UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DESARROLLO DEL SOFTWARE DE CONTROL PARA UN GENERADOR DE FUNCIONES DE ALTA TENSIÓN

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO ELÉCTRICO ELECTRÓNICO

PRESENTA

GABRIEL ANDRÉS PARRA RODRÍGUEZ

DIRECTOR DE TESIS:

ING. ENRIQUE RAMÓN GÓMEZ ROSAS

México, D. F. 2008

“Mucho sería de estimar un dardo
que supiese diferenciar los
buenos de los ruines”

Agradecimientos

A mis padres, a quienes debo el desarrollo de este trabajo.

A mi hermana Lorena, por su apoyo constante.

A la Universidad Nacional Autónoma de México por brindarme la oportunidad de estudiar y formarme profesionalmente.

Al Instituto de Ingeniería de la UNAM, especialmente a la Coordinación de Instrumentación, por contribuir a mi formación académica al permitirme participar en un proyecto de investigación.

Al Ing. Enrique R. Gómez Rosas por su valiosa dirección, su confianza y apoyo constante.

Al Ing. Rodolfo Peters Lammel por todo el apoyo brindado durante la realización de este trabajo.

Al Dr. Efraín Ovando Shelley por su ayuda y colaboración en este trabajo.

Al M. en C. Mario Flores Guzmán por su ayuda y apoyo en este trabajo.

A mi amigo de toda la carrera Carlos Urquieta por su amistad y apoyo a lo largo de estos seis años.

A todas aquellas personas que de una u otra forma contribuyeron con la realización de este trabajo.

Índice

INTRODUCCIÓN **1**

I.1 **PLANTEAMIENTO DE LA TESIS** 1
I.2 **OBJETIVO DE LA TESIS** 3

CAPÍTULO 1: ANÁLISIS PRELIMINAR **5**

1.1 **SELECCIÓN DEL MICROCONTROLADOR** 6
1.2 **DESCRIPCIÓN DEL MICROCONTROLADOR PIC16F873** 7
1.2.1 Arquitectura interna *7*
1.2.2 Organización de la memoria *9*
1.2.3 Subrutinas *11*
1.2.4 Modos de direccionamiento *13*
1.2.5 Interrupciones *14*
1.3 **SELECCIÓN DEL SOFTWARE Y HARDWARE PARA LA PROGRAMACIÓN DEL MICROCONTROLADOR** ... 16
1.3.1 Software de programación *16*
1.3.2 Hardware de programación *17*

CAPÍTULO 2: METODOLOGÍA DE CÁLCULO **19**

2.1 **PARÁMETRO DE AMPLITUD** 19
2.1.1 Voltaje de modulación AM *20*
2.1.2 Sustraendo V_{C301} *20*
2.1.3 Ciclo de trabajo CCP1 ..... *21*
2.2 **PARÁMETRO DE FRECUENCIA** 26
2.2.1 Selector de frecuencia *27*
2.2.2 Corriente de terminales I_T *27*
2.2.3 Ciclo de trabajo CCP2 ..... *28*
2.3 **PARÁMETRO DE FASE** 32
2.4 **PARÁMETRO DE NÚMERO DE CICLOS** 33
2.4.1 Ajuste del número de ciclos *34*
2.4.2 Tiempos de la señal LOAD *35*
2.5 **RESUMEN** 36

CAPÍTULO 3: DESARROLLO DEL SOFTWARE DE CONTROL **37**

3.1 **DESCRIPCIÓN** 37
3.2 **ESTRUCTURA DEL PROGRAMA PRINCIPAL** 38
3.2.1 Descripción del algoritmo *38*
3.3 **ESTRUCTURA DE LA INTERRUPCIÓN** 39
3.3.1 Descripción del algoritmo *39*
3.4 **SUBRUTINAS** 40
3.4.1 Subrutinas de configuración *40*
3.4.2 Subrutinas de procesamiento *50*

CAPÍTULO 4: DESARROLLO DE LA INTERFAZ DE COMUNICACIÓN **59**
4.1 **DESCRIPCIÓN** 59
4.2 **GFAT CAL** 59
4.2.1 Cadena CCP1 ..... *60*
4.2.2 Cadena CCP2 & RELAY *61*
4.2.3 Cadena FASE *62*
4.2.4 Cadena HAB *62*
4.2.5 Cadena NP *62*
4.2.6 Cadena DELA & DELB & HRT *63*
4.3 **GFAT SER** 65

PRUEBAS **67**
5.1 **PRUEBAS DE SIMULACIÓN** 67
5.2 **PRUEBAS EN HARDWARE** 70

RESULTADOS **75**

CONCLUSIONES **81**

REFERENCIAS BIBLIOGRÁFICAS **83**

APÉNDICE A: DIAGRAMAS DE FLUJO **85**

APÉNDICE B: CÓDIGO FUENTE **99**

APÉNDICE C: SET DE INSTRUCCIONES **109**

APÉNDICE D: REGISTROS ESPECIALES **119**

APÉNDICE E: MPLAB IDE **129**

APÉNDICE F: CÓDIGO ASCII **133**

APÉNDICE G: MONTAJE DE CRISTALES **135**

Introducción

Planteamiento de la tesis

Dentro de las actividades de investigación que realiza el Instituto de Ingeniería de la UNAM, las coordinaciones de Instrumentación y Geotecnia desarrollan conjuntamente una nueva metodología basada en la transmisión de ondas para la obtención de los parámetros elásticos de suelos a bajos niveles de deformación (del orden de 10^{-6} a 10^{-5}%). Para ello, se implementa un sistema de cristales piezoeléctricos en una cámara triaxial, que permita evaluar las velocidades de propagación de las ondas de compresión (Vp) y de corte (Vs). Para generar y medir las ondas de compresión, se seleccionaron discos piezoeléctricos; mientras que en el caso de las ondas de corte, se seleccionaron elementos de flexión.

Al aplicar una tensión de excitación a los discos piezoeléctricos, se produce una contracción o expansión del cristal en función de la polaridad y magnitud de la excitación aplicada y al estar fijo uno de los lados, produce ondas de compresión. Los discos piezoeléctricos tienen un diámetro de 8 mm y un espesor de 2 mm.

Por otra parte, los elementos de flexión están formados por dos placas de cristal piezoeléctrico unidas en contrafase mediante una hoja conductiva. Cuando una de las placas se contrae la otra se expande flexionando al conjunto que, al estar empotrado en uno de sus extremos, produce ondas de corte. Los elementos de flexión tienen una altura de 12 mm, un ancho de 8 mm y un espesor de 0.6 mm.

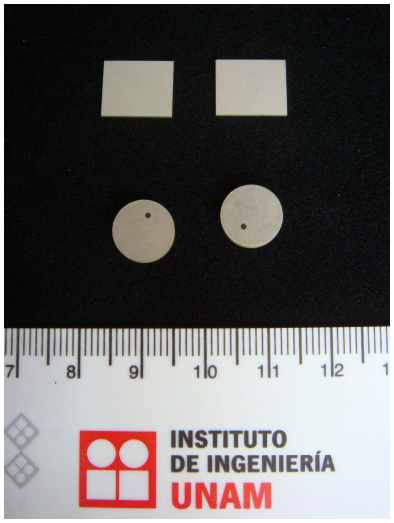


Figura I-1 Elementos de flexión y discos piezoeléctricos

Figura I-2 Operación del disco piezoeléctrico

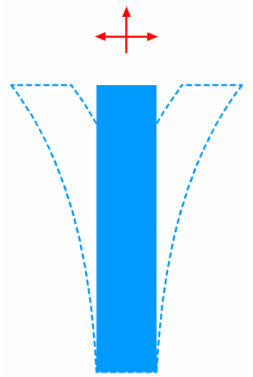


Figura I-3 Operación del elemento de flexión

El sistema de medición (*figura I-4*) se basa en la propiedad de los cristales piezoeléctricos de transformar tensiones eléctricas en deformaciones mecánicas y viceversa, de modo que como se mencionó, pueden emplearse como excitadores o sensores. Para que los cristales se comporten como excitadores, se requiere aplicar tensiones de hasta 140 [V_{RMS}]. Así, la onda generada viajará desde el cristal emisor, a través de la muestra de suelo hasta alcanzar al cristal receptor con lo que puede determinarse la velocidad de propagación al conocer el tiempo transcurrido desde que se aplica la excitación hasta que se tiene respuesta en el receptor.

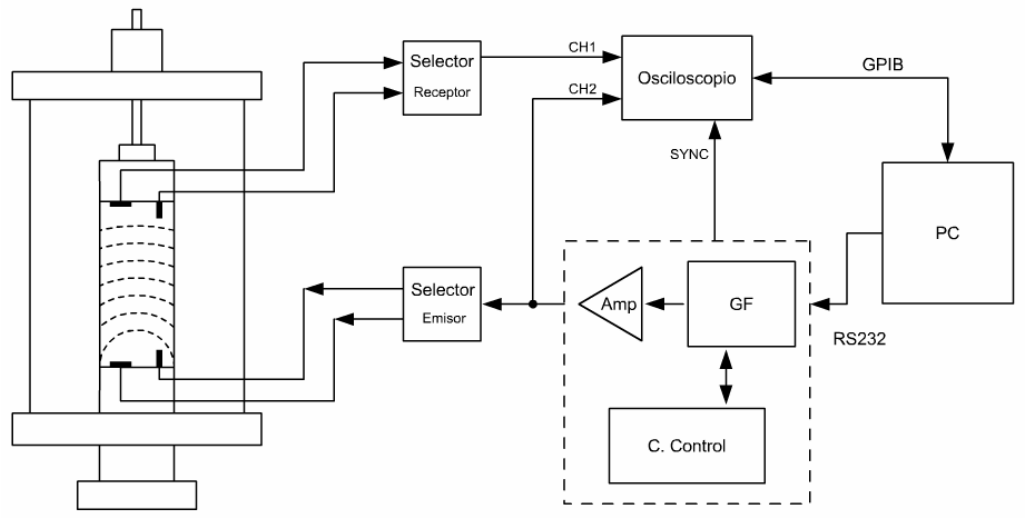


Figura I-4 Instrumentación del sistema de medición

Tradicionalmente, la excitación de los cristales piezoeléctricos consiste en ondas cuadradas; sin embargo, con esta técnica es difícil determinar el instante en el que la onda arriba al cristal receptor. Por otra parte, si la excitación es un tren de ondas sinusoidales, se facilita la detección del tiempo de arribo. Sin embargo, esta técnica requiere de amplificadores lineales con un gran ancho de banda y tensiones altas.

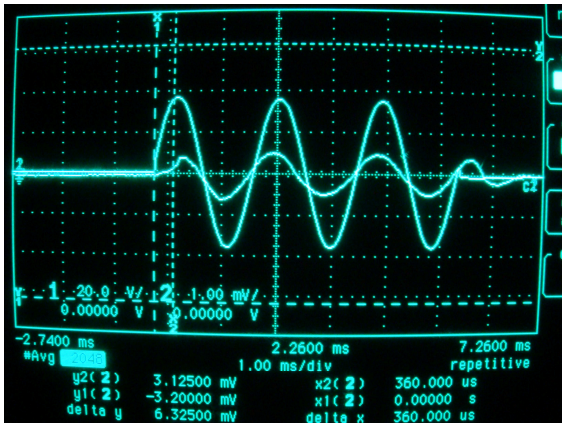


Figura I-5Formas de onda obtenidas para los cristales piezoeléctricos

Objetivo de la tesis

Desarrollar el software de control que permita modificar los parámetros de amplitud, frecuencia, fase y número de ciclos de la señal de excitación proporcionada por el generador de funciones de alta tensión.

Capítulo 1: Análisis Preliminar

Descripción del generador de funciones

Para la realización de este proyecto, se requiere un generador de funciones capaz de proveer un tren de uno a tres ciclos de ondas sinusoidales, con una amplitud de salida de $4\left[\text{ Vpp }\right]$ y una frecuencia máxima de operación de 300 [kHz]. Este tipo de equipo existe en forma comercial pero no se adapta a las necesidades del proyecto y su costo es muy elevado. Por estas razones, se desarrollaron los elementos electrónicos con el fin de reproducirlos en forma económica y llevarlos a aplicaciones posteriores.

Un generador de funciones puede diseñarse desde una perspectiva analógica o digital; en el caso analógico, se parte de una señal triangular que entra a un formador de seno para convertir la onda triangular en una sinusoide de amplitud constante.

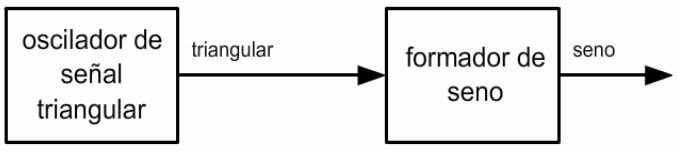


Figura 1-1 Diagrama de bloques de un generador analógico

En un generador de funciones digital, la memoria almacena los valores discretos asociados a las muestras de la señal que desea generarse. Un circuito de control se emplea para direccionar los datos de la memoria a un convertidor digital analógico (DAC). En el caso de una señal sinusoidal de 300 [kHz], se requiere un DAC de al menos 4.8 [MHz] y un sistema digital con reloj de 40 [MHz], lo que implica dispositivos en formato de montaje superficial para los cuales se carece de la herramienta adecuada para su manejo e instalación.

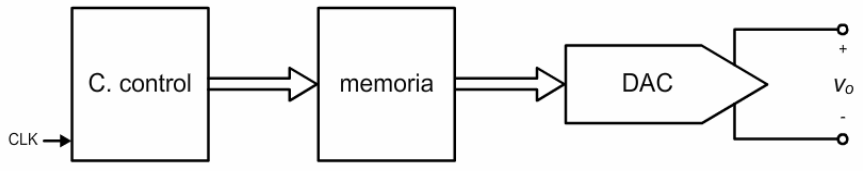


Figura 1-2 Diagrama de bloques de un generador de funciones digital

De lo anterior se concluye que un generador analógico complementado con un sistema digital, es la mejor opción para cumplir con las necesidades del proyecto; ya que permite modificar los parámetros de la onda generada y así, programarlo para tareas específicas.

De esta manera, se seleccionó el circuito integrado XR2206 en el que está implementado un generador de funciones analógico. La amplitud y la frecuencia de la señal generada se fijaron por dos fuentes de corriente controladas analógicamente. Este control, se obtuvo con el uso de un convertidor digital analógico del tipo PWM. Por otra parte, el intervalo de la frecuencia se estableció mediante un arreglo de capacitores controlado por relevadores. Finalmente, para obtener la señal deseada se utilizaron dos interruptores de estado sólido controlados por un contador auxiliar y un sistema digital simple, que al intercalar sus salidas forman el tren de ondas sinusoidales.

A fin de brindar una capacidad de programación similar a la de un generador de funciones digital, se seleccionó al puerto serie como vía de control del dispositivo.

Selección del microcontrolador

De acuerdo con las características anteriores, el circuito de control requirió de un puerto USART, dos generadores PWM y líneas de control adicionales; por lo que se seleccionó el microcontrolador PIC16F873, el cual reúne los requisitos necesarios para el proyecto. Aunado a esto, la abundancia de información lo convierte en una herramienta de fácil implementación.

La *figura 1-3* representa el diagrama final del Generador de funciones de Alta Tensión (*GFAT*).

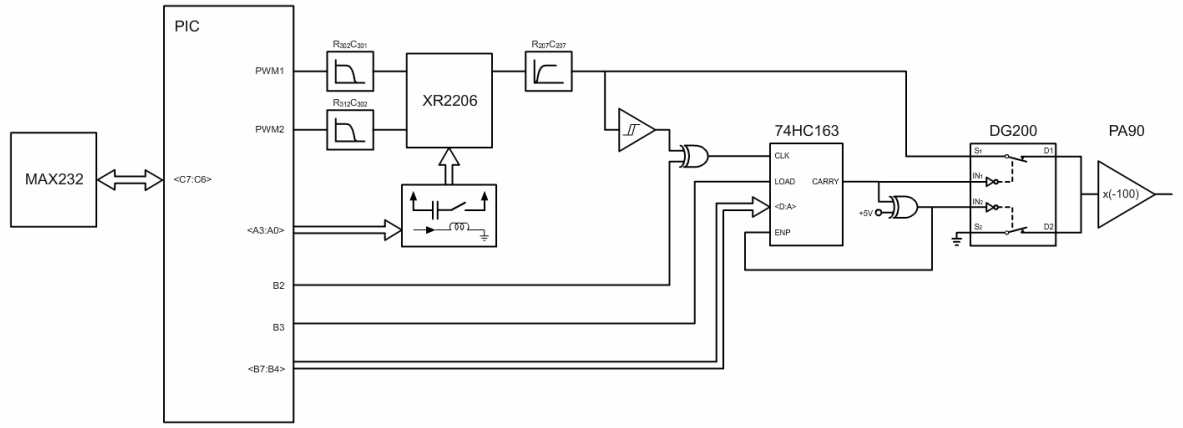


Figura 1-3 Diagrama de bloques del GFAT

Descripción del microcontrolador PIC16F873

Arquitectura interna

El microcontrolador PIC16F873 se caracteriza por:

- Tener una arquitectura Harvard.
- Su procesador es segmentado o *pipeline*.
- Su procesador es del tipo RISC.
- El formato de las instrucciones es ortogonal
- La arquitectura está basada en un banco de registros.

La *figura 1-4* representa el diagrama a bloques del PIC16F873. Destacan los siguientes componentes:

Características	PIC16F873
Frecuencia de operación	DC - 20 MHz
Resets (y retardos)	POR, BOR (PWRT, OST)
Memoria de programa FLASH (palabras de 14 bits)	4K
Memoria de datos (bytes)	192
Memoria de datos EEPROM (bytes)	128
Interrupciones	13
Puertos E/S	Puertos A, B, C
Timers	3
Módulos de Captura / Comparación / PWM	2
Comunicaciones seriales	MSSP, USART
Convertidor analógico digital de 10 bits	5 canales
Set de instrucciones	35 Instrucciones

Tabla 1-1 Características del PIC16F873

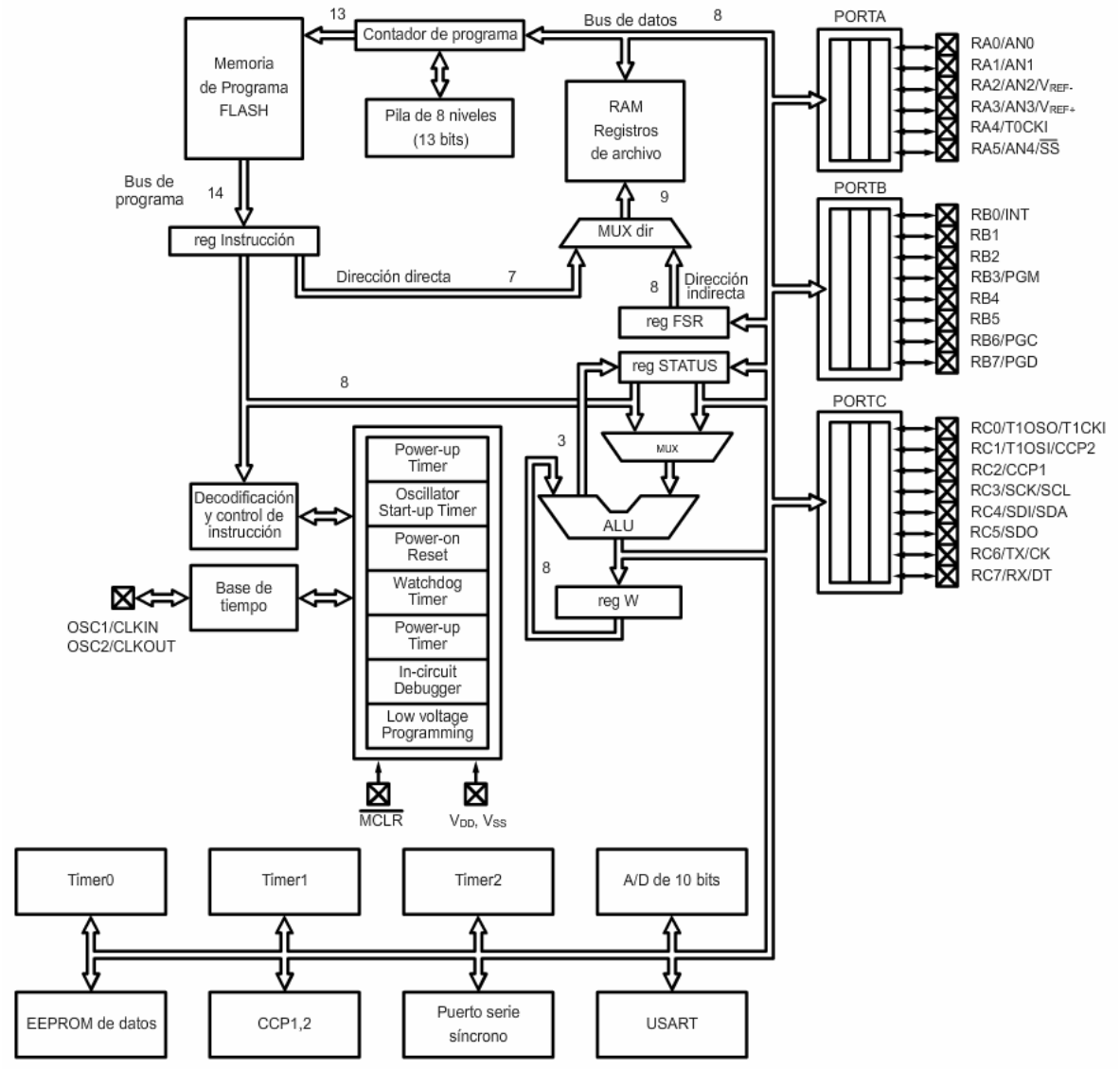


Figura 1-4 Arquitectura interna del PIC16F873

Organización de la memoria

El microcontrolador PIC16F873 contiene tres bloques de memoria:

- **Memoria de programa**. En sus 4096 localidades contiene el programa con las instrucciones que gobiernan la aplicación. Es del tipo no volátil, es decir, el programa se mantiene aunque desaparezca la alimentación.

- **Memoria de datos RAM**. Se destina a guardar las variables y datos. Es volátil, es decir, los datos almacenados se borran cuando desaparece la alimentación.

- **Memoria de datos EEPROM**. Es un área pequeña de memoria de datos de lectura y escritura no volátil, gracias a la cual, un corte en el suministro de la alimentación no ocasiona la pérdida de la información que estará disponible al reiniciarse el programa.

Memoria de programa

La memoria de programa almacena todas las instrucciones del programa de control. Debido a que el programa a ejecutar siempre es el mismo, debe ser grabado de forma permanente. Por esta razón, la memoria de programa del PIC16F873 es no volátil del tipo ROM **FLASH**. Esta característica garantiza que la memoria mantenga su contenido aún sin alimentación, de forma que el programa no necesite volver a ser grabado cada vez que se utilice.

En el PIC16F873, la memoria de programa tiene una capacidad de 4k (4096 localidades) y está organizada en palabras de 14 bits. Así pues, la memoria del programa comienza en la posición 000h (posición inicial de reset) y llega hasta FFFh. El vector de interrupción se encuentra en la posición 004h.

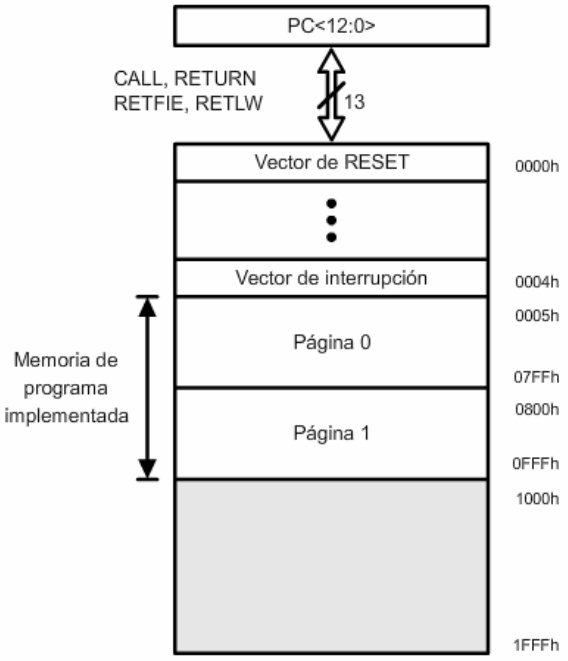


Figura 1-5 Estructura de la memoria de programa del PIC16F873

El contador de programa

El contador de programa (PC) es un registro interno de 13 bits que se utiliza para direccionar las instrucciones del programa de control que están almacenadas en la memoria de programa. Este registro contiene la dirección de la próxima instrucción a ejecutar y se incrementa automáticamente por lo que las instrucciones del programa se ejecutan una después de la otra.

Cuando el microcontrolador se conecta a la alimentación o cuando ocurre un reset, el contador del programa se pone en cero forzando así que la dirección de inicio sea 000h. La primera instrucción ejecutada será la que esté grabada en esta posición.

Memoria de datos RAM

En esta memoria se almacenan los datos que se manejan en un programa. Estos datos varían continuamente, por lo que esta memoria debe ser de lectura y escritura. Se implementa en RAM estática.

La memoria RAM del PIC16F873 se encuentra dividida en diferentes bancos. Los bits RP1(STATUS<6>) y RP0(STATUS<5>) se utilizan para la selección de bancos.

RP1:RP0	Banco
00	0
01	1
10	2
11	3

Tabla 1-2 Selección de banco

Cada banco se extiende hasta 7Fh y sus registros pueden clasificarse dentro de dos grupos:

- **Registros de Funciones Especiales SFR**. Son los primeros registros de cada banco y se emplean para controlar los modos de operación del microcontrolador. Algunos están duplicados en diferentes bancos para reducir el código y acelerar el acceso.

- **Registros de Propósito General GPR**. Son registros de uso general que pueden guardar los datos temporales del programa que se esté ejecutando.

Memoria de datos EEPROM

El PIC16F873 dispone de una zona de 128 bytes de memoria EEPROM para almacenar datos que no se pierdan al desconectar la alimentación. Esto es muy útil ya que permite guardar datos permanentemente.

Subrutinas

En ocasiones un mismo grupo de instrucciones es ejecutado en diferentes partes de un programa y en principio tendría que repetirse tantas veces como el número de ejecuciones dentro del programa. Sin embargo, esto genera un programa ineficiente ya que requiere de una extensión mayor.

La solución más efectiva en términos de ahorro de memoria se obtiene si el grupo de instrucciones que se repite aparece una sola vez en el programa, pero con capacidad para ser ejecutado desde todos los puntos en que aquél se pide. La estructura de programación que implementa esta solución es la *subrutina*.

Una subrutina es un conjunto de instrucciones al que se tiene acceso desde cualquier punto del programa principal. Es decir, una subrutina es un subprograma que se ejecuta cada vez que el programa principal lo necesita.

La acción de pasar del programa principal a la subrutina se denomina *llamada a la subrutina* y se realiza con la instrucción *CALL* que se debe intercalar en el programa principal. Por otra parte, la acción de volver al programa principal después de llevar a cabo las tareas de la subrutina se llama *retorno de la subrutina* y se realiza con la instrucción *RETURN*.

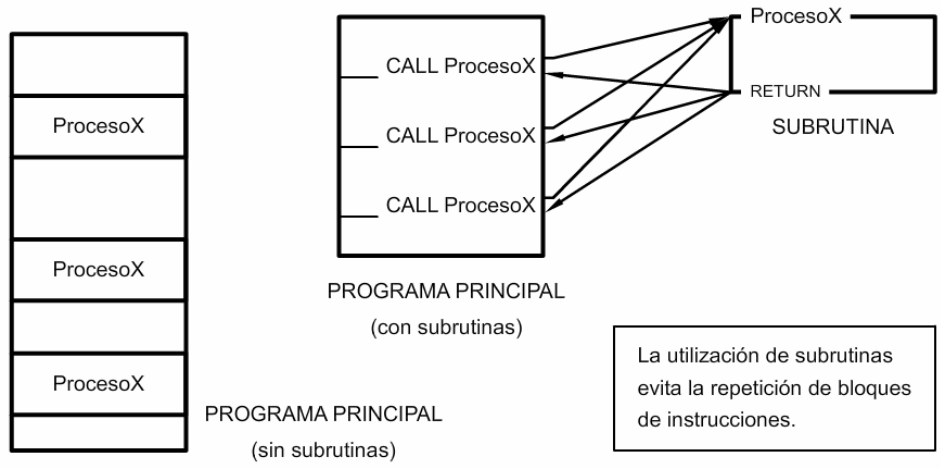


Figura 1-6 Utilización de subrutinas

La principal ventaja de las subrutinas es que la extensión de los programas se hace mucho más corta; sin embargo, provocan una ejecución más lenta debido a que se tienen que ejecutar dos instrucciones extras: una llamada y el obligatorio retorno de la subrutina.

Subrutinas anidadas

Cuando una subrutina llama a otra subrutina se produce una situación conocida como *anidamiento de subrutinas*, es decir, hay subrutinas anidadas dentro de otras. Cada *CALL* sucesivo sin que intervenga un *RETURN* crea un nivel de anidamiento adicional.

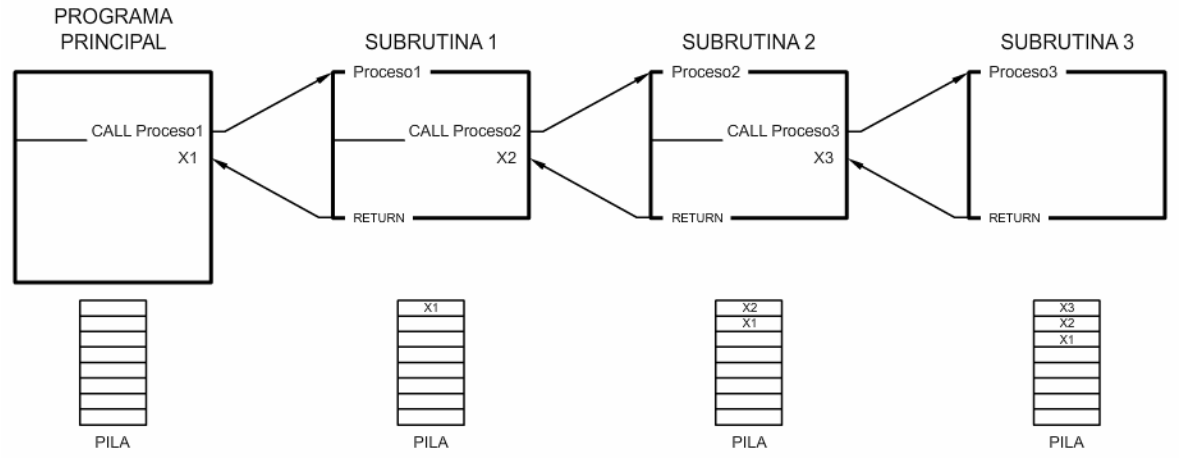


Figura 1-7 Subrutinas anidadas

El nivel de anidamiento está limitado para cada microcontrolador y en el PIC16F873 es de 8 niveles. Esto es, para un PIC16F873 no puede haber más de ocho subrutinas anidadas.

La pila

La pila (stack) es una zona de la memoria que se encuentra separada tanto de la memoria de programa como de la de datos dentro del microcontrolador. Su estructura es del tipo LIFO *(Last In First Out)* por lo que el último dato que se guarda es el primero que sale.

El PIC16F873 dispone de una pila con ocho niveles de una longitud de 13 bits cada uno.

La pila se carga a través de la llamada a la subrutina con la instrucción *CALL*, que almacena el contenido del contador de programa (PC) en la posición superior de la pila. Para recuperar el contenido de la pila en el PC, hay que ejecutar la instrucción de retorno de subrutina *RETURN*.

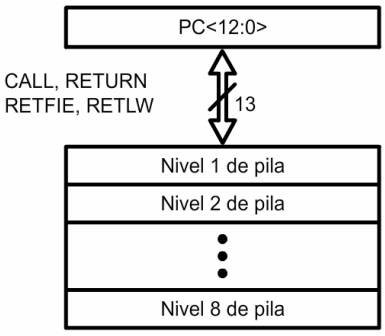


Figura 1-8 Estructura de la pila del PIC16F873

Modos de direccionamiento

Los modos de direccionamiento se refieren a la forma en la que se tiene acceso a la memoria del microcontrolador para fines de almacenamiento y ejecución de datos.

- **Inherente**. El operando está embebido en la instrucción.

  Ejemplo: NOP

- **Inmediato**. El operando se encuentra inmediatamente después de la instrucción.

  Ejemplo: MOVLW 0x0A

- **Directo**. El operando es una dirección de 8 bits de la memoria.

  Ejemplo: CLRF PORTB

- **Indizado**. Tiene acceso a localidades de memoria a partir de un registro intermedio que contiene la dirección efectiva.

  Ejemplo: ADDWF PCL,F

- **Indirecto**. La dirección del dato se encuentra contenida en el registro *INDF*. Cada vez que se hace referencia a éste, se utiliza el contenido del registro apuntador *FSR* para direccionar el operando.

  Ejemplo: MOVWF INDF

Interrupciones

Una interrupción consiste en un mecanismo mediante el cual un evento interno o externo puede interrumpir la ejecución del programa principal en cualquier momento. En ese instante, se produce automáticamente un salto a una *subrutina de atención a la interrupción*. Una vez atendido el evento, se retoma la ejecución del programa principal exactamente donde estaba en el momento de ser interrumpido.

La interrupción ejecuta una subrutina donde la intervención del microcontrolador es urgente. Por esta razón es más eficaz que la técnica de *encuesta*, ya que el microcontrolador no pierde tiempo preguntando por el estado de la línea de entrada, sino que únicamente atiende al periférico cuando éste lo solicita por medio de la interrupción.

La *figura 1-9* representa la lógica de interrupción del PIC16F873 y sus 13 posibles fuentes de interrupción.

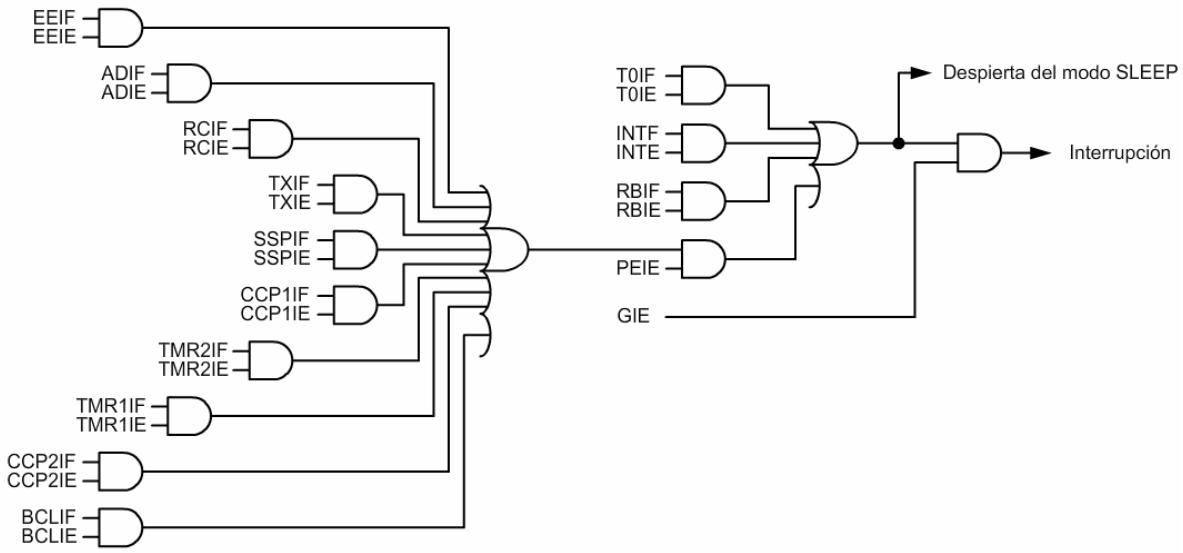


Figura 1-9 Lógica de interrupción del PIC16F873

Consideraciones importantes

- Ya que durante una interrupción, el contenido de los registros del microcontrolador puede modificarse, se debe guardar el valor de los mismos y restaurarlos a su valor original antes de regresar al programa principal.

- El microcontrolador sólo dispone de un vector de interrupción en la dirección 04h, así que sea cual sea la fuente de la interrupción, el contador de programa se carga con la dirección 04h. Por tal motivo, es importante que el programa identifique la causa de la interrupción verificando el estado de las banderas de cada una de las fuentes habilitadas.

