8 LAZO ACTIVO. PROGRAMA DSP_LAZO

8.1 Introducción

Este es el programa para la compensación del campo producido por una línea monofásica al cual se le hace circular una intensidad **I**, desconocida para el sistema, y el cual debe de calcular para poder determinar la intensidad que debe circular por la línea del lazo y así compensar el campo en un punto del espacio.

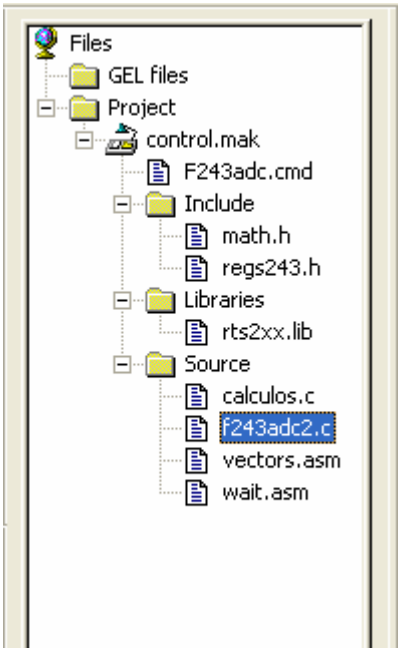

Figura 8.1. Code Composer. Ventana de ficheros

La estructura del programa se divide, según figura 8.1, el los siguientes ficheros:

- F243acd.cmd: fichero donde se especifica al compilador las divisiones de memoria o mapa de memoria del DSP.
- math.h: es la librería de ecuaciones matemáticas.
- regs243.h: es el fichero donde incluye las direcciones de los diferentes registros de que consta el DSP.
- rts2xx.lib: es la librería de interpretación de comandos específicos del DSP.
- calculos.c: es el fichero en C donde se incluye las funciones específicas para el cálculo de la intensidad de la línea a partir del campo como la intensidad necesaria a aplicar al lazo para la compensación.
- f243adc2.c: es el fichero en C donde se encuentra el programa que gestiona el inversor y realiza las lecturas tanto del campo magnético que produce la línea como de la intensidad que circula por el lazo.
- vector.asm: el fichero que incluye las direcciones de las rutinas de interrupción.
- wait.asm: fichero del vector de espera.

Los ficheros principales son el “f243adc2.c” y el “calculos.c” que son donde están los códigos necesarios para el funcionamiento de los dispositivos. A continuación se explicará los diferentes núcleos de que consta cada uno de los ficheros mencionados.

8.2 El fichero f243acd2.c

Este es el fichero donde se encuentra el programa principal. En el se encuentra:

- La inclusión de los ficheros “.h”.
- La definición de los valores de los registros.
- La rutina de interrupción que controla los tiempos de muestreo para la conversión, búsqueda del paso por cero, valor máximo de la intensidad y construcción del desfase de la señal.
- La rutina de interrupción que controla la fase de la señal de referencia interna.
- La carga en los registros de los valores definidos al principio del programa.
- El bucle que chequea el error y actúa sobre el inversor.

Respecto a los ejemplos de temas anteriores, donde se ha ido modelando la construcción de este programa, aquí se han producido diferentes cambios, bien en la situación de diferentes rutinas del programa, como también la inclusión de numerosas instrucciones para mejorar la funcionalidad del mismo. Así que en lo que respecta a su explicación, se hará inciso solo en aquellos puntos que presente novedad y que sean importantes para la compresión de su funcionamiento. En el anexo f243acd.c encontraremos el programa completo.

En esta primera parte se utilizan rutinas del preprocesado del compilador ,con objetivo de no trasladar los valores numéricos y si un conjunto de caracteres fácilmente reconocibles para hacer más legible el código. Además están los `include` para la carga de la librería matemática y los registros del DSP.

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

Acabado la definición de los parámetros de los registros, viene a continuación la definición y creación de los datos, parámetros y variables auxiliares para hacer posible el correcto funcionamiento del código. A continuación se describen sus características:

- PERIOD1: valor numérico para calcular el periodo que actua el conversor.
- PERIOD2: valor numérico para calcular el periodo de formación que determinará la forma de onda de la señal de referencia.

