GLOSARIO

A

Acústica. Parte de la física que estudia las propiedades, producción, reproducción y propagación del sonido.

Audición biaural. Capacidad de algunos animales, incluyendo el hombre, de localizar una fuente de sonido mediante el uso de dos canales independientes.

Ancho de banda. Intervalo de frecuencias útil de cualquier sistema limitado por la baja frecuencia f_L y la de alta frecuencia f_H, frecuencias a las cuales la ganancia del sistema cae 0.707 veces del valor máximo en el intervalo de frecuencias útil. También se define como la diferencia entre el límite de alta frecuencia y el límite de frecuencia baja.

Ancho de potencia media. Diferencia angular para la cual el haz o lóbulo principal dirigido en una dirección dada, su potencia se ha reducido a 3 dB.

Arreglo lineal uniforme. Arreglo formado por sensores distribuidos sobre una línea recta igualmente espaciados.

B

Biosonar. Técnica que utilizan algunos animales para navegar o localizar presas por medio de ondas acústicas en la cual se basa el funcionamiento del sonar.

C

Campo lejano. Región del espacio a la cual la fuente de ondas está lo suficiente alejada para considerar que el frente de onda esférico tiende a ser plano.

Campo cercano. Región del espacio a la cual la fuente de ondas emite un frente de onda esférico.

D

Dirección de arribo (*Direction Of Arrive*, DOA). Ángulo al cual incide un frente de onda sobre un arreglo de sensores.

E

Eco. Fenómeno que se produce cuando un sonido rebota contra una superficie lejana y regresa al receptor con una menor amplitud y con un cierto retardo.

F

Filtrado. Proceso de remover o alterar partes del contenido de las frecuencias de una señal para producir una nueva señal.

Filtro digital. Es aquél que utiliza un procesador digital para realizar operaciones matemáticas sobre valores muestreados de la señal.

Filtro FIR. Filtro de respuesta a impulso finito; es un tipo de filtro digital que si su entrada es un impulso la salida será un número limitado de términos diferentes de cero. Para obtener la salida sólo se emplean valores de la entrada actual y anteriores por lo que no depende del valor de salida actual y anteriores, también se les llama filtros digitales no recursivos.

Filtro IIR. Filtro de respuesta infinita al impulso; si el filtro se excita con un impulso, la salida será un número ilimitado de términos diferentes de cero por lo que no regresará a su estado de reposo. Para obtener la salida se emplean valores de la entrada actual y valores anteriores además de valores de la salida anteriores. Debido a que utiliza valores anteriores de salida, también se le llama filtro digital recursivo.

Formador de haz. Técnica utilizada para obtener un patrón de radiación deseado por medio de un arreglo de sensores.

Fracción de ancho de banda (FB). Parámetro que se utiliza para diferenciar los modos de operación de un arreglo de sensores; para valores menores de 1% se dice que son arreglos de banda angosta, es decir, que funcionan para un ancho de banda de frecuencias reducido, en cambio, para valores mayores del 1% se les llama arreglos de banda ancha, los cuales funcionan para ancho de banda amplio.

L

Lóbulos gratinados. Lóbulos que se presentan en el patrón de radiación cuando se utiliza una señal cuya longitud de onda es menor al doble de la distancia entre sensores y que son debidos a un fenómeno de *aliasing*. Estos lóbulos tienen una amplitud igual o incluso pueden ser mayores a la del lóbulo principal.

Lóbulos laterales. En un patrón de radiación direccionado, son los intervalos angulares en el que se encuentra un valor máximo relativo de potencia y está limitado por dos puntos mínimos y que se encuentran a los lados del lóbulo principal.

Lóbulo principal. También se le conoce como haz, para un patrón de radiación direccionado es el intervalo angular en el cual se encuentra concentrada la mayor potencia, y se mide a partir de los dos puntos mínimos más cercanos al punto máximo de potencia.

M

Micrófono. Transductor que transforma las ondas de sonido en señales eléctricas.

N

Número de onda. Es el número de veces que vibra una onda en una unidad de distancia, puede expresarse como ciclos por metro, pero es más frecuente expresarlo como radianes por metro.

O

Onda plana. Onda cuyos frentes de onda son planos paralelos de amplitud constante, que se propagan en el espacio en una sola dirección.

P

Patrón de radiación. Es la forma en que se distribuye la energía emitida o recibida por un dispositivo, y se representa mediante una gráfica que relaciona la posición angular con la potencia normalizada en dB.

R

Radar (*RAdio Detecting And Ranging*). Dispositivo que detecta la presencia de objetos o blancos, capaz de determinar su ubicación, dirección y distancia de forma parcial o total por medio de la emisión de ondas electromagnéticas concentradas en una zona angular.

S

Sensor. Es un dispositivo que detecta o mide una cantidad física como presión, voltaje, corriente, intensidad luminosa, etc.

Señal. Abstracción de alguna cantidad o fenómeno físico cuantificable que está en función de una o más variables independientes.

Señal CF-FM. Señal que consiste en una onda sinusoidal que en un intervalo de tiempo su frecuencia permanece constante que se le llama componente de frecuencia constante CF, y en otro intervalo de tiempo su frecuencia puede aumentar o disminuir a lo cual se llama componente de frecuencia modulada FM.

Señal continua. Señal presente para todo instante de tiempo.

Señal discreta. Señal presente únicamente en puntos discretos de tiempo. Por lo general las señales discretas son versiones muestreadas de señales continuas.

Sonar (*SOund Navigation And Ranging*). Dispositivo capaz de detectar y determinar la ubicación, dirección y distancia de objetos por medio de la emisión de ondas de presión.

Sonido. Sensación producida en el cerebro cuando el oído recibe un movimiento ondulatorio longitudinal que se propaga en aire producido por la vibración de un cuerpo.

T

Transductor. Dispositivo que transforma una magnitud física en otra distinta.

ANEXOS

A

Características generales del DSP TMS320F2812

El DSP TMS320F2812 pertenece a la generación TMS320C28x de Texas Instruments, y son dispositivos de alto nivel de integración y de alto desempeño para la solución de aplicaciones de control. El TMS320F2812 contiene características tanto de un microcontrolador como de un procesador digital de señales (DSP), combinando las características de la arquitectura Hardvard modificada, direccionamiento circular, ejecución de una instrucción por ciclo de reloj, operaciones registro a registro propias de un DSP, aunado al fácil manejo a través de un conjunto de instrucciones intuitivo y la manipulación de paquetes de bytes y bits de forma sencilla.

A continuación se describen las características principales de la familia C28x a la cual pertenece el DSP TMS320F2812.

A.1 UNIDAD CENTRAL DE PROCESAMIENTO CPU C28X

El CPU es un procesador digital de señales de 32 bits de punto flotante de bajo costo, que se encarga de realizar las operaciones aritméticas y lógicas sobre los datos, el DSP F2812 ejecuta 150 millones de instrucciones por segundo (Mips), lo cual permite al usuario realizar aplicaciones en tiempo real. En la Figura A.1 se muestra el diagrama del CPU.

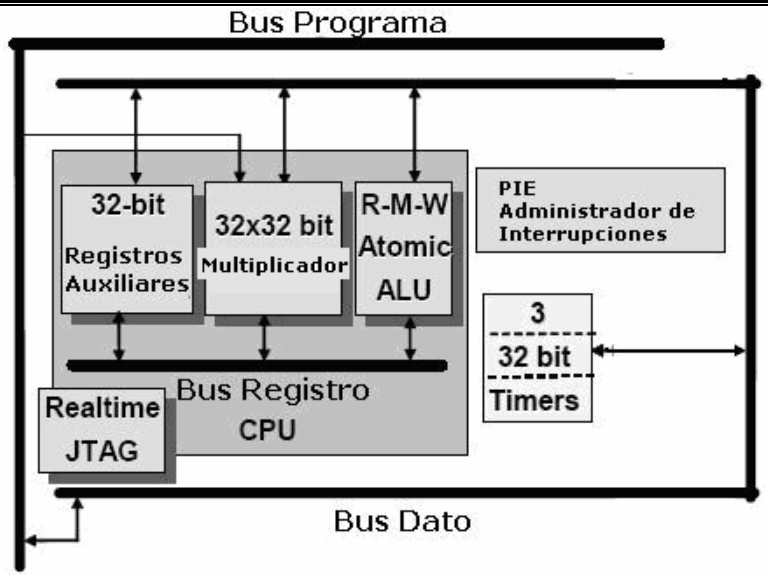


Figura A.1 Diagrama general del CPU C28x.

