Sistemas Digitales II

Universidad Politécnica de Madrid
Departamento de Ingeniería Electrónica
E.T.S.I. de Telecomunicación

Memoria del proyecto desarrollado en Sistemas Digitales II (SDII)

Curso 2014/2015

Título del proyecto desarrollado
"Sistema de Visualización de Señales en el Dominio de la Frecuencia basado en el MCF5272"

Autores (orden alfabético): Francisco García de la Corte

Carlos Santos Rancaño

Código de la pareja: LT-07

ÍNDICE GENERAL

1 Introducción

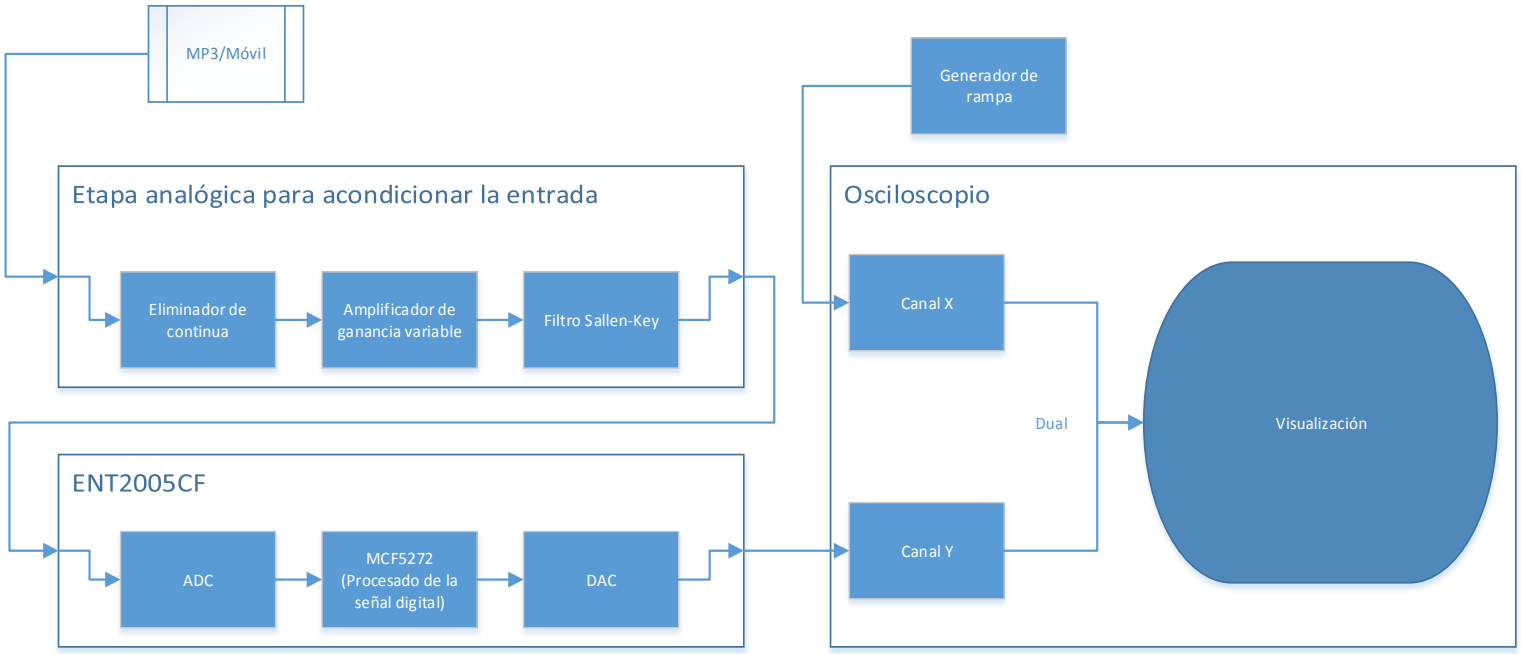
El proyecto desarrollado pretende obtener la información espectral de una señal de entrada de audio mediante la realización de la DFT de la misma. Para ello contamos con dos subsistemas, uno analógico para acondicionar la señal de entrada y uno digital que será el encargado de realizar la DFT.

El subsistema analógico consta de dos partes, la primera prepara la señal del MP3/Móvil variando su amplitud y filtrando frecuencias para que el subsistema digital pueda trabajar con ella. La segunda parte se encargará de generar una señal en diente de sierra para sincronizar los dos canales del osciloscopio.