Fases de una interrupción

A continuación se enumeran las acciones que realiza automáticamente el microcontrolador y que deben tomarse en cuenta durante la implementación del programa:

1. El programa debe habilitar las interrupciones correspondientes mediante una instrucción en la inicialización.

2. Cuando ocurre una interrupción, la bandera correspondiente se activa. Si el bit de permiso correspondiente está a ‘1’ y los bits de habilitación GIE y PEIE están a ‘1’, se produce la interrupción.

3. Para evitar que se produzca otra interrupción, mientras se está atendiendo a una anterior, el bit GIE se pone automáticamente a ‘0’ por hardware.

4. El valor del contador de programa (PC) se guarda en la pila.

5. El PC se carga con el valor correspondiente al vector de interrupciones y se produce un salto a esa sección de la memoria iniciando así, la ejecución de las instrucciones correspondientes a la interrupción.

6. Una vez dentro de la interrupción, el programa debe guardar todos los registros que puedan ser modificados durante la interrupción.

7. Si están habilitadas varias vías de interrupción, el programa debe determinar la causa, verificando el estado de las banderas.

8. Dependiendo de la causa de la interrupción, se bifurca a la subrutina correspondiente.

9. Una vez finalizado el tratamiento de la interrupción el programa debe devolver los valores que tenían los registros antes de producirse la interrupción.

10. El programa debe borrar las banderas que indican las fuentes de las interrupciones antes de regresar al programa principal.

11. Cuando llega a la última instrucción *RETFIE*, el contador del programa se carga con el valor que se guardó inicialmente en la pila y el bit GIE se pone automáticamente a ‘1’.

Selección del software y hardware para la programación del microcontrolador

Software de programación

El lenguaje ensamblador utiliza un grupo de caracteres alfanuméricos, llamados **mnemónicos**, para simbolizar las órdenes o tareas a realizar en cada instrucción. Los mnemónicos se corresponden con las iniciales de la instrucción en inglés. Por ejemplo:

• Instrucción:	suma 58 al registro de trabajo W y guarda el resultado en este mismo registro.
• Ensamblador:	ADDLW d'58'
• Máquina:	11 1110 0011 1010(expresado en binario) 3E3A(expresado en hexadecimal)

Tabla 1-3 Ejemplo de mnemónicos

Programa Ensamblador

El programa ensamblador es un software que se encarga de traducir los mnemónicos y símbolos alfanuméricos del programa escrito en ensamblador por el usuario a código de máquina, para que pueda ser interpretado y ejecutado por el microcontrolador.

El programa escrito en lenguaje ensamblador recibe la denominación de **código fuente** y tiene la extensión **.asm*. La mayoría de los ensambladores proporcionan a su salida un archivo que suele tener la extensión **.hex*. Este archivo puede ser grabado en la memoria de programa mediante la utilización de un grabador de microcontroladores.

El ensamblador más utilizado para los PIC es el **MPASM**, que trabaja dentro de un entorno de software denominado MPLAB.

MPLAB

El **MPLAB IDE** es un software de "Entorno de Desarrollo Integrado" (*Integrated Development Enviroment, IDE*) que se ejecuta bajo Windows.

El MPLAB permite editar, ensamblar y simular en pantalla el código fuente del proyecto para comprobar como evoluciona tanto la memoria de datos, como la de programa y los registros de funciones especiales, según progresa la ejecución del programa.

Hardware de Programación

El PICSTART Plus es un grabador de microcontroladores que permite programar el código ensamblado del usuario en dispositivos PIC.

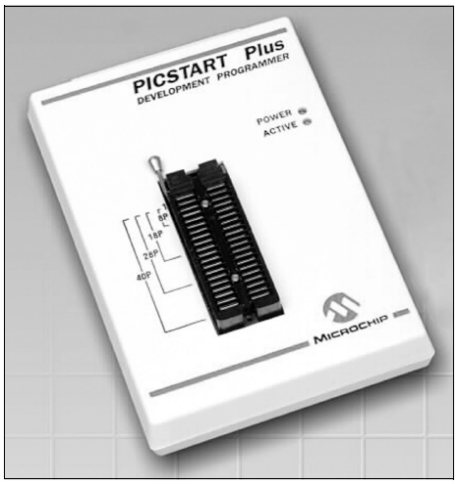


Figura 1-10 Programador PICSTART Plus

El sistema del PICSTART Plus permite:

- Programar microcontroladores PIC, incluyendo memoria de programa, bits de configuración y localidades ID.
- Ser operado como una aplicación de *Windows* a través de MPLAB IDE.
- Comunicarse con la computadora mediante un cable RS232 estándar.
- Con el MPLAB IDE, se puede crear, desplegar y editar los datos que serán programados en el microcontrolador.
- Adicionalmente, puede verificarse que el MCU se encuentre en blanco, verificar que el código en el microcontrolador corresponde al *firmware* del usuario y leer desde el MPLAB IDE el código de un microcontrolador sin protección de lectura para ser depurado y programado en otros dispositivos.

Capítulo 2: Metodología de Cálculo

Parámetro de amplitud

La *figura 2-1* representa la circuitería involucrada en el procesamiento del parámetro de amplitud.

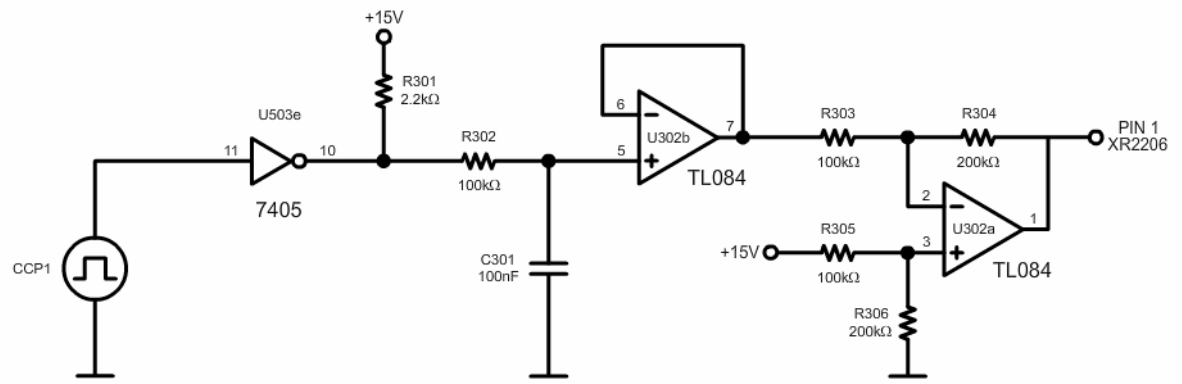


Figura 2-1 Procesamiento del parámetro de amplitud

El pin CCP1 del microcontrolador proporciona una señal cuadrada de 10 [kHz] modulada por ancho de pulso. Esta señal ingresa al inversor U_{503e} que eleva su amplitud de 5 a 15 [V]. La señal amplificada entra al filtro $R_{302}C_{301,}$ para convertirse en un nivel de DC, y posteriormente al seguidor U_{302b}. Finalmente el restador U_{302a} ajusta el nivel para la entrada de modulación AM del generador XR2206.

A fin de determinar la relación entre el ancho de pulso de la señal CCP1 y la amplitud generada debemos seguir los siguientes pasos:

a) Determinar el voltaje de modulación AM en función de la amplitud deseada.

b) Determinar el sustraendo del restador U_{302a} (V_{C301}) en función del voltaje de modulación AM.

c) Determinar el ciclo de trabajo de la señal CCP1 en función de V_{C301}.

Voltaje de modulación AM

De la *figura 2-2*, encontramos que la amplitud de la senoide generada depende de la entrada de modulación AM del XR2206. La ecuación correspondiente es:

$$V_{mod} = 2\,Amplitud + 7.5 \tag{2.01}$$

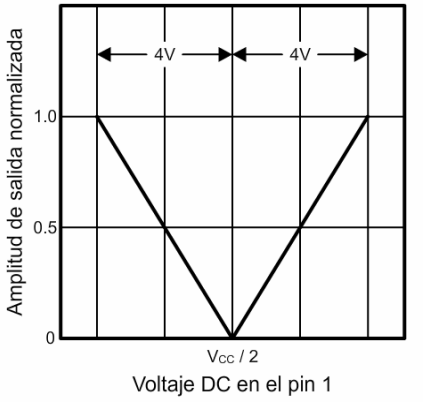


Figura 2-2 Amplitud de salida normalizada en función del voltaje de modulación AM

Sustraendo V_{C301}

Para el restador U_{302a} tenemos que:

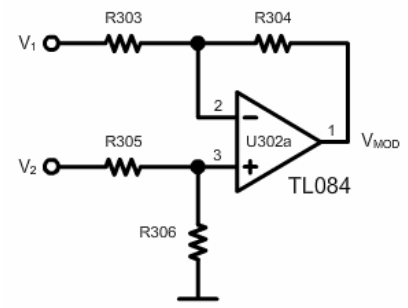


Figura 2-3 Restador U_{302a}

$$V_{mod} = \left(\frac{R_{304}}{R_{303}} + 1\right)\left(\frac{R_{306}}{R_{305} + R_{306}}\right)V_2 - \left(\frac{R_{304}}{R_{303}}\right)V_1$$

si $R_{303} = R_{305} = R_i$ y $R_{304} = R_{306} = R_f$

$$V_{mod} = \frac{R_f}{R_i}\left(V_2 - V_1\right)$$

sustituyendo $R_f = 200\,[\,\mathrm{k\Omega}\,]$, $R_i = 100\,[\,\mathrm{k\Omega}\,]$ y $V_2 = 15\,[\,\mathrm{V}\,]$

$$V_{mod} = 2\left(15 - V_1\right)$$

despejando:

$$V_1 = 15 - \frac{V_{mod}}{2} \tag{2.02}$$

Ciclo de trabajo CCP1

Para determinar el ciclo de trabajo de la señal CCP1 en función de V_{C301}, debemos modelar la respuesta del filtro $R_{302}C_{301}$ para diferentes ciclos de trabajo. Esto implica obtener la función de transferencia del filtro y la expresión para una onda cuadrada cuyo ciclo de trabajo es variable.

Función de transferencia

La *figura 2-4* representa el diagrama esquemático del filtro paso bajas $R_{302}C_{301}$.

Analizando la malla, tenemos que:

$$i_{C_{301}} = C_{301}\frac{d}{dt}v_{C_{301}}$$

$$i_{R_{302}} = i_{C_{301}}$$

$$v_{i_a} = v_{R_{302}} + v_{C_{301}}$$

$$v_{i_a} = R_{302}i_{R_{302}} + v_{C_{301}}$$

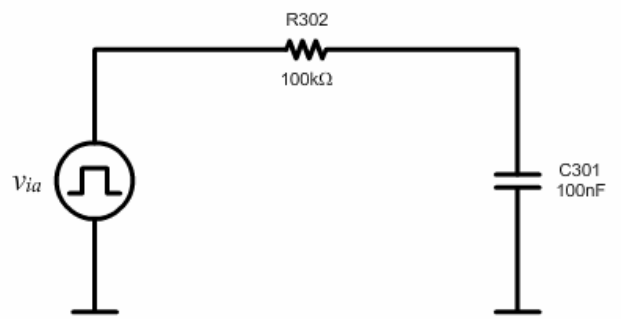


Figura 2-4 Filtro $R_{302}C_{301}$

$$v_{i_a} = R_{302}i_{C_{301}} + v_{C_{301}}$$

$$v_{i_a} = R_{302}C_{301}\dot{v}_{C_{301}} + v_{C_{301}}$$

$$\dot{v}_{C_{301}} + \frac{1}{R_{302}C_{301}}v_{C_{301}} = \frac{1}{R_{302}C_{301}}v_{i_a}$$

Aplicando la transformada de Laplace:

$$\left(s + \frac{1}{R_{302}C_{301}}\right)V_{C_{301}}(s) = \frac{1}{R_{302}C_{301}}V_{i_a}(s)$$

$$H_a(s) = \frac{V_{C_{301}}(s)}{V_{i_a}(s)} = \frac{\dfrac{1}{R_{302}C_{301}}}{s + \dfrac{1}{R_{302}C_{301}}} \qquad (2.03)$$

Si $z_a = \frac{1}{R_{302}C_{301}}$ y $p_a = \frac{1}{R_{302}C_{301}}$ la función de transferencia puede expresarse como:

$$H_a(s) = \frac{z_a}{s + p_a} \tag{2.04}$$

Para calcular la frecuencia de corte del filtro partimos de la ecuación:

$$|H_a(\omega)| = \frac{1}{\sqrt{2}}$$

sustituyéndola en (2.03), encontramos que:

$$\omega_{c_a} = \frac{1}{R_{302}C_{301}}$$

como $\omega = 2\pi f$:

$$f_{c_a} = \frac{1}{2\pi R_{302}C_{301}} \tag{2.05}$$

sustituyendo $R_{302} = 100\,[\,\mathrm{k\Omega}\,]$ y $C_{301} = 100\,[\,\mathrm{nF}\,]$ en (2.05), encontramos que:

$$f_{c_a} = 15.91\,[\,\mathrm{Hz}\,]$$

Señal de entrada

La entrada del filtro es una señal cuadrada cuyo ciclo de trabajo es variable y ya que es una función periódica, puede expresarse en la forma de una serie trigonométrica de Fourier.

$$x(t) = a_0 + \sum_{n=1}^{\infty}\left[a_n \cos(n\,\omega_0 t) + b_n sen(n\,\omega_0 t)\right] \tag{2.06}$$

Sabiendo que la onda cuadrada tiene una frecuencia $f_{0_a} = 10\,[\,\mathrm{kHz}\,]$ y que el filtro $R_{302}C_{301}$ corta en 15.91 [Hz], la serie puede simplificarse a:

$$x_a(t) = a_{0_a}$$

donde:

$$a_{0_a} = \frac{1}{T_0}\int_0^T V_p dt = \frac{V_p}{T_0}\int_0^T dt = \frac{V_p}{T_0} t\Big|_0^T = V_p \frac{T}{T_0} = V_p \delta_{\text{CCP1}}$$

$$a_{0_a} = V_p \delta_{\text{CCP1}}$$

que sustituyendo en (3.06) con $a_n = b_n = 0$ resulta:

$$x(t) = V_p \delta_{\text{CCP1}} \tag{2.07}$$

Respuesta

Ya que el filtro elimina las armónicas de orden superior, la respuesta del filtro se simplifica a una respuesta escalón.

$$V_{C_{301}}(s) = X_a(s) H_a(s)$$

$$V_{C_{301}}(s) = \frac{a_{0_a}}{s} \frac{z_a}{s + p_a}$$

descomponiendo en fracciones parciales:

$$V_{C_{301}}(s) = a_{0_a}\left[\frac{z_a}{s(s + p_a)}\right] = a_{0_a}\left(\frac{A_a}{s} + \frac{B_a}{s + p_a}\right)$$

donde $A_a = \lim_{s\to 0} \frac{z_a}{s + p_a} = \frac{z_a}{p_a}$ y $B_a = \lim_{s\to -p_a} \frac{z_a}{s} = -\frac{z_a}{p_a}$. Sustituyendo:

$$V_{C_{301}}(s) = a_{0_a}\left(\frac{A_a}{s} + \frac{B_a}{s + p_a}\right) = a_{0_a}\left[\frac{z_a}{p_a}\left(\frac{1}{s}\right) - \frac{z_a}{p_a}\left(\frac{1}{s + p_a}\right)\right] = a_{0_a}\left[\frac{z_a}{p_a}\left(\frac{1}{s} - \frac{1}{s + p_a}\right)\right]$$

Aplicando la transformada inversa de Laplace:

$$v_{C_{301}}(t) = a_{0_a}\left[\frac{z_a}{p_a}\left(1 - e^{-p_a t}\right)\right] \tag{2.08}$$

En el dominio del tiempo, la respuesta del filtro $R_{302}C_{301}$ está dada por la ecuación:

$$v_{C_{301}}(t) = V_p \delta_{\text{CCP1}} \left(1 - e^{\frac{-t}{R_{302}C_{301}}} \right)$$

La *tabla 2-1* muestra los valores de tensión en el capacitor C_{301} en función del ciclo de trabajo de la onda cuadrada.

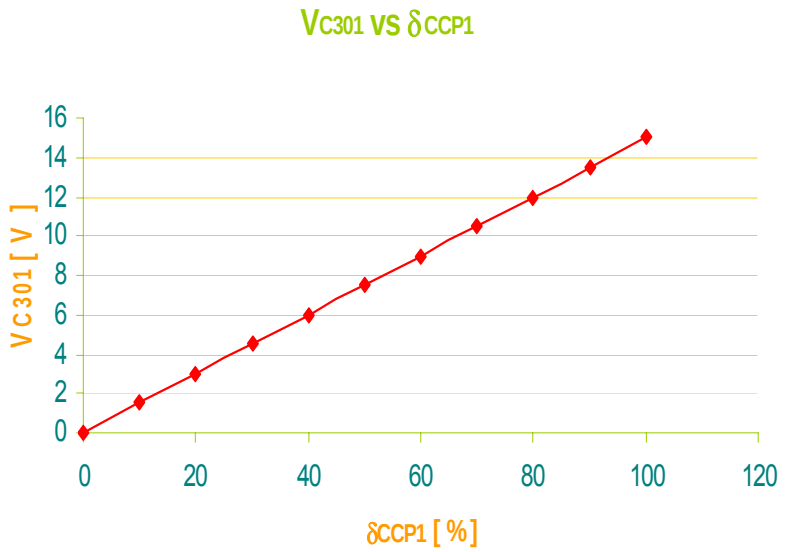


Figura 2-5 Respuesta del filtro $R_{302}C_{301}$

δ_{CCP1} [%]	/δ_{CCP1} [%]	V_{C301} [V]
0	100	14.9980
10	90	13.4920
20	80	11.9870
30	70	10.4830
40	60	8.9809
50	50	7.4803
60	40	5.9813
70	30	4.4837
80	20	2.9877
90	10	1.4931
100	0	0.0000

Tabla 2-1 Respuesta del filtro $R_{302}C_{301}$

Para encontrar el modelo matemático lineal del comportamiento del ciclo de trabajo CCP1 en función de V_{C301} partimos de la expresión:

$$\delta_{\text{CCP1}}\left[\,\%\,\right] = m_1\left[\frac{\%}{\text{V}}\right]\cdot V_{C_{301}}\left[\,\text{V}\,\right] + b_1\left[\,\%\,\right]$$

y de la siguiente tabla que condensa los datos requeridos por el método de mínimos cuadrados.

x	y	x·y	x^2
14.9980	0	0.0000	224.9400
13.4920	10	134.9200	182.0341
11.9870	20	239.7400	143.6882
10.4830	30	314.4900	109.8933
8.9809	40	359.2360	80.6566
7.4803	50	374.0150	55.9549
5.9813	60	358.8780	35.7759
4.4837	70	313.8590	20.1036
2.9877	80	239.0160	8.9264
1.4931	90	134.3790	2.2293
0.0000	100	0.0000	0.0000

Donde:

· n	11
· Σx	82.3670
· Σy	550
· Σx·y	2468.5330
· Σx^2	864.2022
· $(\Sigma x)^2$	6784.3227

Por lo que:

$$m = \frac{n\cdot\sum_{i=1}^{n}\left(x_i\cdot y_i\right) - \sum_{i=1}^{n}x_i\cdot\sum_{i=1}^{n}y_i}{n\cdot\sum_{i=1}^{n}x_i^2 - \left(\sum_{i=1}^{n}x_i\right)^2}$$

$$b = \frac{\sum_{i=1}^{n}y_i\cdot\sum_{i=1}^{n}x_i^2 - \left[\sum_{i=1}^{n}\left(x_i\cdot y_i\right)\cdot\sum_{i=1}^{n}x_i\right]}{n\cdot\sum_{i=1}^{n}x_i^2 - \left(\sum_{i=1}^{n}x_i\right)^2}$$

$$m_1 = \frac{\left(11\cdot 2468.53\right) - \left(82.367\cdot 550\right)}{\left(11\cdot 864.2022\right) - \left(6784.3227\right)}$$

$$b_1 = \frac{\left(550\cdot 864.2022\right) - \left(2468.53\cdot 82.367\right)}{\left(11\cdot 864.2022\right) - \left(6784.3227\right)}$$

$$m_1 = -6.6674$$

$$b_1 = 99.9248$$

Finalmente:

$$\delta_{\text{CCP1}}\left[\,\%\,\right] = -6.6674\left[\frac{\%}{\text{V}}\right]\cdot V_{C_{301}}\left[\,\text{V}\,\right] + 99.9248\left[\,\%\,\right] \qquad (2.09)$$

Parámetro de frecuencia

La *figura 2-6* representa la circuitería involucrada en el procesamiento del parámetro de frecuencia.

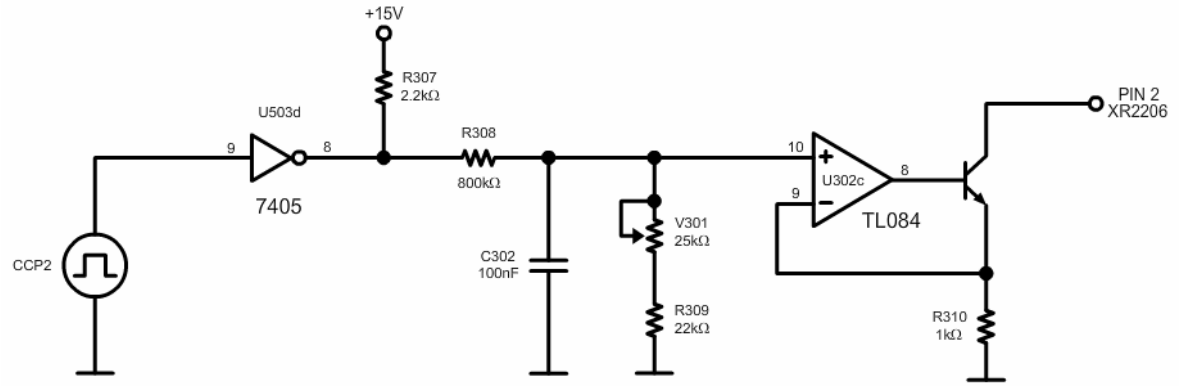


Figura 2-6 Procesamiento del parámetro de frecuencia

El pin CCP2 del microcontrolador proporciona una señal cuadrada de 10 [kHz] modulada por ancho de pulso. Esta señal ingresa al inversor U_{503d} que eleva su amplitud de 5 a 15 [V]. La señal amplificada entra al filtro $R_{308}C_{302}R_{309}R_{301}$ y su respuesta corresponde a una tensión de corriente directa. Finalmente el seguidor U_{302c} activa al transistor Q_{301} que sirve como interruptor de la fuente de corriente del generador XR2206.

Por otra parte, la palabra $A_3A_2A_1A_0$ del puerto A del microcontrolador ajusta el selector de frecuencia para el valor deseado.

A fin de determinar la relación entre el ancho de pulso de la señal CCP2 y la frecuencia generada debemos seguir los siguientes pasos:

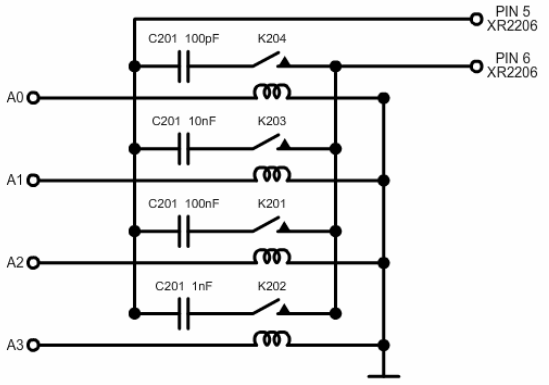


Figura 2-7 Selector de frecuencia

a) Determinar el valor del selector de frecuencia en función de la frecuencia deseada.

b) Determinar la corriente de terminales I_T en función de la frecuencia deseada.

c) Determinar la respuesta del filtro (V_{C302}) en función de la corriente I_T.

d) Determinar el ciclo de trabajo de la señal CCP2 en función de V_{C302}.

Selector de frecuencia

De la ecuación (2.10), encontramos que la frecuencia de la senoide generada depende de la corriente I_T suministrada por el XR2206 y de un capacitor. A fin de que pueda variar dentro del intervalo de 100 [Hz] a 1 [MHz], se emplea un selector de frecuencia de acuerdo con la siguiente tabla:

Intervalo [Hz]	C [μF]
100 ≤ f < 1000	0.1
1000 ≤ f < 10000	0.01
10000 ≤ f < 100000	0.001
100000 ≤ f ≤ 1000000	0.0001

Tabla 2-2 Selector de frecuencia

Corriente de terminales I_T

Una vez seleccionado el capacitor podemos calcular I_T para la frecuencia deseada.

$$I_T = \frac{f\,[\,\text{Hz}\,]C\;[\,\mu\text{F}\,]}{320}\;[\,\text{mA}\,] \tag{2.10}$$

De la *figura 2-8*, podemos observar que la corriente I_T en función del voltaje V_{C302} se define por la siguiente expresión:

$$I_T = \frac{V_{C_{302}}}{R_{310}}\;[\,\text{A}\,] \tag{2.11}$$

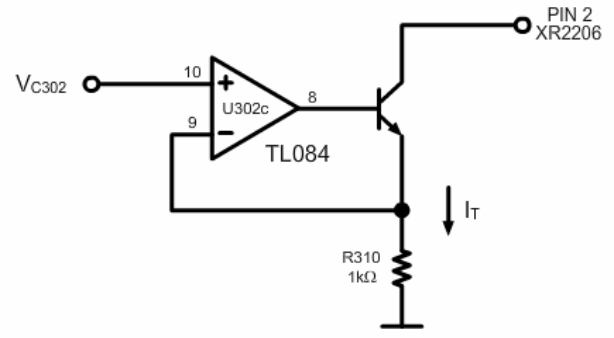


Figura 2-8Corriente I_T

Sustituyendo $R_{310} = 1\;[\,\text{k}\Omega\,]$, $I_T = V_{C_{302}}\;[\,\text{mA}\,]$.

Ciclo de trabajo CCP2

Para determinar el ciclo de trabajo de la señal CCP2 en función de V_{C302}, debemos modelar la respuesta del filtro $R_{308}C_{302}$ para diferentes ciclos de trabajo.

Función de transferencia

La *figura 2-9* representa el diagrama esquemático del filtro.

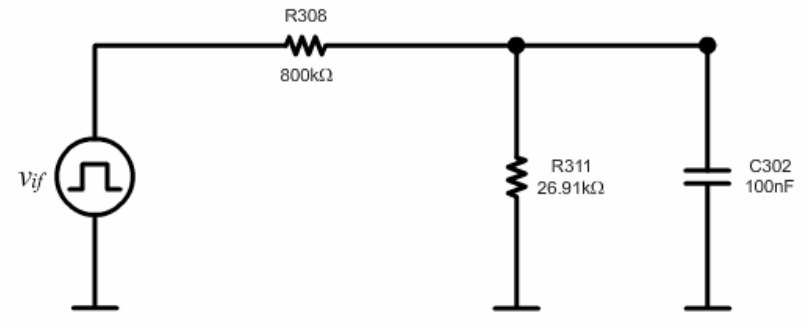


Figura 2-9 Filtro $R_{308}C_{302}R_{311}$

donde

$$R_{311} = R_{309} + V_{301}$$

Realizando el equivalente Thevenin, el circuito se reescribe como:

con:

$$R_{311} = R_{309} + V_{301}$$

$$R_{312} = \frac{R_{308}R_{311}}{R_{308} + R_{311}}$$

$$V_{i_f'}(s) = \frac{R_{312}}{R_{308} + R_{312}} V_{i_f}(s)$$

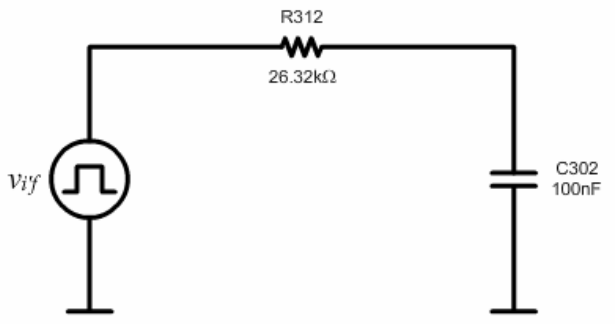


Figura 2-10 Filtro $R_{312}C_{302}$

Como puede observarse, el filtro $R_{312}C_{302}$ tiene la misma forma que el $R_{302}C_{301}$, de modo que su función de transferencia resulta:

$$H_f(s) = \frac{V_{C_{302}}(s)}{V_{i_f}(s)} = \frac{\dfrac{1}{R_{312}C_{302}}}{s + \dfrac{1}{R_{312}C_{302}}} \tag{2.12}$$

Si $z_f = \dfrac{1}{R_{312}C_{302}}$ y $p_f = \dfrac{1}{R_{312}C_{302}}$ la función de transferencia se reescribe como:

$$H_f(s) = \frac{z_f}{s + p_f} \tag{2.13}$$

donde la frecuencia de corte está definida por:

$$f_{c_f} = \frac{1}{2\,\pi\, R_{312}C_{302}} \tag{2.14}$$

Si $R_{308} = 1.2\,[\,\mathrm{M\Omega}\,]$ y $R_{311} = 26.91\,[\,\mathrm{k\Omega}\,]$, $R_{312} = 26.32\,[\,\mathrm{k\Omega}\,]$ que al sustituir con $C_{302} = 100\,[\,\mathrm{nF}\,]$ en (2.14), encontramos que $f_{c_f} = 60.47\ [\,\mathrm{Hz}\,]$.

Respuesta

Al igual que en el caso de amplitud, el filtro $R_{312}C_{302}$ elimina las componentes de orden superior de la serie trigonométrica de la ecuación (2.06), de modo que las ecuaciones (2.07) y (2.08) son válidas y podemos afirmar que:

$$x(t) = V_p \delta_{\mathrm{CCP2}} \tag{2.15}$$

$$v_{C_{302}}(t) = a_{0_f}\left[\frac{z_f}{p_f}\left(1 - e^{-p_f t}\right)\right] \tag{2.16}$$

En el dominio del tiempo, la respuesta del filtro $R_{312}C_{302}$ está dada por la ecuación:

$$v_{C_{302}}(t) = V_p \delta_{\text{CCP2}} \left(1 - e^{\frac{-t}{R_{312}C_{302}}} \right)$$

La *tabla 2-3* muestra los valores de tensión en el capacitor C_{302} en función del ciclo de trabajo de la onda cuadrada.

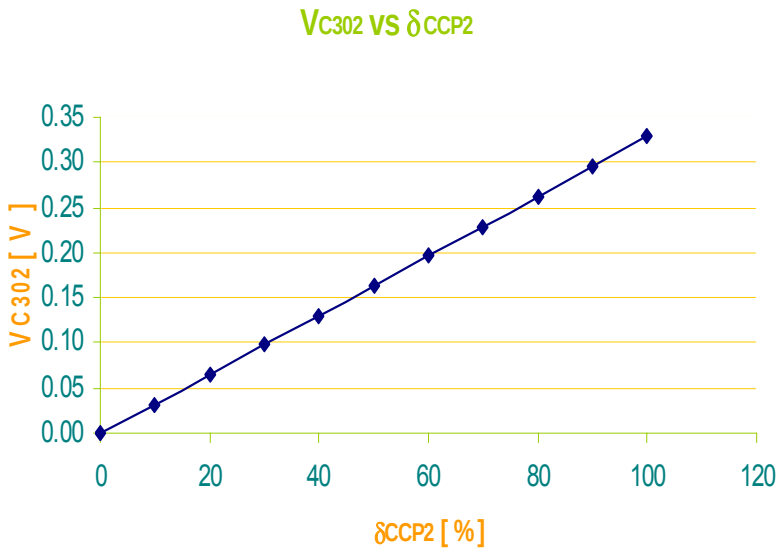


Figura 2-11 Respuesta del filtro $R_{312}C_{302}$

δ_{CCP2} [%]	/δ_{CCP2} [%]	V_{C302} [V]	I_T [mA]
0	100	0.3290	0.3290
10	90	0.2955	0.2955
20	80	0.2622	0.2622
30	70	0.2290	0.2290
40	60	0.1959	0.1959
50	50	0.1629	0.1629
60	40	0.1301	0.1301
70	30	0.0974	0.0974
80	20	0.0648	0.0648
90	10	0.0323	0.0323
100	0	0.0000	0.0000

Tabla 2-3 Respuesta del filtro $R_{312}C_{302}$

Para encontrar el modelo matemático lineal del comportamiento del ciclo de trabajo en función de la corriente partimos de la expresión:

$$\delta_{\text{CCP2}}\left[\,\%\,\right]=m_2\left[\frac{\%}{\text{mA}}\right]\cdot I_T\left[\,\text{mA}\,\right]+b_2\left[\,\%\,\right]$$

y de la siguiente tabla que condensa los datos requeridos por el método de mínimos cuadrados.

x	y	x·y	x^2
0.3290	0	0.0000	0.1082
0.2955	10	2.9553	0.0873
0.2622	20	5.2438	0.0687
0.2290	30	6.8694	0.0524
0.1959	40	7.8360	0.0384
0.1629	50	8.1470	0.0265
0.1301	60	7.8060	0.0169
0.0974	70	6.8174	0.0095
0.0648	80	5.1843	0.0042
0.0323	90	2.9107	0.0010
0.0000	100	0.0000	0.0000

Donde:

- $n = 11$
- $\Sigma x = 1.7992$
- $\Sigma y = 550$
- $\Sigma x\cdot y = 53.7699$
- $\Sigma x^2 = 0.4133$
- $(\Sigma x)^2 = 3.2370$

Por lo que:

$$m=\frac{n\cdot\sum_{i=1}^{n}\left(x_i\cdot y_i\right)-\sum_{i=1}^{n}x_i\cdot\sum_{i=1}^{n}y_i}{n\cdot\sum_{i=1}^{n}x_i^2-\left(\sum_{i=1}^{n}x_i\right)^2}$$

$$b=\frac{\sum_{i=1}^{n}y_i\cdot\sum_{i=1}^{n}x_i^2-\left[\sum_{i=1}^{n}\left(x_i\cdot y_i\right)\cdot\sum_{i=1}^{n}x_i\right]}{n\cdot\sum_{i=1}^{n}x_i^2-\left(\sum_{i=1}^{n}x_i\right)^2}$$

$$m_2=\frac{\left(11\cdot 53.7699\right)-\left(1.7992\cdot 550\right)}{\left(11\cdot 0.4133\right)-\left(1.7992^2\right)}$$

$$b_2=\frac{\left(550\cdot 0.4133\right)-\left(53.7699\cdot 1.7992\right)}{\left(11\cdot 0.4133\right)-\left(1.7992^2\right)}$$

$$m_2=-303.9519$$

$$b_2=99.7148$$

Finalmente:

$$\delta_{\text{CCP2}}\left[\,\%\,\right]=-303.95\left[\frac{\%}{\text{mA}}\right]\cdot I_T\left[\,\text{mA}\,\right]+99.7148\left[\,\%\,\right] \qquad (2.17)$$

Parámetro de fase

La *figura 2-12* representa la circuitería involucrada en el procesamiento del parámetro de fase.

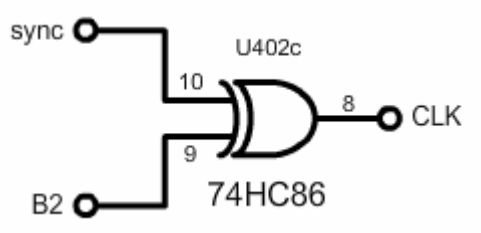


Figura .2-12 Procesamiento del parámetro de fase

Las entradas de la compuerta U402c, del tipo *or exclusiva*, son la señal de sincronía y el bit B_2 del microcontrolador. La señal resultado es el reloj del contador 74HC163.

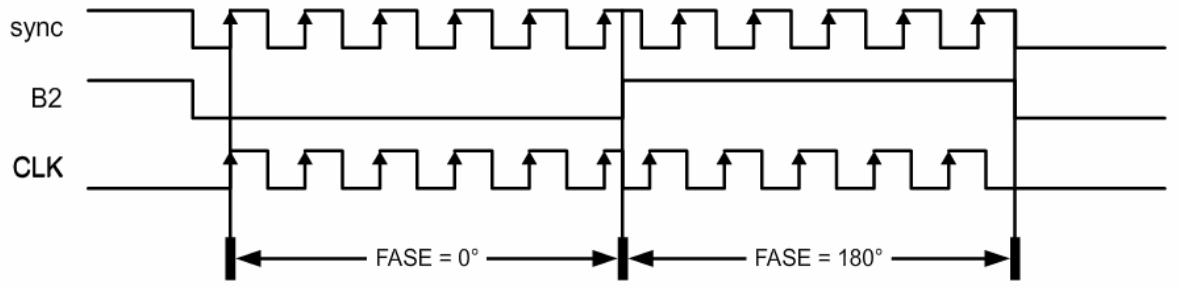


Figura 2-13 Formas de onda asociadas al parámetro de fase

Es importante hacer notar que el amplificador PA90 está implementado como inversor, de modo que si el bit B2 es igual a '0', la fase del tren generado es de 180° mientras que si es '1' la fase es de 0°.

