ANALIZADOR DE ESPECTROS PORTÁTIL UTILIZANDO LA FAMILIA DSP5680X DE MOTOROLA

YAIR DE JESÚS RUIDÍAZ PALOMINO

OMAR JAVIER TÍJARO ROJAS

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2005

ANALIZADOR DE ESPECTROS PORTÁTIL UTILIZANDO LA FAMILIA DSP5680X DE MOTOROLA

YAIR DE JESÚS RUIDÍAZ PALOMINO
OMAR JAVIER TÍJARO ROJAS

Este proyecto es presentado como requisito para optar al título de
Ingeniero Electrónico

DIRECTOR
JAIME GUILLERMO BARRERO PÉREZ
Magíster en Potencia Eléctrica (MPE)

Codirector
JORGE EDUARDO HIGUERA PORTILLA
Ingeniero Electrónico

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2005

DEDICATORIA

Para mis mejores amigos, ustedes padres y hermanos,
ustedes que siempre he amado y siempre amaré,
y sin ninguna duda a ti Diana.
Yair de Jesús Ruidíaz Palomino

A mis padres y hermanos, este triunfo es suyo.
Omar Javier Tíjaro Rojas

AGRADECIMIENTOS

Agradecemos en primera instancia a Dios que fue nuestro apoyo espiritual y el que nos dio la fortaleza necesaria en todo momento para poder culminar esta etapa.

A nuestros padres, que son las personas que nos han dado la mano en aquellos momentos en donde hemos tropezado. También agradecemos a ellos su confianza y apoyo ilimitado, el cual ha sido fundamental para hacer de nosotros personas capaces de solucionar los problemas que se nos presentan en la vida, tratando siempre que estas soluciones no perjudiquen, sino que beneficien a la sociedad.

A nuestras familias y amigos, que siempre nos han escuchado, apoyado y orientado. Agradecemos a éstas personas maravillosas, que le dan sentido a nuestra vida con momentos de alegría y tristeza.

Agradecemos especialmente a nuestros directores: el ingeniero Jorge Higuera y el profesor Jaime Barrero, además a los profesores Jorge Chacón, César Duarte, José Amaya, Daniel Sierra, Gabriel Ordoñez y a los ingenieros Samuel Pinzón y Javier Mier; los cuales fueron las personas que fortalecieron los pilares de nuestros conocimientos.

TABLA DE CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

Pág.

LISTA DE ANEXOS

Pág.

RESUMEN

TITULO: ANALIZADOR DE ESPECTROS PORTÁTIL UTILIZANDO LA FAMILIA DSP5680X DE MOTOROLA[*].

AUTORES: Yair de Jesús Ruidíaz Palomino y Omar Javier Tíjaro Rojas[**].

PALABRAS CLAVES: Transforma rápida de Fourier (FFT), procesador de señales digitales (DSP), DSP56F807, tarjetas de desarrollo, pantallas de cristal líquido (LCD), filtros analógicos.

DESCRIPCIÓN:

En este proyecto se diseñó un equipo de bajo costo capaz de realizar un análisis en frecuencia basado en la FFT, a partir del DSP56F807 de Motorola. Para la ejecución del mismo fue necesario implementar tarjetas de desarrollo y algoritmos con el fin de estudiar y evaluar de una mejor forma el dispositivo programable (DSP), y poder llevar a cabo de una manera adecuada el aprendizaje de cada uno de los módulos embebidos en éste.

Este libro consta de varios capítulos con figuras y tablas para un mejor entendimiento de las técnicas utilizadas para el diseño del equipo, entre ellos se encuentran: Procesadores de señales digitales, tratamiento digital de señales, pantallas de cristal líquido, pruebas, análisis de resultados, conclusiones y recomendaciones. Además se presentan anexos que sirven para el aprendizaje de los programas de computadora que se utilizaron, software implementado en el equipo, manuales de conexiones e implementación de hardware.

El equipo final presentado funciona de la siguiente manera: primero adquiere una señal la cual es filtrada analógicamente y luego muestreada mediante un DSP, que es el dispositivo que luego se encarga de realizar el tratamiento digital y enviar los datos hacia una LCD tipo gráfico, donde es visualizado el espectro. En la pantalla se muestra un diagrama de barras perteneciente al espectro de frecuencias de la señal de entrada.

[*] Trabajo de grado

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones. Director: Jaime Guillermo Barrero Pérez.

ABSTRACT

TITLE: PORTABLE SPECTRUM ANALYZER USING DSP5680X MOTOROLA'S FAMILY*.

AUTHORS: YAIR DE JESÚS RUIDÍAZ PALOMINO and OMAR JAVIER TÍJARO ROJAS**

KEYWORDS: Fast Fourier Transform(FFT), Digital signals processor(DSP), DSP56F807, development board, liquid crystal displays(LCD), analogical filters.

DESCRIPTION

In this wok is designed a low cost equipment able to perform a frequency analyses based on Fast Fourier Transform, using the digital signals processor Motorola DSP56F807. For the execution of this work was necessary implement development boards and algorithms for a better study and evaluation of the programmable device (DSP), and to carry out a good learning of its several embebbed modules.

The document presented consists of several chapters and includes figures and tables for a better understanding of the techniques used in the design of the equipment, some of these are: Digital signals processors, fundamentals on digital signals processing, liquid crystal displays, tests, performance analyses, conclusions and recommendations. In addition, are presented annexes that allow the learning of the computer programs used, the software implemented in the device, connections manuals and hardware implementation.

For its operation the equipment presented performs the next tasks: first, it acquires a signal which is analogically filtered and sampled using a DSP, this device then performs the digital processing of the incoming signal and finally sends the output data to a liquid crystal display, where the spectrum is visualized. The LCD shows using a bars scheme the spectrum corresponding to the input signal.

* Final Project

** Physical and Mechanical Engineering Faculty; School of Electrical, Electronics and Telecommunications Engineering; Jaime Guillermo Barrero Pérez.

INTRODUCCIÓN

Como se puede ver hoy en día, el desarrollo del mundo va de la mano de la electrónica, por lo tanto, a menudo se observa la creciente necesidad de los equipos digitales en la sociedad, los cuales tienen diversas aplicaciones que buscan satisfacer las necesidades de los usuarios o su gusto por la tecnología. Por ejemplo si se analiza el funcionamiento de equipos como los celulares, se puede encontrar que el motor, o el cerebro de estos equipos son los procesadores digitales de señal (DSP), por eso es una parte fundamental en el estudio de los sistemas digitales.

Los DSP son dispositivos robustos que permiten a partir de señales analógicas o digitales realizar un procesamiento rápido y con poca incertidumbre.

La finalidad de esta investigación fue, a partir de la implementación de tarjetas de desarrollo para los DSP 56F801 y 56F807, desplegar aplicaciones (en uno de estos procesadores), que permitieron visualizar los resultados en una pantalla LCD gráfica. El estudio realizado presenta varias etapas en donde se adquirieron señales (por medio del conversor analógico a digital del DSP) a las cuales se les realizó un análisis en frecuencia para poder visualizar el espectro de las señales utilizando el método de la FFT (Fast Fourier Transform), y calcular de este modo la Transformada Discreta de Fourier con resultados confiables en menor tiempo.

1. PROCESADORES DE SEÑALES DIGITALES

Hoy en día los equipos digitales han demostrado ser más eficientes y mejor que los equipos analógicos en cuanto a número de aplicaciones o de análisis se refiere. Esto se debe a que por medio de estos se pueden tomar muestras de la señal y almacenarlas para luego procesarlas de forma digital, obteniendo algunos resultados que en un equipo analógico no se podrían tener.

Los Procesadores de Señales Digitales (DSP-Sigla en Inglés) son dispositivos que se caracterizan por su alto *throughput*[1], bajo consumo de potencia y bajo costo, en comparación con los microcontroladores; por estas razones estos procesadores se convierten en una de las mejores herramientas para el procesamiento digital de señales, tales como las vistas en las siguientes aplicaciones: Implementación de sistemas telefónicos, aplicaciones de reducción de ruidos, filtros digitales, control y monitoreo remoto, control de luz de tráfico, control de motores, etc.

1.1 FAMILIA DSP56F800 DE MOTOROLA

1.1.1 Arquitectura

Esta familia de DSP con tecnología CMOS de 16 "bits", presenta una CPU de propósito general diseñada para el eficiente procesamiento digital de una señal y el control de operaciones. El repertorio de instrucciones que presenta es amplio facilitando la programación al usuario.

A continuación se mencionarán algunas características generales de la familia DSP56800:

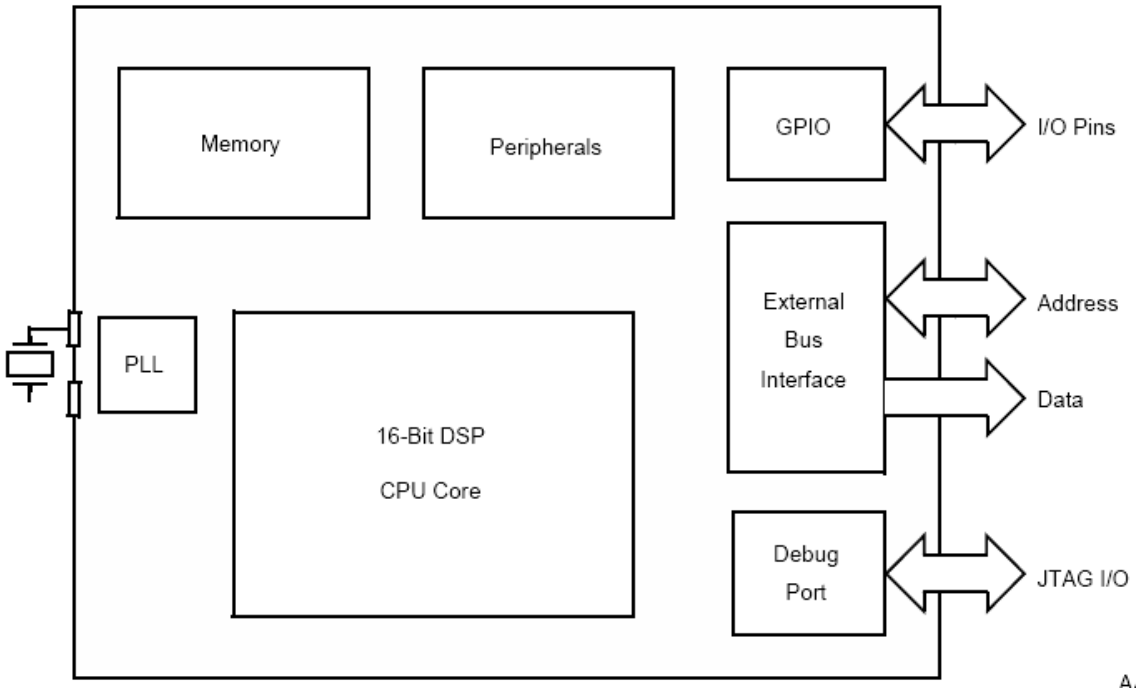
- Es capaz de procesar hasta 35 millones de instrucciones por segundo (MIPS).
- Requiere solo una fuente de alimentación entre 2,7 a 3,6 volt.
- Presenta dos registros acumuladores de 36 "bits", incluyendo extensión de "bits".
- "Hardware" DO y REP.
- Dos pines de interrupciones externas.
- Tres buses de 16 "bits" de datos.
- Tres buses de 16 "bits" de direcciones.

[1] Throughput: Cantidad de datos procesados en un tiempo determinado.

- Repertorio de instrucciones que dan soporte al procesamiento digital de la señal y control de funciones.
- Soporta compilador en C (*Codewarrior*).

Los DSP56800 son un pequeño sistema encapsulado conformado por subsistemas principales como se visualizan en la Figura 1. Estos módulos son: memoria, *Phase Lock Loop* (PLL), Unidad Central de Procesos (CPU), periféricos, puerto de entrada salida, "bus" de interfaz externa [Motorola, DSP56F800 Family Manual, 2003].

Figura 1. Diagrama de bloques de las unidades principales de la familia DSP56800



Fuente: [Motorola, DSP56F800 Family Manual, 2003]

La memoria de los DSP56800 utiliza dos espacios de memorias independientes, los espacios de datos y los de programación. En la tabla 1 se describe la configuración de la memoria y en las figuras 2 y 3, se observan el mapa de la memoria de programación y datos respectivamente.

Tabla 1. Configuración de memoria del DSP.

On-Chip Memory	56F801	56F802	56F803	56F805	56F807
Program Flash (PFLASH)	8K x 16	8K x 16	32252 x 16	32252 x 16	60K x 16
Data Flash (XFLASH)	2K x 16	2K x 16	4K x 16	4K x 16	8K x 16
Program RAM (PRAM)	1K x 16	1K x 16	512 x 16	512 x 16	2K x 16
Data RAM (XRAM)	1K x 16	1K x 16	2K x 16	2K x 16	4K x 16
Program Boot Flash	2K x 16				

Fuente: [Motorola, DSP56F800 User Manual, 2004]

Figura 2. Mapa de memoria de programación.

Begin/ End Address	Mode 0A[1]				Mode 0B[1]				Mode 3[1]
	801/802	803	805	807	801/802	803	805	807	803, 805, 807
0000 0003	BFlash 4	BFlash 4	BFlash 4	BFlash 4	Not Supported	BFlash 4	B Flash 4	BFlash 4	PExternal 64K
0004 1FFF	PFlash 8K-4	PFlash 31.5K-4	PFlash 31.5K-4	PFlash 32K-4		PFlash 31.5K-4	P.Flash 31.5K-4	PFlash 28K-4	
2000 6FFF	Reserved								
7000 73FF								PRAM 2K	
7400 77FF									
7800 7BFF								BFlash 2K	
7C00 7DFF	PRAM 1K								
7E00 7FFF		PRAM 512	PRAM 512			PRAM 512	PRAM 512		
8000 87FF	BFlash 2K	BFlash 2K	BFlash 2K	PFlash 28K		PExternal 32K	PExternal 32K	PExternal 32K	
8800 EFFF	Reserved	Reserved	Reserved						
F000 F3FF				PRAM 2K					
F400 F7FF									
F800 FFFF				BFlash 2K					

Fuente: [Motorola, DSP56F800 User Manual, 2004]

Figura 3. Mapa de memoria de datos.

Begin/ End Address	EX=0				EX=1 803, 805, 807
	801/802	803	805	807	
0000 03FF	XRAM 1K	XRAM 2K	XRAM 2K	XRAM 4K	XExternal 64K
0400 07FF	Reserved				
0800 0BFF		Reserved	Reserved		
0C00 0FFF	Peripherals	Peripherals	Peripherals		
1000 13FF	XFlash 2K	XFlash 4K	XFlash 4K	Peripherals	
1400 17FF					
1800 1FFF	Reserved			Reserved	
2000 2FFF		XExternal 56K-128	XExternal 56K-128	XFlash 8K	
3000 3FFF					
4000 FF7F				XExternal 48K-128	
FF80 FFFF	Core Regs 128	Core Regs 128	Core Regs 128	Core Regs 128	Core Regs 128

Fuente: [Motorola, DSP56F800 User Manual, 2004]

A continuación se nombrarán los módulos periféricos presentes en los DSP56F80x.

- Conversor analógico a digital (ADC).

  Este módulo permite tomar una señal analógica y convertirla en una señal digital para luego procesarla (figura 4).

  El número de módulos de ADC depende del integrado DSP.

  Características (IPBUS 40MHz):

  12 “bits” de resolución.

  Rata de muestreo de hasta 1,666 millones de muestras por segundo.

  Frecuencia de reloj del ADC 5 MHz.

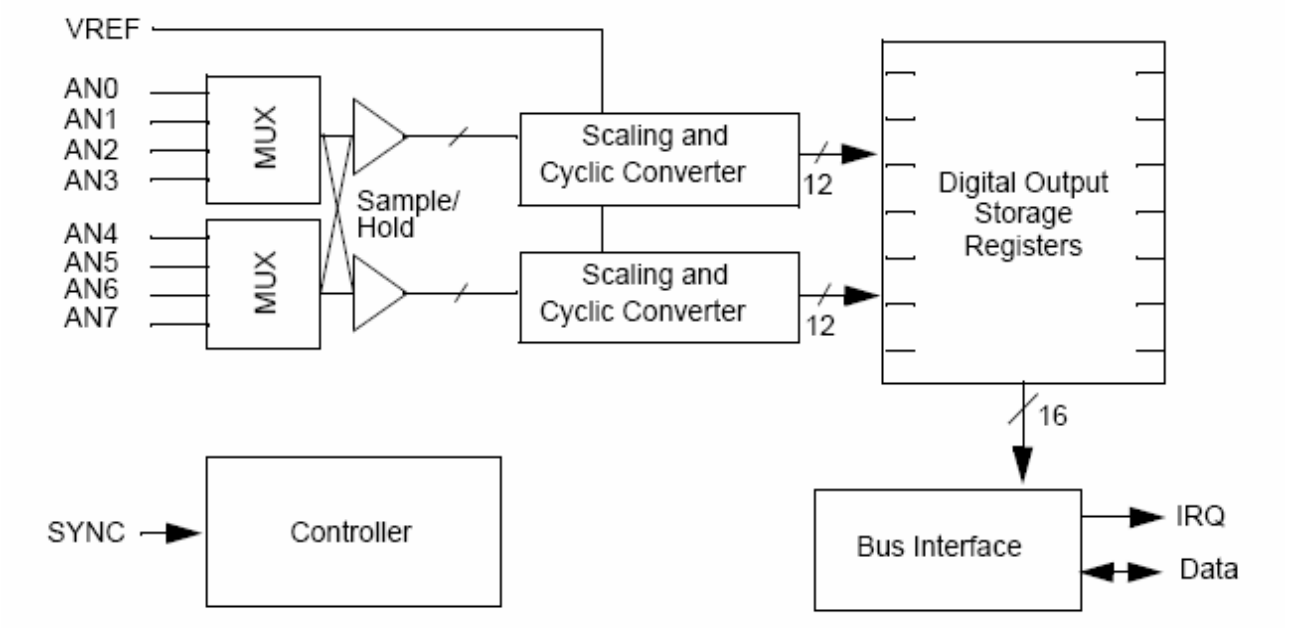
  Tiempo de conversión simple igual a 8,5 ciclos de reloj (8,5*200ns = 1,7μs) y para muestras adicionales 6 ciclos de reloj (6*200ns = 1,2μs). Si se desea realizar 8 conversiones se tardará 26,5 ciclos de reloj (5,3 μs).

  Muestreo secuencial o simultáneo.

Banderas de interrupción opcionales: termino de conversión, cruce por cero, desborde de los límites configurados.

Sincronización con el PWM.

Figura 4. Diagrama de bloque del ADC.

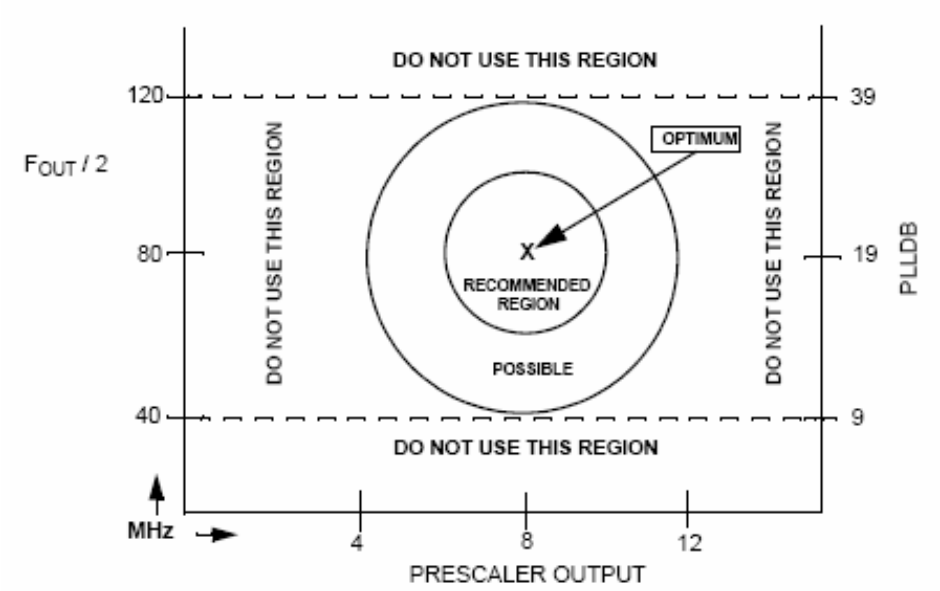


Fuente: [Motorola, DSP56F800 User Manual, 2004]

- Módulo PLL:

  El PLL tiene la propiedad de permitir programar la frecuencia a la cual trabajará el procesador. En la figura 5 se muestra el rango de frecuencias en donde el diseñador recomienda que se trabaje.

Figura 5. Rango de operación del DSP.



Fuente: [Motorola, DSP56F800 User Manual, 2004]

- Interfaz de comunicación serial (SC I):

  Características:

  Este módulo permite la comunicación serial asíncrona con dispositivos periféricos y control de otros.

  Características:

  Comunicación simple y "*full-duplex*".

  Separadamente habilita transmisión y recepción.

  Operación controlada con siete banderas de interrupciones.

  Receptor con detección de error.

  Polaridad programable de recepción y transmisión.

- Interfaz de periféricos seriales (SPI).

  Características:

  Comunicación "*full-duplex*".

  Modo de operación maestro y esclavo.

  Ocho posibles frecuencias de trabajo en modo maestro.

Trabaja a la frecuencia del reloj en modo esclavo.

Programación del orden de desplazamiento al momento de la transmisión y recepción.

Esta interfaz permite la comunicación serial "*full-duplex*", síncrona entre el DSP y dispositivos periféricos, incluyendo otros controladores de 16 "bits".

- Contador/temporizador.

  Este módulo permite contar eventos externos y/o internos. También se utiliza para generar ondas cuadradas a diferentes frecuencias.

  El número de módulos depende del DSP. Por ejemplo los DSP56f801/802 contienen dos, el C y el D. mientras que los DSP56F803/805/807 tienen cuatro: A, B, C y D.

  Características:

  Cada módulo contador/temporizador contiene cuatro contadores/temporizadores de 16 "bits".

  Cuenta de forma ascendente o descendente.

  Se puede configurar para que cuente en cascada.

  Máxima rata de conteo igual a la mitad de la frecuencia de reloj si cuenta eventos externos.

  Varios contadores pueden compartir el mismo pin como entrada.

  Cada contador se puede preescalar de forma independiente.
- Modulación por ancho de pulso (PWM).
- Puertos de entradas/salidas:

  Los DSP56800 presentan pines de propósito general de entrada y/o salidas y el número de estos varía según el DSP.

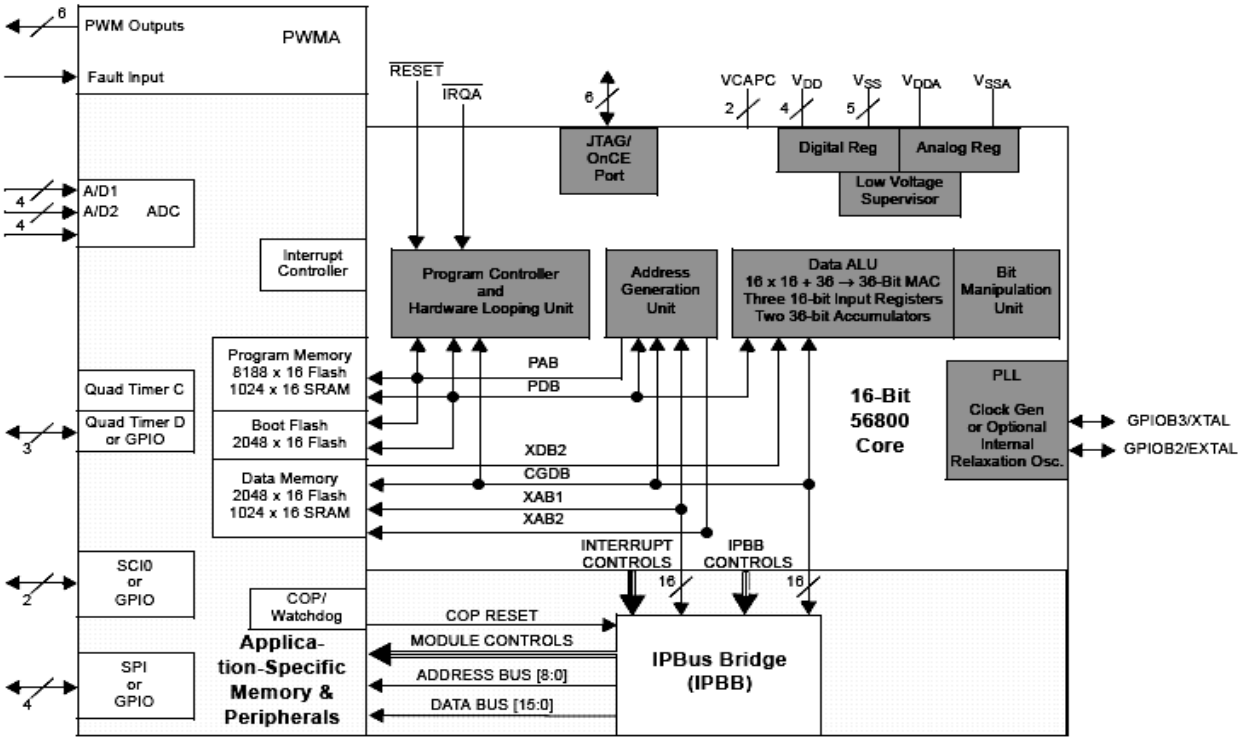
Los DSP56800 están conformados por unidades funcionales que se encargan de realizar un conjunto de operaciones específico. Estas unidades trabajan con 16 "bits" y se mencionan a continuación (ver figura 6):

- Unidad Aritmético Lógica de datos (dato ALU):

  En esta unidad se encarga de realizar todas las operaciones aritméticas y lógicas entre datos.
- Unidad de Generación de Direcciones (AGU):

  Se encarga de realizar el cálculo de la dirección efectiva.

Figura 6. Diagrama de las unidades funcionales del DSP.

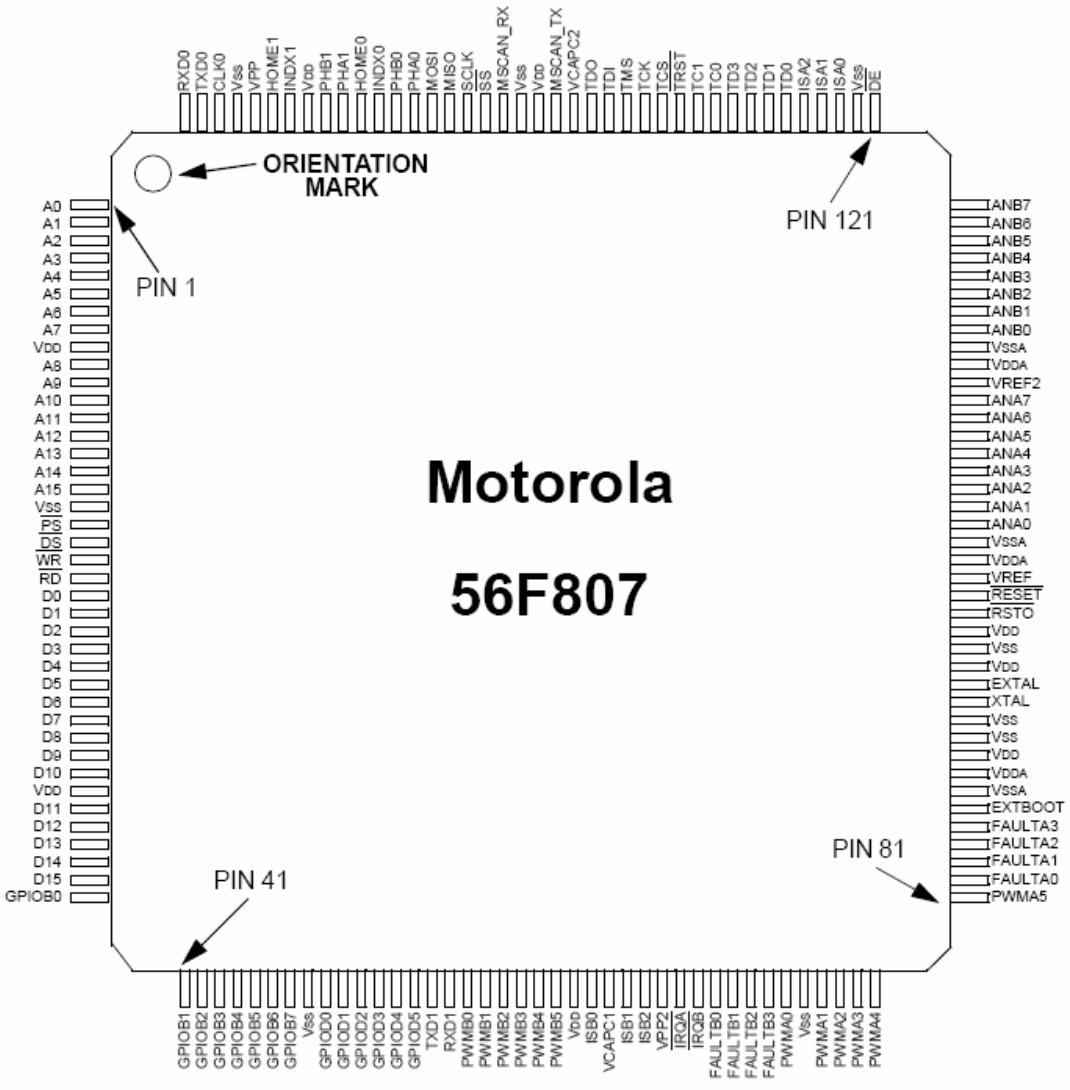


Fuente: [Motorola, DSP56F800 User Manual, 2004]

1.2 DIAGRAMAS DE PINES

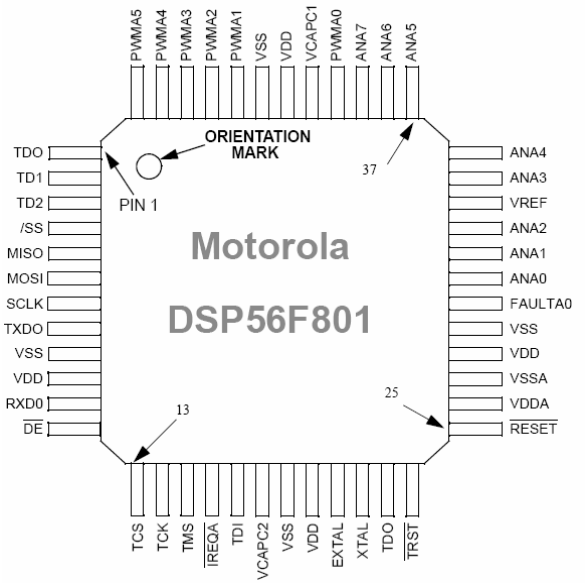
En esta sección se mostrarán los diagramas de pines de los DSPs que se implementaron en las tarjetas de desarrollo. La figura 7 muestra el 56F807 y la figura 8 el 56F801.

Figura 7. Diagrama de pines del 56F807.



Fuente: [Motorola, *DSP56F807*]

Figura 8. Diagrama de pines del 56F801.



Fuente: [Motorola, *DSP56F801*]

1.3 TARJETAS DE DESARROLLO

Los dispositivos como los microcontroladores y los DSPs necesitan ser probados antes de llevarlos a una aplicación específica, es por esta razón que las tarjetas que sirven para desarrollo, son fundamentales para hacer pruebas de los módulos que vienen embebidos en estos circuitos integrados.

Las tarjetas de desarrollo implementadas son las correspondientes al DSP56F801 y DSP56F807, los cuales son el más pequeño y el más grande de la familia de acuerdo al tamaño, la cantidad de pines y módulos que tienen, la capacidad de almacenamiento en memoria y la corriente que consume cada uno. En el anexo D se describen con más detalle cada una de estas tarjetas.

1.3.1 Tarjeta de desarrollo del DSP56F801

Para la construcción de esta aplicación se utilizó el módulo de evaluación de "hardware" para el 56F801, proporcionado por el fabricante (*Motorola*), el cual sirvió como base para realizar este diseño, pero debido a que éste tiene una aplicación para control de motores por PWM, y a que algunos circuitos integrados eran de difícil acceso por su costo, algunos módulos no se

implementaron, como por ejemplo: los sensores de corriente, el módulo de comunicación RS232, las protecciones de sobretensión, los leds del PWM, entre otros [Motorola, DSP56F801EVMUM, 2003].

Se utilizaron tres tarjetas. Una fue para programación por el puerto paralelo del computador, otra para aplicación y una última de acople para conectar o desconectar el DSP. Comunicadas cada una por correas o conectores. A continuación se describirán cada una de las tarjetas.

Para poder descargar en el DSP el programa elaborado en *Codewarrior*, se necesita que sea conectado el cable al puerto paralelo del computador y en el DSP al puerto JTAG (*Join Test Action Group*), cuyo significado en español es grupo de acción para prueba (Anexo F – jtag.pdf).

