  /* 6     : TENABLE    : 1     Habilita GPT1                    */
#define T2CON_TCLKS       00
  /* 5-4   : TCLKS1,0   : 00 Fuente de reloj:interno       */
#define T2CON_TCLD        1
   /* 3-2   : TCLD1,0    : 01 Recarga el timer 2 al llegar a 0 o a T */
#define T2CON_TECMPR      1
   /* 1     :  TECMPR     : 1 habilita operación de comparación del timer */
#define T2CON_SELT1PR 0
  /* 0 : SELT1PR :0 use su propio registro de periodo */
```

```
/*******************************************************/

/*************     SETUP del registro EVIMRA *************/
#define T1OFINT     0  /* 10 : Timer 1 overflow interrupt         */
#define T1UFINT     0  /*  9 : Timer 1 underflow interrupt        */
#define T1CINT            0  /*  8 : Timer 1 compare interrupt         */
#define T1PINT            0  /*  7 : Timer 1 period interrupt    */
#define CMP3INT     0  /*  3 : Compare 3 interrupt                */
#define CMP2INT     0  /*  2 : Compare 2 interrupt                */
#define CMP1INT     0  /*  1 : Compare 1 interrupt                */
#define PDPINT            0  /*  0 : Power Drive Protect Interrupt*/
/*******************************************************/

/*************     SETUP del registro EVIMRB *************/
#define T2OFINT     0  /*  3 : Timer 2 overflow interrupt         */
#define T2UFINT     0  /*  2 : Timer 2 underflow interrupt  */
#define T2CINT            0  /*  1 : Timer 2 compare interrupt   */
#define T2PINT            1  /*  0 : Timer 2 period interrupt    */
/*******************************************************/

/*************     SETUP del registro EVIMRC *************/
#define CAP3INT     0  /*  2 : Capture Unit 3 interrupt    */
#define CAP2INT     0  /*  1 : Capture Unit 2 Interrupt    */
#define CAP1INT     0  /*  0 : Capture unit 1 interrupt    */
/*******************************************************/

/*************     SETUP del IMR    **********************/
#define INT6        1  /*  5 : Level INT6  no emascarada */
#define INT5        0  /*  4 : Level INT5  enmascarada   */
#define INT4        0  /*  3 : Level INT4  enmascarada   */
#define INT3        1  /*  2 : Level INT3  enmascarada   */
#define INT2        0  /*  1 : Level INT2  enmascarada   */
#define INT1        0  /*  0 : Level INT1  enmascarada   */
/*******************************************************/

/*************        SETUP del registro ADCCTRL1 ************/
#define SOFTFREE    2  /* 15-14 : 10 completa conversión antes de parar */
#define ADCIMSTART  0  /* 13 :  no comienza conversión inmediatamente */
#define ADC2EN            1  /* 12 :  unidad ADC2 habilitada       */
#define ADC1EN            1  /* 11 :  unidad ADC1 habilitada       */
#define ADCCONRUN   0  /* 10 :  conversión en modo no continuo          */
#define ADCINTEN    1  /* 9  :  habilita interrupción del ADC    */
#define ADCINTFLAG  1  /* 8  :  borra interrupción del ADC */
#define ADC2CHSEL   0  /* 6-4:  000 canal 0 para la unidad ADC2  */
#define ADC1CHSEL   1  /* 3-1:  001 canal 1 para la unidad ADC1  */
#define ADCSOC            1  /* 0  :  Inicio de conversión    */
/*******************************************************/

/*************        SETUP del registro ADCCTRL2 ************/
#define ADCIM             1  /* 14 : flag de interrupción del ACD */
#define ADCINTPRI   1  /* 11 :prioridad de interrupción de ADC nivel 6  */
#define ADCEVSOC    1  /* 10 : habilitado inicio ADC por gestor de Eventos */
#define ADCEXTSOC   0  /*  9 :deshabilitado inicio externo del ADC */
#define ADCPSCALE   4  /* 2-0: 100 prescaler CLKOUT/12     */
/*******************************************************/
#define PERIOD1 500 /* 500--->40Khz    T1 PERIOD = 50ns * 1 * 500 = 0.000020s
(40Khz)*/

