UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERIA

TRABAJO DE GRADUACIÓN

"DISEÑO Y CONSTRUCCION DE UN OSCILOSCOPIO Y ANALIZADOR ESPECTRAL DIGITAL"

PRESENTAN:

MARIO DE JESUS MARTINEZ SANCHEZ

CAMILO DE JESUS MERINO MELGAR

SOYAPANGO MARZO 2004

UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERIA

ESCUELA DE INGENIERIA ELECTRONICA

RECTOR:

ING. FEDERICO MIGULE HUGUET RIVERA

SECRETARIO GENERAL:

LIC. MARIO OLMOS

DECANO DE LA FACULTAD DE INGENIERIA:

ING. GODOFREDO GIRON

TUTOR:

ING. WENCESLAO RIVAS

ASESOR

ING. FRANCISCO TAURA

JURADOS:

ING. EDGARDO ZELEDÓN

ING MILTON ZAMORA

ING ERIK BLANCO

DEDICATORIA

“DEDICAMOS ESTE TRABAJO A NUESTRA FAMILIA, POR TODO EL APOYO BRINDADO DURANTE ESTOS AÑOS DE ESUDIO Y EN ESPECIAL A DIOS TODO PODEROSO”

“DISEÑO Y CONSTRUCCION DE UN OSCILOSCOPIO Y ANALIZADOR ESPECTRAL DIGITAL”

TUTOR

Ing. Wenceslao Rivas

ASESOR

Ing. Francisco Taura

JURADOS

Ing. Edgardo Zeledón

Ing. Miltón Zamora

Ing. Erik Blanco

INDICE

INTRODUCCION

Uno de los instrumentos más útiles y versátiles a la hora de desarrollar un circuito electrónico es sin dudas el Osciloscopio. Estos dispositivos han sido del dominio de grandes compañías como HP o Tektronix que los ofrecen a costos muy elevados. Costos elevados implica que dichos instrumentos usualmente quedan fuera del alcance de la mayoría de personas que realmente podrían utilizarlos.

Además de esto, la filosofía de diseño de estas compañías tiende a favorecer la construcción de dispositivos portátiles; es decir, dispositivos que tienen su propia computadora, un sistema operativo, pantalla, unidades de almacenamiento de datos, fuente de alimentación y por último el circuito de prueba mismo. Esto resulta adecuado para grandes empresas o instituciones que pueden adquirir equipos sofisticados, por que la naturaleza de sus trabajos se los exige y su poder adquisitivo se los permite.

Para el publico en general esto sería desperdiciar los recursos con los que ya se cuentan, especialmente si observamos las computadoras disponibles hoy en día. Estas computadoras cuentan con procesadores muy veloces, con monitores de muy buena resolución y discos duros con capacidades que fácilmente superan algún dispositivo portátil.

En el área de telecomunicaciones es necesario también estudiar señales en el dominio de la frecuencia, para lo cual se emplea un dispositivo conocido como "Analizador Espectral"y que también es de costos elevados. A pesar de ser muy útiles en el análisis de señales en aplicaciones de telecomunicación, la mayoría de instituciones en el país posee muy pocos de estos dispositivos.

Por las razones expresadas anteriormente, se quiere mediante este proyecto de tesis diseñar y construir un osciloscopio digital y analizador espectral que sea económico, para que este al alcance de estudiantes de carreras técnicas. El diseño se basa en la elaboración de una tarjeta que pueda conectarse a través del puerto serial a una computadora huésped que procese los datos y los presente en pantalla.

Este documento contiene la información del diseño de una tarjeta adquisidora de datos
A continuación presentamos una visión general del contenido de este documento.

En el capítulo I se presenta la teoría general del osciloscopio, se establecen las diferencias entre un osciloscopio análogo y un osciloscopio digital y se presenta la teoría del análisis espectral.

En el capítulo II se presenta un panorama general de la familia microcontroladores PIC. Se discute el funcionamiento y organización de la memoria del microcontrolador PIC16f84 así como teoría de los programadores lógicos de alta densidad (PLD). Se incluye la descripción del diseño del hardware y software de la tarjeta adquisidora de datos así como la forma en que obtiene los datos y la comunicación con la computadora huésped.

El capítulo III explora los distintos leguajes de programación, lenguajes de alto nivel y el lenguaje de programación Visual Basic. Se discute la programación orientada a objetos, que será la base para el desarrollo del software de la computadora huésped. Se

explica la forma de manejar los puertos a través de Visual Basic y en especial el componente de software llamado MSComm encargado de la comunicación serial.

El capítulo IV cubre el software del programa que se alojará en la computadora. El capitulo esta dividido en dos partes. La primera presenta la interfaz grafica con sus distintos componentes y su tabla de propiedades. En la segunda parte se presenta el código para manejar la interfaz gráfica y sus distintas clases. El funcionamiento del código es presentado mediante explicaciones escritas de distintos eventos y a través de flujogramas.

El capítulo V describe el algoritmo empleado para obtener la Transformada Rápida de Fourier.

El capitulo VI examina la técnica de sub-muestreo, sus efectos en el dominio del tiempo y la frecuencia.

En capitulo VII se presenta una lista de todos los elementos empleados en la construcción de la tarjeta adquisidora de datos y su correspondiente costo.

Finalmente, junto con este documento se incluye un CD que contiene una copia de este documento, hojas técnicas de los integrados utilizados y un diagrama completo del circuito construido.

DEFINICIÓN DEL TEMA.

Construir un osciloscopio y analizador espectral digital mediante una tarjeta adquisidora de datos que muestree señales variantes en el tiempo, y que puede ser adaptada a todo tipo de PC bajo ambiente Windows a través del puerto serial. La adquisición de los datos será auto-rango tanto en magnitud como en frecuencia. Estos datos serán procesados dentro de la PC por un lenguaje de alto nivel y presentados gráficamente en la pantalla con la opción de poder representar la señal tanto en el dominio del tiempo como en frecuencia.

OBJETIVOS

GENERAL

Diseñar y construir un sistema de adquisición de datos para muestrear señales variantes en el tiempo, que con ayuda de una PC facilite a los estudiantes el análisis de señales tanto en dominio del tiempo como de la frecuencia, y que reduzca los costos con respecto a los equipos de medición tradicionales.

ESPECIFICOS*:*

Construir un sistema:

- Capaz de determinar las componentes de frecuencia que conforman una señal.
- Que obtenga de forma automática (auto-rango) la frecuencia y amplitud de una señal que varia con el tiempo.
- Capaz de visualizar cualquier señal tanto en el dominio del tiempo como el dominio de la frecuencia.
- Cuyos costos sean considerablemente menores frente a los existentes en el mercado.
- Que se adapte fácilmente a cualquier tipo de PC bajo ambiente Windows (Windows 95 en adelante).
- Cuya interfaz grafica sea de fácil manejo para el usuario.

CAPITULO I: TEORIA DEL OSCILOSCOPIO Y ANALIZADOR DE ESPECTRO.

El osciloscopio es básicamente un dispositivo de visualización gráfica que muestra señales eléctricas variables en el tiempo. El eje vertical, denominado Y, representa el voltaje; mientras que el eje horizontal, denominado X, representa el tiempo.

Un osciloscopio resulta útil para:

- Determinar el periodo y el voltaje de una señal.
- Determinar la frecuencia de una señal.
- Determinar que parte de la señal es DC y cual AC.
- Localizar averías en un circuito.
- Medir la fase entre dos señales.
- Determinar que parte de la señal es ruido y como varia este en el tiempo.

Los osciloscopios se dividen en dos grandes grupos:

- Análogos
- Y Digitales

1.1 Osciloscopios Análogos

Estos trabajan directamente con la señal aplicada, la cual una vez amplificada desvía un haz de electrones en sentido vertical y proporcional a su valor a través de un tubo de rayos catódicos (CRT). Los electrones se estrellan con una *pantalla fluorescente* que está cubierta de fósforo. Por medio de circuito se manipula el haz de electrones para que forme una replica de la señal de entrada.

1.2 Osciloscopios Digitales.

Los osciloscopios digitales utilizan un sistema adicional de proceso de datos que permite almacenar y visualizar la señal. Por ejemplo, puede ser una unidad microprocesador, un microcontrolador, etc. como unidades de procesamiento. Un LCD o un monitor para la visualización de la señal. Utiliza un convertidor analógico digital que muestrea la señal a intervalos de tiempo determinado y convierte una señal continua a valores discretos.
Los valores digitales muestreados se almacenan en una memoria como puntos de señal. El número de los puntos de señal utilizados para reconstruir la señal en pantalla se denomina registro. La sección de visualización recibe estos puntos del registro, una vez almacenados en la memoria, para presentar en pantalla la señal.

1.3 El espectro De Una Señal.

Teóricamente, una onda cuadrada tiene un espectro de frecuencia que contiene la frecuencia fundamental y todos sus armónicos impares hasta el infinito.
La magnitud de los armónicos disminuye a medida que aumenta el orden de los mismos. Por ejemplo, la magnitud del tercer armónico es más grande que la del quinto, y así sucesivamente.
En la práctica, el número de armónicos que se considera que tienen una amplitud significativa está limitada alrededor de nueve, sin embargo, para transportar una onda cuadrada con flancos realmente agudos se necesita un ancho de banda muy grande.

1.4 Introducción Al Análisis Espectral.

El análisis de señales eléctricas es un problema fundamental para muchos ingenieros y científicos. Aun cuando el problema inmediato no es eléctrico, se cambian a menudo los parámetros básicos de interés en las señales eléctricas por medio de los transductores.
Las ventajas de transformar los parámetros físicos de las señales eléctricas son notables, una gran cantidad de instrumento disponible para el análisis de señales eléctricas en el dominio del tiempo y la frecuencia.
La manera tradicional de observar las señales eléctricas es verlas en el dominio de tiempo usando un osciloscopio. El dominio del tiempo se usa para recuperar el tiempo relativo de fase, información que es necesaria para caracterizar la conducta del circuito eléctrico. Sin embargo, no todos los circuitos pueden caracterizarse singularmente sólo con la información del dominio del tiempo. Elementos de los circuitos como amplificadores, los osciladores, mezcladores, modulador-demoduladores, y filtros son caracterizados mejor por la información que reposa en la frecuencia. Esta información de frecuencia es obtenida viendo las señales eléctricas en el dominio de frecuencia. Para desplegar el dominio de la frecuencia se requiere un dispositivo que pueda diferenciar entre frecuencias midiendo los niveles de energía de cada una. Para mostrar gráficamente voltajes o energía como una función de la frecuencia se utiliza un CRT (tubo de rayos catódicos)
En el dominio del tiempo, se ven todos los componentes de la frecuencia de una señal sumados y juntos. En el dominio de la frecuencia, las señales complejas son separadas en sus componentes de frecuencia, y el nivel de energía de cada frecuencia es desplegado. El dominio de frecuencia contiene información no encontrada en el dominio de tiempo y por consiguiente, el analizador del espectro tiene ciertas ventajas comparadas con un osciloscopio.
El analizador alcanza el mayor nivel de sensibilidad a un bajo nivel de distorsión. Las ondas del seno pueden parecer buenas en el dominio de tiempo, pero en el dominio de frecuencia, puede verse distorsión armónica. La sensibilidad y el ancho de rango dinámico del analizador del espectro es útil para medir modulaciones de bajo nivel. Puede usarse para medir FM y pulso RF.

1.5 Tipos De Analizadores Del Espectro.

Hay dos tipos básicos de Analizadores del Espectro, punto-barrido y Analizadores de tiempo-real. Los Analizadores punto-barrido son puestos a punto barriendo eléctricamente encima de su rango de frecuencia. Por consiguiente, se prueban secuencialmente a tiempo los componentes de frecuencia de un espectro. Esto habilita la señal periódica para ser desplegada al azar. Los analizadores de tiempo-real, por otro lado, despliegan simultáneamente la amplitud de todas las señales. Esto conserva la dependencia de tiempo entre las señales que la información de fase le permita ser desplegada. Los analizadores del tiempo-real son capaces de desplegar señales periódicas y señales al azar.

CAPITULO II: TARJETA ADQUISIDORA DE DATOS.

2.1 LA FAMILIA DE MICROCONTROLADORES PIC.

Un microcontrolador o MCU es básicamente un microprocesador al que se le ha integrado los bloques de memoria de datos, programa, y los periféricos de entrada/salida, esto en cuanto al hardware, y como principal característica del software es que posee instrucciones orientadas al bit. Esto quiere decir que posee instrucciones en las cuales los operandos fuente y/o destino es una variable de simplemente un bit de longitud, a parte de contar con las clásicas instrucciones byte a byte.

En la estructura del hardware del microcontrolador se puede encontrar los siguientes sistemas (Pero no siempre todos) dependiendo del fabricante y familia microcontrolador.

CPU
Memoria de datos volátil: RAM
Memoria de datos no volátil: EEPROM
Memoria de programa: PROM, EPROM, EEPROM, FLASH
Entradas/Salidas discretas.
Contadores y temporizadores
Conversores A/D y D/A
Comparadores analógicos.
Gestión de interrupciones
Gestión de consumo de energía
Registros especiales
Reloj de tiempo real
Unidad de comunicaciones seriales: I2C, UART configurables.
Interfase para la Programación en Serie: SPI

2.1.1 Breve Reseña Histórica.

En 1965, la empresa GI creó una división de microelectrónica, GI Microelectronics División, que comenzó fabricando memorias EPROM y EEPROM, que conformaban las familias AY3-XXXX y AY5-XXXX. A principios de los años 70 diseñó el microprocesador de 16 bits CP1600, razonablemente bueno pero que no manejaba eficazmente las Entradas y Salidas. Para solventar este problema, en 1975 diseñó un chip destinado a controlar E/S: el PIC (Peripheral Interfase Controller). Se trataba de un controlador rápido pero limitado y con pocas instrucciones pues iba a trabajar en combinación con el CP1600.

La arquitectura del PIC, que se comercializó en 1975, era sustancialmente la misma que la de los actuales modelos PIC16C5X. En aquel momento se fabricaba con tecnología NMOS y el producto sólo se ofrecía con memoria ROM y con un pequeño pero robusto microcódigo.

La década de los 80 no fue buena para GI, que tuvo que reestructurar sus negocios, concentrando sus actividades en los semiconductores de potencia. La GI

Microelectronics División se convirtió en una empresa subsidiaria, llamada GI Microelectronics Inc. Finalmente, en 1985, la empresa fue vendida a un grupo de inversores de capital de riesgo, los cuales, tras analizar la situación, rebautizaron a la empresa con el nombre de Arizona Microchip Technology y orientaron su negocio a los PIC, las memorias EPROM paralelo y las EEPROM serie. Se comenzó rediseñando los PIC, que pasaron a fabricarse con tecnología CMOS, surgiendo la familia de gama baja PIC16CSX, considerada como la "clásica".

Una de las razones del éxito de los PIC se basa en su utilización. Cuando se aprende a manejar uno de ellos, conociendo su arquitectura y su repertorio de instrucciones, es muy fácil emplear otro modelo.

Microchip cuenta con su factoría principal en Chandler, Arizona, en donde se fabrican y prueban los chips con los más avanzados recursos técnicos. En 1993 construyó otra factoría de similares características en Tempe, Arizona. También cuenta con centros de ensamblaje y ensayos en Taiwan y Tailandia. Para tener una idea de su alta producción, hay que tener en cuenta que ha superado el millón de unidades por semana en productos CMOS de la familia PIC16CSX.

2.1.2 Características Relevantes.

Se comienza describiendo las características más representativas de los PIC.

1ª. La arquitectura del procesador sigue el modelo Harvard

En esta arquitectura, el CPU se conecta de forma independiente y con buses distintos con la memoria de instrucciones y con la de datos. Figura 1-1.

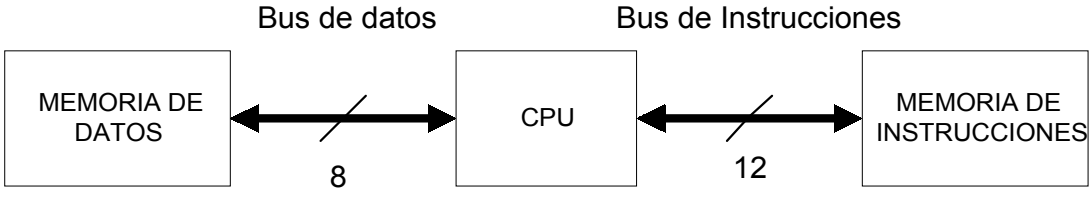


Figura 2 1 Modelo Harvard

La arquitectura Harvard permite al CPU acceder simultáneamente a las dos memorias. Además, propicia numerosas ventajas al funcionamiento del sistema como se irán describiendo.

2ª. Se aplica la técnica de segmentación (“pipe-line”) en la ejecución de las instrucciones.

La segmentación permite al procesador realizar al mismo tiempo la ejecución de una instrucción y la búsqueda del código de la siguiente. De esta forma se puede ejecutar cada instrucción en un ciclo (un ciclo de instrucción equivale a cuatro ciclos de reloj). Figura 3.2.

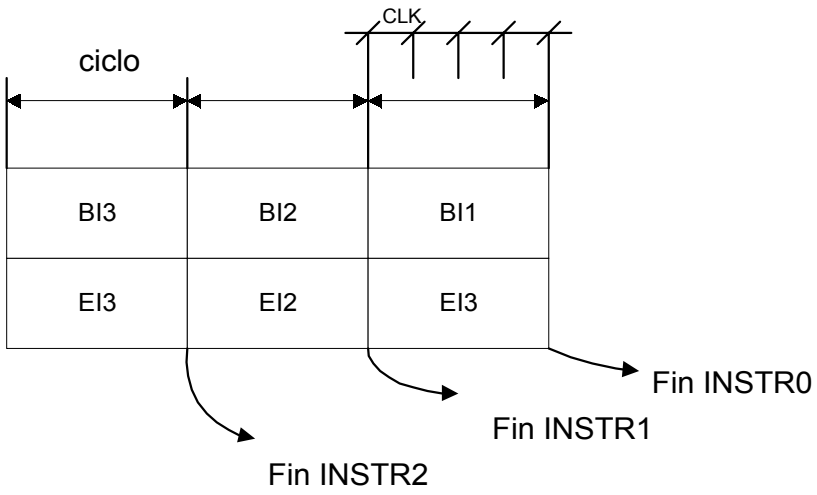


Figura 2 2

La segmentación permite al procesador ejecutar cada instrucción en un ciclo de instrucción equivalente a cuatro ciclos de reloj. En cada ciclo se realiza la búsqueda de una instrucción y la ejecución de la anterior.

Las instrucciones de salto ocupan dos ciclos al no conocer la dirección de la siguiente instrucción hasta que no se haya completado la de bifurcación.

3ª. El formato de todas las instrucciones tiene la misma longitud

Todas las instrucciones de los microcontroladores de la gama baja tienen una longitud de 12 bits. Las de la gama media tienen 14 bits y más las de la gama alta. Esta característica es muy ventajosa en la optimización de la memoria de instrucciones y facilita enormemente la construcción de ensambladores y compiladores.

4ª. Procesador RISC (Computador de Juego de Instrucciones Reducido)

Los modelos de la gama baja disponen de un repertorio de 33 instrucciones, 35 los de la gama media y casi 60 los de la alta.

5ª. Todas las instrucciones son ortogonales

Cualquier instrucción puede manejar cualquier elemento de la arquitectura como fuente o como destino.

6ª. Arquitectura basada en un banco de registros.

Esto significa que todos los objetos del sistema (puertos de E/S, temporizadores, posiciones de memoria, etc.) están implementados físicamente como registros.

7ª. Diversidad de modelos de microcontroladores con prestaciones y recursos diferentes.

La gran variedad de modelos de microcontroladores PIC permite que el usuario pueda seleccionar el más conveniente para su proyecto.

8ª. Herramientas de soporte potentes y económicas

La empresa Microchip y otras que utilizan los PIC ponen a disposición de los usuarios numerosas herramientas para desarrollar hardware y software. Son muy abundantes los programadores, los simuladores software, los emuladores en tiempo real, ensambladores, Compiladores C, Intérpretes y Compiladores Basic, etc.

2.1.3 Las Tres Gamas De PIC.

Para resolver aplicaciones sencillas se precisan pocos recursos; en cambio, las aplicaciones grandes requieren numerosos y potentes. Siguiendo esta filosofía, Microchip construye diversos modelos de microcontroladores orientados a cubrir, de forma, las necesidades de cada proyecto. Así, hay disponibles microcontroladores sencillos y baratos para atender las aplicaciones simples y otros complejos y más costosos para las de mucha envergadura.

Entre los fabricantes de microcontroladores hay dos tendencias para resolver las demandas de los usuarios:

1ª. Microcontroladores de arquitectura cerrada

Cada modelo se construye con un determinado CPU, cierta capacidad de memoria de datos, cierto tipo y capacidad de memoria de instrucciones, un número de E/S y un conjunto de recursos auxiliares muy concreto. El modelo no admite variaciones ni ampliaciones.
La aplicación a la que se destina debe encontrar en su estructura todo lo que precisa y, en caso contrario, hay que desecharlo. Microchip ha elegido principalmente este modelo de arquitectura.

2ª. Microcontroladores de arquitectura abierta

Estos microcontroladores se caracterizan porque, además de disponer de una estructura interna determinada, pueden emplear sus líneas de E/S para sacar al exterior los buses de datos, direcciones y control, con lo que se posibilita la ampliación de la memoria y las E/S con circuitos .integrados externos. Microchip dispone de modelos PIC con arquitectura abierta, sin embargo, esta alternativa se escapa de la idea de un microcontrolador incrustado y se asemeja a la solución que emplean los clásicos microprocesadores.

En nuestra opinión, los verdaderos microcontroladores responden a la arquitectura cerrada y permiten resolver una aplicación con un solo circuito integrado y a precio muy reducido.

La mayoría de los sistemas de control incrustados requieren CPU, memoria de datos, memoria de instrucciones, líneas de E/S, y diversas funciones auxiliares como temporizadores, comunicación serie y otras. La capacidad y el tipo de las memorias, el número de líneas de E/S y el de temporizadores, así como circuitos auxiliares, son parámetros que dependen exclusivamente de la aplicación y varían mucho de unas

situaciones a otras. Quizás se pueda considerar la decisión más importante del proyecto la elección del modelo de microcontrolador. Para adaptarse de forma óptima a las necesidades de los usuarios, Microchip oferta tres gamas de microcontroladores de 8 bits

Con las tres gamas de PIC se dispone de gran diversidad de modelos y encapsulados, pudiendo seleccionar el que mejor se acople a las necesidades de acuerdo con el tipo y capacidad de las memorias, el número de líneas de E/S y las funciones auxiliares precisas. Sin embargo, todas las versiones están construidas alrededor de una arquitectura común, un repertorio mínimo de instrucciones y un conjunto de opciones muy apreciadas, como el bajo consumo y el amplio margen del voltaje de alimentación.

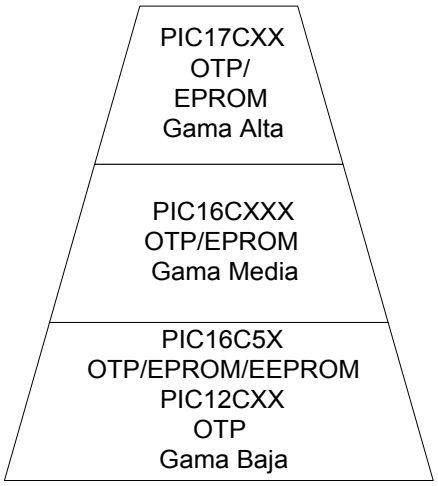


Figura 2 3

En la figura 2.3 se presentan las prestaciones de cada modelo de PIC.

Junto con los microcontroladores, Microchip ha creado una serie de herramientas de ayuda al desarrollo del hardware y software de los proyectos de aplicación, que son válidas para la mayoría de sus modelos y que se citan a continuación.

1º Ensamblador MPASM.

2° Simulador software MPSlM. No soporta los modelos PICI7CXX.

3º Compilador de lenguaje C, MP-C.

4º Programador universal PRO MATE.

5º Emulador universal PIC MASTER.

6° Herramienta de desarrollo para Lógica difusa FUZZY TECH-MP.

7º Entorno de Desarrollo Integrado MPLAB

Figura 2 4

La figura 3.4 muestra un gráfico que aclara la relación "precio/prestaciones de los modelos de PIC.

2.1.3.1 Gama Baja.

La gama baja de los PIC encuadra nueve modelos fundamentales en la actualidad, cuyas principales características aparecen en las figuras anteriores.

La memoria de programa puede contener 512, 1 k. y 2 k palabras de 12 bits, y ser de tipo ROM, EPROM. También hay modelos con memoria OTP, que sólo puede ser grabada una vez por el usuario. La memoria de datos puede tener una capacidad comprendida entre 25 y 73 bytes. Sólo disponen de un temporizador (TMR0), un repertorio de 33 instrucciones y un número de pines para soportar las E/S comprendido entre 12 y 20. El voltaje de alimentación admite un valor muy flexible comprendido entre 2 y 6,25 V, lo cual posibilita el funcionamiento mediante pilas corrientes teniendo en cuenta su bajo consumo (menos de 2 mA a 5 V y 4 MHz).

Al igual que todos los miembros de la familia PIC16/17, los componentes de la gama baja se caracterizan por poseer los siguientes recursos.

1. Sistema POR (POWER ON RESET).
Todos los PIC tienen la facultad de generar una autoreinicialización o autoreset al conectarles la alimentación.

2. Perro guardián, (Watchdog).

Existe un temporizador que produce un reset automáticamente si no es recargado antes que pase un tiempo prefijado. Así se evita que e! sistema quede "colgado" dado en esa situación el programa no recarga dicho temporizador y se genera un reset.

3. Código de protección.
Cuando se procede a realizar la grabación del programa, puede protegerse para evitar su lectura. También disponen, los PIC de posiciones reservadas para registrar números de serie, códigos de identificación, prueba, etc.

4. Líneas de E/S de alta corriente.
Las líneas de E/S de los PIC pueden proporcionar o absorber una corriente de salida comprendida entre 20 y 25 mA, capaz de excitar directamente ciertos periféricos.

5. Modo de reposo (bajo consumo o SLEEP).
Ejecutando una instrucción (SLEEP), el CPU y el oscilador principal se detiene y se reduce notablemente el consumo.

Para terminar conviene nombrar dos restricciones importantes sobre los componentes de la gama baja.

1ª) La pila o "stack" sólo dispone de dos niveles lo que supone no poder encadenar más de dos subrutinas.

2ª) Los microcontroladores de la gama baja no admiten interrupciones.

2.1.3.2 Gama Media.

En esta gama sus componentes añaden nuevas prestaciones a las que poseían los de la gama baja, haciéndoles más adecuados en las aplicaciones complejas. Admiten interrupciones, poseen comparadores de magnitudes analógicas, convertidores A/D, puertos serie y diversos temporizadores.

Algunos modelos disponen de una memoria de instrucciones del tipo OTP ("One Time Programmable"), que sólo la puede grabar una vez el usuario y que resulta mucho más económica en la implementación de prototipos y pequeñas series.

Hay modelos de esta gama que disponen de una memoria de instrucciones tipo EEPROM, que, debido a que se pueden borrar eléctricamente, son mucho más fáciles de reprogramar que las EPROM, que tienen que ser sometidas a rayos ultravioleta durante un tiempo determinado para realizar dicha operación.

Comercialmente el fabricante ofrece cuatro versiones de microcontroladores en prácticamente todas las gamas.

1ª. Versión EPROM borrable con rayos ultravioleta. La cápsula dispone de una ventana de cristal en su superficie para permitir el borrado de la memoria de programa al someterla durante unos minutos a rayos ultravioleta mediante lámparas fluorescentes especiales.

2ª. Versión OTP. “Programable una sola vez”. Son similares a la versión anterior, pero sin ventana y sin la posibilidad de borrar lo que se graba.

3ª. Versión QTP. Es el propio fabricante el que se encarga de grabar el código en todos los chips que configuran pedidos medianos y grandes.

4ª. Versión SQTP. El fabricante solo graba unas pocas posiciones de código para labores de identificación, numero de serie, palabra clave, checksum, etc.

El temporizador TMR1 que hay en esta gama tiene un circuito oscilador que puede trabajar de forma asíncrona y que puede incrementarse aunque el microcontrolador se halle en el modo de reposo ("sleep"), posibilitando la implementación de un reloj en tiempo real.

Las líneas de E/S del puerto B presentan una carga "pull-up" activada por software.

2.1.3.3 Gama Alta

En la actualidad, esta gama esta formada por tres modelos cuyas prestaciones mas representativas se mostraron en las tablas anteriores.

Los dispositivos PIC17C4X responden a microcontroladores de arquitectura abierta pudiéndose expansionar en el exterior al poder sacar los buses de datos, direcciones y control. Así se pueden configurar sistemas similares a los que utilizan los microprocesadores convencionales, siendo capaces de ampliar la configuración interna del PIC añadiendo nuevos dispositivos de memoria y de E/S externas. Esta facultad obliga a estos componentes a tener un elevado número de pines comprendido entre 40 y 44. Admiten interrupciones, poseen puerto serie, varios temporizadores y mayores capacidades de memoria, que alcanza las 8 k palabras en la memoria de instrucciones y 454 bytes en la memoria de datos.

2.2 Microcontrolador PIC 16F84A.

El PIC16F84 se encuentra en la gama media de microcontroladores de Microchip, cuenta con un total de 35 instrucciones, lo cual lo hace idóneo para aplicaciones comunes.
Características

- 1024 (1K) palabras de memoria de programación.
- 68 bytes de RAM
- 64 bytes de EEPROM.
- Palabras de instrucción con un ancho de 14 bits
- 15 registros de funciones especiales
- Frecuencia máxima 20 MHz

La arquitectura del procesador es del modelo Harvard. El CPU se conecta de forma independiente y con buses distintos con la memoria de instrucciones y con la de datos. La arquitectura del modelo Harvard permite al CPU acceder simultáneamente a las 2 memorias.

Se aplica la técnica de segmentación ("pipe-line") en la ejecución de las instrucciones. La segmentación permite al procesador realizar al mismo tiempo la ejecución de una instrucción y la búsqueda del código de la siguiente. De esta forma se puede ejecutar cada instrucción en un ciclo (un ciclo de instrucción equivale a cuatro ciclos de reloj)

Las instrucciones de salto ocupan dos ciclos al no conocer la dirección de la siguiente instrucción hasta que no se haya completado la bifurcación.

El formato de todas las instrucciones es de la misma longitud; todas las instrucciones tienen 14 bits. Dicha característica optimiza la memoria de instrucciones y facilita enormemente la construcción de ensambladores y compiladores.

El procesador es tipo RISC (Computador de juego de Instrucciones Reducido) que cuenta con un total de 35 instrucciones. Cualquier instrucción puede manejar cualquier elemento de la arquitectura como fuente o como destino

Su arquitectura se basa en "bancos de registros" es decir (puertos de E/S, temporizadores, posiciones de memoria, etc.) están implementados físicamente como registros.

Tal como se muestra en la figura 3.5, posee 2 puertos, el A de 5 terminales y el B de 8, estas terminales son nombradas RA0 a RA4 y RB0 a RB7 y pueden ser programados como entradas o salidas. Cada puerto cuenta con una resistencia de pull-up interna que puede ser desconectada mediante programación.

El pin RA4 cumple una doble función de puerto E/S y de entrada de un temporizador, RB0 también es entrada de interrupción y el pin 4 (MCLR/VPP) que también cumple una doble función, es el pin de programación y en funcionamiento es el pin de reset.

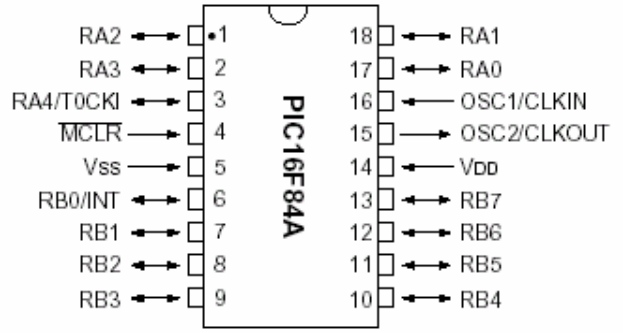
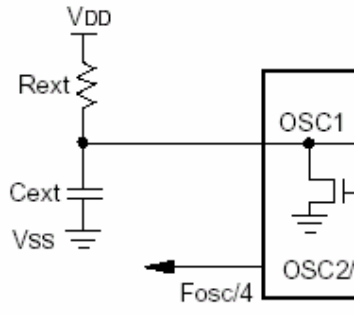
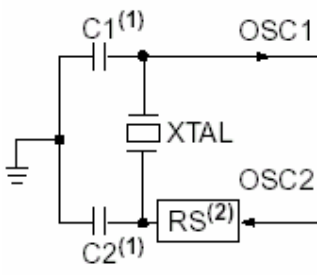


Figura 2 5

Los pines 15 y 16 son únicamente para el oscilador externo. Para que el dispositivo funcione adecuadamente, los pines 14 y 5 deben ser conectados a una fuente de voltaje cuyo valor no sobrepase los 5V.

El oscilador externo puede ser conectado de 4 formas distintas, tal como se muestra en la siguiente tabla:

	Oscilador RC	Oscilador compuesto por una resistencia y un condensador. Generalmente es empleado en aplicaciones que no requieren de mucha precisión
	Oscilador XT	Oscilador compuesto por dos capacitores y un cristal Para aplicaciones de un poco mas de precisión
	Oscilador HS	Oscilador compuesto por un cristal de alta velocidad.
	Oscilador LP	Oscilador Compuesto por un cristal de baja frecuencia y bajo consumo de potencia

2.2.1 Organización De La Memoria.

La memoria de datos está dividida en 2 bloques. Estos bloques son “la memoria de programa” y “la memoria de datos”. Cada bloque tiene su propio bus, por lo que el acceso a cada bloque puede ocurrir durante el mismo ciclo de reloj.

La memoria de datos puede aun ser subdividida en ***RAM de propósito general*** (***GPRs,*** General Purpose Resgisters) y ***registros de funciones especiales*** (***SFRs***, Special Function Registers). Los SFRs controlan la operación de los dispositivos.

El área de los bancos GPR permiten 116 bytes de propósito general de RAM. Los bancos requieren el uso de bits de control para la selección de los bancos. Estos bits de control son localizados en el registro STATUS. La figura 3.6 muestra la organización del mapa de la memoria de datos.

El banco 0 es seleccionado limpiando el bit RP0 (STATUS<5>), colocándolo a 1 se selecciona el banco1. Cada banco tiene una extensión de 128 bytes (7F), las primeras 12 localidades de cada banco son reservados para los registros de funciones especiales(SFR), los sobrantes son registros de propósitos generales como SRAM

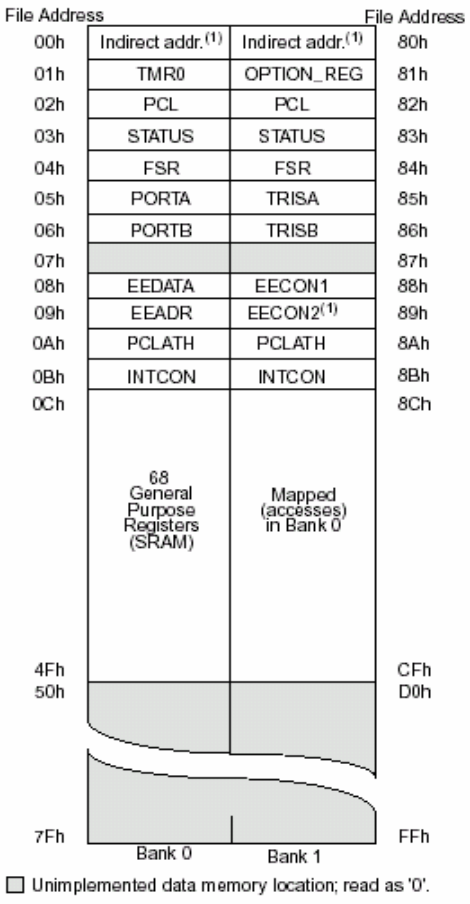


Figura 2 6

Para que los puertos A y B funcionen como entrada o salida, deben ser configurados según sea el caso. La asignación de pines se hace programando los registros ***TRISA*** y ***TRISB***. Si se asigna un 0 a un bit del registro, el pin correspondiente en el puerto será salida y si se le asigna 1 este será entrada. Ejemplo: si TRISA = 10110 (El PUERTO A solo posee 5 pines), los pines del puerto A van a adquirir el estado mostrado en la siguiente tabla.

