ELABORACIÓN DEL SOFTWARE PARA LA CARACTERIZACIÓN DE UNA CELDA ELECTROQUIMICA UTILIZANDO DSP FAMILIA 56800 DE MOTOROLA

JEAN PIERRE AMARIS DOMINGUEZ
JOSE ALBERTO LOPEZ PATIÑO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECANICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA
2004

ELABORACIÓN DEL SOFTWARE PARA LA CARACTERIZACIÓN DE UNA CELDA ELECTROQUIMICA UTILIZANDO DSP FAMILIA 56800 DE MOTOROLA

JEAN PIERRE AMARIS DOMINGUEZ
JOSE ALBERTO LOPEZ PATIÑO

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

DIRECTOR:
JAIME GUILLERMO BARRERO PEREZ
Magíster en potencia Eléctrica (MPE)

CODIRECTOR:
JOSE ALEJANDRO AMAYA
Ingeniero Electricista

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECANICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA
2004

RESUMEN

TÍTULO: ELABORACIÓN DEL SOFTWARE PARA LA CARACTERIZACIÓN DE UNA CELDA ELECTROQUÍMICA UTILIZANDO DSP FAMILIA 56800 DE MOTOROLA *

AUTORES: AMARIS DOMÍNGUEZ, JEAN PIERRE, y, LÓPEZ PATIÑO, JOSÉ ALBERTO **

PALABRAS CLAVES: Celda electroquímica, medidor de impedancia electroquímica (MIE), DSP, 56F801, regresión circular, regresión senoidal, Metrowerks Codewarrior, pantalla de cristal líquido.

DESCRIPCIÓN:
En este documento se explica cómo se implementaron los algoritmos de programación para calcular los parámetros de una celda electroquímica. El desarrollo del código fuente del proyecto se realizó bajo la plataforma Metrowerks Codewarrior versión 6.0, a través del DSP 56F801 de Motorola y consta de tres etapas: Digitalización de la señal, procesamiento y visualización de los resultados.

Inicialmente se describe el DSP utilizado para la implementación del prototipo. Por lo tanto, se describe en forma general el 56F801, la memoria y los periféricos del DSP utilizados en la implementación.

En el Capítulo 2, se describe el hardware del proyecto, el circuito de alimentación de la pantalla de cristal líquido, el circuito de alimentación del proyecto, los circuitos de atenuación y amplificación de las señales provenientes de la celda electroquímica, el registro de desplazamiento, el acople del DSP 56F801 con el medidor de impedancia electroquímica. En el Capítulo 3 de software, se describen las funciones y rutinas utilizadas para la programación del DSP, la pantalla de cristal líquido y, por último, se describen los algoritmos de la regresión senoidal, regresión circular y se da el diagrama de flujo del programa. Finalmente, en el Capítulo 4, se describen y explican las pruebas realizadas al prototipo medidor de impedancia electroquímica. Se presentan los resultados obtenidos gráficamente, como el de la pantalla de cristal líquido.

---

* Trabajo de Grado.
** Facultad de Ingeniería Físico-mecánicas. Ingeniería Electrónica. Jaime Guillermo Barrero Pérez.

SUMMARY

TITLE: ELABORATION OF THE SOFTWARE FOR THE CHARACTERIZATION OF AN ELECTROCHEMICAL CELL USING DSP FAMILY 56800 DE MOTOROLA *

AUTHORS: AMARIS DOMÍNGUEZ, JEAN PIERRE, AND, LÓPEZ PATIÑO, JOSÉ ALBERTO **

KEY WORDS: Electrochemical cell, meter of electrochemical impedance (MIE), DSP, 56F801, circular regression, sine regression, Metrowerks Codewarrior, liquid crystal display.

DESCRIPTION:
In this document it is explained how were implemented the programming algorithms to calculate the parameters of an electrochemical cell. The development of the code source of the project was carried out under the platform Metrowerks Codewarrior version 6.0, through the DSP 56F801 of Motorola and it consists of three stages: Digitalization of the sign, prosecution and visualization of the results.

Initially is described the DSP used for the implementation of the prototype. Therefore, it is described in general form the 56F801, the memory and the peripherals of the DSP used in the implementation.

In the Chapter 2, is described the hardware of the project, the circuit of feeding of the liquid crystal display, the circuit of feeding of the project, the attenuation and amplification circuits of the signs coming from the electrochemical cell, the displacement registration, the couples of the DSP 56F801 with the meter of electrochemical impedance. In the Chapter 3 of software, are described the functions and routines used for the programming of the DSP, the screen of liquid glass and, lastly, are described the algorithms of the senoidal regression, circular regression and is given the diagram of flow of the program. Finally, in the Chapter 4, are described and explain the tests carried out to the prototype meter of electrochemical impedance. The results are presented obtained graphically, as that of the liquid crystal display.

---

* Work of Grade.
** Faculty of Engineering Physical-mechanics. Engineering Electronic. Jaime Guillermo Barrero Pérez

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

INTRODUCCIÓN

Los rápidos avances alcanzados por la electrónica, en especial en las técnicas de fabricación de circuitos integrados, han tenido, y sin duda continuaran teniendo gran importancia e impacto en la industria y la sociedad, estos avances han hecho posible la fabricación de I.C altamente sofisticados capaces de realizar funciones y tareas del procesado de señales digitales que normalmente eran demasiado difíciles o caras con circuitería o sistemas de procesado de señales analógicas. De aquí que muchas de las tareas del procesado de señales que convencionalmente se realizaban analógicamente se realicen hoy mediante hardware digital, con un menor costo y con una mayor precisión.

Este proyecto, hace parte de una investigación en Maestría Electrónica y presenta la implementación de un sistema autónomo; que con la lectura de voltaje y corriente, previamente amplificados, provenientes de una celda Electroquímica, encuentra los parámetros característicos de ésta: Rp, Rs y Cp; para así continuar fortaleciendo trabajos de investigación que en el área de corrosión de la Escuela de Ingeniería Metalúrgica se vienen adelantando.

En el capitulo 1 se hace una breve descripción a los DSPs familia Motorola, especialmente el DSP56F801, su memoria y los periféricos de este DSP utilizados en la implementación del prototipo para la caracterización la celda electroquímica.

El capitulo 2 detalla el Hardware utilizado para la implementación del prototipo. Por lo tanto, Fuente de alimentación, circuito atenuador, circuito amplificador, el registro para el corrimiento y por ultimo una descripción del acople con el medidor de impedancia electroquímica.

En el capitulo 3 se explican los algoritmos utilizados para la inicialización de la pantalla, escribir texto y graficar, configuración del ADC, algoritmo de la regresión circular.

Por ultimo en el capitulo 4 se describen las pruebas realizadas al circuito equivalente de la celda de Randles.

1. PROCESAMIENTO DIGITAL DE SEÑALES

1.1 DEFINICION

Una señal es definida como cualquier cantidad física que varia en el tiempo y que lleva información, generalmente acerca del estado o comportamiento de un sistema, como por ejemplo: radar, música, voz, sonar, etc.

Ahora bien procesar una señal se entiende como la operación o transformación sobre la señal. El procesamiento digital de señal (dsp) es una operación o transformación de una señal en un hardware digital según reglas bien definidas las cuales son introducidas al hardware a través de un software específico que puede o no manejar lenguajes tanto de alto como de bajo nivel.

En estricto rigor, digital ´´signal processing´´ se refiere al procesamiento electrónico de señales tales como sonido, radio, microondas, tensión y corriente de una celda electroquímica, usando técnicas matemáticas para realizar transformaciones o extraer información.

1.2 UN DSP PARA CADA APLICACIÓN

Una forma de clasificar los DSP's y aplicaciones es a través de su rango dinámico. El rango dinámico es un conjunto de números, desde pequeños a grandes, que deben ser procesados en el curso de una aplicación. Por ejemplo, para representar una forma de onda entera de una señal particular es necesario un cierto rango de números para manejar sus valores mayores y menores. El DSP debe ser capaz de manejar los números generados tanto en la transformación análoga – digital como durante los cálculos (multiplicaciones, sumas, divisiones) con dicha señal. Si no es capaz de manejar todo el rango de números ocurrirá "overflow" o "underflow", lo cual producirá errores en los cálculos.

1.3 CRITERIO DE SELECCIÓN DE DSP'S

Actualmente el mercado se ha ampliado enormemente en cuanto a la oferta de DSP's.
Existen diversos fabricantes, cada uno con un tipo especial y particular de arquitectura, uso y/o aplicación. Entre los más conocidos se destacan:

- Motorola (http://www.motorola.com/semiconductors)

Familias 56300 56800 56800E MSC8100 (StarCore)

- Texas Instruments (http://dspvillage.ti.com/docs/dsppproducthome.jhtml)

Familias TMS320C6000 TMS320C5000 TMS320C2000

- Analog Devices (http://www.analog.com/technology/dsp/index.html)

Familias Blackfin Familia Sharc TigerSharc ADSP-21xx
El uso de un DSP está justificado cuando se tienen los siguientes requerimientos:

- Ahorro de dinero.
- Tamaño más reducido.
- Bajo consumo de potencia.
- Procesamiento de varias señales de “alta” frecuencia en tiempo real.

Para este proyecto se escogió trabajar con un dsp del fabricante Motorola debido a que ofrece una gran variedad de DSP´s, los cuales están organizados en varias familias, cada una de ellas destinada a aplicaciones más concretas. Un factor mas relevante fue el precio y el gran soporte que ofrece Motorola gratuitamente como el software para cada DSP, Manuales y mejor desempeño en cuanto a funcionalidad.

1.4 CARACTERÍSTICAS DE LOS DSP`S

Una de las más importantes características de un DSP es su capacidad de realizar operaciones de multiplicación y acumulación (MACs) en sólo un ciclo de reloj. No obstante ello, es necesario que el dispositivo posea la característica de manejar aplicaciones críticas en tiempo real. Esto requiere de una arquitectura que soporte un flujo de datos a alta velocidad hacia y desde la unidad de cálculo y memoria. Esta ejecución a menudo requiere el uso de unidades DMA (“Direct Memory Acess”) y generadores de direcciones duales (DAG's) que operan en paralelo con otras partes del chip. Los DGA's realizan los cálculos de direcciones, permitiendo al DSP buscar dos datos distintos para operar con ellos en un solo ciclo de reloj, de tal forma que es posible ejecutar algoritmos complejos en tiempo real.
Es importante para DSP's tener un mecanismo efectivo de salto para la ejecución de lazos ya que el código generalmente programado es altamente repetitivo. La arquitectura permite realizar estos lazos sin instrucciones adicionales ni demoras, las que al ejecutarse millones de veces empiezan a generar retardos significativos.

Los DSP's deben manejar rangos dinámicos extendidos y de precisión para evitar ´´overflow y underflow´´ y para minimizar los errores de redondeo. Para acomodarse a esta capacidad, los DSP's incluyen acumuladores dedicados con registros más anchos que el tamaño nominal de los datos para así conservar la precisión (por ejemplo, DSP's de 16 bits poseen acumuladores de 32 bits para manejar el resultado de las multiplicaciones). También deben soportar el manejo de “buffers” circulares para la ejecución de funciones algorítmicas, tales como filtros. En estos tipos de “buffers” el puntero se actualiza en paralelo con otras funciones del chip en cada ciclo de reloj. En cada ciclo el “buffer” circular realiza una comprobación de "fin de buffer" para verificar si es necesario volver al inicio de éste sin demorar así la ejecución

del algoritmo a causa de la ejecución de instrucciones adicionales de comparación y salto.

1.5 ARQUITECTURA HARVARD

En la arquitectura clásica de Von Neumann la ALU y la unidad de control están conectadas a una sólo unidad de memoria que almacena tanto instrucciones de programa como datos.
Durante la ejecución de un programa, una instrucción es leída desde la memoria y decodificada, los operandos necesarios son obtenidos desde la memoria, y, finalmente, la instrucción es ejecutada. La principal desventaja es que la memoria se transforma en el cuello de botella de esa arquitectura.

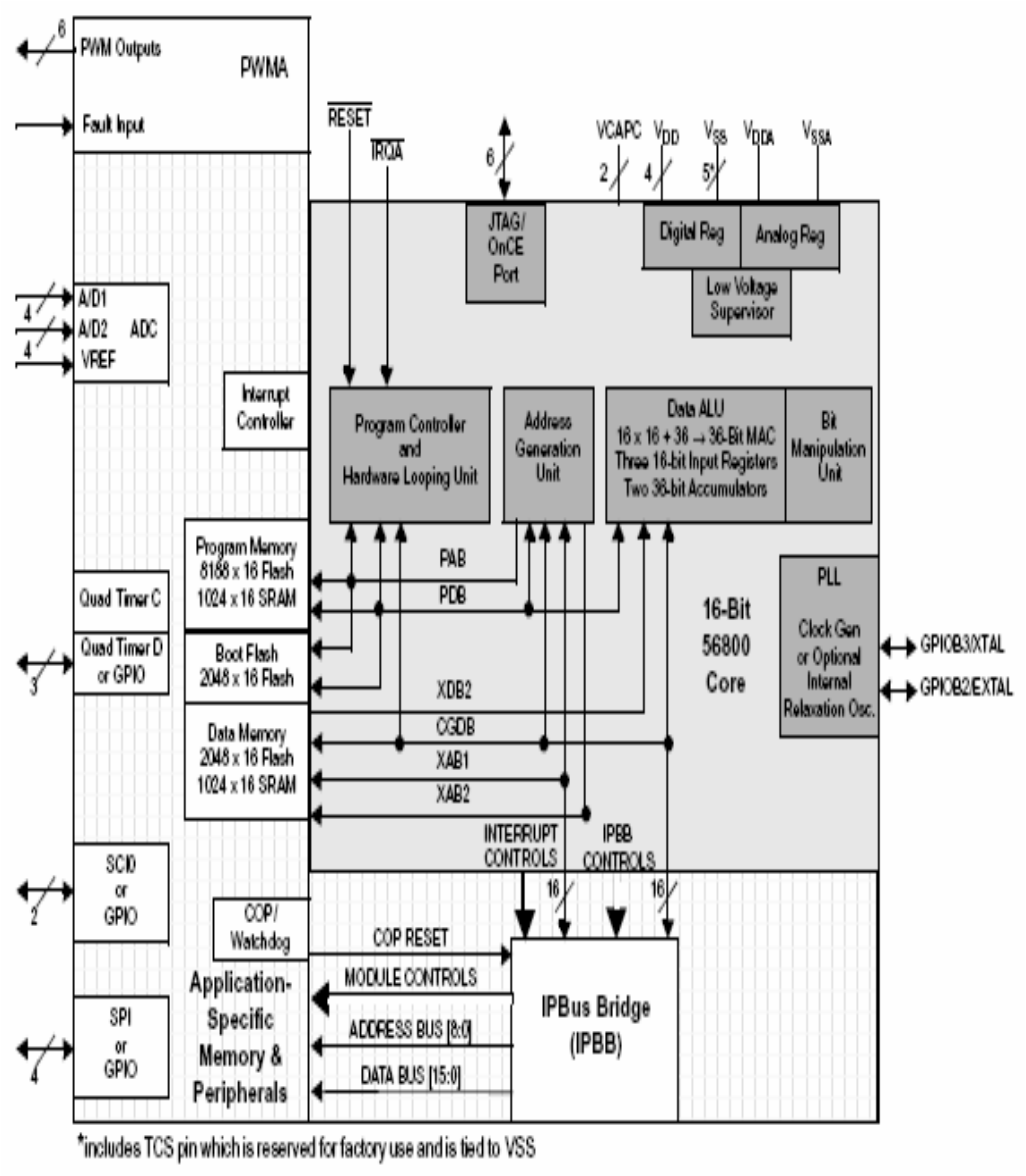
La arquitectura *Harvard* trata de aumentar el ancho de banda de los accesos aumentando el paralelismo de la memoria: Para ello se dispone de dos espacios de memoria independientes, uno para almacenar el código a ejecutar y el otro para los datos. Cada uno de estos espacios dispone de su propio grupo de buses, con lo cual es posible acceder simultáneamente a ambos espacios.

1.6 PROCESADOR DIGITAL DE SEÑALES 56F801 de MOTOROLA

El pilar fundamental tecnológico de este proyecto es el procesador digital de señales (*DSP*) 56F801 de *Motorola.*
El 56F801 es un *DSP* de 16-bit, con una arquitectura *´´Harvard dual´´* haciéndolo mas eficiente, de bajo consumo y es complementado con un gran rango de periféricos dentro del chip, una memoria de programa flash y una memoria *RAM* de acceso dual. Las características principales de este *DSP* son: un puerto de comunicación serial (SCI- 2 pines -), puerto de Interfaz serial de periférico (*SPI – 4 pines*), modulación de ancho de pulso (PWM) de 6 canales, puerto *JTAG* y posibilidad de configuración de 11 pines dedicados

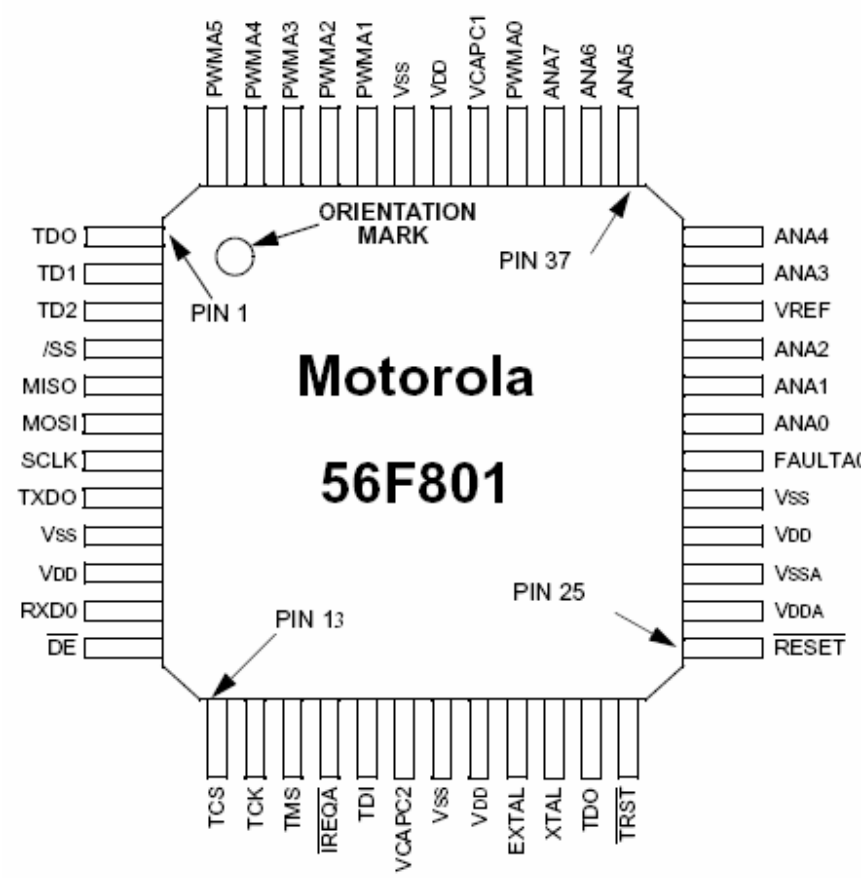
a pines de propósito general (GPIO). A continuación se presentan estas características en las figuras 1 y 2.

Figura 1. Diagrama de Bloques de la Arquitectura de la CPU – 56F8001



Las anteriores características hacen del 56F801 le permitan desempeñarse en una gran variedad de aplicaciones de control y de procesamiento de señales con un alto rendimiento y confiabilidad.

Figura 2 DSP 56F801



La tabla 1 presenta la asignación de pines utilizados del DSP 56F801 como de propósito general, con su respectiva función en este proyecto. En el capitulo de software se explican con mas detalle.

Tabla 1 Pines Configurados como GPIO

	Pin dedicado a:	**GPIO**	**Configurado para**	**I/O**
1	Temporizador, TD0	GPIOA0	Pulsador 1 - Pantalla	I
1	Temporizador, TD1	GPIOA1	Pulsador 2 - Pantalla	I
1	Temporizador, TD2	GPIOA2	Pulsador 3 - Pantalla	I
1	Transmisión de datos serial, TXD0	GPIOB0	Enter- Esclavo MIE	O
1	Recepción de datos serial, RXD0	GPIOB1	Seguir – Esclavo MIE	O
1	Entrada oscilador externo de cristal, EXTAL	GPIOB2	Resistencia Shunt –MIE	I
1	Salida de oscilador de cristal, XTAL	GPIOB3	Ganancia Corriente - MIE	I
1	Reloj Serial del SPI, SCLK	GPIOB4	Dedicado a SPI, SCLK	O
1	Salida maestro/entrada esclavo SPI, MOSI	GPIOB5	Dedicado a SPI, MOSI	O
1	Entrada maestro/salida esclavo SPI, MISO	GPIOB6	Escritura-pantalla	O
1	Selección de esclavo SPI, $\overline{SS}$	GPIOB7	Comando-pantalla	O

1.6.1 INTRODUCCIÓN AL DSP 56F801

Este procesador pertenece a la familia de procesadores 56F800 de *Motorola*, el cual en un chip, combina el poder de procesamiento de un DSP y la funcionalidad de un microcontrolador, que con el conjunto de periféricos lo hacen una alternativa económica a las unidades de microcontroladores tradicionales.

El 56F801 posee una gran velocidad de la unidad de procesamiento central (*CPU*) (40 millones de instrucciones por segundo) 40 MIPS, posee una

arquitectura basada en palabras de 16-bit con registros de 32-bit para almacenamiento de resultados intermedios.

Los periféricos del 56F801 utilizados en este proyecto son:

- El conversor análogo-digital dual (*ADC*)
- La Interfase de Periférico Serial (*SPI*), configuración maestro.
- 9 pines configurados para funcionar como pines de propósito general:
    - 2 pines de interfase de comunicación serial(SCI)
    - 2 pines de conexión de oscilador externo.
    - 3 pines de “TIMER”
    - 2 pines de SPI, de configuración esclavo.

A continuación se describen los periféricos utilizados y los registros del DSP que permiten controlan su funcionamiento.

1.6.1.1 CONVERSOR ANÁLOGO-DIGITAL (A/D)

Este periférico es de gran relevancia debido a que por él se tomaron todas las muestras para los cálculos a desarrollar y los resultados a visualizar en la pantalla de cristal líquido. El 56F801 tiene un conversor análogo-digital dual de 12 bit, cada uno con cuatro canales, llamado ADCA.

Para comenzar se describirán las características que hacen parte del conversor A/D:

- Frecuencia máxima de reloj es de 5MHz con un periodo de 200ns.
- Tiempo de conversión máxima 1.7μS, frecuencia de 588.23 kHz.
- Configuración para adquirir datos en forma simultanea por dos canales, ANA1 y ANA5
- Voltaje de referencia para la conversión 3 V D.C.

Registros para su configuración:

- El Registro de Control ADC1 (ADCR1) se utilizo en general para configurar el tipo de muestreo (referencial y simultáneo), los pulsos de inicio y detención, y para deshabilitar todas las interrupciones.

La figura 3 Registro de control ACDR1

ADC Control Register 1 (ADCR1) ADC_BASE+$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	STOP	0	SYNC	EOSIE	ZCIE	LLMTIE	HLMTIE	CHNCFG[3:0]				0	SMODE[2:0]		
	Write			START													
	Reset	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1

- Registro de Control ADC2 (ADCR2) se utiliza para ajustar el valor del bus (reloj) interno (5MHz máximo) del periférico ADC.

Figura 4 Registro de control ADCR2

ADC Control Register 2 (ADCR2) ADC_BASE+$1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	DIV[3:0]			
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

- El registro de deshabilitar muestra (ADSDIS), tiene un bit por cada canal del ADC, que sirve para habilitar o deshabilitar su conversión en el proceso de muestreo del ADC, para este trabajo se habilitan los canales ANA0 y ANA4, por tener la posibilidad de adquirir sus muestras en modo simultaneo.

Figura 5 Registro deshabilitar muestra ADSDIS

ADC Sample Disable Register (ADSDIS) ADC_BASE+$5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	TEST[1:0]		0	0	0	0	0	0	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1.6.1.2 INTERFAZ DE PERIFÉRICO SERIAL (*SPI*)

El modulo de interfaz de Periférico serial (*SPI*), permite comunicación *full duplex*, sincronía y comunicación serial entre controladores híbridos y dispositivos periféricos. Sus principales características son:

- Tiene dos modos de operación Maestro y esclavo.
- Operación *Full-Duplex*.
- Transmisiones de longitud programables de 2 a 16 bit.
- De sus cuatro pines dos manejan la comunicación periférica como maestro (SCLK Y MOSI), y los otros dos (MISO Y $\overline{SS}$), que se utilizan para la configuración esclavo, son programados como pines de propósito general.

En el capitulo de software se presenta la configuración de sus registros.

1.6.1.3 MEMORIA DE DATOS

La memoria de datos del DSP 56F801, esta dividida en memoria de datos de RAM y FLASH, la primera tiene 1K palabras y la segunda 2K palabras.

Figura 6 Diagrama de Bloques de la Memoria de Datos

Begin/ End Address	801/802
0000 03FF	XRAM 1K
0400 07FF	Reserved
0800 0BFF	
0C00 0FFF	Peripherals
1000 13FF	XFlash 2K
1400 17FF	
1800 1FFF	Reserved
2000 2FFF	
3000 3FFF	
4000 FF7F	
FF80 FFFF	Core Regs 128

.

1.6.1.4 MEMORIA DE PROGRAMA

La memoria de programa del DSp 56F801 tiene 8K palabras de FLASH de programa *on chip* y 1K palabras de RAM de programa *on-chip.*

La figura 4 muestra la distribución de la memoria de programa para la RAM de programa y la FLASH de programa.

La memoria de programa es donde se almacena el código fuente, operandos inmediatos y vectores de información.

Figura 7 Diagrama de bloques de la memoria de Programa.

Begin/ End Address	801/802
0000 0003	BFlash 4
0004 1FFF	PFlash 8K-4
2000 6FFF	Reserved
7000 73FF	
7400 77FF	
7800 7BFF	
7C00 7DFF	PRAM 1K
7E00 7FFF	
8000 87FF	BFlash 2K
8800 EFFF	Reserved
F000 F3FF	
F400 F7FF	
F800 FFFF	

2. HARDWARE

En esta sección se describen los dispositivos electrónicos implementados en el sistema para caracterizar una celda electroquímica. El sistema consta de una tarjeta principal, donde es ubicado el DSP, una tarjeta que comunica el DSP con el computador, la tarjeta de alimentación en donde además se encuentran los conectores que comunican con el DSP y la pantalla de cristal liquido. Los circuitos de atenuación y amplificación.

También en esta sección se describe el circuito de registro de entrada serie y salida paralela que maneja la pantalla de cristal liquido para su funcionamiento y el circuito para el contraste de la pantalla de cristal liquido.

Por último se describe el conector del Medidor de Impedancia Electroquímica con el DSP.