B2	FASE
0	180°
1	0°

Tabla 2-4 Parámetro de fase

Parámetro de número de ciclos

La *figura 2-14* representa la circuitería involucrada en el procesamiento del parámetro de número de ciclos.

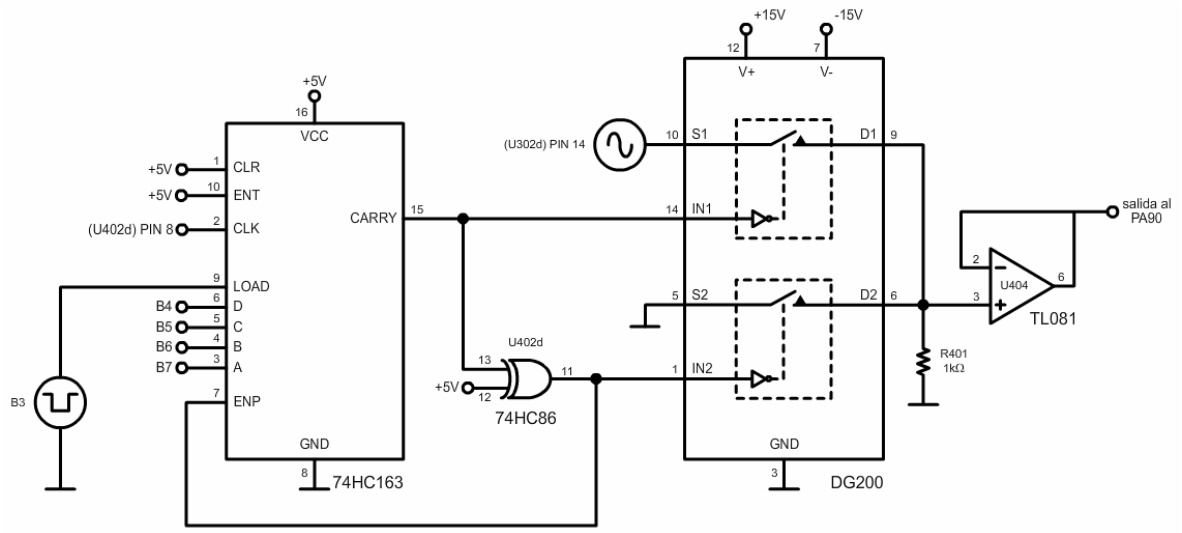


Figura 2-14 Procesamiento del parámetro de número de ciclos

Los bits $B_7B_6B_5B_4$ del microcontrolador definen la carga paralela del contador 74HC163 mientras que la salida CARRY habilita o deshabilita la operación de conteo. Esto se debe a que ingresa a la compuerta U_{402d} (implementada como inversor) y la salida de ésta a la entrada de habilitación ENP.

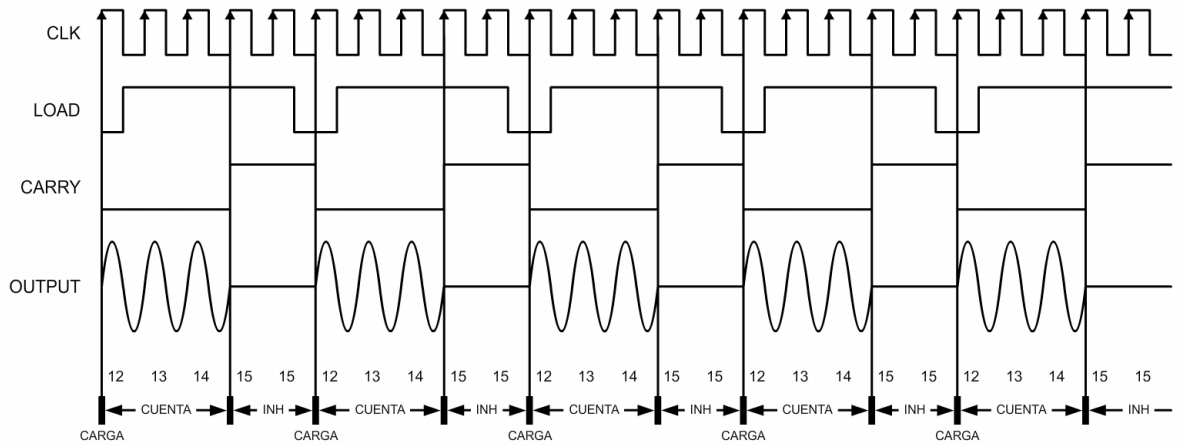


Figura 2-15 Formas de onda asociadas al parámetro de número de ciclos

El bit B_3 del microcontrolador genera una señal cuadrada de periodo y ciclo de trabajo variables. Esta señal se usa como la habilitación LOAD de la carga paralela del contador.

Por otra parte, el interruptor analógico DG200 se encarga de formar el tren de ondas sinusoidales al conmutar alternadamente sus dos interruptores. Las líneas CARRY y ENP del contador proporcionan las señales de habilitación de los interruptores mientras que las entradas son una onda sinusoidal sin nivel de DC y la línea de tierra.

A fin de determinar el número de pulsos debemos seguir los siguientes pasos:

a) Ajustar el número de ciclos.

b) Cálculo de tiempos de la señal LOAD.

Ajuste del número de ciclos

Ya que el bit más significativo del puerto B corresponde al bit menos significativo de la carga paralela del contador, es necesario ajustar el valor de la carga paralela de acuerdo con la siguiente tabla.

# ciclos	$(CP)_{10}$	$(CP)_{16}$	$(CP)_2$	$(CPA)_2$	$(CPA)_{16}$	$(CPA)_{10}$
0	15	F	1111	1111	F	15
1	14	E	1110	0111	7	7
2	13	D	1101	1011	B	11
3	12	C	1100	0011	3	3
4	11	B	1011	1101	D	13
5	10	A	1010	0101	5	5
6	9	9	1001	1001	9	9
7	8	8	1000	0001	1	1
8	7	7	0111	1110	E	14
9	6	6	0110	0110	6	6
10	5	5	0101	1010	A	10
11	4	4	0100	0010	2	2
12	3	3	0011	1100	C	12
13	2	2	0010	0100	4	4
14	1	1	0001	1000	8	8
15	0	0	0000	0000	0	0

Tabla 2-5 Ajuste de la carga paralela del contador

Tiempos de la señal LOAD

La *figura 2-16* muestra el tren de ondas sinusoidales obtenido a la salida del interruptor DG200. Durante el tiempo RET , la salida del generador es nula y por lo tanto, la operación de conteo debe ser deshabilitada.

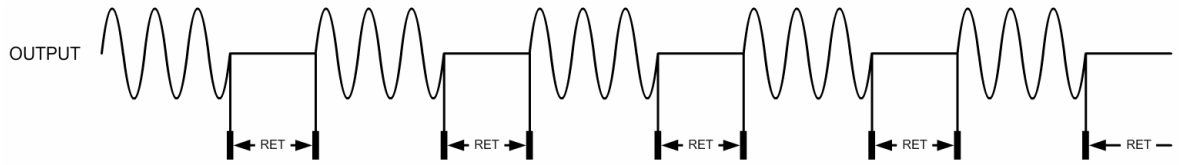


Figura 2-16 Salida del DG200

Para garantizar que un dato se cargue a la cuenta del 74HC163 el bit LOAD debe ser '0' durante la transición de bajo a alto del reloj. Ya que la frecuencia del reloj es la misma que la frecuencia de la senoide generada, el tiempo en bajo se define como:

$$DELB = \frac{1}{f} \tag{2.18}$$

Para el tiempo en alto, además del retardo, es necesario considerar el tiempo correspondiente al número de ciclos menos uno.

$$DELA = RET + (np - 1)DELB \tag{2.19}$$

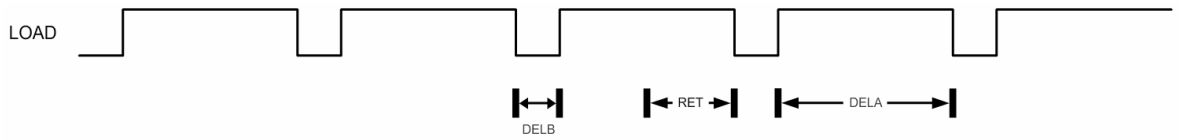


Figura 2-17 Habilitación de la carga paralela LOAD

Resumen

Para el parámetro de amplitud se tiene el siguiente procedimiento:

1. Cálculo del voltaje de modulación. $V_{mod} = 2Amplitud + 7.5$
2. Cálculo del voltaje en C_{301}. $V_{C_{301}} = 15 - \frac{V_{mod}}{2}$
3. Cálculo del ciclo de trabajo. $\delta_{CCP1} = -6.6674\ V_{C_{301}} + 99.9248$

Para el parámetro de frecuencia el procedimiento es:

1. Selección del capacitor. (*Tabla 2-2*)
2. Cálculo de la corriente. $I_T = \frac{f\,[\,\mathrm{Hz}\,]C\,[\,\mu\mathrm{F}\,]}{320}\,[\,\mathrm{mA}\,]$
3. Cálculo del ciclo de trabajo. $\delta_{CCP2} = -303.95\ I_T + 99.7148$

Para el parámetro de fase se lleva acabo el siguiente procedimiento:

1. Selección del bit B2. (*Tabla 2-4*)

Para el parámetro de número de ciclos las etapas del procedimiento son:

1. Ajuste del número de ciclos. (*Tabla 2-5*).
2. Cálculo del tiempo en bajo. $DELB = \frac{1}{f}$
3. Cálculo del tiempo en alto. $DELA = RET + (np - 1)DELB$

Capítulo 3: Desarrollo del software de control

Descripción

El software de control condensa las instrucciones necesarias para que el circuito de control gobierne la operación del generador de funciones. Después de procesar un mensaje ASCII, administra los recursos del microcontrolador para proporcionar los parámetros involucrados en la generación de la señal de excitación.

Las tareas que realiza el software de control son las siguientes:

- Inicializar el sistema.
- Recibir caracteres ASCII en forma serial.
- Verificar y decodificar los caracteres recibidos.
- Generar dos ondas cuadradas moduladas por ancho de pulso.
- Proporcionar el valor del selector de frecuencia.
- Proporcionar el bit de fase: 0° o 180°.
- Proporcionar la carga paralela (CP) del contador.
- Generar la señal de habilitación para la carga paralela (LOAD) del contador.

Ya que el programa está diseñado en forma modular, cada tarea está asociada a un grupo de subrutinas para su ejecución y pueden clasificarse en dos grupos:

- **Tareas Principales**. Son aquellas tareas que se ejecutan dentro del programa principal, como son la inicialización y la generación de la señal LOAD.
- **Tareas de Interrupción**. Son las tareas que se ejecutan en la sección de interrupciones, como la recepción, validación y decodificación del mensaje de control, así como la actualización de los parámetros del generador.

Consideraciones importantes

- La frecuencia de operación de los módulos CCP1 y CPP2 es de 10 [kHz].
- El tiempo de oscilación T_{osc} se define como el inverso de la frecuencia del oscilador. El cristal empleado es de 16 [MHz], de modo que $T_{osc} = 62.5$ [ns].
- Un ciclo de máquina está compuesto por cuatro ciclos de reloj por lo que la frecuencia de operación es un cuarto de la frecuencia de oscilación $f_{op} = 0.25\ f_{osc}$.

Estructura del programa principal

Dentro del software de control, el programa principal tiene las funciones de inicializar los módulos del sistema y generar la señal de habilitación para la carga paralela (LOAD) del contador. Dicha señal se obtiene mediante la transición cíclica de un nivel alto a un nivel bajo de la salida B_3 del microcontrolador.

Descripción del algoritmo

La inicialización contiene las instrucciones necesarias para configurar los módulos del microcontrolador. Esta sección realiza las siguientes tareas:

a) Configurar los puertos A, B y C como salidas. En el caso del puerto A, define sus líneas como digitales.

b) Seleccionar la operación PWM de los módulos CCP1 y CCP2 estableciendo una frecuencia de operación de 10 [kHz].

c) Configurar el módulo USART como *full duplex* para la interacción con la interfaz de comunicación

d) Habilitar los permisos de interrupción para la recepción asíncrona del mensaje de control por medio del módulo USART.

Para generar la señal LOAD, el programa principal enciende y apaga la salida B_3 del microcontrolador. El tiempo que B_3 permanece prendido o apagado se calcula con las ecuaciones (3.18) y (3.19), pudiendo variar entre 1 [μs] y 10 [ms]. Debido a que el intervalo es muy amplio, se necesitan dos subrutinas de retardo. La primera de un nivel de anidación y la segunda de dos niveles.

Ya que cada llamada a subrutina conlleva un retardo inherente de 1 [μs][1], es necesario implementar el primer retardo dentro del programa principal. De este modo, se garantiza el intervalo de operación del generador de funciones.

[1] La llamada y el retorno de subrutina suman cuatro ciclos de instrucción.

Estructura de la interrupción

La función principal de la interrupción consiste en identificar el instante en el que la interfaz de comunicación inicia la transmisión del mensaje de control. Para lograrlo, se emplea la interrupción por recepción del modulo USART.

Como tareas secundarias, la interrupción establece el inicio para validar, decodificar y actualizar los parámetros del generador de funciones una vez que la recepción del mensaje de control ha sido completada.

Descripción del algoritmo

Cada vez que el módulo USART recibe un dato de la interfaz de comunicación, la interrupción detiene la generación de la señal LOAD y destina los recursos del microcontrolador para guardar, en memoria, el dato recibido. Cuando la tarea es completada, se retoma el programa donde estaba en el momento de ser interrumpido, reanudando así la generación de la señal LOAD.

La interrupción seguirá guardando los datos recibidos hasta que la memoria destinada al mensaje se llene. En este punto, se verificará que el último carácter recibido corresponda al terminador, que en este caso es el *carry return*. Si el terminador es correcto, se procede a validar y decodificar cada uno de los caracteres que conforman al mensaje.

Una vez que se ha determinado que el mensaje es válido y que ha sido decodificado, la interrupción ordena y actualiza los datos recibidos garantizando así, que la onda presente los nuevos parámetros que el usuario desea.

Como último paso, se apunta al inicio de memoria del mensaje en espera de un nuevo cambio en los parámetros. Si se presenta un error en la verificación del terminador o en la validación del mensaje, se termina el procesamiento y también se reinicializan los apuntadores.

Subrutinas

La sección de subrutinas contiene las instrucciones necesarias para que el microcontrolador ejecute las tareas del software de control. Por su naturaleza, las subrutinas pueden separarse en dos grupos:

- **Subrutinas de configuración**. Son aquellas que establecen el modo de operación de los diferentes módulos del microcontrolador.

- **Subrutinas de procesamiento**. Son las que intervienen en el procesamiento del mensaje de control.

Subrutinas de configuración

INICIALIZA

Esta subrutina inicializa los módulos del PIC16F873 con la siguiente secuencia:

1. Inicializa los puertos de entrada/salida con la subrutina INIES.
2. Inicializa los módulos CCP1 y CCP2 con la subrutina INIPWM.
3. Inicializa el módulo USART con la subrutina INIUSART.
4. Habilita la interrupción por recepción del USART con la subrutina HABINT.
5. Carga la configuración predeterminada mediante la subrutina DEFCON.
6. Decodifica el mensaje predeterminado con la subrutina VDMEN.
7. Ordena el mensaje predeterminado con la subrutina ORDENA.
8. Actualiza los parámetros del GFAT con la subrutina ACTUALIZA.
9. Reinicia los apuntadores con la subrutina INIAP.

INIES

Esta subrutina inicializa los puertos A, B y C con la siguiente secuencia:

1. Limpia el registro correspondiente al puerto.
2. En el caso del puerto A, establece la naturaleza de sus líneas: analógica o digital.
3. Define si las líneas del puerto serán entradas o salidas.

Registro del puerto

TRISA

---	---	Registro para la dirección de datos del puerto A

bit7 bit0

TRISB

Registro para la dirección de datos del puerto B

bit7 bit0

TRISC

Registro para la dirección de datos del puerto C

bit7 bit0

Líneas analógicas o digitales

Ya que el puerto A está multiplexado con el convertidor analógico digital, debemos especificar la función que tomarán sus líneas. Esto se logra escribiendo a los bits PCFG3:PCFG0 del registro ADCON1 de acuerdo con la siguiente tabla:

ADCON1

ADFM	---	---	---	**PCFG3**	**PDFG2**	**PCFG1**	**PCFG0**

bit7 bit0

PCFG3: PCFG0	AN4 RA4	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	V_{REF+}	V_{RE-}
0000	A	A	A	A	A	V_{DD}	V_{SS}
0001	A	V_{REF+}	A	A	A	RA3	V_{SS}
0010	A	A	A	A	A	V_{DD}	V_{SS}
0011	A	V_{REF+}	A	A	A	RA3	V_{SS}
0100	D	A	D	A	A	V_{DD}	V_{SS}
0101	D	V_{REF+}	D	A	A	RA3	V_{SS}
011x	**D**	**D**	**D**	**D**	**D**	**V_{DD}**	**V_{SS}**
1000	A	V_{REF+}	V_{REF-}	A	A	RA3	RA2
1001	A	A	A	A	A	RA3	V_{SS}
1010	A	V_{REF+}	A	A	A	RA3	V_{SS}
1011	A	V_{REF+}	V_{REF-}	A	A	RA3	RA2
1100	A	V_{REF+}	V_{REF-}	A	A	RA3	RA2
1101	D	V_{REF+}	V_{REF-}	A	A	RA3	RA2
1110	D	D	D	D	A	V_{DD}	V_{SS}
1111	D	V_{REF+}	V_{REF-}	D	A	RA3	RA2

Tabla 3-1 Configuración del puerto A

A = Entrada analógica
D = Entrada/Salida digital

Entradas o salidas

A fin de establecer una línea como entrada o salida, debemos escribir al registro de direccionamiento del puerto correspondiente. Para una entrada se escribe un ‘1’ mientras que para una salida se escribe un ‘0’.

Para la aplicación:

- Como primer paso, se escribe 00h a los registros PORTA, PORTB y PORTC.
- El puerto A se define como digital, escribiendo 06h al registro ADCON1.
- Como los tres puertos son salidas, se escribe 00h a los registros TRISA, TRISB y TRISC.

INIPWM

Esta subrutina establece la operación PWM en los módulos CCP1 y CCP2 con la siguiente secuencia:

1. Definir el periodo de operación.
2. Establecer el ciclo de trabajo.
3. Establecer el pin CCPx como salida.
4. Habilitar el Timer2.
5. Configurar el módulo CCPx para la operación PWM.

Periodo de operación

El Timer2 es empleado por módulos CCP1 y CCP2, de manera que ambos presentan el mismo periodo de operación. Éste se establece escribiendo al registro PR2 mediante la siguiente expresión:

$$PWM_{per} = \left[(PR2)+1\right] \cdot 4 \cdot T_{osc} \cdot \left(TMR2_{pres}\right) \qquad (3.01)$$

donde:

· $PWM_{per} = 100\,[\,\mu s\,]$ · $T_{osc} = 62.5\,[\,ns\,]$ · $TMR2_{pres} = 16$

Despejando PR2:

$$PR2 = 24$$

PR2

Registro del periodo del Timer2	
bit7	bit0

Ciclo de trabajo

El ciclo de trabajo del PWM se especifica al escribir en el registro CCPRxL y en los bits <5:4> del registro CCPxCON. El CCPRxL contiene los 8 bits más significativos y el CCPxCON<5:4> contiene los dos bits menos significativos.

Este valor de 10 bits se representa por CCPx (CCPRxL:CCPxCON<5:4>). La siguiente ecuación se utiliza para calcular el ciclo de trabajo del PWM en el tiempo:

$$PWM_{\delta} = (CCP) \cdot T_{osc} \cdot (TMR2_{pres}) \tag{3.02}$$

donde:

· $PWM_{\delta} = \left(\frac{\delta\%}{100}\right)(PWM_{per})$ · $T_{osc} = 62.5\,[\text{ ns }]$ · $TMR2_{pres} = 16$

Despejando CCP:

$$CCP = \delta\% \tag{3.03}$$

CCPR1L / CCPR2L

Resgistro captura/comparación/PWM (bits menos significativos)

bit7 bit0

CCP1CON / CCP2CON

---	---	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0

bit7 bit0

Salida CCPx

Para establecer el pin CCPx como salida, se escribe un ‘0’ al bit <2> o al bit <1> del registro TRISC dependiendo del módulo deseado.

TRISC

TRISC<7>	TRISC<6>	TRISC<5>	TRISC<4>	TRISC<3>	**TRISC<2>**	**TRISC<1>**	TRISC<0>

bit7 bit0

Habilitación del Timer2.

Para habilitar el Timer2, se escribe ‘1’ al bit TMR2ON del registro T2CON y para que $TMR2_{pres} = 16$, se escribe ‘11’ a los bits T2CKPS1:T2CKPS0 del registro T2CON.

T2CON

---	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	**TMR2ON**	**T2CKPS1**	**T2CKPS0**

bit7 bit0

Operación PWM.

Para configurar el módulo CCPx como PWM se escribe ‘1100’ a los bits CCPxM3:CCPxM0 del registro CCPxCON.

CCP1CON / CCP2CON

---	---	CCPxX	CCPxY	**CCPxM3**	**CCPxM2**	**CCPxM1**	**CCPxM0**
bit7							bit0

Para la aplicación:

- Especificamos una frecuencia de operación de 10 [kHz] escribiendo 18h al registro PR2.
- El ciclo de trabajo se establece escribiendo el valor de CCP en los registros CCPR1L:CCP1CON<5:4>
- Los bits CCP1 y CCP2 se habilitan como salidas al escribir ‘0’ a los bits <2:1> del registro TRISC.
- El Timer2 se enciende escribiendo ‘1’ al bit TMR2ON del registro T2CON.
- La operación PWM se selecciona al escribir ‘1100’ a los bits CCP1M3:CCP1M0 y CCP2M3:CCP2M0 de los registros CCP1CON y CCP2CON respectivamente.

INIUSART

Esta subrutina configura la recepción asíncrona del módulo USART con la siguiente secuencia:

1. Configurar las líneas <7:6> del puerto C como canales del USART.
2. Definir la velocidad de transmisión.
3. Habilitar el puerto serie asíncrono.
4. Configurar la recepción de 8 ó 9 bits.
5. Habilitar la recepción.

Líneas de entrada-salida del USART

Ya que las líneas <7:6> del puerto C están multiplexadas con el módulo USART, es necesario especificar su función. Escribir '11' a los bits TRISC<7:6> resultará en la habilitación de los canales de recepción y transmisión (RX y TX) del módulo USART.

TRISC

TRISC<7>	**TRISC<6>**	TRISC<5>	TRISC<4>	TRISC<3>	TRISC<2>	TRISC<1>	TRISC<0>
bit7							bit0

Generador de velocidad de transmisión binario (BRG)

El BRG es un generador de 8 $^{bits}/_{seg}$. El registro SPBRG controla el periodo de un contador que corre libre de 8 bits. En el modo asíncrono, el bit BRGH (TXSTA<2>) también controla la velocidad de transmisión.

SPBRG

Registro para la velocidad de transmisión							
bit7							bit0

TXSTA

CSRC	TX9	TXEN	SYNC	---	**BRGH**	TRMT	TX9D
bit7							bit0

Vel. de transmisión [kbytes]	kbaud	%error	$(SPBRG)_{10}$
0.3	---	---	---
1.2	1.202	0.17	207
2.4	2.404	0.17	103
9.6	**9.615**	**0.16**	**25**
19.2	19.231	0.16	12
28.8	27.778	3.55	8
33.6	35.714	6.29	6
57.6	62.500	8.51	3
HIGH	0.977	---	255
LOW	250.000	---	0

Tabla 3-2a BRGH=0

Vel. de transmisión [kbytes]	kbaud	%error	$(SPBRG)_{10}$
0.3	---	---	---
1.2	---	---	---
2.4	---	---	---
9.6	**9.615**	**0.16**	**103**
19.2	19.231	0.16	51
28.8	29.412	2.13	33
33.6	33.333	0.79	29
57.6	58.824	2.13	16
HIGH	3.906	---	255
LOW	1000.000	---	0

Tabla 3-2b BRGH=1

Puerto serie asíncrono

El módulo USART se configura como asíncrono al escribir ‘0’ al bit SYNC (TXSTA<4>) mientras que el puerto serie se habilita al escribir ‘1’ al bit SPEN (RCSTA<7>).

TXSTA

CSRC	TX9	TXEN	**SYNC**	---	BRGH	TRMT	TX9D
bit7							bit0

RCSTA

SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit7							bit0

Recepción en 8 bits

Para configurar la recepción de 8 bits se escribe ‘0’ al bit RX9 (RCSTA<6>).

RCSTA

SPEN	**RX9**	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit7							bit0

Habilitación de la recepción

Para habilitar la recepción continua del módulo USART se escribe ‘1’ al bit CREN (RCSTA<4>).

RCSTA

SPEN	RX9	SREN	**CREN**	ADDEN	FERR	OERR	RX9D
bit7							bit0

Para la aplicación:

- Se establece una tasa de transmisión de 9.6 kbytes al escribir 19h al registro SPBRG.
- Se configura el USART en modo asíncrono escribiendo 20h al registro TXSTA.
- Para habilitar el puerto serie y la recepción continua de 8 bits se escribe 90h al registro RCSTA.

HABINT

Esta subrutina habilita la interrupción por recepción del módulo USART con la siguiente secuencia:

1. Habilitar el permiso de interrupción por recepción del módulo USART.

2. Habilitar el permiso de interrupción de periféricos y el permiso global.

Permiso local

Para habilitar la interrupción de recepción escribimos ‘1’ al bit TMR2IE del registro PIE1.

PIE1

PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	**TMR2IE**	TMR1IE
bit7							bit0

Permiso global y de periféricos

Para habilitar el permiso de periféricos y el permiso global, escribimos ‘1’ a los bits PEIE y GIE del registro INTCON.

INTCON

GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit7							bit0

Subrutinas de procesamiento

INIAP

Esta subrutina apunta a la primera localidad de memoria del mensaje de control con la siguiente secuencia:

1. Obtener la dirección de la primera localidad del mensaje de control.

2. Transferir la dirección al registro FSR.

CARGAT

Esta subrutina carga el carácter *"carry return"* en el registro W y apunta al último carácter del mensaje recibido con la siguiente secuencia:

1. Se obtiene la dirección del primer carácter del mensaje.

2. A la dirección obtenida, se le suma el tamaño del mensaje.

3. El resultado obtenido se transfiere al registro apuntador FSR.

4. Se resta 01h al contenido de FSR. Como resultado, FSR se encuentra apuntando al último carácter del mensaje.

5. Finalmente, se carga el registro W con el carácter *"carry return"*.

GDATO

Esta subrutina guarda los datos recibidos con la siguiente secuencia:

1. Transfiere el dato recibido a la zona de memoria destinada para el mensaje mediante direccionamiento indirecto.

2. Incrementa el apuntador.

VDMEN

Esta subrutina valida el mensaje de control con la siguiente secuencia:

1. Borra la bandera de error.

2. Apunta al inicio del mensaje y establece el número de datos a evaluar.

3. Ejecuta la subrutina VALDEC para determinar si el dato actual es válido o no.

4. Si el dato es inválido, enciende la bandera de error VDMEF.

5. Si el dato es válido, incrementa el apuntador y verifica si se han validado todos los caracteres del mensaje.

6. Si se ha validado todo el mensaje se termina la subrutina. Si faltan datos por evaluar, se vuelve al paso 2.

VALDEC

Esta subrutina valida y decodifica los datos recibidos con la siguiente secuencia:

1. Borra la bandera de error.

2. Verifica el límite inferior del primer intervalo. De ser erróneo, enciende la bandera de error VDEF y termina la subrutina.

3. Verifica el límite superior del primer intervalo. Si el dato pertenece a este intervalo, se decodifica restándole 30h y termina la subrutina.

4. Si el límite superior del primer intervalo es incorrecto, se evalúa el límite inferior del segundo intervalo. De ser erróneo, enciende la bandera de error VDEF y termina la subrutina.

5. Verifica el límite superior del segundo intervalo. Si el dato pertenece a este intervalo, se decodifica restándole 37h y termina la subrutina.

6. Si el límite superior del segundo intervalo es incorrecto, se enciende la bandera de error VDEF y termina la subrutina.

ORDENA

Esta subrutina reorganiza los datos decodificados del mensaje de control con la siguiente secuencia:

1. Los datos LIST01, LIST02 y LIST03 se reorganizan en los registros dc1H y dc1L.

2. Los datos LIST04, LIST05 y LIST06 se reorganizan en los registros dc2H y dc2L.

3. Los datos LIST07, LIST08, LIST09 y LIST10 se transfieren a los registros FASE, RELAY, NP y NPA respectivamente.

4. Los datos LIST11 y LIST12 se reorganizan en el registro DELA.

5. Los datos LIST13 y LIST14 se reorganizan en el registro DELB.

6. Los datos LIST15 y LIST16 se transfieren a los registros HRT y HAB.

ACTUALIZA

Esta subrutina actualiza los parámetros de control con la siguiente secuencia:

1. Actualiza el parámetro de amplitud, escribiendo el nuevo ciclo de trabajo a CCPR1L:CCP1CON<5:4>.

2. Actualiza el parámetro de frecuencia, escribiendo el nuevo ciclo de trabajo a CCPR2L:CCP2CON<5:4> y escribiendo el nuevo valor del selector de frecuencia al puerto A.

3. Actualiza el número pulsos escribiendo el valor de NPA a los bits <7:4> del puerto B.

4. Actualiza la fase encendiendo o apagando el bit <2> del puerto B.

DEFCON

Esta subrutina carga la configuración predeterminada del GFAT con la siguiente secuencia:

1. Apunta al inicio del mensaje predeterminado y a la primera localidad del mensaje de control.

2. Mediante la subrutina TABLA, se transfiere un carácter del mensaje predeterminado a la localidad de memoria correspondiente del mensaje de control.

3. Incrementa los apuntadores y verifica si se ha llegado al fin del mensaje predeterminado.

4. Si se ha cargado todo el mensaje predeterminado, se termina la subrutina. Si aún faltan caracteres por cargar, se vuelve al paso 2.

TABLA

Esta subrutina apunta al mensaje predeterminado con la siguiente secuencia:

1. Apunta al inicio del mensaje predeterminado.

2. Por medio de direccionamiento indizado, tiene acceso al caracter especificado por el *offset*.

3. Carga el caracter seleccionado al registro W.

DELAY1

Esta subrutina genera un retardo de un nivel de anidación con la siguiente secuencia:

1. Se carga el único contador con el número de eventos.

2. Se decrementa el contador en una unidad.

3. Se verifica el valor del contador. Si es diferente de cero se vuelve al paso 2. Si es igual a cero, se termina la subrutina.

El código empleado para el ciclo de iteración es el siguiente:

```
LOOP1N   DECFSZ   B1,F     1
         GOTO     LOOP1N   2
```

$3B1+1$

Podemos observar que:

- La instrucción DECFSZ decrementa el contador y evalúa si su contenido es igual a cero en 1 ciclo de instrucción.

- La instrucción GOTO reposiciona el contador de programa para un nuevo decremento en 2 ciclos de instrucción.

- Estos 3 ciclos de instrucción se repiten tantas veces como el valor del contador.

- Cuando la instrucción DECFSZ determina que el valor del contador es cero, se ejecuta una NOP en lugar la instrucción siguiente, haciendo que DECFSZ tenga un ciclo de instrucción adicional.

La duración del retardo es:

$$t = (3B1+1)\left(f_{op}^{-1}\right)$$

y despejando B1 tenemos que:

$$B1 = \frac{t(f_{op})-1}{3} \qquad (3.04)$$

Finalmente:

- El error teórico máximo es de 12.5% mientras que el error teórico promedio es de 0.74%.

DELAY2

Esta subrutina genera un retardo de dos niveles de anidación con la siguiente secuencia:

1. Se cargan el número de eventos para los contadores de 1er y 2do nivel. Así mismo, se crea un respaldo del contador del 1er nivel.
2. Se decrementa el contador de primer nivel hasta que llega a cero. A continuación se restaura su valor con ayuda del respaldo.
3. Se decrementa el contador de segundo nivel y se verifica su valor. Si es diferente de cero se vuelve al paso 2. Si es igual a cero, se termina la subrutina.

El código empleado para el ciclo de iteración es el siguiente:

LOOP2N	DECFSZ	B1,F	1	$3B1+1$	$(3B1+6)B2+1$
	GOTO	LOOP2N	2		
	MOVF	BH1,W	1	5	
	MOVWF	B1	1		
	DECFSZ	B2,F	1		
	GOTO	LOOP2N	2		

Podemos observar que:

- La primera parte del código corresponde a un retardo de 1 nivel.
- Las instrucciones que restauran el valor del primer contador y que evalúan el contenido del segundo suman 5 ciclos de instrucción.
- Estos $3B1+6$ ciclos de instrucción se repiten tantas veces como el valor del segundo contador.
- Cuando el segundo contador llega a cero se produce un ciclo de instrucción adicional.

La duración del retardo es:

$$t=\left[(3B1+6)B2+1\right]\left(f_{op}{}^{-1}\right)$$

y despejando B2 tenemos que:

$$B2=\frac{t\left(f_{op}\right)-1}{3B1+6} \tag{3.05}$$

Finalmente:

- B1 se asigna con 41h. El error teórico máximo es de 0.75% mientras que el error teórico promedio es de 0.36%.

DELAY3

Esta subrutina genera un retardo de tres niveles de anidación con la siguiente secuencia:

1. Se cargan el número de eventos para los contadores de primer, segundo y tercer nivel. Así mismo, se crea un respaldo de los contadores de primer y segundo nivel.

2. Se decrementa el contador de 1er nivel hasta que llega a cero y se restaura su valor.

3. Se decrementa el contador de 2do nivel. Si es diferente de cero se vuelve al paso 2. Si es igual a cero se restaura su valor.

4. Se decrementa el contador de 3er nivel y se verifica su valor. Si es diferente de cero se vuelve al paso 2. Si es igual a cero, se termina la subrutina.

El código empleado para el ciclo de iteración es el siguiente:

LOOP3N	DECFSZ	B1,F	1	$3B1+1$	$(3B1+6)B2+1$	$(3B1B2+6B2+6)B3+1$
	GOTO	LOOP3N	2			
	MOVF	BH1,W	1	5		
	MOVWF	B1	1			
	DECFSZ	B2,F	1			
	GOTO	LOOP3N	2			
	MOVF	BH2,W	1	5		
	MOVWF	B2	1			
	DECFSZ	B3,F	1			
	GOTO	LOOP3N	2			

Podemos observar que:

- La primera parte del código corresponde a un retardo de 2 niveles.
- Las instrucciones que restauran el valor del segundo contador y que evalúan el contenido del tercero suman 5 ciclos de instrucción.
- Estos $3B1B2+6B2+6$ ciclos de instrucción se repiten tantas veces como el valor del tercer contador.
- Cuando el tercer contador llega a cero se produce un ciclo de instrucción adicional.

La duración del retardo es:

$$t=\left[\left(3B1B2+6B2+6\right)B3+1\right]\left(f_{op}{}^{-1}\right)$$

y despejando B1 tenemos que:

$$B3=\frac{t\left(f_{op}\right)-1}{3B1B2+6B2+6} \tag{3.06}$$

Finalmente:

- B1 y B2 se asignan con FFh y 03h respectivamente. El error teórico máximo es de 1.89% mientras que el error teórico promedio es de 0.37%.

Capítulo 4: Desarrollo de la interfaz de comunicación

Descripción

La interfaz de comunicación se encarga de generar el mensaje de control a partir de los parámetros de amplitud, frecuencia, fase, número de ciclos y retardo suministrados por el usuario y está conformada por los siguientes bloques:

- **Captura de datos**. En este bloque se introducen los parámetros que presentará la onda generada.