El subsistema digital se encarga de calcular la DFT muestreando la señal proveniente del subsistema analógico y posteriormente procesándola, para ello utilizaremos el microcontrolador MCF5272 integrado en la plataforma ENT2005CF.

Por último podremos visualizar la DFT de la señal de entrada en el osciloscopio gracias a la señal de sierra generada.

2 Diagrama de subsistemas



Empezando el diagrama, la señal a tratar proviene de un MP3 o un móvil reproduciendo música, que entrará en la etapa analógica. En esta etapa, acondicionamos la señal para que la etapa digital no tenga problemas a la hora de procesar las muestras: se elimina la componente continua, se amplifica hasta entrar en un rango de valores bien distinguibles y se filtra la señal para que no haya solapamiento espectral.

Después de la etapa analógica, la señal entra en el entrenador, y el módulo ADC muestrea la señal, cuantificando los valores que ésta toma para su tratado digital. El microcontrolador MCF5272 hace las operaciones necesarias con las muestras para obtener la información espectral de la señal (DFT), escala estos valores y los envía al DAC para obtener una señal analógica y terminar la etapa del entrenador.

Del entrenador salen los valores de la DFT que entran en el canal 2 del osciloscopio, y un circuito aparte genera una rampa que entra en el canal 1, de modo que con la opción "dual" la rampa hará el barrido del eje X y los valores de la DFT de eje Y.

3 Descripción del subsistema Hardware

3.1 Etapa analógica de entrada

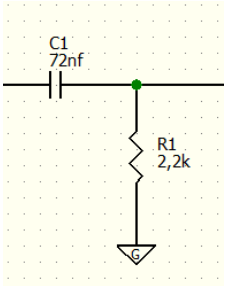
Antes de conectar la señal del MP3 a la plataforma ENT2005CF ha de pasar a través de una primera etapa analógica consistente en dos filtros y un amplificador de ganancia variable.

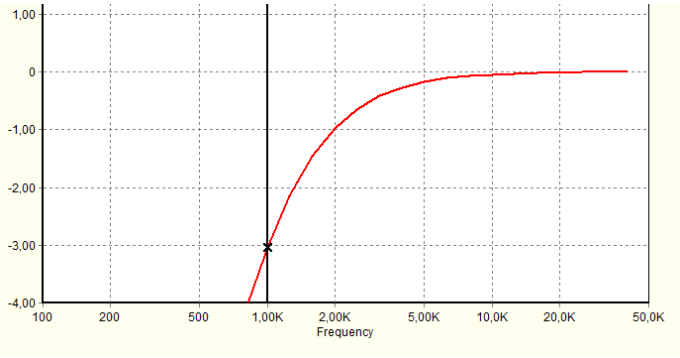
3.1.1 Filtro eliminador de continua

Este filtro RC se encarga de eliminar la componente continua de la señal, para ello decidimos que su frecuencia de corte fuese 1KHz.

Elegimos los valores del condensador y de la resistencia según la fórmula de la frecuencia de corte y después redondeando a valores comerciales.

$$fc = \frac{1}{2\pi * R * C}$$

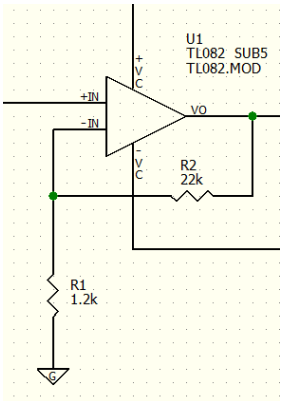


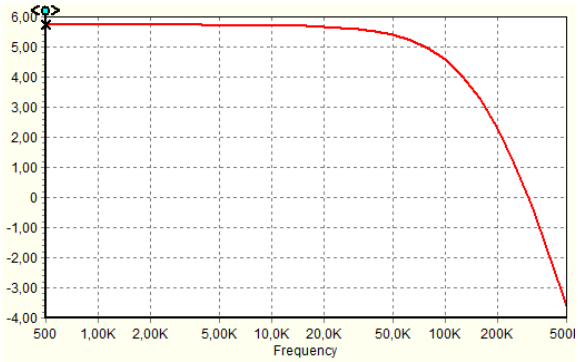


3.1.2 Amplificador de ganancia variable

Se trata de un amplificador lineal para todas las frecuencias cuya ganancia se puede variar usando un potenciómetro en la realimentación negativa, siendo la ganancia $G = \frac{R1+R2}{R1}$

Elegimos el potenciómetro (R2 en este caso) de 47k de forma que variándolo pudiésemos obtener ganancias de entre 12dB y -15dB o lo que es lo mismo obteniendo hasta 4V si la entrada son 100mV de amplitud.



