UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERIA

Tarjeta debugger y hardware de aplicaciones para Microcontroladores PIC16F877/A.

TRABAJO DE GRADUACION PARA OPTAR AL TITULO DE

INGENIERO EN AUTOMATIZACION

PRESENTADO POR:

CHRISTIAM GEOVANI FLORES RAMIREZ

ASESOR:

ING. HECTOR RUBEN CARIAS JUAREZ

ENERO 2010.

EL SALVADOR, CENTRO AMERICA

UNIVERSIDAD DON BOSCO

RECTOR

ING. FEDERICO MIGUEL HUGUET RIVERA

SECRETARIA GENERAL

INGA. YESENIA XIOMARA MARTINEZ OVIEDO

DECANO DE LA FACULTAD DE INGENIERIA

ING. ERNESTO GODOFREDO GIRON

ASESOR DEL TRABAJO DE GRADUACION

ING. HECTOR RUBEN CARIAS JUAREZ

LECTOR

ING. NESTOR ROMAN LOZANO LEIVA

ADMINISTRADOR DEL PROCESO

ING. HERBER CARDONA

UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERIA

EVALUACION DEL TRABAJO DE GRADUACION

TARJETA DEBUGGER Y HARDWARE DE APLICACIONES PARA MICROCONTROLADORES PIC16F877/A.

ING. HECTOR CARIAS	**ING. NESTOR LOZANO**
ASESOR	LECTOR

ING. HERBER CARDONA

ADMINISTRADOR DEL PROCESO

AGRADECIMIENTOS

Gracias a Dios y a la virgen por permitirme obtener este logro, gracias padre por todos lo bellos y pequeños detalles que me ayudaron a construir esto.

Gracias a mi madre por la vida ,el amor y la paciencia con la que a compartido su vida conmigo, ella ha sido mi pilar fundamental a lo largo de estos años.

Gracias a mi padre por su apoyo incondicional,por sus consejos ,por su ejemplo de lucha y por inculcarme la idea de siempre seguir adelante.

A mis abuelas Angela y Marta por su cariño ,a mis tios, familia y amigos que a lo largo de estos años me han servido de apoyo y con quienes he compartido muchas estapas bonitas de mi vida.

A mis hermanos por su comprension y paciencia en muchos momentos en los cuales su apoyo fue fundamental.

A mis amigos Roberto, Roberto Carlos, Jonathan, Carlos, Armando,Francisco, monika y Gladys por su apoyo y compañerismo a lo largo de la carrera, momentos inolvidables ,recuerdos valiosos y amistades largas.

Personas que a lo largo de mi vida han dejado buenos recuerdos y enseñanzas .

Dedicado para Carlos Ramirez (QDDG) , se que le hubiera gustado estar aqui.

“Siempre es demasiado pronto para renunciar”

INDICE GENERAL

INTRODUCCIÓN

Debido a la necesidad de realizar tareas de una mejor manera y de utilizar menos recursos, la tecnología está sufriendo gran cantidad de cambios. Es así como podemos enfocarnos en los sistemas embedidos, los cuales poseen diferentes subsistemas en un solo encapsulado, siendo ampliamente utilizados en distintos campos como la industria automovilística ,la industria de los electrodomésticos, así como en la industria de la maquinas, etc. Para obtener el máximo provecho en el uso de estos sistemas se hace necesario tener conocimiento de herramientas que nos ayuden a la comprensión de su funcionamiento y así aplicarlos para dar solución a problemas.

En el mercado existen diferentes herramientas de las cuales valerse para entender de manera practica el funcionamiento especifico de los sistemas embedidos conocidos como microcontroladores, las herramientas más conocidas son los depuradores o tarjetas debugger, las cuales sirven para comprobar como se comporta el programa instalado en el microcontrolador, observando de manera real los cambios que sufren los registros internos del sistema. Estas tarjetas se complementan con equipo hardware para comprobar el funcionamiento de dicho programa utilizando indicadores de datos digitales, entradas digitales, análogas, salidas a dispositivos, etc.

Estos equipos para realizar debug están orientados específicamente a realizar su operación para un microcontrolador específico como los Motorola con la tarjeta HC12, Hitachi con su tarjeta debugger, nuestro enfoque estará en el desarrollo y uso de la herramienta para realizar depuración en los microcontroladores PIC[1] de gamma media 16F877 y 16F877A.

[1] **PIC** Controlador de interfaz de Periféricos.

ANTECEDENTES

En la introducción nos referimos de manera general a las herramientas con las cuales podemos valernos a la hora de tratar con microcontroladores ,sin embargo ahora nos enfocaremos específicamente en la materia de microcontroladores impartida en la Universidad Don Bosco. Los alumnos se apoyan en distintas herramientas para la realización de practicas en la materia de microcontroladores; por la razón de no contar con una herramienta especialmente enfocada para realizar las distintas practicas usando específicamente los PIC, dentro de éstas se encuentran los comúnmente llamados simuladores que muchas veces nos ayudan de gran manera pero no tiene las ventajas de una tarjeta debugger, esto se debe a que con estas tarjetas debugger se puede verificar de manera real los cambios que sufren los registros internos del microcontrolador usado, mientras que en los simuladores simplemente se obtiene una idea de como cambiarán los registros internos que contiene el microcontrolador.

El contenido de la materia de microcontroladores esta enfocada hacia el aprendizaje de los microcontroladores Motorola HC12 y PICs (específicamente los 16F877/A) .Estos microcontroladores poseen, en un solo encapsulado, una gran cantidad de subsistemas ampliamente utilizados en la electrónica moderna. Las herramientas con que se cuentan en el laboratorio son la tarjeta debugger HC12 y tarjeta para quemado de PICs, no se cuenta con una tarjeta específica para desarrollar depuración con los PICs y el alumno se ve forzado a desarrollar sus aplicaciones a prueba y error, sin mencionar que además debe contar con el hardware específico para su aplicación. Al contar con la tarjeta debugger y su hardware de aplicaciones podrá tenerse la ventaja de desarrollar practicas y de depurar éstas practicas en tiempo real, basándose en la tutoría de ejercicios que estarán incluidos en el manual de usuario creado y apoyándose además en el hardware existente.

OBJETIVOS

GENERAL

Desarrollar un sistema didáctico en el cual los diferentes usuarios puedan a través del sistema hardware y del software MPLAB-IDE realizar depuración y programación; además de poder realizar prácticas utilizando los diferentes ejemplos creados para comprender los distintos subsistemas básicos presentes en los microcontroladores PIC16F877 o PIC16F877A.

ESPECIFICOS

- Implementar el prototipo de un sistema modular que incluya las herramientas básicas para realizar debug en los PIC877/A, diseñando el hardware de aplicaciones y la tarjeta debug.

- Crear un sistema con hardware básico para la verificación del programa creado que se desea depurar o implementar, que incluya interruptores, LEDs visualizadores, salidas digitales y análogas, salidas de potencia y conexión a los tipos de comunicación soportadas por el microcontrolador.

- Implementar en el sistema un módulo para comunicación I²C[2], SPI[3] para el aprendizaje de la utilización de dichos protocolos, así como un módulo para el control de un motor paso y el control de relés.

- Elaborar un manual de usuario que incluya instrucciones para el uso del sistema, además de ejercicios explicados orientados al estudio y aprendizaje de los subsistemas del microcontrolador apoyándose en el hardware de aplicaciones.

[2] **I²C** Interface entre circuitos integrados.
[3] **SPI** Interface serial entre periféricos.

ALCANCES

- Realizar la depuración en tiempo real a través del sistema operativo del microcontrolador en conjunto con el software MPLAB. La utilización del software será estrictamente para el envío del programa hacia el microcontrolador y la visualización de los registros internos del PIC provenientes del sistema operativo.

- El hardware consta de entradas y salidas de datos digitales para manejo de puertos I/O y salida de potencia para manejo de dispositivos que demanden alta corriente según el límite establecido.

- Utilizando la entrada análoga se leen datos con un rango definido de voltaje y frecuencia ya sea de dispositivos externos o internos al sistema.

- El hardware es capaz de conectarse a dispositivos externos o internos como pueden ser PICs o Chips que soporten las comunicaciones SPI ó I²C.

- El sistema incluye salida análoga variable con un rango definido de Voltaje y de corriente.

- la tarjeta debug es capaz de reconocer diferentes tipos de PICs además de los utilizados en el sistema didáctico, esto queda delimitado por el software MPLAB. Además esta tarjeta puede ser utilizada como programador o como depurador.

- Utilizando la tutoría de ejercicios incluidos en el manual de usuario, el cual ha sido creado específicamente para este sistema, el alumno será capaz de comprender el uso de los subsistemas presentes en el PIC y de practicarlos en el hardware de aplicaciones.

LIMITACIONES

- El hardware de aplicaciones está desarrollado para la utilización de los Microcontroladores 16F877 ó 16F877A por lo que las bases en cuanto al diseño del hardware quedarán si se desea implementar tarjetas para otras series.

- La depuración es realizada de forma básica cargando el programa en la memoria del microcontrolador, corriendo el programa y visualizando los cambios de los registros internos en la PC y en el hardware de aplicaciones.

- El rango de entrada análogo es de 0 a 10Vdc utilizando la entrada externa y de 0 a 5Vdc utilizando la entrada interna.

- El rango de salida análogo es de 0-10Vdc con una corriente máxima de 1 Amperio.

- La frecuencia de entrada para la conversión A/D depende del tiempo en que se carga el capacitor Chold, este tiempo es de 19.72us, siendo la frecuencia máxima de 50KHz.

- Para la salida análoga el tiempo de conversión máximo es de 10us dada por el conversor D/A MCP4822 controlado por comunicación serial (SPI), ofreciendo una frecuencia de salida máxima de 100KHz.

- La tarjeta de potencia está dispuesta para manejar un máximo de 4 relés y un motor paso de manera individual, es decir solo puede estar funcionando ya sea el motor paso o solo los relés.

- La tarjeta I²C/SPI consta de lo básico para estudiar los protocolos I²C y SPI como PIC16F877A, memoria EEPROM I²C y conversor digital-análogo MCP4822.

Capitulo Uno: Microcontroladores PIC16F877 / A .

http://ww1.microchip.com/downloads/en/DeviceDoc/33023a.pdf
http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

1.1 INTRODUCCION A LOS CONTROLADORES Y MICROCONTROLADORES.

Un controlador es un dispositivo utilizado para manejar uno o varios procesos. Al principio estos dispositivos estaban formados exclusivamente por sistemas discretos, más tarde se emplearon procesadores rodeados de memorias y circuitos de entrada /salida incluidos en una sola placa impresa, actualmente los controladores integran todos los dispositivos en un solo chip. Las partes principales de un microcontrolador son: CPU[4] o procesador encargado de ejecutar las ordenes , líneas de E/S para comunicarse con los periféricos, memorias RAM conteniendo los datos que se intercambian con los buses de E/S , memoria tipo ROM/PROM/EEPROM que mantiene el código programado ,módulos para el control de periféricos (puerto serie y paralelo, conversor analógico/digital,etc)incluyendo generador de impulsos de reloj que sincronizan el funcionamiento de todo el sistema definiendo la velocidad de proceso del controlador**.** Cada fabricante dispone de un a gran cantidad de modelos que van desde los más sencillos a los más complejos, es posible con esto elegir la capacidad de las memorias, numero de líneas de E/S, velocidad de funcionamiento y más. La clasificación más importante es entre los microcontroladores de 4, 8, 16 y 32 bits , siendo los MCU[5] de 16 y 32 bits los que ofrecen más ventajas ,pero los más utilizados en el mercado son los de 8 bits por ser muy adecuados para la mayoría de aplicaciones.

1.2 CARACTERISTICAS DE LOS MICROCONTROLADORES PIC 16F877 /A.

Los microcontroladores PIC16F877 y 16F877A son de la familia Microchip de 40 pines, son MCUs de medio rango que poseen múltiples subsistemas en un solo chip.

---

[4] **CPU** unidad central de procesos.
[5] **MCU** micro unidad de control.

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

Tabla.1.1 *Códigos de acceso a cada banco*.

Cada banco posee hasta 128 bytes, las localidades más bajas están reservadas para los registros de funciones especiales, arriba de estos registros se sitúan los registros de propósito general. Los bancos y sus direcciones son mostrados en la figura 1.1.
Los registros de propósito general (registros creados por el usuario *Ej. Contador, dato1*), pueden ser accesados de manera directa utilizando sus nombres específicos o de manera indirecta, colocando en el registro FSR la dirección de estos registros .Para detalles de utilización del registro FSR refiérase a la sección 1.3.5 del capitulo 1.

definicion de registros de propósito general en un programa.

```
Contador   equ  0x40
Datol      equ  0x42
```

nombre del registro. **direccion del registro.**

Fig. 1.2 *Ejemplo de Registros de propósito general*.

Los registros de funciones especiales son registros usados por la CPU y módulos periféricos para controlar las operaciones deseadas en el microcontrolador, estos registros son implementados como memoria RAM estática (su valor cambia cada vez que existe un reset).

registros de funciones especiales.

Nombre	Direccion
ADCON1	(9Fh)
PIR1	(0Ch)
PORTA	(05h)

Fig.1.3 *Ejemplos de registros de funciones especiales.*

1.3.3 CONTADOR DE PROGRAMA (PCL) Y PCLATH.

El contador de programa PC tiene un campo de 13 bits, el byte menos significativo (0:7) es obtenido del registro PCL, el cual es para lectura y escritura; Los bits restantes (8:12) no pueden leerse, pero pueden cambiarse indirectamente utilizando el registro PCLATH. En cualquier operación de reset los bits (8:12) serán configurados como"0", los diagramas mostrados en la figura 1.2 muestran las distintas situaciones para cargar el contador de programa.

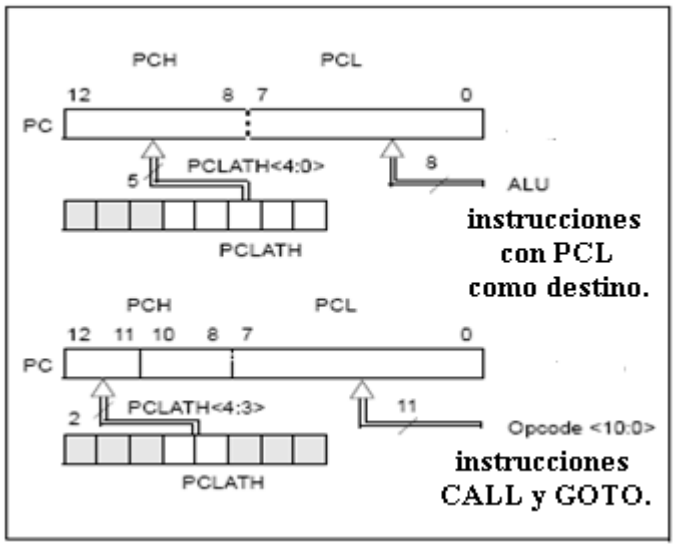


Fig.1.4 *Como cargar contador de programa.*

El primer diagrama muestra cuando se realiza una carga al PC por medio de PCLATH, el segundo diagrama indica cuando se carga el contador en una operación de salto de instrucción GOTO (el cual le agrega un offset al contador de programa) o una instrucción CALL detallada a continuación.Los PICs 16F877/A tiene una pila hardware con espacio

para 8 niveles de 13 bits cada uno, la pila no forma parte del espacio de memoria ni de datos, por lo que no se puede leer ni escribir en ella, el contador de programa es colocado en la pila cuando una instrucción CALL (llamada)es ejecutada o una interrupción causa un salto a otra dirección; la pila es vaciada al ejecutar las instrucciones RETURN,RETLW o RETFIE ,debe tenerse en cuenta que el registro PCLATH no es afectado ni al meter ni sacar datos de la pila (operaciones PUSH o POP).

La pila opera como un buffer circular, esto quiere decir que cuando la pila ha recibido 8 datos al ingresar un noveno dato este ocupa el lugar del dato guardado en la primera posición ,sobrescribiendo esa posición , así el décimo byte sobrescribe el segundo byte y así sigue esa secuencia. No existe ningún bit que indique cuando la pila ha sido sobrepasada, debe tenerse en mente que solo puede mantener 8 datos.

1.3.4 PAGINACIÓN DE MEMORIA DE PROGRAMA.

Los dispositivos 16F877/A son capaces de direccionar un bloque continuo de memoria de programa de 8K, las instrucciones GOTO o CALL proveen solo 11 bits de dirección para permitir saltar dentro de cualquiera de los bancos que posee el microcontrolador (cada banco posee 2K de memoria),por lo que los 2 bits restantes son adquiridos del registro PCLATH <4:3>,al realizar una instrucción GOTO o CALL el usuario debe asegurarse que los bits de selección de pagina estén bien configurados al banco al cual deseamos direccionar, si una instrucción de retorno o interrupción normal se ejecuta después de una llamada (CALL),el contador de programa completo(13 bits) son recuperados desde la pila para seguir en la instrucción que continua al salto hecho. El contenido del registro PCLATH no se ve afectado después de realizar un retorno con RETURN o RETFIE.

1.3.5 DIRECCIONAMIENTO DIRECTO /INDIRECTO.

Para el direccionamiento directo es necesario hacer referencia a la dirección a la cual se desea realizar una operación ya sea de lectura o escritura, teniendo en cuenta las configuraciones necesarias de bancos y espacio de memoria.

En el direccionamiento indirecto no se hace referencia directa a la dirección que se desea accesar, aquí se utilizan los registros FSR e INDF (este último no es un registro físico sino una localidad de memoria que posee el contenido de la dirección que se desea accesar). Para accesar de manera indirecta a una dirección de un banco determinado el valor de la dirección se guarda en el registro FSR y el contenido de esa dirección estará guardada en el registro INDF. Solo el registro FSR puede ser escritotomando un valor determinado, si se desea realizar un direccionamiento con 9 bits la dirección es obtenida concatenado el registro FSR con el bit IRP (STATUS<7>) como se muestra en el diagrama de la figura 1.5.

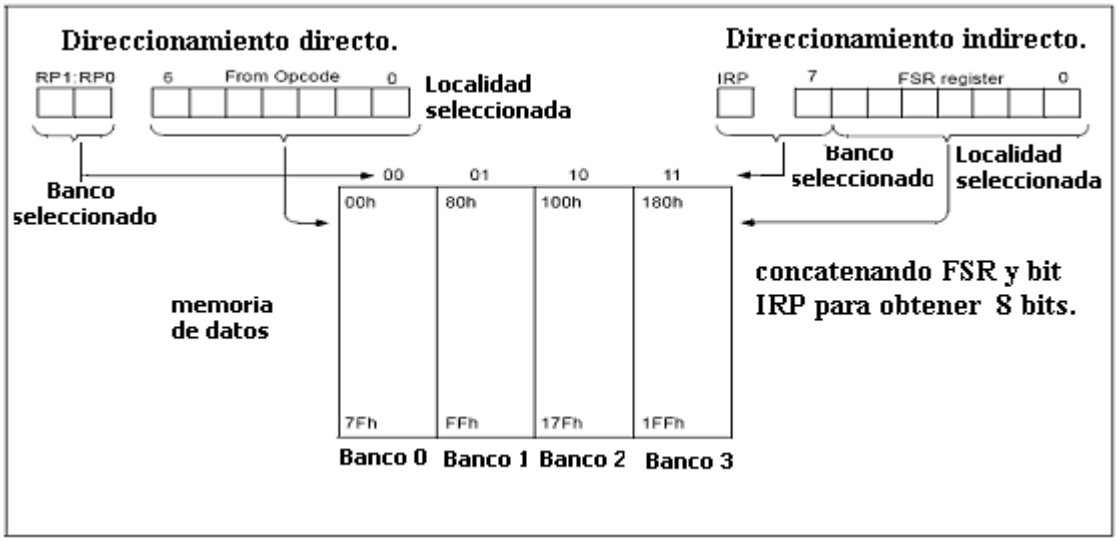


Fig.1.5 *Direccionamiento directo e indirecto*.

1.4 MÓDULO ENTRADA/SALIDA DIGITAL.

Los microcontroladores PIC 16F877/A posee 5 puertos de E/S digital de los cuales dos de estos puertos pueden ser multiplexados para actuar como entradas análogas.

Los puertos que contiene el microcontrolador 16F877/A son:

- Puerto A (E/S digital, entrada análoga, 6 pines RA0 – RA5).
- Puerto B (E/S digital, 8 pines RB0 –RB7).
- Puerto C (E/S digital, 8 pines RC0 –RC7).
- Puerto D (E/S digital ,8 pines RD0 –RD7).
- Puerto E (E/S digital, entrada análoga ,3 pines RE0 – RE2).

1.4.1 PUERTO A

El puerto A contiene 6 pines que pueden ser bidireccionales si están configurados para E/S digital a través de su registro de configuración TRISA; cada pin puede ser activado ya sea como entrada ("1" en cada bit) o salida digital ("0" en cada bit).

Con una lectura al registro PORTA se lee el estado de los pines configurados como entradas, una escritura a este registro cambiará el estado de los pines configurados como salidas.