Con referencia al conector JTAG de 14 pines propuesto por el fabricante, se decidió realizar un cambio ya que 4 pines en este puerto no se conectan. Se rediseñó éste con solo 10 pines haciendo más eficiente el diseño y ganando espacio. Este tipo de interfaz esta detallado en la norma IEEE 1149.1a *StandardTest Access Port and Boundary Scan Architecture*.

La tarjeta de aplicación del DSP56F801 se muestra en el anexo F (tarjetadedesarrollo801.pdf). En esta, se encuentran los siguientes módulos periféricos:

- ADC
- PWM
- GPIO
- Timer
- SPI
- SCI
- JTAG

1.3.2 Tarjeta de desarrollo del DSP56F807

La implementación de esta tarjeta se basó en el módulo de evaluación de "hardware" para el 56F807, proporcionado por el fabricante (*Motorola*), pero ya que éste tiene una aplicación para

control de motores por PWM, y a que algunos circuitos integrados eran de difícil acceso por su costo, algunos módulos no se tuvieron en cuenta [Motorola, DSP56F807EVMUM, 2003].

Se fabricaron dos tarjetas (a diferencia de las tres implementadas en el DSP56F801), una de ellas con los módulos de programación y desarrollo, este último con todos los pines disponibles como puertos o como módulos embebidos y con un conector tipo NEB21R[2] el cual se alimentó con una fuente que tuviese una tensión DC entre 7,5 V - 12 V y una corriente de 500mA típicamente; y otra para acoplar el DSP con la tarjeta anteriormente descrita (ver anexo F – tarjetadedesarrollo807.pdf).

1.3.3 Tarjeta de desarrollo para la aplicación de analizador de espectros

Para el diseño del analizador de espectros se escogió el DSP 56F807 debido a que esta aplicación amerita una cantidad de memoria RAM que el 56F801 no poseía.

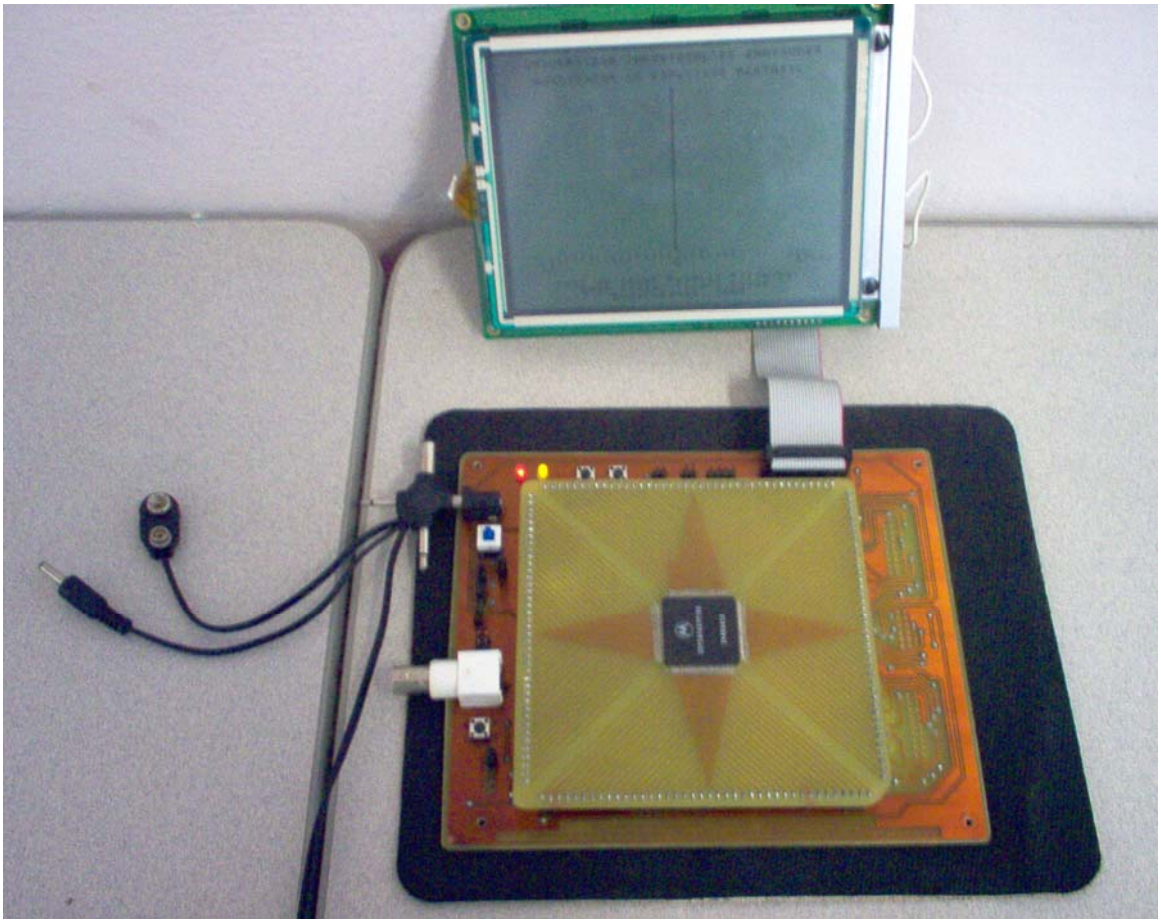
Los módulos utilizados en esta tarjeta fueron los de ADC (ANA0 y ANA4), GPIO (GPIOB, GPIOE0 y GPIOE1), RESET, JTAG, IRQA, IRQB, memoria externa (dirección – puerto A, datos – puerto D, y control), SCI1, y terminales para conexión de dos baterías de aproximadamente 5 V a 1600 mAh. A su vez, se implementaron *jumpers* para utilizar la memoria externa o interna, el oscilador por cristal o una señal de reloj externa, y uno para habilitar el filtro en el canal ANA0. Si se activa el filtro, la entrada presenta un conector de tipo coaxial; si no está conectado el *jumper*, la entrada ANA0 es por el conector tipo *jumper*. Para ver el diagrama de pines remítase al anexo F – equipoFFTportatil.pdf.

Los canales del ADC ANA0 y ANA4 fueron implementados debido a que estos se pueden configurar en modo diferencial. Para la visualización en la pantalla LCD fueron utilizados el puerto B para datos y dos pines del puerto E para control. El RESET se implementó por seguridad. Para una posible programación del DSP con mejoras a la aplicación se añadió el puerto JTAG. Las interrupciones IRQA, IRQB y la memoria externa se adicionaron para brindarle flexibilidad a la tarjeta en el diseño de otra aplicación donde se utilice control externo y/o expansión de memoria.

En la figura 9 se puede observar la tarjeta diseñada para la aplicación del analizador de espectros.

[2] Conector para entrada de tensión 7.5 V - 12 V de polarización (librería con-lumberg de *Eagle 4.01*)

Figura 9. Tarjeta de desarrollo para el analizador de espectros.



Fuente: Los autores.

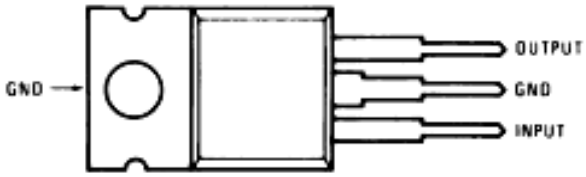
El circuito esquemático de esta aplicación se encuentra en el anexo F en el archivo equipoFFTportatil.pdf.

1.3.4 Fuente de alimentación

En las tarjetas de desarrollo se utilizaron diferentes implementos para la correcta alimentación del circuito. Para el 56F801 se utilizó un conector para una entrada de 3,3 V DC a 300 mA, mientras que en el 56F807 se usó el mismo conector pero con 9 V DC a 500 mA, en el cual se colocó en serie un regulador de 5,0 V LM78L05 (figura 10), y el TPS7133QP (figura 11) regulador de 3,3 V. El

LM78L05 puede entregar una corriente de 1 A y se colocó para garantizar que cuando se alimentara con una fuente de un valor mayor que el de tensión recomendada, no se presentaran problemas en el regulador fijo a 3,3 V. Los problemas de calentamiento de este dispositivo se mejoraron con un disipador comercial de calor colocado en la parte superior del elemento.

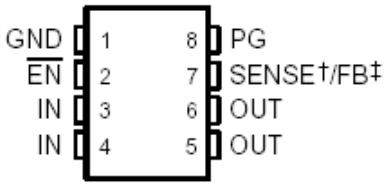
Figura 10. Diagrama de pines del LM78L05**.**



Fuente: LM78LXX.pdf

Se escogió el TPS7133QP (figura 11) producido por la empresa *Texas Instruments,* debido a que las características del dispositivo coincidían con las necesidades del circuito. En términos generales, el regulador puede recibir una entrada de tensión entre 3,5 V y 10 V y su salida es fija de 3,3V entregando máximo 500 mA.

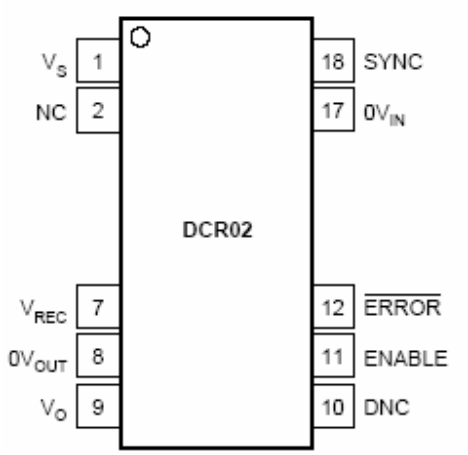
Figura 11. Diagrama de pines del regulador a 3,3 V.



Fuente: www.ti.com

Para la alimentación de la tarjeta de desarrollo del analizador de espectros, se utilizaron los integrados DCR021205 fabricados por *Texas Instruments*, los cuales regulan de 12 a 5 V y además estas tensiones son aislados por un transformador interno entregando una potencia máxima de 2W. En la figura 12 se visualiza el diagrama de pines de este circuito integrado.

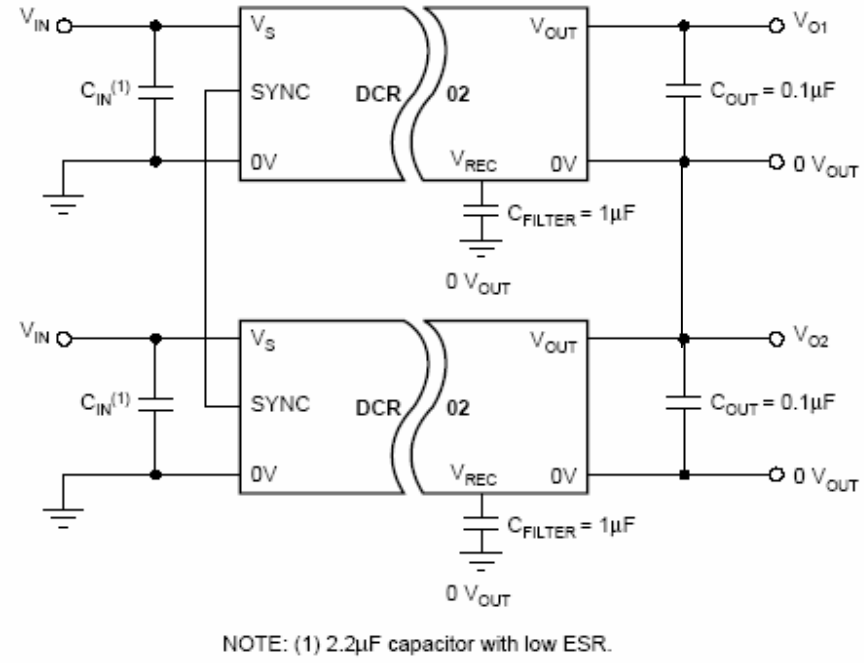
Figura 12. Diagrama de pines del DCR021205.



Fuente: www.ti.com

Otro objetivo por el cual se implementó este integrado, fue por que se necesitaba alimentación dual de 5 V para el integrado TL084 del cual se comentará en la sección 2.1.1. El diagrama de conexiones de este dispositivo es mostrado en la figura 13.

Figura 13. Diagrama de conexiones para alimentación dual.



Fuente: www.ti.com

Para la alimentación del DSP se utiliza el TPS76833 de *Texas Instruments*, el cual tiene el mismo diagrama de pines mostrado en la figura 10, y sus características son parecidas pero, el empaquetado de este es SOIC y la corriente que puede entregar es de 1 A.

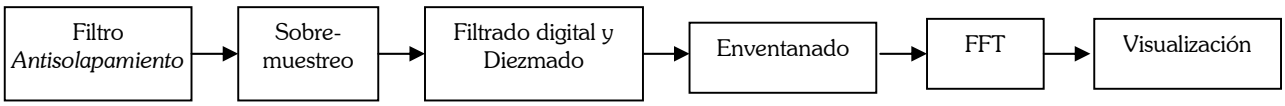
2. TRATAMIENTO DIGITAL DE SEÑALES

En la actualidad los equipos de medición digitales han demostrado ser más confiables, tener mayor precisión y aplicabilidad que los equipos analógicos, esto se debe en gran parte al avance tecnológico y a los numerosos estudios realizados sobre tratamiento digital de señales, que han sido muy importantes en el desarrollo de la electrónica.

2.1 ETAPAS DEL EQUIPO DIGITAL

Todo equipo en donde se necesite un procesado digital de señal debe por lo menos contener las etapas observadas en la figura 14.

Figura 14. Diagrama de bloques general de un sistema de tratamiento digital de señales.



Fuente: Los autores.

2.1.1 Filtro antisolapamiento

Cuando se muestrea una señal continua en el dominio del tiempo, existe la posibilidad de encontrar solapamiento (*aliasing*[3]) de frecuencias. Debido a esto, se debe implementar un filtro analógico pasabajas que evite que se muestreen señales que no corresponden a las componentes de la frecuencia de la señal.

Se diseñó para esto en "hardware" un filtro *Butterworth* pasabajas, ya que la respuesta es máximamente plana en amplitud en el ancho de banda deseado (20 kHz). El cálculo de los parámetros de éste se halló utilizando el "software" *FilterPro* de *Texas Instruments*. Esta herramienta arroja un esbozo gráfico del circuito esquemático y la respuesta en frecuencia (magnitud y fase)

[3] Aliasing: Fenómeno que se presenta cuando se muestrea una señal a una frecuencia menor que la mitad de ella. En éste, las frecuencias altas se pueden observar como bajas.

donde los argumentos de entrada son: tipo de filtro, orden, frecuencia de corte y clase de filtro (LP, BP, HP), entre otros.

El orden del filtro para lograr las atenuaciones adecuadas fue de orden cuarto, cuya frecuencia de paso (-3dB) fue de 20kHz, y la frecuencia de corte (-80dB) de 280kHz. En un diseño de detección de armónicos de la señal eléctrica existe el criterio de -98dB de atenuación, pero la razón por la que se escogió la atenuación de -80dB y no la de -98dB (que corresponde a una ganancia de 12,5 uV/V), es que el nivel de cuantificación del ADC del DSP es de 806 uV aproximadamente, es decir cualquier señal de menor valor a esta se pierde y por lo tanto el filtro no tiene que ser tan estricto. Esto se puede observar claramente en la ecuación 1 [Motorola, DSP56F800 User Manual, 2004].

$$\Delta = \frac{Vref}{4096} = \frac{3.3}{4096} = 805{,}6640 \text{ uV} \qquad \Rightarrow \qquad \Delta dB = -61.8769 \text{ dB} \quad (1)$$

Este filtro se diseñó usando el integrado TL084, que presentan características deseables como son: 4 amplificadores operacionales, adecuado ancho de banda (3 MHz), rechazo en modo común de 86dB típicamente, y voltaje de offset de 3 a 15 mV.

La simulación de este filtro se realizó en *Orcad 9* (figura 15 y figura 16).

Figura 15. Diagrama esquemático del sistema acondicionador de la señal analógica.

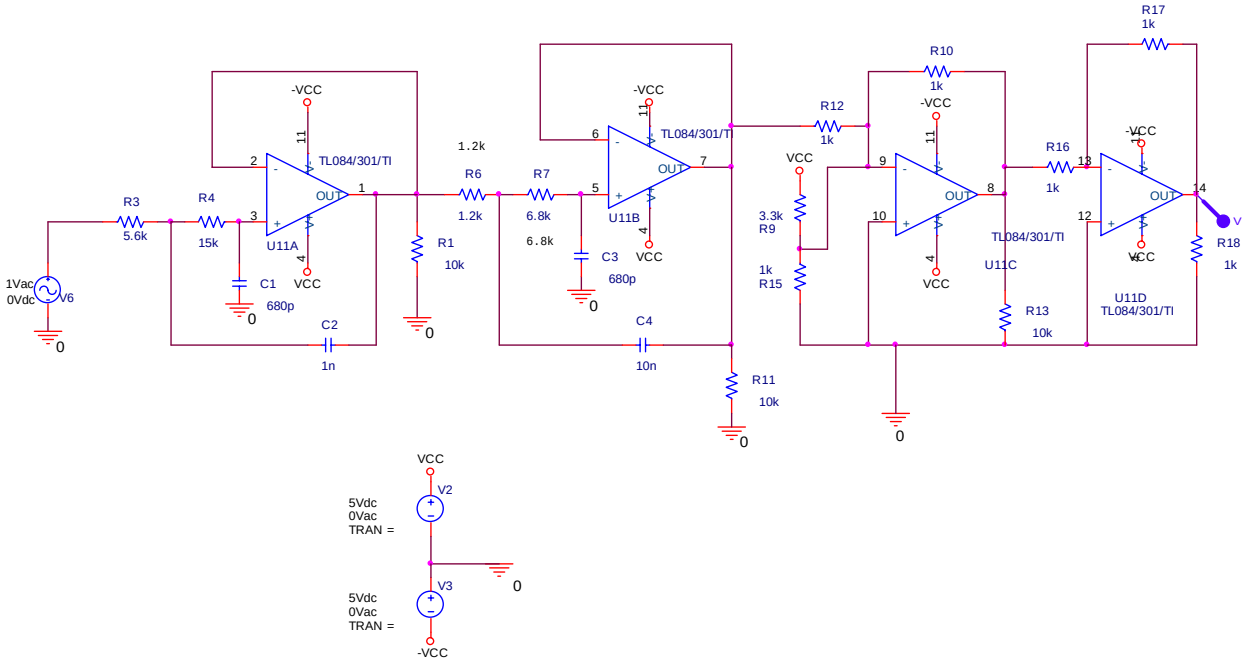
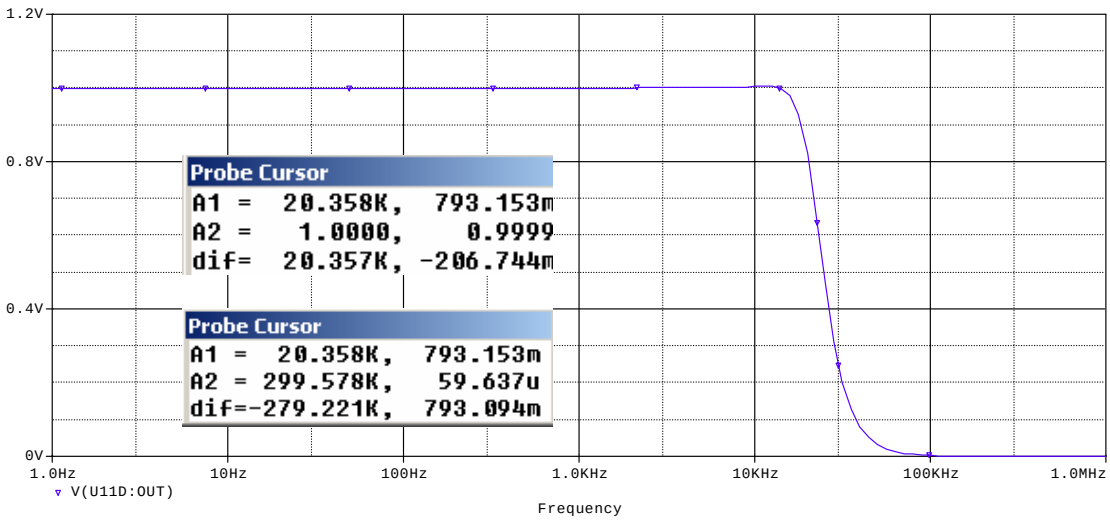


Figura 16. Respuesta en frecuencia del sistema acondicionador de la señal analógica.



2.1.2 Sobremuestreo

En esta etapa se adquiere la señal por medio del conversor analógico a digital del DSP a una frecuencia de muestreo de 600kHz ±12kHz (esta tolerancia de la frecuencia se explica en la sección

4.2.1, y la frecuencia de muestro corresponde al doble de la frecuencia de corte diseñada en el filtro en la sección 2.1.1) tomando aproximadamente 29 muestras en un periodo de la señal de más alta frecuencia (20kHz). Además se toman 1442 muestras para tener una mejor resolución en frecuencia (la justificación de este número de muestras se presenta en la sección 2.1.3).

2.1.3 Filtrado digital

Teniendo las muestras almacenadas en la memoria RAM del DSP, se procede a realizar un filtrado. Este filtrado tiene como objetivo atenuar de forma más selectiva las frecuencias que deja pasar el filtro analógico (sección 2.1.1). Este filtro se escogió FIR ya que estos presentan fase lineal, éste se obtuvo utilizando los comandos de Matlab ***fir2*** y ***buttord***. El comando *buttord* se utilizó para encontrar el orden del filtro a partir de los argumentos: frecuencia de paso, frecuencia de corte, atenuación frecuencia de paso y de frecuencia de corte. El resultado se utilizó como argumento de la función *fir2* para hallar los coeficientes del filtro digital.

El orden del filtro obtenido fue de N=417, dándole al comando *buttord* los siguientes argumentos: frecuencia de paso 20 kHz y frecuencia de corte de 20,5kHz, atenuación de -3db y –90 dB respectivamente, estos últimos parámetros se utilizaron para darle más selectividad al filtro. El comando fir2 arrojó como resultado un vector de coeficientes de 418 elementos que fueron almacenados en la memoria *Flash* del DSP (ver anexo C – vector coeFIR). Se realizó la convolución en el tiempo de este vector y el de muestras, lo que equivale al filtrado de la señal. En la sección 2.1.2. se mencionó de 1442 muestras, las cuales corresponden a 1024 (2^N, para realizar el diezmado en 8 y luego poder implementar el algoritmo de *radix-2*) más 418 (para garantizar el filtrado de todos los elementos del vector de muestras).

Al finalizar la operación de filtrado se obtiene un vector de 1024.

2.1.4 Diezmado

En este sistema el diezmado tiene como fin eliminar componentes en el tiempo discreto para que la resolución en frecuencia se visualice mejor. El resultado obtenido de esta operación es un vector más pequeño, en donde se encuentra distribuido de una mejor manera el ancho especificado.

El diezmado realizado es por 8, teniéndose como vector final un arreglo 128 elementos para calcular la FFT (ver anexo C – función main).

2.1.5 Enventanado

Esta técnica se utiliza para obtener una mejor respuesta en la amplitud de los espectros de las muestras. Se implementó la ventana de *Hanning*, ya que era la que mejor se adaptaba al sistema para la visualización de los resultados en la pantalla gráfica en comparación con la rectangular, es decir, permitía diferenciar dos componentes espectrales en regiones cercanas, y además su ecuación es de fácil ejecución ya que realiza menos operaciones que ventanas como la *Blackman* que tienen una respuesta similar con mayor carga computacional.

La respuesta al impulso de la ventana de *Hanning* se observa en la ecuación 2 [Proakis y Manolakis, 1998].

$$h[n] = 0.5 - 0.5\cos\left(\frac{2\pi \cdot n}{N+1}\right) \qquad (2)$$

N: número de muestras.
n: variable independiente.

El comando *TFR1_tfr16CosPIx(x)* de *Codewarrior* se encuentra en la librería *tfr16.h*, y se utilizó para calcular el coseno(π·x). De esta forma se puede implementar la ventana *Hanning* (ver anexo B y C).

2.1.6 Transformada de Fourier

Para realizar un análisis en frecuencia se necesita conocer la teoría básica de señales y por ende la Transformada Discreta de Fourier (DFT del inglés *Discrete Fourier Transform*), que es el equivalente discreto de las Series de Fourier. La Transformada de Fourier X(w) de una señal analógica x(t) se muestra en la ecuación 3.

$$X(\varpi) = \int_{-\infty}^{+\infty} x(t) \cdot e^{-j\omega t} dt \quad (3)$$

La Transformada Discreta de Fourier (ecuación 4) es un método muy eficiente para determinar el espectro en frecuencia de una señal. Esta, permite convertir una secuencia de valores en el dominio del tiempo a una secuencia de valores equivalente en el dominio de la frecuencia. La Inversa de la Transformada Discreta de Fourier (IDFT) realiza el proceso contrario (ecuación 5).

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot W^{nk} \qquad \text{k=0,1,…, N-1} \qquad (4)$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \cdot W^{-nk} \qquad \text{n=0,1,…, N-1} \qquad (5)$$

La constante "*W*" es conocida como ***factor de Fourier*** y definida como lo muestra la ecuación 6.

$$W = e^{-j2\pi/N} \quad (6)$$

Se observa que 'W' es una función que depende de N, por ello, también suele expresarse como W_N. El inconveniente de realizar unos algoritmos que implementen tal cual estas ecuaciones es la cantidad de tiempo requerido para obtener la salida. Esto es debido a que los índices k y n deben variar de 0 a N-1 para conseguir el rango de salida completo y, por lo tanto, se deben realizar N^2 operaciones. Un algoritmo que realiza eficientemente la DFT es la Transformada Rápida de Fourier (FFT del inglés *Fast Fourier Transform*). Con la FFT se obtienen los mismos resultados que la DFT pero más rápidamente debido a que reduce el número de cálculos requeridos. El término genérico Transformada Rápida de Fourier abarca distintos algoritmos con distintas características, ventajas y desventajas.

En la ecuación de la Transformada Discreta de Fourier, obtener X(k) para un 'k' determinado requiere aproximadamente N sumas complejas y N productos complejos (ecuación 7).

$$X(k) = x(0) + x(1) \cdot W^{k} + x(2) \cdot W^{2k} + x(3) \cdot W^{3k} + \ldots + x(N-1) \cdot W^{(N-1)k} \qquad (7)$$

Para k = 0, 1,..., N-1.

Si lo que se desea es obtener X(0), X(1), ..., X(N-1) entonces se necesitarían un total de aproximadamente N^2 sumas complejas y N^2 productos complejos. Esto quiere decir que los requerimientos computacionales de la DFT pueden ser excesivos especialmente si el tamaño de N es grande.

La FFT aprovecha la periodicidad y simetría del factor de Fourier 'W' para el cálculo del Transformada Discreta de Fourier. La periodicidad de 'W' implica la ecuación 8 y su simetría implica la ecuación 9.

$$W^k = W^{k+N} \quad (8)$$

$$W^k = -W^{k+N/2} \quad (9)$$

La FFT descompone la DFT de N puntos en transformadas más pequeñas. Una DFT de N puntos es descompuesta en dos DFT's de (N/2) puntos. Cada DFT de (N/2) puntos se descompone a su vez en dos DFT's de (N/4) puntos y así sucesivamente. Al final de la descomposición se obtienen (N/2) DFT´s de 2 puntos cada una. La transformada más pequeña viene determinada por la base de la FFT. Para una FFT de base 2, N debe ser una potencia de 2 y la transformada más pequeña es la DFT de 2 puntos. Para implementar la FFT existen dos procedimientos: diezmado en frecuencia (DIF del inglés *Decimation In Frequency*) y diezmado en el tiempo (DIT del inglés *Decimation In Time*).

Para explicar el diezmado en frecuencia de la Transformada Discreta de Fourier se utilizará una secuencia x(n) de N puntos como la mostrada en la ecuación 4.

Si la secuencia se separa en dos mitades se tiene:

$$x(0), x(1), \ldots, x\left(\frac{N}{2}-1\right) \qquad x\left(\frac{N}{2}\right), x\left(\frac{N}{2}+1\right), \ldots, x(N-1)$$

La ecuación para la transformada rápida de Fourier también se puede separar en dos sumatorias como se visualiza en la ecuación 10.

$$X(k) = \sum_{n=0}^{(N/2)-1} x(n) \cdot W^{nk} + \sum_{n=N/2}^{N-1} x(n) \cdot W^{nk} \qquad (10)$$

Si en el segundo término de la sumatoria se hace un cambio de variable (n = n + N/2), se puede obtener la ecuación 11.

$$X(k) = \sum_{n=0}^{(N/2)-1} x(n) \cdot W^{nk} + W^{Nk/2} \sum_{n=0}^{(N/2)-1} x(n + \frac{N}{2}) \cdot W^{nk} \qquad (11)$$

Donde $W^{Nk/2}$ se toma fuera de la segunda sumatoria porque no depende de n. Esto se puede observar en la ecuación 12.

$$W^{Nk/2} = \ell^{-jk\pi} = (\ell^{-j\pi})^k = (\cos\pi - jsen\pi)^k = (-1)^k \qquad (12)$$

Por lo que la ecuación 11 puede convertirse en la ecuación 13.

$$X(k) = \sum_{n=0}^{(N/2)-1} \left[x(n) + (-1)^k \cdot x\left(n + \frac{N}{2} \right) \right] \cdot W^{nk} \qquad (13)$$

Debido a que $(-1)^k = 1$ para k pares y –1 para k impares, la ecuación anterior puede ser dividida en dos ecuaciones, una para los k pares y otra para los k impares (ecuaciones 14 y 15).

$$X(k) = \sum_{n=0}^{(N/2)-1} \left[x(n) + x\left(n + \frac{N}{2} \right) \right] \cdot W^{nk} \qquad (14) \qquad \text{para k par.}$$

$$X(k) = \sum_{n=0}^{(N/2)-1} \left[x(n) - x\left(n + \frac{N}{2} \right) \right] \cdot W^{nk} \qquad (15) \qquad \text{para k impar.}$$

Sustituyendo k = 2k para los k pares, y k = 2k+1 para los k impares se obtienen las ecuaciones 16 y 17.

$$X(2k) = \sum_{n=0}^{(N/2)-1} \left[x(n) + x\left(n + \frac{N}{2} \right) \right] \cdot W^{2nk} \qquad (16) \qquad \text{k=0,1,…, (N/2)-1.}$$

$$X(2k+1) = \sum_{n=0}^{(N/2)-1} \left[x(n) - x\left(n + \frac{N}{2}\right) \right] \cdot W^{n} \cdot W^{2nk} \quad (17) \qquad k=0,1,\ldots, (N/2)-1.$$

Debido a que "W" es una función de longitud N, puede escribirse como "W_N'" y, de la misma manera, "$(W_N)^{2}$" puede escribirse como "$W_{N/2}$". Esto permite escribir de forma más clara las ecuaciones anteriores. Para facilitar el manejo matemático, se llamará a(n) y b(n) a las expresiones mostradas en la ecuación 18 y 19.

$$a(n) = x(n) + x\left(n + \frac{N}{2}\right) \quad (18)$$

$$b(n) = x(n) - x\left(n + \frac{N}{2}\right) \quad (19)$$

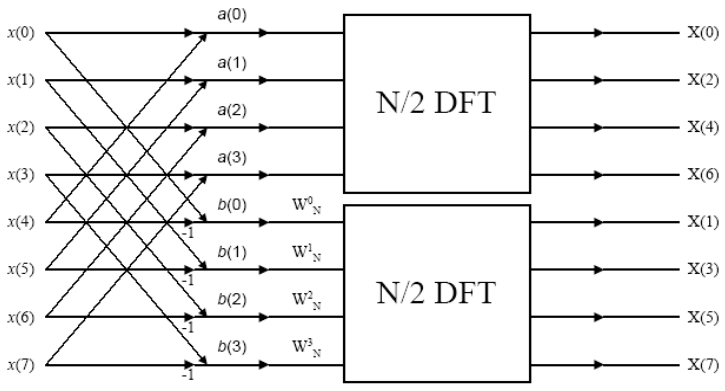
Después de tal tratamiento se tienen las ecuaciones 20 y 21.

$$X(2k) = \sum_{n=0}^{(N/2)-1} a(n) \cdot W_{N/2}^{nk} \qquad (20) \qquad k=0,1,\ldots, (N/2)-1.$$

$$X(2k+1) = \sum_{n=0}^{(N/2)-1} b(n) \cdot W_{N}^{n} \cdot W_{N/2}^{nk} \qquad (21) \qquad k=0,1,\ldots, (N/2)-1.$$

La figura 17 muestra la descomposición de una DFT de *N* puntos en dos DFT de *N*/2 puntos para el caso de $N = 8$. Aplicando (16) y (17) es posible llegar a obtener los *X*'s de la primera etapa de la FFT. Este procedimiento se denomina mariposa.

