GENERADOR DE SEÑALES PARA MEDICIONES DE IMPEDANCIA ELECTROQUÍMICA BASADO EN DSP.

Autores:

ADRIANA MARCELA MARTÍNEZ RIVERA
EDGAR LEONARDO ARENAS RANGEL

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA
2006

GENERADOR DE SEÑALES PARA MEDICIONES DE IMPEDANCIA ELECTROQUÍMICA BASADO EN DSP.

Autores:

ADRIANA MARCELA MARTÍNEZ RIVERA
EDGAR LEONARDO ARENAS RANGEL

TRABAJO DE GRADO

Director:

Ing. JOSÉ ALEJANDRO AMAYA PALACIO. M.I.(c)

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA
2006

DEDICATORIA

A mis padres, Hernando y Evangelina
Por su amor, apoyo y paciencia.

A mi hermano, Edgar
Por su cariño y confianza.

A mis tíos, María del Carmen, Marlene y Pablo
Por su apoyo incondicional.

A mi compañero de proyecto, Edgar Leonardo
Por su dedicación y trabajo.

Adriana Marcela

DEDICATORIA

A mis padres, Eduardo y Maria Jazmina
Por su amor, paciencia, confianza y apoyo incondicional.

A mi hermano, Andrés
Por la seguridad que me ha transmitido.

A mi novia, Ivonne
Por su amor, inspiración y apoyo.

A mi compañera de proyecto, Adriana
Por su dedicación, esfuerzo y trabajo.

Edgar Leonardo

AGRADECIMIENTOS

A Dios Todopoderoso, por guiarnos, darnos fortaleza, entendimiento, acompañarnos en nuestro camino y así culminar exitosamente esta etapa de nuestra vida.

A nuestro director de proyecto, Jose Alejandro Amaya Palacio, por su dedicación, paciencia, constante motivación y orientación durante el desarrollo de este proyecto.

A los profesores: Jaime Guillermo Barrero, Edwin Saavedra por su valiosa ayuda. A Javier Mier y Jairo Mantilla, por su constante y eficaz ayuda.

A la Universidad Industrial de Santander, especialmente a la Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones por los equipos suministrados.

A la empresa Analog Devices por su generosidad en el suministro de implementos electrónicos.

A nuestros familiares y amigos por su apoyo incondicional a lo largo de nuestra carrera universitaria.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTAS DE TABLAS

LISTA DE ANEXOS

RESUMEN

TITULO: GENERADOR DE SEÑALES PARA MEDICIÓN DE IMPEDANCIA ELECTROQUÍMICA BASADO EN DSP [*]

AUTORES: MARTÍNEZ RIVERA, ADRIANA MARCELA, y, ARENAS RANGEL, EDGAR LEONARDO [**]

PALABRAS CLAVES: Generador de señales, Impedancia Electroquímica, DSP, LCD, Amplitud, Frecuencia y PCB.

DESCRIPCIÓN:

En el documento se describe detalladamente el diseño e implementación de un Generador de Señales Senoidales con pantalla LCD y teclado para realizar medición de impedancia electroquímica basado en un DSP de la familia 56F8323 de Motorola.

En el primer capítulo se presenta la fundamentación teórica necesaria para el desarrollo del proyecto, así como una pequeña descripción del DSP e información sobre ruido, interferencias y normatividad para la construcción del Generador.

El Hardware implementado en este proyecto está formado por 2 tarjetas electrónicas: Tarjeta de desarrollo del DSP 56F8323 y la tarjeta de generación de funciones, cuyo diseño se encuentra explicado detalladamente en el segundo capítulo. En el tercer capítulo, se presentan los algoritmos necesarios para la generación de la señal senoidal, control de la pantalla LCD y control del teclado.

En el cuarto capítulo se presentan los resultados de las pruebas realizadas al integrado AD9833 utilizado para la generación de la señal, con el fin de evaluar su funcionamiento dentro de los rangos establecidos. También se mencionan las pruebas realizadas, para verificar el correcto funcionamiento de la pantalla LCD y el teclado. Todas estas pruebas permiten establecer los rangos de funcionamiento del equipo tanto en frecuencia como en amplitud.

Finalmente en el último capítulo, se presentan las conclusiones y observaciones del trabajo realizado.

---

[*] Trabajo de Grado

[**] Facultad de Ingenierías Físico-mecánicas. Ingeniería Electrónica. Director: M.I.(c) Jose Alejandro Amaya Palacio.

SUMMARY

TITLE: DSP BASED SIGNAL GENERATOR FOR ELECTROCHEMICAL IMPEDANCE MEASUREMENT*

AUTHORS: MARTÍNEZ RIVERA, ADRIANA MARCELA, y, ARENAS RANGEL, EDGAR LEONARDO**

KEY WORDS: Signal Generator, Electrochemical Impedance, DSP, LCD, Amplitude, Frequency and PCB.

DESCRIPTION:

This document describes the design and performance of a Sine Signals Generator with alpha-numeric LCD and keyboard to perform electrochemical impedance measurement based on a Motorola 56F8323 DSP.

Chapter one describes the basic theory for the development of the project, as well as a small description of the DSP, some information about noise, interference and a number of standards on the construction of the Generator.

The implemented hardware on this project is formed by tree electronic boards:
A development Board for the 56F8323 DSP, a Function Generation Board, and also a user interface board (LCD and keyboard). The design and performance is explained with detail on chapter two. On chapter three is shown the essential algorithm for the generation of the sine signal, control algorithm of LCD and control algorithm of the keyboard.

Chapter four presents the results of the tests performed on the AD9833 used on the generation of the signal, with the purpose of evaluating its operation within the established ranks. Also the made tests are mentioned for verify the correct operation of LCD and the keyboard. All these tests allow establishing the operation rates of the equipment in frequency and in amplitude.

Finally, on the last chapter are presented the project conclusions, observations and some recommendations about the equipment.

* Degree Work

** Faculty of Physical-Mechanical Engineering. Electronics Engineering. Director: M.I.(c) Jose Alejandro Amaya Palacio.

INTRODUCCIÓN

Uno de los procesos más importantes en el laboratorio de corrosión de Ingeniería Metalúrgica y Ciencia de Materiales, es la medición de impedancia electroquímica. Este proceso se utiliza generalmente en la caracterización de baterías, celdas de combustible y semiconductores. Es una técnica muy poderosa que usa señales de AC para determinar la impedancia en una celda electroquímica.

Este proyecto plantea el diseño e implementación de un generador de señales sinusoidales basado en un DSP, que complementa y mejora los medidores de impedancia electroquímica del laboratorio de corrosión. Dichos medidores presentan deficiencias en la parte de generación de la señal de AC. Con este proyecto se superan los inconvenientes presentados anteriormente y se logra un avance significativo en dicho proceso.

En el primer capítulo, se presenta la fundamentación teórica y alguna información adicional que se necesitó para implementar el equipo, como es, factores de ruido, interferencias y normatividad para la construcción del generador.

El segundo capítulo describe el hardware que se utilizó en la construcción del generador de señales. Se muestra en detalle la tarjeta de desarrollo del DSP, la tarjeta de generación de funciones, la fuente de alimentación, la pantalla LCD y el teclado.

En el capítulo 3, se muestra el software utilizado para el control del generador. Se presentan los algoritmos para la generación de la señal, el

control de frecuencia, la visualización en la pantalla y la recepción de datos por medio del teclado.

El capítulo 4 detalla las distintas pruebas de laboratorio que se realizaron tanto al integrado (AD9833) como al equipo en general. Estas pruebas se realizaron para comprobar el correcto funcionamiento del prototipo dentro de los márgenes establecidos para medición de impedancia electroquímica. Este capítulo también muestra el diseño del PCB y el prototipo final del equipo.

Por último, en el capítulo 5, se presentan las conclusiones y observaciones que se obtuvieron en el desarrollo del proyecto.

1. FUNDAMENTACIÓN TEÓRICA.

En el presente capítulo se realiza un estudio del proceso de corrosión y la técnica del seno simple. También se define que es un oscilador, posibles fuentes de ruido e interferencia que se pueden presentar en el diseño, las normas que se deben tener en cuenta para la construcción del generador de señales sinusoidal y una breve descripción del DSP que se utilizó en el proyecto.

1.1 PROCESOS ELECTROQUÍMICOS

Cuando hablamos de procesos electroquímicos, nos referimos a procesos donde existe una relación entre las corrientes eléctricas y las reacciones químicas. Son procesos donde ocurren fenómenos químicos causados por la acción de corrientes y tensiones, o se producen efectos eléctricos debido a reacciones químicas.

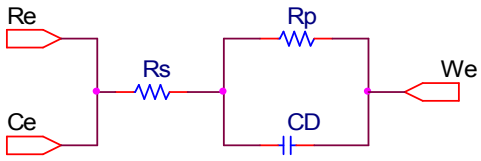
Los procesos electroquímicos pueden modelarse mediante elementos sencillos de circuitos tales como resistencias, inductores, condensadores y fuentes de tensión. Como la reacción de corrosión, es un proceso de este tipo, se puede modelar por medio de estos elementos.

1.1.1 Corrosión

Utilizando la teoría de circuitos AC, un proceso de corrosión simple se puede caracterizar mediante un circuito equivalente que modela el comportamiento

de la impedancia electroquímica cuando éste se excita en determinado rango de frecuencias. El circuito equivalente es como se muestra en la Figura 1.

Figura 1 Circuito equivalente de una celda electroquímica.



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

Donde,

R_S: Representa el efecto resistivo de la solución.

R_P: Representa la resistencia a la polarización.

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

Al aumentar la frecuencia, la impedancia del condensador disminuye. Por tanto, a altas frecuencias la impedancia del condensador C_D es mucho más pequeña que la impedancia de la resistencia R_P y como C_D está en paralelo con R_P, el condensador actúa como un corto y elimina a la resistencia del circuito. Entonces a altas frecuencias la celda es modelada casi enteramente por R_S.

A frecuencias muy bajas, el condensador actúa como un circuito abierto y éste puede ser excluido del circuito. La impedancia de la celda es entonces la combinación de dos resistencias en serie, R_S y R_P. Es decir, en los límites de alta y baja frecuencia, la celda se comporta principalmente como una resistencia. La componente imaginaria de la impedancia es muy pequeña, por tanto el ángulo de fase entre la tensión y la corriente en la celda es cercano a cero grados.

A frecuencias intermedias, la impedancia del condensador empieza a tener efecto y la celda tiende a comportarse de forma capacitiva. El ángulo de fase puede llegar a ser hasta de 90 grados.

1.1.2 Técnica del seno-simple

La técnica del seno-simple es utilizada para determinar la impedancia de una celda como una función de la frecuencia. Consiste en aplicar una señal de tensión senoidal de pequeña amplitud y de frecuencia fija a la celda de prueba y se mide la señal de corriente de la misma.

La señal de respuesta (señal de corriente) determina las componentes en fase (parte real) y fuera de fase (parte imaginaria) de la impedancia total. Con esta información, se determina el grado de desplazamiento de fase entre las ondas de entrada y salida, y se calcula la magnitud de la impedancia. Para determinar la impedancia, se debe realizar un espectro de frecuencia con varios valores por década y hacer una medición discreta en cada valor.

Esta técnica presenta varias ventajas. Por ejemplo, los datos obtenidos con la técnica, son muy confiables porque la forma de onda de prueba es generada a una frecuencia simple. También se pueden realizar mediciones rápidas a alta frecuencia y solo se requiere de un generador de frecuencias para producir la onda seno.

El problema se presenta cuando se realizan mediciones a baja frecuencia. La velocidad de medición disminuye. A una frecuencia de 1 Hz, se gastaría 1 segundo para aplicar la forma de onda de excitación. A 1mHz, este podría tomar 1000 segundos o alrededor de 16 minutos para aplicar la forma de onda.

1.2 RUIDO ELÉCTRICO E INTERFERENCIAS

El ruido eléctrico y las interferencias son dos fenómenos que se deben tener en cuenta cuando se van a construir equipos electrónicos.

1.2.1 Ruido eléctrico

Se designa con el nombre de ruido a toda señal no deseada que se encuentra superpuesta a una señal útil. Generalmente el ruido se presenta como señales que se introducen en el sistema de medida, afectando la medición de la señal deseada e incrementando los errores aleatorios.

En la práctica se presentan dos clases de ruido: la interferencia que se debe a la interacción entre campos eléctricos y magnéticos externos y el circuito del sistema de medida, y el ruido aleatorio producido por el movimiento aleatorio de los electrones y otros portadores de carga en los componentes. Este ruido aleatorio se produce de diversas formas entre las que cabe mencionar el ruido Johnson, el ruido de fluctuación y el ruido de granalla.

El ruido Jonson es el más común en los conductores y también es conocido como ruido térmico. Los electrones libres responsables de la conducción eléctrica en un material conductor, al estar sometidos a agitación térmica dan origen a pequeñas corrientes en todas las direcciones y sentidos dentro del material. En ausencia de campo eléctrico externo no hay direcciones privilegiadas, por tanto debido a la enorme cantidad de electrones libres se produce una compensación estadística que tiende a anular la corriente resultante. Esta compensación no es perfecta, generándose una corriente, que fluctúa aleatoriamente y que constituye el denominado ruido térmico.

El ruido de fluctuación es el que se presenta en algunos elementos como el diodo, transistor y termistor. Es un ruido producido por el flujo de portadores de carga en un medio discontinuo. Solo aparece a frecuencias bajas, alrededor de 1kHz.

Otro ruido que se presenta en los componentes electrónicos es el ruido de granalla. Es producido por el movimiento de portadores de carga a través de barreras de potencial, como en el caso de una unión p-n. Los portadores de carga fluyen a través de la barrera, en forma constante, pero tienen un elemento aleatorio superpuesto a ese movimiento. Por esta razón, hay fluctuaciones aleatorias conocidas como ruido de granalla.

1.2.2 Interferencias

Existen tres tipos de interferencias que se presentan cuando se desea diseñar y construir un equipo electrónico. Se clasifican de acuerdo al fenómeno físico responsable de su generación.

La primera interferencia es la producida por acoplamiento inductivo, electromagnético o magnético. Cuando una corriente cambiante se encuentra cerca de un circuito, se produce un cambio en el campo magnético, presentándose una fuerza electromotriz en el sistema de medida como consecuencia de la inducción magnética. Estas interferencias son causadas por los cables de potencia de las paredes cercanas y por los cables de cada instrumento.

El segundo tipo de interferencia, es el producido por el acoplamiento capacitivo. Los cables de potencia, la tierra y otros elementos están separados por un dieléctrico (aire), de los conductores del sistema de

medida. Entre estos elementos se forman condensadores. Un cambio en la tensión aplicada en una de las placas de dichos condensadores, afecta la tensión de la otra. Por tanto, estos condensadores acoplan los conductores del sistema de medida a otros sistemas y en consecuencia, las señales de esos sistemas pasan al sistema de medida como interferencia.

Otro tipo de interferencias son las producidas por las tierras múltiples. Esta interferencia se presenta cuando el sistema de medida tiene más de una conexión a tierra, dado que todos los puntos del circuito no están al mismo potencial. Esto se conoce como bucle de masa. Las tierras múltiples deben ser evitadas, ya que producen interferencia al sistema de medida. Por esta razón es recomendable tener únicamente un punto de tierra.

1.3 TEORÍA SOBRE OSCILADORES

En el diseño de sistemas electrónicos, con frecuencia surge la necesidad de señales que tengan formas de ondas normalizadas prescritas (sinusoidales, cuadradas, triangulares). Hay dos métodos claramente diferentes para la generación de ondas sinusoidales.

- *Osciladores lineales:* Son circuitos que generan ondas sinusoidales a través de fenómenos de resonancia. Utiliza un lazo de retroalimentación positiva que consiste en un amplificador y una red selectiva de frecuencia RC o LC. La amplitud de las ondas sinusoidales se ajusta, por medio de un mecanismo no lineal, implementado ya sea con un circuito separado o mediante las no linealidades del dispositivo amplificador.

- *Osciladores no lineales:* Son circuitos que generan ondas sinusoidales al conformar adecuadamente una onda triangular. Son los llamados circuitos generadores de funciones. Éstos utilizan una clase especial de circuitos conocidos como multivibradores. Hay tres tipos de multivibradores: biestables, monoestables y astables.

Un oscilador es un circuito capaz de convertir la corriente continua en una corriente que varía de forma periódica en el tiempo; estas oscilaciones pueden ser senoidales, cuadradas, triangulares, dependiendo de la forma que tenga la onda producida. Fundamentalmente un oscilador es un amplificador cuya señal de entrada se toma de su propia salida a través de un circuito de realimentación. Se puede considerar que está compuesto por un circuito cuyo desfase depende de la frecuencia, un elemento amplificador y un circuito de realimentación.

Existen varias configuraciones para osciladores, como por ejemplo, los multivibradores, osciladores LC, osciladores RF, osciladores de cristal, osciladores Seiler, osciladores Pierce, osciladores Colpitts, osciladores Hartley y osciladores Vackar.

1.4 PROCESADOR DIGITAL DE SEÑALES (DSP)

El generador de señales senoidales que se diseñó, esta basado en un procesador digital de señales (DSP), que es el encargado de controlar la frecuencia de la señal generada. Además es el punto de conexión entre el circuito integrado generador de funciones, la pantalla y teclado.

El procesador digital de señales utilizado es el DSP 56F8323 de la familia Motorola. Este DSP combina, en un solo chip, la capacidad de procesamiento de un DSP y la funcionalidad de un microcontrolador. Gracias

a su bajo costo, flexibilidad de configuración, y un compacto código de programación, el 56F8323 se ajusta perfectamente a muchas aplicaciones. El dispositivo incluye varios periféricos que son usados especialmente en control automático, control industrial, control de movimiento, inversores de propósito general, sensores inteligentes, sistemas de seguridad, y aplicaciones de monitoreo médico.

Los 56800E se basan en una arquitectura al estilo Harvard que consiste en tres unidades de ejecución operando en paralelo, que permiten ciclos de hasta seis operaciones por instrucción. El set de instrucciones trabaja eficientemente para compiladores C que habilitan un rápido desarrollo de aplicaciones de control. Este dispositivo cuenta con una línea externa de interrupción y más de 27 líneas de entrada/salida de propósito general, dependiendo de la configuración periférica.

1.5 NORMATIVIDAD PARA EL DISEÑO DEL GENERADOR DE SEÑALES.

El generador de señales que se construyó, hace parte de los equipos de impedancia electroquímica de los laboratorios de corrosión. Por este motivo, el generador de señales debe cumplir con ciertas normas que se mencionan a continuación:

- Límites en la frecuencia que se va a manejar:

  "Como límite normativo para procesos de espectroscopia de impedancia electroquímica, la frecuencia debe estar limitada a un rango que va desde los 10[mHz] hasta los 100[kHz]."[1]

[1] Impedance Instrumentation and Techniques. Aplication Note AC-3.

- Límites en la amplitud de tensión que se va a manejar:

  "La amplitud de la señal AC debe ser lo suficientemente baja para asegurar que la respuesta del instrumento de medición de impedancia se comporte de forma lineal, por esta razón la amplitud está limitada a valores de 10[mV] a 60[mV]."[2]

El generador de señales senoidales se va a utilizar en el laboratorio de corrosión y va a ser manipulado por expertos e inexpertos en la materia. Por tal motivo el generador debe cumplir con las normas de seguridad eléctrica para ser manipulado por cualquier persona. Esta legislación se encuentra contenida en la CEI 61010 – Parte 1, expedida por la Comisión Electrotécnica Internacional.

En esta norma se especifican las condiciones que deben cumplir los equipos para garantizar una protección adecuada del operador y del medio circundante, como por ejemplo, condiciones contra choques eléctricos, quemaduras, temperaturas excesivas, placas de advertencia y blindajes.[3]

[2] Impedance Instrumentation and Techniques. Aplication Note AC-3.
[3] Norma CEI 61010-Parte 1. Artículos 6, 9, 10 y notas aclaratorias.