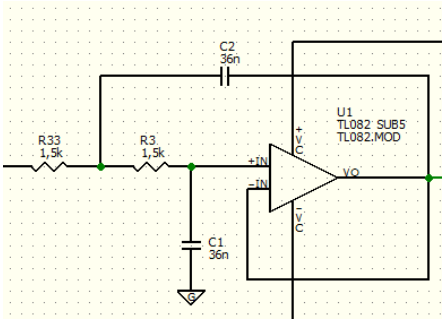


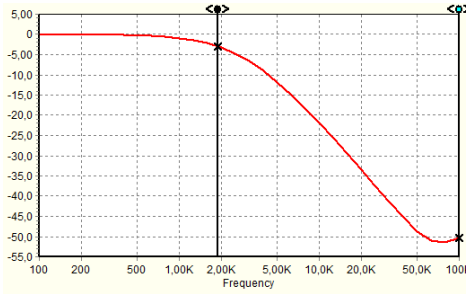
3.1.3 Filtro Sallen-Key de segundo orden

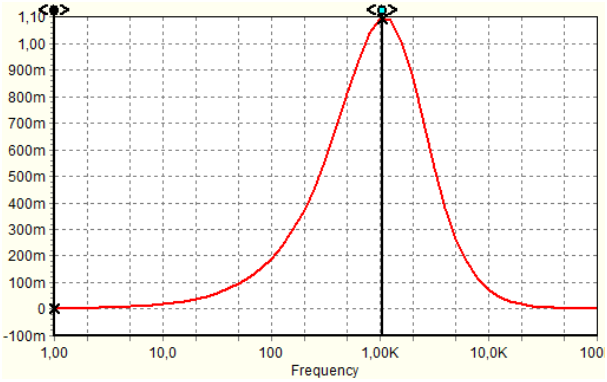
Este filtro debe estar diseñado para cortar frecuencias mayores a 2kHz. En la práctica decidimos eliminar la rama de realimentación para que fuese un filtro sin ganancia y los cálculos fuesen más fáciles

además de que así podemos variar la ganancia de la etapa entera con el potenciómetro del amplificador anterior.

Para hallar los valores de resistencias y condensadores usamos la documentación disponible en el Moodle de la asignatura de Circuitos Electrónicos y después ajustamos a los valores reales más apropiados según las simulaciones.





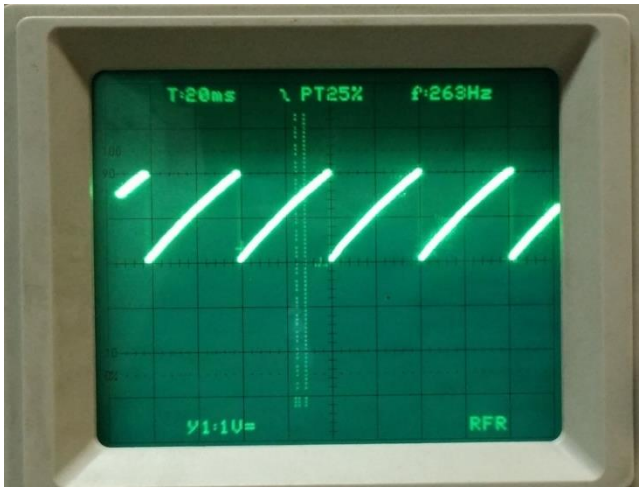


De este modo se nos queda la etapa analógica entera como un filtro paso banda cuya banda de paso se sitúa por encima de la contínua y por debajo de 2kHz, consiguiendo sin problemas 1,2V a la salida si la entrada son 100mV

3.2 Generador de rampa

Este módulo es necesario para poder sincronizar la información del canal Y del osciloscopio (Magnitud en dB) con el canal X (frecuencia).