/* Periodo de formación de la onda senoidal */
```

```
#define PERIOD2 3125 /* 50ns*128*3125=.002s o sea 50Hz. Si 360/3125 = 0.1125º
*/

#define MUESTRA 9000 /* 0.02*3000=60 segundos*/
#define DESFASE 26  /* Es el desfase que introduce el circuito lector del
campo magnético debido a los amplificadores operacionales */

/* definimos las variables para el seno */

float pi=3.14159;
float seno[360];
signed int senor[360];

signed int maxi;  /*señales de referencia internas y ref int retrasada*/

unsigned int ADC0_result,ADC1_result;
unsigned int medperiod,desf=DESFASE;  /*360-26*/
signed int ref,io;
signed int refmax_pos=0,refmax_neg=0;
signed int err;
unsigned int y=0,n,t=0,calmax=0;
signed int ip=0,cero=0,tz=0;
unsigned int t1=0;
unsigned int contad=0;
signed int adc1,adc2=0; /* tabla de dos elementos para guardar las ultimas
conversiones*/

float bfx,bfy=1,blx,bly;
float absil;
interrupt void ADC_ISR(void);
interrupt void ONDA_SEN(void);


void c_dummy1(void);
extern _out_wsgr(); /*declaración de función externa para
       configurar WSGR */
void muestra(void);
float int_lazo(float, float, float ,float );


void c_dummy1(void)
{
      while(1);          /*función para atrapar interrupciones
             espurias */
}
interrupt void ADC_ISR(void) /*rutina de interrupción*/
{

 if((PIVR-0x0004)==0)     /*Verifica la interrupción ( 4 =
                  ADC )  */
      {
   /*lo primero que hacemos es leer el tiempo de la onda que generamos*/
      t1=T2CNT; /*valor entre 0 y 3125*/
      ADC0_result=ADCFIFO2>>6;          /* Intensidad*/
 /*   asm (" LACL 7036h");     /* Vacío las FIFO */
 /*   asm (" LACL 7036h");  */
      ADC1_result=ADCFIFO1>>6;    /* Referencia de la
                                 línea */
 /*asm (" LACL 7038h");
```

```
   asm (" LACL 7038h"); */
io= (ADC0_result-565);
ref= (ADC1_result-555);
       if(y)
       {
              n=PERIOD2/360;    /*ángulo de la onda*/
              n=t1/n;
       }
/*******************************************************/
/*muestreo de la onda busqueda de 0 */
/*lo que hacemos aquí es desfasar el tiempo, y con ello*/
/*la onda desfasada*/
/*******************************************************/

       if(cero==1)
       {
                muestra(); /*llamada función muestra*/

    }

/*******************************************************/
/*******************************************************
búsqueda del máximo AMPLITUD DE LA SEÑAL DE SALIDA.
T2CNT esta sincronizado con la señal de referencia, así que aproximadamente a
PERIOD2/4-> 90º o 3*PERIOD2/4-> 270º está nuestro máximo
*******************************************************/

       if(t==MUESTRA)
       {

        contad++;
       if (contad<1000)
       {


              if(refmax_pos<=ref)
                  {
                       refmax_pos=ref;
                   }
                  if(refmax_neg>=ref)
                   {
                        refmax_neg=ref;
                   }
               }
              else
               {
               contad=0;
                     maxi=(refmax_pos+refmax_neg*(-1))/2;


                     if(maxi>512)  maxi=512;
                     bfy=0.01224*maxi+0.0339;   /*para pasar de voltios pico a
campo amplitud*/

                     absil= int_lazo( bfx, bfy, blx, bly); /*valor de la
intensidad del lazo*/

                     maxi=385.024*absil; /*convertir la intensidad del lazo en
tensión */
```

```
                    if(maxi>512) maxi=512; /*asegurar que sobrepasa este
valor, protección del circuito*/
                        /*borrar*/

                    refmax_pos=0;
                    refmax_neg=0;
                    calmax=1; /*activar creación vector senor*/
                    t=0;
                }