TRISA	CONDICION	ESTADO DEL PUERTO A
RA0	0	SALIDA
RA1	1	ENTRADA
RA2	1	ENTRADA
RA3	0	SALIDA
RA4	1	ENTRADA

El PUERTOB es igual. Ejemplo, si TRISA = 00110001, el puerto B funcionaría de acuerdo a la siguiente tabla.

TRISB	CONDICION	ESTADO DEL PUERTO B
RB0	1	ENTRADA
RB1	0	SALIDA
RB2	0	SALIDA
RB3	0	SALIDA
RB4	1	ENTRADA
RB5	1	ENTRADA
RB6	0	SALIDA
RB7	0	SALIDA

Cuando el PIC arranca siempre se encontrara en banco 0, y los registros TRISA y TRISB para configurar los puertos se encuentran en el banco 1. Se bebe de modificar el registro STATUS (Ver figura 3.7).

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R	R	R/W	R/W	R/W
IRP	RP1	RP0	/TO	/PD	Z	DC	C

C: Acarreo en el 8º bit.
1 = acarreo en la suma y no en la resta. 0 = acarreo en la resta y no en la suma
DC: Acarreo en el 4º bit de menor peso.
Igual que C.
Z: Zero.
1 = El resultado de alguna operación es 0. 0 = El resultado es distinto de 0
/PD: Power Down.
1 = Recién encendido o tras CLRWDT. 0 = Tras ejecutar una instrucción SLEEP
/TO: Timer Out.
1 = Recién encendido, tras CLRWDT, o SLEEP. 0 = Saltó el WDT

Figura 2 7

Si el bit RP0 es 0 entonces se accede al banco 0, si es 1 se accede al banco 1.
Para modificar este bit del registro se puede utilizar las siguientes instrucciones:
BSF (bit set file register) pone uno lógico en la localidad de RAM especificada.
BCF (bit clear file register) pone cero en la localidad de RAM especificada.

Por ejemplo; para poner a uno el bit RP0 seria

```
BSF   STATUS,5
```

2.3 DISPOSITIVOS LOGICOS PROGRAMABLES

Un dispositivo lógico programable o PLD(Programmable Logic Device), es un dispositivo lógico cuyas características pueden ser modificadas y almacenadas mediante programación. El dispositivo más simple es la PAL(Programmable Array Logia). El circuito interno de una PAL consiste en una matriz de conexiones, una matriz de compuertas AND y un arreglo de compuertas OR. Una matriz de conexiones es una red de conductores distribuidos en filas y columnas con un fusible en cada punto de intersección. Mediante el cual se seleccionan cuales entradas del dispositivo serán conectadas al arreglo AND cuyas salidas serán conectadas al arreglo OR y de esta manera obtener una función lógica en forma de productos.

La mayoría de los PLDs están formados por una matriz de conexión, una matriz de compuerta AND y una matriz de compuertas OR y algunos además con registros. Las matrices pueden ser fijas o programables. Con estos recursos se implementan las funciones lógicas deseadas mediante un software especial y un programador de dispositivos.
La figura 2.8 muestra la estructura básica para la mayoría de los PLD:

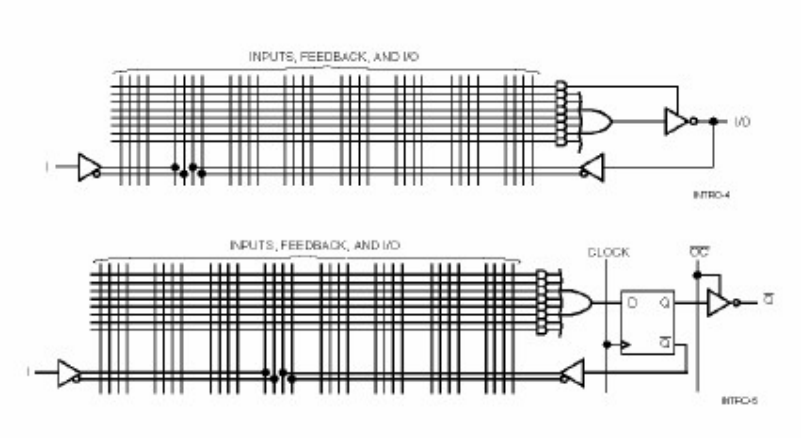


Figura 2 8 Estructura básica del PLD

Un CPLD (Complex Programmable Logia Device) extiende el concepto de un PLD a un mayor nivel de integración ya que permite implementar sistemas mas eficientes por que utiliza menos espacio, mejoran la confiabilidad del circuito y reducen los costos. Un CPLD se forma con múltiples **bloques lógicos**, cada uno similar a un PLD. Los bloques lógicos se comunican entre si utilizando una matriz programable.

2.3.1 LA FAMILIA ISPLSI 1000 DE LATTICE

La unidad básica para esta familia es un cuadro genérico del bloque de lógica (GLB). El ispLSI 1016 contiene 16 de estos bloques. La figura 2.9 muestra la estructura interna del 1016

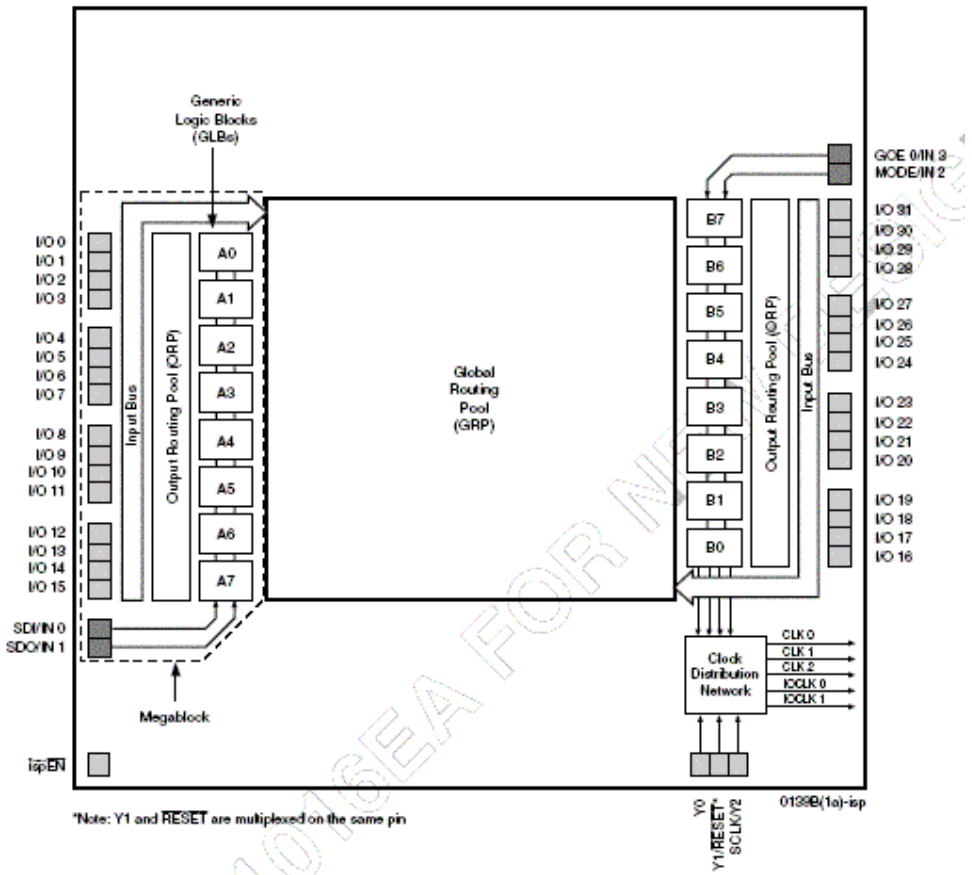


Figura 2 9 Diagrama del Bloque Funcional

EL BLOQUE GENÉRICO DE LÓGICA

El bloque genérico de lógica (GLB) es el bloque estándar de la lógica de los dispositivos del highdensity ispLSI de Lattice Semiconductor Corporation (LSC). Un GLB tiene 18 entradas de información, cuatro salidas y la lógica necesaria para poner la mayoría de las funciones de la lógica en ejecución del estándar. La lógica interna del GLB se divide en cuatro secciones separadas: la matriz AND, el término del producto que comparte la matriz (PTSA), los registros de Reconfigurable, y las funciones de control (véase la figura 2.10). La matriz AND consiste en 20 términos del producto, que pueden producir el producto lógico de cuales quiera de las 18 entradas de información de GLB. Los dieciséis de las entradas de información vienen de la Global Routin Pool, y son señales de retorno de cualquiera del GLBs o entradas de información de las celdas externas de la entrada-salida. Las dos entradas de información restantes vienen directamente a partir de dos contactos dedicados de la entrada de información. Estas

señales están disponibles para los términos del producto en la lógica de verdad y las formas complementadas que hace la reducción boleana de la lógica más eficiente.

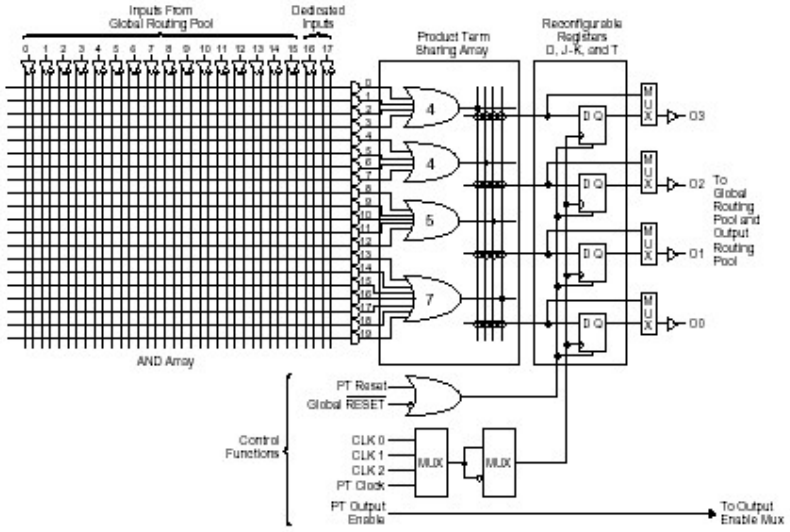


Figura 2 10: Ejemplo de Matriz de Termino de Producto Compartido

El PTSA toma los 20 términos del producto y los encamina a las cuatro salidas de GLB. Hay cuatro compuertas OR, con el producto cuatro, cuatro, cinco y siete productos términos cada uno (véase figura 2.10). La salida de cualesquiera de estas compuertas OR se puede encaminar a cualesquiera de las cuatro salidas de GLB, y si más términos del producto son necesarios, el PTSA puede combinarlas como necesarias. Además, el PTSA puede compartir los términos del producto similares a un dispositivo de FPLA. Si la preocupación principal del utilizador es velocidad, el PTSA puede utilizar un circuito de derivación que proporcione a cuatro términos del producto a cada salida, para aumentar el funcionamiento de la celda. Esto se puede hacer a cualesquiera o a todas las cuatro salidas del GLB.

Los registros Reconfigurables consisten en cuatro flip-flop del tipo D con una puerta de XOR en la entrada. La puerta de XOR en el GLB puede ser utilizada como elemento de la lógica o configurar de nuevo el flip-flop del tipo D para emular un flip-flop J-k o del tipo T. Esto simplifica grandemente el diseño de contadores, de comparadores y las funciones del tipo de ALU. Los registros pueden ser puenteados si el usuario necesita una salida combinatoria. Cada salida del registro se trae nuevamente dentro de Global Routing Pool y también se trae a las celdas de I/O vía la Output Routing Pool. Los registros Reconfigurables no están disponibles cuando se utiliza puente de producto de cuatro términos. El PTSA es bastante flexible permitir que estas características sean utilizadas en virtualmente cualquier combinación que el usuario desee

Celda de I/O

La celda de I/O (véase la figura) se utiliza para encaminar la entrada, la salida o las señales bidireccionales conectadas con el pin de I/O. Una entrada de la lógica viene del ORP, y la otra viene de puente más rápido de ORP . Un par de multiplexores selecciona qué señal será utilizada, y su polaridad.

Como con la trayectoria de datos, un multiplexor selecciona la polaridad de la señal. La salida se puede fijar a una lógica alta (permitido) cuando se desea un pin de la salida, o a lógica bajo (inhabilitado) cuando un pin de la entrada es necesario. La señal global del reajuste (REAJUSTE) es conducida por el pin bajo activo del reajuste de la ruta. Este reajuste está conectado siempre con todos los registros de GLB y de I/O. Cada celda de I/O puede seleccionar individualmente una de las dos señales del reloj (IOCLK 0 o IOCLK 1). Estas señales del reloj son generadas por la red de distribución de reloj.

Usando los multiplexores, la celda de la entrada-salida se puede configurar como una entrada de información, una salida, una salida de 3 estados o entrada-salida bidireccional. El registro del tipo D se puede configurar como un nivel transparente sensible latch o un flip-flop accionado borde para salvar los datos entrantes. El cuadro 2.11 ilustra algunas de las varias configuraciones de la celda I/O posibles.

Hay un resistor activo pull-up en los contactos de la entrada-salida que se utiliza automáticamente cuando el pin no está conectado. Una opción existe para tener activo el resistor pull-up conectados con todos los pines. Esto mejora la inmunidad de ruido y reduce Icc para el dispositivo. Los pines de la entrada-salida usados por el diseño funcional se pueden configurar individualmente para utilizar o no utilizar pull-up.

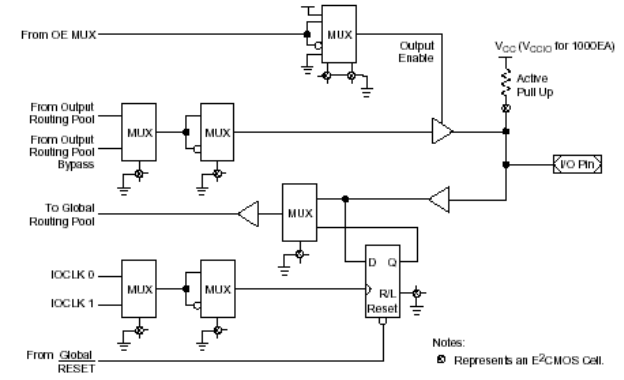
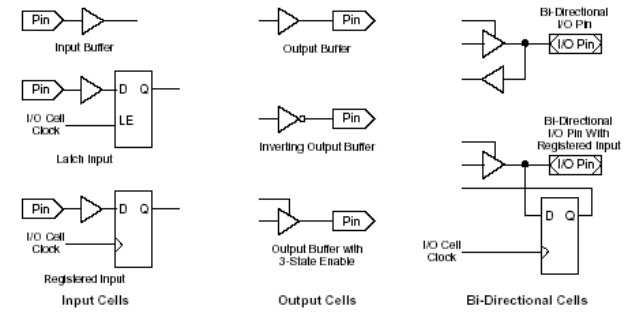


Figure 10. Examples of I/O Cell Configurations



.

Figura 2 11 Arquitectura de las celdas I/O

Red de Distribución de Reloj

Las redes de distribución de reloj se muestran en el cuadro 13. Generan cinco señales globales CLK 0, CLK 1, CLK 2, IOCLK 0 e IOCLK 1del reloj. Los primeros tres, CLK 0, CLK 1 y CLK 2 se pueden utilizar para registrar todo el GLBs en el dispositivo. Semejantemente, las señales de IOCLK 0 y de IOCLK 1 se utilizan para registrar todas las celdas de la entrada-salida en el dispositivo. Hay cuatro contactos dedicados del reloj del sistema (Y0, Y1, Y2, Y3) (tres para el ispLSI 1016 (Y0, Y1, Y2)), que se pueden dirigir a cualquier GLB o a cualquier celda de la entrada-salida usando la red de distribución de reloj. Las otras entradas de información a la red de distribución de reloj son las cuatro salidas de un reloj dedicado GLB (" C0 " para el ispLSI 1032 se muestra en la figura 2.12 a). Estas salidas del reloj GLB se pueden utilizar para crear un esquema interno definido por el usario el registrar.

Por ejemplo, el reloj GLB se puede registrar usando el contacto principal externo Y0 del reloj conectado con la señal global CLK 0 del reloj. Las salidas del reloj GLB alternadamente pueden generar una señal de dividir por la señal del CLK 0 que se puede conectar con las líneas globales del reloj de CLK 1, de CLK 2, de IOCLK 0 o de IOCLK 1.

Todos los GLBs tienen la capacidad de generar sus propios relojes asíncronos usando el término del producto del reloj (PT12). Alimentación de CLK 0, de CLK 1 y de CLK 2 a sus entradas de información correspondientes del reloj MUX en todo el GLBs (véase figura 2.12 b). Los dos relojes de la entrada-salida generados en la red de distribución de reloj IOCLK 0 e IOCLK 1, se traen a toda la celda de las celdas de la entrada-salida y los programas de utilizador de la entrada-salida para utilizar uno de los dos.

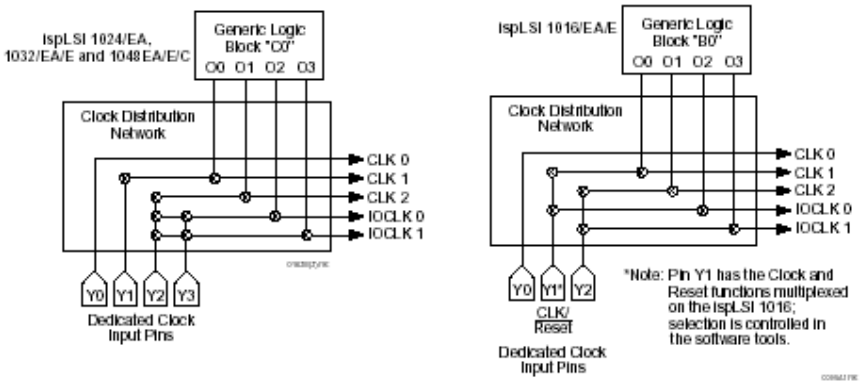


Figura 2 12 Clock Distribution Networks

2.4 HARDWARE Y SOFTWARE DE LA TARJETA.

La tarjeta adquisidora de datos es básicamente una maquina virtual, en la cual la esencia de la maquina es el microcontrolador PIC, lo que hace que las instrucciones del PIC se vuelvan microcodigos, es decir que la maquina tiene sus propio códigos y registros, esta maquina no usa instrucciones como XOR o DECFSZ propias del PIC, en lugar de eso utiliza instrucciones para manipular registros, muestrear, descargar muestras, etc. Lo que hace que el programador o usuario no necesite saber como programar un PIC por lo que resulta muy fácil para el la programación de la tarjeta.

Ya que la maquina posee sus propia arquitectura hace claramente la diferencia entre la PC y la tarjeta, hace que el programador desarrolle su programa en la plataforma que el desee o que maneje mejor ya que solo nesecita una interfase serial que puede ser utilizada en cualquier sistema; por ejemplo si se nesecita cargar cierto registro como R6 con un valor de 5A solo se debe de ejecutar

[6]@[5A]S

que corresponden a una cadena de caracteres ASCII en el puerto Serie, pudiendo se implementar el código fácilmente en cualquier lenguaje de programación, y siendo manejado por cualquier tipo de PC; por lo que la tarjeta trabaja en forma independiente de la PC es decir no necesitan el mismo reloj o sincronía entre los 2, a lo que el software solo se encarga de graficar y procesar la señal. La tarjeta será conectada a una PC con la interfase Visual Basic.

La tarjeta adquisidora de datos solo requiere de tres pasos:

- Programar los registros internos.
- Usar los comandos de adquisición de datos
- y recuperar los datos cuando estén disponibles.

Cada vez que un comando es enviado de la PC a la tarjeta esta le devuelve un eco es decir el mismo comando, para establecer un handshake

2.4.1 REGISTROS DEL PIC

El PIC maneja alrededor de 15 registros para el funcionamiento de la maquina. La operación de la tarjeta y sus instrucciones son referidas al uso de ellos. Todas las fuentes de los registros en el PIC están fuera de la tarjeta adquisidora.
La función y descripción de estos registros se hace a continuación:

R0
Registro de entrada:
Todo dato proveniente de la PC es almacenado en este registro.

R1
Registro puntero:
Datos que son traspasados a otros registros son trasladados de R0 a R1

R2
Registro fuente:
En este se almacena el número de registro donde se almacenara el dato en R1

R3
Adress virtual low:
Parte baja del registro virtual del contador de direcciones.

R4
Adress virtual high:
Parte alta del registro virtual del contador de direcciones

R5
Trigger lógico:
Patrón de niveles lógicos para generar disparo

R6
Trigger mascara:
Bits a tomar en cuenta en la comparación.

R7
Option:
Registro de un solo byte al PLD para configuración de disparo.

R8
MODO:
Selección del tipo de muestreo que se va a ejecutar

R9
Adrres counter LO:
Parte baja del registro de direcciones de la RAM.

R10
Adress counter HI:
Parte alta del registro de direcciones de la RAM.

R11
Post trigger delay LO:
Registro para retraso del post disparo.

R12
Post Trigger Delay Hi:
Registro para retrazo del post disparo.

R13
TB.
Registro que contiene la base de tiempo de expansión.

R14
Canal:
Registro que contiene los parámetros de cada canal.

2.4.2 SET DE INSTRUCCIONES DE LA MAQUINA

El set de instrucciones es una secuencia de byte que están entre 0 a 255. Estos están en código ASCII entre los valores de 0 a 128.

El esquema general para la utilización de los valores de código-byte y sus símbolos ASCII es el siguiente:

Números	**usados para entrar datos**
Operadores	**manipulan el valor de los registros**
Letras minúsculas	**Usados para entrar datos**
Letras mayúsculas	**operaciones**

00 Reset

23 # carga al registro fuente
Guarda el valor del registro R0 en el registro R2 el cual es la registro fuente

2b + incrementar registro
Incrementa el registro apuntado por R1

2d - decrementar registro
Decrementa el registro apuntado por R1

30 0 entre el nible 0
Incrementa R0 en 0.

31 1 entre el nible 1
Incrementa R0 en 1.

32 2 entre el nible 2
Incrementa R0 en 2

33 3 entre el nible 3
Incrementa R0 en 3.

34 4 entre el nible 4
Incrementa R0 en 4.

35 5 entre el nible 5
Incrementa R0 en 5.

36 6 entre el nible 6
Incrementa R0 en 6.

37 7 entre el nible 7
Incrementa R0 en 7.

38 8 entre el nible 8
Incrementa R0 en 8.

39 9 entre el nible 9
Incrementa R0 en 9.

30 0 entre el nible 0
Incrementa R0 en 0.

3c < obtener el valor del contador de RAM
Descarga los 16 bits de contador de direcciones

3e > programa los registros del PLD
Carga los 3 byte de datos dentro de PLD

40 @ cargar dato
Carga el contenido de R0 en R1

30 0 entre el nible 0
Incrementa R0 en 0

53 S descargar muestras
Descarga toda la RAM hacia la PC hasta la dirección apuntada por el contador de direcciones

54 T trazar
Comienza a muestrear según el modo seleccionado.

5b [limpiar R0

5d] intercambio de nibbles R0
R0(0..3) es intercambiado con R0(4..7).

61 a enter nibble 'a' hex
Incrementa R0 en 10.

62 b enter nibble 'b' hex
Incrementa R0 en 11.

63 c enter nibble 'c' hex
Incrementa R0 en 12.

64 d enter nibble 'd' hex
Incrementa R0 en 13.

65 e enter nibble 'e' hex
Incrementa R0 en 14.

66 f enter nibble 'f' hex
Incrementa R0 en 15.

6c l carga R0 con R2
El contenido del registro que apunta R2 es almacenado en R0

6e n siguiente dirección
Incrementa el registro R1

73 u guardar R0 en R1
Copia los contenidos de R0 al registro apuntado por R1.

2.4.3 CARACTERISTICAS GENERALES

El comando T (trazar) es el comando principal de la tarjeta adquisidora de datos. Antes de utilizar este comando es necesario descargar ciertos registros al PLD correctamente.

2.4.3.1 REGISTROS DEL PLD

Antes de comenzar a muestrear es necesario colocar dentro del PLD ciertos registros, los cuales son cargados con el comando ">", estos registros solo necesitan ser programado una sola vez, a menos que se necesiten cambiar algunos parámetros de ellos.

Ejemplo:
Cargar al PLD con 01,0f, aa dentro de los registros R7, R6, R5 respectivamente
[3]@[1]sns[0f]ns[aa]ns>

[3]@	**entra 3 dentro de R0, luego lo copias a R1**
[1]s	**entra 01 en R0, y lo guarda en el registro apuntado por R1**
ns	**incrementa R1 a 4, luego guarda R0 en R1**
[0f]ns	**entra 0f en R0 incrementa R1 y almacena en lo apuntado por R1**
[aa]ns	**entra aa en R0 incrementa R1 y almacena en lo apuntado por R1**
>	**carga el PLD con R7, R6, R5.**

Registros de canal
El registro CANAL controla los canales A/B y la selección de la atenuación del el mismo durante el muestreo. Este registro contiene 2 nibbles el primario (bajo, Bits 3..0) y el secundario(alto, Bits 7..4), el primario es el canal donde se capturan datos, este es puesto dentro del puerto A y el secundario sirve para las funciones del modo DUAL donde se alternara con el primario para la captura de muestras. El formato para ambos nibbles es el mismo

Los Bits 1,0 : *controlan el rango de atenuación de la señal a muestrear.*
Bit 2: *selección del canal a muestrear canal A/B*
Bit 3: *estado del reloj.*

El rango de atenuación para la entrada BNC, también puede ser utilizada una punta de prueba X 10

Rango de atenuación	**BNC x 1**	**BNC x 10**
00	± 130 mV	± 1.3 V
01	± 600 mV	± 6.0 V
10	± 1.2 V	± 12.0V
11	± 3.1	± 31.0V

El registro MODO de muestrear, determina si solo actúa el canal primario o si ambos canales son usados. Si esta en modo de muestreo Dual los 2 nibbles son intercambiados repetidamente después que un disparo se ha detectado.

2.4.3.2 OPCIONES DE MUESTREO

Cuando la tarjeta adquisidora comienza a muestrear, esta caerá en un lazo en la espera de un disparo. El registro MODO contiene un número para diferentes algoritmos para lazos. Los 4 bits menores de este registro son programados para seleccionar uno dentro de 4 modos disponibles de muestreos.

	Modo	**Canal**
0	Modo simple de muestreo	Un solo canal
1	Modo simple de muestreo	2 canales
2	Expansión de tiempo	un solo canal
3	Expansión de tiempo	2 canales

2.4.3.3 EXPANSIÓN DE TIEMPO

El registro TB contiene un byte que expande la base de tiempo. En los modos de expansión que puede detener el reloj par un conteo de este registró y luego dejar correr el reloj para abrir una ventana para muestrear la señal.

2.4.3.4 POST DISPARO

Estos son 2 registros que contienen un contador de 16 bits que cuenta ascendentemente después que un disparo es alcanzado. Este tiempo es un retrazo lento, donde el reloj no se detiene hasta que este retrazo expira.

2.4.3.5 COMANDO " T "

El comando T causa que la tarjeta adquisidora de datos comience a muestrear y quede en un lazo cerrado hasta la espera de un disparo, cuando la condición de disparo se encuentre el retrazo de post disparo es inicializado antes de detener la maquina. Y el contenido del contador de direcciones es leído y enviado a la PC por el puerto paralelo.

Un ejemplo del uso del comando T :

```
>T              recarga los registros del PLD y luego muestrea
7304            después del retrazo, el contador de dirección es descargado
S               se descargan las muestras hasta la dirección 7304
00,03,05,09,0c,1b   datos validos
```

2.4.4 HARDWARE

2.4.4.1 EL CPU

El cerebro de la tarjeta es el microcontrolador PIC 16F84 el cual controla todas las funciones de la tarjeta como lo son el reloj, la conversión, el almacenamiento etc.
El PIC 16F84 es un microcontrolador de fácil uso y altamente adecuado para aplicaciones donde se requiere una interfase entre circuitos lógicos. Su rápido set de instrucciones y su capacidad de puertos I/O y 3-state son una gran ventaja para el control de la tarjeta adquisidora.
Un diseño cuidadoso y planeado hace posible que sus 13 puertos sean suficientes para esta aplicación, ya que algunos pines son utilizados para más de una función, la descripción de pines es la siguiente:

RA0 **Rango 0**
RA1 **Rango 1**

Salidas que en modo de conteo controlan la ganancia de los amplificadores (ver diagrama)

RA2 **Canal A/B**

Pin de salida que cambia la fuente de la señal entre el canal A y canal B

RA3 **ZZ-clk**

Este es un pin 3-state. Este pin controla al reloj principal cuando este es entrada el reloj corre libremente a través de un flip-flop tipo D y cuando este es salida el PIC controla el comportamiento del reloj. Este control es importante, ya que por este podemos rolar los datos del PLD al PIC y viceversa y también para modular el muestreo para baja frecuencia en las medidas con base de tiempo expandido

RA4 **TRIGER**

Pin de entrada que sirve para detectar cuando ha ocurrido un disparo y entrar a post trigger

RB0 **Modo de conteo**
Entrada conectada a la salida del multiplexor (convertidor paralelo/serie)) para leer los 8 data bits provenientes del bus ADC-RAM . y puede ser utilizado leyendo el bit 7 de datos para detectar un cruce por cero en adición al sistema de disparo del PLD

Modo de registro
Señal de salida de datos para alimentar al PLD en cual nesecita 3 registros que se seteados .

RB1 **Modo de conteo**
Salida A0 para el multiplexor(convertidor paralelo/serie)

Modo de registro
Señal de entrada de datos del PLD. Esta es la entrada serial donde se descarga el contador de direcciones proveniente del PLD

RB2 **SEL 1**
RB3 **SEL2**
Pines de salida, para A1, A2 del multiplexor (convertidor paralelo/serie)

RB4 **CONTEO/REGISTRO**
Pin de salida que controla en que modo se encuentra la tarjeta. Los modos son conteo y registros. Un valor de 1 causa que el PLD cambie para recibir un nuevo dato dentro de sus registros. Un valor de 0 causa que el PLD muestree señales, dispare y descargue las muestras

RB5 **Serial OUT**
Salida serial hacia el puerto paralelo.
RB6 **Serial IN**
Entrada serial del puerto paralelo.

RB7 **STORE/READ**
Salida de control n para la RAM y el ADC. Esta señal selecciona lectura o escritura de la RAM, permitiendo escribir datos en la RAM, o para leerlos mas tarde.

2.4.4.2 EL PLD LOGICO

Las 2 principales funciones del PLD son direccionar la RAM y la generación de un comparador para la formación de un disparo

Todas las funciones de este dispositivo son manejados por el reloj en ambos modos
Modo de Conteo: RB4 en bajo, el direccionamiento de la RAM comienza y se monitorea el pin RA4 para la comparación de un disparo.

Modo de Registros RB4 en alto, es introducido los datos provenientes de RB0 y expulsados los datos hacia RB1. esto significa que se pueden alterar los registros cada vez que se desee después de después de cada muestreo

El PLD contiene 3 registros internos

R0 8 bit patron de disparo
R1 8 bit mascara de disparo
R2 1 bits de option

OPERACIÓN DE DISPARO

Los bits provenientes del bus análogo son comparados con un patrón de registros o ignorados cuando la mascara de disparo es 1
Por ejemplo:

Patrón	01100100
Mascara	00001111
Trigger	0110xxxx

Entonces el sistema de disparo compara con los datos análogos solo los cuatro bits más significativos para generar un disparo. Si se desea detectar los cruces por ceros se debe de colocar una mascara de 0111 1111 el cual se debe de esperar un cambio en el MSB del ADC.
Enmascarando los 4 bits LSB se puede seleccionar de 1 a 16 bandas de las salidas del ADC y dispararlo cuando estos crucen esta banda, muchas de las otras combinaciones no son necesarias

2.4.4.3 FUENTE DE PODER

El socket P1 conecta un transformador el cual suministra un voltaje de 10Vac, los diodos D1 y D2 forman un rectificador de media onda y pasa por un filtro el cual proporciona 10Vdc aproximadamente.

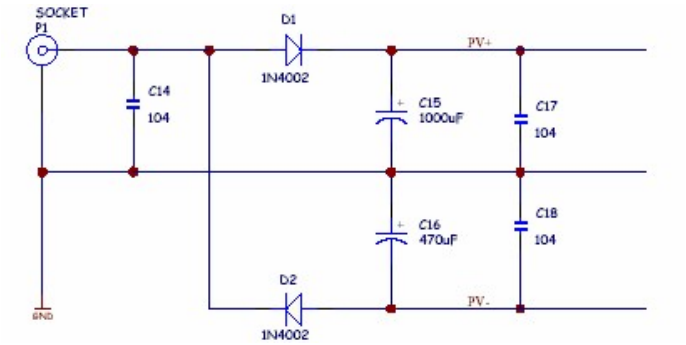


Figura 2 13

Los reguladores de voltaje U11 y U20 regulan para obtener un voltaje de +5, -5 que sirve para la alimentación de circuitos digitales

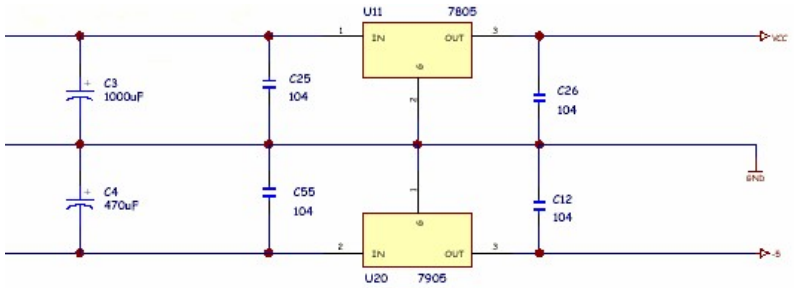


Figura 2 14

La fuente análoga suple +5 y -5 que son regulado por los U9 y U10, el inductor L1 aísla el tierra análogo del tierra digital a altas frecuencias, Para evitar ruido digital, tales como rebote. Esta fuente alimenta al ADC.