El condensador se irá cargando con pendiente constante hasta que el bit de reset se ponga a '1' en cuyo caso la tensión bajará a 0V. Esto ocurre cada 40ms y la tensión en bornas del condensador debe llegar a 2V con lo que resolviendo las ecuaciones del circuito llegamos a que



$$Ir = \frac{Vb + 0{,}7}{R} = \frac{0.8625}{200k} = 4.3125\mu A$$

$$C = \frac{t * Ir}{Vout} = 86.25nF$$

Ajustamos el valor del condensador a 100nF ya que es un valor comercial fácil de encontrar. Rehaciendo las ecuaciones para este valor de condensador necesitábamos una resistencia de 172kΩ con lo que usando un potenciómetro de 200kΩ no tendremos ningún problema.

4 Descripción del subsistema Software

El software del proyecto hace la función principal, que es obtener la información espectral de la señal de audio. Cada rutina del programa tiene una función distinta: una interrupción obtiene las muestras, hace los cálculos necesarios (a veces llamando a otras subrutinas) y saca los resultados; una función actualiza los valores de los senos y cosenos a distintas frecuencias usados en los cálculos; otra función convierte los valores obtenidos en decibelios escalados para el DAC, y una rutina inicializa valores iniciales de variables y configura las interrupciones.

4.1 Proceso del programa principal (interrupción TIMER0)

Función rutina_tout0: entran las muestras del ADC y salen los valores de la DFT a mostrar.

Obtener muestra
Para cada componente de frecuencia de la DFT
  Acumular parte real e imaginaria, y desescalar

Actualizar valores de los punteros al seno y coseno para la próxima muestra

Si calculo = verdadero y es la muestra 79
  Para cada componente de frecuencia de la DFT
        Sumar partes real e imaginaria acumuladas y al cuadrado
  Convertir a decibelios
  Calculo = falso
  Contador de muestras = 0
  Contador de interrupciones = 0
  Reiniciar acumuladores

Si calculo = falso o no es la muestra 79
 Aumentar el contador de muestras

Si es la muestra 80
   Reiniciar acumuladores
   Número de muestra = 0

Mostrar componente de la DFT calculada (la misma durante 8 interrupciones)
Aumentar contador de interrupciones

Si contador de interrupciones = 160
  Reiniciar acumuladores
  Calculo = verdadero
  Contador de interrupciones = 0
  Contador de muestras = 0
  Activar bit del generador de rampa

Si contador de interrupciones no es 160
  Desactivar bit del generador de rampa

El proceso del programa principal es la rutina de atención a la interrupción del TIMER0, el cual es el encargado de obtener las muestras, hacer los cálculos necesarios (con ayuda de otras subrutinas secundarias) y sacar los datos finales. Esta rutina se ejecuta 4000 veces por segundo, por lo que actúa como un bucle.

4.1.1 Función para convertir valores a decibelios

Función conversión_dB: entran los valores lineales de la DFT y salen valores logarítmicos y escalados para el DAC.

```
Para cada componente de la DFT
  Si el valor se sale de rango de umbrales
    Asignar valor mínimo o máximo según no llegue o se salga del rango
  Si el valor está contenido en el rango de umbrales
   Calcular de qué nivel está más cerca
   Asignarle ese nivel
```

Esta función convertirá los valores calculados de la DFT según valores guardados en un array a su correspondiente valor en dB (valores reconocidos por el DAC) de otro array.

4.1.2 Función para refrescar los punteros a seno y coseno

Función actualiza_Índices: entran los punteros a seno y coseno y el contador de muestras, y salen los mismos punteros actualizados.

```
Si contador de muestras = 79
  Para  cada componente en frecuencia de la DFT
    Puntero a valores del seno hacia primer valor
    Puntero a valores del coseno hacia muestra 100 del seno
Si contador de muestras no es 79
  Para cada componente en frecuencia de la DFT
    Calcular posible desborde (si se pasa de las 400 muestras del seno)
    Sumar salto o desborde correspondiente a los punteros
```

Esta función cambiará la dirección a la que apuntan los punteros usados en el cálculo de la DFT, dependiendo de la componente en frecuencia. En la rutina de atención a la interrupción TIMER0 varían los punteros según el número de muestra, por lo que el objetivo de esta función se realiza conjuntamente entre la rutina principal y la función aquí descrita.

4.2 Función de inicio del programa

Función __init: inicializa todas las variables que no están inicializadas fuera de las funciones.