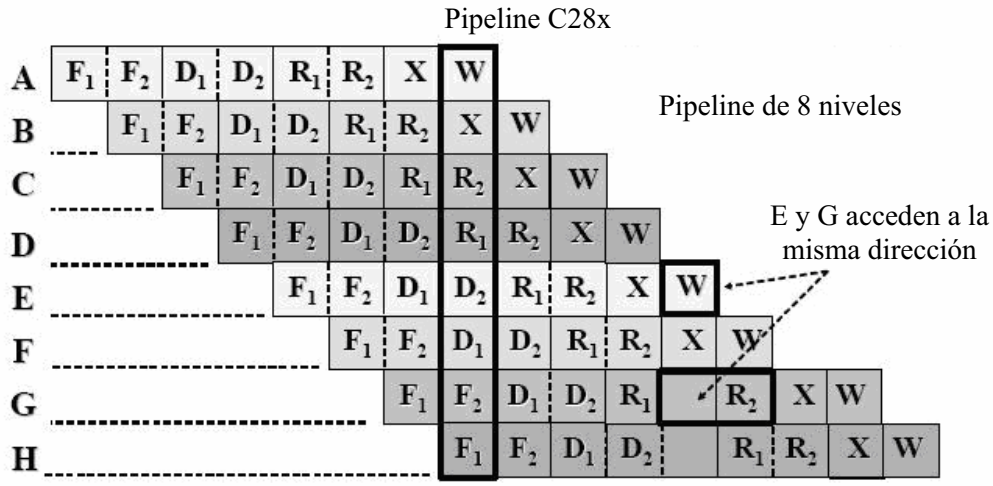
A continuación se listan sus principales características:

- Pipeline protegida. El CPU implementa un pipeline de ocho fases, tal como se muestra en Figura A.2.
- Unidad Aritmética Lógica ALU de 32 bits. Ejecuta operaciones de lógica booleana y aritmética con dos operandos.
- Unidad aritmética de registros direccionamiento ARAU. Genera direcciones de memoria dato, incrementa y decrementa apuntadores en paralelo con operaciones de la ALU.
- Registro de corrimiento. Ejecuta corrimiento hacia la derecha o izquierda de hasta 16 bits.
- Multiplicación en punto flotante. El C28x presenta un hardware multiplicador que puede ejecutar multiplicaciones de 16 x 16 bits o 32 x 32 bits en punto flotante.
- Visibilidad y emulación en tiempo real.

Para tener una idea de la capacidad del DSP, como ejemplo se muestra la instrucción de lenguaje ensamblador.

IMACL P,loc32,*XAR7

Esta instrucción indica la multiplicación de 32 x 32 bits entre loc32, la cual indica una variable direccionada de forma directa o forma indirecta, por el dato de localidad de 32 bits que apunta XAR7 el cual puede indicarse que solo se incremente. Se realiza el producto de 32 x 32 bits signados y el resultado de 64 bits se suma con el contenido del registro P.



F1: Dirección de instrucción
F2: Código de instrucción
D1: Decodifica instrucción
D2: Resuelve operando instrucción
R1: Operando dirección
R2: Obtiene operando
X: CPU ejecuta instrucción
W: Almacena contenido en memoria

Figura A.2 Pipeline protegida del C28x.

Por si sola esta instrucción se realiza en 2 ciclos de reloj, pero si esta se declara junto con la instrucción RPT la cual repite una instrucción N veces el producto acumulación de 32 x 32 bits se realiza en N+2 ciclos de reloj.

A.2 BUSES

Como muchos dispositivos DSP, se hace uso de múltiples buses para mover datos entre la memoria, los periféricos y el CPU. La arquitectura del bus de memoria contiene:

- Un bus de lectura programa: líneas de dirección de 22 bits y líneas dato de 32 bits.
- Un bus de lectura de datos: líneas de dirección de 32 bits y líneas dato de 32 bits.
- Un bus de escritura de datos: líneas de dirección de 32 bits y líneas dato de 32 bits.

Los buses dato de 32 bits de ancho permiten operaciones de 32 bits en un ciclo, esta arquitectura de bus múltiple, conocida como arquitectura Harvard permite buscar, leer un valor de dato y escribir un valor de dato en un solo ciclo. Todos los periféricos y memorias están conectados al bus de memoria y dará prioridad al acceso a memoria. La Figura A.3 se muestra la estructura de bus del C28x.

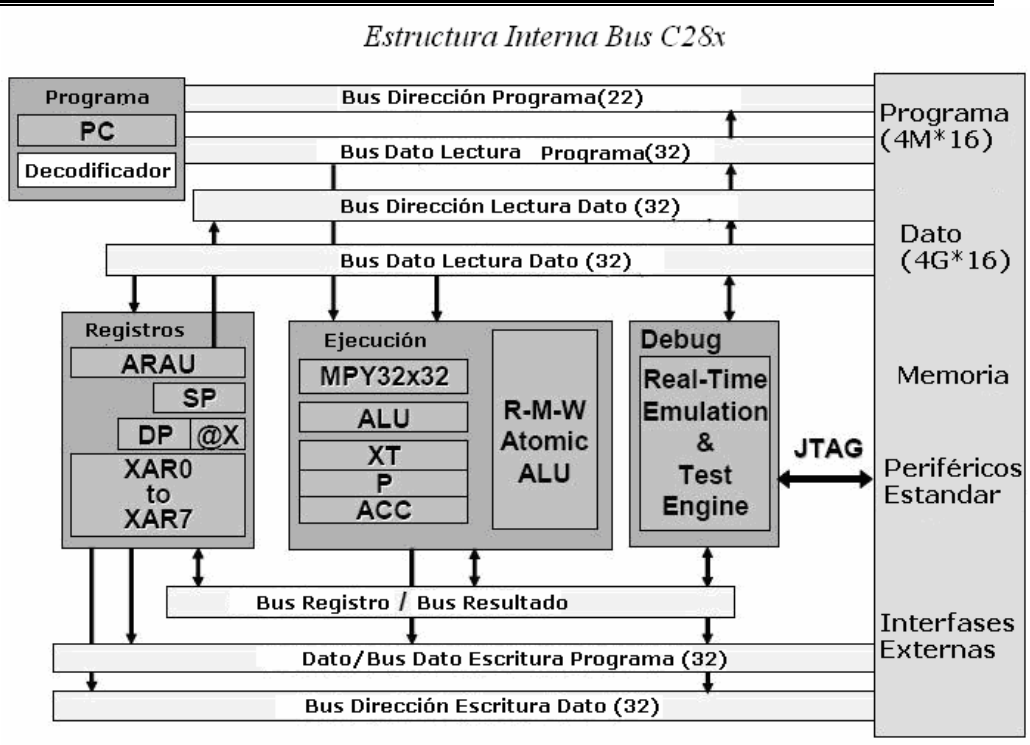


Figura A.3 Diagrama de buses del C28x.

A.3 MEMORIA

El C28x utiliza 32 bits de dirección datos y 22 bits para direcciones de programa. La Figura A.4 muestra el mapa de memoria del C28x. El mapa de memoria está divido en los siguientes segmentos:

- Programa/Dato on chip. Todos los dispositivos C28x contienen dos bloques de memoria de acceso único on chip referidos como M0 y M1. Cada bloque es de 1K words (1 word = 16 bits). M0 está ubicado de las direcciones 0x000000 a la dirección 0x0003FF. Y M1 esta mapeado en las direcciones 0x000400 a 0x0007FF. M0 y M1 están mapeados tanto para espacio programa y espacio dato. Por lo que M0 y M1 pueden utilizarse para ejecutar código o para datos.
- Reservado. Las direcciones 0x00000800 a 0x000009FF en espacio dato están reservados para registros emulación de CPU.
- Vectores Interrupción CPU. Sesenta y cuatro direcciones en espacio programa son utilizados para una tabla de 32 vectores interrupción de CPU.

Mapa de Memoria TMS320F2812

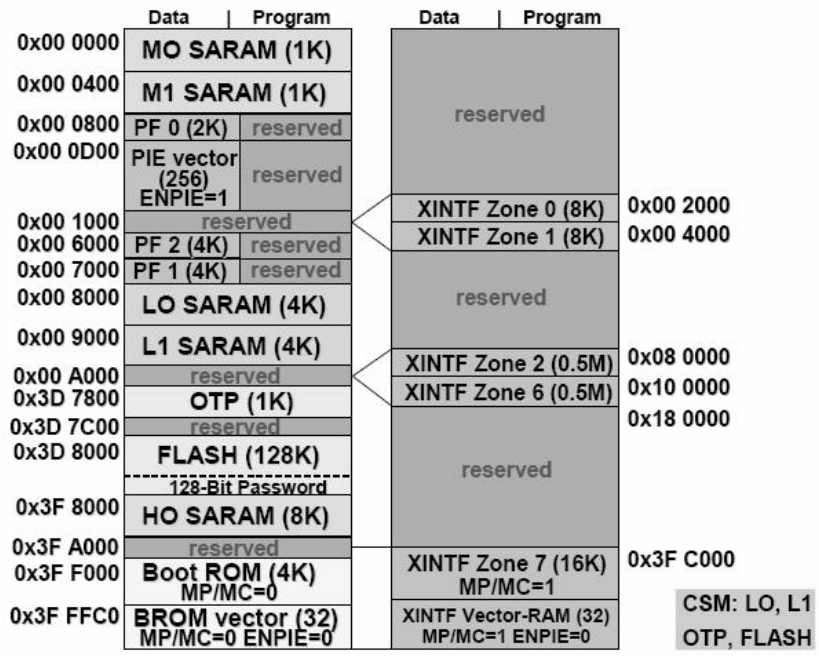


Figura A.4 Mapa de memoria del C28x.