2. DISPOSITIVOS UTILIZADO PARA LA CONSTRUCCIÓN DEL GENERADOR DE SEÑALES SENOIDALES

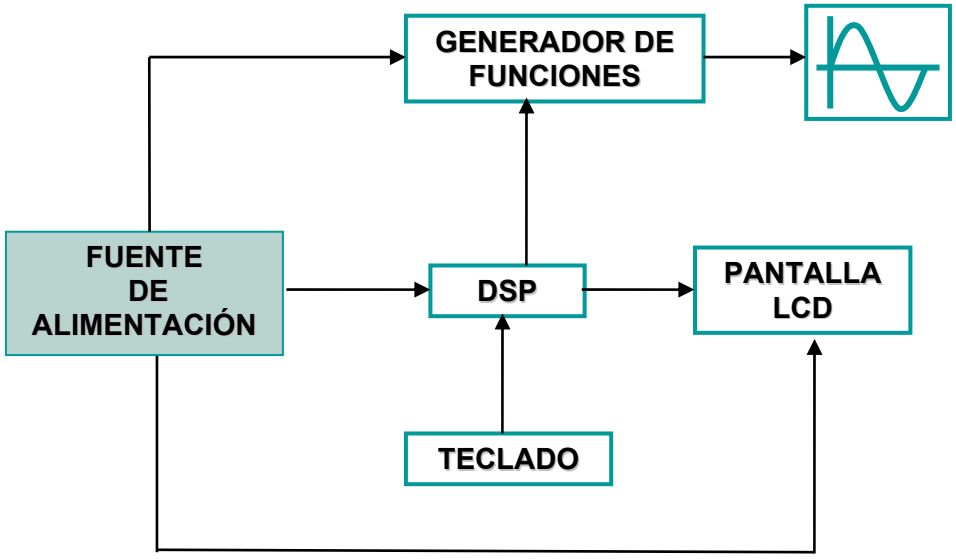
Como se ha mencionado anteriormente, se desea construir un generador de señales senoidales para mejorar los medidores de impedancia electroquímica de los laboratorios de corrosión. Este generador de señales cumple con las especificaciones que se describen a continuación:

- El equipo genera señales senoidales de tensión dentro de un rango de amplitud entre 30-50 [mV].
- Genera señales senoidales de tensión con frecuencias que varían dentro de un rango de 0,1Hz-100 kHz.
- El generador presenta una distorsión armónica de 1,14% y una impedancia de salida igual a cero.

Para lograr este objetivo, el diseño general del generador de señales se dividió en 4 etapas:

- Control basado en el DSP 56F8323
- Generación de la señal (AD9833)
- Interfaz de comunicación con el usuario
- Fuente de alimentación

Figura 2 Diagrama de Bloques del Generador de Señales Senoidales.



Fuente: Autores del Proyecto.

A continuación se describen las 4 etapas. Se realiza la descripción de todos los dispositivos que intervienen en cada una de ellas y se presentan los parámetros más importantes para su implementación.

2.1 CONTROL BASADO EN EL DSP 56F8323

El generador de señales senoidales es la primera etapa de un macro-proyecto, para implementar un medidor de impedancia electroquímica controlado por DSP. Por esta razón, este diseño se basa en el uso de un DSP que controla todo el funcionamiento del generador y es el punto de comunicación entre el generador de funciones, el teclado y la pantalla LCD.

Se optó por la utilización del procesador digital de señales DSP 56F8323 de Motorola, puesto que además de ser sencillo de manejar, es de bajo costo, es robusto y cuenta con un software de programación con licencia gratuita,

llamado Code Warrior, que permite trabajar en lenguaje de alto nivel, simplificando el trabajo en el entorno de programación.

Las funciones que realiza el DSP 56F8323 son las siguientes:

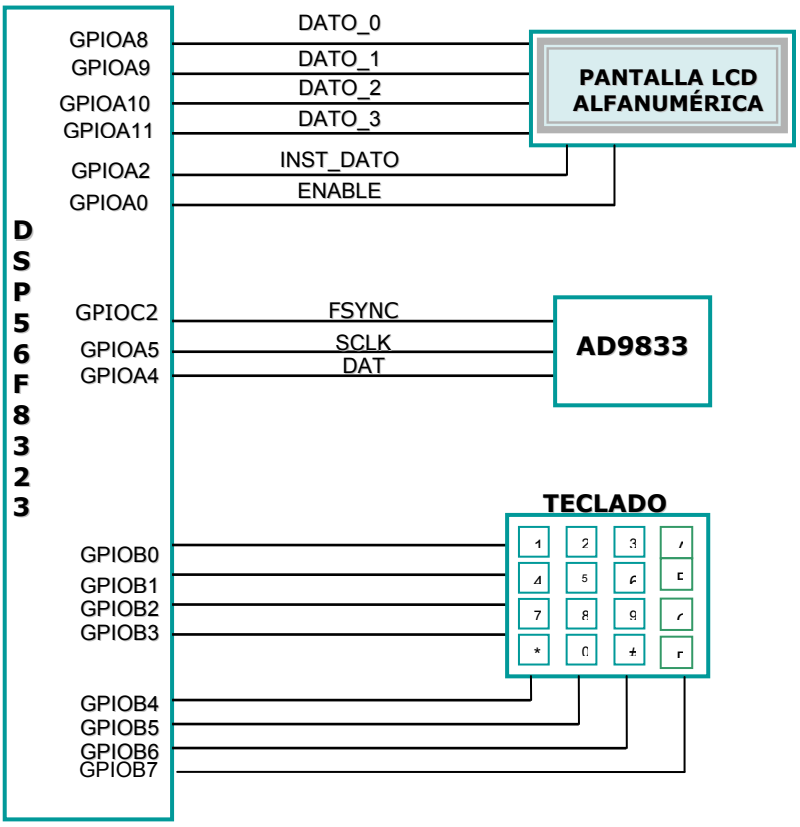
- Control de frecuencia para la señal senoidal mediante el envío de códigos al generador de funciones de la empresa Analog Devices para lograr un rango de frecuencia entre 0,1Hz-100kHz, como se mencionó anteriormente.

- Control de la pantalla LCD, necesaria para la visualización de los parámetros de frecuencia escogidos por el usuario.

- Adquisición de la información de frecuencia de la señal senoidal deseada, que es introducida por el usuario por medio del teclado.

Todos estos procesos de control se lograron mediante la utilización de los pines de propósito general (GPIO), el módulo SPI (Serial Peripheral Interface) y las interrupciones externas con que cuenta el DSP. Para tener una idea mas clara de la distribución de pines utilizados en el DSP, en la Figura 3, se muestra un diagrama de control del DSP 56F8323 y los elementos a controlar.

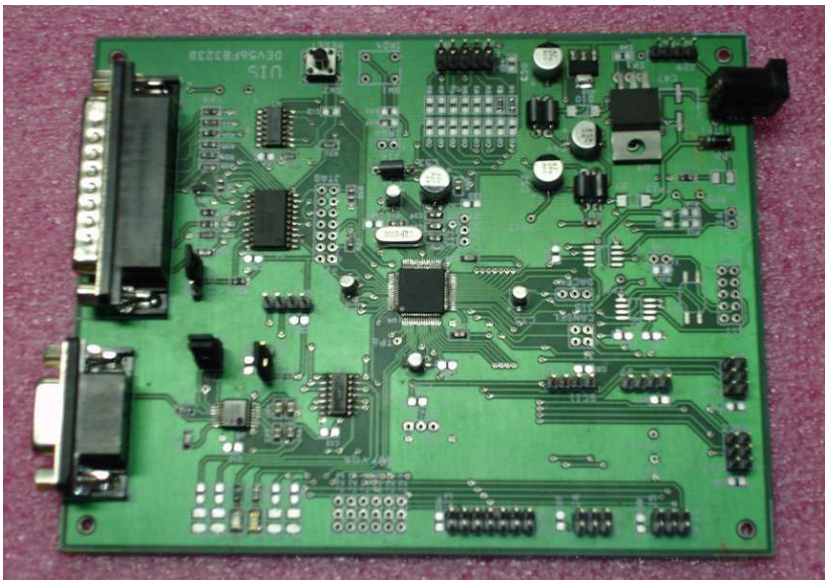
Figura 3 Diagrama de Control del DSP 56F8323.



Fuente: Autores del Proyecto.

Para este proyecto se implementó la tarjeta de desarrollo del DSP 56F8323. La tarjeta fue facilitada por la universidad y los implementos fueron importados por los autores del proyecto. Ésto fue de gran ayuda para realizar la conexión de los pines del DSP y sirvió como interfaz entre el DSP, el generador de funciones, la pantalla LCD y el teclado. En la Figura 4, se muestra la tarjeta de desarrollo que se utilizó.

Figura 4 Tarjeta de Desarrollo del DSP 56F8323.



Fuente: Autores del Proyecto.

2.2 GENERACIÓN DE LA SEÑAL

La principal parte del diseño del generador de señales es precisamente la generación de las señales senoidales de tensión. Para tal fin se utilizó el generador de forma de onda programable, AD9833, de la empresa Analog Devices. Este generador, mediante programación, toma la información proveniente del DSP y la convierte en una señal analógica con valores de frecuencia y amplitud específicos.

Este circuito integrado es un generador programable de funciones de baja potencia, capaz de producir formas de ondas cuadradas, triangulares y sinusoidales. Las frecuencias y fases de salida son totalmente programables por software, lo que permite una fácil sintonización. Opera con una fuente de alimentación de 2,3V a 5,5V.

Una de las características más atractivas de este dispositivo es que sólo requiere una referencia de reloj y algunos capacitores de desacople para su funcionamiento. Además, está diseñado para una interfaz serial de tres hilos que puede operar a tasas de reloj superiores a 40 [MHz], siendo de esta manera compatible con DSPs y micro-controladores.

2.2.1 Descripción del circuito

El AD9833 es un chip sintetizador digital directo (DDS) totalmente integrado. Requiere de un reloj de referencia, un resistor de baja precisión, y un capacitor de desacople para producir ondas sinusoidales creadas digitalmente de hasta 12,5 [MHz].

Las ondas sinusoidales son típicamente mostradas en términos de su magnitud a(t) = sin(ωt). Sin embargo, estas son no lineales y no son fáciles de generar excepto cuando se usa construcción por tramos. Por otro lado, la información angular es por naturaleza lineal. Esto es, que el ángulo de fase va rotando a través de un ángulo fijo por cada unidad de tiempo. La frecuencia angular depende de la frecuencia de la señal en la cantidad tradicional de

$$\omega=2\pi f. \qquad (1)$$

Conociendo que la fase de la onda seno es lineal y dando un intervalo de referencia (periodo de reloj), la rotación en fase para ese periodo puede ser determinada por la ecuación 2,

$$\Delta Phase = \omega\Delta t \qquad (2)$$

Resolviendo para ω

$$\omega = \Delta Phase / \Delta t = 2\pi f \qquad (3)$$

Resolviendo para f y sustituyendo la frecuencia de reloj de referencia por el periodo de referencia $(1/f_{MCLK} = \Delta t)$,

$$f = \Delta Phase \times f_{MCLK} / 2\pi \qquad (4)$$

El AD9833 construye la salida basándose en la ecuación (4). El integrado puede implementar la ecuación con la ayuda de los siguientes sub-circuitos importantes:

- **Oscilador controlado numéricamente (NCO) mas moduladores de frecuencia y fase:** Esta formado por 2 registros selectivos de frecuencia, 1 acumulador de fase de 28 bits (componente principal), 2 registros de fase inicial y 1 un sumador de fase.

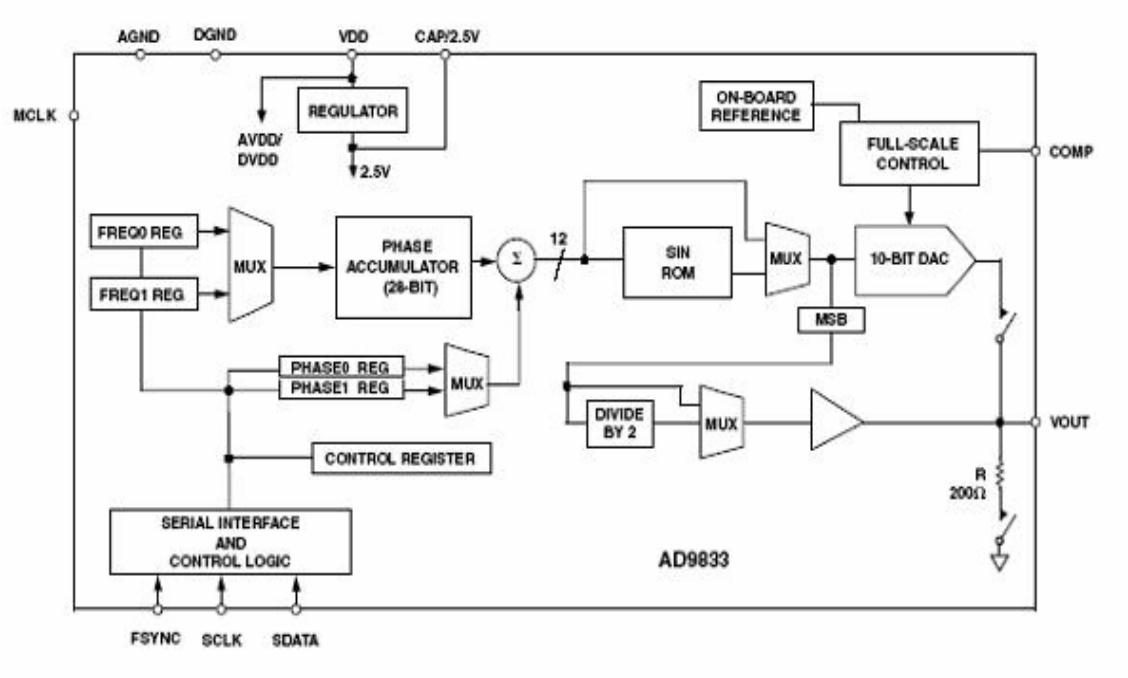
- **SIN-ROM:** Utiliza la información de fase digital como una dirección que es buscada en una tabla y convierte la información de fase en una amplitud. Es habilitada utilizando el bit MODE (D_1) en el registro de control.

- **Convertidor Digital-Analógico:** Este recibe las palabras digitales del SIN-ROM y las convierte en las correspondientes tensiones analógicas. Genera una salida de tensión de $0{,}6V_{pp}$.

- **Regulador:** El regulador interno disminuye la tensión aplicada por V_{DD} a 2,5[V] que requiere la sección digital interna del integrado, para su correcta operación.

En la Figura 5, se muestra el diagrama de bloques funcional del integrado, donde se encuentra el conjunto de módulos anteriormente mencionados.

Figura 5 Diagrama de Bloques Funcional del integrado AD9833.

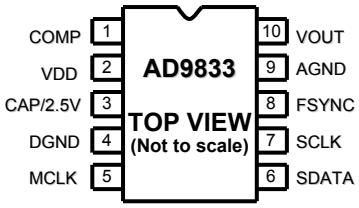


Fuente: Hoja de datos del Fabricante.

2.2.2 Configuración de pines

El generador de funciones AD9833 cuenta con 10 pines como se pueden observar en la Figura 6. Los pines 2, 3, 4 y 9 están configurados para la fuente de alimentación. Los pines 1 y 10 para señal analógica y los pines 5, 6, 7 y 8 para control e interfaz digital. En la Tabla 1, se presenta una breve descripción de cada pin.

Figura 6 Configuración de pines AD9833.



Fuente: Hoja de datos del Fabricante.

Tabla 1 Descripción de pines AD9833

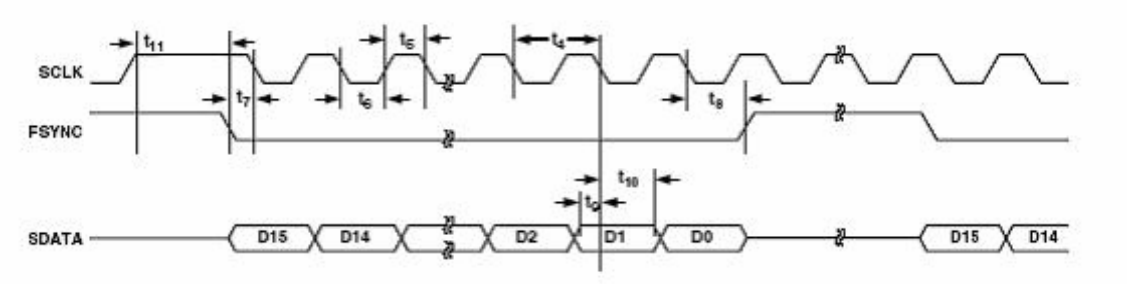
NÚMERO DEL PIN	NOMBRE DEL PIN	DESCRIPCIÓN
1	COMP	Pin especial para el DAC. Es utilizado para desacoplar la tensión de polarización del DAC.
2	VDD	Fuente de alimentación positiva para las secciones de interfase analógica y digital.
3	CAP/2.5V	Tensión para la operación de la parte digital generada por VDD.
4	DGND	Tierra Digital.
5	MCLK	Entrada de reloj digital. La salida DDS está expresada como una fracción binaria de este reloj.
6	SDATA	Entrada serial de datos. En esta entrada se aplican los 16 bits de datos seriales.
7	SCLK	Entrada serial de reloj.
8	FSYNC	Señal de sincronización para la entrada de datos. Trabaja con flanco de bajada.
9	AGND	Tierra analógica.
10	VOUT	Tensión de salida. No requiere resistencia de carga en la salida.

Fuente: Autores del Proyecto.

2.2.3 Modo de funcionamiento

El generador de funciones que se utilizó es compatible con QSPI, MICROWIRE, SPI y DSP's gracias a la interfaz de comunicación serial de 3 hilos. El dato es cargado en el dispositivo como una palabra de 16 bits según el diagrama de tiempos serial que se muestra en la Figura 7.

Figura 7 Diagrama de Tiempos Serial.



Fuente: Hoja de datos del Fabricante.

En el diagrama se observan las 3 entradas que tiene el integrado: SCLK, FSYNC y SDATA. La entrada FSYNC es la entrada de disparo que sincroniza y habilita el chip. Para que se inicie la transferencia de un dato al dispositivo, la entrada FSYNC debe estar en nivel bajo, de esta manera el dato es desplazado mediante el registro de desplazamiento durante 16 pulsos de reloj. Cuando se ha terminado la transferencia, FSYNC toma un nivel alto. Por tanto, un continuo flujo de palabras de 16 bits puede ser cargado mientras FSYNC es mantenida en nivel bajo, lo cual ocurre hasta después del dieciseisavo flanco de bajada de SCLK. Este reloj puede ser continua o alternativamente alto o bajo entre operaciones de escritura, pero debe ser alto mientras FSYNC esté en bajo.

Al momento de colocar a funcionar el AD9833 se deben tener en cuenta algunas consideraciones:

El integrado debe reiniciarse. Con esto se logra que los registros internos se coloquen en cero para proporcionar una salida analógica de media escala. El bit RESET debe estar en 1 mientras el dispositivo esté leyendo para empezar a generar una salida y en 0 al empezar a generar una salida. El bit RESET no resetea la fase, frecuencia o registros de control.