- **Procesamiento**. Este bloque genera un mensaje de control de 16 caracteres ASCII a partir de los parámetros deseados. Así mismo, proporciona la habilitación para iniciar la transmisión del mensaje de control.

- **Transmisión serial**. Este bloque transmite en forma serial el mensaje de control por medio del protocolo VISA.

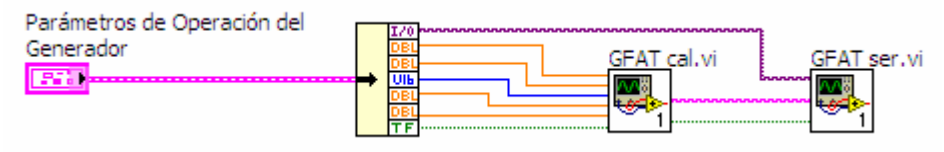


Figura 4-1 Diagrama de bloques de la interfaz de comunicación GFATcom

El procesamiento y la transmisión serial se encuentran en instrumentos virtuales (*VIs*) independientes y se describen a continuación.

GFAT cal

Este VI genera el mensaje de control a partir de los parámetros de la onda que se desea generar. Para ello, cada parámetro se procesa hasta obtener un equivalente de caracteres ASCII. Posteriormente se concatenan los caracteres procesados en una sola cadena denominada *Mensaje de control* y se determina si se concede o no el permiso de transmisión.

Cadena CCP1

1. Se calcula el ciclo de trabajo para la amplitud deseada (ecuación 3.09) y se expresa en binario.
2. Se agrupan los bits en tres números binarios. El primero se forma con los bits <9:6>, el segundo con los bits <5:2> y el tercero con los bits <1:0>.
3. Cada número binario se convierte a una cadena hexadecimal.
4. Se concatenan las tres cadenas.

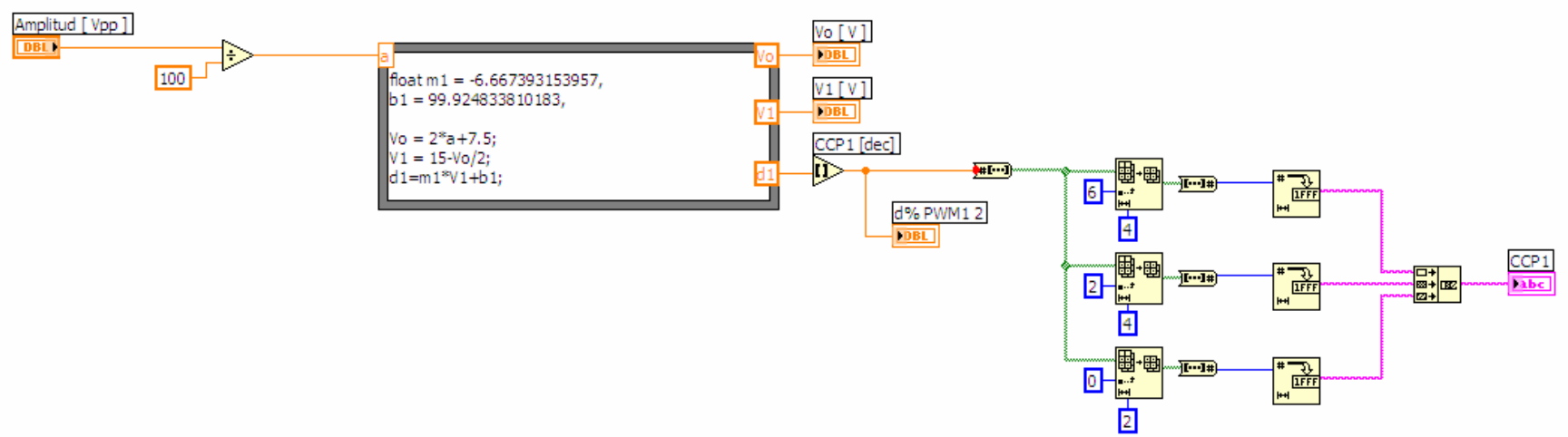


Figura 4-2 Cálculo de CCP1

Cadena CCP2 & RELAY

1. Se selecciona el valor del selector de frecuencia y se calcula el ciclo de trabajo para la frecuencia deseada (ecuación 3.17). El resultado se expresa en binario.
2. Se agrupan los bits en tres números binarios. El primero se forma con los bits <9:6>, el segundo con los bits <5:2> y el tercero con los bits <1:0>.
3. Cada número binario se convierte a una cadena hexadecimal.
4. Se convierte el valor del selector de frecuencia a una cuarta cadena hexadecimal.
5. Se concatenan las cuatro cadenas.

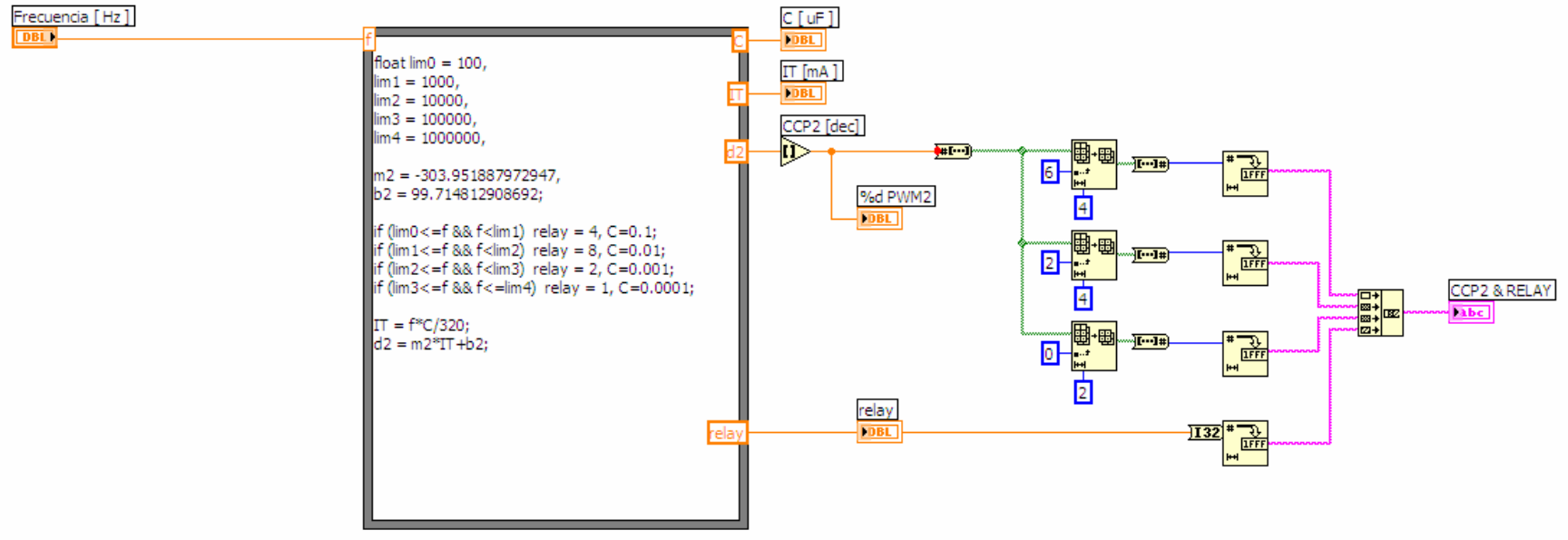


Figura 4-3 Cálculo de CCP2 & RELAY

Cadena FASE

1. El valor de fase (0 ó 1) se convierte a una cadena hexadecimal.

Figura 4-4 Cálculo de FASE

Cadena HAB

1. El valor de encendido (0 ó 1) se convierte a una cadena hexadecimal.

Figura 4-5 Cálculo de HAB

Cadena NP

1. El número de ciclos se convierte a una cadena hexadecimal. Si el GFAT está apagado, np = 0.
2. Se calcula el valor ajustado del número de ciclos y se convierte a una cadena hexadecimal.
3. Se concatenan las dos cadenas.

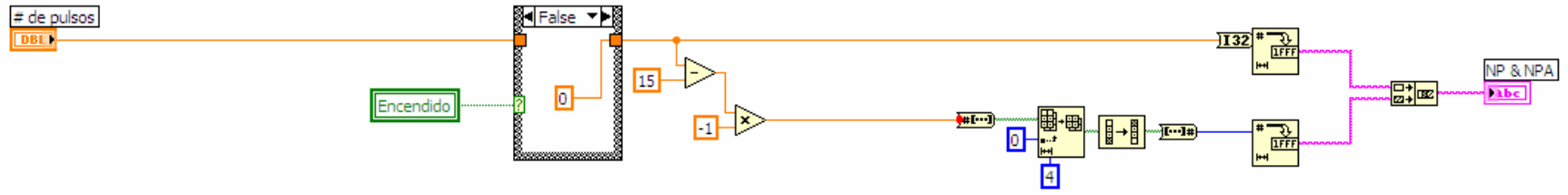


Figura 4-6 Cálculo de NP & NPA

Cadena DELA & DELB & HRT

1. Se calcula el número de iteraciones para el retardo deseado (ecuación 4.06) y se expresa en binario.
2. Se agrupan los bits en dos números binarios. El primero se forma con los bits <7:4> y el segundo con los bits <3:0>.
3. Se calcula el número de iteraciones para el tiempo en bajo de la señal LOAD (ecuaciones 4.04 y 4.05) y se expresa en binario.
4. Se agrupan los bits en dos números binarios. El primero se forma con los bits <7:4> y el segundo con los bits <3:0>.
5. Cada número binario se convierte a una cadena hexadecimal.
6. A partir del intervalo del selector se obtiene un bit auxiliar (HRT) y se convierte a una cadena hexadecimal.
7. Se concatenan las cinco cadenas

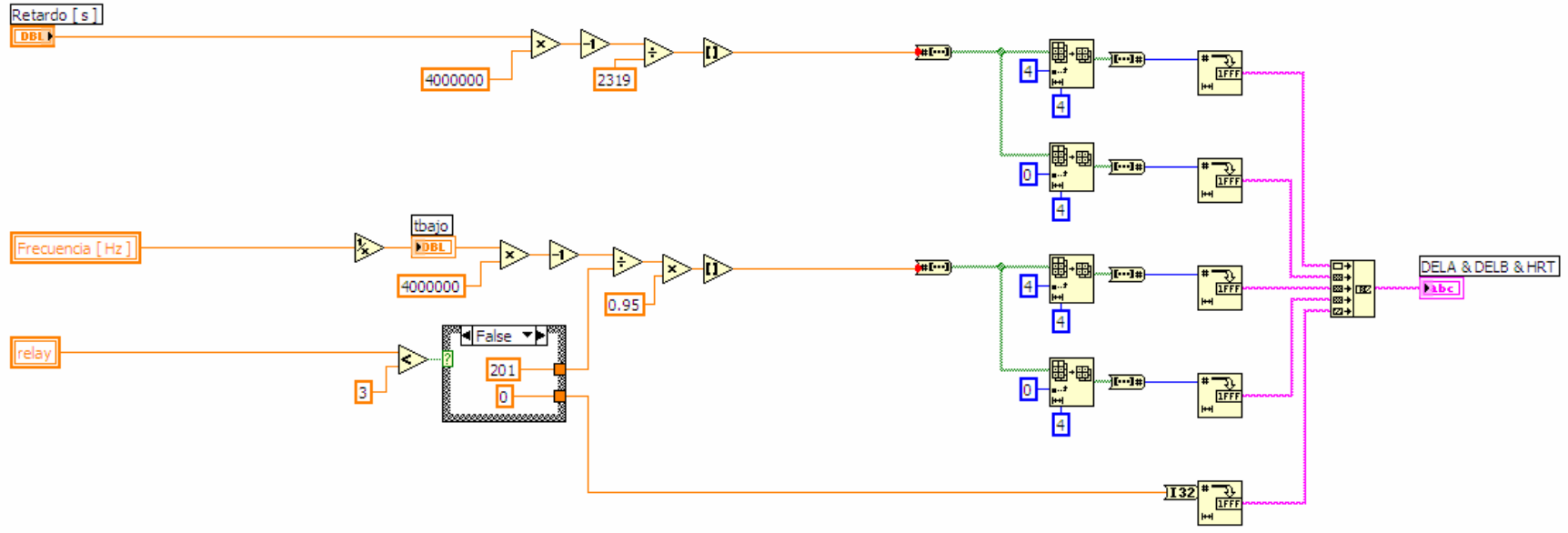


Figura 4-7 Cálculo de DELA & DELB & HRT

Una vez que se han obtenido las cadenas correspondientes a cada parámetro, se concatenan para formar el mensaje de control.

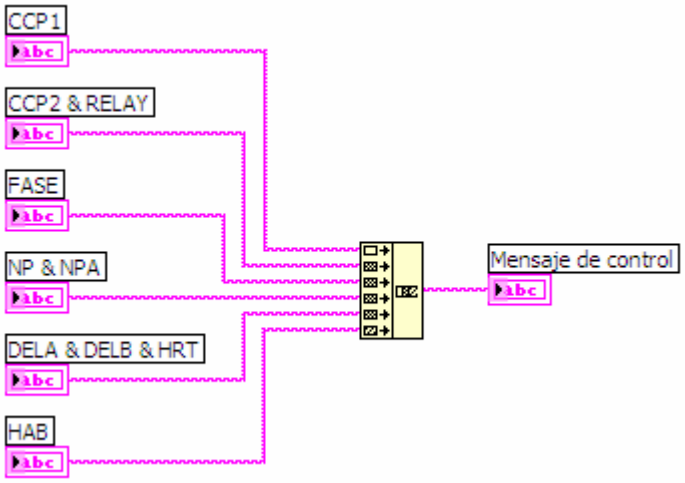


Figura 4-8 Mensaje de control

Permiso de transmisión

Para iniciar la transmisión de datos, se compara el mensaje de control con una cadena auxiliar. Si el resultado arroja que las dos cadenas son diferentes, implicando que el mensaje de control ha sido modificado, se concede el permiso de transmisión y se copia el contenido del mensaje de control a la cadena auxiliar.

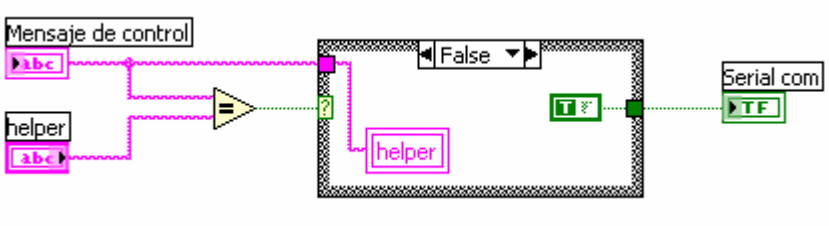


Figura 4-9 Permiso de transmisión concedido

Si en la comparación, el mensaje de control y la cadena auxiliar son iguales, el permiso de trasmisión no se concede.

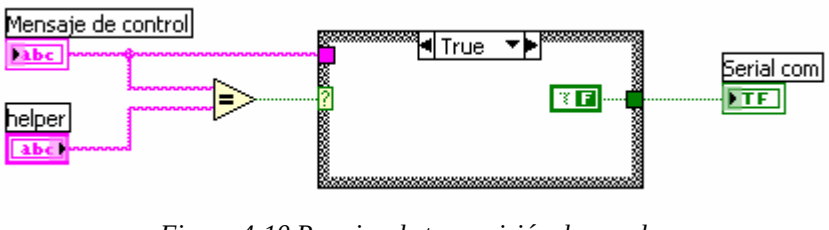


Figura 4-10 Permiso de transmisión denegado

Es así que el mensaje de control solo se transmite cuando se modifica alguno de los parámetros del generador de funciones.

GFAT ser

Este VI se encarga de transmitir el *mensaje de control* al microcontrolador PIC16F873 vía RS232. Para ello, se configura el puerto serie mediante el protocolo VISA y la transmisión se ejecuta en el instante que el permiso de transmisión es concedido.

1. Inicializa el puerto especificado con los siguientes parámetros:

Parámetro	Valor
Caracter terminador	habilitado
Terminador	carry return
Velocidad de transmisión	9600b/s
Bits de datos	8
Paridad	ninguna
Bits de parada	ninguno
Control de flujo	ninguno

Tabla 4-1 Configuración del puerto serie

2. Habilita la transmisión del *mensaje de control* mediante el permiso de transmisión.

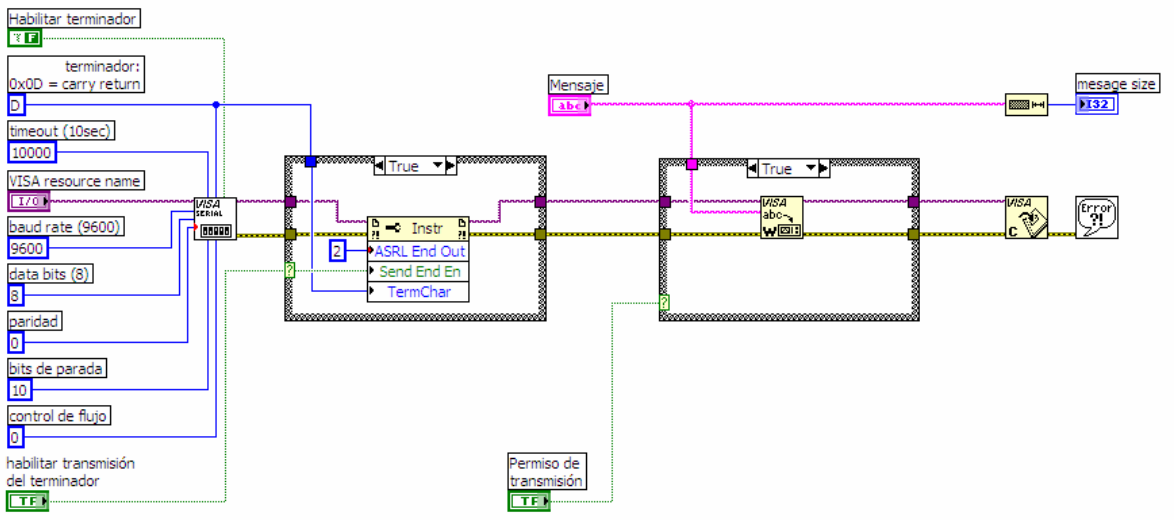


Figura 4-10 Transmisión del mensaje de control

Pruebas

Para probar la operación del software de control se realizaron dos tipos de pruebas:

- **Pruebas en simulación**. En estas pruebas, se aprovecha la aplicación MPLAB SIM del MPLAB IDE para verificar el procesamiento de los datos y el estado de los parámetros en memoria.

- **Pruebas en hardware**. En estas pruebas, se diseñó una tarjeta de evaluación que permite identificar el estado de los bits de carga paralela, fase y del selector de frecuencia así como las señales de CCP1, CCP2 y LOAD.

Pruebas de simulación

En esta etapa, se propusieron cuatro pruebas que evalúan la operación de las tareas de interrupción del software de control. Para ello, se emplea el *mensaje de control* para diferentes condiciones de la señal de excitación y se simula la operación del software de control con la herramienta *MPLAB SIM*. La *tabla 5-1a* enlista los parámetros usados en cada prueba.

Prueba	amp [Vpp]	fre [Hz]	fase	No. ciclos	ret [ms]	GFAT
1	50	100	negativa	1	10	apagado
2	100	1000	positiva	1	20	encendido
3	150	10000	negativa	2	30	encendido
4	200	100000	positiva	3	40	encendido

Tabla 5-1a Parámetros de la señal de excitación

La *tabla 5-1b* muestra el valor esperado que los registros de control deben presentar después de cada prueba. Todos los valores representan números hexadecimales.

Prueba	CCP1	CCP2	RELAY	FASE	NP	NPA	DELA	DELB	HRT	HAB
1	070C	162C	4	0	0	F	11	BD	0	0
2	080C	162C	8	1	1	7	22	13	0	1
3	083C	162C	2	0	2	B	34	7E	1	1
4	092C	162C	1	1	3	3	45	0C	1	1

Tabla 5-1b Valores esperados de los registros de control

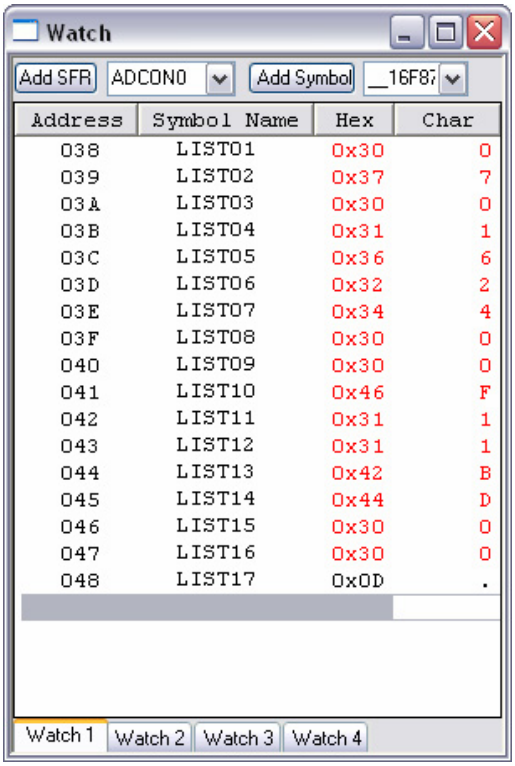


Figura 5-1a Prueba 1 Mensaje de control

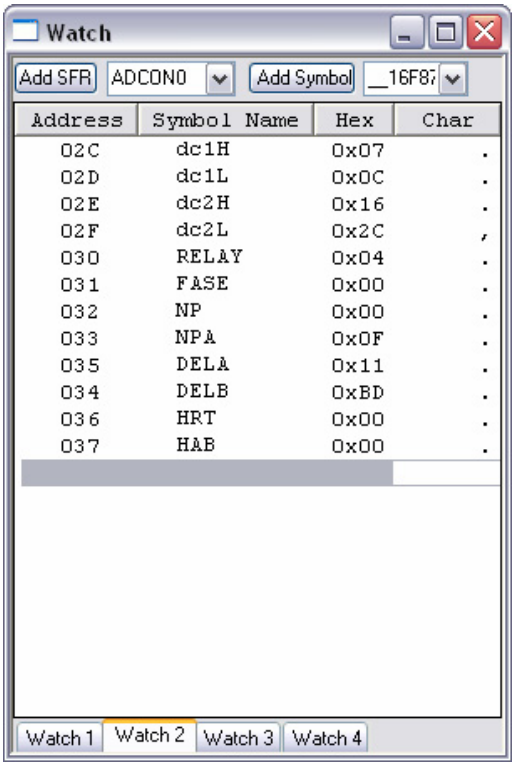


Figura 5-1b Prueba 1 Parámetros decodificados

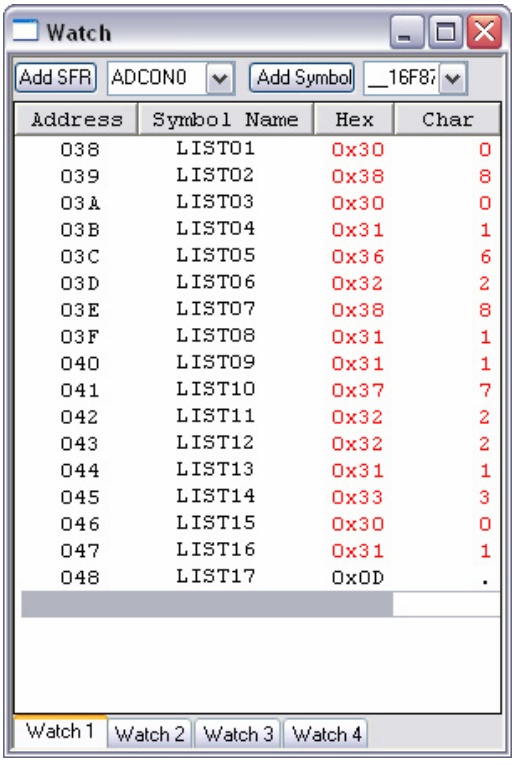


Figura 5-2a Prueba 2 Mensaje de control

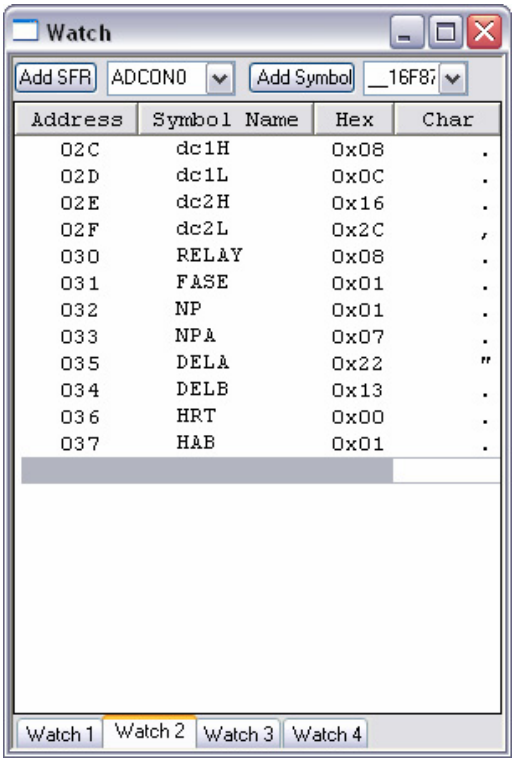


Figura 5-2b Prueba 2 Parámetros decodificados

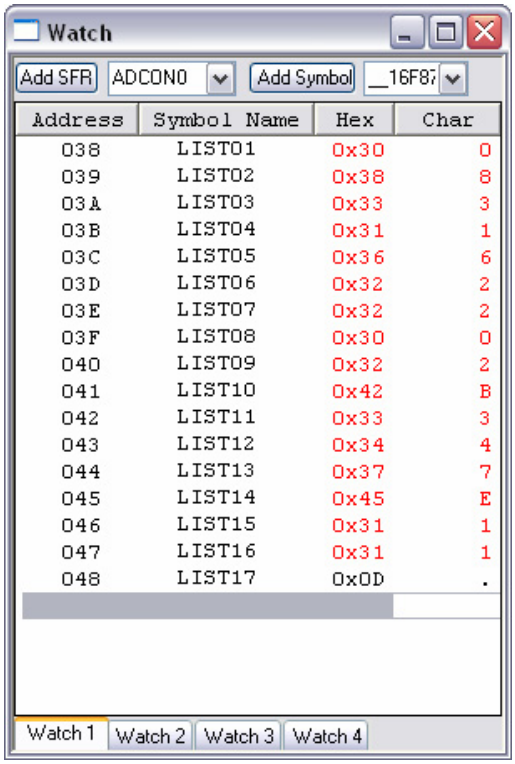


Figura 5-3a Prueba 3 Mensaje de control

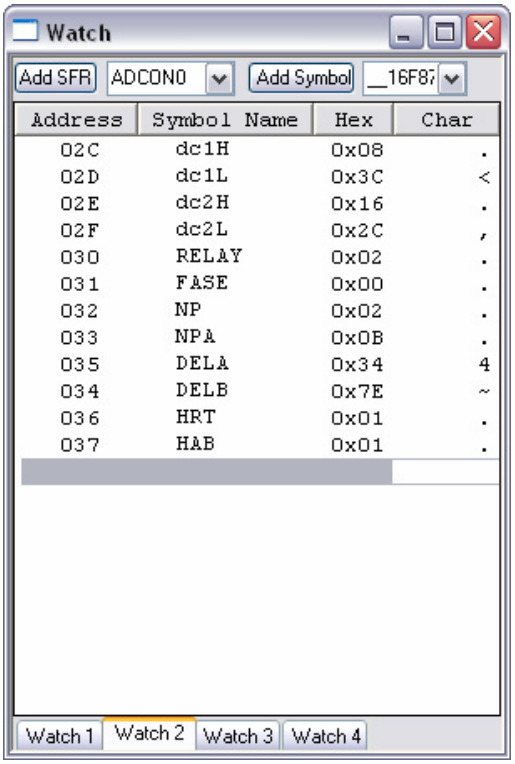


Figura 5-3b Prueba 3 Parámetros decodificados

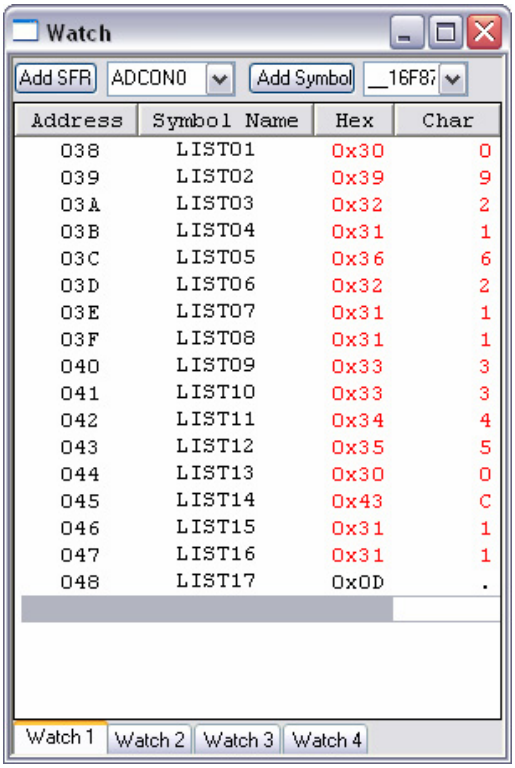


Figura 5-4a Prueba 4 Mensaje de control

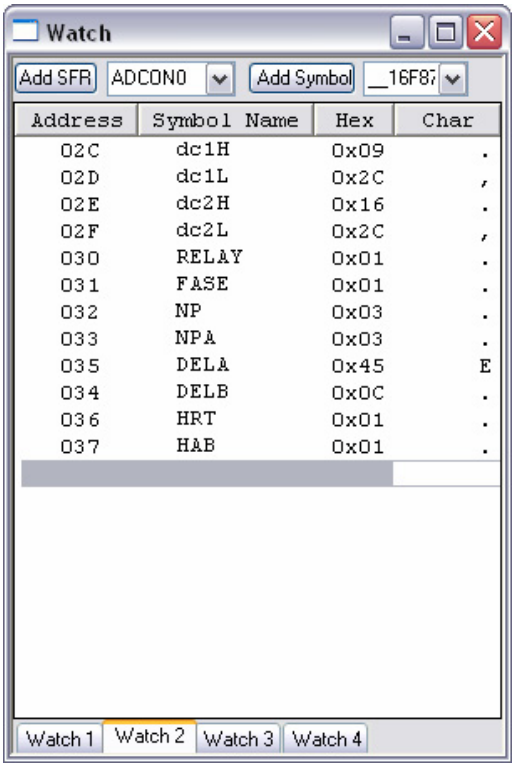


Figura 5-4b Prueba 4 Parámetros decodificados

Después de las pruebas de simulación encontramos que los parámetros decodificados presentan los valores esperados, por lo que puede concluirse que las tareas de interrupción del software de control funcionan correctamente.

Pruebas en hardware

En estas pruebas se evalúa que las salidas del PIC16F873 sean correctas mediante el uso de la siguiente tarjeta:

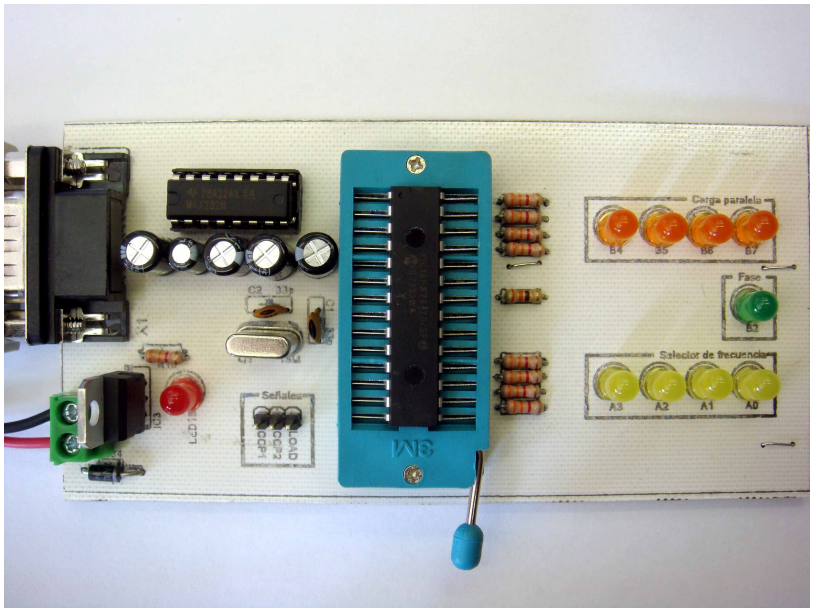


Figura 5-5 Tarjeta de evaluación

Prueba 1	δ_{CCP1} [%]	δ_{CCP2} [%]	LOAD [ms]	$(CP)_{16}$	$(FASE)_{16}$	$(SF)_{16}$
1	28	90	10	F	0	4
2	32	90	1	7	1	8
3	35	90	0.1	B	0	2
4	38	90	0.01	3	1	1

Tabla 5-2 Pruebas en hardware: salidas CCP1, CCP2, LOAD, Carga paralela, Fase y Selector de frecuencia

Prueba 1

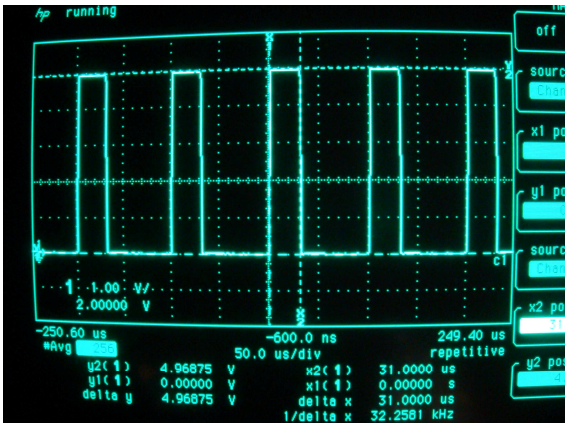


Figura 5-6a Prueba 1 salida CCP1

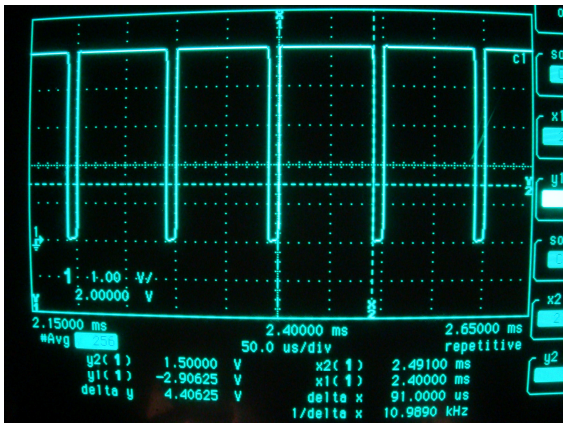


Figura 5-6b Prueba 1 salida CCP2

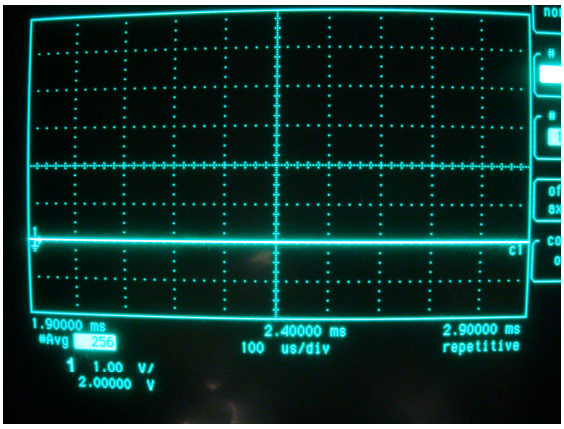


Figura 5-6c Prueba 1 salida LOAD

Figura 5-6d Prueba 1 salidas CP, FASE y SF

A [Vpp]	$\delta_{CCP1\text{-}TEO}$ [%]	$\delta_{CCP1\text{-}EXP}$ [%]	$\%E_{CCP1}$ [%]
50	28.2504	29	2.6536

Tabla 5-3a: $\%E_{CCP1}$

f [Hz]	$\delta_{CCP2\text{-}TEO}$ [%]	$\delta_{CCP2\text{-}EXP}$ [%]	$\%E_{CCP2}$ [%]
100	90.2163	91	0.8687