Figura 17. Descomposición de una DFT de *N* puntos en dos DFTs de *N*/2 puntos, para $N = 8$.



Fuente: [Posadas, 1998]

Esta sección estuvo basada en el documento [Posadas, 1998].

En *Codewarrior* se utilizó el comando *DFR1_dfr16CFFT(a,b,c)* de la librería *dfr16.h* para calcular la FFT de los valores de b, donde b es un vector de entrada de longitud 2^N, a es un vector temporal y c es el vector resultado (FFT).

Se aprovechó esta librería encontrada en los *Beans* por la optimización en memoria.

2.1.7 Visualización

Para la visualización de los resultados se utilizó una pantalla gráfica de cristal líquido fabricada por *Powertip Technology Corporation* [LCD Gráfico], cuya resolución fue de 320x240 píxeles. La interfaz (gráfica y caracter) de la pantalla fue controlada con el DSP, en donde se diseñaron funciones para facilitar la programación y optimizar el algoritmo mostrado en el anexo C.

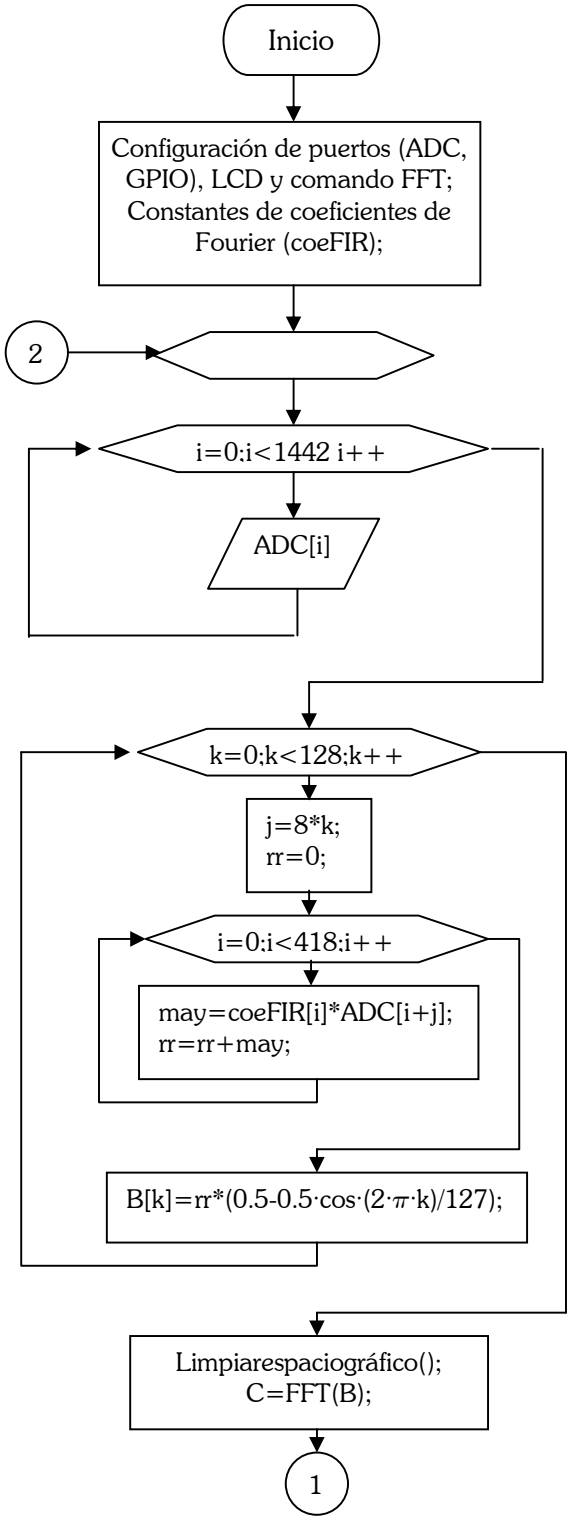
En esta pantalla se mostró la FFT con una resolución en frecuencia de 585,93 Hz $\left(\frac{600k}{1024}\right)$, y un SPAM de 2343,75 Hz (585,93 x 4). Para garantizar que se pueda observar la componente espectral de mayor potencia, el vector de la FFT fue normalizado con el elemento de mayor valor absoluto, siendo el valor de la unidad 160 píxeles.

El programa de la aplicación realizado se remite al Anexo C, donde se encuentra el código en C del algoritmo implementado.

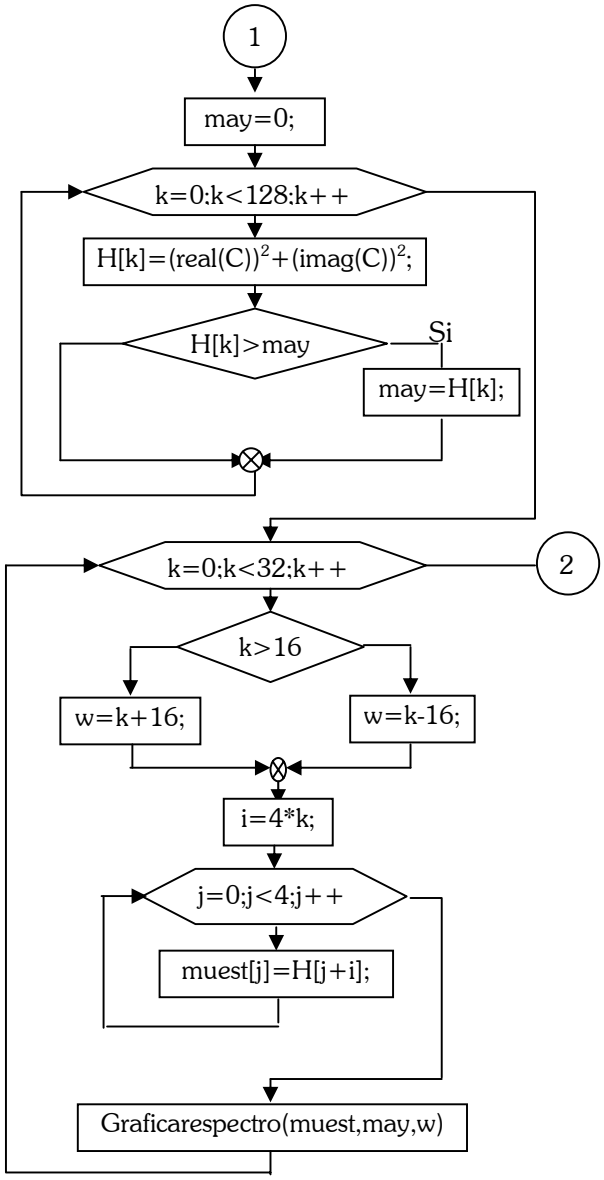
2.2 DIAGRAMA DE FLUJO DEL PROCESADO

En la figura 18 se muestra el diagrama de flujo del tratamiento digital de la señal, la cual es la aplicación realizada en *Codewarrior* para implementar el analizador de espectros. Esta es mostrada en el anexo C, donde se encuentran cada una de las funciones del programa que no se detallan en la figura 18.

Figura 18. Diagrama de flujo del procesado de la señal.



Continúa en la siguiente página



Fuente: Los autores.

3. PANTALLAS DE CRISTAL LÍQUIDO

En la actualidad existen equipos de visualización tanto mecánicos como electrónicos, que permiten a un operador tener una idea aproximada de lo que está ocurriendo en un proceso. Los equipos mecánicos se caracterizan por tener agujas indicadoras y en algunos casos engranajes que indican el conteo de una variable determinada; sin embargo, este tipo de medidores son lentos y poco precisos teniendo en cuenta la resolución de los aparatos y las incertidumbres de visualización. Por otro lado los equipos de visualización electrónicos son más rápidos y precisos, haciéndolos más robustos y prácticos cuando se van a medir variables donde la velocidad y resolución son importantes.

Las pantallas de cristal líquido (LCD - *Liquid Crystal Display*) son los elementos de visualización más comunes en el mercado. Estos dispositivos se encuentran en diferentes tamaños y por lo general de tipo carácter y gráfico.

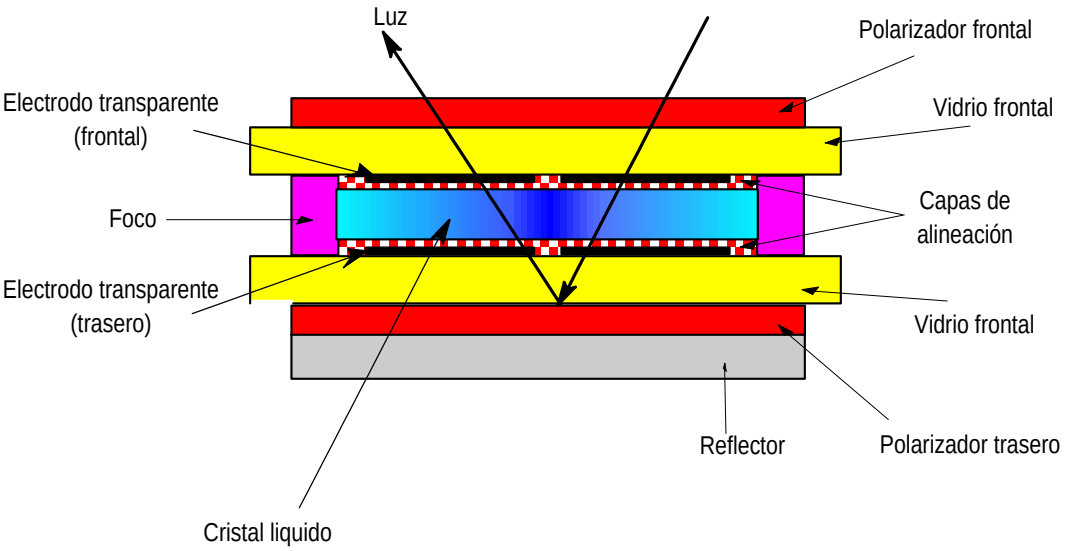
3.1 ASPECTOS GENERALES

Las pantallas de cristal líquido más conocidas como LCD (*Liquid Crystal Display*), fueron desarrolladas al inicio de la década de los 70's y están físicamente compuestas por dos capas de vidrios que encierran en sus paredes líquidos con propiedades ópticas especiales. A diferencia de los demás tipos de pantallas, las LCD son ópticamente pasivas, es decir, no emiten luz sino que bloquean su paso, por lo tanto se caracterizan por consumir cantidades mínimas de energía. [Jaquenod LCD]

El término "cristal líquido" se refiere a la sustancia contenida entre las dos capas de la pantalla, que está compuesta por un enorme número de cristales en forma de hebras en suspensión en un líquido. En las caras internas de las capas de vidrio están impresos electrodos transparentes con las formas que definen los segmentos, píxeles u otros símbolos de la pantalla, sobre esos electrodos existe una capa de polímero con unas micro ranuras que sirven para alinear la orientación pasiva de las moléculas de cristal del líquido (donde esas micro ranuras se orientan perpendicularmente entre sí en las dos capas de vidrio). A su vez, en las caras externas de las capas de vidrio están laminados

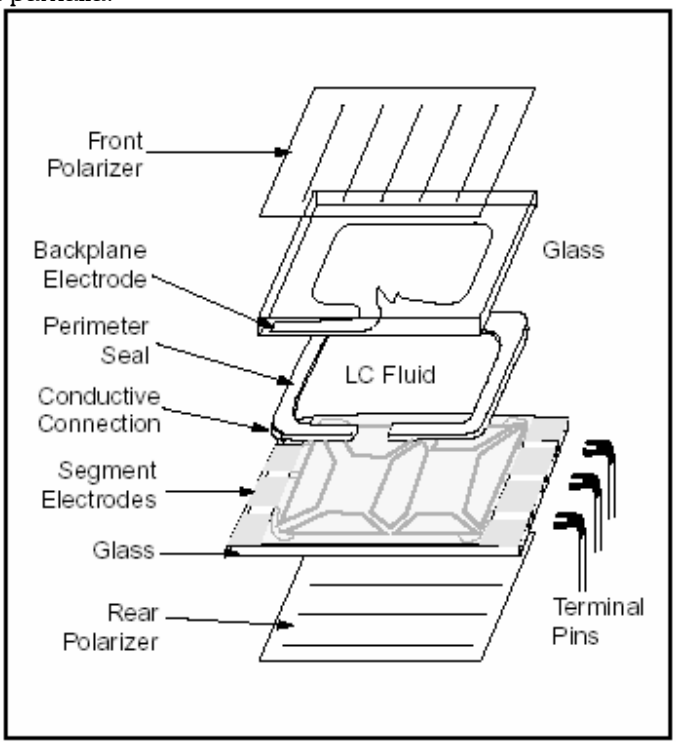
films de polarizadores orientados entre si con una rotación de 90 grados (en los más comunes), o con una idéntica orientación (véase figura 19 y 20). [Jaquenod LCD]

Figura 19. Descripción física de una pantalla LCD común, vista lateral.



Fuente: Los autores.

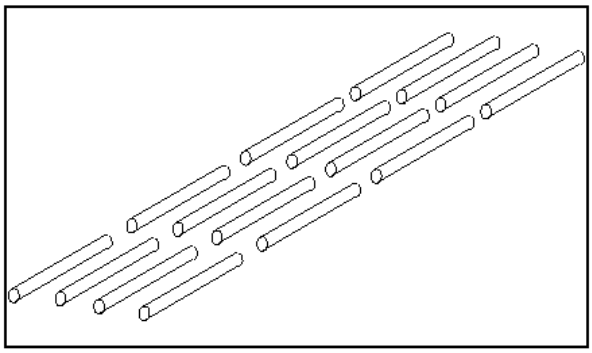
Figura 20. Capas de la pantalla.



Fuente: [Microchip, *LCD*]

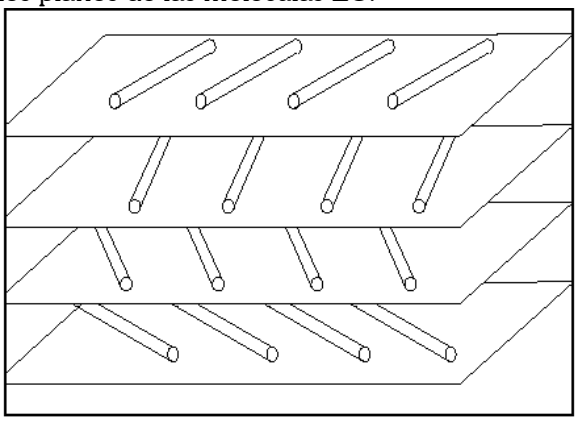
Los segmentos de electrodos vistos en la figura 20, son los encargados de polarizar el fluido del líquido del cristal (LC fluid). Las moléculas de cristal líquido son largas y cilíndricas y en cada plano con fluido LC se alinean quedando todas en paralelo (como lo muestra la figura 21); en los otros planos de fluidos sus moléculas LC también se alinean quedando una orientación de 90 grados entre cada plano como se mencionó anteriormente (figura 22). [Microchip, *LCD*]

Figura 21. Orientación de las moléculas LC.



Fuente: [Microchip, *LCD*]

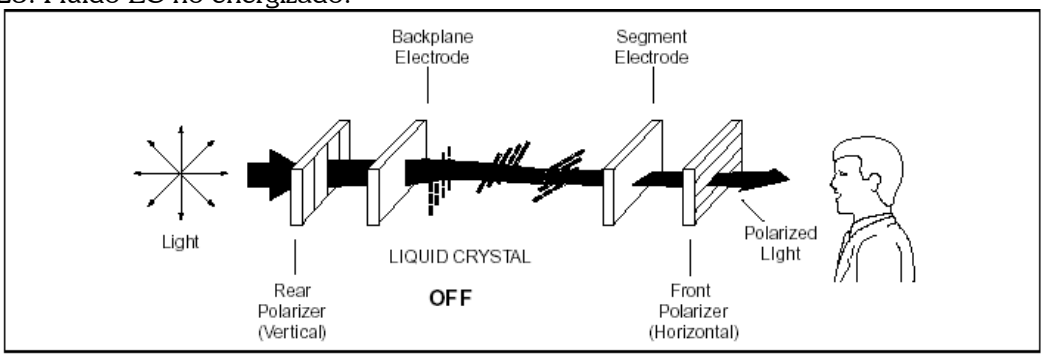
Figura 22. Orientación de los planos de las moléculas LC.



Fuente: [Microchip, *LCD*]

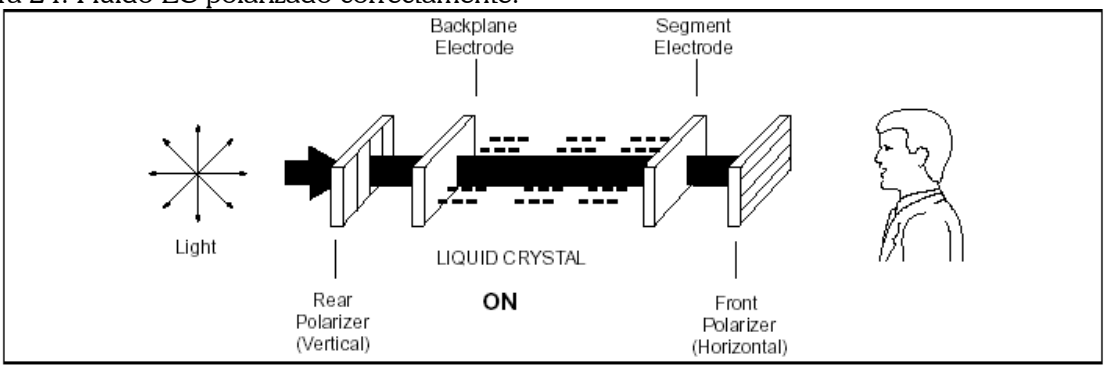
El movimiento de las moléculas LC son la base para la operación de las LCD. La figura 23 muestra como funcionan estos módulos cuando las moléculas no están polarizadas. Y en la figura 24 se visualiza cuando está polariza adecuadamente una pantalla LCD. [Microchip, *LCD*]

Figura 23. Fluido LC no energizado.



Fuente: [Microchip, *LCD*]

Figura 24. Fluido LC polarizado correctamente.



Fuente: [Microchip, *LCD*]

La diferencia en las figuras 23 y 24, es que si las moléculas están polarizadas como es debido, el fluido no hace rotar la luz y la deja pasar como está debidamente programada.

3.2 TIPOS DE PANTALLA

Debido al gran número de aplicaciones que se hacen necesarias, se han creado pantallas LCD tipo carácter y tipo gráfico. Estas, vienen en diferentes resoluciones y tamaños según los requerimientos del usuario, y por lo tanto existen varias empresas que se dedican al desarrollo de estas pantallas como *Powertip Technology, Hyundai, Optrex y Display Tech*, entre otras.

Las aplicaciones implementadas en el desarrollo del proyecto mostraron la necesidad de una pantalla LCD gráfica, por que al observar medidas en ésta, se presenta más robustez debido a las funciones que realiza, teniendo por lo general los módulos de visualización tipo carácter y gráfico.

Por esta razón se escogió una pantalla LCD gráfica y no una tipo carácter, ya que en esta, solo se podría visualizar los resultados numéricos de las medidas.

La resolución de estas pantallas es otro parámetro que las diferencia, por lo general existen pantallas de tipo carácter de 1 línea por 8 caracteres y de tipo gráfico de 640 x 256 píxeles aproximadamente según el controlador que utilice.

Últimamente con el desarrollo de equipos portátiles, la producción de pantallas LCD ha evolucionado y se han añadido módulos como el color y diferentes tipos de "control" como el *Touch Screen*. Estos parámetros pueden ser también un criterio para la elección del tipo de pantalla que se requiera.

El tipo de pantalla escogido para el desarrollo del proyecto fue la PG320240FRF-DE4-H-A1-SA producida por *Powertip Technology Corporation* cuyas características se amoldaban a las necesidades requeridas [LCD Gráfico].

3.3 DESCRIPCIÓN GENERAL DE LA PANTALLA SELECCIONADA

La resolución proporcionada en la PG320240FRF-DE4-H-A1-SA fue de una matriz de 320x240 píxeles, de tipo FSTN (*Film Supertwisted nematic*) lo que indica que tienen compensación de la película para rotaciones indeseadas; colores blanco y negro, vista en ángulo de 6 en punto, 8 "bits" de datos de entrada en paralelo, controlados por el SED1335 (circuito integrado mostrado dentro del circulo verde en la figura 25); construido con circuito generador de tensión negativa, luz de fondo de CCFL (*Cold Cathode Fluorescent Lamp*), módulo de *Touch Screen* y compensación de temperatura. Las especificaciones mecánicas se muestran en la tabla 2.

Tabla 2. Descripción mecánica de la pantalla.

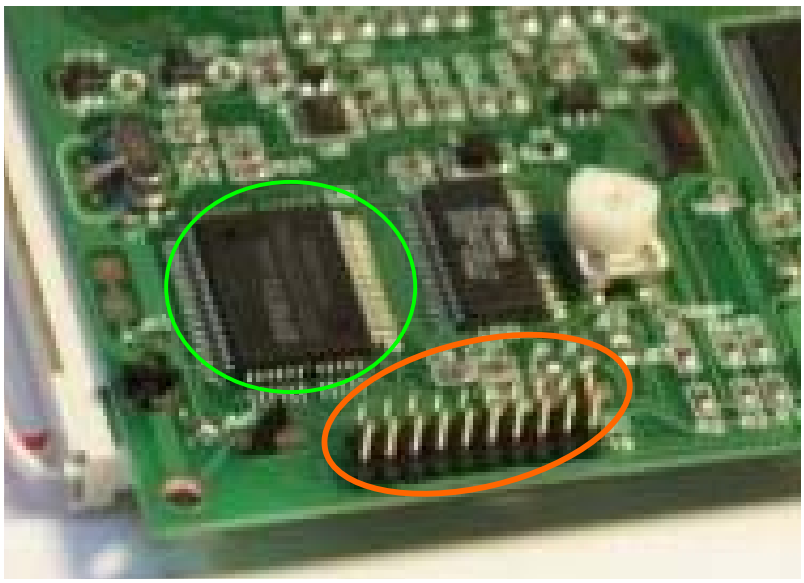
Item	***Valor***	***Unidades***
Dimensiones (L x W x H)	148,02 x 120,24 x 20,3	mm
Área de vista (L x W)	120,14 x 92,14	mm
Área activa (L x W)	115,17 x 92,14	mm
Tamaño de píxel (L x W)	0,33 x 0,33	mm
Profundidad del píxel	0,36 x 0,36	mm

Los pines para la debida programación y funcionamiento se describen en la tabla 3 y se muestran en la figura 25 en un óvalo rojo.

Tabla 3. Descripción de las terminales de la pantalla.

Pin #	***Símbolo***	***Función***
1	V_{SS}	Señal de tierra (V_{SS} =0)
2	V_{DD}	Fuente de alimentación (V_{DD}> V_{SS})
3	V_{LCD}	Fuente de tensión (controlador LCD). No se conecta
4	~RD	Leer datos (escribe datos a otro módulo colocando un nivel bajo "L")
5	~WR	Escribir datos (lee datos desde el módulo colocando un nivel bajo "L")
6	A0	Señal de control de direcciones de la pantalla[4]
7-14	DB0-DB7	Bus de datos (DB0=MSB, DB7=LSB)[5]
15	~CS	Selector del chip SED1335[6]
16	~RES	Reset del chip SED1335
17	V_{EE}	Fuente de tensión negativa. No se conecta
18	FG	Tierra mecánica
19	NC	No se conecta
20	NC	No se conecta

Figura 25. Controlador SED1335 y pines de control y datos.



Fuente: Los autores.

[4] Este terminal es controlado por el GPIOE0 del DSP56F807
[5] Estos pines son controlados por el GPIOB del DSP56F807
[6] Es controlado por el GPIOE1 en el DSP56F807

La figura 26 muestra la foto de la pantalla que se utilizó para desarrollar la aplicación.

Figura 26. Pantalla LCD gráfica PG320240FRF-DE4-H-A1-SA

Fuente: Los autores.

3.4 CONEXIONES FINALES Y PROGRAMACIÓN DE LA PANTALLA

Para la aplicación que requiere este proyecto no es necesario controlar con el DSP algunos terminales de la pantalla, es decir, como en la pantalla no se leían posiciones de memoria desde el DSP, algunas conexiones se dejaron con niveles altos o bajos, según como se necesitase, como es el caso de los pines ~RD (en alto) y ~WR (en bajo), de la misma forma que el ~RES (nivel alto).

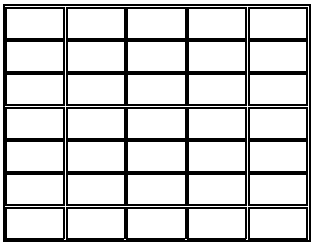
La pantalla se programa a través del controlador SED1335 [Controlador LCD], el cual está diseñado específicamente para el control de pantallas LCD. Con un código adecuado se pueden mostrar variables de texto y gráficas.

En general esta pantalla posee tres capas de memoria las cuales se pueden programar de la siguiente forma: La capa 1 se puede programar como texto o como gráfica, las capas 2 y 3 como gráfica únicamente.

El direccionamiento de la memoria de la pantalla gráfica está dada por parámetros de configuración que emplean una pantalla real (visible) y varias posiciones de memoria más para almacenar valores en pantalla que luego puedan ser vistos realizando desplazamientos de la misma.

La memoria interna CG-ROM es la que contiene la lista de caracteres, la cual concuerda con los caracteres ASCII desde la posición 0 hasta la 127. Los caracteres implementados dentro de la pantalla desde la posición 128 en adelante corresponden a caracteres orientales y espacios negros. Los caracteres para esta implementación tienen una altura de 7 y un ancho de 5 píxeles (ver figura 27).

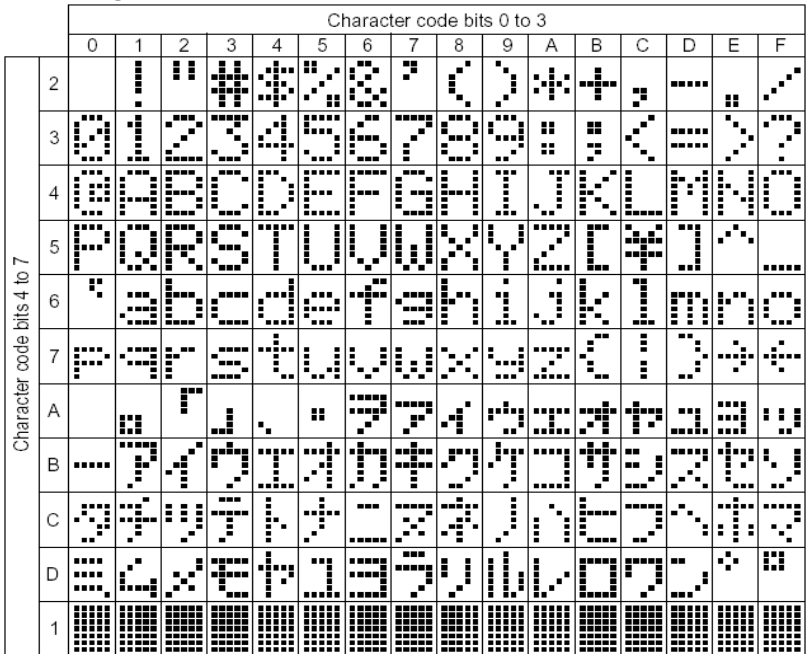
Figura 27. Matriz usada en la CGROM para implementar los caracteres



La razón por la cual las variables de texto tienen esta altura y este ancho, es para poder garantizar los espacios entre letras y entre líneas, es decir, que las letras no queden superpuestas para que sea legible el texto que se escriba.

La figura 28 muestra el diagrama del generador de caracteres para el controlador SED1335 [Controlador LCD].

Figura 28. Fuente interna generadora de caracteres.



Fuente: [Controlador LCD]

Para utilizar la pantalla gráfica, se debe dibujar por píxeles, pero se debe tener en cuenta que para colocar un píxel en negro se debe primero saber la posición en la cual se encuentra en la pantalla y luego si dibujarlo. Las direcciones de memoria donde se colocan datos para visualizar en la pantalla gráfica son manejadas por *bytes*. Dada esta aclaración es fácil entender que para hacer este trabajo se debe tener claro en que posición dentro del *byte* se debe fijar el nivel lógico alto.

Para la programación de la pantalla se implementaron funciones en lenguaje C a través del "software" de programación del DSP (*Codewarrior*). El listado de estas funciones se presenta a continuación.

InicioLCD(): Configurar la pantalla.
Direccioncur(): Movimiento automático del cursor dentro de la pantalla.
Limpiarram(): Limpiar la pantalla (blanqueo de pantalla).
diregcur(): Acceder a una posición de memoria.
comando(): Fijar en el puerto de datos y control el comando de la pantalla.
configp(): Fijar en el puerto de datos y control la configuración de comando().
formacur(): Fija la forma del cursor.

displayon(): Enciende la pantalla.
escribir(): Escribe en la pantalla una palabra.

La descripción detallada de estas funciones se pueden observar en el anexo C.

4. PRUEBAS Y ANÁLISIS DE RESULTADOS

En este capítulo se mencionan las pruebas realizadas al *"hardware"* y *"software"* implementado, y se realiza el análisis de los resultados.

4.1 PRUEBAS DEL FILTRO.

Como primera medida se hizo una prueba del filtro *antisolapamiento Butterworth* de cuarto orden simulado en el capítulo 2, los resultados se muestran en la tabla 4; y en la figura 29 se hace una comparación entre los valores obtenidos experimentalmente y los simulados (*"software" – Orcad 9.1*), donde se puede observar que el comportamiento es el esperado.

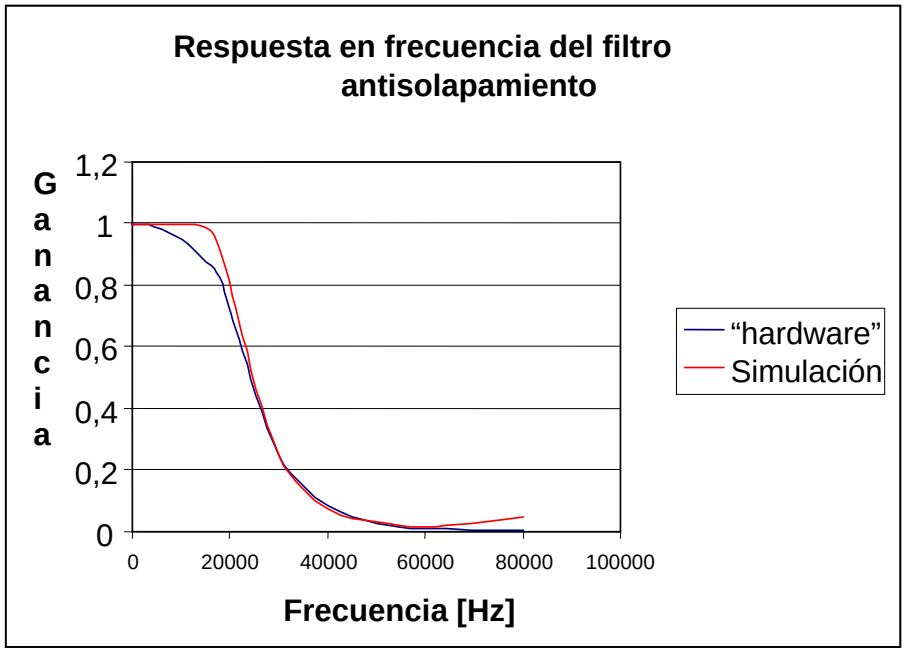
Tabla 4. Respuesta del filtro *antisolapamiento* de "hardware".

EXPERIMENTAL				***SIMULACION***	
Vi [V]	***Vo [V]***	***Frecuencia [Hz]***	***Atenuación en dB***	***Atenuación en dB***	***% Error de atenuación***
2	2	1	0	0	0
2	2	60	0	0	0
2	2	200	0	0	0
2	2	500	0	0	0
2	2	1000	0	0	0
2	2	2000	0	0	0
2	2	3300	0	0	0
2	1,975	5000	-0,1	0	1,25
2	1,9	10000	-0,44	0	5
2	1,75	15000	-1,15	-0,09	11,34
2	1,65	18000	-1,67	-0,75	9,93
2	1,45	20000	-2,79	-1,70	11,15
2	1,25	22000	-4,08	-3,33	8,22
2	0,5	30000	-12,04	-12,04	0
2	0,17	40000	-21,41	-21,18	0,82
2	0,05	50000	-32,04	-29,86	2,13
4	0,05	60000	-38,06	-38,19	1,46
4	0,02	80000	-46,02	-46,55	6,38

Nota: Debido a que la alimentación del sistema de filtrado estaba entre ±5V DC, la mayor amplitud de señal de entrada permitida debía ser 4 Vp (para no entrar en el rango de saturación de los op amp), en la frecuencia de 80 kHz se encuentra una atenuación de -46dB lo cual representa un valor

de tensión de aproximadamente 20 mV y por lo tanto, los aparatos de medición que se utilizaron captaban una señal defectuosa que era la combinación de la señal atenuada y el ruido ambiental.