A.4 RECURSOS DEL SISTEMA

A continuación se describe los recursos necesarios del DSP para el programa desarrollado en esta la tesis, cuanta memoria ocupa el programa así como que periféricos del DSP ocupa.

El programa se ubica de la localidad 0x3F8002 hasta la localidad 0x3F87C3 en memoria SARAM tanto el programa principal así como las subrutinas, interrupciones y otras funciones que se utilizan para configurar los periféricos, los datos se guardan en memoria SARAM a partir de la localidad 0x3F9000 hasta la localidad 0x3F958F. Adicionalmente se utilizaron las localidades 0x8094 a la localidad 0x9036 de la memoria SARAM para poder observar el comportamiento de las señales con ayuda de CCS.

En la Tabla A.1 se mencionan los recursos del DSP TMS320F2812 y se menciona cómo el sistema ocupa estos recursos.

Tabla A.1 Periféricos utilizados del DSP para desarrollado en esta la tesis	
Recurso	**Utilizado por el trabajo de tesis**
Memoria Flash y memoria programable una sola vez (*One Time Programmable* OTP)	No se utiliza
Modulo de Código de Seguridad	No se utiliza
Oscilador PLL controla la frecuencia de funcionamiento del DSP	150MHz
Temporizadores 0/1/2 CPU de 32 bits	Se utiliza el temporizador 0 para controlar la frecuencia de muestreo 44.1kHz
Watchdog	Deshabilitado
Registros GPIO MUX que se utilizan para multiplexar o conmutar la operación de pines compartidos	No se utiliza
Periférico de Expansión de Interrupciones PIE para multiplexar numerosas fuentes de interrupción en un conjunto reducido de entradas de interrupción	Se utilizan 2 interrupciones de las 96 disponibles
Interfase Externa (*External Interface,* XINTF)	No se utiliza
Controlador Mejorado de Área de Red (*Enhanced Controller Area Network,* eCAN)	No se utiliza
Administrador de Eventos (*Event Manager,* EV)	No se utiliza
Convertidor Analógico Digital (*Analogic to Digital Converter*, ADC)	Se ocupan los 16 canales disponibles del convertidor y se operan en modo cascada
Puerto Serial Buffereado Multicanal (*Multichannel Buffered Serial Port,* McBSP)	No se utiliza
Interfase Comunicación Serial (*Serial Communications Interface,* SCI)	No se utiliza
Interfase Periférico Serial (*Serial Peripheral Interface,* SPI)	No se utiliza

A.5 OTROS DIAGRAMAS

En la Figura A.5 se representa el funcionamiento del convertidor analógico digital ADC del TMS320F2812 operando de forma secuencial y en modo cascada. El convertidor ADC convierte cada canal uno a uno en el orden programado de acuerdo con los registros

Ch Sel(CONV00), de ahí el nombre de modo cascada, los registros Ch Sel(CONV00) seleccionan los canales con el selector del multiplexor analógico, y se guardan de forma secuencial en los registros ADCRESULT con el selector del multiplexor de resultados, El proceso da inicio con la señal de inicio de secuencia que puede provenir del administrador de eventos EVA o EVB, software, o el pin externo GPIOE1. El convertidor de 12 bits recibe la señal SOC y al terminar la conversión emite la señal EOC, todo esto controlado por la máquina de estado auto secuenciada.

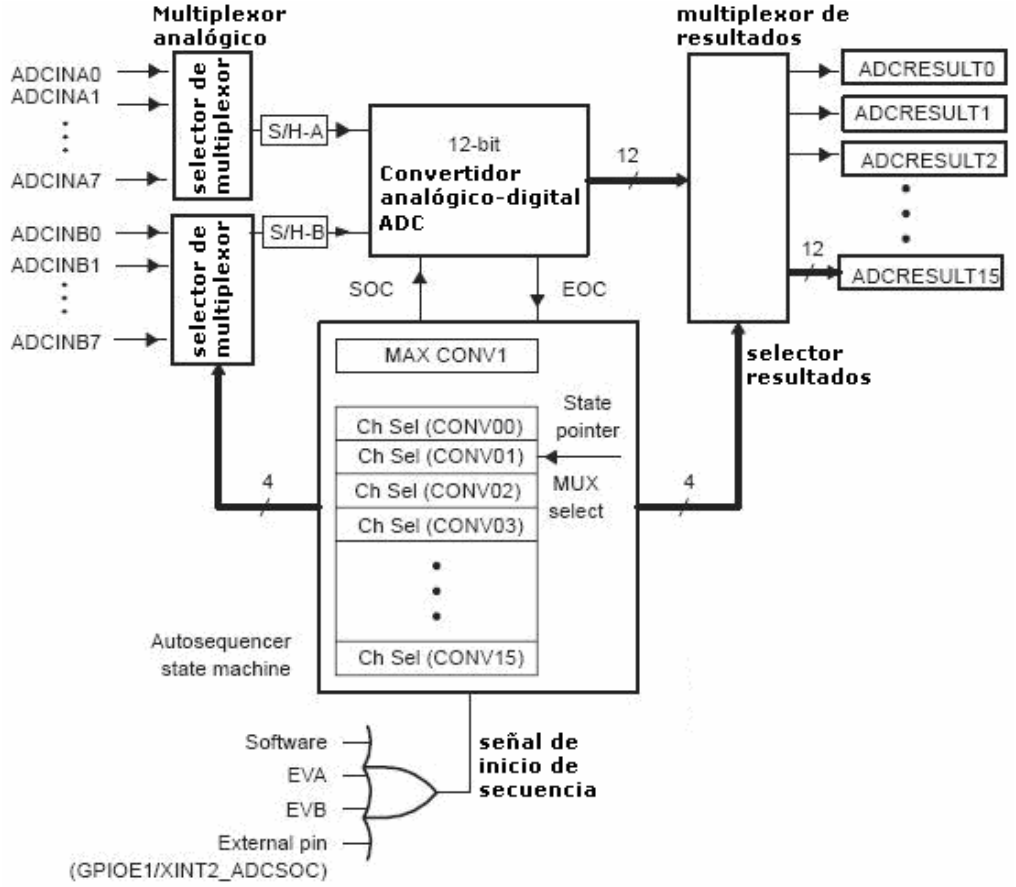


Figura A.5 Diagrama ADC auto secuenciado, en modo cascada.

El CPU del DSP TMS320F2812 dispone de 12 interrupciones, las cuales mediante el periférico PIE se pueden expandir hasta 96 lo cual se hace mediante el control de multiplexor los cuales expanden cada interrupción del CPU por 8. La Figura A.6 muestra esto. Las interrupciones propias del CPU se habilitan mediante el registro IER el cual contiene 12 bits mas otros 4 no disponibles, si se habilita un bit de este registro y en el registro IFR el bit llamado bandera correspondiente se activa, se efectúa la interrupción en el CPU con INTM. Las banderas en el registro IFR se activan mediante la expansión PIE, para ello la compuerta PIEACKx habilita la expansión donde x puede ser una de las doce interrupciones (1, 2, …, 12) y los registros PIEIERx(8:1) y PIEIFR(8:1) a su vez controlan las interrupciones secundarias INTx.X, donde X es igual a 1, 2, …, 8, las cuales se asignan

a una fuente de interrupción proveniente de algún periférico o interrupciones externas como señales provenientes de algún pin.

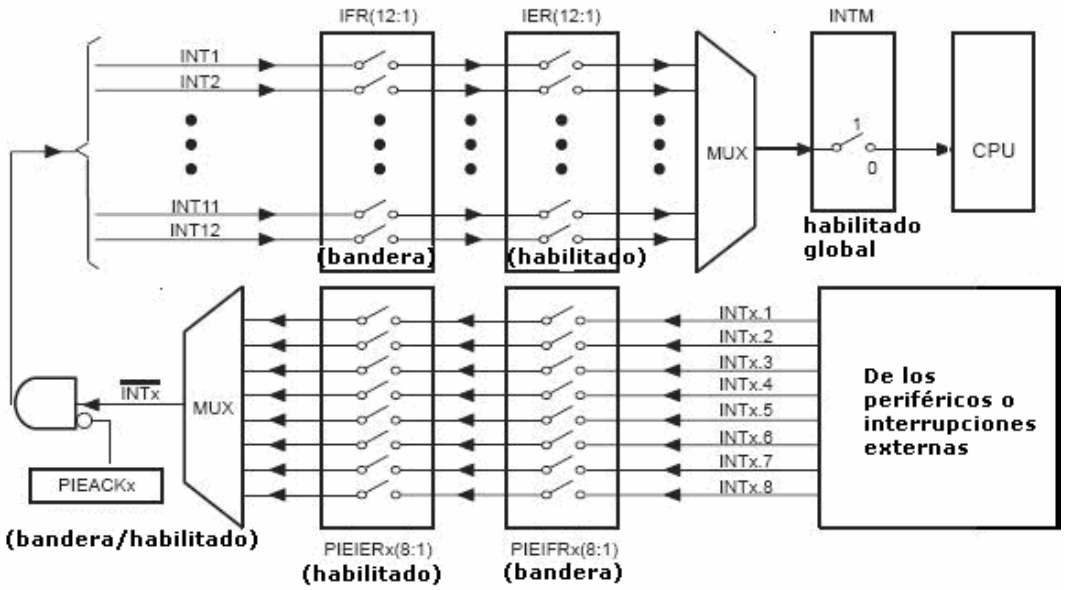


Figura A.6 Diagrama de multiplexaje de interrupciones utilizando el bloque PIE.