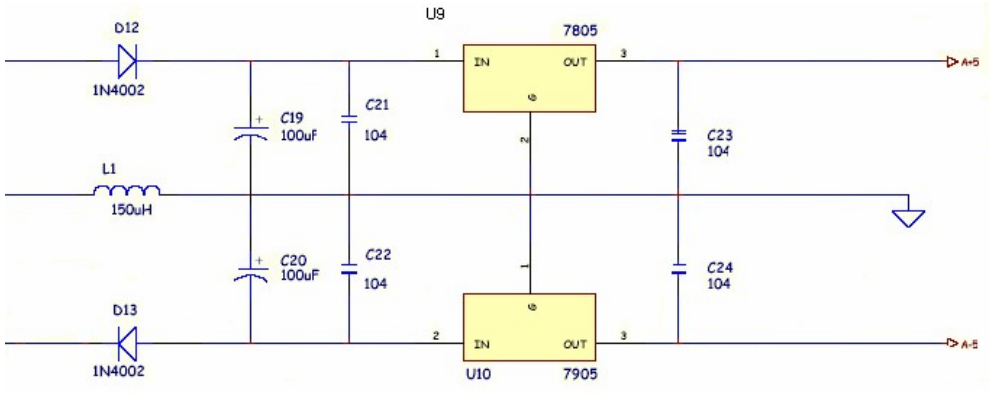


Figura 2 15

2.4.4.4 VECTOR DE INTERRUPCIÓN.

Como se mencionó anteriormente, cuando el PIC arranca, los puertos son configurados y este queda en un lazo infinito hasta que ocurre una interrupción. Durante la configuración se programa al PIC para que responda a un solo tipo de interrupción, esta interrupción es por cambio de estado del puerto B, dicha interrupción solo funciona para los pines de la parte alta del PUERTOB que son configurados como entrada. En nuestro caso hacemos uso del pin RB6.

Las interrupciones son habilitadas a través del registro INTCOM (0Bh, ver figura 2.16).

GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit7							bit0

Figura 2 16

Colocando a 1 los bits **GIE** (bit habilitador de interrupciones generales) y **RBIE** (habilita la interrupción por cambio de estado), el pin RB6 sólo responderá a los cambios de estado. Por lo tanto, el pin RB6 servirá para recibir datos desde el puerto serie RS-232. Cuando se configura la

interrupción, la línea se encuentra en uno lógico (1); es decir, línea inactiva (idle), y el microcontrolador sólo responderá cuando la línea cambie a cero lógico (0).

Cuando la computador huésped solicita datos a la tarjeta adquisidora de datos, esta envía un bit de inicio poniendo la línea a nivel bajo como se muestra en la figura, y disparando la bandera correspondiente a esa interrupción RBIF

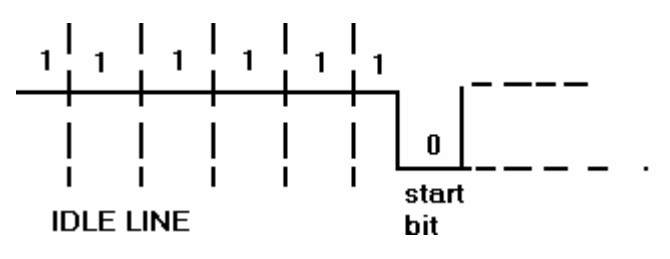


Figura 2 17

Una vez la bandera RBIF está activada, el programa salta a la dirección PC: 0004 que es el vector donde saltan todas las interrupciones. Cuando ha entrado a la subrutina de servicio de la interrupción se debe deshabilitar el bit GIE (Global Interrupt Enable) colocándole un cero, con esto se evita que salte otra vez al vector de interrupción. Una vez deshabilitada toda solicitud de interrupción, el programa se ejecuta normalmente hasta que se entregan ala PC las 8K muestras y sale de la interrupción para regresar al lugar donde se encontraba antes de la interrupción a esperar otra solicitud del a computadora.

La subrutina para habilitar la interrupción es:

```
BSF   INTCON, GIE     ; enable general interupts
BSF   INTCON, RBIE    ; enable interrupt on change
return
```

Antes de salir de la interrupción hay que limpiar la bandera de interrupción por cambio RBIF y volver habilitar la interrupción general, la secuencia es la siguiente

```
bcf      INTCON,RBIF
bsf      INTCON,GIE
retfie    ;exit
```

2.4.4.5 INTERFACE SERIAL

La comunicación con la PC y la tarjeta adquisidora de datos es a través de la interfase seria RS-232. Los niveles de voltaje de dicha interfase que representan un 0 lógico es +12 V y el nivel lógico 1 es –12 V.
El formato utilizado es transmisión asíncrona es 81N, no bit de paridad, 1 bit de inicio, 1 bit de paro, y 8 bit de información; la figura muestra el una trama

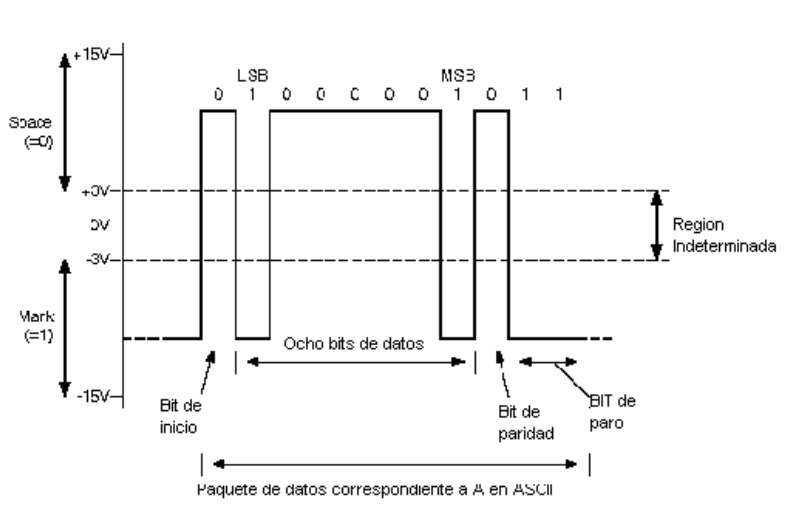


Figura 2 18

La velocidad de comunicación depende principalmente de la frecuencia del reloj del PIC, este trabaja a una frecuencia de 20MHz (un clock interno de 200 nS.) la velocidad mayor que se puede conseguir con este PIC es de 115.2KBaudios (115,200 baudio), por lo que el software utilizado debe de ser colocado a la misma velocidad .

La conversión de RS-232 a TTL es hecha por el IC MAX 232 el cual posee 2 canales para la comunicación, su configuración es sencilla y muy practica, solo nesecita 4 capacitores que su función es la de dobladores de voltaje y puede alcanzar velocidades máximas de 256K baudios, su alimentación es de 5 voltios; las conexiones con el puerto serie se hacen a través de los pines Rx y Tx para recibir y transmitir respectivamente, puede ser conectado dependido del puerto de la PC, ya sea DB9 o DB25, Los pines de transmisión y recepción de TTL son conectados al puerto B del PIC a los pines RB5 y RB6 respectivamente.

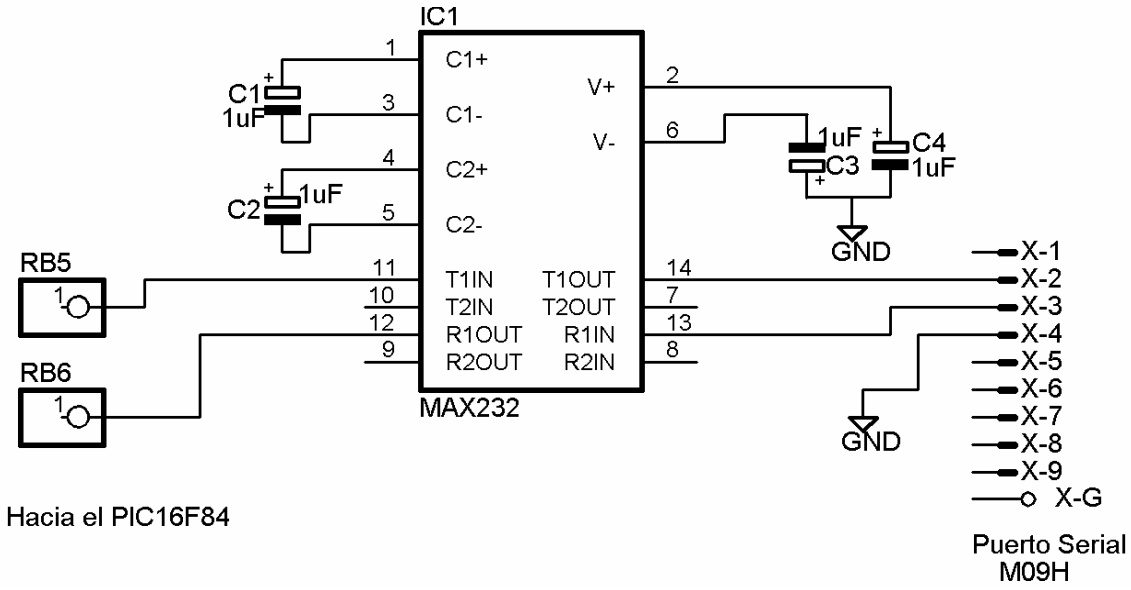


Figura 2 19

2.4.4.6 PARTE ANALOGA

El circuito análogo de la tarjeta adquisidor consta de 5 etapas bien definidas, las cuales incluye un convertidor ADC tipo flash, un buffer con ajuste de offset, un multiplexor para los rango de atenuación de la entrada, multiplexeo de canales y un buffer para la señal de entrada.

2.4.4.6.1 EL ADC

El convertidor ADC es tipo Flash el TLC 5540 el cual es el corazón de la parte análoga de la tarjeta adquisidora. Este trabaja a una velocidad de muestreo típica de 25 MHz y que posee un ancho de banda de 100 MHz.
La interfase es muy simple, el bus de datos análogo se desplaza hacia la entrada de datos de la RAM, al PLD y al convertidor paralelo/serie. La frecuencia de muestreo del ADC es la misma frecuencia con la que trabaja el PLD por lo que a cada ciclo de reloj se ejecuta una conversión de datos validos a lo que a la vez se almacena en una dirección de memoria que es incrementada a la misma frecuencia; la conversión de datos y almacenamiento de los mismos es controlada por la señal |STORE proveniente del pin RB7 del PIC cuando este se encuentra en estado bajo. Cuando la señal es colocada en alto el ADC se coloca en estado de alta impedancia y la RAM es estado de escritura.
La colocación del ADC es a través de un modulo, el cual consta de un paquete de 24 pines el cual es alimentado por fuentes análoga y digital. En el paquete se encuentra el ADC, resistencias y diodos los cuales regulan los voltajes de referencia.
El modulo del ADC se muestra en la siguiente figura:

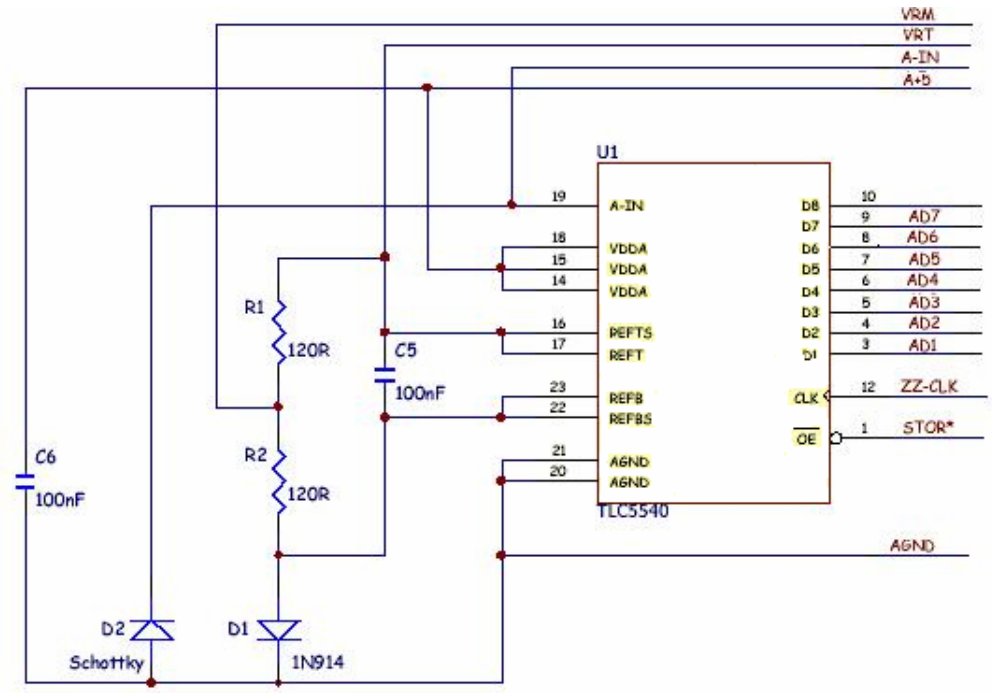


Figura 2 20

Las resistencias R1 y R2 son del mismo valor(120) y estas generan un Vm y permiten recortar el span para exactamente 2.0 V el voltaje Vm sirve para calibrar el ADC para la mitad del rango de entrada; el diodo D2 engancha la señal de entrada cuando esta se desplaza hacia la zona negativa; el diodo D1 mueve el rango de tierra a 0.4V para mejorar la linealidad entre la zona cerca de 00H y los capacitares desacoplan el tierra minimizando el ruido.†(sacado de APLICACIONES TIPICAS para el TLC 5540)

Entre las características del ADC TLC 5540†† están:

- Posee un bus de 8 bits de datos.
- La función de muestrear y enganchar (sample and hold) que es una característica que debe de poseer para el submuestreo.
- Un gran ancho de banda arriba de los 75MHz.
- Un span de 2 voltios el cual esta centrado en 1.62V.
- Es de tecnología de superficie.

Como es un ADC de muestrear y enganchar la exactitud de la conversión depende grandemente de la velocidad de muestreo, y se obtiene mayor exactitud a altas frecuencias pero que no sobrepasen los 40 MHz

2.4.4.6.2 ENTRADAS VERTICALES

La tarjeta adquisidora de datos captura datos posee 2 entradas verticales el **CANAL A** y **CANAL B** a través de 2 conectores tipo BNC que es compatible con puntas de prueba (10 : 1). Posee un interruptor en cada canal para acoplar señales AC/DC vía capacitor de 100 nF(ver figura). Se ha colocado un resistor de 1 MΩ para proveer a la punta de entrada con alta impedancia; se utiliza un resistor de 10 KΩ con un capacitor de 100 pF y un arreglo de 2 diodos colocados entre +5 V a -5V para proveer protección contra sobrevoltaje mayores de dichos límites.

Para el desacople de impedancia se utiliza un seguidor de voltaje creado por los transistores BC548 y el FET J309. y para ajustar el offset creado por la punta se regula a través del potenciómetro 200 Ω, ya que estos JFET poseen una alta impedancia de salida se utiliza otro seguidor de voltaje a través del MAX 477 el cual posee un ancho de banda de 300 MHz con una muy baja impedancia de salida.

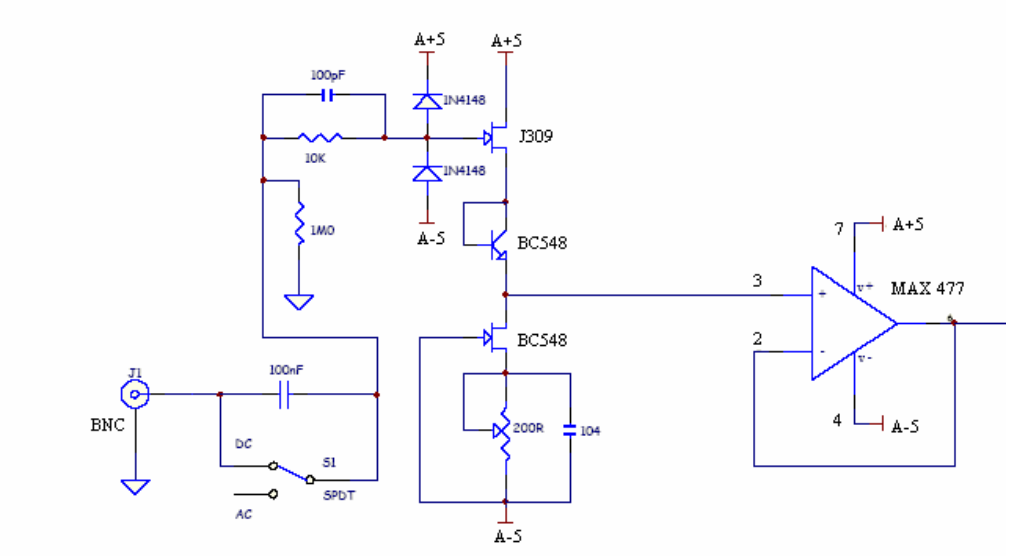


Figura 2 21

2.4.4.6.3 MULTIPLEXEO DE CANALES

La tarjeta adquisidora de datos posee 2 entradas vía BNC canal A y B. el max 4052 que es un multiplexor con switch análogo con un gran ancho de banda y excelentes características en altas frecuencia; este el encargado de seleccionar que canal es el que captura datos, la selección de la entrada del canal es vía PIC por el pin RA2: selección de canal A/B. A la vez que se selecciona un canal, un led indica que canal esta en uso. Si RA2 se encuentra en bajo el canal seleccionado es el A y si se encuentra en alto el canal seleccionado es el B, los switch sobrantes son colocados a tierra. Su alimentación requiere 5, -5 V que son suplidos por la fuente análoga.
El diagrama de conexión se muestra a continuación:

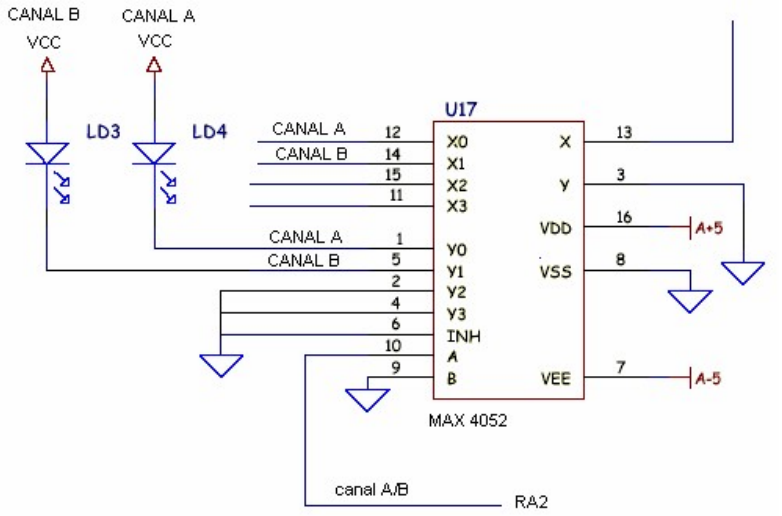


Figura 2 22

2.4.4.6.4 BUFFER DE ATENUACION

Se pueden obtener 4 rangos de ganancia las cuales son switchadas por el MUX 4052 que es diseccionado por los puertos RNG0 y RGN1 del PIC .
Cuando se selecciona 00 en las respectivas entradas se selecciona el buffer de ganancia de 4.58 generado por un amplificador no inversor y su configuración de resistencia, para pequeñas señales.
Para las otras condiciones restantes se atenúa en un factor de 1, 2, 5.2 generadas por una red de resistencias

2.4.4.6.5 BUFFER ADC.

Luego que ha sido seleccionado el rango la señal es amplificada en un factor de 1.66 y sirve para centrar la señal a este valor cuando la entradas es 0V, ya que debido que el span del ADC es 2 voltios centrados en 1.6V. Para ajustar el offset para este valor posee un ajuste de cero el cual esta construido por un transistor y un potenciómetro.

2.4.5 SOFTWARE Y ARQUITECTURA DE LA TARJETA

2.4.5.1 TRACE

Como se menciono anteriormente el comando T es el comando mas poderoso de la tarjeta ya que hace funcionar todo el sistema por completo

2.4.5.2 PROGRAMANDO EL TRACE

Después que todos los registros han sido programados el comando trace deberá ser ejecutado (T). los registros que entran en juego son los siguientes:

R8 modo de selección de Trace.
R11 retrazo de disparo (low)
R12 retrazo de disparo (high)
R13 factor de expansión de tiempo.

2.4.5.3 DESCRIPCION DE LOS MODOS DE CAPTURA

El registro más importante al ejecutar el comando (T) es el registro R8. los 4 bits menos significativos sirven para programar el modo de disparo que se desee seleccionar, los modos de programación son:

MODO 0 modo de trazado simple un solo canal
MODO 1 modo de trazado simple dual
MODO 2 modo con base de expacion de tiempo un solo canal
MODO 3 modo con base de expansión de tiempo dual

Los otros 4 bits más significativos deben de ser programados a cero.

ESTADOS DE TRACE

Todos los modos de TRACE tienen los mismos pasos o estados, los cuales son 2:

Estado 1 *Trigger habilitado* Después de que se ejecuta el comando T el sistema comienza a capturar datos, en la espera de un disparo, se mantiene en este estado hasta encontrar un disparo.

Estado 2 *Post trigger delay* Cuando se encuentra un disparo, se entra al estado 2 donde sigue capturando datos pero el contador de direcciones es reseteado para que los datos capturados válidos comienzan desde la dirección 0000 H, el tiempo que sigue capturando depende de el valor de los registro R11, R12, R13.

Base de expansión de tiempo:
Este registro es usado en el post trigger como se explico anteriormente y se utiliza en todos los modos de disparo, aunque su verdadera aplicación esta en los modos 2 y 3 de trace
esta expansión de tiempo sirve para reducir la velocidad de muestreo para poder medir frecuencias menores por ejemplo de 125 KHz hacia abajo. Ya que para frecuencias menores si se muestrea a la misma frecuencia se llenaría la memoria RAM con muy poca información de la señal
La captura de datos con expansión de tiempo consiste en una serie de ventanas donde se obtienen pocas capturas de muestras validas muestreadas a la misma velocidad nominal de 25 MHz, de las cuales se obtiene un promedio de cada ventana que puede ser considerada como una sola muestra.

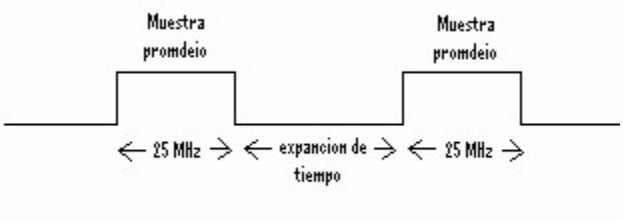


Figura 2 23

±En los modos 0 y 2 solo se obtiene capturas de un solo canal, la entrada seleccionada es referida como la el canal **primario.** Para los modos 1 y 3 soportan el modo de operación dual. En este caso el primario y el **secundario** canal son intercambiados repetidamente durante el comando T así que la adquision de datos contiene una serie alternativa de muestras provenientes del canal primario y secundario. El canal que será el primario o secundario es programado en el registro R14 donde los 4 MSB representan el canal secundario y los 4 LS bits representan el canal primario.

Todos los retrasos calculados son expresados en unidades de *PIC ciclos*, cada ciclo dura 200 ns (por ejemplo 4 PIC ciclos = 800 ns).

2.3.5.3.1 MODO 0

Este modo es el modo mas básico y solo soporta entrada por el canal primario que es programado en R14. después que se ejecuta el comando T, el estado 1 entra en uso y la captura de datos es realizada, el contador de direcciones de la RAM comienza a direccional en busca de un disparo.
Cuando un valido disparo es encontrado por el PLD este es transferido al PIC, lo que hace que este entre al estado 2 de post trigger, reinicie el contador de direcciones para capturar muestras validas. la duración de este retrazo **T(PTD)** es generado por los registro R13, R12 y R11. el retrazo es generado por la siguiente subrutina:

```
Loop_1
        MOVF        R13 , W
        CALL        delay
```

```
        DECFSZ      R12, F
        GOTO        loop_1
        DECFSZ      R11, F
        GOTO        loop_1
```

Esta subrutina produce un retardo de acuerdo a la siguiente formula:

T[PTD] = `((PTD + 1) * (5 + 3 * (TB+1))) + 2 * (PTD[hi] + 1) + U`

DONDE:

T[PTD]	post trigger delay (en el PIC).
PTD	16 bit valor de post trigger delay (**R11, R12** -> 0 a 65536).
PTD[hi]	8 bit high byte de post trigger delay (**R11** -> 0 a 255).
TB	8 bit time-base expansion (**R13** -> 1 a 256, 0 leido como 256).
U	± 3 PIC de error

Por ejemplo si programamos :
R11 = 01, R12 = 02, R13 = 05 entonces T[PTD] = 5961 *PIC ciclos.* = 1.1922 mS

El mínimo de post trigger es aproximadamente 2.6 uS ± 1.2 uS a un máximo de varios segundos.

Después que ha transcurrido este retrazo el contador de las direcciones es descargado a la PC para quedar luego en la espera del comando S (descargar muestras).

2.4.5.3.2 MODO 1

Este modo soporta un solo canal o dual. Durante el estado 1 solo el canal primario captura datos y el dual es deshabilitado, al encontrar un disparo y entrar al estado 2 la opción dual se activa intercambiándose el canal primario con el secundario y viceversa. Si ambos canales son programados con la misma atenuación y entrada la captura es efectivamente un solo canal.

La operación dual entre el primario y secundario canal se ejecuta N veces y el numero de muestras por canal que se desea capturar es P PIC ciclos, transcurrido P se intercambia los canales por otro P ciclos y así repetidamente N veces ciclos, en un tiempo total de T[PTD] PIC CICLOS.

La subrutina para este modo es la siguiente:

```
                NOP
Loop_5          NOP
                NOP
Loop_4          MOVF R13,W
                CALL        delay
```

```
SWAP        var_swap,F          ; intercambio entre canal A y B
MOVF        var_swap, W
MOVWF       PORTA
DECFSZ      R12, F
GOTO        Loop_5
DECFSZ      R11, F
GOTO        Loop_4
```

El valor de retardo es :

P = (9 + 3 (TB + 1))
N = ((1 + PTD) * P)
T[PTD] = N + PTD [hi] + 5 ± U

Donde:

T[PTD]	post trigger delay (en el PIC).
PTD	16 bit valor de post trigger delay (**R11, R12** -> 0 a 65536).
PTD[hi]	8 bit high byte de post trigger delay (**R11** -> 0 a 255).
TB	8 bit time-base expansion (**R13** -> 1 a 256, 0 leido como 256).
P	tiempo de cuantas muestras se desea por canal
N	numero de ciclos del intercambio de canales
U	± 3 PIC de error

El mínimo de muestras que se puede tomar por canal es de 7

2.4.5.3.3 MODO 2

Este modo es similar al modo 0, este captura dato solo del canal primario, la diferencia con el modo 0 es que el tiempo de captura de datos es expandido. La expansión de base de tiempo opera habilitando la captura de datos por 2 PIC ciclos cada P PIC ciclos donde P es la base de expansión de tiempo controlado por el R13 de acuerdo ala siguiente formula:

P = 7 + 3 * (TB + 1)

Donde:

P	periodo de expansión de tiempo (en PIC ciclos).
TB	8 bit tiempo-base expansión (**R13** -> 1 a 256, 0 leído como 256).

El mínimo periodo que se puede obtener es de 2.6 uS y un máximo de 155 uS. A cualquier velocidad de base de expansión de tiempo, son adquiridas 10 muestras en una ventana a una frecuencia de muestreo de 25 MHZ

2 PIC ciclo = 400 nS;
1 / 25 MHz = 40 nS;
400 nS / 40 nS = 10 muestras

Cuando se encuentra un disparo, entra al estado 2 el post trigger es ejecutado por la siguiente subrutina:

```
Loop_3
        BSF INDF, 3         ; Se habilita el reloj para capturar datos
        NOP                 ; 2 ciclos de reloj
        BCF INDF, 3         ; se deshabilita el reloj para capturar datos
        MOVF R13, W
        CALL    delay
        DECFSZ  R12, F
        GOTO    Loop_3
        DECFSZ  R11, F
        GOTO    Loop_3
```

La anterior subrutina produce la siguiente formula que define el periodo de ejecución:

$$T[PTD] = ((\ P * (PTD + 1))\ +\ 2 * (\ PTD[hi] + 1\ )\ + U$$

Donde:

T[PTD]	post trigger delay (en el PIC).
PTD	16 bit valor de post trigger delay (**R11, R12** -> 0 a 65536).
PTD[hi]	8 bit high byte de post trigger delay (**R11** -> 0 a 255).
TB	8 bit time-base expansion (**R13** -> 1 a 256, 0 leido como 256).
U	± 3 PIC de error

2.4.5.3.4 MODO 3

Este el segundo modo con base de expansión de tiempo y soporta simple y dual canal. En este caso la captura de datos es habilitado por 3 PIC ciclos para cada canal cada P PIC ciclos donde P es controlado por R13

$$P\ =\ 12\ +\ 3\ (\ TB\ +\ 1\ )$$

Donde:

P	periodo de expansión de tiempo (en PIC ciclos).
TB	8 bit tiempo-base expansión (**R13** -> 1 a 256, 0 leído como 256).

El post trigger se calcula de la misma forma que en el modo anterior. Estos dos modos son utilizados para muestrear frecuencias menores a 125 KHz

2.4.5.4 DESCARGA DE DATOS.

La descarga de los datos hacia la PC se hace a través del comando S, cada ves que se utiliza este comando una serie de datos son descargado hacia el puerto serial. El tamaño de cada descarga varia y depende del registro R9 y R10 que contienen el valor del contador de direcciones es decir el numero de muestras que se tomo; la velocidad a que se descargan las muestras es de 115200 baudio como se menciona en la sección de *interfase serial* por ejemplo para descargar un promedio de 10,000 muestras 1.066 segundos.

Cuando se envía el comando S desde la PC el PIC este lo recibe y lo devuelve dando la confirmación del mismo comando. El formato para descargar los datos es el siguientes:

S = > S < cr > aaaaaaaa……..aaa < cr >

Donde a = es la muestra análoga de 8 bits.

2.4.5.5 EL PLD

El generador de disparo y direccionador de la memoria es hecho por el PLD (Programador lógico de Alta Densidad); el PLD utilizado es 1016E de la compañía LATTICE[11], el cual contiene un 1 K de compuertas y 16 GLB de ahí su numero, en un paquete de 44 pines y de montaje superficial, consta de 33 pines I/O y que puede operar a una frecuencia máxima de 100 MHz.
Para su programación se utilizo las herramientas de diseño que proporciona la misma compañía ISPDESIGN que sirve para modelar, compilar, enrutar; e ISPVMSYSTEM para quemar el dispositivo.

2.4.5.5.1 ESTADOS DEL PLD

El PLD básicamente posee 4 estados básicos que definen el funcionamiento:

Estado 1 : Programación de registros.

Antes de un comando Trace (T) los registros deben de ser programado (R5, R6, R7) que deben de ser descargados del PIC al PLD, estos son descargados a través del comando ‘ > ‘ . los registros son transferidos de forma serial a través del pin RB0 del PIC y llamado igual en el PLD

Estado 2: Habilitación del disparo y guarda muestras validas.

Una ves programados los registró el PLD queda en la espera del comando T cuando este es realizado, entra al estado 2 en el cual se encuentra un disparo valido y este direccionar para almacenar muestras validas.

Estado 3: Descarga del contador de registros

Una ves terminado de almacenar se descarga del PLD al PIC el contador de la RAM en el valor en el que se quedo. Esta descarga se realiza al ejecutar el comando ‘ < ‘ y es almacenado en los registro R9 y R10 del PIC que son descargados hacia la PC.

Estado 4: Descarga de la muestras

Cuando se desea descargar las muestras de la RAM, el PLD solo entra en modo de conteo normal si espera de un disparo y se detiene hasta la dirección definida por los registro R9 y R10 del PIC descargados en el estado anterior.

2.4.5.5.2 GENERACION DEL TRIGGER

Ya que para graficar continuamente se nesecita encontrar un punto de referencia que debe de ser el mismo es decir que los datos deben de ser almacenados desde que se encuentra un disparo.
Para ello se debe configurar donde se desea disparar en un valor exacto. Pero como es muy difícil de encontrar un punto exacto se programa no una muestra en especial sino una banda de datos a través de la mascara (como se explico en la sección de PLD lógico).

El disparo es generado aplicándole una mascara(R6) a la entrada análoga de datos que posee el PLD que es de 8 bits y comparando el resultado con un byte llamado trigger lógico que también es programado(R5) cuando se de esta condición es que se encuentra en la banda de disparo. Solo queda definir si la señal se encuentra creciendo o decreciendo, se debe de encontrar un disparo valido cuando se cumplen las dos condiciones, que se encuentre en la banda de disparo y que la señal se encuentre creciendo. Como se mira en la figura, es un ejemplo para una señal senoidal:

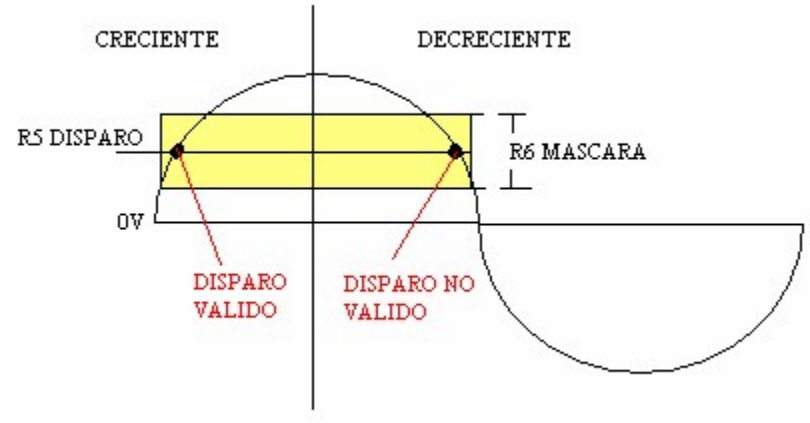


Figura 2 24

Cuando este disparo es encontrado el contador de las direcciones de la RAM es reseteado para comenzar a guardar muestras validas por un tiempo postrigger. Si no se encuentra disparo ya sea por que es una señal de DC o por un disparo mal programado, posee un sistema de auto disparo, que genera este disparo al quedar la RAM llena y son descargadas todas las muestras.

EL PIC

El programa del PIC fue creado en ASM y fue compilado en MPLAB herramienta que proporciona la compañía MICROCHIP fabricante de los PIC, el quemado del programa se realiza con el programa ICPROGRAM y la rutina de retrazo generada en la subrutina de transmisión y recepción serial con la PC son creadas en el programa PICDELAY.

En las secciones previas se explica la estructura, configuración y comandos del PIC.

CAPITULO III: LENGUAJES DE PROGRAMACIÓN.

Como se ha dicho anteriormente, una de las partes importantes de este proyecto será el software empleado para recibir y procesar los datos dentro de la computadora. En este sentido se comenzará estableciendo un panorama general acerca de los lenguajes de programación

3.1 Lenguajes De Programación.

Los lenguajes de programación son lenguajes especiales que ayudan al usuario a comunicarse con la computadora. Establecen una comunicación entre el humano que prefiere usar palabras, el sistema decimal, etc y la computadora, que trabaja solo con números binarios (0's y 1's).

Los lenguajes de programación pueden clasificarse como:

Lenguaje de máquina: que representa el más bajo nivel en la programación. Este lenguaje es en esencia números y códigos que solamente el microprocesador entiende. Este lenguaje es fácil de entender por la computadora, pero difícil para el usuario. El punto a su favor es que un programa escrito en lenguaje de máquina puede ejecutarse más rápido que los programas escritos en otros lenguajes, y que sus dimensiones son menores. Es el lenguaje original de la computadora el cual es generado por el "software", y no por el programador.

Lenguaje de bajo nivel: Es un lenguaje de programación bien cercano al lenguaje de máquina. Es difícil de entender por las personas y requiere que los programadores codifiquen las instrucciones con muchos detalles. Ejemplo: lenguaje ensamblador.

Lenguaje de alto nivel: Es un lenguaje que se asemeja más al lenguaje humano que a un lenguaje de máquina o ensamblador. Es más fácil escribir programas en este lenguaje, pero luego deben ser traducidos por compiladores o intérpretes para que la computadora los entienda.

En el caso de este proyecto de tesis se ha elegido como plataforma de desarrollo del software Visual Basic debido a la experiencia que se ha adquirido en los últimos años con respecto a este lenguaje de alto nivel y por su facilidad para desarrollar programas con ambiente gráfico y de ventanas.