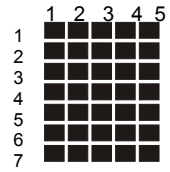
2.1 PANTALLA DE CRISTAL LÍQUIDO

Se selecciono un LCD 256x128 (256 líneas de 128 caracteres por línea) donde se van a visualizar los parámetros de una celda electroquímica, el conjunto de mensajes y las opciones de menú programadas a través del dsp.

La pantalla de cristal líquido utiliza como pines de entrada de datos, 8 pines (DB0 a DB7) y a través de un ponteciometro se gradúa su contraste.

La memoria para texto en la LCD va desde 0x0000 hasta 0x0FFF (4095 bytes), y la memoria para gráficos desde 0x1000 hasta 0x1FFF (4095 bytes).

La pantalla de cristal líquido que se utiliza en el sistema es de marca Hyundai de referencia HG25504NG-01 y pertenece a la gama de pantallas graficas que trabaja con base en matrices de 5x7 puntos:



Matriz de Puntos 5x7

La LCD posee internamente grabados 160 caracteres de 5x7 píxeles ya determinados, los cuales se pueden tener acceso por medio del bus de datos colocando el código ASCII del correspondiente carácter. Para mas información acerca de la LCD gráfica ver anexo F.

2.1.1 CIRCUITO DE ALIMENTACION DE LA LCD

El circuito electrónico de la figura 8 muestra las conexiones de alimentación de la pantalla de cristal líquido. Esta trabaja con una alimentación de 5V C.D., y otro voltaje de alimentación de -25 voltios para el ajuste del contraste, este se gradúa con un potenciómetro de 10KΩ.

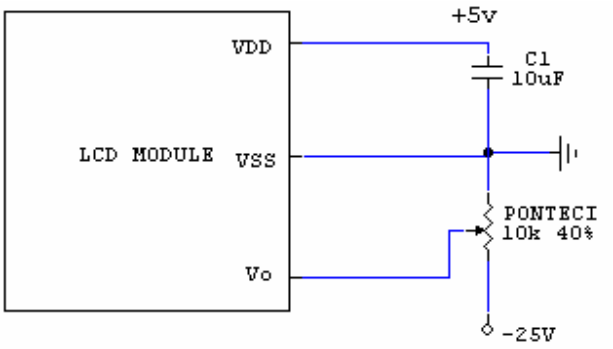


Figura 8 Circuito de alimentación

2.1.2 Especificaciones Físicas

Tabla 2 Características de la LCD HG25504-NG de Hyundai

Ítem	Especificación	Unidad
Tamaño Lcd (W X H X T)	147.0 x 116.0 x 12.0	mm
Área de despliegue(W X H)	127.0 x 70.0	mm
Tamaño de punto (W X H)	0.43 x 0.43	mm
Tamaño de puntos (W X H)	0.47 x 0.47	mm
Peso	155	g

Fuente: Hoja de datos de la LCD.

2.1.3 Asignación de Pines de interfase

En el tabla 3 se identifica los pines de la pantalla de cristal líquido, con su respectiva función. Para mas información favor dirigirse al anexo F.

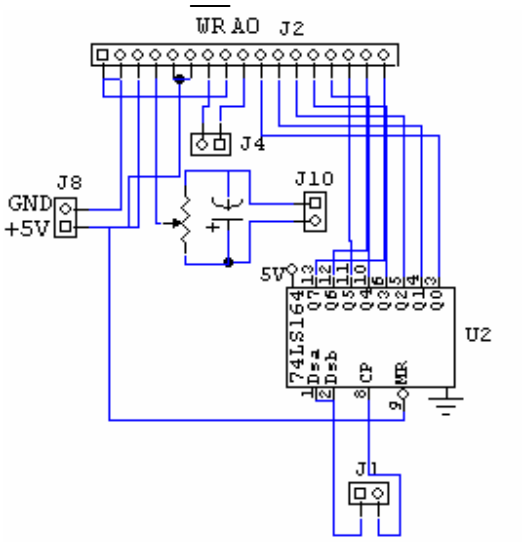
Tabla 3 Pines de la LCD

Pin No.	Símbolo	Nivel	Función
1	FG	0v	Tierra de carcasa
2	Vss(Gnd)	0v	Tierra
3	V_{DD}(Vcc)	+5v	Fuente de Voltaje para la lógica y LCD
4	Vo	-25	Voltaje variable para el contraste de la LCD
5	/RES	H/L	Señal de Reset
6	/RD	H/L	Señal de Lectura
7	/WR	H/L	Señal de Escritura-DSP
8	/CS	H/L	Señal Chip Select
9	A0	H/L	Señal para seleccionar el tipo de datos-DSP
10	DB0	H/L	Dato bit 0
11	DB1	H/L	Dato bit 1
12	DB2	H/L	Dato bit 2
13	DB3	H/L	Dato bit 3
14	DB4	H/L	Dato bit 4
15	DB5	H/L	Dato bit 5
16	DB6	H/L	Dato bit 6
17	DB7	H/L	Dato bit 7

Fuente: Hoja de datos de la LCD.

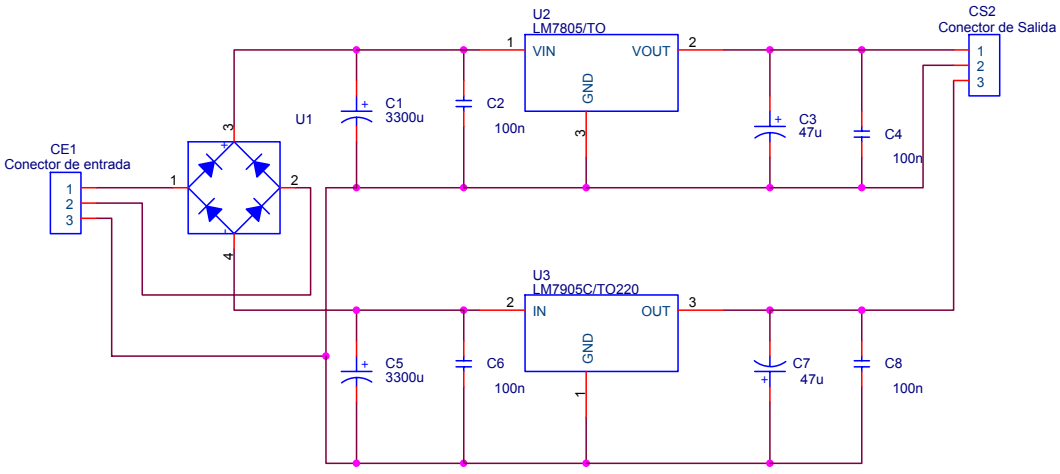
La figura 9 muestra las conexiones realizadas a la pantalla de cristal líquido, en donde el conector J2 reúne los 17 pines que posee la pantalla para su funcionamiento. Del pin 10 al 17 son pines de datos, estos se conectan al I.C de registro de desplazamiento, de entrada serie y salida paralela de referencia 74LS164, el conector J1 tiene las entradas del 74LS164 del ´´clear´´ y el ´´clock´´ que van al DSP. Por ultimo en el conector J8 esta la tierra y +5V D.C y en el conector J10 el conector de contraste.

Figura 9 Conexiones de los pines de la LCD



2.1.4 CIRCUITO DE ALIMENTACION

Figura 10 Fuente Dual ±5[V]



La Figura 10 muestra el diagrama circuital de la fuente dual de ±5[V] utilizada en el proyecto. En el conector de entrada (CE1) se suministran las señales de un trasformador 120[V] / 26[V] con tap central. Las dos señales de A.C.

de 26[V] se conectan a las entradas 1 y 2 de CE1 mientras que la tierra proveniente del tap central se conecta a la entrada 3 del mismo. Los circuitos integrados utilizados para la regulación de las señales de alimentación son el LM7805 para la señal de +5[V] y LM7905 para la señal de -5[V] junto con un arreglo de condensadores (3300μF, 100nF, 47μF y 100nF) para mejorar la estabilidad en las señales de alimentación de C.C y disminuir la tensión de rizado.

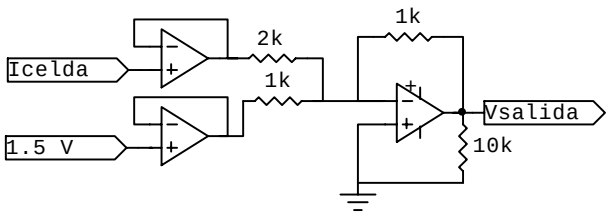
En el conector de salida (CS2) se obtienen señales rectificadas y reguladas de +5[V] en el pin 1, –5[V] en el pin 3 y tierra en el pin 2.

2.2 ACONDICIONAMIENTO DE SEÑALES

2.2.1 CIRCUITO ATENUADOR

Los conversores A/D del DSP toman muestras en un rango de 0 a 3V, siendo necesario realizar un circuito que acondicione la señal de salida de corriente de la celda de +/- 2.25V, a un rango de salida que va de 0.375V hasta 2.625V, a continuación se presenta el circuito utilizado para atenuar la señal de corriente de la celda. El voltaje de 1.5V lo suple el circuito de alimentación de la figura 10.

Figura 11 Circuito atenuador

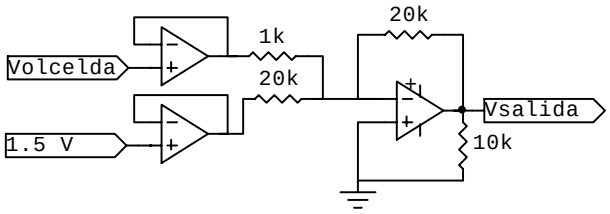


El OPA 4131PJ realiza la función del amplificador de atenuación. En la figura 11 se muestra como la corriente de salida de la celda pasa por un seguidor de tensión, luego se implementa la configuración sumador inversor donde la tensión 1 es la señal de Corriente de la celda y la tensión 2 es un nivel de offset de 1.5 V que se le suma a la señal. Para este caso se utilizo el OPA 4131PJ, este I.C posee un bajo nivel de ´´*offset*´´ (750uV máximo), un voltaje de ruido muy pequeño ideal 15 nV/$\sqrt{\ }$ Hz a 1 kHz, posee un producto de ganancia por ancho de 4Mhz, el cual cumple con el rango de frecuencias trabajadas. Se alimenta con una fuente dual de +/-5 V D.C, en donde el pin 4 se alimenta con +5V D.C y el pin 11 -5V D.C.

2.2.2 CIRCUITO AMPLIFICADOR

La señal de voltaje de salida de la celda es de +/- 30mV, siendo necesario amplificar esta señal con una ganancia de 20 v/v y sumarle un nivel de voltaje continuo de 1.5V para que el ADC muestree la señal de voltaje en el rango de 0 a 3V. El voltaje de salida del circuito amplificador va de 0.9V hasta 2.1V cumpliendo con el requerimiento del ADC. El circuito equivalente es el siguiente.

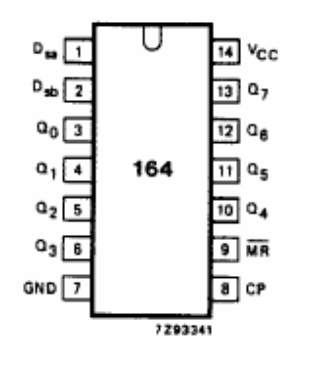
Figura 12 circuito amplificador



2.3 REGISTRO DE DESPLAZAMIENTO 74LS164.

Para la comunicación del DSP con la pantalla se hizo necesaria la implementación de un registro de desplazamiento de entrada serie y salida paralelo, debido a la limitación en número de pines de propósito general del DSP.

Figura 13 Registro de desplazamiento



Como se ilustra en la figura 9 los pines Q0 a Q7 son los pines que van conectados al la pantalla LCD, en la figura 13 se muestra el esquema en donde el pin 8 ´´CP´´ la entrada del reloj y la entrada de datos seriales los pines 1(Dsa) y 2(Dsb) van conectadas al SPI del DSP, pines SCLK y MOSI respectivamente, tal como aparece en la tabla 1.

2.4 MEDIDOR DE IMPEDANCIA ELECTROQUIMICA (MIE)

El Medidor de Impedancia Electroquímica (*MIE)** es un equipo que permite el desarrollo de pruebas para una celda Electroquímica usando un barrido en frecuencias con señales senoidales de AC para determinar sus parámetros característicos. Este prototipo entrega dos señales senoidales correspondientes a la tensión y corriente aplicada a la celda, cuyos valores estarán en el rango de 30mVpp para el voltaje y 2.25Vpp para la señal correspondiente a la corriente, estas señales tendrán un barrido en frecuencia desde 139mHz hasta 63kHz, con ocho frecuencias por década; para su operación se necesitara un pulso de inicio ("ENTER") y otro ("SEGUIR") que se encargara de hacer el cambio de frecuencias, adicionalmente se tomaran dos señales provenientes del PIC interno a este proyecto que permitirán conocer la amplificación a la cual fue sometida la corriente para ser entregada como señal de voltaje, cuyos valores dados por los realizadores de dicho proyecto son:

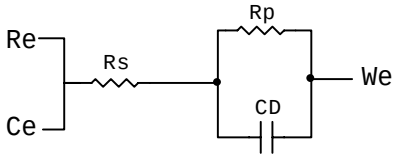
Tabla 4 Combinaciones de Ganancia

PIC pin #40 GANANCIA	PIC pin #26 RESISTENCIA DE SHUNT	RESISTENCIA EQUIVALENTE (Ω)	GANANCIA (V/V)	GANANCIA TOTAL
0	0	47// 18	100	37.22687
0	1	47//18	10	3726.872
1	0	18	100	1800
1	1	18	10	180

Estos valores de GANANCIA TOTAL permiten calcular el valor de la corriente a partir de la señal de voltaje que la esta caracterizando, simplemente dividiendo el valor del voltaje por la ganancia total.

El circuito equivalente de una celda electroquímica es:

Figura 14 Circuito Equivalente de la celda Electroquímica.



Fuente: SILVERMAN, D.C. Primer on the AC impedance Technique.

Donde,

Rs: Representa el efecto resistivo de la solución.

Rp: Representa la resistencia de polarización.

C_D: Representa el efecto capacitivo de la interfase de unión probeta solución.

Ecuación de Impedancia de la celda de Randles:

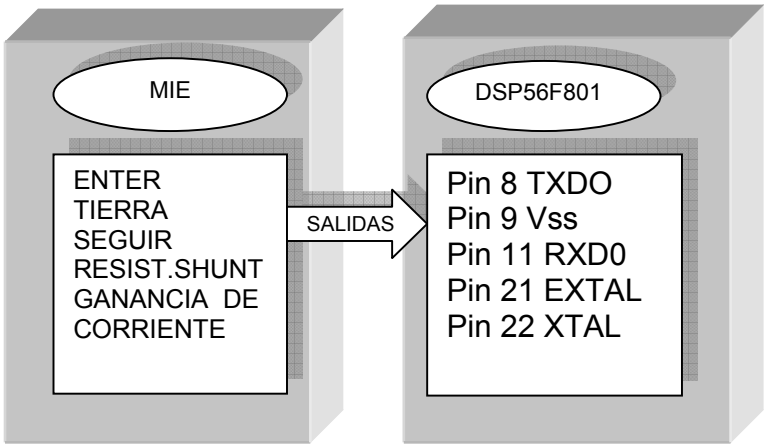
$$Z = R_S + \frac{R_P}{1+(\omega R_P C)^2} + j\left(\frac{-(\omega R_P^2 C)}{1+(\omega R_P C)^2}\right) = Z + jZ''$$ Ecuación 1

Para este proyecto se trabajo con el Medidor de impedancia Electroquímica en modo esclavo siendo el DSP el maestro. Los pines utilizados para tal propósito se muestran en la siguiente figura.

---

*Mayores detalles en la tesis de pregrado “MEDIDOR DE IMPEDANCIA ELECTROQUIMICA”, realizado JORGE HUMBERTO RODRÍGUEZ PACHECO y SERGIO ANDRÉS RUIZ GÓMEZ.

Figura 15 Acople del MIE y el DSP56F801



Los pines 21 EXTAL y 22 XTAL del DSP 56801, inicialmente eran los pines para el oscilador de relajación del DSP, debido a la falta de pines de propósito general (GPIO), se quito el oscilador y se habilitaron como GPIOs para controlar la ganancia de corriente de la celda y la resistencia de shunt.

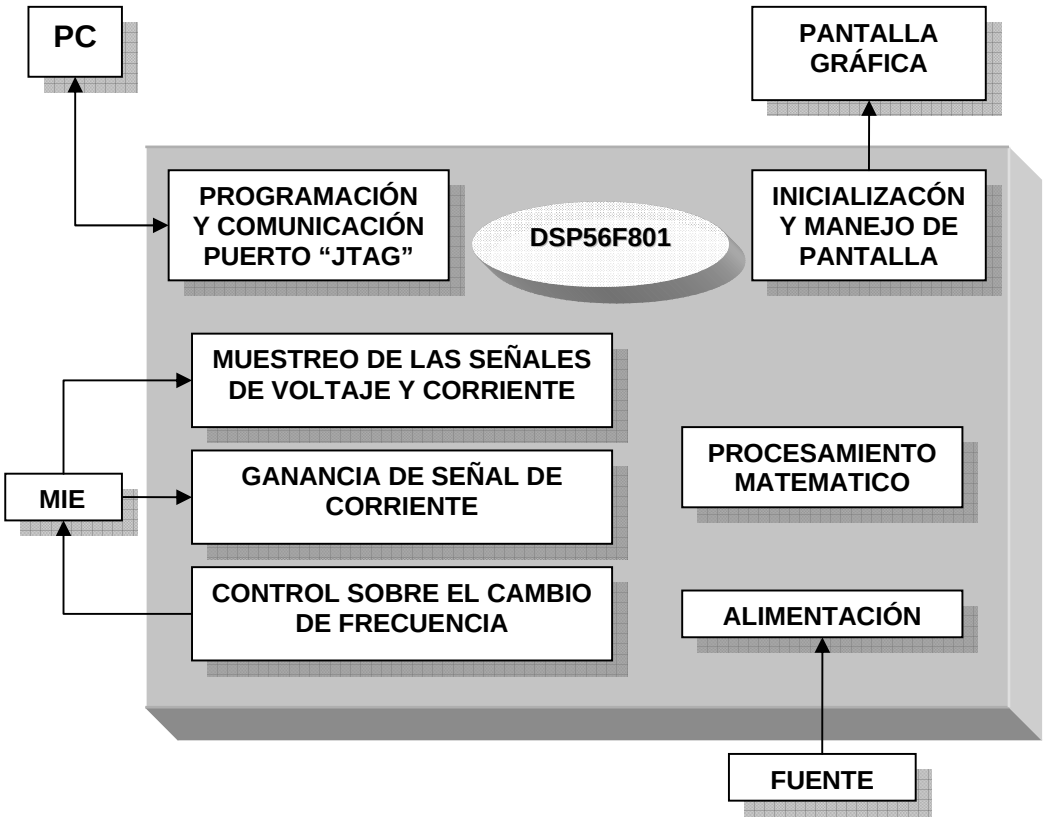
*Mayores detalles en la tesis de pregrado “MEDIDOR DE IMPEDANCIA ELECTROQUIMICA”, realizado JORGE HUMBERTO RODRÍGUEZ PACHECO y SERGIO ANDRÉS RUIZ GÓMEZ.

3. SOFTWARE.

La totalidad del software para el manejo del DSP se realizo en lenguaje de alto nivel C, bajo la plataforma de programación ***CodeWarrior™ Development Studio for Motorola™ DSP56800/E Hybrid Controllers, version 6.0***, que facilita la programación, ayudando al usuario con la asignación de memoria, su direccionamiento, interrupciones, registros, vectores, y demás conceptos estrictamente técnicos de su arquitectura

El programa general se divide en tres grandes bloques de trabajo: la inicialización y manejo de la pantalla, el muestreo de las señales provenientes del MIE y el procesamiento matemático de las señales adquiridas para encontrar los parámetros de la celda electroquímica; en la figura 15 se muestra el diagrama de bloques del proyecto, con toda las comunicaciones que necesita el DSP para el correcto funcionamiento del mismo; no se presenta el programa en general, solo se expondrán los criterios, para detalles específicos del programa ver Anexo D.

Figura 16. Diagrama de bloques del proyecto.



3.1 Configuración del DSP:

Antes de la operación del DSP para cualquier trabajo, se tiene que preparar o configurar: la frecuencia interna del BUS (PLL interno), pines de propósito general y otros periféricos que se tengan que utilizar como lo son la interfase serial de periféricos (SPI) y el conversor análogo-digital ADC.

3.1.1 Frecuencia Interna del DSP

La principal idea del DSP es adquirir mayor velocidad de operaciones, y con este DSP se ha logrado, pero en el momento de interactuar con otros periféricos mas lentos en operación y comunicación, como el registro de desplazamiento 74LS164, utilizado en este proyecto, se tiene que disminuir su velocidad de operación al mismo nivel de su periférico, esto se logro por

software, graduando la frecuencia interna del bus del DSP, con cálculos muy sencillos. Este proyecto utiliza dos configuraciones o valores de frecuencia para el bus del DSP, una primera de 14MHz que opera lo relacionado con la comunicación que se hace con la pantalla gráfica (registro de desplazamiento y SPI) y la segunda frecuencia que se configura es de 40MHz (la máxima posible), utilizada para las operaciones matemáticas, el tratamiento de señales y el muestreo por medio del ADC.

Para hacer esta configuración básicamente se manejan tres registros del PLL, estos son:

- Registro de control del PLL (PLLCR).
- Registro de División-Por del PLL (PLLDB).
- Registro de Estado del PLL (PLLSR).

3.1.1.1 Registro de control del PLL (PLLCR):

Con este registro se puede tener control general sobre la configuración del PLL, desde la habilitación de preescalador o postescalador hasta interrupciones, ver figura 16.

Figura 17. Registro de control del PLL, PLLCR.

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PLLIE1		PLLIE0		LOCIE	0	0	0	LCKON	CHPMPTRI	0	PLLPD	0	PRECS	ZSRC[1:0]	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Tabla 6 Configuración del Registro ZSRC

ZSRC[1:0]	**Fuente de reloj.**
01	Salida de Preescalador.

10	**Salida de Postescalador.**
00,11	Reservado.
PRECS	**Selección de reloj preescalador Aplicado únicamente a DSP56F801**
0	**Oscilador de relajación interno.**
1	Oscilador de cristal externo.

Tabla 7 Configuración del registro PLLPD

PLLPD	**Habilitación de PLL**
0	Habilitación de PLL
1	**Deshabilitación de PLL**

Tabla 8 Configuración el registro LCKON

LCKON	**Detector de cierre**
0	Detector de cierre deshabilitado.
1	**Detector de cierre habilitado.**

3.1.1.2 Registro de División-Por del PLL, PLLDB:

Con el registro División-por (ver figura 18.) se tiene la posibilidad de hacer división del postescalador y del preescalador, además de configurar un valor de frecuencia que se necesite.

Figura 18. Registro de División-Por, PLLDB.

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	LORTP[3:0]				PLLCOD[1:0]		PLLCID[1:0]		0	PLLDB[6:0]						
Write																
Reset	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	1

Tabla 9 Configuración del registro PLLDB

PLLDB	**PLL Dividido-por**
0 - 127	Fout = Frecuencia de salida de PLL. $f_{out} = \frac{FREFx(n+1)}{2}, donde$ $FREF = \frac{Frecuencia_{referncia}(8MHz)}{PLLCID[1:0]}$ n+1=Valor de escala. n= Valor de PLLDB[6:0], [0-127]

Tabla 10 Configuración del registro PLLCID

PLLCID	**Salida de reloj de PLL dividido-por(Preescalador)**
00	Dividido por 1
01	Dividido por 2
10	Dividido por 4
11	Dividido por 8

Tabla 11 Configuración del registro PLLCOD

PLLCOD	**Salida de reloj de PLL dividido-por(Postescalador)**
00	Dividido por 1
01	Dividido por 2
10	Dividido por 4
11	Dividido por 8

3.1.1.3 Registro de Estado del PLL, PLLSR:

El registro de Estado del PLL (ver figura 19), selecciona la fuente de reloj de salida, el estado de preescalador.

Figura 19. Registro de Estado del PLL, PLLSR.

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read / Write	LOLI1	LOLI0	LOCI	0	0	0	0	0	0	LCK1	LCK0	PLLPDN	0	**PRECS**	ZSRC[1:0]	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Tabla 12 Configuración del registro PRECS

PRECS	**Estado de selección de preescalador(56F801)**
0	**Reloj de oscilador de relajación interno.**
1	Reloj de oscilador de cristal externo.

Tabla 13 Configuración del registro LCK0

LCK0	**Perdida de cierre 0**
0	PLL esta abierto(en curso)
1	**PLL esta cerrado(en curso)**

Después de conocer lo necesario de los registros para la configuración de la frecuencia del PLL interno del DSP, se procede a operarlo en lenguaje C de la siguiente manera:

```
clrRegBit(PLLCR, PRECS);
while(getRegBit(PLLSR, PRECS)){}
setRegBit(PLLCR, PLLPD);
setReg(PLLDB, n);
setRegBit(PLLCR, LCKON);
clrRegBit(PLLCR, PLLPD);
while (!getRegBit(PLLSR, LCK0)){}
setRegBitGroup(PLLCR, ZSRC, 2);
```

El primer paso es seleccionar el Oscilador de relajación interno, poniendo en cero el bit PRECS del Registro PLLCR, luego se espera a que el reloj se establezca leyendo el bit PRECS del Registro PLLSR hasta que se ponga en cero, se deshabilita el PLL llevando a uno el bit PLLPD del registro PLLCR, ahora se tiene que introducir el valor(n) que se encargara de configurar la frecuencia de reloj interna del DSP, en el registro PLLDB se coloca el valor necesario, según la formula que a continuación se presenta:

$$frecuencia_{\text{interna}} = \frac{8MHz * (n+1)}{4}$$

Donde los 8MHz se tienen del oscilador de relajación interno, y el valor de n es el colocado en el registro PLLDB, con el valor de n =19, se configura la frecuencia de 40MHz y con n =6 la de 14MHz, después de configurar el valor de división para el PLL, se deshabilita el lazo detector seteando el bit LCKON del registro PLLCR, se habilita de nuevo el PLL poniendo en cero el bit PLLPD del registro PLLCR, se espera a que el PLL se cierre, que es hasta que se ponga en alto el bit LCK0 del registro PLLSR y finalmente se selecciona la salida de frecuencia por post-escalador, llevando a valor dos la mascara ZSRC del registro PLLSR.

3.1.2 PINES DE PROPOSITO GENERAL

El DSP utilizado para el proyecto, no posee puertos de pines de propósito general, todos sus pines tienen su destinación de trabajo; pero el DSP ofrece la posibilidad de configurar ciertos pines dedicados, a algún periférico, para que sean utilizados como pines de propósito general (GPIO), ver cuadro1, *Codewarrior* hace la lista de registros con los nombres y posiciones tal cual aparece en la literatura que Motorola hace de sus DSP, por eso se hace fácil

la configuración de registros por sus nombres y no por su dirección de registro directamente.