Figura 29. Respuesta en frecuencia del filtro analógico experimentado y simulado.



4.2 PRUEBAS DEL DSP

En esta sección se comprueba el funcionamiento de los diferentes módulos del DSP que se necesitarán en el desarrollo del proyecto.

4.2.1 Pruebas del conversor analógico-digital

Todo conversor analógico a digital presenta un error en el momento de cuantificar una muestra, en el caso del DSP, el fabricante proporciona un error de cuantificación de Vref/4096.

Para verificar que la frecuencia de muestreo era la especificada por el fabricante, se configuró el puerto ADC de tal forma que tuviese la máxima frecuencia de reloj. Luego se tomaron 1442 muestras de una señal cuadrada de la cual se conocía previamente su frecuencia y su amplitud, ya

que éstas se midieron con el generador de señales y un osciloscopio marca *Fluke*. Se realizaron varias pruebas que se visualizaron en *Codewarrior,* y algunos de los resultados fueron los siguientes:

- Onda cuadrada de 4kHz: Tuvo su primer cambio de signo en la muestra 20 y su tercer cambio de signo en la muestra 169, al hacer la resta se obtiene el número de muestras existentes en un periodo, y por lo tanto como se sabe la frecuencia de la señal, el cálculo de la frecuencia de muestreo está dada por la ecuación 22.

$$fs = (N+1) * fm \qquad (22)$$

Donde fs es la frecuencia de muestreo, N+1 el número de muestras y fm la frecuencia de la señal de entrada.

Aplicando la ecuación 22 se obtiene que la frecuencia de muestreo del ADC es 600kHz.

- Onda cuadrada de 4,6kHz: Con esta onda se procedió igual que la anterior, donde el número de muestras contenidas en un periodo fue de 128. Reemplazando nuevamente en la ecuación 22, fs=588kHz.

- Onda cuadrada de 5kHz: Se realizó el mismo procedimiento, N+1=122, por lo tanto fs=610kHz.

Los resultados tabulados se pueden observar en la tabla 5.

Tabla 5. Pruebas de la frecuencia de muestro ADC del DSP.

Frecuencia de la señal (kHz)	***Número de muestras (cambio de signo)***	***Frecuencia de muestreo (kHz)***
4	150	600
4,6	128	588,8
5	122	610

Para comprobar el error de cuantificación del ADC del DSP se tomaron muchas pruebas de las cuales se obtuvieron los resultados mostrados en la tabla 6. El error en porcentaje se calculó

utilizando el error promedio, el cual arrojó un resultado del 9,84 %. Este resultado se obtuvo de la siguiente manera: $\left(\frac{0,325}{3,3}\right)*100\%$

Tabla 6. Error de cuantificación con señal de tierra y 1,5 V.

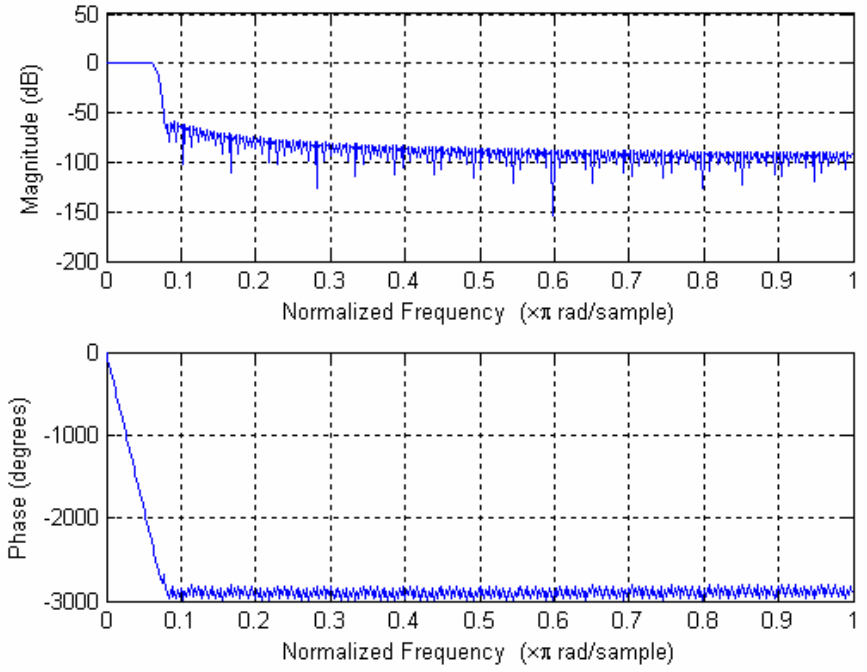
Señal de entrada [V]	***Límite inferior [Δ]***	***Límite superior [Δ]***	***Error [Δ]***	***Error [V]***
0	5	427	422	0,339
	10	388	378	0,304
	0	408	408	0,328
	11	388	377	0,303
	4	400	396	0,319
1,5	1659	2079	420	0,338
	1649	2072	423	0,340
	1677	2069	392	0,315
	1638	2068	430	0,346
	1680	2071	391	0,315
Promedio			403,7	0,325
Desviación estándar			19,476	0,015

4.2.2 Pruebas de los algoritmos.

En esta sección se realizaron pruebas a los códigos implementados en el DSP (*Codewarrior*) contrastándose con funciones ya implementadas en el "software" *Matlab 6.5*. Las funciones probadas en esta sección son las correspondientes al filtrado digital de la señal, enventanado y FFT.

1. El filtrado digital se realizó como se mencionó en la sección 2.1.3. El resultado arrojado se visualizó con la función *freqz* (respuesta en frecuencia de filtro digital) de *Matlab* en la figura 30. En esta gráfica, la frecuencia está normalizada de 0 a π (0 a 300 kHz) y se observa que la respuesta en magnitud cumple con las características requeridas, además se observa la característica de fase lineal que brindan los filtros FIR.

Figura 30. Respuesta en frecuencia del filtro digital pasabajos.



2. Para el enventanado también se probó la respuesta en frecuencia. La figura 31 ilustra la respuesta en frecuencia del enventanado rectangular y la figura 32 el enventanado de *Hanning* (implementados en el DSP).

Figura 31. Respuesta en frecuencia del enventanado rectangular.

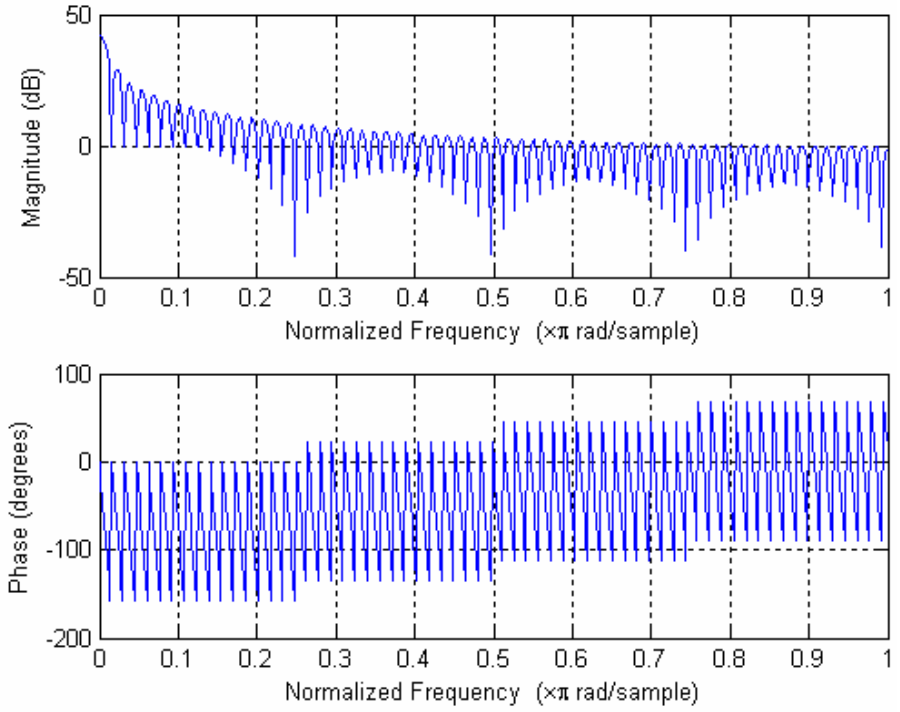
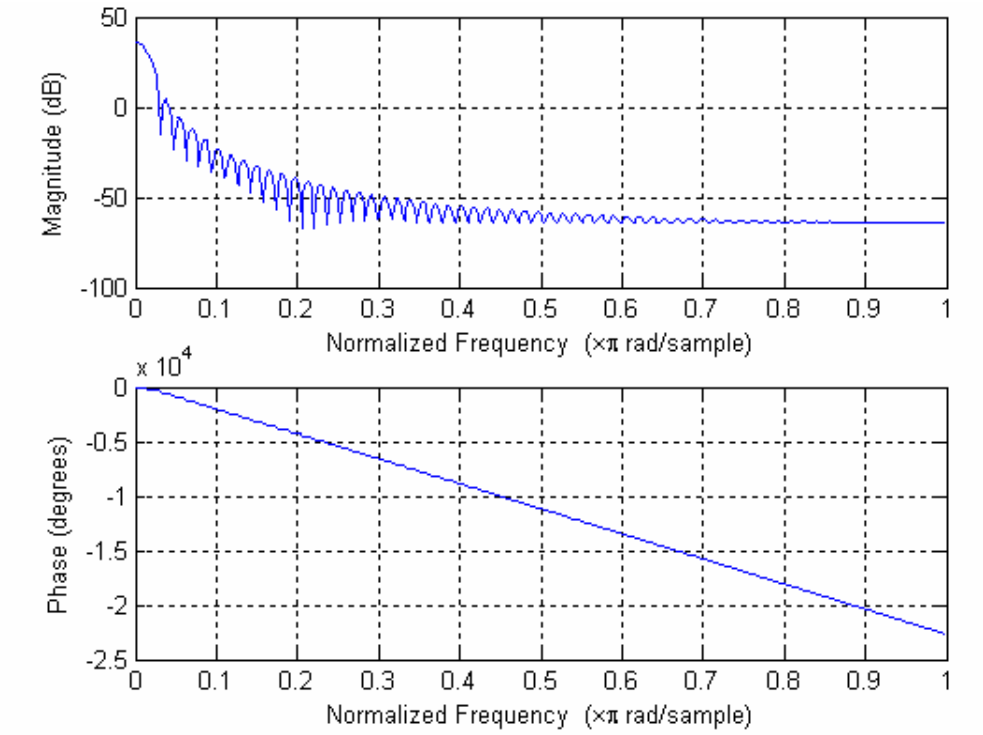


Figura 32. Respuesta en frecuencia del enventanado de Hanning.



En el equipo se utilizó la ventana de *Hanning*, ya que en los resultados arrojados por la respuesta en frecuencia es mucho mejor que la ventana rectangular, teniendo en cuenta que se visualizaba mejor el resultado con la ventana *Hanning,* ya que presenta mayor atenuación de pequeñas componentes no deseadas en todo el ancho de banda que aparecen en la ventana rectangular por no tener un periodo entero de la señal al calcular la FFT; otra características por la cual se escogió esta ventana es que esta representa menor carga computacional en el DSP comparándola con la *Blackman*.

3. Para la prueba de la FFT se simularon varias señales, de las cuales se tomaron 128 muestras en un periodo. De éstas se obtuvo que su magnitud presentó las siguientes características.

- Onda senoidal [100*sen(2*pi/128)]:
  - *Matlab*: En la posición 2 y en la 128 del vector aparecía una magnitud de 64.
  - *Codewarrior*: En la posición 2 y en la 128 del vector aparecía una magnitud de 63,619.

  El error de esta prueba fue de 0,595%

- Onda DC [10].
  - Matlab: En la posición 1 del vector aparecía una magnitud de 16384.
  - *Codewarrior*: En la posición 1 del vector aparecía una magnitud de 16384.

  El error en DC fue de 0%.

- Onda cuadrada [Amplitud 10, T=64muestras].
  - *Matlab*: En la posición 2 y en la 128 del vector aparecía una magnitud de 814,955
  - *Codewarrior*: En la posición 2 y en la 128 del vector aparecía una magnitud de 810,246

  El error de esta prueba fue de 0,5%.

  - *Matlab*: En la posición 4 y en la 126 del vector aparecía una magnitud de 271,87
  - *Codewarrior*: En la posición 4 y en la 126 del vector aparecía una magnitud de 271,73.

  El error de esta prueba fue de 0,04%

  - *Matlab*: En la posición 64 del vector aparecía una magnitud de 20,006.
  - *Codewarrior*: En la posición 4 y en la 126 del vector aparecía una magnitud de 20.

El error de esta prueba fue de 0,03%

4.3 CARACTERIZACIÓN DEL EQUIPO

Características principales del analizador de espectros portátil:

- Alimentación con una fuente de 9 V DC y 500mA (adaptador).
- Bajo consumo de potencia.
- Ancho de banda de 20 KHz.
- Spam de 2343,75 Hz.
- Resolución en frecuencia de 585,9375 Hz.
- 2 baterías recargables de 5 V, se recomienda UBC425085/PCM de la empresa *Ultralife Baterries Inc*. Las cuales tienen una capacidad de 1600mAh, lo cual representa una autonomía para el equipo de 5 horas aproximadamente.
- Tamaño 17 x 14 x 6 cms.
- Corriente de consumo del sistema 238,5 mA.
- Impedancia de entrada de la señal 56kΩ.
- Posibilidad de expansión de Memoria y conexiones al PC por puerto USB o serial.
- Dos canales de ADC con resolución de 12 "bits" de adquisición de la señal.
- Procesado de la señal con 16 "bits".
- Despliegue de la FFT en una pantalla LCD de 320 x 240 píxeles.
- Interfaz grafica de fácil entendimiento para el usuario.
- Puerto JTAG para permitir programar al equipo para realizar mejoras al *"software"*. O si desea implementar otra aplicación.
- Costo total del equipo $1'200.000,oo.

Nota: Con respecto a las baterías recargables, se escogió la UBC425085/PCM de la empresa *Ultralife Baterries Inc. la cual fue la que más se ajustaba al diseño por sus dimensiones. Para esta etapa se realizaron los diseños para una posible alimentación de este tipo, pero no se hicieron pruebas debido a los pocos recursos económicos para la compra de las mismas.*

Alguna de las mejoras que se le pueden realizar a este equipo son:

- Expansión de memoria a 64 K x 16 de RAM, previendo que este dispositivo solo tiene 4K x 16 de RAM. Esta memoria se podría manejar por los puertos disponibles en la tarjeta del equipo, los cuales son: direcciones, datos y control.
- Adquisición con ADC externo que muestree a una frecuencia de por lo menos 5 MHz, ya que en éste solo se pueden adquirir señales con una frecuencia de muestreo de 600 kHz aproximadamente; además que presente mejor estabilidad al realizar la cuantificación.
- Comunicación mediante el puerto USB con el objetivo de transferir datos rápidamente (1 Mbaudio) al PC que serán almacenados, para un posible historial, y/o realizar un análisis posterior (estadístico, en frecuencia, en tiempo, etc.). Este análisis se puede elaborar mediante herramientas de "software" como *Lab-view, Visual C++, Matlab,* etc.
- Manejo del módulo de *Touch Screen* de la pantalla de cristal líquido con el objetivo de controlar eventos tales como los menús. Esta mejora se puede implementar por medio del canal disponible del conversor analógico a digital.
- Carcasa metálica para introducir el equipo. La finalidad de esta carcasa es capturar las señales de ruido presentes en el ambiente y conectarla con la referencia a tierra del sistema.

Algunas de estas mejoras son posibles realizarlas al software ya implementado en este equipo si se adquiere una licencia que tenga más capacidad a la entregada por la empresa *Metrowerks* (gratuitamente) en el programa de Codewarrior de 16 Kbytes.

5. OBSERVACIONES, CONCLUSIONES Y RECOMENDACIONES

5.1 OBSERVACIONES

Algo que es importante resaltar del equipo analizador de espectro portátil, es que este tendrá el puerto JTAG habilitado en la tarjeta, para ser programado de nuevo si así se desea, también incluye los puertos para expansión de memoria externa(*bytes* de dirección y datos) y dos canales de ADC.

El soporte presentado por el fabricante del DSP (*Motorola*) fue una de las herramientas más importantes, ya que proporcionan sin ningún valor los circuitos esquemáticos de sistemas específicos con la opción de obtener al mismo tiempo una tarjeta de desarrollo para la prueba de múltiples aplicaciones.

En la familia DSP56800 de *Motorola* se pudo observar que a medida que se avanzaba en cada uno de los dispositivos (801-807), los módulos como el ADC y Contador/temporizador presentaban más estabilidad, teniendo en cuenta también que el tamaño y el número de pines por lo general aumentaban.

Los dispositivos de visualización como la pantalla LCD gráfica, permitió mostrar medidas de tipo gráfico, además, comentarios de tipo texto que permitieron darle al prototipo un entorno más dinámico.

Codewarrior como "software" de programación, presenta la ventaja de utilizar un lenguaje de alto nivel como C, el cual facilita la programación al usuario.

En la configuración en *Codewarrior* de cada uno de los módulos realizadas por el programador, se pudo observar que para los puertos de propósito general, éstos funcionaron con mayor velocidad comparándolos con los configurados por los "*Beans"*; sin embargo, al insertar un *Bean* en el proyecto, el *Processor Expert* genera códigos que se implementan como librerías .h y archivos .C, opciones que reducen el tamaño del programa en el DSP.

El "software" de implementación ("*Eagle*") para circuitos impresos es una herramienta muy útil y de fácil manejo, ya que tiene herramienta para diseño de integrados (físicos) y permite realizar fácilmente planos de tierra.

La resolución es una propiedad importante en la aplicación del analizador de espectros, y se pudo resaltar que la pantalla tiene una buena resolución para mostrar los datos.

5.2 CONCLUSIONES

Por medio de dispositivos como los DSP se pueden elaborar aplicaciones en donde se necesiten una considerable carga computacional para realizar algoritmos de procesado de señal sin la necesidad de una máquina tan potente como un computador, permitiendo así a partir de una debida programación, construir equipos con visualización de eventos en pantalla LCD gráfica.

Las variables globales que crea el programador dentro de los archivos .C generados por el *Processor Expert*, son solo visibles por tal archivo, por ejemplo, si se crea una variable global en el archivo *events.c* (interrupciones), esta variable no será reconocida por el archivo *main.c*. Como alternativa de solución a este problema se pueden implementar códigos en donde se utilice la memoria *Flash* para descargar algunos datos de un archivo y luego recuperarlos sin inconvenientes desde otro.

Las aplicaciones presentadas por *Codewarrior* a través de la herramienta *Processor Expert* y en específico los "*Beans*", sirven de mucha ayuda para aprender a utilizar el DSP como tal, es decir, con los debidos pasos para realizar una correcta configuración de un módulo y las funciones que emplea, además el entorno gráfico para la configuración de cada *Bean*, hacen ver fácil la programación del DSP.

En este equipo se presentaron limitaciones de "hardware" tales como velocidad de conversión del ADC del DSP (600 kHz aproximadamente), alimentación de entrada al mismo (3,3 V); las cuales arrojaron características para el analizador de espectros como: la señal de entrada debe tener una frecuencia no mayor a 20kHz y su tensión no debe exceder de ±1 V, para garantizar su correcto

funcionamiento. Las limitaciones de "software" se dieron principalmente por la licencia entregada por Codewarrior para programación del dispositivo (16 KB).

Los dispositivos programables como los DSP y microcontroladores tienen una amplia ventaja con respecto a los dispositivos no programables, debido a que permiten el cambio o la mejora de los códigos que se implementan en ellos sin necesidad de adicionar más "hardware". Esto se puede observar fácilmente en la elaboración de "software" para la pantalla LCD gráfica, en donde éste puede cambiarse según las necesidades que el usuario desee implementar.

Al trabajar con dispositivos de montaje superficial el diseño del circuito impreso se dificulta, ya que al disminuir el tamaño de los elementos, los caminos de cobre buscan otras capas para poder unirse sin inconvenientes. Al solo tener en Bucaramanga tecnología para enrutamiento en dos capas, en tarjetas de desarrollo de dispositivos como el DSP56F807 es muy difícil enrutar si se utilizan solo elementos de montaje superficial y se quiere acceso a todos los puertos y/o pines que posee el integrado.

El módulo del ADC presenta principalmente dos problemas:

-Oscilación de la frecuencia de muestreo (600 kHz ± 12 kHz). Esto se debe a que se tuvo que preescalar el PLL del DSP, para poder almacenar los datos sin afectar la frecuencia de muestreo. La frecuencia de operación del DSP al ser preescalado quedó en 120 MHz, el cual es límite entregado por el fabricante.
-Inestabilidad al momento de la cuantificación. Este defecto es propio del DSP.

5.3 RECOMENDACIONES

El conversor analógico a digital del 56F80X es crítico cuando se necesita que los datos analógicos obtenidos sean exactos (esta observación la hace el fabricante y se comprobó en el capítulo 4), si se necesita que la aplicación no falle por este motivo, este procesador no es el indicado a utilizar, sería conveniente entonces utilizar otro DSP con mejores características, pero si se desea mayor precisión y exactitud en el momento de capturar la señal analógica se recomienda utilizar un conversor analógico a digital externo.

Para la portabilidad del equipo sería importante utilizar baterías recargables e implementar todos los circuitos integrados con la misma tensión de entrada (no bipolar), evitando el alto consumo de potencia producido por los integrados que se alimentan con tensión dual.

Para la etapa de soldadura de los diferentes elementos, se deben tener los siguientes cuidados para que la tarjeta no sufra daños y los elementos funcionen correctamente. Para empezar se debe utilizar un cautín con una punta apropiada y limpia (debe observarse brillante), limpiar constantemente el cautín con un elemento no cortante y crema para soldar. El cautín no se debe dejar mucho tiempo sobre las pistas, esto puede producir calentamiento de las mismas y un posterior levantamiento o corte del camino. No se recomienda tener expuesto por mucho tiempo el cautín a los pines de los integrados, ya que el calor puede causarle daños.

Para obtener una mejor resolución en frecuencia se podría trabajar con una memoria RAM externa, con el objetivo de tomar un número mayor de muestras y no estar restringido por la memoria RAM del DSP, que es uno de los cuellos de botella más fuerte en los sistemas en donde se requiere procesado digital; y además trabajar con un ADC que posea una velocidad de conversión mayor que la del DSP (la memoria y la velocidad depende de la precisión que se requiera) y menor error de cuantificación.

BIBLIOGRAFÍA

[Amaris & Lopez, 2004] AMARIS Jean Pierre y LOPEZ José Alberto. Elaboración del "software" para la caracterización de una celda electroquímica utilizando DSP familia 56800 de Motorota. Bucaramanga, 2004. Trabajo de grado (Ingeniero Electrónico). Universidad Industrial de Santander. Facultad Físico Mecánica. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones.

[Flórez y Herrera, 2004] FLOREZ Jairo Iván y HERRERA Shirley Paola. Diseño y construcción de un prototipo de un medidor de armónicos de corriente basado en un procesador de señales digitales (*DSP*). Bucaramanga, 2004. Trabajo de grado (Ingeniero Electrónico). Universidad Industrial de Santander. Facultad Físico Mecánica. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones.

[DSP56800 "hardware" Interface Techniques]. DSP56f80x User Manuals Motorola *digital dna, MCU-DSP 56800 Accelerated Development System. Technical Documentation, 56800 Documenation (PDF), Technical Articles*, Application Notes, AN1920/D, 06, 2001.

[Motorola, DSP56F800 Family Manual, 2003] *Motorola digital dna, DSP56F800 16-Bit Digital Signal Processor (PDF)*, *DSP56F800FM/D, 03,* 2003.

[Motorola, DSP56F800 User Manual, 2004] *Freescale semiconductor, DSP56F800 User Manual, 56800 Documentation(PDF)*, *DSP56F801-7UM,* 06, 2004.

[Motorola, *DSP56F801*] Motorola digital dna*, 56F801 16 bit Hybrid Processor. Technical Data, Data Sheets DSP56F801, DSP56F801, 7, 2002.*

[Motorola, DSP56F801EVMUM, 2003] F*reescale Semiconductor, 56801 Evaluation Module "hardware" User's Manual. Technical Documentation,* DSP56F801EVMUM (PDF) Application Notes, DSP56801EVMUM, 02, 2003.

[Motorola, *DSP56F807*] F*reescale Semiconductor, 56F807 16 bit Hybrid Processor. Technical Data, Data Sheets DSP56F807, DSP56F807, 12, 2004*

[Motorola, DSP56F807EVMUM, 2003] F*reescale Semiconductor, 56F807 Evaluation Module "hardware" User's Manual. Technical Documentation,* DSP56F807EVMUM (PDF) Application Notes, DSP56807EVMUM, 02, 2003.

[Posadas, 1998] POSADAS Juan Luis. Transformada Rápida de Fourier (FFT) e Interpolación en Tiempo Real. Valencia, 1998. Universidad Politécnica de Valencia. Departamento de informática de sistemas y computadores.

[Proakis y Manolakis, 1998] Proakis J, Manolakis D. Tratamiento Digital de Señales. Prentice Hall. España 1998.

[Jaquenod LCD] JAQUENOD Guillermo A. Tecnologías de los Displays LCD. Análisis de los productos de DISPLAYTECH.

[Creus, 1998] CREUS Solé antonio. Instrumentación industrial. Marcombo, Barcelona. 1998. 6ª edición. P.192-196.

[Microchip, *LCD*] Microchip Technology Inc, LCD Fundamentals Using PIC16C92X Microcontrollers (PDF). *Technical Documentation, AN658.1997*

[Fink y Christiansen, 1992] FINK Donald G. y Christiansen Donald. Manual de Ingeniería Electrónica. Tomo II. Mc Graw Hill. Madrid (España). 1992.

[Freescale] http://www.freescale.com/. Fabricante DSPs.
[Metrowerks] http://www.metrowerks.com/. Proveedor de "software".
[Controlador LCD] http://www.elateceurope.com/displays/support/progr_sed1335.shtml. Controlador LCD gráfico.
[LCD Gráfico] http://www.powertipusa.com/. Fabricantes pantalla gráfica.

ANEXOS

ANEXO A. PROGRAMA PARA EL DISEÑO DE CIRCUITOS IMPRESOS

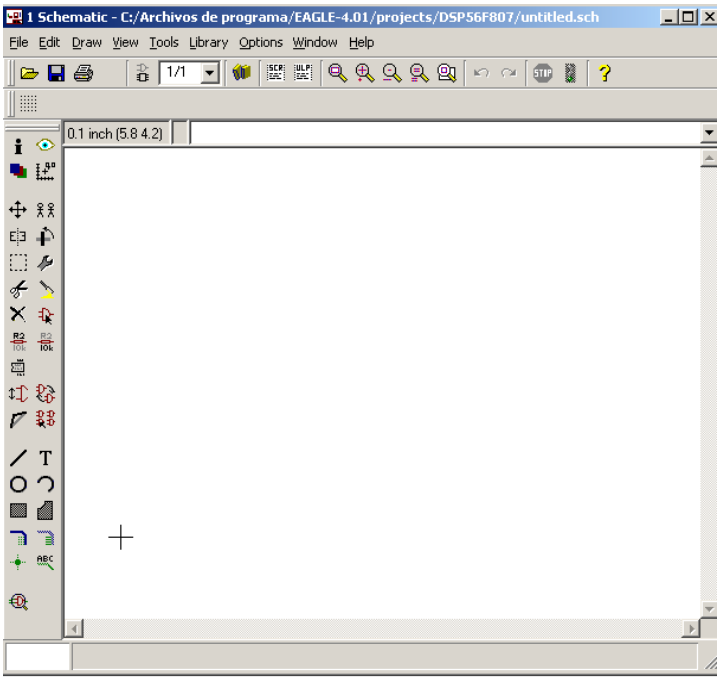
Para la implementación de los diseños de "hardware" se utilizó una versión de prueba del "software" *Eagle* 4.01, la cual permitió cumplir correctamente con los objetivos trazados en cuanto al diseño de las tarjetas de desarrollo.

Esta versión, permitió a partir de un circuito esquemático obtener un diagrama del impreso de las aplicaciones que se necesitaban. Para los dispositivos que no se encontraban en la base de datos se debieron crear librerías, lo que incluye, los símbolos esquemáticos, los empaquetados y las conexiones de estos dos últimos, quedando el dispositivo listo para utilizarse en el "software".

En esta sección se hará una breve introducción a este "software", explicando los comandos usados para la elaboración de los impresos.

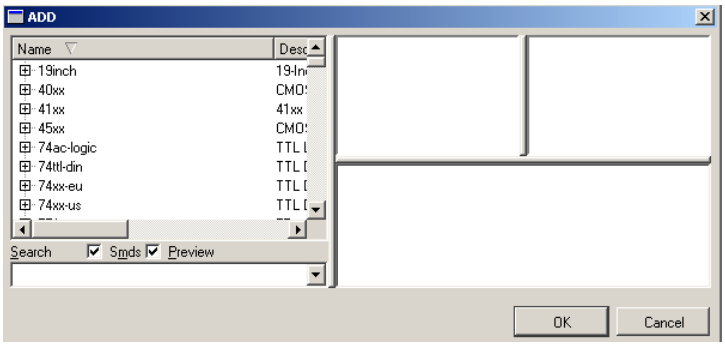
Para empezar, se supondrá que la base de datos del programa tiene todos los dispositivos que se van a utilizar. Por lo tanto, se inicia ejecutando el comando S*chematic*, del submenú *New*, del menú *File*. Este comando abre la ventana que se muestra en la figura A1, y es allí donde se crea el circuito esquemático de la aplicación definida. En el entorno de *Eagle* se muestra una barra de herramientas estándar (parte superior de la figura A1), debajo de ésta, una barra gris que activa opciones en algunos comandos y un cuadro para escribir texto, éste último puede utilizarse para llamar los comandos que se observan en la barra de edición (parte izquierda de la figura A1).

Figura A1. Ventana donde se implementa el circuito esquemático.



Para iniciar cualquier diseño se deben agregar los dispositivos a usar, esto se elige con el comando *add*, el cual se puede escribir en la barra de texto (al igual que todos los comandos), del mismo modo se encuentra en el menú edición, o haciendo clic con el ratón en el icono  , después de ejecutar esta acción, aparece la ventana que tiene la base de datos de los dispositivos (ver figura A2).

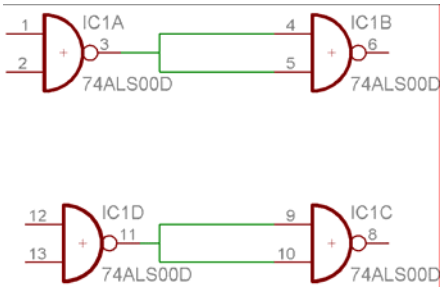
Figura A2. Base de datos del dispositivo (comando ADD).



Cuando se escoge un elemento, el puntero del *mouse* se convierte en una cruz con un esbozo del elemento a anexar a la ventana. Cuando se ha anexado éste, el "software" espera que se sigan anexando elementos del mismo tipo y solo hasta cuando se hace clic en el icono *cancel*  (o se pulsa dos veces la tecla *Esc*) se dejan de anexar dispositivos (el comando *cancel* detiene la acción de cualquier comando). Si lo que se quiere es anexar todos los símbolos esquemáticos de los dispositivos del diseño del circuito, no se elige la opción cancel, si no que se pulsa una sola vez la tecla *Esc* del teclado, retornando a la ventana de la base de datos de los dispositivos.

Para corregir la posición de los elementos se utiliza el comando *move*, su icono es  y su función es desplazar los objetos sobre la ventana. Con el clic derecho del ratón se pueden realizar rotaciones a 90º en sentido contrario a las manecillas del reloj, si previamente se ha escogido un elemento. Teniendo los elementos en los lugares determinados se prosigue con la conexión de los mismos. El trazo de estas líneas se realiza con el comando *net*, su icono es  y cuando éste se elige, el ratón adapta la posición de cruz. Este comando funciona así: al hacer el primer clic, el usuario se sitúa en el lugar donde va a empezar la línea, luego, se arrastra el puntero hasta donde se necesite y se hace de nuevo clic (muchas veces se utiliza para hacer ángulos de 90º y seguir desplazándose), para terminar una conexión se hace doble clic, o un solo clic si se sitúa el ratón en la terminal de un elemento (ver figura A4).

Figura A4. Utilización del comando net.