El bit RA4 está multiplexado con la entrada de reloj T0CK1 del modulo TIMER0.Cabe mencionar que todos los pines del PUERTOA tienen niveles de entrada TTL y las salidas son tipo CMOS.El pin RA5 esta multiplexado con la entrada de selector de esclavo (SS), este pin es utilizado cuando se activa el modulo para comunicación SPI en modo esclavo (sección 1.7.2.2).

El registro ADCON1 se utiliza para configurar el PUERTOA ya sea como E/S digital o como entrada análoga. Refiérase a la figura 1.4.3 de la sección 1.4 del capitulo 1 para verificar detalles de configuración de bits.

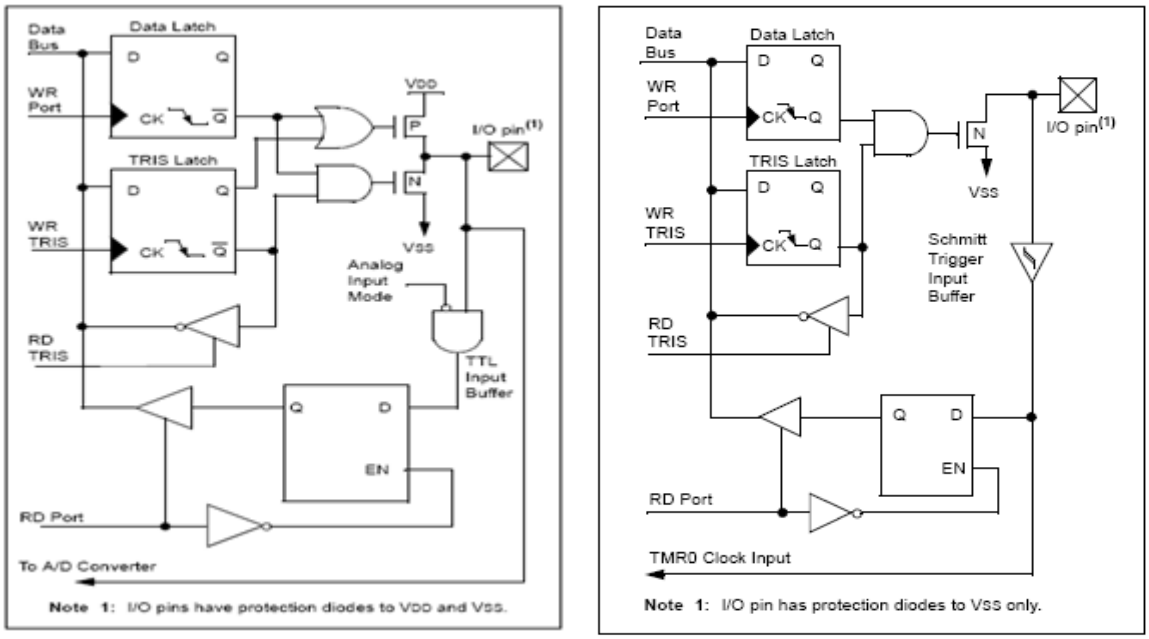


Fig.1.3.1a *Pines RA0-RAJHq5jTN:q=LS:LN5IT2:H=LNR□ILLNx:HNRM5tT2:JJ=J=q5RTBqv:2vv==*

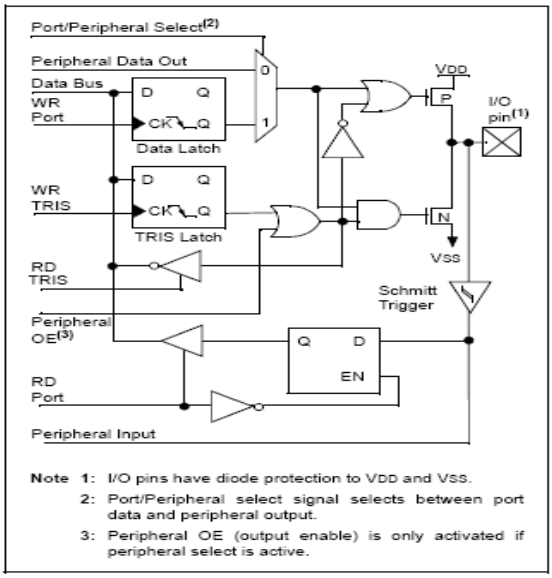


Fig.1.3.3a *Pines RC0:RC2 ,RC5:RC7*.

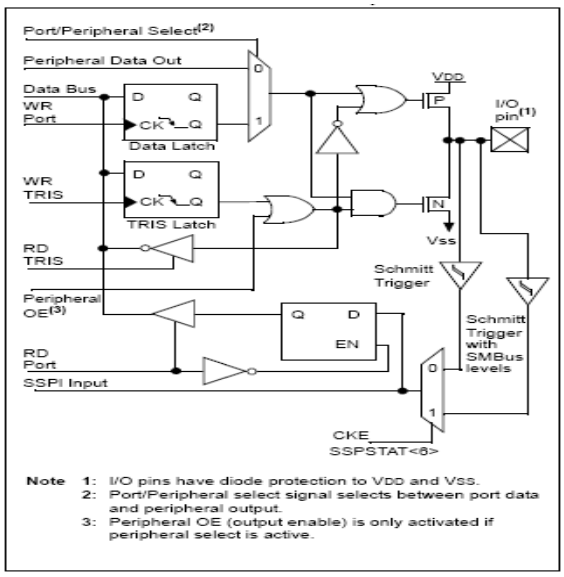


Fig.1.3.3b *Pines RC3,RC4.*

1.4.4 PUERTO D.

El puerto D contiene 8 pines con buffers del tipo Schmitt Trigger , los cuales pueden ser dibireccionales,cada pin individualmente puede ser activado como entrada o salida a través del registro TRISD,este puerto puede ser configurado para ser utilizado como puerto paralelo esclavo (PSP) que ofrece la ventaja de ser utilizado para comunicación con microprocesadores utilizando el protocolo paralelo (sección 1.3.6).

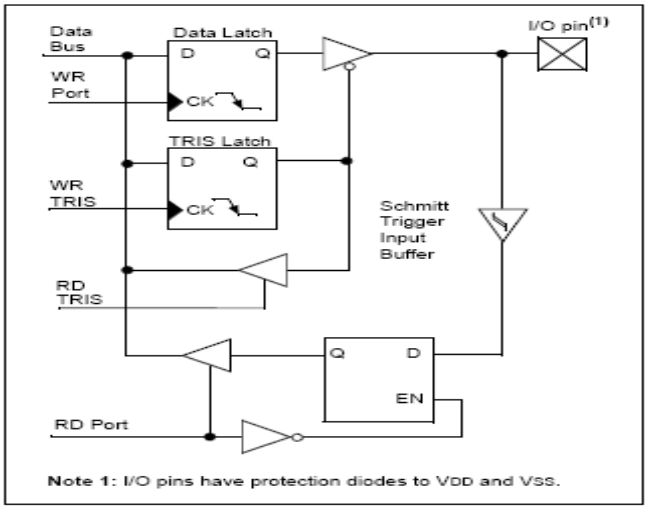


Fig.1.3.4 *Diagrama de bloques puertoD*.

1.4.5 PUERTO E.

El puerto E tiene 3 pines (RE0:RE2), los cuales son configurables individualmente usando el registro TRISE, estos pines poseen buffer de entrada Schmitt Trigger ; el PUERTOE es utilizado para controlar el PUERTOD (en modo PSP)activando el bit PSPMODE del registro TRISE<4>.el PUERTOE puede usarse ademas como entrada análoga al igual que el PUERTOA.

Nombre	Bit	Buffer	Función
RE0/RD/AN5	Bit 0	ST	Pin de E/S,pin para lectura en módulo esclavo paralelo activo en cero, entrada análoga AN5.
RE1/WR/AN6	Bit 1	ST	Pin de E/S,pin para escritura en módulo esclavo paralelo activo en cero, entrada análoga AN6.
RE2/CS/AN7	Bit 2	ST	Pin de E/S,pin para selección de dispositivo en módulo esclavo paralelo activo en cero, entrada análoga AN7.

Tabla 1.3.2.*Funciones de los pines del puerto E.*

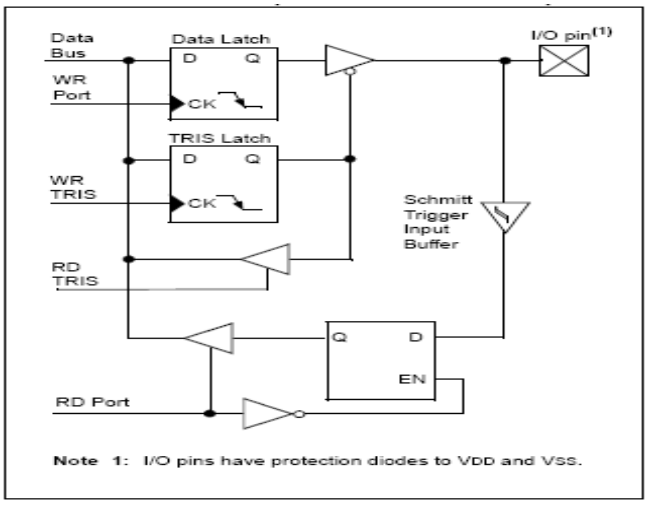


Fig.1.3.5.*Diagrama de bloques puerto E.*

1.4.6 PUERTO ESCLAVO PARALELO PSP.

En el modo esclavo paralelo PSP o puerto de microprocesador ,se utilizan los pines del puerto D y puerto E para establecer conexión entre el dispositivo y el bus de 8 bits de un microprocesador.El puerto paralelo se activa utilizando el bit PSPMODE (TRISE<4> figura 1.3.7).En modo PSP la escritura y lectura de datos es controlada asincronamente por los pines del puerto E. El diagrama de bloques del puerto PSP se muestra en la figura 1.3.6.

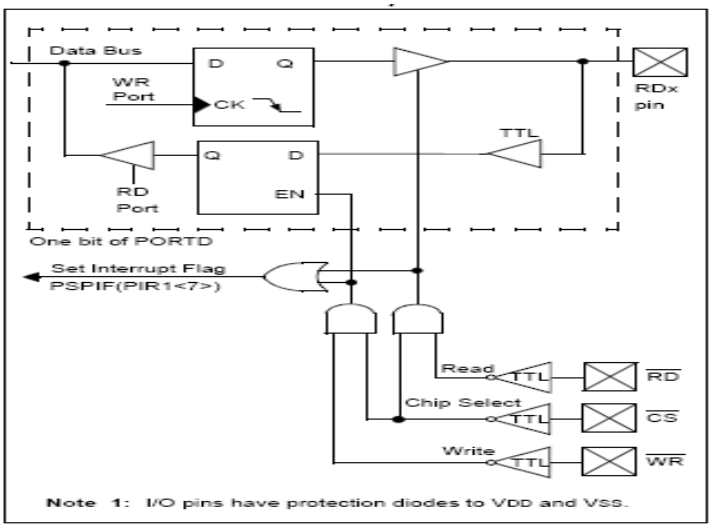


Fig.1.3.6.*Diagrama de bloques puerto esclavo PSP*.

R-0	R-0	R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1
IBF	OBF	IBOV	PSPMODE	—	Bit2	Bit1	Bit0
bit 7							bit 0

Parallel Slave Port Status/Control Bits:

IBF: Input Buffer Full Status bit

1 = A word has been received and is waiting to be read by the CPU
0 = No word has been received

OBF: Output Buffer Full Status bit

1 = The output buffer still holds a previously written word
0 = The output buffer has been read

IBOV: Input Buffer Overflow Detect bit (in Microprocessor mode)

1 = A write occurred when a previously input word has not been read (must be cleared in software)
0 = No overflow occurred

PSPMODE: Parallel Slave Port Mode Select bit

1 = PORTD functions in Parallel Slave Port mode
0 = PORTD functions in general purpose I/O mode

Unimplemented: Read as '0'

PORTE Data Direction Bits:

Bit2: Direction Control bit for pin RE2/$\overline{CS}$/AN7

1 = Input
0 = Output

Bit1: Direction Control bit for pin RE1/$\overline{WR}$/AN6

1 = Input
0 = Output

Bit0: Direction Control bit for pin RE0/$\overline{RD}$/AN5

1 = Input
0 = Output

Fig.1.3.7.*Registro TRISE*.

para la escritura de un dato de 8 bits hacia el puerto paralelo se utilizan los pines CS y WR los cuales son activados con nivel bajo, al poner estos pines a nivel bajo se captura el dato en los pines del puerto D ,cuando ambos pines estan en alto la bandera de estado de buffer lleno IBF es activada indicando que la escritura esta hecha,además el bit PSPIF es activado, el cual puede generar una interrupción ;para limpiar la bandera IBF solo es necesario una lectura al puertoD. Cuando otro dato ha sido escrito al puerto antes de que el anterior es leido se activa el bit de sobreflujo IBOV (TRISE<5>) el cual es limpiado por software.

Para una lectura de datos desde el puerto PSP se utilizan los pines CS y RD ,se escriben los datos en el puerto D y se pueden leer desde el dispositivo cuando CS y RD estan en bajo,el bit de buffer de salidad OBF sepone a “0” indicando que el puerto D esta esperando por una lectura por un bus externo ,al cambiar los bits CS y RD desde bajo a alto el bit PSPIF (PIR1<7>)se activa generando una interrupcion (si se desea),este bit se limpia por software,el bit OBF permanece en bajo hasta que se de otra lectura desde el puertoD.

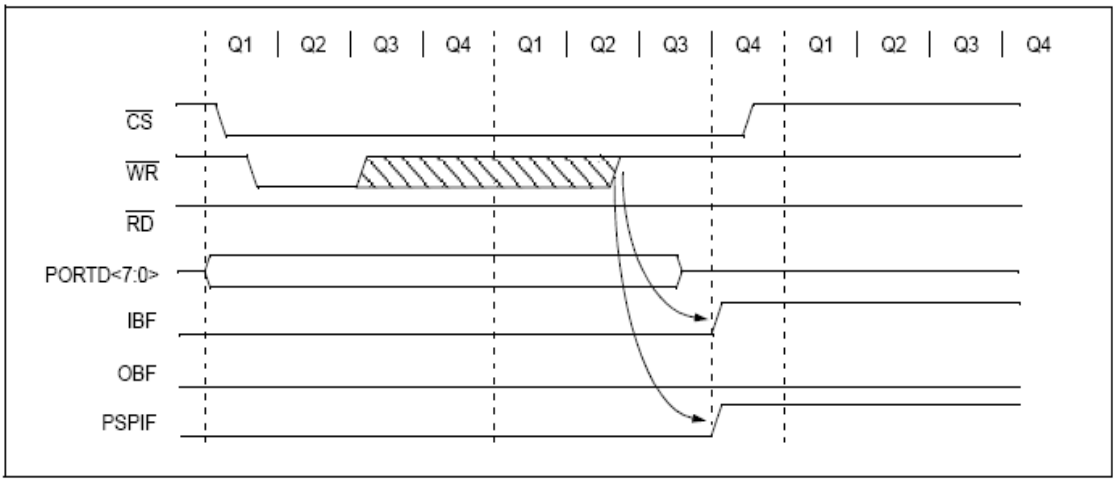


Fig.1.3.8.*Escritura de datos modo PSP*.

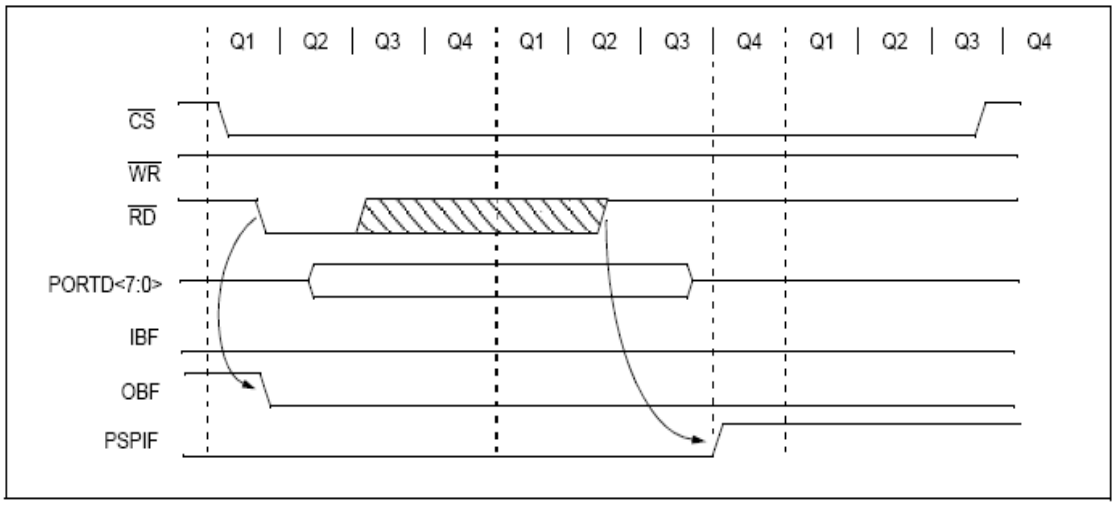


Fig. 1.3.9.*Lectura de datos modo PSP.*

1.5 MÓDULO CONVERSOR ANALÓGICO-DIGITAL A/D.

Encargado de la realizar las conversiones análogas /digitales utilizando el PUERTOA y ofreciendo un valor digital de 10 bits, la generación se realiza por aproximaciones sucesivas .El módulo A/D consta de un rango de voltaje de entrada de referencia que se puede seleccionar mediante software, los registros que intervienen en el uso de este modulo son:

- Registro de control 0 ADCON0.
- Registro de control 1 ADCON1.
- Registro de resultado alto ADRESH.
- Registro de resultado bajo ADRESL.

El registro ADCON0 mostrado en la figura 1.4.2 es el encargado de controlar la operación del módulo A/D, en este registro se encuentra el bit de inicio de conversión, se puede escoger el reloj de conversión y determinar cual entrada análoga será la que se utilizará. El registro ADCON1 de la figura 1.4.3 configura los pines ya sea como entradas análogas o digitales y determina el rango de voltaje de referencia. Si los pines se configuran como entradas análogas cualquier lectura a estos pines resultará en la lectura de ceros en todos los pines del PUERTOA.

Los registros ADRESH y ADRESL contienen los 10 bits que resultan de la conversión, antes de iniciar el proceso de conversión debe de activarse a través del registro TRISA y del registro ADCON1 que entrada análoga será leída. Debido a que se cuenta con 16 bits y solo se utilizan 10 bits, el módulo A/D ofrece la flexibilidad para justificar el resultado ya sea a lado izquierdo o derecho, el diagrama se muestra en la fig.1.4.1 , siendo el bit ADFM el encargado de realizar la justificación , los bits restantes son leídos como ceros.

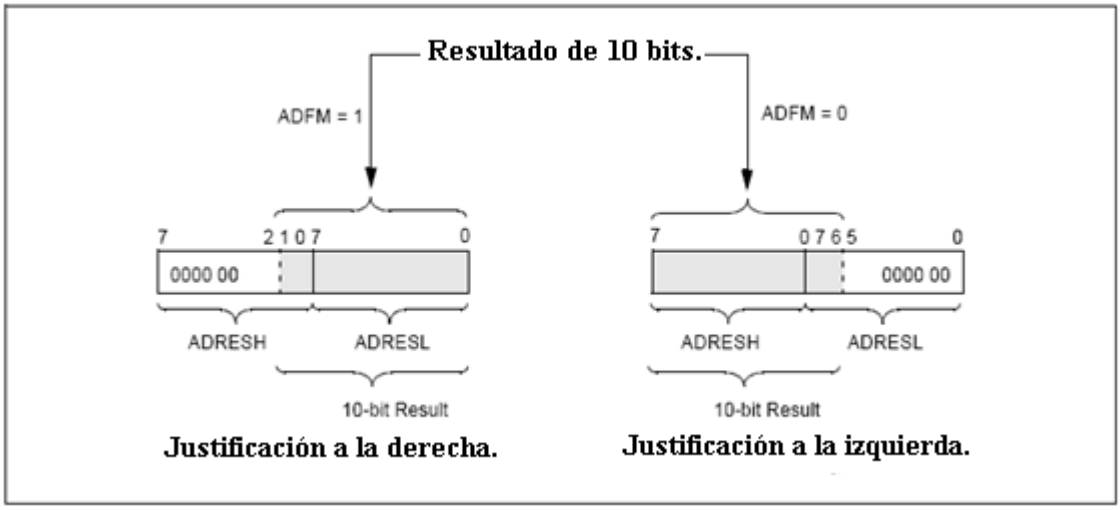


Fig.1.4.1.*Justificación del resultado obtenido por el convertidor*.

La conversión se inicia activando el bit GO/DONE, cuando la conversión es completada automáticamente los registros ADRESH y ADRESL contienen el valor convertido. Cuando la conversión está completa automáticamente se desactiva el bit GO/DONE (ADCON0<2>) y se activa el bit de interrupción ADIF (si el bit de interrupciones globales fue activado).