Para la configuración de los pines como propósito general se debe tener completo conocimiento de los siguientes registros:

- Registro para habilitación de "*pull-up*", GPIO_X_PUR.
- Registro para dato (valor de salida), GPIO_X_DR.
- Registro para dirección de pin (entrada/salida), GPIO_X_DDR.
- Registro para habilitación de periférico, GPIO_X_PER.

3.1.2.1 Registro para habilitación de "*pull-up*", GPIO_X_PUR

Este registro permite habilitar en el pin de salida la resistencia de ***"pull-up".***

Figura 20 Registro para habilitación de "*pull-up*

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	PU[7:0]							
Write																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Para los pines de salida que necesiten resistencia de ***pull-up,*** el valor de su correspondiente bit en el registro PUR debe ir a uno. Para el GPIO_A, el bit 0, del registro PUR, corresponde al pin de entrada TDO del temporizador; el bit 0 en el GPIO_B, corresponde al pin TXDO y así sucesivamente (ver Tabla 1).

3.1.2.2 Registro para dato (valor de salida), GPIO_X_DR.

El valor que se coloque en este registro, es el valor lógico que se va a presentar a la salida del pin correspondiente, por ejemplo, un uno lógico en el

bit 0, para el GPIO_B, tendría como resultado una salida alta (3.3v) en el pin correspondiente al GPIOB0 (ver Tabla1)

Figura 21 Registro para dato

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	D[7:0]							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3.1.2.3 Registro para dirección de pin (entrada/salida), GPIO_X_DDR.

Este registro configura la dirección para cual se necesita el pin; un uno lógico configura el pin como salida, un cero lógico para entrada.

Figura 22 Registro para dirección de pin (entrada/salida)

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	DD[7:0]							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3.1.2.4 Registro para habilitación de periférico, GPIO_X_PER.

En éste registro se configura al pin para que realice el trabajo como pin de propósito general, o bien, para que trabaje como el pin dedicado por defecto; un 0 lógico, configura al pin para trabaje como GPIO, un uno para que siga como pin dedicado.

Figura 23 Registro para habilitación de periférico

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	DD[7:0]							
Write																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Después de conocidos los registros necesarios para la configuración y operación de los pines de propósito general (GPIO), para el DSP56F801, se presenta a continuación el programa en lenguaje C, que se encarga de la configuración necesaria para el proyecto, Ver tabla 1.

```
setReg(GPIO_A_PUR,0x07);
setReg(GPIO_B_PUR,0x0C);

setReg(GPIO_A_DDR,0x07);
setReg(GPIO_B_DDR,0x0C);

setReg(GPIO_A_PER,0x00);
setReg(GPIO_B_PER,0x30);

setRegBit(GPIO_B_DR,D0);
setRegBit(GPIO_B_DR,D1);
```

De los pines dedicados presentados en la tabla 1 se utiliza en su función destinada únicamente los pines MOSI y SCLK.

Para los demás pines que han sido configurados como GPIO, únicamente se necesita cambiar el valor en su bit correspondiente en el registro DR, para cambiar la salida al estado que el programa lo requiera.

3.1.3 INTERFASE SERIAL DE PERIFERICOS (SPI)

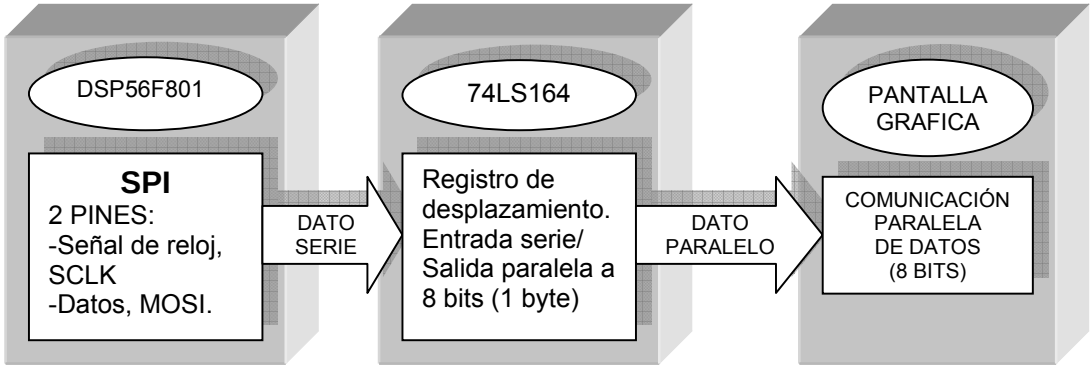
El SPI se conecta al registro de desplazamiento serie-paralelo 74LS164 (ver figura 13), para lo cual se hicieron ajustes de frecuencia en su sincronización (ver frecuencia interna del DSP sección 3.1.1), posee tres registros encargados de configurarlo, habilitarlo y controlarlo, estos son:

- Registro de estado y control del SPI, SPSCR.

- Registro de tamaño de transmisión del SPI, SPDSR.

- Registro para transmisión de dato del SPI, SPDTR.

Figura 24. Diagrama de bloques, comunicación de datos a la pantalla gráfica.



3.1.3.1 Registro de estado y control del SPI, SPSCR.

Con el registro de estado y control del SPI, SPSCR, principalmente se puede seleccionar la rata de Baudios de transmisión y leer banderas de estado.

Figura 25 Registro de estado y control del SPI

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		DSO	SPRF	EERIE	OVRF	MODF	SPTE	MODFEN	**SPR1**	**SPR0**	SPRIE	**SPMSTR**	CPOL	CPHA	**SPE**	SPTIE
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabla 14 Configuración del registro SPE

SPE	**Habilitación de SPI**
0	Deshabilitar modulo SPI
1	Habilitar modulo SPI.

Tabla 15 Configuración del registro SPMSTR

SPMSTR	SPI como maestro/esclavo
0	Selección en modo de operación esclavo
1	Selección en modo de operación maestro.

Tabla 16 Selección de rata de Baudios de transmisión

Selección de rata de Baudios de transmisión		
SPR1	SPR0	BD
0	0	2
0	1	8
1	0	16
1	1	32
Rata de Baudios=(clk / BD), donde clk = Reloj interno del bus del DSP. BD = Divisor de rata de Baudios		

3.1.3.2 Registro de tamaño de transmisión del SPI, SPDSR.

Con este registro se configura el tamaño del dato que se vaya a transmitir.

Figura 26 Registro de tamaño de transmisión SPDSR

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read													DS3	DS2	DS1	DS0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabla 17 Registros DS3-DS0

DS3-DS0	Tamaño del dato a transmitir
0000	No permitido
0001	2 Bits

0010	3 Bits
0011	4 Bits
0100	5 Bits
0101	6 Bits
0110	7 Bits
0111	8 Bits
1000	9 Bits
1001	10 Bits
1010	11 Bits
1011	12 Bits
1100	13 Bits
1101	14 Bits
1110	15 Bits
1111	**16 Bits**

3.1.3.3 Registro para transmisión de dato del SPI, SPDTR.

En este registro se coloca el valor que se vaya a dar como salida de dato serie, una vez escrito el valor en el registro, éste es transmitido inmediatamente.

Figura 27 Registro para transmisión de dato del SPDTR

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read																
Write	T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Con el registro SPSCR se habilita el periférico, se controla los Baudios de transmisión (a lo más lento posible del DSP) por la lentitud del registro, a **32**

baudios, y se configura el SPI únicamente a modo maestro. Con el registro SPSCR se controla el tamaño de datos a enviar, siendo éste valor de **16 bits**, aunque solo se este utilizando la transmisión del byte menos significativo del registro, simplemente se selecciono 16 para colaborar con el retardo de transmisión que se requiere para sincronizar con el registro de desplazamiento.

Enviar un dato por el SPI es muy fácil, pues como éste posee su propia señal de reloj, se sincroniza muy bien, únicamente se procede a escribir el valor a enviar en el registro de transmisión del SPI, SPDTR, una vez terminado de escribirlo en el registro es enviado a salida, en este caso al registro de desplazamiento.

En el lenguaje C, se configura de la siguiente forma:

```
setReg(SPSCR,0XD8); / / 32 baudios de transmisión, modo maestro
setReg(SPDSR,0x0F);  / / transmisión de 16 bits.
setRegBit(SPSCR,SPE);  / / habilitación del periférico.
```

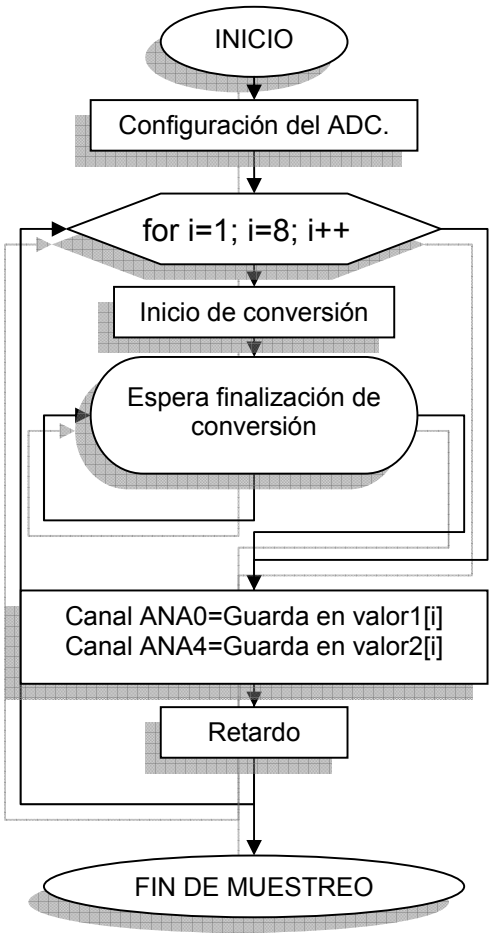
Por ejemplo para enviar el valor de 165 al puerto serie, simplemente escribimos el valor en el registro SPDTR:

```
setReg(SPDTR,165);  / / transmisión inmediata de 165, serial.
```

3.1.4 CONVERSION ANALOGO – DIGITAL

La configuración máxima para muestreo del ADC es de 588,235KHz, con un Δt=1,7μs; la configuración máxima que se realizo en este proyecto fue de 500 kHz, con un Δt=2μs, porque al calcular la regresión senoidal para la frecuencia mas alta de muestreo , que es 62.2kHz, se necesita aproximar a 8 muestras por periodo, es decir muestrear lo mas aproximado a 62200x8=497.6kHz, éste valor fue de 500kHz; la velocidad de conversión del ADC no fue cambiada en todo el programa, para tomar muestras de frecuencias menores, se introdujeron retardos entre muestras y así aumentar el Δt, disminuyendo la frecuencia de muestreo(ver figura 28)

Figura 28 Diagrama de flujo para muestreo de las señales de entrada.



Para la configuración, utilización y aplicación del ADC se trabajo con los siguientes registros:

- Registro1 de Control de ADC, ADCR1.

- Registro2 de Control de ADC, ADCR2.

- Registro para habilitación de muestras de ADC, ADSDIS.

- Registro de Estado de ADC, ADSTAT.

- Registros de Resultado de ADC, ADRSLT0 Y ADRSLT4.

3.1.4.1 Registro1 de Control de ADC, ADCR1.

En este registro se puede configurar el modo de muestreo, la configuración de canales y el inicio de conversión.

Figura 29 Registro1 de Control de ADC, ADCR1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	STOP	0	SYNC	EOSIE	ZCIE	LLMTIE	HLMTIE	CHNCFG[3:0]				0	SMODE[2:0]		
Write			START													
Reset	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1

Tabla 18 Configuración del registro SMODE

SMODE	**Modo de muestreo**
000	Una vez secuencial
001	Una vez simultanea
010	Lazo secuencial
011	Lazo simultaneo.
100	Secuencial por disparo.
101	**Simultaneo por disparo.**
110	Reservado.

111	Reservado.

Tabla 19 Configuración de Canales

CHNCFG Configuración de canales			
Entrada simple. (Referenciado a tierra)	Muestreo de canales simple.	Entrada Diferencial.	Muestreo de canales diferencial
Bit 4 = 0	**AN0** AN1	Bit 4 = 1	AN0 es +, AN1 es -
Bit 5 = 0	AN2 AN3	Bit 5 = 1	AN2 es +, AN3 es -
Bit 6 = 0	**AN4** AN5	Bit 6 = 1	AN4 es +, AN5 es -
Bit 7 = 0	AN6 AN7	Bit 7 = 1	AN6 es +, AN7 es -

Tabla 20 Configuración del Registro START

START	**Inicio de conversión**
0	Ninguna acción.
1	Inicio de conversión.

3.1.4.2 Registro2 de Control de ADC, ADCR2.

Con este registro se configura la frecuencia de conversión del ADC.

Figura 30 Registro2 de Control de ADC, ADCR2

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	DIV[3:0]			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Tabla 21 Selección de divisor de reloj

DIV	**Selección de divisor de reloj**
0-15	N= Div+1 $F_{ADC}=(F_{IPR})/2N$. donde: N=Valor de selección de división de reloj.

	F_{ADC}=Frecuencia de conversión de ADC. F_{IPR}=Frecuencia de reloj del bus periférico de ADC.

3.1.4.3 Registro para habilitación de muestras de ADC, ADSDIS.

Este registro permite habilitar las muestras de canales que se requieran para el proceso de muestreo, por ejemplo, en este proyecto el proceso de muestreo únicamente abarca las muestras para los canales AN0 y AN4, entonces se habilitan las muestras solo de esos dos canales. Se habilitan con un cero lógico en su respectivo bit de registro; AN0 con bit 0, AN1 con bit 1, AN2 con bit 2 y así sucesivamente.

Figura 31 Registro para habilitación de muestras de ADC, ADSDIS

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TEST[1:0]		0	0	0	0	0	0	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3.1.4.4 Registro de Estado de ADC, ADSTAT.

De este registro de banderas básicamente se ha de trabajar con el bit CIP, que da señal sobre la terminación de un muestreo, necesario para guardar los valores y realizar un nuevo muestreo. Este bit CIP es el encargado de tomar los ciclos de muestreo del o los canales que se estén usando.

Figura 32 Registro de Estado de ADC, ADSTAT

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CIP	0	0	0	EOSI	ZCI	LLMTI	HLMTI	RDY7	RDY6	RDY5	RDY4	RDY3	RDY2	RDY1	RDY0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabla 22 Configuración del Registro CIP

CIP	Conversión en progreso
0	Desocupado.
1	Ciclo de muestreo en progreso.

3.1.4.5 Registros de Resultado de ADC, ADRSLT0 Y ADRSLT4.

Estos registros, que en total son 8, uno por cada canal del ADC, permiten leer el valor de conversión logrado por el ADC, es decir la muestra del canal, ésta muestra es de 12 bits, desplazados los tres menos significativos se tendría un orden de conversión en decimal de hasta 32760 niveles de conversión.

Figura 33 Registros de Resultado de ADC, ADRSLT0 Y ADRSLT4

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	SEXT	RSLT[11:0]												0	0	0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

El resultado se almacena en los bits 14-3, siendo el bit SEXT, el signo del valor, utilizado cuando se resta un offset de la señal de entrada. El canal AN0 guarda en registro ADRST0, el canal AN1 en registro ADRST1 y así sucesivamente.

Con el registro de control, ADCR1, se configuro para que realice un muestreo en modo simultaneo y con referenciado a tierra, o modo simple como se conoce, el modo simultaneo se selecciono “Triggered”, o espera por disparo, por permitir la realización de un nuevo “scan” dando pulso al bit “star” del registro de control ADCR1, y al terminar la conversión activa la bandera CIP del registro de estado del ADC, ADSTAT.

En lenguaje C, la forma en que se configuró fue:

```
setReg(ADCA_ADCR1,5);
setReg(ADCA_ADCR2,0);
setReg(ADCA_ADSDIS,0xCC);
```

Y la forma de realizar el muestreo:

```
setRegBit(ADCA_ADCR1,START);
while(getRegBit(ADCA_ADSTAT,CIP)){}
adc_volt[i]=getReg(ADCA_ADRSLT0);
adc_crte[i]=getReg(ADCA_ADRSLT4);
RETARDO_ADC(RETAR_MUES[j]);
```

Donde RETAR_MUES[ ], es un vector de retardos calculados para cubrir en tiempo, con 8 muestras, un periodo completo de la señal a muestrar, y la variable j, maneja el valor de frecuencia en el cual se va; para mayores detalles(valores) ver Anexo D, el programa general.

Como se observa, iniciar una adquisición de datos es bien sencillo, pues una vez configurada la forma en que se vaya a realizar el "*scan*" se da señal a "star" y por la bandera de conversión CIP, del registro ADSTAT, se conocerá cuando se haya terminado la conversión, quedando listo para leer del respectivo registro de resultado para cada canal, continua con un nuevo pulso de *star* hasta completar las ocho muestras que se necesitan de la señal(ver diagrama de flujo para muestreo de las señales de entrada, figura 28)

3.2 INICIALIZACION Y MANEJO DE LA PANTALLA.

3.2.1 INICIALIZACIÓN

El manejo y programación de esta pantalla se debe entender como particular y solo aplicable a esta referencia de pantalla. Se realizaron una serie de subrutinas o funciones que hicieron fácil esa programación y manejo.

Para la pantalla de cristal líquido se debe diferenciar entre un byte de información como datos y un byte de información como comando, la programación consiste en una lista de comandos que van siendo utilizados según la necesidad que se tenga, seguido de todo comando va un dato.

Con ayuda de la información ofrecida por el fabricante se seleccionaron las siguientes características principales:

- Generador de caracteres interno.
- 8 líneas por carácter.
- Panel simple
- Tamaño vertical del carácter: 8 píxeles.
- Tamaño horizontal del carácter: 8 píxeles.
- 128 líneas de *display*.
- Ancho de la pantalla virtual: 32 direcciones.
- Bloque en memoria de texto: 0x0000 a 0x0FFF. (4096 Espacios.)
- Bloque en memoria de grafico: 0x1000 a 0x1FFF. (4096 Espacios.)

La pantalla de cristal líquido tiene dos pantallas, una de texto y otra grafica, lógicamente ambas sobre la misma LCD, únicamente que son traslapadas, para dar su forma final de presentación.

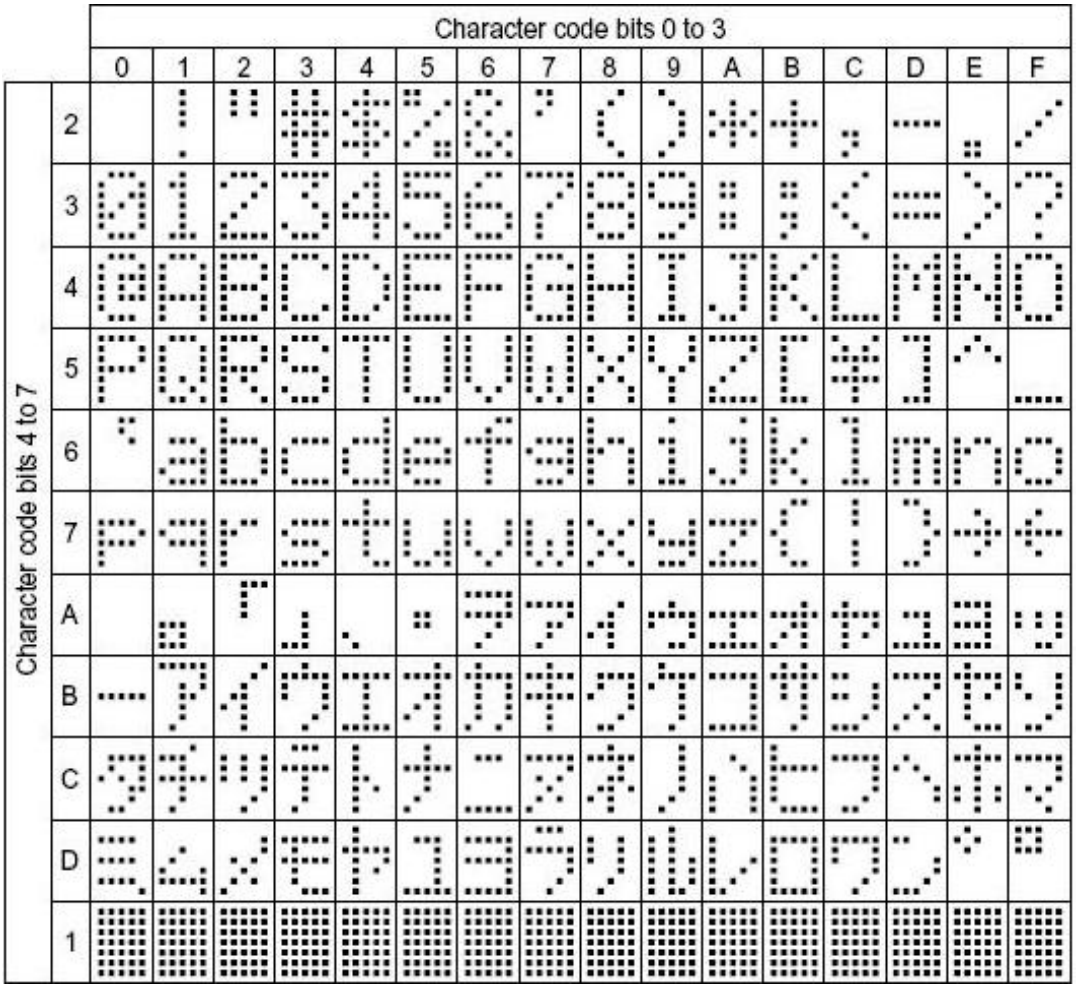
La distribución de cada una de ellas se puede apreciar en las figuras 34 y 35; esta distribución abarco la totalidad de la pantalla tanto en texto como en grafico.

Para la pantalla de texto el generador de código ASCII presenta los caracteres según la figura 36.

Figura 34 Pantalla de Texto

Figura 35 Pantalla de Grafico

Figura 36. Caracteres presentados por generador ASCII interno de la LCD.



3.2.2 DESPLEGAR TEXTO:

La matriz de direcciones que se tiene como resultado de la inicialización de la pantalla es de: 16 filas por 32 columnas. Si se quisiera colocar texto en la primera fila primera columna, se debe posicionar el cursor en la dirección 0x0000, elegir el movimiento del cursor hacia la derecha y enviar los códigos ASCII de cada uno de los caracteres que se necesiten desplegar; en el lenguaje C seria:

```
POS_CURSOR(0x0000);
MOV_CURSOR(Derecha);
```

```
ESCRIBIR_DATO('T');
ESCRIBIR_DATO('E');
ESCRIBIR_DATO('X');
ESCRIBIR_DATO('T');
ESCRIBIR_DATO('O');
```

Función ESCRIBIR_DATO(dato): Envía "dato" a la pantalla.

```
void ESCRIBIR_DATO(byte Dato)
{
setReg(SPDTR,Dato);
ESCRITURA_WR(Bajo);
MICROSEC(46);
ESCRITURA_WR(Alto);
}
```

El carácter entre comillas es el valor de código ASCII que envía a la pantalla; otra característica especial que se debe conocer es que una vez escrito un carácter el cursor se desplaza a su siguiente posición (derecha, izquierda, arriba o abajo) según se configure el movimiento del mismo (ver programa general Anexo D).

Se tienen 32 columnas por 16 filas, entonces se tendría que tener en cuenta un espacio en memoria de 512 caracteres, lo que se direccione en una determinada memoria de esos 512 bytes se va a ver en pantalla, lo demás no tiene lógica, no se vería nada; se recorren las filas sumando 0x20h a las posiciones a partir de cero y para avanzar en columnas se suman de a 0x01h, si se quisiera un texto en la fila 6 con columna 4:

0x0000h+6*0x020h+4d=196d.

Se posiciona el cursor en 196d y se envía la cadena de caracteres.

El programa se optimiza creando funciones especificas, entre estas se puede resaltar las de posicionar el cursor, la dirección de movimiento, escribir un comando, escribir un dato, pulso de escritura y pulso de comando.(ver anexo D programa general).

3.2.3 DESPLEGAR GRAFICOS.

No hay mucha diferencia en el modo en que se procesa un grafico a un texto, solo que el controlador va a convertir los datos de memoria a un mapa binario y no a un código ASCII de texto.

Para gráficos hay que pensar en una matriz de las mismas 32 columnas pero con 128 filas, es decir un espacio en memoria de 4096 bytes, que empezaría en la dirección de memoria 0x1000h, para desplazarse en la pantalla grafica por columnas se suma de a 0x01 para hacerlo por filas de a 0x20h o 32d.

Por ejemplo si se traza un marco a todo el contorno de la pantalla, se posiciona en la dirección 0x1000h, se hace que el cursor direccione hacia la derecha, y se envian 32 datos de bytes 0xFF, y se tendrá la línea superior; para la línea izquierda, se tendría que posicionar en la segunda fila y primera columna, es decir 0x1000+1*0x20=0x1020, el cursor direccionaria hacia abajo por 126 filas con el dato 0x80, que en binario es b10000000 y de la misma manera se haría para la línea inferior y la línea izquierda, solo que la línea izquierda se tendría que enviar el dato 1.
En lenguaje C, la línea superior seria:

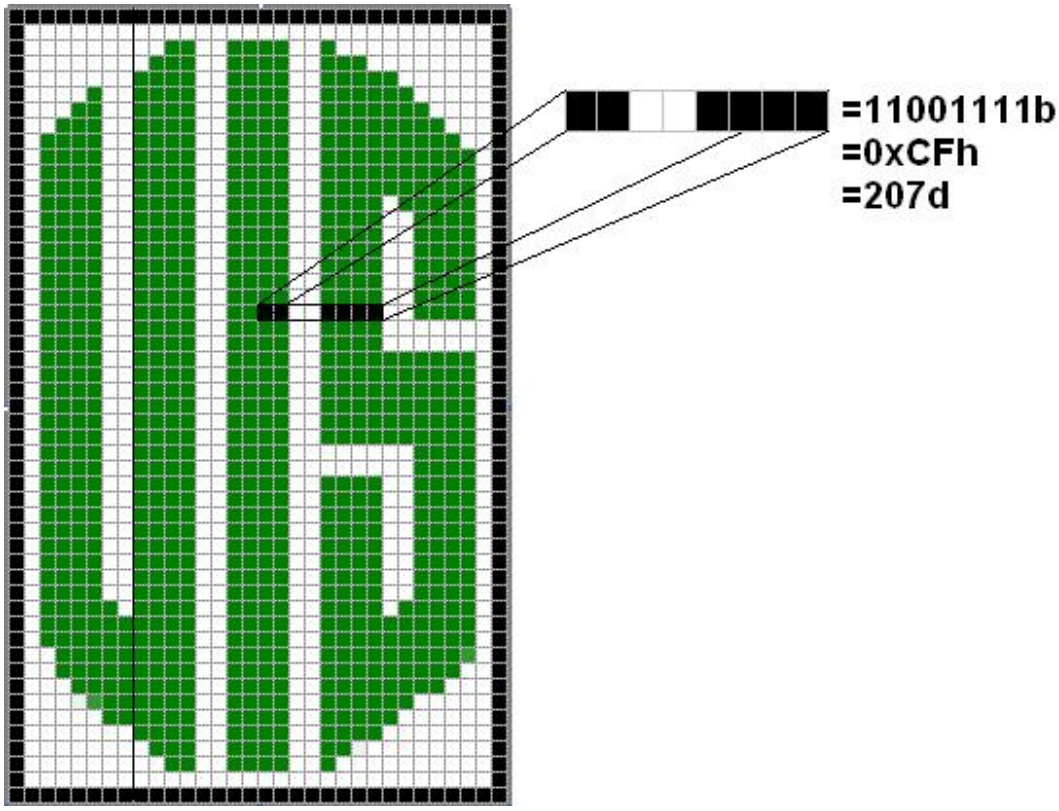
```
POS_CURSOR(0x1000);
MOV_CURSOR(Derecha);
for(i=1;i<=32;i++)
     ESCRIBIR_DATO(0xFF);
```

3.2.3.1 COMO SE LOGRA EL LOGO UIS

Para presentar en pantalla cualquier dibujo (Logo UIS) en particular lo primero que se debe hacer es la representación en mapa de bits, que para la utilización de la pantalla gráfica seria representación en mapa de píxeles.
Con ayuda del software “Paint” de Windows se puede lograr, utilizando una ampliación del 800% se cuadricula la imagen que se vaya a trabajar. Por facilidad de código y distribución de pantalla es preferible hacer la cuadricula para un ancho completo en bytes (8 píxeles), es decir un ancho de 8, 16, 24, 32,40,…píxeles, en la configuración e iniciación de la pantalla se ha seleccionado un ancho de 8 bits (píxeles) para el cursor; de esta forma se estaría aprovechando completamente el byte de memoria tanto del DSP como de la pantalla, y el espacio en grafica.