3.2 Lenguajes De Alto Nivel.

Los lenguajes de alto nivel más comunes son:

BASIC (Beginners All-purpose Symbolic Instruction Code) Fue el lenguaje de programación interactivo más popular en la década de los 70. Es un lenguaje de propósito general. Desarrollado por John Kemeny y Thomas Kurtz en "Dartmouth College" en 1963. Existen numerosas versiones, algunas son compiladores y otras son intérpretes.

COBOL (Common Business Oriented Language) Es un lenguaje compilador diseñado para aplicaciones de negocios. Desarrollado en 1959 por el gobierno federal de los Estados Unidos y fabricantes de computadoras bajo el liderazgo de Grace Hopper. Es el más utilizado por los "mainframe". COBOL está estructurado en cuatro divisiones; las cuales son:

- División de identificación - identifica el programa.
- División ambiental - identifica a las computadoras fuente y objeto.
- División de datos - identifica las memorias "buffer", constantes y áreas de trabajo.
- División de procedimiento - describe el procesamiento (la lógica del programa).

PASCAL Este programa recibió su nombre en honor a Blaise Pascal. Fue desarrollado por el científico suizo Niklaus Wirth en 1970 y diseñado para enseñar técnicas de programación estructurada. Es fácil de aprender y de usar y no utiliza línea sino ";" (punto y coma). Existen versiones de compilador, como de intérprete. Estas varían según la versión.

FORTRAN (FORmula TRANslator) Es uno de los primeros lenguajes de alto nivel desarrollado en 1954 por John Backus y un grupo de programadores de IBM. Es un lenguaje compilador que se diseñó para expresar con facilidad las fórmulas matemáticas, resolver problemas científicos y de ingeniería.

ADA Es un lenguaje desarrollado como una norma del Departamento de Defensa de los Estados Unidos. Es un lenguaje basado en PASCAL, pero más amplio y específico. Fue diseñado tanto para aplicaciones comerciales como científicas. Es un lenguaje de multitareas que puede ser compilado por segmentos separados. Se llama ADA en honor de Augusta Ada Byron, condesa de Lovelace e hija del poeta inglés Lord Byron.

APL (A Programming Language) Este programa fue desarrollado por Kenneth Inverson a mediados de la década de 1960 para resolver problemas matemáticos. Este lenguaje se caracteriza por su brevedad y por su capacidad de generación de matrices y se utiliza en el desarrollo de modelos matemáticos.

PL/1 (Programming Language 1) Este programa fue desarrollado por IBM. Es un lenguaje de propósito general que incluye características de COBOL y de FORTRAN. Su principal utilidad es en los "mainframes".

RPG (Report Program Generator) Fue desarrollado por IBM en 1964 y diseñado para generar informes comerciales o de negocios.

Lenguaje C Fue desarrollado a principios de la década de los 70 en los laboratorios Bell por Brian Kernigham y Dennis Ritchie. Ellos necesitaban desarrollar un lenguaje que se pudiera integrar con UNIX, permitiendo a los usuarios hacer modificaciones y mejorías fácilmente. Fue derivado de otro lenguaje llamado BCPL.

Lenguaje C++: Fue desarrollado por Bjarne Stroustrup en los laboratorios Bell a principios de la década de los '80. C++ introduce la programación orientada al objeto en C. Es un lenguaje extremadamente poderoso y eficiente. C++ es un súper conjunto de C, para aprender C++

significa aprender todo acerca de C, luego aprender programación orientada al objeto y el uso de éstas con C++.

Visual BASIC Este programa fue creado por Microsoft. Es un programa moderno que da apoyo a las características y métodos orientados a objetos.

Programación orientada al objeto Las metas de la programación orientada al objeto es mejorar la productividad de los programadores haciendo más fácil de reutilizar y extender los programas y manejar sus complejidades. De esta forma, se reduce el costo de desarrollo y mantenimiento de los programas. En los lenguajes orientados al objeto los datos son considerados como objetos que a su vez pertenecen a alguna clase. A las operaciones que se definen sobre los objetos son llamados métodos. Ejemplo de programas orientados al objeto: Visual BASIC y C++.

El lenguaje de alto nivel que se ha elegido para este proyecto de tesis es Visual Basic, utilizando sobre todo la programación orientada a objetos que facilitará el desarrollo y reutilización del software. Estas cualidades de Visual Basic serán desarrolladas a continuación.

3.3 Lenguaje De Programación Visual Basic.

Visual Basic 6.0 es uno de los lenguajes de programación que más entusiasmo despiertan entre los programadores de PCs, tanto expertos como novatos. En el caso de los programadores expertos por la facilidad con la que pueden desarrollar aplicaciones complejas en muy poco tiempo (comparado con lo que cuesta programar en Visual C++, por ejemplo). En el caso de los programadores novatos por el hecho de ver de lo que son capaces a los pocos minutos de empezar su aprendizaje. El precio que hay que pagar por utilizar Visual Basic 6.0 es una menor velocidad o eficiencia en las aplicaciones.

Visual Basic fue desarrollado por Microsoft, su programación se basa en uno de los lenguajes de programación más conocidos a nivel mundial, *Basic*. Lo que permite al programa, de manera conveniente, desarrollar aplicaciones para Microsoft Windows uno de los sistemas operativos de mayor uso a nivel mundial. Proporciona la clase de capacidad y desempeño muy altas necesarias para desarrollar sistemas para las empresas. Hace al lenguaje sumamente adecuado para aplicaciones en Internet, y dotado con características que las personas realmente necesitan tales como: cadenas de caracteres, gráficos, componentes con interfaces graficas, manejo de errores, multimedia (audio, imágenes, animación y video), procesamiento de archivos, procesamiento de bases de datos, redes de cliente/servidor basadas en Internet, buscadores para la World Wide Web, mejoramiento de documentación de sitios en la World Wide Web con Visual Basic Script (VBScript), y componentes prediseñados. Hace al lenguaje extensible para que vendedores independientes puedan desarrollar componentes para un gran número de aplicaciones. Estos aspectos son precisamente lo que los negocios y organizaciones necesitan para cumplir con los requisitos del procesamiento de información de hoy en día
Visual Basic 6.0 es un lenguaje de programación visual, también llamado lenguaje de 4ª generación. Esto quiere decir que un gran número de tareas se realizan sin escribir código, simplemente con operaciones gráficas realizadas con el ratón sobre la pantalla.

3.3.1 Programación Orientada a Objetos.

La programación orientada a objetos (OOP: Object Oriented Programming) es una manera natural de pensar en el mundo y de escribir programas de cómputo. Al observar el mundo real que nos rodea, se nota que dondequiera que se vaya hay *objetos*. Gente, animales, plantas, automóviles, aviones, edificios, computadoras y demás. Los seres humanos pensamos en términos de objetos. Tenemos la maravillosa capacidad de la abstracción, que nos permite ver imágenes en la pantalla como objetos, tales como gente, aviones, árboles y montañas, en lugar de cómo puntos de colores (llamados píxeles). Podemos, si lo deseamos, pensar en términos de una playa, en lugar de granos de arena; de un bosque, en lugar de árboles; y de una casa, en lugar de ladrillos.

Se podría inclinar a dividir los objetos en dos categorías: objetos animados y objetos inanimados. Los objetos animados están "vivos" en algún sentido. Se mueven y hacen cosas. Los objetos inanimados, como las toallas, no hacen gran cosa. Sólo están ahí. Sin embargo, todos estos objetos tienen algo en común. Cuentan con atributos como tamaño, forma, color, peso, etc. Todos presentan comportamientos; por ejemplo, una pelota rueda, rebota, se infla y desinfla; un bebé llora, duerme, gatea, camina y parpadea; un automóvil acelera, frena, da vuelta; una toalla absorbe agua; etcétera.

Los seres humanos aprenden sobre los objetos estudiando sus atributos y observando sus comportamientos. Objetos diferentes pueden tener atributos similares y presentar comportamientos similares. Pueden hacerse comparaciones; por ejemplo, entre los bebés y los adultos, y entre los humanos y los chimpancés. Los automóviles, camiones, carritos y patinetas tienen mucho en común.

La OOP simula objetos reales con equivalentes de software. Aprovecha las relaciones de *clase* en las que los objetos de cierta *clase* (digamos una clase de vehículos) tienen las mismas características. Aprovecha las relaciones de *herencia* e incluso de herencia múltiple, en las que se derivan clases nuevas de objetos que heredan características de clases que ya existen y que, sin embargo, contienen características propias únicas. Un objeto de la clase convertible tiene las características de la clase automóvil, pero de manera adicional: el techo del convertible sube y baja.

La OOP nos da una forma más natural e intuitiva de ver el proceso de programación, por medio de la simulación de objetos reales, sus atributos y sus comportamientos. La OOP también simula la comunicación entre objetos. Al igual que las personas se envían mensajes entre ellas (por ejemplo, el sargento que ordena a la tropa prestar atención), los objetos también se comunican por medio de mensajes.

La OOP *encapsula* datos (atributos) y funciones (comportamientos) en paquetes llamados objetos; los datos y las funciones de un objeto están íntimamente ligados. Los objetos tienen la propiedad de ocultamiento de información. Esto significa que, aunque los objetos tal vez sepan cómo comunicarse entre ellos a través de interfaces bien definidas, normalmente no se les permite saber la manera en que se implementan otros objetos; los detalles de implementación

están ocultos en los objetos mismos. Con seguridad es posible manejar un automóvil sin que sea necesario saber cómo funcionan internamente el motor, la transmisión, y el sistema de escape. El ocultamiento de información es crucial para la buena ingeniería de software.

En la OOP la unidad de programación es la ***clase***, a partir de la cual los objetos son creados en algún momento. Los sustantivos de la especificación de un sistema son los que ayudan al programador a determinar el conjunto de clases a partir del cual se crearán objetos que funcionarán juntos, implementando el sistema. *Los planos son para las casas lo que las clases son para los objetos*. Se pueden construir muchas casas a partir de un plano y podemos crear muchos objetos a partir de una clase.

Cuando se empaqueta el software en clases, éstas se vuelven componentes que pueden reutilizarse en otros sistemas de software. Muchos programadores de hoy en día consideran que con la tecnología de objetos se construirá la mayor parte del software del futuro, combinando partes "estandarizadas e intercambiables" llamadas clases.

La OOP requiere tres tecnologías básicas: *encapsulación*, *herencia* y *polimorfismo*. Visual Basic no permite herencia de la misma forma en que C++ y Java lo hacen, pero si permite la encapsulación y el polimorfismo. No obstante, Visual Basic permite a los programadores realizar muchos de los beneficios de la herencia a través del uso de *interfaces* y una técnica llamada *delegación*.

3.3.2 Manejo De Los Puertos Con Visual Basic.

Existen dos formas tradicionales para la comunicación de la computadora con algún periférico: a través del *puerto paralelo* y a través del *puerto serial*.

Con respecto a la comunicación a través del puerto paralelo, Visual Basic carece de comandos o componentes que le permitan escribir datos o leer datos del puerto. Sin embargo, esto se puede solucionar haciendo uso de algún archivo DLL (Dynamic Linked Library).

Como su nombre lo indica, los DLLs pueden permitir a Visual Basic enlazar (un paso antes de compilar) código (librerías que han sido desarrolladas en otros lenguajes como: Delphi, Borland C++ o Microsoft's Visual C++) en tiempo de ejecución (dinámicamente).

La comunicación por puerto paralelo no es una opción a utilizar en nuestro proyecto; por tal razón, no profundizaremos más en los detalles de este tema. Si el lector está interesado en la comunicación a través del puerto paralelo, puede consultar la bibliografía para encontrar información acerca del libro llamado "Parallel Port Complete". Este libro es una buena fuente de información para aquellos interesados en desarrollar programas en Visual Basic que se comuniquen con el puerto paralelo. O bien pueden consultar algunas de los sitios en Internet sugeridos en la bibliografía.

Visual Basic 2.0 y versiones superiores han sido equipadas con capacidades para la comunicación serial. El componente de software encargado de estas funciones en Visual Basic se llama MSCOMM. La versión 2.0 y 3.0 estaba equipada con MSCOM.VBX y la versión 4.0 incluía tanto a MSCOM16.OCX y MSCOMM32.OCX. La versión 5.0 y la más reciente la 6.0 son de 32 bit, por esta razón solo incluyen MSCOMM32.OCX. Las versiones VBX y OCX tienen ligeras diferencias en sus características.

MSCOMM encapsula las API (Application Programming Interface) para comunicación de Windows, lo que permite funciones de comunicación tan simples como lectura y escritura de propiedades. Las API permiten notificación por medio de eventos que ocurren en la comunicación.

MSCOMM está oculto en tiempo de ejecución. Cuando se está ejecutando, MSCOMM no tiene interfase visual, todas las entradas y salidas deben realizarse por medio de código.

El control MSCOMM proporciona dos formas diferentes de tratamiento de las comunicaciones:

- Las comunicaciones controladas por eventos son un método muy poderoso para el tratamiento de interacciones con el puerto serie. En muchas situaciones resulta útil que se notifique cuándo tiene lugar un evento; por ejemplo, cuándo llega un carácter o cuándo se produce un cambio en las líneas de Detección de portadora (CD) o solicitud de envío (RTS). En tales casos se utiliza el evento *OnComm* del control MSCOMM para interceptar y tratar estos eventos de comunicaciones. El evento OnComm también detecta y trata los errores en las comunicaciones. En la propiedad CommEvent se puede ver una lista completa de todos los eventos y errores posibles en las comunicaciones.

- También se puede sondear los eventos y errores si se comprueba el valor de la propiedad CommEvent después de cada función crítica de un programa. Esta alternativa es preferible si la aplicación es pequeña y autónoma. Por ejemplo, si se está escribiendo un marcador telefónico sencillo, no tiene sentido generar un evento después de recibir cada carácter, ya que los únicos caracteres que piensa recibir son las respuestas de aceptación que envía el módem.

Cada uno de los controles MSCOMM que se usen corresponde a un puerto serie. Si se necesita tener acceso a más de un puerto serie en una aplicación, se debe usar más de un control MSCOMM. La dirección del puerto y la dirección de la interrupción pueden cambiarse desde el Panel de control de Windows.

Aunque el control MSCOMM tiene muchas propiedades importantes, hay algunas con las que un usuario debe familiarizarse primero.

Propiedades	**Descripción**
CommPort	Establece y devuelve el número del puerto de comunicaciones.
Settings	Establece y devuelve la velocidad en baudios, paridad, bits de datos y bits de parada en forma de cadena.
PortOpen	Establece y devuelve el estado de un puerto de comunicaciones. También abre y cierra un puerto.
Input	Devuelve y quita caracteres del búfer de recepción.
Output	Escribe una cadena de caracteres en el búfer de transmisión.

CAPITULO IV: SOFTWARE DEL OSCILOSCOPIO DIGITAL.

De forma general el programa deberá realizar las siguientes funciones:

- Cuando el programa arranque debe presentar una **interfaz gráfica**.
- Dar una señal al hardware para **iniciar el proceso de muestreo**.
- Esperar que el hardware indique el **fin de conversión** (EOC).
- **Leer buffers de memoria** del hardware.
- **Presentar señal en pantalla**.

Cada uno de los puntos mencionados anteriormente serán desarrollados con más detalles a continuación.

4.1 Interfaz Gráfica.

La figura 4.1 muestra la interfaz gráfica en modo de diseño.

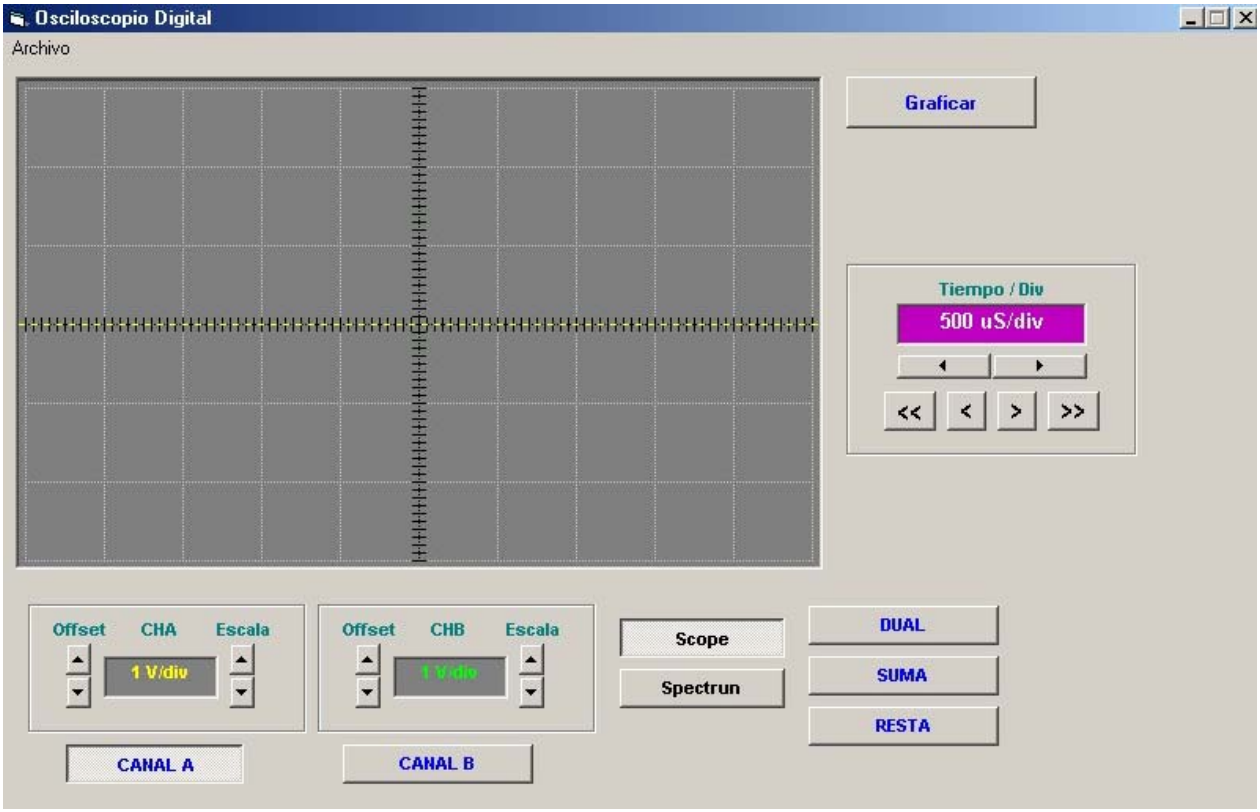


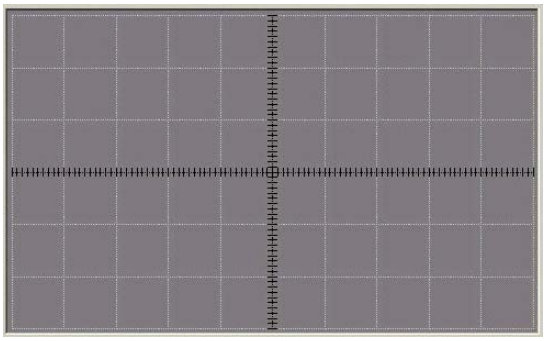
Figura 4 1 Interfaz gráfica del osciloscopio digital

El formulario en la figura 4.1 muestra, entre otras cosas, un menú que nos permite guardar o abrir archivos de señales muestreadas. Por el momento es lo único que se ha incorporado a este menú. A medida el software vaya evolucionando se incorporarán más funciones.

El formulario incluye un componente de Visual Basic llamado PictureBox, en cuyo fondo se ha colocado una imagen cuadriculada y graduada para graficar las señales muestreadas por la tarjeta adquisidora de datos. Un botón de comando "**Obtener Datos**" permitirá ordenarle a la tarjeta adquisidora cuando debe **iniciar el proceso de muestreo**. También posee 3 recuadros (frame); dos de ellos sirven para ajustar el offset de una señal, ajustar la escala, así como visualizar los voltios por división en una caja de texto. El tercer recuadro sirve para el tiempo por división que contiene controles que nos permiten elegir un tiempo por división adecuado para una determinada señal. Al lado de los dos recuadros, que controlan los parámetros de cada canal, aparecen dos botones de opción para elegir AC o DC.

Dos botones de opción (option bottons) permiten al programa seleccionar el canal a presentar en pantalla (Canal A o Canal B). Las propiedades establecidas en tiempo de diseño para todos estos controles son presentados en la siguientes tabla. Para la descripción de los controles se han empleado nombres en ingles en lugar de su equivalente en español. Estos nombres no deberían resultar extraños a una persona familiarizada con la programación en Visual Basic.

4.1.1 Tablas De Propiedades De Los Objetos.

Objeto	**Propiedad**	**Valor**
Form	Name	frmScope
	Caption	Osciloscopio digital
	BackColor	&H8000000B&
 **Figura 4 2 PictureBox**		
PictureBox	Name	picPantalla
	AutoRedraw	True
	BackColor	&H00808080&
	Picture	Agregar imagen llamada ***Cuadricula.jpg***
	ScaleHeight	309
	ScaleMode	3 – Píxel
	ScaleWidth	509
Objeto	**Propiedad**	**Valor**
 **Figura 4 3 Componente MSComm para comunicación serial.**		

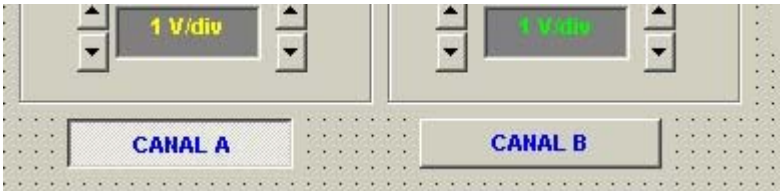
MSComm	Name	MScomm1
	CommPort	2
	InBufferSize	8193 (8k de memoria)
	OutBufferSize	512
	Settings	115200,n,8,1

Figura 4 4 Recuadro para establecer propiedades del canal A.

Frame	Name	frmCHA
Label	Name	lblOffsetA
	Caption	Offset
	ForeColor	&H00808000&
Label	Name	lblCHA
	Caption	CHA
	ForeColor	&H00808000&
Label	Name	lblEscalaA
	Caption	Escala
	ForeColor	&H00808000&
UpDown	Name	udOffset
	Increment	5
	Index	0
	Max	128
	Min	428
	Orientation	0 - cc2OrientationVertical
	Value	280
UpDown	Name	udEscala
	Increment	1
	Index	0
	Max	11
	Min	0
	Orientation	0 - cc2OrientationVertical
	Value	6
TextBox	Name	txtEscala
	Font	Arial, Negrita, 8
	BackColor	&H00808080&
	ForeColor	&H0000FFFF&
	Index	0
	Text	1 V/div

Objeto	Propiedad	Valor
 **Figura 4 5 Recuadro para establecer propiedades del canal B.**		
Frame	Name	frmCHB
Label	Name	lblOffsetB
	Caption	Offset
	ForeColor	&H00808000&
Label	Name	lblCHB
	Caption	CHB
	ForeColor	&H00808000&
Label	Name	lblEscalaB
	Caption	Escala
	ForeColor	&H00808000&
UpDown	Name	udOffset
	Increment	5
	Index	1
	Max	128
	Min	428
	Orientation	0 - cc2OrientationVertical
	Value	280
UpDown	Name	udEscala
	Increment	1
	Index	1
	Max	11
	Min	0
	Orientation	0 - cc2OrientationVertical
	Value	6
TextBox	Name	txtEscala
	Font	Arial, Negrita, 8
	BackColor	&H00808080&
	ForeColor	&H0000FF00&
	Index	1
	Text	1 V/div

Objeto	Propiedad	Valor
 **Figura 4 6 Recuadro para establecer propiedades del Tiempo/División.**		
Frame	Name	frmTimeDiv
Label	Name	lblTimeDiv
	Caption	Tiempo / Div
	ForeColor	&H00808000&
TextBox	Name	txtTimeDiv
	Font	Arial, Negrita, 8
	BackColor	&H00C000C0&
	ForeColor	&H0000FF00&
	Text	500 uS/div
UpDown	Name	udTimeDiv
	Increment	1
	Max	29
	Min	0
	Orientation	0 - cc2OrientationHorizontal
	Value	17
 **Figura 4 7 Botones de comandos para desplazar la señal graficada a lo largo del eje horizontal**		
CommandButton	Name	cmdShift
	Caption	<<
	Font	Arial Narrow, Negrita, 12
	Index	0
CommandButton	Name	cmdShift
	Caption	<
	Font	Arial Narrow, Negrita, 12
	Index	1
CommandButton	Name	cmdShift
	Caption	>
	Font	Arial Narrow, Negrita, 12
	Index	2
CommandButton	Name	cmdShift
	Caption	>>
	Font	Arial Narrow, Negrita, 12
	Index	3

Objeto	Propiedad	Valor
 **Figura 4 8 Botones de Opción para elegir Canal A o Canal B**		
OptionButton	Name	optCanalA
	Caption	CANAL A
	Font	Arial Narrow, Negrita8
	ForeColor	&H00FF0000&
	Value	True
OptionButton	Name	optCanalB
	Caption	CANAL B
	Font	Arial Narrow, Negrita8
	ForeColor	&H00FF0000&
	Value	True

4.2 Software y Flujogramas.

El software se divide en dos partes: Clases y Eventos. Las clases son componentes de software que nos permiten desarrollar objetos, en cambio los eventos son subrutinas que se ejecutan cuando el usuario realiza una acción; por ejemplo, hacer clic, presionar o soltar una tecla, etc.

4.2.1 Clases.

Se han definido 3 clases para el software del osciloscopio:

- La clase **VoltDiv**.
- La clase **TimeDiv**.
- Y la clase **Grafica**.

➢ La clase **VoltDiv**

Esta clase está formada por dos matrices, que contienen los voltios por división que el Osciloscopio digital podrá manejar. Una de las matrices contiene los valores en formato ***texto*** (string) y la otra el ***valor numérico***. Estos valores son: "10mV/div", "20mV/div", "50mV/div", "0.1V/div", "0.2V/div", "0.5V/div", "1V/div", "2V/div", "5V/div", "10V/div", "20V/div" y "50V/div". También ha sido dotada con una propiedad denominada **índice**. La propiedad índice puede apuntar a ambas matrices (texto y valor numérico). El valor de este índice puede cambiarse por medio del botón UpDown ubicado en los frames (recuadros) que controlan cada canal. Al cambiar este índice se cambia inmediatamente la caja de texto para mostrar el valor correspondiente a los voltios por división que se ha seleccionado. Contiene también dos funciones públicas; una de ellas devuelve el valor numérico de la escala que se ha elegido y la otra devuelve la relación de escala entre la escala actual y la escala original a la que una señal fue muestreada. Esto permitirá aumentar o reducir una señal sobre el eje vertical.

- La clase **TimeDive**

Está clase es muy parecida a la clase VoltDiv. Incluye dos matrices apuntadas por un único índice. Una almacena el tiempo por división en formato texto y la otra en valor numérico. Los valores de tiempo incluidos para el software son: "1 nS/div", "2 nS/div", "5 nS/div", "10 nS/div", "20 nS/div" "50 nS/div", "100 nS/div", "200 nS/div", "500 nS/div", "1 uS/div", "2 uS/div", "5 uS/div", "10 uS/div", "20 uS/div", "50 uS/div", "100 uS/div", "200 uS/div", "500 uS/div", "1 mS/div", "2 mS/div", "5 mS/div", "10 mS/div", "20 mS/div", "50 mS/div", "100 mS/div", "200 mS/div", "500 mS/div", "1 S/div", "2 S/div" y "5 S/div".

Posee una única propiedad llamada índice con la cual se puede apuntar a ambas matrices (texto y valor numérico). El valor de este índice puede cambiarse mediante el botón UpDown que controla el tiempo por división de cada canal. Al cambiar este índice se actualiza inmediatamente la caja de texto para mostrar el valor correspondiente al tiempo por división que se ha elegido. Contiene también dos funciones públicas; una de ellas devuelve el valor numérico de la escala que se ha elegido y la otra devuelve la relación de escala entre la escala actual y la escala original a la que una señal fue muestreada. Esto permitirá aumentar o reducir una señal determinada sobre el eje horizontal.

- La clase **Grafica**:

Para desarrollar esta clase se ha empleado una técnica llamada ***composición***. Esta técnica consiste en crear nuevas clases agregando objetos de clases existentes como miembros. Así, la clase ***Grafica*** incorpora a las clases ***VoltDiv*** y ***TimeDiv***. En otras palabras, esta clase hereda todas las características y comportamiento de las clases anteriores.

La función principal de la clase Grafica es dibujar la señal que se este monitoreando en un determinado canal. Para tal tarea, posee una función que recibe una matriz de datos desde el puerto serial. Cada dato es procesado y ubicado en un lugar adecuado de la pantalla para poder reconstruir la señal monitoreada. Posee propiedades que le permiten aplicar un offset horizontal o vertical para poder desplazar la señal a cualquier lugar de la pantalla.

Debido a que incorpora las clases VoltDiv y TimeDiv, la clase Grafica puede controlar los voltios por división y el tiempo por división; es decir, puede reducir o aumentar una señal tanto en el eje horizontal como en el eje vertical.

Una vez se han desarrollado las clases necesarias se pueden crear los objetos de la clase Grafica. En el programa se han creado dos objetos de la clase grafica:

- El objeto **Canal A**
- Y el objeto **Canal B**

Tanto el objeto Canal A como Canal B tienen la capacidad de recibir una cadena de caracteres, graficar dicha cadena en la pantalla, y reducir o aumentar la señal en pantalla. Es importante destacar que sería posible crear más objetos para poder tener más canales; sin embargo, para este prototipo solo se incorporarán dos canales.

4.2.1 Eventos.

Ya se han descrito las clases necesarias para el software del osciloscopio. Ahora se describirán los eventos que tendrán lugar cuando se ejecute el software.

- Evento **Carga del formulario (Form_Load)**:

Cuando el formulario se ejecuta por primera vez, tendrá lugar el evento "Carga del formulario" (Form Load). Este evento debe establecer ciertas propiedades iniciales, como por ejemplo: establecer el puerto de comunicación (COM1 o COM2), la velocidad de la comunicación, el canal por defecto, etc. El flujograma se muestra en la figura 5.10

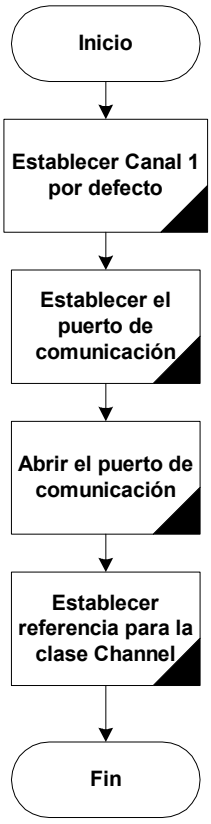


Figura 4 9 Flujograma del evento FormLoad

- Evento **"Obtener Datos"**

Cuando el usuario haga clic en el botón nombrado "*Obtener Datos*" se depositará un número en formato ASCII en la propiedad Output de MSComm1. Este número indicará a la tarjeta adquisidora de datos la velocidad a la que debe muestrear una señal. La Clase TimeDiv mantiene almacenado este número en la propiedad Indice, de tal manera que únicamente basta con transferir esta propiedad a la propiedad Output de MSComm1 para que la tarjeta inicie su proceso. Debido a que este evento es relativamente sencillo no se presenta flujograma.

- Evento **"OnComm"**

Una vez el usuario haga clic en el botón "Obtener Datos" la tarjeta adquisidora de datos muestreará la señal y la almacenará en su memoria. Cuando la memoria esté llena será transferida

automáticamente a los buffers de entrada del puerto serial. Los datos llegan al puerto serial en formato ASCII y encapsulados en un tipo de variable conocida como String o "cadena de caracteres". También, debido a la forma en que se realiza el muestreo, por cada punto de una señal se han tomado 10 muestras a alta frecuencia, lo que implica realizar un promedio de esos 10 datos para obtener un punto de muestra.

Los buffers de entrada del puerto serial serán descargados a un tipo de variable especial. Esta variable contendrá los voltios por división, el tiempo por división, el voltaje máximo y las muestras en formato ASCII. Este tipo de variable es conocida en Visual Basic como ***"Variable Creada por el Usuario"***. El objetivo que se persigue con este tipo de variables es el facilitar el almacenamiento de los datos en archivos de acceso aleatorio. Por la naturaleza de los archivos de acceso aleatorio, resulta sumamente fácil guardar o leer registro de longitud fija. En nuestro caso, el software maneja una variable llamada ***mArchivo*** del tipo "creada por el usuario" con las siguientes propiedades:

Propiedades	**Descripción**
sSamples	Almacena las muestras obtenidas de una señal en formato String (cadena de caracteres). Su longitud es de 8K bytes
Vmax	Variable de tipo entera que almacena el máximo valor de las 8k muestras obtenidas.
VoltDiv	Variable de tipo entera que almacena una constante que representa los voltios por división al que fueron tomadas las muestras.
TimeDiv	Variable de tipo entera que almacena una constante que representa el tiempo por división al que fueron tomadas las muestras.

La nomenclatura de las líneas de software para acceder a estas propiedades es sencilla. Por ejemplo si se quieren almacenar las 8k muestras recibidas desde el puerto serial, la nomenclatura sería la siguiente.

```
mArchivo.sSamples = MSComm1.Input
```

La línea anterior especifica que los Buffers de entrada del puerto serial serán almacenados en la variable sSampes, que pertenece a un de tipo especial de variable "creada por el usuario" llamada mArchivo.

El flujograma de este evento se presenta en la figura 4.10

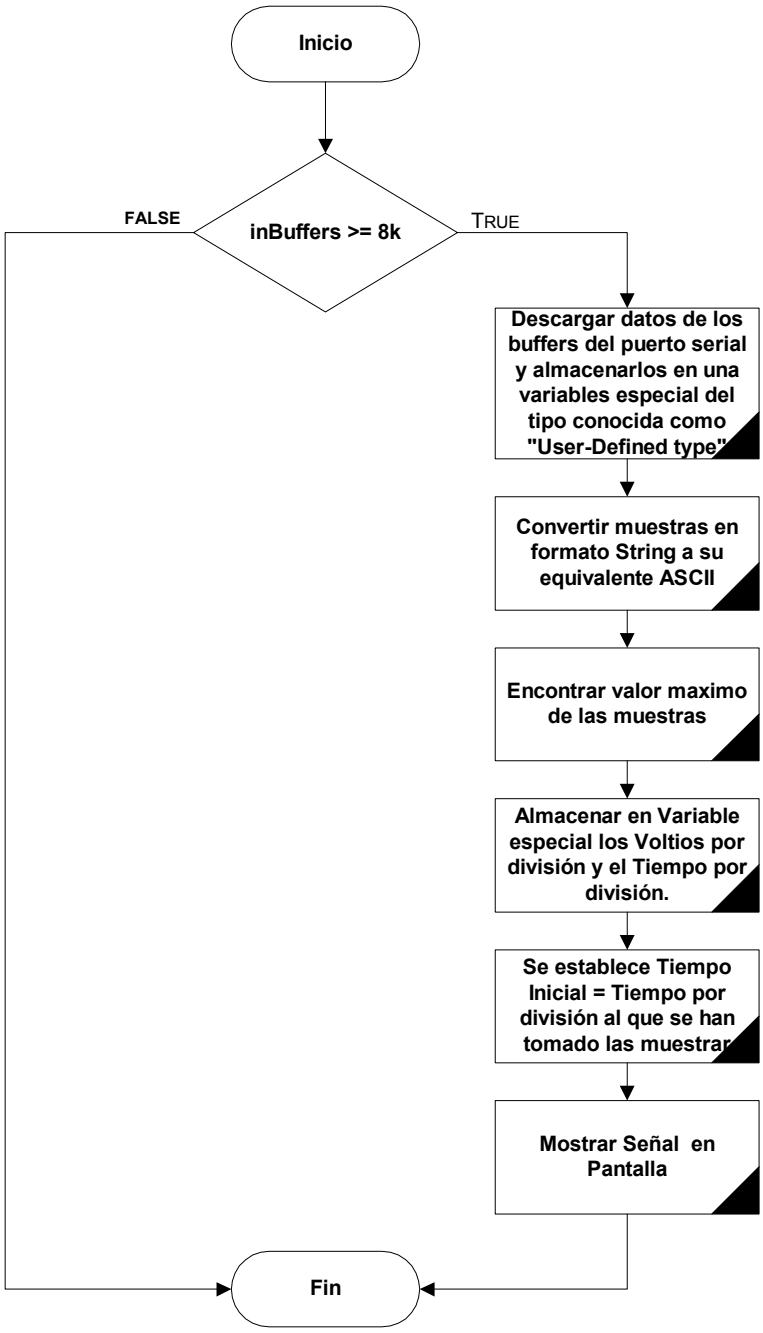


Figura 4 10 Evento OnComm

.