Tabla 5-3b: $\%E_{CCP2}$

$DELB_{TEO}$ [µs]	$DELB_{EXP}$ [µs]	$\%E_{DELB}$ [%]
∞	∞	0

Tabla 5-3c: $\%E_{DELB}$

Prueba 2

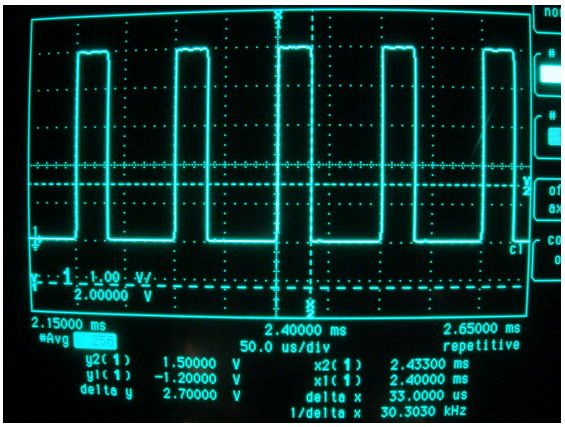


Figura 5-7a Prueba 2 salida CCP1

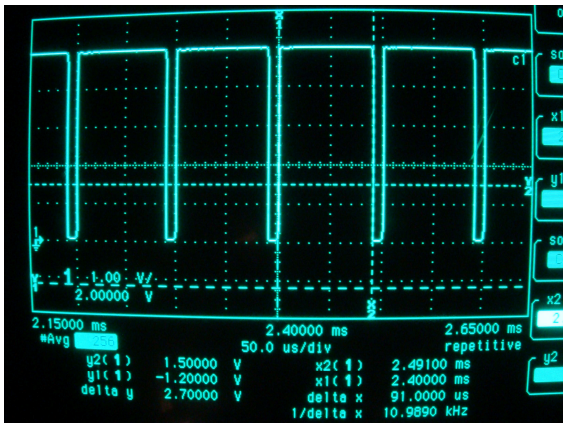


Figura 5-7b Prueba 2 salida CCP2

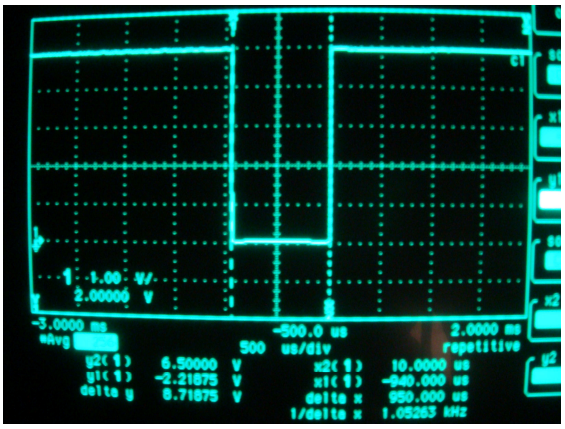


Figura 5-7c Prueba 2 salida LOAD

Figura 5-7d Prueba 2 salidas CP, FASE y SF

A [Vpp]	$\delta_{CCP1\text{-}TEO}$ [%]	$\delta_{CCP1\text{-}EXP}$ [%]	$\%E_{CCP1}$ [%]
100	31.5841	33	4.4831

Tabla 5-4a: $\%E_{CCP1}$

f [kHz]	$\delta_{CCP2\text{-}TEO}$ [%]	$\delta_{CCP2\text{-}EXP}$ [%]	$\%E_{CCP2}$ [%]
1	90.2163	91	0.8687

Tabla 5-4b: $\%E_{CCP2}$

$DELB_{TEO}$ [µs]	$DELB_{EXP}$ [µs]	$\%E_{DELB}$ [%]
1000	950	5

Tabla 5-4c: $\%E_{DELB}$

Prueba 3

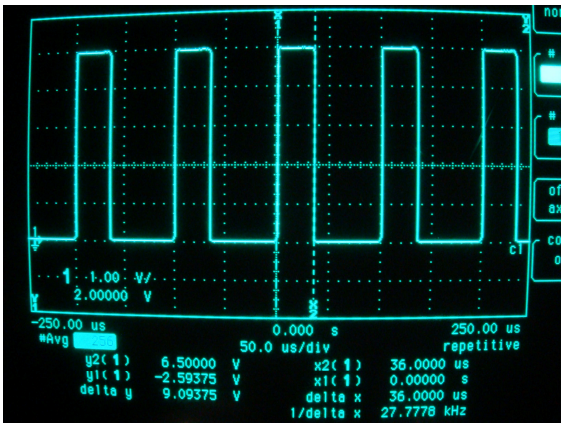


Figura 5-8a Prueba 3 salida CCP1

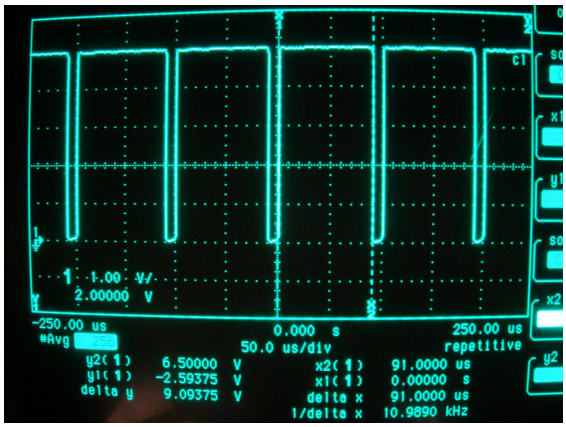


Figura 5-8b Prueba 3 salida CCP2

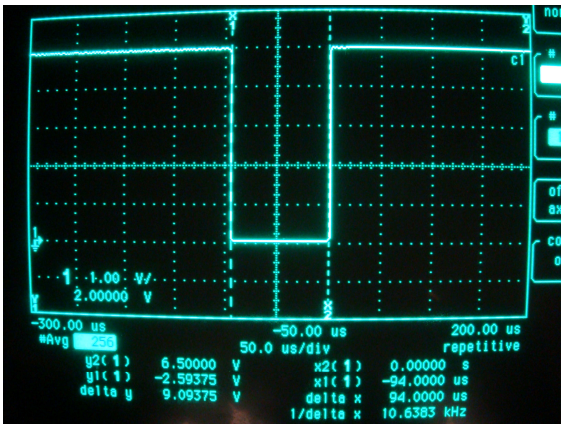


Figura 5-8c Prueba 3 salida LOAD

Figura 5-8d Prueba 3 salidas CP, FASE y SF

A [Vpp]	$\delta_{CCP1\text{-}TEO}$ [%]	$\delta_{CCP1\text{-}EXP}$ [%]	$\%E_{CCP1}$ [%]
150	34.9178	36	3.0994

Tabla 5-5a: $\%E_{CCP1}$

f [kHz]	$\delta_{CCP2\text{-}TEO}$ [%]	$\delta_{CCP2\text{-}EXP}$ [%]	$\%E_{CCP2}$ [%]
10	90.2163	91	0.8687

Tabla 5-5b: $\%E_{CCP2}$

$DELB_{TEO}$ [µs]	$DELB_{EXP}$ [µs]	$\%E_{DELB}$ [%]
100	94	6

Tabla 5-5c: $\%E_{DELB}$

Prueba 4

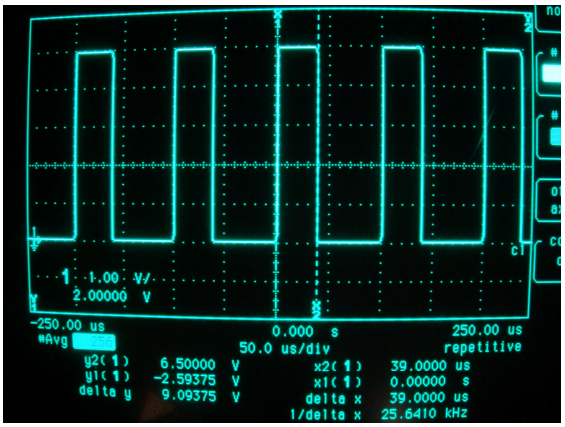


Figura 5-9a Prueba 4 salida CCP1

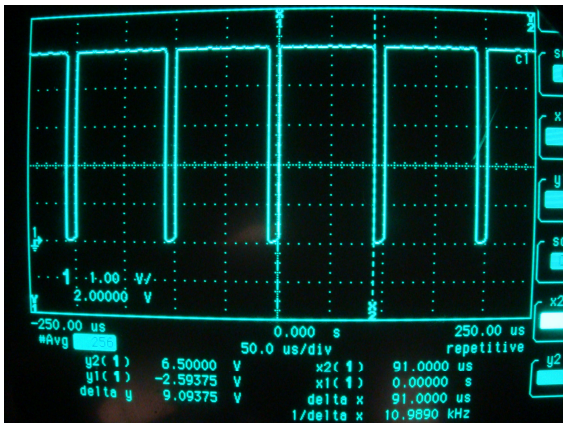


Figura 5-9b Prueba 4 salida CCP2

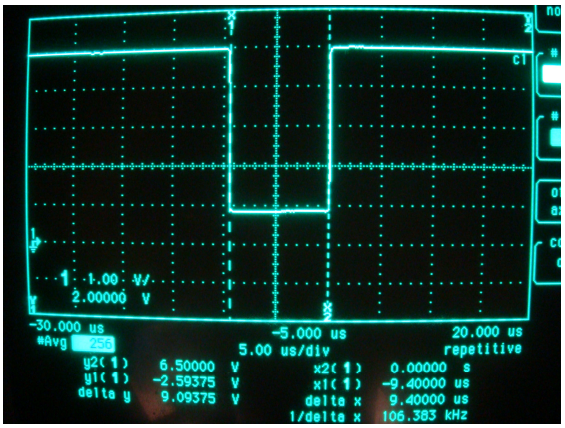


Figura 5-9c Prueba 4 salida LOAD

Figura 5-9d Prueba 4 salidas CP, FASE y SF

A [Vpp]	$\delta_{CCP1\text{-}TEO}$ [%]	$\delta_{CCP1\text{-}EXP}$ [%]	$\%E_{CCP1}$ [%]
200	38.2514	39	1.9569

Tabla 5-6a: $\%E_{CCP1}$

f [kHz]	$\delta_{CCP2\text{-}TEO}$ [%]	$\delta_{CCP2\text{-}EXP}$ [%]	$\%E_{CCP2}$ [%]
100	90.2163	91	0.8687

Tabla 5-6b: $\%E_{CCP2}$

$DELB_{TEO}$ [µs]	$DELB_{EXP}$ [µs]	$\%E_{DELB}$ [%]
10	9.4	6

Tabla 5-6c: $\%E_{DELB}$

Resultados

A continuación se presentan los resultados que se obtuvieron en el prototipo implementado.

Figura 6-1 Prototipo del GFAT

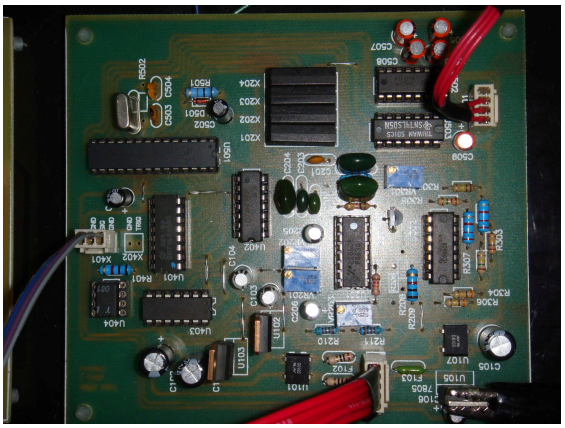


Figura 6-2 Prototipo del generador

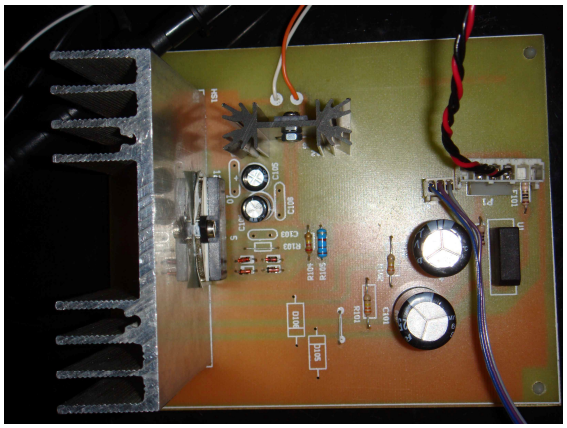


Figura 6-3 Prototipo del amplificador

Evaluando la respuesta de los filtros $R_{302}C_{301}$ y $R_{312}C_{302}$ se encontraron variaciones de alrededor del 10% con respecto a los valores teóricos.

δ_{CCP1} [%]	$V_{C301\ TEO}$ [V]	$V_{C301\ EXP}$ [V]
0	14.9980	14.940
10	13.4920	13.430
20	11.9870	11.920
30	10.4830	10.430
40	8.9809	8.940
50	7.4803	7.450
60	5.9813	5.980
70	4.4837	4.500
80	2.9877	3.046
90	1.4931	1.587
100	0.0000	0.136

Tabla 6-1Respuesta experimental del filtro $R_{302}C_{301}$

δ_{CCP2} [%]	$I_{T\ TEO}$ [mA]	$I_{T\ EXP}$ [mA]
0	0.3290	0.338
10	0.2955	0.305
20	0.2622	0.272
30	0.2290	0.238
40	0.1959	0.206
50	0.1629	0.173
60	0.1301	0.139
70	0.0974	0.106
80	0.0648	0.073
90	0.0323	0.040
100	0.0000	0.007

Tabla 6-2Respuesta experimental del filtro $R_{312}C_{302}$

Recalculando las ecuaciones (2.09) y (2.17) para los valores experimentales se encontró que:

$$\delta_{CCP1}[\,\%\,] = -6.755\left[\frac{\%}{\mathrm{V}}\right]\cdot V_{C_{301}}[\,\mathrm{V}\,] + 100.5764[\,\%\,] \qquad (6.01)$$

$$\delta_{CCP2}[\,\%\,] = -309.16\left[\frac{\%}{\mathrm{mA}}\right]\cdot I_T[\,\mathrm{mA}\,] + 102.2481[\,\%\,] \qquad (6.02)$$

A continuación se sustituyeron las ecuaciones (6.01) y (6.02) en la interfaz de comunicación y se evaluaron las características de las señales δ_{CCP1}, δ_{CCP2} y *LOAD*. Los resultados se muestran en las *tablas 6-3*, *6-4* y *6-5*. Por último, las *figuras 6-4, 6-5 y 6-6* muestran diferentes señales de excitación producidas con el *GFAT*.

A [Vpp]	$\delta_{CCP1\ TEO}$ [%]	$\delta_{CCP1\ EXP}$ [%]	$\%E_{\delta CCP1}$ [%]
50	27.9595	28	0.1450
75	29.6482	30	1.1865
100	31.3370	31	1.0754
125	33.0257	33	0.0780
150	34.7145	35	0.8224
175	36.4033	36	1.1078
200	38.0920	38	0.2416

Tabla 6-3 Pruebas experimentales de amplitud

f [Hz]	$\delta_{CCP2\ TEO}$ [%]	$\delta_{CCP2\ EXP}$ [%]	$\%E_{\delta CCP2}$ [%]
1000	92.5868	93	0.4463
2000	82.9256	83	0.0898
3000	73.2643	73	0.3607
4000	63.6030	63	0.9481
5000	53.9418	54	0.1079
6000	44.2805	43	2.8919
7000	34.6193	35	1.0997
8000	24.9580	25	0.1682
9000	15.2968	15	1.9401
9900	6.6017	7	6.0341

Tabla 6-4 Pruebas experimentales de frecuencia

f [Hz]	$DELB_{TEO}$ [µs]	$DELB_{EXP}$ [µs]	$\%E_{DELB}$ [%]
100	10000	9400	6
1000	1000	960	4
10000	100	95	5
100000	10	9.5	5
800000	1.25	1.26	0.8
1000000	1	1.26	26

Tabla 6-5 Pruebas experimentales del tiempo en bajo de la señal LOAD

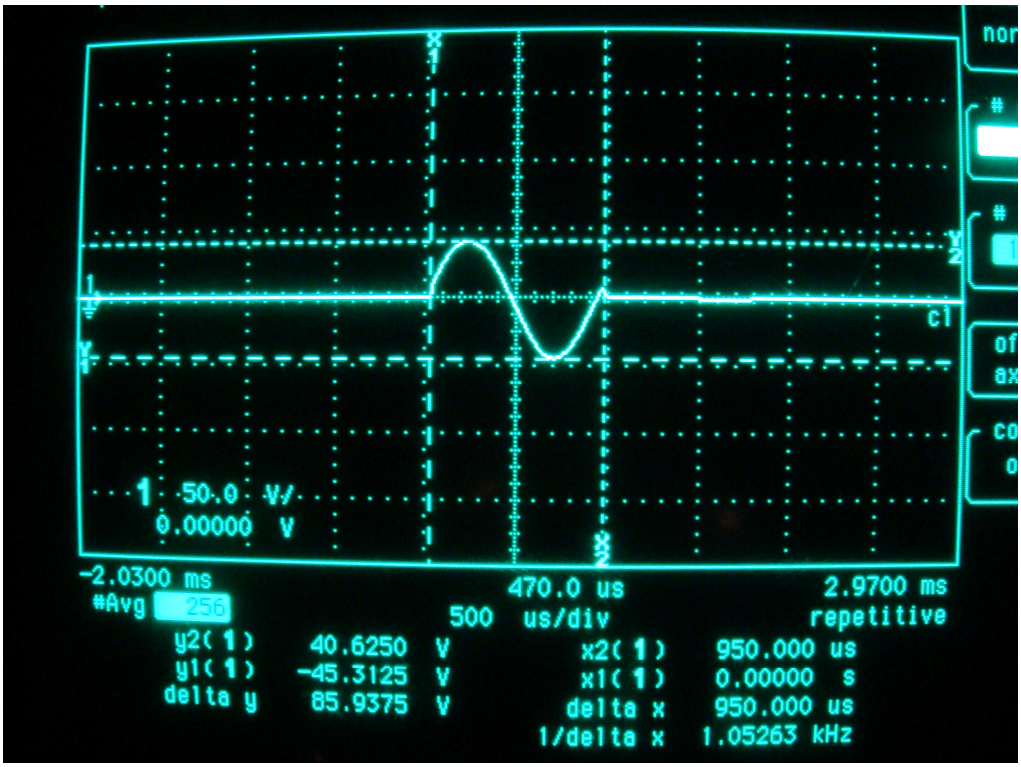


Figura 6-4

Parámetro	TEO	EXP	%E [%]
A [Vpp]	100	85.9375	14.0625
f [Hz]	1000	1052.63	5.263

Tabla 6-6

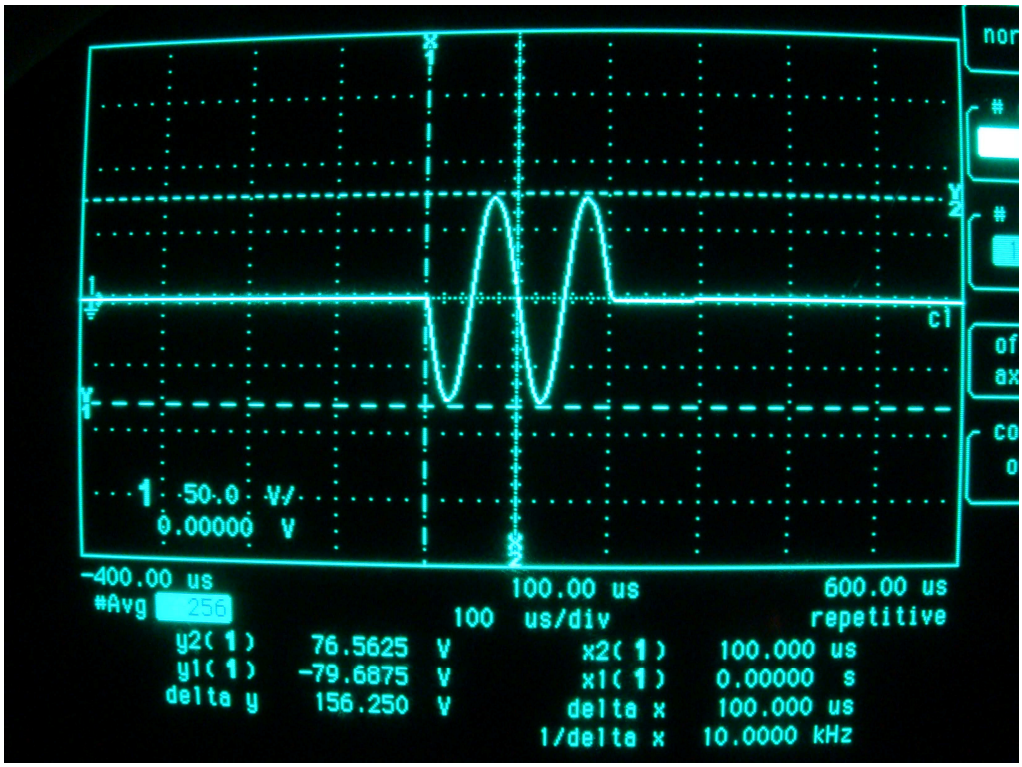


Figura 6-5

Parámetro	TEO	EXP	%E [%]
A [Vpp]	150	156.25	4.1667
f [Hz]	10000	10000	0.0000

Tabla 6-7

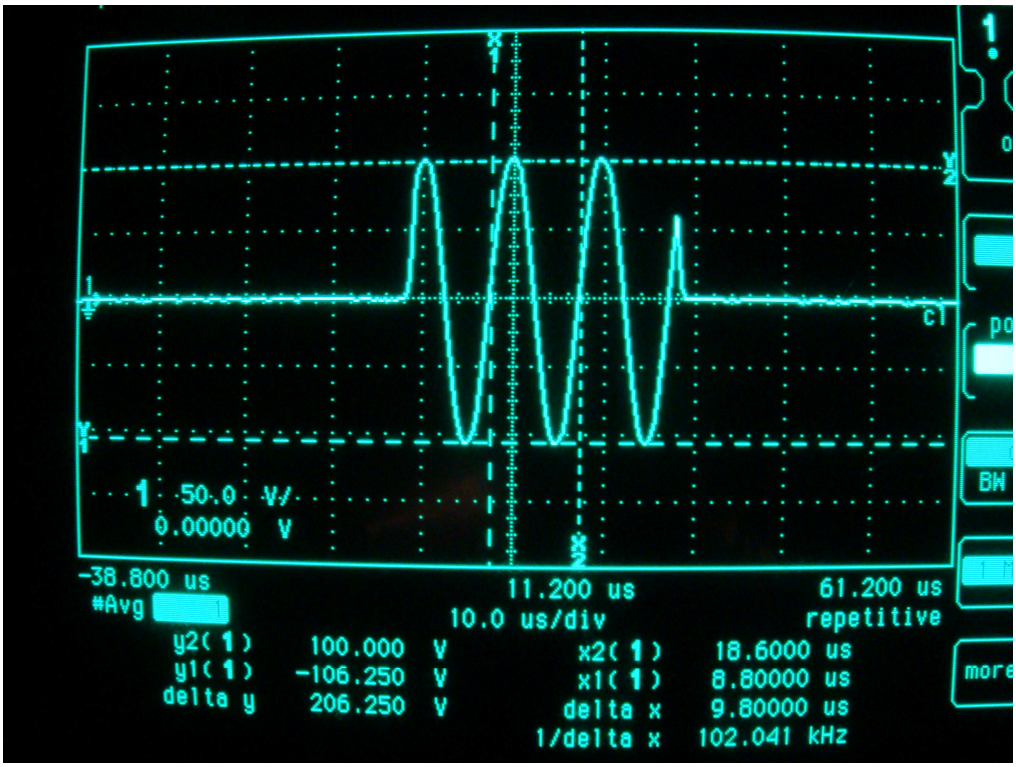


Figura 6-6

Parámetro	TEO	EXP	%E [%]
A [Vpp]	200	206.25	3.1250
f [Hz]	100000	102041	2.0410

Tabla 6-8

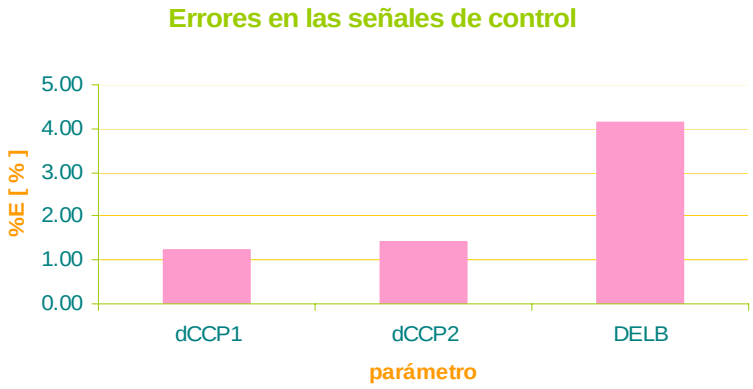
Conclusiones

En el presente trabajo se describe cómo se desarrolló el software de control para un generador de funciones de alta tensión. Este software, implementado en el PIC16F873, permitió modificar los parámetros de amplitud, frecuencia, fase y número de ciclos de la señal proporcionada por el generador.

Para ello, la interfaz de comunicación desarrollada en *LabVIEW*, generó un conjunto de caracteres ASCII a partir de los parámetros deseados y los transmitió vía RS232 al PIC16F873. Una vez que la recepción fue completada, el software de control validó, decodificó y actualizó las señales de control del generador de funciones garantizando así que la señal presentara los nuevos parámetros.

El software de control, *GFAT.asm*, se desarrolló en lenguaje ensamblador con la aplicación *MPLAB IDE v7.50* y se cargó en el microcontrolador con el programador *PICSTART Plus*. Para probar su funcionamiento, se realizaron pruebas de simulación y pruebas en hardware; encontrándose las siguientes observaciones:

- Los errores promedios de las señales de control δ_{CCP1}, δ_{CCP2} y DELB son del 1.22%, 1.41% y 4.16%, respectivamente.



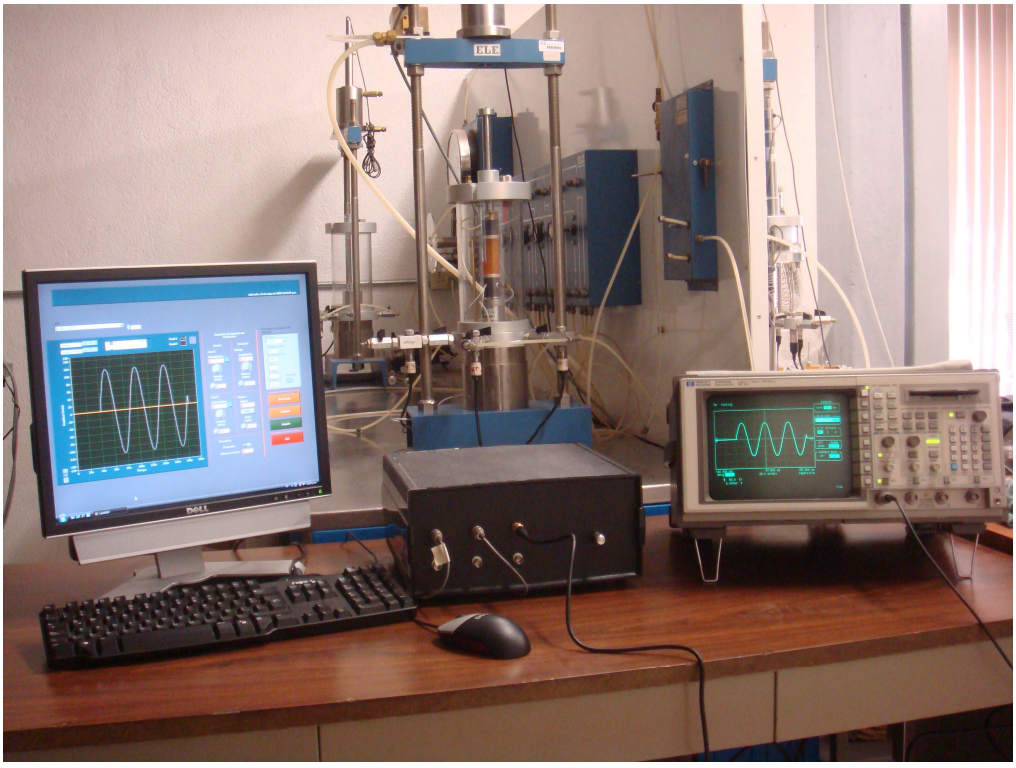
- Por otra parte, los selectores de frecuencia, fase y carga paralela no presentaron error alguno.

- El tiempo mínimo en que la señal *LOAD* permanece en bajo es 1.26 [µs]. Esto corresponde al tiempo requerido para que el microcontrolador ejecute una instrucción.

El software de control desarrollado constituye una aplicación específica que permite calcular el módulo de rigidez de suelos, y complementar el diseño de cimentación de estructuras.

A pesar de haber sido desarrollado para un caso particular, su diseño modular permite que el código generado se lleve a otras aplicaciones tales como el control dinámico de displays gráficos y monitoreo y control de temperatura. Así mismo, se puede utilizar el Generador de Funciones de Alta Tensión en pruebas no destructivas para identificar cambios de densidad en una estructura a través de los patrones de propagación de ondas.

Con el fin de mejorar el desempeño del generador de funciones, se puede seleccionar un microcontrolador que presente una mayor velocidad en la ejecución de instrucciones y que cuente con un puerto USB para facilitar la conexión con la computadora.