En el proyecto se han presentado dos logos UIS(dos tamaños), uno para la portada y otro para el despliegue de graficas; la manera en que se procede es crear un vector (1 dimensión) que contenga en código Hexadecimal (o decimal) cada 8 píxeles de la imagen (1 byte-ver figura-37); se toma el primer byte de la esquina superior izquierda, luego se desplaza hacia la derecha(cambio de columna) y se toma el byte, así se continua hasta cubrir la figura por el ancho(# de columnas); después se salta de fila y se continua con el mismo procedimiento hasta cubrir toda la imagen; lo importante es tener en cuenta la forma en que se tomo los bytes de la imagen, pues se va a guardar en único vector toda la imagen, y en el momento de desplegarla hay que hacerlo en la misma forma en que se haya guardado.

Figura 37 Logo UIS



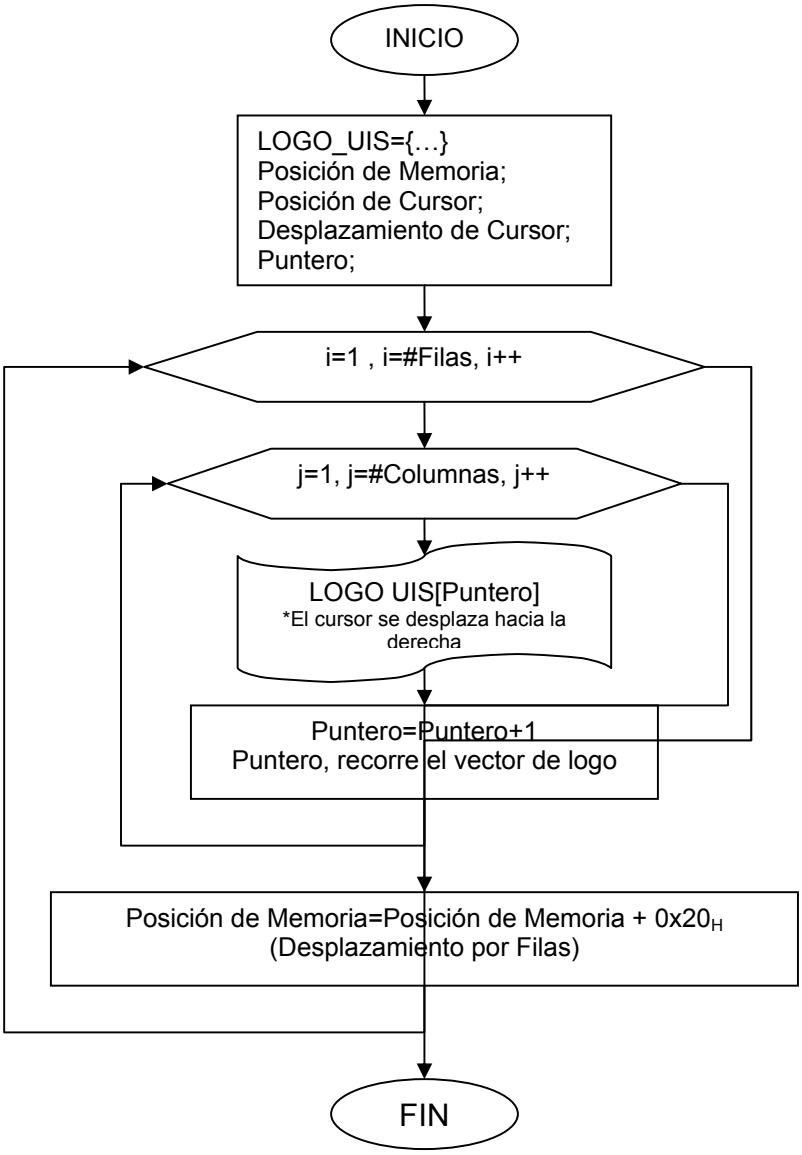
Finalizado el proceso de caracterización en mapa de bits(píxeles), con longitudes conocidas, se establece en la pantalla un punto de inicio para presentar la gráfica, ayudado de las funciones POS_CURSOR(Dirección), MOV_CURSOR(Derecha), ESCRIBIR_COMANDO(0x42) y ESCRIBIR_DATO (UIS[]), un diagrama de flujo representativo del despliegue de la imagen seria el uso de dos bucles "for"(ver figura 38), uno interno que manejara la cantidad de columnas y otro externo se encargara de recorrer el numero de filas, el movimiento que se hace del cursor debe ser de la misma forma en que se caracterizo el mapa de bits, es decir si se hizo hacia la derecha por filas completas y hacia abajo con saltos de columna, esa debe ser la forma de graficar; hay que tener muy presente que para la configuración lograda para la pantalla desplazarse por columnas es sumarle una posición a la dirección de memoria, para desplazarse por filas hay que sumar $0x20_H$ o 32_d .

En lenguaje C, seria:

```
Direccion=0x1000;
MOV_CURSOR(Derecha);
for (i=1;i<=#filas;i++)
 {
   POS_CURSOR(Direccion);
   ESCRIBIR_COMANDO(0x42);
   for (j=1;j<=#columnas;j++)
    {
     ESCRIBIR_DATO(UIS[puntero]);
     puntero++;
    }
 Direccion=Direccion+0x20;
}
```

Donde la dirección inicial (0x1000), no necesariamente debiera ser el inicio del direccionamiento de la pantalla grafica.

Figura 38 .Diagrama de flujo de presentación de logo UIS.



3.2.3.2 DIBUJAR UNA FUNCION

Este punto es el más importante y el que posee mas dificultad dentro del manejo de la pantalla para desplegar gráficos; la pantalla esta configurada con un cursor de ancho 8 píxeles, por conveniencia para texto, y que para gráficos va a resultar un poco molesto, pues el calculo que se hace píxel a píxel no se puede desplegar, de la misma manera, en pantalla hasta no tener la información completa del byte.

La estrategia planteada es utilizar un vector para enmascarar las amplitudes que vayan resultando con los valores que tenga la fila del byte donde se este realizando el calculo, este vector es de ocho elementos (10000000b,01000000b,00100000b,00010000b,00001000b,00000100b,00000010b,00000001b); el procedimiento es el siguiente:

1- Crear un vector de dimensión igual al alto de filas (píxeles), donde se almacenara la información de una columna (8 píxeles de ancho), para ser enviada totalmente cuando se haya calculado su último píxel.

2- Calcular la amplitud, ya sea con función, o con entrada de ADC, y escalarla al alto en píxeles que se tenga disponible.

3- El valor escalado dará la altura en la grafica, lo que representaría la posición en el vector creado en el primer paso, luego si el calculo realizado de amplitud es del primer píxel del byte (8 píxeles) se enmascara el valor del primer elemento del vector de mascara a la posición dada por la amplitud escalada, con una simple operación “or” píxel a píxel (bit a bit); este procedimiento se realiza para no borrar las amplitudes anteriores guardadas del mismo byte.

4 –Después de completar el byte se despliega en pantalla la información que contenga el vector de altura, hay que tener presente la forma en que se haya calculado la altura de la señal, para que sea desplegada de esa misma forma, siendo lo mas practico de abajo hacia arriba para trabajar siempre con valores positivos; al desplegar el vector columna puede presentar vacíos entre filas, entonces con pocos condicionales se podrá completar la línea hasta que se encuentre el valor de amplitud guardada.

5 – Finalizada la primera columna se desplaza a la siguiente para realizar el mismo procedimiento; así se continúa hasta abarcar el número de bytes de ancho que se tengan para la grafica (22 bytes).

3.2.4 TRASLAPE DE TEXTO CON GRAFICA

Como se sabe, la configuración de la pantalla consta de una pantalla de texto y una pantalla grafica que abarcan toda la LCD, en el momento en que se presente un traslape entre una parte de grafico y una de texto, se mostrara en pantalla un contraste entre las dos, que consiste en dejar el texto en *off* , es decir no presentar el píxel, en los píxeles en que se presente el traslape, dando un aspecto de sombra del grafico sobre el texto. Esto es muy importante tenerlo en cuenta pues el texto podría ser oculto (cancelarse) dentro de un grafico, por no ser pantalla grafica a colores no se podría hacer diferencia sobre el uno o el otro (texto o grafico).

3.3 FUNCIONES MATEMATICAS

Partiendo del hecho que no se tienen acceso a las librerías matemáticas, se implementaron una serie de métodos numéricos para solucionar cálculos matemáticos necesarios dentro del flujo completo del programa; las series de “Taylor” para la funciones de seno y coseno, aproximaciones por regresión en cuarto orden para la función de arcotangente, y funciones de cálculos acumulativos como, factorial y potencia, y por ultimo obtener la raíz cuadrada.

Por series de Taylor:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \ldots$$

El coseno se implemento desplazando al seno

cos(x)=sin(x+π/2);

Con el fin de mejorar la regresión cuadrática para la función arcotangente se dividió en dos partes, una abarca desde 0 hasta 6, y la otra desde 6 hasta 60, de ahí en adelante se suponen ángulos muy cercanos a los 89.9 grados, por eso se aproxima a un valor en radianes de 1.5535.

Entre 0 y 6,(radianes)

$$a\tan(x) =$$
$$-3.389E-3*x^4+5.7207E-2*x^3-3.619E-1*x^2+1.0775*x+5.1949E-3$$

Entre 6 y 60,(radianes)

$$a\tan(x) =$$
$$-1.029E-7*x^4+1.616E-5*x^3-9.2818E-4*x^2+2.3820E-2*x+1.3054$$

En el momento de pasar a programa de Codewarrior, en lenguaje C, se hizo necesario programar las funciones básicas como factorial y potenciación:

Factorial

```
long double fac(char x)
{
 int i;
 long double result=1;

 for(i=2;i<=x;i++)
    result*=i;  / / acumula 2*3*4*5*6… hasta x.

 return result;
}
```

Potenciación

```
long double pot(float val,char x)
```

```
{
 int i;
 double result=1;

 for(i=1;i<=x;i++)
   result*=val; / / acumula val*val*val*.. x veces

 return result;
}
```

Finalmente las funciones a implementar quedarían así:

Seno

```
float sin(float x1)
        {
                float xx;
                int y1,signo=1;
                xx=x1;

                if(x1>=pi)
                {
                y1=x1/pi;
                signo=pot(-1,y1);
                xx=x1-y1*pi;
                }

                return(signo*(xx-pot(xx,3)/fac(3)+pot(xx,5)/fac(5)-pot(xx,7)/fac(7)
                          +pot(xx,9)/fac(9)-pot(xx,11)/fac(11)+pot(xx,13)/fac(13)
                        -pot(xx,15)/fac(15)+pot(xx,17)/fac(17)-pot(xx,19)/fac(19)
                        +pot(xx,21)/fac(21)-pot(xx,23)/fac(23)+pot(xx,25)/fac(25)
                        -pot(xx,27)/fac(27)+pot(xx,29)/fac(29)-pot(xx,31)/fac(31)
                       +pot(xx,33)/fac(33)-pot(xx,35)/fac(35)));
        }
```

Coseno

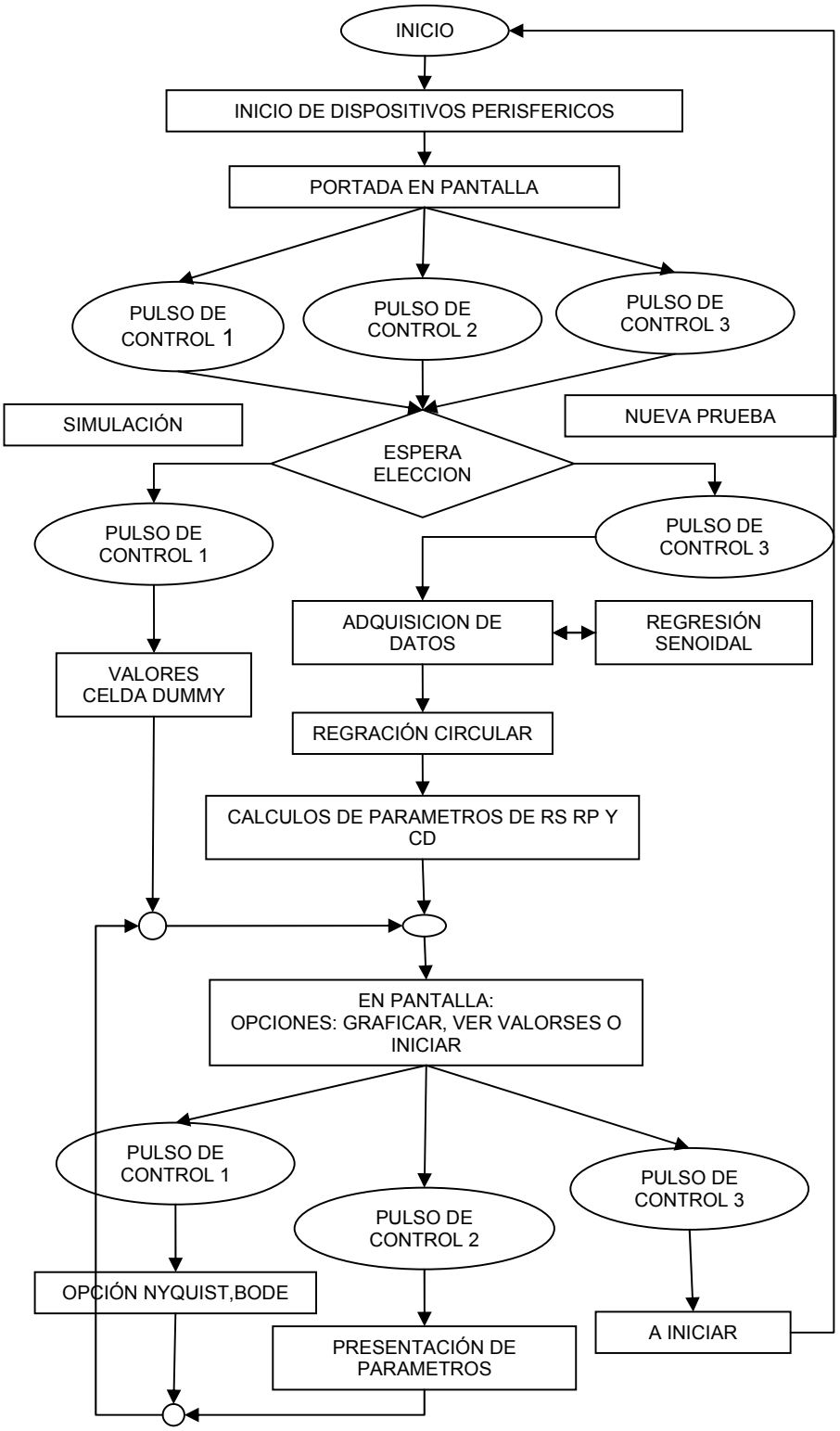
```
        float cos(float x)
        {
        return sin(x+1.570796327);
        }
```

Arcotangente

```
long double atan(float x1) //atan= x - x3/3 + x5/5 - x7/7 + x9/9...
{
        long double xx=x1;
        long double signo=1;
        if(xx==0) return 0;
        if(x1<0)
        {
        signo=-1;
        xx=-x1;
        }

        if(xx<=6) return(signo*(-(3.38905e-3)*pot(xx,4)+
                (5.72072e-2)*pot(xx,3)-(3.61935e-1)*xx*xx+
                1.07748*xx+(5.19498e-3)));

        else if(xx<=60) return(signo*((-1.0297955e-7)*pot(xx,4)+
                (1.6160681e-5)*pot(xx,3)-(9.2818849e-4)*xx*xx+
                (2.3820753e-2)*xx+1.305445));

        else return  signo*1.55335;
}
```

3.4 DIAGRAMA DE FLUJO GENERAL DEL PROGRAMA

Figura 39 Diagrama de Flujo general

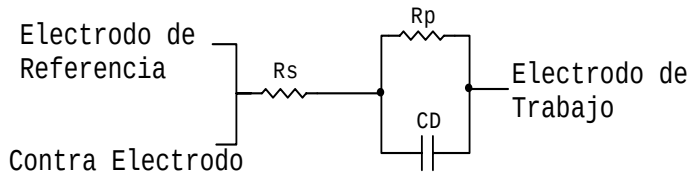

INICIO
INICIO DE DISPOSITIVOS PERISFERICOS
PORTADA EN PANTALLA
PULSO DE CONTROL 1
PULSO DE CONTROL 2
PULSO DE CONTROL 3
SIMULACIÓN
NUEVA PRUEBA
ESPERA ELECCION
PULSO DE CONTROL 1
PULSO DE CONTROL 3
ADQUISICION DE DATOS
REGRESIÓN SENOIDAL
VALORES CELDA DUMMY
REGRACIÓN CIRCULAR
CALCULOS DE PARAMETROS DE RS RP Y CD
EN PANTALLA: OPCIONES: GRAFICAR, VER VALORSES O INICIAR
PULSO DE CONTROL 1
PULSO DE CONTROL 2
PULSO DE CONTROL 3
OPCIÓN NYQUIST,BODE
PRESENTACIÓN DE PARAMETROS
A INICIAR

4. PRUEBAS

Con el firme propósito de conocer la funcionalidad del proyecto, es necesaria la realización de diferentes tipos de prueba que alimenten al proyecto, y así conocer y analizar su desempeño.

Antes de entrar en detalle, las pruebas finales se realizaron mediante el uso de una celda *''Dummy'',* la cual simula el comportamiento básico de una celda electroquímica. La celda utilizada consta de una resistencia (Rs), la cual en uno de sus extremos tiene conectada un electrodo de referencia (RE) y un contra electrodo para cerrar el circuito, y en el otro extremo de la resistencia (Rs), están en paralelo una resistencia (Rp) y el condensador (CD) conectados al electrodo de trabajo. Tal como se ilustra en el Tabla 22.

Tabla 22 Valores de la celda Dummy

Elemento	Circuito Equivalente
Rsmin= $10\,\Omega \pm 0.1\%$ Rsmax=$20\,\Omega \pm 0.1\%$ Rpmin=$100\,\Omega \pm 0.1\%$ Rpmax=$200\,\Omega \pm 0.1\%$ CD=100uF $\pm$ 20%	

Fuente: [G106-89]

Los valores están dentro de la normas ASTM para la verificación de impedancia electroquímica.

4.1 PRUEBAS EN LA PANTALLA GRÀFICA HG25504NG-01 DE HYUNDAI

4.1.1 Despliegue de texto.

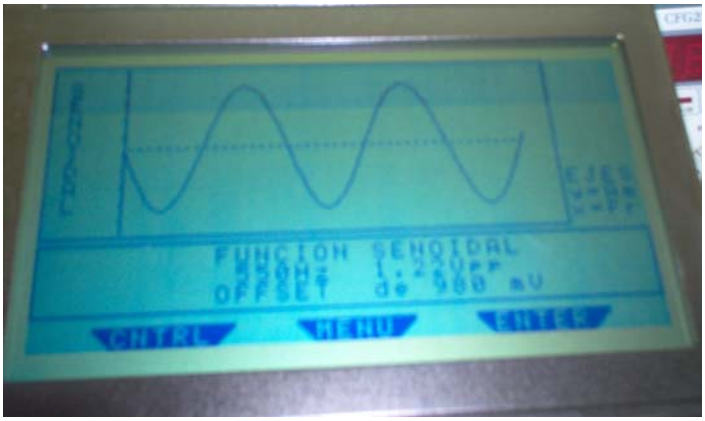
Figura 40 Despliegue de Texto

Figura 41 Portada Logo UIS

4.1.2 Despliegue Gráfico

A continuación se trabajo con la capa destinada para Gráficos correspondiente a las direcciones desde la 0x1000 hasta 0x1FFF (4095 bytes) combinando la capa destinada para texto. La figura 42 muestra una señal senoidal, generada por medio de un generador en donde el ADC del DSP esta configurado para tomar esta señal y para luego ser procesada y visualizada en la pantalla de cristal liquido de modo que se puedan ver las formas de ondas seleccionadas en forma correcta.

Figura 42 Onda Senoidal



4.2 Pruebas en el medidor de impedancia electroquímica

Al Medidor de impedancia electroquímica se le realizaron varias pruebas para conocer el rango de frecuencias en la que trabaja el equipo, dando como resultado el cuadro 25.

Tabla 23 Rango de Frecuencias del MIE

FRECUENCIAS (Hz)
62200
54750
48270
34950
19620
17180
12270
10160
10610
8157
5870
3286
2874
2049
1698
950.6
826.4
729.9
524.1
293.1
256.4
182,5
150.9
97.28
84.18
74.29
53.19
29.62
25.77
18.45
15.08
8.881
7.728
6.803

4.909
2.732
2.388
1.724
1.420
0.8929
0.7576
0.6727
0.4902
0.2688
0.2358
1.1678
0.1397

Como resultado de varias pruebas la tensión de salida del medidor de impedancia es constante en 30mVp.

Las pruebas realizadas se basaron en una ´´*Celda Dummy*´´ con los valores del cuadro 3 de acuerdo a las normas ASTM para la verificación de la celda electroquímica.

El programa desarrollado en ´´codewarrior´´ permite caracterizar la respuesta en frecuencia de una celda electroquímica. Esta respuesta en frecuencia es el diagrama de ´´Nyquist´´ y el diagrama de ´´Bode´´ en magnitud y fase.

A continuación se muestra la respuesta en frecuencia para el equivalente de la celda electroquímica. Los valores de la celda utilizada fueron : Rs=10Ω, Rp=100 Ω y C_D=100μF.

Figura 45 Diagrama de Nyquist

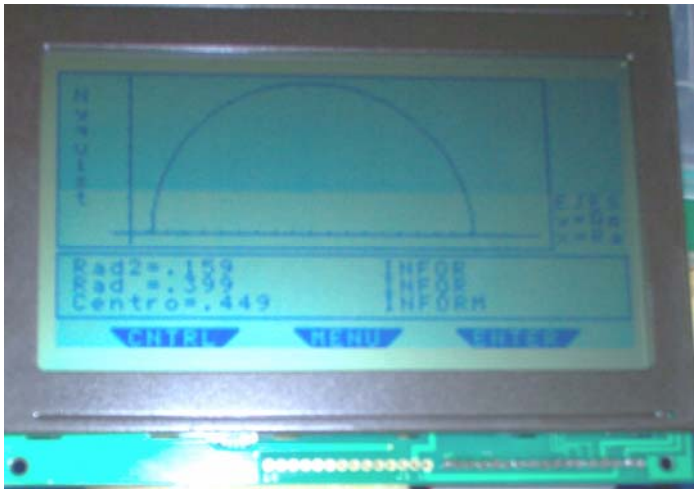
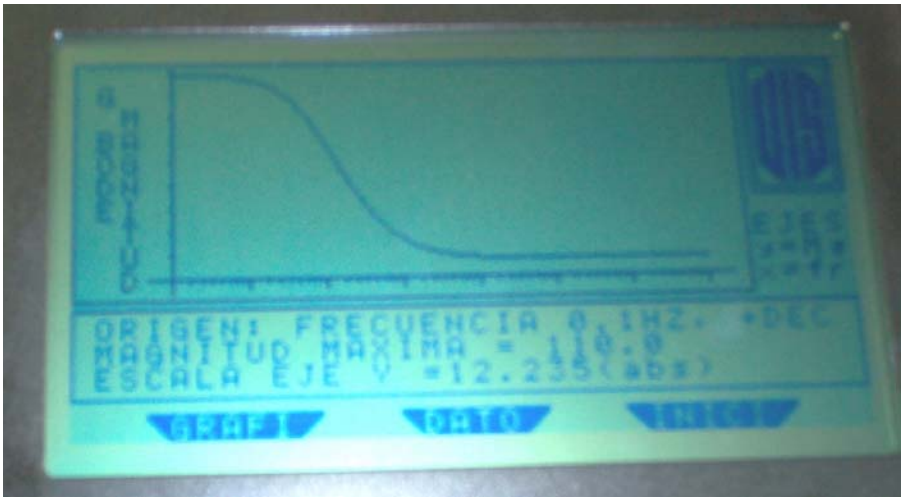
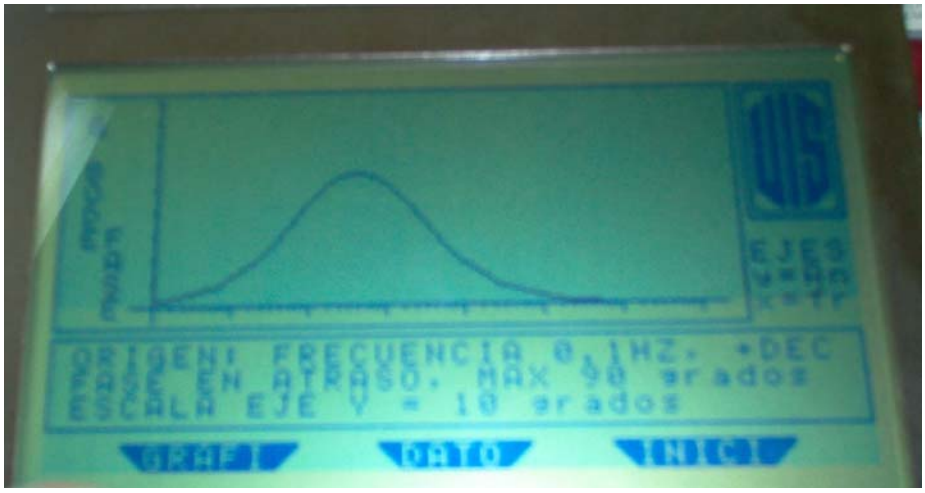


Figura 46 Diagrama De Bode Magnitud



En la grafica 46 se muestra el diagrama de bode en magnitud para la celda electroquímica con los valores mostrados en la tabla 22. En donde la escala del eje y es igual a 12.235 en valor absoluto (magnitud).