- MUESTRA: valor numérico, es el tiempo que es refrescada la amplitud de la señal de referencia.
- DESFASE: valor numérico obtenido experimentalmente. Es el desfase que introduce el sensor de campo utilizado en este experimento.
- pi: número pi, dato.
- seno[360]: tabla donde se almacena el valor del seno, por unidad y sin escalar.
- Senor[360]: valor donde se almacena el valor del seno desfasado.
- maxi: variable donde se almacena el valor máximo de la amplitud de la señal de referencia, la obtenida de la conversión y posteriormente la obtenida de los cálculos. Al final es la que se utiliza para ser el factor de escala para construir la señal de referencia desfasada.
- ADC0_result, ACD1_result: variables auxiliares para almacenar los datos obtenidos de la conversión del ACD del DSP.
- medperiod: semiperiodo de la señal de referencia a la misma escala utilizada en PERIOD2.
- desf: como variable es el DESFASE.
- ref: valor obtenido de la conversión de la señal de referencia.
- io: valor obtenido de la conversión de la señal de salida.
- refmax_pos, refmax_neg: variable auxiliar donde se van almacenando los diferentes valores, positivos y negativos, obtenidos de la conversión de la señal de referencia (línea) para encontrar la amplitud máxima de la señal
- err: es la variable donde se almacena la diferencia entre la señal de salida o del lazo y la señal de referencia creada en el DSP a partir de los datos obtenidos de las lecturas externas.
- y, cero, calmax: son utilizadas como “banderas” para activar “etapas” en las líneas de código dependiendo de los requerimientos que se precise.
- n, ip ,t, tz, contad: variable auxiliar de uso general.
- adc1, adc2: variables auxiliares para almacenar los valores n-1 y n de la iteración en la búsqueda del paso por cero
- t1: almacena el valor actual del temporizador 2 para poder determinar el valor que tiene la señal de referencia creada en la tabla
- blx, bly, bfx, bfy: variables auxiliares utilizadas para el cálculo de los valores de intensidad de la línea
- absil: intensidad que debe circular por el lazo.

```
#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;
```

Una vez definidas las variables en función del tipo de dato que va a almacenar, se registra las funciones para poder llamadas desde el programa. Las creadas a partir de las rutinas de interrupción *ADC_ISR* y *ONDA_SEN.* La primera controla la conversión y captura de datos y la segunda la gestión de la señal de referencia interna. Las dos siguientes son funciones internas del programa para controlar las interrupciones espurias y el control de accesos respectivamente. Por último aparece la función `muestra` que forma parte un conjunto de rutinas para muestrear la señal de referencia externa, o sea, la línea que queremos mitigar. La función `int_lazo` nos da la intensidad del lazo que debe circular.

```
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 */
}
```

A diferencia de los programas vistos en el tema 6, aquí esta rutina cobra una importancia vital al centrarse gran parte del código para obtener la información de la línea y calcular la referencia interna. Lo normal es crear

rutinas de interrupción con cortas instrucciones para no obstaculizar el funcionamiento del programa, pero a las implementaciones que se han realizado para llegar a tal fin fueron infructuosas, teniendo que acumular mucho del código en la propia rutina. Se vio que esta rutina se dedicaba casi exclusivamente al escaneo de los puertos de conversión con el objetivo de obtener los datos de la referencia externa y de la señal de se crea en el inversor. Ahora, aparte de lo anteriormente, se le añade nuevas funciones de gran importancia como son la búsqueda del paso por cero, la búsqueda de la máxima amplitud, la obtención de la intensidad del lazo (haciendo una llamada a la función correspondiente) y la creación de la señal de referencia interna. Cada grupo será descrito con más detalle posteriormente.

```
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"); */
```

Las siguientes dos ecuaciones se utilizan para la el valor, en tensión de las lecturas de las señales de referencia procedente del lector de campo, `ref`, y de la señal del inversor, `io`. A diferencia, otra vez, de los programas vistos en el tema 6, el intervalo en que están comprendidos estos valores va desde 0 a 1023, o sea, 1024 valores. Ahora bien, debido a un offset que aparece en la conversión y que es interno, procedente del DSP, y externo a la circuitería de la placa de adaptación de señales, se ha tenido que cambiar el valor del 0 de la señal (512), pasar para `io` a 565 y para `ref` 555. Se calcula de forma experimental.

```
      io= (ADC0_result-565);
      ref= (ADC1_result-555);
```