Anexos

B

Código del programa

En este anexo se muestra el código fuente del programa principal, así como de otras funciones que utiliza el programa. Cabe mencionar que en el CCS se incluyen los archivos del proyecto que se muestran en la Figura B.1, y al final de este anexo los archivos descritos como archivos encabezado de periféricos que incluyen funciones que facilitan la programación de los periféricos de los dispositivos DSPC281x [24].

```
#include <DSP281x_Device.h>
#include <DSP281x_GlobalPrototypes.h>
#include <DSP281x_DefaultIsr.h>
#include <DSP281x_Examples.h>

interrupt void Timer0_interrup(void);
interrupt void adc_interrup(void);

void RETRA(void);
/************************DECLARACION DE VARIABLES************************************/
int k=0, m=0, kl1=0, kl2=0;
long long int sumas=0;
float POT[37],*APOT;
unsigned int *leeA1;
float AC;
int Y=0;
long int APOYO=0,SUMA1=0,SALIDA1=0,*SALE1,DTs, X1=0;
long int contenido=0;
long int X16=0,ETs=0;
long int muE=0;
float ACUMULA=0, POTENCIA=0, Z,POTANG=0;
long int BL[15], *blok;
long int
Wa[16]={4096,1562,1800,2017,2203,2351,2454,2506,2506,2454,2351,2203,2017,1800,1562,4096},*Wa
cof,cont;//Qi=14
long int Wc[15]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},*Wccof;
long int mu=410;// mu=0.1 Qi=12

long int FIRFILT[16]={76, -171, 212, -42, -344, 713, -741, 317, 317, -741, 713, -344, -42,
212, -171, 76}, *Cofiltro;//Qi=12 filtro pasabanda15kHz A 18kHz
```

```
int MATRET[278];/*si d=0.01*/
long int *filtro1;
long int  DSUMA=0,ESUMA=0;

int *SEN[16];
int **ap1;
short int dirr;
/*
long long  *prueba,das;
int t[4]={1,2,4,8};*/
/*si d=0.01m*/
short int MATOFFSET[37][16]={{19,18,16,15,14,12,11,10, 9, 7, 6, 5, 3, 2, 1, 0},//90-1
{19,18,16,15,14,12,11,10, 9, 7, 6, 5, 3, 2, 1, 0},//85-2
{19,17,16,15,14,12,11,10, 8, 7, 6, 5, 3, 2, 1, 0},//80-3
{18,17,16,15,13,12,11,10, 8, 7, 6, 5, 3, 2, 1, 0},//75-4
{18,17,15,14,13,12,10, 9, 8, 7, 6, 4, 3, 2, 1, 0},//70-5
{17,16,15,14,12,11,10, 9, 8, 7, 5, 4, 3, 2, 1, 0},//65-6
{16,15,14,13,12,11,10, 8, 7, 6, 5, 4, 3, 2, 1, 0},//60-7
{15,14,13,12,11,10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0},//55-8
{14,13,12,11,10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 0},//50-9
{13,12,11,11,10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 0},//45-10
{12,11,10,10, 9, 8, 7, 6, 5, 5, 4, 3, 2, 1, 0, 0},//40-11
{11,10, 9, 8, 8, 7, 6, 5, 5, 4, 3, 2, 2, 1, 0, 0},//35-12
{ 9, 9, 8, 7, 7, 6, 5, 5, 4, 3, 3, 2, 1, 1, 0, 0},//30-13
{ 8, 7, 7, 6, 6, 5, 4, 4, 3, 3, 2, 2, 1, 1, 0, 0},//25-14
{ 6, 6, 5, 5, 4, 4, 3, 3, 3, 2, 2, 1, 1, 0, 0, 0},//20-15
{ 5, 4, 4, 4, 3, 3, 3, 2, 2, 2, 1, 1, 1, 0, 0, 0},//15-16
{ 3, 3, 2, 2, 2, 2, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0},//10-17
{ 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0},//5-18
{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},//0-19
{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1},//5-20
{ 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3},//10-21
{ 0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5},//15-22
{ 0, 0, 0, 1, 1, 2, 2, 3, 3, 3, 4, 4, 5, 5, 6, 6},//20-23
{ 0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 6, 6, 7, 7, 8},//25-24
{ 0, 0, 1, 1, 2, 3, 3, 4, 5, 5, 6, 7, 7, 8, 9, 9},//30-25
{ 0, 0, 1, 2, 2, 3, 4, 5, 5, 6, 7, 8, 8, 9,10,11},//35-26
{ 0, 0, 1, 2, 3, 4, 5, 5, 6, 7, 8, 9,10,10,11,12},//40-27
{ 0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,11,12,13},//45-28
{ 0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14},//50-29
{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15},//55-30
{ 0, 1, 2, 3, 4, 5, 6, 7, 8,10,11,12,13,14,15,16},//60-31
{ 0, 1, 2, 3, 4, 5, 7, 8, 9,10,11,12,14,15,16,17},//65-32
{ 0, 1, 2, 3, 4, 6, 7, 8, 9,10,12,13,14,15,17,18},//70-33
{ 0, 1, 2, 3, 5, 6, 7, 8,10,11,12,13,15,16,17,18},//75-34
{ 0, 1, 2, 3, 5, 6, 7, 8,10,11,12,14,15,16,17,19},//80-35
{ 0, 1, 2, 3, 5, 6, 7, 9,10,11,12,14,15,16,18,19},//85-36
{ 0, 1, 2, 3, 5, 6, 7, 9,10,11,12,14,15,16,18,19}};//90-37
short int *offset1, *offset2;

int teta1=0,conte=0,jkl,angulo=5,dir=0;

float POT2[37],*APOT2,ACUMULA2;
/*********************************PROGRAMA PRINCIPAL***********************************/
void main(void){
asm("  SETC           INTM");                //Deshabilita interrupciones
/**********************se les asigna valor a apuntadores*****************************/
APOT=&POT[0];
APOT2=&POT2[0];
Cofiltro=(long int*)&FIRFILT[0];
blok    =(long int*)&BL[0];// LMS
Wccof   =(long int*)&Wc[0];

SEN[0 ]= (int*)&MATRET[  0];
SEN[1 ]= (int*)&MATRET[ 20];
SEN[2 ]= (int*)&MATRET[ 39];
SEN[3 ]= (int*)&MATRET[ 55];
SEN[4 ]= (int*)&MATRET[ 70];
SEN[5 ]= (int*)&MATRET[ 85];
SEN[6 ]= (int*)&MATRET[ 98];
SEN[7 ]= (int*)&MATRET[110];
```

```
SEN[8 ]= (int*)&MATRET[121];
SEN[9 ]= (int*)&MATRET[132];
SEN[10]= (int*)&MATRET[144];
SEN[11]= (int*)&MATRET[157];
SEN[12]= (int*)&MATRET[172];
SEN[13]= (int*)&MATRET[188];
SEN[14]= (int*)&MATRET[205];
SEN[15]= (int*)&MATRET[224];
filtro1= (long int*)&MATRET[244];
ap1=&SEN[0];
SALE1 =   (long int*)0x08094;
leeA1 =   (unsigned int*)0x07108;

DisableDog();//Se deshabilita WatchDog
asm("   SETC           OBJMODE");
/*********************************Se configura PLL del DSP*****************************/
InitPll(0xA);

EALLOW;
SysCtrlRegs.PLLCR.bit.DIV=0xA;         // SYSCLK =150MHz
SysCtrlRegs.HISPCP.bit.HSPCLK=0x0;     // HSPCLK=SYSCLK
SysCtrlRegs.PCLKCR.bit.ADCENCLK=1;   //Se Habilita el reloj del ADC
/*********************************Se configura timer 0*********************************/
InitCpuTimers();                      //INICIA CPU TIMERS
CpuTimer0Regs.TCR.bit.TSS=0x1;       //CONTADOR DETENIDO
CpuTimer0Regs.PRD.all=0x0D49;//PRD = 300X10^6
CpuTimer0Regs.TIM.all=0x0D49;
CpuTimer0Regs.TCR.bit.FREE=0x0;
//Configura que el contador se detenga en la posición en que se encuentre