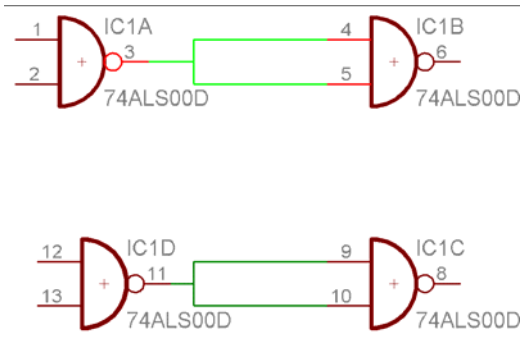


Si se está completamente seguro de que los terminales de los elementos quedaron unidos por medio de la línea, el circuito se puede dejar como está y pasarlo a la *board*, pero en muchos casos, no se sabe si en realidad las líneas están bien conectadas. Para tener certeza, se pueden realizar una de

dos tareas: verificar que cada una de las líneas se conectaron de forma correcta (*show*), o colocar puntos (*junction*) en cada terminal donde haya una unión.

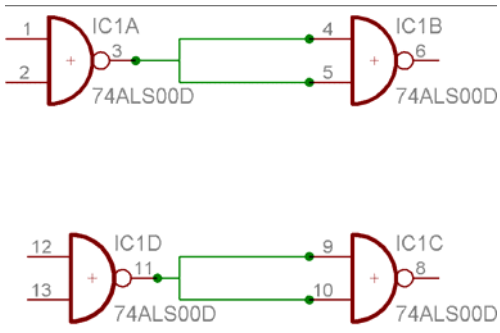
La verificación de estas terminales se hace por medio del comando show  , el cual "ilumina" el objeto o la línea en donde se haga clic. La figura A5 muestra las líneas de arriba más brillantes, lo cual indica que se hizo clic sobre una de ellas y los elementos están unidos.

Figura A5. Aplicación del comando *show*.



La opción de colocar puntos en cada terminal que se vaya a unir es guiada por el comando *junction*  , el cual acopla líneas con terminales o entre si. Esta opción es más utilizada, ya que cuando la cuadrícula (*grid*) del "software" se hace muy pequeña se dificulta la unión de las líneas (ver figura A6).

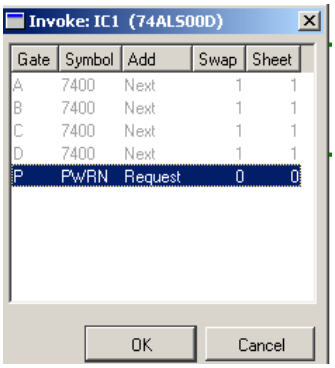
Figura A6. Puntos de unión.



Teniendo las terminales unidas, se procede con la alimentación de los elementos.

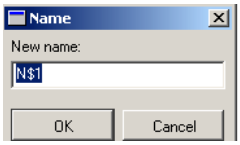
La alimentación de los dispositivos es un punto importante en el diseño de los circuitos impresos, y por eso, debido a que en algunos circuitos integrados solo están los pines relacionados con las funciones que estos realizan y no los terminales de alimentación, por lo tanto se utiliza el comando *invoke*  para poder llamar los pines de alimentación del circuito integrado. En la figura A7 se muestra un ejemplo con el integrado 74ALS00D y al hacer clic en OK, los pines de alimentación aparecen para colocar en el diagrama esquemático.

Figura A7. Aplicación del comando *invoke*.



Realizados estos pasos, se le puede dar nombre a cada uno de los elementos, para esta función se utiliza *name*  que en la figura B8 se muestra como ejemplo en una línea.

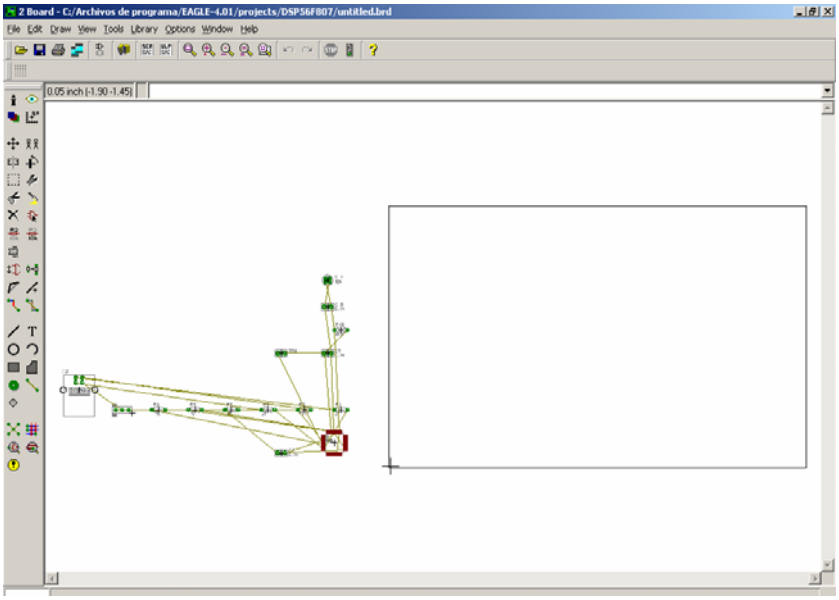
Figura A8. Ejemplo de la utilización del comando *name*.



Finalmente si en el diagrama esquemático se necesita borrar algún componente, se usa el comando delete  , el cual después de ejecutarse elimina los elementos que se seleccionen.

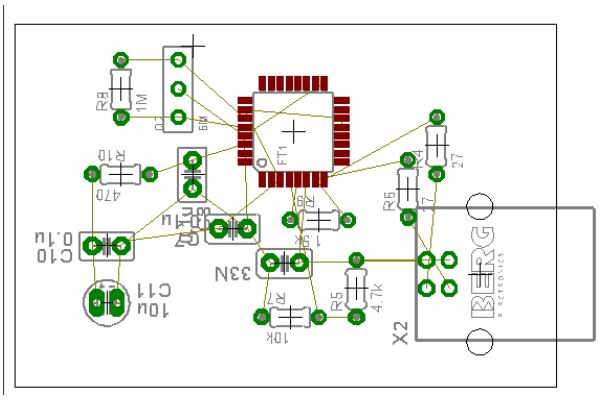
Cuando se termina con el diseño esquemático se procede a diseñar la tarjeta. Para llevar los elementos de un diagrama a otro se utiliza el comando *board*  . Con este, se realiza el diseño del circuito impreso en si, ya que es en este entorno donde se realiza el enrutamiento de la tarjeta y se posicionan los dispositivos (ver figura A9).

Figura A9. Ejemplo de posición de los dispositivos en la board (primera vez).



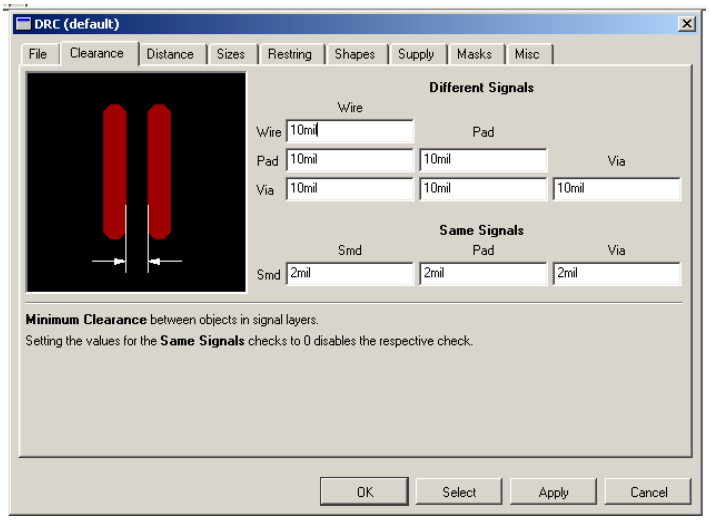
En la figura A9 se observa que los implementos en la tarjeta no tienen un orden y, que estos quedan por fuera del área determinada para realizar el enrutado. Lo primero que se debe hacer es organizar los elementos, esto se hace igual que en panel del diagrama esquemático con el comando *move* (ver figura A10).

Figura A10. Vista de los elementos y el área de enrutado debidamente organizados.



Teniendo los elementos en posición para enrutar se ajustan los parámetros para que el "software" se encargue del enrutado. Esto se realiza con el comando *DRC*  del menú *Tools* (figura A11).

Figura A11. Comando DRC, parámetros de los caminos y las vías.

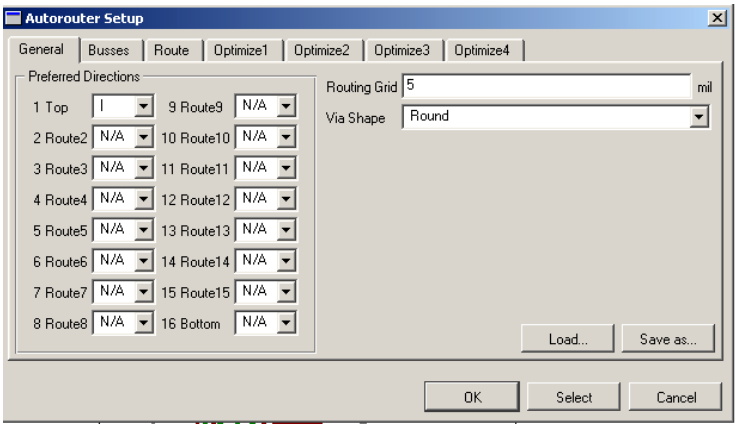


En la figura A11 se puede observar la pestaña *Clearance*, del comando *DRC*, en donde se ajustan los tamaños de separación entre caminos (*wire*), *pads* y *vias*. Un camino es una línea de cobre que comunica dos elementos; los *pads* son las áreas de cobre donde van soldados los pines de los integrados (características típicas de los dispositivos DIP), y las *vias* son el puente de comunicación cuando se utiliza más de una cara para el enrutamiento del circuito impreso.

En las diferentes pestañas del comando DRC, se encuentran las opciones de configuración que por lo general, son de los parámetros explicados anteriormente (*wire*, *pads* y *vias*). En este se pueden especificar los tamaños (*size*), las distancias (*distance*) y otras características en diseños más específicos.

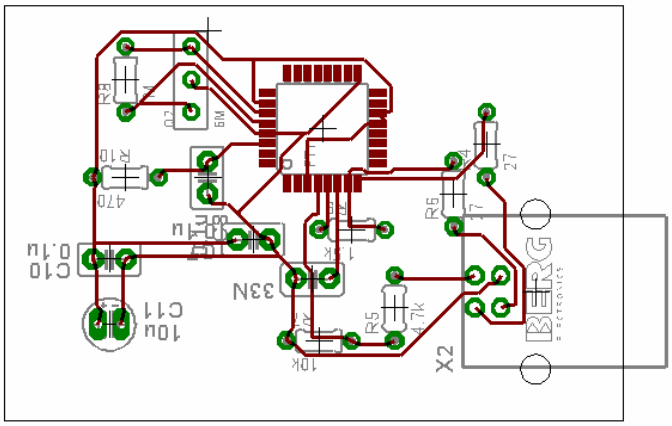
Aprovechando la opción de autoenrutado del "software", se procede entonces a ejecutar el comando *auto*, el cual se encuentra en el menú *Tools*, o como icono en la barra de edición  (ver figura A12), en donde se escogen las capas donde se va a realizar el enrutamiento. Cuando se pulsa el botón OK, el "software" empieza a realizar el enrutamiento, que será rápido o lento dependiendo del procesador del computador.

Figura A12. Comando auto para escoger las caras del impreso y realizar enrutamiento por "software".



Si los elementos no se encuentran en una posición correcta, el enrutado automático puede no arrojar los resultados esperados (*autoroute* 100%). Por lo tanto, se recomienda antes de realizar el enrutado automático guardar el archivo, ya que si algún elemento no permite efectuar la operación correctamente, se puede cerrar el archivo sin guardar, luego cambiar la posición, guardar y volver a autoenrutar. La figura A13 muestra el diagrama final de un enrutado realizado con *Eagle*.

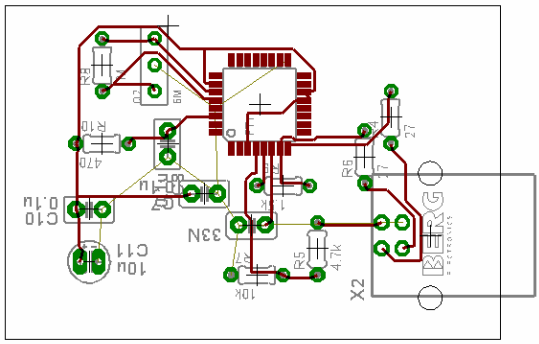
Figura A13. Enrutado realizado por "software".



Teniendo en cuenta, que muchas veces se necesitan trazar planos de tierra para eliminar ruido, es importante que éstos cubran la mayor área posible en el circuito impreso. A continuación se presentará el procedimiento para poder hacer los debidos planos.

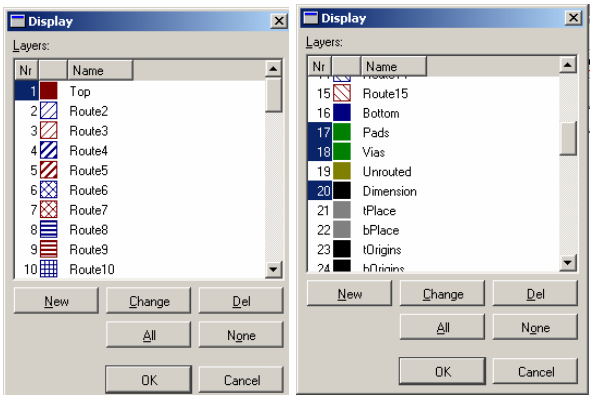
Primero se deben desenrutar los caminos correspondientes a la señal que se desea para el área de cobre. Esto se realiza mediante el comando *Ripup*  , el cual sirve para desenrutar cada una de las líneas con el *mouse*, o toda una señal (si se conoce el nombre) en la barra de texto. En el ejemplo se usará la señal de tierra (GND), véase en la figura A14 las líneas amarillas opacas.

Figura A14. Señal de tierra desenrutada (amarilla).



Luego, con el comando *Display*  , se esconden las vistas que no correspondan a la cara donde se va a colocar el área de tierra, quedando también los *pad*s, las *vias* y la dimensión (figura A15).

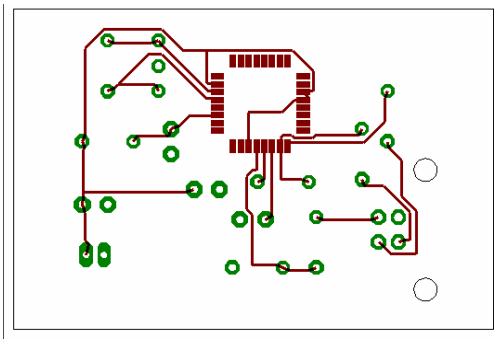
Figura A15. Comando *Display*, vistas para hacer planos de tierra.



Después de ejecutar este comando solo se observan en el impreso las capas seleccionadas por el usuario (figura A16). Luego, se procede a diseñar el área de cobre, ésta se hace por medio del comando *polygon*  . Cuando se utiliza este comando la barra de herramientas del comando se

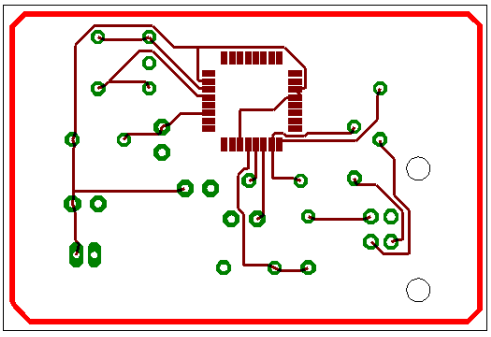
instala debajo de la barra de herramientas estándar y muestra los parámetros de configuración de usuario para la construcción de esta área.

Figura A16. Capas necesarias para implementar el área de cobre.



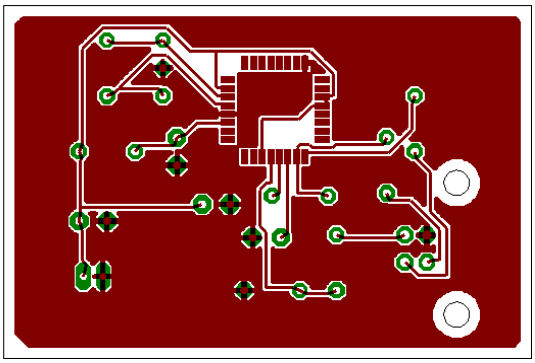
Por defecto se escogen los valores de ancho de 20 – 24 (*width*), la capa (*top* ó *bottom*) y luego se coloca en la barra de texto el nombre de la señal que quedará dentro del área de cobre (GND). Después se dibuja el área que se utilizará, y cuando se termina se hace doble clic, o se unen el principio y el final de la línea con un solo clic (figura A17).

Figura A17. Aplicación del comando *polygon*.



Finalmente, se utiliza el comando *Ratsnest*  , el cual escoge el mejor camino para enrutamiento (en las líneas sin enrutar), pero en este caso, se usa para que la tierra se disperse por el área que se le ha asignado. En la figura A18 se visualiza como queda finalmente el circuito impreso.

Figura A18. Etapa final con área de tierra.

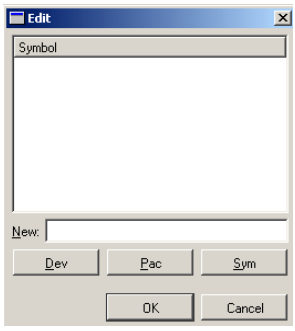


Hasta este punto, el usuario de *Eagle*, puede implementar los diseños de "hardware" en donde todos los elementos que necesite utilizar se encuentren en las librerías incorporadas por el "software", pero debido a que algunos no se encuentran en las librerías de *Eagle*, se requiere entonces que se realice el diseño de los mismos utilizando desde el panel de control la opción de *Library* del submenú *New*, del menú *File*.

El entorno de diseño de dispositivos, es muy parecido al del panel de diagramas esquemáticos, pero en éste, para empezar a diseñar se deben utilizar los iconos *Device* , *Package* y *Symbol* , con los cuales se implementan dispositivos, paquetes y símbolos esquemáticos respectivamente.

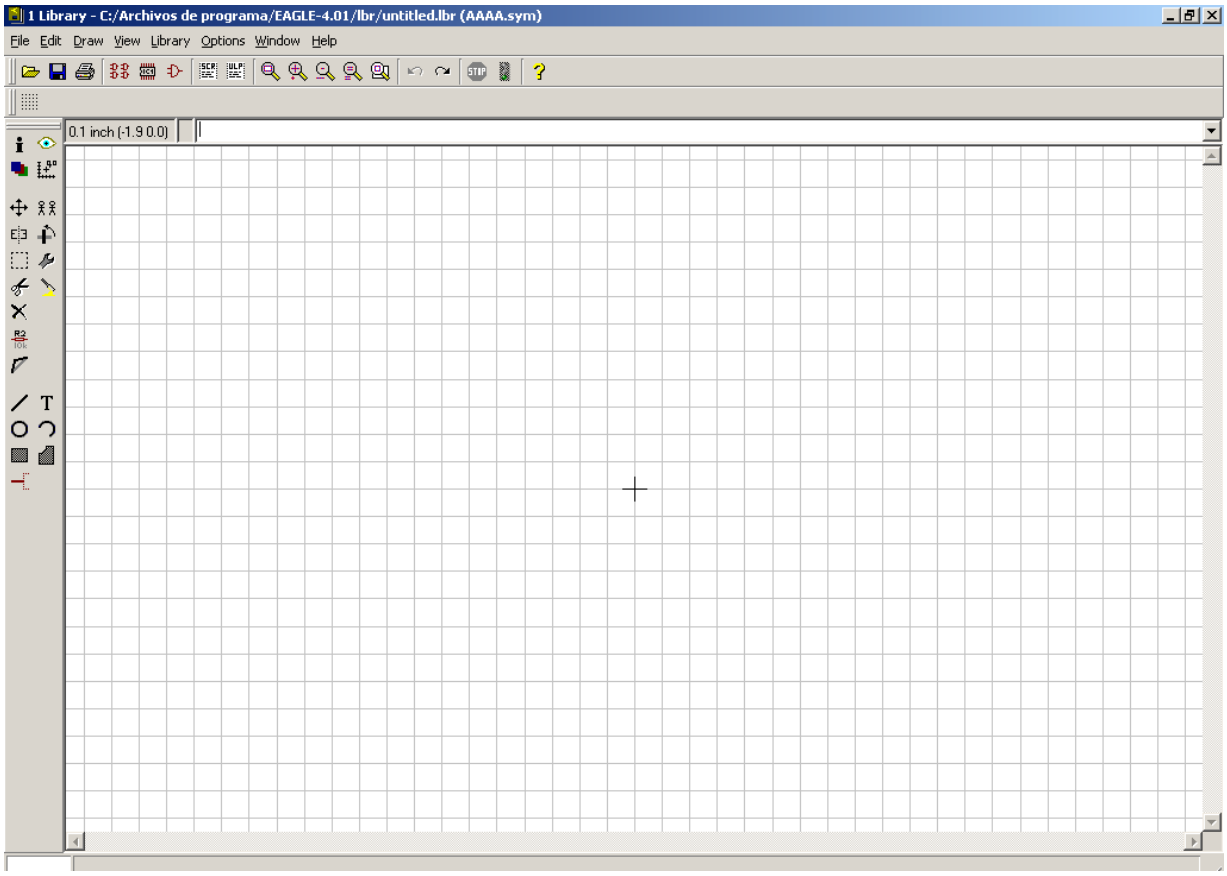
El primer paso es hacer clic en el icono *Symbol*, y aparece una ventana (figura A19), en donde se coloca el nombre del símbolo a crear (en esta ventana se puede volver a escoger si lo que se quiere es un paquete o un dispositivo).

Figura A19. Cuadro de diálogo para crear un símbolo.



Cuando se presiona el botón OK, aparece un cuadro de diálogo preguntando si desea crear el elemento, si la respuesta es afirmativa, la barra de herramientas de edición que se encuentra al lado izquierdo cambia, mostrando las nuevas posibilidades que se presentan (figura A20). En esta ventana se ocultan varias opciones que se observaban en el panel de diagramas esquemáticos, pero se aprecia una opción muy importante, que es la de crear pines (icono mostrado en la parte inferior izquierda de la barra de edición).

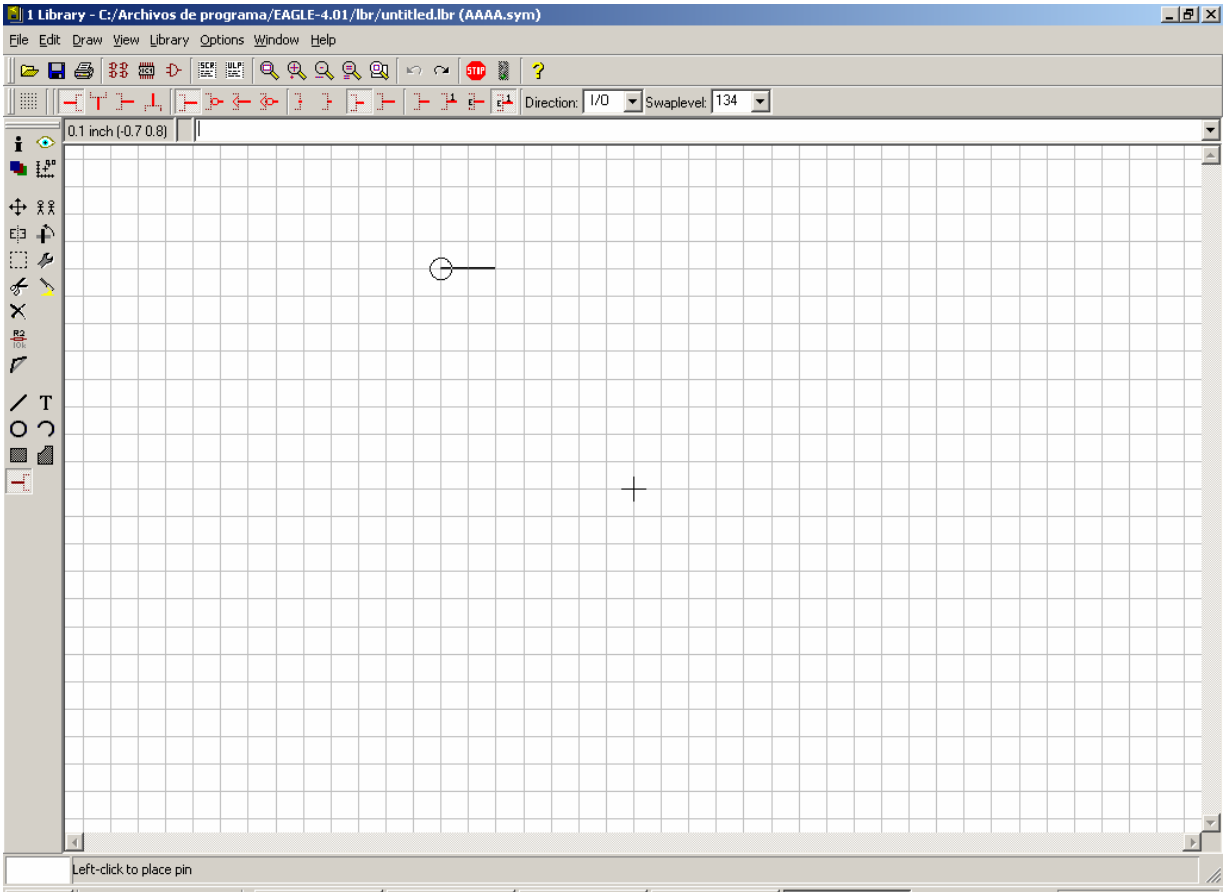
Figura A20. Módulo para crear símbolos esquemáticos.



Las figuras se crean con la herramienta para hacer líneas *Wire* , dibujando a través del módulo. Cuando se ha realizado el dibujo del símbolo a utilizar, entonces se empiezan a dibujar los pines correspondientes a este símbolo. El comando *pin* , es el usado para este propósito, y cuando éste se ejecuta, aparecen una serie de opciones (debajo de la barra de herramientas estándar) relacionados con la dirección del pin, el tipo de pin (si es reloj, entrada negada, o ninguno), el

tamaño del pin y la función del pin (entrada, salida, alimentación, etc.), entre otros (véase la figura A21).

Figura A21. Comando pin en ejecución.



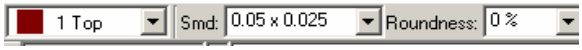
De igual forma para crear paquetes se elige el icono *Package*, y en la barra de herramientas de diseño aparecen (a diferencia con el panel de *board*) los iconos *pad* y *smd* los cuales sirven para crear el paquete.

El comando *pad* es utilizado para el diseño de elementos tipo DIP, ésta activa la barra de herramientas mostrada en la figura A22, en la que se puede escoger la forma del *pad* a utilizar y el tamaño del orificio (*Drill*) y el diámetro externo de dicho *pad*.

Figura A22. Barra de herramientas del comando *pad*.

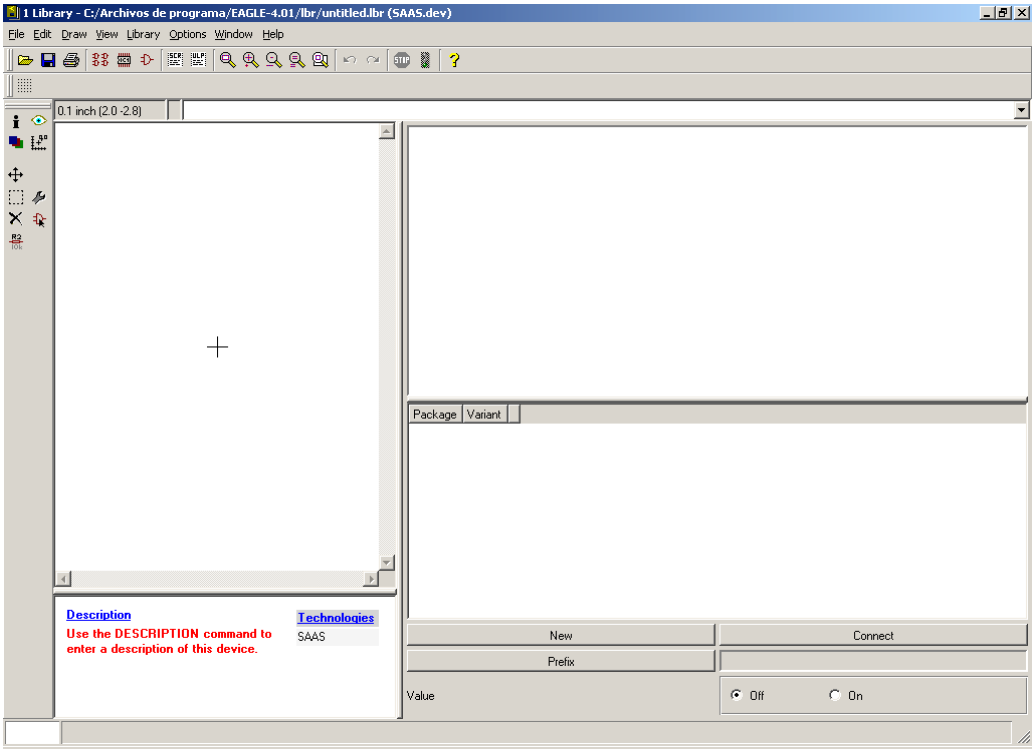
Por su parte el comando smd, es para el diseño de dispositivos superficiales, la barra de herramientas que activa (figura A23), tiene las opciones para elegir si se quiere el dispositivo en la cara superior o inferior, los diferentes tamaños disponibles y el tipo de redondeo que se desea.

Figura A23. Barra de herramientas del comando *smd*.



La última parte es la creación del dispositivo y en esta se observa el entorno visto en la figura A24, en el lado izquierdo, se observa un entorno parecido al de diagrama esquemático y allí con el comando *add*, se inserta el esquemático creado. Y en la parte derecha se observa el diagrama de la *board*, con el botón *new*, se toma el paquete creado para colocar en la tarjeta (la parte física). En el *prefix* (botón debajo de *New*), se colocan dos comillas sencillas (‘ ‘) y en C*onect* se hace la conexión entre el diagrama esquemático y el paquete.

Figura A24. Entorno de creación del dispositivo.



ANEXO B. ENTORNO DE PROGRAMACIÓN

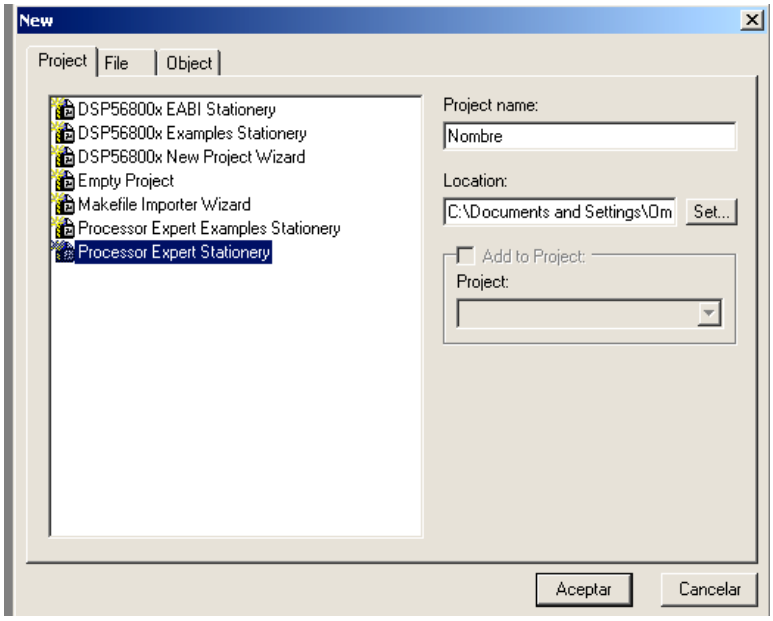
Gracias al avance de la tecnología, cada día se pueden desarrollar aplicaciones en donde el computador es fundamental para llevar a cabo los objetivos propuestos.

La empresa *Metrowerks* ha desarrollado por su parte un "software" gratuito muy robusto para la programación de diferentes dispositivos programables como microcontroladores y procesadores digitales de señal. Para la familia DSP56800 esta empresa proporciona una licencia de 16K en la versión 6.1 que resulta muy útil para desarrollos de equipos de diferente aplicabilidad. *Codewarrior Development Studio for Motorola* 56800 v 6.1 presenta las herramientas necesarias para utilizar por medio del *Processor Expert* aplicaciones con cualquier módulo del DSP.

Para elaborar un programa en *Codewarrior* se deben seguir los siguientes pasos:

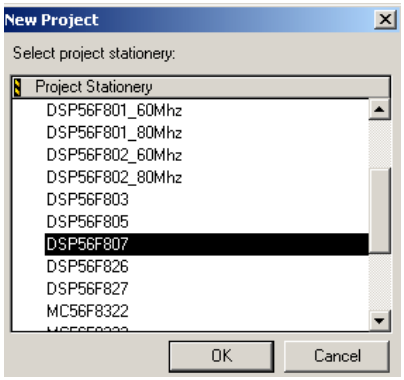
El primer paso para realizar una aplicación en el DSP es crear un proyecto nuevo por medio del comando *New* del menú *file*, o haciendo clic en el icono *New*  que se encuentra en la parte superior izquierda de la barra de herramientas. Al ejecutar esta acción, aparece un cuadro de diálogo en donde se escoge el tipo de proyecto a implementar, la ruta de almacenamiento en disco y el nombre que se le va a asignar al proyecto. La figura B1 muestra una ventana en donde se escogen las opciones de programación, en este caso, se opta por la última opción perteneciente al *Processor Expert Stationary*, se asigna un nombre y una dirección en disco. Finalmente se hace clic en el botón aceptar.