CAPITULO V: SOFTWARE ANALIZADOR DE ESPECTROS.

5.1 Transformada Rapida De Fourier

Básicamente la Transformada de Fourier se encarga de convertir una señal del dominio del tiempo, al dominio de la frecuencia, de donde se puede realizar su antitransformada y volver al dominio temporal.

Un ejemplo de representación en frecuencia, puede ser el ecualizador de un equipo de música. Las barritas que suben y bajan, indican las diferentes componentes frecuenciales de la señal sonora que se está escuchando. Esta tarea la realiza un integrado aplicando la transformada de Fourier de la forma más rápida posible (FFT, o Fast Fourier Transform).

El trabajo con la señal en frecuencia, no solo sirve como información, sino que se puede modificar, de forma que es ampliamente utilizada en filtros, procesamiento de imágenes y de sonido, comunicaciones (modulaciones, líneas de transmisión, etc.) y otro tipo de aplicaciones más curiosas: estadística, detección de fluctuaciones en los precios, análisis sismográfico, etc.

La transformada rápida de Fourier es simplemente un algoritmo rápido para la evaluación numérica de integrales de Fourier desarrollado en los laboratorios de IBM, y su importancia radica en la rapidez de cálculo conseguida.

Evidentemente hacemos uso del la FFT en el programa para obtener rápidamente el espectro de la señal a partir de la señal temporal de entrada, aunque se podría haber hecho a partir de la integral discreta de Fourier, pero consumiría mucho más tiempo de cálculo.

La diferencia de velocidad de cálculo entre la tradicional transformada discreta y la FFT aumenta según aumenta el número de muestras a analizar, según se puede apreciar en la figura 5.1, ya que mientras una aumenta el número de operaciones necesarias para la resolución de forma exponencial, la otra lo hace de forma prácticamente lineal.

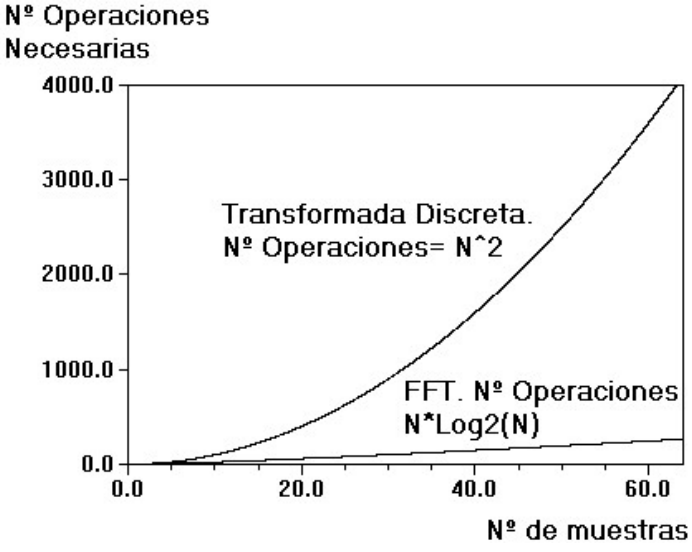


Figura 5 1

El codigo de la FFT que estamos empleando fue desarrollado por Murphy McCauley **[1].** Desde el sitio web de este autor se puede descargar el código ya sea para Visual Basic o el archivo DLL que es mas rápido. En nuestro caso estamos usando el archivo llamado FFT.DLL que esta desarrollado en lenguaje C y por ser una librería dinámica puede ser usado por nuestro programa principal. El uso del archivo es gratuito, aunque el autor exige se hagan los créditos correspondientes.

5.2 Interfaz Gráfica.

Para la interfaz grafica del espectro de una señal se empleara la misma cuadricula que se emplea para la representación en el tiempo. La figura 5.2 muestra una señal cuadrada en el dominio en el tiempo, mientras que la figura 6.3 muestra su correspondiente espectro. Como se puede ver en estas dos figuras se ha hecho uso de la misma cuadricula.

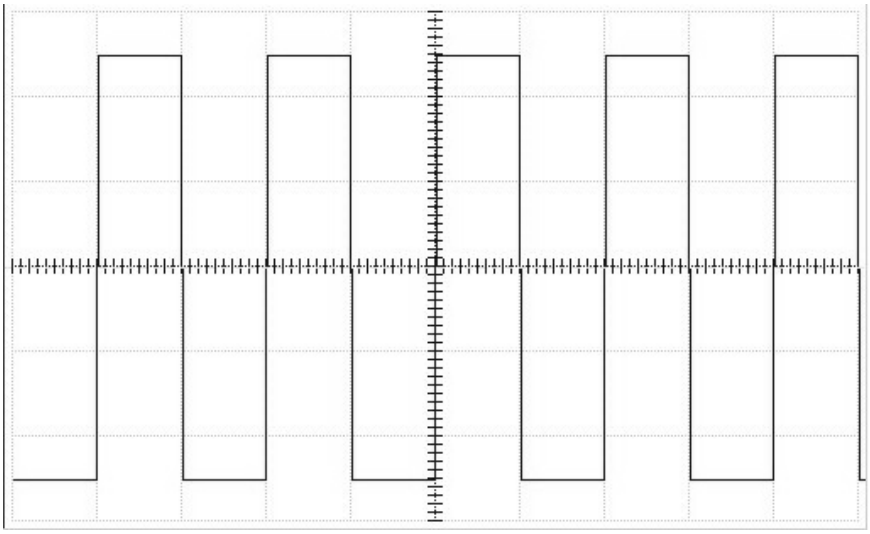


Figura 5 2 Señal cuadrada en el dominio del tiempo

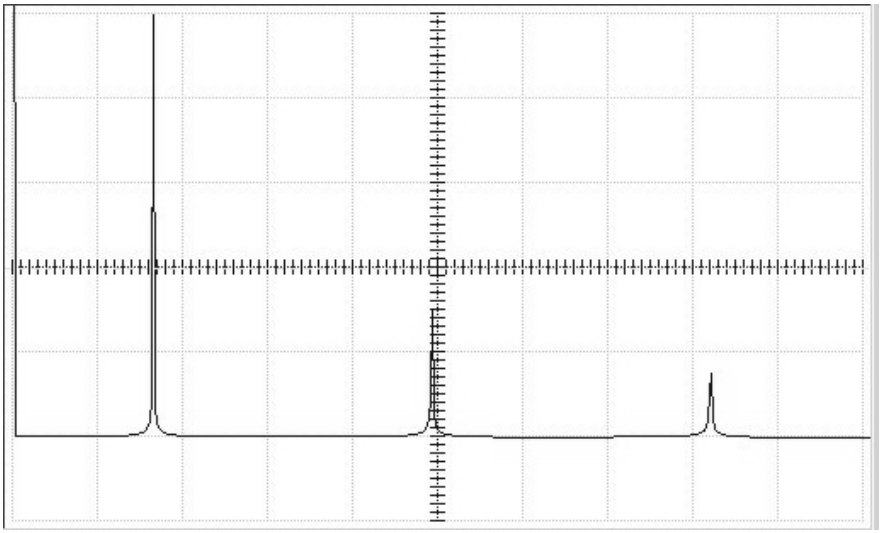


Figura 5 3 Espectro en frecuencia de una onda cuadrada

5.3 Algoritmo de la Transformada Rápida de Fourier (FFT)

El código completo de este algoritmo se encuentra en los anexos. Los fundamentos teóricos que se aplican en el algoritmo se describen a continuación:
El algoritmo FFT lo único que busca es resolver de la manera más eficiente posible la siguiente expresión **[2]:**

$$X[n] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-jk\Omega n}$$

Donde $\Omega = 2\pi / N$
La evaluación directa de estas sumatoria implica **N^2** multiplicaciones. Pero haciendo una serie de reordenamientos, se consigue reducir el número de operaciones de la FFT a **N*Log2(N)**.

Para lograr esto, primero se deben separar las muestras pares y las impares:

$$X[n] = \frac{1}{N}\left(\sum_{n=0}^{N/2-1} x[2n]e^{-2jk\Omega n} + \sum_{n=0}^{N/2-1} x[2n+1]e^{-(2n+1)jk\Omega} \right)$$

A continuación sacamos fuera de la sumatoria impar el exponencial $e^{-jk\Omega}$:

$$X[n] = \frac{1}{N}\left(\sum_{n=0}^{N/2-1} x[2n]e^{-2jk\Omega n} + e^{-jk\Omega} + \sum_{n=0}^{N/2-1} x[2n+1]e^{-2njk\Omega} \right)$$

De la expresión anterior podemos observar que si aplicamos:

Y=FFT(x[0], x[2], x[4], ..., x[N-2])

y

Z=FFT(x[1], x[3], x[5], ..., x[N-1])

Entonces obtenemos:

$$x[k] = \frac{1}{2}\left(Y[k] + e^{-jk\Omega} Z[k]\right) \quad \text{Si } 0 \le k \lhd N/2$$

$$x[k] = \frac{1}{2}\left(Y[k-N/2] - e^{-jk\Omega} Z[k-N/2]\right) \quad \text{Si } N/2 \le k \lhd N$$

Así el problema se ha reducido al cálculo de dos FFTs de tamaño N/2 y realizar N multiplicaciones complejas. Es conveniente observar que el bit menos significativo de k determina siempre si k es par o impar. Repitiendo este proceso reiteradamente, se consigue extraer la transformada de x..

La forma en que todo lo anterior se traslada a código se explica a continuación:
Inicialmente se envía una cantidad de datos a dicho algoritmo. Mediante una subrutina se determina que el número de datos recibidos sea potencia de 2, pues esta es la base de la FFT. Si la cantidad de datos recibidos no es potencia de 2 se agrega una cantidad de ceros para que lo sea.

La siguiente subrutina que se ejecuta dentro del algoritmo, es conocida como "Inversión de bits", el objetivo de esta subrutina es dividir la matriz de datos en dos partes, la primera parte contiene los datos pares y la siguiente contiene los datos impares. El que una muestra sea par o impar es determinado por el índice que ocupa la muestra dentro de la matriz de datos.

La última subrutina y la de mayor grado de dificultad se encarga de realizar las operaciones complejas aplicando el método de la Mariposa. No profundizaremos en este tema pero se puden consultar las referencias **[3]**

[1] http://www.fullspectrum.com/deeth/

[2] http://www.arrakis.es/~ppriego/fourier/explica.htm

[3] http://www.spd.eee.strath.ac.uk/~interact/fourier/fft/fftbutfy.html

CAPITULO VI: TÉCNICA DEL SUB-MUESTREO.

6.1 Fundamentos Del Sub-Muestreo.

Nuevos ADCs de banda ancha, de baja distorsión resultan útiles para aplicar una técnica conocida como submuestreo. Para poder entender estas aplicaciones es necesario revisar los fundamentos del muestreo.

El concepto de muestrear un señal de amplitud determinada en períodos discretos de tiempo es ilustrado en la figura 6.1. La señal análoga continua debe ser muestreada en intervalos discretos de tiempo, ts, que deben ser elegidos cuidadosamente para evitar la perdida de información. Queda claro entonces que a mayor cantidad de muestras (velocidad de muestreo más rápida), la representación digital de la señal análoga será más exacta, pero si se comienza a disminuir la cantidad de muestras (reducción de la velocidad de muestreo) se alcanza un punto en el cual se pierde información crítica de la señal. Fue todo esto lo que llevo al establecimiento del teorema de información de Shannon y el criterio de Nyquist.

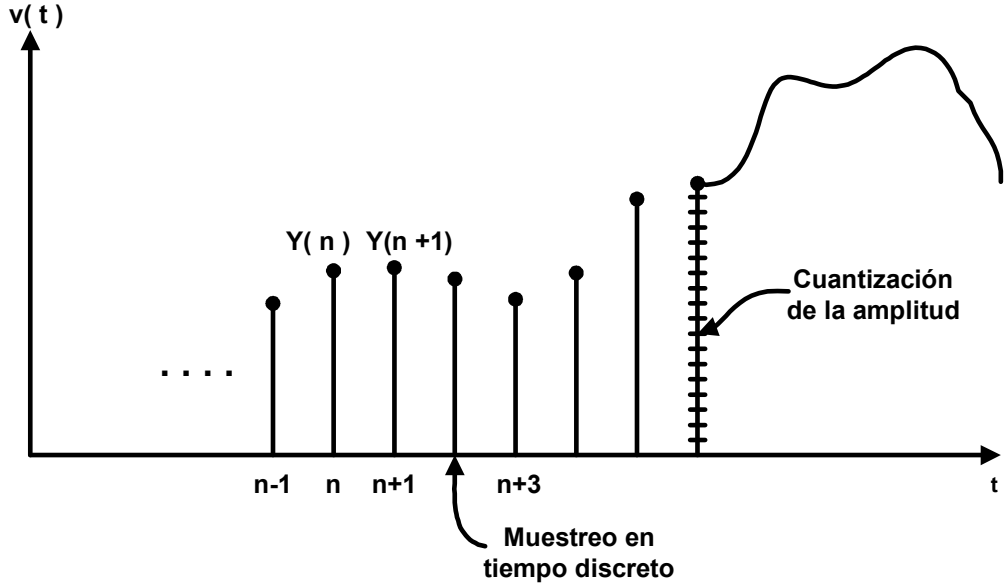


Figura 6 1 Muestreo de una señal análoga.

La mayoría de libros de texto definen el teorema de Nyquist de la siguiente manera: "Una señal debe ser muestreada a una velocidad mayor al doble de su frecuencia máxima para asegurarse su posterior reconstrucción. Se asume de forma general que la señal tiene componentes de frecuencia que van desde dc (0 Hz) hasta una frecuencia máxima, fa. Así, el criterio de Nyquist requiere una velocidad de muestreo fs > 2fa para evitar el traslape de componentes (aliasing). Sin embargo, para señales que no van desde dc (0 Hz) hasta fa, la velocidad de muestreo mínima requerida es una función del ancho de banda de la señal, así como de su posición en el espectro de frecuencia.

Entonces, una definición más completa del los teoremas de muestreo se presenta a continuación

Shannon:

- *Una señal análoga con una ancho de banda fa debe ser muestreada a una velocidad de fs > 2fa para evitar la perdida de información.*
- *El ancho de banda de una señal puede extenderse desde DC hasta fa (muestreo de bandabase) o desde f1 hasta f2, donde fa = f2 – f1 (Submuestreo: undersampling, bandpass sampling, harmonic sampling, o Super-Nyquist)*

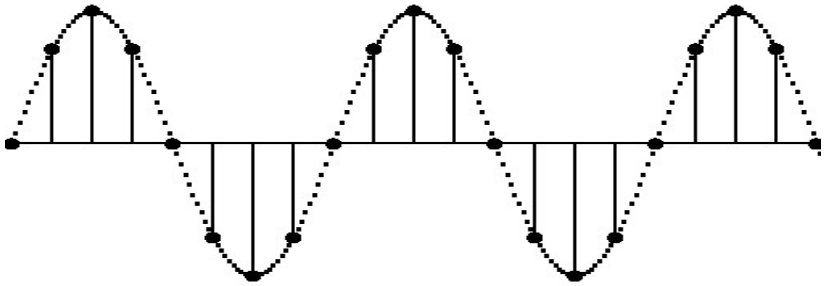
Nyquist:

- *Si fs < 2fa ocurrirá un fenómeno conocido como Aliassing.*
- *El fenómeno de Aliassing resulta de mucha utilidad en aplicaciones de submuestreo.*

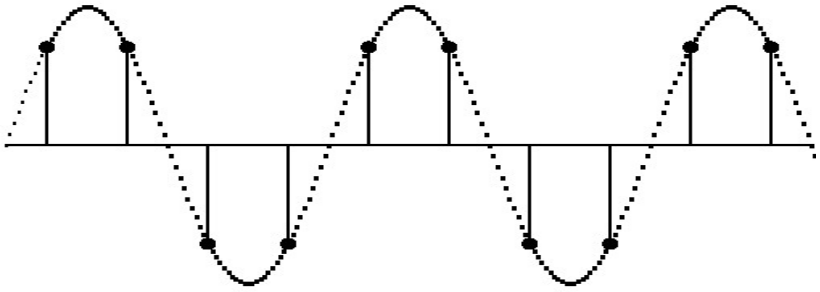
6.2 Efectos Del Sub-Muestreo En El Dominio Del Tiempo.

Para entender las implicaciones del aliasing en el dominio del tiempo, se presentan cuatro casos en la figura 6.2. En el caso 1, resulta evidente que se han tomado un número adecuado de muestras para mantener la forma de la onda senoidal. En el caso 2, solamente se han tomado cuatro muestras por ciclo; siendo aun un número adecuado para mantener la información. El caso 3 representa una condición ambigua donde fs = 2fa. Si la relación entre los puntos de muestra y la forma de onda senoidal fuera tal que la onda senoidal esté siendo muestreada en el preciso momento en que cruza por cero (en lugar de los puntos máximos como aparecen en la figura), entonces se perdería toda la información. El caso 4 ilustra la situación en la cual fs < 2fa, y la información obtenida de la muestras indica que se tiene una señal con frecuencia inferior a fs/2; es decir, aparece una copia (alias) de fa dentro del ancho de banda de Nyquist entre dc y fs/2. Si la velocidad de muestreo continua siendo disminuida, y la frecuencia de entrada fa se aproxima a la frecuencia de muestreo fs, la señal copia se aproximará a dc (0 Hz) en el espectro de frecuencia.

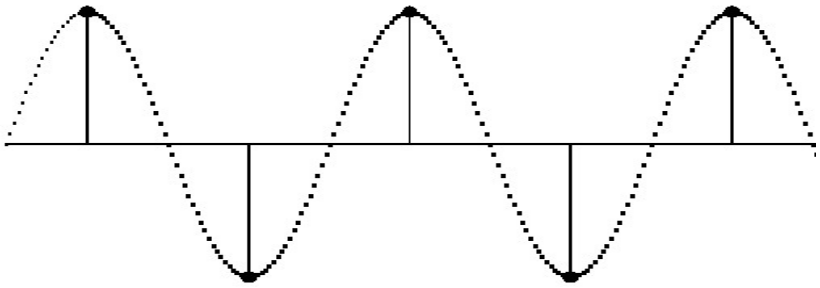
CASO 1: fs = 8fa



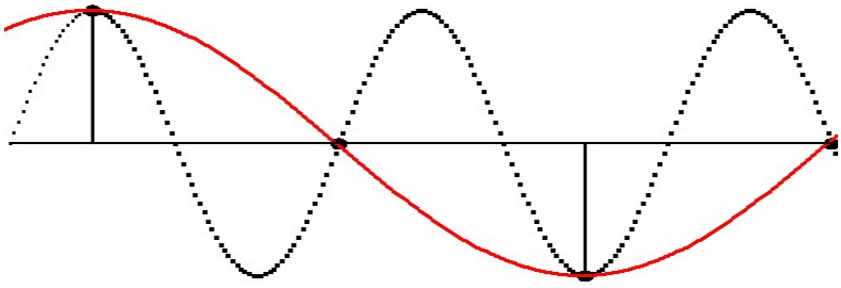
CASO 2: fs = 4fa



CASO 3: fs = 2fa



CASO 4: fs = 1.3fa



Efectos del Aliasing en el dominio del tiempo

Figura 6 2 Muestreo de una señal en el dominio del tiempo.

6.3 Efectos Del Sub-Muestreo En El Dominio De La Frecuencia.

En la figura 6.3 se presenta el escenario anterior en el dominio de la frecuencia. Se observa que una señal fa siendo muestreada a una velocidad fs produce dos alias, una en fs + fa y la otra en fs – fa. El alias superior raras veces representará un problema, debido a que se encuentra fuera del ancho de banda de Nyquist. Es el alias inferior, fs – fa, que provoca problemas cuando la señal de entrada excede el ancho de banda de Nyquist, fs/2.

De esta figura podemos realizar una conclusión muy importante: ***que sin importar en que lugar del espectro de frecuencia se encuentre la señal a ser muestreada (siempre y cuando no se encuentre en algún múltiplo de fs/2), el efecto del muestreo provocará ya sea que la misma señal o un alias de esta caiga dentro del ancho de banda de Nyquist entre dc y fs/2***.

Por tal razón, cualquier señal que caiga fuera del ancho de banda de interés, ya sea que se trate de tonos esporádicos o ruido aleatorio, debe ser filtrada adecuadamente antes del muestreo. Pues si no se hace así, el mismo proceso de muestreo puede provocar alias de dicho ruido dentro del ancho de banda de Nyquist y alterar la señal deseada.

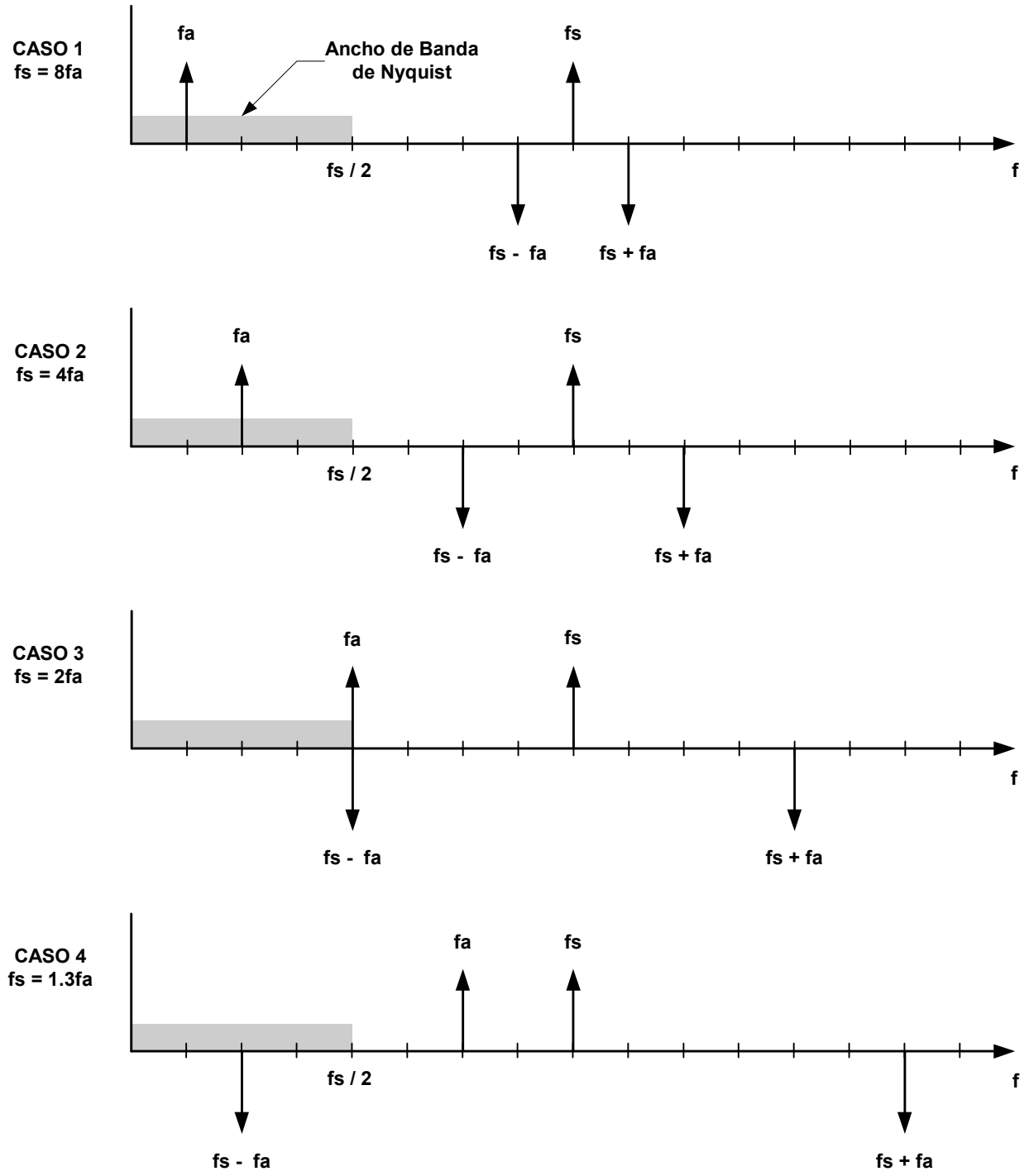


Figura 6 3 Muestreo de una señal en el dominio de la frecuencia.

Existen métodos que hacen uso del fenómeno de aliasing como una ventaja en aplicaciones de procesamiento de señales. La figura 6.4 muestra cuatro casos donde una señal con un ancho de banda de 1MHz es ubicada en diferentes porciones del espectro de frecuencia. La frecuencia de muestreo debe ser elegida de tal manera que no se produzca traslape entre los alias. En general la

frecuencia de muestreo debe ser de al menos dos veces el ancho de banda, y la señal muestreada no debe cruzar ningún múltiplo entero de fs/2.

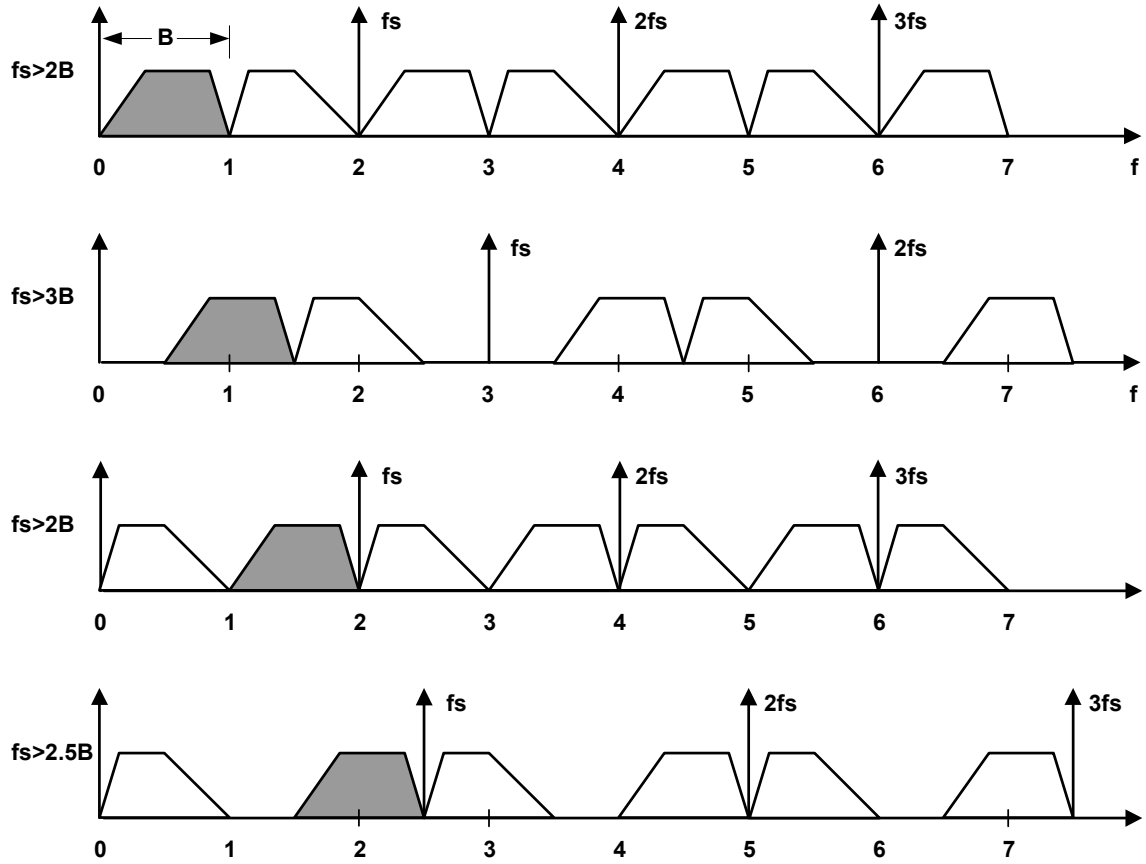


Figura 6 4

En el primer caso, la señal ocupa una banda desde dc hasta 1MHz, y por ello debe ser muestreada a frecuencias mayores a 2MSPS. El segundo caso muestra una señal de 1MHz que ocupa la banda desde 0.5 hasta 1.5MHz. Esta señal debe ser muestreada a una frecuencia de al menos 3MSPS para evitar el traslape de sus distintos alias. En el tercer caso, la señal ocupa la banda desde 1MHz hasta 2MHz, y la frecuencia mínima requerida es nuevamente de al menos 2MHz. El último caso muestra una señal que ocupa la banda desde 1.5 hasta 2.5MHz. Esta señal debe ser muestreada a un mínimo de 2.5MSPS para evitar el traslape de sus distintos alias.

Este análisis puede ser generalizado como se muestra en la figura 6.5. La velocidad mínima de muestreo requerida es una función de la relación entre la componente de mayor frecuencia, fMAX, y el ancho de banda total, B. Si la relación entre fMAX y B se hace mayor, la frecuencia de muestreo mínima requerida se aproxima a 2B.

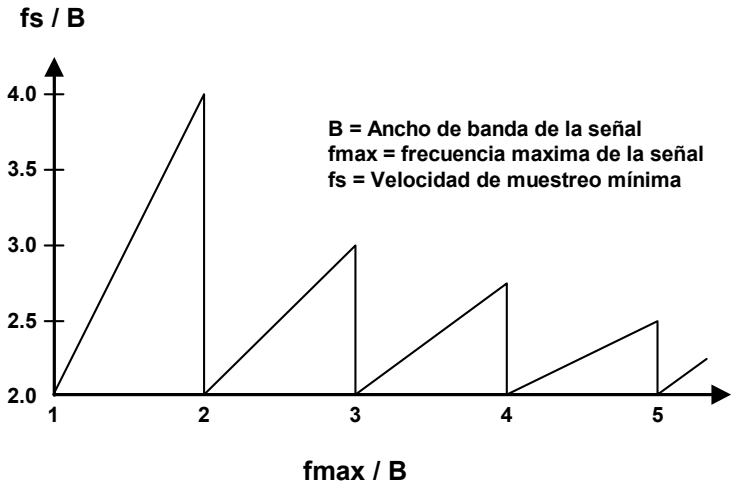


Figura 6 5

Ahora si se considera el caso de una señal que tiene un ancho de banda de 1MHz y se encuentra entre 6 y 7MHz del espectro de frecuencia es como el que se muestra en la figura 6.6. El teorema de información de Shannon establece que la señal (ancho de banda de 1MHz) debe ser muestreada a una velocidad de muestreo de al menos 2MSPS para retener toda la información (evitar traslape de los alias). Asumiendo que se tiene un ADC con velocidad de muestreo, fs, 2MSPS, frecuencias de muestreo adicionales son generadas en todos los múltiplos enteros de fs: 4MHz, 6MHz, 8MHz, etc. Entonces aparecerán alias de la señal en cada una de estas frecuencias harmónicas de muestreo, fs, 2fs, 3fs, 4fs, …., de ahí el termino "muestreo de harmónicas" (harmonic sampling). Se observa también que cualquiera de los alias es una representación exacta de la señal original (la inversión de frecuencia que ocurre a la mitad de cada alias puede ser removida mediante software). En particular, la componente que se encuentra en la región de banda base entre dc y 1MHz podría ser calculada usando la transformada rápida de Fourier (FFT), y sería también una representación exacta de la señal original, asumiendo que no hay errores de conversión del ADC. La FFT obtendría todas las características de la señal excepto su posición original en el espectro de frecuencia.

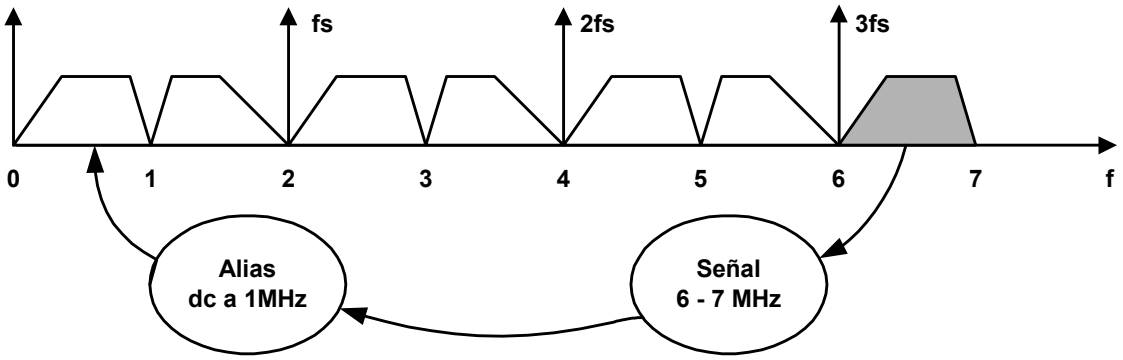


Figura 6 6

VII ANALISIS DE RESULTADOS

A continuación se presentan algunas señales capturadas por la tarjeta. Se presentan las opciones que ofrece el programa como son la suma y resta de señales.

Las señales de prueba que se han utilizado son señales de tipo senoidal, cuadrada y triangular obtenidas de un generador de señales en un rango de 1KHz hasta 1MHz.

La siguiente señal es una señal triangular de 902.77 KHz muestreada a 25MHz, a 1 V/Div y 1uS/Div. Debido a que el semiciclo negativo de la señal supera los ±4.5V de la escala que se había seleccionado, la señal aparece recortada.

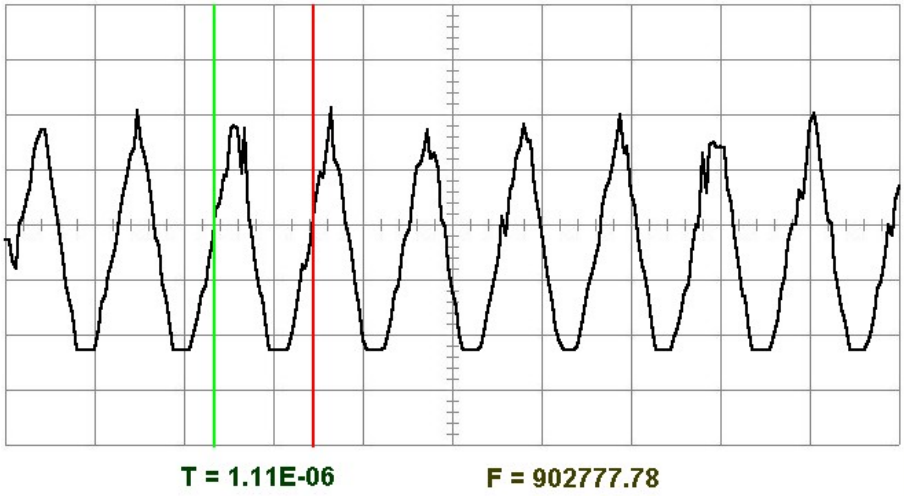


Figura 7 1 Señal triangular, 1V/Div, y 1uS/Div

El espectro de la señal anterior se muestra en la figura 7.2.