      }

/*********************************************************/
/*********************************************************/
/*creación del vector seno con las variables max */
/*********************************************************/
      if(calmax) /* calmax=1 calcula vector)*/
      {
         for(n=0;n<360;n++)
         {
                    if(n+desf<360)
                    {
                          senor[n+desf]=maxi*seno[n];     /*angulo
comprendido entre desf y 360*/
                    }
                    else
                    {
                          senor[(n+desf)-360]=maxi*seno[n];    /*comprendido
entre 0 y desf*/
                    }
         }
         calmax=0;/*finalizacion de la tabla*/

         cero=1;


    }
/*********************************************************/



ADCTRL1 |= 0x0100;   /*activar conversor*/
      }
} /*fin rutina de interrupción */

/*********************************************************
generador de tiempos para la construcción de la onda senoidal y pausa de la
onda para muestreo de la señal de referencia
*********************************************************/

interrupt void ONDA_SEN(void)   /* Para construir la señal senoidal*/
{
 if((PIVR-0x002B)==0) /*verifica interrupción Compara Timer2 */
  {
```

```
      EVIFRB=T2PINT;       /*habilita la interrupción escribiendo un 1*/
      if(tz<20)
            {     tz++;
                  if(tz==20)
                        {     cero=1;
                              tz=0;
                              }
            }
      if (t<MUESTRA)
        {
            t++;      /*bandera para escanear la referencia*/
            if(t==MUESTRA)
            {
                  adc2=0; /*inicializar valor muestreo*/
                  y=0; /* el inversor apagado*/
            }
        }

  }
} /*fin rutina de interrupción*/

/*********************************************************/
/*********************************************************/
/* INICIO PROGRAMA PRINCIPAL*/
/*********************************************************/

void main(void)
{
 asm ("  clrc XF");/*Pone 0 en la salida XF */
 asm (" setc INTM"); /*Deshabilita todas las interrupciones
            */
 asm (" clrc SXM");/*Borra el bit de extensión de signo*/
 asm (" clrc OVM");/*Borra bit de modo de desbordamiento*/
 asm (" clrc CNF");/*Configura B0 como memoria de datos */

 WDCR=((WDDIS<<6)+(WDCHK2<<5)+(WDCHK1<<4)+(WDCHK0<<3)+WDSP);
                              /*Inicializa el registro WDCR   */

 SCSR = ((CLKSRC<<14)+(LPM<<12)+ILLADR); /* Inicializa SCSR
      */

 out_wsgr((BVIS<<9)+(ISWS<<6)+(DSWS<<3)+PSWS);
                  /* Función externa para configurar WSGR        */

 OCRB = ((OCRB9<<9)+(OCRB8<<8)+
               (OCRB7<<7)+(OCRB6<<6)+(OCRB5<<5)+(OCRB4<<4)+
               (OCRB3<<3)+(OCRB2<<2)+(OCRB1<<1)+OCRB0);
                              /* Inicializa  OCRB      */


 OCRA = ((OCRA15<<15)+(OCRA14<<14)+(OCRA13<<13)+(OCRA12<<12)+
          (OCRA11<<11)+(OCRA10<<10)+(OCRA9<<9)+(OCRA8<<8)+
          (OCRA7<<7)+(OCRA6<<6)+(OCRA5<<5)+(OCRA4<<4)+
          (OCRA3<<3)+(OCRA2<<2)+(OCRA1<<1)+OCRA0);
                        /* Inicializa OCRA       */

 GPTCON=((GPTCON_T2TOADC<<9)+
          (GPTCON_T1TOADC<<7)+
          (GPTCON_TCOMPOE<<6)+
```

```
          (GPTCON_T2PIN<<2)+
          (GPTCON_T1PIN)); /*   Inicia el registro de control
                    del temporizador  */


PBDATDIR = 0x0000;    /*configure IOPB..B7 como entradas*/


T1PR = PERIOD;              /*configura el periodo del
     temporizador T1    */
T1CNT= 0x0000;              /* valor inicial del contador
     */

T1CON=((T1CON_FREESOFT<<14)+
           (T1CON_TMODE<<11)+
           (T1CON_TPS<<8)+
           (T1CON_TCLKS<<4)+
           (T1CON_TCLD<<2)+
           (T1CON_TECMPR<<1)); /*Configura el registro de
                         control del temporizador 1*/


T2PR = PERIOD2;             /*configure el periodo del
            temporizador T2     */
T2CNT= 0x0000;              /*valor inicial del contador
            */