CpuTimer0Regs.TCR.bit.SOFT=0x0;
CpuTimer0Regs.TCR.bit.TIE=0x1;//habilita interrupción por contador
/*********************************Se configura periférico ADC del DSP******************/
InitAdc();
AdcRegs.ADCTRL3.bit.ADCEXTREF=0x0;
AdcRegs.ADCTRL1.bit.CPS=0x1;
//Preescalador de reloj. Si es igual a 0 Fclk=clk/1. Si es igual a 1 Fclk = clk/2
//75MHz conversion
AdcRegs.ADCTRL1.bit.SEQ_OVRD=0x0;
AdcRegs.ADCTRL3.bit.ADCCLKPS=0x01;   //0X06 Divisor de reloj ADCLK = HSPCLK/(ADCCLKPS+1)
75MHz/2=37.5MHz
AdcRegs.ADCTRL3.bit.SMODE_SEL=0x0;
AdcRegs.ADCTRL1.bit.SUSMOD=0x2;
//Modo emulación-suspensión. Bits que determinan que sucede cuando sucede una //suspensión
ocurre como por un breakpoint
AdcRegs.ADCTRL1.bit.ACQ_PS=0x1;             //Ancho del pulso SOC periodo = ADCTRL1[11:8]+1
veces el ADCLK
AdcRegs.ADCTRL1.bit.CONT_RUN=0x0;
//Modo de conversión. Si es 1 la conversión continua después de alcanzar EOS, si es 0 //la
conversión se detiene antes de alcanzar EOS
AdcRegs.ADCTRL1.bit.SEQ_CASC=0x1;
//Si es 0 SEQ1 y SEQ2 operan como dos secuenciadores distintos, si es 1 SEQ1 y SEQ2 //operan
como un único SEQ de 16 bits y opera en modo cascada
AdcRegs.ADCMAXCONV.bit.MAX_CONV1=0xF;//Numero máximo de conversiones por sesión
AdcRegs.ADCCHSELSEQ1.bit.CONV00=0;
AdcRegs.ADCCHSELSEQ1.bit.CONV01=1;
AdcRegs.ADCCHSELSEQ1.bit.CONV02=2;
AdcRegs.ADCCHSELSEQ1.bit.CONV03=3;
AdcRegs.ADCCHSELSEQ2.bit.CONV04=4;
AdcRegs.ADCCHSELSEQ2.bit.CONV05=5;
AdcRegs.ADCCHSELSEQ2.bit.CONV06=6;
AdcRegs.ADCCHSELSEQ2.bit.CONV07=7;
AdcRegs.ADCCHSELSEQ3.bit.CONV08=8;
AdcRegs.ADCCHSELSEQ3.bit.CONV09=9;
AdcRegs.ADCCHSELSEQ3.bit.CONV10=0xA;
AdcRegs.ADCCHSELSEQ3.bit.CONV11=0xB;
AdcRegs.ADCCHSELSEQ4.bit.CONV12=0xC;
AdcRegs.ADCCHSELSEQ4.bit.CONV13=0xD;
AdcRegs.ADCCHSELSEQ4.bit.CONV14=0xE;
AdcRegs.ADCCHSELSEQ4.bit.CONV15=0xF;
AdcRegs.ADCTRL2.bit.RST_SEQ1=0x1;
AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;
```

```
AdcRegs.ADCTRL2.bit.INT_ENA_SEQ1=0x1;//Habilita interrupción INT SEQ1
AdcRegs.ADCTRL2.bit.INT_MOD_SEQ1=0x0;

EALLOW;
/************************Configuración de interrupciones con periférico PIE************/
PieCtrlRegs.PIEIER1.bit.INTx7=0x1;  //habilita interrupción INT1.7 en PIE
PieVectTable.TINT0=&Timer0_interrup;//Liga rutina interrupción con la tabla vector
PieCtrlRegs.PIEIER1.bit.INTx6=0x1;  //Habilita interrupción PIE INT1.6
PieVectTable.ADCINT=&adc_interrup;  //Asocia el PIE tabla vector de interrupciones con la
función interrupción adc_interrup
PieCtrlRegs.PIECRTL.bit.ENPIE=0x1;  //Habilita interrupciones PIE
IER            |=M_INT1;             //Habilita INT1 en registro IER
EINT;                                //Habilita interrupciones INTM = 0
ERTM;                                //Habilita interrupciones en modo debugger

CpuTimer0Regs.TCR.bit.TSS=0x0;      //Inicia conteo

while(1){}                           //CICLO INFINITO

/******INTERRUPCIÓN TIMER 0 ************/
interrupt void Timer0_interrup(void){
AdcRegs.ADCTRL2.bit.SOC_SEQ1=0x1;
PieCtrlRegs.PIEACK.bit.ACK1=0x1;//Cambia
estado de PIEIFR asociado a INT1
return;
}
/**********INTERRUPCIÓN ADC ************/
interrupt void adc_interrup(void){

AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;

ap1=&SEN[8];
leeA1 = (unsigned int*)0x07108;
blok =(long int*)&BL[0];
**(ap1++)=((*(leeA1++))>>4)-2008;
//mic1-mic8-----------Qi=12
**(ap1++)=((*(leeA1++))>>4)-1979;
**(ap1++)=((*(leeA1++))>>4)-1975;
**(ap1++)=((*(leeA1++))>>4)-1928;
**(ap1++)=((*(leeA1++))>>4)-1958;
**(ap1++)=((*(leeA1++))>>4)-1938;
**(ap1++)=((*(leeA1++))>>4)-1951;
**(ap1++)=((*(leeA1++))>>4)-1920;
ap1=&SEN[0];
**(ap1++)=((*(leeA1++))>>4)-2020;
//mic9-mic16-----------Qi=12
**(ap1++)=((*(leeA1++))>>4)-1960;
**(ap1++)=((*(leeA1++))>>4)-1988;
**(ap1++)=((*(leeA1++))>>4)-2016;
**(ap1++)=((*(leeA1++))>>4)-1956;
**(ap1++)=((*(leeA1++))>>4)-1981;
**(ap1++)=((*(leeA1++))>>4)-1954;
**ap1=(*leeA1>>4)-2020;
Wacof=(long int*)&Wa[0];

/**ELECCIÓN DE VECTOR DIRECCIONAMIENTO**/

switch (teta1){

case 0:

offset1=MATOFFSET[18];
dirr=0;
break;

case  1:

offset1=MATOFFSET[0];
dirr=-90;
break;
case  2:

offset1=MATOFFSET[1];
dirr=-85;

break;
case  3:

offset1=MATOFFSET[2];

dirr=-80;

break;
case  4:

offset1=MATOFFSET[3];
dirr=-75;

break;

case  5:

offset1=MATOFFSET[4];
dirr=-70;

break;
case  6:

offset1=MATOFFSET[5];
dirr=-65;
break;
case  7:

offset1=MATOFFSET[6];
dirr=-60;

break;
case  8:

offset1=MATOFFSET[7];
dirr=-55;

break;
case  9:

offset1=MATOFFSET[8];
dirr=-50;

break;
case  10:
```

```
offset1=MATOFFSET[9];
dirr=-45;

break;
case  11:

offset1=MATOFFSET[10];
dirr=-40;

break;
case  12:

offset1=MATOFFSET[11];
dirr=-35;

break;

case  13:

offset1=MATOFFSET[12];

dirr=-30;

break;
case  14:

offset1=MATOFFSET[13];
dirr=-25;

break;
case  15:

offset1=MATOFFSET[14];
dirr=-20;

break;
case  16:

offset1=MATOFFSET[15];
dirr=-15;

break;
case  17:

offset1=MATOFFSET[16];
dirr=-10;

break;
case  18:

offset1=MATOFFSET[17];
dirr=-05;

break;
case  19:

offset1=MATOFFSET[18];
dirr=0;

break;
case  20:

offset1=MATOFFSET[19];
dirr=05;

break;

case  21:

offset1=MATOFFSET[20];
dirr=10;

break;
case  22:

offset1=MATOFFSET[21];
dirr=15;

break;
case  23:

offset1=MATOFFSET[22];
dirr=20;

break;
case  24:

offset1=MATOFFSET[23];
dirr=25;

break;
case  25:

offset1=MATOFFSET[24];
dirr=30;

break;
case  26:

offset1=MATOFFSET[25];
dirr=35;

break;
case  27:

offset1=MATOFFSET[26];
dirr=40;

break;
case  28:

offset1=MATOFFSET[27];
dirr=45;

break;

case  29:

offset1=MATOFFSET[28];
dirr=50;

break;
case  30:

offset1=MATOFFSET[29];
dirr=55;
break;
case  31:

offset1=MATOFFSET[30];
dirr=60;
break;
case  32:

offset1=MATOFFSET[31];
dirr=65;
break;
case  33:

offset1=MATOFFSET[32];
dirr=70;
```

```
break;

case  34:

offset1=MATOFFSET[33];
dirr=75;
break;
case  35:

offset1=MATOFFSET[34];
dirr=80;
break;