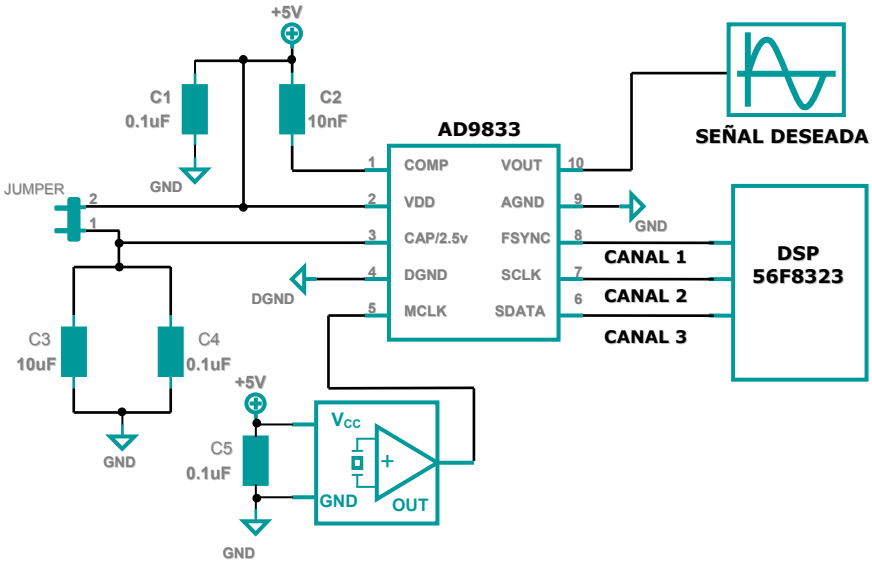
El AD9833 contiene un registro de control de 16 bits que puede ser operado como se desee según la aplicación. Todos los bits de control, excepto MODE, son muestreados en el flanco de bajada del MCLK. Para informar al AD9833 que los contenidos del registro de control serán alterados, D15 y D14 deben colocarse en 0.

La SIN ROM es usada para convertir la información de fase desde los registros de frecuencia y fase, en información de amplitud que resulta en una señal senoidal a la salida. Para tener una salida senoidal desde el pin VOUT, se debe colocar D1 (MODE) en 0 y el D5 (OPBITEN) en 0.

2.2.4 Esquema de conexión del AD9833

Para lograr el funcionamiento del AD9833 y obtener la senoidal deseada, se realizaron las conexiones que se muestran en la Figura 8.

Figura 8 Esquema de Conexión del AD9833

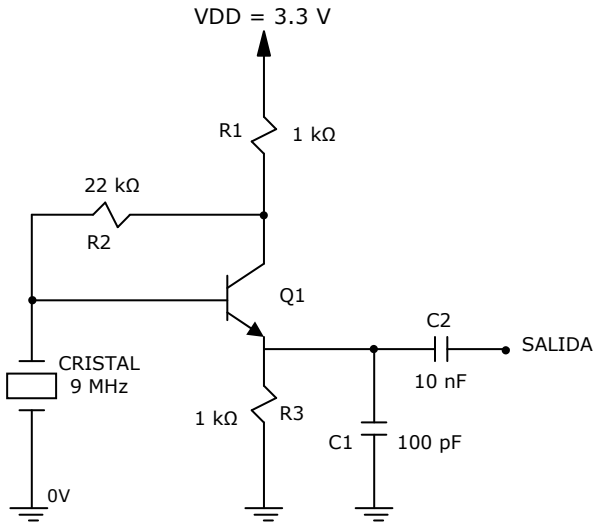


Fuente: Autores del Proyecto.

Se utilizaron condensadores de desacople según recomendaciones del fabricante del integrado. Las entradas SDATA, SFYNC y SCLK provienen del DSP. Todo el circuito es alimentado a una tensión de 3,3 [V].

Es importante tener en cuenta que el integrado necesita de un oscilador externo para su funcionamiento. Este oscilador se conecta al pin 6 (MCLK) del integrado. Para este proyecto se implementó un oscilador de cristal, como el que se muestra en la figura 9.

Figura 9 Circuito del Oscilador Externo.



Fuente: Autores del Proyecto

Se escogió un oscilador de cristal por sus características de resonancia electromecánica que son muy estables y altamente selectivas. El oscilador opera a una sola frecuencia, la cual se define por el cristal, que en este caso es de 9MHz. Se realizaron pruebas con otros cristales y se comprobó que la configuración funciona adecuadamente para cristales entre 5 y 10MHz.

2.2.5 Circuito atenuador para la amplitud de la señal

Una de las especificaciones que debe cumplir el generador de señales diseñado es que la amplitud de las funciones generadas esté en el rango de 30-50 [mV]. La onda sinusoidal que se obtiene del generador de funciones AD9833 tiene una amplitud de tensión pico fijo de 316 [mV]. Para lograr colocar la amplitud en el rango deseado es necesario un circuito atenuador.

Para este proyecto se realizaron pruebas con diferentes circuitos como por ejemplo, divisores de tensión, configuraciones de amplificadores operacionales con ganancias cercanas a la unidad, atenuadores basados en diodos zener, configuraciones pi y potenciómetros. Después de comparar los resultados obtenidos con cada uno de estos circuitos, se decidió realizar la atenuación de la señal por medio de una resistencia de 10 [kΩ] colocada en el pin de salida del integrado AD9833.

La resistencia realiza una atenuación considerable de la señal de salida, obteniéndose una tensión pico a pico de 82 [mV], valor que si cumple con el objetivo propuesto. Además la señal obtenida después de la resistencia conserva la forma senoidal de la señal y presenta una distorsión máxima de 1.14%.

Para poder calcular el valor de la resistencia y obtener la tensión deseada, se realizó el cálculo de la impedancia de salida del AD9833. Este cálculo se presenta en el capítulo 4. Teniendo dicho valor de impedancia, el valor de la resistencia se calculó por medio de un divisor de tensión. Si no se tiene en cuenta este valor, se debe utilizar una resistencia más y por tanto se produce mas ruido en la señal de salida.

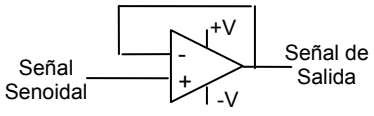
2.2.6 Circuito seguidor de tensión

Cuando se empezaron a hacer las pruebas de las señales generadas, colocando diferentes valores de cargas resistivas, se observó que el nivel de tensión caía considerablemente. Por tal razón, se decidió colocar un amplificador operacional en configuración de seguidor de tensión para fortalecer la etapa de salida del generador diseñado.

Esta configuración fue de gran ayuda, no solo para mantener el nivel de tensión al colocarle una carga sino también para lograr que la impedancia de salida del generador diseñado fuera muy pequeña (en este caso cero), uno de los objetivos del proyecto.

El circuito implementado se muestra en la figura 10. Se realizaron pruebas con diferentes integrados como por ejemplo, LM741, LF356 y LF353. El circuito se implementó con el último integrado, LF353, ya que presentó el nivel de offsett mas bajo de los tres. Además presenta un ancho de banda grande, el cual garantiza que las señales de frecuencias altas no se pierdan ni se distorsionen.

Figura 10 Circuito seguidor de tensión



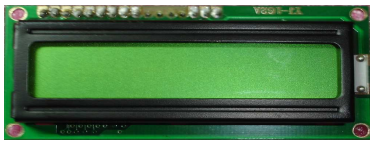
Fuente: Autores del Proyecto

2.3 INTERFACE DE COMUNICACIÓN CON EL USUARIO

2.3.1 Pantalla LCD

Para la comunicación entre el usuario y el dispositivo, se utilizó una pantalla LCD alfanumérica de 2x16 líneas, en donde se puede visualizar las funciones que ofrece el generador y los datos introducidos mediante el teclado. La pantalla se muestra en la Figura 11.

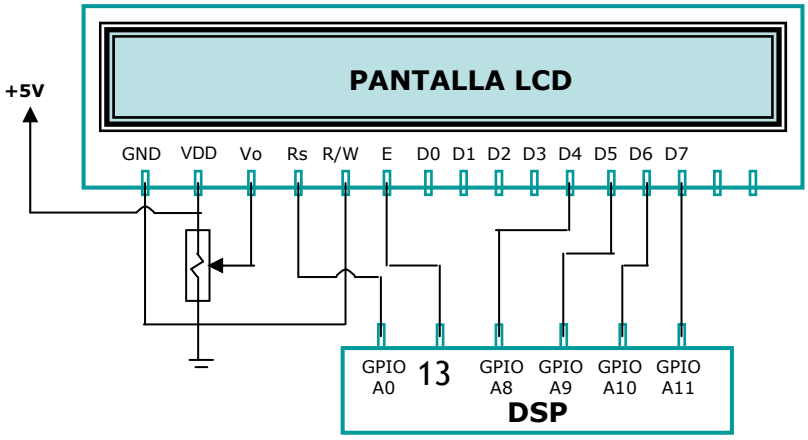
Figura 11 Pantalla LCD alfanumérica 2x16.



Fuente: Autores del Proyecto.

El DSP controla el funcionamiento de la pantalla LCD mediante seis pines de propósito general que son los encargados de llevar los datos, habilitar la pantalla y habilitar el registro de datos o el registro de control de la pantalla. Además para el funcionamiento la pantalla, se utiliza una tensión de alimentación de 5 [V] y se utiliza un potenciómetro para controlar el contraste del cristal líquido. El circuito de conexión se muestra en la Figura 12.

Figura 12 Esquema de conexión para la Pantalla LCD.



Fuente: Autores del Proyecto.

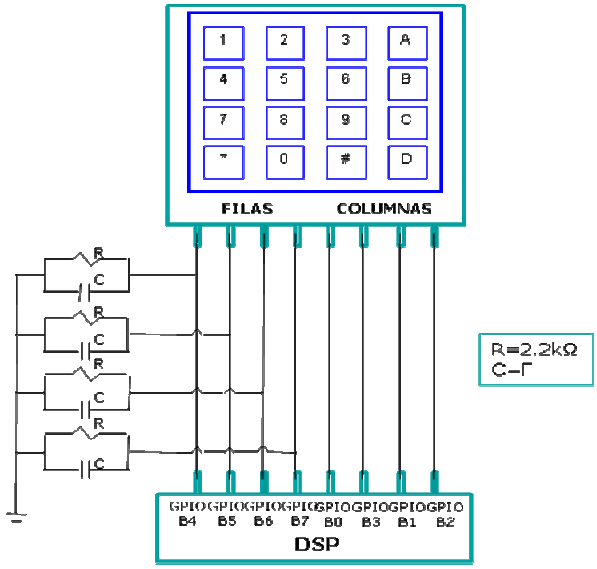
2.3.2 Teclado 4x4

La selección de los valores de frecuencia para el generador de señales diseñado se realiza por medio de un teclado matricial 4x4 como el mostrado en la figura 13. Este teclado recibe los valores de frecuencia de la onda sinusoidal que el usuario desea obtener. Para su correcto funcionamiento se implementó el esquema de conexión que se muestra en la figura 14. Los filtros que se muestran en el esquema son necesarios para eliminar los rebotes que se presentan al oprimir cada una de las teclas.

Figura 13 Teclado Matricial 4x4.

Fuente: Autores del Proyecto.

Figura 14 Esquema de conexión para el teclado matricial 4x4.



Fuente: Autores del Proyecto.

2.4 FUENTE DE ALIMENTACIÓN

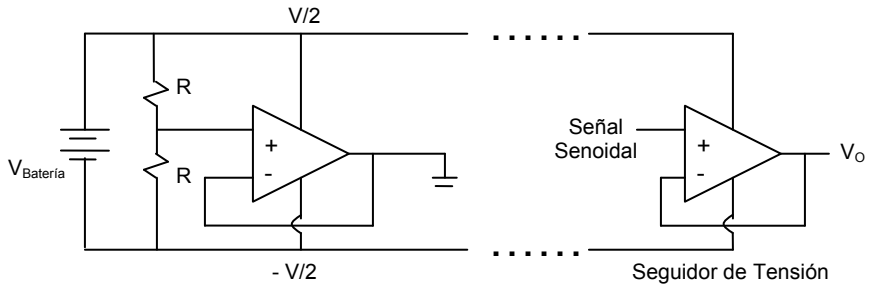
El equipo diseñado fue implementado con tres tarjetas: la primera es la tarjeta de desarrollo del DSP, la segunda corresponde a la tarjeta del generador de señales diseñado y la tercera tarjeta contiene los circuitos implementados para la pantalla LCD y el teclado.

La tensión de alimentación de la tarjeta de desarrollo se realiza por medio de un adaptador de 6V.

La segunda tarjeta contiene la configuración del generador de funciones AD9833. En esta tarjeta se manejan tensiones de alimentación de 3,3 [V] y -3,3 [V]. El integrado AD9833 y el oscilador externo se alimentan a 3.3 [V]. Este valor fue tomado de uno de los pines de la tarjeta del desarrollo del DSP.
Por otro lado, en esta tarjeta también se encuentra el integrado LF353. Este integrado es el seguidor de tensión que se implementó a la salida. Para su correcto funcionamiento, el integrado necesita una tensión de alimentación dual. Ya que la tarjeta de desarrollo del DSP no tiene tensiones negativas y no se pudo conseguir un regulador de tensión que realizara el cambio de polaridad, se analizaron dos posibilidades que se describen a continuación.

- Se probó una configuración basada en un amplificador operacional, dos resistencias de igual valor y una batería como se muestra en la figura 15. Teóricamente con esta configuración se obtiene una tensión dual igual a la mitad de la tensión de la batería de tal forma que cuando se conecte el seguidor de tensión, éste se alimenta de las tensiones obtenidas. Esta configuración es interesante ya que la misma batería alimenta al circuito y a la vez se obtiene la tensión deseada.

Figura 15 Regulador de tensión dual con un amplificador operacional

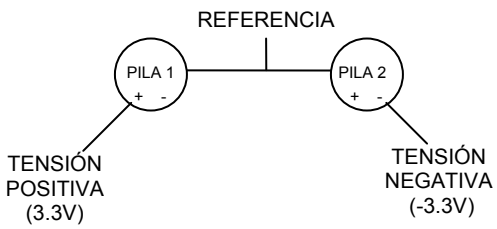


Fuente: Manual de Electrónica Básica. SENA

Cuando se implementó esta configuración, se utilizó una batería de 5 [V] y efectivamente se obtuvieron los niveles de tensión deseados pero al observarlos en el osciloscopio, la señal de continua presentaba mucha distorsión y algunos picos los cuales afectaban notablemente el funcionamiento del seguidor de tensión, deteriorando la señal de salida del generador.

- Como no se logró los resultados esperados, la fuente de alimentación dual se implementó por medio de dos pilas de 3.3 [V] conectadas en serie. La configuración se muestra en la figura 16.

Figura 16 Esquema de conexión para la fuente dual.



Fuente: Autores del Proyecto

La tercera tarjeta contiene los circuitos para la LCD y el teclado. Esta tarjeta tiene una alimentación de 5 [V]. Dicho valor se tomó también de la tarjeta de desarrollo del DSP. Para mayor claridad, los dispositivos, sus valores de alimentación y la potencia consumida por cada uno, se encuentran resumidos en la tabla 2.

Tabla 2 Alimentación utilizada para cada uno de los dispositivos.

DISPOSITIVO	NIVEL DE TENSIÓN [V]	CONSUMO DE CORRIENTE [mA]	CONSUMO DE POTENCIA [mW]
AD9833	3,3	5,5	18,15
LF353	3,3	3,2	10,56
Pilas	3,3	160	528
Pantalla LCD Genérica	5,0	3	15
Teclado 4x4	5,0	0,1	0,5
Oscilador de Cristal	3,3	1,27	4,19
TOTAL		173,07	576,4

Fuente: Autores del Proyecto

3. SOFTWARE UTILIZADO PARA LA CONSTRUCCIÓN DEL GENERADOR DE SEÑALES SENOIDALES.

En el diseño del generador de señales se utilizaron diferentes dispositivos controlados digitalmente los cuales necesitan de rutinas de programación para su inicialización y correcto funcionamiento.

Dichas rutinas se realizaron en lenguaje de programación C bajo la plataforma CodeWarrior Development Studio for Freescale 56800/E Digital Signal Controllers y el Processor Expert diseñado para facilitar el desarrollo de aplicaciones, por intermedio de componentes denominados Embedded Beans. Para el uso de los Beans, se requiere configurar algunas propiedades y métodos que controlen los periféricos, los módulos PWM entre otros.

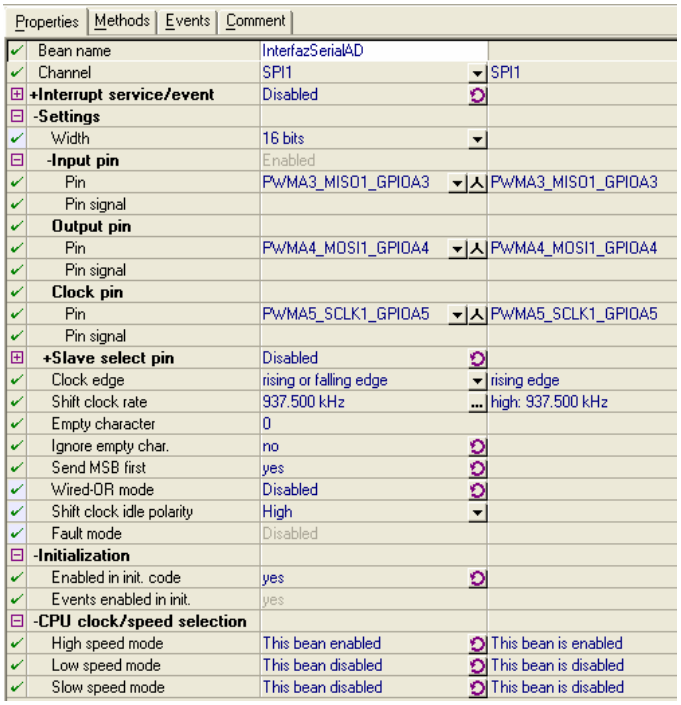
3.1 PROGRAMACIÓN PARA EL DSP56F8323.

El DSP56F8323 es el encargado de controlar casi todos los dispositivos del generador de señales. Este control lo realiza por medio de los diferentes módulos con que cuenta, los cuales deben ser configurados según su utilidad. A continuación se describe brevemente la configuración de cada módulo.

3.1.1 MÓDULO SPI

El módulo SPI, como su nombre lo indica es una interfaz periférica serial, que permite al DSP comunicarse sincrónicamente y de forma independiente con cualquier dispositivo externo. En este caso, el módulo se utilizó para comunicar el DSP con el generador de señales AD9833. En la figura 17 se muestra la configuración utilizada.

Figura 17 Propiedades del Módulo SPI.



Fuente: Autores del Proyecto.

El módulo SPI se configura en modo maestro. Los datos van desde el dispositivo maestro, que en este caso es el DSP, hasta el dispositivo esclavo, que es el integrado AD9833. Los datos van siempre en una sola dirección (comunicación simplex). Para configurar el módulo se deben tener en cuenta dos bits:

Bit SCLK: Es el encargado de la frecuencia del reloj. Para el proyecto se utilizó el pin 10 del DSP, llamado GPIOA5.
Bit MOSI: Es el encargado del envío de datos seriales desde el DSP al AD9833. Se utilizó el pin 9 del DSP, llamado GPIOA4.

Además de la configuración de los dos bits anteriormente descritos, se debe tener en cuenta las características del integrado AD9833 como por ejemplo el ancho de los datos (16 bits), la frecuencia de reloj del módulo SPI (937,5 kHz), interrupciones deshabilitadas y envío del bit mas significativo.

Por último, para el correcto funcionamiento del módulo es necesario habilitar el módulo antes de enviar cualquier dato y deshabilitarlo cuando el módulo no se esté usando. Estas órdenes se realizan por medio de los comandos enable y disable.