T2CON=((T2CON_FREESOFT<<14)+
           (T2CON_TMODE<<11)+
           (T2CON_TPS<<8)+
           (T2CON_TCLKS<<4)+
           (T2CON_TCLD<<2)+
           (T2CON_TECMPR<<1)+
          T2CON_SELT1PR);      /*configuración del
                         registro timer 2*/




ADCTRL1= ((SOFTFREE<<14)+(ADCIMSTART<<13)+(ADC2EN<<12)+
            (ADC1EN<<11) +(ADCCONRUN<<10) +(ADCINTEN<<9)+
            (ADCINTFLAG<<8)+(ADC2CHSEL<<4)+(ADC1CHSEL<<1));
                  /* Configura registro de control 1 de ADC*/

ADCTRL2= ((ADCIM<<14)+(ADCINTPRI<<11)+
           (ADCEVSOC<<10)+(ADCEXTSOC<<9)+ADCPSCALE);
                  /*Configura registro de control 2 de ADC*/

asm (" LACL 7036h");      /* Vacío las FIFO */
asm (" LACL 7036h");
asm (" LACL 7038h");
asm (" LACL 7038h");

EVIFRA=0xFFFF;       /* Registro de flags del grupo A     de
                  EV*/

EVIFRB=0xFFFF;       /* Registro de flags del grupo B     de
                  EV*/
```

```
 EVIFRC=0xFFFF;     /* Registro de flags del grupo C      de
                   EV*/


 EVIMRA=((T1OFINT<<10)+
            (T1UFINT<<9)+
            (T1CINT<<8)+
            (T1PINT<<7)+
            (CMP3INT<<3)+
            (CMP2INT<<2)+
            (CMP1INT<<1)+
            (PDPINT)); /*Registro de máscara del grupo A del
             EV*/

 EVIMRB=((T2OFINT<<3)+
          (T2UFINT<<2)+
          (T2CINT<<1)+
          (T2PINT));  /*Registro de máscara del grupo B del
             EV*/

 EVIMRC=((CAP3INT<<2)+
              (CAP2INT<<1)+
              (CAP1INT)); /*Registro de máscara del grupo C
                  del EV*/

 IFR=0xFFFF; /*Borra todas las interrupciones pendientes */

 IMR=((INT6<<5)+
       (INT5<<4)+
       (INT4<<3)+
       (INT3<<2)+
       (INT2<<1)+
       (INT1));      /*Registro de máscaras de interrupción
      */
/*********************************************************/
/*Programa generación tabla del seno */
/*********************************************************/

for(n=0;n<180;n++)
 {
      seno[n]=sin(pi*n/180);      /*el valor de lo almacenado va desde 0
hasta n-1*/
     seno[180+n]=-seno[n];
  }

 /*********************************************************/
medperiod=PERIOD2/2;

calculos (&bfx,bfy,&blx,&bly);

asm (" clrc INTM");  /*Habilita todas las interrupciones
             */

 T1CON=T1CON+(T1CON_TENABLE<<6); /*Habilita el GPT1 */
 T2CON=T2CON+(T2CON_TENABLE<<6); /*timer 2 funcionando */

 t=MUESTRA; /* busca el cero ref al principio */
```

```
while(1){


       err=io-senor[n];  /*error de seguimiento*/


     if(y)   /* y=1 on inversor, y=0 off inversor */
        {
        if(err>5)
        {
                asm ("clrc XF"); /*XF es 0*/
                         /*PDDATDIR= 0xFF00;*/
        }

        else if(err<-5)
            {
               asm ("setc XF"); /*XF es 1*/
                                /*PDDATDIR= 0xFFFF;*/
            }
      }
   }
}
/*********************************************************/
/* FIN DEL PROGRAMA PRINCIPAL */
/*********************************************************/
/*********************************************************/
/* FUNCIÓN MUESTRA */
/*********************************************************/

void muestra(void)
{

    adc1=adc2;
    adc2= ref;


          if ((adc1<0 & adc2>0)|(adc1>0 & adc2<0))
          {
             ip=0;
              if(adc2>0) /*positiva*/     /*adc2>0*/
              {

                   T2CNT= 0;  /*desfases entre 0 y 180 grados*/
                   cero=0;
           y=1; /*activar inversor*/

                   }

        }  /*fin de if de busqueda de cero*/

} /*fin de muestra*/
```

6 Calculos.c

Rutina para calcular la frecuencia y amplitud de de la señal que se inyectará en el lazo a partir de los datos suministrados por los sensores.