Figura B1. Propiedades generales del proyecto.



Luego, aparece una ventana en donde se escoge el tipo de procesador a utilizar. Por ejemplo para el proyecto que se desarrolló, se puede escoger el procesador DSP56F801 o el DSP56F807 (ver figura B2) y se pulsa el botón OK.

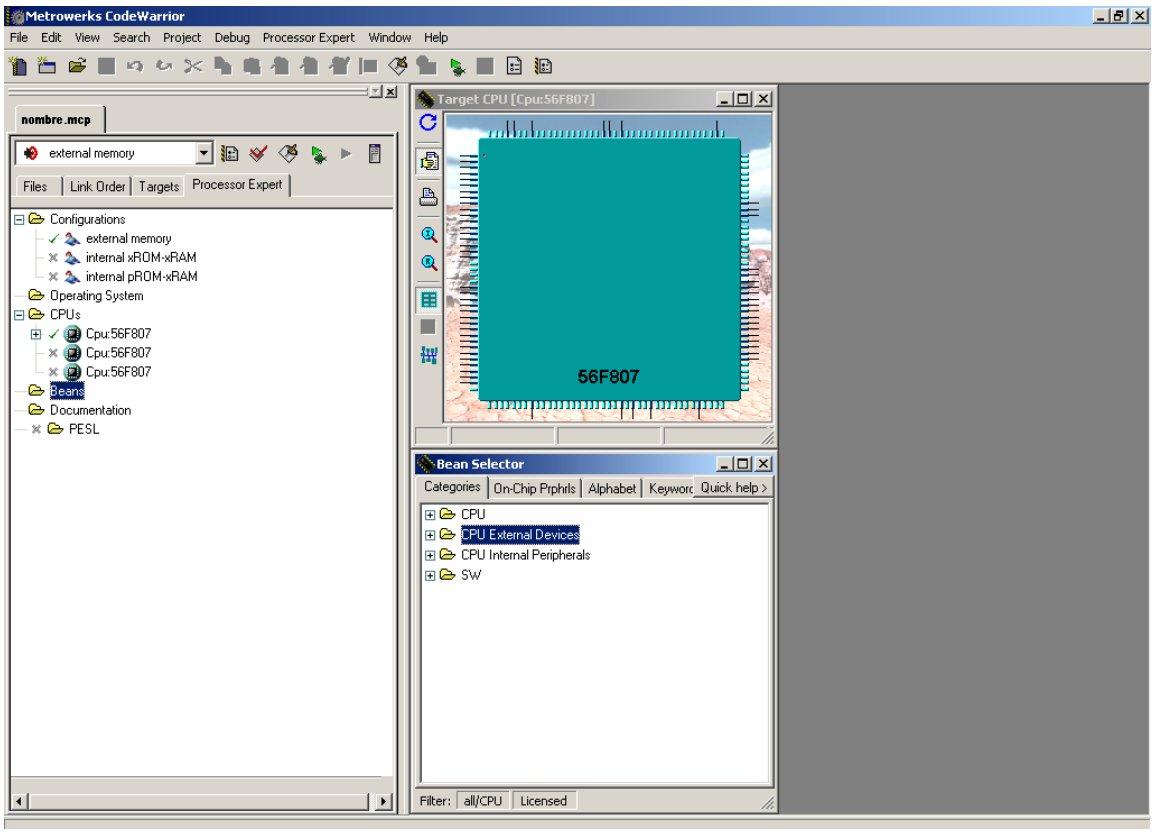
Figura B2. Ventana para escoger procesador.



En este paso se pueden añadir los *beans* que sean necesarios para desarrollar en el proyecto. Los *beans* son aplicaciones desarrolladas por los diseñadores del "software" (*Metrowerks*) para inicializar interfaces y aplicar algoritmos que se necesiten para realizar un debido procesamiento (Figura B3).

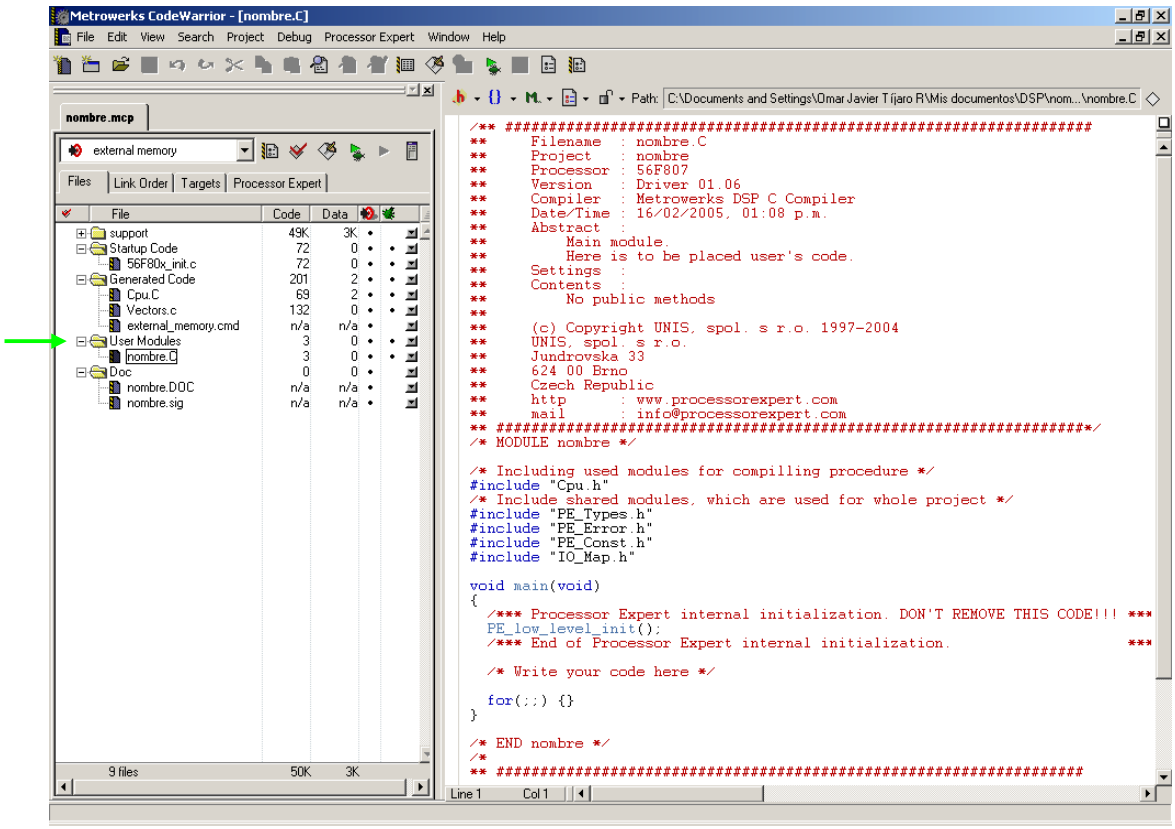
El DSP56F801 y el DSP56F807 se diferencian en este paso en que la memoria usada por este último puede ser externa, o interna, mientras que en el 56F801 no tiene acceso a memoria externa.

Figura B3. Módulos de inicio con el *Processor Expert*.



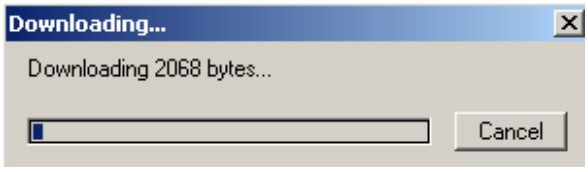
Después de esta etapa, es necesario hacer clic en el botón *make*  para que se generen los códigos de la CPU, memoria y *beans* escogidos, y se habilitan las funciones e interrupciones previamente seleccionadas. La figura B4 muestra en la parte izquierda los códigos generados por el *Processor Expert* y en la parte derecha el programa principal donde se encuentra la función *main*. Esta función se genera en un archivo que tiene el mismo nombre del proyecto con extensión .C dentro de la carpeta *User Modules* (penúltima carpeta vista de arriba hacia abajo en la figura B4 señalada con una flecha verde); y es el archivo donde el usuario realiza los algoritmos que quedarán como *firmware* dentro del DSP. La programación se trabaja en lenguaje C.

Figura B4. Programa principal y códigos generados



Cuando se terminan de programar los algoritmos se compila el algoritmo para observar los debidos errores y advertencias con el comando *make*, luego, para programar el dispositivo se utiliza el comando *Debug*, que se encuentra en el menú Project, o se hace clic en el botón  que se encuentra en la barra de herramientas. Cuando se está programando el dispositivo, el "software" presenta una ventana que muestra el estado de programación del dispositivo con el tamaño del programa en *bytes* (ver figura B5).

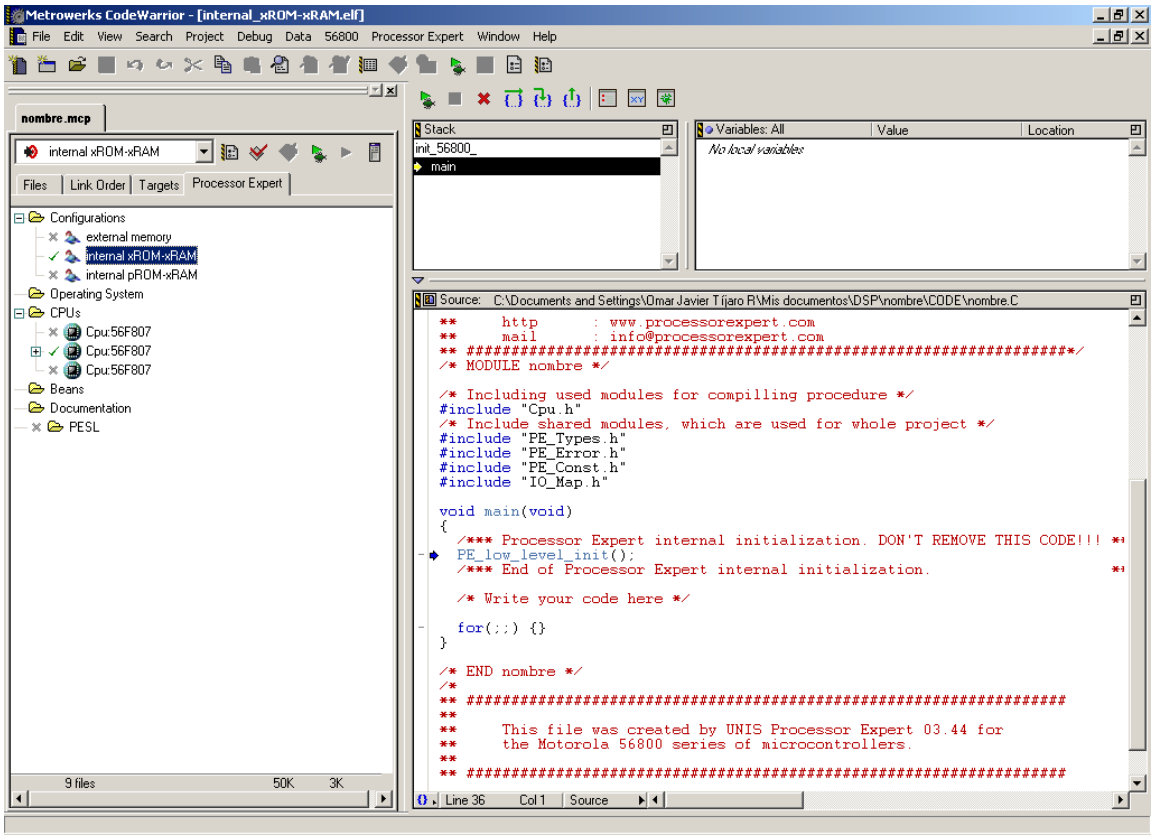
Figura B5. Descarga del programa al dispositivo.



Nota: En el DSP56F801 los valores predeterminados después del primer make son valederos para programar con Debug, pero para el 56F807 toca observar si las opciones de memoria están bien configuradas, es decir, si se quiere programar el dispositivo con memoria externa, las opciones se dejan como están por defecto, pero si desea trabajar con memoria interna, se debe activar en la pestaña Processor Expert la configuración de internal xROM-xRAM o pROM-xRAM, que se encuentra en la carpeta de Configurations.

Cuando el DSP se está programando no se deben tocar los botones de interrupciones ni reset de la tarjeta de desarrollo, por lo tanto si el "hardware" está conectado correctamente, se debe llegar a la ventana mostrada en la figura B6.

Figura B6. Entorno de la interfaz Debug.

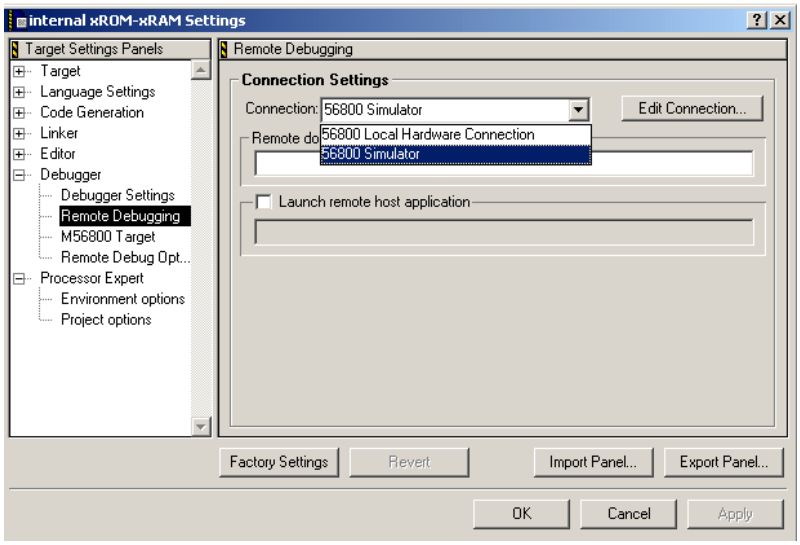


En el panel *Debug* se muestran los comandos necesarios para recorrer el algoritmo. Los comandos utilizados para este son: *Run* (el mismo símbolo del botón *Debug*) que se usa para correr el

programa libremente. *Break* que sirve para detener el modo *Run*. *Kill* utilizado para salir del panel y dejar programado el DSP. *Step Over* para recorrer el programa paso a paso. *Step Info* entrar a una función indicada y recorrerla paso a paso si no hay otras declaraciones. *Step Out* sirve cuando se está recorriendo una función (con *Step Info*) y se desea terminar de ejecutar los comandos pertenecientes a esta declaración, al finalizar la ejecución de este comando el puntero indicador de paso del programa aparece una línea hacia abajo de donde se había ejecutado *Step Info*.

Codewarrior permite también sin necesidad de crear otros proyectos, simular en "software" los algoritmos realizados para observar mediante variables locales los resultados que se están obteniendo en cada función. Es así como en el comando *internal xROM – xRAM settings*, del menú *Edit*, o pulsando *Alt+F7* en el teclado, se pueden llegar a las opciones donde se escoge si se quiere trabajar con el "hardware" conectado, o si se desea el modo simulador. La figura B7 este procedimiento.

Figura B7. Ventana de configuración.

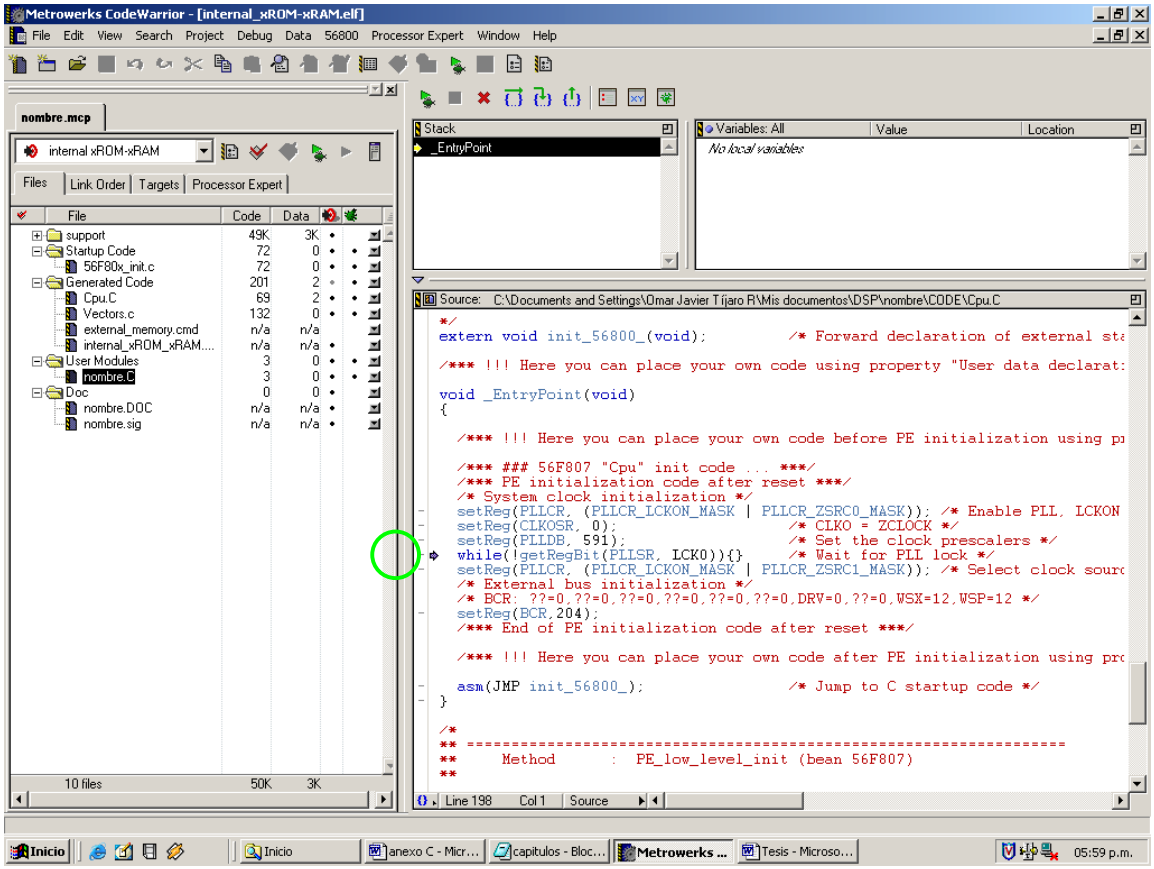


En la opción *Debugger* de la ventana vista en la figura B7 se encuentra la opción *Remote Debugging* y en la parte superior central se especifica el tipo de conexión, es decir, si está conectado en

"hardware" o se quiere trabajar con la opción del simulador. Después de escoger la característica deseada se hace clic en el botón OK.

Cuando se escoge la opción de simulación es importante tener en cuenta que cuando se hace clic en el botón *Debug* (descrito anteriormente), el programa se queda esperando la bandera donde se cierra el lazo del PLL (como lo muestra la figura B8), debido a que ésta, es una opción de "hardware" se procede a arrastrar la flecha azul (puntero indicador de recorrido del programa encerrado en un círculo verde en la figura B8), una línea hacia abajo para que no realice dicha confirmación.

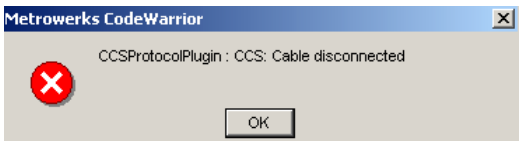
Figura B8. CPU.C. Modo simulación (detenido esperando bandera externa).



Después de correr el cursor a la línea de abajo se vuelve a pulsar el botón *Debug*, y el resultado de esta acción lleva el programa a la figura B6.

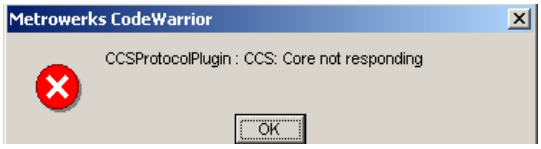
Los errores que se pueden presentar durante la programación del "hardware" se deben por lo general a fallas en los diferentes acoples, sin embargo, un *jumper* mal conectado puede deshabilitar el puerto JTAG y el computador a su vez observaría al dispositivo como si estuviese desconectado. Este error lo presenta el "software" como se ve en la figura B9.

Figura B9. Error de conexión de cable LPT1 desconectado o *jumpers* deshabilitando el JTAG.



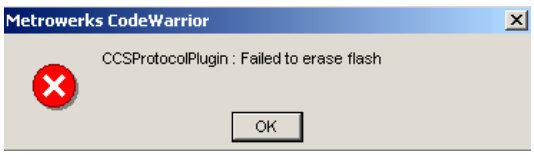
Otro error común es el que se debe a la conexión física del dispositivo y los problemas con la alimentación (figura B10). En este caso, se puede producir conflicto por no tener conectada la alimentación del sistema, o por no limpiar debidamente los integrados después de soldar, quedando en la superficie crema o impurezas que generan conducción por lo general entre los pines relacionados con V_{SS} y V_{DD}.

Figura B10. Error debido a mala conexión del dispositivo.



Un error generado por las interrupciones externas y por el reset mientras se está programando es que cuando se presiona uno de estos pulsadores (en "hardware") el dispositivo no puede borrar la memoria flash. Además, si los cables de la alimentación (+V y GND) chocan, también se puede presentar este problema (ver figura B11). Una posible solución es resetear el sistema con el botón reset o manualmente desconectando y conectando la alimentación.

Figura B11. Error de la programación de la memoria Flash.



ANEXO C. PROGRAMA DE APLICACIÓN DE LA FFT

En este anexo se muestra el "software" de la aplicación (analizador de espectros) programada en lenguaje C por medio de *Codewarrior* 6.1, para el DSP56F807 de *Motorola* con una entrada de tensión y salida en la pantalla LCD gráfica PG320240 de *Powertip Technology Corporation*.

```
#include "Cpu.h"
#include "DFR1.h"
#include "MEM1.h"
#include "TFR1.h"
#include "MFR1.h"
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

byte txt[]="   UNIVERSIDAD INDUSTRIAL DE SANTANDER ";
byte med[]="     ANALIZADOR DE ESPECTROS PORTATIL  ";
byte yar[]="Yair de Jesus Ruidiaz Palomino";
byte oma[]="Omar Javier Tijaro Rojas";
byte buc[]="Bucaramanga. 2005";
byte Limizq[4]="-37";
byte centro[2]="0";
byte Limider[6]="37kHz";

//Coeficientes del Filtro FIR antisolapamiento (tomados      de Matlab funcion FIR2 y buttord)
const Frac16 coeFIR[]={     1 ,    0 ,   -1 ,   -2 ,   -3 ,   -3 ,   -4 ,
   -4 ,   -5 ,   -5 ,   -4 ,   -4 ,   -3 ,   -3 ,   -2 ,   -1 ,    0 ,
    2 ,    3 ,    4 ,    5 ,    6 ,    6 ,    6 ,    6 ,    6 ,    6 ,
    5 ,    4 ,    2 ,    1 ,   -1 ,   -3 ,   -4 ,   -6 ,   -7 ,   -9 ,
   -9 ,  -10 ,  -10 ,   -9 ,   -8 ,   -7 ,   -5 ,   -3 ,   -1 ,    2 ,
    5 ,    7 ,   10 ,   12 ,   14 ,   15 ,   15 ,   15 ,   14 ,   13 ,
   10 ,    8 ,    4 ,    0 ,   -4 ,   -8 ,  -12 ,  -15 ,  -18 ,  -21 ,
  -22 ,  -23 ,  -23 ,  -21 ,  -18 ,  -15 ,  -10 ,   -5 ,    0 ,    6 ,
   12 ,   18 ,   23 ,   27 ,   31 ,   33 ,   33 ,   32 ,   30 ,   26 ,
   21 ,   14 ,    6 ,   -2 ,  -10 ,  -19 ,  -27 ,  -34 ,  -40 ,  -44 ,
  -47 ,  -47 ,  -45 ,  -41 ,  -35 ,  -27 ,  -18 ,   -7 ,    5 ,   16 ,
   28 ,   39 ,   49 ,   57 ,   62 ,   65 ,   65 ,   62 ,   56 ,   47 ,
   36 ,   22 ,    7 ,   -9 ,  -25 ,  -41 ,  -56 ,  -69 ,  -79 ,  -86 ,
  -90 ,  -89 ,  -84 ,  -76 ,  -63 ,  -47 ,  -28 ,   -7 ,   16 ,   38 ,
   60 ,   80 ,   98 ,  111 ,  120 ,  124 ,  123 ,  115 ,  102,   84 ,
   61 ,   34 ,    4 ,  -27 ,  -59 ,  -89 , -117 , -141 , -159 , -171 ,
 -176 , -173 , -161 , -142 , -115 ,  -82 ,  -42 ,    1 ,   47 ,   93 ,
  137 ,  178 ,  213 ,  240 ,  258 ,  265 ,  260 ,  242 ,  213 ,  171 ,
  118 ,   56 ,  -13 ,  -87 , -163 , -237 , -306 , -367 , -417 , -451 ,
 -468 , -464 , -438 , -388 , -313 , -215 ,  -92 ,   51 ,  214 ,  394 ,
```

```
 585 ,  785 ,  989 , 1191 , 1387 , 1571 , 1739 , 1887 , 2010 , 2106 ,
2171 , 2204 , 2204 , 2171 , 2106 , 2010 , 1887 , 1739 , 1571 , 1387 ,
1191 ,  989 ,  785 ,  585 ,  394 ,  214 ,   51 , -92 , -215 , -313 ,
-388 , -438 , -464 , -468 , -451 , -417 , -367 , -306 , -237 , -163 ,
 -87 ,  -13 ,   56 ,  118 ,  171 ,  213 ,  242 ,  260 ,  265 ,  258 ,
 240 ,  213 ,  178 ,  137 ,   93 ,   47 ,    1 , -42 , -82 , -115 ,
-142 , -161 , -173 , -176 , -171 , -159 , -141 , -117 ,  -89 ,  -59 ,
 -27 ,    4 ,   34 ,   61 ,   84 ,  102 ,  115 ,  123 ,  124 ,  120 ,
 111 ,   98 ,   80 ,   60 ,   38 ,   16 ,   -7 ,  -28 ,  -47 ,  -63 ,
 -76 ,  -84 ,  -89 ,  -90 ,  -86 ,  -79 ,  -69 ,  -56 ,  -41 ,  -25 ,
  -9 ,    7 ,   22 ,   36 ,   47 ,   56 ,   62 ,   65 ,   65 ,   62 ,
  57 ,   49 ,   39 ,   28 ,   16 ,    5 ,   -7 ,  -18 ,  -27 ,  -35 ,
 -41 ,  -45 ,  -47 ,  -47 ,  -44 ,  -40 ,  -34 ,  -27 ,  -19 ,  -10 ,
  -2 ,    6 ,   14 ,   21 ,   26 ,   30 ,   32 ,   33 ,   33 ,   31 ,
  27 ,   23 ,   18 ,   12 ,    6 ,    0 ,   -5 ,  -10 ,  -15 ,  -18 ,
 -21 ,  -23 ,  -23 ,  -22 ,  -21 ,  -18 ,  -15 ,  -12 ,   -8 ,   -4 ,
   0 ,    4 ,    8 ,   10 ,   13 ,   14 ,   15 ,   15 ,   15 ,   14 ,
  12 ,   10 ,    7 ,    5 ,    2 ,   -1 ,   -3 ,   -5 ,   -7 ,   -8 ,
  -9 ,  -10 ,  -10 ,   -9 ,   -9 ,   -7 ,   -6 ,   -4 ,   -3 ,   -1 ,
   1 ,    2 ,    4 ,    5 ,    6 ,    6 ,    6 ,    6 ,    6 ,    6 ,
   5 ,    4 ,    3 ,    2 ,    0 ,   -1 ,   -2 ,   -3 ,   -3 ,   -4 ,
  -4 ,   -5 ,   -5 ,   -4 ,   -4 ,   -3 ,   -3 ,   -2 ,   -1 ,    0 ,
   1 };

//Función para cambiar de tipo float a Frac16
Frac16 conver(float k)
{Frac16 c;
        c=k*0x7FFF;
        return c;
}
////Función para cambiar de tipo Frac16 a float
float frac2float(Frac16 k)
{float c;
        c=k/0x7FFF;
        return c;
}
//Retardos necesarios para la pantalla LCD gráfica
static void retardo30ns(void)
{
        asm
        {
        nop;
        }
}
static void retardo130ns(void)
{
        asm
        {
        nop;
        nop;
        nop;
        nop;
        nop;
```

```
        nop;
        }
}
/*
 _____________________________________
| PANTALLLA   | /CS  |   A0  |  DB0-7    |       ==================
|_____________|______|_____|_________ |              ~RD=1          ~WR=0;
| DSP56F807   | GPE1 |GPE0|GPIOB0-7 |       ==================
|_____________|______|_____|_________ |
*/
static void comando(byte com)    // Comandos de la pantalla
{
                GPIO_E_DR=0x01;       //A0=1; /CS=0
                retardo30ns();
                GPIO_B_DR=com;  //Palabra de comando
                retardo130ns();
                GPIO_E_DR=0x03;       //A0=1; /CS=1;
                retardo130ns();
                retardo130ns();
                retardo130ns();
                retardo130ns();
                retardo30ns();
}
static void configp(byte com)       //Configuración del comando
{
                GPIO_E_DR=0x00;  //A0=0; /CS=0
                retardo30ns();
                GPIO_B_DR=com;   //pn
                retardo130ns();
                GPIO_E_DR=0x02;  //A0=0; /CS=1
                retardo130ns();
                retardo130ns();
                retardo130ns();
                retardo30ns();
}
//Coloca el cursor en una direccion de memoria dado
static void diregcur(word ADD)                          //ADD-> Dirección de 16 bits
{byte HL,LL;
                //CSRW
                LL=ADD*0x00FF;
                HL=ADD>>8;
                comando(0x46);          //Comando -> Direccion de registro del cursor
                retardo30ns();
                configp(LL);  //p1
                retardo30ns();
                configp(HL);      //p2
                retardo30ns();
                //Se situa en el espacio de memoria 0x0000
}

/*Indica la direccion del cursor    0->Derecha      1->Izquierda
                                        2->Arriba       3->Abajo                */
static void Direccioncur(byte k)
```

```
{
        k=k+0x4C;
        comando(k);             //Comando CSRDIR
        retardo30ns();
}

static void Limpiarram(void)
{int i;
        //MWRITE
                diregcur(0);
                retardo30ns();
                Direccioncur(0);
                retardo30ns();
                comando(0x42);
                retardo30ns();
                for (i=0;i<1200;i++)
                {
                        configp(0x20);
                }
                retardo30ns();
                // Limpiado el espacio de caracteres

                comando(0x46);
                retardo30ns();
                configp(0xE8);
                retardo30ns();
                configp(0x03);
                retardo30ns();
                comando(0x42);
                retardo30ns();
                for (i=0;i<16000;i++)
                {
                        configp(0x00);
                }
                retardo30ns();
                //Limpiado el espacio gráfico

}

static void formacur(void)
{
        //CSRFORM
        comando(0x5D);          //Comando -> CSRFORM
        retardo30ns();
        configp(0x04);  //p1
        retardo30ns();
        configp(0x86);
        retardo30ns();
        //Forma del cursor cuadrada
}

static void InicioLCD(byte a,byte b)
{
```

```
//Inicio del LCD

//Configurar sistema
comando(0x40);          //Comando -> Fijar Sistema
retardo30ns();
configp(0x30);  //p1:  En este se fija el generador de caracteres, si se usa la
retardo30ns();   //                  CGRAM, el alto de los caracteres, si se usan dos paneles
                                    //                  y la correcion vertical.

configp(0x87);  //p2:     Ancho de caracter (FX) y control de forma de onda (WF)
retardo30ns();   //       el WF se utiliza para que a un ángulo dado se pueda ver bien
configp(a);      //p3:    Altura del caracter (0-F)
retardo30ns();
configp(0x27);  //p4:     Rango de direcciones cubierto por una línea de caracteres
retardo30ns();
configp(0x2A);  //p5:     TC/R: Fija la longitud horizontal de la línea
retardo30ns();
configp(0xEF);  //p6: Longitud de la altura de la pantalla
retardo30ns();
configp(0x28);  //p7  Rango de direcciones de la pantalla virtual APL
retardo30ns();
configp(0x00);  //p8  Rango de direcciones de la pantalla virtual APH
retardo30ns();
//Termino de configurar sistema

//Cursor
comando(0x44);          //Comando -> Scroll
retardo30ns();
configp(0x00);  //p1      SAD1L Posiciones de memoria
retardo30ns();
configp(0x00);  //p2      SAD1H Posiciones de memoria
retardo30ns();
configp(0xF0);  //p3      SL1=L/F
retardo30ns();
configp(0xB0);  //p4      SAD2L Posiciones de memoria
retardo30ns();
configp(0x04);  //p5   SAD2H Posiciones de memoria
retardo30ns();
configp(0xF0);  //p6   SL2=L/F
retardo30ns();
configp(0xC0);  //p7
retardo30ns();
configp(0x44);  //p8
retardo30ns();
configp(0x00);  //p9
retardo30ns();
configp(0x00);  //p10
retardo30ns();                  //Termino de configurar el cursor

//Posición Horizontal
comando(0x5A);          //Comando -> HDOT
retardo30ns();
configp(0x00);  //p1      Empieza en la esquina superior izquierda (sin desplazar)
```

```
	retardo30ns();
	//Termino de fijar la posición horizontal de la pantalla

	//OVLAY
	comando(0x5B);		//Comando -> OVLAY
	retardo30ns();
	configp(b);  //p1  Define pantalla gráfica y/o texto según las capas
	retardo30ns();				//Configuradas las capas de la pantalla
}


static void lineahoriz(int k,byte a)
{int i;
	Direccioncur(0);
	retardo30ns();
	comando(0x42);
	retardo30ns();
	for(i=0;i<k;i++)
	{
		configp(a);
		retardo30ns();
	}

}

static void lineasvert(byte a)	//a->Byte que se desea dibujar verticalmente
{
	configp(a);
	retardo30ns();
}
static void displayon(void)
{
	//Display
	comando(0x59);		//Comando -> Display ON
	retardo30ns();
	configp(0x16);
	retardo30ns();
	//Display encendido

}

static void escribir(const byte *p,int n)
{int i;
	comando(0x4C);
	retardo30ns();
	comando(0x42);
	for(i=0;i<n;i++)
	{
		configp(*(p+i));
		retardo30ns();

	}
}
```

```
/*************************************************************
Este código sirve para graficar en la pantalla LCD gráfica
                el espectro en potencia.
*************************************************************/
static void graficarFFT(float a[4],float may,int t)
{int b[4],temp; int k,j;byte c;

//Se busca la norma para llevar el mayor a 160
for (k=0;k<4;k++)
{
        b[k]=(int)(((float)a[k]/may)*160);
        a[k]=b[k];
}

//Se ordena el vector b de menor a mayor
for(k=0;k<3;k++)
{
        for(j=k+1;j<4;j++)
        {
                if(b[j]<b[k])
                {
                        temp=b[k];
                        b[k]=b[j];
                        b[j]=temp;
                }
        }
}

//La variable c es la máscara para graficar
c=0xFF;
diregcur(0x229C-t);
retardo30ns();
Direccioncur(2);        //Direccion hacia arriba
retardo30ns();
comando(0x42);
retardo30ns();
for (k=0;k<b[0];k++)
{
        lineasvert(c);
        retardo30ns();
}

//Enmascarar a c
for(k=1;k<4;k++)
{
        if(a[0]<=b[k-1])
                c=c&0x3F;
        if(a[1]<=b[k-1])
                c=c&0xCF;
        if(a[2]<=b[k-1])
                c=c&0xF3;
```

```
        if(a[3]<=b[k-1])
                c=c&0xFC;
        for (j=b[k-1];j<b[k];j++)
        {
                lineasvert(c);
                retardo30ns();
        }
}
}

/*************************************************************
Borrado de la parte gráfica donde se dibuja el espectro
*************************************************************/
static void borrarespectro(void)
{int i,j;
        for(j=0;j<32;j++)
        {
                diregcur(0x229C-j);
                retardo30ns();
                comando(0x42);
                retardo30ns();
                for (i=0;i<160;i++)
                {
                        lineasvert(0x00);
                }
        }
}
/*********************************************************
Configuración de los puertos de control y datos de
                        la pantalla
*********************************************************/
static void iniptos(void)
{
        GPIO_E_IAR=0;
        GPIO_E_IENR=0;
        GPIO_E_IPOLR=0;
        GPIO_E_IESR=0;
        GPIO_E_PER=0;
        GPIO_E_DDR=0xFFFF;
        GPIO_E_DR=0;

        GPIO_B_IAR=0;
        GPIO_B_IENR=0;
        GPIO_B_IPOLR=0;
        GPIO_B_IESR=0;
        GPIO_B_PER=0;
        GPIO_B_DDR=0xFFFF;
        GPIO_B_DR=0;
}

/*********************************************************
Configuración inicial del conversor analógico a digital
*********************************************************/
```

```
static void configADC(void)
{
 //STOP=1,START=0,SYNC=0,EOSIE=1,ZCIE=0,LLMTIE=0,HLMTIE=0,CHNCFG=0,SMODE=4
  setReg(ADCA_ADCR1,18436);
  //OFFSET=7832
  setReg(ADCA_ADOFS0,7832);
  //HLMT=32760
  setReg(ADCA_ADHLMT0,32760);
  //LMT=0
  setReg(ADCA_ADLLMT0,0);
//CIP=0,EOSI=1,ZCI=0,LLMTI=0,HLMTI=0,RDY7=0,RDY6=0,RDY5=0,RDY4=0,RDY3=0,RDY2=0//,R
DY1=0,RDY0=0
  setReg(ADCA_ADSTAT,2048);
  // ADCA_ADSDIS: TEST=0,DS7=1,DS6=1,DS5=1,DS4=1,DS3=1,DS2=1,DS1=1,DS0=0
  setReg(ADCA_ADSDIS,254);
  // ADCA_ADLST1: SAMPLE3=0,SAMPLE2=0,SAMPLE1=0,SAMPLE0=0
  setReg(ADCA_ADLST1,12816);
  // ADCA_ADCR2: DIV=3
  setReg(ADCA_ADCR2,3);
  clrRegBit(ADCA_ADCR1,STOP);
}

/*******************************************************
        Títulos, nombres y textos del equipo
*******************************************************/
static void Textos(void)
{int k;byte p=0x7C;
  Limpiarram();
  displayon();
  formacur();
  diregcur(0);
  escribir(txt,40);                     //UNIVERSIDAD INDUSTRIAL DE SANTANDER
  diregcur(0xB1);
  escribir(med,40);                     //ANALIZADOR DE ESPECTROS
  diregcur(0x4C3);
  escribir(yar,30);                     //YAIR RUIDIAZ
  diregcur(0x498);
  escribir(oma,24);                     //OMAR TIJARO
  diregcur(0x46C);
  escribir(buc,18);                     //Bucaramanga
  diregcur(0x4F5);
  escribir(Limider,6);                  //37 kHz
  diregcur(0x304);
  escribir(centro,2);                   //0
  diregcur(0x316);
  escribir(Limizq,3);                   //-37
  diregcur(0x33C);
  for (k=0;k<32;k++)
  {
        escribir(&p,1);
  }
}
```

```
void main(void)
{
CFrac16 b[128],c[128];Frac16 muestras,err;float may,muest[4],h[128];
int k,i,j,w,ADC[1442];dfr16_tCFFTStruct *a;unsigned int y;float yy,zz,rr;

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/

iniptos();
configADC();

InicioLCD(0x07,0x01);
a=DFR1_dfr16CFFTCreate(128,FFT_DEFAULT_OPTIONS);
DFR1_dfr16CFFTInit(a,128,FFT_DEFAULT_OPTIONS);
Textos();

  for(;;) {

//Adquisición de la señal
        for(i=0;i<1442;i++)
        {
                setRegBit(ADCA_ADCR1,START); // Inicio de conversión
                while(getRegBit(ADCA_ADSTAT,CIP)){} // Conversion en Progreso
                ADC[i]=ADCA_ADRSLT0;
        }

//Filtro antisolapamiento,diezmado y enventanado
for (k=0;k<128;k++)
        {
        j=k*8; rr=0;
        for (i=0;i<419;i++)
        {
                may=coeFIR[418-i]*((int)(100*((float)(ADC[i+j])/0x7FFF)));
                rr=rr+may;
        }
        rr=rr/32636;
        if (k<64)
        muestras=conver((float)k/63.5);
        else
        {
                i=k-127;
                muestras=conver((float)i/63.5);
        }
        muestras=muestras+0x01;
        j=(int)(100*((float)(TFR1_tfr16CosPIx(muestras))/0x7FFF));
        i=(99-j);
        b[k].real=(int)((float)(rr*i)/19.8);
        b[k].imag=0;
}
borrarespectro();
DFR1_dfr16CFFT(a,b,c);                  //FFT
```

```
//Se halla la magnitud al cuadrado de la señal y se busca el mayor
may=0;
        for (k=0;k<128;k++)
        {
                yy=(float)(c[k].real)*(float)(c[k].real);
                zz=(float)(c[k].imag)*(float)(c[k].imag);
                rr=(yy+zz)/10000;
                h[k]=rr;
                if(rr>may)
                may=rr;
        }

//Se preparan las muestras para graficar
for(k=0;k<32;k++)
{
        if(k>=16)
        {
                w=k-16;
        }
        else
        {
                w=k+16;
        }
        i=4*k;
        for(j=0;j<4;j++)
        {
                muest[j]=h[j+i];
        }
        graficarFFT(muest,may,w);
 }
}
}
```