Las siguientes instrucciones, las encerradas en el if, tienen el objetivo de situar, en función del temporizador, en que parte del periodo del seno se encuentra para poder obtener a partir de la tabla dicho valor. Para ello primeramente se obtiene el valor del tiempo en función de una escala de grados, que posteriormente se calcula dividiendo el tiempo actualmente empleado por el resultado de la ecuación anterior la fase actual del seno.

```
      if(y)
      {
            n=PERIOD2/360;    /*ángulo de la onda*/
```

```
        n=t1/n;
    }
```

Llegado a este punto comienza los tres bloques que se construyen dentro de la rutina. En este primero a su vez se ha creado una función externa (que también podría estar internamente incorporada). El objetivo de esta función es la búsqueda del paso del cero de la señal de referencia. Más adelante se explicará su funcionamiento.

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

Este segundo bloque representa la búsqueda de la amplitud máxima de la señal de referencia, la de la línea que se quiere mitigar y posteriormente calcular la intensidad que circula por la línea, introducir estos datos en la ecuación correspondiente y extraer el valor en tensión que hay que multiplicar la tabla de la función seno para tener la tabla del seno desfasado. Es necesario trabajar en tensiones pues todas las lecturas que hace el DSP las hace en tensiones, por tanto los valores en intensidad hay que convertirlos, aplicando la ecuación V=R*I, donde R es el valor de la resistencia que se pone en serie con el lazo para obtener la I del lazo, aplicando a su vez dicha ecuación. V es el valor que nos aporta el DSP.

Activado este bloque por medio de la bandera MUESTRA, llegamos a un grupo de instrucciones condicionales (if) con la función de buscar el valor de pico a partir de un cierto número de muestras. Como se desconoce a partir de que punto de la señal senoidal comienza, lo que se hace es realizar un muestreo amplio para asegurar haber recorrido el 100% de un periodo. Con ello aseguramos tener los 2 máximos. Estos se van almacenando en las variables refmax_pos para el máximo positivo, refmax_neg para el negativo.

Una vez muestreado la señal hay que obtener el máximo haciendo una media entre máximo positivo y el negativo. Con ello se elimina en parte la aparición de algún pico como también del offset (si este todavía existe). A continuación se calcula la componente bfy del campo. Posteriormente por medio de la función int_lazo se obtiene el valor de la intensidad del lazo. Por último se convierte este valor en tensión para su seguimiento por el DSP. Existe una pequeña rutina para salvaguardar el circuito del montaje, pues si sobrepasa la variable maxi el valor de 512, la señal que se introduce en el inversor es una señal senoidal truncada por lo que aumentaría mucho la intensidad y quemaría los circuitos (existe unos fusibles en el montaje para que esto no pase).

```
    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;
              }



      }
```

Ya llegado a este punto, solo queda crear la tabla del seno desfasado, senor, y activar el inversor para su funcionamiento. La tabla del senor se hace en dos pasos. En primer lugar se multiplica la amplitud por el valor unitario del seno, seno[n], y este se desfasa la cantidad indicada en la variable `desf`, almacenando el valor en la tabla `senor`. Posteriormente se introduce el resto de valores que no fueron posibles en el primer paso. Se activa la búsqueda de cero y con ello está ya operativo el sistema.

```
      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 interrupción*/
```

La segunda rutina de interrupción tiene la funcionalidad de controlar la base de tiempos de la señal de referencia, creada internamente, para que está sea lo más próxima posible a 50 Hz. En este existe dos bloques diferenciados, el bloque condicional sobre `tz` y el que corresponde a `MUESTRA`. El bloque condicional de `tz` se utiliza para sincronizar cada 20*0.02 segundos la señal senoidal creada con la señal de la línea. El motivo es obvio al ser la creada una señal de 50 Hz exactos y la de la línea oscilar entre 49,5 y 50,5 Hz. De esta forma se va refrescando la onda minimizando el desfasado en el tiempo. El segundo bloque tiene como objetivo la de refrescar tanto la búsqueda de cero como la amplitud, o sea, activar todas las funciones de la interrupción `ADC_ISR`.

```
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*/
```

El programa principal. Al estar explicado ya no se entra en profundidad.

```
void main(void)
{
. . .
```