ADCON0 REGISTER (ADDRESS 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

ADCS1:ADCS0: A/D Conversion Clock Select bits (ADCON0 bits in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

CHS2:CHS0: Analog Channel Select bits

000 = Channel 0 (AN0)
001 = Channel 1 (AN1)
010 = Channel 2 (AN2)
011 = Channel 3 (AN3)
100 = Channel 4 (AN4)
101 = Channel 5 (AN5)
110 = Channel 6 (AN6)
111 = Channel 7 (AN7)

Note: The PIC16F873A/876A devices only implement A/D channels 0 through 4; the unimplemented selections are reserved. Do not select any unimplemented channels with these devices.

GO/$\overline{\text{DONE}}$: A/D Conversion Status bit

When ADON = 1:

1 = A/D conversion in progress (setting this bit starts the A/D conversion which is automatically cleared by hardware when the A/D conversion is complete)
0 = A/D conversion not in progress

Unimplemented: Read as '0'

ADON: A/D On bit

1 = A/D converter module is powered up
0 = A/D converter module is shut-off and consumes no operating current

Fig.1.4.2.*Registro ADCON0*.

ADCON1 REGISTER (ADDRESS 9Fh)

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

ADFM: A/D Result Format Select bit

1 = Right justified. Six (6) Most Significant bits of ADRESH are read as '0'.
0 = Left justified. Six (6) Least Significant bits of ADRESL are read as '0'.

ADCS2: A/D Conversion Clock Select bit (ADCON1 bits in shaded area and in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

Unimplemented: Read as '0'

PCFG3:PCFG0: A/D Port Configuration Control bits

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C/R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2/1
011x	D	D	D	D	D	D	D	D	—	—	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1/2

A = Analog input D = Digital I/O
C/R = # of analog input channels/# of A/D voltage references

Fig.1.4.3.*Registro ADCON1*.

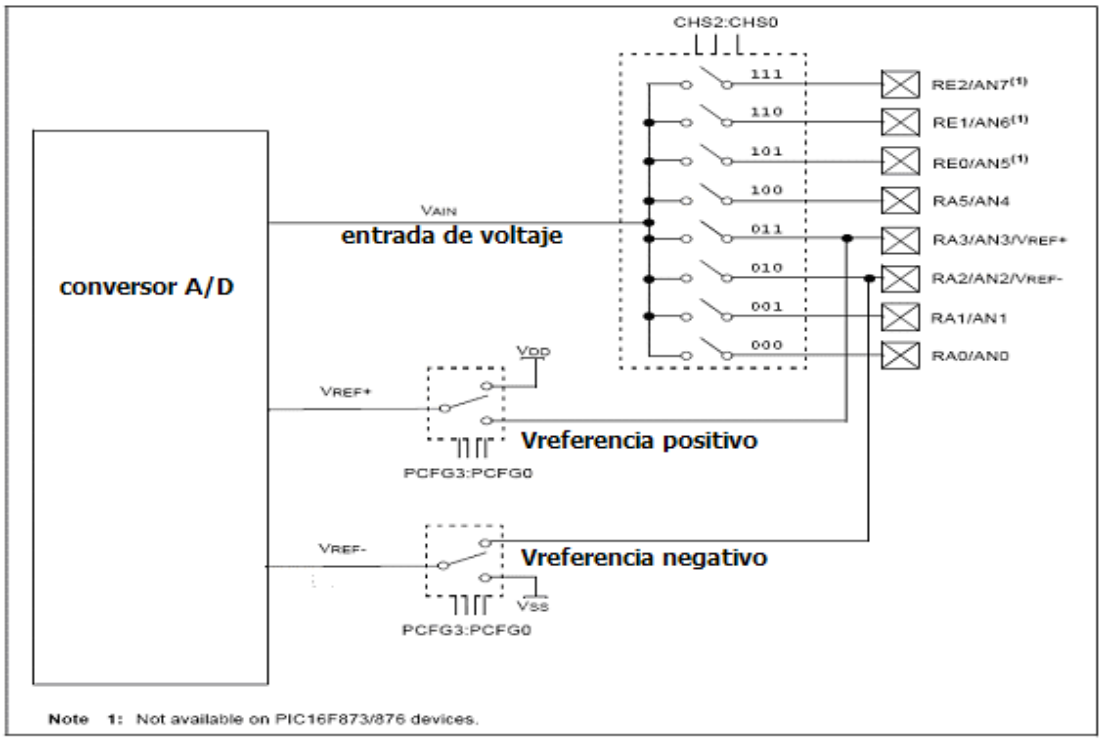


Fig.1.4.4.*Diagrama de bloques conversor A/D*.

El tiempo de adquisición esta determinado por el tiempo que se necesita para cargar el condensador de mantenimiento (CHold); para calcular el mínimo tiempo de adquisición se utiliza la ecuación de la fig.1.4.5 ,aquí se asume un error de 0.5LSB (son 1024 pasos para el convertidor) que es el máximo error permitido para conocer la resolución real. con esto se obtiene la frecuencia máxima de entrada que es de *Fmax=1/Tacq=50KHz.*

TACQ = Amplifier Settling Time + Hold Capacitor Charging Time + Temperature Coefficient

TACQ=Tiempo amplificado+tiempo de carga del capacitor+coeficiente de Temp.

= 2 µs + TC + [(Temperature – 25°C)(0.05 µs/°C)]

TC = CHOLD (RIC + RSS + RS) In(1/2047)

= - 120 pF (1 kΩ + 7 kΩ + 10 kΩ) In(0.0004885)

= 16.47 µs

TACQ = 2 µs + 16.47 µs + [(50°C – 25°C)(0.05 µs/°C)

= 19.72 µs

Fig.1.4.5.*Formulas para calcular tiempo de adquisición*.

El reloj de conversión se establece por medio de software, se necesita un tiempo de 12Tad mostrado en la figura 1.4.6 para realizar la conversión a 10 bits.

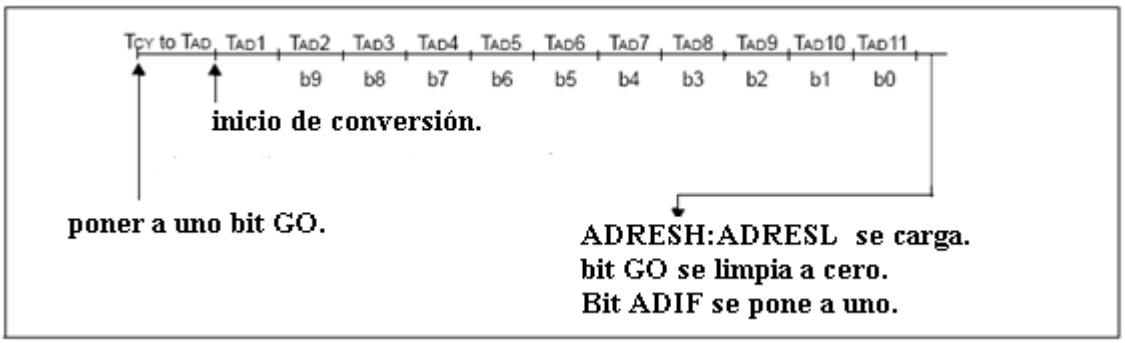


Fig.1.4.6.*Tiempo total para realizar la conversión*.

Para generar una correcta conversión, debe de asegurarse un tiempo Tad mínimo de 1.6us.La tabla 1.4.1 muestra los valores Tad derivados de la frecuencia de operación del dispositivo y de las diferentes fuentes de reloj que se pueden utilizar.

Fuente de reloj del conversor A/D (TAD) .		frecuencia maxima
Operación	ADCS2:ADCS1:ADCS0	
2 Tosc	000	1.25 MHz
4 Tosc	100	2.5 MHz
8 Tosc	001	5 MHz
16 Tosc	101	10 MHz
32 Tosc	010	20 MHz
64 Tosc	110	20 MHz
RC(1, 2, 3)	x11	———

Tosc=Tiempo de oscilación.

Tabla.1.4.1.*Fuentes de reloj (Tad).*

Al limpiar el bit GO/DONE durante una conversión esto hará que se aborte esta conversión y el valor de los registros ADRESH: ADRESL guardará el valor de la última conversión completa. Debe de tomarse en cuenta que el bit de inicio de conversión no debe ser activado en el mismo instante que se activa el módulo conversor.

El módulo A/D puede operar aún cuando el microcontrolador esté en modo sleep (dormido), esto se logra seleccionando como reloj de conversión el oscilador RC interno del dispositivo (ADSC1:ADCS0 =11) ,utilizando este oscilador se espera un ciclo de instrucción antes de iniciar la conversión permitiendo que se realice la instrucción de sleep , luego se realizará una secuencia de conversión normal, al finalizar la conversión si están habilitadas las interrupciones el dispositivo se despertará.
Si están desactivadas las interrupciones el módulo A/D se apagará aunque el bit ADON (A/D encendido) este a “1”. Sino se utiliza un reloj RC al ejecutarse la instrucción sleep se abortará la operación de conversión que se este ejecutando y se apagará el subsistema A/D aunque el bit ADON este a “1”.
Un reset al dispositivo provocará que los registros del conversor A/D sean configurados a sus valores predeterminados, ocasionando que se aborte cualquier conversión y se apague el módulo A/D .

1.6 MEMORIA DE DATOS EEPROM Y MEMORIA DE PROGRAMA FLASH.

Ambas memorias (EEPROM y FLASH) ofrecen la ventaja de escribir y leer en cualquiera de sus localidades. La secuencia deseada se realiza durante una operación normal dentro del rango Vdd (5V) del dispositivo .Estas operaciones utilizan un solo byte de datos para memoria EEPROM y una palabra de datos (2 bytes) para memoria flash. En una operación de escritura interviene la secuencia de borrado/escritura en la memoria a utilizar (EEPROM o FLASH), con esto se logra que antes de escribir el dato en la localidad especificada, el valor anterior se borra.
Una operación (lectura o escritura) en la memoria EEPROM no altera el proceso normal del microcontrolador pero, una secuencia de escritura a la memoria flash si detiene la operación normal del dispositivo, por lo que la memoria de programa no puede ser accesada durante una escritura a memoria flash. Para realizar las instrucciones tanto de lectura como escritura en ambas memorias se utilizan los registros:

- EEDATA
- EECON1
- EECON2
- EEADR
- EEADRH
- EEDATH

Cuando se realiza operaciones en un bloque de memoria EEPROM, el registro EEADR mantiene la dirección que se desea accesar, el registro EEDATA mantiene el dato que será escrito en la memoria (escritura) o recibe el dato que ha sido leído desde la memoria EEPROM (lectura), los dispositivos 16F877/A poseen un campo de memoria de datos EEPROM de 256 bytes (00h-FFh) accesado por medio del registro EEADR.
La memoria de programa flash está direccionada por los registros EEADRH: EEADR los cuales mantiene los 13 bits necesarios para accesar a cualquier localidad de memoria flash, los registros EEDATH: EEDATA mantienen los 14 bits de datos para escritura o reflejan el contenido de la memoria en una operación de lectura, el valor de la dirección debe estar dentro del rango que posee la memoria flash, normalmente está en las direcciones (0000h -3FFFh).

1.6.1 REGISTROS EECON1 Y EECON2.

El registro EECON1 es el registro utilizado para configurar e inicializar las operaciones que se realizarán en la memoria deseada.

R/W-x	U-0	U-0	U-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	—	—	—	WRERR	WREN	WR	RD
bit 7							bit 0

EEPGD: Program/Data EEPROM Select bit
1 = Accesses program memory
0 = Accesses data memory
(This bit cannot be changed while a read or write operation is in progress)

Unimplemented: Read as '0'

WRERR: EEPROM Error Flag bit
1 = A write operation is prematurely terminated
(any MCLR Reset or any WDT Reset during normal operation)
0 = The write operation completed

WREN: EEPROM Write Enable bit
1 = Allows write cycles
0 = Inhibits write to the EEPROM

WR: Write Control bit
1 = Initiates a write cycle. (The bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.)
0 = Write cycle to the EEPROM is complete

RD: Read Control bit
1 = Initiates an EEPROM read. (RD is cleared in hardware. The RD bit can only be set (not cleared) in software.)
0 = Does not initiate an EEPROM read

Fig.1.5.1.*Registro EECON1.*

El bit EEPGD determina si el acceso será a la memoria EEPROM o FLASH, por lo que si este bit es “0” todas las operaciones serán hechas en la memoria de datos.Si es”1” todas las operaciones serán hechas en la memoria de programa .Los bit RD y WR inician una lectura o escritura respectivamente al activarlos por software , se limpian por hardware al terminar una secuencia de escritura o lectura ;el bit WREN permite que se realice una operación de escritura , en una operación de reset este bit queda como “0” por lo que no permite la escritura a la memoria EEPROM,siendo necesario activarlo en una operación de escritura. El bit WRERR es una bandera que me indica cuando en una operación de escritura o borrado se ejecuta un reset en MCLR o debido al perro guardian ,por lo que despues del reset es necesario activar nuevamente la secuencia de escritura, ya que los registros que contienen la direccion y datos no se alteran.

Si las interrupciones están activas, el bit EEIF del registro PIR2 sirve para indicar que ha terminado una escritura de datos, este bit debe ser limpiado por software.

El registro EECON2 sólo es utilizado en la secuencia de escritura ;despues de haber activado todos los bits necesarios para realizar la operación de escritura se carga el valor de 55h y el valor de AAh en orden , con esto se logra prevenir escrituras parásitas.

1.7 MÓDULO SERIAL SINCRONO /ASINCRONO (USART).

Conocido más comúnmente como interface de comunicación serial SCI, este módulo puede ser configurado como sistema asíncrono full dúplex para comunicarse con terminales CRT[8] y computadores personales, o ser configurado como sistema síncrono half dúplex para establecer comunicación con periféricos como conversores A/D, D/A, EEPROMS seriales, etc.

El modulo USART puede ser configurado en estos tres modos.

- Comunicación asíncrona full dúplex.
- Comunicación síncrona modo maestro half dúplex.
- Comunicación síncrona modo esclavo half dúplex.

Los registros asociados con este módulo son:

- Registros TXSTA y RCSTA mostrados en las figuras 1.6.1. y 1.6.2.
- Registro de generación de baudios SPBRG.
- Registro para transmisión TXREG.
- Registro para recepción RCREG.

Para activar el modulo USART se coloca a “1” el bit SPEN (RCSTA<7>), con esto se configuran los pines RC6 (TX salida), RC7 (RX entrada).

[8] **CRT** Tubo de rayos catódicos.

TXSTA

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
bit 7							bit 0

CSRC: Clock Source Select bit
Asynchronous mode:
Don't care
Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

TX9: 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

TXEN: Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled

Note: SREN/CREN overrides TXEN in SYNC mode.

SYNC: USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

Unimplemented: Read as '0'

BRGH: High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode

TRMT: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

TX9D: 9th bit of Transmit Data, can be parity bit

Fig.1.6.1. *Registro TXSTA.*

En el registro TXSTA es en donde puede configurarse la fuente de reloj a utilizar , activar la transmisión de datos ,el modo del reloj ya sea síncrono o asíncrono,especificar la transmisión de 9 u 8 bits y verificar cuando se ha terminado el envío de un dato.

RCSTA

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
bit 7							bit 0

SPEN: Serial Port Enable bit
1 = Serial port enabled (configures RC7/RX/DT and RC6/TX/CK pins as serial port pins)
0 = Serial port disabled

RX9: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

SREN: Single Receive Enable bit
Asynchronous mode:
Don't care
Synchronous mode - master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode - slave:
Don't care

CREN: Continuous Receive Enable bit
Asynchronous mode:
1 = Enables continuous receive
0 = Disables continuous receive
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

ADDEN: Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enables interrupt and load of the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received, and ninth bit can be used as parity bit

FERR: Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

OERR: Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

RX9D: 9th bit of Received Data (can be parity bit, but must be calculated by user firmware)

Fig.1.6.2.*Registro RCSTA*.

En el registro RCSTA es en donde se activa el modulo USART, se activa la función de recepción continua, se especifica si se recibirán datos de 9 u 8 bits, se verifican los bits que indican error de sobre flujo (cuando un dato es recibido sin haber leído el anterior) y error de framming (error indicado porque el bit de STOP recibido es "1", el cual debería ser "0").

Cuando se detecta cualquiera de estos errores, los datos involucrados en la transferencia se pierden, por lo se necesita realizar la operación transmisión/recepción de estos datos nuevamente.

1.7.1 REGISTRO DE GENERACIÓN DE BAUDIOS SPBRG.

El registro de generación de baudios soporta tanto modo asíncrono como modo síncrono. Este registro tiene un generador de baudios de 8 bits y es el encargado de definir la velocidad de transferencia ya sea para transmisión o recepción ; la velocidad depende del valor cargado en el regitro SPBRG y del bit BRGH.Este ultimo tiene comportamiento diferente según el tipo de comunicación utilizado: en modo asíncrono el bit BRGH (TXSTA<2>) tambien controla los baudios ,en modo síncrono el bit BRGH es ignorado .Para definir la tasa de transferencia la figura 1.6.3. muestra la formula utilizada.

Bit SYNC	BRGH=0 (Velocidad baja)	BRGH=1 (Velocidad alta)
0	Asíncrono,Baudios=Fosc/(64(X+1))	Baudios= Fosc/(16(X+1))
1	Síncrono, Baudios=Fosc/(4(X+1))	No aplica

X=valor a colocar en registro SPBRG(0 a 255).

Fig 1.6.3.*Tasas de transferencia utilizadas en el modulo SCI.*

Para obtener el valor a cargar en el registro SSPBRG se despeja x de la ecuación utilizando el valor de velocidad que se desea (ejemplo. 9600 bps), muchas veces no se conseguiran valores enteros , siempre existirá un pequeño error pero, al utilizar la formula con BRGH=1(velocidad alta) el error se ve minimizado.

Escribir un nuevo valor en SSPBRG causaría que el timer interno vuelva a iniciar el conteo.

1.7.1.1 OPERACIÓN DE MUESTREO.

Los datos en el pin RX/DT son muestreados 3 veces para obtener con mayor precisión el valor presente en el pin de recepción, estos valores pueden ser alto o bajo. El receptor recibe cada bit en el registro RCREG al ritmo que marca su generador de baudios o reloj , el cual debe sincronizarse con la aparición del bit de START en la linea de datos.

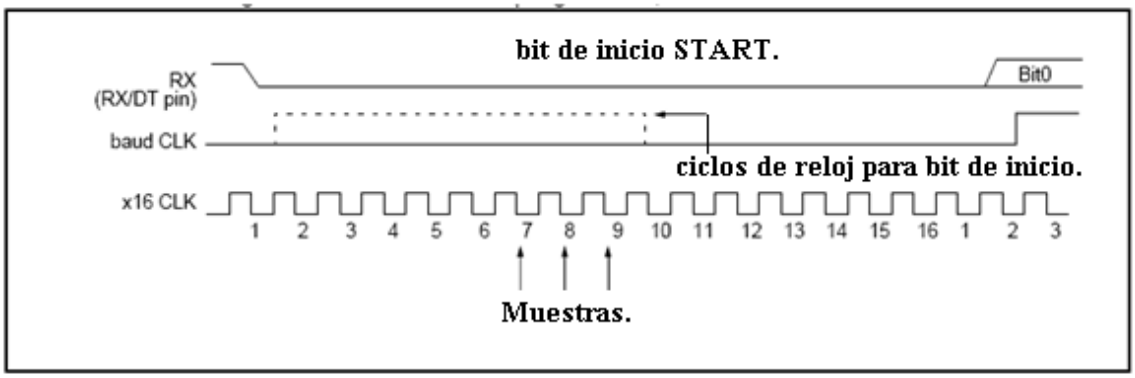


Fig.1.6.4 *Secuencia de muestreo de datos en el pin RX.*

El muestreo en la transmisión se realiza colocando en el registro TXREG cada bit del byte a enviar ,estos bits son guardados uno a uno ,iniciando con el Bit0, solamente en los flancos positivos del reloj controlado por el generador de baudios.En el registro TXREG es en donde se guardan los bytes que se desean transmitir.

1.7.2 MÓDULO USART EN MODO ASÍNCRONO.

Este tipo de comunicación es el más adecuado para conexión con equipos alejados, ya que los niveles lógicos de las señales corresponden a los niveles de voltaje de alimentación del mcrocontrolador.

Existen varias normas para la transmisión serie asíncrona como la norma RS232 que emplea niveles de tensión mas inmunes al ruido, norma RS485 y norma RS422 que emplean tensiones diferenciales y que sirven para distancias más largas entre dispositivos.la ventaja de esta comunicación es que la transferencia puede ser unidireccional , bidireccional o simultanea.

Para la transferencia asíncrona se emplean relojes de la misma frecuencia. Cada paquete de bits es de tamaño fijo y se enmarca con un bit de arranque (START) y un bit de parada (STOP) que sirven para sincronizar los relojes tanto del emisor como del receptor.

La línea de datos en estado “1” está inactiva, si se desea enviar un dato se manda un bit de inicio (START) que activa la línea de datos llevándola al valor “0” durante el tiempo correspondiente a un bit, al finalizar el envío del dato la línea se vuelve a colocar a “1” durante el tiempo de un bit que corresponde al bit de STOP.