Apéndice A: Diagramas de Flujo

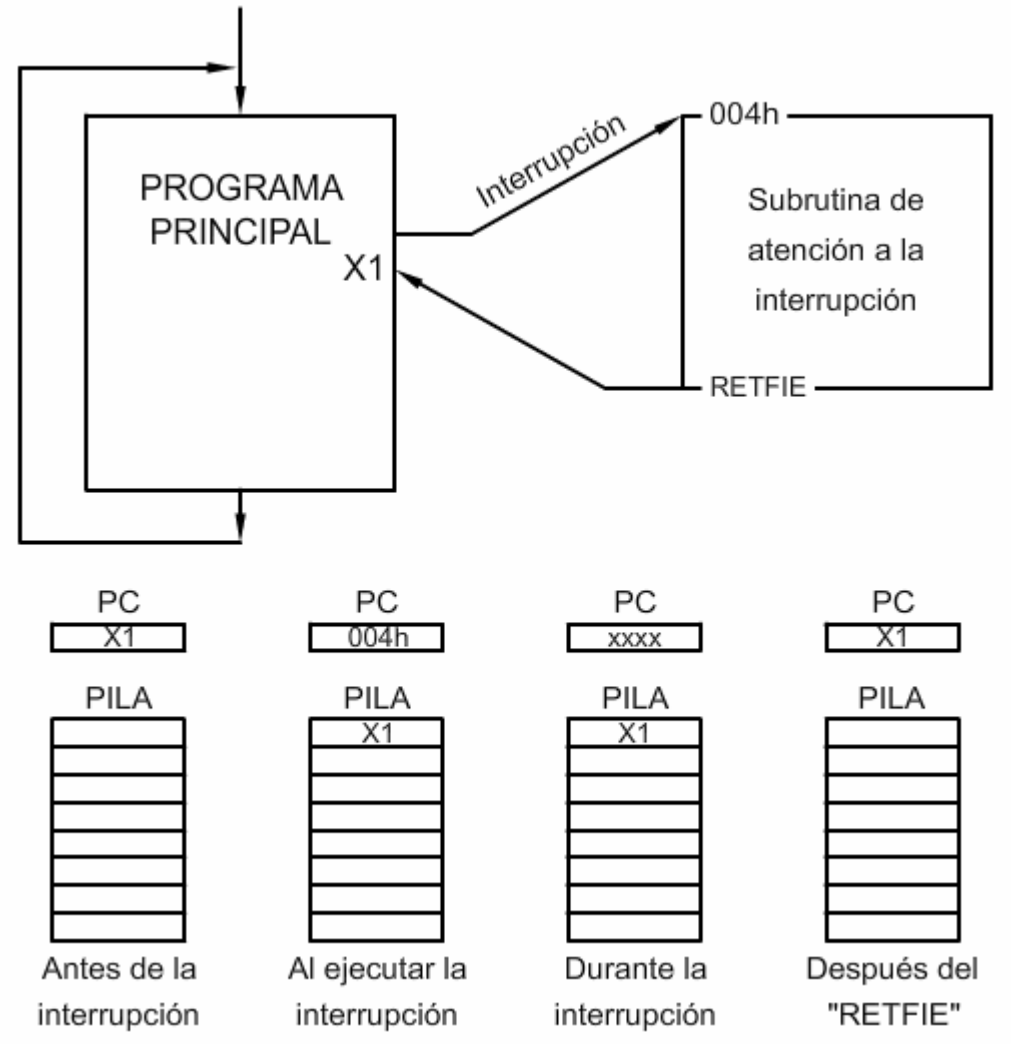


Figura A-1 Diagrama general del software de control

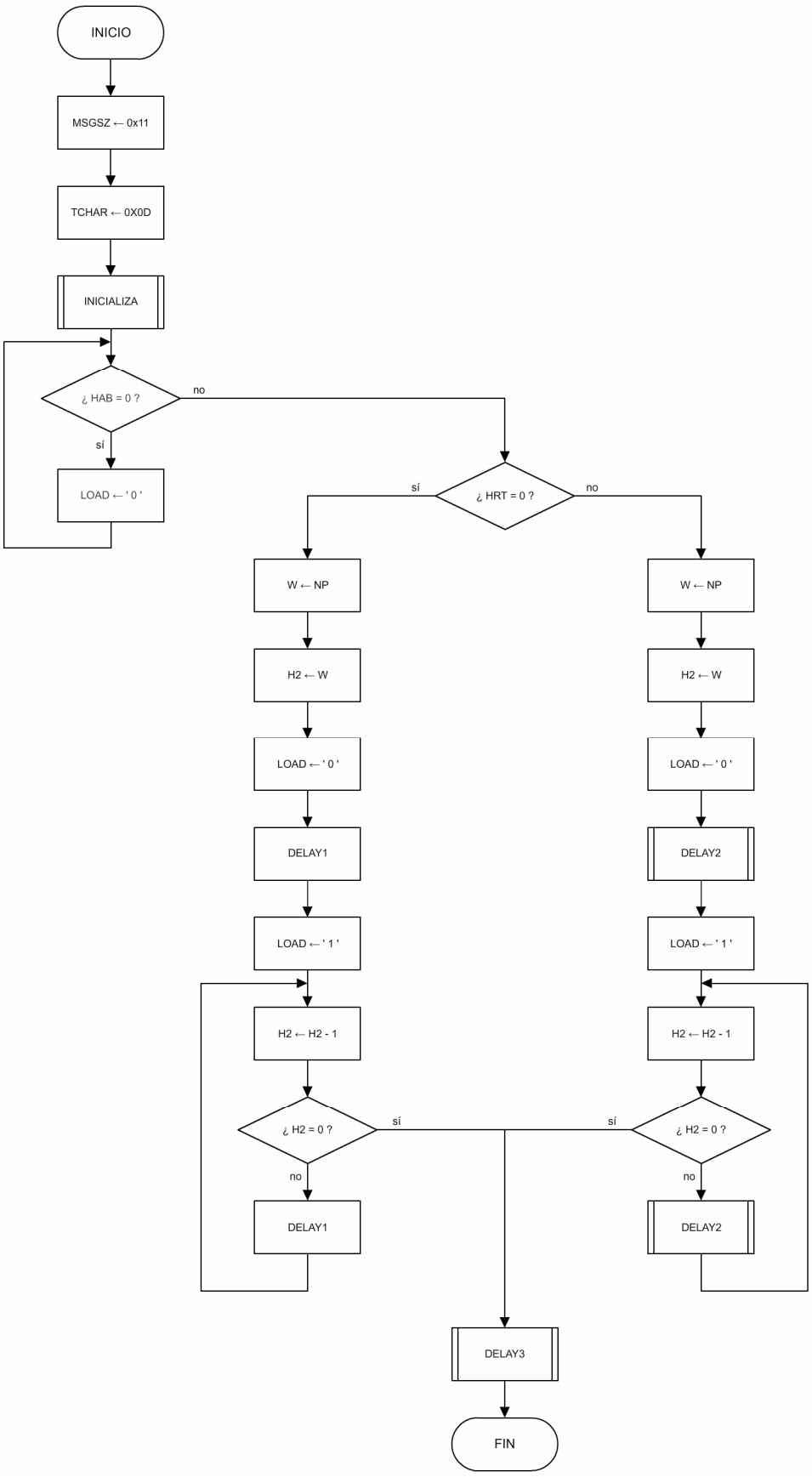


Figura A-2 Programa Principal

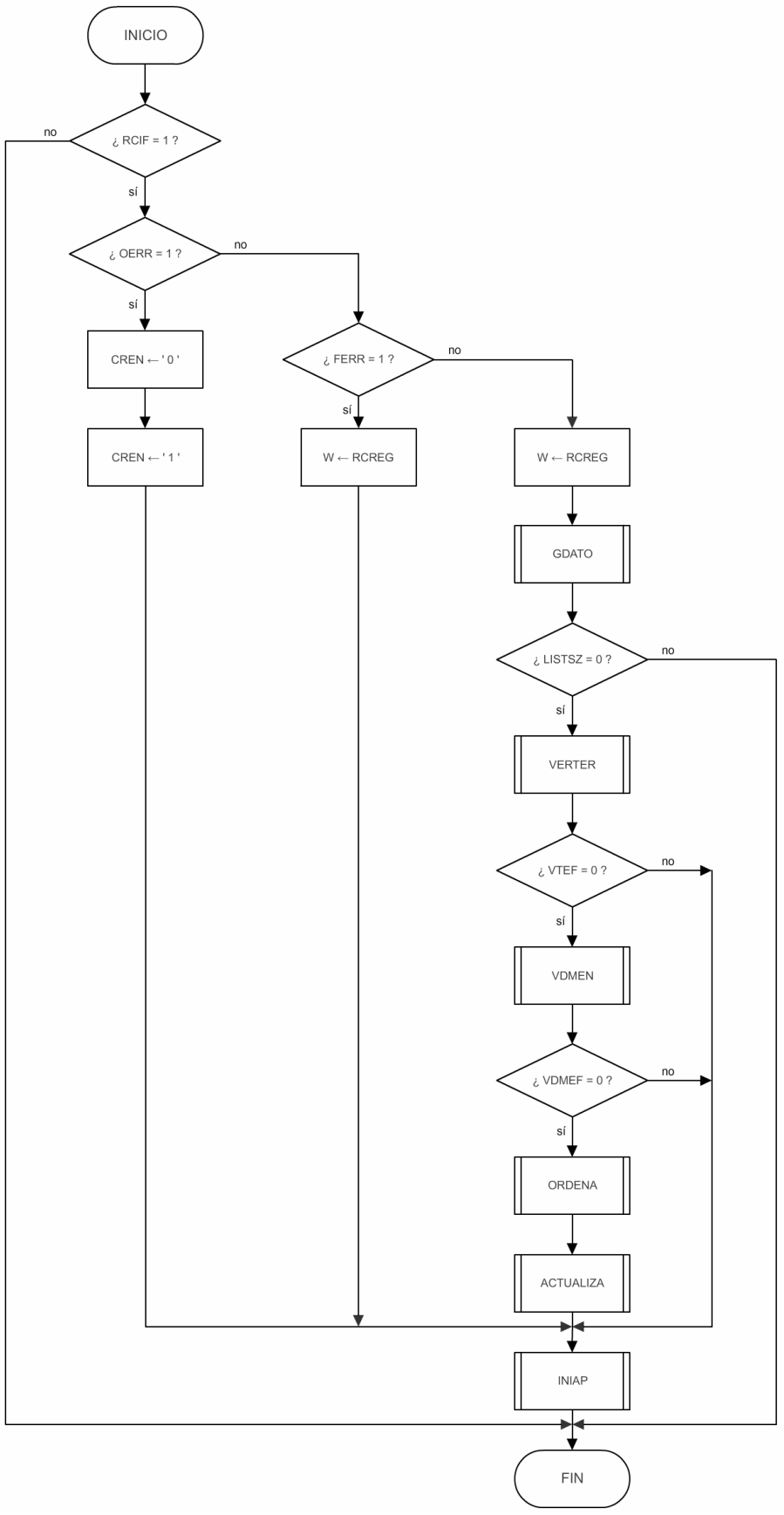


Figura A-3 Interrupción

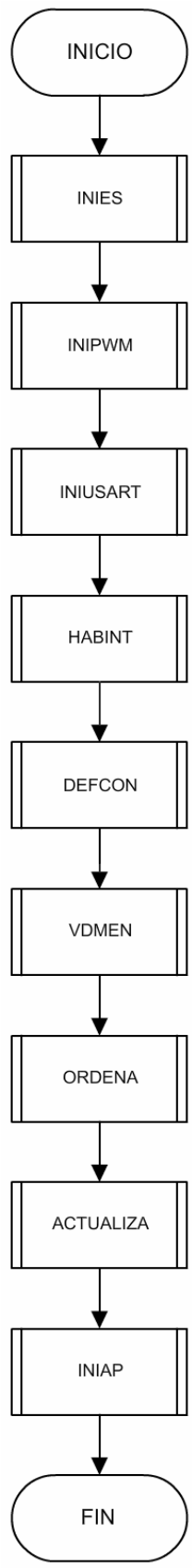


Figura A-4 Subrutina INICIALIZA

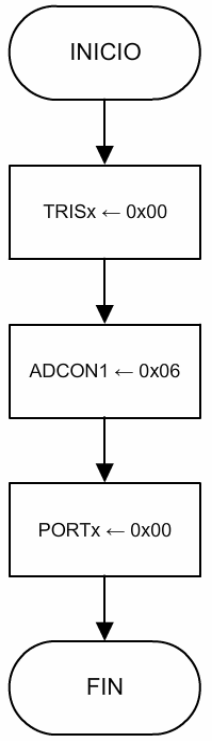


Figura A-5 Subrutina INIES

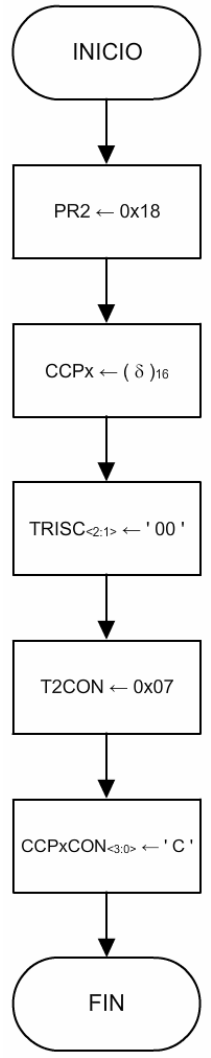


Figura A-6 Subrutina INIPWM

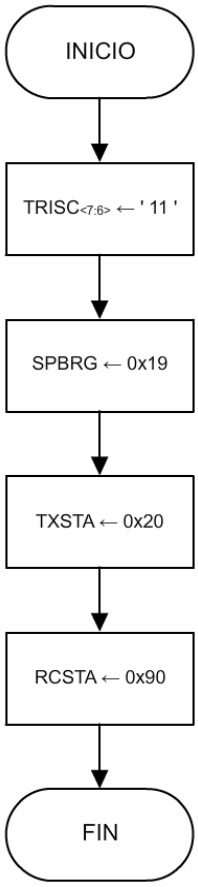


Figura A-7 Subrutina INIUSART

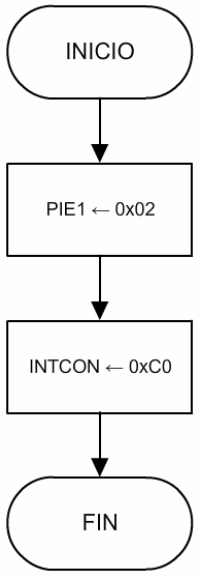


Figura A-8 Subrutina HABINT

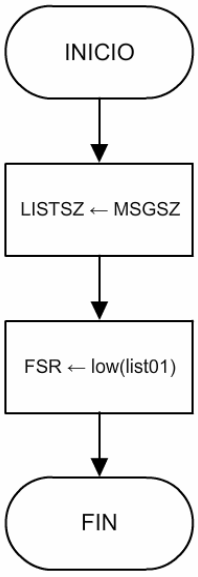


Figura A-9 Subrutina INIAP

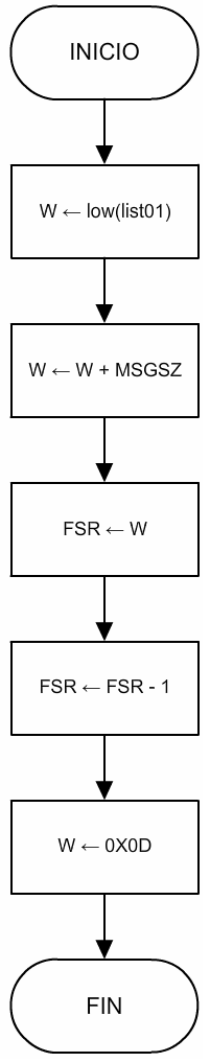


Figura A-10 Subrutina CARGAT

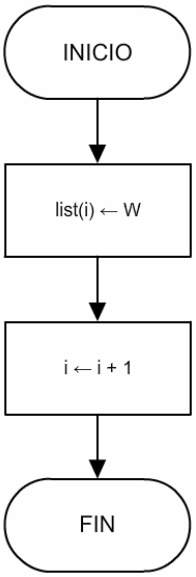


Figura A-11 Subrutina GDATO

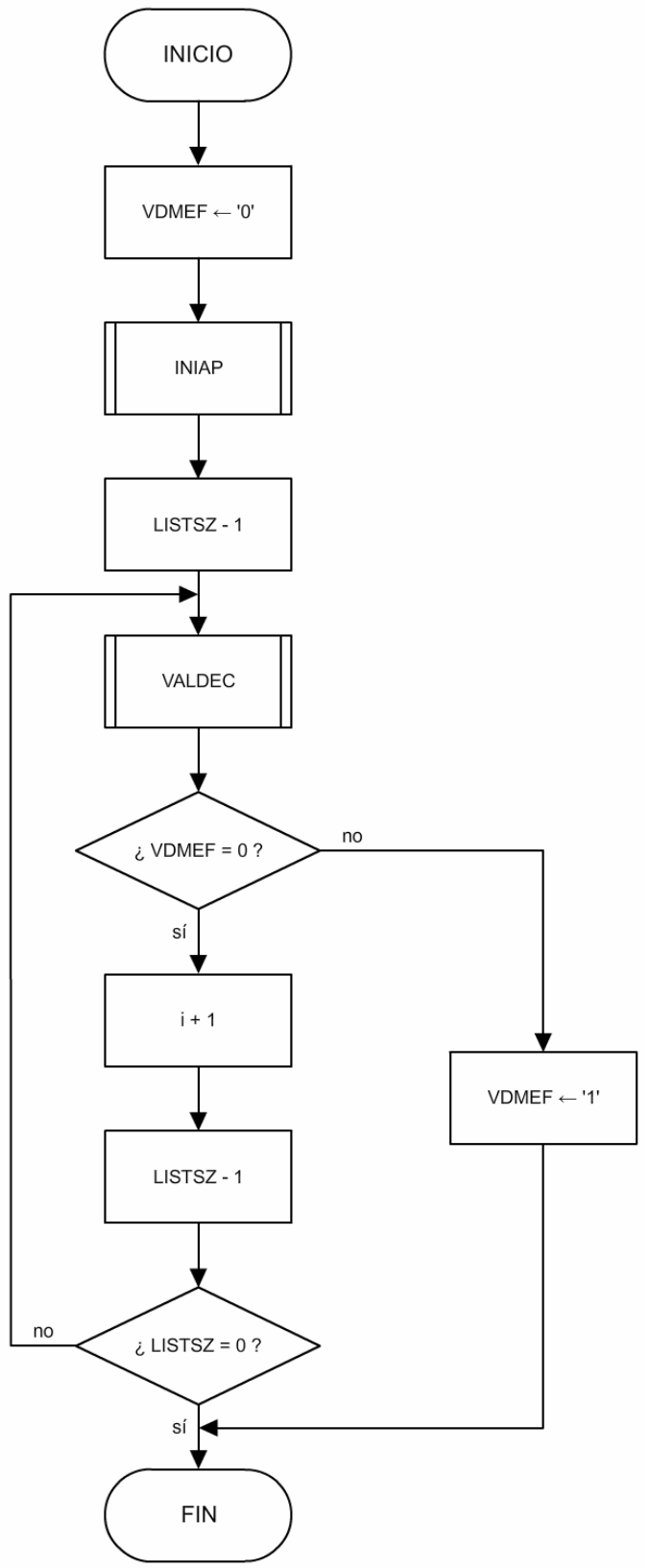


Figura A-12 Subrutina VDMEN

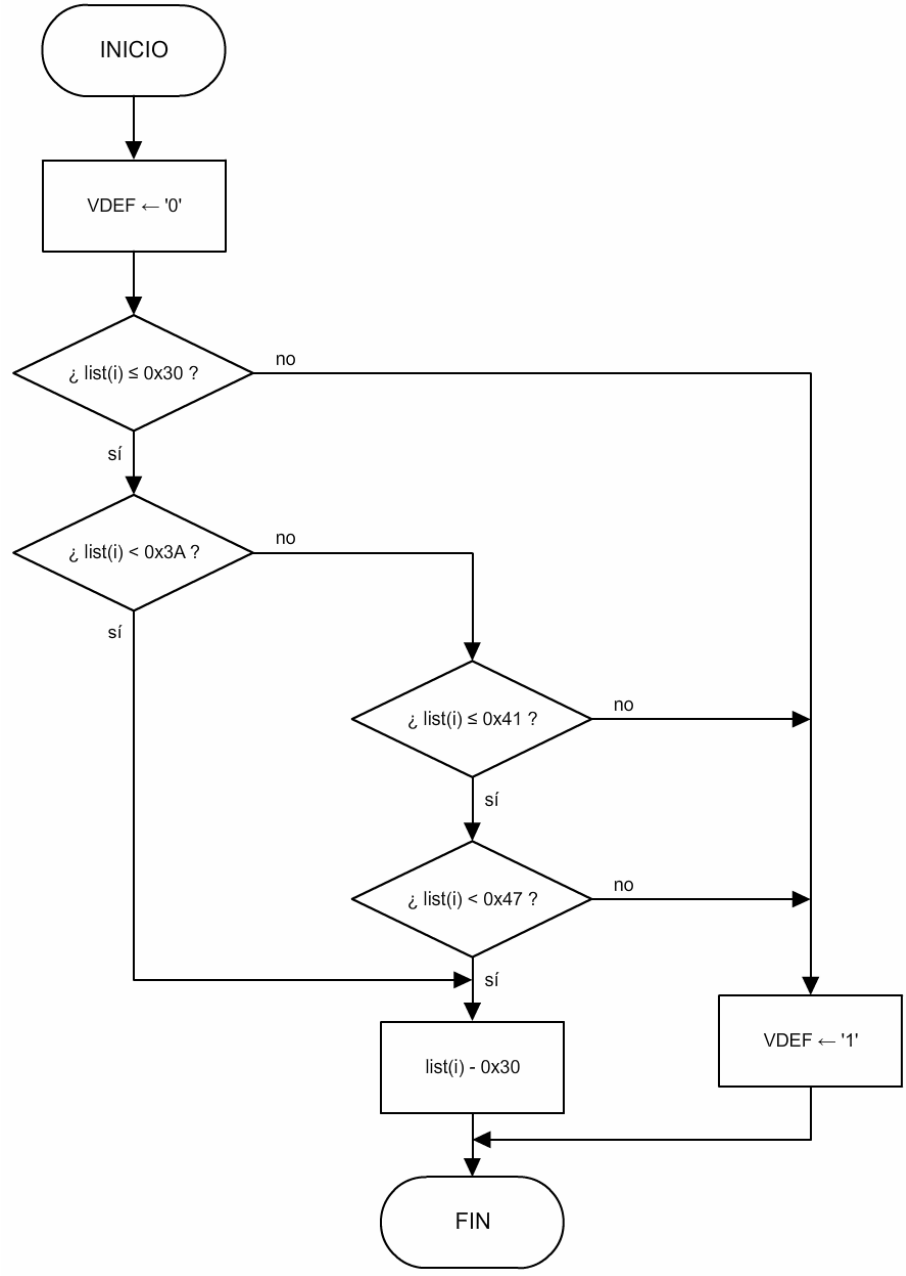


Figura A-13 Subrutina VALDEC

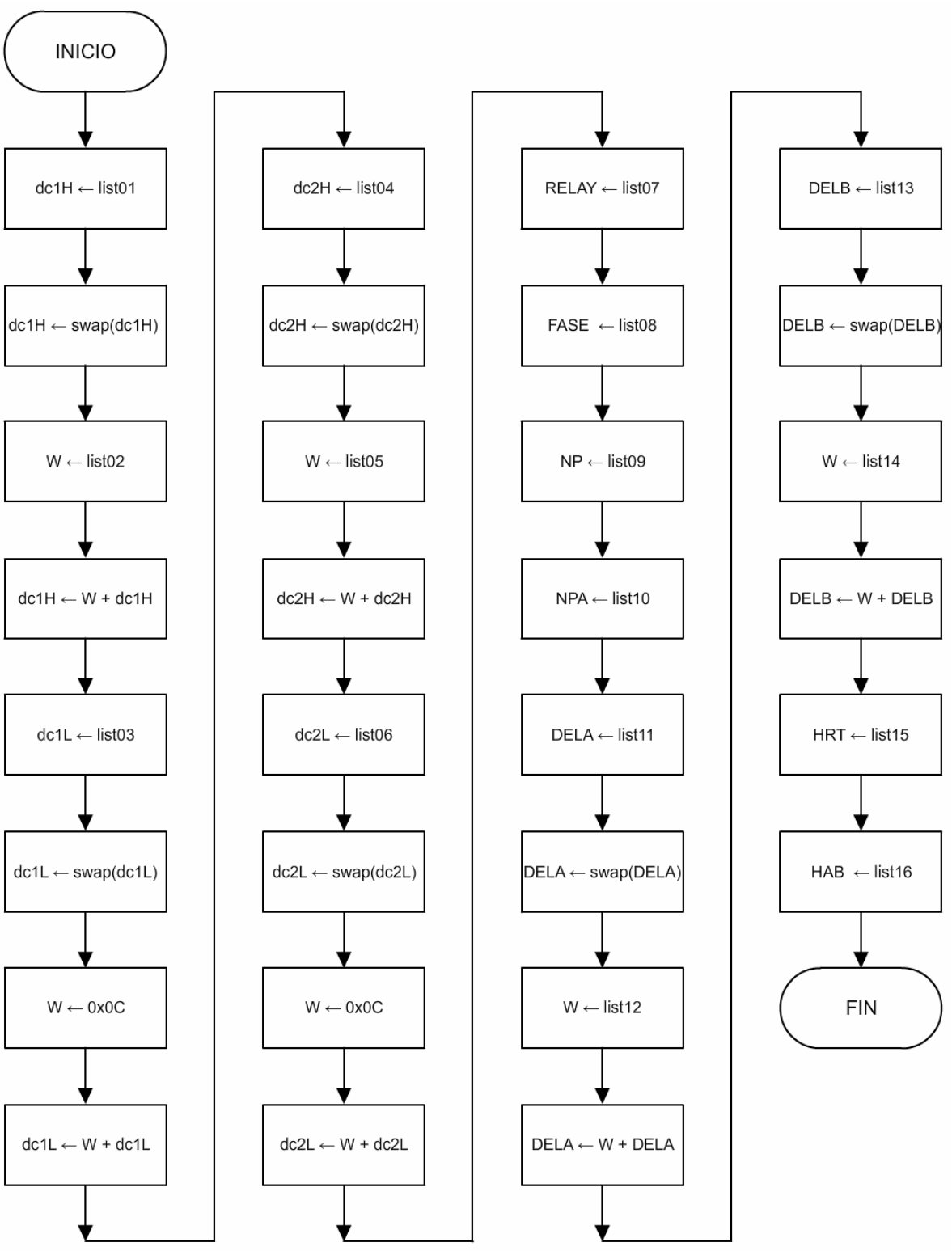


Figura A-14 Subrutina ORDENA

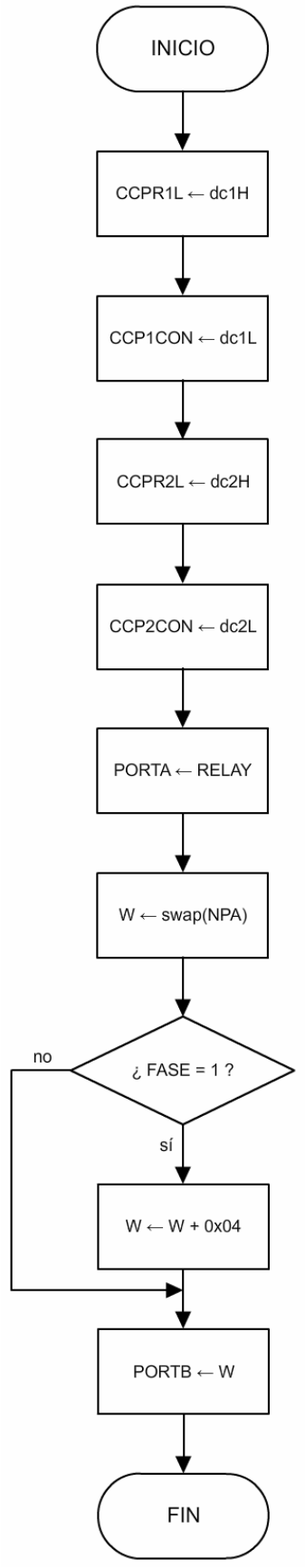


Figura A-15 Subrutina ACTUALIZA

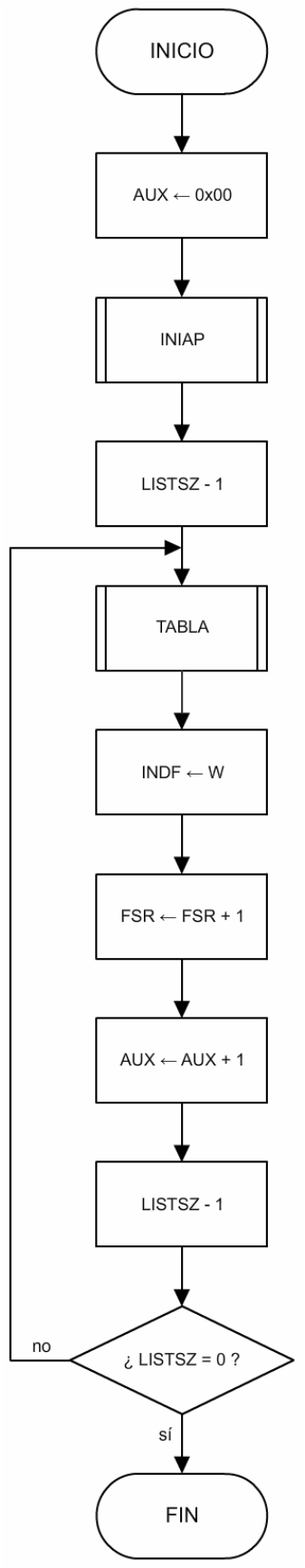


Figura A-16 Subrutina DEFCON

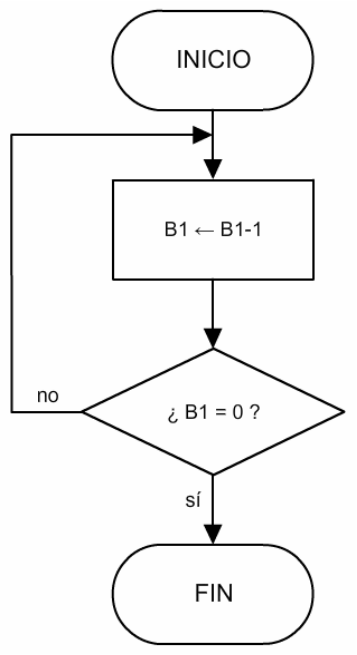


Figura A-17 Subrutina DELAY1

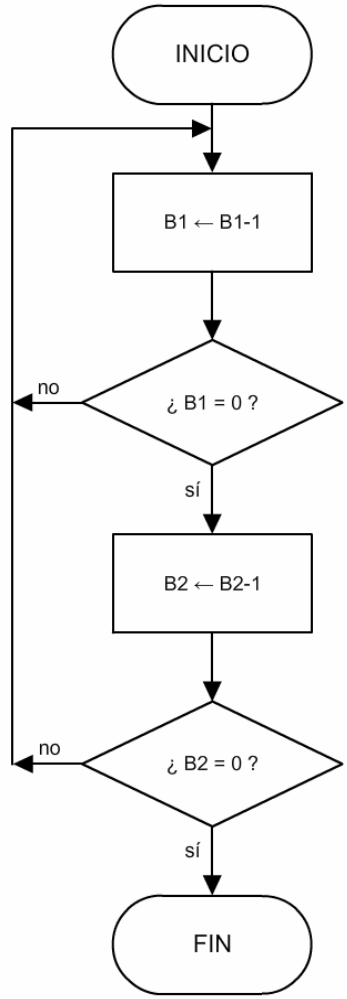


Figura A-18 Subrutina DELAY2

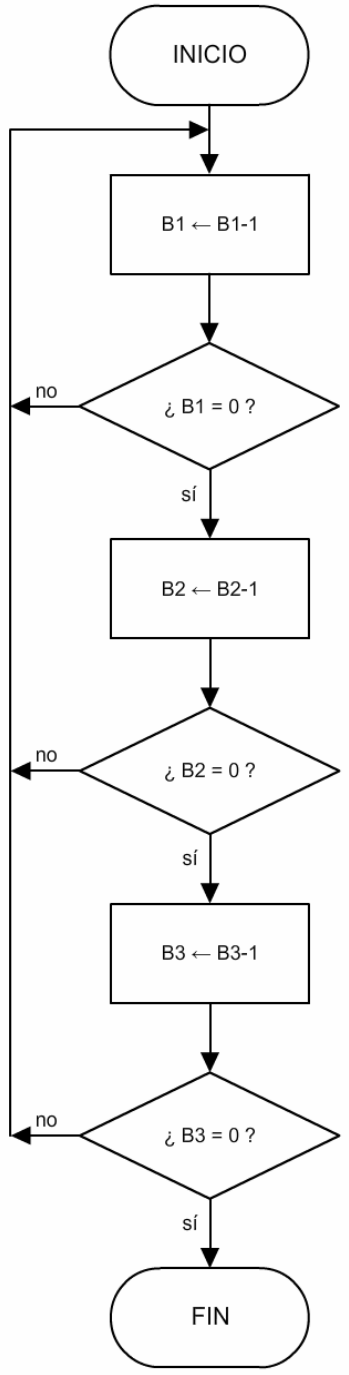


Figura A-19 Subrutina DELAY3

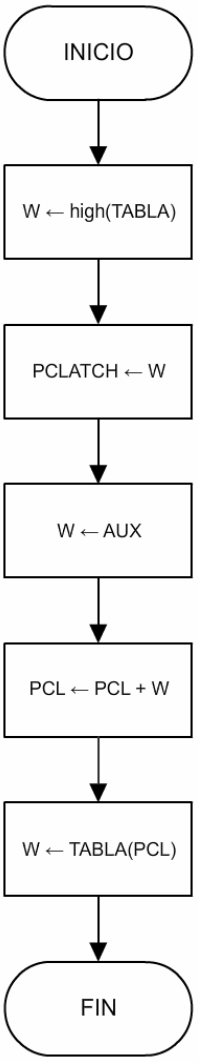


Figura A-20 Subrutina TABLA

Apéndice B: Código Fuente

```
;*******************************************************************************
;                                                                              *
;   Microprospección de ondas dinámicas en suelos                              *
;   saturados a bajos niveles de deformación                                   *
;                                                                              *
;   SOFTWARE DE CONTROL                                                        *
;                                                                              *
;*******************************************************************************
;
;   Nombre del archivo:     GFAT.asm                                           *
;   Fecha:                  28-04-08                                           *
;   Versión:                1                                                  *
;   Autor:                  Gabriel Andrés Parra Rodríguez                     *
;   Compañía:               Instituto de Ingeniería, UNAM                      *
;
;*******************************************************************************
;                                                                              *
;   Archivos necesarios     P16F873.INC                                        *
;                                                                              *
;*******************************************************************************


                LIST            P=16F873        ; directiva para definir procesador
                #INCLUDE        <P16F873.INC>   ; definiciones específicas del procesador
                ERRORLEVEL      -302            ; suprime mensaje: "not in bank 0"

;*******************************************************************************
; bits de configuración                                                        *
;*******************************************************************************
                __CONFIG _PWRTE_OFF & _HS_OSC & _WDT_OFF & _LVP_OFF & _BODEN_OFF

;*******************************************************************************
; definición de variables                                                      *
;*******************************************************************************
w_temp          EQU             0x20            ; variable para el respaldo de entorno
status_temp     EQU             0x21            ; variable para el respaldo de entorno

;*******************************************************************************
; declaración de registros auxiliares                                          *
;*******************************************************************************
                CBLOCK          0x02B
                                PWMp            ; Periodo del PWM
                                dc1H             ;Parte alta ciclo de trabajo PWM1
                                dc1L            ; Parte baja ciclo de trabajo PWM1
                                dc2H            ; Parte alta ciclo de trabajo PWM2
                                dc2L            ; Parte baja ciclo de trabajo PWM2
                                RELAY           ;Relay selector de frecuencia
                                FASE            ;Defasamiento
                                NP              ;# de pulsos
```

```
                        NPA             ;# de pulsos ajustado
                        DELB            ;Tiempo en bajo (load)
                        DELA            ;Tiempo en alto (load)
                        HRT             ;Criterio para la selección del retardo
                        HAB             ;ON/OFF
                        LIST01          ;list(01)
                        LIST02          ;list(02)
                        LIST03          ;list(03)
                        LIST04          ;list(04)
                        LIST05          ;list(05)
                        LIST06          ;list(06)
                        LIST07          ;list(07)
                        LIST08          ;list(08)
                        LIST09          ;list(09)
                        LIST10          ;list(10)
                        LIST11          ;list(11)
                        LIST12          ;list(12)
                        LIST13          ;list(13)
                        LIST14          ;list(14)
                        LIST15          ;list(15)
                        LIST16          ;list(16)
                        LIST17          ;list(17) localidad del terminador
                        LISTSZ          ;tamaño de la lista
                        MSGSZ           ;tamaño del mensaje
                        TCHAR           ;terminador
                        H1              ;auxiliar 1
                        H2              ;auxiliar 2
                        AUX             ;auxiliar 3
                        B1              ;delay helper1
                        B2              ;delay helper2
                        B3              ;delay helper3
                        BH1             ;delay helper1b
                        BH2             ;delay helper2b
                        BH3             ;delay helper3b
                        FLAG            ;banderas
                ENDC

#DEFINE         VTEF            FLAG,0          ; bandera de error, verifica terminador
#DEFINE         VDEF            FLAG,1          ; bandera de error, valida y decodofica dato
#DEFINE         VDMEF           FLAG,2          ; bandera de error, valida y decodofica mensaje
#DEFINE         LOAD            PORTB,3         ; habilitación del contador (LOAD)

;*******************************************************************************
; vector de reset                                                              *
;*******************************************************************************
                ORG             0x000           ; localidad del vector de reset
                CLRF            PCLATH          ; página 0
                GOTO            MAIN            ; apunta al inicio del programa

;*******************************************************************************
; vector de interrupción                                                       *
;*******************************************************************************
                ORG             0x004           ; localidad del vector de interrupción
                MOVWF           w_temp          ; w_temp ← W
                MOVF            STATUS,W        ; W ← STATUS
                BCF             STATUS,RP0      ; banco 0
```

```
                MOVWF       status_temp     ; status_temp ← W

                BANKSEL     PIR1
                BTFSS       PIR1,RCIF       ;¿interrupción por recepción?
                GOTO        FINRC           ;no, vuelve al código principal
                BANKSEL     PORTA           ;sí, verifica errores de recepción
                BTFSC       RCSTA,OERR      ;¿OVERRUN?
                GOTO        EOERR           ;sí, arregla
                BTFSC       RCSTA,FERR      ;¿FRAMING?
                GOTO        EFERR           ;sí, arregla
                MOVF        RCREG,W         ;W ← RCREG
                CALL        GDATO           ;guarda dato
                DECFSZ      LISTSZ,F        ;¿lista llena?
                GOTO        FINRC           ;no, termina interrupción
                CALL        VERTER          ;sí, verifica terminador
                BTFSC       VTEF            ;¿terminador correcto?
                GOTO        FINACRC         ;no, termina interrupción y reinicializa apuntadores
                CALL        VDMEN           ;sí, valida mensaje
                BTFSC       VDMEF           ;¿mensaje válido?
                GOTO        FINACRC         ;no, termina interrupción y reinicializa apuntadores
                CALL        ORDENA          ;sí, reasigna parámetros
                CALL        ACTUALIZA       ;actualiza parámetros
                GOTO        FINACRC         ;termina interrupción y reinicializa apuntadores
EOERR           BCF         RCSTA,CREN      ;reset la lógica del receptor
                BSF         RCSTA,CREN      ;habilita la recepción de nuevo
                GOTO        FINACRC
EFERR           MOVF        RCREG,W         ;descarta el dato recibido que tiene el error
FINACRC         CALL        INIAP


FINRC           BCF         STATUS,RP0      ; banco 0
                MOVF        status_temp,w   ; W← status_temp
                MOVWF       STATUS          ; STATUS← W
                SWAPF       w_temp,f
                SWAPF       w_temp,w
                RETFIE                      ; retorno de interrupción

;*******************************************************************************
; programa principal                                                           *
;*******************************************************************************
MAIN:
                MOVLW       0x11
                MOVWF       MSGSZ           ; mensaje = 17 caracteres
                MOVLW       0x0D
                MOVWF       TCHAR           ; terminador: carry return
                CALL        INICIALIZA

MLOOP           BTFSC       HAB,0           ; ¿generador encendido?
                GOTO        GLOOP           ; sí, genera load
                BCF         LOAD            ; no, LOAD ← '0
                GOTO        MLOOP

GLOOP           BTFSS       HRT,0           ; ¿RELAY<3?
                GOTO        LOAD2           ; no, ejecuta retardo de 2 niveles
                GOTO        LOAD1           ; sí, ejecuta retardo de 1 nivel
```

```
LOAD1       MOVF        NP,W
            MOVWF       H2
            BCF         LOAD            ; load ? '0
            MOVF        DELB,W
            MOVWF       B1
LL1N1a      DECFSZ      B1,F
            GOTO        LL1N1a
            BSF         LOAD            ; load ? '1'
LOOPL1      DECF        H2,F
            BTFSC       STATUS,Z        ; ¿tren completado?
            GOTO        RET             ; sí, ejecuta retardo de usuario
            MOVF        DELB,W
            MOVWF       B1
LL1N1b      DECFSZ      B1,F
            GOTO        LL1N1b
            GOTO        LOOPL1

LOAD2       MOVF        NP,W
            MOVWF       H2
            BCF         LOAD            ; load ? '0'
            CALL        DELAY2          ; tiempo en bajo = periodo de la senoide
            BSF         LOAD            ; load ? '1'
LOOPL2      DECF        H2,F
            BTFSC       STATUS,Z        ; ¿tren completado?
            GOTO        RET             ; sí, ejecuta retardo de usuario
            CALL        DELAY2          ; no, espera pulso
            GOTO        LOOPL2

RET         CALL        DELAY3          ; retardo establecido (10ms a 100ms)
            GOTO        MLOOP

;*****************************************************************************
; subrutinas                                                                 *
;*****************************************************************************

;====================================
; inicializa módulos                =
;====================================
INICIALIZA:
            CALL        INIES           ; inicializa puertos: entradas/salidas
            CALL        INIPWM          ; inicializa PWM1 & PWM2
            CALL        INIUSART        ; inicializa USART como full duplex
            CALL        HABINT          ; habilita interrupción de recepción
            CALL        DEFCON          ; carga configuración predeterminada
            CALL        VDMEN           ; decodifica mensaje predeterminado
            CALL        ORDENA          ; ordena parámetros
            CALL        ACTUALIZA       ; reasigna parámetros
            CALL        INIAP           ; inicializa apuntadores
            RETURN

;====================================
; inicializa entradas y salidas     =
;====================================
INIES:
            BANKSEL     PORTA
            CLRF        PORTA
```

```
                CLRF        PORTB
                CLRF        PORTC
                BANKSEL     ADCON1
                MOVLW       0x06
                MOVWF       ADCON1      ; PORTA as digital
                CLRF        TRISA       ; PORTA<5:0> → output
                CLRF        TRISB       ; PORTB<7:0> → output
                CLRF        TRISC       ; PORTC<7:0> → output
                RETURN

;=====================================
; inicializa CCP1 y CCP2 como PWM     =
;=====================================
INIPWM:
                BANKSEL     PORTA
                MOVLW       0x18
                MOVWF       PWMp        ; PWMf = 10 kHZ
                MOVLW       0x09
                MOVWF       dc1H        ; ciclo de trabajo 1 = 38%
                MOVLW       0x17
                MOVWF       dc2H        ; ciclo de trabajo 2 = 93%
                MOVF        PWMp,W
                BANKSEL     PR2
                MOVWF       PR2         ; periodo TMR2(PWM1 & PWM2)
                BANKSEL     CCPR1L
                MOVF        dc1H,W
                MOVWF       CCPR1L      ; MSBdc PWM1 ciclo de trabajo
                MOVF        dc2H,W
                MOVWF       CCPR2L      ; MSBdc PWM2 ciclo de trabajo
                BANKSEL     TRISC
                BCF         TRISC,2     ; C2 → PWM1 output
                BCF         TRISC,1     ; C1 → PWM2 output
                BANKSEL     T2CON
                MOVLW       0x07
                MOVWF       T2CON       ; TMR2 → 'ON'; pre → '1:16'
                MOVLW       0x2C        ; LSBdc PWM1 (2)
                MOVWF       CCP1CON     ; CCP1mode → 'PWM' (...C)
                MOVLW       0x1C        ; LSBdc PWM2 (1)
                MOVWF       CCP2CON     ; CCP2mode → 'PWM' (...C)
                RETURN

;=====================================
; inicializa USART como full duplex   =
;=====================================
INIUSART:
                BANKSEL     TRISC
                BSF         TRISC,6     ; PORTC<6> input
                BSF         TRISC,7     ; PORTC<7> input
                BANKSEL     SPBRG
                MOVLW       0x19
                MOVWF       SPBRG       ; baud rate: 9.6K
                MOVLW       0x20
                MOVWF       TXSTA       ; habilita transmisión
                BANKSEL     RCSTA
                MOVLW       0x90
                MOVWF       RCSTA       ; habilita puerto serie y recepción
```

```
                RETURN

;=====================================
; habilita interrupción por recepción =
;=====================================
HABINT:
                BANKSEL     PIE1
                MOVLW       0x20
                MOVWF       PIE1        ; habilita interrupción de recepción
                BANKSEL     INTCON
                MOVLW       0xC0        ; habilita permiso global
                MOVWF       INTCON      ; y permiso de periféricos
                RETURN

;=====================================
; inicializa apuntadores              =
;=====================================
INIAP:
                MOVF        MSGSZ,W
                MOVWF       LISTSZ      ; list size = 17
                MOVLW       LOW LIST01
                MOVWF       FSR         ; apuntando al inicio de la lista
                RETURN

;=====================================
; carga terminador                    =
;=====================================
CARGAT:
                MOVLW       LOW LIST01
                ADDWF       MSGSZ,W
                MOVWF       FSR
                DECF        FSR,F       ; apuntando al fin de la lista
                MOVF        TCHAR,W
                RETURN

;=====================================
; guarda dato recibido                =
;=====================================
GDATO:
                MOVWF       INDF        ; list(i) ← dato recibido
                INCF        FSR,F       ; incremento del apuntador
                RETURN

;=====================================
; verifica terminador                 =
;=====================================
VERTER:
                BCF         VTEF        ; limpia bandera
                CALL        CARGAT      ; FSR ← list(17)
                SUBWF       INDF,W      ; W ← carry return
                BTFSS       STATUS,Z    ; ¿terminador correcto?
                BSF         VTEF        ; no, enciende bandera de error
                RETURN

;=====================================
; valida y decodifica mensaje         =
```

```
;=====================================
VDMEN:
                BCF             VDMEF           ; limpia bandera
                CALL            INIAP
                DECF            LISTSZ,F        ; list size = 16
LOOPVD          CALL            VALDEC
                BTFSC           VDEF            ; ¿dato válido?
                GOTO            MENF            ; no, mensaje inválido
                INCF            FSR,F           ; sí, incrementa apuntador
                DECFSZ          LISTSZ,F        ; ¿fin de la lista?
                GOTO            LOOPVD          ; no, valida otro dato
                RETURN
MENF            BSF             VDMEF
                RETURN

;=====================================
; valida y decodifica dato                  =
;=====================================
VALDEC:
                BCF             VDEF
I1P1            MOVLW           0x30
                SUBWF           INDF,W
                BTFSC           STATUS,C        ; ¿H1 ≤ '0'?
                GOTO            I1P2            ; sí, prueba H1 ≤ '9'
                GOTO            DATF            ; no, dato inválido
I1P2            MOVLW           0x3A
                SUBWF           INDF,W
                BTFSS           STATUS,C        ; ¿H1 ≤ '9'?
                GOTO            DATVI1          ; sí, dato válido
                GOTO            I2P1            ; no, prueba 2do intervalo
I2P1            MOVLW           0x41
                SUBWF           INDF,W
                BTFSC           STATUS,C        ; ¿H1 ≤ 'A'?
                GOTO            I2P2            ; sí, prueba H1 ≤ 'F'
                GOTO            DATF            ; no, dato inválido
I2P2            MOVLW           0x47
                SUBWF           INDF,W
                BTFSS           STATUS,C        ; ¿H1 ≤ 'F'?
                GOTO            DATVI2          ; sí, dato válido
                GOTO            DATF            ; no, dato inválido
DATVI1          MOVLW           0x0F
                ANDWF           INDF,F
                RETURN
DATVI2          MOVLW           0x37
                SUBWF           INDF,F
                RETURN
DATF            BSF             VDEF
                RETURN

;=====================================
; ordena                                    =
;=====================================
ORDENA:
                MOVF            LIST01,W
                MOVWF           dc1H
                SWAPF           dc1H,F
```

```
                MOVF        LIST02,W
                ADDWF       dc1H,F          ; parte alta de PWM1 (amplitud)
                MOVF        LIST03,W
                MOVWF       dc1L
                SWAPF       dc1L,F
                MOVLW       0x0C
                ADDWF       dc1L,F          ; parte baja de PWM1 (amplitud)
                MOVF        LIST04,W
                MOVWF       dc2H
                SWAPF       dc2H,F
                MOVF        LIST05,W
                ADDWF       dc2H,F          ; parte alta de PWM2 (frecuencia)
                MOVF        LIST06,W
                MOVWF       dc2L
                SWAPF       dc2L,F
                MOVLW       0x0C
                ADDWF       dc2L,F          ; parte baja de PWM2 (frecuencia)
                MOVF        LIST07,W
                MOVWF       RELAY           ; selector de frecuencia
                MOVF        LIST08,W
                MOVWF       FASE            ; ajuste de fase
                MOVF        LIST09,W
                MOVWF       NP              ; # de pulsos
                MOVF        LIST10,W
                MOVWF       NPA             ; # de pulsos ajustado
                MOVF        LIST11,W
                MOVWF       DELA
                SWAPF       DELA,F
                MOVF        LIST12,W
                ADDWF       DELA,F          ; tiempo en alto B3 (retardo de 3 niveles)
                MOVF        LIST13,W
                MOVWF       DELB
                SWAPF       DELB,F
                MOVF        LIST14,W
                ADDWF       DELB,F          ; tiempo en bajo B2 o B1 (retardo de 1 o 2 niveles)
                MOVF        LIST15,W
                MOVWF       HRT             ; decisión de retardo
                MOVF        LIST16,W
                MOVWF       HAB             ; habilitación
                RETURN

;=====================================
; actualiza parámetros                =
;=====================================
ACTUALIZA:
                BANKSEL     CCPR1L
                MOVF        dc1H,W
                MOVWF       CCPR1L          ; MSBdc PWM1 ciclo de trabajo
                MOVF        dc1L,W
                MOVWF       CCP1CON         ; actualiza amplitud
                MOVF        dc2H,W
                MOVWF       CCPR2L          ; MSBdc PWM2 ciclo de trabajo
                MOVF        dc2L,W
                MOVWF       CCP2CON         ; actualiza frecuencia
                BANKSEL     PORTA
                MOVF        RELAY,W
```

```
                MOVWF       PORTA       ; actualiza el selector de frecuencia
                SWAPF       NPA,W
                BTFSC       FASE,0
                ADDLW       0x04
                MOVWF       PORTB       ; actualiza # de pulsos ajustado y fase
                RETURN

;====================================
; configuración predeterminada      =
;====================================
DEFCON:
                BANKSEL     PORTA
                CLRF        AUX
                CALL        INIAP
                DECF        LISTSZ,F    ; list size = 16
LOOPDC          CALL        TABLA
                MOVWF       INDF        ; list(i) ← tabla(i)
                INCF        FSR,F       ; apuntando list(i+1)
                INCF        AUX,F       ; apuntando tabla(i+1)