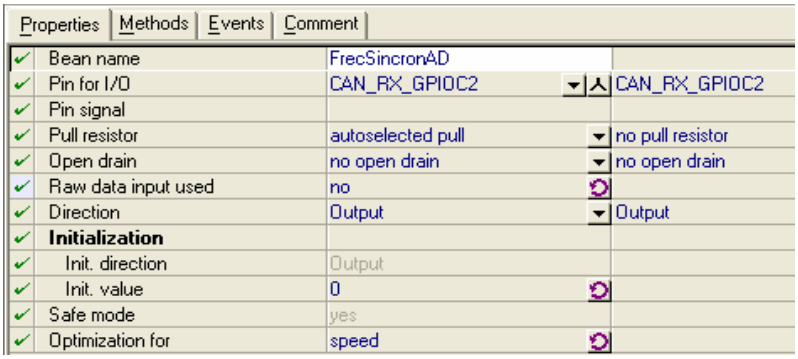
3.1.2 PINES DE PROPÓSITO GENERAL (GPIO)

Los pines de propósito general se utilizaron para comunicar el DSP con el teclado, la LCD alfanumérica y para obtener la frecuencia de sincronismo del AD9833. El DSP56F8323 cuenta con tres puertos que se pueden utilizar para propósito general. A continuación se muestra detalladamente la configuración de los Bean que se utilizaron para el teclado, la LCD y la frecuencia de sincronismo.

- *FRECUENCIA DE SINCRONISMO:* Esta frecuencia es la encargada de dejar pasar los datos al DSP. Cuando la frecuencia de sincronismo está en cero, el DSP permite la entrada de datos. Cuando la frecuencia se sincronismo está en uno, se deshabilita la entrada de datos.

Para implementar esta frecuencia, se utilizó un pin de propósito general, el cual se coloca en uno o en cero según la necesidad.

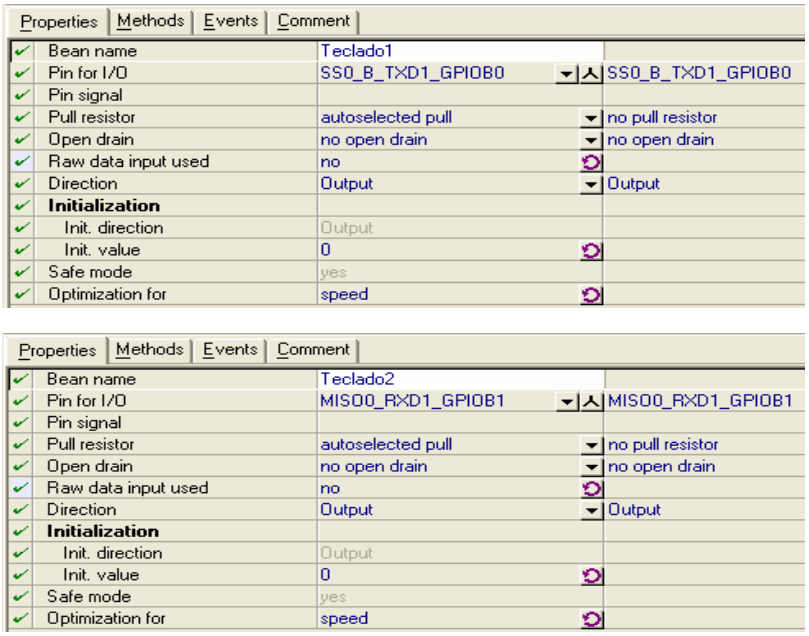
Figura 18 Propiedades del Bean Frecuencia de Sincronismo.

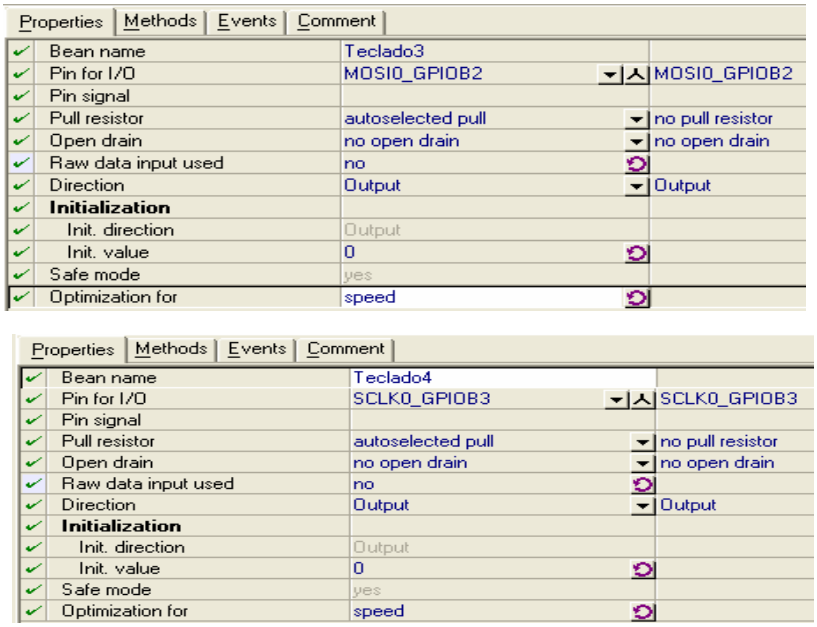


Fuente: Autores del Proyecto.

- *TECLADO:* En el diseño se utilizó el puerto B del DSP ya que se necesitó 8 pines de propósito general. Se implementaron cuatro pines como salida y cuatro pines como interrupciones externas. En la figura 19 y 20 se muestran con detalle las propiedades de cada Bean.

Figura 19 Propiedades del Bean del Teclado utilizados como salidas.



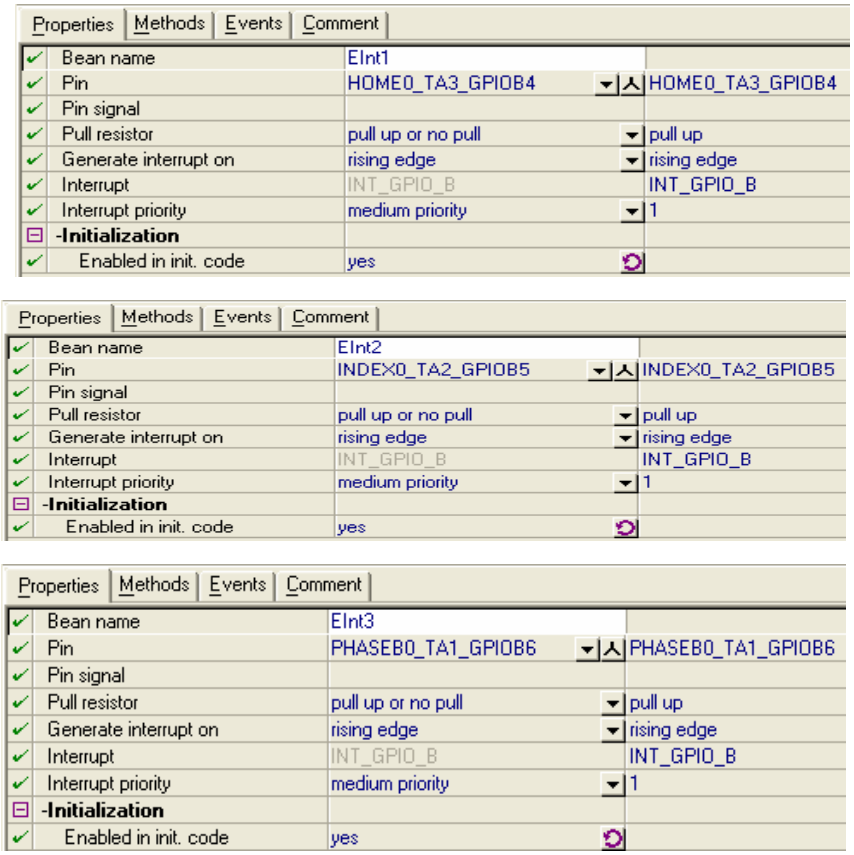


Properties	Methods	Events	Comment	
Bean name	Teclado3			
Pin for I/O	MOSI0_GPIOB2			MOSI0_GPIOB2
Pin signal				
Pull resistor	autoselected pull			no pull resistor
Open drain	no open drain			no open drain
Raw data input used	no			
Direction	Output			Output
Initialization				
Init. direction	Output			
Init. value	0			
Safe mode	yes			
Optimization for	speed			

Properties	Methods	Events	Comment	
Bean name	Teclado4			
Pin for I/O	SCLK0_GPIOB3			SCLK0_GPIOB3
Pin signal				
Pull resistor	autoselected pull			no pull resistor
Open drain	no open drain			no open drain
Raw data input used	no			
Direction	Output			Output
Initialization				
Init. direction	Output			
Init. value	0			
Safe mode	yes			
Optimization for	speed			

Fuente: Autores del Proyecto.

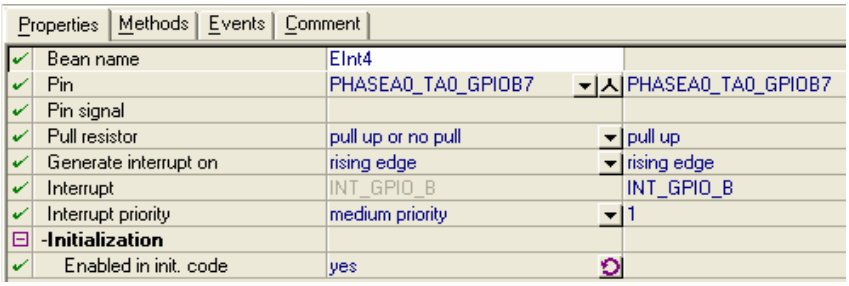
Figura 20 Propiedades del Bean del teclado utilizadas como interrupciones.



Properties	Methods	Events	Comment	
Bean name	EInt1			
Pin	HOME0_TA3_GPIOB4			HOME0_TA3_GPIOB4
Pin signal				
Pull resistor	pull up or no pull			pull up
Generate interrupt on	rising edge			rising edge
Interrupt	INT_GPIO_B			INT_GPIO_B
Interrupt priority	medium priority			1
-Initialization				
Enabled in init. code	yes			

Properties	Methods	Events	Comment	
Bean name	EInt2			
Pin	INDEX0_TA2_GPIOB5			INDEX0_TA2_GPIOB5
Pin signal				
Pull resistor	pull up or no pull			pull up
Generate interrupt on	rising edge			rising edge
Interrupt	INT_GPIO_B			INT_GPIO_B
Interrupt priority	medium priority			1
-Initialization				
Enabled in init. code	yes			

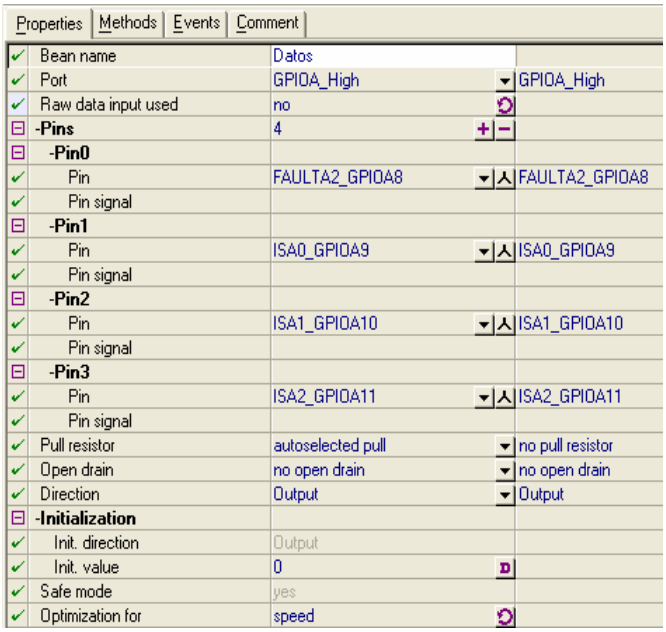
Properties	Methods	Events	Comment	
Bean name	EInt3			
Pin	PHASEB0_TA1_GPIOB6			PHASEB0_TA1_GPIOB6
Pin signal				
Pull resistor	pull up or no pull			pull up
Generate interrupt on	rising edge			rising edge
Interrupt	INT_GPIO_B			INT_GPIO_B
Interrupt priority	medium priority			1
-Initialization				
Enabled in init. code	yes			



Fuente: Autores del Proyecto.

- *PANTALLA LCD:* Se utilizó una pantalla alfanumérica 2x16. Para su control se utilizaron seis bits en total. Cuatro de ellos se configuraron como salida, por medio de los cuales el DSP envía un dato o una instrucción a la pantalla. Estos bits se agruparon en un solo Bean BitsIO.

Figura 21 Propiedades del Bean Datos LCD.

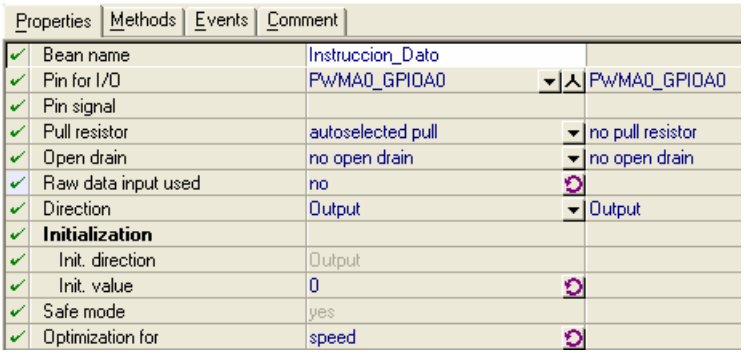


Fuente: Autores del Proyecto.

Los otros dos bits restantes, corresponden a los bits de habilitación y el bit que identifica si es una instrucción o un dato lo que se le está

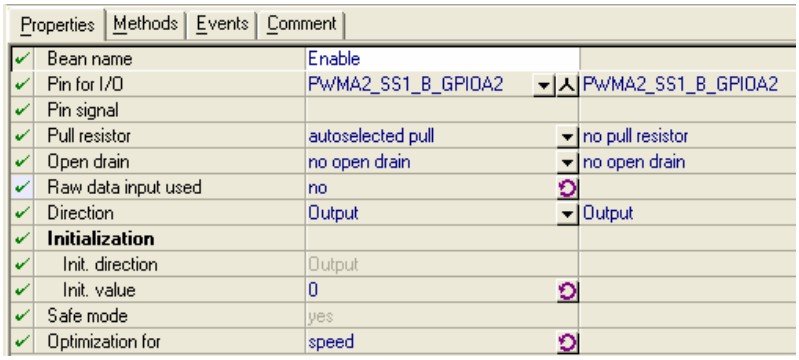
enviando a la pantalla. Estos bits fueron configurados como salidas y mediante un Bean BitIO como se muestra en las figuras.

Figura 22 Propiedades del Bean Instrucción o Dato a la LCD.



Fuente: Autores del Proyecto.

Figura 23 Propiedades del Bean Habilitación de la LCD.



Fuente: Autores del Proyecto.

Para aclarar cualquier confusión sobre los pines utilizados en el diseño del generador de señales, en la tabla 3, se hace un resumen de todos los pines utilizados y algunas características importantes.

Tabla 3 Pines del DSP utilizados en el proyecto.

PIN	NOMBRE	CONFIGURACIÓN DEL BEAN	INPUT / OUTPUT
PWMC2 (Pin 61)	FrecsincronAD	BitIO	Output
GPIOA8 (Pin 15)	Datos	BitsIO	Output
GPIOA9 (Pin 16)			Output
GPIOA10 (Pin 18)			Output
GPIOA11 (Pin 19)			Output
GPIOA0 (Pin 3)	Instrucción_Dato	BitIO	Output
GPIOA2 (Pin 7)	Enable	BitIO	Output
GPIOA5 (Pin 10)	SCLK	SPIMaster (SynchroMaster)	Output
GPIOA4 (Pin 9)	MOSI		Output
GPIOB0 (Pin 21)	Teclado	BitsIO	Output
GPIOB1 (Pin 22)			Output
GPIOB2 (Pin 24)			Output
GPIOB3 (Pin 25)			Output
GPIOB4 (Pin 49)		BitsIO	ExtInt
GPIOB5 (Pin 50)			ExtInt
GPIOB6 (Pin 51)			ExtInt
GPIOB7 (Pin 52)			ExtInt

Fuente: Autores del Proyecto.

3.2 PROGRAMACIÓN Y MANEJO DE LA PANTALLA LCD.

Como se ha mencionado en el capítulo anterior, para que el usuario visualice los datos e instrucciones que maneja el generador de señales, se utilizó una pantalla LCD alfanumérica de 2x16 líneas, cuya distribución de pines con las respectivas funciones que desempeñan se encuentran en la tabla 4.

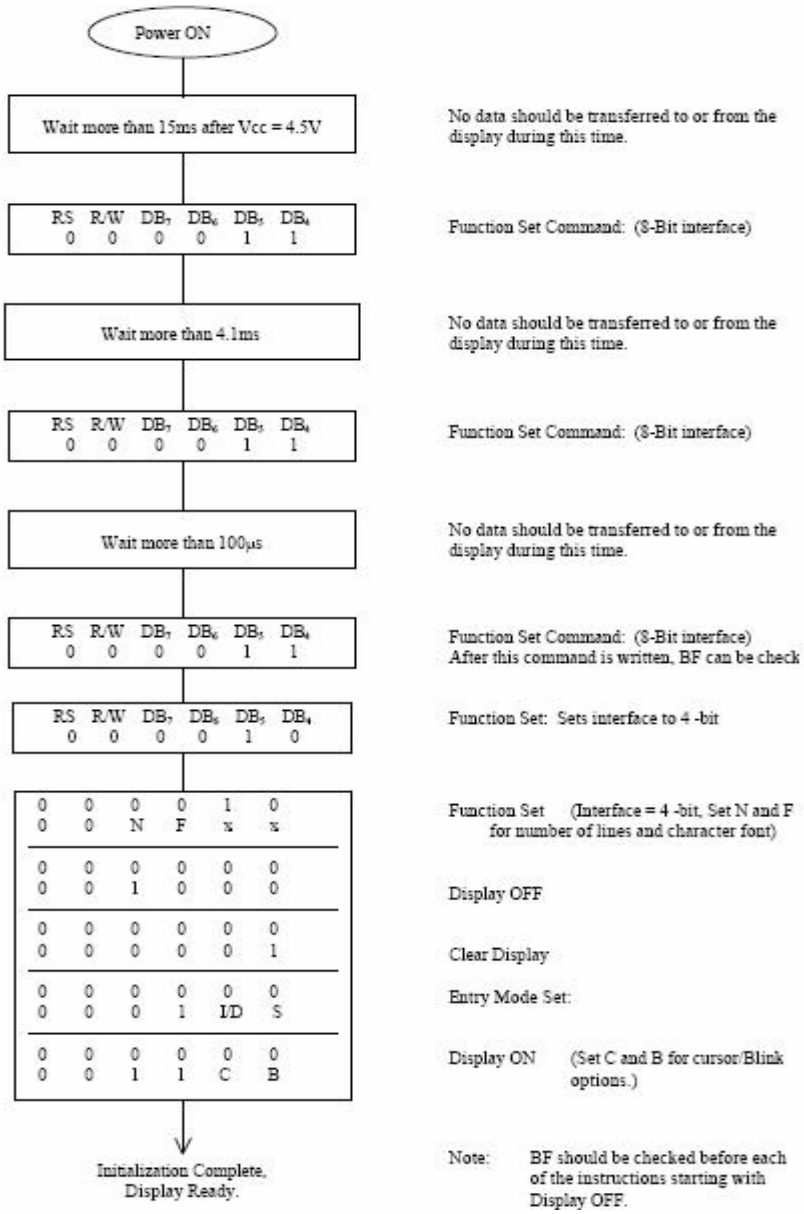
Tabla 4 Descripción y funciones de la pantalla LCD alfanumérica.

NÚMERO DEL PIN	SÍMBOLO DEL PIN	FUNCIÓN DEL PIN
1	V_{SS}	Conectar a 0 [V]
2	V_{DD}	Conectar a 5 [V]
3	V_0	Terminal para que la LCD conduzca la fuente de potencia.
4	R_S	Bit de entrada que selecciona el registro: Cero: Reg. de Instrucción. Uno: Reg. de Dato
5	R/W	Bit de entrada que selecciona la operación “Leer Dato (1)” ó “Escribir Dato (0)”
6	E	Bit de entrada que habilita las operaciones de lectura y escritura mientras esté activado.
7	D0	Bits LSBs de datos de entrada y de salida que comunica el DSP con el módulo.
8	D1	
9	D2	
10	D3	
11	D4	Bits MSBs de datos de entrada y de salida que comunica el DSP con el módulo. El bit D7 también indica “ocupado”.
12	D5	
13	D6	
14	D7	
15	E1	Bit de entrada que habilita las operaciones de lectura y escritura para 2 líneas MSBs.
16	E2	Bit de entrada que habilita las operaciones de lectura y escritura para 2 líneas LSBs.