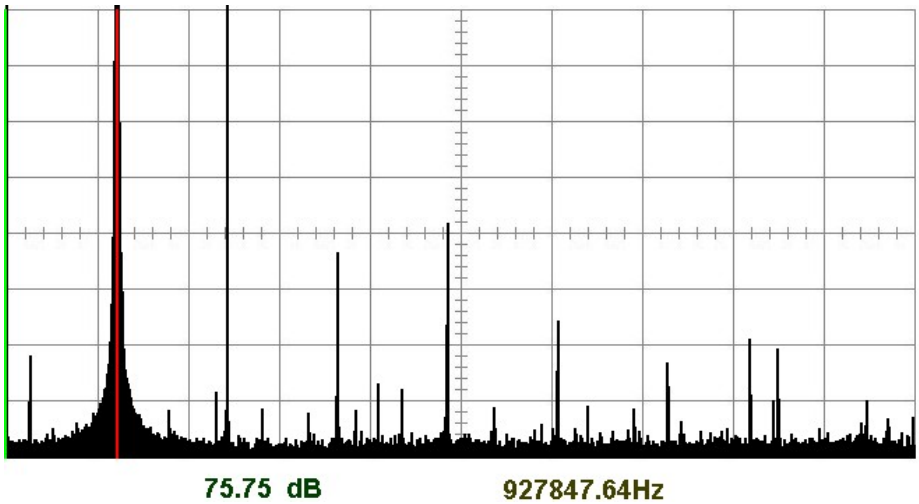


Figura 7 2 Espectro de una señal triangular

En la figura 7.2 se puede observar que el primer armónico ocurre a una frecuencia aproximadamente igual a la mostrada en la figura 7.1

La figura 7.3 muestra una señal cuadrada de 619.04 KHz, 1V/Div, 1 uS/Div y su correspondiente espectro se muestra en la figura 7.4

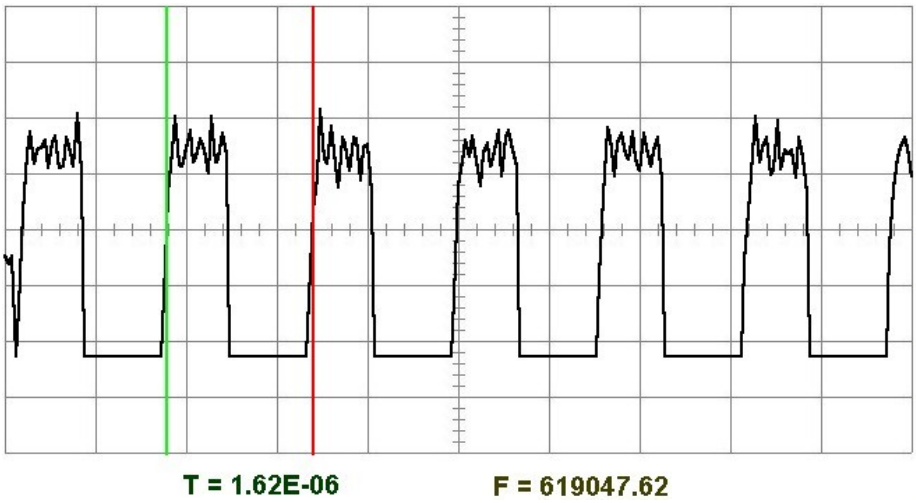


Figura 7 3 Señal cuadrada de 619.04KHz

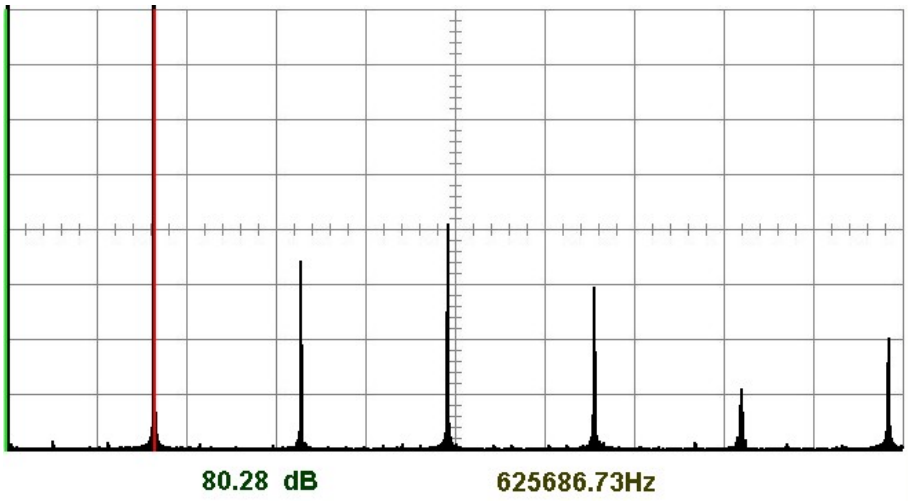


Figura 7 4 Espectro de una señal cuadrada

En la figura 7.5 y 7.6 se muestra una señal senoidal de 44.22 KHz, 1 V/Div, 10u S/Div.

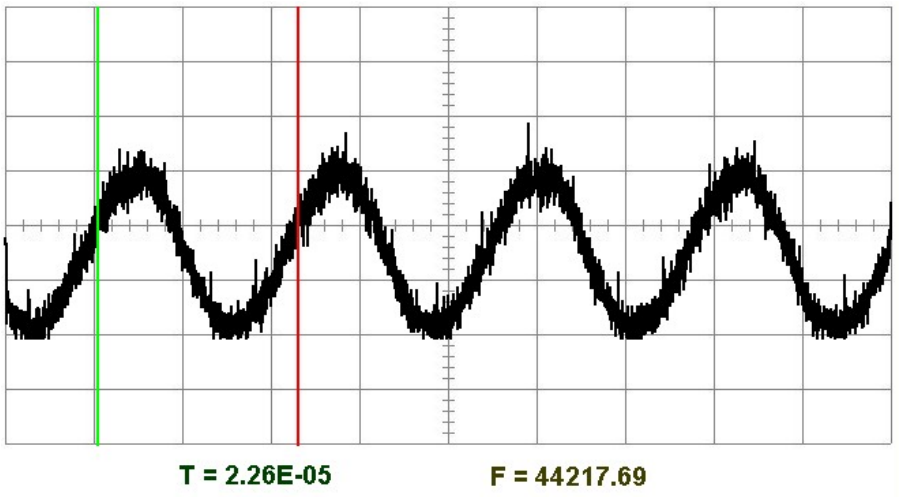


Figura 7 5 Señal senoidal de 44.22KHz

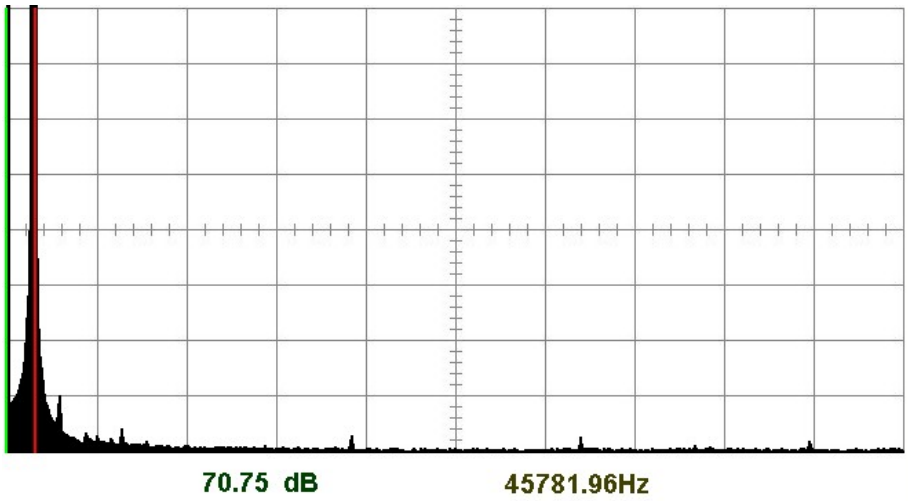


Figura 7 6 Espectro de una señal senoidal

Las siguientes imágenes muestran las opciones del osciloscopio; esto es, sumar o restar dos señales. La figura 7.7 muestra dos señales, una senoidal y la segunda triangular. Ambas han sido muestreadas a 2 V/Div y 20u S/Div. La figura 7.8 muestra el resultado de la suma. La figura 7.9 muestra el resultado de la resta.

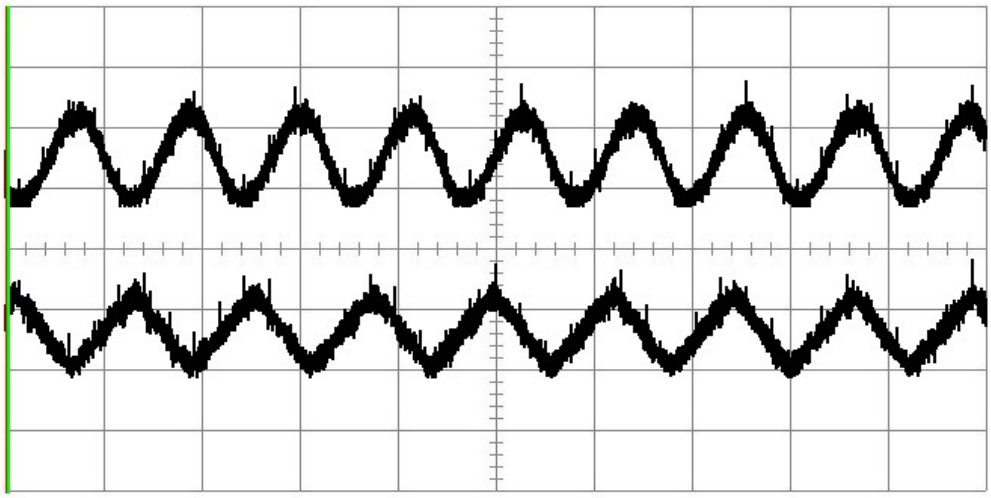


Figura 7 7 Forma dual: señal senoidal y triangular

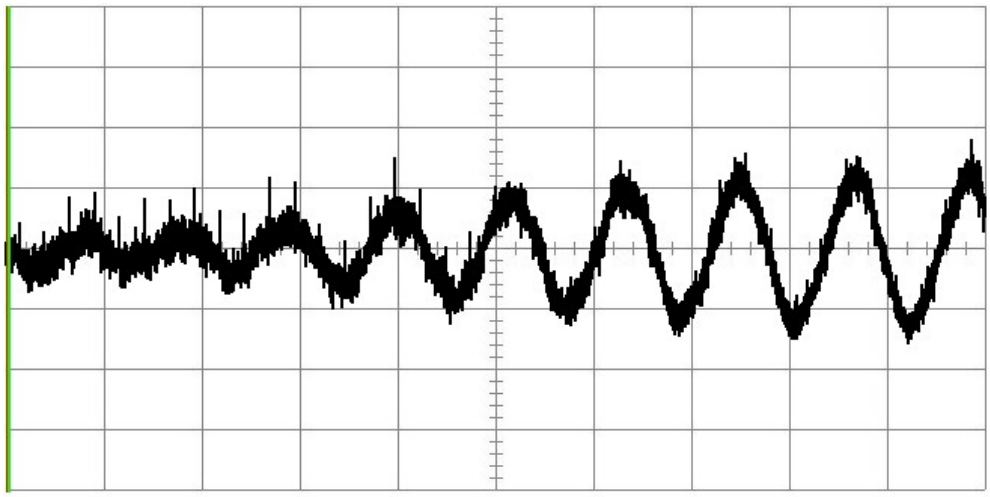


Figura 7 8 Suma de señales senoidal y triangular

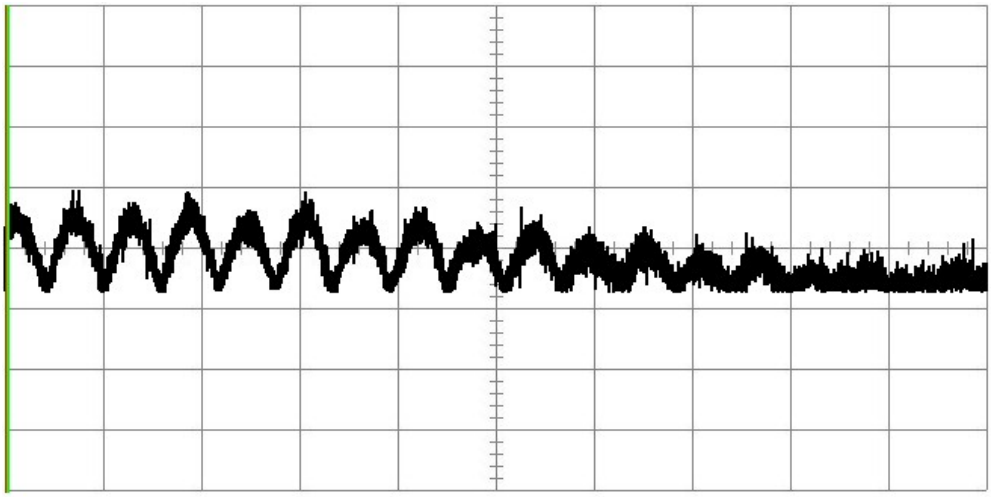


Figura 7 9 Resta de señales senoidal y triangular

VIII LISTA DE ELEMENTOS Y COSTOS

En esta sección se presenta la lista de materiales empleados en la construcción de la tarjeta así como el costo de cada uno de ellos. En el caso de los elementos que no se encuentra disponibles en el país se incluye una referencia que especifica el distribuidor en Estados Unidos.
Los nombres y descripción de los elementos se han establecido en ingles, pues esto facilita la búsqueda y compra a través de Internet.
El convertidor Análogo a Digital TLC5540 fue un caso especial, ninguno de los distribuidores lo ofrecía en pequeñas cantidades. Afortunadamente se pudo obtener muestras gratuitas desde el sitio web de "Texas Instruments" cuya dirección incluimos en las referencias.

TIPO	DESCRIPCIÓN	ELEMENTO		FABRICANTE	QTY	COSTO /U	TOTAL
IC	Op Amp 300MHz	MAX477	[1]	MAXIM	3	$4.57	$13.71
IC	250 MHz, General Purpose Voltage Feedback Op Amp	AD8048AN	[2]	ANALOG DEVICES	2	$4.14	$8.28
IC	CMOS Analog Multiplexers/Switches	MAX4051ACPE	[1]	MAXIM	1	$3.59	$3.59
IC	CMOS Analog Multiplexers/Switches	MAX4052ACPE	[1]	MAXIM	2	$3.59	$7.18
IC	Quad EXCLUSIVE-OR gate	74HC86	[3]		1	$0.39	$0.39
IC	Dual D flip-flop	74HC74	[3]		1	$0.35	$0.35
IC	CMOS PIC Microcontroller	PIC16F84A-20/P	[3]	MICROCHIP	1	$5.95	$5.95
IC	IC SRAM 32KX8 ASYNC PD 28-SDIP	CY7C199-15PC	[4]	CYPRESS SEMICONDUCTOR CORP	1	$2.30	$2.30
IC	8-BIT HIGH-SPEED ANALOG-TO-DIGITAL CONVERTER	TLC5540CNS	[5]	TEXAS INSTRUMENTS	1	$3.52	$3.52
IC	Interface Series, dual RS232 transmitter/receiver	MAX232ACPE	[3]	MAXIM	1	$2.19	$2.19
IC	HIGH-DENSITY PROGRAMMABLE LOGIC	ISP1016E100LT44	[6]	LATTICE SEMICONDUCTOR	1	$8.75	$8.75
IC		IC7805			2	$1.05	$2.10
IC		IC7905			2	$0.79	$1.58
XTAL	TTL Crystal Oscillator CRY,OSC,50MHz,100ppm,	OSC50	[3]	RAKON	1	$1.99	$1.99
XTAL	Crystal Oscillator	CY20A	[3]		1	$0.79	$0.79
	PUSH PULL	SS152			2	$0.40	$0.80
	HEAT SINK	HS220			4	$0.20	$0.80
RES	RESISTOR	510K 1/2W		GENERIC	4	$0.30	$1.20
RES	RESISTOR	10K 1/4W		GENERIC	3	$0.25	$0.75
RES	RESISTOR	220 1/4W		GENERIC	6	$0.25	$1.50

RES	RESISTOR	330 1/4W		GENERIC	2	$0.25	$0.50
RES	RESISTOR	100 1/4W		GENERIC	2	$0.25	$0.50
RES	RESISTOR	470 1/4W		GENERIC	1	$0.25	$0.25
RES	RESISTOR	120 1/4W		GENERIC	1	$0.25	$0.25
RES	RESISTOR	1K 1/4W		GENERIC	2	$0.25	$0.50
RES	RESISTOR	2.2K 1/4W		GENERIC	1	$0.25	$0.25
RES	RESISTOR	47K 1/4W		GENERIC	2	$0.25	$0.50
POT	1/2-Watt, 3/8" Square, Multi-Turn Trimming Potentiometer (±10%)	500R	[3]		2	$2.09	$4.18
POT	1/2-Watt, 3/8" Square, Multi-Turn Trimming Potentiometer (±10%)	1K			2	$2.09	$4.18
CAP	CAP,MONO,0.1uF,50V,20%	104	[3]	GENERIC	26	$0.07	$1.82
CAP	CAP,MYLAR,0.1uF,100V,10 % -10	100nF	[3]	GENERIC	2	$0.17	$0.34
CAP	CAP,MONO,1uF,50V,20%	1uF	[3]	GENERIC	10	$0.48	$4.80
CAP	DIPPED TANTALUM CAP, 25, 10%	10uF	[3]	GENERIC	2	$0.50	$1.00
CAP	CAP, CERM, DISC, 50V, 20% -10	10pF	[3]	GENERIC	4	$0.07	$0.28
CAP	CAP,CERM, DISC, 50V, 20% -10	100pF	[3]	GENERIC	3	$0.07	$0.21
CAP	CAP,RADIAL, 25V,105C 10mm x 20mm	1000uF		GENERIC	2	$0.35	$0.70
CAP	CAP,RADIAL, 25V,105C 10mm x 20mm	470uF		GENERIC	2	$0.17	$0.34
CAP	CAP,RADIAL, 25V,105C 10mm x 20mm	100uF		GENERIC	2	$0.07	$0.14
D	DIODE,SWITCH,75V PRV, Ir=50u,Vf=1v@10mA(10)	1N4148		GENERIC	6	$0.04	$0.24
D	DIODE,SIL REC,1N4002,1A, 100V PRV (10)	1N4002		GENERIC	4	$0.03	$0.12
L	150uH Axial RF Choke	L500		GENERIC	1	$0.35	$0.35
JFET	N-Ch, VHF Amp/Mixer	NTE312		GENERIC	4	$1.00	$4.00
BJT	NPN Signal	BC548 / 2N2222	[3]	GENERIC	2	$0.45	$0.90
BJT	PNP Signal	BC558	[3]	GENERIC	1	$0.45	$0.45
LED	Yellow 3mm LED	LED		GENERIC	2	$0.11	$0.22
CON	CONNECTOR,BNC,LOW PROFILE,RA PC MOUNT	CH-7044S	[3]	GENERIC	2	$1.95	$3.90
CON	CONNECTOR,HOUSING,MA LE,DB9	DB9	[3]	GENERIC	1	$0.50	$0.50
CON	JACK,DC PWR,MALE,2.5MM,PC TERM PC MOUNT	JACK CONECTOR		GENERIC	1	$0.45	$0.45

CABLE	CABLE,9PIN,SERIAL,10 DB9 FEMALE TO DB25 MALE	CABLE		GENERIC	1	$3.95	$3.95
HARDWARE	COPPER PLATED 21x14 cm MAIN BOARD	COPPER PLATED		GENERIC	1	$32.00	$32.00
HARDWARE	COPPER PLATED 5.6x3 cm PLD SOCKET	COPPER PLATED		GENERIC	1	$3.00	$3.00
HARDWARE	COPPER PLATED 4.2x2.8 cm ADC SOCKET	COPPER PLATED		GENERIC	1	$2.00	$2.00
PINS	HEADER,.1ST MALE,1RW, 3PIN, (10) .025PST,.23GOLDTAIL	PIN HEADERS	[3]	GENERIC	1	$0.17	$0.17
PINS	HEADER,.1ST MALE,1RW, 2PIN, (10) .025PST,.23GOLDTAIL	PIN HEADERS	[3]	GENERIC	4	$0.17	$0.68
PINS	HEADER,.1ST MALE,1RW, 12PIN, (10) .025PST,.23GOLDTAIL	PIN HEADERS	[3]	GENERIC	2	$0.25	$0.50
PINS	HEADER,.1ST MALE,1RW, 22PIN, (10) .025PST,.23GOLDTAIL	PIN HEADERS	[3]	GENERIC	2	$0.25	$0.50
HARDWARE	MODULAR OSCILLOSCOPE PROBE	OSCILLOSCOPE PROBE	[3]	GENERIC	2	$25.00	$50.00
SOURCE	TRANS,WALL,12VAC, 1600mA, 2.5mm X 5.5mm	TRANSFORMER	[3]	GENERIC	1	$9.95	$9.95
						TOTAL	**$202.34**

[1] http://www.maxim-ic.com/index.cfm

[2] http://www.newark.com

[3] http://www.jameco.com

[4] http://www.digikey.com

[5] www.ti.com

[6] http://www.avnet.com/

CONCLUSIONES

- La unidad central de la tarjeta adquisidora de datos es el PIC, un microcontrolador muy confiable, fácil de manejar y programar debido a su tipo de arquitectura. Se encarga de controlar todos los procesos del sistema; tales como: conversión, direccionamiento, almacenamiento, y de facilitar el desplazamiento de los datos hacia la computadora de manera serial. Su inclusión en el sistema ha facilitado el diseño de hardware y software debido a que hace que la tarjeta sea independiente de la computadora, dejándole las tareas más fuertes como procesamiento de datos y gráficas obteniendo así mejor rendimiento en ambas partes. Puede funcionar con diferentes velocidades y tipos de microprocesadores con muy pocas variaciones en el código ensamblador.

- La tarjeta adquisidora de datos posee registros dinamicos que rigen el comportamiento del sistema, es decir que el usuarios los puede modificar cualquier registro en cualquier momento, lo que da ventaja a diferencia de los registros estaticos que si se desea cambiar el registro debe de ser retirado el PIC para programar su nuevo valor.

- Cada vez que se envia un comando este es devuelto como confirmacion lo que sirve para indicar al programa que la tarjeta esta conectada, de lo contrario este mandaria un mensaje de error que indica verificación del dispositivo.

- Al hacer que la tarjeta funcione con comandos con un set de instrucciones compacto y de muy facil entendimiento da una gran ventaja al programador ya que este no nesecita saber de programación de microcontroladores o PLDs simplemente se dedica a programar a base de envio de comandos para la modificacion de registros, ejecución de operaciones. Por lo que puede ser prgramada con cualquier lenguaje de programación siempre y cuando tenga la capacidad de manejar el puerto serial

- Al incrementar la velocidad comunicacion de 57600 baudios a 115200 baudios hace que se pueda descargar mas muestras en menos tiempos, por lo que el buffer de almacenamiento ha crecido en la misma proporcion. El incremento en la velocidad de comunicación se ha obtenido colocando una unidad PIC con mayor frecuencia nominal de trabajo de 10 MHz a 20 MHz que es la versión mas rapida para cualquier modelo del 16F84A, colocando otro microcontrolador con mayor capacidad de reloj se podrian coseguir velocidades donde ya se puede utilizar el puerto USB disminuyendo el tiempo de refrescamiento.

- Debido a su interface serial hace facil su conexión con la PC sin nesecidad de abrir el case de la PC como lo hacen las tarjetas basadas en el bus ISA o PCI

- El sistema de PosTrigger da la ventaja de que si se de controlar el numero de muestras que se desee almacenar, dando un buen ajuste a los registros que controlan este retrazo podemos llenar la RAM o tomar un pequeño numero de muestras para graficar.

- La introducion de un sistema complejo de disparo facilita obtener un eje de referencia de donde debe de comenzar la grafica para que en su refrescamiento este pueda ser graficada desde el mismo punto o con un valor cercano.

- El sistema de *Mascara* en la obtención de un disparo ayuda grandemente en la obtención de un disparo. Controlando los registros que intervienen en el disparo podemos obtener un margen para encontrar un disparo.

- La utilización de un Programador Logico ha reducido en forma sensible el tamaño y el costo de la placa y el poder manejar mayores frecuencia para el direccionamiento ya que puede trabajar a la misma o mayor frecuecia de muestreo que se este utilizando.

- Los 4 modos de trazado o muestreo se puede obtener captura de datos de un solo canal o de ambos; o de muestrear a baja o alta frecuencia. Modelando otros modos de Trace como medicion de frecuencia, medicion de pulsos de alta frecuencia, etc. Pueden ser agregados fácilmente al PIC ya que este posee 1 k de memoria.

- El rango de voltaje de entrada es muy limitado, no consiguiendo poder medir valores de voltaje muy alto, debido a su sistema de atenuación y al span de entrada del ADC.

- El costo de la tarjeta es considerablemente bajo con respecto a un osciloscopio, ya que esta construido por componentes de gama media de costo relativamente bajo, aunque la mayoria de elementos no son encontrados en el mercado local, estos deben de ser comprados con un proveedor de semiconductores en el extranjero ya sea por Internet o por una tienda dedicad a esto. Por lo que podria ser construido por estudiantes o aficionados que cuentan con una PC

- Al agregar un software que realice la Transformada Rapida de Fourier se puede obtener de forma rapida el espectro de una señal y sus componente en fecuencia y magnitud, sin nesecidad de modificar la tarjeta adquisidora dandole una doble funcion.

- Las metas de la programación orientada al objeto es mejorar la productividad de los programadores haciendo más fácil de reutilizar y extender los programas y manejar sus complejidades. De esta forma, se reduce el costo de desarrollo y mantenimiento de los programas. En este tipo de lenguajes los datos son considerados como objetos que a su vez pertenecen a alguna clase.

.

REFERENCIAS

1. http://www.ic-prog.com/index1.htm
   Desde este sitio se puede descargar el programa ICPROG, software que permite programar algunos circuitos integrados a través del puerto serial.

2. http://www.todopic.com.ar
   Excelente sitio argentino, tipo foro, donde se puede encontrar mucha información acerca de los microcontroladores PIC, documentación, preguntas más frecuentes, etc.

3. http://www.fe.up.pt/~victorm/TTL.htm
   Sitio desde el cual se pueden descargar hojas técnicas de componentes TTL.

4. http://www.microchip.com
   Sitio oficial del fabricante de microcontroladores Microchip. Desde este sitio se pueden descargar hojas técnicas, herramientas de desarrollo como el simulador MPLAB, documentación de distintas aplicaciones así como el código ensamblador, etc.

5. http://www.geocities.com/un-2000/jdm.htm
   Este sitio contiene un circuito sencillo para un programador de microcontroladores PIC empleando del puerto serial.

6. www.analog.com/UploadedFiles/Associated_Docs/ 163524457Section5.pdf
   Documento en formato pdf desarrollado por Walk Kester con muy buena información acerca de la técnica de Sub-muestreo.

7. http://msdn.microsoft.com/developer/related/vbasic.htm
   Sitio oficial de Microsoft Visual Basic. Contiene noticias, información de productos y documentación para Visual Basic. También se pueden encontrar vínculos para descargas, recursos, consejos, soporte técnico, etc.

8. http://www.boondog.com/tutorials/dlltutor/dlltutor.htm
   Excelente sitio donde aparecen desarrollados muchos circuitos incluidos sus software. En especial, resultan de mucha utilidad para este proyecto de tesis la información proporcionada para manejar los puertos de la computadora con Visual Basic, como desarrollar archivos dll que puedan ser empleados por Visual Basic y el documento sobre el microcontrolador 16F84.

9 http://elm-chan.org/
   Este sitio ha sido desarrollado por Takeshi Akamatsu. En este sitio aparece el diseño de un Osciloscopio Digital de baja frecuencia, pero que presenta un diseño muy interesante; especialmente por que la comunicación con la computadora es vía serial y hace uso de un microcontrolador.

10. http://www.essi.fr/SSI/
Sitio interesante donde se presentan algunas de las ventajas del procesamiento de señales a través de las computadoras. Trata especialmente el tema de Sub-muestreo. Tiene especial valor, el documento llamado EAEEIE.DOC desarrollado por Jean-Paul Stromboni.

11 www.latticesemi.com
Sitio oficial de la compañía lattice semiconductor, donde se obtiene la hoja tecnica del ispLSI 1016E y siti de descarga de los programas ISPVMSYSTEM y ISPDESIGEXPERT para la programacion y compilación respectivamente de los PLDs

12. Deitel y Deitel, C++ Como Programar, Segunda Edición, 1999, pg. 35 – 38.

13. Deitel y Deitel, Visual Basic 6 How To Program, 1999.

14. Richard Grier, Visual Basic Programmer's Guide To Serial Communications, Tercera Edición, 15 De Febrero de 2002.

15. Jan Axelson, Parallel Port Complete: Programming, Interfacing & Using the PC'S Parallel Printer Port, 199

16 José Ignacio Artiga Maestre, Luis Ángel Barragán, , Electrónica Digital Aplicaciones Y Problemas con VHDL

ANEXOS

ANEXO 1: DIAGRAMAS ESQUEMATICOS

ANEXO 2: PROGRAMA PLD

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity scope9_0 is
            port (clk : in std_logic;
                rst   : in std_logic;
                rb4   : in std_logic;
                rb0   : in std_logic;
                 rb7  : in std_logic;
                rb1   : out std_logic;
                analogo: in std_logic_vector(7 downto 0);
--               equal_temp : buffer std_logic;
                ram :  out  std_logic_vector(14 downto 0);
       --      actual, ad1 : buffer std_logic_vector(7 downto 0);
--              reg_serial : buffer std_logic_vector(32 downto 0);
       --       a, b, q0,  c, d,h, m, st_byte, q, qn,n : buffer std_logic);
                n : out std_logic);
end;

architecture behavioral of scope9_0 is
       signal equal_temp: std_logic;
       signal cnt1 :  std_logic_vector(1 downto 0);
       signal cntram, load: std_logic_vector(15 downto 0);
       signal option_byte, triger_mask, triger_logic : std_logic_vector(7 downto 0);
       signal reg_serial, reg_temp: std_logic_vector(16 downto 0);
       signal ad1, actual: std_logic_vector(7 downto 0);
       signal positivo, negativo,k,l,m,o,p,q,r,u,s,v: std_logic;
       signal antes, t: std_logic_vector(7 downto 0);
       signal ps, ps_temp: std_logic_vector(15 downto 0);



begin

       process (rst, clk )
              --
              begin
                     --
                     if rst = '1' then
              --
                            reg_serial <= (others => '0');
       --
                            ps <= (others => '0');
                     elsif (clk'event and clk ='1') then                                    --
                            reg_serial <= reg_temp;
              --
                            ps <= ps_temp;
                     end if;
                     --
              end process;
              --

                            --
       process (rb4,rb0, reg_serial, rb0, cntram,  ps)
              --
              begin
                     --
                     if rb4 = '1' then
              --
                            reg_temp <= reg_serial(15 downto 0) & rb0;  -- entrada de datos del
pic
                            ps_temp <= ps(14 downto 0) & ps(15); -- salida de registros al pic
                            rb1 <= ps(15); -- pin de salida
                     else
                     --
                            reg_temp <= reg_serial;
              --
                            ps_temp <= cntram;
                            rb1 <= 'Z';    -- pin dee salida en alta impedancia
```

```
                    end if;
                    --
             end process;
             --
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
ad1 <= ((not reg_serial(7 downto 0)) and analogo);
------------------comparador de el disparo--------------------------------------
equal_temp <= '1' when ((reg_serial(15 downto 8)= ad1 ) and rb4 ='0') else '0';
-------------------------------------------------------------------------------

       process (equal_temp, u)
       begin
              if equal_temp = '1' then s <= '1';
               elsif   u = '0' then s <= '0';
              end if;
       end process;

u <= not positivo;
t  <= analogo and "11110000";
v <= '1' when (t > reg_serial(15 downto 8))else  '0';
positivo <= s and v;

-------------------------contadores------------------------------------------------
----------------------contador de dirreciones de la RAM--------------------------


                        --
             process (clk, rst, rb4, cntram, positivo  )
                    --
             begin
                    --

                    if rst='1' or (rb4 = '1') or (positivo = '1') then
                                        --
                           cntram <= (others => '0');
       --
                    elsif (clk'event and clk ='1') then                                    --
                           if rb4 = '0'  and (not (cntram(15) = '1')) then                 --
                                  cntram <= cntram +1;
                           end if;                                              --
                    end if;                                                     --
             end process;                                                       --
       ram <= cntram(14 downto 0);

-------------------------------------------------------------------------------


--multiplexor
       q <= positivo and reg_serial(16);

       r <= cntram(15) and reg_serial(16);
       l <= q or r;
       o <= l and rb7;

process (o, rb4 )
       begin
              if o = '1' and rb4 = '0' then p <= '1';
               elsif rb4 = '1' and o = '0' then p <= '0';
              end if;
       end process;

       n <= p;


-------------------------------------------------------------------------------------------
end behavioral;
```

ANEXO 3: CODIGO ASM DEL PIC

```
        list    p=16F84,    n=92
     include <p16f84a.inc>           ; include for this chip
;*****************************************************************
;       Serial Communications Interface Subroutines
;       File Name: scope3.asm
;       Date: 27/05/2003
;       Author: Camilo Merino Melgar / U.D.B.
;       Modificado: 26/02/2004
;       Protocol: 8 N 1 NRB (transmits LSB -> MSB) (receives LSB-> MSB)
;       external constant: BAUD_SEL - selects baud rate constant value
;*****************************************************************

INDF                        EQU     H'0000'
TMR0                        EQU     H'0001'
PCL                         EQU     H'0002'
STATUS                      EQU     H'0003'
FSR                         EQU     H'0004'
PORTA                       EQU     H'0005'
PORTB                       EQU     H'0006'
PCLATH                      EQU     H'000A'
INTCON                      EQU     H'000B'
GIE                                         EQU     H'0007'
RBIF                         EQU     H'0000'
OPTION_REG                  EQU     H'0081'
TRISA                       EQU     H'0085'
TRISB                       EQU     H'0086'

RA0                 EQU             H'0000'
RA1                 EQU             H'0001'
RA2                 EQU             H'0002'
RA3                 EQU             H'0003'
RA4                 EQU             H'0004'

RB0                 EQU             H'0000'
RB1                 EQU             H'0001'
RB2                 EQU             H'0002'
RB3                 EQU             H'0003'
RB4                 EQU             H'0004'
RB5                 EQU             H'0005'
RB6                 EQU             H'0006'
RB7                 EQU             H'0007'


Txbuff              EQU             0x20    ;transmit buffer (1 byte)
Rxbuff              EQU             0x21    ;receive buffer (1 byte)
delay_var           EQU                     0x22    ; variable para retardo
bit_count           EQU                     0x24    ;counts # of bits to be transmitted or
received
wait_cnt            EQU                     0x0f    ;wait loop counter
temp_transf         EQU                     0x26    ;ram para almacenar un valor temporal
temp_transfA   EQU                  0x27
data_byte           EQU                     0x28    ;contador de los datos del multiplexor
byte_d              EQU                     0x2a    ;el dato a transmitir
var_swap            EQU                     0x2b