```
/*******************************************************
 cálculos de la frecuencia y amplitud de la señal del lazo
 *******************************************************/


 #define A_LINEA    0.59         /*Altura línea*/
 #define A_LAZO     0.2          /*Altura lazo*/
 #define S_LINEA    0.72         /*Separación conductores
                      línea*/
 #define S_LAZO     0.67         /*Separación conductores
                      lazo*/
 #define XLC        -0.9         /*Posición X del lector
                             campo*/
 #define YLC        0.53         /*Posición Y del lector
                             campo*/
  #define XPC              -0.50  /*Posición X del punto de
                             mitigación*/
 #define YPC        .03          /*Posición Y del punto de
                             mitigación*/

/*función para calcular la amplitud de la intensidad del lazo*/
/*Valor de entrada a la función: B(campo). Valor de salida: I del lazo*/
void calculos (float *pbfx,float bfy,float *pblx,float *pbly)
{
      float hf=A_LINEA;         /*altura conductores  linea*/

      float hl=A_LAZO;          /*altura conductores lazo*/
      float df=S_LINEA;         /*separacion conductores
                      linea*/
      float dl=S_LAZO;          /*separacion conductores
                      lazo*/
      float xp=XLC,yp=YLC;      /*posicion sensor de campo*/
      float xc=XPC,yc=YPC;      /*posición punto a
                             compensar*/

/*definimos las variables intermedias o auxiliares*/

      float xf1,yf1,xf2,yf2;
      float xl1,xl2,yl1,yl2;
      float aux,aux1,aux2,aux3,aux4;
      float bfx,blx,bly;
      float absif;

/* calculamos las posiciones de los conductores de linea y lazo*/
            xf1=-df*0.5;
            yf1=hf;
            xf2=df*0.5;
            yf2=hf;
            xl1=-dl*0.5;
            yl1=hl;
            xl2=dl*0.5;
            yl2=hl;

/* dicha amplitud de campo*/

/*aux1=(xp-xf1)^2+(yp-yf1)^2;*/

      aux1=(xp-xf1)*(xp-xf1)+(yp-yf1)*(yp-yf1);
```

```
/*aux2=(xp-xf2)^2+(yp-yf2)^2;*/

      aux2=(xp-xf2)*(xp-xf2)+(yp-yf2)*(yp-yf2);

      aux=(xp-xf1)/aux1-(xp-xf2)/aux2;

      absif=(bfy/(0.2*aux));
      if(absif<0) absif=-absif;   /*valor absoluto*/

      bfx=0.2*absif*((yf1-yp)/aux1-(yf2-yp)/aux2);

/* Calculamos la amplitud de la corriente por el lazo*/
/*aux3=(xc-xl1)^2+(yc-yl1)^2;*/

      aux3=(xc-xl1)*(xc-xl1)+(yc-yl1)*(yc-yl1);

/*aux4=(xc-xl2)^2+(yc-yl2)^2;*/

      aux4=(xc-xl2)*(xc-xl2)+(yc-yl2)*(yc-yl2);

      blx=0.2*((yl1-yc)/aux3-(yl2-yc)/aux4);
      bly=0.2*((xc-xl1)/aux3-(xc-xl2)/aux4);
/*retornamos los valores de las variables locales a las variables exteriores
por medio de los punteros*/
      *pbfx=bfx;
      *pblx=blx;
      *pbly=bly;


} /*fin de calculos*/

/*calculo de la intensidad del lazo a partir de los parámetros de campo de la
línea a compensar y del propio lazo*/
float int_lazo(float bfx,float bfy,float blx,float bly)
{


      /*absil=abs((-blx*bfx-bly*bfy)/(blx^2+bly^2));*/
     float absil;
      absil=(blx*bfx+bly*bfy)/(blx*blx+bly*bly); /* intensidad el lazo */
      if(absil<0) absil=-absil; /* valor absoluto */
     return(absil);
```