Fuente: Autores del Proyecto.

Después de tener listo todos los dispositivos, para que la pantalla LCD funcione necesita de una rutina de inicialización. Para este proyecto se programó una de las tantas rutinas encontradas en el Manual del Usuario OPTREX DMC LCD para pantallas LCD genéricas. La rutina y la programación en C de la misma se muestran en las figuras 24 y 25.

Figura 24 Propiedades del Bean Habilitación de la LCD.



Fuente: Manual del Usuario de OPTREX DMC LCD.

Figura 25 Propiedades del Bean Habilitación de la LCD.

```
void INIC_LCD (void)
{
        RETARDO(7000);
        ESCRIBIR_INSTRUCCION(3); // Operación con 8 bits
        RETARDO(3000);
        ESCRIBIR_INSTRUCCION(3); // Operación con 8 bits
        RETARDO(3000);
        ESCRIBIR_INSTRUCCION(3); // Operación con 8 bits
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(2); // Operación con 4 bits
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(2); // Operación con 4 bits
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(8); // Se fija N=1 (2-Líneas) y F=0 (5X7 dot)
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(0); // Display Apagado
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(8); // Display Apagado
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(0); // Limpiar Display
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(1); // Limpiar Display
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(0); // Fija el Modo (Incremento/Decremento)
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(6); // I/D=1 Se fija en modo Incremento
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(0); // Display encendido
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(15); // C=1 (Encendido) B=1 (Parpadeo Encendido)
        RETARDO(2000);
}
```

Fuente: Autores del Proyecto

Además de esta rutina de inicialización, se debe tener en cuenta que la pantalla muestra dos líneas de escritura. Se debe ubicar el cursor al comienzo de cada línea. Para esto, se utiliza el siguiente código:

```
COMANDO(1);          // Limpia la LCD
ESCRIBIR(frase1);    // Escribe la frase 1 en la primera línea
COMANDO(192);        // Posiciona el cursor en la segunda línea
ESCRIBIR(frase2);    // Escribe la frase 2 en la segunda línea
```

3.3 PROGRAMACIÓN Y MANEJO DEL TECLADO 4x4.

Cualquier teclado 4x4 o 4x3 se asemeja a una matriz 4x4 o 4x3. Cada vez que se oprime una tecla, se escoge una posición de la matriz. Este principio se tuvo en cuenta para programar el código y lograr el funcionamiento del mismo.

Para que el DSP sepa cual tecla está oprimida, se configuraron 4 pines de propósito general como entradas. Estas entradas son interrupciones externas, que se activan al momento de oprimir una tecla. También se configuraron 4 pines de propósito general como salidas. Por medio de estructuras if y for como las que se muestra en la figura 26 se logró la detección de las teclas.

Figura 26 Propiedades del Bean Habilitación de la LCD.

```
////*********************************************************
 Bit3_PutVal(1);
 Bit4_PutVal(0);
 Bit5_PutVal(0);
 Bit6_PutVal(0);
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 Bit3_PutVal(0);
 Bit4_PutVal(0);
 Bit5_PutVal(0);
 Bit6_PutVal(0);
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
```

```
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}

if( alarm10==5 ){alarm10=0;alarm11=0;alarm12=0;alarm13=0;cont2=1;ZZ=1; cont=0;}
 if( alarm11==5 ){alarm10=0;alarm11=0;alarm12=0;alarm13=0;LE=1;ZZ=1;cont=cont+1;}
  if( alarm12==5 ){alarm10=0;alarm11=0;alarm12=0;alarm13=0;LE=2;ZZ=1;cont=cont+1;}
   if( alarm13==5 ){alarm10=0;alarm11=0;alarm12=0;alarm13=0;LE=3;ZZ=1;cont=cont+1;}
```

Fuente: Autores del Proyecto

Todo el programa en lenguaje C que se utilizó para el funcionamiento del generador de señales diseñado, se muestra en el anexo 1.

3.4 PROGRAMACIÓN DE LAS FRECUENCIAS DEL GENERADOR DE SEÑALES.

El generador de señales diseñado, genera señales de tensión con una amplitud fija y varias frecuencias. Estas últimas son implementadas todas por software, utilizando el generador de funciones AD9833. Como se dijo anteriormente este integrado trabaja con tres señales: SCLK, DATA y FRECSINCRON.

Para inicializar el AD9833 se utilizó la rutina mostrada en la figura 27. En ella se habilita el módulo SPI, se activa el bit de la frecuencia de sincronismo, se adecua el integrado para que reciba los datos y se inicializan los registros de frecuencia del integrado. Por último, se deshabilita el bit de la frecuencia de sincronismo para que no reciba mas datos.

Figura 27 Propiedades del Bean Habilitación de la LCD.

```
void INICIOAD9833(void)                 //Inicialización del Generador de Funciones AD9833
{
        RETARDO(100);
        FrecSincronAD_PutVal(0);        //Fija en cero el bit FSYNC para habilitar los datos
        RETARDO(100);
        InterfazSerialAD_Enable()       //Habilita el módulo SPI
        RETARDO(100);
        InterfazSerialAD_SendChar(0x2100);  //Coloca B28=1,RESET=1,FSELECT=0,PSELECT=0
        RETARDO(100);
        InterfazSerialAD_SendChar(0x4003);  //Coloca los 14LSB de 0.1Hz en FREQ0 REG
        RETARDO(100);
        InterfazSerialAD_SendChar(0x4000);  //Coloca los 14MSB de 0.1Hz en FREQ0 REG
        RETARDO(100);
        InterfazSerialAD_SendChar(0xB333);  //Coloca los 14LSB de 100kHz en FREQ1 REG
        RETARDO(100);
        InterfazSerialAD_SendChar(0x80CC);  //Coloca los 14MSB de 100kHz en FREQ1 REG
        RETARDO(100);
        InterfazSerialAD_SendChar(0xC000);  //Coloca FASE 0 en PHASE0 REG
        RETARDO(100);
        InterfazSerialAD_SendChar(0xE000);  //Coloca FASE 0 en PHASE1 REG
        RETARDO(100);
        InterfazSerialAD_SendChar(0x2000);  //Coloca B28=1,FSELECT=0,PSELECT=0
        RETARDO(100);
        FrecSincronAD_PutVal(1);        //Fija en uno el bit FSYNC para desabilitar datos
        RETARDO(100);
}
```

Fuente: Autores del Proyecto

Teniendo ya inicializado el AD9833, se pueden adquirir los datos que se introducen por medio del teclado. Esto se logra con las funciones que se muestran a continuación.

```
void SELEC_FREC_SENOIDAL(unsigned int LSB0, unsigned int MSB0, unsigned int LSB1, unsigned
int MSB1)
{
        RETARDO(100);
        FrecSincronAD_PutVal(0);        // Fija en cero el bit FSYNC para habilitar los datos
        RETARDO(100);
        InterfazSerialAD_SendChar(LSB0);// Escribe los 14LSB de la Frec deseada en FREQ0 REG
```

```
	RETARDO(100);
	InterfazSerialAD_SendChar(MSB0);// Escribe los 14MSB de la Frec deseada en FREQ0 REG
	RETARDO(100);
	InterfazSerialAD_SendChar(LSB1);// Escribe los 14LSB de la Frec deseada en FREQ1 REG
	RETARDO(100);
	InterfazSerialAD_SendChar(MSB1);// Escribe los 14MSB de la Frec deseada en FREQ1 REG
	RETARDO(100);
	InterfazSerialAD_SendChar(0x2000);// Bits de Control: B28=1, FSELECT=0, PSELECT=0, los
demás bits en cero
	RETARDO(100);
	FrecSincronAD_PutVal(1);		// Fija en uno el bit FSYNC para deshabilitar los datos
	RETARDO(100);
}

void FREQ1_REG(void)
{
	RETARDO(100);
	FrecSincronAD_PutVal(0);		// Fija en cero el bit FSYNC para habilitar los datos
	RETARDO(100);
	InterfazSerialAD_SendChar(0x2C00);// Bits de Control: B28=1, FSELECT=1, PSELECT=1, los
demás bits en cero
	RETARDO(100);
	FrecSincronAD_PutVal(1);		// Fija en uno el bit FSYNC para deshabilitar los datos
	RETARDO(100);
}
```

Cuando el usuario digita una frecuencia, se hace un barrido de las frecuencias diseñadas para encontrar el valor deseado por el usuario. Este barrido se logra por medio de estructuras if similares a la que se muestra a continuación.

```
	if(FRECUENCIA==10000)
	{
		FREQ1_REG();
		RETARDO(100);
		SELEC_FREC_SENOIDAL(0x4D15,0x4012,0x8D15,0x8012);
		RETARDO(100);
	}
```

Las direcciones que se muestra en el código, corresponden a la frecuencia en Hexa. El generador produce señales con frecuencia entre 0.1Hz y 100kHz. Todas las frecuencias posibles se muestran en la tabla 5.

Tabla 5 Descripción de pines AD9833

Frecuencia Seleccionada [Hz]	Registro FREQ0		Registro FREQ1	
	LSB0	MSB0	LSB1	MSB1
0,1	4002_H	4000_H	8002_H	8000_H
0,2	4005_H	4000_H	8005_H	8000_H
0,3	4008_H	4000_H	8008_H	8000_H
0,4	$400B_H$	4000_H	$800B_H$	8000_H
0,5	$400E_H$	4000_H	$800E_H$	8000_H
0,6	4011_H	4000_H	8011_H	8000_H
0,7	4014_H	4000_H	8014_H	8000_H
0,8	4017_H	4000_H	8017_H	8000_H
0,9	$401A_H$	4000_H	$801A_H$	8000_H
1,0	$401D_H$	4000_H	$801D_H$	8000_H
1,5	$402C_H$	4000_H	$802C_H$	8000_H
2,0	$403B_H$	4000_H	$803B_H$	8000_H
2,5	$404A_H$	4000_H	$804A_H$	8000_H
3,0	4059_H	4000_H	8059_H	8000_H
3,5	4068_H	4000_H	8068_H	8000_H
4,0	4077_H	4000_H	8077_H	8000_H
4,5	4086_H	4000_H	8086_H	8000_H
5,0	4095_H	4000_H	8095_H	8000_H
5,5	$40A4_H$	4000_H	$80A4_H$	8000_H
6,0	$40B2_H$	4000_H	$80B2_H$	8000_H
6,5	$40C1_H$	4000_H	$80C1_H$	8000_H
7,0	$40D0_H$	4000_H	$80D0_H$	8000_H
7,5	$40DF_H$	4000_H	$80DF_H$	8000_H
8,0	$40EE_H$	4000_H	$80EE_H$	8000_H
8,5	$40FD_H$	4000_H	$80FD_H$	8000_H
9,0	$410C_H$	4000_H	$810C_H$	8000_H
9,5	$411B_H$	4000_H	$811B_H$	8000_H
10	$412A_H$	4000_H	$812A_H$	8000_H
15	$41BF_H$	4000_H	$81BF_H$	8000_H
20	4254_H	4000_H	8254_H	8000_H
25	$42E9_H$	4000_H	$82E9_H$	8000_H
30	$437E_H$	4000_H	$837E_H$	8000_H
35	4413_H	4000_H	8413_H	8000_H
40	$44A9_H$	4000_H	$84A9_H$	8000_H
45	$453E_H$	4000_H	$853E_H$	8000_H
50	$45D3_H$	4000_H	$85D3_H$	8000_H
55	4668_H	4000_H	8668_H	8000_H

60	$46FD_H$	4000_H	$86FD_H$	8000_H
65	4792_H	4000_H	8792_H	8000_H
70	4827_H	4000_H	8827_H	8000_H
75	$48BC_H$	4000_H	$88BC_H$	8000_H
80	4952_H	4000_H	8952_H	8000_H
85	$49E7_H$	4000_H	$89E7_H$	8000_H
90	$4A7C_H$	4000_H	$8A7C_H$	8000_H
95	$4B11_H$	4000_H	$8B11_H$	8000_H
100	$4BA6_H$	4000_H	$8BA6_H$	8000_H
150	5179_H	4000_H	9179_H	8000_H
200	$574D_H$	4000_H	$974D_H$	8000_H
250	$5D20_H$	4000_H	$9D20_H$	8000_H
300	$62F3_H$	4000_H	$A2F3_H$	8000_H
350	$68C7_H$	4000_H	$A8C7_H$	8000_H
400	$6E9A_H$	4000_H	$AE9A_H$	8000_H
450	$746D_H$	4000_H	$B46D_H$	8000_H
500	$7A41_H$	4000_H	$BA41_H$	8000_H
550	4014_H	4001_H	8014_H	8001_H
600	$45E7_H$	4001_H	$85E7_H$	8001_H
650	$4BBB_H$	4001_H	$8BBB_H$	8001_H
700	$518E_H$	4001_H	$918E_H$	8001_H
750	5761_H	4001_H	9761_H	8001_H
800	$5D34_H$	4001_H	$9D34_H$	8001_H
850	6308_H	4001_H	$A308_H$	8001_H
900	$68DB_H$	4001_H	$A8DB_H$	8001_H
950	$6EAE_H$	4001_H	$AEAE_H$	8001_H
1000	7482_H	4001_H	$B482_H$	8001_H
1500	$6EC3_H$	4002_H	$AEC3_H$	8002_H
2000	6904_H	4003_H	$A904_H$	8003_H
2500	6345_H	4004_H	$A345_H$	8004_H
3000	$5D86_H$	4005_H	$9D86_H$	8005_H
3500	$57C7_H$	4006_H	$97C7_H$	8006_H
4000	5208_H	4007_H	9208_H	8007_H
4500	$4C49_H$	4008_H	$8C49_H$	8008_H
5000	$468A_H$	4009_H	$868A_H$	8009_H
5500	$40CB_H$	$400A_H$	$80CB_H$	$800A_H$
6000	$7B0C_H$	$400A_H$	$CB0C_H$	$800A_H$
6500	$754E_H$	$400B_H$	$B54E_H$	$800B_H$
7000	$6F8F_H$	$400C_H$	$AF8F_H$	$800C_H$
7500	$69D0_H$	$400D_H$	$A9D0_H$	$800D_H$
8000	6411_H	$400E_H$	$A411_H$	$800E_H$
8500	$5E52_H$	$400F_H$	$9E52_H$	$800F_H$
9000	5893_H	4010_H	9893_H	8010_H
9500	$52D4_H$	4011_H	$92D4_H$	8011_H
10000	$4D15_H$	4012_H	$8D15_H$	8012_H

15000	$53A0_H$	$401B_H$	$93A0_H$	$801B_H$
20000	$5A2B_H$	4024_H	$9A2B_H$	8024_H
25000	$60B6_H$	$402D_H$	$A0B6_H$	$802D_H$
30000	6740_H	4036_H	$A740_H$	8036_H
35000	$6DCB_H$	$403F_H$	$ADCB_H$	$803F_H$
40000	7456_H	4048_H	$B456_H$	8048_H
45000	$7AE1_H$	4051_H	$BAE1_H$	8051_H
50000	$416C_H$	$405B_H$	$816C_H$	$805B_H$
55000	$47F6_H$	4064_H	$87F6_H$	8064_H
60000	$4E81_H$	$406D_H$	$8E81_H$	$806D_H$
65000	$550C_H$	4076_H	$950C_H$	8076_H
70000	$5B97_H$	$407F_H$	$9B97_H$	$807F_H$
75000	6222_H	4088_H	$A222_H$	8088_H
80000	$68AC_H$	4091_H	$A8AC_H$	8091_H
85000	$6F37_H$	$409A_H$	$AF37_H$	$809A_H$
90000	$75C2_H$	$40A3_H$	$B5C2_H$	$80A3_H$
95000	$7C4D_H$	$40AC_H$	$BC4D_H$	$80AC_H$
100000	$42D8_H$	$40D6_H$	$82D8_H$	$80D6_H$

Fuente: Autores del Proyecto

Estos valores se calcularon teniendo en cuenta el oscilador externo utilizado. Este oscilador utilizó un cristal de 9MHz. Los cálculos se hicieron basados en la información suministrada por el manual del integrado AD9833. Es importante mencionar, que las direcciones son las mismas para los dos registros, pero cuando se dividen en los bits mas significativos y los menos significativos, los 2 primeros bit de cada vector cambia. Para el registro cero, son 01 y para el registro uno, son 10.

4. PRUEBAS DE LABORATORIO

Para comprobar el correcto funcionamiento del generador de señales sinusoidales implementado, se realizaron diferentes pruebas a cada uno de los componentes del mismo.

4.1 PRUEBAS PARA LA PANTALLA LCD DE 2x16 Y TECLADO 4x4

Para empezar se comprobó que la pantalla mostrara el nombre del equipo, el nombre de los autores del proyecto, los logos de la Universidad Industrial de Santander y la Escuela de Ingeniería Eléctrica, Electrónica y Telecomunicaciones. Posteriormente se comprobó que la pantalla mostrara las indicaciones para escoger la frecuencia de la señal deseada.

Figura 28 Presentación del Generador de Señales Sinusoidales.

Fuente: Autores del Proyecto

Figura 29 Indicaciones para escoger la frecuencia de la señal.

Fuente: Autores del Proyecto

Por último, se probó si la pantalla mostraba los datos que se introducían por el teclado. Se comprobó el funcionamiento de cada uno de los números, la tecla A y el enter (tecla #), este último necesario para enviar la información al AD9833 y generar la señal.

Figura 30 Frecuencia seleccionada mediante el teclado.

Fuente: Autores del Proyecto

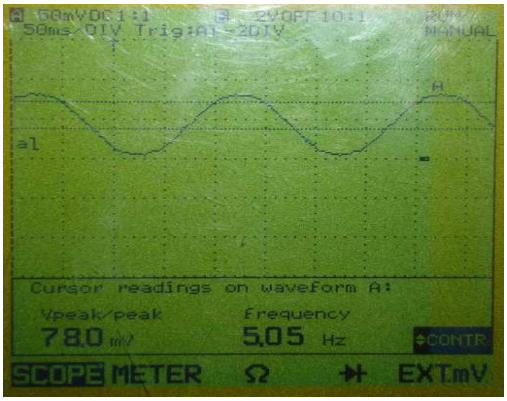
4.2 PRUEBAS EXPERIMENTALES PARA LA GENERACIÓN DE LA SEÑAL.

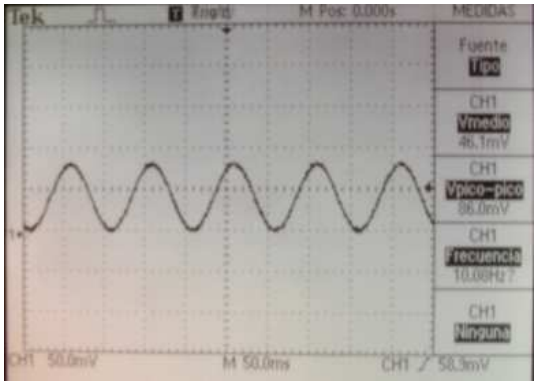
El funcionamiento del generador de señales sinusoidales diseñado, se comprobó mediante 4 pruebas experimentales. Se realizaron mediciones de las señales generadas sin carga y con carga. Se halló la impedancia de salida del integrado AD9833. Se visualizó el espectro en frecuencia de algunas de las señales generadas y se calculó la distorsión armónica. Todas estas pruebas se describen a continuación.