La recepción o transmisión son de forma independiente compartiendo el generador de baudios. En la transferencia de datos ya sea actuando como transmisor o receptor intervienen los pines y registros:

TX pin para transmisión de datos (RC6).

RX pin de recepción de datos (RC7).

Registro de generación de baudios SPBRG.

Circuito de muestreo.

1.7.3 COMUNICACIÓN ASÍNCRONA EN MODO TRANSMISOR.

En este modo de comunicación el microcontrolador es el encargado de realizar la transmisión de datos hacia un dispositivo que puede ser una PC, otro microcontrolador o periférico que soporte este tipo de comunicación.

Para activar este modo en el PIC debe de llevarse el bit TXEN (TXSTA<5>) a “1”, ya habiendo establecido el valor del reloj por medio del generador de baudios. La transmisión no se genera automáticamente sino hasta que un valor ha sido cargado al registro TXREG por software, automáticamente el valor en TXREG es transferido al registro TSR que es el corazón del sistema transmisor y no acepta otro dato hasta que el bit de STOP haya sido transmitido ; para indicarme que el registro TXREG esta vacío se activa la bandera de interrupción TXIF (PIR1<4>) la cual es activada por medio del bit TXIE (PIE1<4>) ,cuando se carga TXREG con otro dato se limpia el bit TXIF .

Otra bandera usada es el bit TMRT (TXSTA<1>) que me indica cuando se ha transmitido un dato desde el registro TSR, es decir indica cuando el registro TSR esta vacío, esta bandera es de solo lectura por eso es la más usada ya que me indica que la transmisión ha sido hecha hacia el otro dispositivo y no genera una interrupción.

El sistema es capaz de transmitir datos de 8 bits y de 9 bits, la más común es la transmisión de 8 bits. Para seleccionar la opción de transmisión de 9 bits se activa el bit TX9 (TXSTA<6>) y se coloca el noveno bit en TX9D (TXSTA <0>), el noveno bits debe de ser cargado a TX9D antes de cargar el dato (8 bits) en el registro TXREG.

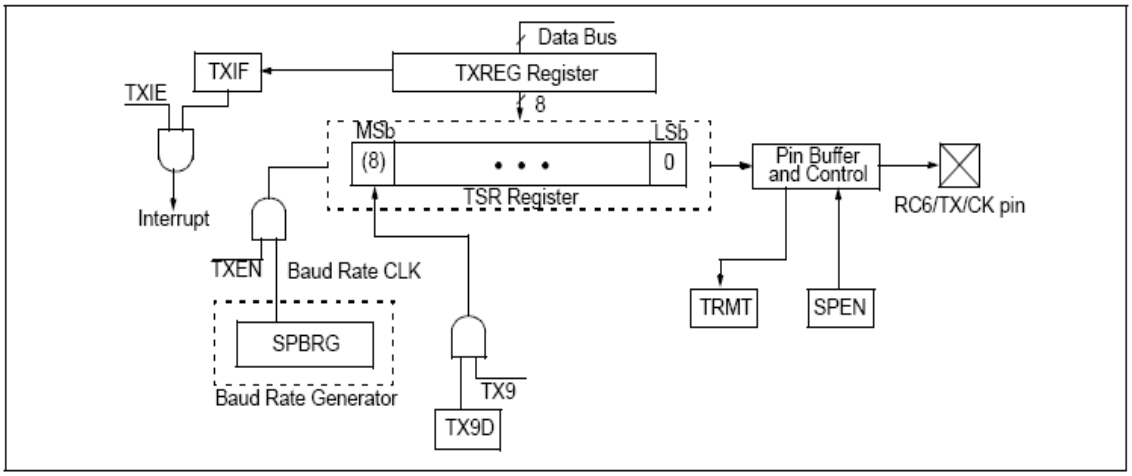


Fig.1.6.4.*Diagrama sistema transmisor SCI*.

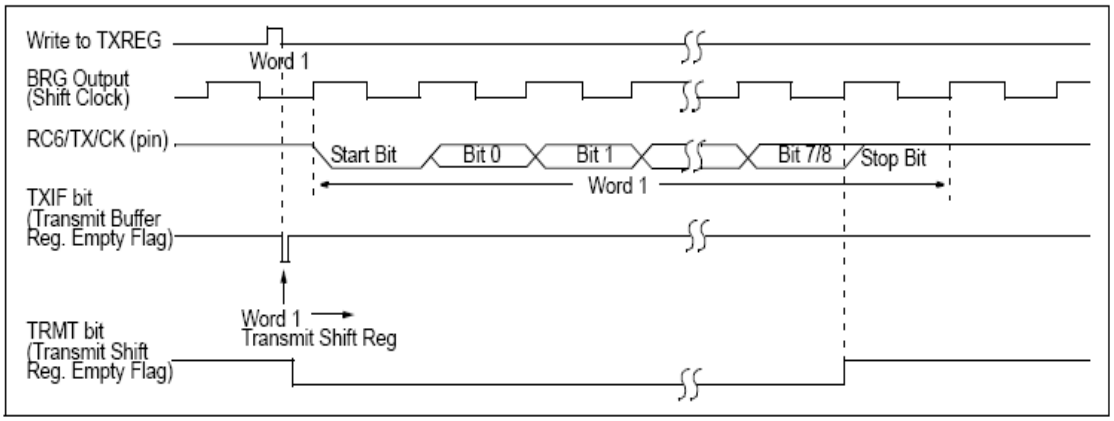


Fig.1.6.5.*Secuencia de transmisión asíncrona.*

La secuencia se inicia cargando el dato al registro TXREG ,cada bit se transmite en cada flanco de subida del reloj, el bit TMRT se activa cuando se ha transmitido todo el dato ,la secuencia es la misma para cada dato a enviar.

1.7.4 COMUNICACIÓN ASÍNCRONA EN MODO RECEPTOR.

Cuando el dispositivo es configurado en este modo, acepta datos provenientes de otro dispositivo. Los datos son recibidos externamente por medio del pin RX y guardados en el registro RCREG para posteriormente ser leídos, el diagrama de bloques se muestra en la figura 1.6.6.

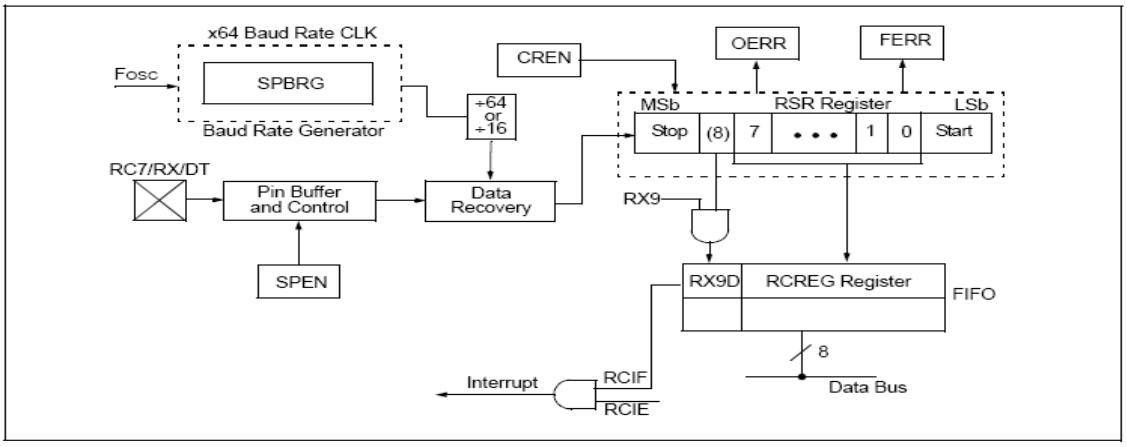


Fig.1.6.6.*Diagrama de bloques sistema receptor SCI.*

para activar este modo se utiliza el bit CREN (RCSTA<4>) colocándolo a “1”, el dato es recibido primero por el registro TSR, después que se ha muestreado el bit de STOP, el valor del registro TSR es transferido al registro RCREG ,activándose el bit RCIF (PIR1<5>)que indica dato listo para ser leído ,este bit es limpiado automáticamente cuando se lee RCREG; si se desea generar una interrupción se debe activar el bit RCIE (PIE1<5>). En este modo existe la posibilidad de reconocer errores que son indicados por los bits OERR (RCSTA<1>) el cual me indica cuando ha existido un error de sobre escritura (recepción de otro dato antes de que el anterior se haya leído) este bit es limpiado por software o realizando un reset en el mecanismo de recepción, también el bit FERR (RCSTA<2>) que indica error de trama cuando se recibe “1” en el bit STOP el cual debería ser “0”.

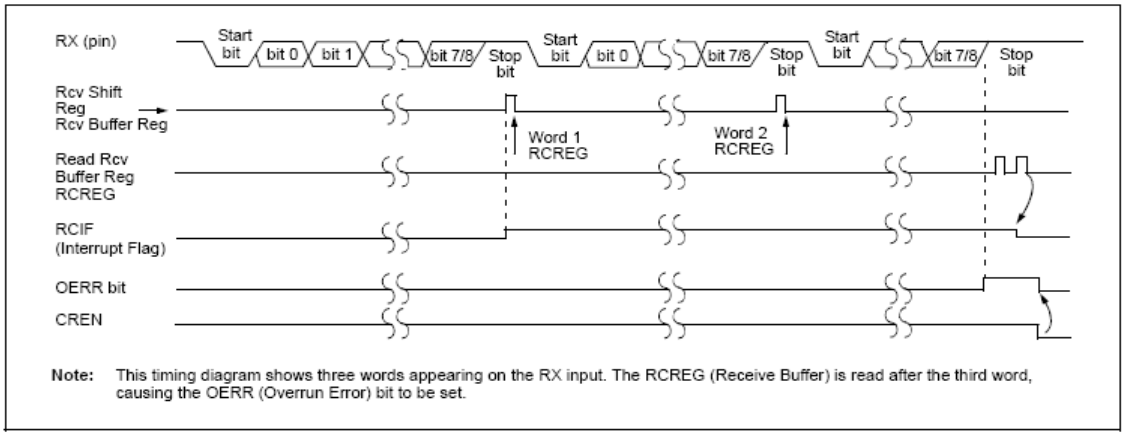


Fig.1.6.7.*Secuencia de recepcion asincrona.*

1.8 INTERFACE DE COMUNICACIÓN SERIE SÍNCRONA MAESTRA (MSSP).

Este modulo es utilizado para la comunicación con otros periféricos incluyendo microcontroladores, memorias EEPROMs, conversores A/D o D/A y LCDs que soporten este tipo de comunicación serial síncrona.

Existen 2 tipos de comunicación que están dentro de este modulo las cuales son:

- Interface de periféricos serie (SPI).
- Interface serie entre circuitos integrados (I^2C).

1.8.1 REGISTROS DE CONTROL PARA MÓDULO MSSP.

Existen 2 registros de control asociados al módulo MSSP y un registro de estado, con los cuales pueden activarse los dos sistemas de comunicación que soporta el subsistema MSSP:

- SSPCON
- SSPSTAT
- SSPCON2

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

WCOL: Write Collision Detect bit
Master mode:
1 = A write to SSPBUF was attempted while the I2C conditions were not valid
0 = No collision
Slave mode:
1 = SSPBUF register is written while still transmitting the previous word (must be cleared in software)
0 = No collision

SSPOV: Receive Overflow Indicator bit
In SPI mode:
1 = A new byte is received while SSPBUF holds previous data. Data in SSPSR is lost on overflow. In Slave mode, the user must read the SSPBUF, even if only transmitting data, to avoid overflows. In Master mode, the overflow bit is not set, since each operation is initiated by writing to the SSPBUF register. (Must be cleared in software.)
0 = No overflow
In I^2C mode:
1 = A byte is received while the SSPBUF is holding the previous byte. SSPOV is a "don't care" in Transmit mode. (Must be cleared in software.)
0 = No overflow

SSPEN: Synchronous Serial Port Enable bit
In SPI mode,
When enabled, these pins must be properly configured as input or output
1 = Enables serial port and configures SCK, SDO, SDI, and SS as the source of the serial port pins
0 = Disables serial port and configures these pins as I/O port pins
In I^2C mode,
When enabled, these pins must be properly configured as input or output
1 = Enables the serial port and configures the SDA and SCL pins as the source of the serial port pins
0 = Disables serial port and configures these pins as I/O port pins

CKP: Clock Polarity Select bit
In SPI mode:
1 = Idle state for clock is a high level
0 = Idle state for clock is a low level
In I^2C Slave mode:
SCK release control
1 = Enable clock
0 = Holds clock low (clock stretch). (Used to ensure data setup time.)
In I^2C Master mode:
Unused in this mode

SSPM3:SSPM0: Synchronous Serial Port Mode Select bits
0000 = SPI Master mode, clock = Fosc/4
0001 = SPI Master mode, clock = Fosc/16
0010 = SPI Master mode, clock = Fosc/64
0011 = SPI Master mode, clock = TMR2 output/2
0100 = SPI Slave mode, clock = SCK pin. $\overline{SS}$ pin control enabled.
0101 = SPI Slave mode, clock = SCK pin. $\overline{SS}$ pin control disabled. $\overline{SS}$ can be used as I/O pin.
0110 = I^2C Slave mode, 7-bit address
0111 = I^2C Slave mode, 10-bit address
1000 = I^2C Master mode, clock = Fosc / (4 * (SSPADD+1))
1011 = I^2C Firmware Controlled Master mode (slave idle)
1110 = I^2C Firmware Controlled Master mode, 7-bit address with START and STOP bit interrupts enabled
1111 = I^2C Firmware Controlled Master mode, 10-bit address with START and STOP bit interrupts enabled
1001, 1010, 1100, 1101 = Reserved

Fig.1.7.2.*Registro SSPCON.*

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
bit 7							bit 0

SMP: Sample bit
SPI Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time
SPI Slave mode:
SMP must be cleared when SPI is used in slave mode
In I^2C Master or Slave mode:
1 = Slew rate control disabled for standard speed mode (100 kHz and 1 MHz)
0 = Slew rate control enabled for high speed mode (400 kHz)

CKE: SPI Clock Edge Select (Figure 9-2, Figure 9-3 and Figure 9-4)
SPI mode:
For CKP = 0
1 = Data transmitted on rising edge of SCK
0 = Data transmitted on falling edge of SCK
For CKP = 1
1 = Data transmitted on falling edge of SCK
0 = Data transmitted on rising edge of SCK
In I^2C Master or Slave mode:
1 = Input levels conform to SMBus spec
0 = Input levels conform to I^2C specs

D/A: Data/$\overline{\text{Address}}$ bit (I^2C mode only)
1 = Indicates that the last byte received or transmitted was data
0 = Indicates that the last byte received or transmitted was address

P: STOP bit
(I^2C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.)
1 = Indicates that a STOP bit has been detected last (this bit is '0' on RESET)
0 = STOP bit was not detected last

S: START bit
(I^2C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.)
1 = Indicates that a START bit has been detected last (this bit is '0' on RESET)
0 = START bit was not detected last

R/W: Read/Write bit Information (I^2C mode only)
This bit holds the R/W bit information following the last address match. This bit is only valid from the address match to the next START bit, STOP bit or not $\overline{\text{ACK}}$ bit.
In I^2C Slave mode:
1 = Read
0 = Write
In I^2C Master mode:
1 = Transmit is in progress
0 = Transmit is not in progress
Logical OR of this bit with SEN, RSEN, PEN, RCEN, or ACKEN will indicate if the MSSP is in IDLE mode.

UA: Update Address (10-bit I^2C mode only)
1 = Indicates that the user needs to update the address in the SSPADD register
0 = Address does not need to be updated

BF: Buffer Full Status bit
Receive (SPI and I^2C modes):
1 = Receive complete, SSPBUF is full
0 = Receive not complete, SSPBUF is empty
Transmit (I^2C mode only):
1 = Data transmit in progress (does not include the ACK and STOP bits), SSPBUF is full
0 = Data transmit complete (does not include the $\overline{\text{ACK}}$ and STOP bits), SSPBUF is empty

Fig.1.7.3.Registro SSPSTAT.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
bit 7							bit 0

GCEN: General Call Enable bit (In I^2C Slave mode only)
1 = Enable interrupt when a general call address (0000h) is received in the SSPSR
0 = General call address disabled

ACKSTAT: Acknowledge Status bit (In I^2C Master mode only)
In Master Transmit mode:
1 = Acknowledge was not received from slave
0 = Acknowledge was received from slave

ACKDT: Acknowledge Data bit (In I^2C Master mode only)
In Master Receive mode:
Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.
1 = Not Acknowledge
0 = Acknowledge

ACKEN: Acknowledge Sequence Enable bit (In I^2C Master mode only)
In Master Receive mode:
1 = Initiate Acknowledge sequence on SDA and SCL pins and transmit ACKDT data bit. Automatically cleared by hardware.
0 = Acknowledge sequence idle

RCEN: Receive Enable bit (In I^2C Master mode only)
1 = Enables Receive mode for I^2C
0 = Receive idle

PEN: STOP Condition Enable bit (In I^2C Master mode only)
SCK Release Control:
1 = Initiate STOP condition on SDA and SCL pins. Automatically cleared by hardware.
0 = STOP condition idle

RSEN: Repeated START Condition Enable bit (In I^2C Master mode only)
1 = Initiate Repeated START condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Repeated START condition idle

SEN: START Condition Enable bit (In I^2C Master mode only)
1 = Initiate START condition on SDA and SCL pins. Automatically cleared by hardware.
0 = START condition idle

Fig.1.7.1.Registro SSPCON2.

1.8.2 MODO DE COMUNICACIÓN SPI.

www.ate.**uniovi**.es/fernando/Doc2003/SED/**SSP_SPI**_BREVE.pdf

Este modo de comunicación permite la transmisión y recepción simultánea de manera síncrona de 8 bits de datos (1 byte), físicamente se utilizan 3 pines que son:

- Salida de datos serial (SDO).
- Entrada de datos serial (SDI).
- Reloj serial (SCK).

Adicionalmente se utiliza el pin de selección de esclavo (SS), solamente cuando el modulo actúa en modo esclavo SPI.

Cuando se inicializa el modulo SPI ciertas opciones necesitan ser especificadas, por lo que es necesario utilizar los bits de los registros SSPCON<5:0>y SSPSTAT<7:6>, con estos bits se logra especificar distintas funciones del módulo síncrono SPI.

- Modo maestro (SCK es la salida de reloj desde el dispositivo maestro).
- Modo esclavo (SCK es entrada hacia dispositivo esclavo).
- Polaridad / Flanco / Velocidad del reloj.
- Muestreo del dato de entrada.
- Modo de selección de esclavo.

Para activar el módulo SPI el bit SSPEN (SSPCON 5) debe colocarse a "1", con esto se configuran los pines que interviene en esta comunicación SDO, SDI, SCK llegando a ser pines seriales, no olvidando en realizar sus respectivas especificaciones de entrada o salida a través del registro TRISC correspondiente.

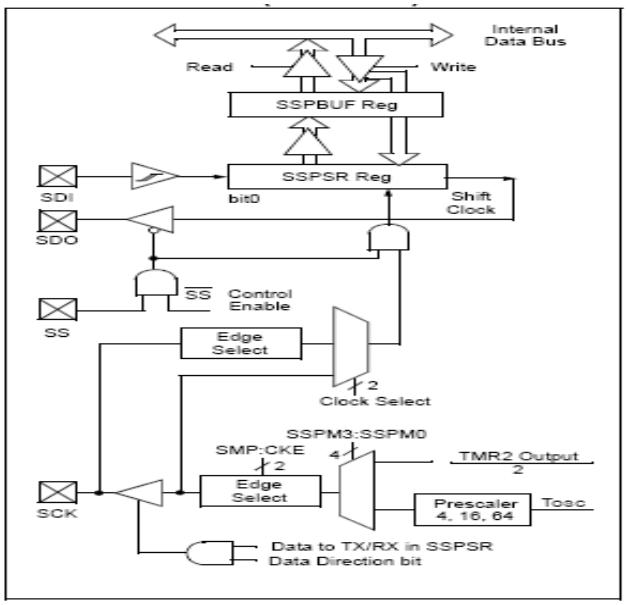


Fig.1.7.4.*Diagrama de bloques módulo SPI.*

1.8.2.1 MODO MAESTRO SPI.

Cuando el dispositivo se configura para que desarrolle la función de maestro, el MCU es el encargado de controlar el reloj del sistema que lleva la sincronía de los datos, además el maestro es el que decide cuando se inicia la comunicación.

Cuando el registro SSPBUF contiene un dato se puede iniciar una transmisión o recepción, si el módulo maestro sólo va a recibir puede obviarse el pin SDO. La polaridad del reloj puede lograse programando apropiadamente el bit CKP (SSPCON 4), escogiendo el tipo de forma de onda que servirá de reloj, el usuario puede utilizar cualquiera de estas opciones:

- Fosc/4 (Tcy).
- Fosc /16 (2 * Tcy).
- Fosc /64 (16 * Tcy).
- Salida del timer 2 /2.