case  36:

offset1=MATOFFSET[35];
dirr=85;
break;
case  37:

offset1=MATOFFSET[36];
dirr=90;
break;

default:

offset1=MATOFFSET[18];
dirr=00;
break;
}
/**SE OBTIENE EL FORMADOR DE HAZ FIJO CON
VENTANA Y LA MATRIZ DE BLOQUEO B********/

APOYO=(*(SEN[0] +*(offset1++)));
X1=APOYO;
SUMA1 =(APOYO*(*(Wacof++)));
APOYO=(*(SEN[1] +*(offset1++)));
*(blok++)=(X1-APOYO);
X1=APOYO;
SUMA1+=(APOYO*(*(Wacof++)));
APOYO=(*(SEN[2] +*(offset1++)));
*(blok++)=(X1-APOYO);
X1=APOYO;
SUMA1+=(APOYO*(*(Wacof++)));
APOYO=(*(SEN[3] +*(offset1++)));
*(blok++)=(X1-APOYO);
X1=APOYO;
SUMA1+=(APOYO*(*(Wacof++)));
APOYO=(*(SEN[4] +*(offset1++)));
*(blok++)=(X1-APOYO);
X1=APOYO;
SUMA1+=(APOYO*(*(Wacof++)));
APOYO=(*(SEN[5] +*(offset1++)));
*(blok++)=(X1-APOYO);
X1=APOYO;
SUMA1+=(APOYO*(*(Wacof++)));
APOYO=(*(SEN[6] +*(offset1++)));
*(blok++)=(X1-APOYO);
X1=APOYO;
SUMA1+=(APOYO*(*(Wacof++)));
APOYO=(*(SEN[7] +*(offset1++)));
*(blok++)=(X1-APOYO);
X1=APOYO;
SUMA1+=(APOYO*(*(Wacof++)));
APOYO=(*(SEN[8] +*(offset1++)));
*(blok++)=(X1-APOYO);
X1=APOYO;
SUMA1+=(APOYO*(*(Wacof++)));
APOYO=(*(SEN[9] +*(offset1++)));
*(blok++)=(X1-APOYO);
X1=APOYO;
SUMA1+=(APOYO*(*(Wacof++)));
APOYO=(*(SEN[10] +*(offset1++)));
*(blok++)=(X1-APOYO);
X1=APOYO;
SUMA1+=(APOYO*(*(Wacof++)));
APOYO=(*(SEN[11] +*(offset1++)));
*(blok++)=(X1-APOYO);
X1=APOYO;
SUMA1+=(APOYO*(*(Wacof++)));
APOYO=(*(SEN[12] +*(offset1++)));
*(blok++)=(X1-APOYO);
X1=APOYO;
SUMA1+=(APOYO*(*(Wacof++)));
APOYO=(*(SEN[13] +*(offset1++)));
*(blok++)=(X1-APOYO);
X1=APOYO;
SUMA1+=(APOYO*(*(Wacof++)));


APOYO=(*(SEN[14] +*(offset1++)));
*(blok++)=(X1-APOYO);
X1=APOYO;
SUMA1+=(APOYO*(*(Wacof++)));
APOYO=(*(SEN[15] +*offset1));
*blok=(X1-APOYO);
X1=APOYO;
SUMA1+=(APOYO*(*Wacof));

filtro1= (long int*)&MATRET[244];
*filtro1=SUMA1<<2;//Qi(SUMA1)=26,
Qi(*filtro1)=15
/*****SE EJECUTA EL FILTRO FIR PASO BANDA
DESPUÉS DEL FORMADOR DE HAZ FIJO********/
Cofiltro=(long int*)&FIRFILT[0];

X16 =((*(filtro1++))*(*(Cofiltro++)));
X16+=((*(filtro1++))*(*(Cofiltro++)));
X16+=((*(filtro1++))*(*(Cofiltro++)));
X16+=((*(filtro1++))*(*(Cofiltro++)));
X16+=((*(filtro1++))*(*(Cofiltro++)));
X16+=((*(filtro1++))*(*(Cofiltro++)));
X16+=((*(filtro1++))*(*(Cofiltro++)));
X16+=((*(filtro1++))*(*(Cofiltro++)));
X16+=((*(filtro1++))*(*(Cofiltro++)));
X16+=((*(filtro1++))*(*(Cofiltro++)));
X16+=((*(filtro1++))*(*(Cofiltro++)));
X16+=((*(filtro1++))*(*(Cofiltro++)));
X16+=((*(filtro1++))*(*(Cofiltro++)));
X16+=((*filtro1)*(*Cofiltro));

blok =(long int*)&BL[0];
Wccof=(long int*)&Wc[0];
/*********************LMS*****************/
/*************************************/
/**SUMATORIA DEL PRODUCTO COEFICIENTES Wc
CON MATRIZ DE BLOQUEO B*****************/
DTs =((*(blok++))*(*(Wccof++)));
DTs+=((*(blok++))*(*(Wccof++)));
DTs+=((*(blok++))*(*(Wccof++)));
DTs+=((*(blok++))*(*(Wccof++)));
DTs+=((*(blok++))*(*(Wccof++)));
DTs+=((*(blok++))*(*(Wccof++)));
DTs+=((*(blok++))*(*(Wccof++)));
DTs+=((*(blok++))*(*(Wccof++)));
DTs+=((*(blok++))*(*(Wccof++)));
DTs+=((*(blok++))*(*(Wccof++)));
DTs+=((*(blok++))*(*(Wccof++)));
DTs+=((*blok)*(*Wccof));
```

```
blok =(long int*)&BL[0];
Wccof=(long int*)&Wc[0];

ETs=(X16-DTs)>>13;

muE=(ETs*mu)>>11;
/*CALCULO DE LOS SIGUIENTES COEFICIENTES
DEL ALGORITMO ADAPTABLE Wc*/
*(Wccof++)=*Wccof+((muE*(*(blok++)))>>11);
*(Wccof++)=*Wccof+((muE*(*(blok++)))>>11);
*(Wccof++)=*Wccof+((muE*(*(blok++)))>>11);
*(Wccof++)=*Wccof+((muE*(*(blok++)))>>11);
*(Wccof++)=*Wccof+((muE*(*(blok++)))>>11);
*(Wccof++)=*Wccof+((muE*(*(blok++)))>>11);
*(Wccof++)=*Wccof+((muE*(*(blok++)))>>11);
*(Wccof++)=*Wccof+((muE*(*(blok++)))>>11);
*(Wccof++)=*Wccof+((muE*(*(blok++)))>>11);
*(Wccof++)=*Wccof+((muE*(*(blok++)))>>11);
*(Wccof++)=*Wccof+((muE*(*(blok++)))>>11);
*(Wccof++)=*Wccof+((muE*(*(blok++)))>>11);
*(Wccof++)=*Wccof+((muE*(*(blok++)))>>11);
*(Wccof++)=*Wccof+((muE*(*(blok++)))>>11);
*Wccof    =*Wccof+((muE*(*blok))>>11);
/*******CÁLCULO DE LA POTENCIA**********/
*SALE1=ETs;
SALE1++;
AC=ETs;
AC=AC*AC;
ACUMULA=ACUMULA+AC;
AC=X16;
AC=AC*AC;
ACUMULA2=ACUMULA2+AC;
 if (k<2000){
   k++;
        }
 else {
        Wccof=(long int*)&Wc[0];
        *(Wccof++)=0;
        *(Wccof++)=0;
        *(Wccof++)=0;
        *(Wccof++)=0;
        *(Wccof++)=0;
        *(Wccof++)=0;
        *(Wccof++)=0;
        *(Wccof++)=0;
        *(Wccof++)=0;
        *(Wccof++)=0;
        *(Wccof++)=0;
        *(Wccof++)=0;
        *(Wccof++)=0;
        *(Wccof++)=0;
        *Wccof    =0;

        k=0;

        SALE1   =   (long int*)0x08094;
        ACUMULA =   0.0005*ACUMULA;
        ACUMULA2=   0.0005*ACUMULA2;
        POTENCIA=   ACUMULA;
*APOT=ACUMULA;

        *APOT2=ACUMULA2;
        ACUMULA =  0;
        APOT++;
        APOT2++;

/***Control para realizar el escaneo***/

        if(POTENCIA>POTANG){
                        POTANG=POTENCIA;
                        angulo=dirr;
        }
        teta1++;
        if(teta1>38){
                APOT=&POT[0];
                APOT2=&POT2[0];
                teta1=0;

                if(POTANG<4500){
                        dir=0;
                }
                else{
                dir=angulo;
//Se determina la dirección de arribo
        }
        POTANG=0;
}

}

AdcRegs.ADCTRL2.bit.RST_SEQ1=0x1;
PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;

RETRA();
return;
}

/*********FUNCION RETRA.ASM************/

           .def        _RETRA
           .sect       "ramfuncs"
           .global     __RETRA
PES        .long       01h
EQUIS7     .long       01h

_RETRA:

           CLRC        VMAP
           SETC        OBJMODE
           MOVL        XAR2, #03F9315h
           RPT         #255
           ||DMOV      *--XAR2
           RPT         #22
           ||DMOV      *--XAR2
           MOVL        XAR2, #03F9315h
           RPT         #33
           ||DMOV      *--XAR2
           LRT
```