Inicialización de:
ADC/DAC
Dirección de vectores de interrupción
Interrupción TIMER0

Variables para las demás sub/rutinas:
Calculo = true
Contador de muestras = 0
Contador de interrupciones = 0
Para cada componente en frecuencia de la DFT
Valores DFT y acumuladores = 0
Punteros a seno y coseno a posiciones iniciales

Esta función se ejecuta nada más empezar el programa, y es necesaria para iniciar todas las variables a los valores iniciales.

5 Principales problemas encontrados

El primer problema que nos encontramos fue en la parte analógica, conseguimos demasiada amplitud saturando el amplificador del filtro Sallen-Key y recibiendo una señal más parecida a una señal cuadrada que a un seno. Esto lo resolvimos eliminando la ganancia del Sallen-Key de forma que la ganancia de toda la etapa analógica dependa exclusivamente del potenciómetro y de esta forma sea más fácil de manejar.

También nos encontramos errores en el cálculo de la DFT, por ejemplo valores erróneos, fallos en los algoritmos de cálculo… que solucionamos en las horas de laboratorio revisando el código y realizando las convenientes pruebas.

6 Manual de usuario

Lo primero de todo ha de ser alimentar la parte analógica con alimentación simétrica +-5Vy encender la plataforma ENT2005CF.

La entrada de audio proveniente del MP3/Móvil se conecta a la entrada de la etapa analógica y hay que ajustar la ganancia de la misma (variando el potenciómetro) hasta conseguir la amplitud deseada, la salida de la misma será la entrada analógica de la ENT2005CF. Posteriormente se digitalizará y se harán los cálculos pertinentes.

Para visualizar la DFT de una forma intuitiva conectaremos la salida de la ENT2005CF a un canal del osciloscopio y la señal diente de sierra de la etapa analógica en el otro canal. Preferiblemente se conectará la DFT en el canal 2 y el diente de sierra en el 1 de forma que al ponerlo en modo dual salgan los valores de la DFT en el eje Y.

Por último se pueden ajustar las escalas en ambos ejes del osciloscopio para poder visualizar la DFT mejor.

Este proyecto actualmente es la versión básica y no incluye ninguna mejora ni opciones de configuración, su funcionamiento se queda en lo descrito en el manual.

7 Bibliografía

ENUNCIADO DEL PROYECTO

Ha supuesto el pilar básico en este proyecto pues nos dio la estructura para comenzar y una propuesta temporal para la realización del mismo. Nos ha proporcionado las bases y bibliografía adicional así como tutoriales y teoría de utilidad.

DATASHEETS DE COMPONENTES

Para saber cómo conectar los componentes analógicos (sobre todo los transistores) recurrimos a las datasheets de los mismos (www.alldatasheet.com). También las utilizamos para conocer las entradas/salidas del entrenador.

PÁGINAS TUTORIALES SOBRE PROGRAMACION EN C

Ya que en un principio no estábamos muy familiarizados con este nuevo lenguaje a la hora de programar tuvimos que recurrir a tutoriales online (la gran mayoría de las veces a Stack Overflow) y para entender las librerías y poder modificar alguna nos beneficiamos de los tutoriales de la asignatura.

TEORÍA DE OTRAS ASIGNATURAS

Usamos los apuntes de otras asignaturas como ayuda adicional. Sobre todo usamos los apuntes de Teoría Digital de Señales para refrescar nuestros conocimientos sobre la idea de nuestro proyecto, la Transformada de Fourier.

Otra de las asignaturas a cuyos apuntes recurrimos fue Circuitos Electrónicos para la parte analógica y el diseño del filtro Sallen-Key.