Con esto se logra un máximo de frecuencia de reloj de 5.0MHZ, utilizando un oscilador de 20MHz .la gráfica siguiente muestra las formas de onda cuando el dispositivo se configura como maestro.

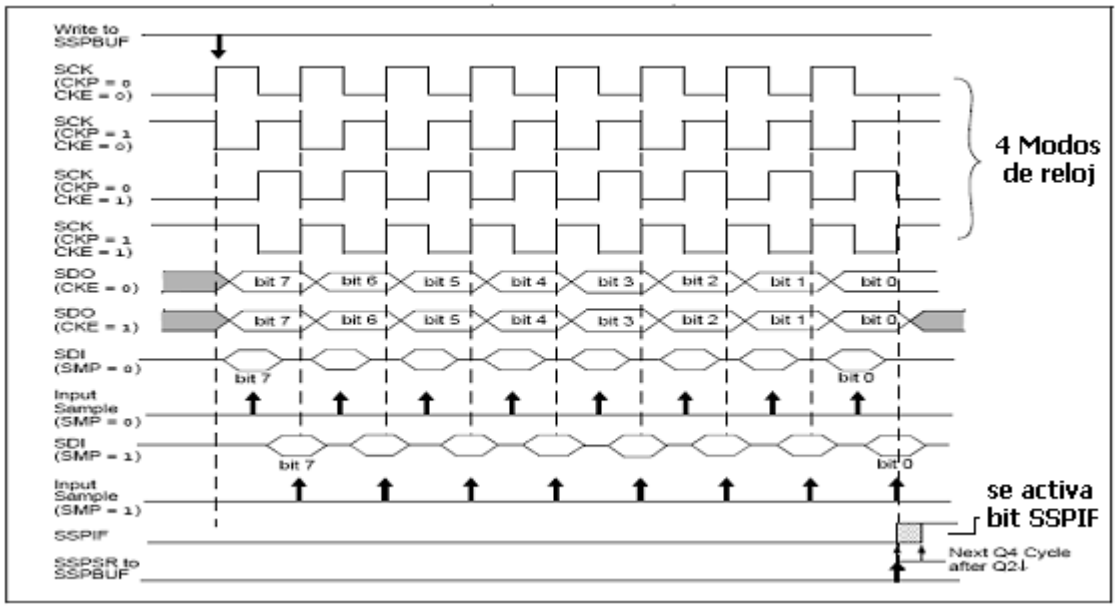


Fig.1.7.5.*Secuencia modo maestro SPI.*

Puede observarse que cuando el bit CKE es activado con “1”, el dato presente en SDO es válido antes de que exista un flanco de reloj en SCK, el muestreo del dato es controlado por el bit SMP y el tiempo necesario para cargar el registro SSPBUF con el dato recibido se muestra también.

1.8.2.2 MODO ESCLAVO SPI.

En modo esclavo la recepción o transmisión de datos es hecha solo cuando los pulsos de reloj aparecen en la entrada SCK provenientes de un dispositivo maestro, cuando el ultimo bit es recibido en el modo de recepción se activa la bandera de interrupciones SSPIF (PIR1<3>) indicando que existe un dato en el registro SSPBUF. En este modo es utilizado el pin SS, el cual sirve para saber cuando un maestro a escogido un determinado dispositivo esclavo para establecer comunicación, siempre que en el pin SS del esclavo se reciba un “0”.

Si el bit CKE esta activado con “1” es necesario activar el bit SS, si este bit es colocado a VDD el dispositivo esclavo recibirá un reset.

Para el modo esclavo se dan dos tipos de secuencias que están controladas por el bit CKE y se muestran en las siguientes gráficas.

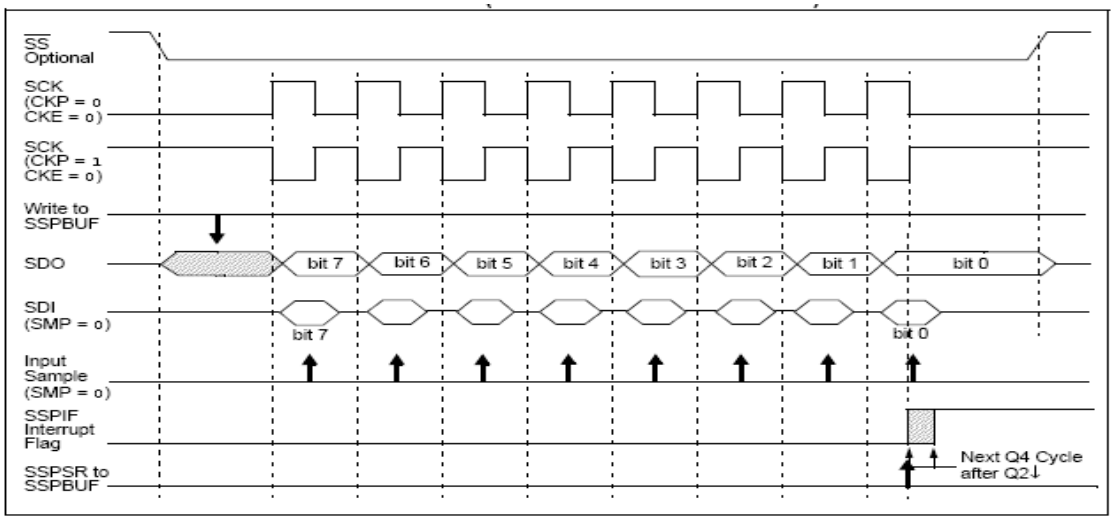


Fig.1.7.6.*Secuencia modo esclavo SPI con CKE=0.*

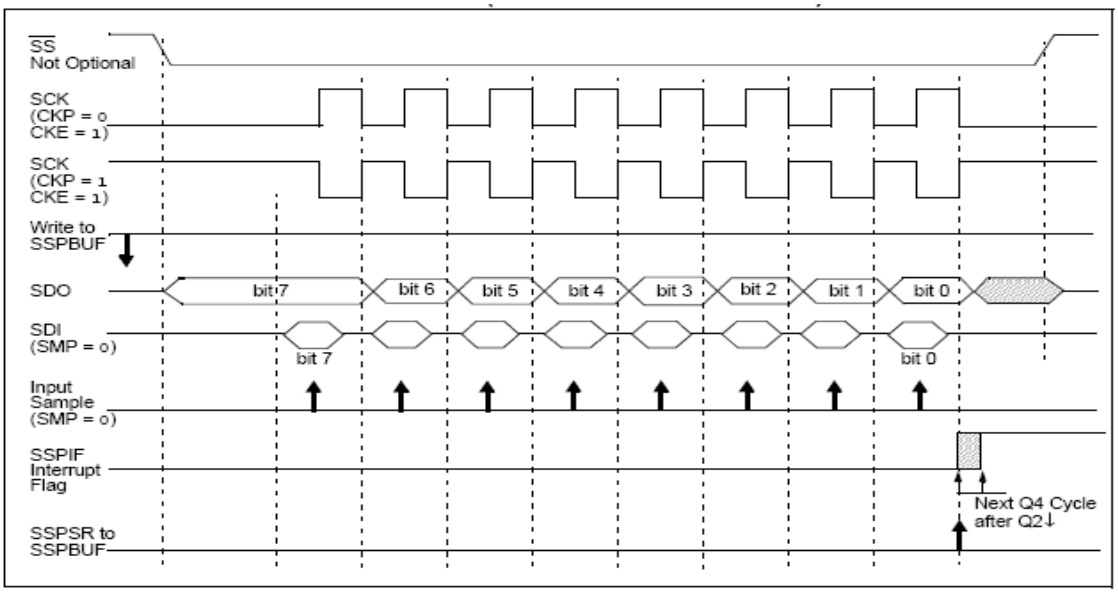


Fig.1.7.7.*Secuencia modo esclavo SPI con CKE=1.*

La tabla 1.7.1 muestra de una forma resumida los registros que intervienen en la utilización del modulo SPI.

Direccion	Nombre	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valor en POR, BOR	Valor en $\overline{MCLR}$, WDT
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
94h	SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	0000 0000

Tabla.1.7.1.*Registros que intervienen en modo SPI*.

1.8.3 MÓDULO DE COMUNICACIÓN I²C.

http://www.ate.uniovi.es/fernando/Doc2003/SED/SSP_I2C.pdf

El modulo MSSP utilizado en modo I²C, ofrece las funciones tanto de maestro como de esclavo (incluyendo soporte general de llamadas), además de ofrecer interrupciones al generar las señales de START y STOP para verificar si el bus de comunicación esta libre (Modo multimaestro). Los pines usados en el modo I²C son:

- Reloj serial (SCL) pin físico RC3/SCK/SCL.
- Datos seriales (SDA) pin físico RC4/SDI/SDA.

Los registros que interviene en la configuración del modo I²C son:

- Registro de control SSPCON.
- Registro de control 2 SSPCON2.
- Registro de estado SSPSTAT.
- Registro buffer para recepción/transmisión de datos seriales SSPBUF.
- Registro de direcciones SSPADD.

El usuario debe configurar estos pines como entrada o salida a través del registro TRISC.

Para seleccionar el modo I²C debe de activarse el bit SSPEN del registro SSPCON (figura 1.7.2), con esto se definen los pines SCL y SDA como reloj y datos respectivamente; resistores pull-up deben ser colocados externamente en SCL /SDA para la satisfactoria operación del modulo I²C.
El bit CKE es el encargado de establecer los niveles en los pines SCL y SDA ya sea en modo maestro o modo esclavo, si el bit CKE =0 los niveles estarán sujetos a la especificación I²C, si el bit CKE=1 los niveles estarán sujetos a las especificaciones SMBus. El registro SSPSTAT (figura 1.7.3) ofrece el estado de la transferencia de datos que incluye detección de los bits de START y STOP, especifica si el byte recibido fue dato o dirección, si el siguiente byte es el complemento de la dirección de 10 bits e indica si la operación es de lectura o escritura de datos.

El registro SSPBUF es en donde aparece el dato que se ha recibido o se desea transmitir, en el caso de transmisión cuando se ha completado esta operación se activa la bandera de interrupción SSPIF al igual que en el modo SPI, en el caso que otro byte sea recibido antes que se haya leído el registro SSPBUF se ejecuta un sobreflujo que es indicado por el bit SSPOV (SSPCON<6>) perdiendo el bit anterior.
El registro SSPADD es el que contiene la dirección para seleccionar el modulo esclavo, en el caso que se utilice el direccionamiento por 10 bits primero se escriben los bits más significativos y luego el resto de la dirección.

1.8.3.1 MODO ESCLAVO I^2C.

En el modo esclavo los pines SCL y SDA deben ser configurados como entradas, para establecer comunicación con un dispositivo esclavo debe de existir coherencia entre la dirección enviada por el maestro y la contenida por el esclavo, automáticamente el esclavo genera una señal de reconocimiento ACK, (el valor de la dirección enviada por el maestro se guarda en SSPBUF) y se activa el bit BF (buffer lleno).

Existen condiciones que causaran que el modulo MSSP no genere el pulso de reconocimiento.

- El bit de buffer lleno BF (SSPSTAT<0>), se activo antes de que la transferencia fue recibida.
- El bit de sobreflujo SSPOV (SSPCON<6>) se activo antes de que la transferencia fue recibida.

Bits de estado de dato recibido.		SSPSR → SSPBUF	genera pulso ACK.	se pone a uno bit SSPIF.
BF	SSPOV			
0	0	Si	Si	Si
1	0	No	No	Si
1	1	No	No	Si
0	1	Si	No	Si

Tabla.1.7.2.*Banderas involucradas en la transferencia de datos.*

Una vez el bit BF se active indicando que el buffer esta lleno, la manera para limpiarlo consiste en la lectura del registro SSPBUF, a diferencia del bit SSPOV que se limpia a través de software.

1.8.3.1a DIRECCIONAMIENTO.

Un vez el módulo MSSP ha sido configurado en el modo I²C como esclavo, se queda esperando una condición de START, cuando esta ocurre los bits que le siguen son guardados en el registro SSPSR, estos bits son muestreados en los flancos positivos de la señal de reloj. Una vez ya en el registro SSPSR este dato es comparado con el valor del registro SSPADD, esta comparación es hecha en el flanco negativo del octavo ciclo de reloj, si existe una igualdad entre la dirección recibida y la del dispositivo, además de estar desactivados los bits BF y SSPOV se dan los siguientes eventos.

- El valor del registro SSPSR es cargado en el registro SSPBUF en el flanco negativo del octavo pulso de reloj.
- El bit BF es activado en el flanco de bajada del octavo ciclo de reloj.
- El pulso de reconocimiento ACK es generado por el sistema.
- El bit SSPIF (PIR1<3>) es activado en el flanco negativo del noveno ciclo de reloj me india que se recibió un dato.

Existe la posibilidad del usar direcciones que constan de 10 bits, en este modo 2 bytes de direcciones necesitan ser recibidos por el esclavo, debe tomarse en cuenta que deben recibirse 2 bytes por lo que se hace uso del bit R/W (SSPSTAT<2>) indicándole al dispositivo que se realizará una escritura para aceptar el otro byte de direcciones.

1.8.3.1b RECEPCIÓN EN MODO ESCLAVO.

Primero debe seleccionarse el dispositivo esclavo colocando “0” en el pin SS ,cuando ocurre una validación de dirección y el bit R/W que acompaña a la dirección es “0” (escritura),el dato recibido (dirección) es cargado en el registro SSPBUF generando que el bit R/W del registro SSPSTAT se ponga a “0” , el bit BF se ponga a “1” ,el bit de interrupción SSPIF (PIR1<3>) se ponga a “1” y se genere la señal ACK.

Después que el esclavo generó la señal ACK el maestro le envía un dato, que se guardará en SSPBUF y se indicará con el bit BF que el dato puede ser leído.
Cada vez que reciba un dato el esclavo generara una señal ACK, hasta que se de una señal de STOP por parte del maestro.

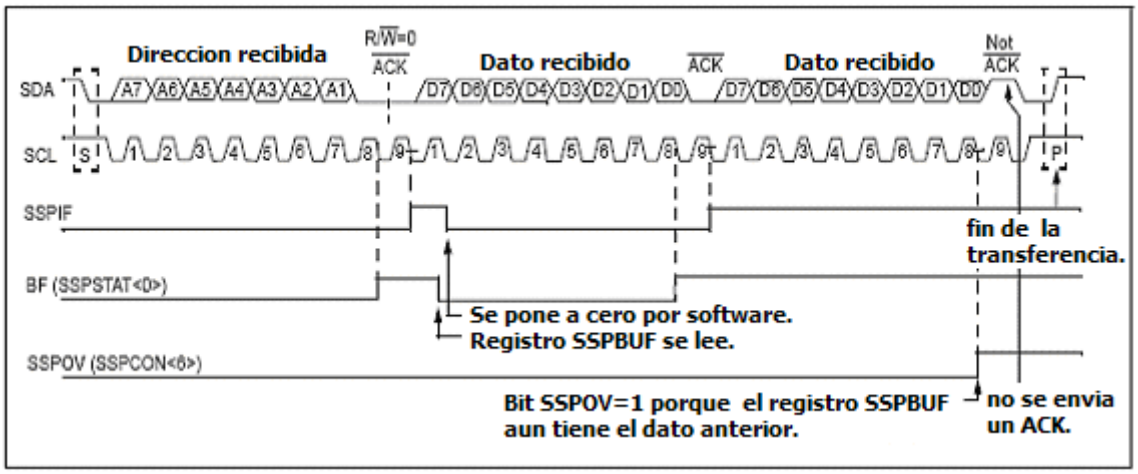


Fig.1.7.8.*Secuencia de recepción con direccionamiento de 7 bits*.

1.8.3.1c TRANSMISIÓN EN MODO ESCLAVO.

Cuando se da una coincidencia en la dirección y además el bit R/W que acompaña a la dirección es “1”(lectura) el bit R/W del registro SSPSTAT se pone a “1” ,el bit BF se activa “1” ,se genera el pulso de reconocimiento en el pin SDA y la dirección recibida se guarda en el registro SSPBUF.

El maestro retiene el reloj a cero esperando a que el esclavo prepare la transmisión del dato (el bit CKP vale “0”) ;para liberar la línea SCL debe activarse el bit CKP con “1” después de cargar el valor a transmitir al registro SSPBUF que a su vez lo carga en el registro SSPSR ; al finalizar el envío del byte desde el esclavo se debe esperar a que se reciba un pulsos de reconocimiento ACK desde el maestro para saber si se continuará enviando datos, si por el contrario no se recibe este pulso ACK se asume que la transferencia terminó . El esclavo se queda esperando nuevamente el bit de START, el bit SSPIF se activa cada vez que se transmite un dato y debe de limpiarse por software al iniciar la transmisión de otro dato .

El bit BF se activa cada vez que se carga un valor en SSPBUF y se limpia en el octavo flanco de bajada del reloj (cuando el buffer esta vacio).

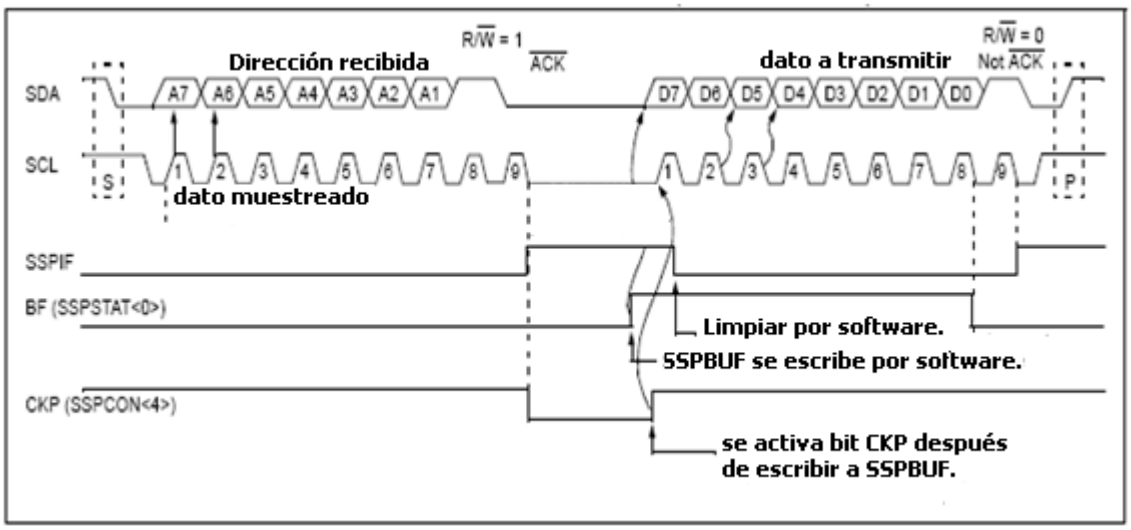


Fig.1.7.9.*Secuencia de transmisión modo esclavo I²C.*

1.8.3.1d MODO DE LLAMADA GENERAL EN EL BUS I²C.

La opción de llamada general al bus se desarrolla enviando una dirección válida para todos los esclavos presentes en el bus I²C, la dirección de 7 bits que envía el maestro consiste en 7 ceros seguidos del valor de R/W =0. En este caso todos los esclavos presentes deberían responder con un pulso de reconocimiento ACK. En los microcontroladores PIC la llamada general se indica al activarse el bit GCEN (SSPCON2<7>).

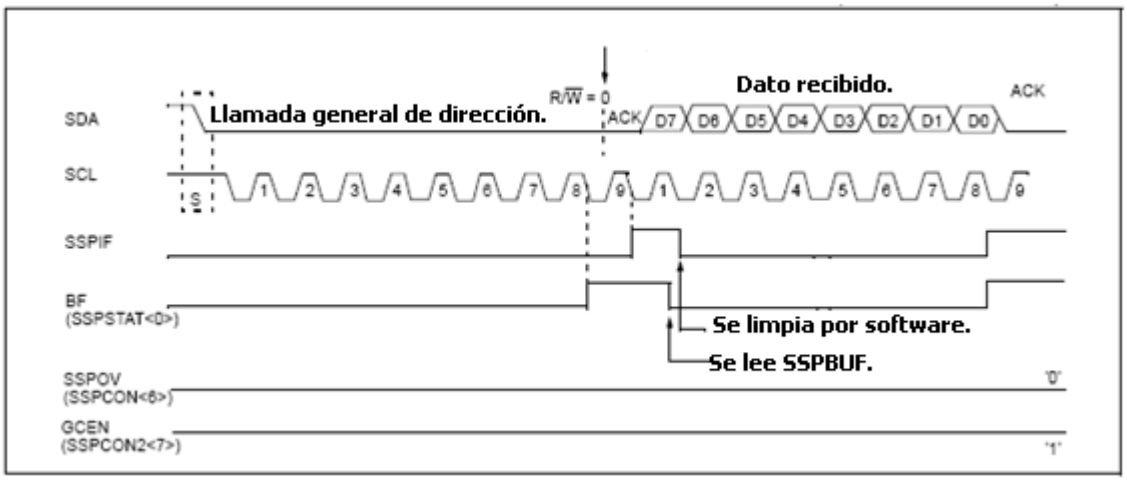


Fig.1.7.10.*Secuencia de llamada general en el bus I²C.*

El proceso inicia con el bit de START , seguido de los 8 bits que se cargan al registro SSPSR comparándose con el valor del registro SSPADD; si existe una coincidencia entre ambos este valor en SSPSR se transfiere al registro SSPBUF , entonces los bits BF y SSPIF se colocan a “1”. Para verificar si el direccionamiento es por llamada general se puede leer el registro SSPBUF.