Figura 47 Diagrama de Bode Fase



La figura 47 es el diagrama de bode en fase, de la celda equivalente de la tabla 22, en donde la escala del eje y es igual a 10 grados.

Los valores obtenidos durante las pruebas se muestran en la tabla 24.

Tabla 24 Valores de Resistencia y Condensadores obtenidos

Parámetro	Valor obtenido	Valor experimental	Error
Rs	18.89	19.90	5%
Rp	82.25	97.98	8%
Cd	1.30E-4	1.04E-4	23%

Parámetro	Valor obtenido	Valor experimental	Error
Rs	8.78	9.91	11.4%
Rp	78.85	97.98	19%
Cd	1.25E-4	1.04E-4	20%

Parámetro	Valor obtenido	Valor experimental	Error
Rs	11.10	9.91	10.72%
Rp	81.79	97.98	19%
Cd	0.8179-4	1.04E-4	21.35%

Parámetro	Valor obtenido	Valor experimental	Error
Rs	9.78	9.91	1.32%
Rp	98.35	97.98	0.37%
Cd	0.99E-4	1.04E-4	2%

Parámetro	Valor obtenido	Valor experimental	Error
Rs	9.96	9.91	0.5%
Rp	94.59	97.98	3.58%
Cd	1.01E-4	1.04E-4	2.97%

CONCLUSIONES

- Se ha implementado un prototipo de unidad de procesamiento basado en *´´DSP´´* para la caracterización de una celda electroquímica. Se visualizan los resultados en una pantalla de cristal líquido. Las opciones implementadas para la caracterización de una celda electroquímica son: Seleccionar Simulación, el cual realiza el diagrama de Nyquist, el diagrama de bode en magnitud y fase, introduciéndole los valores de las resistencias Rs, Rp y el Condensador C_D. Seleccionar Nuevo, el cual realiza el proceso de adquisición de las señales de tensión y corriente de la celda electroquímica, para luego calcular sus valores y graficar la respuesta en frecuencia.

- Se diseñaron funciones en alto nivel de programación prácticas para facilitar la programación de la pantalla de cristal líquido HG25504NG-01.

- Se logro acoplar el Medidor de Impedancia Electroquímica con el DSP. En donde el DSP es el encargado de trabajar en modo Maestro y el MIE en modo esclavo.

- La configuración de pines dedicados a algún periférico, como pines de propósito general GPIO, hace que estos pines trabajen como un puerto de Entrada/Salida normal, este proyecto utilizo al limite la capacidad sobre estos pines (11pines), llegando hasta la utilización de los pines utilizados para una conexión de reloj de cristal externo, fueron utilizados como GPIO, debido a la falta de GPIO.

- Dentro de los grandes avances de la tecnología en los últimos 20 años, cabe anotar el gran desempeño y versatilidad que ha demostrado el trabajo sobre microcontroladores y microprocesadores; llegando a integrar en un DSP funciones mas eficaces junto con manejos periféricos y procesadores matemáticos; alcanzando gran velocidad de procesamiento y ejecución, pero cabe resaltar, que la tecnología no puede estar creciendo de una sola rama, el verdadero crecimiento de la tecnología depende de una gran cantidad de factores como avances en tecnología de materiales, en almacenamiento de energía (pilas), y avances en paralelo. En este proyecto se experimento un gran desnivel o tropiezo con la comunicación que se debiera realizar entre la pantalla y el DSP por medio del registro de desplazamiento; donde se tuvo la necesidad de restarle capacidad de procesamiento y operación al DSP, simplemente porque es demasiado rápido para otro periférico; la posibilidad de configurar frecuencias de trabajo o ciclos de reloj para el DSP, es prueba de que se tiene que estar dando la posibilidad de acople entre lo nuevo y lo que se esta trabajando en el momento.

- Las señales provenientes de medidor de impedancia electroquímica, no son señales muy optimas, ya que son señales que vienen como muchas señales no deseadas (ruido) y no se tiene un control eficiente en cuanto a la frecuencia de las señales provenientes del aparato y esto ocasiona errores en la sincronización y esos factores alteran los resultados en una forma significativa

RECOMENDACIONES

- Utilizar un DSP mas avanzado que se encargara de la generación y adquisición de las señales, ayudaría enormemente a minimizar los errores de lectura de frecuencia y sincronizar la generación con la lectura, no se tendría que hacer configuraciones de tipo maestro-esclavo, e incluso trabajar con una pantalla grafica de colores, para tener una mejor presentación.

- Para el acople de las diferentes señales entre dispositivos hacer etapas previas a la adquisición de datos como el filtrado, seguidores de tensión, para minimizar el error en el calculo final de los parámetros.

- Contar con la capacidad ilimitada de la licencia para la plataforma de codewarrior, para que la cantidad de código no pase a ser la restricción del programa sino la capacidad de memoria del DSP. Ya que en este proyecto se llego al limite de programación en un valor de 20210 bytes en cuanto a Codewarrior pero no por parte del DSP.

- Implementar el proyecto con un DSP de mayor disposición de pines o puertos de propósito general, para evitar trabajar al límite y tener la posibilidad de realizar mas funciones destinas a entradas y salidas.

BIBLIOGRAFIA

[DSP56F800FM, 03] *Motorola*. "DSP56800 *Family Manual*". 2003.

[DSP56F801-7UM/D, 03] *Motorola*. "DSP56801/802/803/805/807 *User`s Manual*". 2003.

[DSP56F801/D, 03] *Motorola*. "DSP56801 *Technical Data Sheet*". 2003.

[Smith, 99] Smith, Steven. "*The Scientist and Engineer's Guide to Digital Signal Processing*". Segunda edición, *California Technical Publishing*, 1999.

[Proakis & Manolakis, 98] Proakis, John y Manolakis, Dimitris. "Tratamiento digital de señales". Tercera edición, *Prentice Hall*, Madrid, 1998.

[Sedra & Smith, 98] Sedra, Adel y Smith, Kenneth. "Circuitos Microelectrónicos". Cuarta Ediciòn, Oxford University Press, Inc., U.S.A 1998.

[SED 1330, 95] S-MOS Systems, Inc. Datasheet "LCD Controller ICs Technical Manual**".** http://ertw.dhis.org/LCDPro/LCDInfo.htm

RODRÍGUEZ PACHECO, Jorge Humberto. RUIZ GÓMEZ, Sergio Andrés. "Medidor de Impedancia Electroquímica". Bucaramanga 2004, 110 páginas. Trabajo de Grado (Ingeniero Electrónico). Universidad Industrial de Santander, Escuela de Ingeniería Eléctrica, Electrónica y Telecomunicaciones.

ANEXO A ENTORNO A CODEWARRIOR

´´Metrowerks´´ y Motorola han dejado disponible una nueva versión especial del entorno de desarrollo para la familia de ´´DSP56800/E Hybrid Controllers: CodeWarrior Development Studio for Motorola 56F800 v.6.0´´ es una interfaz tipo Windows que posee menús, barras de herramientas que ayudan a construir, revisar y probar aplicaciones en tiempo real, la cual es gratuita y puede compilar hasta 8kbytes de código C. Con lo cual se cuenta con una máquina virtual que permite simular la CPU, periféricos e interrupciones de toda la familia de DSP56800 actuales de Motorola, lo que facilita el proceso de depuración de las aplicaciones desarrolladas en lenguaje C.

A continuación se procederá a realizar una introducción rápida para crear proyectos en lenguaje C, utilizando el compilador ´´Metrowerks CodeWarrior v6.0´´.

Antes de empezar la creación de proyectos se debe escribir que los parámetros básicos que necesita este software para la realización de un programa adecuado, codewarrior necesita un ´´DLL´´ (Dynamic Link Libraries) para iniciar el puerto paralelo, ya que trabaja con el puerto de emulación ´´JTAG´´ (Joint Test Action Group), este ´´DLL´´ se adquiere de la pagina www.motorola.com y tener conectada las tarjetas del DSP para poder

trabajar en codewarrior. Solo si se va a trabajar en modo Simulador no se necesita tener conectada la tarjeta del Puerto JTAG y la tarjeta del DSP 56F801 de motorola.

El primer paso es crear el proyecto. Para esto se selecciona **File | New…**

Con esto aparecerá la ventana que se muestra en la figura A1. Escoja **DSP56800x ´´New Project Wizard´´,** la carpeta donde va a ubicar su proyecto y el nombre del proyecto.

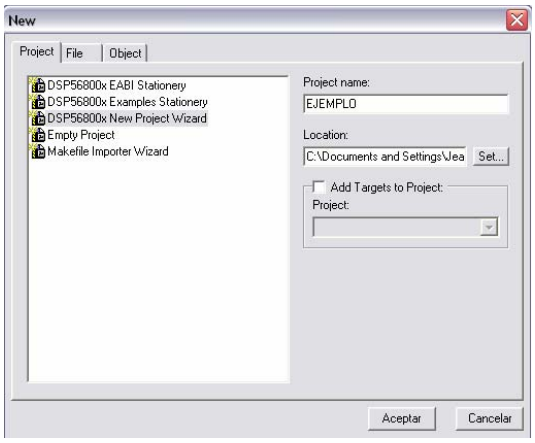


Figura A1. Crear proyecto nuevo

Cuando presione Aceptar, aparecerá otra ventana para escoger el tipo de la familia DSP, y el DSP correspondiente el cual para este caso es DSP56F801 a 80Mhz.Tal como aparece en la figura A2.

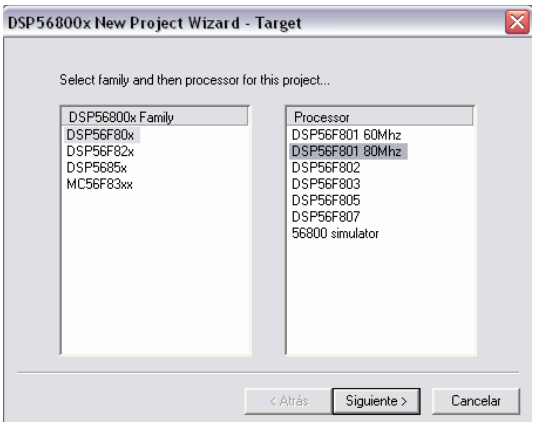


Figura A2 escoger el DSP

Luego de presionar ´´Siguiente´´ aparecerá una ventana donde se tiene la posibilidad de escoger el lenguaje en el cual se desea trabajar. En este caso se procedió trabajar con Procesador Experto.

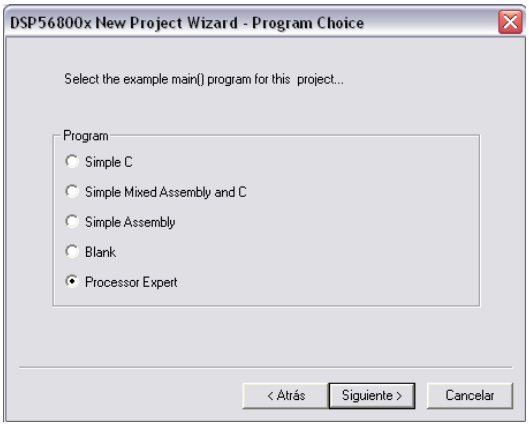


Figura A3 Lenguaje de Programación

Cuando presione ´´siguiente´´ *Codewarrior* creará una carpeta con el nombre que le dio a su proyecto y creará un proyecto de ejemplo en el lenguaje que escogió, en este caso lenguaje C. La siguiente figura muestra la ventana principal de Metrowers Codewarrior.

En ella se aprecia que aparece una ventana que contiene el nombre del proyecto con la extensión mcp, en la parte superior derecha aparece la ventana Target CPU la cual permite ver el DSP escogido que para este caso es el dsp58f8001, en esta ventana se puede ver la configuración de pines del DSP. Por ultimo esta la ventana de ´´Bean Selector´´ que permite a un usuario seleccionar un Bean deseado y agregarlo al proyecto.

Cabe decir que un ´´Bean´´ es un componente que puede usarse en el Procesador Experto, el cual encapsula la funcionalidad de elementos básicos

incluidos en sistemas (como CPU core, CPU on-chip, componentes periféricos ordenados independientemente, dispositivos virtuales, y los algoritmos del software puros) y proporciona el mando de estos dispositivos vía propiedades, métodos, y eventos.

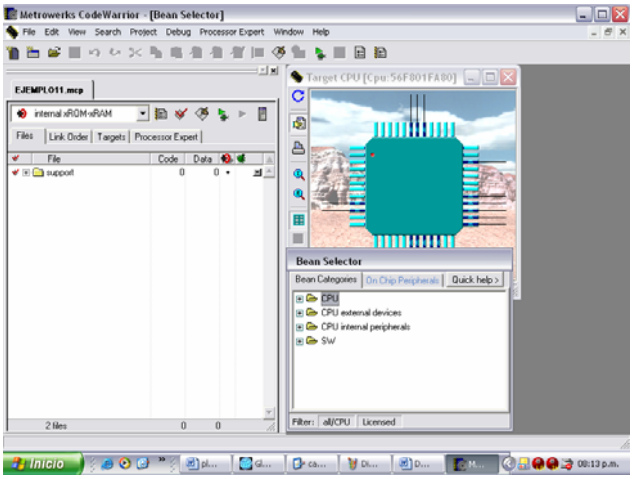


Figura A4 Ventana principal utilizando Procesador Experto

A Continuación se tiene que ejecutar el comando **Make**  ubicado en la parte superior de la ventana principal de ´´codewarrior´´ el cual crea varias carpetas, entre ellas la carpeta donde se creara el proyecto, como se aprecia en la siguiente figura. ´´Make´´ también se encarga de encontrar errores de programación y de posibles advertencias que existen en el programa que se este ejecutando.

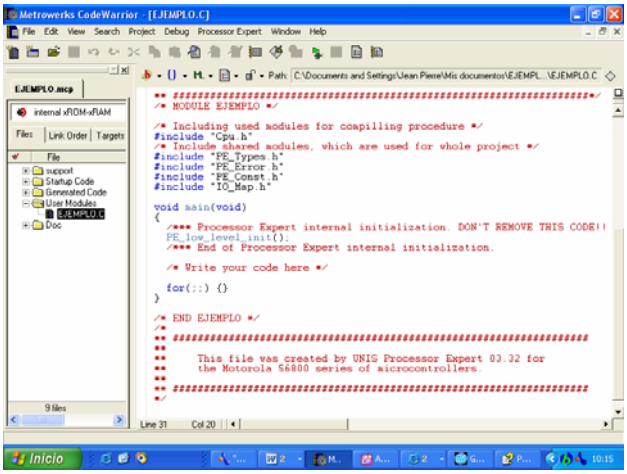


Figura A5 Archivo generado para empezar el proyecto

En la carpeta **User Modules** se crea por defecto el archivo con el nombre que que se asigno al proyecto, donde esta la función principal ´´main´´ con un esquema para editar programas en C, bajo procesador experto. Tal como aparece en la figura A6.

```
** ###################################################################*/
/* MODULE Ejemplo */

/* Including used modules for compilling procedure */
#include "Cpu.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

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

  /* Write your code here */

  for(;;) {}
}

/* END Ejemplo */
/*
** ###################################################################
**
**     This file was created by UNIS Processor Expert 03.32 for
**     the Motorola 56800 series of microcontrollers.
**
** ###################################################################
*/
```

Figura A6 Archivo en Lenguaje de programación en C.

Después de haber editado el programa se vuelve a ejecutar ´´Make´´ para ver los posibles errores y advertencias .Paso seguido se hace un ´´Debug´´  , este comando ubicado esta en la parte superior de la ventana principal de codewarrior, el cual hace que el programa este en modo ´´Run´´ y para asi simular el DSP o ya sea ver los resultados en el software de ´´codewarrior IDE´´.

´´Codewarrior´´ tiene la posibilidad de mostrar el tamaño en bytes del programa generado, tal como se puede apreciar en la figura A7 la cual indica el tamaño en bytes del programa realizado, esto aparece inmediatamente después de ejecutar el comando debug.

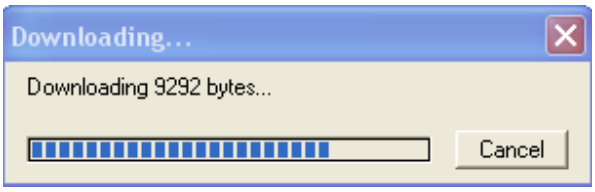


Figura A7 Tamaño en bytes del Programa

En la siguiente figura, se muestra la ventana de simulación de CodeWarrior. En ella se pueden observar el contenido de los registros, memoria, variables globales y locales y el código fuente en C y en ensamblador. Finalmente, solo queda grabar el programa en el dsp. Por defecto el puntero queda siempre en la inicialización interna del Procesador experto, la cual no se debe remover. Según se quiere el programa se puede ejecutar en diferentes formas, tal y como se explica a continuación.

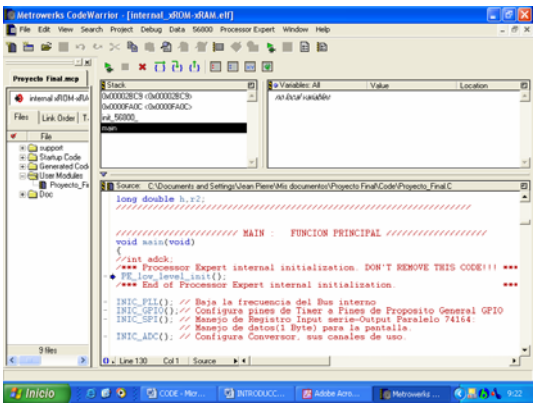


Figura A8 Panel frontal en Modo Run

Entre los comandos utilizados para el proyecto tenemos:

- El comando ´´RUN´´ se encarga de enviar el programa al DSP directamente.
- El comando ´´Kill Tread´´ termina la sesión de ejecución del programa completamente, llevándolo al programa original.
- El comando ´´Step Over´´ permite literalmente recorrer el código fuente, ejecutando cada declaración hecha en el programa.
- El comando ´´Step into´´ realiza un recorrido similar al comando Step Over pero con la diferencia que este entra a la declaración y la realiza paso a paso.
- El comando ´´Step Out´´ se encarga de ejecutar el resto de las rutinas en que valla el programa y se detiene la ejecución cuando termina de ejecutar todas las rutinas.

- El comando ´´breakpoints´´  es uno de los comandos mas importantes para saber si un programa realiza lo que tiene que hacer el coloca un punto en el que el programa se detiene para permitir detectar y eliminar errores.

Finalmente, la figura de abajo muestra el momento en que el programa se esta ejecutando y simulando al DSP.

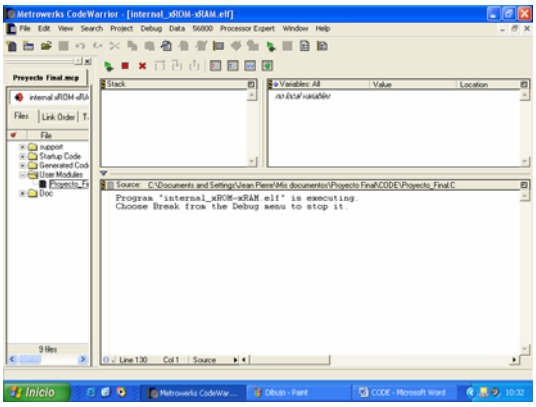


Figura A9 Ventana de Ejecución

Una forma eficiente de depurar los programas realizados en codewarrior es el Modo Simulador, el cual da soporte al uso de los periféricos internos y los periféricos externos de la familia de DSP`s de motorola. Permitiendo trabajar sin tener conectada la tarjeta del puerto JTAG y la tarjeta del DSP 56F801. A continuación se presenta en la figura de abajo la forma de trabajar en Modo Simulador.

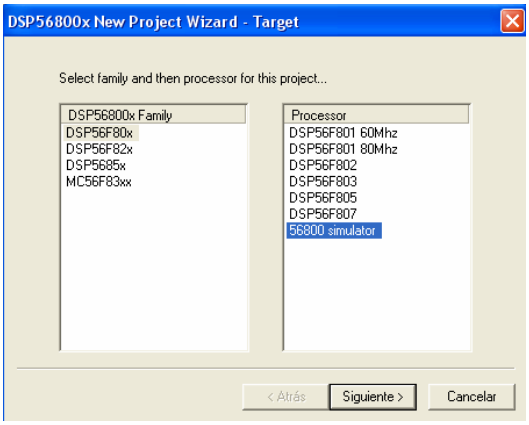


Figura A10 Escoger Modo Simulador

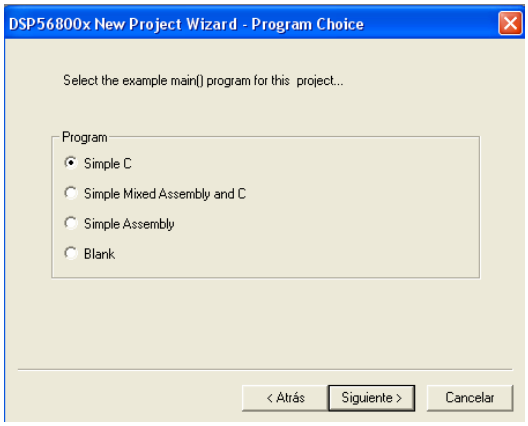


Figura A11 Escogencia Lenguaje de Programación

En la figura de abajo se puede observar cómo está estructurado el programa. Una primera ventana (superior izquierda) contiene los archivos que componen el proyecto que se están realizando. En otra ventana (lado derecho), podemos editar el archivo seleccionado.

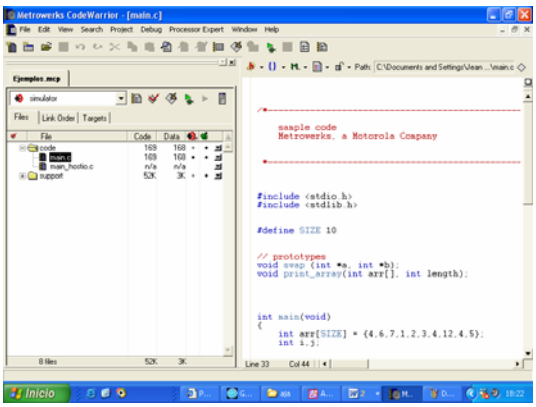


Figura A12 Ventana Principal de Modo Simulador

En la siguiente figura de abajo se muestra la ventana principal de codewarrior en Modo Simulador, en ella se ven tres ventanas principales, la primera ventana (superior izquierda) los archivos generados por la compilación, la segunda ventana (superior derecha) la cual contiene las variables que se crean en el proyecto y finalmente una tercera ventana (inferior derecha), donde esta el código escrito. Además se tiene una serie de menús con los que se puede configurar y depurar el proyecto. Todos los iconos que están alrededor de las ventanas son accesos directos a las funcionalidades de los menús. Los comandos utilizados son los mismos que fueron expuestos anteriormente.

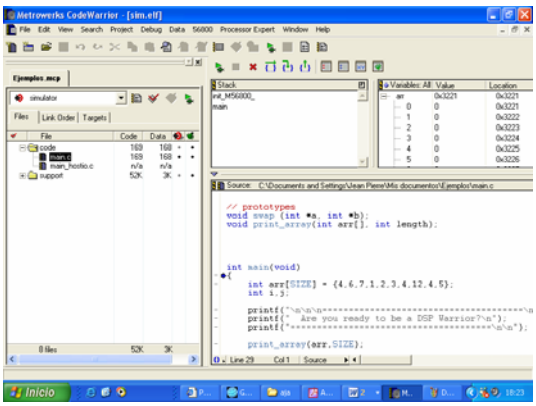


Figura A13 Modo Simulador

Paso seguido se le da RUN al programa para poder ver los resultados.

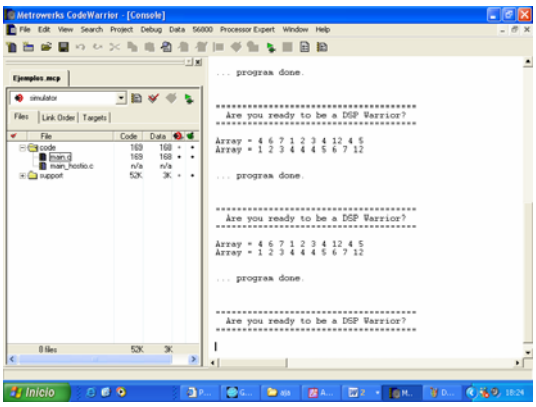


Figura A14 Ventana de simulación final

Por ultimo cuando no se tiene conectada la tarjeta al computador o no se ha ejecutado el DLL y se va a trabajar en Procesador experto, el programa le arroja el siguiente mensaje recordándole estos parámetros mínimos para trabajar.

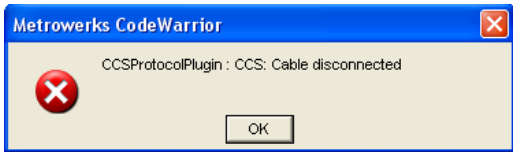


Figura A15 Cable desconectado o Protocolo no iniciado

ANEXO B DETERMINACION DE LA REGRESIÒN CIRCULAR

En este anexo se determina la regresión circular, partiendo de la ecuación de una circunferencia.

Siendo X y Y de la siguiente forma:

$$X = [x_1, x_2, x_3, ...x_i .....x_n]$$

$$Y = [y_1, y_2, y_3, ...y_i ....y_n]$$

Se tiene que la ecuación de una circunferencia con centro en C (h, 0) y radio R, es de la forma:

$$Y^2 + (x-h)^2 = R^2 \quad \text{Ò} \quad Y^2 = R^2 - (x_i - h)^2$$

Se plantea un problema de optimización.

Minimizando. Ecuación 1 $e = \sum_{i=1}^{n} \left(Y^2 - Y_i^2\right)^2$ $\therefore$ Error en el cuadrado de la variable vertical.

Reemplazando términos se obtiene que: $e = \sum_{i=1}^{n} \left[R^2 - (x_i - h)^2 - Y_i^2\right]^2$ Ecuación 2.