8 ANEXO I: Código del programa del proyecto final

Ejemplo de código en C:

```
//Francisco García de la Corte
//Carlos Santos Rancaño

#include "m5272.h"
#include "m5272lib.c"
#include "m5272adc_dac(mod).c" //Librería modificada para que el ADC sea bipolar
#include "m5272gpio.c"

//Constantes para la configuración de la interrupción TIMER0
#define FONDO_ESCALA 0xFFF      // Valor de lectura máxima del ADC
#define V_MAX 5
#define V_BASE 0x40                     // Dirección de inicio de los vectores de interrupción
#define DIR_VTMR0 4*(V_BASE+5)                // Dirección del vector de TMR0
#define FREC_INT 4000                         // Frec. de interr. TMR0 = 4000 Hz (cada 0.25ms)
#define PRESCALADO 2                              //parece que el preescalado no hace nada
#define CNT_INT1 MCF_CLK/(FREC_INT*PRESCALADO*16)    // Valor de precarga del temporizador de interrupciones TRR0
#if CNT_INT1>0xFFFF
#error PRESCALADO demasiado pequeño para esa frecuencia (CNT_INT1>0xFFFF)
#endif
#define BORRA_REF 0x0002                // Valor de borrado de interr. pendientes de tout1 para TER0
#define FONDO_ESCALA 0xFFF      // Valor de lectura máxima del ADC
#define V_MAX 5
//Constantes para el cálculo de la DFT
#define NUM_MUESTRAS_PERIODO_10HZ 400 //Muestras del seno a 10Hz
#define ESC_ADC 819 //Valor máximo del DAX
#define N 80 //Usamos 80 muestras de la entrada
#define DESESCALADO 1024
#define N_FRECS 20 //Orden de la DFT: 20 muestras de la TF
#define NIVELES 32 //32 niveles para los umbrales
#define N_INT_DFT 160 //160 interrupciones para mostrar las 20 componentes de una DFT
```

```
#define N_INT_COMPONENTE 8 //Mostrar la misma componente de una DFT durante 8 interrupciones
//Variables necesitadas en el código
int j; //iterar en los umbrales y escalado
int distanciaArriba; //cuánto dista un valor del nivel de arriba
int distanciaAbajo; //cuánto dista un valor del nivel de abajo

int umbrales[NIVELES]={63095, 84924, 114304, 153849, 207075, 278715, 375140, 504923,
 679607, 914724, 1231182, 1657123, 2230422, 3002059 , 4040653 , 5438559 , 7320085,
 9852544 ,13261133, 17848959, 24023991, 32335339, 43522083, 58578997, 78845006,
 106122250, 142836338, 192252044, 258763624, 348285572, 468778561, 686492401};

int escalado[NIVELES]={122, 245, 368, 491, 614, 737, 860, 983, 1105, 1228,
 1351, 1474, 1597, 1720, 1843, 1966, 2088, 2211, 2334, 2457, 2580,
 2703, 2826, 2949, 3072, 3194, 3317, 3440, 3563, 3686, 3809, 3932};

int sinusoide10Hz[NUM_MUESTRAS_PERIODO_10HZ]={0, 12, 25, 38, 51, 64, 77, 89, 102, 115, 128, 140, 153, 166, 178,
 191, 203, 216, 228, 240, 253, 265, 277, 289, 301, 313, 325, 336, 348, 360, 371, 383, 394, 405, 416, 427, 438,
 449, 460, 470, 481, 491, 501, 511, 521, 531, 541, 551, 560, 569, 579, 588, 596, 605, 614, 622, 630, 639, 647,
 654, 662, 669, 677, 684, 691, 698, 704, 711, 717, 723, 729, 735, 741, 746, 751, 756, 761, 766, 770, 774, 778,
 782, 786, 790, 793, 796, 799, 802, 804, 806, 809, 810, 812, 814, 815, 816, 817, 818, 818, 819, 819, 819, 818,
 818, 817, 816, 815, 814, 812, 811, 809, 807, 804, 802, 799, 796, 793, 790, 786, 783, 779, 775, 771, 766, 761,
 757, 752, 746, 741, 736, 730, 724, 718, 712, 705, 698, 692, 685, 678, 670, 663, 655, 647, 639, 631, 623, 615,
 606, 597, 588, 579, 570, 561, 552, 542, 532, 522, 512, 502, 492, 482, 471, 461, 450, 439, 428, 417, 406, 395,
 384, 372, 361, 349, 338, 326, 314, 302, 290, 278, 266, 254, 242, 229, 217, 204, 192, 179, 167, 154, 142, 129,
 116, 103, 91, 78, 65, 52, 39, 27, 14, 1, -12, -25, -38, -51, -63, -76, -89, -102, -115, -127, -140, -153, -165,
 -178, -190, -203, -215, -228, -240, -252, -264, -277, -289, -301, -313, -324, -336, -348, -360, -371, -382,
 -394, -405, -416, -427, -438, -449, -460, -470, -481, -491, -501, -511, -521, -531, -541, -551, -560, -569,