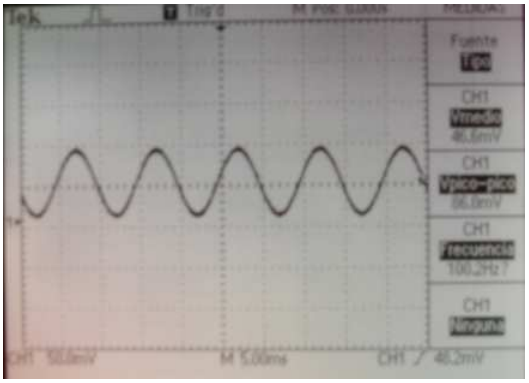
4.2.1 Señales generadas sin carga

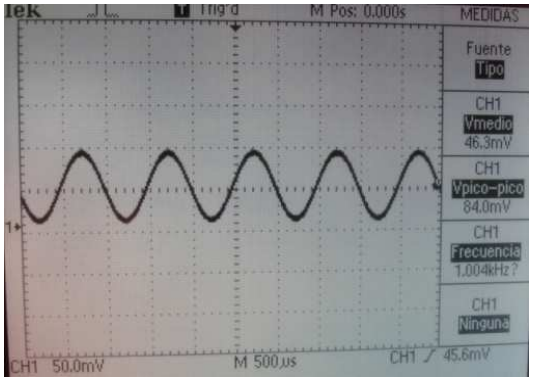
Esta prueba se realizó tomando la señal de tensión en la salida del seguidor. Se introdujeron los diferentes valores de frecuencia por medio del teclado. En la figura 31, se muestran las señales obtenidas.

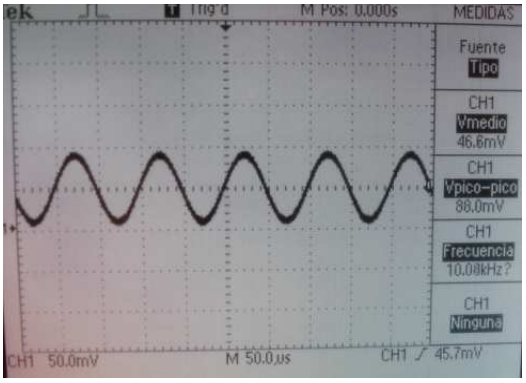
Figura 31 Señal Senoidal para 0.5Hz, 5Hz, 10Hz, 100Hz, 1kHz, 10kHz y 100kHz.

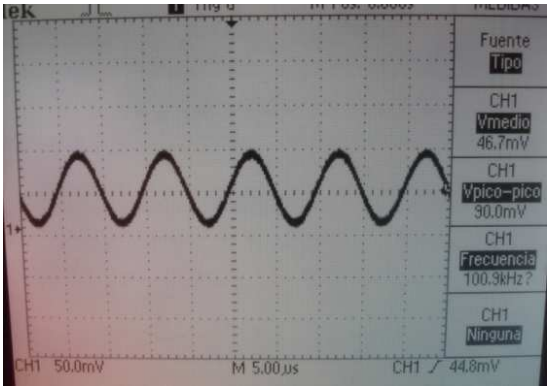












Fuente: Autores del Proyecto

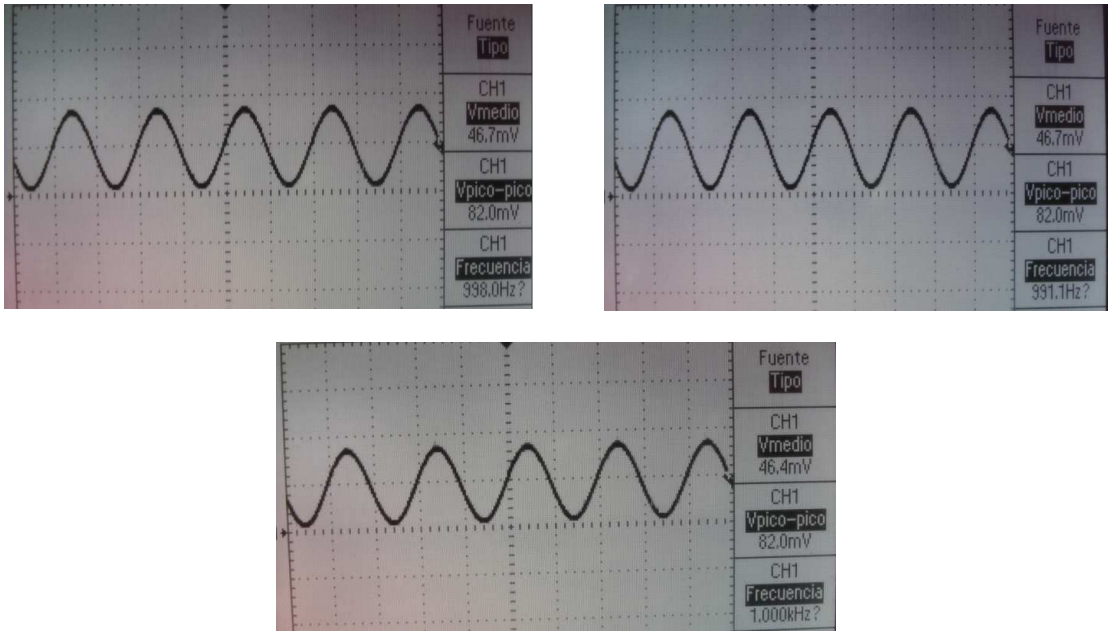
Como se puede observar en la figura 31, el generador de señales senoidales diseñado, genera señales dentro del rango de frecuencia de (0,1Hz-100kHz). Observamos que la forma de la señal y la amplitud de la misma no se alteran a medida que se varía la frecuencia. También se observa que a medida que aumenta la frecuencia a la señal se le adiciona ruido de alta frecuencia.

4.2.2 Señales generadas con carga

Para realizar las pruebas con carga, se tomaron dos tipos de cargas: una carga resistiva y una celda electroquímica dummy.

- Carga Resistiva: Se generó una señal senoidal de 82 [mV_{pp}] y con una frecuencia de 1 [kHz]. Se colocaron diferentes cargas resistivas (100Ω, 1kΩ y 33kΩ) y se observó que no se presentan caídas de tensión por efecto de la carga. Esto se logro gracias al amplificador operacional, con configuración de seguidor, colocado a la salida del generador de señales. Las señales obtenidas se muestran en la figura 32.

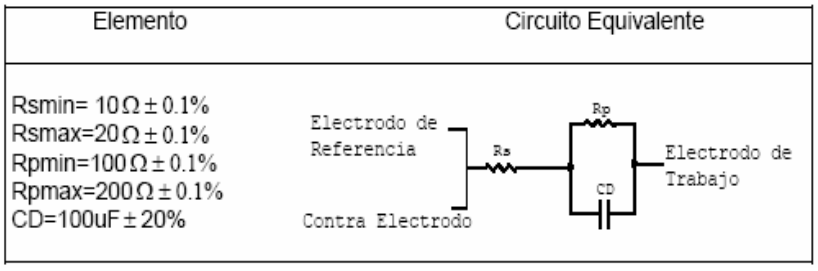
Figura 32 Señales obtenidas para una carga de 100Ω, 1kΩ y 33kΩ.



Fuente: Autores del Proyecto

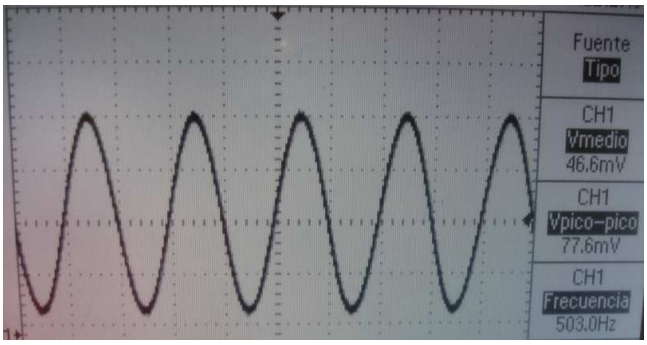
- Celda Electroquímica Dummy: Se generó una señal senoidal de amplitud 82 [mVpp] y frecuencia 500 [Hz]. Dicha señal, se conectó a una celda electroquímica dummy, la cual se muestra en la figura 33. Se observó que se presenta una disminución de la tensión aplicada. De un valor de 82 [mV_{pp}], cae a un valor de 77 [mV_{pp}], produciendo un decremento de 5 [mV]. La señal en la carga se muestra en la figura 34.

Figura 33 Celda Electroquímica Dummy



Fuente: Normas ASTM para la verificación de impedancia electroquímica

Figura 34 Señal en la Celda Electroquímica Dummy utilizada como carga



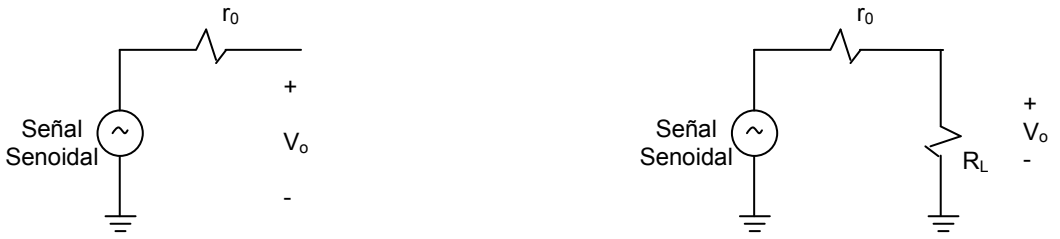
Fuente: Autores del Proyecto

4.2.3 Cálculo de la impedancia de salida del AD9833

La tercera prueba que se realizó fue el cálculo de la impedancia de salida del integrado AD9833. Esta prueba fue de gran ayuda para optimizar la parte de atenuación de la señal del generador.

El cálculo de la impedancia de salida del AD9833 se realizó por medio de un circuito de prueba, midiendo primero la tensión de circuito abierto, en los terminales de salida del integrado y luego la tensión de salida en una resistencia de carga específica.

Figura 35 Circuito de Prueba para cálculo de r_0



Fuente: Autores del Proyecto

Como se mencionó en el capítulo 2, la señal generada por el integrado AD9833 tiene una amplitud fija de 632 [mV_{pp}]. Este valor sería el valor de la

señal senoidal del circuito de la figura 35. Realizando un divisor de tensión se obtiene una expresión para hallar r_0:

$$r_0 = \frac{R_L\,(V_{señalsen} - V_0)}{V_0} \qquad (5)$$

La ecuación se resolvió para 13 valores de R_L. Luego se halló el promedio de las r_0 halladas y se obtuvo la impedancia de salida del AD9833. En la tabla 6 se muestran los cálculos realizados.

Tabla 6 Cálculos para hallar la impedancia de salida del integrado AD9833

$V_{señalsen}$ [mVpp]	V_0 [mVpp]	R_L [Ω]	r_0 [Ω]
632	168	100	276,2
	280	217	272,8
	340	324	278,2
	424	553	271,2
	496	990	271,4
	560	2150	276,4
	584	3270	268,7
	600	4910	261,8
	608	6500	256,6
	616	9800	254,5
	616	12000	311,6
	620	21500	416,12
	624	32500	416,6
Promedio r_0			**294,8**

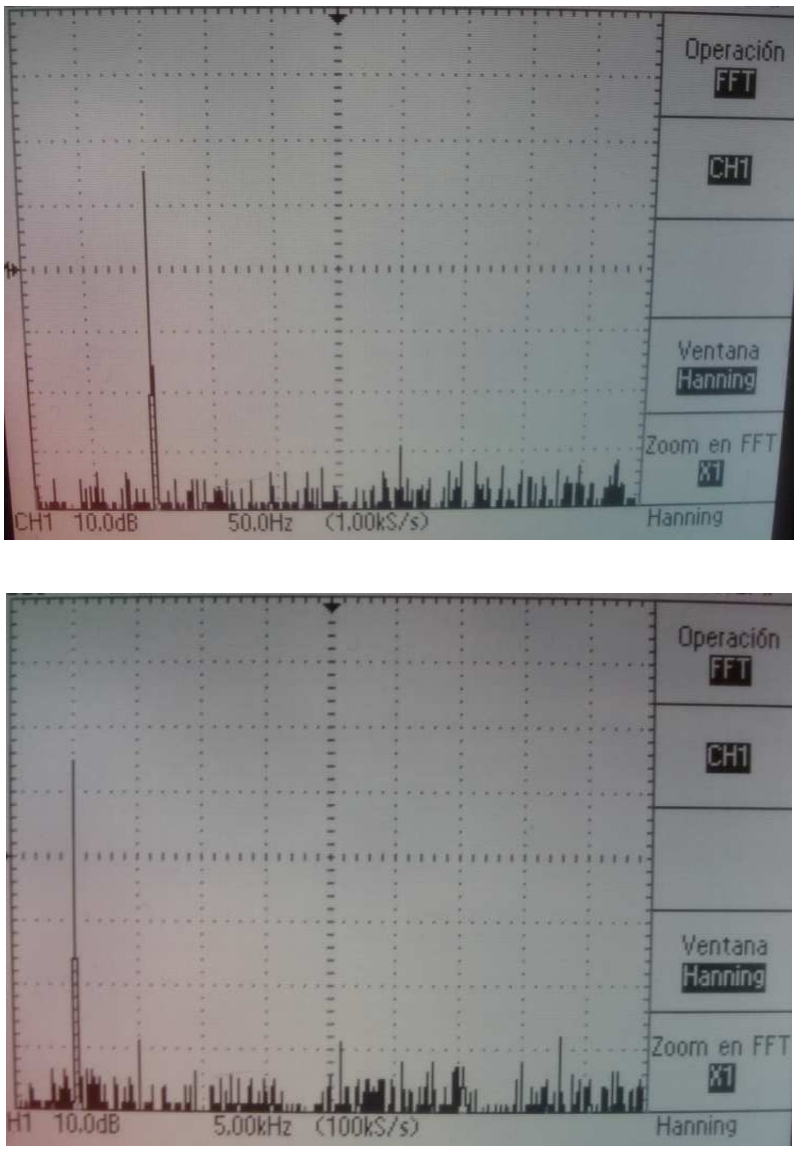
Fuente: Autores del Proyecto

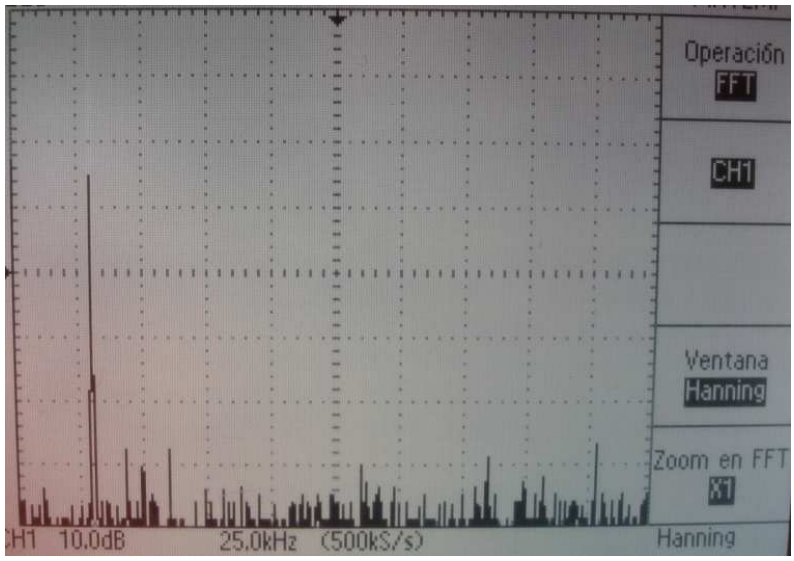
4.2.4 Espectro en frecuencia y distorsión armónica.

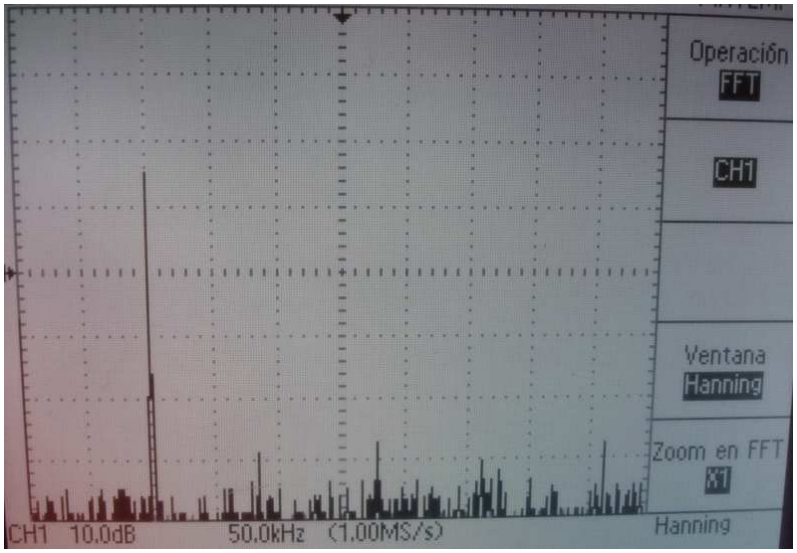
La última prueba que se realizó para comprobar el correcto funcionamiento del generador de señales fue el análisis del espectro en frecuencia y el cálculo de la distorsión armónica.

Gracias a la función math con que cuentan algunos osciloscopios del laboratorio de electrónica pudimos observar la FFT de algunas de las señales senoidales generadas por el equipo.

Figura 36 FFT de las señales senoidales 100 [Hz], 5 [kHz], 30 [kHz] y 100 [kHz]







Fuente: Autores del Proyecto

También se calculó la distorsión armónica de las señales observadas. En la tabla 7 se muestra la frecuencia de cada señal, los valores de los armónicos y el cálculo de la distorsión armónica.

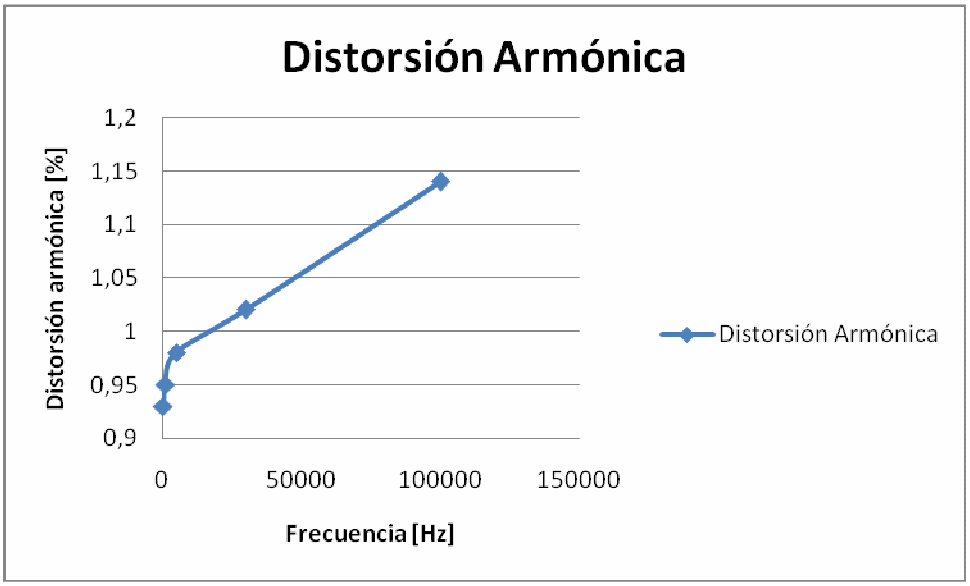
Tabla 7 Distorsión Armónica

FRECUENCIA DE LA SEÑAL SENOIDAL	ARMÓNICOS	DISTORSIÓN ARMÓNICA (THD)
100 [Hz]	a_0 = 16 [dB] = 4.46 [mVrms] a_1 = -34 [dB] = 0.0141 [mVrms] a_2 = -30 [dB] = 0.0223 [mVrms] a_3 = -34 [dB] = 0.0141 [mVrms] a_4 = -30 [dB] = 0.0223 [mVrms] a_5 = -32 [dB] = 0.0177 [mVrms]	THD = 0.93 %
5 [kHz]	a_0 = 16 [dB] = 4.46 [mVrms] a_1 = -31 [dB] = 0.0199 [mVrms] a_2 = -34 [dB] = 0.0141 [mVrms] a_3 = -34 [dB] = 0.0141 [mVrms] a_4 = -28 [dB] = 0.0281 [mVrms] a_5 = -32 [dB] = 0.0177 [mVrms]	THD = 0.98 %
30 [kHz]	a_0 = 16 [dB] = 4.46 [mVrms] a_1 = -28 [dB] = 0.0281 [mVrms] a_2 = -34 [dB] = 0.0141 [mVrms] a_3 = -34 [dB] = 0.0141 [mVrms] a_4 = -32 [dB] = 0.0177 [mVrms] a_5 = -34 [dB] = 0.0141 [mVrms]	THD = 1.02 %
100 [kHz]	a_0 = 16 [dB] = 4.46 [mVrms] a_1 = -28 [dB] = 0.0281 [mVrms] a_2 = -34 [dB] = 0.0141 [mVrms] a_3 = -32 [dB] = 0.0177 [mVrms] a_4 = -30 [dB] = 0.0223 [mVrms] a_5 = -28 [dB] = 0.0281 [mVrms]	THD = 1.14 %

Fuente: Autores del Proyecto

Como se observa en la tabla 7, la máxima distorsión armónica que presenta el generador es de 1,14% a una frecuencia de 100 [kHz]. Se pudo observar que a medida que se aumenta la frecuencia de las señales generadas la distorsión armónica aumenta en un pequeño porcentaje. El comportamiento de la distorsión armónica se observa en la figura 37.