Derivando con *e* con respecto a *R*: $\frac{\partial e}{\partial R} = \sum_{i=1}^{n} 2\left[R^2 - \left(x_i - h\right)^2 - Y_i^{\,2}\right] * 2R = 0$

Ecuación 3

Utilizando las propiedades de las sumatorias da como resultado la expresión:

$$nR^2 - \sum_{i=1}^{n}\left(x_i^{\,2} - 2x_i^{\,2}h + h^2\right) - \sum_{i=1}^{n} Y_i^{\,2} = 0 \text{ Ecuación 4.}$$

Ecuación 5 después de derivar: $nR^2 + 2h\sum_{i=1}^{n} x_i - nh^2 = \sum_{i=1}^{n} x_i^{\,2} + \sum_{i=1}^{n} Y_i^{\,2}$ Ecuación 5.

Ahora se deriva a e con respecto a h: $\frac{\partial e}{\partial h} = \sum_{i=1}^{n} 2\left[R^2 - \left(x_i - h\right)^2 - Y_i^{\,2}\right] * 2\left(x_i - h\right)$=0

$$\sum_{i=1}^{n}\left[R^2 - x_i^{\,2} + 2hx_i - h^2 - Y_i^{\,2}\right] * \left(x_i - h\right) = 0 \text{ Ecuación 6}$$

Multiplicando término a término se tiene:

$$R^2\sum_{i=1}^{n} x_i - \sum_{i=1}^{n} x_i^{\,3} + 2h\sum_{i=1}^{n} x_i^{\,2} - h^2\sum_{i=1}^{n} x_i - \sum_{i=1}^{n} x_i Y_i^{\,2}$$
$$-nhR^2 + h\sum_{i=1}^{n} x_1^{\,2} - 2h^2\sum_{i=1}^{n} x_i + nh^3 + h\sum_{i=1}^{n} Y_i^{\,2} = 0 \text{ Ecuación 7}$$

La ecuación 7 es resultado de las operaciones realizadas anteriormente:

$$R^2\sum_{i=1}^{n} x_i - \sum_{i=1}^{n} x_i^{\,3} + 3h\sum_{i=1}^{n} x_i^{\,2} - 3h^2\sum_{i=1}^{n} x_i + nh^3 - \sum_{i=1}^{n} x_i Y_i^{\,2} - nhR^2 + h\sum_{i=1}^{n} Y_i^{\,2} = 0$$

Agrupando términos de la ecuación de arriba se obtiene la ecuación 8

$$R^2 * \left\{\sum_{i=1}^{n} x_i - hn\right\} + h * \left\{3\sum_{i=1}^{n} x_i^{\,2} + \sum_{i=1}^{n} y_i^{\,2}\right\} - h^2 * \left\{3\sum_{i=1}^{n} x_i\right\} + h^3 n = \sum_{i=1}^{n} x_i^{\,3} + \sum_{i=1}^{n} x_i Y_i^{\,2} \quad (8)$$

Reagrupando, se obtiene la ecuación 9.

$$\left(\sum_{i=1}^{n} x_i\right) * R^2 - nhR^2 + \left\{3\sum_{i=1}^{n} {x_1}^2 + \sum_{i=1}^{n} {Y_i}^2\right\} * h - \left\{3\sum_{i=1}^{n} x_i\right\} * h^2 + nh^3 - \sum_{i=1}^{n} {x_i}^3 - \sum_{i=1}^{n} x_i {Y_i}^2 = 0$$

Las expresiones resultantes después de derivar con respecto a R y h son las ecuaciones 5 y 8:

$$nR^2 + \left(2\sum_{i=1}^{n} x_i\right) * h - nh^2 = \sum_{i=1}^{n} {x_i}^2 + \sum_{i=1}^{n} {Y_i}^2 \quad \text{Ecuación 5}$$

Ecuación8

$$\left(\sum_{i=1}^{n} x_i\right) * R^2 - nhR^2 + \left\{3\sum_{i=1}^{n} {x_i}^2 + \sum_{i=1}^{n} {Y_i}^2\right\} * h - \left\{3\sum_{i=1}^{n} x_i\right\} * h^2 + nh^3 - \sum_{i=1}^{n} {x_i}^3 - \sum_{i=1}^{n} x_i {Y_i}^2 = 0$$

De la ecuación (5) se despeja R^2 dando la ecuación (A):

$$R^2 = h^2 - \left(\frac{2}{n}\sum_{i=1}^{n} x_i\right) * h + \frac{1}{n} * \left\{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} {Y_i}^2\right\} \qquad \text{Ecuación (A)}$$

La ecuación (A) en la ecuación (8),

$$\left(\sum_{i=1}^{n} x_i\right) * h^2 - \left[\frac{2}{n}\left(\sum_{i=1}^{n} x_i\right)^2\right] * h + \left[\frac{1}{n}\sum_{i=1}^{n} x_i\right] * \left[\sum_{i=1}^{n} {x_i}^2 + \sum_{i=1}^{n} {Y_i}^2\right] - nh^3 + \left(2\sum_{i=1}^{n} x_i\right) * h^2 - \left\{\sum_{i=1}^{n} {x_i}^2 + \sum_{i=1}^{n} {Y_i}^2\right\} * h +$$
$$\left\{3\sum_{i=1}^{n} {x_i}^2 + \sum_{i=1}^{n} {Y_i}^2\right\} * h - \left\{3\sum_{i=1}^{n} x_i\right\} * h^2 + nh^3 - \sum_{i=1}^{n} {x_i}^3 - \sum_{i=1}^{n} x_i {Y_i}^2 = 0$$

Después de varias manipulaciones matemáticas se obtiene la ecuación (C)

$$\left\{2\sum_{i=1}^{n} x_i^{\,2} - \frac{2}{n}\left(\sum_{i=1}^{n} x_i\right)^2\right\} * h = -\left\{\frac{1}{h}\sum_{i=1}^{n} x_i\right\} * \left\{\sum_{i=1}^{n} x_i^{\,2} + \sum_{i=1}^{n} Y_i^{\,2}\right\} + \sum_{i=1}^{n} x_i^{\,3} + \sum_{i=1}^{n} x_i Y_i^{\,2}$$

Ecuación C

Finalmente se despeja h de la ecuación (C) dando la expresión para la ecuación (D):

$$h = \frac{\sum_{i=1}^{n} x_i^{\,3} + \sum_{i=1}^{n} x_i Y_i^{\,2} - \left\{\frac{1}{n}\sum_{i=1}^{n} x_i\right\} * \left\{\sum_{i=1}^{n} x_i^{\,2} + \sum_{i=1}^{n} Y_i^{\,2}\right\}}{2\sum_{i=1}^{n} x_i^{\,2} - \frac{2}{n}\left(\sum_{i=1}^{n} x_i\right)^2}$$ Ecuación D.

Como resultado de la solución de la ecuación de la circunferencia con centro en C (h, 0) y radio R, se obtuvieron que las ecuaciones A y D son las soluciones.

Resultado final:

$$R^2 = h^2 - \left(\frac{2}{n}\sum_{i=1}^{n} x_i\right) * h + \frac{1}{n} * \left\{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} Y_i^{\,2}\right\}$$

Ecuación A

$$h = \frac{\sum_{i=1}^{n} x_i^{\,3} + \sum_{i=1}^{n} x_i Y_i^{\,2} - \left\{\frac{1}{n}\sum_{i=1}^{n} x_i\right\} * \left\{\sum_{i=1}^{n} x_i^{\,2} + \sum_{i=1}^{n} Y_i^{\,2}\right\}}{2\sum_{i=1}^{n} x_i^{\,2} - \frac{2}{n}\left(\sum_{i=1}^{n} x_i\right)^2}$$

Ecuación D

ANEXO C REGRESION SENOIDAL

En este anexo se determina la regresión senoidal.

Siendo X y Y de la siguiente forma:

$$X = [x_1, x_2, x_3, ...x_i .....x_n]$$

$$Y = [y_1, y_2, y_3, ...y_i ....y_n]$$

Se plantea la ecuación 1

$\overline{Y} = BcCos(WoX) + BsSen(WoX)$ Ecuación 1.

Se plantea un problema de optimización ecuación 2.

$$e = \sum_{i=1}^{n} [Yi - BcCos(WoXi) + BsSen(WoXi)]^2 \quad \text{Ecuación 2.}$$

Se deriva e con respecto a Bc, dando la ecuación 3

$$\frac{\partial e}{\partial Bc} = \sum_{i=1}^{n} 2[Yi - BcCos(WoXi) - BsSen(WoXi)] * [-Cos(WoXi)] = 0$$

La ecuación resultante de derivar la ecuación 3.

$$\sum_{i=1}^{n} \{YiCos(WoXi) - BcCos^2(WoXi) - BsSen(WoXi)\}Cos(WoXi) = 0 \quad \text{Ecuación 4}$$

Ahora se deriva e con respecto a Bs.

$$\frac{\partial e}{\partial Bs} = \sum_{i=1}^{n} 2[Yi - BcCos(WoXi) - BsSen(WoXi)][-Sen(WoXi)] = 0 \quad \text{Ecuación 4.1}$$

La ecuación resultante es la ecuación 5.

$$\sum_{i=1}^{n} [YiSen(WoXi) - BcSen(WoXi)Cos(WoXi) - BsSen^2(WoXi)] = 0 \quad \text{Ecuación 5.}$$

Haciendo uso de las propiedades de las sumatorias la ecuación resulta de la forma:

$$Bc\sum_{i=1}^{n} Cos^2(WoXi) + Bs * \frac{1}{2} * \sum_{i=1}^{n} Sen(2WoXi) = \sum_{i=1}^{n} YiCos(WoXi) \quad \text{Ecuación 6.}$$

Reagrupando términos se llega a una ecuación sencilla. Ecuación 7.

$$Bc * \frac{1}{2}\sum_{i=1}^{n} Sen(2WoXi) + Bs\sum_{i=1}^{n} Sen^2(WoXi) = \sum_{i=1}^{n} YiSen(WoXi)$$

Despejando Bc de la ecuación 7 resulta la ecuación 8.

$$Bc = \frac{\left\{\left(\sum_{i=1}^{n} YiCos(WoXi)\right) * \left(\sum_{i=1}^{n} Sen^2(WoXi)\right) - \left(\sum_{i=1}^{n} YiSen(WoXi)\right) * \left(\frac{1}{2} * \sum_{i=1}^{n} (2WoXi)\right)\right\}}{\left\{\left(\sum_{i=1}^{n} Cos^2(WoXi)\right) * \left(\sum_{i=1}^{n} Sen^2(WoXi)\right) - \frac{1}{4} * \left(\sum_{i=1}^{n} (2WoXi)\right)^2\right\}}$$

Ecuación 8

De igual forma que Bc, para Bs se aplican las mismas propiedades de sumatorias dando la ecuación final para Bs. La ecuación 9

$$Bs = \frac{\left\{\left(\sum_{i=1}^{n} Cos^2(WoXi)\right) * \left(\sum_{i=1}^{n} YiSen(WoXi)\right) - \frac{1}{2}\sum_{i=1}^{n} Sen(2WoXi) * \left(\sum_{i=1}^{n} YiCos(WoXi)\right)\right\}}{\left\{\left(\sum_{i=1}^{n} Cos^2(WoXi)\right) * \left(\sum_{i=1}^{n} Sen^2(WoXi)\right) - \frac{1}{4} * \left(\sum_{i=1}^{n} (2WoXi)\right)^2\right\}}$$