El bloque de creación de la tabla del seno. Este, respecto a otros programas anteriores, ha sido simplificado dejando solamente la parte correspondiente a la operación de la función seno. Este se hace, al igual que el senor, en dos partes, calculando la mitad de los valores (positivos) y el resto multiplicando por –1 la tabla anteriormente creada.

```
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;
```

Haciendo una llamada a la función *calculos* se extraen los valores bfx, blx y bly del campo pues bfy la obtenemos de la lectura del lector de campos.

```
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 */
```

El bucle que comanda el inversor. Obtenido el error, que aquí se ha escalado de forma diferente, se compara con la banda de error que se desea, entre 5 y –5, y con ello se van conmutando los transistores.

```
 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 main() */
```

La función `muestra` tiene la finalidad de buscar el paso por cero de la señal de referencia (la señal procedente de la línea). Para ello lo que se hace es ir almacenando las muestras *n* y las *n-1* con el objetivo de compararlas y ver su evolución. Encontrado una variación de signo en la comparación de adc1 (n-1) y adc2 (n) nos encontramos ante un paso por cero y es ahí donde se activa la señal haciendo `T2CNT=0` e `y=1`. Como se aprecia en la condición **`((adc1<0 & adc2>0)|(adc1>0 & adc2<0))`** esta preparada para buscar tanto de un paso negativo a positivo, 0 grados, como de positivo a negativo, 180 grados. No se ha utilizado el de 180 grados por simplicidad del programa.

```
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*/
```

8.3 El fichero calculos.c

Este es el fichero donde se calculan los valores de campo de la línea, la intensidad que circula por la línea y la intensidad que es necesaria circular por el lazo. El fichero se divide en dos funciones:

- `calculos`: devuelve los valores de campo magnético (x, y) tanto de la línea como del lazo. Los datos los obtiene del propio programa.
- `int_lazo`: devuelve la intensidad que debe circular por el lazo en función de los valores de campo introducidos.

Para el cálculo de la intensidad del lazo necesaria se debe de suministrar dos tipos de datos: la posición de los elementos (línea y lazo) y el valor de la componente del campo Y, Bfy. Utilizaremos las siguientes ecuaciones para calcular los valores deseados.

Partiendo del conocimiento de la posición del lazo y la línea

hf=0.59	altura de los conductores de la línea
hl=0.2	altura de los conductores del lazo
df=0.72	separación entre los conductores de la línea
dl=0.67	separación entre los conductores del lazo
xp=-0.9;yp=0.53	posición sensor de campo
xc=-0.5;yc=0.03	posición punto de compensación

Calculamos siguientes posiciones

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

Con esto y el valor del campo, Bfy procedemos a calcular la componente x del campo

$$aux1 = (xp - xf1)^2 + (yp - yf1)^2$$

$$aux2 = (xp - xf2)^2 + (yp - yf2)^2$$

$$aux = \frac{(xp - xf1)}{aux1} - \frac{(xp - xf2)}{aux2}$$

La intensidad de la línea, en su valor absoluto.

$$absIf = abs\left(\frac{Bfy}{(0.2 \cdot aux)}\right)$$

El valor del campo magnético, producido por la línea, en su componente x.

$$Bfx = 0.2 \cdot absIf\left(\frac{(yf1 - yp)}{aux1} - \frac{(yf2 - yp)}{aux2}\right)$$

Calculado el valor Bfx y conocido el Bfy se procede al cálculo de la intensidad que debe de circular por el lazo.

$$aux3 = (xc - xl1)^2 + (yc - yl1)^2$$

$$aux4 = (xc - xl2)^2 + (yc - yl2)^2$$

Valor de la componente del campo magnético x del lazo.

$$blx = 0.2 \cdot \left(\frac{(yl1 - yc)}{aux3} - \frac{(yl2 - yc)}{aux4} \right)$$

Valor de la componente del campo magnético y del lazo.

$$bly = 0.2 \cdot \left(\frac{(xc - xl1)}{aux3} - \frac{(xc - xl2)}{aux4} \right)$$

Valor absoluto de la intensidad que circula por la línea.

$$absIl = abs\left(\frac{(-blx \cdot Bfx - bly \cdot Bfy)}{blx^2 + bly^2} \right)$$

Obtenido el valor absoluto de la intensidad, la función int_lazo procede a devolverlo a través de la instrucción return

```
return(absil);
```

El programa completo se puede ver en el anexo *Calculos.c*