C                   EQU             0x00    ;status C bit= 0
RP0                 EQU             0x05    ;Status RP0 bit = bit 5
Z                   equ             0x02    ;status z bit = 2
f                   equ             0x01
w                   equ             0x00

R0                  equ     0x30    ; el registro R0 de manejo
R1                  equ             0x31
R2                  equ             0x32
R3                  equ     0x33
R4                  equ             0x34
R5                  equ             0x35
R6                  equ             0x36
R7                  equ             0x37
R8                  equ     0x38
R9                  equ             0x39
```

```
R10                     equ             0x3A
R11                     equ             0x3b
R12                     equ             0x3c
R13                     equ     0x3d
R14                     equ             0x3e
R15                     equ             0x3f

SER_OUT                 EQU             0x05    ;I/O pin for serial transmit SER_OUT <PORT,BIT>
       PORTB,RB5
SER_IN                  EQU             0x06    ;I/O pin for serial receive SER_IN <PORT,BIT>
PORTB,R6

;*****************************************************************
       org 0x0000

       call init_0    ; inicializa el PIC
       call init_bc             ; initialize the ByteCode machine
       nop
idle   call idle

;*****************************************************************
;***** SERIAL RECEIVE ROUTINE [receive:]
;***** Receives data from SER_IN (I/O pin) LSB first returns data in 'W' reg.
;***** Frame error: 'C'=0 , no errors: 'C'=1
       org 0x004
Interrupciones
       bcf     INTCON,GIE           ;Asegura la
       btfsc   INTCON,GIE           ;deshabilitacion
       goto    Interrupciones

receive:       bcf     STATUS,RP0            ; bank 0
               btfsc   PORTB,SER_IN                  ; see if start bit = 0
               goto    Rxerror               ; Rx line idle
receive1       movlw   D'9'                  ; get 9 bits...
               movwf   bit_count             ; initialize counter
               clrf    Rxbuff                ; clear the input buffer
               call    bit_star

               bcf     STATUS,C              ; make sure 'C' is clear or will write into buffer
get_bit
read_bit       bsf     Rxbuff,0              ; assume a 1
               btfss   PORTB,SER_IN                  ; read the actual state
               bcf     Rxbuff,0              ; make a 0
               rrf     Rxbuff,F              ; NEW_BIT -> 'C'

               call    delay_bit
               decfsz  bit_count,F           ; on last bit ?
               goto    get_bit               ; no, get next bit
got_stop       btfss   STATUS,C       ; should be a '1'
               goto    Rxerror               ; return an error flag

               movf    Rxbuff,W              ; put return value in W-reg.
               movwf   Txbuff
               bcf     INTCON,RBIF           ;limpia la bandera de interupcion
               MOVLW   0x03                  ;cargara la pagina 3xx de la memoria
                MOVWF   PCLATH               ;en el PC
               MOVF    Rxbuff   , W
                IORLW   0x80                 ;exor de Rxbuff con #80
               MOVWF   PCL

;*****************************************************************
exit_int       retfie                        ; exit

;--------------envio del acknowledg al host------------------------------
ack
       call transmit
       call exit_int
       return
;*****************************************************************
;******************** envia el dato almacenado en W ********************
send
```

```
        MOVWF Txbuff
        CALL transmit
        RETURN
;********************carga del numero de bits a transmitir***************
bits_num
        MOVLW 0x08
        MOVWF bit_count
;--------------------descarga del counter spock al pic---------------------
down_counter
                BTFSC temp_transf,7
                BSF PORTB,0                     ; !!Bank!! PORTB - TRISB
                BTFSS temp_transf,7
                BCF PORTB,0                     ; !!Bank!! PORTB - TRISB
                BTFSC PORTB,1                   ; !!Bank!! PORTB - TRISB
                BSF STATUS,C
                BTFSS PORTB,1                   ; !!Bank!! PORTB - TRISB
                BCF STATUS,C
                BCF PORTA,3                     ; !!Bank!! PORTA - TRISA
                NOP
                BSF PORTA,3                     ; !!Bank!! PORTA - TRISA
                RLF temp_transf,F
                DECFSZ bit_count,F
                GOTO down_counter
                RETURN
;-------------------------------------------------------------------------
;*************************************************************************
;****************salida de interrupcion y retorno a linea idle ************
exit_idle
        call exit_int
        goto 0x0003
;*************************************************************************
;*****************envio del ack y dde regreso al estado idle**************
ack_idle
        call init_0
        call ack
        goto 0x0003
;********************limpia el registro R0*********************************
clear_r0
        CLRF R0
        goto ack_idle
;*****************incrementar el registro R0 ****************************
niblle_f        incf  R0,f
niblle_e        incf  R0,f
niblle_d        incf  R0,f
niblle_c        incf  R0,f
niblle_b        incf  R0,f
niblle_a        incf  R0,f
niblle_9        incf  R0,f
niblle_8        incf  R0,f
niblle_7        incf  R0,f
niblle_6        incf  R0,f
niblle_5        incf  R0,f
niblle_4        incf  R0,f
niblle_3        incf  R0,f
niblle_2        incf  R0,f
niblle_1        incf  R0,f
niblle_0        SWAPF R0,F              ;intercambia los niblles
                         goto ack_idle
;-------------------------------------------------------------------------
;***************transfiere R0 a R1 ************************************
load_adress
        MOVF R0,W
        MOVWF R1
        goto ack_idle
;*************************************************************************
;******************** load R2 a R0****************************************
load_r2_r0
        MOVF R2,W
        ADDLW 0x30
        MOVWF FSR
        MOVF INDF,W
```

```
        MOVWF R0
        goto ack_idle
;*****************************************************************************
;********************* copia R0 a R1*************************************
R0_R1
        MOVF R1,W
        ADDLW 0x30
        MOVWF FSR
        MOVF R0,W
        MOVWF INDF
        goto ack_idle
;*****************************************************************************
;*********************incrementa R1 "n" ***********************************
next_adress
        INCF R1,F
        goto ack_idle
;*******************rutina de decrementar el registro r1 *****************
dec_reg
        MOVF  R1,W
        ADDLW 0x30
        MOVWF FSR

        decf INDF,F
        goto ack_idle
;*****************************************************************************
;*******************rutina de incrementar el registro r1 *****************
inc_reg
        MOVF  R1,W
        ADDLW 0x30
        MOVWF FSR
        INCF INDF,F
        goto ack_idle
;*****************************************************************************
;*****************descargar los registros al spock************************
spock_reg
        MOVLW TRISA
        MOVWF FSR
        BCF INDF,3
        MOVLW TRISB
        MOVWF FSR
        BSF INDF,1
        BSF PORTB,4                     ;
        BCF PORTB,7                     ;
        BSF PORTA,3                     ;
        BCF INDF,0
        MOVF R7,W
        MOVWF temp_transf
        CALL bits_num
        MOVF R5,W
        MOVWF temp_transf
        CALL bits_num
        MOVF R6,W
        MOVWF temp_transf
        CALL bits_num
        BSF INDF,0
        goto ack_idle
;********obtener transferir el contador a la pc*****************************
get_counter
        MOVLW TRISA
        MOVWF FSR
        BCF INDF,3
        MOVLW TRISB
        MOVWF FSR
        BSF INDF,1
        BSF PORTB,4                     ;
        BCF PORTB,7                     ;
        BSF PORTA,3                     ;
        BCF INDF,0
        MOVF R7,W
        MOVWF temp_transf
        CALL bits_num
```

```
        MOVF temp_transf,W
        MOVWF R10
        MOVF R5,W
        MOVWF temp_transf
        CALL bits_num
        MOVF temp_transf,W
        MOVWF R9
        MOVF R6,W
        MOVWF temp_transf
        BSF INDF,0
        RETURN
;*************************************************************************
;*****************descargar las muestras hacia la PC************************
dump
        BSF PORTA,3                  ; coloca el zz clock a 1
        MOVLW TRISA                  ;hace que el zz clock se convierta
        MOVWF FSR                    ; en salida para podeer manipular
        BCF INDF,3                   ; direccion tras direccion
        MOVLW TRISB                  ;
        MOVWF FSR                    ;coloca los puertos RB1 como salida y RB4,RB7
        BCF INDF,1                   ; a 0
        BCF PORTB,4                  ;
        BCF PORTB,7                  ;
        call ack                     ; envia al host la confirmacion
        MOVLW 0x0D                   ; caracter < CR > a enviar
        CALL send
        clrf   R3
        clrf   R4
next_byte      CALL multiplexeo      ; llama a la rutina de convercion paralelo serie
               MOVF temp_transf,W
               MOVWF byte_d          ; el valor obtenido se almacena  en byte_d
               call send             ;envia la muestra valida
               BCF PORTA,3           ; coloca a 0 el zz clock
               BSF PORTA,3           ; coloca a 1 zz clock para cambio en el spock
               INCF R3,f
               BTFSC STATUS,Z        ; se increnmentara la direcion dee la muestra actual
               INCF R4,f             ;
               movf R9,w
               xorwf R3,w            ; cuando es igual al spock counter
               BTFSs STATUS,Z        ; esta se detendra y de obtener muestras
               goto next_byte
               movf R10,w
               xorwf R4,w
               BTFSs STATUS,Z
               goto next_byte        ;
               MOVLW 0x0D            ; cuando se han trasmitido todas las muestras
               CALL  send            ; se envia < CR > de confirmacion
               call  init_0
               goto 0x0003
;*************************************************************************
;***** SERIAL TRANSMIT ROUTINE [transmit:]
;***** Call with data to transmit in W-reg. / sends LSB first on SER_OUT (I/O pin)
;***** Sets 'C' bit after completion if no errors
transmit:      bcf    STATUS,RP0                  ; select Bank 0
               movlw  D'9'                        ; transmit 9 bits (+ stop bit)
               movwf  bit_count                   ; initialize bit counter
               bsf    STATUS,C                          ; set 'C' for stop bit (& frame error
check)
start_bit      bcf    PORTB,SER_OUT               ; send start bit '0'
bit_delay      call   delay_bit_1
next_bit       decfsz bit_count,F           ; all 9 bits transmitted ?
               goto   send_data                   ; no, go and transmit next bit
               goto   $+1                                ; pad for timing (2 cycles)
stop_bit       rrf    Txbuff,F                    ; rotate to get the stop bit
               btfss  STATUS,C                    ; see if 'C' is set
               goto   transmit_err                ; will exit with an error flag (C=0)
               bsf    PORTB,SER_OUT                      ; send stop '1'
stop_bit1      nop
               nop
               call delay_bit_1
               goto   transmitx                   ; go to exit (with 'C' set)
```

```
send_data       rrf     Txbuff,F                ; rotate to put next bit in 'C'
                        btfsc   STATUS,C                ; see if 'C' is clear
                        goto    transmit_hi             ; go send a '1'
transmit_lo     bcf     PORTB,SER_OUT           ; send a '0'
                        goto    bit_delay               ; wait for bit time
transmit_hi     bsf     PORTB,SER_OUT           ; send a '1'
                        goto    bit_delay               ; wait for bit time
transmit_err    bsf     PORTB,SER_OUT           ; make sure send line is in idle state
transmitx
Rxerror         bcf     STATUS,C                ; return an error flag (C=0)
                        return
;*******************************************************************************
;*************** rutina de convercion y guardar los datos *************************
trace
       call channel          ;configura los canales;
       call ack              ; envia una confirmacion del mismo comando y sale de int.
       MOVF R11,W
       MOVWF temp_transf             ; carga el post trigger delay low
       INCF temp_transf,F            ; se aumenta a 1 para evitar que se haga 00
       MOVF R12,W                    ;
       MOVWF temp_transfA            ; carga el post trigger delay high
       INCF temp_transfA,F           ; se aumenta a 1 para evitar que se haga 00
       MOVLW 0x03
       MOVWF PCLATH          ; segun el modo de trace saltara a la 350
       movf R8,w
       ANDLW 0x0F
       ADDLW 0x50
       MOVWF PCL
;*******************************************************************************
descarga
       call get_counter
       MOVLW 0x0D            ; se mandara < CR >
       CALL send
       MOVF R10,W
       call send
       MOVF R9,W
       call send
       MOVLW 0x0D
       CALL send
       goto 0x0003
;*******************************************************************************
;******* rutina de transferencia de datos y conversion de datos paralelos(ADC) a serie (al PIC)
multiplexeo
              BCF PORTB,1                   ; !!Bank!! PORTB - TRISB
              BCF PORTB,2                   ; !!Bank!! PORTB - TRISB
              BCF PORTB,3                   ; !!Bank!! PORTB - TRISB
              BCF PORTB,4                   ; a sacar muestras
              MOVLW 0x07
              MOVWF data_byte
nx_bit
              BTFSC PORTB,0                 ; entrada de datos al pic
              BSF STATUS,C
              BTFSS PORTB,0                 ; al carry
              BCF STATUS,C
              RRF temp_transf,F
              MOVF PORTB,W                  ; !!Bank!! PORTB - TRISB
              ADDLW 0x02
              MOVWF PORTB                   ; !!Bank!! PORTB - TRISB
              DECFSZ data_byte,F
              GOTO nx_bit
              BTFSC PORTB,0                 ; !!Bank!! PORTB - TRISB
              BSF STATUS,C
              BTFSS PORTB,0                 ; !!Bank!! PORTB - TRISB
              BCF STATUS,C
              RRF temp_transf,F
              BCF PORTB,4                   ; !!Bank!! PORTB - TRISB
              RETURN
;*********************************************************************
source_adress  movf R0,w
              MOVWF R2
              goto ack_idle
```

```
;*********************************************************************
;*************delay bit star Rx ***************************************
bit_star
              movlw     .12         ; 1 set numero de repeticion
        movwf     wait_cnt    ; 1 |
PLoop0  nop                   ; 1 nop
        decfsz    wait_cnt, 1        ; 1 + (1) es el tiempo 0  ?
        goto      PLoop0      ; 2 no, loop
        return
;*********************************************************************
;*************** delay bit rx ****************************************
delay_bit
              movlw     .8           ; 1 set numero de repeticion
        movwf     wait_cnt     ; 1 |
PLoop1  nop                   ; 1 clear watchdog
        decfsz    wait_cnt, 1  ; 1 + (1) es el tiempo 0  ?
        goto      PLoop1      ; 2 no, loop
        return                ; 2+2 Fin.
;*********************************************************************
;***************delay bit tx*****************************************
delay_bit_1
              movlw     .7        ; 1 set numero de repeticion
        movwf     wait_cnt     ; 1 |
PLoop2  nop                ; 1 clear watchdog
        decfsz    wait_cnt,1  ; 1 + (1) es el tiempo 0  ?
        goto      PLoop2    ; 2 no, loop
        return
;*********************************************************************
;****** inicializacion portx ****************************************
init_0
       movlw  0x3f
       movwf  PORTB
       movlw  0x86
       movwf  FSR
       movlw  0x43
       movwf  INDF
       movlw  0x03
       movwf  PORTA
       movlw  0x85
       movwf  FSR
       movlw  0x10
       movwf  INDF
       return
init_bc
       MOVLW 0x81
       MOVWF FSR
       MOVLW 0xef
       MOVWF INDF
       bcf INTCON,RBIF
       BSF   INTCON, GIE             ; enable general interupts
    BSF   INTCON, RBIE       ; enable interrupt on change
       return
;*********************************************************************************
channel
       movf R14,w            ; el registro de configuracion de canales
       movwf PORTA           ; es introducido al puerto A
       movf TRISA
       movwf FSR
       movlw 0x10            ; RB4 como entrada las demas salidas
       movwf INDF
       MOVLW 0xEF
       MOVWF PORTB                  ; !!Bank!! PORTB - TRISB
       MOVLW TRISB
       MOVWF FSR
       MOVLW 0x41            ;RB7, RB0 entradas y los demas salidas
       MOVWF INDF
       movf Txbuff,W
       RETURN
;*******************************  MODO 0 ***************************************
modo_0
       MOVLW TRISA
```

```
        MOVWF FSR
        MOVLW 0xFF                      ;se escribe  ff en el timer
        MOVWF TMR0                      ;
        BCF INTCON,T0IF                 ;se limpia la bandera del timer overflow
        BSF INDF,3                      ; deja correr libre el zz clock
espera_1
        BTFSC PORTA,4                   ; espera la interrupcion del triger
        GOTO trigger_event_0
        BTFSC INTCON,T0IF               ; espera que expire el timer
        GOTO trigger_event_0
        GOTO espera_1

trigger_event_0
        MOVF R13,W                      ;se provoca un retardo = time_base
        CALL delay
        DECFSZ temp_transf,F            ;retardo = post_trigger low
        GOTO trigger_event_0
        DECFSZ temp_transfA,F           ; retardo = post_trigger high
        goto trigger_event_0
        BCF INDF,3                      ; se detiene el zz clock
        call descarga                   ; subrutina para dedscargar el spock counter
;*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
;***************************** MODO 1 *********************************************
modo_1
        movf R14,W
        movwf var_swap
        MOVLW TRISA
        MOVWF FSR
        movf R5,f
        btfsc STATUS, Z
        goto cruce_cero_1
        MOVLW 0xFF                      ;se escribe  ff en el timer
        MOVWF TMR0                      ;
        BCF INTCON,T0IF                 ;se limpia la bandera del timer overflow
        BSF INDF,3                      ; deja correr libre el zz clock
espera_3
        BTFSC PORTA,4                   ; espera la interrupcion del triger
        GOTO trigger_event_1
        BTFSC INTCON,T0IF               ; espera que expire el timer
        GOTO trigger_event_1
        GOTO espera_3

cruce_cero_1
        bcf    INTCON,INTF
        bsf INTCON, INTE
        BSF INDF,3                      ; deja correr libre el zz clock
espera_4
        BTFSC PORTB,0                   ; espera la interrupcion del triger
        GOTO trigger_event_1
        BTFSC INTCON,INTF               ; espera que expire el timer
        GOTO trigger_event_1
        GOTO espera_4
trigger_event_1
                nop
loop_5  nop
                nop
loop_4  movf R13,w
                call delay
                swapf var_swap,F
                movf var_swap,w
                movwf PORTA
                decfsz temp_transf,F
                goto loop_5
                decfsz temp_transfA,F
                goto loop_4
                bcf    INDF,3
                bcf    INTCON,INTF
                bcf INTCON, INTE
                call descarga
;***************************** MODO 2 *********************************************
modo_2
```

```
        MOVLW TRISA
        MOVWF FSR
        movf R5,f
        btfsc STATUS, Z
        goto cruce_cero_2
        MOVLW 0xFF                  ;se escribe  ff en el timer
        MOVWF TMR0                  ;
        BSF INDF,3

espera_5
        BTFSC PORTA,4               ; espera la interrupcion del triger
        GOTO trigger_event_2
        BTFSC INTCON,T0IF           ; espera que expire el timer
        GOTO trigger_event_2
        GOTO espera_5
cruce_cero_2
        bcf    INTCON,INTF
        bsf INTCON, INTE

        BSF INDF,3                  ; deja correr libre el zz clock
espera_6
        BTFSC PORTB,0               ; espera la interrupcion del triger
        GOTO trigger_event_2
        BTFSC INTCON,INTF           ; espera que expire el timer
        GOTO trigger_event_2
        GOTO espera_6

trigger_event_2
loop_3  bsf INDF,3
              nop
              bcf    INDF,3
              movf R13,w
              call delay
              decfsz temp_transf,f
              goto loop_3
              decfsz temp_transfA,f

              goto loop_3
              bcf    INTCON,INTF
              bcf INTCON, INTE
              call descarga
;**-*-*-*--*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
;*************************** MODO 3 ***********************************************
modo_3
        MOVLW TRISA
        MOVWF FSR
        movf R5,f
        btfsc STATUS, Z
        goto cruce_cero_3
        MOVLW 0xFF                  ;se escribe  ff en el timer
        MOVWF TMR0                  ;

espera_7
        BTFSC PORTA,4               ; espera la interrupcion del triger
        GOTO trigger_event_3
        BTFSC INTCON,T0IF           ; espera que expire el timer
        GOTO trigger_event_3
        GOTO espera_7
cruce_cero_3
        bcf    INTCON,INTF
        bsf INTCON, INTE
        MOVLW TRISA
        MOVWF FSR
        BSF INDF,3                  ; deja correr libre el zz clock
espera_8
        BTFSC PORTB,0               ; espera la interrupcion del triger
        GOTO trigger_event_3
        BTFSC INTCON,INTF           ; espera que expire el timer
        GOTO trigger_event_3
        GOTO espera_8
```

```
trigger_event_3
loop_6  bsf INDF,3
               swapf var_swap,F
               movf var_swap,w
               movwf PORTA
               nop
               nop
               bcf    INDF,3
               movf R13,w
               call delay
               decfsz temp_transf,f
               goto loop_6
               decfsz temp_transfA,f
               goto loop_6
               bcf    INTCON,INTF
               bcf INTCON, INTE
               call descarga
;*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
;************************* MODO 4  ********************************************
modo_4

;*****************************DELAY ********************************************
delay
               MOVWF delay_var
lazo_01
               DECFSZ delay_var,F
               GOTO lazo_01
               RETURN
;*******************************************************************************************
       org 0x350
       goto modo_0
       goto modo_1
       goto modo_2
       goto modo_3
       goto modo_4

       org  0x380
       goto 0x00
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto ack_idle
       goto exit_idle
       goto exit_idle
       goto ack_idle
       goto exit_idle
       goto exit_idle
       goto ack_idle
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto exit_idle
       goto ack_idle
       goto ack_idle
```

```
goto ack_idle
goto source_adress            ;  #
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto inc_reg                  ; +
goto ack_idle
goto dec_reg                  ; -
goto ack_idle
goto ack_idle
goto niblle_0                 ; 0
goto niblle_1                 ; 1
goto niblle_2                 ; 2
goto niblle_3                 ; 3
goto niblle_4                 ; 4
goto niblle_5                 ; 5
goto niblle_6                 ; 6
goto niblle_7                 ; 7
goto niblle_8                 ; 8
goto niblle_9                 ; 9
goto ack_idle
goto ack_idle
goto get_counter              ; <
goto ack_idle
goto spock_reg                ; >
goto ack_idle
goto load_adress              ; @
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle

goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto dump                     ; S
goto trace                    ; T
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto clear_r0                 ; [
goto ack_idle
goto niblle_0                 ; ]
goto ack_idle
goto ack_idle
goto ack_idle
goto niblle_a                 ; a
goto niblle_b                 ; b
goto niblle_c                 ; c
goto niblle_d                 ; d
goto niblle_e                 ; e
goto niblle_f                 ; f
goto ack_idle
```

```
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto load_r2_r0                     ; l
goto ack_idle
goto next_adress            ; n
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto R0_R1                  ; s
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto ack_idle
goto exit_idle
end
```

ANEXO 4: CODIGO EN VISUAL BASIC

```
Option Explicit
Dim Ack As String
Dim PLD_Counter As Double
Dim Channel(2) As cCanal
Dim OperationMode As Integer
Dim Cadena As String
Dim TI As Double, Tf As Double, Tt As Double
Dim ArrayTem(MrySize) As Double
Dim Promedio As Integer

Private Type RndFile
   VoltDiv As Integer
   TimeDiv As Integer
   Vmax As Integer
   Vmin As Integer
   sSamples As String * MrySize
End Type

Dim I As Integer

Dim mArchivo As RndFile

Private Sub chkRunStop_Click()
   Dim I As Integer
   Dim U As Integer
   Dim A1 As Double, A2 As Double, A3 As Double, A4 As Double
   Promedio = 10
   If chkRunStop = vbChecked Then
      chkRunStop.Caption = "Detener"

      For I = 1 To Promedio
         'Se envía el Trace y se espera devolución del caracter
         'y cuatro caracteres mas
         Acknowlege (">")
         MSComm1.Output = "T"
         TI = Timer
         Do
            Tf = Timer
            If Tf - TI >= 1 Then
               MsgBox "Error en la descarga"
               Exit Sub
            End If
             DoEvents
         Loop While MSComm1.InBufferCount < 5
         Cadena = MSComm1.Input

         A1 = Asc(Mid$(Cadena, 2, 1))
         A2 = Asc(Mid$(Cadena, 3, 1))
         A3 = Asc(Mid$(Cadena, 4, 1))
         A4 = Asc(Mid$(Cadena, 5, 1))
         PLD_Counter = A2 * 256 + _
                     A3 - 1

         'Se envía S para descargar los datos
         'se devuelve S seguido de todos los datos
         MSComm1.Output = "S"
         TI = Timer
         Do
            Tf = Timer
            If Tf - TI >= 4 Then
               MsgBox "Error en la descarga"
               Exit Sub
            End If
            DoEvents
         Loop While MSComm1.InBufferCount < PLD_Counter

         Cadena = MSComm1.Input
         U = Len(Cadena)
         Cadena = Mid$(Cadena, 3, Len(Cadena))
         Sumatoria
```

```
        Next I
        ToString Promedio

        mArchivo.sSamples = Cadena
        mArchivo.VoltDiv = Channel(mCanal).VoltIndex
        mArchivo.TimeDiv = Channel(mCanal).TimeIndex
        With Channel(mCanal)
           .sMuestras = mArchivo.sSamples
           mArchivo.Vmax = .VoltMax
           mArchivo.Vmin = .VoltMin
           .OffsetY = udOffset(mCanal).Value
        End With
        Graficar_1
        mnuGuardar.Enabled = True
        MsgBox "Ya"        'Linea de prueba
                    'Debe borrarse en el programa final
    Else
       chkRunStop.Caption = "Graficar"
    End If
End Sub

Private Sub Command1_Click()
    Dim Cade As String
    Dim U As Double

    Cade = MSComm1.Input
    Cade = Mid$(Cade, 2, Len(Cade))
    U = Len(Cade)
    mArchivo.sSamples = Cade
    mArchivo.VoltDiv = Channel(mCanal).VoltIndex
    mArchivo.TimeDiv = Channel(mCanal).TimeIndex
    With Channel(mCanal)
       .sMuestras = mArchivo.sSamples
       mArchivo.Vmax = .VoltMax
       mArchivo.Vmin = .VoltMin
       .OffsetY = udOffset(mCanal).Value
    End With
    Graficar_1
    mnuGuardar.Enabled = True

End Sub



Private Sub optDominioF_Click()
    Channel(0).Domain = False
    Channel(1).Domain = False
    Channel(2).Domain = False
    Graficar_1
End Sub

Private Sub optDominioT_Click()
    Channel(0).Domain = True
    Channel(1).Domain = True
    Channel(2).Domain = True
    Graficar_1
End Sub

Private Sub optDual_Click()
    frmCHA.Enabled = True
    frmCHB.Enabled = True
    linXRef(0).Visible = True
    linXRef(1).Visible = True
    Channel(1).TimeIndex = Channel(0).TimeIndex
    Channel(0).OffsetX = 5
    Channel(1).OffsetX = 5
    Graficar_1
End Sub

Private Sub optResta_Click()
```

```
    Dim Resta(MrySize) As Double
    frmCHA.Enabled = True
    frmCHB.Enabled = False
    mCanal = 2
    linXRef(0).Visible = True
    linXRef(1).Visible = False
    If Channel(0).TimeIndex = Channel(1).TimeIndex Then
       Channel(mCanal).CH = 0
       Channel(mCanal).VoltIndex = Channel(0).VoltIndex
       Channel(mCanal).TimeIndex = Channel(0).TimeIndex
       Channel(0).Get_Matriz Resta
       Channel(1).Get_Matriz Resta, True
       Channel(2).Let_Matriz Resta
       Channel(2).OffsetY = udOffset(0).Value
       Graficar_1
    End If
End Sub

Private Sub optSuma_Click()
    Dim Suma(MrySize) As Double
    frmCHA.Enabled = True
    frmCHB.Enabled = False
    mCanal = 2
    linXRef(0).Visible = True
    linXRef(1).Visible = False
    If Channel(0).TimeIndex = Channel(1).TimeIndex Then
       Channel(0).Get_Matriz Suma
       Channel(1).Get_Matriz Suma
       Channel(2).Let_Matriz Suma
       Channel(mCanal).CH = 0
       Channel(mCanal).VoltIndex = Channel(0).VoltIndex
       Channel(mCanal).TimeIndex = Channel(0).TimeIndex
       udEscala(1).Value = udEscala(0).Value
       udOffset(1).Value = udOffset(0).Value
       Channel(2).OffsetY = udOffset(0).Value
       Graficar_1
    End If
End Sub

Private Sub trmRefresh_Timer()
    Dim C As String
    C = Channel(mCanal).TimeIndex
    trmRefresh.Interval = 1000

    If C = 17 Then
       C = 0
       chkModo1.Value = vbUnchecked
    ElseIf C >= 19 And C <= 23 Then
       C = C - 18
       chkModo1.Value = vbUnchecked
    ElseIf C = 15 Then
       If chkModo1.Value = vbChecked Then
          C = 6
       Else
          MsgBox "Tiempo valido solamente en modo libre", vbCritical
          chkGetData.Value = vbUnchecked
          chkGetData_Click
          trmRefresh.Interval = 10
          Exit Sub
       End If
    Else
       MsgBox "Tiempo de muestreo no valido", vbCritical
       chkGetData.Value = vbUnchecked
       chkGetData_Click
       trmRefresh.Interval = 10
       Exit Sub
    End If

    MSComm1.Output = C
End Sub
```

```
Private Sub chkModo1_Click()
   If chkModo1.Value = vbChecked Then
      Ventana = 1
   Else
      Ventana = 5
   End If
End Sub

Private Sub cmdShift_Click(Index As Integer)
   Dim I As Integer
   Dim Max As Integer
   Dim Idx As Integer

   Max = IIf(optDual.Value = True, 1, 0)
   For I = 0 To Max
      Idx = IIf(OperationMode = 2, I, mCanal)
      Select Case Index
         Case 0
            Channel(Idx).OffsetX = Channel(mCanal).OffsetX - 50
         Case 1
            Channel(Idx).OffsetX = Channel(mCanal).OffsetX - 1
         Case 2
            Channel(Idx).OffsetX = Channel(mCanal).OffsetX + 1
         Case 3
            Channel(Idx).OffsetX = Channel(mCanal).OffsetX + 50
      End Select
   Next I
   Graficar_1
End Sub

Private Sub Form_Load()

   mCanal = 0
   Set Channel(0) = New cCanal
   Set Channel(1) = New cCanal
   Set Channel(2) = New cCanal

   Channel(0).OffsetX = 5
   Channel(1).OffsetX = 5
   Channel(0).Color = &HFFFF& 'Color amarillo
   Channel(1).Color = &HFF00& 'Color verde
   Channel(2).Color = &HFFFF&

   mnuGuardar.Enabled = False
   'Prepara el puerto serial
   MousePointer = 0
   MSComm1.CommPort = 2
   MSComm1.InputLen = 0
   MSComm1.PortOpen = True
   MSComm1.RThreshold = 1

   Cadena = "[5]@[80]s[6]@[7f]s[7]@[ff]s[8]@[0]s[b]@[0]s[c]@[01]s[d]@[ff]s[e]@[9]s"
   Acknowlege (Cadena)

   If chkModo1.Value = vbChecked Then
      Ventana = 1
   Else
      Ventana = 5
   End If
   linXRef(0).Visible = True
   linXRef(1).Visible = False
   linXRef(0).BorderColor = &HFFFF&
End Sub

Private Sub mnuAbrir_Click()
   Dim recLenght As Long
   Dim FileName As String
   Dim FileNum As Integer
```

```
'*** SE ABRE EL ARCHCIVO***
   On Error Resume Next
   cmmDialog.Filter = "Data Files (*.rnd)|*.rnd|"
   cmmDialog.ShowOpen

   If Err.Number = 0 Then
      'Determina el numero de bytes en un objeto archivo
      recLenght = LenB(mArchivo)
      FileNum = FreeFile
      FileName = cmmDialog.FileName
      'Abrir Archivo para lectura
      Open FileName For Random Access Read As FileNum _
         Len = recLenght
      Get #FileNum, 1, mArchivo

      With Channel(mCanal)
         .CH = mCanal
         .sMuestras = mArchivo.sSamples
         .VoltIndex = mArchivo.VoltDiv  'Se recupera volt/div al que fue muestreada la señal
         .TimeIndex = mArchivo.TimeDiv  'Se recupera time/div al que fue muestreada la señal
         .OffsetY = udOffset(mCanal).Value
      End With
      Graficar_1
      Close #FileNum
   End If
End Sub

Private Sub MSComm1_OnComm()
   'If MSComm1.InBufferCount = Spock_Counter Then
   '    Ack = MSComm1.Input
   'End If
   'If MSComm1.InBufferCount = MrySize + 1 Then
   '    mArchivo.sSamples = MSComm1.Input
   '    mArchivo.VoltDiv = Channel(mCanal).VoltDiv.Indice
   '    mArchivo.TimeDiv = Channel(mCanal).TimeDiv.Indice
   '    With Channel(mCanal)
   '        .sMuestras = mArchivo.sSamples
           '.InitialTime = .TimeValue 'El tiempo inicial se establece como el tiempo de referencia
   '        mArchivo.Vmax = .VoltMax
   '        mArchivo.Vmin = .VoltMin
   '        .Offset = mCanal
   '        .Graficar
   '    End With

   '    mnuGuardar.Enabled = True
   'End If
End Sub

Private Sub mnuGuardar_Click()
   Dim recLenght As Long
   Dim FileName As String
   Dim FileNum As Integer

'*** SE GUARDA EL ARCHCIVO***
   On Error Resume Next
   cmmDialog.Filter = "Data Files (*.rnd)|*.rnd|"
   cmmDialog.ShowSave

   If Err.Number = 0 Then
      'Determina el numero de bytes en un objeto archivo
      recLenght = LenB(mArchivo)
      FileNum = FreeFile
      FileName = cmmDialog.FileName
      'Abrir Archivo para escritura
      Open FileName For Random Access Write As FileNum _
         Len = recLenght

      Put #FileNum, 1, mArchivo
```

```
        Close #FileNum
    End If
End Sub

Private Sub mnuSalir_Click()
    End
End Sub

Private Sub optCanalA_Click()
    frmCHA.Enabled = True
    frmCHB.Enabled = False
    mCanal = 0
    udTimeDiv.Value = Channel(mCanal).TimeIndex
    udOffset(mCanal).Value = Channel(mCanal).OffsetY
    linXRef(mCanal).BorderColor = &HFFFF&
    linXRef(0).Visible = True
    linXRef(1).Visible = False
    Graficar_1
End Sub

Private Sub optCanalB_Click()
    frmCHA.Enabled = False
    frmCHB.Enabled = True
    mCanal = 1
    udTimeDiv.Value = Channel(mCanal).TimeIndex
    udOffset(mCanal).Value = Channel(mCanal).OffsetY
    linXRef(mCanal).BorderColor = &HFF00&
    linXRef(0).Visible = False
    linXRef(1).Visible = True
    Graficar_1
End Sub

Private Sub udEscala_DownClick(Index As Integer)
    mCanal = IIf(optDual.Value = True, Index, mCanal)
    Channel(mCanal).CH = Index
    Channel(mCanal).VoltIndex = udEscala(Index).Value
    Channel(mCanal).OffsetY = udOffset(Index).Value
    Graficar_1
End Sub

Private Sub udEscala_UpClick(Index As Integer)
    udEscala_DownClick (Index)
End Sub

Private Sub udOffset_DownClick(Index As Integer)
    mCanal = IIf(optDual.Value = True, Index, mCanal)
    Channel(mCanal).OffsetY = udOffset(Index).Value
    linXRef(Index).Y1 = Channel(mCanal).OffsetY
    linXRef(Index).Y2 = Channel(mCanal).OffsetY
    Graficar_1
End Sub

Private Sub udOffset_UpClick(Index As Integer)
    udOffset_DownClick (Index)
End Sub

Private Sub udTimeDiv_DownClick()
    Channel(mCanal).TimeIndex = udTimeDiv.Value
    Graficar_1
End Sub