Figura 37 Gráfica Distorsión Armónica vs. Frecuencia



Fuente: Autores del Proyecto

5. CONCLUSIONES Y RECOMENDACIONES

- Se implementó un generador de señales senoidales con una amplitud constante de 82 [mVpp], que genera señales de tensión en un rango de 0,1[Hz] a 100[kHz], como se especifica en la tabla 5. Además el equipo presenta una distorsión armónica máxima del 1,14% y una impedancia de salida cero, cumpliendo así, con los objetivos trazados al inicio del trabajo de grado.

- Se logró mejorar y optimizar la parte de generación de la señal senoidal del medidor de impedancia electroquímica realizado por los estudiantes Jorge Humberto Rodríguez y Sergio Andrés Ruiz. Esto es un logro importante ya que en dicho trabajo de grado se necesitaban de una gran cantidad de condensadores y resistencias para lograr el cambio de frecuencia en las señales generadas. Con este equipo, las frecuencias son cambiadas según el deseo del usuario por medio del teclado y todas están programadas por software.

- Se diseñaron e implementaron dos tarjetas electrónicas (Interfaz de comunicación y generador de señales senoidales) para el manejo de una pantalla alfanumérica LCD de 2x16, teclado 4x4 y para la obtención de las señales de tensión senoidal. Adicionalmente, se utilizó el DSP 56F8323 de Motorola, para la generación y control de las señales, así como para el manejo de la pantalla LCD y el teclado, empleando el módulo la interfaz serial, los pines de propósito general y los beans con los que cuenta el DSP.

- Durante el desarrollo del hardware del equipo, se presentaron problemas de distorsión y ruido producidos por conexiones en el protoboard y ruido introducido por algunas resistencias. Estos inconvenientes se superaron en gran medida cuando se implementó las dos tarjetas electrónicas. En cuanto al ruido introducido por las resistencias, concluimos que se deben utilizar resistencias de precisión o de muy bajo consumo de potencia, ya que con estas obtuvimos notables mejoras al visualizar la señal de salida del generador. Lo ideal es trabajar con elementos para montaje superficial, elementos con los cuales se obtienen mejores resultados.

- El generador de funciones AD9833 cumple eficazmente con los requerimientos del proyecto. La señal de salida del integrado tiene una amplitud fija de 632 [mV_{pp}], impedancia de salida baja (r_0 = 294,8Ω) y una distorsión armónica menor del 1%. Su control y funcionamiento es bastante sencillo y práctico. Se puede controlar totalmente por software y solo requiere una referencia de reloj y algunos condensadores de desacople.

- El equipo es la primera etapa de un Medidor de Impedancia Electroquímica que será implementado mas adelante. Maneja toda la parte de generación de la señal, dejando la puerta abierta para que futuros proyectos de grado implementen la parte de medición de impedancia. El DSP utilizado cuenta todavía con pines de propósito general y PWM para implementar las partes faltantes del medidor.

- Los resultados de las diferentes pruebas realizadas permiten asegurar invariabilidad de la amplitud con respecto a la frecuencia y que el equipo genera las frecuencias dentro del rango programado, independientemente

de la carga. Además, las señales generadas por el equipo cumplen con la norma ASTM para medición de impedancia electroquímica.

- Durante el desarrollo de este trabajo de grado se ha puesto en práctica y se ha profundizado gran parte de los conocimientos adquiridos en las diferentes materias cursadas a lo largo de la carrera tales como tratamiento de señales, procesamiento digital de señales y circuitos electrónicos. Además se adquirió experiencia en el diseño y desarrollo de proyectos y trabajos relacionados con nuestro futuro laboral como Ingenieros electrónicos.

Por último se plantean algunas observaciones y recomendaciones:

- Al momento de realizar la programación en CodeWarrior, es de gran ayuda utilizar el Procesador Experto con que cuenta el software. Hacer uso de los Bean, facilita la programación del DSP debido a que maneja los módulos y los periféricos de entrada y de salida por medio de propiedades, métodos y eventos, de tal forma que solo arroja un código que es incluido en el programa principal.

- Es importante tener en cuenta que la velocidad a la que trasmite el DSP es mucho mayor que la velocidad de recepción de la pantalla LCD y del Generador de Funciones AD9833. Por tal motivo, es necesario introducir retardos entre las instrucciones del algoritmo de cada dispositivo para eliminar el embotellamiento o acumulación que se presentan por dicha “lentitud”.

- Para obtener la salida deseada en el generador de señales senoidales de tensión se implementó un seguidor de tensión. Este seguidor de tensión requiere de una tensión de alimentación dual de 3,3 [V]. Como se mencionó en el capitulo 2, esta tensión dual se logró por medio de dos pilas de 3,3 [V], ya que no se consiguió un regulador de tensión con estas especificaciones. El proceso de importación de dicho integrado retrasaba considerablemente la culminación del proyecto. Por todo ésto, recomendamos que para la realización del macro-proyecto del Medidor de Impedancia Electroquímica, estas pilas sean reemplazadas por un regulador de tensión que invierta la polaridad de la tensión. Si no es factible esta posibilidad, dejar el montaje como está, pero estar pendiente del cambio de las baterías cuando sea necesario.

6. BIBLIOGRAFÍA

SEDRA, Adel y SMITH, Kenneth. Circuitos Microelectrónicos. 4 Ed. Oxford University Press, 1999.

Tektronix. Manual del Usuario de Osciloscopios del Tiempo Real Digital de la Serie TDS 200.

CUBIDES DIAZ, Astrid y MIELES PINTO, Fidel. Diseño de una Fuente de corriente senoidal y de pulsos bifásicos para medición en tejido humano. Trabajo de Grado. Universidad Industrial de Santander. 2006.

RODRIGUEZ PACHECO, Jorge y RUIZ, Sergio Andrés. Medidor de Impedancia Electroquímica. Trabajo de Grado. Universidad Industrial de Santander.

Impedance Instrumentation and Techniques. Aplication Note AC-3. Base de Datos de la IEEE. Biblioteca Universidad Industrial de Santander.

Norma CEI 61010-Parte 1. Artículo 6, 9, 10 y notas aclaratorias. Base de Datos de la IEEE. Biblioteca Universidad Industrial de Santander.

AMARIS DOMINGUEZ, Jean Pierre y LÓPEZ PATIÑO, Jose Alberto. Elaboración del Software para la caracterización de una celda electroquímica utilizando DSP familia 56800 Motorola. Trabajo de Grado. Universidad Industrial de Santander. 2004.

ANEXO A. MANUAL DEL USUARIO DEL GENERADOR DE SEÑALES PARA MEDICIÓN DE IMPEDANCIA ELECTROQUÍMICA.

MANUAL DEL USUARIO

MANUAL DEL USUARIO

ESPECIFICACIONES GENERALES

Tensión de Alimentación: 6 [V], 300 [mA]

Conectores de Salida: 1

Pulsadores: 1

Impedancia de Salida del Generador: 0 [Ω]

Amplitud de Señal del Generador: 82,0 [mV_{pp}]

Rango de Frecuencia del Generador: 0,1[Hz] – 100[kHz]

(20 valores de frecuencia por década)

Potencia Total Consumida: 576,4 [mW]

Distorsión Armónica: 1.14 %

Margen de carga: 10Ω-33kΩ

Rango de Temperatura: 20°C-100°C

MODO DE FUNCIONAMIENTO

1. Energice el dispositivo de acuerdo a las especificaciones de tensión de alimentación.

2. Después de encendido se observan en la pantalla LCD, el nombre del equipo, los logos de la Universidad Industrial de Santander y de la Escuela de Ingeniería Eléctrica, Electrónica y Telecomunicaciones, junto con los nombres de los autores del proyecto.
   NOTA: Si no se visualiza la presentación anteriormente descrita por favor, pulse el botón reset.

3. Posteriormente aparece la indicación: "Oprima A para seleccionar la frecuencia:"

4. Después de oprimir la tecla A, la pantalla LCD muestra la indicación: "Digite Frec.:". Es aquí, donde debe escoger el valor de frecuencia deseado, teniendo en cuenta que el generador produce 20 frecuencias por década. (Ejemplo: De 1[Hz] a 10[Hz], se puede escoger, 1[Hz], 1.5 [Hz], 2 [Hz], 2.5 [Hz], 3 [Hz], 3.5 [Hz], 4 [Hz], 4.5[Hz]……10 [Hz]). Además al introducir los números, estos se van corriendo hacia la izquierda hasta obtener la cifra deseada.

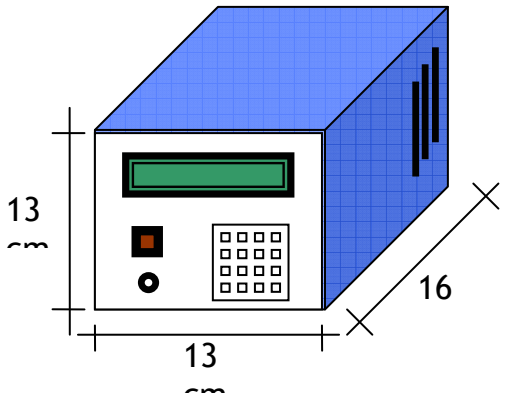
5. Para que el generador de señales produzca la señal deseada, después de seleccionar la frecuencia se debe oprimir la tecla # que es el ENTER del generador.

6. Finalmente, en los terminales del conector, se obtiene la señal deseada, que puede ser observada por un osciloscopio.

7. El equipo utiliza dos pilas de 3.3V. Se recomienda cambiar las pilas después de 4 meses de uso ya que en uso continuo tiene 2890 horas de vida útil.

NOTA:

MEDIDAS DEL EQUIPO



ANEXO B. PROGRAMACIÓN EN CODEWARRIOR PARA EL GENERADOR DE SEÑALES PARA MEDICIÓN DE IMPEDANCIA ELECTROQUÍMICA.

```
/* Including used modules for compilling procedure */
/* Including used modules for compilling procedure */
#include "Cpu.h"
#include "Events.h"
#include "Instruccion_Dato.h"
#include "Enable.h"
#include "Instruccion_Dato.h"
#include "Bit3.h"
#include "Bit4.h"
#include "Bit5.h"
```

```
#include "Bit6.h"
#include "EInt1.h"
#include "EInt2.h"
#include "EInt3.h"
#include "EInt4.h"
#include "InterfazSerialAD.h"
#include "FrecSincronAD.h"
#include "Datos.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"

const byte fuente1[]="  Generador de   ";
const byte fuente2[]="Seîales Senoidales ";
const byte uis[]="    * UIS *      ";
const byte uis1[]="    * E3T *      ";
const byte est[]="Adriana Martinez";
const byte est1[]=" Edgar Arenas   ";
const byte frase[]="Oprima  A  para            ";
const byte frase1[]="la frecuencia.     ";
const byte frec[]="Digite  Frec.:       ";
const byte frec12[]="  Frecuencia:       ";

byte puntos[16];
int i;
float LE1,LE2,LE3,LE4,LE5,LE6,LE7,LE8,LE9,LE10,MOS;
int cont,cont1,cont2,cont3,cont4;
float al,al1,al2,al3,a14,a15,a16,a17,a18,a19;
float FRECUENCIA;
int cont13,cont23,cont33,cont43;
int LU;
int ENTER=0;

extern int alarm;
extern int alarm2;
extern int alarm3;
extern int alarm5;
extern int alarm10;
extern int alarm11;
extern int alarm12;
extern int alarm13;
int LE;
float MOST;
int ZZ;

void SELEC_FREC_SENOIDAL(unsigned int LSB0, unsigned int MSB0, unsigned int LSB1,
unsigned int MSB1);
void FREQ1_REG(void);
void INICIOAD9833(void);
void MENU2 (const byte *frase1,const byte *frase2);
```

```
void MENU (const byte *frase1,const byte *frase2);
void INIC_LCD (void);
void COMANDO(byte p);
void ESCRIBIR1(const byte *c);
void ESCRIBIR(const byte *c);
void ESCRIBIR_DATO(byte Dato);
void ESCRIBIR_INSTRUCCION(byte Instruccion);
void ESPERA(void);
void RETARDO(word Retardo);
void RECEP(void);
void CONS(void);
void FRECDESEADA(void);

void main(void)
{
  byte inicio=0;

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

  INIC_LCD();                          // Inicializa la LCD

  MENU(fuente1,fuente2);               // Visualiza el título del proyecto
  ESPERA();
  MENU(uis,uis1);                      // Visualiza los logos de la universidad y de la
escuela
  ESPERA();
  MENU(est,est1);                      // Visualiza los autores
  ESPERA();
  MENU(frase,frase1);                  // Visualiza indicaciones para el usuario
  ESPERA();

   for(;;)
  {
   RECEP();                             // Algoritmo para la Selección de Frecuencia
  CONS();
  FRECDESEADA();
   }
}

  void RECEP(void)
  {
 Bit3_PutVal(1);
 Bit4_PutVal(0);
 Bit5_PutVal(0);
 Bit6_PutVal(0);
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 Bit3_PutVal(0);
 Bit4_PutVal(0);
 Bit5_PutVal(0);
```

```
Bit6_PutVal(0);
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}

if( alarm10==5 ){alarm10=0;alarm11=0;alarm12=0;alarm13=0;cont2=1;ZZ=1; cont=0;}
 if( alarm11==5 ){alarm10=0;alarm11=0;alarm12=0;alarm13=0;LE=1;ZZ=1;cont=cont+1;}
  if( alarm12==5 ){alarm10=0;alarm11=0;alarm12=0;alarm13=0;LE=2;ZZ=1;cont=cont+1;}
   if( alarm13==5 ){alarm10=0;alarm11=0;alarm12=0;alarm13=0;LE=3;ZZ=1;cont=cont+1;}

 for(i=0;i<5000;i++){}
////************************************************************
Bit3_PutVal(0);
Bit4_PutVal(1);
Bit5_PutVal(0);
Bit6_PutVal(0);
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
Bit3_PutVal(0);
Bit4_PutVal(0);
Bit5_PutVal(0);
Bit6_PutVal(0);
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}
for(i=0;i<5000;i++){}

if(alarm10==5){alarm10=0;alarm11=0;alarm12=0;alarm13=0;cont13=1;ZZ=1;cont4=cont4+1;}
 if( alarm11==5 ){alarm10=0;alarm11=0;alarm12=0;alarm13=0;LE=7;cont=cont+1;ZZ=1;}
  if( alarm12==5 ){alarm10=0;alarm11=0;alarm12=0;alarm13=0;LE=8;cont=cont+1;ZZ=1;}
   if( alarm13==5 ){alarm10=0;alarm11=0;alarm12=0;alarm13=0;LE=9;cont=cont+1;ZZ=1;}

  for(i=0;i<5000;i++){}
////************************************************************
Bit3_PutVal(0);
```

```
 Bit4_PutVal(0);
 Bit5_PutVal(1);
 Bit6_PutVal(0);
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 Bit3_PutVal(0);
 Bit4_PutVal(0);
 Bit5_PutVal(0);
 Bit6_PutVal(0);
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}

 if( alarm10==5 ){alarm10=0;alarm11=0;alarm12=0;alarm13=0;cont33=1;ZZ=1;cont4=cont4-
1;}
  if( alarm11==5 ){alarm10=0;alarm11=0;alarm12=0;alarm13=0;LE=52;cont=cont+1;ZZ=1;}
   if( alarm12==5 ){alarm10=0;alarm11=0;alarm12=0;alarm13=0;LE=0;cont=cont+1;ZZ=1;}
    if( alarm13==5 ){alarm10=0;alarm11=0;alarm12=0;alarm13=0;cont3=1;}

  for(i=0;i<5000;i++){}
////************************************************************
 Bit3_PutVal(0);
 Bit4_PutVal(0);
 Bit5_PutVal(0);
 Bit6_PutVal(1);
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 Bit3_PutVal(0);
 Bit4_PutVal(0);
 Bit5_PutVal(0);
 Bit6_PutVal(0);
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
```

```
 for(i=0;i<5000;i++){}
 for(i=0;i<5000;i++){}
  if( alarm10==5 ){alarm10=0;alarm11=0;alarm12=0;alarm13=0;ZZ=1;cont23=1;}
  if( alarm11==5 ){alarm10=0;alarm11=0;alarm12=0;alarm13=0;LE=4;cont=cont+1;ZZ=1;}
   if( alarm12==5 ){alarm10=0;alarm11=0;alarm12=0;alarm13=0;LE=5;cont=cont+1;ZZ=1;}
    if( alarm13==5 ){alarm10=0;alarm11=0;alarm12=0;alarm13=0;LE=6;cont=cont+1;ZZ=1;}

  for(i=0;i<5000;i++){}
////*****************************************//////////////////PARA LA FRECUENCIA/////////////////////////
 if( cont2==1 )
   {
   if( ZZ==1 )
   {
          if( cont==1 )
            {
              LE6=0;
              LE5=0;
              LE4=0;
              LE3=0;
              LE2=0;
              LE1=0;
              LE10=LE;
            }
          if( cont==2 )
            {
              LE6=0;
              LE5=0;
              LE4=0;
              LE3=0;
              LE2=0;
              LE1=LE10;
              LE10=LE;
            }
           if( cont==3 )
            {
              LE6=0;
              LE5=0;
              LE4=0;
              LE3=0;
              LE2=LE1;
              LE1=LE10;
              LE10=LE;
            }
           if( cont==4 )
            {
              LE6=0;
              LE5=0;
              LE4=0;
              LE3=LE2;
              LE2=LE1;
              LE1=LE10;
              LE10=LE;
            }
            if( cont==5 )
```

```
            {
                LE6=0;
              LE5=0;
              LE4=LE3;
              LE3=LE2;
              LE2=LE1;
              LE1=LE10;
              LE10=LE;
            }
            if( cont==6 )
            {
              LE6=0;
              LE5=LE4;
              LE4=LE3;
              LE3=LE2;
              LE2=LE1;
              LE1=LE10;
              LE10=LE;
            }
            if( cont==7 )
            {
                      LE6=LE5;
                      LE5=LE4;
                      LE4=LE3;
                      LE3=LE2;
                      LE2=LE1;
                      LE1=LE10;
                      LE10=LE;
             }
           if( cont==8 )
            {
              LE6=0;
              LE5=0;
              LE4=0;
              LE3=0;
              LE2=0;
              LE1=0;
              LE10=0;
              cont=0;
           }
  }
  }
///*********************************************************************
        if( cont3==1 )////////// ENTER
           {
           if( cont2==1 )////////// FRECUENCIA
           {
            cont=0;
            cont2=0;///////AMPLITUD
            cont3=0;////////ENTER
            MENU2(frec12, est);
            cont23=0;
      FRECUENCIA=LE6*100000+LE5*10000+LE4*1000+LE3*100+LE2*10+LE1+LE10*0.1;
            LE4=0;
```

```
            LE3=0;
            LE2=0;
            LE1=0;
            LE5=0;
            LE6=0;
            LE10=0;
            ENTER=1;
           }
           }
///**************************************************************************
  }
 ////////////////////////////////////**********************************************************************
void CONS(void)
 {
 ////////////////////////////////////////////////////
 if( cont2==1 )////////////FRECUNCIA
  {
  if( ZZ==1 )//////////////TODOS LOS NUMEROS
     {
   MENU2(frec, est);

   ZZ=0;

     }
  }
////////////////////////////////////////////////
 }
//////////////////////////////////////////**************************************************************************
void FRECDESEADA(void)                  //Cambio de frecuencia por el usuario
{
if(ENTER==1)
{
        if(FRECUENCIA==0.1)
        {
                FREQ1_REG();
                RETARDO(100);
                SELEC_FREC_SENOIDAL(0x4002,0x4000,0x8002,0x8000);
                RETARDO(100);
        }
        if(FRECUENCIA==0.2)
        {
                FREQ1_REG();
                RETARDO(100);
                SELEC_FREC_SENOIDAL(0x4005,0x4000,0x8005,0x8000);
                RETARDO(100);
        }
        if(FRECUENCIA==0.3)
        {
                FREQ1_REG();
                RETARDO(100);
                SELEC_FREC_SENOIDAL(0x4008,0x4000,0x8008,0x8000);
                RETARDO(100);
        }
        if(FRECUENCIA==0.4)
```

```
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x400B,0x4000,0x800B,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==0.5)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x400E,0x4000,0x800E,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==0.6)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4011,0x4000,0x8011,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==0.7)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4014,0x4000,0x8014,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==0.8)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4017,0x4000,0x8017,0x8000);
        RETARDO(100);
}

if(FRECUENCIA==0.9)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x401A,0x4000,0x801A,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==1)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x401D,0x4000,0x801D,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==1.5)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x402C,0x4000,0x802C,0x8000);
```

```
        RETARDO(100);
}
if(FRECUENCIA==2)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x403B,0x4000,0x803B,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==2.5)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x404A,0x4000,0x804A,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==3)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4059,0x4000,0x8059,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==3.5)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4068,0x4000,0x8068,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==4)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4077,0x4000,0x8077,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==4.5)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4086,0x4000,0x8086,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==5)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4095,0x4000,0x8095,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==5.5)
{
        FREQ1_REG();
```

```
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x40A4,0x4000,0x80A4,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==6)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x40B2,0x4000,0x80B2,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==6.5)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x40C1,0x4000,0x80C1,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==7)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x40D0,0x4000,0x80D0,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==7.5)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x40DF,0x4000,0x80DF,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==8)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x40EE,0x4000,0x80EE,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==8.5)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x40FD,0x4000,0x80FD,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==9)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x410C,0x4000,0x810C,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==9.5)
```

```
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x411B,0x4000,0x811B,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==10)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x412A,0x4000,0x812A,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==15)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x41BF,0x4000,0x81BF,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==20)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4254,0x4000,0x8254,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==25)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x42E9,0x4000,0x82E9,0x8000);
        RETARDO(100);
}

if(FRECUENCIA==30)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x437E,0x4000,0x837E,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==35)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4413,0x4000,0x8413,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==40)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x44A9,0x4000,0x84A9,0x8000);
```

```
        RETARDO(100);
}
if(FRECUENCIA==45)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x453E,0x4000,0x853E,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==50)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x45D3,0x4000,0x85D3,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==55)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4668,0x4000,0x8668,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==60)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x46FD,0x4000,0x86FD,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==65)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4792,0x4000,0x8792,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==70)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4827,0x4000,0x8827,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==75)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x48BC,0x4000,0x88BC,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==80)
{
        FREQ1_REG();
```

```
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4952,0x4000,0x8952,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==85)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x49E7,0x4000,0x89E7,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==90)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4A7C,0x4000,0x8A7C,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==95)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4B11,0x4000,0x8B11,0x8000);
        RETARDO(100);
}

if(FRECUENCIA==100)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4BA6,0x4000,0x8BA6,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==150)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x5179,0x4000,0x9179,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==200)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x574D,0x4000,0x974D,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==250)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x5D20,0x4000,0x9D20,0x8000);
        RETARDO(100);
}
```

```
if(FRECUENCIA==300)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x62F3,0x4000,0xA2F3,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==350)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x68C7,0x4000,0xA8C7,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==400)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x6E9A,0x4000,0xAE9A,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==450)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x746D,0x4000,0xB46D,0x8000);
        RETARDO(100);
}
if(FRECUENCIA==500)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x7A41,0x4000,0xBA41,0x8000);
        RETARDO(100);
}

if(FRECUENCIA==550)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4014,0x4001,0x8014,0x8001);
        RETARDO(100);
}
if(FRECUENCIA==600)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x45E7,0x4001,0x85E7,0x8001);
        RETARDO(100);
}
if(FRECUENCIA==650)
{
        FREQ1_REG();
        RETARDO(100);
```

```
        SELEC_FREC_SENOIDAL(0x4BBB,0x4001,0x8BBB,0x8001);
        RETARDO(100);
}
if(FRECUENCIA==700)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x518E,0x4001,0x918E,0x8001);
        RETARDO(100);
}
if(FRECUENCIA==750)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x5761,0x4001,0x9761,0x8001);
        RETARDO(100);
}
if(FRECUENCIA==800)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x5D34,0x4001,0x9D34,0x8001);
        RETARDO(100);
}
if(FRECUENCIA==850)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x6308,0x4001,0xA308,0x8001);
        RETARDO(100);
}
if(FRECUENCIA==900)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x68DB,0x4001,0xA8DB,0x8001);
        RETARDO(100);
}
if(FRECUENCIA==950)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x6EAE,0x4001,0xAEAE,0x8001);
        RETARDO(100);
}
if(FRECUENCIA==1000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x7482,0x4001,0xB482,0x8001);
        RETARDO(100);
}
if(FRECUENCIA==1500)
{
```

```
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x6EC3,0x4002,0xAEC3,0x8002);
        RETARDO(100);
}
if(FRECUENCIA==2000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x6904,0x4003,0xA904,0x8003);
        RETARDO(100);
}
if(FRECUENCIA==2500)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x6345,0x4004,0xA345,0x8004);
        RETARDO(100);
}
if(FRECUENCIA==3000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x5D86,0x4005,0x9D86,0x8005);
        RETARDO(100);
}
if(FRECUENCIA==3500)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x57C7,0x4006,0x97C7,0x8006);
        RETARDO(100);
}
if(FRECUENCIA==4000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x5208,0x4007,0x9208,0x8007);
        RETARDO(100);
}
if(FRECUENCIA==4500)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4C49,0x4008,0x8C49,0x8008);
        RETARDO(100);
}
if(FRECUENCIA==5000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x468A,0x4009,0x868A,0x8009);
        RETARDO(100);
}
```

```
if(FRECUENCIA==5500)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x40CB,0x400A,0x80CB,0x800A);
        RETARDO(100);
}
if(FRECUENCIA==6000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x7B0C,0x400A,0xBB0C,0x800A);
        RETARDO(100);
}
if(FRECUENCIA==6500)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x754E,0x400B,0xB54E,0x800B);
        RETARDO(100);
}
if(FRECUENCIA==7000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x6F8F,0x400C,0xAF8F,0x800C);
        RETARDO(100);
}
if(FRECUENCIA==7500)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x69D0,0x400D,0xA9D0,0x800D);
        RETARDO(100);
}
if(FRECUENCIA==8000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x6411,0x400E,0xA411,0x800E);
        RETARDO(100);
}
if(FRECUENCIA==8500)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x5E52,0x400F,0x9E52,0x800F);
        RETARDO(100);
}
if(FRECUENCIA==9000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x5893,0x4010,0x9893,0x8010);
```

```
        RETARDO(100);
}
if(FRECUENCIA==9500)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x52D4,0x4011,0x92D4,0x8011);
        RETARDO(100);
}
if(FRECUENCIA==10000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4D15,0x4012,0x8D15,0x8012);
        RETARDO(100);
}
if(FRECUENCIA==15000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x53A0,0x401B,0x93A0,0x801B);
        RETARDO(100);
}
if(FRECUENCIA==20000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x5A2B,0x4024,0x9A2B,0x8024);
        RETARDO(100);
}
if(FRECUENCIA==25000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x60B6,0x402D,0xA0B6,0x802D);
        RETARDO(100);
}
if(FRECUENCIA==30000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x6740,0x4036,0xA740,0x8036);
        RETARDO(100);
}
if(FRECUENCIA==35000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x6DCB,0x403F,0xADCB,0x803F);
        RETARDO(100);
}
if(FRECUENCIA==40000)
{
        FREQ1_REG();
```

```
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x7456,0x4048,0xB456,0x8048);
        RETARDO(100);
}
if(FRECUENCIA==45000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x7AE1,0x4051,0xBAE1,0x8051);
        RETARDO(100);
}
if(FRECUENCIA==50000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x416C,0x405B,0x816C,0x805B);
        RETARDO(100);
}
if(FRECUENCIA==55000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x47F6,0x4064,0x87F6,0x80F6);
        RETARDO(100);
}
if(FRECUENCIA==60000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x4E81,0x406D,0x8E81,0x806D);
        RETARDO(100);
}
if(FRECUENCIA==65000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x550C,0x4076,0x950C,0x8076);
        RETARDO(100);
}
if(FRECUENCIA==70000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x5B97,0x407F,0x9B97,0x807F);
        RETARDO(100);
}
if(FRECUENCIA==75000)
{
        FREQ1_REG();
        RETARDO(100);
        SELEC_FREC_SENOIDAL(0x6222,0x4088,0xA222,0x8088);
        RETARDO(100);
}
if(FRECUENCIA==80000)
```

```
        {
                FREQ1_REG();
                RETARDO(100);
                SELEC_FREC_SENOIDAL(0x68AC,0x4091,0xA8AC,0x8091);
                RETARDO(100);
        }
        if(FRECUENCIA==85000)
        {
                FREQ1_REG();
                RETARDO(100);
                SELEC_FREC_SENOIDAL(0x6F37,0x409A,0xAF37,0x809A);
                RETARDO(100);
        }
        if(FRECUENCIA==90000)
        {
                FREQ1_REG();
                RETARDO(100);
                SELEC_FREC_SENOIDAL(0x75C2,0x40A3,0xB5C2,0x80A1);
                RETARDO(100);
        }
        if(FRECUENCIA==95000)
        {
                FREQ1_REG();
                RETARDO(100);
                SELEC_FREC_SENOIDAL(0x7C4D,0x40AC,0xBC4D,0x80AC);
                RETARDO(100);
        }
        if(FRECUENCIA==100000)
        {
                FREQ1_REG();
                RETARDO(100);
                SELEC_FREC_SENOIDAL(0x42D8,0x40B6,0x82D8,0x80B6);
                RETARDO(100);
        }
 }
ENTER=0;
}
void SELEC_FREC_SENOIDAL(unsigned int LSB0, unsigned int MSB0, unsigned int LSB1,
unsigned int MSB1)
{
        RETARDO(100);
        FrecSincronAD_PutVal(0);            // Fija en 0 el bit Frec. para habilitar los datos
        RETARDO(100);
        InterfazSerialAD_SendChar(LSB0); // Escribe 14LSB de Frec. deseada en FREQ0
        RETARDO(100);
        InterfazSerialAD_SendChar(MSB0); // Escribe 14MSB de Frec. deseada en FREQ0
        RETARDO(100);
        InterfazSerialAD_SendChar(LSB1); // Escribe 14LSB de Frec. deseada en FREQ1
        RETARDO(100);
        InterfazSerialAD_SendChar(MSB1); // Escribe 14MSB de Frec.deseada en FREQ1
        RETARDO(100);
        InterfazSerialAD_SendChar(0x2000);// Bits de Control: B28=1, FSELECT=0,
PSELECT=0, los demás bits en cero
        RETARDO(100);
```

```
        FrecSincronAD_PutVal(1);            // Fija en uno el bit Frec para deshabilitar datos
        RETARDO(100);
}
void FREQ1_REG(void)
{
        RETARDO(100);
        FrecSincronAD_PutVal(0);                // Fija en cero el bit Frec para habilitar datos
        RETARDO(100);
        InterfazSerialAD_SendChar(0x2C00);  // Bits de Control: B28=1, FSELECT=1,
PSELECT=1, los demás bits en cero
        RETARDO(100);
        FrecSincronAD_PutVal(1);                // Fija en 1 el bit Frec para deshabilitar datos
        RETARDO(100);
}
void INICIOAD9833(void)                 //Inicialización del Generador de Funciones AD9833
{
        RETARDO(100);
        FrecSincronAD_PutVal(0);                //Fija en 0 el bit FSYNC para habilitar  datos
        RETARDO(100);
        InterfazSerialAD_Enable();              //Habilita el módulo SPI
        RETARDO(100);
        InterfazSerialAD_SendChar(0x2100);  //B28=1,RESET=1,FSELECT=0,PSELECT=0
        RETARDO(100);
        InterfazSerialAD_SendChar(0x4003);  //Coloca los 14LSB de 0.1Hz en FREQ0
        RETARDO(100);
        InterfazSerialAD_SendChar(0x4000);  //Coloca los 14MSB de 0.1Hz en FREQ0
        RETARDO(100);
        InterfazSerialAD_SendChar(0xB333);  //Coloca los 14LSB de 100kHz en FREQ1
        RETARDO(100);
        InterfazSerialAD_SendChar(0x80CC);  //Coloca los 14MSB de 100kHz en FREQ1
        RETARDO(100);
        InterfazSerialAD_SendChar(0xC000);  //Coloca FASE 0 en PHASE0 REG
        RETARDO(100);
        InterfazSerialAD_SendChar(0xE000);  //Coloca FASE 0 en PHASE1 REG
        RETARDO(100);
        InterfazSerialAD_SendChar(0x2000);  //Coloca B28=1,FSELECT=0,PSELECT=0
        RETARDO(100);
        FrecSincronAD_PutVal(1);                //Fija en 1 el bit Frec para desabilitar datos
        RETARDO(100);
}
void MENU2 (const byte *frase1,const byte *frase2)
{
        COMANDO(1);                     // Limpia la LCD
        ESCRIBIR(frase1);               // Escribe la frase 1 en la primera línea
        COMANDO(192);                   // Posiciona el cursor en la segunda línea 192
        ESCRIBIR1(frase2);              // Escribe la frase 2 en la segunda línea
}
void MENU (const byte *frase1,const byte *frase2)
{
        COMANDO(1);             // Limpia la LCD
        ESCRIBIR(frase1);       // Escribe la frase 1 en la primera línea
        COMANDO(192);           // Posiciona el cursor en la segunda línea
   ESCRIBIR(frase2);    // Escribe la frase 2 en la segunda línea
}
```

```
void INIC_LCD (void)
{
        RETARDO(7000);
   ESCRIBIR_INSTRUCCION(3);         // Operación con 8 bits
        RETARDO(3000);
        ESCRIBIR_INSTRUCCION(3); // Operación con 8 bits
        RETARDO(3000);
        ESCRIBIR_INSTRUCCION(3); // Operación con 8 bits
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(2); // Operación con 4 bits
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(2); // Operación con 4 bits
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(8); // Se fija N=1 (2-Líneas) y F=0 (5X7 dot)
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(0); // Display Apagado
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(8); // Display Apagado
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(0); // Limpiar Display
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(1); // Limpiar Display
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(0); // Fija el Modo (Incremento/Decremento)
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(6); // I/D=1 Se fija en modo Incremento
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(0); // Display encendido
        RETARDO(2000);
        ESCRIBIR_INSTRUCCION(15); // C=1 (Cursor Encend) B=1 (Parpadeo Encend)
        RETARDO(2000);
}
void COMANDO(byte p)
{byte MSB,LSB;
        RETARDO(75);
        MSB=0xF0&p;
        LSB=0x0F&p;
        MSB=MSB>>4;
        ESCRIBIR_INSTRUCCION(MSB);
        RETARDO(75);
        ESCRIBIR_INSTRUCCION(LSB);
        RETARDO(75);
}
void ESCRIBIR1(const byte *c)
{byte MSB,LSB; int i;
 ////////////////////////////////////////////
 LE7=12;
 LE8=12;
  al=LE1+47;     /////se le suman 48 de un solo digito
 al1=LE2+47;    /////se le suman 48 de un solo digito
 al2=LE3+47;    /////se le suman 48 de un solo digito
 al3=LE4+47;    /////se le suman 48 de un solo digito
 a14=LE5+47;   /////se le suman 48 de un solo digito
 a15=LE6+47;   /////se le suman 48 de un solo digito
```

```
 a16=LE7+59;
 a17=LE8+109;
 a18=LE9+47;
 a19=LE10+47;

 for(i=0;i<16;i++)          { puntos[i]=32;}////////////////////
 puntos[4]=1+a15;
 puntos[5]=1+a14;
 puntos[6]=1+al3;
 puntos[7]=1+al2;
 puntos[8]=1+al1;
 puntos[9]=1+al;
 puntos[10]=1+a18-2;
 puntos[11]=1+a19;
 puntos[13]=1+a16;
 puntos[14]=1+a17;
////////////////////////////////////////////////
          for(i=0;i<16;i++)
          {
                    RETARDO(75);
                    MSB=0xF0&(puntos[i]);
                    LSB=0x0F&(puntos[i]);
             MSB=MSB>>4;
                    ESCRIBIR_DATO(MSB);
                    RETARDO(75);
                    ESCRIBIR_DATO(LSB);
                    RETARDO(75);
          }
}
void RETARDO(word Retardo)
{ word a,b;
  Retardo=Retardo*2;
  for(a=0;a<=Retardo;a++)
  {
          for(b=0;b<=2;b++)
          {
                    asm
             {
                              nop;
                              nop;
                              nop;
                              nop;
                              nop;
             }
          }
  }
}
void ESPERA(void)      // TIEMPO ESPERA PARA VISUALIZAR PRESENTACIÓN
{int i;
          for(i=0;i<20;i++)
          {
                    RETARDO(50000);
          }
}
```

```
void ESCRIBIR_INSTRUCCION(byte Instruccion)
{
   Instruccion_Dato_ClrVal(); // Para enviar una instrucción
   RETARDO(75);
   Datos_PutVal(Instruccion);
   RETARDO(75);
   Enable_SetVal();
   RETARDO(75);
   Enable_ClrVal();
   RETARDO(75);
}
void ESCRIBIR_DATO(byte Dato)
{
   Instruccion_Dato_SetVal(); // Para enviar un dato
   RETARDO(75);
   Datos_PutVal(Dato);
   RETARDO(75);
   Enable_SetVal();
   RETARDO(75);
   Enable_ClrVal();
   RETARDO(75);
}
void ESCRIBIR(const byte *c)
{byte MSB,LSB; int i;
        for(i=0;i<16;i++)
        {
                RETARDO(75);
                MSB=0xF0&(c[i]);
                LSB=0x0F&(c[i]);
                 MSB=MSB>>4;
                ESCRIBIR_DATO(MSB);
                RETARDO(75);
                ESCRIBIR_DATO(LSB);
                RETARDO(75);
        }
}
```