 -579, -588, -597, -605, -614, -622, -631, -639, -647, -655, -662, -670, -677, -684, -691, -698, -705, -711,
 -718, -724, -730, -735, -741, -746, -752, -757, -762, -766, -771, -775, -779, -783, -787, -790, -794, -797,
 -800, -802, -805, -807, -809, -811, -813, -815, -816, -817, -818, -819, -819, -820, -820, -820, -819, -819,
 -818, -817, -816, -815, -813, -812, -810, -808, -806, -803, -800, -798, -794, -791, -788, -784, -780, -776,
 -772, -768, -763, -758, -753, -748, -743, -737, -731, -725, -719, -713, -707, -700, -693, -686, -679, -672,
 -665, -657, -649, -641, -633, -625, -616, -608, -599, -590, -581, -572, -563, -554, -544, -534, -524, -515,
 -504, -494, -484, -473, -463, -452, -441, -431, -420, -408, -397, -386, -375, -363, -351, -340, -328, -316,
 -304, -292, -280, -268, -256, -244, -231, -219, -207, -194, -182, -169, -156, -144, -131, -118, -106, -93,
 -80, -67, -55, -42, -29, -16};
```

```
int muestra; //Valor obtenido por el ADC

int dft[N_FRECS]; // = dftCos^2 + dftSin^2
int dftCos[N_FRECS]; //parte real
int dftSin[N_FRECS]; //parte imaginaria

int i, n, k; //Iteradores para bucles for
int cont160Inter;
int calculo; //Si calculo o no la DFT en función de si me quedan por sacar por pantalla

int salto[N_FRECS] = {5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190};

int *pSin[N_FRECS], *pCos[N_FRECS]; //Punteros al array del seno, a variar dependiendo de la frecuencia deseada

int offsetSin, offsetCos; //Para el buffer circular (actualizacíón de los índices)
```

```
//PROGRAMA-----------------------------------------------------------------------------------------------------

//-------------------------------------------------------
// void conversion_dB(void)
//
// Descripción:
//  Función que, una vez calculado el módulo al cuadrado de la
//  DFT, convierte esos valores en función de unos umbrales
//  a valores en decibelios adaptados para el DAC
//-------------------------------------------------------
void conversion_dB(void) {
 for (i=0; i< N_FRECS; i++) {
       if (dft[i] > umbrales[NIVELES-1])
             dft[i] = escalado[NIVELES-1];
       else if (dft[i] < umbrales[0])
             dft[i] = escalado[0];
       else { //Si está dentro del rango de valores
             for (j=0; j< (NIVELES-1); j++) {
                   if ((dft[i]>=umbrales[j]) && (dft[i]<=umbrales[j+1])) {
                         distanciaArriba = umbrales[j+1] - dft[i];
                         distanciaAbajo = dft[i] - umbrales[j];
                         if (distanciaArriba >= distanciaAbajo) //Asigna el nivel más cercano
                                     dft[i] = escalado[j];
                         else
                               dft[i] = escalado[j+1];
                   }
             }
       }
 }
}
```

```
//-----------------------------------------------------
// void actualiza_Indices(void)
//
// Descripción:
//  Función que gestiona los senos y cosenos en la fórmula del
//  cálculo de la DFT. Dependiendo de n y de k, modifica los
//  punteros al array de muestras del seno a 10Hz.
//-----------------------------------------------------
void actualiza_Indices(void) {
 if (n == (N-1))
       for (i=0; i<N_FRECS; i++) {
             pSin[i] = sinusoide10Hz;
             pCos[i] = sinusoide10Hz + NUM_MUESTRAS_PERIODO_10HZ/4; //empieza en sinusoide10Hz[100]
       }
 else
       for (i=0; i<N_FRECS; i++) {
             offsetSin = ((pSin[i]-sinusoide10Hz) + salto[i]) % NUM_MUESTRAS_PERIODO_10HZ;
             offsetCos = ((pCos[i]-sinusoide10Hz) + salto[i]) % NUM_MUESTRAS_PERIODO_10HZ;
             pSin[i] = sinusoide10Hz + offsetSin;
             pCos[i] = sinusoide10Hz + offsetCos;
       }
}


//-----------------------------------------------------
// void reinicia_Acumuladores(void)
//
// Descripción:
//  Función para resetear los acumuladores de las partes real