Ecuación 9

ANEXO D. CODIGO GENERAL DEL PROGRAMA

```
////////////////////////////////////////////////////////////////////
/* Including used modules for compilling procedure */
#include "Cpu.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

#include <math.h> //Libreria Matématica
////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////
// Constantes guardadas en Memoria Ram de Datos.

#define Alto        1
#define Bajo        0
#define Derecha     0x4C
#define Izquierda   0x4D
#define Arriba      0x4E
#define Abajo       0x4F
#define P_Grafico   0x1000
#define pi          3.14159265
#define TOLER       0.0001
////////////////////////////////////////////////////////////////////
// Constantes guardadas en Memoria Flash de Datos.
//---------------------- Vectores de texto ----------------------
const byte UNI[]={"UNIVERSIDAD INDUSTRIAL"};
const byte SAN[]={" DE SANTANDER"};
const byte ESC[]=
{"Escuela de Ingenierias ElectricaElectronica y Telecomunicaciones"};
const byte PRO[]={"*** PROYECTO MIE ***"};
const byte CEM[]={"GRUPO  >> CEMOS <<"};
const byte JEA[]={"JEAN PIERRE AMARIS D."};
const byte JOS[]={"JOSE ALBERTO LOPEZ P."};
const byte INI[]={"INICI"};
const byte ENT[]={"ENTER"};
const byte CON[]={"CNTRL"};
const byte DAT[]={"DATO"};
const byte NYQ[]={"NYQUIST"};
const byte GRA[]={"GRAFI"};
const byte BMA[]={"B MA"};
const byte BFA[]={"B FAS"};
```

```
const byte PRE[]={"PRESIONE -ENTER- PARA"};
const byte ADQ[]={"INICIAR LA ADQUISICION DE DATOS "};
const byte EST[]={"ADQUIRIENDO DATOS..."};
const byte POR[]={"POR FAVOR ESPERE"};
const byte FIN[]={"FINALIZADA LA ADQUISICION"}; /
const byte PUE[]={"OPERACION MANUAL"};
const byte PRG[]={"PRESENTACION DE GRAFICOS"};
const byte IBM[]={"ESCALA EJE Y = 10 grados"};
const byte GUA[]={"GUARD:Ultimo prodeso realizado."};
const byte SIM[]={"SIMUL:Simulacion de datos."};
const byte NUE[]={"NUEVO:Nueva toma de datos."};
const byte DEC[]={"ORIGEN: FRECUENCIA 0.1HZ, +DEC"};
const byte MAG[]={"MAGNITUD MAXIMA = "};
const byte ABS[]={"(abs)"};
const byte FAS[]={"FASE EN ATRASO, MAX 90 grados"};
const byte BOD[]={"G BODE"};
const byte RAD[]={"Radio = "};
const byte RAD2[]={"Radio2 = "};
const byte CEN[]={"Centro = "};
const byte ESCA[]={"Escala ejes Y y X = "};
const byte OHM[]={" Ohmios"};
const byte RES1[]={"Resistencia Rp ="};
const byte RES2[]={"Resistencia Rs ="};
const byte CAP[]={"Capacitancia Cd ="};
const byte MIC[]={" uF"};
const byte Im[]={"Im"};
const byte Re[]={"Re"};
const byte Mg[]={"Mg"};
const byte fr[]={"fr"};
const byte An[]={"An"};

const byte PUNTOS[] = {0x80,0x40,0x20,0x10,0x8,0x4,0x2,0x1};

const float RESIS_GAN[]={3726.872247,372.6872247,18000,1800};

const byte LOGARI1[]= {0x82,0x24,0x95,0x41,0x12,0x4A,0xA0,0x89,0x25,
                       0x50,0x44,0x92,0xA8,0x22,0x49,0x54,0x11,0x24,
                       0xAA,0x08,0x92,0x55};

const byte LOGARI2[]= {0x80,0x00,0x00,0x40,0x00,0x00,0x20,0x00,0x00,
                       0x10,0x00,0x00,0x08,0x00,0x00,0x04,0x00,0x00,
                       0x02,0x00,0x00,0x01};

const unsigned int RETAR_MUES[]={0,0,0,5,18,23,38,49,58,68,77,109,
                    200,235,340,415,605,703,800,1130,2000,2330,
```

```
3290,3950,6560,7620,8705,12180,21790,24950,
35050,42140,733,838,961,1340,2410,2650,3878,
4687,6968,8082,9130,12709,22824,26142,36744,
44385};

const double
FRECUENCIAS[]={62200,54750,48270,34950,19620,17180,12270,
    10160,10610,9250,8157,5870,3286,2874,2049,1698,950.6,
    826.4,729.9,524.1,293.1,256.4,182,5,150.9,97.28,84.18,
    74.29,53.19,29.62,25.77,18.45,15.08,8.881,7.728,6.803,
    4.909,2.732,2.388,1.724,1.420,0.8929,0.7576,0.6727,
    0.4902,0.2688,0.2358,1.1678,0.1397};

const float DELTA[]={2E-6,2.7E-6,2.7E-6,3.675E-6,6.25E-6,7.25E-6,
    1.02E-5,1.235E-5,1.425E-5,1.6125E-5,1.7875E-5,2.4125E-5,
    4.2E-5,4.9E-5,6.95E-5,8.45E-5,1.22E-4,1.4125E-4,1.6E-4,
    2.25E-4,3.95E-4,4.6E-4,6.5E-4,7.8E-4,1.2875E-3,1.5E-3,
    1.7125E-3,2.4E-3,4.3E-3,4.9E-3,6.9E-3,8.3E-3,
    1.4875E-2,1.72E-2,1.9625E-2,2.725E-2,4.9E-2,5.4E-2,
    7.9E-2,9.5E-2,0.4125,0.16375,0.185,0.2575,0.465,
    0.53,0.745,0.905};

const byte UIS_P[]=
  { 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x80,0x00,0x00,0x00,0x00,0x01
   ,0x80,0x01,0xCF,0xF3,0x00,0x01,0x80,0x07,0xCF,0xF3,0xE0,0x01
   ,0x80,0x1F,0xCF,0xF3,0xF8,0x01,0x80,0x1F,0xCF,0xF3,0xFE,0x01
   ,0x81,0x9F,0xCF,0xF3,0xFF,0x81,0x83,0x9F,0xCF,0xF3,0xFF,0xC1
   ,0x87,0x9F,0xCF,0xF3,0xFF,0xE1,0x8F,0x9F,0xCF,0xF3,0xFF,0xF1
   ,0x9F,0x9F,0xCF,0xF3,0xFF,0xF9,0xBF,0x9F,0xCF,0xF3,0xFF,0xFD
   ,0xBF,0x9F,0xCF,0xF3,0xFF,0xFD,0xBF,0x9F,0xCF,0xF3,0xFF,0xFD
   ,0xBF,0x9F,0xCF,0xF3,0xF9,0xFD,0xBF,0x9F,0xCF,0xF3,0xF9,0xFD
   ,0xBF,0x9F,0xCF,0xF3,0xF9,0xFD,0xBF,0x9F,0xCF,0xF3,0xF9,0xFD
   ,0xBF,0x9F,0xCF,0xF3,0xF9,0xFD,0xBF,0x9F,0xCF,0xF3,0xF9,0xFD
   ,0xBF,0x9F,0xCF,0xF3,0xF9,0xFD,0xBF,0x9F,0xCF,0xF3,0xF9,0xFD
   ,0xBF,0x9F,0xCF,0xF3,0xF8,0x01,0xBF,0x9F,0xCF,0xF3,0xF8,0x01
   ,0xBF,0x9F,0xCF,0xF3,0xFF,0xFD,0xBF,0x9F,0xCF,0xF3,0xFF,0xFD
   ,0xBF,0x9F,0xCF,0xF3,0xFF,0xFD,0xBF,0x9F,0xCF,0xF3,0xFF,0xFD
   ,0xBF,0x9F,0xCF,0xF3,0xFF,0xFD,0xBF,0x9F,0xCF,0xF3,0xFF,0xFD
   ,0xBF,0x9F,0xCF,0xF3,0xFF,0xFD,0xBF,0x9F,0xCF,0xF3,0xFF,0xFD
   ,0xBF,0x9F,0xCF,0xF0,0x01,0xFD,0xBF,0x9F,0xCF,0xF0,0x01,0xFD
   ,0xBF,0x9F,0xCF,0xF3,0xF9,0xFD,0xBF,0x9F,0xCF,0xF3,0xF9,0xFD
   ,0xBF,0x9F,0xCF,0xF3,0xF9,0xFD,0xBF,0x9F,0xCF,0xF3,0xF9,0xFD
   ,0xBF,0x9F,0xCF,0xF3,0xF9,0xFD,0xBF,0x9F,0xCF,0xF3,0xF9,0xFD
   ,0xBF,0x9F,0xCF,0xF3,0xF9,0xFD,0xBF,0x9F,0xCF,0xF3,0xF9,0xFD
   ,0xBF,0x9F,0xCF,0xF3,0xF9,0xFD,0xBF,0x9F,0xCF,0xF3,0xF9,0xFD
```

```
      ,0xBF,0x9F,0xCF,0xF3,0xF9,0xFD,0xBF,0x9F,0xCF,0xF3,0xFF,0xFD
      ,0xBF,0xFF,0xCF,0xF3,0xFF,0xFD,0xBF,0xFF,0xCF,0xF3,0xFF,0xFD
      ,0xBF,0xFF,0xCF,0xF3,0xFF,0xFD,0xBF,0xFF,0xCF,0xF3,0xFF,0xFD
      ,0x9F,0xFF,0xCF,0xF3,0xFF,0xF9,0x8F,0xFF,0xCF,0xF3,0xFF,0xF1
      ,0x87,0xFF,0xCF,0xF3,0xFF,0xE1,0x83,0xFF,0xCF,0xF3,0xFF,0xC1
      ,0x81,0xFF,0xCF,0xF3,0xFF,0x81,0x80,0xFF,0xCF,0xF3,0xFF,0x01
      ,0x80,0x3F,0xCF,0xF3,0xFC,0x01,0x80,0x1F,0xCF,0xF3,0xF8,0x01
      ,0x80,0x07,0xCF,0xF3,0xE0,0x01,0x80,0x00,0xCF,0xF3,0x00,0x01
      ,0x80,0x00,0x07,0xE0,0x00,0x01,0x80,0x00,0x00,0x00,0x00,0x01
      ,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF};

const byte UIS[]=
   {127,255,255,254,192,  0,  0,  3,159,255,255,249,176,  0,  0, 13,
    160, 27,216,  5,160,123,222,  5,160,123,223,  5,161,123,223,133,
    163,123,223,197,167,123,223,229,175,123,223,245,175,123,223,245,
    175,123,223,245,175,123,222,245,175,123,222,245,175,123,222,245,
    175,123,222,245,175,123,222,245,175,123,222,245,175,123,222,245,
    175,123,222,  5,175,123,223,245,175,123,223,245,175,123,223,245,
    175,123,223,245,175,123,223,245,175,123,223,245,175,123,192,245,
    175,123,222,245,175,123,222,245,175,123,222,245,175,123,222,245,
    175,123,222,245,175,123,222,245,175,123,222,245,175,123,222,245,
    175,251,223,245,175,251,223,245,175,251,223,245,167,251,223,229,
    163,251,223,197,161,251,223,133,160,251,223,  5,160,123,222,  5,
    160, 27,216,  5,176,  0,  0, 13,159,255,255,249,192,  0,  0,  3,
    127,255,255,254};

const byte TRIANGULO1[]=
                {0xFF,0xFF,0x7F,0x3F,0x1F,0x0F,0x07,0x03,0x01};
const byte TRIANGULO2[]=
                {0xFF,0xFF,0xFE,0xFC,0xF8,0xF0,0xE0,0xC0,0x80};
////////////////////////////////////////////////////////////////////////
// Funciones declaradas, necesario en el lenguaje C, para que sean
// reconocidas en el momento de ser utilizadas.
void INIC_PLL(byte tiempo);
void INIC_GPIO(void);
void INIC_SPI(void);
void INIC_LCD(void);
void INIC_ADC(void);
void PORTADA(void);
void LOGO_UIS(void);
void SIMUL(void);
void PRESEN_DATOS(void);
void FFT(float adc_volt[8],float adc_crte[8],byte j);
void Reg_Circular(float amp_cte[48],float amp_vol[48]);
void IMPRIMIR(const byte texto[],int posicion,byte tamaño,byte direc);
```

```
void IMPRIMIR_V(byte val,int posicion,byte tamaño,byte direc);
void GRAF_NYQUI(void);
void GRAF_BODE_MAG(void);
void GRAF_BODE_FAS(void);
byte GRAFICA(byte nivel_a,byte NIVEL[8],byte SALIDA[78],byte i);
void GRAFICA_EJE_LOG(void);
void TEXTOS(void);
void GRAFICA_EJES(void);
void SOMBRA_CONTROLES(void);
void LIMPIAR_TEXTO(void);
void LIMPIAR_GRAFICO(void);
void LIMPIAR_GRAFICO_P(void);
void LIMPIAR_INFOR(void);
void GRAFICAR_MARCOS(void);
void ESCRIBIR_COMANDO(byte Comando);
void ESCRIBIR_DATO(byte Dato);
void MODO_A0(bool estado);
void ESCRITURA_WR(bool estado);
void TEXTO(char *cadena);
void RETARDO_ADC(unsigned int tiempo);
void RETARDO_ADC_L(unsigned int tiempo);
void MICROSEC(word tiempo);
void SEGUNDO(word  tiempo);
void POS_CURSOR(word Pos);
void MOV_CURSOR(word Mov);
void DES_NUM(float num);
void ENTER_ESC(void);
void SEGUIR_ESC(void);
bool CNTRL(void);
bool MENU(void);
bool ENTER(void);
void OBTEN_ADC(void);
byte Gan_Cte(void);
float sin(float x);
float cos(float x);
long double atan(float x);
long double fac(char x);
long double pot(float val,char x);
float sqrt(float x);
//////////////////////////////////////////////////////////////////
// Variables Generales, pueden ser utilizadas dentro de cualquier
// función, y puede ser modificados.
volatile float amplvolt,amplcrte,desfase,centro,radio2,radio1;
volatile double cd,Rs,Rp;
```

```
////////////////////////////////////////////////////////////////////
// Función main(): Punto de inicio para la ejecución del programa
// desde aqui se ordenan las demas funciones.
void main(void)
{
  PE_low_level_init();

INIC_PLL(6);
INIC_SPI();
INIC_GPIO();
INIC_LCD();
PORTADA();

for(;;)
{
 if(!CNTRL()){while(!getRegBit(GPIO_A_DR,D2)){} goto CONTI;}

 if(!MENU()) {while(!getRegBit(GPIO_B_DR,D6)){} goto CONTI;}

 if(!ENTER()){while(!getRegBit(GPIO_B_DR,D7)){} goto CONTI;}
}

CONTI:

LIMPIAR_TEXTO();
LIMPIAR_GRAFICO();
GRAFICAR_MARCOS();
SOMBRA_CONTROLES();
LOGO_UIS();

PULSO3:
LIMPIAR_TEXTO();
IMPRIMIR(SIM,0x181,26,Derecha);
IMPRIMIR(NUE,0x1A1,26,Derecha);
IMPRIMIR(SIM,0x1E4,5,Derecha);
IMPRIMIR(NUE,0x1F7,5,Derecha);
for(;;)
{
  if(!CNTRL())
   {
   while(!getRegBit(GPIO_A_DR,D2)){}
   LIMPIAR_TEXTO();
   goto SIMULADOS;
   }
```

```
  if(!ENTER())
   {
   while(!getRegBit(GPIO_B_DR,D7)){}
   LIMPIAR_TEXTO();
   goto NUEVA;
   }
}

SIMULADOS:
SIMUL();
goto MANUAL;


NUEVA:

LIMPIAR_INFOR();
IMPRIMIR_V(0x20,0x1F7,5,Derecha);
IMPRIMIR(EST,0x181,20,Derecha);
IMPRIMIR(POR,0x1AF,17,Derecha);

OBTEN_ADC();

INIC_PLL(6);
LIMPIAR_INFOR();
IMPRIMIR(FIN,0x181,25,Derecha);
IMPRIMIR(PUE,0x1B0,17,Derecha);

MANUAL:

IMPRIMIR(GRA,0x1E4,5,Derecha);
IMPRIMIR(DAT,0x1EE,4,Derecha);
IMPRIMIR(INI,0x1F7,5,Derecha);

for(;;)
{
  if(!CNTRL())
   {
   while(!getRegBit(GPIO_A_DR,D2)){}
   LIMPIAR_GRAFICO_P();LIMPIAR_TEXTO();
   goto PULSO1;
   }

  if(!MENU())
   {
   while(!getRegBit(GPIO_B_DR,D6)){}
```

```
   LIMPIAR_GRAFICO_P();LIMPIAR_TEXTO();
   goto PULSO2;
   }

  if(!ENTER())
   {
   while(!getRegBit(GPIO_B_DR,D7)){}
   LIMPIAR_GRAFICO_P();
   goto PULSO3;
   }
}

PULSO2:
PRESEN_DATOS();
goto MANUAL;

PULSO1:
IMPRIMIR(PRG,0x181,24,Derecha);
IMPRIMIR(NYQ,0x1E4,5,Derecha);
IMPRIMIR(BMA,0x1EE,4,Derecha);
IMPRIMIR(BFA,0x1F7,5,Derecha);

for(;;)
{
  if(!CNTRL())
   {
   while(!getRegBit(GPIO_A_DR,D2)){}
   LIMPIAR_GRAFICO_P();LIMPIAR_TEXTO();
   goto PULSO4;
   }

  if(!MENU())
   {
   while(!getRegBit(GPIO_B_DR,D6)){}
   LIMPIAR_GRAFICO_P();LIMPIAR_TEXTO();
   goto PULSO5;
   }

  if(!ENTER())
   {
   while(!getRegBit(GPIO_B_DR,D7)){}
   LIMPIAR_GRAFICO_P();LIMPIAR_TEXTO();
   goto PULSO6;
   }
}
```

```
PULSO4:
GRAF_NYQUI();
goto MANUAL;

PULSO5:
GRAF_BODE_MAG();
goto MANUAL;

PULSO6:
GRAF_BODE_FAS();
goto MANUAL;
}

////////////////////////////////////////////////////////////////////
// Función INIC_PLL2: Configura la frecuencia del bus interno del DSP
// a 28MHz, que se utiliza para todo lo relacionado con el manejo de
// la pantalla Gráfica.
void INIC_PLL(byte tiempo)
{
  clrRegBit(PLLCR, PRECS);
  while(getRegBit(PLLSR, PRECS)){}
  setRegBit(PLLCR, PLLPD);
  setReg(PLLDB, tiempo);
  setRegBit(PLLCR, LCKON);
  clrRegBit(PLLCR, PLLPD);
  while(!getRegBit(PLLSR, LCK0)){}
  setRegBitGroup(PLLCR, ZSRC, 2);
}

////////////////////////////////////////////////////////////////////
// Función INIC_GPIO(): Es llamada para que configure los pines del
// DSP dedicados(SPI.SCI,TRM y XTAL), para un uso como proposito
// general.
void INIC_GPIO(void)
{
 setReg(GPIO_A_PUR,0x04);
 setReg(GPIO_B_PUR,0xCC);
 setReg(GPIO_A_DDR,0x03);
 setReg(GPIO_B_DDR,0x03);
 setReg(GPIO_A_PER,0x00);
 setReg(GPIO_B_PER,0x30);
 setRegBit(GPIO_B_DR,D0);
 setRegBit(GPIO_B_DR,D1);
}
```

```
////////////////////////////////////////////////////////////////////
// Función SPI(): Prepara el periférico de comunicación serial SPI
// para la comunicación en forma de maestro con el registro de
// desplazamiento entrada serie salida paralela,74LS164, que es el
// encargado de enviar datos a la pantalla.
void INIC_SPI(void)
{
 setReg(SPSCR,216);
 setReg(SPDSR,0xF);
 setRegBit(SPSCR,SPE);
}

////////////////////////////////////////////////////////////////////
// Función INIC_LCD(): Configura y prepara la pantalla, con los datos
// particulares de la LCD HYUNDAI HG25601-C con controlador SED1330F.
void INIC_LCD(void)
{
ESCRITURA_WR(Alto);
ESCRIBIR_COMANDO(0x40);  // Inicio de comando
 ESCRIBIR_DATO(0x30);// M0: CG ROM Interna
                        // M1: CG RAM es 32 caracteres maximo
                     // M2: 8 lineas por caracter
                        // W/S: Simple - Panel drive
 ESCRIBIR_DATO(0x87);// FX: Tamaño Horizontal del caacter = 8 pixeles
 ESCRIBIR_DATO(0x07);// FY : .111: ALTO DE CARACTER
                       // FY: Tamaño Vertical del Caracter = 8 pixels
 ESCRIBIR_DATO(0x4F);//AREA DE DIBUJO 64 por linea
 ESCRIBIR_DATO(0x52); //TC/R: Total rango de direcciones por linea = 82
 ESCRIBIR_DATO(0x7F); // L/F: 128 lineas de display
 ESCRIBIR_DATO(0x20); //APL:Tamaño de Pantalla Virtual.
 ESCRIBIR_DATO(0x00); //APH:   32 direcciones
 //--COMANDOS DE DESPLAZAMIENTO ------------------------------------
ESCRIBIR_COMANDO(0x44); //SCROLL: DESPLAZAMIENTO
//**** PARAMETROS PRIMERA PANTALLA
 ESCRIBIR_DATO(0x00);  // Primer bloque inicia direccon 0x000h
 ESCRIBIR_DATO(0x00); // PAGINA DE TEXTO
 ESCRIBIR_DATO(0x7F);  // desplega 64 lineas el primer bloque
//**** PARAMETROS SEGUNDA PANTALLA
 ESCRIBIR_DATO(0x00); // Segundo inicia 0x1000h direccion
 ESCRIBIR_DATO(0x10); // PAGINA GRAFICA
 ESCRIBIR_DATO(0x7F); // 64 lineas el segundo bloque
//**** PARAMETROS TERCERA PANTALLA
 ESCRIBIR_DATO(0x00);
 ESCRIBIR_DATO(0x20);
```

```
 //---- PUNTO DE DESPLAZAMIENTO HORIZONTAL --------------------------
ESCRIBIR_COMANDO(0x5A);
 ESCRIBIR_DATO(0x00);
//----- OVERLAY COMMAND ----------------------------------------------
ESCRIBIR_COMANDO(0x5B); //MODO DE LA PANTALLA
TEXTO/GRAFICA
 ESCRIBIR_DATO(0x01);
//------------------
LIMPIAR_TEXTO(); // LIMPIA PANTALLA DE TEXTO
LIMPIAR_GRAFICO(); // LIMPIA LA PANTALLA GRAFICA
//--- FORMA DEL CURSOR ---------------------------------------------
ESCRIBIR_COMANDO(0x5D);
 ESCRIBIR_DATO(0x00);
 ESCRIBIR_DATO(0x81);
//---- COMANDO ON/OFF-----------------------------------------------
ESCRIBIR_COMANDO(0x59); // DISPLAY ON
  ESCRIBIR_DATO(0x56);
}

//////////////////////////////////////////////////////////////////////
// Función INIC_ADC: Configura el conversor interno del DSP, para que
// tome datos de los canales ANA0 y ANA4 en forma simultanea, ademas
// se configura a su maxima frecuencia de conversión.(588kHz)
void INIC_ADC(void)
{
setReg(ADCA_ADCR1,5);
setReg(ADCA_ADCR2,0);
setReg(ADCA_ADSDIS,0xCC);
INIC_PLL(19);
}

//////////////////////////////////////////////////////////////////////
// Funcion PORTADA: Encargada de hacer la presentación de la UIS y los
// realizadores del proyecto,no tiene argumentos, no devuelve valores;
// esta función solo se implementa en el encendido de la pantalla.
void PORTADA(void)
{
 byte i,j=0,x;
 word Direccion=0x100D;

 IMPRIMIR(UNI,0x105,23,Derecha);
 IMPRIMIR(SAN,0x129,14,Derecha);
 IMPRIMIR(ESC,0x140,64,Derecha);
 IMPRIMIR(CEM,0x187,18,Derecha);
 IMPRIMIR(PRO,0x1A6,20,Derecha);
```

```
 IMPRIMIR(JEA,0x1C6,21,Derecha);
 IMPRIMIR(JOS,0x1E6,21,Derecha);

 for (i=1;i<=63;i++)
 {
   POS_CURSOR(Direccion); ESCRIBIR_COMANDO(0x42);
   for (x=1;x<=6;x++){ESCRIBIR_DATO(UIS_P[j]);j++;}
   Direccion=Direccion+0x20;
 }

}

////////////////////////////////////////////////////////////////////
// Función LOGO_UIS:Presenta en pantalla el logo de la Universidad en
// el tamaño pequeño, siendo el que aparece junto al cuadro grafico
// que visualiza las funciones; por si sola se posiciona en ese punto.
void LOGO_UIS(void)
{
  byte i,x,j=0;
  word Direccion;
  Direccion=0x101C;
    MOV_CURSOR(Derecha);
    for (i=1;i<=49;i++)
     {POS_CURSOR(Direccion); ESCRIBIR_COMANDO(0x42);
     for (x=1;x<=4;x++) {ESCRIBIR_DATO(UIS[j]);j++;}
     Direccion=Direccion+0x20; }
}

////////////////////////////////////////////////////////////////////
// Función SIMUL: Encargada de darle valores comunes a los parametros
// de una Celda Elctroquímica, con el fin de visualizar el proceso
// gráfico que se hace.
void SIMUL(void)
{
float rr;
cd=100e-6;Rs=10;Rp=100;
radio1=Rp/2;
centro= radio1+Rs;
}

////////////////////////////////////////////////////////////////////
// Función PRESEN_DATOS(): Despliega los valos de los parametros de
// la celda Electroquímica
void PRESEN_DATOS(void)
{
```

```
 float Cd;byte i;
 Cd=cd/(1e-6);

IMPRIMIR(RES1,0x161,16,Derecha);DES_NUM(Rp);
for(i=0;i<=6;i++) ESCRIBIR_DATO(OHM[i]);

IMPRIMIR(RES2,0x181,16,Derecha);DES_NUM(Rs);
for(i=0;i<=6;i++) ESCRIBIR_DATO(OHM[i]);

IMPRIMIR(CAP,0x1A1,17,Derecha);DES_NUM(Cd);
for(i=0;i<=2;i++) ESCRIBIR_DATO(MIC[i]);
}

////////////////////////////////////////////////////////////////////
// Función OBTEN_ADC: Realiza el proceso de adquisición, ajustando
// retardos para muestreo, que dependen de la frecuencia a la cual se
// este adquiriendo, tambien se encarga de llamar a las funciones que
// necesite para calcular los parametros de la celda.
void OBTEN_ADC(void)
{
 byte i,j=0,k,posicion; float GanVol,GanCte,mayor=0;
 int adc_volt[8],adc_crte[8];
 float real[48],imag[48],vol,cor,adc_volt1[8],adc_crte1[8];
 byte R_FASE[]={9,4,4,4,4,4,4,20,20,15,20,25,30,38,40,40};

GanVol=20;  /// Ganancias de Amplificación
GanCte=0.5;

INIC_ADC();
ENTER_ESC(); SEGUNDO(15);
j=0;
 for(i=0;i<=7;i++) //Primera Muestra
 {
setRegBit(ADCA_ADCR1,START);
while(getRegBit(ADCA_ADSTAT,CIP)){}
adc_volt[i]=getReg(ADCA_ADRSLT1);
adc_crte[i]=getReg(ADCA_ADRSLT5);
}
for(i=0;i<=7;i++)
{
vol=adc_volt[i];
cor=adc_crte[i];
adc_volt1[i]=((9.15751e-5)*vol-1.5); //(1.5*vol/16384-1.5)
adc_crte1[i]=((9.15751e-5)*cor-1.5); //(1.5*cor/16384-1.5)
}
```

```
FFT(adc_volt1,adc_crte1,j);
amplcrte=amplcrte/(RESIS_GAN[Gan_Cte()]*GanCte);
amplvolt=amplvolt/GanVol;
real[j]=(amplvolt/amplcrte)*cos(desfase);
imag[j]=(amplvolt/amplcrte)*sin(desfase);

SEGUNDO(3);
SEGUIR_ESC();
SEGUNDO(13);

for(j=1;j<=31;j++)// muestras de 2 a 32 frecuencia
{

SEGUIR_ESC();SEGUNDO(3);
if(j==8|j==16|j==24)SEGUNDO(9);
for(i=0;i<=7;i++)
{
setRegBit(ADCA_ADCR1,START);
while(getRegBit(ADCA_ADSTAT,CIP)){}
adc_volt[i]=getReg(ADCA_ADRSLT1);
adc_crte[i]=getReg(ADCA_ADRSLT5);
RETARDO_ADC(RETAR_MUES[j]);
}

for(i=0;i<=7;i++) //Conversión a nivel de Voltaje
{
vol=adc_volt[i];
cor=adc_crte[i];
adc_volt1[i]=((9.15751e-5)*vol-1.5);  //(1.5*vol/16384-1.5)
adc_crte1[i]=((9.15751e-5)*cor-1.5);  //(1.5*cor/16384-1.5)
}
FFT(adc_volt1,adc_crte1,j);
amplcrte=amplcrte/(RESIS_GAN[Gan_Cte()]*GanCte);
amplvolt=amplvolt/GanVol;
real[j]=amplvolt/amplcrte*cos(desfase);
imag[j]=amplvolt/amplcrte*sin(desfase);

SEGUIR_ESC();SEGUNDO(12); // pulso de fase
}

for(j=32;j<=47;j++) // muestras de 33 a 48 frecuencia
{

SEGUIR_ESC();SEGUNDO(R_FASE[j-32]); // Pulso volt
```

```
for(i=0;i<=7;i++)
{
setRegBit(ADCA_ADCR1,START);
while(getRegBit(ADCA_ADSTAT,CIP)){}
adc_volt[i]=getReg(ADCA_ADRSLT1);
adc_crte[i]=getReg(ADCA_ADRSLT5);
RETARDO_ADC_L(RETAR_MUES[j]);
}

for(i=0;i<=7;i++) //Conversión a nivel de Voltaje
{
vol=adc_volt[i];
cor=adc_crte[i];
adc_volt1[i]=((9.15751e-5)*vol-1.5); //(1.5*vol/16384-1.5)
adc_crte1[i]=((9.15751e-5)*cor-1.5); //(1.5*vol/16384-1.5)
}
FFT(adc_volt1,adc_crte1,j);
amplcrte=amplcrte/(RESIS_GAN[Gan_Cte()]*GanCte);
amplvolt=amplvolt/GanVol;
real[j]=amplvolt/amplcrte*cos(desfase);
imag[j]=amplvolt/amplcrte*sin(desfase);

SEGUIR_ESC();SEGUNDO(12); // pulso de fase
}//fin de 33 a 48

Reg_Circular(real,imag);

Rs=centro-radio1;
Rp=2*radio1;

mayor=imag[0];
posicion=0;

for(i=1;i<=47;i++)
 if(imag[i]>mayor)
  {mayor=imag[i];
   posicion=i;}

cd=sqrt(Rp/(real[posicion]-Rs)-1)/(Rp*6.2831853*FRECUENCIAS[posicion]);

}

//////////////////////////////////////////////////////////////////////
// Función GRAF_NYQUI(): Toma los valores de centro y radio, los
// proporciona a valores dentro del rango de graficos de la pantalla
```

```
// y con la ecuación de la circunferencia realiza el grafico de un
// semicirculo caracteristico de los diagramas de Nyquist.
void GRAF_NYQUI(void)
{
  byte NIVEL[8],i,j;
  byte SALIDA[78],x=0,pixel=0,nivel_a=0;
  float fac_radio,radio,centro_gra,muestra,conv,d_conv,escx,
        r2_g,cc_g;

TEXTOS();
GRAFICA_EJES();

IMPRIMIR(Im,0x11E,2,Derecha);//y
IMPRIMIR(Re,0x13E,2,Derecha);//x

IMPRIMIR(NYQ,0x42,7,Abajo);

IMPRIMIR(CEN,0x181,9,Derecha);DES_NUM(centro);
IMPRIMIR(RAD,0x161,8,Derecha);DES_NUM(radio1);

radio=72;
fac_radio=72/radio1;
centro_gra=centro*fac_radio;
if((centro+radio1)>=142)
{
fac_radio=(centro+radio1)/142;
cc_g=centro/fac_radio;
radio=radio1/fac_radio;
escx=8*fac_radio;
}
else
{
cc_g=centro*fac_radio;
escx=8/fac_radio;
}

r2_g=radio*radio;

IMPRIMIR(ESCA,0x1A1,20,Derecha);DES_NUM(escx);

for(i=0;i<=21;i++)
  {
  if(i!=0) for(j=0;j<=73;j++) SALIDA[j]=0;
  else for(j=0;j<=73;j++) SALIDA[j]=128;
   for(j=0;j<=7;j++)
```

```
        {
        muestra=r2_g-(pixel-cc_g)*(pixel-cc_g);
        if(muestra<=0) muestra=0;
        muestra=sqrt(muestra);
        NIVEL[j]=muestra;
        pixel++;
        }
        SALIDA[1]=0x80;
        nivel_a=GRAFICA(nivel_a,NIVEL,SALIDA,i);
  }
}

////////////////////////////////////////////////////////////////////////
// Función GRAF_BODE_MAG(): Procesa los parametros de la celda con su
// función de transferencia caracteristica, para implementar una
// gráfica de Bode en Magnitud.
void GRAF_BODE_MAG(void)
{
 byte i,j,pixel=0,nivel_a=0,x,NIVEL[8];
 byte SALIDA[78];
 double muestra,xx,rr1,rr2,rr,ii,zz;
 float esc;

TEXTOS();
IMPRIMIR(Mg,0x11E,2,Derecha);//y
IMPRIMIR(fr,0x13E,2,Derecha);//x

GRAFICA_EJE_LOG();

IMPRIMIR(DEC,0x161,31,Derecha);

esc=(Rp+Rs);
IMPRIMIR(MAG,0x181,18,Derecha);DES_NUM(esc);

esc=(Rp+Rs)/8.99;
IMPRIMIR(IBM,0x1A1,14,Derecha);DES_NUM(esc);
for(i=0;i<=4;i++) ESCRIBIR_DATO(ABS[i]);

IMPRIMIR(BOD,0x21,6,Abajo);

IMPRIMIR(MAG,0x42,8,Abajo);

for(i=0;i<=21;i++)
  {
  if(i!=0) for(j=0;j<=73;j++) SALIDA[j]=0;
```

```
  else for(j=0;j<=73;j++) SALIDA[j]=128;
   for(j=0;j<=7;j++)
          {
          xx=pixel;
          xx=exp((xx+25)*0.06375);
          rr1=xx*Rp*cd;
          rr2=rr1*rr1;
          rr=(Rs+Rp/(1+rr2));
          ii=(rr1*Rp)/(1+rr2);
          zz=rr*rr+ii*ii;
          muestra=(72/(Rp+Rs))*sqrt(zz);
          NIVEL[j]=muestra;
          pixel++;
          }
          nivel_a=GRAFICA(nivel_a,NIVEL,SALIDA,i);
 }
}

////////////////////////////////////////////////////////////////////
// Función GRAF_BODE_FAS(): Procesa los parametros de la celda con su
// función de transferencia caracteristica, para implementar una
// gráfica de Bode en Fase.
void GRAF_BODE_FAS(void)
{
 byte i,j,pixel=0,nivel_a=0,x,NIVEL[8];
 byte SALIDA[78];
 double muestra,xx,ang,rr1,rr2,rr,ii,zz;
 float esc;

TEXTOS();
IMPRIMIR(An,0x11E,2,Derecha); //y
IMPRIMIR(fr,0x13E,2,Derecha); //x

GRAFICA_EJE_LOG();

IMPRIMIR(DEC,0x161,31,Derecha);

IMPRIMIR(FAS,0x181,30,Derecha);

IMPRIMIR(IBM,0x1A1,24,Derecha);

IMPRIMIR(BOD,0x21,6,Abajo);

IMPRIMIR(FAS,0xC2,4,Abajo);
```

```
for(i=0;i<=21;i++)  // los 22 bytes de grafico
  {
  if(i!=0) for(j=0;j<=73;j++) SALIDA[j]=0;
  else for(j=0;j<=73;j++) SALIDA[j]=128;
   for(j=0;j<=7;j++)
          {
          xx=pixel;
          xx=exp((xx+25)*0.06375);
          rr1=xx*Rp*cd;
          rr2=rr1*rr1;
          rr=(Rs+Rp/(1+rr2));
          ii=(-rr1*Rp)/(1+rr2);
          zz=rr/ii;
          muestra=45.83662361*(1.5707963+atan(zz));
          NIVEL[j]=muestra;
          pixel++;
          }
  nivel_a=GRAFICA(nivel_a,NIVEL,SALIDA,i);
  }
}

////////////////////////////////////////////////////////////////////////
// Función GRAFICA(nivel_a,NIVEL,SALIDA,i): Se encarga de presentar en
// pantalla los datos calculados por las diferentes funciones.
byte GRAFICA(byte nivel_a,byte NIVEL[8],byte SALIDA[78],byte i)
{

  byte j,x;

  for(j=0;j<=7;j++)
   {
    if(nivel_a==NIVEL[j]) SALIDA[NIVEL[j]]=SALIDA[NIVEL[j]]|PUNTOS[j];
          else if(nivel_a<NIVEL[j])  for(x=0;x<=NIVEL[j]-nivel_a-1;x++)
                        SALIDA[nivel_a+x+1]=SALIDA[nivel_a+x+1]|PUNTOS[j];
          else for(x=0;x<=nivel_a-NIVEL[j]-1;x++)
                        SALIDA[nivel_a-x-1]=SALIDA[nivel_a-x-1]|PUNTOS[j];
          nivel_a=NIVEL[j];
          }
          SALIDA[0]=0xFF;
          POS_CURSOR(P_Grafico+0x964+i);MOV_CURSOR(Arriba);
          for (j=0;j<=73;j++) ESCRIBIR_DATO(SALIDA[j]);

return nivel_a;

}
```

```
////////////////////////////////////////////////////////////////////
// Función FFT(): Aplica la Transformada de Fourier a la señal, que
// se haya muestreado, siendo un requisito saber la frecuencia de la
// señales y ademas que sean senoidal pura. Calcula la amplitud de y
// fase; se minimiza el error tomando 8 muestras por periodo con un
// mismo espaciamiento.
void FFT(float adc_volt[7],float adc_crte[7],byte j)
{
 byte i;
 double yc1,yc2,s2,ys1,ys2,s22,c2,w,D,E1,E2,F1,F2,Bc1,
        Bc2,Bs1,Bs2,Q1,Q2,z1,z2,z3,A1,A2,Q3,x[8],dt;

yc1=0;yc2=0;s2=0;ys1=0;ys2=0;s22=0;c2=0;

for(i=1;i<=8;i++)
{x[i-1]=i*DELTA[j];}

w=6.283185307*FRECUENCIAS[j];

for(i=0;i<=7;i++)
      {
      z1=cos(w*x[i]);
      z2=sin(w*x[i]);
      z3=sin(2*w*x[i]);

      yc1=yc1+z1*adc_volt[i];
      yc2=yc2+z1*adc_crte[i];
      s2=s2+(z2*z2);
      ys1=ys1+(z2*adc_volt[i]);
      ys2=ys2+(z2*adc_crte[i]);
      s22=s22+z3;
      c2=c2+(z1*z1);
      }

D=c2*s2-0.25*s22*s22;

E1=yc1*s2-ys1*0.5*s22;
E2=yc2*s2-ys2*0.5*s22;

F1=c2*ys1-0.5*s22*yc1;
F2=c2*ys2-0.5*s22*yc2;

Bc1=E1/D;
Bc2=E2/D;
```

```
Bs1=F1/D;
Bs2=F2/D;

A1=sqrt(Bc1*Bc1+Bs1*Bs1);//VOLTAJE
A2=sqrt(Bc2*Bc2+Bs2*Bs2);//CORRIENTE

Q1=atan(Bc1/Bs1);
Q2=atan(Bc2/Bs2);

Q3=Q2-Q3;//RETRASO DE FASE, SEÑALES INVERTIDAS

amplvolt=A1;
amplcrte=A2;
desfase=Q3;
}

////////////////////////////////////////////////////////////////////////
// Función Reg_Circular(real,imag): Con los vectores de la parte real
// y parte imaginaria de los datos adquiridos, realiza una regresión
// circular; cambia los datos de radio y centro, variables generales.
void Reg_Circular(float real[48],float imag[48])
{
byte i;
float xi3=0,xiyi2=0,xi=0,xi2=0,yi2=0,h,r2;
for(i=0;i<=47;i++)
     { xi3=real[i]*real[i]*real[i]+xi3;
       xiyi2=real[i]*imag[i]*imag[i]+xiyi2;
       xi=real[i]+xi;
       xi2=real[i]*real[i]+xi2;
       yi2=imag[i]*imag[i]+yi2;
     }
h=(xi3+xiyi2-(0.02083333*xi)*(xi2+yi2))/(2*xi2-0.04166666*xi*xi);
  // h=(xi3+xiyi2-(1/48*xi)*(xi2+yi2))/(2*xi2-2/48*xi*xi)
r2=h*h-(0.04166666*xi)*h+0.02083333*(xi2+yi2);
 // r2=h*h-(2/48*xi)*h+1/48*(xi2+yi2);
centro=h;
radio2=r2;
radio1=sqrt(radio2);
}

////////////////////////////////////////////////////////////////////////
// Función GRAFICA_EJE_LOG(): Presenta la escala logaritmica para el
// eje x, con siete decadas, que se utiliza en la visualización del
// diagrama de Nyquist.
```

```
void GRAFICA_EJE_LOG(void)
{
 GRAFICA_EJES();

 IMPRIMIR(LOGARI1,P_Grafico+0x984,22,Derecha);

 IMPRIMIR(LOGARI2,P_Grafico+0x9A4,22,Derecha);
 IMPRIMIR(LOGARI2,P_Grafico+0x9C4,22,Derecha);
}

////////////////////////////////////////////////////////////////////////
// Función DES_NUM(): Despliega en pantalla numeros de punto flotante
// en formato decimal con sigo, con tres cifras decimales; siguiente
// a la posición actual del cursor.
void DES_NUM(float numero)
{
unsigned long int Aj1,Aj2,Aj3,Aj4,Aj5,Aj6,Aj7,numero_ent;
char salida[8],j;
byte flag=0;

for(j=0;j<=9;j++) ESCRIBIR_DATO(0x20);
MOV_CURSOR(Izquierda); for(j=0;j<=9;j++) ESCRIBIR_DATO(0x20);
MOV_CURSOR(Derecha);

if(numero<=0){numero=-numero;TEXTO("-");}
numero=numero*1000;
numero_ent=numero;

Aj1=numero_ent/10;
Aj2=numero_ent/100;
Aj3=numero_ent/1000;
Aj4=numero_ent/10000;
Aj5=numero_ent/100000;
Aj6=numero_ent/1000000;
Aj7=numero_ent/10000000;

numero_ent=numero_ent+Aj1*0x06+Aj2*0x60+Aj3*0x600+
           Aj4*0x6000+Aj5*0x60000+Aj6*0x600000+
           Aj7*0x6000000;

salida[0]=((numero_ent & 0XF0000000)>>28)+48;
salida[1]=((numero_ent & 0X0F000000)>>24)+48;
salida[2]=((numero_ent & 0X00F00000)>>20)+48;
salida[3]=((numero_ent & 0X000F0000)>>16)+48;
salida[4]=((numero_ent & 0X0000F000)>>12)+48;
```

```
salida[5]='.';
salida[6]=((numero_ent & 0X00000F00)>>8)+48;
salida[7]=((numero_ent & 0X000000F0)>>4)+48;
salida[8]=((numero_ent & 0X0000000F))+48;

flag=0;
ESCRIBIR_COMANDO(0x42);
  for(j=0;j<=3;j++)
  {
  if(salida[j]!='0' & flag==0) flag=1;
  if(flag==1) ESCRIBIR_DATO(salida[j]);
  }
  ESCRIBIR_DATO(salida[4]);
  ESCRIBIR_DATO('.');
  if(salida[8]=='0' & salida[7]=='0') ESCRIBIR_DATO(salida[6]);
  else if(salida[8]=='0')
        {
        ESCRIBIR_DATO(salida[6]);
        ESCRIBIR_DATO(salida[7]);
        }
  else for(j=6;j<=8;j++) ESCRIBIR_DATO(salida[j]);
}

//////////////////////////////////////////////////////////////////////
// Función MOV_CURSOR(Dirección): Programa a la pantalla para que
// efectue el desplazamiento del cursor hacia la derecha, izquierda
// arriba, abajo, segun el arguemto "Dirección".
void MOV_CURSOR(unsigned int Mov)
{ESCRIBIR_COMANDO(Mov);ESCRIBIR_COMANDO(0x42);}

//////////////////////////////////////////////////////////////////////
// Función POS_CURSOR(Posición): Posiciona al cursor en la dirección
// de memoria, de argumento "Posición".
void POS_CURSOR(unsigned int Pos)
{ESCRIBIR_COMANDO(0x46); ESCRIBIR_DATO(Pos);
ESCRIBIR_DATO(Pos>>8);}

//////////////////////////////////////////////////////////////////////
// Función GRAFICAR_MARCOS(): Traza las líneas de marco, tanto de la
// parte gráfica como la de texto.
void GRAFICAR_MARCOS(void)
{
 //***** LINEAS DESCENDENTES //////

IMPRIMIR_V(0x80,P_Grafico+0x20,81,Abajo); // Linea Izquierda de Grafico
```

```
IMPRIMIR_V(0x04,P_Grafico+0x3B,81,Abajo); // Linea Derecha de Grafico

IMPRIMIR_V(0x80,P_Grafico+0xAC0,28,Abajo); // Linea Izquierda de Texto

IMPRIMIR_V(0x01,P_Grafico+0xADF,28,Abajo); // Linea Derecha de Texto

 //**** LINEAS DERECHA

IMPRIMIR_V(0xFF,P_Grafico,27,Derecha);    // Linea Superior de Grafico
ESCRIBIR_DATO(0xFC);

IMPRIMIR_V(0xFF,P_Grafico+0xA40,27,Derecha); //Linea Inferior de Grafico
ESCRIBIR_DATO(0xFC);

IMPRIMIR_V(0xFF,P_Grafico+0xAA0,32,Derecha); //Linea Superior de Texto

IMPRIMIR_V(0xFF,P_Grafico+0xE40,32,Derecha); //Linea Inferior de Texto
}

////////////////////////////////////////////////////////////////////////
// Función SOMBRA_CONTROLES(): Grafica un recuadro sobre cada uno de
// los tres controles del programa, dando un aspecto de sombra.
void SOMBRA_CONTROLES(void)
{
  byte j,x;
  word Direccion;

 IMPRIMIR(TRIANGULO1,P_Grafico+0xEE3,9,Abajo); //CONTROL 1

 Direccion= P_Grafico+0xEE4; //Sombra control1
  for (x=0;x<=8;x++)
   {
    IMPRIMIR_V(0xFF,Direccion,5,Derecha);
    Direccion=Direccion+0x20;
   }

IMPRIMIR(TRIANGULO2,P_Grafico+0xEE9,9,Abajo); //CONTROL 1

IMPRIMIR(TRIANGULO1,P_Grafico+0xEED,9,Abajo); //CONTROL 2

 Direccion= P_Grafico+0xEEE; //Sombra control2
  for (x=0;x<=8;x++)
   {
    IMPRIMIR_V(0xFF,Direccion,4,Derecha);
```

```
    Direccion=Direccion+0x20;
   }

IMPRIMIR(TRIANGULO2,P_Grafico+0xEF2,9,Abajo); //CONTROL 2

IMPRIMIR(TRIANGULO1,P_Grafico+0xEF6,9,Abajo); //CONTROL 3

 Direccion= P_Grafico+0xEF7; // Sombra control3
  for (x=0;x<=8;x++)
   {
    IMPRIMIR_V(0xFF,Direccion,5,Derecha);
    Direccion=Direccion+0x20;
   }

IMPRIMIR(TRIANGULO2,P_Grafico+0xEFC,9,Abajo); //CONTROL 3
}

////////////////////////////////////////////////////////////////////////
// Función GRAFICA_EJES(): Presenta la linea del eje Y, junto con sus
// divisiones. y la linea de eje X.
void GRAFICA_EJES(void)
{
  byte x,i;
  word Direccion;

IMPRIMIR_V(0x80,P_Grafico+0x44,79,Abajo);

  Direccion=P_Grafico+0x63;//separaciones
  for(i=0;i<=9;i++)
 {
  POS_CURSOR(Direccion); ESCRIBIR_COMANDO(0x42);
  ESCRIBIR_DATO(0x01); Direccion=Direccion+0x100;
 }
 IMPRIMIR_V(0xFF,P_Grafico+0x963,24,Derecha); // Linea de eje X
}

////////////////////////////////////////////////////////////////////////
// Función Textos(): Despliega el texto que permanece en forma
// constante en la pantalla de gráficos.
 void TEXTOS(void)
{
 POS_CURSOR(0xFC); MOV_CURSOR(Derecha); TEXTO("EJES");
 POS_CURSOR(0x11C);ESCRIBIR_COMANDO(0x42);TEXTO("y=");
 POS_CURSOR(0x13C);ESCRIBIR_COMANDO(0x42);TEXTO("x=");
}
```

```
////////////////////////////////////////////////////////////////////
// Función LIMPIAR_TEXTO():Encargada de limpiar toda la pantalla de
// texto, escribiendo espacios en su memoria.
void LIMPIAR_TEXTO(void)
 { IMPRIMIR_V(0x20,0x00,512,Derecha);}

////////////////////////////////////////////////////////////////////
// Función LIMPIAR_INFOR(): Limpia la pantalla de texto en la posición
// superior, no alcanza la parte de los controles.
void LIMPIAR_INFOR(void)
 {  IMPRIMIR_V(0x20,0x00,480,Derecha);}

////////////////////////////////////////////////////////////////////
// Función LIMPIAR_GRAFICO(): Limpia completamente la pantalla grafica
// enviando ceros a la memoria de la pantalla.
void LIMPIAR_GRAFICO(void)
 {  IMPRIMIR_V(0x00,P_Grafico,4096,Derecha);}

////////////////////////////////////////////////////////////////////
// Función LIMPIAR_GRAFICO_P():Limpia parcialmente la pantalla grafica
// en la zona de recuadro superior.
void LIMPIAR_GRAFICO_P(void)
{
  byte i;
  word Direccion;

  Direccion=P_Grafico;
   for (i=1;i<=82;i++)
    {
     IMPRIMIR_V(0x00,Direccion,28,Derecha);
     Direccion=Direccion+0x20;
    }
  GRAFICAR_MARCOS();
}

////////////////////////////////////////////////////////////////////
// Función TEXTO(texto): Despliega una cadena de caracteres de
// argumento "texto".
void TEXTO(char *cadena)
{
 char letra;
 letra = *cadena;
  while (letra != 0)
  {
```

```
    ESCRIBIR_DATO(letra);
    cadena++;
    letra = *cadena;
  }
}

////////////////////////////////////////////////////////////////////
// Función IMPRIMIR(texto,posicion,tamaño,direc): despliega los
// vectores texto que han sido guardados en memoria flash de datos con
// argumento "texto","posición" "tamaño" y "direccion".
void IMPRIMIR(const byte texto[],int posicion,byte tamaño,byte direc)
{
  byte i;

  POS_CURSOR(posicion); MOV_CURSOR(direc);
  for(i=0;i<=tamaño-1;i++) ESCRIBIR_DATO(texto[i]);
}


void IMPRIMIR_V(byte val,int posicion,byte tamaño,byte direc)
{
  byte i;

  POS_CURSOR(posicion); MOV_CURSOR(direc);
  for(i=0;i<=tamaño-1;i++) ESCRIBIR_DATO(val);
}

////////////////////////////////////////////////////////////////////
// Función ESCRIBIR_COMANDO(comando): Se utiliza para enviar datos
// de comando a la pantalla, de argumento "comando".
void ESCRIBIR_COMANDO(byte Comando)
{
MODO_A0(Alto);
setReg(SPDTR,Comando);
ESCRITURA_WR(Bajo);
MICROSEC(46);
ESCRITURA_WR(Alto);
MODO_A0(Bajo);
}

////////////////////////////////////////////////////////////////////
// Función ESCRIBIR_DATO(dato): Envia "dato" a la pantalla.
void ESCRIBIR_DATO(byte Dato)
{
setReg(SPDTR,Dato);
```

```
ESCRITURA_WR(Bajo);
MICROSEC(46);
ESCRITURA_WR(Alto);
}

//////////////////////////////////////////////////////////////////////
//Función MODO_A0(estado):cambia a "estado" el pin A0(TRD0).
void MODO_A0(bool estado)
{
if(estado==1) setRegBit(GPIO_A_DR,D0);
 else clrRegBit(GPIO_A_DR,D0);
}

//////////////////////////////////////////////////////////////////////
// Función ESCRITURA_WR(estado):cambia a "estado" el pin WR(TRD1).
void ESCRITURA_WR(bool estado)
{
if(estado==1) setRegBit(GPIO_A_DR,D1);
 else clrRegBit(GPIO_A_DR,D1);
}

//////////////////////////////////////////////////////////////////////
// Función ENTER_ESC(): Da un pulso al pin "Enter" del Esclavo
void ENTER_ESC(void)
{
clrRegBit(GPIO_B_DR,D0);SEGUNDO(1);setRegBit(GPIO_B_DR,D0);
}

//////////////////////////////////////////////////////////////////////
// Función SEGUIR_ESC(): Da pulso a pin "Seguir" del Esclavo
void SEGUIR_ESC(void)
{
clrRegBit(GPIO_B_DR,D1);SEGUNDO(1);setRegBit(GPIO_B_DR,D1);
}

//////////////////////////////////////////////////////////////////////
// Funciones encargadas de leer el estado de los pulsadores de control
// y retornarlo a donde se haya solicitado.
bool CNTRL(void) { return getRegBit(GPIO_A_DR,D2);}

bool MENU(void) { return getRegBit(GPIO_B_DR,D6);}

bool ENTER(void) { return getRegBit(GPIO_B_DR,D7);}

//////////////////////////////////////////////////////////////////////
```

```
// Función Gan_Cte(): Toma el estado de dos bits del "Esclavo", que se
// relacionan con el factor de ganancia de corriente para la señal de
// voltaje(medida de corriente) que se toma con el ADC.
byte Gan_Cte(void)
{
if(!getRegBit(GPIO_B_DR,D2)&&!getRegBit(GPIO_B_DR,D3)) return 0;
else if(!getRegBit(GPIO_B_DR,D2)&&getRegBit(GPIO_B_DR,D3)) return 1;
else if(getRegBit(GPIO_B_DR,D2)&&!getRegBit(GPIO_B_DR,D3)) return 2;
return 3;           // D2 SHUNT
}                  // D3 GANANCIA

//////////////////////////////////////////////////////////////////////
// Funciones utilizadas para dar retardos al flujo del programa
void RETARDO_ADC(unsigned int tiempo)
{ unsigned int a;    for(a=0;a<=tiempo;a++){}     }

void RETARDO_ADC_L(unsigned int tiempo)
{word a,b;    for(a=0;a<=tiempo;a++) for(b=0;b<=80;b++) {}     }

void MICROSEC(unsigned int tiempo)
{  word a,b;for(a=0;a<=tiempo;a++){ asm{nop
                                       nop
                                       nop} } }
void SEGUNDO(word  tiempo)
{
 byte c;
 for(c=1;c<=tiempo;c++) RETARDO_ADC_L(61225);

}
//////////////////////////////////////////////////////////////////////
// Funciones matemáticas
float sin(float x1)  //    sen x = x-x^3/3!+x^5/5!-x^7/7! ...
{
float xx;
int y1,signo=1;
xx=x1;

if(x1>=pi)
{
y1=x1/pi;
signo=pot(-1,y1);
xx=x1-y1*pi;
}
return(signo*(xx-pot(xx,3)/fac(3)+pot(xx,5)/fac(5)-pot(xx,7)/fac(7)
          +pot(xx,9)/fac(9)-pot(xx,11)/fac(11)+pot(xx,13)/fac(13)
```

```
        -pot(xx,15)/fac(15)+pot(xx,17)/fac(17)-pot(xx,19)/fac(19)
        +pot(xx,21)/fac(21)-pot(xx,23)/fac(23)+pot(xx,25)/fac(25)
        -pot(xx,27)/fac(27)+pot(xx,29)/fac(29)-pot(xx,31)/fac(31)
        +pot(xx,33)/fac(33)-pot(xx,35)/fac(35)));
}
//---------------------------------------------------------------
long double atan(float x1) //atan= x - x3/3 + x5/5 - x7/7 + x9/9...
{
long double xx=x1;
long double signo=1;
if(xx==0) return 0;
if(x1<0)
{
signo=-1;
xx=-x1;
}

if(xx<=6) return(signo*(-(3.38905e-3)*pot(xx,4)+
      (5.72072e-2)*pot(xx,3)-(3.61935e-1)*xx*xx+
      1.07748*xx+(5.19498e-3)));

else if(xx<=60) return(signo*((-1.0297955e-7)*pot(xx,4)+
      (1.6160681e-5)*pot(xx,3)-(9.2818849e-4)*xx*xx+
      (2.3820753e-2)*xx+1.305445));

else return  signo*1.55335;
}
//---------------------------------------------------------------
long double pot(float val,char x)
{
double result=1; int i;
for(i=1;i<=x;i++) result*=val;
return result;
}
//---------------------------------------------------------------
float cos(float x)
{ return sin(x+1.570796327);}
//---------------------------------------------------------------
long double fac(char x)
{
int i;long double result=1;
for(i=2;i<=x;i++)result*=i;
return result;
}
//---------------------------------------------------------------
```

```
float sqrt(float x)
{
 if(x==0) return 0;
  else
   {
    if(x==1) return 1;
     else
      {
       float i=x/2,j=2*TOLER;
       while(j>=TOLER)
        {
        i=(i+x/i)/2;
        if((i*i-x)*100/x>=0)
          {j=(i*i-x)*100/x;}
        else
          {j=(x-i*i)*100/x;}
        }
        return i;
      }
   }
}
//----------------------- FIN DEL PROGRAMA -------------------------
```