ANEXO D. MANUAL DE LAS TARJETAS DE DESARROLLO

En este anexo se muestran las tarjetas de desarrollo (figuras D1-D5) y se especifican cada uno de sus puertos, sus dispositivos indicadores de alimentación, *jumpers* y pulsadores. A continuación se describirán los pasos requeridos para programar los DSPs.

La alimentación para el DSP56F801 debe ser de 3,3 V DC y su corriente típica debe ser de 350mA. Para el DSP56F807 su alimentación debe ser de 9VDC a 500mA. El cable del puerto paralelo del computador debe ser conectado a la entrada PC-LPT1 del DSP. En el caso del 56F801 se debe conectar el puerto de programación (JTAG) con el respectivo conector de la tarjeta de desarrollo del DSP. Los *jumpers* deben estar conectados de la siguiente manera: en el 56F801, se debe colocar el *jumper* J activar JTAG, y el de alimentación Buffer 1 se puede conectar entre 1 y 2, o entre 2 y 3. En el caso del 56F807 los *jumpers* a conectar son el de memoria (J MEMOR), el cual debe estar conectado si se trabaja con memoria interna; y los dos de cristal externo o señal de reloj externa. Si se utiliza el oscilador de cristal, se conecta el *jumper* J XTAL y el J EXTAL entre 2 y 3. Si se usa oscilador externo, se levanta el *jumper* J XTAL y se introduce la señal por el terminal 1, el *jumper* J EXTAL se conecta entre 1 y 2.

Figura D1. Puerto de programación (JTAG - *Join Test Action Group*) para DSP56F801.

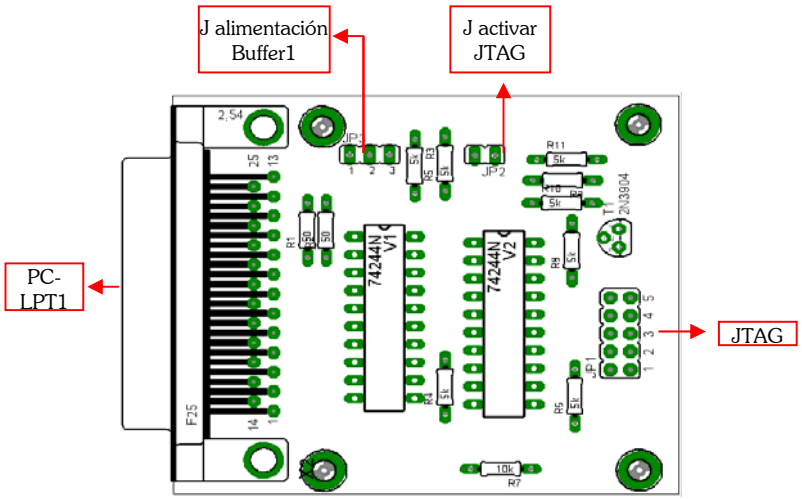


Figura D2. Tarjeta de desarrollo del DSP56F801.

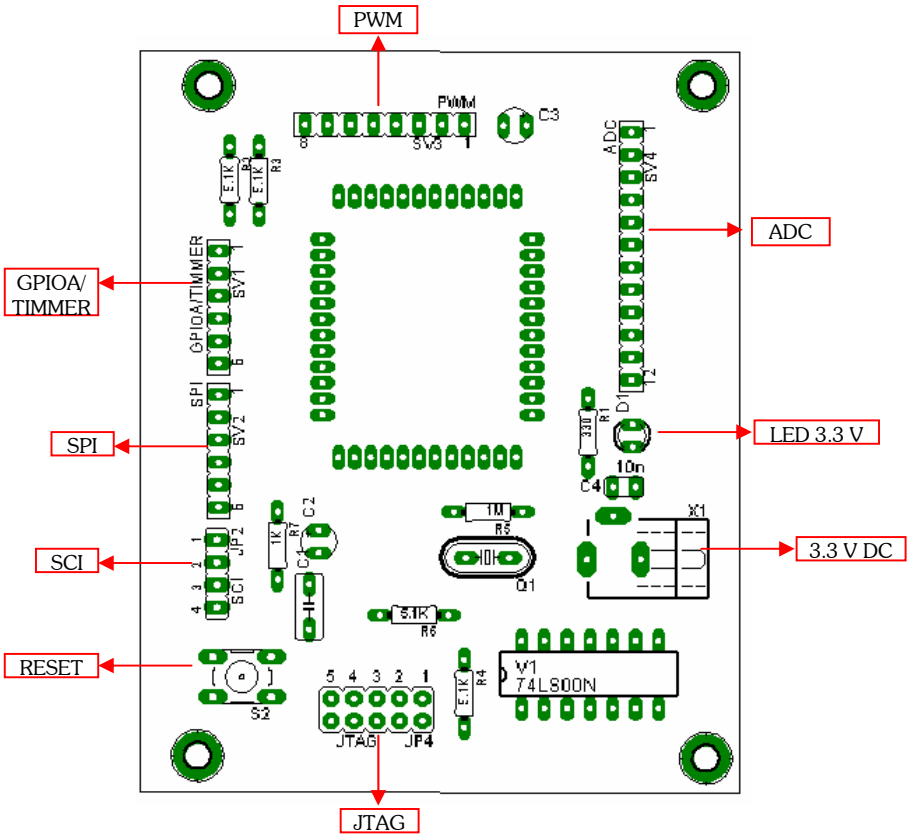


Figura D3. Tarjeta de acople entre el DSP y tarjeta de desarrollo del DSP56F801.

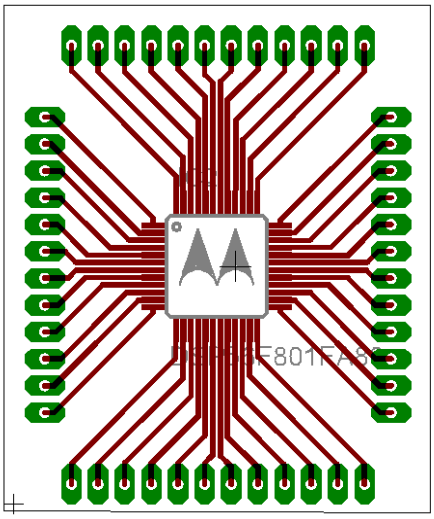


Figura D4. Tarjeta de desarrollo del DSP56F807.

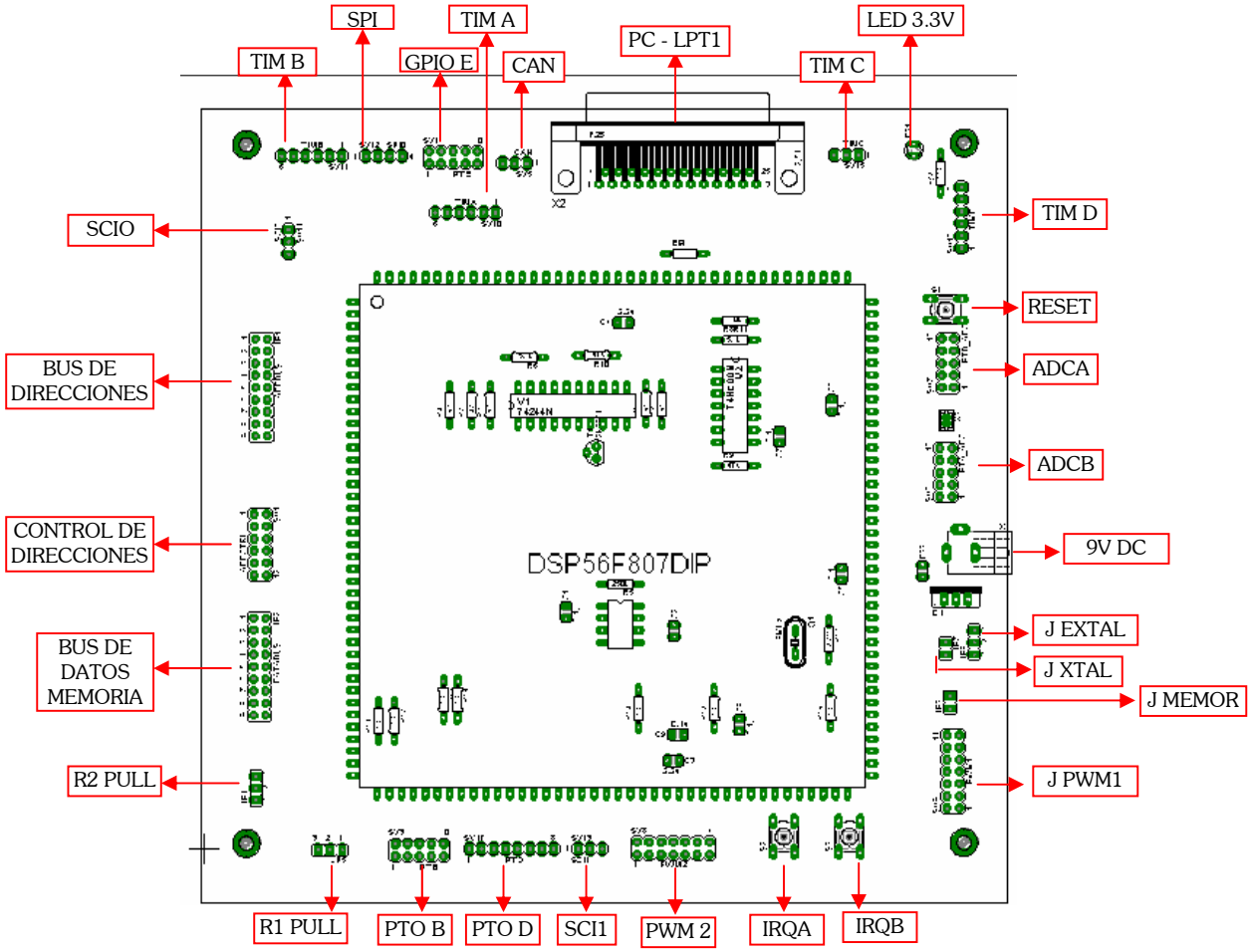
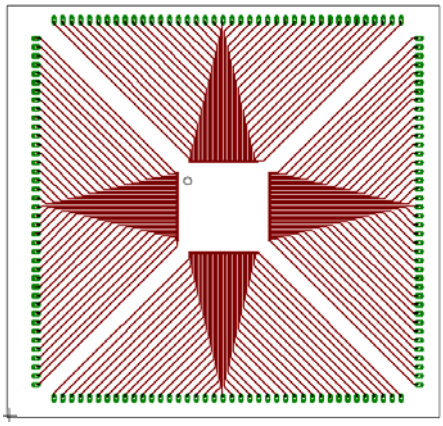


Figura D5. Tarjeta de acople entre el DSP y tarjeta de desarrollo del DSP56F807.



Se recomienda en ambos casos no introducir al conversor analógico a digital del DSP señales mayores a 3,3V y menor a 0 V.

ANEXO E. IMPLEMENTACIÓN DETALLADA DE TARJETAS DE DESARROLLO

En este anexo se muestra la implementación completa del hardware de las tarjetas de desarrollo de dos Procesadores de Señal Digitales (DSP) de la familia 56F800 de Motorola, que son los dispositivos que se utilizan en la actualidad para desarrollar funciones que requieran procesamiento de señal.

TARJETAS DE DESARROLLO

Las tarjetas de desarrollo implementadas son las correspondientes al DSP56F801 y DSP56F807, los cuales son el más pequeño y el más grande de la familia de acuerdo al tamaño, la cantidad de pines y módulos que tienen, la capacidad de almacenamiento en memoria y la corriente que consume cada uno.

Tarjeta de desarrollo del DSP56F801

Se utilizaron tres tarjetas. Una fue para programación por el puerto paralelo del computador, otra para aplicación y una última para conectar o desconectar el DSP, alimentadas con 3,3 V (para todo el sistema), y comunicadas cada una por correas o conectores. El procedimiento fue el siguiente:

- Primero. Se escogió el software para realizar el diseño de hardware. El resultado de esta elección fue "*Eagle 4,01 Lite Edition*" por su fácil manejo, sus múltiples funciones y porque su base de datos tiene la mayoría de dispositivos que se necesitaban (Ver anexo A).
- Segundo. Se creó la librería para el correspondiente DSP y los dispositivos que no estaban en la base de datos del "software", esto implicaba el diseño el símbolo esquemático (figura E1), del empaquetado (figura E2) y la conexión del dispositivo (figura E3).

Figura E1. Símbolo esquemático del DSP56F801

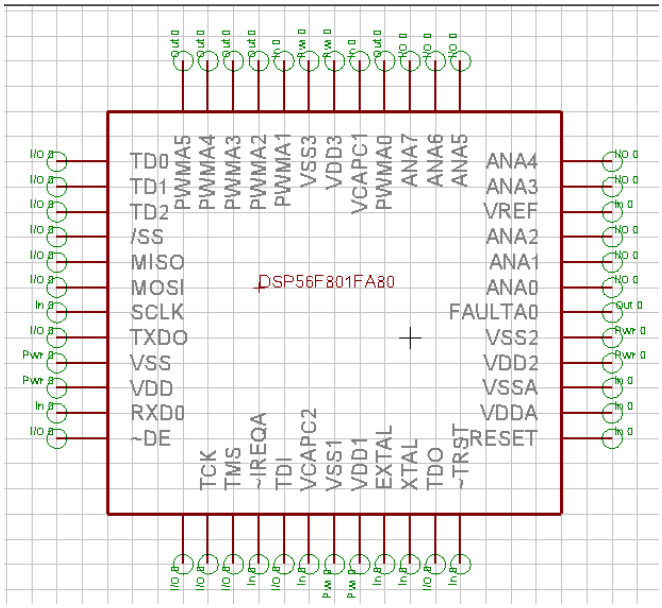


Figura E2. Empaquetado LQFP 48 pines.

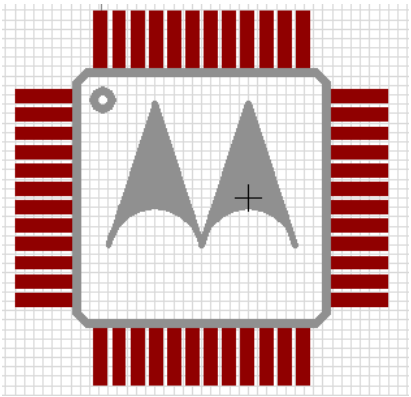
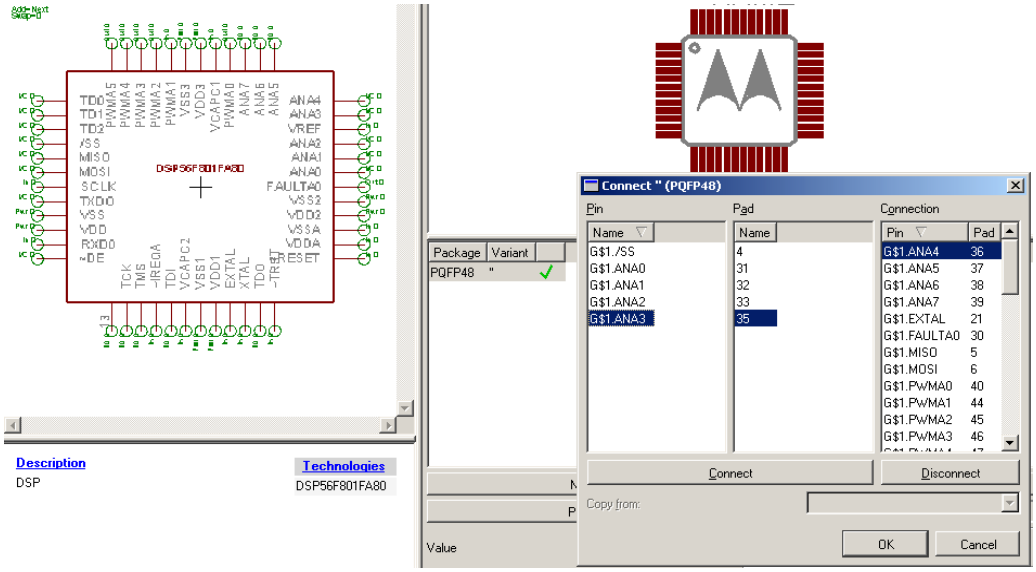


Figura E3. Fase final del diseño del circuito integrado.



Siguiendo los mismos pasos se diseñaron los dispositivos que no estaban en la base de datos.

- Tercero. Se escogieron los dispositivos a utilizar con el comando ***add***  y se trazaron los caminos con el comando ***net***  en un circuito esquemático. En esta etapa se diseñó una tarjeta para programación (figura E4), una para aplicación (figura E5) y otra para el DSP (figura E6).

Figura E4. Puerto de programación (JTAG - Join Test Action Group)

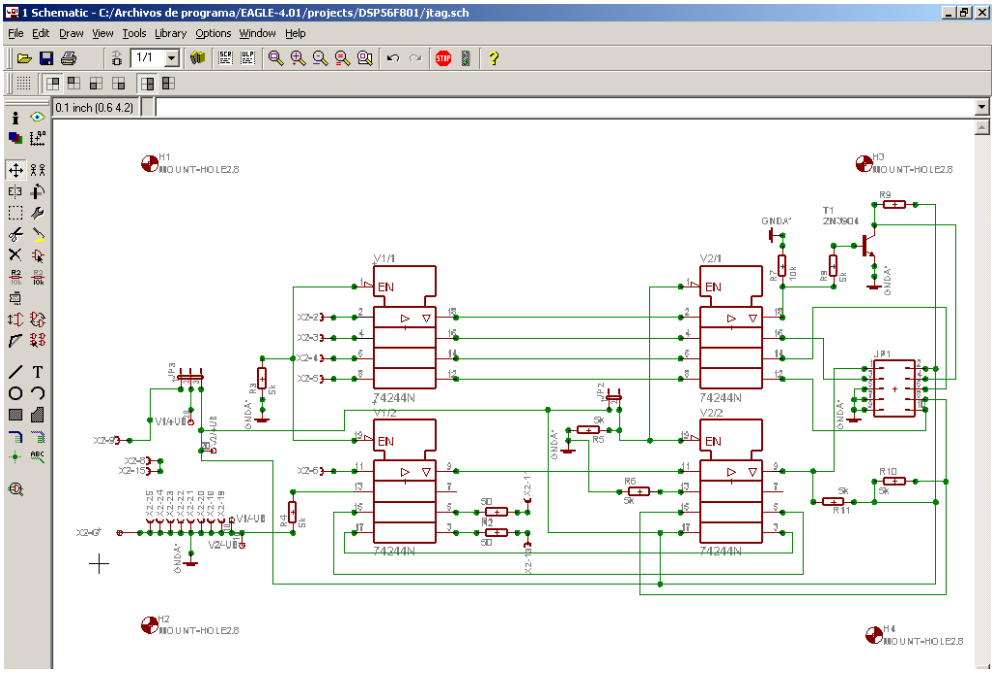


Figura E5. Tarjeta para desarrollo del 56F801.

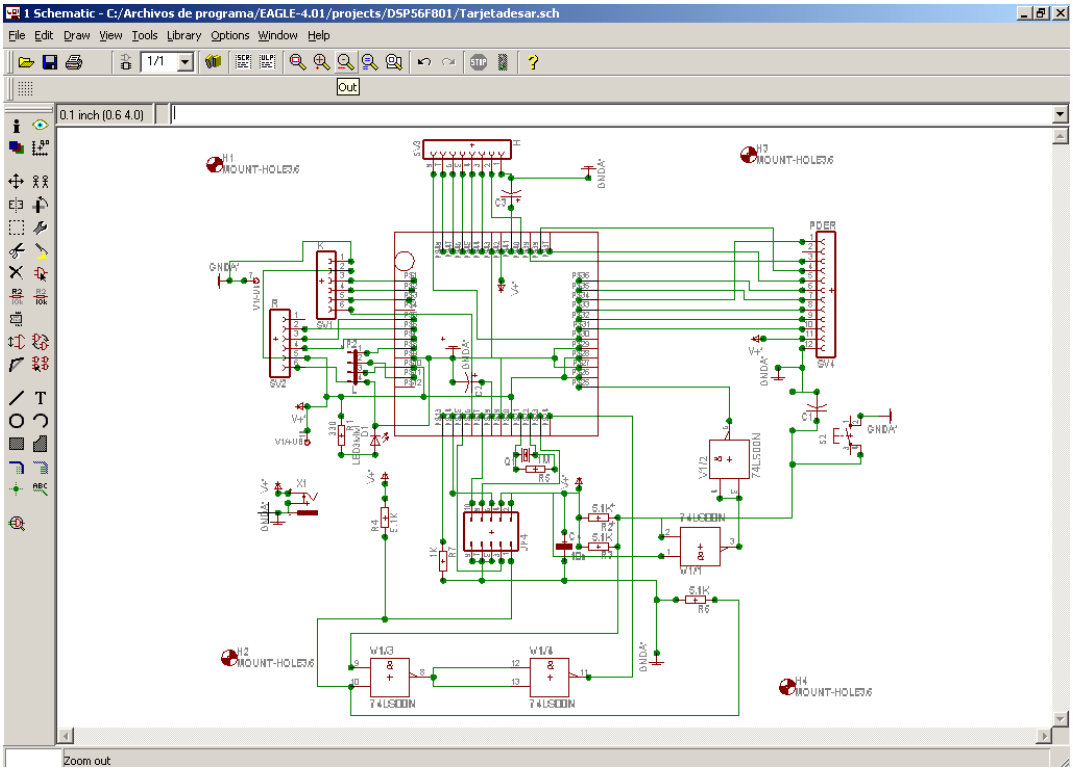
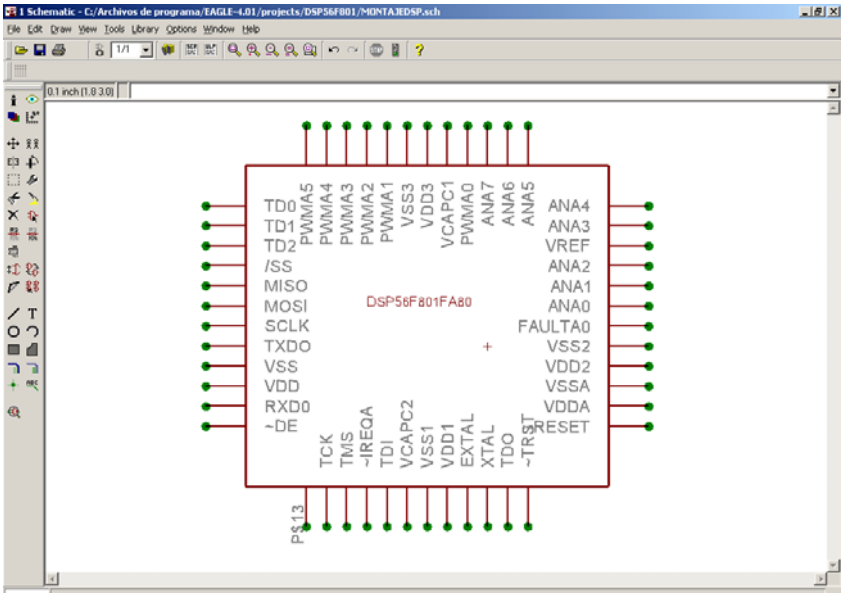


Figura E6. Tarjeta de acople de empaquetado LQFP del DSP a DIP. El símbolo en ésta es el mismo, por lo tanto solo se ve un dispositivo.