                DECFSZ      LISTSZ,F    ; ¿fin de la lista?
                GOTO        LOOPDC      ; no, asigna otro dato
                RETURN

;====================================
; retardo de 1 nivel                =
;====================================
DELAY1:
                MOVF        DELB,W
                MOVWF       B1
LOOP1N          DECFSZ      B1,F
                GOTO        LOOP1N
                RETURN

;====================================
; retardo de 2 niveles              =
;====================================
DELAY2:
                MOVLW       0x41
                MOVWF       B1
                MOVWF       BH1
                MOVF        DELB,W
                MOVWF       B2
LOOP2N          DECFSZ      B1,F
                GOTO        LOOP2N
                MOVF        BH1,W
                MOVWF       B1
                DECFSZ      B2,F
                GOTO        LOOP2N
                RETURN

;====================================
; retardo de 3 niveles              =
;====================================
DELAY3:
                MOVLW       0xFF
                MOVWF       B1
```

```
                MOVWF       BH1
                MOVLW       0x03
                MOVWF       B2
                MOVWF       BH2
                MOVF        DELA,W
                MOVWF       B3
LOOP3N          DECFSZ      B1,F
                GOTO        LOOP3N
                MOVF        BH1,W
                MOVWF       B1
                DECFSZ      B2,F
                GOTO        LOOP3N
                MOVF        BH2,W
                MOVWF       B2
                DECFSZ      B3,F
                GOTO        LOOP3N
                RETURN

;=====================================
; mensaje de configuración           =
;=====================================
TABLA:
                MOVLW       HIGH TABLA
                MOVWF       PCLATH
                MOVF        AUX,W           ; apuntando a tabla(i)
                ADDWF       PCL,F
                DT '0','9','2','1','7','1','2','0'     ; f=10kHz a=2Vpp fase='+'
                DT '0','F','1','1','7','E','1','0'     ; 3 pulsos ret=10ms APAGADO


;*****************************************************************************
; fin del programa                                                           *
;*****************************************************************************

                END

;*****************************************************************************
```

Apéndice C: Set de Instrucciones

Cada instrucción del PIC16F873 es una palabra de 14 bits dividida en un OPCODE (Código de Operación) que especifica el tipo y los operandos de la instrucción. La *tabla C-1* muestra la descripción del campo opcode.

Campo	Descripción
f	Dirección del registro (0x00 a 0x7F)
W	Registro de trabajo (acumulador)
b	Dirección de un bit dentro de un registro de 8 bits
k	Campo para constante, literal o etiqueta
x	Localidad no importa (0 o 1) El ensamblador generará el código con x=0. Esta es la forma recomendada para la compatibilidad con todas las herramientas de software de Microchip
d	Selección del destino; d = 0: guarda resultado en W, d = 1: guarda resultado en el registro f. La condición predeterminada es d = 1.
PC	Contador de programa
TO	Bit Time-out
PD	Bit Power-down

Tabla C-1 Descripción de los campos OPCODE

El set de instrucciones es altamente ortogonal y está agrupado en tres categorías:

- **Operaciones orientadas a bytes**. Para estas instrucciones, ‘f’ representa el registro de archivo designado que será usado por la instrucción mientras que ‘d’ representa el destino designado en el que se guardará el resultado de la operación. Si ‘d’ es cero, el resultado se almacena en el registro W. Si ‘d’ es uno, el resultado se guarda en el registro de archivo especificado en la instrucción.

- **Operaciones orientadas a bits**. Para estas instrucciones, ‘b’ representa el campo para el bit designado que selecciona el número de bit afectado por la operación, mientras que ‘f’ representa la dirección del archivo en el que se localiza el bit.

- **Operaciones orientadas a control y literales**. Para estas instrucciones, ‘k’ representa una constante de ocho u once bits o el valor de una literal.

Todas las instrucciones se ejecutan dentro de un solo ciclo de instrucción, a menos que una prueba condicional resulte verdadera o que el contador de programa sea modificado como resultado de una instrucción. En este caso, la ejecución requiere de dos ciclos de instrucción y el segundo ciclo se ejecuta como un NOP. Un ciclo de instrucción consiste en cuatro periodos del oscilador. Por lo tanto, para una frecuencia de oscilación de 4 MHz, el tiempo de ejecución es de 1 μs. Si una prueba condicional resulta verdadera, o el contador de programa se modifica como resultado de una instrucción, el tiempo de ejecución de la instrucción es de 2 μs.

La *tabla C-2* enlista las instrucciones reorganizadas por el ensamblador MPASM y la *figura C-1* muestra el formato general que las instrucciones pueden presentar.

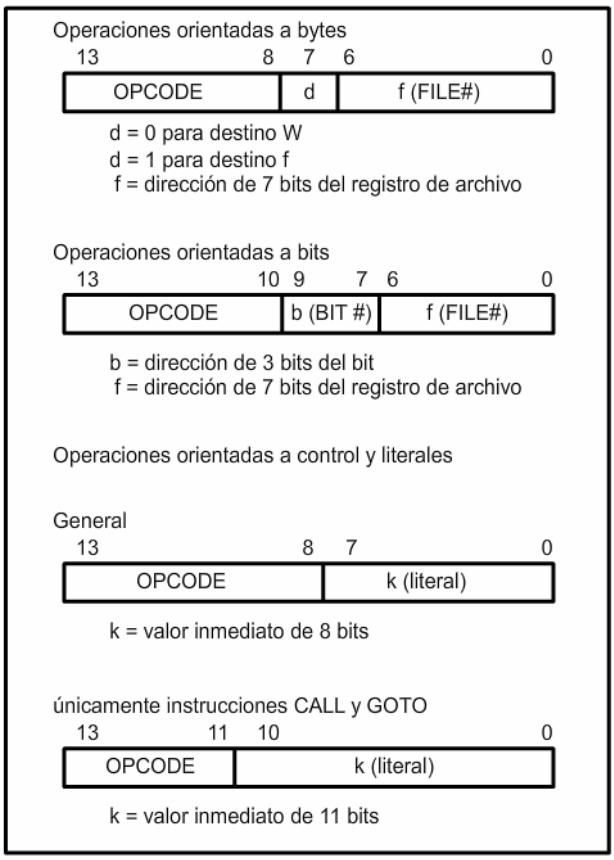


Figura C-1 Formato general de las instrucciones

Todos los ejemplos emplean el siguiente formato para representar un número hexadecimal:

0xhh

donde h significa un dígito hexadecimal.

Nemónic operan		Descripción	ciclos	14-bit Opcode MSb LSb	Status afectado	Nota
Operaciones orientadas a bytes						
ADDWF	f, d	Suma W y f	1	00 0111 dfff ffff	C, DC, Z	1, 2
ANDWF	f, d	AND W con f	1	00 0101 dfff ffff	Z	1, 2
CLRF	f	Borra F	1	00 0001 1fff ffff	Z	2
CLRW	'-	Borra W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complementa f	1	00 1001 dfff ffff	Z	1, 2
DECF	f, d	Decrementa f	1	00 0011 dfff ffff	Z	1, 2
DECFSZ	f, d	Decrementa f, salta si es 0	1 (2)	00 1011 dfff ffff		1, 2, 3
INCF	f, d	Incrementa f	1	00 1010 dfff ffff	Z	1, 2
INCFSZ	f, d	Incrementa s, salta si es 0	1 (2)	00 1111 dfff ffff		1, 2, 3
IORWF	f, d	OR W con f	1	00 0100 dfff ffff	Z	1, 2
MOVF	f, d	Mueve f	1	00 1000 dfff ffff	Z	1, 2
MOVWF	f, d	Mueve W a f	1	00 0000 1fff ffff		
NOP	'-	No operes	1	00 0000 0xx0 0000		
RLF	f, d	Rota f a la izquierda através de Carry	1	00 1101 dfff ffff	C	1, 2
RRF	f, d	Rota f a la derecha através de Carry	1	00 1100 dfff ffff	C	1, 2
SUBWF	f, d	Resta W de f	1	00 0010 dfff ffff	C, DC, Z	1, 2
SWAPF	f, d	Intercambia los nibbles de f	1	00 1110 dfff ffff		1, 2
XORWF	f, d	XOR W con f	1	00 0110 dfff ffff	Z	1, 2
Operaciones orientadas a bits						
BCF	f, d	Borra bit de f	1	01 00bb bfff ffff		1, 2
BSF	f, d	Enciende bit de f	1	01 01bb bfff ffff		1, 2
BTFSC	f, d	Prueba bit de f, salta si es 0	1 (2)	01 10bb bfff ffff		3
BTFSS	f, d	Prueba bit de f, salta si es 1	1 (2)	01 11bb bfff ffff		3
Operaciones de control y literales						
ADDLW	k	Suma literal y W	1	11 111x kkkk kkkk	C, DC, Z	
ANDLW	k	AND literal con W	1	11 1001 kkkk kkkk	Z	
CALL	k	Llamada a subrutina	2	10 0kkk kkkk kkkk		
CLRWDT	-	Borra el Watchdog Timer	1	00 0000 0110 0100	$\overline{TO}$, $\overline{PD}$	
GOTO	k	Salta a dirección	2	10 1kkk kkkk kkkk		
IORLW	k	OR literal con W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Mueve literal a W	1	11 00kk kkkk kkkk		
RETFIE	-	Retorno de interrupción	2	00 0000 0000 1001		
RETLW	k	Retorno con literal en W	2	11 01kk kkkk kkkk		
RETURN	-	Retorno de subrutina	2	00 0000 0000 1000		
SLEEP	-	Cambia a modo de espera	1	00 0000 0110 0011	$\overline{TO}$, $\overline{PD}$	
SUBLW	k	Resta W de literal	1	11 110k kkkk kkkk	C, DC, Z	
XORLW	k	XOR literal con W	1	11 1010 kkkk kkkk	Z	

Tabla C-2 Set de instrucciones de PIC16F873

Nota 1: Cuando un registro de entrada/salida es modificado como una función de sí mismo, (e.g. MOVF PORTB, 1), el valor empleado será aquel presente en las mismas terminales. Por ejemplo, si el dato en el latch es '1' para una terminal configurada como entrada y es llevada a un nivel bajo por un dispositivo externo, el dato se escribirá como '0'.

Nota 2: Si la instrucción se ejecuta en el registro TMR0 (y, si es aplicable, d=1), el preescalador se borrará si está asignado al modulo del Timer0.

3: Si el contador de programa es modificado o una prueba condicional resulta verdadera, la instrucción requiere dos ciclos. El segundo ciclo se ejecuta como un NOP.

Descripción de las instrucciones

ADDLW	**AND Literal and W**
Sintaxis:	[etiqueta] ADDLW k
Operandos:	0 ≤ k ≤ 255
Operación:	(W) + k → (W)
STATUS afectado:	C, DC, Z
Descripción	El contenido del registro W se suma a literal de ocho bits 'k' y el resultado se guarda en el registro W.

ANDWF	**AND W with f**
Sintaxis:	[etiqueta] ANDWF f,d
Operandos:	0 ≤ f ≤ 127 d ∈ [0,1]
Operación:	(W) .AND. (f) → (destino)
STATUS afectado:	Z
Descripción	Se realiza la operación AND entre el registro W y el registro 'f'. Si 'd' es 0, el resultado se guarda en el registro W. Si 'd' es 1, el resultado se almacena en el registro 'f'.

ADDWF	**AND W and f**
Sintaxis:	[etiqueta] ADDWF f,d
Operandos:	0 ≤ f ≤ 127 d ∈ [0,1]
Operación:	(W) + (f) → (destino)
STATUS afectado:	C, DC, Z
Descripción	Suma el contenido del registro W con el registro 'f'. Si 'd' es 0, el resultado se guarda en el registro W. Si 'd' es 1, el resultado se vuelve a guardar en el registro 'f'.

BCF	**Bit Clear f**
Sintaxis:	[etiqueta] BCF f,b
Operandos:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operación:	0 → (f<b>)
STATUS afectado:	Ninguno
Descripción	El bit 'b' del registro 'f' es limpiado.

ANDLW	**AND Literal with W**
Sintaxis:	[etiqueta] ADDLW k
Operandos:	0 ≤ k ≤ 255
Operación:	(W) .AND. (k) → (W)
STATUS afectado:	Z
Descripción	Se realiza la operación AND entre el contenido del registro W y una literal 'k' de ocho bits. El resultado se guarda en el registro W.

BSF	**Bit Set f**
Sintaxis:	[etiqueta] BCF f,b
Operandos:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operación:	1 → (f<b>)
STATUS afectado:	Ninguno
Descripción	El bit 'b' del registro 'f' es encendido.

BTFSS	**Bit Test f, Skip if Set**
Sintaxis:	[etiqueta] BTFSS f,b
Operandos:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operación:	salta si (f<b>) = 1
STATUS afectado:	Ninguno
Descripción	Si el bit 'b' del registro 'f' es '0', la siguiente instrucción es ejecutada. Si el bit 'b' es '1', la siguiente instrucción se descarta y una NOP es ejecutada en su lugar haciendo de esta, una instrucción de $2T_{CY}$.

BTFSC	**Bit Test f, Skip if Clear**
Sintaxis:	[etiqueta] BTFSC f,b
Operandos:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operación:	salta si (f<b>) = 0
STATUS afectado:	Ninguno
Descripción	Si el bit 'b' del registro 'f' es '1', la siguiente instrucción es ejecutada. Si el bit 'b' es '0', la siguiente instrucción se descarta y una NOP es ejecutada en su lugar haciendo de esta, una instrucción de $2T_{CY}$.

CALL	**CALL Subroutine**
Sintaxis:	[etiqueta] CALL k
Operandos:	0 ≤ k ≤ 2047
Operación:	(PC) + 1 → TOS, k → PC<10:0>, (PCLATCH<4:3>)→ PC<12:11>
STATUS afectado:	Ninguno
Descripción	Llamada a subrutina. Primero, la instrucción de retorno (PC+1) se guarda en la pila. La dirección inmediata de 11 bits se carga a los bits <10:0> de PC. La parte alta de PC se carga desde PCLATCH. CALL se ejecuta en dos ciclos de instrucción.

CLRF	**Clear f**
Sintaxis:	[etiqueta] CLRF f
Operandos:	0 ≤ f ≤ 127
Operación:	00h → (W) 1 → Z
STATUS afectado:	Z
Descripción	El contenido del registro 'f' es borrado y el bit Z es encendido.

CLRW	**Clear W**
Sintaxis:	[etiqueta] CLRW
Operandos:	Ninguno
Operación:	00h → (W) 1 → Z
STATUS afectado:	Z
Descripción	El contenido del registro W es borrado y el bit Z es encendido.

CLRWDT	**Clear Watchdog Timer**
Sintaxis:	[etiqueta] CLRWDT
Operandos:	Ninguno
Operación:	00h → WDT 0 → WDT prescaler 1 → $\overline{TO}$ 1 → $\overline{PD}$
STATUS afectado:	$\overline{TO}$, $\overline{PD}$
Descripción	La instrucción CLRWDT reinicia el Watchdog Timer. Además, se reinicia el preescalador del WDT y los bits de estado $\overline{TO}$ y $\overline{PD}$ se encienden.

COMF	**Complement f**
Sintaxis:	[etiqueta] COMF f,b
Operandos:	0 ≤ f ≤ 127 d ∈ [0,1]
Operación:	($\bar{f}$) → (destino)
STATUS afectado:	Z
Descripción	El contenido del registro 'f' se complementa. Si 'd' es 0, el resultado se almacena en W. Si 'd' es 1, el resultado se vuelve a guarda en el registro 'f'.

GOTO	**Unconditional Branch**
Sintaxis:	[etiqueta] GOTO k
Operandos:	0 ≤ k ≤ 2047
Operación:	k → PC<10:0> PCLATCH<4:3> → PC<12:11>
STATUS afectado:	Ninguno
Descripción	GOTO es un salto incondicional. El valor inmediato de 11 bits es cargado en PC<10:0>. La parte alta de PC se carga desde PCLATCH<4:3>. GOTO se ejecuta en dos ciclos de instrucción.

DECF	**Decrement f**
Sintaxis:	[etiqueta] DECF f,b
Operandos:	0 ≤ f ≤ 127 d ∈ [0,1]
Operación:	(f) - 1 → (destino)
STATUS afectado:	Z
Descripción	Decrementa el registro 'f'. Si 'd' es 0, el resultado se almacena en W. Si 'd' es 1, el resultado se vuelve a guarda en el registro 'f'.

INCF	**Increment f**
Sintaxis:	[etiqueta] INCF f,d
Operandos:	0 ≤ f ≤ 127 d ∈ [0,1]
Operación:	(f) + 1 → (destino)
STATUS afectado:	Z
Descripción	El contenido del registro 'f' es incrementado. Si 'd' es 0, el resultado se almacena en el registro W. Si 'd' es 1, el resultado se vuelve a guardar en el registro 'f'.

DECFSZ	**Decrement f, Skip if 0**
Sintaxis:	[etiqueta] DECFSZ f,b
Operandos:	0 ≤ f ≤ 127 d ∈ [0,1]
Operación:	(f) - 1 → (destino); salta si resultado = 0
STATUS afectado:	Ninguno
Descripción	El contenido del registro 'f' es decrementado. Si 'd' es 0, el resultado se almacena en W. Si 'd' es 1, el resultado se vuelve a guarda en el registro 'f'. Si el resultado es 1, la próxima instrucción es ejecutada. Si el resultado es 0, un NOP es ejecutado y la instrucción se realiza en dos ciclos de instrucción.

INCFSZ	**Increment f, Skip if 0**
Sintaxis:	[etiqueta] INCFSZ f,d
Operandos:	0 ≤ f ≤ 127 d ∈ [0,1]
Operación:	(f) + 1 → (destino); salta si resultado = 0
STATUS afectado:	Ninguno
Descripción	El contenido del registro 'f' es incrementado. Si 'd' es 0, el resultado se almacena en W. Si 'd' es 1, el resultado se vuelve a guarda en el registro 'f'. Si el resultado es 1, la próxima instrucción es ejecutada. Si el resultado es 0, un NOP es ejecutado y la instrucción se realiza en dos ciclos de instrucción.

IORLW	**Inclusive OR Literal with W**
Sintaxis:	[etiqueta] IORLW k
Operandos:	0 ≤ k ≤ 255
Operación:	(W) .OR. (k) → (W)
STATUS afectado:	Z
Descripción	Se realiza la operación OR entre el contenido del registro W y una literal 'k' de ocho bits. El resultado se guarda en el registro W.

MOVLW	**Move Literal to W**
Sintaxis:	[etiqueta] MOVLW k
Operandos:	0 ≤ k ≤ 255 d ∈ [0,1]
Operación:	(k) → (W)
STATUS afectado:	Ninguno
Descripción	La literal 'k' de 8 bits es cargada en el registro W.

IORWF	**Inclusive OR W with f**
Sintaxis:	[etiqueta] IORWF f,d
Operandos:	0 ≤ f ≤ 127 d ∈ [0,1]
Operación:	(W) .OR. (f) → (destino)
STATUS afectado:	Z
Descripción	Se realiza la operación OR entre el registro W y el registro 'f'. Si 'd' es 0, el resultado se guarda en el registro W. Si 'd' es 1, el resultado se almacena en el registro 'f'.

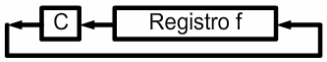
MOVWF	**Move W to f**
Sintaxis:	[etiqueta] MOVWF f
Operandos:	0 ≤ f ≤ 127
Operación:	(W) → (f)
STATUS afectado:	Ninguno
Descripción	Transfiere el dato del registro W al registro 'f'.

MOVF	**Move f**
Sintaxis:	[etiqueta] MOVF f,d
Operandos:	0 ≤ f ≤ 127 d ∈ [0,1]
Operación:	(f) → (destino)
STATUS afectado:	Z
Descripción	El contenido del registro f se transfiere a un destino que depende del estado de d. Si d = 0, el destino es el registro W. Si d = 1, el destino es el mismo registro f. Para probar un registro, la condición d = 1 es útil, ya que el bit Z se ve afectado.

NOP	**No Operation**
Sintaxis:	[etiqueta] NOP
Operandos:	Ninguno
Operación:	Ninguna operación
STATUS afectado:	Ninguno
Descripción	Ninguna operación

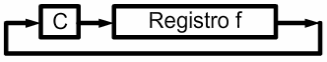
RETFIE	**Return from Interrupt**
Sintaxis:	[etiqueta] RETFIE
Operandos:	Ninguno
Operación:	TOS → PC 1 → GIE
STATUS afectado:	Ninguno
Descripción	Retorno de interrupción. El PC se carga con la parte alta de la pila y se habilita el permiso de interrupción. RETFIE se ejecuta en dos ciclos de instrucción.

RLF	**Rotate Left f through Carry**
Sintaxis:	[etiqueta] RLF f,d
Operandos:	0 ≤ f ≤ 127 d ∈ [0,1]
Operación:	ver detalle
STATUS afectado:	C
Descripción	El contenido del registro 'f' es rotado un bit a la izquierda pasando por el bit C. Si 'd' es 0, el resultado se almacena en el registro W. Si 'd' es 1, el resultado se vuelve a guardar en el registro 'f'.



RETLW	**Return with Literal in W**
Sintaxis:	[etiqueta] RETLW k
Operandos:	0 ≤ k ≤ 255
Operación:	k → (W); TOS → PC
STATUS afectado:	Ninguno
Descripción	El registro W es cargado con la literal de 8 bits 'k'. El PC es cargado con la parte alta de la pila (dirección de retorno) RETLW se ejecuta en dos ciclos de instrucción.

RRF	**Rotate Right f through Carry**
Sintaxis:	[etiqueta] RRF f,d
Operandos:	0 ≤ f ≤ 127 d ∈ [0,1]
Operación:	ver detalle
STATUS afectado:	C
Descripción	El contenido del registro 'f' es rotado un bit a la derecha pasando por el bit C. Si 'd' es 0, el resultado se almacena en el registro W. Si 'd' es 1, el resultado se vuelve a guardar en el registro 'f'.



RETURN	**Return from Subroutine**
Sintaxis:	[etiqueta] RETURN
Operandos:	Ninguno
Operación:	TOS → PC
STATUS afectado:	Ninguno
Descripción	Retorno de subrutina. El PC se carga con la parte alta de la pila. RETURN se ejecuta en dos ciclos de instrucción.

SLEEP	
Sintaxis:	[etiqueta] SLEEP
Operandos:	Ninguno
Operación:	00h → WDT 0 → $\overline{\text{WDT}}$ prescaler 1 → $\overline{\text{TO}}$ 0 → $\overline{\text{PD}}$
STATUS afectado:	$\overline{\text{TO}}$, $\overline{\text{PD}}$
Descripción	El bit de status $\overline{\text{PD}}$ es limpiado. El bit de status $\overline{\text{TO}}$ es encendido. El WDT y su preescalador son limpiados. El procesador cambia al modo SLEEP con el oscilador detenido.

SUBLW	**Subtract W from Literal**
Sintaxis:	[etiqueta] SUBLW k
Operandos:	0 ≤ k ≤ 255
Operación:	k - (W) → (W)
STATUS afectado:	C, DC, Z
Descripción	El contenido del registro W se sustrae (por complemento a 2) de la literal de ocho bits 'k'. El resultado se guarda en el registro W.

SUBWF	**Subtract W from f**
Sintaxis:	[etiqueta] SUBWF f,d
Operandos:	0 ≤ f ≤ 127 d ∈ [0,1]
Operación:	(f) - (W) → (destino)
STATUS afectado:	C, DC, Z
Descripción	Resta (por complemento a 2) el contenido del registro W del registro 'f'. Si 'd' es 0, el resultado se almacena en el registro W. Si 'd' es 1, el resultado se vuelve a guardar en el registro f'.

SWAPF	**Swap Nibbles in f**
Sintaxis:	[etiqueta] SWAPF f,d
Operandos:	0 ≤ f ≤ 127 d ∈ [0,1]
Operación:	(f<3:0>) → (destino<7:4>), (f<7:4>) → (destino<3:0>)
STATUS afectado:	Ninguno
Descripción	Los nibbles alto y bajo del registro 'f' son intercambiados. Si 'd' es 0, el resultado se almacena en el registro W. Si 'd' es 1, el resultado se vuelve a guardar en el registro 'f'.

XORLW	**Exclusive OR Literal with W**
Sintaxis:	[etiqueta] XORLW k
Operandos:	0 ≤ k ≤ 255
Operación:	(W) .XOR. (k) → (W)
STATUS afectado:	Z
Descripción	Se realiza la operación XOR entre el contenido del registro W y una literal 'k' de ocho bits. El resultado se guarda en el registro W.

XORWF	**Exclusive OR W with f**
Sintaxis:	[etiqueta] XORWF f,d
Operandos:	0 ≤ f ≤ 127 d ∈ [0,1]
Operación:	(W) .XOR. (f) → (destino)
STATUS afectado:	Z
Descripción	Se realiza la operación XOR entre el registro W y el registro 'f'. Si 'd' es 0, el resultado se guarda en el registro W. Si 'd' es 1, el resultado se almacena en el registro f'.

Apéndice D: Registros Especiales

Registro D-1: Registro STATUS (direcciones 03h, 83h, 103h, 183h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit7							bit0

bit 7: **IRP:** Bit de selección para el banco de registro (usado para direccionamiento indirecto)
1 = Banco 2, 3 (100h – 1FFh)
0 = Banco 0, 1 (00h – FFh)

bit 6-5: **RP1:RP0:** Bits de selección para el banco de registros (usados para direccionamiento directo)
11 = Banco 3 (180h – 1FFh)
10 = Banco 2 (100h – 17Fh)
01 = Banco 1 (80h – FFh)
00 = Banco 0 (00h – 7Fh)
Cada banco contiene 128 bytes

bit 5: **$\overline{TO}$:** Time-out bit
1 = Después de encendido o de las instrucciones CLRWDT o SLEEP
0 = Ha ocurrido un WDT time-out

bit 4: **$\overline{PD}$:** Power-down bit
1 = Después del encendido o por la instrucción CLRWDT
0 = Por ejecución de la instrucción SLEEP

bit 3: **Z**: Cero
1 = El resultado de una operación aritmética o lógica es cero.
1 = El resultado de una operación aritmética o lógica es no cero.

bit 2: **DC:** Digit carry/borrow
1 = Ha ocurrido un desbordamiento en la parte baja (bit 4).
0 = No ha ocurrido un desbordamiento en la parte baja (bit 4).

bit 1: **C:** Digit carry/borrow
1 = Ha ocurrido un desbordamiento del bit más significativo.
0 = No ha ocurrido un desbordamiento del bit más significativo

Registro D-2: Registro INTCON (dirección 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit7							bit0

bit 7: **GIE:** Permiso global de interrupción
1 = Habilita todas las interrupciones sin máscara
0 = Deshabilita todas las interrupciones

bit 6: **PEIE:** Permiso interrupción por periféricos
1 = Habilita todas las interrupciones periféricas sin máscara
0 = Deshabilita todas las interrupciones periféricas

bit 5: **T0IE:** Permiso de interrupción por desbordamiento del TMR0
1 = Habilita interrupción del TMR0
0= Deshabita interrupción del TMR0

bit 4: **INTE:** Permiso de interrupción externa por RB0/INT
1 = Habilita interrupción externa por RB0/INT
0= Deshabilita interrupción externa por RB0/INT

bit 3: **RBIE**: Permiso de interrupción por cambio del puerto RB
1 = Habilita interrupción por cambio del puerto RB
0 = Deshabilita interrupción por cambio del puerto RB

bit 2: **T0IF:** Bandera de interrupción por desbordamiento del TMR0
1 = Ha ocurrido un desbordamiento del TMR0 (debe limpiarse por software)
0 = Sin desbordamiento del TMR0

bit 1: **INTF:** Bandera de interrupción externa por RB0/INT
1 = Ha ocurrido una interrupción externa por RB0/INT (debe limpiarse por software)
0 = No ha ocurrido una interrupción externa por RB0/INT

bit 0: **RBIF:** Bandera de interrupción por cambio del puerto RB
1 = Al menos uno de los bits RB7:RB4 ha cambiado de estado (debe limpiarse por software)
0 = Ninguno de los bits RB7:RB4 ha cambiado de estado

Registro D-3: Registro PIE1 (dirección 8Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit7							bit0

bit 7: **PSPIE(1):** Permiso de interrupción por lectura o escritura del puerto paralelo esclavo
1 = Habilita interrupción por lectura/escritura del PSP
0 = Deshabilita interrupción por lectura/escritura del PSP

bit 6: **ADIE:** Permiso interrupción por el convertidor A/D
1 = Habilita la interrupción del convertidor A/D
0 = Deshabilita la interrupción del convertidor A/D

bit 5: **RCIE:** Permiso de interrupción por recepción del USART
1 = Habilita la interrupción por recepción del USART
0 = Deshabilita la interrupción por recepción del USART

bit 4: **TXIE:** Permiso de interrupción por transmisión del USART
1 = Habilita la interrupción por transmisión del USART
0 = Deshabilita la interrupción por transmisión del USART

bit 3: **SSPIE**: Permiso de interrupción del puerto serie síncrono
1 = Habilita la interrupción del SSP
0 = Deshabilita la interrupción del SSP

bit 2: **CCP1IE:** Permiso de interrupción del CCP1
1 = Habilita la interrupción del CCP1
0 = Deshabilita la interrupción del CCP1

bit 1: **TMR2IE:** Permiso de interrupción por igualación de TMR2 y PR2
1 = Habilita la interrupción por igualación de TMR2 y PR2
0 = Deshabilita la interrupción por igualación de TMR2 y PR2

bit 0: **TMR1IE:** Permiso de interrupción por desbordamiento del TMR1
1 = Habilita la interrupción por desbordamiento del TMR1
0 = Deshabilita la interrupción por desbordamiento del TMR1

Nota 1: PSPIE está reservado, por lo que este bit siempre debe mantenerse en ‘0’.