//  e imaginaria de la DFT, a usar para que no se guarden los
//  valores de una ventana de cálculo a la siguiente.
//-----------------------------------------------------
void reinicia_Acumuladores(void) {
 for (k=0; k<N_FRECS; k++) {
             dftCos[k] = 0;
             dftSin[k] = 0;
       }
}
```

```
//-----------------------------------------------------
// void rutina_tout0(void)
//
// Descripción:
//   Función de atención a la interrupción para TIMER0, configurada
//   para que se ejecute 4000 veces por segundo. Hace de función
//   principal del programa.
//-----------------------------------------------------
void rutina_tout0(void) {
 //mbar_writeShort(MCFSIM_TER0,BORRA_REF);  // Reset del bit de fin de cuenta

 muestra = ADC_dato(); //Obtenemos muestra

 for (k=0; k<N_FRECS; k++) {    //Calcular y acumular aportaciones real e imaginaria de la muestra
       dftCos[k] += muestra*(*pCos[k])/DESESCALADO;
       dftSin[k] += muestra*(*pSin[k])/DESESCALADO;
 }

 actualiza_Indices(); //actualizar índices del seno y coseno para la siguiente muestra

 if (calculo && (n == (N-1))) { //Calculo del módulo de la DFT al cuadrado
       for (k=0; k<N_FRECS; k++)
             dft[k] = (dftCos[k]*dftCos[k]) + (dftSin[k]*dftSin[k]);

       conversion_dB();
       calculo = 0;
       n=0; //Empieza la siguiente ventana de muestras
       cont160Inter = 0;
       reinicia_Acumuladores();
 }
 else
       n++;

 if(n==N) { //Reinicio acumuladores si hemos acabado una ventana, y no hemos entrado en el anterior if
       reinicia_Acumuladores();
       n = 0;
 }
```

```
	DAC_dato(dft[cont160Inter/N_INT_COMPONENTE]); //Sacar cada componente de la DFT durante 8 interrupciones

	cont160Inter++;

	if(cont160Inter == N_INT_DFT) {
		reinicia_Acumuladores();
		calculo = 1;
		cont160Inter = 0;
		n = 0;
		set16_puertoS(0xFFFF); //Bit para el generador de rampa
	}
	else
		set16_puertoS(0x0000);
}
```

```
//-----------------------------------------------------
// void __init(void)
//
// Descripción:
//  Función por defecto de inicialización del sistema.
//  Configuración de las interrupciones, ADC/DAC e inicialización
//  de variables-
//-----------------------------------------------------
void __init(void) {
  DAC_ADC_init();
  mbar_writeByte(MCFSIM_PIVR,V_BASE);             // Fija comienzo de vectores de interrupción en V_BASE.
  ACCESO_A_MEMORIA_LONG(DIR_VTMR0)= (ULONG)_prep_TOUT0; // Escribimos la dirección de la función para TMR0
  output("COMIENZA EL PROGRAMA\r\n");
  mbar_writeShort(MCFSIM_TMR0, (PRESCALADO-1)<<8|0x3D);        // TMR0: PS=1-1=0 CE=00 OM=1 ORI=1 FRR=1 CLK=10 RST=1
  mbar_writeShort(MCFSIM_TCN0, 0x0000);           // Ponemos a 0 el contador del TIMER0
  mbar_writeShort(MCFSIM_TRR0, CNT_INT1);    // Fijamos la cuenta final del contador
  mbar_writeLong(MCFSIM_ICR1, 0x8888C888);  // Marca la interrupción del TIMER0 como no pendiente y de nivel 4
  sti();

 calculo = 1;
 n = 0;
 cont160Inter = 0;
 for (i=0; i<N_FRECS; i++) {
             dft[i] = escalado[0]; //inicializado al valor que DAC reconoce como cero, 122
             dftSin[i] = 0;
             dftCos[i] = 0;
       pSin[i] = sinusoide10Hz;
       pCos[i] = sinusoide10Hz + NUM_MUESTRAS_PERIODO_10HZ/4; //empieza en sinusoide10Hz[100]
 }

}
```

```
//-------------------------------------------------------
// void bucleMain(void)
//
// Descripción:
//   Función del programa principal
//-------------------------------------------------------
void bucleMain(void) {}
//-------------------------------------------------------
// Definición de rutinas de atención a la interrupción
// Es necesario definirlas aunque estén vacías
void rutina_int1(void){}
void rutina_int2(void){}
void rutina_int3(void){}
void rutina_int4(void){}
void rutina_tout1(void){}
void rutina_tout2(void){}
void rutina_tout3(void){}
```