Como se puede ver en la Figura B.1 El proyecto incluye los archivos que se incluyen en SPRC097 C281x C/C++ Header Files and Peripheral Examples. Al agregar los archivos DSP281x_Device.h, DSP281x_DevEmu.h, DSP281x_GlobalPrototypes.h, DSP281x_SysCtrl.h, rts2800_ml.lib, DSP281x_DefaultIsr.c, DSP281x_GlobalVariableDefs.c, con lo que el sistema automáticamente agrega el resto de los archivos. Los archivos BFSTEER.c donde se incluye el programa principal, y el archivo RETRA.asm el cuales un programa escrito en ensamblador para realizar una serie de corrimientos de datos, son los archivos programados que se realizaron para la tesis.

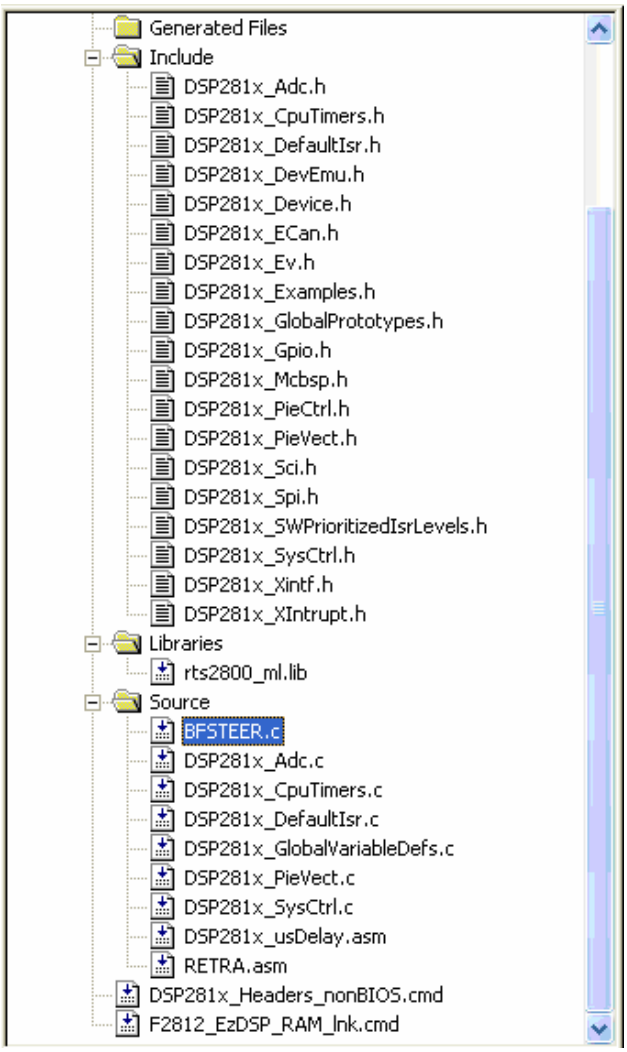


Figura B.1 Ventana parte de CCS

Anexos

C

Circuitos impresos

En este anexo se muestran los circuitos impresos necesarios para el arreglo de sensores y el circuito de acondicionamiento que sirve de interfase entre la tarjeta de desarrollo DSK DSP TMS320F2812 y el arreglo de sensores. Los circuitos impresos fueron realizados en el programa Eagle (*Easily Applicable Graphical Layout Editor*) V 4.1 para *Windows*.

En la Figura C.1 se muestra el circuito del arreglo de micrófonos, como se puede observar los micrófonos electret están soldados a este circuito, a su vez el circuito se conecta con el circuito B que se muestra en la Figura C.3 mediante un cable plano.

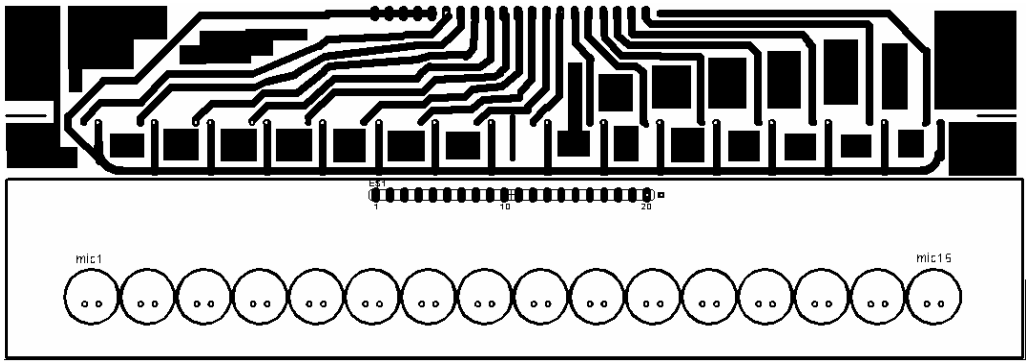


Figura C.1 Arreglo de micrófonos electret em-926.

En la Figura C.2 se muestra el circuito B, que es el circuito donde se encuentra el circuito acondicionador de señal con 16 filtros pasa banda con un ancho de banda de 20 Hz hasta 18 kHz, el circuito recibe como entradas ya sea las señales del arreglo o las señales recibidas en las entradas auxiliares recibidas en el circuito B. En la parte inferior del

circuito A se encuentran 30 pines, que corresponden al puerto ADC de la tarjeta DSK DSP TMS320F2812, a la derecha se encuentran otros 30 pines los cuales están conectados a los voltajes de alimentación y a tierra de los cuales se pueden utilizar para polarizar otros circuitos.

Se hizo la omisión de los valores de los elementos del circuito ya que afectaría la visibilidad del circuito.

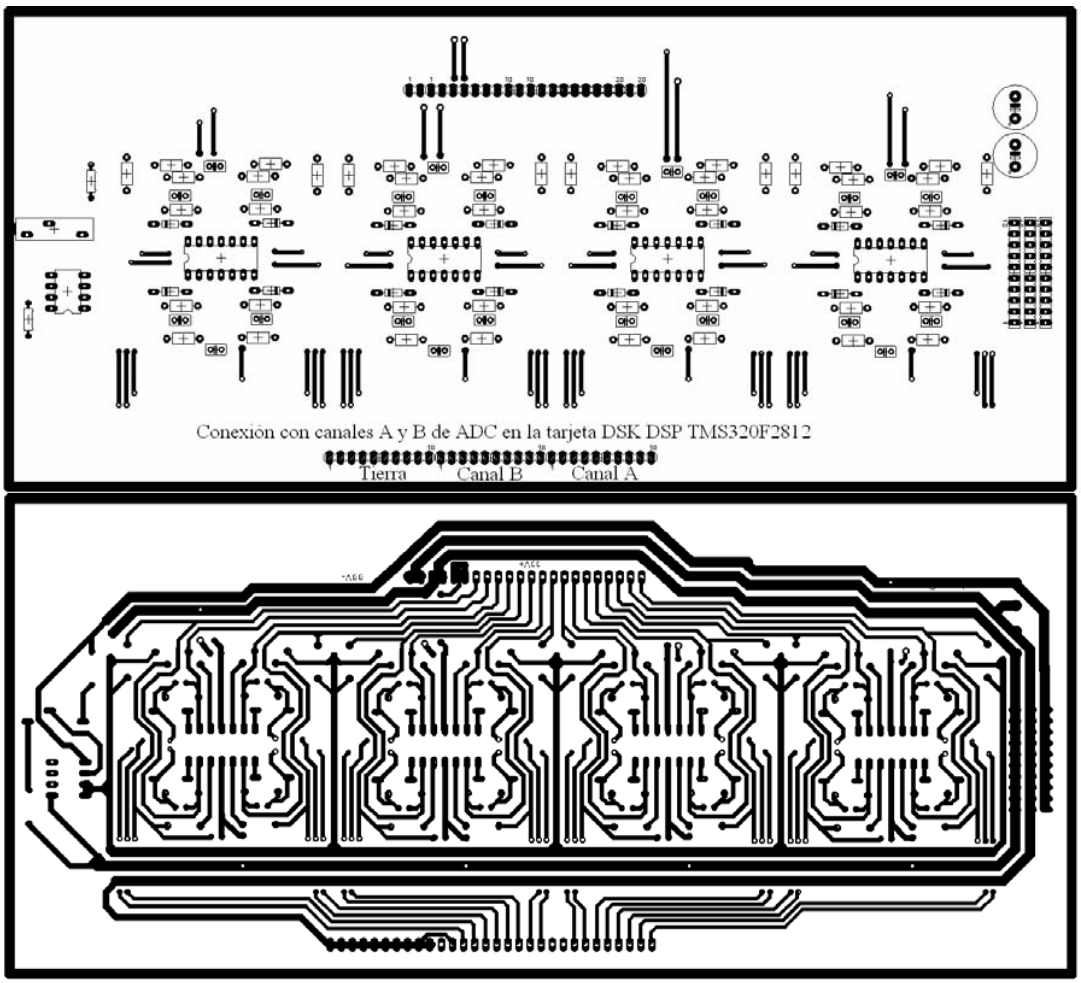


Figura C.2 Circuito B que incluye los amplificadores operacionales TL074.

En la Figura C.3 se muestra el circuito A que es donde se reciben las entradas tanto los voltajes de polarización del circuito, que son -5 V, +5 V y tierra. En éste se incluye la conexión con el arreglo de micrófonos, que se pueden conmutar mediante los 16 switches con las entradas auxiliares, cabe mencionar que las entradas auxiliares están diseñadas para tener una ganancia de 1 y un ancho de banda de 20 Hz hasta 30 kHz.