ANEXO E PROGRAMAS EN MATLAB

Este anexo contiene los programas realizados en Matlab para la regresión Senoidal y la regresión circular.

A continuación se presenta el programa implementado para la regresión Senoidal. El programa se encuentra grabado en el CD que se entrega con el nombre de “mie2” con extensión *.m.

```
clear
clc
disp(  '*********************************************');
disp(  '*******   Regresion Senoidal  ***********');
disp(  '*********************************************');
disp(  '  ');
disp(  '  ');
f=100.1e3;
d=1/(16*f);

x=[d 3*d 7*d 12*d 14*d];
n=length(x);
an1=input('Introduzca fase de la tension en grados = ');
an2=input('Introduzca fase de la corriente en grados = ');
qv=an1*pi/180;
qi=an2*pi/180;

y=12.7*sin(2*pi*f*x+qv);
g=2.8*sin(2*pi*f*x+qi);

w=2*pi*f;
yc=0;
gc=0;
s2=0;
s2g=0;
ys=0;
gs=0;
s22=0;
s22g=0;
c2=0;
c2g=0;
l=1;
for i=1:n,
   yc=yc+cos(w.*x(i)).*y(i);
   gc=gc+cos(w.*x(i)).*g(i);
```

```
    s2=s2+(sin(w.*x(i))).^2;
    s2g=s2g+(sin(w.*x(i))).^2;
    ys=ys+(sin(w.*x(i))).*y(i);
    gs=gs+(sin(w.*x(i))).*g(i);
    s22=s22+sin(2*w.*x(i));
    s22g=s22g+sin(2*w.*x(i));
    c2=c2+(cos(w.*x(i))).^2;
    c2g=c2g+(cos(w.*x(i))).^2;
end
D=c2*s2-0.25*(s22)^2;
E=yc*s2-ys*0.5*s22;
F=c2*ys-0.5*s22*yc;
Bc=E/D;
Bs=F/D;
Dg=c2g*s2g-0.25*(s22g)^2;
Eg=gc*s2g-gs*0.5*s22g;
Fg=c2g*gs-0.5*s22g*gc;
Bcg=Eg/Dg;
Bsg=Fg/Dg;
A=sqrt(Bc^2+Bs^2);
Ag=sqrt(Bcg^2+Bsg^2);

tc=sign(Bc);
ts=sign(Bs);
if tc==1
    if ts==1
        Q=atan(Bc/Bs)*180/pi;
    else
        Q=atan(Bc/Bs)*180/pi+180;
    end
else
    if ts==1
        Q=atan(Bc/Bs)*180/pi;
    else
        Q=atan(Bc/Bs)*180/pi-180;
    end
end

tcg=sign(Bcg);
tsg=sign(Bsg);
if tcg==1
    if tsg==1
        Qg=atan(Bcg/Bsg)*180/pi;
    else
        Qg=atan(Bcg/Bsg)*180/pi+180;
```

```
    end
else
    if tsg==1
        Qg=atan(Bcg/Bsg)*180/pi;
    else
        Qg=atan(Bcg/Bsg)*180/pi-180;
    end
end

if ts==0
    if tc==1
        Q=90;
    else
        Q=-90;
    end
end

if tsg==0
    if tcg==1
        Qg=90;
    else
        Qg=-90;
    end
end


disp('La magnitud de la tension es ='),disp(A);
disp('La fase de la tension en grados es ='),disp(Q);

disp('La magnitud de la corriente es ='),disp(Ag);
disp('La fase de la corriente en grados es ='),disp(Qg);

t=[0:d:1/f];
y1=A*sin(w*t+Q*pi/180);%señal reconstruida
g1=Ag*sin(w*t+Qg*pi/180);%señal reconstruida
figure
stem(x,y);hold on
plot(t,y1,'r');stem(x,g);plot(t,g1,'g');hold off;
```

La regresión Circular implementada es la que se encuentra en el anexo B. El programa se encuentra grabado en el CD con el nombre de mie1 con extensión *.m.

```
disp(    '*********************************************************');
disp(    '************ REGRESION CIRCULAR  ********************');
disp(    '*********************************************************');
disp('  ');
disp('  ');
disp('  ');

%Para importar datos de un archivo de excel guardado en work
%llamado'datos3' tomados de pruebas GIC Guatiguara

a=xlsread('datos1');
q=length(a);

%Definiendo los datos en los vectores x e y
for j=1:q
   x(j)=a(j);
   y(j)=a(q+j);
end


%Definicion inicial de acumuladores%

x1=0;
x2=0;
x3=0;
y2=0;
x1y2=0;
n=length(x);

for i=1:n
   x1=x1+x(i);
   x2=x2+x(i).^2;
   x3=x3+x(i).^3;
   y2=y2+y(i).^2;
   x1y2=x1y2+x(i).*y(i)^2;
end

%Calculo del Radio [R] y el centro [h] de acuerdo con la regresion circular
%de minimos cuadrados para y^2

h=vpa((x3+x1y2-(1/n)*x1*(x2+y2))/(2*(x2-(1/n)*x1^2)),6)
```

```
R=vpa(sqrt(h^2-(2/n)*x1*h+(1/n)*(x2+y2)),6)

o=sym('o');
p=sym('p');
disp(' La ecuacion de la circunferencia es =');
subs('y^2+(x-o)^2=p^2',{o,p},{h,R})

t1=vpa(-abs(-h+R));
t2=vpa(2*R+abs(h-R));
t3=vpa(-0.01*R);
t4=vpa(1.5*R);
f=subs('y=sqrt(p^2-(x-o)^2)',{o,p},{h,R});

ezplot(f,[double(sym(t1)),double(sym(t2))],[double(sym(t3)),double(sym(t4))]);
hold on;
title('Analisis de Regresion Circular EIS');
xlabel('Real(Z) [Ohm]');
ylabel('Imag(Z) [Ohm]');
plot(x,y,'ro'); hold off;
```

Regresión Circular

El programa resuelve las 2 ecuaciones finales para el centro de la media circunferencia y el radio de esta. A continuación se presenta el algoritmo realizado y los datos obtenidos.

La tabla E1 presenta los datos tomados de la celda electroquímica típica, guardados en Excel con el nombre de mie1 con extensión *.xls guardado en el CD que se entrega a Biblioteca.

Tabla E1 Datos 1

x	y
2	6,928
3	7,416
4	7,746
5	7,937
6	8
7	7,937
8	7,746
9	7,416

El programa mie1 lee los datos tomados de una celda electroquímica guardados en Excel con extensión .xls, para luego realizar las operaciones asignadas y dando como resultado el centro y radio de la circunferencia, grafica el semicírculo y por ultimo arroja la ecuación característica del semicírculo. A continuación se presentan los resultados obtenidos en Matlab.

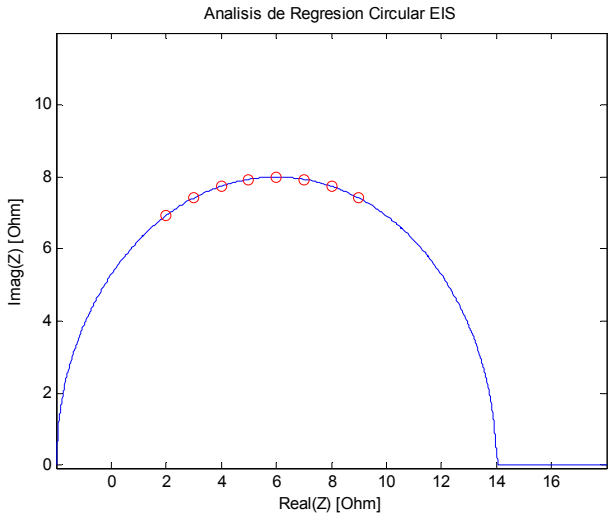
****
********** REGRESION CIRCULAR **********
****

h =6.00004

R =7.99988

La ecuacion de la circunferencia es = ans =y^2+(x-6.00004)^2 = 63.9980800144

Figura 43 Regresión Circular datos 1



Regresión Senoidal.

Se implemento una regresión lineal que reconstruyera las señales de entradas, para poder reconstruir la amplitud y fase de la señal de tensión de la celda y la corriente de la celda electroquímica.

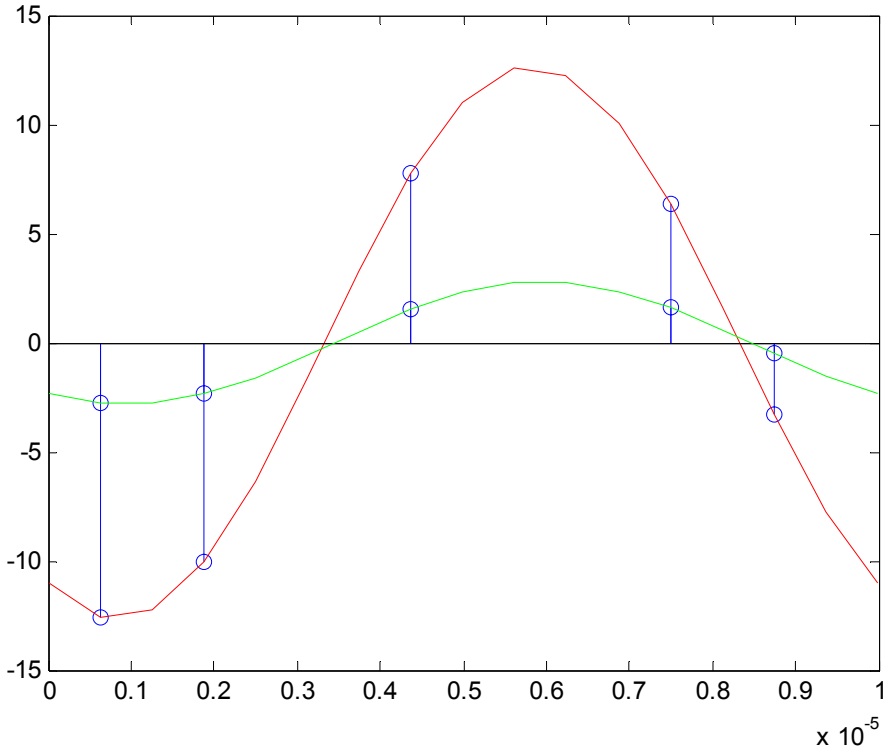
La deducción de la regresión senoidal aparece explicada en el anexo B. El programa reconstruye señales tomando mínimo 3 muestras, haciendo

posible reconstruir las señales. El programa es realizado en Matlab guardado con el nombre ´´mie2´´ con extensión *.m, guardado en el CD que se entrega.

Los resultados del programa se presentan a continuación.

```
*******************************************
*******     Regresion  Senoidal   ***********
*******************************************
Introduzca fase de la tension en grados = -30
Introduzca fase de la corriente en grados = -50
La magnitud de la tension es =   12.7000

La fase de la tension en grados es =  -30.0000

La magnitud de la corriente es =    2.8000

La fase de la corriente en grados es =  -50.0000
```

Grafica 44 Regresión Senoidal



La señal de color rojo es la señal de tensión introducida y la señal de color verde es la corriente.

ANEXO F PANTALLA DE CRISTAL LIQUIDO