1.8.3.2 MODO MAESTRO.

En este modo el dispositivo maneja el reloj que esta en el bus I²C, las líneas SDA y SCL son manejadas por el hardware por lo que no es necesaria su especificación con el registro TRISC.

El hardware interno del dispositivo dispone de circuitos lógicos para la detección de las condiciones START y STOP reflejadas en los bits S y P respectivamente, estos bits son solo de lectura y se limpian con un reset o cuando el modulo MSSP está desactivado. Para la activación del modo maestro deben usarse los bits SSPM (SSPCON) y el bit SSPEN.

La bandera de interrupciones SSPIF en modo maestro indica cualquiera de estos eventos.

- Condición de START
- Transferencia de un byte.
- Repetición de START.
- Condición de STOPS.
- Transmisión de ACK.

El dispositivo actuando en modo maestro es el encargado de manejar los pulsos del reloj por lo que se debe configurar ciertos parámetros del reloj a utilizar a través del registro de generación de baudios (BRG).
La frecuencia del reloj viene dada por la expresión:

FSCL =Fosc / (4*(SSPADD +1))	En I2C las frecuencias estandar son de 100KHz , 400KHz y 1MHz.

Cuando ocurre una escritura al registro SSPBUF el registro BRG automáticamente inicia el conteo decrementandose 2 veces por cada ciclo de instrucción (Tcy), al llegar a cero se queda esperando hasta que se realice otra escritura a SSPBUF. En el modo I2C BRG es recargado automáticamente a su valor inicial .La figura 1.7.11 muestra el diagrama de bloques del generador de baudios y la tabla 1.7.3 muestra diferentes valores para la generación de baudios.

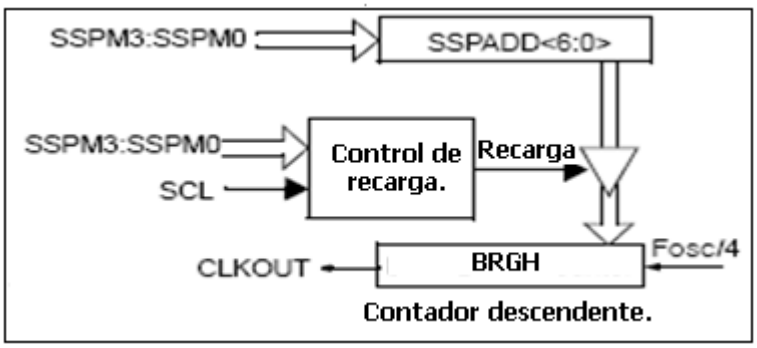


Fig.1.7.11.*Generador de baudios.*

FCY	FCY*2	Valor en BRG	FSCL
10 MHz	20 MHz	19h	400 kHz(1)
10 MHz	20 MHz	20h	312.5 kHz
10 MHz	20 MHz	3Fh	100 kHz
4 MHz	8 MHz	0Ah	400 kHz(1)
4 MHz	8 MHz	0Dh	308 kHz
4 MHz	8 MHz	28h	100 kHz
1 MHz	2 MHz	03h	333 kHz(1)
1 MHz	2 MHz	0Ah	100 kHz
1 MHz	2 MHz	00h	1 MHz(1)

Tabla.1.7.3.*Valores de generación de baudios*.

Las operaciones que genera el dispositivo configurado como maestro son: generación de START, generación de ACK, generación de STOP; utilizadas en los modos de recepción o transmisión.

1.8.3.2a GENERACIÓN DE START.

Para iniciar la condición de START se activa el bit SEN (SSPCON2<0>), si los pines SDA y SCL están a “1” el generador de baudios se carga con el valor de SSPADD<6:0> iniciando el decremento del registro BRG; si al llegar al final de la cuenta los pines SDA y SCL permanecen a “1” el pin SDA cambia a estado bajo “0” activándose el bit S e indicando que se dio la condición de START (la línea SCL permanece en “1”) .La figura 1.7.12 muestra el diagrama de tiempo para la generación de la señal START.

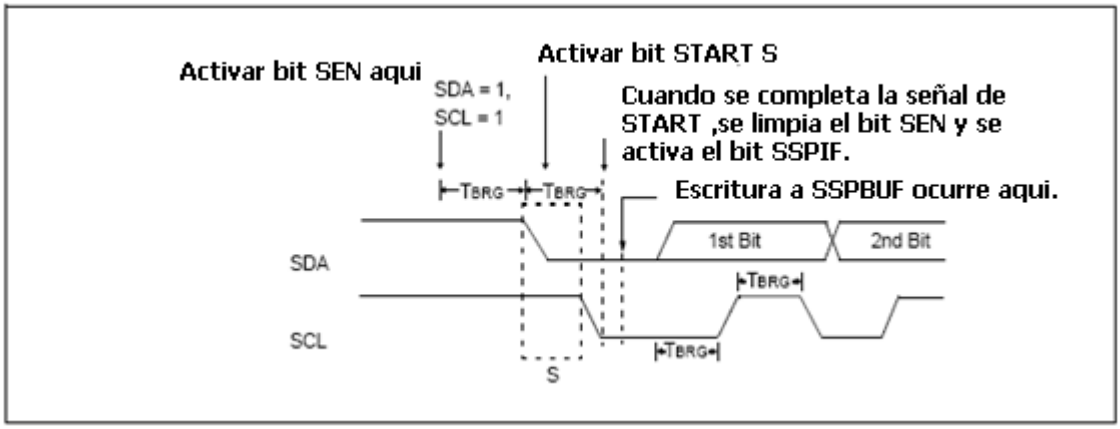


Fig.1.7.12.*Generación señal START*.

Consideraciones:

Si al inicio de una condición de START los pines SCL y SDA están a cero o si la línea SCL está en bajo antes de que se de el cambio en la línea SDA, se da una colisión en el bus; esto es indicado por la activación del bit de interrupción BCLIF y la condición de START es abortada llevando al módulo I2C al estado inactivo.

Si se desea escribir al registro SSPBUF mientras se genera la señal de START, el bit WCOL (SSPCON) es activado y el registro SSPBUF no se altera.

1.8.3.2b GENERACIÓN DE LA SEÑAL DE RECONOCIMIENTO ACK.

La secuencia de generación de la señal ACK generada por el maestro es iniciada al activar el bit ACKEN (SSPCON2<4>), con esto la línea SCL queda retenida con cero y el contenido del bit ACKDT (SSPCON2<5>) se refleja en la línea SDA; si se desea generar una señal de reconocimiento el bit ACKDT debe ser “0”, si por el contrario se desea generar una señal de no reconocimiento el bit ACKDT debe ser “1”.Los tiempos para generar la señal ACK se basan en el empleo del generador de baudios BRG, tras el tiempo TBRG la línea SCL vuelve a “1”.Al detectar el flanco de subida en SCL pasa otro tiempo TBRG , transcurrido este segundo tiempo SCL vuelve a cero ,es entonces que el bit ACKEN se desactiva y el modulo MSSP pasa al estado inactivo. La figura 1.7.13 muestra la secuencia de generación de la señal ACK.

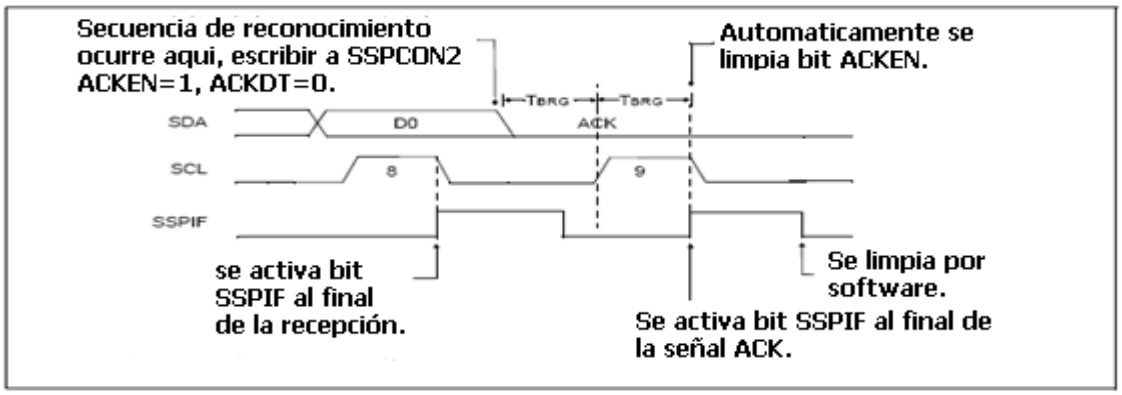


Fig.1.7.13.*Generacion señal ACK.*

1.8.3.2c GENERACIÓN DE LA SEÑAL DE STOP.

La señal de STOP es la encargada de finalizar las secuencias de recepción o transmisión, esta señal es generada por el maestro y se inicia activando el bit PEN (SSPCON2<2>);al terminar una secuencia de transmisión el pin SCL queda en el estado bajo , si el bit PEN se pone a “1” el pin SDA se pone a “0” y el BRG inicia el conteo .Al finalizar el tiempo TBRG la línea SDA pasará a “1”,después de otro tiempo TBRG se liberará la línea SDA activándose el bit P (SSPSTAT<4>) indicando la condición de STOP; después de esto se espera otro tiempo TBRG para que el bit PEN se desactive automáticamente .La figura 1.7.14 muestra la generación de la señal de STOP.

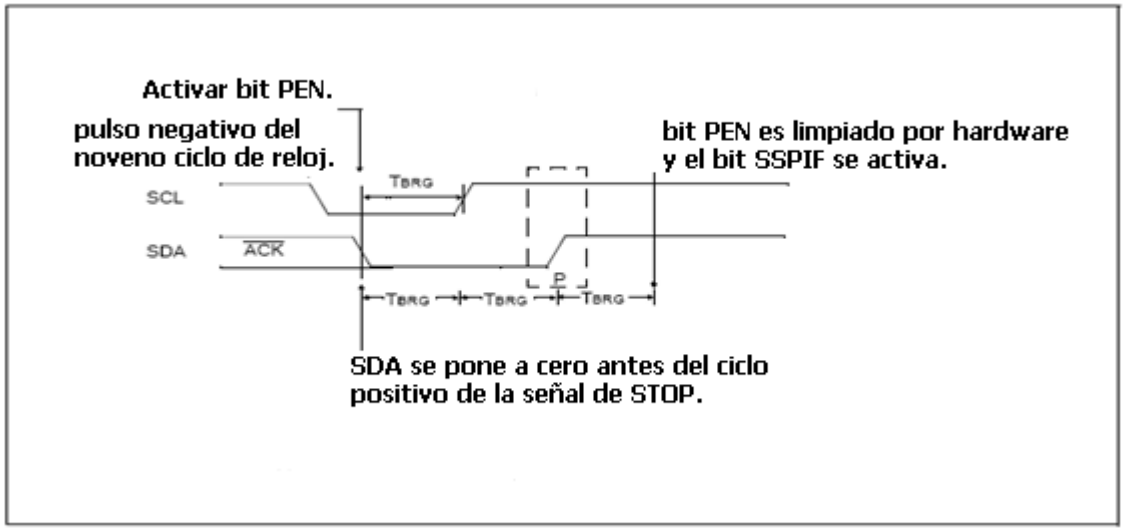


Fig.1.7.14.*Generación señal STOP.*

1.8.3.2d MODO MAESTRO PARA TRANSMISIÓN DE DATOS.

Utilizando este modo pueden transmitirse datos hacia un dispositivo esclavo, estos bytes pueden ser de dirección o datos; después de haber sido generado la condición de START la secuencia de transmisión se inicia cargando el byte al registro SSPBUF, esto hará que el bit BF se active indicando que existe un dato válido en el buffer e iniciando el conteo en el registro BRG. Cada bit será enviado por el pin SDA en los flancos de bajada de la línea SCL. Cuando se transmita el octavo bit por la línea SDA, el bit BF se limpia automáticamente y el maestro libera la línea SDA para esperar la señal de reconocimiento enviada por el dispositivo esclavo. El estado del bit ACK de carga en el bit ACKSTAT (SSPCON2<6>) en el flanco de bajada del noveno pulso de reloj ,si el valor en ACKSTAT en el maestro es “0” indica que si se recibió un bit ACK, sin el valor es “1” indica que no se recibió un bit ACK desde el esclavo ; al terminar el dato se activa el bit SSPIF y el reloj se detiene hasta que se vuelve a cargar un valor en el registro SSPBUF , el pin SCL queda en estado bajo “0” .En la figura 1.7.15 se observa el diagrama de tiempo de la transmisión.

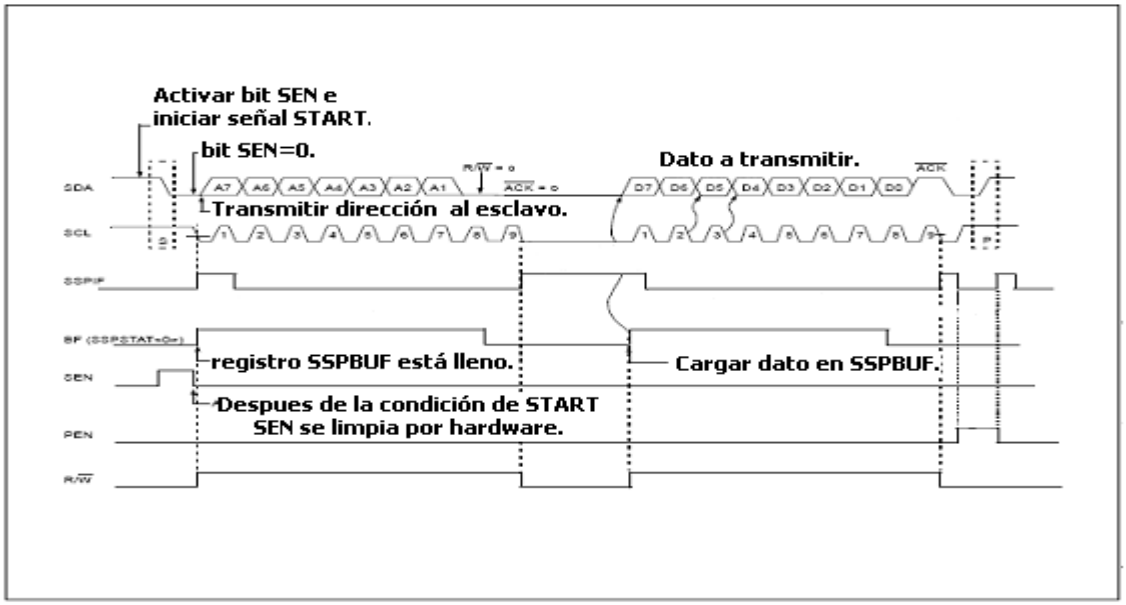


Fig.1.7.15.*Transmision de datos modo maestro I2C.*

1.8.3.2e MODO MAESTRO PARA RECEPCIÓN DE DATOS.

Para la utilización del dispositivo maestro en modo de recepción de datos desde un dispositivo esclavo es necesario activar por software el bit RCEN (SSPCON2<3>), la secuencia se realiza después de que el maestro haya generado la condición de START, después de haber enviado el byte de dirección al esclavo y de haber recibido la señal ACK por parte del esclavo.

Los datos a recibir son guiados por la señal de reloj en la línea SCL, al octavo flanco de bajada el bit RCEN se desactiva (vuelve a cero), con esto se habrá cargado el dato enviado desde el esclavo hacia el maestro en el registro SSPSR para luego cargarse al registro SSPBUF, activándose los bits BF y SSPIF que indican que un dato esta disponible para ser leído. El reloj se detendrá y la línea SCLK se queda a cero; al realizarse la lectura de SSPBUF el bit BF vuelve a cero por hardware, en este tiempo el maestro será el encargado de generar la señal ACK (sección 1.6.3.2b), para hacerle saber al esclavo que recibió el dato. Si se desea leer mas datos la secuencia es la misma iniciándose al activar el bit RCEN, al terminar la secuencia de recepción el maestro debe de generar la condición de STOP (sección 1.6.3.2c).

Consideraciones en la recepción de datos.

- El bit BF se pone a “1” cuando existe un dato en SSPBUF, al ser leído se pone a “0”.
- El bit WCOL se pone a “1” al escribir otro dato en el registro SSPBUF estando una recepción en curso.
- El bit SSPOV se pone a “1” cuando existe un dato completo en SSPSR y el bit BF esta aun activo.

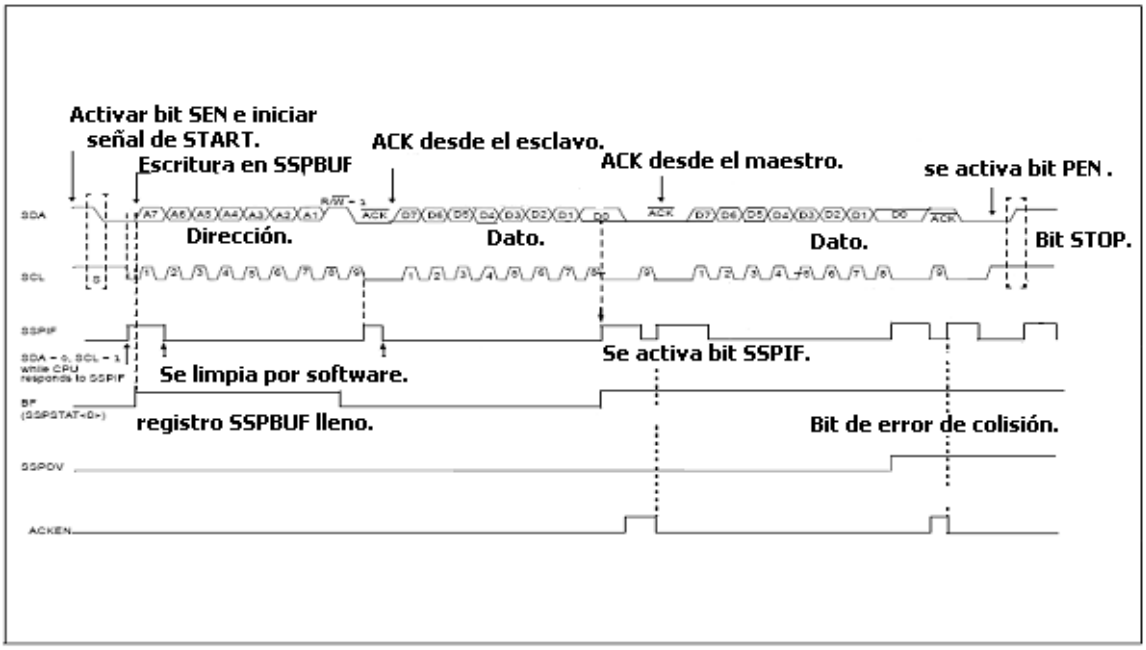


Fig.1.7.16.*Recepción de datos modo maestro I²C.*

1.8.3.2f MODO MULTIMAESTRO ARBITRAJES Y COLISIONES.

En este modo varios maestros pueden funcionar en el mismo bus I²C, pero debe existir un mecanismo de arbitrariedad logrando con esto que un solo maestro sea el encargado de las líneas SDA y SCL en una transferencia de datos.SDA puede cambiar cuando SCL se encuentra estable. Si un maestro coloca un “1” en su salida de datos en este intervalo y aparece un “0” en la línea SDA, se da una colisión en el bus I²C, como puede observarse en la figura 1.7.17.

La colisión genera que el bit BCLIF del maestro en cuestión se ponga a “1” y que se genere un reset en el bus I²C pasando éste al modo inactivo, hasta que el bus este limpio puede generarse otra secuencia de transferencia de datos.