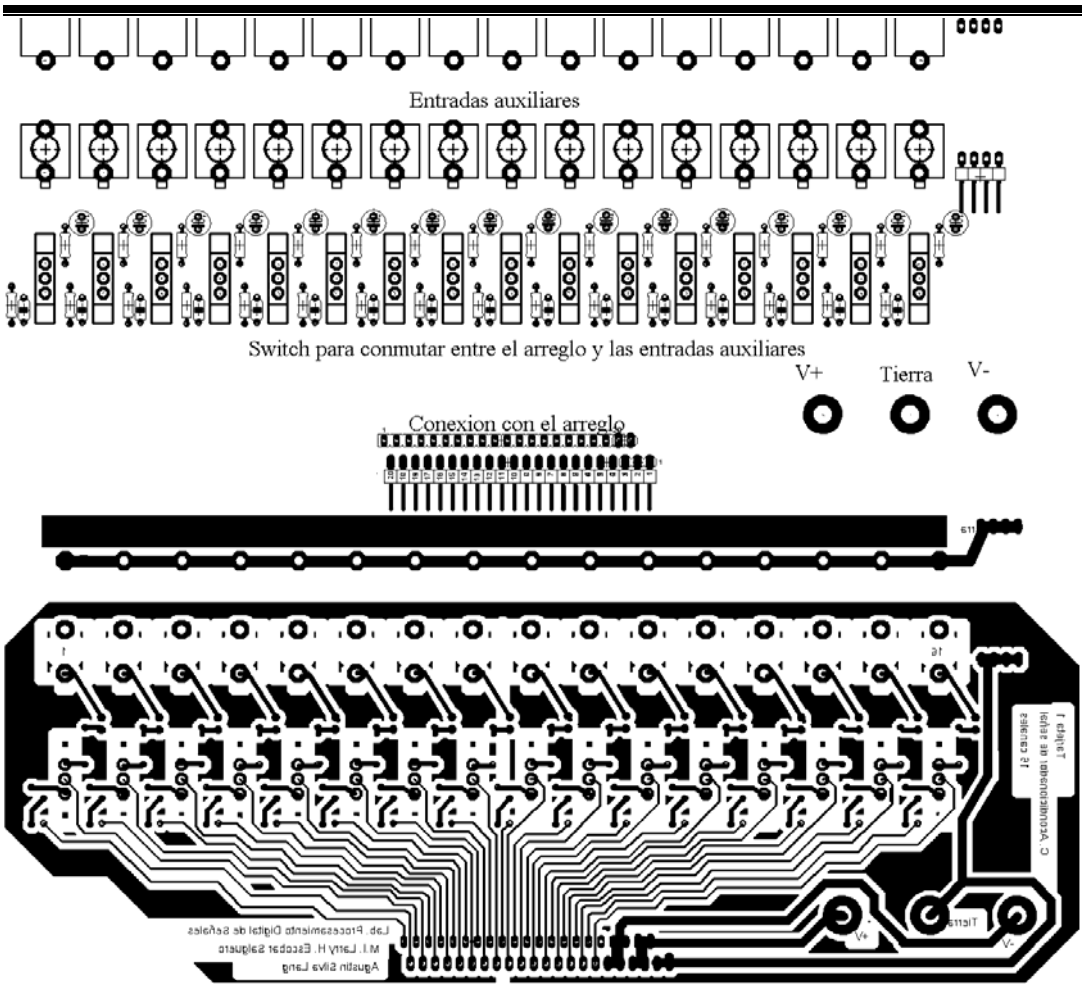


Figura C.3 Circuito A entrada de señales del arreglo de micrófonos y entradas auxiliares

En la Figura C.4 se muestran físicamente los circuitos A y B, el arreglo de micrófonos y la tarjeta DSK DSP TMS320F2812. En la figura C.5 se observa como se conecta el circuito A y B entre si, además en C.5 (b) se observa cómo el circuito B mediante cable plano está conectado con el puerto ADC del DSP. En C.5 (a) se observa como en el circuito A se inserta el cable plano que une al circuito con el arreglo de micrófonos.

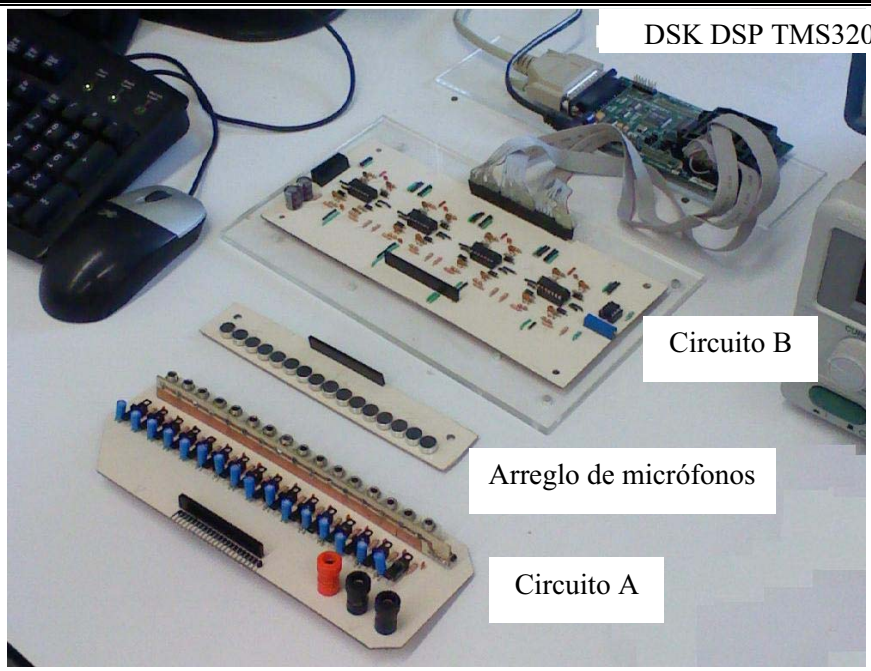


Figura C.4 Circuitos diseñados junto con la tarjeta DSK DSP TMS320F2812.

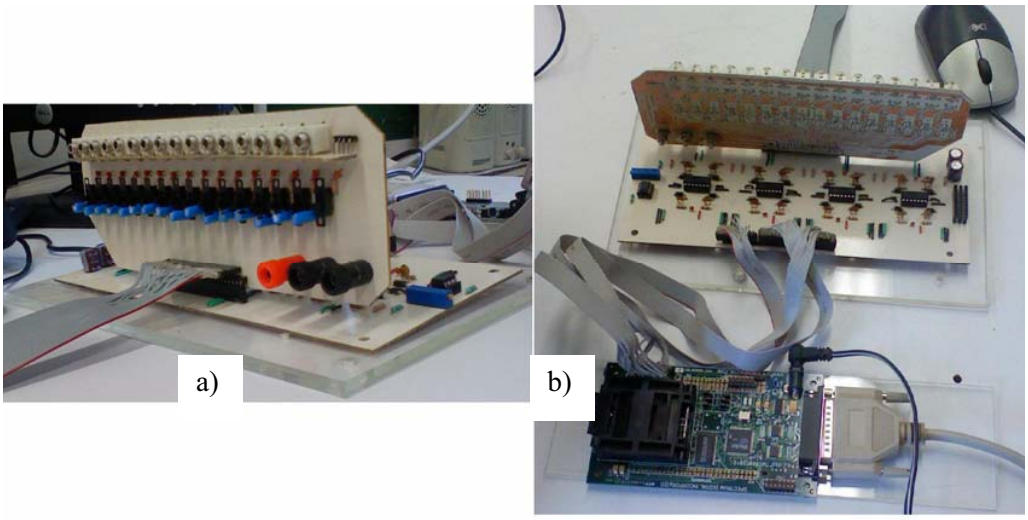


Figura C.5 Circuitos interconectados.

A continuación en la Figura C.6 se observa el arreglo de micrófonos colocado sobre el disco transportador junto con el cable plano que lo une con el circuito A. Y por último en la Figura C.7 se observa todo el sistema.

Figura C.6 Arreglo de micrófonos sobre el disco transportador.

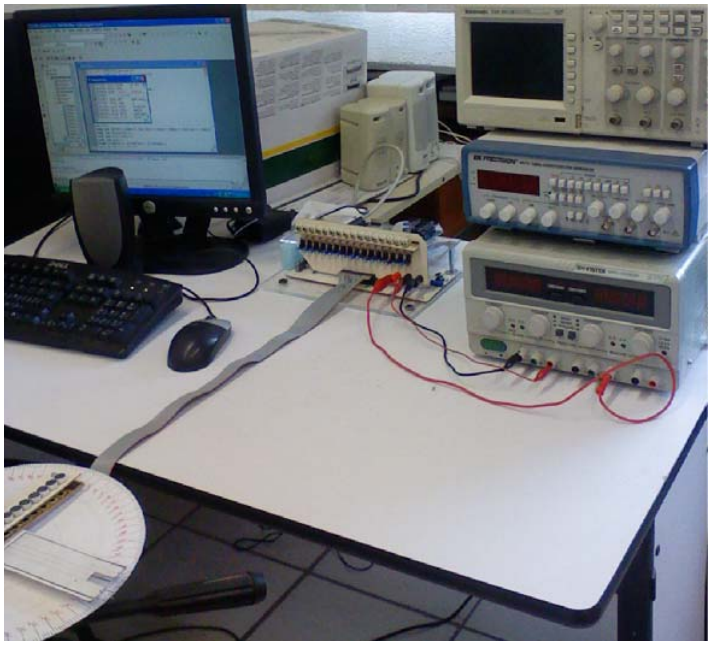


Figura C.7 Sistema completo.