Private Sub udTimeDiv_UpClick()
    udTimeDiv_DownClick
End Sub


Public Sub Graficar_1()
    Dim I As Integer

    frmScope.picPantalla.DrawWidth = 1
```

```
    frmScope.picPantalla.Cls

    If optDual.Value = True Then
        For I = 0 To 1
            Channel(I).Graficar
        Next I
    Else
        Channel(mCanal).Graficar
    End If
End Sub

Public Sub Acknowlege(Caracter As String)
    Dim A As String
    Dim I As Integer
    I = Len(Caracter)
    For I = 1 To Len(Caracter)
        A = Mid$(Caracter, I, 1)
        MSComm1.Output = A

        TI = Timer
        Do
            Tf = Timer
            Tt = Tf - TI
            If Tt >= 1 Then
                MsgBox "Verifique que el hardware esta conectado"
                Exit For
            End If
            DoEvents
        Loop While MSComm1.InBufferCount = 0
        Ack = MSComm1.Input
        If Ack <> A Then
            MsgBox "Caracter no fue devuelto"
        End If
    Next I
End Sub


Public Sub Sumatoria()
    Dim I As Double
    Dim J As Double
    J = Len(Cadena)
    For I = 0 To MrySize
        If I >= J Then Exit For
        ArrayTem(I) = ArrayTem(I) + Asc(Mid$(Cadena, I + 1, 1))
    Next I
End Sub

Public Sub ToString(P As Integer)
    Dim I As Double
    Cadena = ""
    For I = 0 To MrySize
        Cadena = Cadena & Chr(ArrayTem(I) / P)
    Next I
End Sub
```

```
'Definicion de Clase para el canal
Option Explicit
Dim I As Integer, J As Integer
Private MuestrasT(MrySize) As Double
Private FourierRin(MrySize) As Double
Private FourierIin(MrySize) As Double
Private FourierRout(MrySize) As Double
Private FourierIout(MrySize) As Double

Dim mXb As Double
Private mXStep As Double
Private mOffsetX As Double
Private mOffsetY As Integer
Private mOffsetY1 As Integer
Private mDomain As Boolean
Private mColor As Double
Dim NoMuestras As Long
Dim mfourierMax As Double
Dim mEmpty As Boolean
Dim mMuestras As String
Dim mCH As Integer

Private StrTimeDiv(30) As String
Private mTimeDiv(30) As Double
Private mTimeIndex As Integer
Private mInitialTime As Double

Private mVoltMax As Integer
Private mVoltMin As Integer
Private strVoltDiv(12) As String
Private mVoltDiv(12) As Double
Private mVoltIndex As Integer

'*****************************************************
'PROPIEDADES Y FUNCIONES LA CLASE
Public Property Let sMuestras(ByVal Sm As String)
   mMuestras = Sm
   GetSamples MuestrasT, Sm
   mVoltMax = FindMax(MuestrasT)
   'mVoltMin = FindMin(MuestrasT)
   AplicarTransformada
   mfourierMax = FindMax(FourierRout, 1)
   mEmpty = False
   mXb = 0
End Property

Public Property Get sMuestras() As String
   sMuestras = mMuestras
End Property

Public Property Let Domain(ByVal D As Boolean)
   mDomain = D
End Property

Public Property Get pEmpty() As Boolean
   pEmpty = mEmpty
End Property

Public Property Let Color(ByVal C As Double)
   mColor = C
End Property

Public Property Let CH(ByVal C As Integer)
   mCH = C
End Property

'*****************************************************
'PROPIEDADES Y FUNCIONES PARA EL TIEMPO POR DIVISIÓN
Public Property Get TimeIndex() As Integer
   TimeIndex = mTimeIndex
```

```
    frmScope.txtTimeDiv.Text = StrTimeDiv(mTimeIndex)
    frmScope.udTimeDiv.Value = mTimeIndex
End Property

Public Property Let TimeIndex(ByVal I As Integer)
    mXb = IIf(mXb = 0, mTimeDiv(I), mTimeDiv(mTimeIndex))
    mTimeIndex = I
    mXStep = mXStep * (mXb / mTimeDiv(mTimeIndex))
    frmScope.txtTimeDiv.Text = StrTimeDiv(mTimeIndex)
    frmScope.udTimeDiv.Value = mTimeIndex
End Property

'Public Property Let InitialTime(ByVal t As Double)
'    mInitialTime = t
'End Property

'Public Property Get InitialTime() As Double
'    InitialTime = mInitialTime
'End Property

'Public Function TimeValue() As Double
    'La constante 0.0005 es una constante que se adiciona debido
    'a que la ventana de captura de 10 muestras tarda 10uS * 50 pixeles
'    If frmScope.chkModo1.Value = vbUnchecked Then
        'TimeValue = mTimeDiv(mTimeIndex) + 0.0005 - 0.000056
'    Else
        'TimeValue = mTimeDiv(mTimeIndex)
'    End If
'End Function

'Private Function xStep() As Double
'    xStep = mInitialTime / mTimeDiv(mTimeIndex)
'End Function

Private Sub Load_TimeDivValues()
    StrTimeDiv(0) = "1 nS/div"
    StrTimeDiv(1) = "2 nS/div"
    StrTimeDiv(2) = "5 nS/div"
    StrTimeDiv(3) = "10 nS/div"
    StrTimeDiv(4) = "20 nS/div"
    StrTimeDiv(5) = "50 nS/div"
    StrTimeDiv(6) = "100 nS/div"
    StrTimeDiv(7) = "200 nS/div"
    StrTimeDiv(8) = "500 nS/div"
    StrTimeDiv(9) = "1 uS/div"
    StrTimeDiv(10) = "2 uS/div"
    StrTimeDiv(11) = "5 uS/div"
    StrTimeDiv(12) = "10 uS/div"
    StrTimeDiv(13) = "20 uS/div"
    StrTimeDiv(14) = "50 uS/div"
    StrTimeDiv(15) = "100 uS/div"   'Escala free
    StrTimeDiv(16) = "200 uS/div"
    StrTimeDiv(17) = "500 uS/div"   '1o Escala
    StrTimeDiv(18) = "1 mS/div"
    StrTimeDiv(19) = "2.5 mS/div"   '2o Escala  '"2 mS/div"
    StrTimeDiv(20) = "5 mS/div"     '3o Escala
    StrTimeDiv(21) = "12.5 mS/div"  '4o Escala  '"10 mS/div"
    StrTimeDiv(22) = "25.5 mS/div"  '5o Escala  '"20 mS/div"
    StrTimeDiv(23) = "50.1 mS/div"  '6o Escala
    StrTimeDiv(24) = "100 mS/div"
    StrTimeDiv(25) = "200 mS/div"
    StrTimeDiv(26) = "500 mS/div"
    StrTimeDiv(27) = "1 S/div"
    StrTimeDiv(28) = "2 S/div"
    StrTimeDiv(29) = "5 S/div"

    mTimeDiv(0) = 0.000000001       '"1 nS/div"
    mTimeDiv(1) = 0.000000002       '"2 nS/div"
    mTimeDiv(2) = 0.000000005       '"5 nS/div"
    mTimeDiv(3) = 0.00000001        '"10 nS/div"
```

```
    mTimeDiv(4) = 0.00000002       '"20 nS/div"
    mTimeDiv(5) = 0.00000005       '"50 nS/div"
    mTimeDiv(6) = 0.0000001        '"100 nS/div"
    mTimeDiv(7) = 0.0000002        '"200 nS/div"
    mTimeDiv(8) = 0.0000005        '"500 nS/div"
    mTimeDiv(9) = 0.000001         '"1 uS/div"
    mTimeDiv(10) = 0.000002        '"2 uS/div"
    mTimeDiv(11) = 0.000005        '"5 uS/div"
    mTimeDiv(12) = 0.00001         '"10 uS/div"
    mTimeDiv(13) = 0.00002         '"20 uS/div"
    mTimeDiv(14) = 0.00005         '"50 uS/div"
    mTimeDiv(15) = 0.0001          '"100 uS/div"
    mTimeDiv(16) = 0.0002          '"200 uS/div"
    mTimeDiv(17) = 0.0005          '"500 uS/div"
    mTimeDiv(18) = 0.001           '"1 mS/div"
    mTimeDiv(19) = 0.0025          '"2 mS/div"
    mTimeDiv(20) = 0.005           '"5 mS/div"
    mTimeDiv(21) = 0.0125          '"10 mS/div"
    mTimeDiv(22) = 0.0255          '"20 mS/div"
    mTimeDiv(23) = 0.051           '"50 mS/div"
    mTimeDiv(24) = 0.1             '"100 mS/div"
    mTimeDiv(25) = 0.2             '"200 mS/div"
    mTimeDiv(26) = 0.5             '"500 mS/div"
    mTimeDiv(27) = 1               '"1 S/div"
    mTimeDiv(28) = 2               '"2 S/div"
    mTimeDiv(29) = 5               '"5 S/div"
End Sub
'*******************************************************


'*******************************************************
'PROPIEDADES Y FUNCIONES PARA EL VOLTIOS POR DIVISIÓN
Public Property Get VoltIndex() As Integer
    VoltIndex = mVoltIndex
End Property

Public Property Let VoltIndex(ByVal I As Integer)
    mVoltIndex = I
    frmScope.txtEscala(mCH).Text = strVoltDiv(mVoltIndex)
    frmScope.udEscala(mCH).Value = mVoltIndex
End Property

Public Property Get VoltMax() As Integer
    VoltMax = mVoltMax
End Property

Public Property Get VoltMin() As Integer
    VoltMin = mVoltMin
End Property


Private Sub Load_VoltDivValues()
    strVoltDiv(0) = "10mV/div"
    strVoltDiv(1) = "20mV/div"
    strVoltDiv(2) = "50mV/div"
    strVoltDiv(3) = "0.1V/div"
    strVoltDiv(4) = "0.2V/div"
    strVoltDiv(5) = "0.5V/div"
    strVoltDiv(6) = "1V/div"
    strVoltDiv(7) = "2V/div"
    strVoltDiv(8) = "5V/div"
    strVoltDiv(9) = "10V/div"
    strVoltDiv(10) = "20V/div"
    strVoltDiv(11) = "50V/div"

    mVoltDiv(0) = 0.01
    mVoltDiv(1) = 0.02
    mVoltDiv(2) = 0.05
    mVoltDiv(3) = 0.1
    mVoltDiv(4) = 0.2
```

```
    mVoltDiv(5) = 0.5
    mVoltDiv(6) = 1
    mVoltDiv(7) = 2
    mVoltDiv(8) = 5
    mVoltDiv(9) = 10
    mVoltDiv(10) = 20
    mVoltDiv(11) = 50
End Sub

Private Function VoltEscala() As Double
    VoltEscala = (250 / 255) / mVoltDiv(mVoltIndex)
End Function
'*******************************************************


'PROPIEDADES PARA LA GRAFICA
'*******************************************************
Public Property Get OffsetX() As Double
    OffsetX = mOffsetX
End Property

Public Property Let OffsetX(ByVal X As Double)
    mOffsetX = X
End Property

Public Property Get OffsetY() As Integer
    OffsetY = mOffsetY '- mVoltMax / 2 * VoltEscala
End Property

Public Property Let OffsetY(ByVal I As Integer)
    mOffsetY = I
    mOffsetY1 = mVoltMax / 2 * VoltEscala + mOffsetY
End Property
'*******************************************************


Private Sub Class_Initialize()
    mOffsetY = 150 + 5 '155
    mOffsetX = 5
    mXStep = 1

    Load_TimeDivValues
    mTimeIndex = 17
    mInitialTime = mTimeDiv(mTimeIndex)

    Load_VoltDivValues
    mVoltIndex = 6
    mDomain = True
    mEmpty = True
End Sub

Public Sub Graficar()
    If mEmpty = False Then
        If mDomain = True Then
            Plot MuestrasT, mOffsetX, mXStep, mOffsetY1, VoltEscala, mColor, MrySize \ Ventana
        Else
            Plot FourierRout, mOffsetX, mXStep, 255, 250 / mfourierMax, mColor, NoMuestras
        End If
    End If
End Sub

Private Sub AplicarTransformada()
    Dim J As Long
    NoMuestras = MrySize \ Ventana
    For J = 0 To NoMuestras
        FourierRin(J) = MuestrasT(J) / 51
    Next J
    NoMuestras = Agregar_Ceros(FourierRin, NoMuestras)
    FFTDouble NoMuestras, False, FourierRin(0), FourierIin(0) _
        , FourierRout(0), FourierIout(0)
```

```
    'FourierTransform NoMuestras, FourierRin, FourierIin _
        , FourierRout, FourierIout

    For J = 0 To NoMuestras
        FourierRout(J) = Sqr(FourierRout(J) ^ 2 + FourierIout(J) ^ 2)
    Next J
End Sub


Public Sub Get_Matriz(ByRef Matriz() As Double, Optional Sig As Boolean)
    Dim J As Long
    Dim Signo As Integer

    Signo = IIf(Sig = False, 1, -1)
    For J = 0 To MrySize \ Ventana
        Matriz(J) = Abs(Matriz(J) + MuestrasT(J) * Signo)
    Next J
End Sub

Public Sub Let_Matriz(ByRef Matriz() As Double)
    Dim J As Long

    For J = 0 To MrySize \ Ventana
        MuestrasT(J) = Matriz(J)
    Next J
    mVoltMax = FindMax(MuestrasT)
    'mVoltMin = FindMin(MuestrasT)
    AplicarTransformada
    mfourierMax = FindMax(FourierRout, 1)
    mEmpty = False
    mXb = 0
End Sub

'Definicion de Clase para el canal
Option Explicit
Dim I As Integer, J As Integer
Private MuestrasT(MrySize) As Double
Private FourierRin(MrySize) As Double
Private FourierIin(MrySize) As Double
Private FourierRout(MrySize) As Double
Private FourierIout(MrySize) As Double

Private mpX1 As Double
Private mOffset As Integer
Private mDomain As Boolean
Private mColor As Double
Dim NoMuestras As Long
Dim mfourierMax As Double
Dim mEmpty As Boolean
Dim mMuestras As String
Dim mEmpty1 As Boolean
Dim mMuestras1 As String

Private StrTimeDiv(30) As String
Private mTimeDiv(30) As Double
Private mTimeIndex As Integer
Private mInitialTime As Double

Private mVoltMax As Integer
Private mVoltMin As Integer
Private strVoltDiv(12) As String
Private mVoltDiv(12) As Double
Private mVoltIndex As Integer

'******************************************************
'PROPIEDADES Y FUNCIONES LA CLASE
Public Property Let sMuestras(ByVal Sm As String)
    mMuestras = Sm
    mEmpty = False
End Property
```

```
Public Property Let sMuestras1(ByVal Sm As String)
   mMuestras1 = Sm
   mEmpty1 = False
End Property

Public Property Let Suma(ByVal Sum As Boolean)
   If (mEmpty = False And mEmpty1 = False) Then
      Add MuestrasT, mMuestras, mMuestras1, Sum
      mVoltMax = FindMax(MuestrasT)
      mVoltMin = FindMin(MuestrasT)
      AplicarTransformada
      mfourierMax = FindMax(FourierRout, 1)
   End If
End Property


Public Property Let Domain(ByVal D As Boolean)
   mDomain = D
End Property

Public Property Get pEmpty() As Boolean
   pEmpty = mEmpty
End Property

Public Property Let Color(ByVal C As Double)
   mColor = C
End Property

'*******************************************************
'PROPIEDADES Y FUNCIONES PARA EL TIEMPO POR DIVISIÓN
Public Property Get TimeIndex() As Integer
   TimeIndex = mTimeIndex
   frmScope.txtTimeDiv.Text = StrTimeDiv(mTimeIndex)
End Property

Public Property Let TimeIndex(ByVal I As Integer)
   mTimeIndex = I
   frmScope.txtTimeDiv.Text = StrTimeDiv(mTimeIndex)
   frmScope.udTimeDiv.Value = mTimeIndex
End Property

Public Property Let InitialTime(ByVal t As Double)
   mInitialTime = t
End Property

Public Property Get InitialTime() As Double
   InitialTime = mInitialTime
End Property

Public Function TimeValue() As Double
   'La constante 0.0005 es una constante que se adiciona debido
   'a que laventana de captura de 10 muestras tarda 10uS * 50 pixeles
   If frmScope.chkModo1.Value = vbUnchecked Then
      TimeValue = mTimeDiv(mTimeIndex) + 0.0005 - 0.000056
   Else
      TimeValue = mTimeDiv(mTimeIndex)
   End If
End Function

Private Function xStep() As Double
   xStep = mInitialTime / mTimeDiv(mTimeIndex)
End Function

Private Sub Load_TimeDivValues()
   StrTimeDiv(0) = "1 nS/div"
   StrTimeDiv(1) = "2 nS/div"
   StrTimeDiv(2) = "5 nS/div"
   StrTimeDiv(3) = "10 nS/div"
   StrTimeDiv(4) = "20 nS/div"
```

```
    StrTimeDiv(5) = "50 nS/div"
    StrTimeDiv(6) = "100 nS/div"
    StrTimeDiv(7) = "200 nS/div"
    StrTimeDiv(8) = "500 nS/div"
    StrTimeDiv(9) = "1 uS/div"
    StrTimeDiv(10) = "2 uS/div"
    StrTimeDiv(11) = "5 uS/div"
    StrTimeDiv(12) = "10 uS/div"
    StrTimeDiv(13) = "20 uS/div"
    StrTimeDiv(14) = "50 uS/div"
    StrTimeDiv(15) = "100 uS/div"   'Escala free
    StrTimeDiv(16) = "200 uS/div"
    StrTimeDiv(17) = "500 uS/div"   '1o Escala
    StrTimeDiv(18) = "1 mS/div"
    StrTimeDiv(19) = "2.5 mS/div"   '2o Escala  '"2 mS/div"
    StrTimeDiv(20) = "5 mS/div"     '3o Escala
    StrTimeDiv(21) = "12.5 mS/div"  '4o Escala  '"10 mS/div"
    StrTimeDiv(22) = "25.5 mS/div"  '5o Escala  '"20 mS/div"
    StrTimeDiv(23) = "50.1 mS/div"  '6o Escala
    StrTimeDiv(24) = "100 mS/div"
    StrTimeDiv(25) = "200 mS/div"
    StrTimeDiv(26) = "500 mS/div"
    StrTimeDiv(27) = "1 S/div"
    StrTimeDiv(28) = "2 S/div"
    StrTimeDiv(29) = "5 S/div"

    mTimeDiv(0) = 0.000000001       '"1 nS/div"
    mTimeDiv(1) = 0.000000002       '"2 nS/div"
    mTimeDiv(2) = 0.000000005       '"5 nS/div"
    mTimeDiv(3) = 0.00000001        '"10 nS/div"
    mTimeDiv(4) = 0.00000002        '"20 nS/div"
    mTimeDiv(5) = 0.00000005        '"50 nS/div"
    mTimeDiv(6) = 0.0000001         '"100 nS/div"
    mTimeDiv(7) = 0.0000002         '"200 nS/div"
    mTimeDiv(8) = 0.0000005         '"500 nS/div"
    mTimeDiv(9) = 0.000001          '"1 uS/div"
    mTimeDiv(10) = 0.000002         '"2 uS/div"
    mTimeDiv(11) = 0.000005         '"5 uS/div"
    mTimeDiv(12) = 0.00001          '"10 uS/div"
    mTimeDiv(13) = 0.00002          '"20 uS/div"
    mTimeDiv(14) = 0.00005          '"50 uS/div"
    mTimeDiv(15) = 0.0001           '"100 uS/div"
    mTimeDiv(16) = 0.0002           '"200 uS/div"
    mTimeDiv(17) = 0.0005           '"500 uS/div"
    mTimeDiv(18) = 0.001            '"1 mS/div"
    mTimeDiv(19) = 0.0025           '"2 mS/div"
    mTimeDiv(20) = 0.005            '"5 mS/div"
    mTimeDiv(21) = 0.0125           '"10 mS/div"
    mTimeDiv(22) = 0.0255           '"20 mS/div"
    mTimeDiv(23) = 0.051            '"50 mS/div"
    mTimeDiv(24) = 0.1              '"100 mS/div"
    mTimeDiv(25) = 0.2              '"200 mS/div"
    mTimeDiv(26) = 0.5              '"500 mS/div"
    mTimeDiv(27) = 1                '"1 S/div"
    mTimeDiv(28) = 2                '"2 S/div"
    mTimeDiv(29) = 5                '"5 S/div"
End Sub
'*****************************************************


'*****************************************************
'PROPIEDADES Y FUNCIONES PARA EL VOLTIOS POR DIVISIÓN
Public Property Get VoltIndex() As Integer
    VoltIndex = mVoltIndex
End Property

Public Property Let VoltIndex(ByVal I As Integer)
    mVoltIndex = I
    frmScope.txtEscala(mCanal).Text = strVoltDiv(mVoltIndex)
    frmScope.udEscala(mCanal).Value = mVoltIndex
```

```
End Property

Public Property Get VoltMax() As Integer
   VoltMax = mVoltMax
End Property

Public Property Get VoltMin() As Integer
   VoltMin = mVoltMin
End Property


Private Sub Load_VoltDivValues()
   strVoltDiv(0) = "10mV/div"
   strVoltDiv(1) = "20mV/div"
   strVoltDiv(2) = "50mV/div"
   strVoltDiv(3) = "0.1V/div"
   strVoltDiv(4) = "0.2V/div"
   strVoltDiv(5) = "0.5V/div"
   strVoltDiv(6) = "1V/div"
   strVoltDiv(7) = "2V/div"
   strVoltDiv(8) = "5V/div"
   strVoltDiv(9) = "10V/div"
   strVoltDiv(10) = "20V/div"
   strVoltDiv(11) = "50V/div"

   mVoltDiv(0) = 0.01
   mVoltDiv(1) = 0.02
   mVoltDiv(2) = 0.05
   mVoltDiv(3) = 0.1
   mVoltDiv(4) = 0.2
   mVoltDiv(5) = 0.5
   mVoltDiv(6) = 1
   mVoltDiv(7) = 2
   mVoltDiv(8) = 5
   mVoltDiv(9) = 10
   mVoltDiv(10) = 20
   mVoltDiv(11) = 50
End Sub

Private Function VoltValue() As Double
   VoltValue = mVoltDiv(mIndice)
End Function

Public Function VoltEscala() As Double
   VoltEscala = (250 / 255) / mVoltDiv(mVoltIndex)
End Function
'*****************************************************


'PROPIEDADES PARA LA GRAFICA
'*****************************************************
Public Property Get pX1() As Double
   pX1 = mpX1
End Property

Public Property Let pX1(ByVal X As Double)
   mpX1 = X
End Property

Public Property Get Offset() As Integer
   Offset = mOffset - mVoltMax / 2 * VoltEscala
End Property

Public Property Let Offset(ByVal Ind As Integer)
   Dim Off As Integer  'Valor del boton de offset

   Off = frmScope.udOffset(Ind).Value
   mOffset = mVoltMax / 2 * VoltEscala + Off

   frmScope.linXRef(Ind).Y1 = Off
```

```
    frmScope.linXRef(Ind).Y2 = Off
End Property
'*****************************************************


Private Sub Class_Initialize()
    mOffset = 150 + 5 '155
    mpX1 = 5

    Load_TimeDivValues
    mTimeIndex = 17
    mInitialTime = mTimeDiv(mTimeIndex)

    Load_VoltDivValues
    mVoltIndex = 6
    mDomain = True
    mEmpty = True
End Sub

Public Sub Graficar()
    If mEmpty = False Then
        If mDomain = True Then
            Plot MuestrasT, mpX1, xStep, mOffset, VoltEscala, mColor, MrySize \ Ventana
        Else
            Plot FourierRout, mpX1, xStep, 255, 250 / mfourierMax, mColor, NoMuestras
        End If
    End If
End Sub

Public Sub AplicarTransformada()
    Dim J As Long
    NoMuestras = MrySize \ Ventana
    For J = 0 To NoMuestras
        FourierRin(J) = MuestrasT(J) / 51
    Next J
    NoMuestras = Agregar_Ceros(FourierRin, NoMuestras)
    FFTDouble NoMuestras, False, FourierRin(0), FourierIin(0) _
        , FourierRout(0), FourierIout(0)
    'FourierTransform NoMuestras, FourierRin, FourierIin _
        , FourierRout, FourierIout

    For J = 0 To NoMuestras
        FourierRout(J) = Sqr(FourierRout(J) ^ 2 + FourierIout(J) ^ 2)
    Next J
End Sub

Private Sub Add(ByRef Matriz() As Double, ByRef mString As String, _
    mString1 As String, ByVal Operation As Boolean)
    'Si Operacion = True es una suma
    'Si Operation = Fase es una resta
    Dim Suma As Integer
    Dim Suma1 As Integer
    Dim Num As Integer
    Dim prom As Integer
    Dim Signo As Integer
    Num = 1
    For J = 0 To MrySize \ Ventana
        If Num >= MrySize - 1 Then '((MrySize \ Ventana) * 10) Then
            Exit For
        End If
        For I = Num To Num + (Ventana - 1)
            Suma = Suma + Asc(Mid$(mString, I, 1))
            Suma1 = Suma1 + Asc(Mid$(mString1, I, 1))
        Next I
        Num = Num + Ventana
        Signo = IIf(Operation = True, 1, -1)
        prom = Suma / Ventana + Suma1 / Ventana * Signo
        Suma = 0
        Suma1 = 0
        Matriz(J) = prom
```

```
    Next J
End Sub

Global Const MrySize = 8192
Global Const MrySize1 = 32768
Global Ventana As Integer
Global Const CH_A = 0
Global Const CH_B = 1
Global mCanal As Integer
Dim I As Integer, J As Integer

'Mediante esta subrutina se puede encontrar facilmente el
'máximo valor de una matriz cualquiera
Public Function FindMax(ByRef Matriz() As Double, Optional A As Long) As Double
    Dim Max As Double

    Max = Matriz(LBound(Matriz) + A)
    For I = LBound(Matriz) + A To MrySize \ Ventana
        If Matriz(I) > Max Then
            Max = Matriz(I)
        End If
    Next I
    FindMax = Max
End Function

'Mediante esta subrutina se puede encontrar facilmente el
'míniimo valor de una matriz cualquiera
Public Function FindMin(ByRef Matriz() As Double) As Double
    Dim Min As Double

    Min = Matriz(LBound(Matriz))
    For I = LBound(Matriz) To MrySize \ Ventana
        If Matriz(I) < Min Then
            Min = Matriz(I)
        End If
    Next I
    FindMin = Min
End Function

Public Function Agregar_Ceros(ByRef Matriz() As Double, ByVal NumSamples As Long)
    'Si la matriz no es potencia de 2 entonces
    'el bucle do-while le agrega ceros hasta que lo sea.
    Dim NoPot2 As Long
    NoPot2 = (NumSamples And (NumSamples - 1))
    Do While NoPot2 <> False
        NumSamples = NumSamples + 1
        Matriz(NumSamples) = 0
        NoPot2 = (NumSamples And (NumSamples - 1))
    Loop
    Agregar_Ceros = NumSamples
End Function

Public Sub GetSamples(ByRef Matriz() As Double, ByRef mString As String)
    Dim Suma As Integer
    Dim Num As Integer
    Dim prom As Integer
    Num = 1
    For J = 0 To MrySize \ Ventana
        If Num >= MrySize - 1 Then '((MrySize \ Ventana) * 10) Then
            Exit For
        End If
        For I = Num To Num + (Ventana - 1)
            Suma = Suma + Asc(Mid$(mString, I, 1))
        Next I
        Num = Num + Ventana
        prom = Suma / Ventana
        Suma = 0
        Matriz(J) = prom
    Next J
End Sub
```

```
Public Sub Plot(ByRef Matriz() As Double, ByVal pX1 As Double, ByVal pX2 As Double, _
   ByVal Offset As Double, ByVal Yheight As Double, ByVal Color As Double, _
   ByVal NumSamples As Double)

   Dim pY1 As Double
   Dim pY2 As Double
   Dim C As Integer

   C = 0
   Do
      'Graficar en dominio en tiempo
      If (C + 1) >= NumSamples Then
         Exit Do
      End If
      pY1 = Matriz(C)
      pY2 = Matriz(C + 1)
      frmScope.picPantalla.Line (pX1, (Offset - (pY1 * Yheight))) _
         -((pX1 + pX2), (Offset - (pY2 * Yheight))), Color
      C = C + 1
      pX1 = pX1 + pX2
   Loop While pX1 < frmScope.picPantalla.ScaleWidth
End Sub
```

ANEXO 5: CODIGO FFT

```
'---------------------------------------------------------------------
' VB FFT Release 2-B
' by Murphy McCauley (MurphyMc@Concentric.NET)
' 10/01/99
'---------------------------------------------------------------------
' About:
' This code is very, very heavily based on Don Cross's fourier.pas
' Turbo Pascal Unit for calculating the Fast Fourier Transform.
' I've not implemented all of his functions, though I may well do
' so in the future.
' For more info, you can contact me by email, check my website at:
' http://www.fullspectrum.com/deeth/
' or check Don Cross's FFT web page at:
' http://www.intersrv.com/~dcross/fft.html
' You also may be intrested in the FFT.DLL that I put together based
' on Don Cross's FFT C code.  It's callable with Visual Basic and
' includes VB declares.  You can get it from either website.
'---------------------------------------------------------------------
' History of Release 2-B:
' Fixed a couple of errors that resulted from me mucking about with
'   variable names after implementation and not re-checking.  BAD ME.
'  --------
' History of Release 2:
' Added FrequencyOfIndex() which is Don Cross's Index_to_frequency().
' FourierTransform() can now do inverse transforms.
' Added CalcFrequency() which can do a transform for a single
'   frequency.
'---------------------------------------------------------------------
' Usage:
' The useful functions are:
' FourierTransform() performs a Fast Fourier Transform on an pair of
'  Double arrays -- one real, one imaginary.  Don't want/need
'  imaginary numbers?  Just use an array of 0s.  This function can
'  also do inverse FFTs.
' FrequencyOfIndex() can tell you what actual frequency a given index
'  corresponds to.
' CalcFrequency() transforms a single frequency.
'---------------------------------------------------------------------
' Notes:
' All arrays must be 0 based (i.e. Dim TheArray(0 To 1023) or
'  Dim TheArray(1023)).
' The number of samples must be a power of two (i.e. 2^x).
' FrequencyOfIndex() and CalcFrequency() haven't been tested much.
' Use this ENTIRELY AT YOUR OWN RISK.
'---------------------------------------------------------------------

Option Explicit
Const Pi = 3.14159265358979

Function NumberOfBitsNeeded(PowerOfTwo As Long) As Byte
   Dim I As Byte
   For I = 0 To 16
      If (PowerOfTwo And (2 ^ I)) <> 0 Then
         NumberOfBitsNeeded = I
         Exit Function
      End If
   Next
End Function


Function IsPowerOfTwo(X As Long) As Boolean
   If (X < 2) Then IsPowerOfTwo = False: Exit Function
   If (X And (X - 1)) = False Then IsPowerOfTwo = True
End Function


Function ReverseBits(ByVal Index As Long, NumBits As Byte) As Long
   Dim I As Byte, Rev As Long

   For I = 0 To NumBits - 1
```

```
        Rev = (Rev * 2) Or (Index And 1)
        Index = Index \ 2
    Next

    ReverseBits = Rev
End Function


Sub FourierTransform(NumSamples As Long, RealIn() As Double, ImageIn() As Double, RealOut() As Double,
ImagOut() As Double, Optional InverseTransform As Boolean = False)
    Dim AngleNumerator As Double
    Dim NumBits As Byte, I As Long, J As Long, K As Long, n As Long, BlockSize As Long, BlockEnd As
Long
    Dim DeltaAngle As Double, DeltaAr As Double
    Dim Alpha As Double, Beta As Double
    Dim TR As Double, TI As Double, AR As Double, AI As Double

    If InverseTransform Then
        AngleNumerator = -2# * Pi
    Else
        AngleNumerator = 2# * Pi
    End If

    If (IsPowerOfTwo(NumSamples) = False) Or (NumSamples < 2) Then
        Call MsgBox("Error in procedure Fourier:" + vbCrLf + " NumSamples is " + CStr(NumSamples) + ", which
is not a positive integer power of two.", , "Error!")
        Exit Sub
    End If

    NumBits = NumberOfBitsNeeded(NumSamples)
    For I = 0 To (NumSamples - 1)
        J = ReverseBits(I, NumBits)
        RealOut(J) = RealIn(I) / 51     '/51 se agrego al codigo original
        ImagOut(J) = ImageIn(I) / 51    '/51 se agrego al codigo original
    Next

    BlockEnd = 1
    BlockSize = 2

    Do While BlockSize <= NumSamples
        DeltaAngle = AngleNumerator / BlockSize
        Alpha = Sin(0.5 * DeltaAngle)
        Alpha = 2# * Alpha * Alpha
        Beta = Sin(DeltaAngle)

        I = 0
        Do While I < NumSamples
            AR = 1#
            AI = 0#

            J = I
            For n = 0 To BlockEnd - 1
                K = J + BlockEnd
                TR = AR * RealOut(K) - AI * ImagOut(K)
                TI = AI * RealOut(K) + AR * ImagOut(K)
                RealOut(K) = RealOut(J) - TR
                ImagOut(K) = ImagOut(J) - TI
                RealOut(J) = RealOut(J) + TR
                ImagOut(J) = ImagOut(J) + TI
                DeltaAr = Alpha * AR + Beta * AI
                AI = AI - (Alpha * AI - Beta * AR)
                AR = AR - DeltaAr
                J = J + 1
            Next

            I = I + BlockSize
        Loop

        BlockEnd = BlockSize
        BlockSize = BlockSize * 2
```

```
    Loop

    If InverseTransform Then
        'Normalize the resulting time samples...
        For I = 0 To NumSamples - 1
            RealOut(I) = RealOut(I) / NumSamples
            ImagOut(I) = ImagOut(I) / NumSamples
        Next
    End If
End Sub


Function FrequencyOfIndex(NumberOfSamples As Long, ByVal Index As Long) As Double
    'Based on IndexToFrequency().  This name makes more sense to me.

    If Index >= NumberOfSamples Then
        FrequencyOfIndex = 0#
        Exit Function
    ElseIf Index <= NumberOfSamples / 2 Then
        FrequencyOfIndex = CDbl(Index) / CDbl(NumberOfSamples)
        Exit Function
    Else
        FrequencyOfIndex = -CDbl(NumberOfSamples - Index) / CDbl(NumberOfSamples)
        Exit Function
    End If
End Function


Sub CalcFrequency(NumberOfSamples As Long, FrequencyIndex As Long, RealIn() As Double, ImagIn() As
Double, RealOut As Double, ImagOut As Double)

    Dim K As Long
    Dim Cos1 As Double, Cos2 As Double, Cos3 As Double, Theta As Double, Beta As Double
    Dim Sin1 As Double, Sin2 As Double, Sin3 As Double

    Theta = 2 * Pi * FrequencyIndex / CDbl(NumberOfSamples)
    Sin1 = Sin(-2 * Theta)
    Sin2 = Sin(-Theta)
    Cos1 = Cos(-2 * Theta)
    Cos2 = Cos(-Theta)
    Beta = 2 * Cos2

    For K = 0 To NumberOfSamples - 2
        'Update trig values
        Sin3 = Beta * Sin2 - Sin1
        Sin1 = Sin2
        Sin2 = Sin3

        Cos3 = Beta * Cos2 - Cos1
        Cos1 = Cos2
        Cos2 = Cos3

        RealOut = RealOut + RealIn(K) * Cos3 - ImagIn(K) * Sin3
        ImagOut = ImagOut + ImagIn(K) * Cos3 + RealIn(K) * Sin3
    Next
End Sub
```