Registro D-4: Registro PIR1 (dirección 0Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit7							bit0

bit 7: **PSPIF(1):** Bandera de interrupción de lectura/escritura puerto esclavo paralelo
1 = Ha ocurrido una operación de lectura o escritura (debe limpiarse por software)
0 = No ha ocurrido ni lectura ni escritura

bit 6: **ADIF:** Bandera de interrupción del convertidor A/D
1 = Una conversión A/D se ha completado
0 = La conversión A/D no está completa.

bit 5: **RCIF:** Bandera de interrupción por recepción del USART
1 = El búfer de recepción del USART está lleno.
0 = El búfer de recepción del USART está vacío.

bit 4: **TXIF:** Bandera de interrupción por transmisión del USART
1 = El búfer de transmisión del USART está vacío.
0 = El búfer de transmisión del USART está lleno.

bit 3: **SSPIF:** Bandera de interrupción del puerto serie síncrono (SSP)
1 = Ha ocurrido la condición de interrupción del SSP y debe limpiarse por software antes de salir de la rutina de interrupción.
0 = No ha ocurrido la condición de interrupción del SSP..

bit 2: **CCP1IF:** Bandera de interrupción del CCP1
Modo de captura
1 = Ha ocurrido una captura en el registro TMR1.
0 = No ha ocurrido una captura en el registro TMR1.
Modo de comparación
1 = Hay concordancia en la comparación del registro TMR1
0 = No hay concordancia en la comparación del registro TMR1
Modo PWM
Bit sin uso en este modo.

bit 1: **TMR2IF:** Bandera de interrupción de concordancia entre TMR2 y PR2
1 = Hay concordancia entre TMR2 y PR2 (debe limpiarse por software)
0 = No hay concordancia entre TMR2 y PR2.

bit 0: **TMR1IF:** Bandera de interrupción del desbordamiento de TMR1
1 = El registro TMR1 se ha desbordado (debe limpiarse por software)
0 = El registro TMR1 no se ha desbordado

Nota 1: PSPIF está reservado, por lo que este bit siempre debe mantenerse en ‘0’.

Registro D-5: Registro ADCON1 (dirección 9Fh)

U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	---	---	---	PCFG3	PDFG2	PCFG1	PCFG0
bit7							bit0

bit 7: **ADFM:** Selección del formato de resultado A/D
1 = Justificado a la derecha. Los 6 bits más significativos de ADRESH se leen como '0'
0 = Justificado a la izquierda. Los 6 bits menos significativos de ADRESL se leen como '0'

bit 6-4: **No disponibles:** Leídos como '0'

bit 3-0: **PCFG3:PCFG0:** Bits para el control de configuración del puerto A/D

PCFG3: PCFG0	AN4 RA4	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	V_{REF+}	V_{REF-}	CHAN / Refs(1)
0000	A	A	A	A	A	V_{DD}	V_{SS}	8/0
0001	A	V_{REF+}	A	A	A	RA3	V_{SS}	7/1
0010	A	A	A	A	A	V_{DD}	V_{SS}	5/0
0011	A	V_{REF+}	A	A	A	RA3	V_{SS}	4/1
0100	D	A	D	A	A	V_{DD}	V_{SS}	3/0
0101	D	V_{REF+}	D	A	A	RA3	V_{SS}	2/1
011x	D	D	D	D	D	V_{DD}	V_{SS}	0/0
1000	A	V_{REF+}	V_{REF-}	A	A	RA3	RA2	6/2
1001	A	A	A	A	A	RA3	V_{SS}	6/0
1010	A	V_{REF+}	A	A	A	RA3	V_{SS}	5/1
1011	A	V_{REF+}	V_{REF-}	A	A	RA3	RA2	4/2
1100	A	V_{REF+}	V_{REF-}	A	A	RA3	RA2	3/2
1101	D	V_{REF+}	V_{REF-}	A	A	RA3	RA2	2/2
1110	D	D	D	D	A	V_{DD}	V_{SS}	1/0
1111	D	V_{REF+}	V_{REF-}	D	A	RA3	RA2	1/2

A = Entrada analógica D = Entrada/salida digital

Nota 1: Esta columna indica el número de canales analógicos disponibles como entradas A/D y el número de canales analógicos usados como entradas de referencia de voltaje.

Registro D-6: Registro T2CON (dirección 12h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
---	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit7							bit0

bit 7: **No disponible:** Leído como '0'

bit 6-3: **TOUTPS3:TOUTPS0:** Bits de selección para el postescalador de la salida del Timer2
0000 = 1:1 Postescalamiento
0001 = 1:1 Postescalamiento
0010 = 1:1 Postescalamiento
•
•
•
1111 = 1:16 Postescalamiento

bit 2: **TMR2ON:** Bit de encendido del Timer2
1 = Timer2 encendido
0 = Timer2 apagado

bit 1-0: **T2CKPS1:T2CKPS0:** Bits de selección para el preescalador del reloj del Timer2
00 = El factor del preescalador es 1
01= El factor del preescalador es 4
1x = El factor del preescalador es 16

Registro D-7: Registro CCP1CON/CCP2CON (dirección: 17h/1Dh)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
---	---	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit7							bit0

bit 7-6: **No disponibles:** Leídos como ‘0’

bit 5-4: **CCPxX:CCPxY:** Bits menos significativos del PWM
Modo de Captura: sin uso
Modo de Comparación: sin uso
Modo PWM: Bits menos significativos del ciclo de trabajo del PWM. Los ocho bits más significativos se encuentran en el registro CCPRxL.

bit 3-0: **CCPxM3:CCPxM0:** Bits de selección para el modo de CCPx
0000 = Captura / Comparación / PWM apagado (reinicio del módulo CCPX)
0100 = Modo de captura, cada borde de bajada
0101 = Modo de captura, cada borde de subida
0110 = Modo de captura, cada 4° borde de subida
0111 = Modo de captura, cada 16° borde de subida
1000 = Modo de comparación, encender salida
1001 = Modo de comparación, apagar salida
1010 = Modo de comparación, genera interrupción por software
1011 = Modo de comparación, evento especial de disparo
11xx = Modo PWM

Registro D-8: Registro TXSTA (dirección 98h)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	---	BRGH	TRMT	TX9D
bit7							bit0

bit 7: **CSRC:** Bit de selección para el reloj
Modo asíncrono
No importa
Modo síncrono
1 = Modo maestro (Reloj generado internamente por BRG)
0 = Modo esclavo (Reloj generado por fuente externa)

bit 6: **TX9:** Bit de habilitación para la transmisión de 9 bits
1 = Transmisión de 9 bits
0 = Transmisión de 8 bits

bit 5: **TXEN:** Habilitación de la transmisión
1 = Transmisión habilitada
0 = Transmisión deshabilitada

bit 4: **SYNC:** Bit de selección para el modo del USART
1 = Modo síncrono
0 = Modo asíncrono

bit 3: **No disponible**: Leído como '0'

bit 2: **BRGH:** Bit de selección para una elevada velocidad de transmisión
Modo asíncrono
1 = Alta velocidad
0 = Baja velocidad
Modo síncrono
Sin uso en este modo

bit 1: **TRMT:** Bit de status para el registro de corrimiento de transmisión
1 = TSR vacío
0 = TSR lleno TMR1

bit 0: **TX9D:** 9° bit de transmisión. (Puede ser el bit de paridad)

Registro D-9: Registro RCSTA (dirección 18h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit7							bit0

bit 7: **SPEN:** Bit de habilitación del puerto serial
1 = Puerto serial habilitado (Configurar RC7/RX/DT y RC6/TX/CK como pines del puerto serial)
0 = Puerto serial deshabilitado

bit 6: **RX9:** Bit de habilitación para la recepción de 9 bits
1 = Recepción de 9 bits
0 = Recepción de 8 bits

bit 5: **SREN:** Bit de habilitación para la recepción simple
Modo asíncrono
No importa
Modo síncrono - maestro
1 = Habilita recepción simple
0 = Deshabilita recepción simple
Este bit es limpiado después de que la recepción es completada.
Modo síncrono – esclavo
Sin uso en este modo

bit 4: **CREN:** Bit de habilitación para la recepción continua
Modo asíncrono
1 = Habilita recepción continua
0 = Deshabilita recepción continua
Modo síncrono
1 = Habilita recepción continua hasta que el bit CREN es limpiado (CREN overrides SREN)
0 = Deshabilita recepción continua

bit 3: **ADDEN**: Bit de habilitación para la detección de dirección
Modo asíncrono de 9 bits (RX9=1)
1 = Habilita la detección de dirección, habilita interrupción y carga del bufer de recepción cuando RSR<8> está encendido
0 = Deshabilita la detección de dirección, todos los bits son recibidos y el 9° puede usarse como bit de paridad

bit 2: **FERR:** Bit de error de framing
1 = Error de framing (Puede actualizarse leyendo el registro RCREG y recibir el próximo byte válido)
0 = Sin error de framing

bit 1: **OERR:** Bit de error de overrun
1 = Error de overrrun (Puede limpiarse al limpiar el bit CREN)
0 = Sin error de overrun

bit 0: **RX9D:** 9° bit de recepción. (Puede ser el bit de paridad)

	dirección de archivo						
dir. Indir.(*)	00h	dir. Indir.(*)	80h	dir. Indir.(*)	100h	dir. Indir.(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCLATH	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
	08h		88h		108h		188h
	09h		89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDAT	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reservado(1)	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reservado(1)	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h				
T2CON	12h	PR2	92h				
SSPBUF	13h	SSPADD	93h				
SSPCON	14h	SSPSTAT	94h				
CCPR1L	15h		95h				
PPCR1H	16h		96h				
CCP1CON	17h		97h				
RCSTA	18h	TXSTA	98h				
TXREG	19h	SPBRG	99h				
RCREG	1Ah		9Ah				
CCPR2L	1Bh		9Bh				
CCPR2H	1Ch		9Ch				
CCP2CON	1Dh		9Dh				
ADRESH	1Eh	ADRESL	9Eh				
ADCON0	1Fh	ADCON1	9Fh				
	20h		A0h		120h		A0h
Registros de propósito general 96 bytes	7Fh	Registros de propósito general 96 bytes	FFh	accesses 20h – 7Fh	16Fh 170h 17Fh	accesses A0h – FFh	1EFh 1F0h 1FFh
Banco 0		Banco 1		Banco 2		Banco 3	

Localidades de memoria no implementadas, se leen como ‘0’.

***** No es un registro físico

Nota 1: Registros reservados, siempre deben mantenerse en 00h.

Apéndice E: MPLAB IDE

Configuración del MPLAB IDE

1. Elegir el microcontrolador. Para ello, acceder al menú *Configure* → *Select Device* y seleccionar *PIC16F873*.

Ensamblado del programa

1. Para ello, acceder al menú *Project* → *Quickbuild *.asm*.

Programación del dispositivo

1. Seleccionar el programador. Para ello, acceder al menú *Programmer* → *Select Programmer* y seleccionar *PICSTART Plus.*

2. Habilitar el programador. Para ello, acceder al menú *Programmer* → *Enable Programmer*.

3. Iniciar la programación. Para ello, acceder al menú *Programmer* → *Program*.

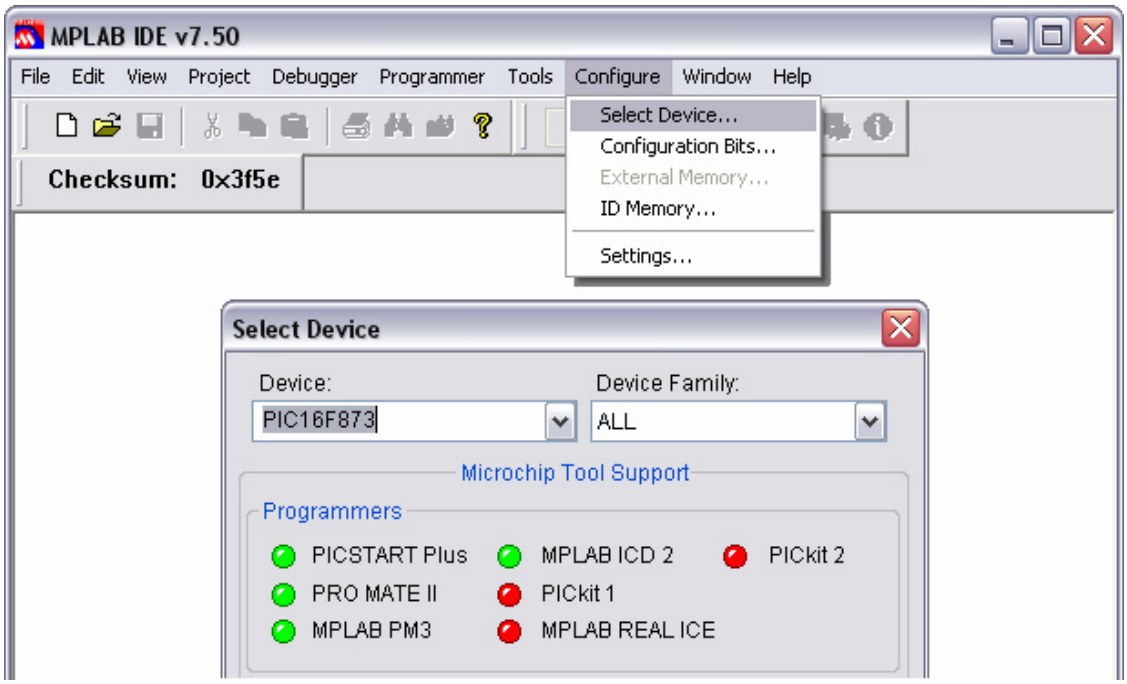


Figura E-1 Selección del microcontrolador

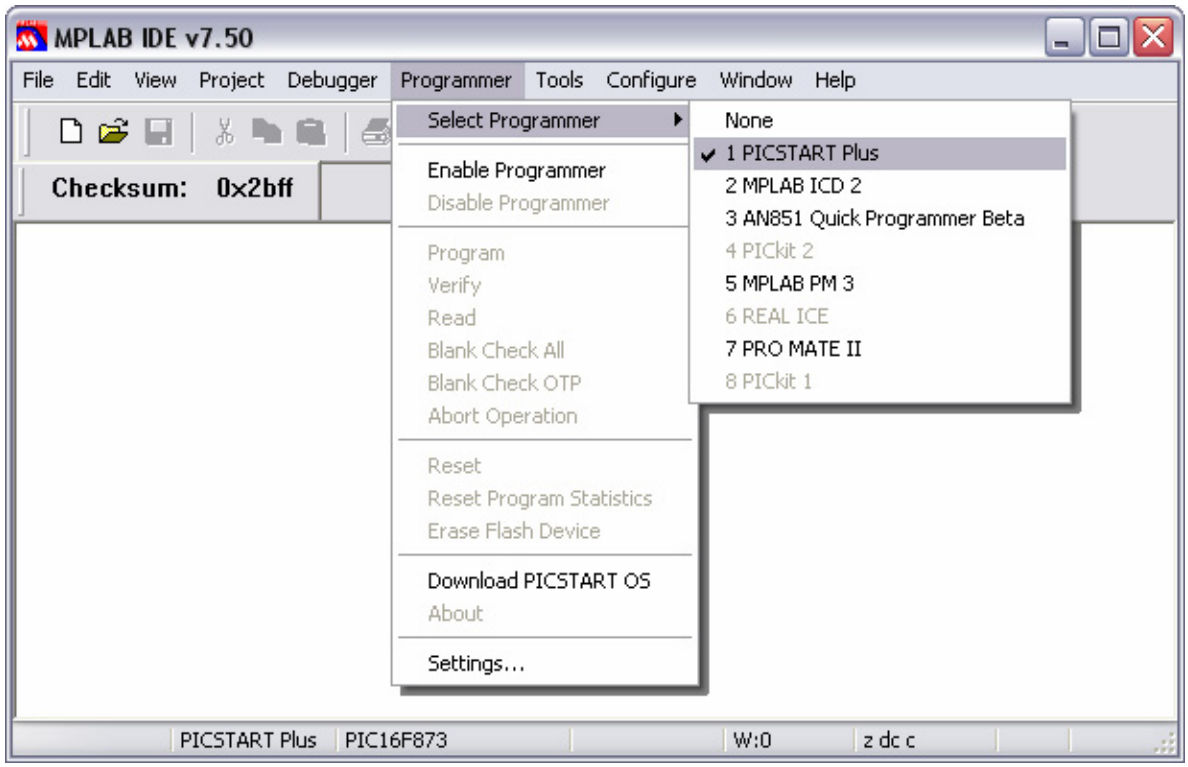


Figura E-2 Selección del programador

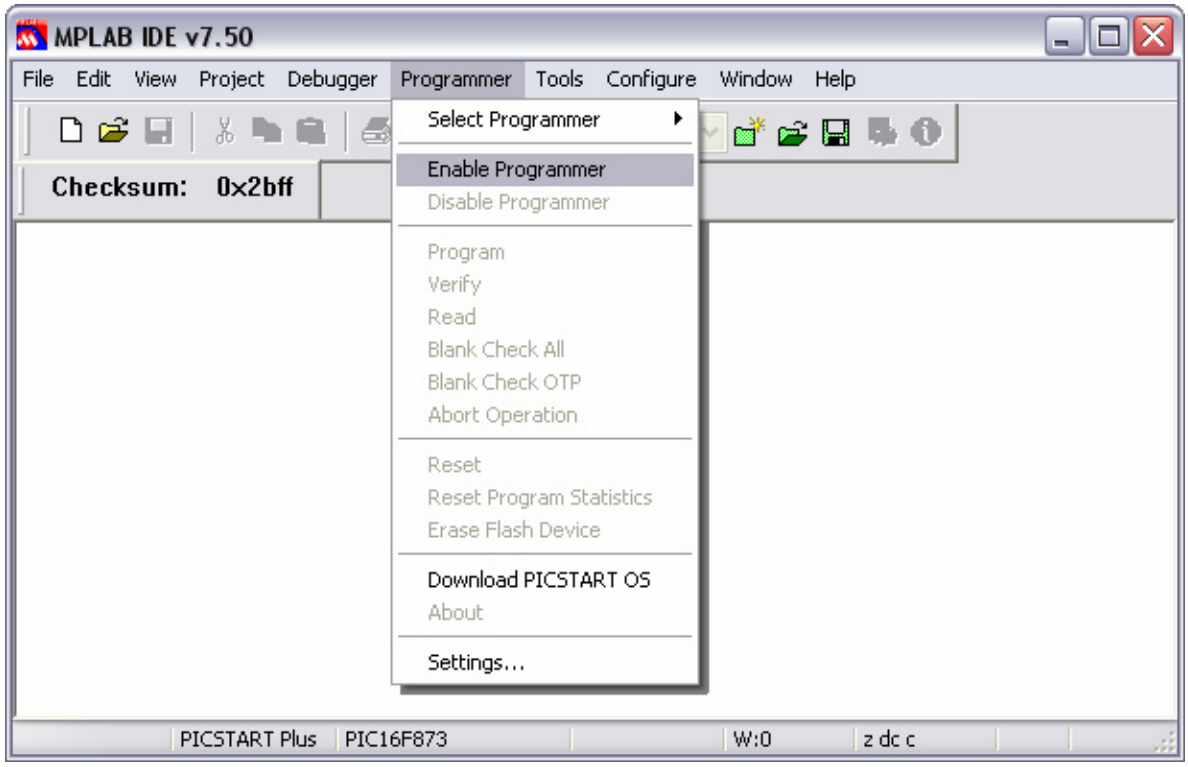


Figura E-3 Habilitación del programador

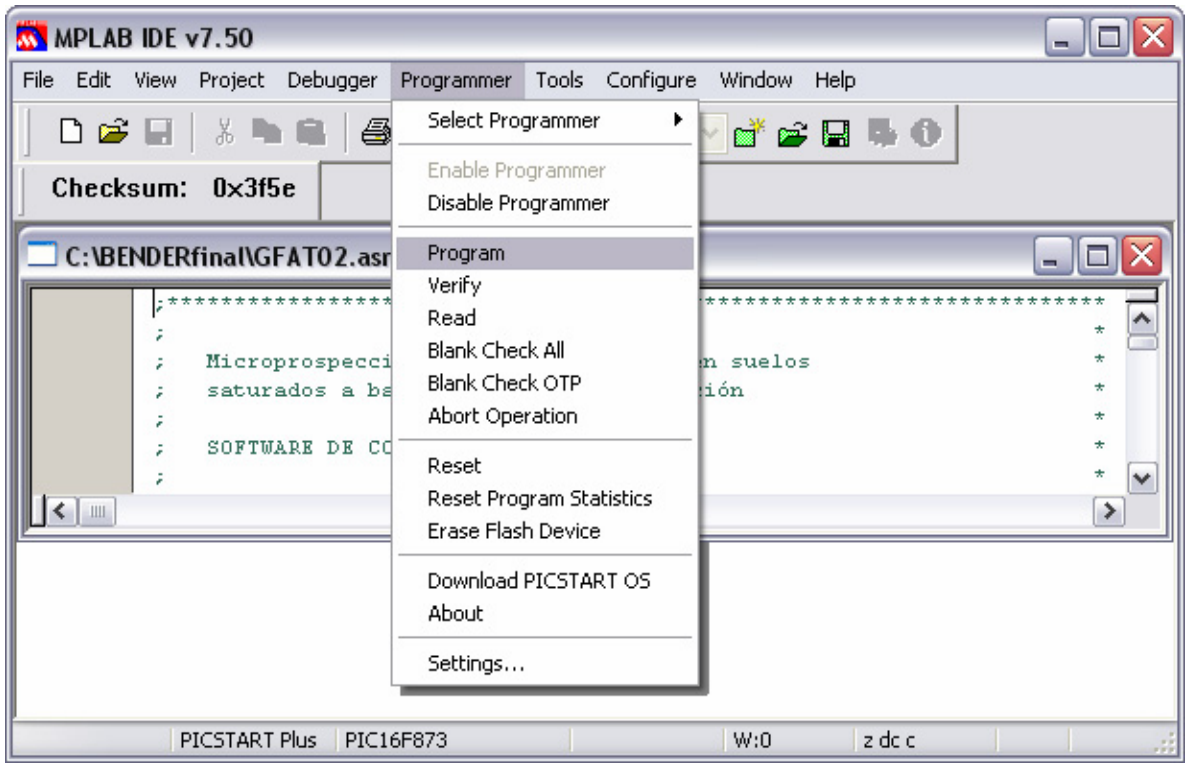


Figura E-4 Programación del microcontrolador

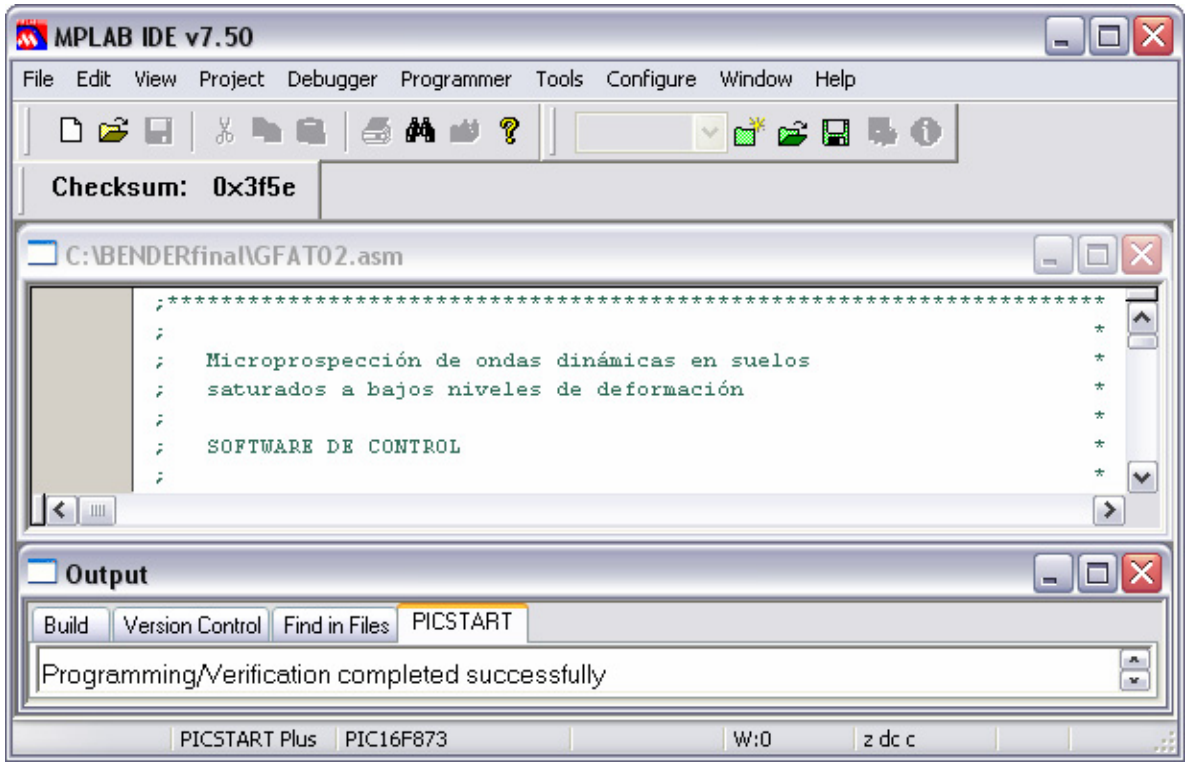


Figura E-5 Programación completada

Apéndice F: Código ASCII

El código ASCCII (*American Standard Code of Information Interchange*) es un sistema de representación utilizado en los sistemas digitales que utiliza un esquema de codificación que asigna valores numéricos a letras, números, signos de puntuación y a otros caracteres.

Códigos de Control				Caracteres Alfanuméricos								
Nombre	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
Null char	0	00	NUL	32	20	Espacio	64	40	@	96	60	`
Start of Header	1	01	SOH	33	21	!	65	41	A	97	61	a
Start of Text	2	02	STX	34	22	"	66	42	B	98	62	b
End of Text	3	03	ETX	35	23	#	67	43	C	99	63	c
End of Transmission	4	04	EOT	36	24	$	68	44	D	100	64	d
Enquiry	5	05	ENQ	37	25	%	69	45	E	101	65	e
Acknowledgment	6	06	ACK	38	26	&	70	46	F	102	66	f
Bell	7	07	BELL	39	27	'	71	47	G	103	67	g
Backspace	8	08	BS	40	28	(	72	48	H	104	68	h
Horizontal Tab	9	09	HT	41	29	)	73	49	I	105	69	i
Line Feed	10	0A	LF	42	2A	*	74	4A	J	106	6A	j
Vertical Tab	11	0B	VT	43	2B	+	75	4B	K	107	6B	k
Form Feed	12	0C	FF	44	2C	,	76	4C	L	108	6C	l
Carriage Return	13	0D	CR	45	2D	-	77	4D	M	109	6D	m
Shift Out	14	0E	SO	46	2E	.	78	4E	N	110	6E	n
Shift In	15	0F	SI	47	2F	/	79	4F	O	111	6F	o
Data Link Escape	16	10	DEL	48	30	0	80	50	P	112	70	p
Device Control 1	17	11	DC1	49	31	1	81	51	Q	113	71	q
Device Control 2	18	12	DC2	50	32	2	82	52	R	114	72	r
Device Control 3	19	13	DC3	51	33	3	83	53	S	115	73	s
Device Control 4	20	14	DC4	52	34	4	84	54	T	116	74	t
Negative Acknowledgement	21	15	NAK	53	35	5	85	55	U	117	75	u
Synchronous Idle	22	16	SYN	54	36	6	86	56	V	118	76	v
End of Trans. Block	23	17	ETB	55	37	7	87	57	W	119	77	w
Cancel	24	18	CAN	56	38	8	88	58	X	120	78	x
End of Medium	25	19	EM	57	39	9	89	59	Y	121	79	y
Substitute	26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
Escape	27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
File Separator	28	1C	FS	60	3C	<	92	5C	\	124	7C	|
Group Separator	29	1D	GS	61	3D	=	93	5D	]	125	7D	}
Record Separator	30	1E	RS	62	3E	>	94	5E	^	126	7E	~
Unit Separator	31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Tabla F-1 Código ASCII estándar

Apéndice G: Montaje de cristales

Con respecto al montaje de los cristales piezoeléctricos, se realizó en dos partes. En la primera parte se conectó un cable coaxial a cada cristal piezoeléctrico y se aisló con un recubrimiento de poliuretano resistente al agua. A continuación, se armó un juego de emisores (disco y elemento de flexión) y un juego de receptores (disco y elemento de flexión).

En la segunda parte, cada juego se fijó a una base con pegamento epóxico y se aplicó pintura conductiva para realizar la conexión a tierra. Finalmente, se añadió otra capa de poliuretano para garantizar que la base se encontrara aislada.

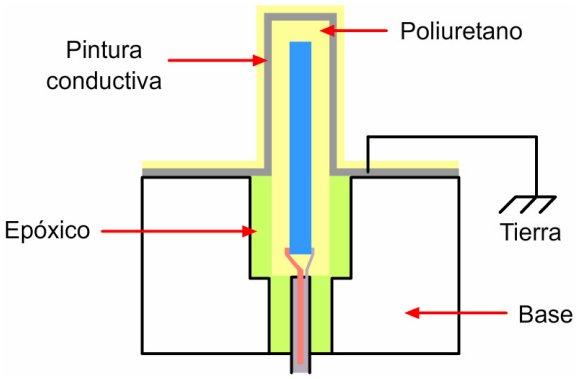


Figura G-1 Montaje de los cristales piezoeléctricos

Es importante señalar que durante el montaje de los cristales, las bases se manufacturaron en tres materiales diferentes: acrílico, nylamid y aluminio; siendo éste último el que permitió solucionar los problemas de fracturas y disipación de energía que se presentaban en los dos anteriores. Así mismo, para el caso de los discos piezoeléctricos fue necesario colocar el cable de forma perpendicular como se muestra en la *figura x-2*.

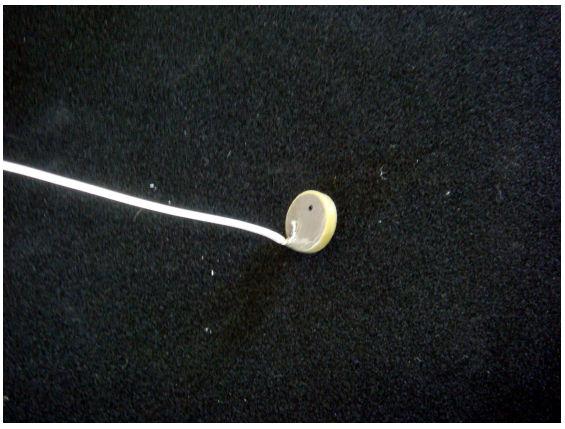


Figura G-2 Disco piezoeléctrico

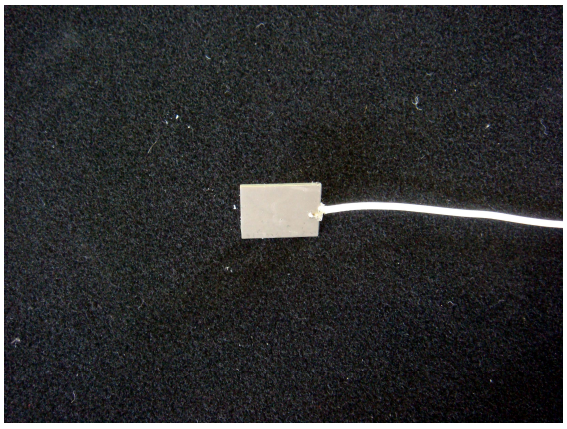


Figura G-3 Elemento de Flexión

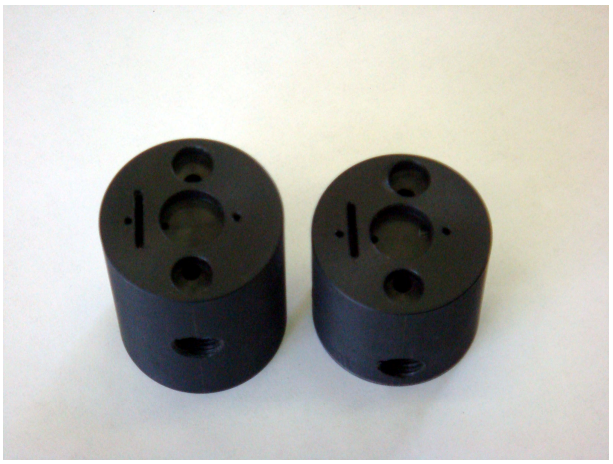


Figura G-4 Bases

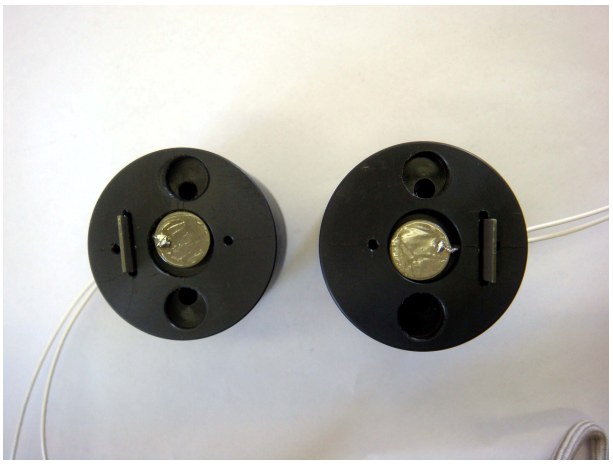


Figura G-5 Etapa 1 del montaje

Figura G-6 Etapa 2 del montaje

Referencias Bibliográficas

- Palacios, E., et. al.; **"Microcontrolador PIC16F84"**; Ed. Alfaomega; 2da ed.; México, 2006; pp. 77-93, 141-147, 157-160, 253-266
- Bishop R. H.; **"LabVIEW 8 Student Edition"**; Ed. Pearson Prentice Hall; 2da ed.; USA, 2007; pp.
- Carmona, I.; **"Ecuaciones Diferenciales"**; Ed. Addison Wesley Longman; 4ta ed.; México, 1998; pp. 463-467
- Garbutt, M.; **"AN774: Asynchronous Communications with the PICmicro® USART"**; USA, 2003
- **"7405 Data Sheet"**, Fairchild Semiconductor; USA, 2000.
- **"74HC86 Data Sheet"**, Hitachi Semiconductor; Japón, 1999
- **"74HC163 Data Sheet"**, Fairchild Semiconductor; USA, 1999
- **"DG200 Data Sheet"**, Maxim Integrated Products; USA, 1996
- **"MAX232 Data Sheet"**, Maxim Integrated Products; USA, 2000
- **"PA90 Data Sheet"**, Apex Microtechnology; USA, 2004
- **"PIC16F87X Data Sheet"**, Microchip Technology; USA, 2001
- **"TL08X Data Sheet"**, Motorola; USA,1997
- **"XR2206"**, EXAR Corporation; USA, 1997