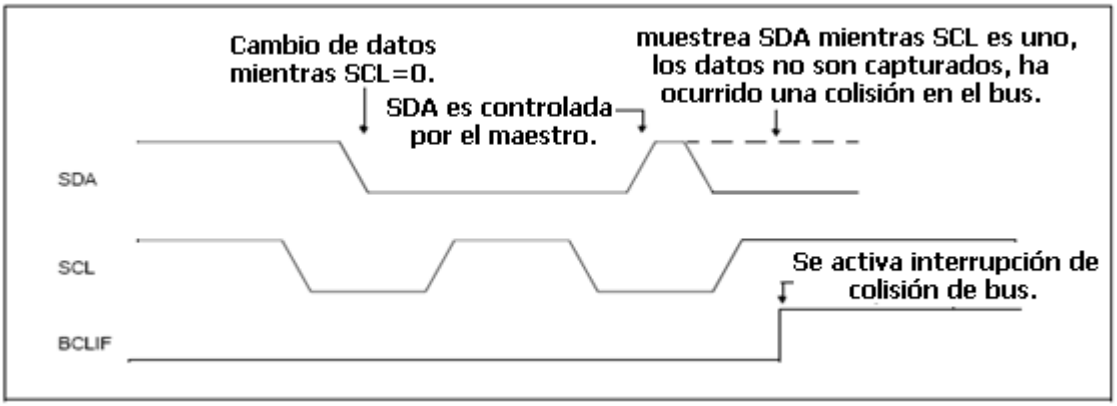


Fig.1.7.17.*Secuencia de colisión en el bus I²C.*

Capitulo Dos:

Programación Serial ICSP y depuración en los PIC16F877/A.

http://ww1.microchip.com/downloads/en/DeviceDoc/30277d.pdf
http://ww1.microchip.com/downloads/en/AppNotes/00656b.pdf

2.1 GENERALIDADES (ICSP).

ISP (In-System Programming) es una técnica donde un dispositivo es programado después de ser colocado en su tarjeta de circuito de aplicación. ICSP[9] es una técnica basada en ISP[10] implementada en los microcontroladores de Microchip como los son OTP[11] y MCU esta técnica utiliza solamente 2 pines de entrada/salida para realizar la programación, por lo que resulta ser una buena manera de ocupar menos espacio y menos recursos en el microcontrolador.

2.2 CONSIDERACIONES DE TIEMPO PARA LA PROGRAMACIÓN.

El tiempo de programación puede ser significativamente diferente al programar microcontroladores OTP y microcontroladores flash.

En los OTP (EPROM) los bytes son programados en el orden de cientos de microsegundos, por el otro extremo los flash requieren un tiempo que oscila entre los milisegundos ya sea para programar bytes o palabras. En la figura 2.1a se puede observar un diagrama donde se visualiza la diferencia de tiempos de programación, es de tomar en cuenta que la grafica se presenta de forma ideal en donde el tiempo gastado es puramente para la programación. La figura 2.1b es más realista ya que se toma un tiempo para realizar la programación que está entre 3 a 5 tiempos teóricos o ideales basado en pruebas con programadores disponibles en el mercado.

La diferencia entre los tiempos de programación resulta significante cuando se necesita programación en masa o cantidades, cabe mencionar que estos datos fueron tomados de la guía técnica de programación serial ICSP desarrollada por la compañía Microchip.

---

[9] **ICSP** Programación serial en circuito.
[10] **ISP** Programación serial en sistema.
[11] **OTP** Dispositivos programables solo una vez.

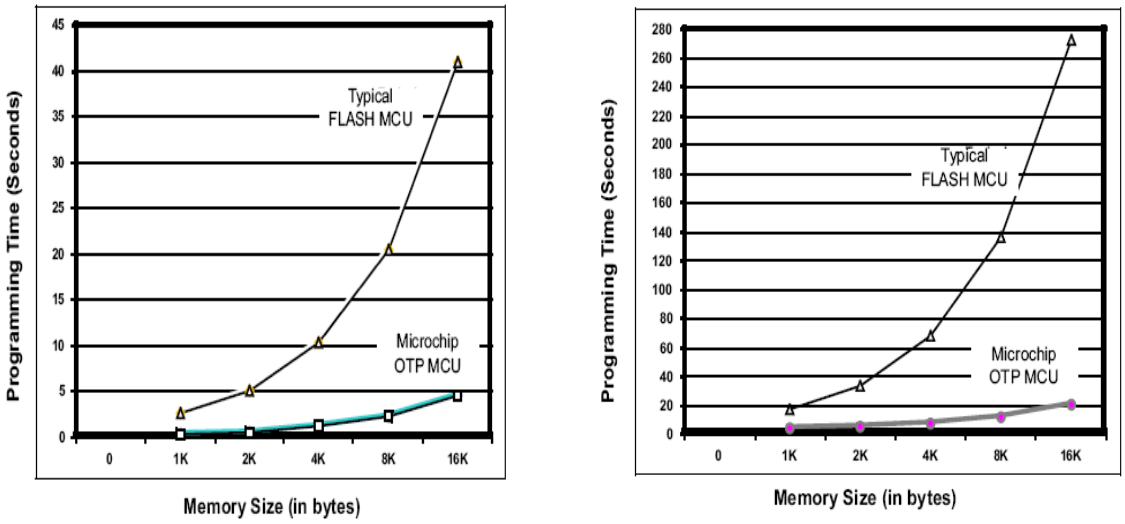


Fig.2.1 (a, b).*Graficas de tiempo en la programación ICSP*.

2.3 PROGRAMACIÓN SERIAL ICSP EN LOS PIC 16F877 /A.

Tanto el microcontrolador 16F877 como el 16F877A tienen la ventaja de ser programados de manera serial usando la técnica ICSP, ofreciendo con esto la ventaja de estar colocados en una tarjeta de aplicaciones, es decir de no tener la necesidad de programarlos fuera del hardware en donde serán utilizados. Existen 2 métodos para realizar la programación en los microcontroladores que son *HVP (high Voltaje Programming) y LVP (low Voltaje programming).*

El método HVP es el más utilizado, pero necesita la aplicación de un voltaje de 13.5V al pin MCLR/VPP, el microcontrolador está siempre alimentado por 5V.

El método LVP de programación ICSP permite programar al PIC usando solamente 5V en el pin MCLR/VPP, además de utilizar el pin PGM (RB3) para entrar a este modo de programación. La secuencia a seguir es:

Poner a “1” el bit de configuración LVP, poner a “1” el pin RB3 y colocar Vdd (5V) al pin MCLR/Vpp para iniciar la programación. Si el bit LVP esta a cero el pin RB3 es un pin I/O normal y el modo de programación es el HVP.

ICSP permite programar la memoria de programa con el código deseado, memoria de datos, localidades especiales usadas para ID y palabra de configuración en los micros a utilizar. En la tabla 2.1 se observan los distintos pines que intervienen en la programación del microcontrolador utilizando la programación serial ICSP.

DESCRIPCION DE PINES : PIC16F87X

Nombre del pin.	DURANTE LA PROGRAMACION.		
	Funcion.	Pin Tipo	Descripcion.
RB3	PGM	I	Entrada de voltaje ICSP para programacion si bit LVP esta activo.
RB6	CLOCK	I	Reloj de entrada.
RB7	DATA	I/O	entrada /salida de datos.
$\overline{\text{MCLR}}$	$V_{TESTMODE}$	P*	selecciona modo de programacion.
V_{DD}	V_{DD}	P	Fuente de voltaje.
V_{SS}	V_{SS}	P	Tierra.

Tabla 2.1 *Descripción de pines utilizados en modo ICSP*.

2.4 MAPA DE MEMORIA DEL PROGRAMA DE USUARIO.

Para los PICs 16F877 y 16F877A la memoria de usuario esta comprendida desde la dirección 0x0000 hasta la dirección 0x1FFF lo que constituye un espacio de 8K Bytes, en modo de programación el espacio total es de la dirección 0x0000 hasta la dirección 0x3FFF, siendo la segunda parte (0x2000 hasta la 0x3FFF)la memoria de configuración , en este espacio el rango de direcciones 0x2000 a 0x200F están físicamente implementadas ,sin embargo solo el rango desde 0x2000 a 0x2007 están disponibles ,las restantes están reservadas, en la figura 2.2 puede verse el mapeo mas detallado.

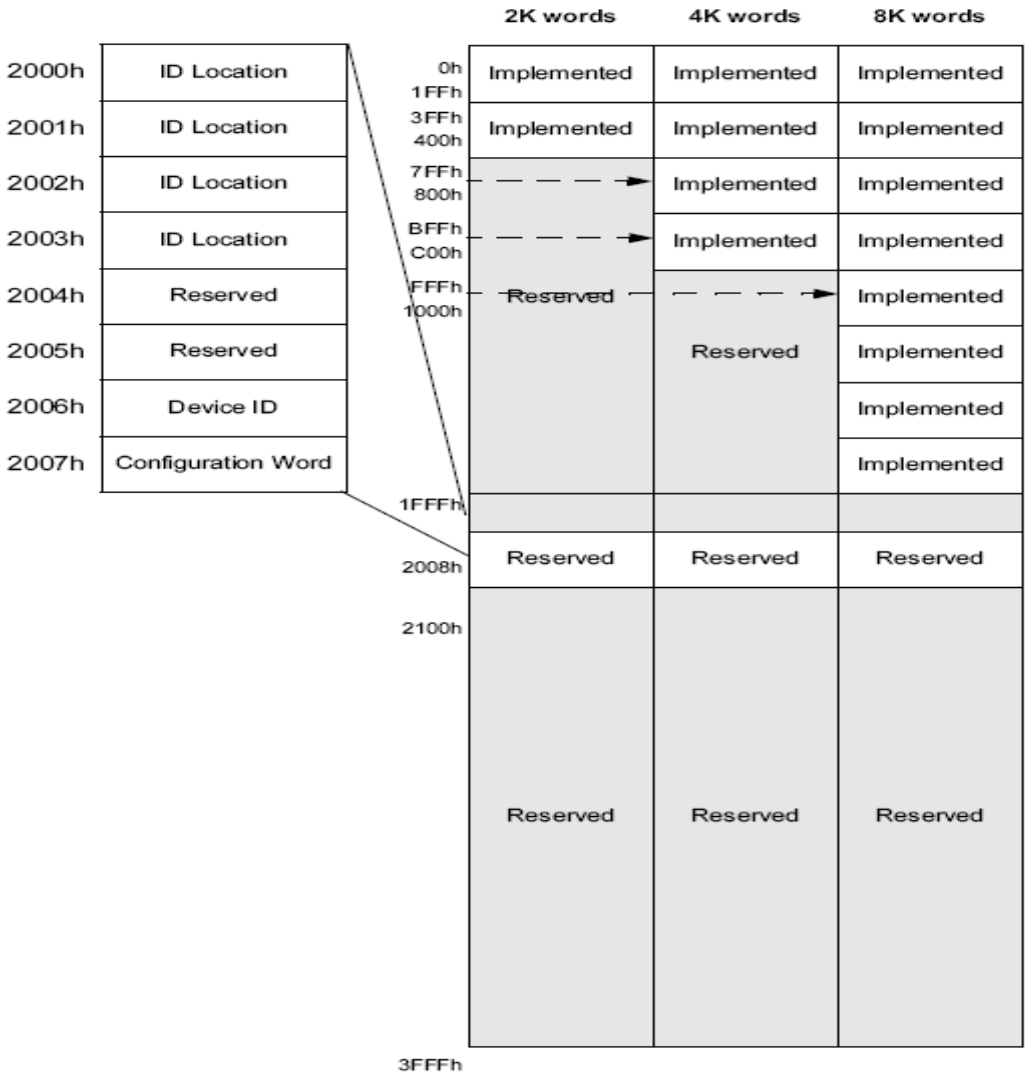


Fig. 2.2 *Mapa de memoria utilizado en modo ICSP.*

2.4.1 LOCALIDADES DE IDENTIFICACIÓN (ID LOCATION).

El usuario puede guardar información en 4 localidades dispuestas para el código de identificación (ID), estas localidades están mapeadas en el rango de direcciones 0x2000 a la 0x2003, como se observa en la figura 2.2 .Se recomienda que el usuario utilice solo los 4 bits menos significativos de cada byte ID.

2.5 MODO DE PROGRAMACIÓN Y VERIFICACIÓN HVP.

Para entrar al modo de programación /verificación los pines PGD y PGC se mantienen en bajo, mientras que el voltaje en el pin MCLR/VPP se incrementa desde VIL (0V) a VIHH (13.5V). Después de un reset el contador de programa se sitúa en la dirección 00h (Para evitar un reset cuando se esta en el modo de programación es necesario desactivar el wacthdog timer (WDT) que posee el microcontrolador). El pin RB6 es el utilizado como entrada de reloj, el pin RB7 es el encargado de leer desde el MCU y escribir datos o comandos hacia el MCU de manera serial.

La secuencia normal de programación/verificación consiste en usar el comando para cargar un dato en una dirección específica, luego verificar leyendo el dato guardado e incrementar la dirección para programar un bloque de memoria. Al existir un reset el contador de programa se coloca en la dirección 00h y al incrementar direcciones se esta incrementando el contador de programa, para configurar el dispositivo se utiliza específicamente el comando para cargar la palabra de configuración y automáticamente el contador de programa se colocara en la dirección 0x2000. Para escribir un comando se necesitan 6 ciclos de reloj obtenidos en el pin PGC (RB6),cada bit del comando es enganchado en el flanco de bajada de cada ciclo de reloj, (iniciando con el bit menos significativo(LSB)),los comandos que tienen datos asociados a ellos como lectura y carga de datos necesitan un tiempo mínimo de retraso de 1us entre el comando y el dato, después de este retraso el reloj ofrece 16 ciclos de reloj con el primer ciclo siendo el bit de START y el ultimo ciclo siendo el bit de STOP ,el orden del dato inicia con el bit menos significativo (LSB).Durante una operación de lectura el bit LSB será transmitido por el pin RB7 en el flanco positivo del segundo ciclo, durante una operación de escritura el bit LSB será cargado en el flanco de bajada de cada ciclo de reloj.

2.6 PALABRA DE IDENTIFICACIÓN DEL DISPOSITIVO (ID).

Cada dispositivo contiene una palabra de identificación ya definida, conocida como ID, la cual se encuentra en la dirección 0x2006. La tabla 2.2 muestra diferentes ID para diferentes dispositivos.

Dispositivo.	Valor ID del dispositivo.	
	Dev	Rev
PIC16F870	00 1101 000	x xxxx
PIC16F871	00 1101 001	x xxxx
PIC16F872	00 1000 111	x xxxx
PIC16F873	00 1001 011	x xxxx
PIC16F874	00 1001 001	x xxxx
PIC16F876	00 1001 111	x xxxx
PIC16F877	00 1001 101	x xxxx

Tabla 2.2 *Diferentes valores de identificación*.

2.7 PALABRA DE CONFIGURACIÓN.

Los microcontroladores PIC 16F877/A poseen diversos bits de configuración que están Incluidos en la palabra de configuración, estos bits son:

- CPx Bits para protección de código en memoria de programa.
- DEBUG Bit de activación o desactivación para realizar depuración.
- WRT Bit de activación para escritura en memoria de programa.
- CPD Bit de activación de protección de código en memoria EEPROM.
- LVP Bit de activación para programación con bajo voltaje.
- BODEN bit de activación de reset BROWN –OUT.
- PWRT Bit de activación de reset por alimentación del dispositivo.
- WDTE Bit de activación del pero guardián o watchdog timer.
- FOSCx Bits para realizar la selección del tipo de oscilador a utilizar.

La palabra de configuración se muestra explícitamente en la figura 2.3.

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/P-1	U-0	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1
CP1	CP0	RESV	—	WRT	CPD	LVP	BODEN	CP1	CP0	$\overline{\text{PWRTE}}$	WDTE	F0SC1	F0SC0
bit 13													bit 0

bit 13-12, bit 5-4 **CP1:CP0:** FLASH Program Memory Code Protection bits(2)

4 K Devices:
11 = Code protection off
10 = 0F00h to 0FFFh code protected
01 = 0800h to 0FFFh code protected
00 = 0000h to 0FFFh code protected

8 K Devices:
11 = Code protection off
10 = 1F00h to 1FFFh code protected
01 = 1000h to 1FFFh code protected
00 = 0000h to 1FFFh code protected

bit 11 **Reserved:** Set to '1' for normal operation

bit 10 **Unimplemented:** Read as '1'

bit 9 **WRT:** FLASH Program Memory Write Enable bit
1 = Unprotected program memory may be written to by EECON control
0 = Unprotected program memory may not be written to by EECON control

bit 8 **CPD:** Data EE Memory Code Protection bit
1 = Code protection off
0 = Data EE memory code protected

bit 7 **LVP:** Low Voltage ICSP Programming Enable bit
1 = RB3/PGM pin has PGM function, low voltage programming enabled
0 = RB3 is digital I/O, HV on $\overline{\text{MCLR}}$ must be used for programming

bit 6 **BODEN:** Brown-out Reset Enable bit(2)
1 = BOR enabled
0 = BOR disabled

bit 3 **$\overline{\text{PWRTE}}$:** Power-up Timer Enable bit
1 = PWRT disabled
0 = PWRT enabled

bit 2 **WDTE:** Watchdog Timer Enable bit
1 = WDT enabled
0 = WDT disabled

bit 1-0 **FOSC1:FOSC0:** Oscillator Selection bits
11 = RC oscillator
10 = HS oscillator
01 = XT oscillator
00 = LP oscillator

Note 1: Enabling Brown-out Reset automatically enables Power-up Timer (PWRT), regardless of the value of bit $\overline{\text{PWRTE}}$. Ensure the Power-up Timer is enabled any time Brown-out Reset is enabled.
2: All of the CP1:CP0 pairs have to be given the same value to enable the code protection scheme listed.

Fig. 2.3 *Palabra de configuración PIC16F877/A (0x2007).*

2.7.1 CARGAR CONFIGURACIÓN POR MEDIO DE LA DIRECTIVA _CONFIG.

La capacidad de poder alterar los bits de la palabra de configuración del microcontrolador , a través de la directiva _CONFIG anexada en el código de usuario que se desea programar al dispositivo ,es una ventaja para poder corroborar que la palabra de configuración sea la deseada para el tipo de aplicación que se desea hacer con el microcontrolador. En la tabla 2.3 se observan los símbolos que se utilizan en el código fuente para alterar la palabra de configuración.

Simbolos de la directiva _CONFIG.

Función	Simbolos.
Osciladores	_RC_OSC
	_EXTRC_OSC
	_EXTRC_OSC_CLKOUT
	_EXTRC_OSC_NOCLKOUT
	_INTRC_OSC
	_INTRC_OSC_CLKOUT
	_INTRC_OSC_NOCLKOUT
	_LP_OSC
	_XT_OSC
	_HS_OSC
Temporizador perro guardian.	_WDT_ON
	_WDT_OFF
Temporizador de alimentación.	_PWRTE_ON
	_PWRTE_OFF
Reset por Brown _out	_BODEN_ON
	_BODEN_OFF
activar limpieza del maestro.	_MCLRE_ON
	_MCLRE_OFF
Protección de código.	_CP_ALL
	_CP_ON
	_CP_75
	_CP_50
	_CP_OFF
Protección de còdigo en EEPROM.	_DP_ON
	_DP_OFF
espacio para protección de código.	_CPC_ON
	_CPC_OFF

Tabla. 2.3 *Símbolos utilizados por la directiva _config.*

La secuencia de uso de la directiva _CONFIG dentro del código fuente se muestra en la figura 2.4 (es aplicable para cualquier microcontrolador).

```
      LIST    p = p16C77          ; Directiva de lista.

;
      #INCLUDE      <P16C77.INC>      ; Archivo de cabecera de dispositivo a usar.
;
      #INCLUDE      <MY_STD.MAC>      ; Archivo incluye macros estandard.
      #INCLUDE      <APP.MAC>         ; además de macros especificos
                                      ; de la aplicación.
;
;  Especifica bits de configuración del dispositivo.
;
      __CONFIG      _XT_OSC & _PWRTE_ON & _BODEN_OFF & _CP_OFF & _WDT_ON
;
      org    0x00                   ; Inicio del programa
RESET_ADDR    :                     ; Primera instrucción despues del reset.


      end                           Fin del programa.
```

Fig.2.4 *Ejemplo de utilización directiva _config.*

2.7.2 PROGRAMACIÓN Y VERIFICACIÓN UTILIZANDO COMANDOS.

Comando	Mapeo (MSB ... LSB)						Dato	Rango de voltaje.
cargar configuración.	X	X	0	0	0	0	0, data (14), 0	2.2V - 5.5V
Cargar dato a memoria de programa.	X	X	0	0	1	0	0, data (14), 0	2.2V - 5.5V
Leer dato de memoria de programa.	X	X	0	1	0	0	0, data (14), 0	2.2V - 5.5V
Incrementar dirección.	X	X	0	1	1	0		2.2V - 5.5V
Iniciar ciclo borrado-programado.	0	0	1	0	0	0		2.2V - 5.5V
Iniciar ciclo de programación.	0	1	1	0	0	0		4.5V - 5.5V
Cargar dato a memoria de datos.	X	X	0	0	1	1	0, data (14), 0	2.2V - 5.5V
Leer dato de memoria de datos.	X	X	0	1	0	1	0, data (14), 0	2.2V - 5.5V
Borrar pila 1.	0	0	0	0	0	1		4.5V - 5.5V
Borrar pila 2.	0	0	0	1	1	1		4.5V - 5.5V

Tabla 2.4 *Comandos utilizados en la programación/verificación modo ICSP.*

2.7.2.1 CARGAR CONFIGURACIÓN UTILIZANDO COMANDO.

Al recibir este comando el contador de programa se colocara en la dirección 0x2000, se aplicaran 16 ciclos de reloj para cargar el dato de 14 bits de la palabra de configuración en la memoria de configuración, después de haber accesado a la memoria de configuración la única manera de regresar a la memoria de usuario es la de salirse del modo de programación y verificación al aplicar 0V (VIL) al pin MCLR.