- Cuarto. Antes de realizar el enrutamiento se determinó cuanta corriente se transportaría en las pistas o caminos, la cual no debería ser mayor de lo que resistiera el material según el ancho de las líneas de enrutado. La corriente que disipa este dispositivo es de 300 mA, pero se tomó un poco superior para construir la tarjeta (500 mA) debido a los dispositivos que se conecten pueden pedir un poco más de corriente; y por las fallas mecánicas que se suelen tener cuando se está soldando, es decir, que las pistas no soporten el calor y se despeguen, la anchura escogida fue 0,024 pulgadas siendo recomendable tomar por lo menos 0,001. [Fink y Christiansen, 1992]
- Quinto. Se procedió con el enrutamiento del circuito impreso, ordenando previamente cada uno de los elementos, para que se facilitara tal actividad. En algunos impresos se pudo utilizar una sola cara para el enrutamiento, mientras en otros se hicieron necesarias dos caras. La figura E7 muestra el impreso final del DSP56F801, el cual se puede sacar de la tarjeta de desarrollo y montarlo en otras aplicaciones. El puerto de programación visto en las figuras E8, E9 y E10 se llevó de 14 a 10 pines con el objetivo de ahorrar espacio. La tarjeta de desarrollo se observa de la misma forma en las figuras E11, E12 y E13.

Figura E7. Circuito impreso de tarjeta de acople del DSP con soporte de 48 pines.

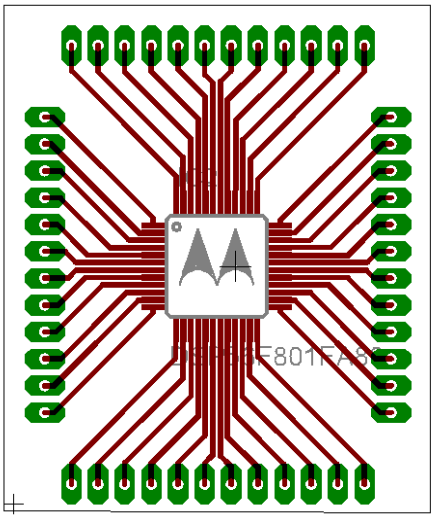


Figura E8. Enrutado inferior de la tarjeta programadora (JTAG).

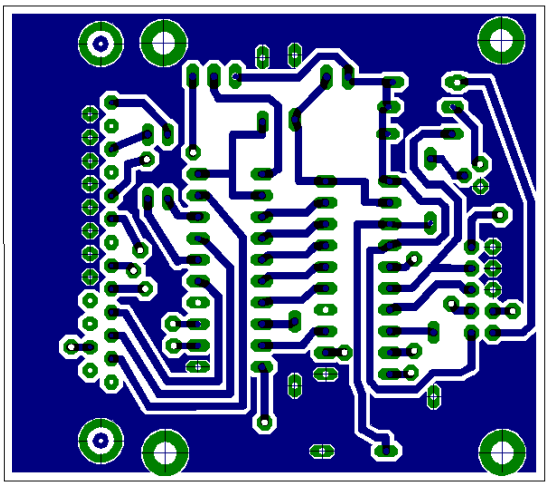


Figura E9. Enrutado superior de la tarjeta programadora (JTAG).

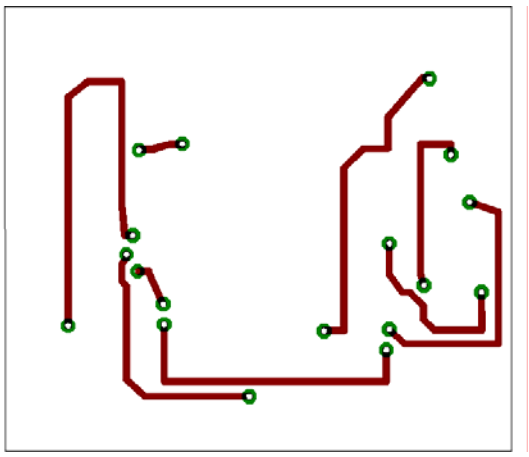


Figura E10. Vista de componentes de la tarjeta programadora (JTAG).

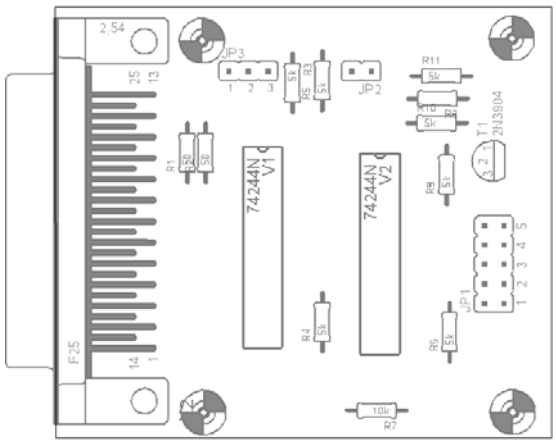


Figura E11. Enrutado inferior de la tarjeta de desarrollo.

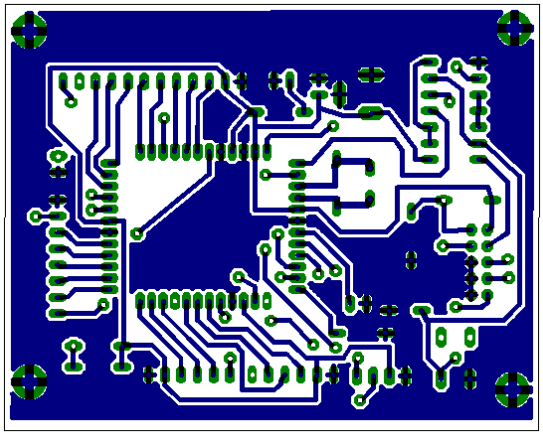


Figura E12. Enrutado superior de la tarjeta de desarrollo.

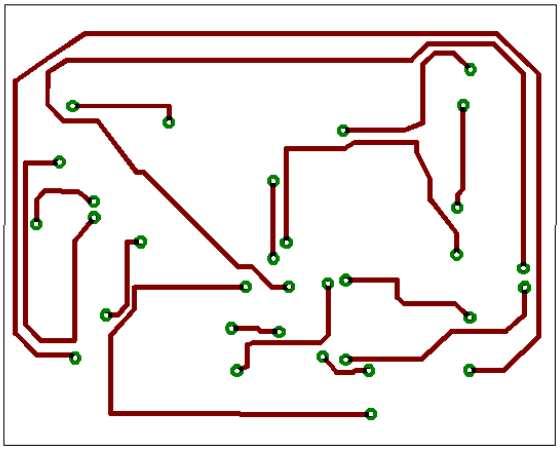
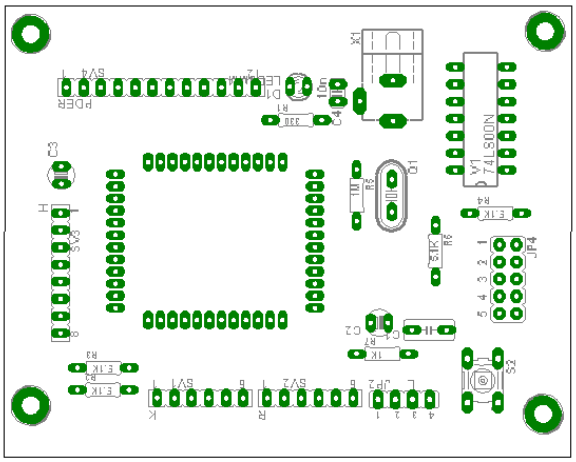


Figura E13. Vista de componentes de la tarjeta de desarrollo.



- Sexto. Se implementaron los circuitos impresos a doble cara en las tarjetas programadora y de desarrollo, y a una cara en la de acople del DSP. Se escogió el material de fibra de vidrio ya que sus características se ajustan a las condiciones requeridas.
- Séptimo. Teniendo la tarjeta con sus debidas perforaciones se continuó con la soldadura, comenzando con las vías (caminos que pasan entre las dos caras) y luego se prosiguió con los implementos.

- Octavo. Se limpia la tarjeta cuidadosamente con tiner y detergente en polvo para eliminar los residuos dejados por la crema soldadora u otras impurezas que se adhieren al momento de soldar, ya que estas pueden causar corto circuito.
- Noveno. Finalmente se realiza la prueba del hardware.

Tarjeta de desarrollo del DSP56F807

Se fabricaron dos tarjetas (a diferencia de las tres implementadas anteriormente en el DSP56F801), una de ellas con los módulos de programación y desarrollo, este último con todos los pines disponibles como puertos o como módulos embebidos y con un conector tipo NEB21R el cual se alimentó con una fuente que tuviese un voltaje DC entre 7,5 V - 12 V y una corriente de 500mA típicamente; y otra para acoplar el DSP con la tarjeta anteriormente descrita.

El procedimiento para el diseño fue el mismo descrito en la tarjeta de desarrollo del DSP56F801, por lo tanto no se mostrarán los pasos a seguir, sino solo las figuras E14-20 con sus respectivos rótulos de descripción, para la creación del circuito integrado se muestra solo el dispositivo, ya que en este se puede ver tanto el símbolo esquemático y el empaquetado.

Nota: Cuando se menciona en esta sección la frase tarjeta de desarrollo del DSP56F807 se hace referencia a los módulos de programación y desarrollo.

Figura E14. Fase final del integrado DSP56F807.

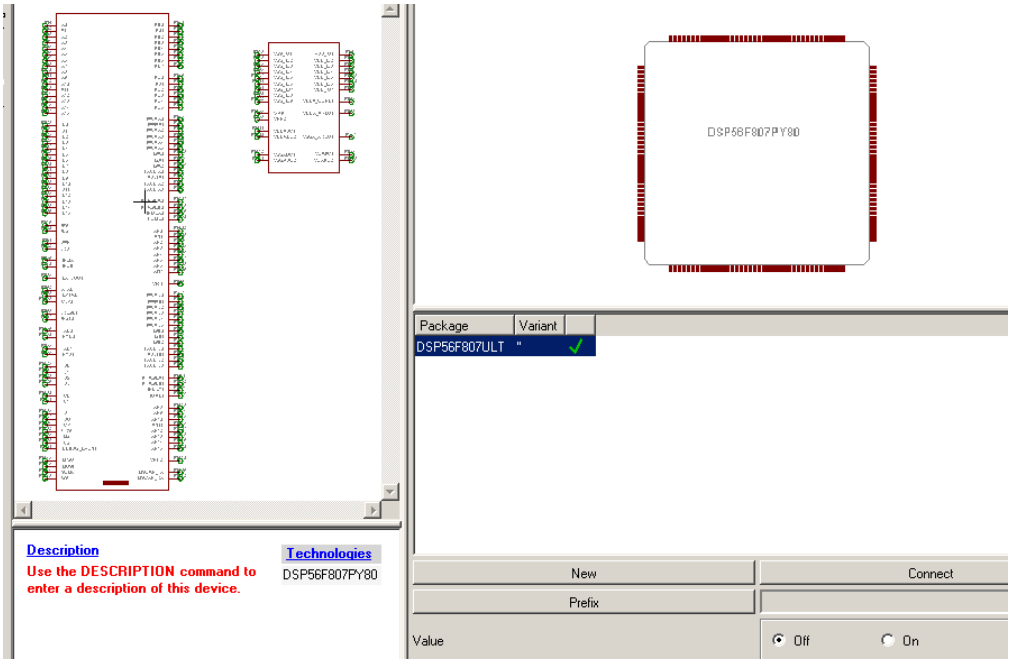


Figura E15. Diagrama esquemático de la tarjeta de desarrollo.

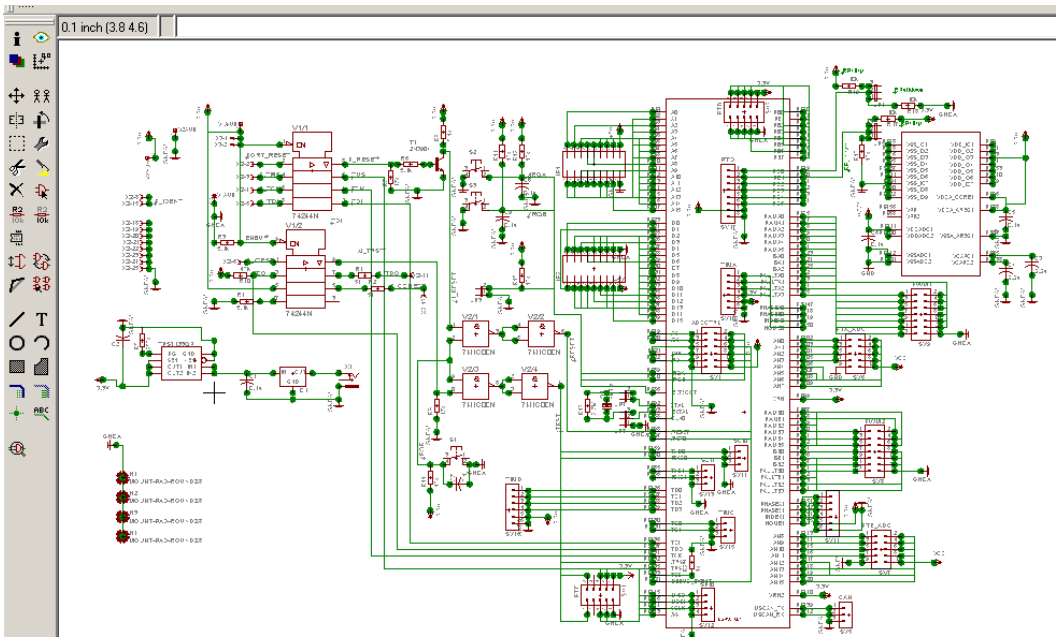


Figura E16. Acople entre el DSP y empaquetado tipo DIP.

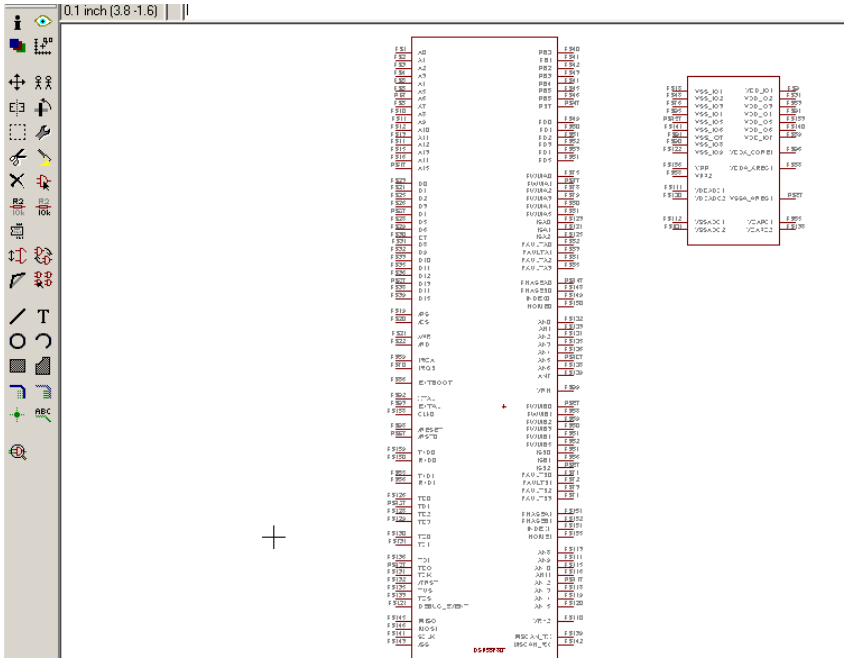


Figura E17. Circuito de acople del DSP56F807 de 160 pines LQFP a tipo DIP.

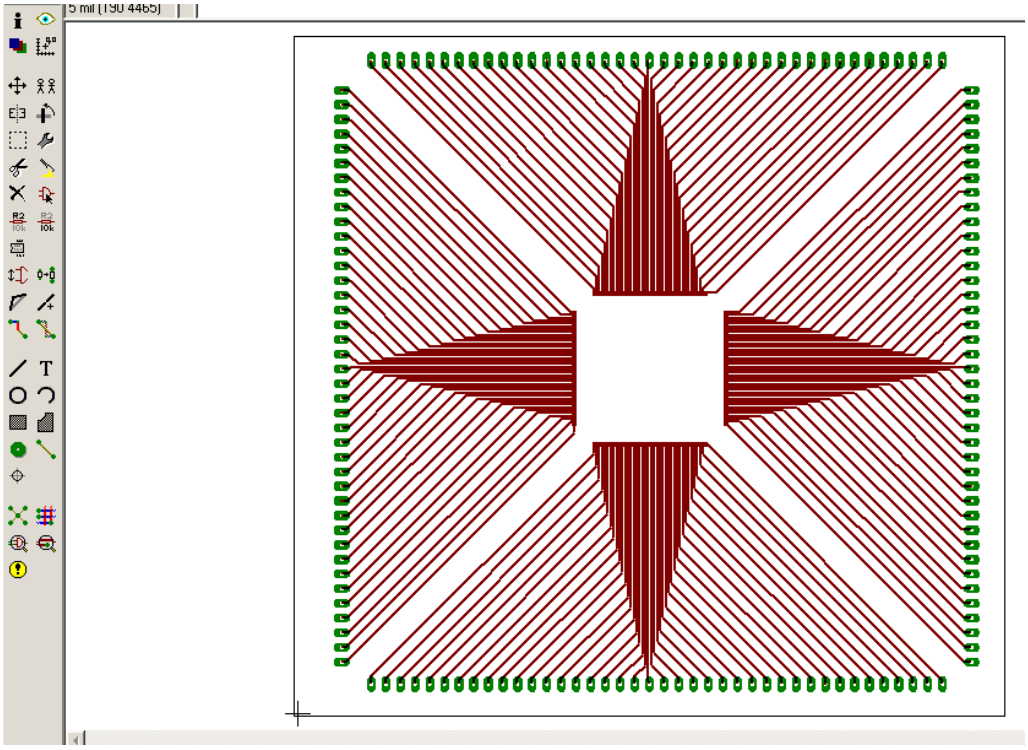


Figura E18. Tarjeta de desarrollo DSP56F807 vista inferior.

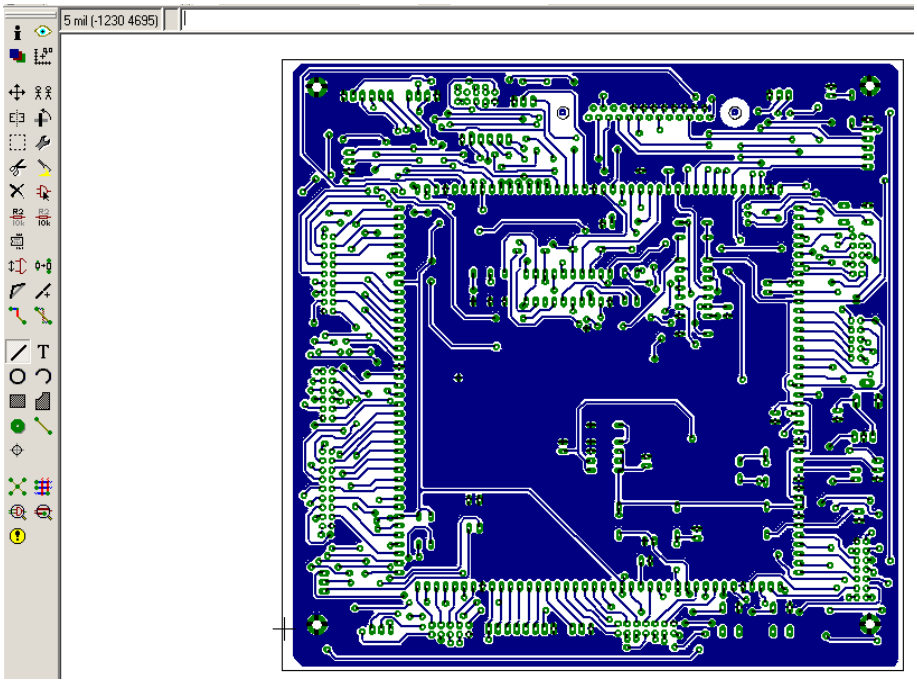


Figura E19. Tarjeta de desarrollo DSP56F807 vista superior.

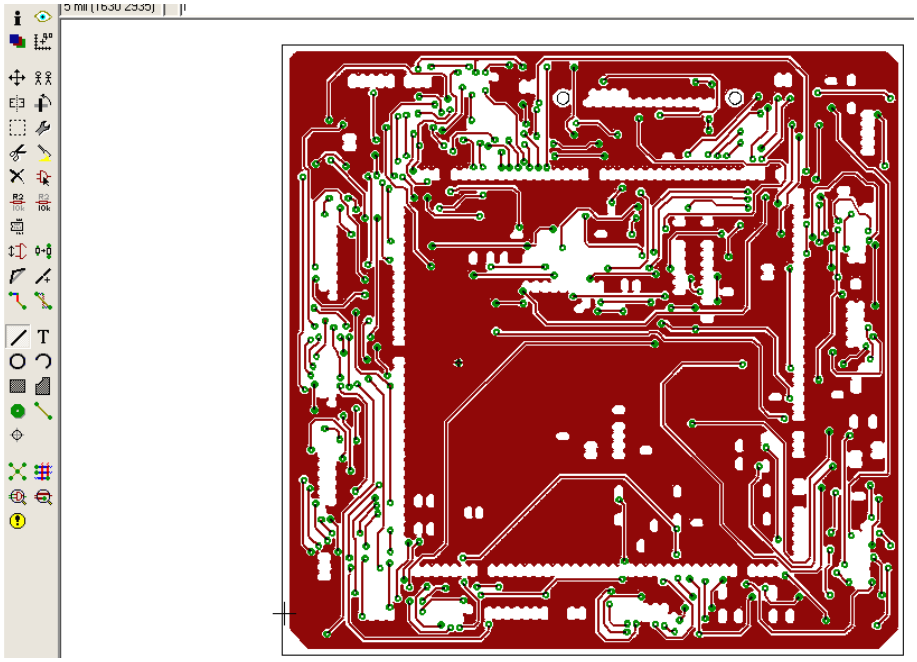
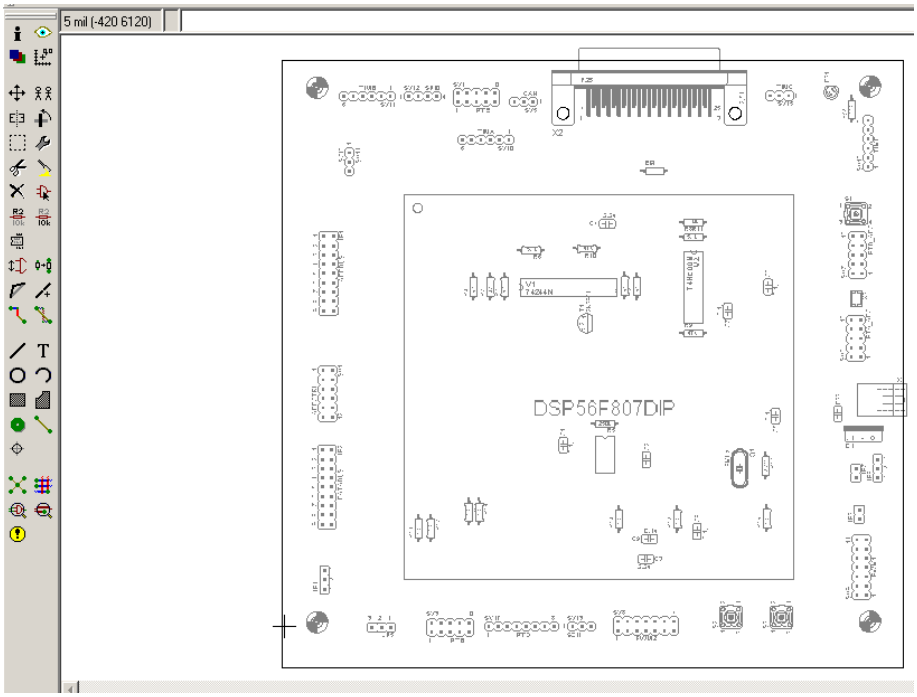
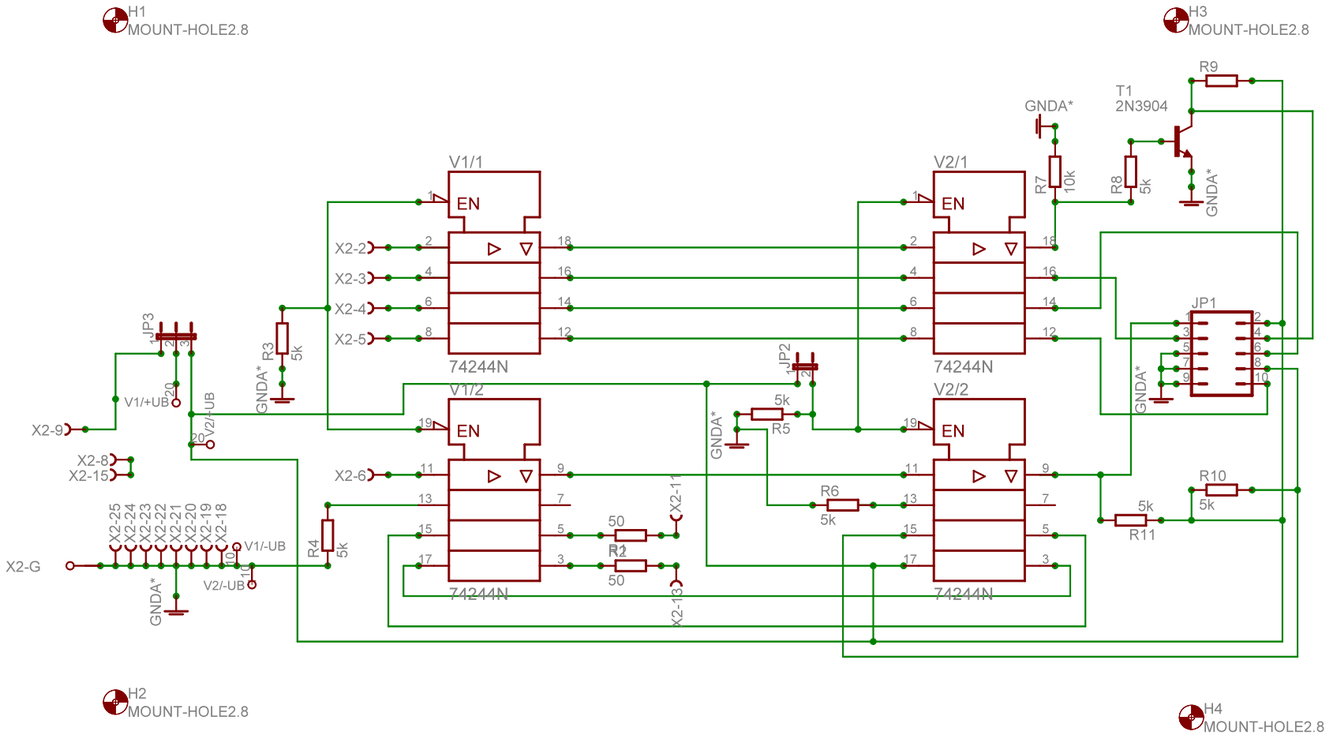
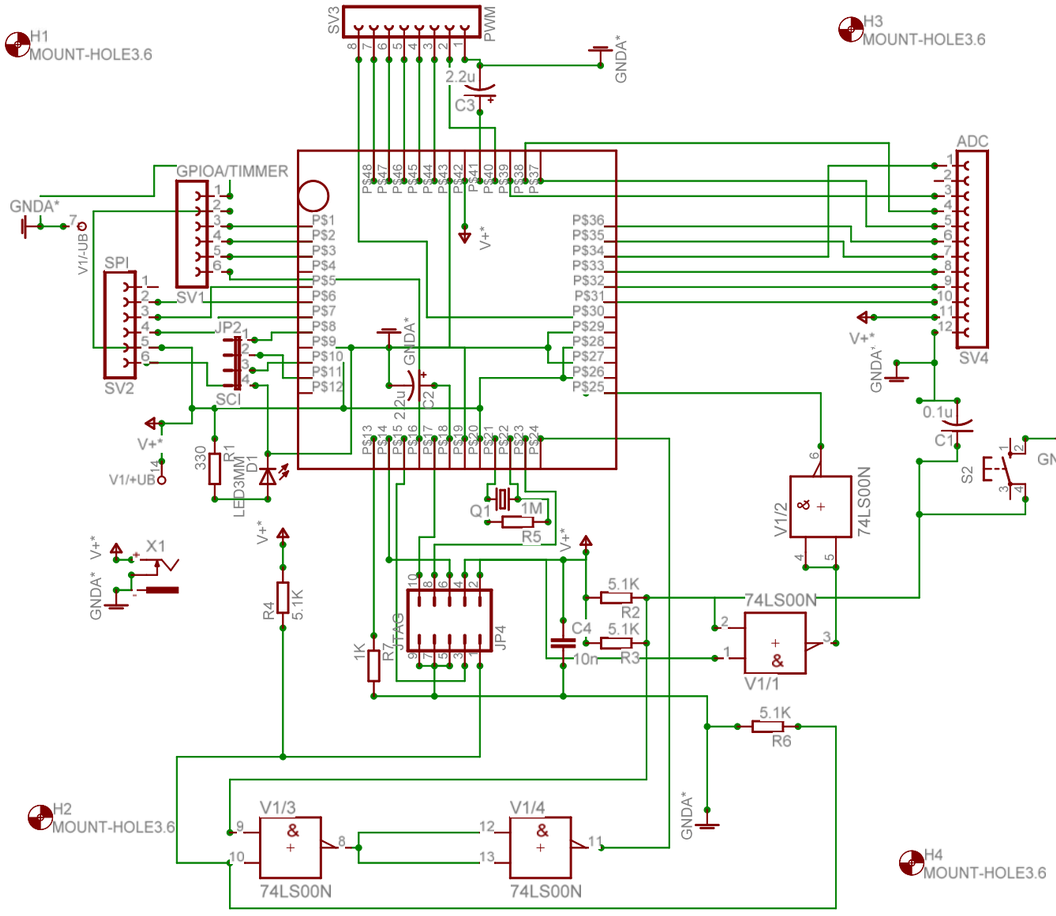


Figura E20. Tarjeta de desarrollo DSP56F807 vista de componentes.

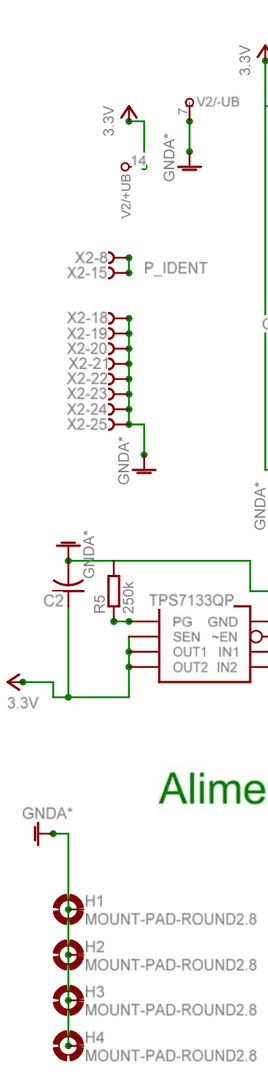


Se recomienda para más claridad de las figuras remitirse al anexo F.


H1
MOUNT-HOLE2.8
H3
MOUNT-HOLE2.8
V1/1
V2/1
V1/2
V2/2
EN
74244N
JP3
JP2
JP1
R3
5k
R4
5k
R5
5k
R6
5k
R7
10k
R8
5k
R9
T1
2N3904
R10
5k
R11
5k
R1
50
R2
50
GNDA*
V1/+UB
V2/+UB
V1/-UB
V2/-UB
X2-9
X2-8
X2-15
X2-6
X2-G
X2-25
X2-24
X2-23
X2-22
X2-21
X2-20
X2-19
X2-18
X2-11
X2-13
H2
MOUNT-HOLE2.8
H4
MOUNT-HOLE2.8


H1
MOUNT-HOLE3.6
H3
MOUNT-HOLE3.6
SV3
PWM
GNDA*
2.2u
C3
GPIOA/TIMMER
SV1
SPI
SV2
JP2
SCL
V+*
V1/+UB
330
R1
LED3MM
D1
V1/-UB
GNDA*
P$1
P$2
P$3
P$4
P$5
P$6
P$7
P$8
P$9
P$10
P$11
P$12
P$13
P$14
P$15
P$16
P$17
P$18
P$19
P$20
P$21
P$22
P$23
P$24
P$25
P$26
P$27
P$28
P$29
P$30
P$31
P$32
P$33
P$34
P$35
P$36
P$37
P$38
P$39
P$40
P$41
P$42
P$43
P$44
P$45
P$46
P$47
P$48
ADC
SV4
0.1u
C1
S2
Q1
1M
R5
X1
R4
5.1K
1K
R7
JTAG
JP4
C4
10n
5.1K
R2
5.1K
R3
74LS00N
V1/1
V1/2
5.1K
R6
H2
MOUNT-HOLE3.6
V1/3
V1/4
74LS00N
74LS00N
H4
MOUNT-HOLE3.6

Alimentación



Alimentación

H1 MOUNT-PAD-ROUND2.8
H2 MOUNT-PAD-ROUND2.8
H3 MOUNT-PAD-ROUND2.8
H4 MOUNT-PAD-ROUND2.8

Filtro

Pantalla

DSP56F807