2.7.2.2 COMANDO PARA CARGAR DATOS EN MEMORIA DE PROGRAMA.

La secuencia de este comando consiste en 6 ciclos de reloj para el comando, 16 ciclos de reloj para el dato de 14 bits a guardar (START, datos, STOP). En el diagrama de tiempo de la figura 2.5 se muestra con más detalle.

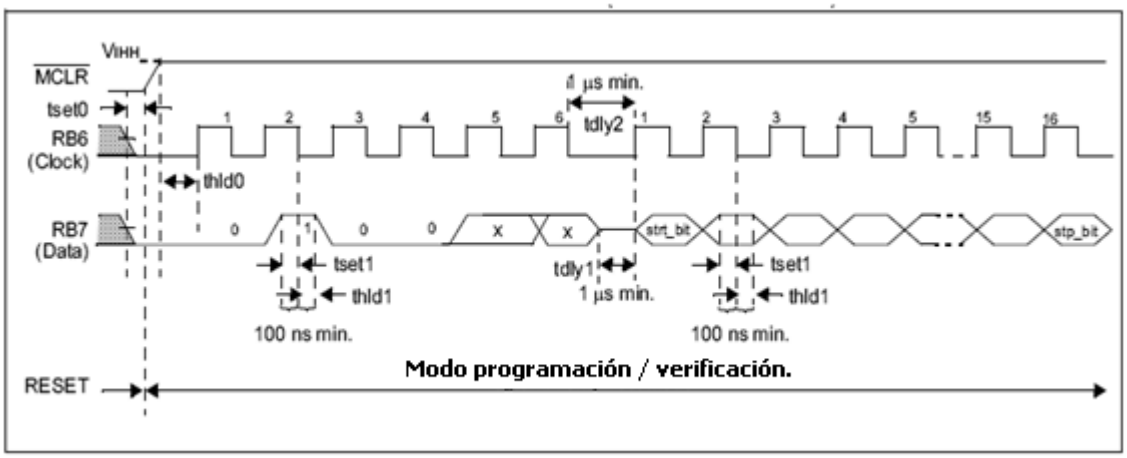


Fig.2.5 *Secuencia del comando para cargar datos (MCLR=Vihh).*

2.7.2.3 COMANDO PARA CARGAR DATOS EN MEMORIA DE DATOS.

La secuencia de este comando es similar en cuanto a bits de datos y ciclos de reloj a la secuencia de cargar datos en la memoria de programa, la diferencia radica en que la memoria de datos solamente posee campo para 8 bits (256 bytes), por eso solo los primeros 8 bits que siguen al bit de START serán programados en la memoria de datos, aunque para que las se de la operación con normalidad son necesarios los 16 ciclos de reloj.

2.7.2.4 COMANDO PARA LECTURA DE DATOS DESDE LA MEMORIA DE PROGRAMA.

Al recibir este comando el MCU iniciara la transmisión de los datos desde la memoria de programa, ya sean de datos o configuración, esto se realiza de manera ordenada. La secuencia inicia después de recibir el comando (primeros 6 ciclos de reloj), los datos son transmitidos a través de el pin RB7 que se comporta como salida, iniciando en el segundo flanco de subida del reloj (16 ciclos restantes), luego RB7 se comportara como entrada con alta impedancia en el ultimo flanco de subida de los 16 ciclos de reloj. En el diagrama de tiempo de la figura 2.6 se observa con más detalle.

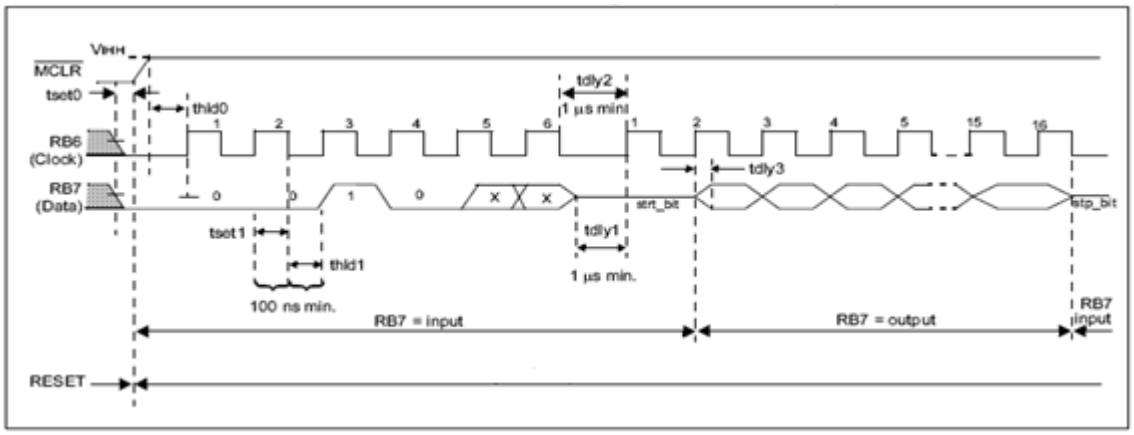


Fig.2.6.*Secuencia del comando para lectura de datos (MCLR=Vihh).*

2.7.2.5 COMANDO PARA LECTURA DE DATOS DESDE LA MEMORIA DE DATOS.

La secuencia de este comando es similar a la que se realiza en lectura de datos desde memoria de usuario, siguiendo el pin RB7 el mismo comportamiento en los mismos ciclos de reloj, se diferencia en que solamente lo primeros 8 bits son validos por el espacio disponible en la memoria de datos.

2.7.2.6 COMANDO PARA INCREMENTAR DIRECCION.

El contador de programa es incrementado cada vez que se recibe este comando siguiendo el diagrama de tiempo de la figura 2.7.

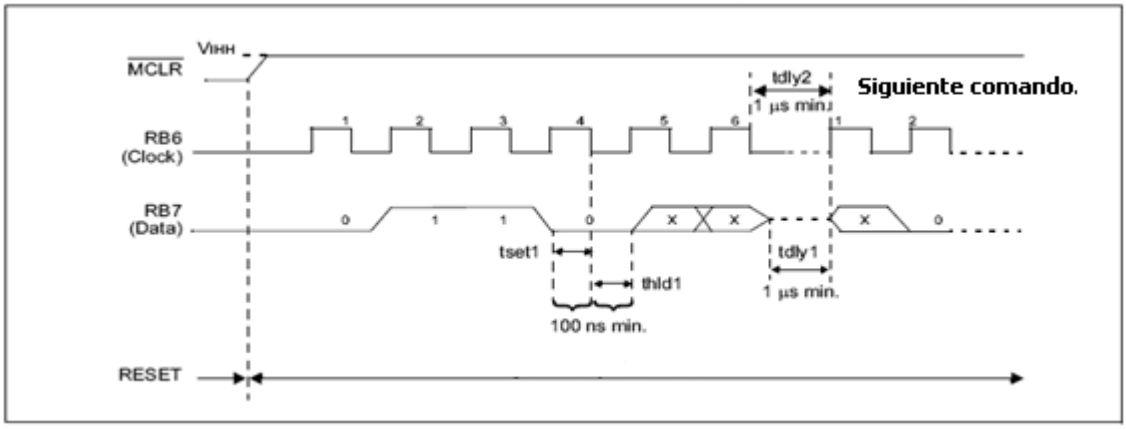


Fig.2.7.*Secuencia del comando para incrementar dirección (MCLR=Vihh).*

2.7.2.7 COMANDO PARA CICLO DE BORRADO / PROGRAMADO.

Siempre un comando de carga debe ser desarrollado antes de un comando de programación. Inmediatamente después de que el comando de carga sea recibido se desarrollara la programación en el espacio de memoria a utilizar (usuario, datos, configuración).Un mecanismo interno es desarrollado para el borrado antes de cada escritura, no se necesita un comando que indique la finalización de la programación.

2.7.2.8 COMANDO PARA INICIO DE LA PROGRAMACIÓN.

Esta operación debe ser desarrollada dentro del rango d voltaje comprendido entre 4.5 V y 5.5V, se necesita aplicar siempre un comando de carga de datos antes de utilizar el comando de programación .Al recibir este comando un mecanismo interno de tiempo se desarrolla para la escritura, es parecido al comando de borrado/programado pro solo realiza la etapa de escritura.

2.7.3 BORRADO DE LA MEMORIA DE PROGRAMA Y MEMORIA DE DATOS.

Dependiendo del estado del bit de protección de código, el borrado de la memoria ya sea de programa o de datos se realiza de diferente manera, cuando no existe código protegido en ambas memorias o cuando ambas memorias están protegidas.

2.7.3.1 BORRAR MEMORIA DE DATOS Y DE USUARIO SIN PROTECCIÓN DE CÓDIGO.

Cuando no existe protección de código en ambas memorias, están deben ser borradas individualmente siguiendo el procedimiento propuesto.

Consideraciones:

- Si una de las dos memorias tiene protección de código puede realizarse un solo procedimiento para el borrado del código.
- Al desarrollar estos procedimientos no se borra la palabra de configuración y ni el ID, sino que debe reprogramarse para activar o desactivar la protección de código.

PROCEDIMIENTO PARA BORRAR LA MEMORIA DE PROGRAMA.

- Ejecutar un comando de carga de datos a memoria de programa escribiendo "1" en todas las localidades (0x3FFF) (000010).
- Ejecutar comando de borrado de pila setup1 (000001).
- Ejecutar comando de borrado de pila setup2 (000111).
- Ejecutar comando de borrado programado (001000).
- Esperar 8 milisegundos.
- Ejecutar comando de borrado de pila setup1 (000001).
- Ejecutar comando de borrado de pila setup2 (000111).

PROCEDIMIENTO PARA BORRADO DE MEMORIA DE DATOS.

- Ejecutar comando para cargar datos a memoria de datos colacando"1" en cada localidad (0x3FFF) (000011).
- Ejecutar comando de borrado de pila setup1 (000001).
- Ejecutar comando de borrado de pila setup2 (000111).
- Ejecutar comando de borrado programado (001000).
- Esperar 8msegundos.
- Ejecutar comando de borrado de pila setup1 (000001).
- Ejecutar comando de borrado de pila setup2 (000111).

2.7.3.2 BORRAR MEMORIA DE USUARIO O DE DATOS CON PROTECCIÓN DE CÓDIGO.

Para los PICs 16F877/A cuando se activa la protección de código en los campos de memorias la lectura ofrece solamente ceros en cualquier localidad de memoria leída. El procedimiento para borrar el dispositivo que posee código protegido es el siguiente (éste desarrollo borra tanto la memoria de usuario, de datos y los bits de configuración además de las localidades ID).

- Ejecutar comando de carga de configuración (000000) colocando “1” en cada localidad (0x3FFF).
- Ejecutar comando de incremento de dirección (000110) para colocar el contador en la localidad 0x2007 que contiene la palabra de configuración.
- Ejecutar comando de borrado de pila setup1 (000001).
- Ejecutar comando de borrado de pila setup2 (000111).
- Ejecutar comando de borrado programado (001000).
- Esperar 8msegundos.
- Ejecutar comando de borrado de pila setup1 (000001).
- Ejecutar comando de borrado de pila setup2 (000111).

2.7.4 CIRCUITO DE APLICACIÓN PARA DESARROLLAR ICSP EN LOS PIC16F877/A.

El circuito de aplicación debe ser diseñado para permitir que todas las señales de programación sean directamente conectadas al microcontrolador a programar serialmente. Se muestra un diagrama típico para realizar ICSP en un determinado MCU, sucede lo mismo para diferentes MCUs que soporten programación ICSP.

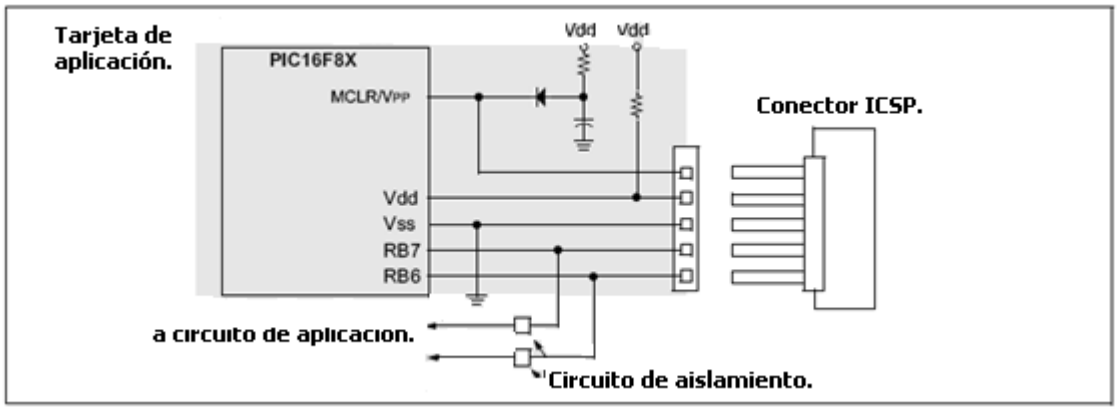


Fig.2.8 *Circuito para realizar programación ICSP.*

El pin MCLR/VPP normalmente esta conectado a un circuito RC, que comprende de un resistor conectado a VDD y de un capacitor conectado a tierra. Este circuito puede afectar el funcionamiento de la programación, dependiendo del tamaño del capacitor, sobre todo si el voltaje VPP debe ser aislado del resto del circuito de aplicación ya que su nivel de voltaje es de 13.5V y el microcontrolador solo maneja 5.5V para su funcionamiento (normalmente una sola resistencia no es capaz de aislar este voltaje), por esa razón se utiliza el circuito RC anexándole un diodo que debe ser diodo schottky para asegurar el aislamiento y protección.

Los pines RB7 (PGD) y RB6 (PGC) son utilizados para la programación serial, RB6 es el reloj para la programación (manejado por el programador), RB7 se comporta como un pin bidireccional siendo manejado por el programador cuando escribe y por el microcontrolador cuando verifica código. Estos pines deben ser aislados del resto del circuito de aplicación para que las señales no se vean afectadas durante la programación (el diseño del aislamiento depende de la aplicación por eso no se muestra en el diagrama), también se debe tomar en cuenta la impedancia de salida del circuito programador.

Recomendaciones para desarrollar el circuito de programación en modo ICSP.

- Es recomendable que se utilice una fuente externa que ofrezca los suficientes niveles de voltaje y nivel de corriente para poder alimentar el circuito de aplicación, además de usar los pines de programación (RB6, RB7) exclusivamente para programar.

- La verificación del código debe de realizarse dentro del rango definido de voltaje (2.7V - 4.5V), el circuito programador debe ser capaz de programar a través de PGD usando 5V y de verificar tanto a 2.7V como a 4.5V para asegurarse que el código no se vera afectado si el micro es alimentado dentro de este rango definido.

- El oscilador en el circuito de aplicación juega un papel importante, el voltaje VPP debe ser colocado en el pin MCLR antes de que el dispositivo ejecute cualquier código ,en el caso de utilizar un oscilador RC debe de tomarse en cuenta que se debe de entrar al modo de programación antes de que el oscilador RC haya ejecutado 4 ciclos ,si se dan 4 o mas ciclos el contador de programa pude incrementarse y obtenerse un offset , en otras palabras la programación del código no se hará a partir de la dirección 00h sino desde 00h mas el offset.

2.8 GENERALIDADES (DEBUGGER ON –CHIP).

http://ww1.microchip.com/downloads/en/DeviceDoc/51242a.pdf

Un sistema debugger –On-Chip es un especial hardware y software que trabaja con un especifico microcontrolador, brindando un método de bajo costo para realizar la depuración de código. El sistema permite colocar puntos de ruptura, ejecutar paso a paso y detener la ejecución del programa, todo esto para visualizar el estado interno del microcontrolador durante una ejecución en tiempo real ya instalado en el circuito de aplicación.

2.9 REGISTROS ASOCIADOS CON LA DEPURACIÓN.

Existen 2 registros internos que poseen los microcontroladores PIC 16F877/A asociados con el control del debug, los cuales son ICKBUG y BIGBUG. Entre estos 2 registros existen 3 bits que son exclusivamente para el control de la depuración, los restantes 13 bits (BKA12-BKA00) son para direccionamiento de los breakpoints (puntos de parada).

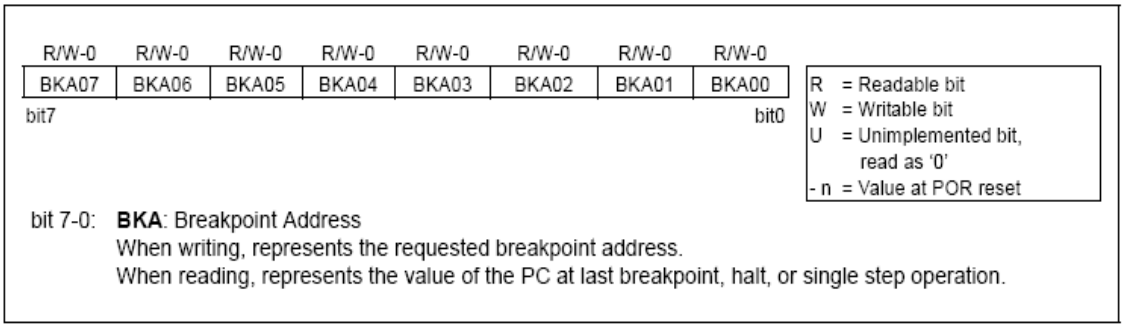


Fig.2.9.*Registro BIGBUG (0x18).*

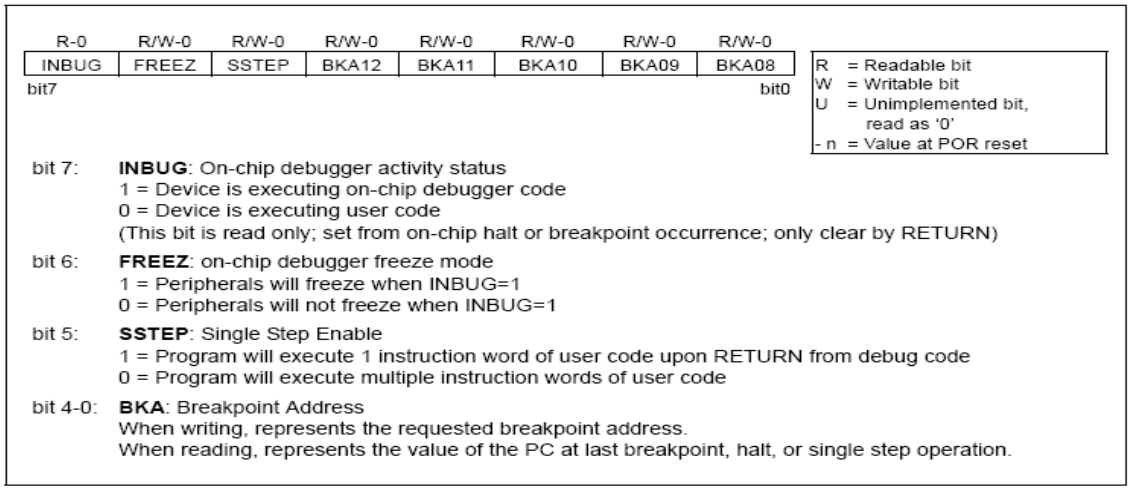


Fig.2.10. Registro ICKBUG.

BIT INBUG: Bit que indica con un "1" cuando el MCU está en modo debug y se lee "0" cuando se sale del modo debug o se esta ejecutando código de usuario. En modo debug las interrupciones son desactivadas al igual que la opción debugger HALT.

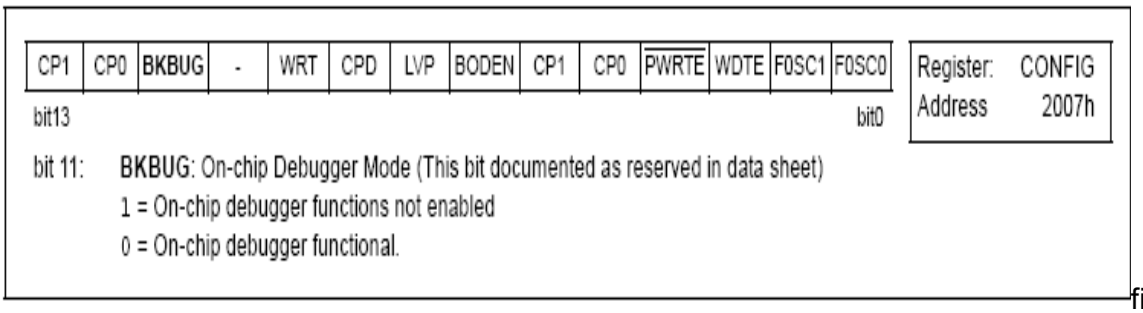
BIT FREEZ: Cuando se coloca un "1" en este bit se congelan los timers TMR0,TMR1,TMR2 y sus pre escalas suspendiendo el conteo ,los sistemas SSP ,A/D, USART se suspenderán también, las banderas de estado que son alterados por los registros no se ven afectadas por el bit FREEZ.

BIT SSTEP: Se tiene la opción de ejecutar un único paso a través del código al escribir un "1" en este bit.

BITS BKAxx: Son 13 bits que contienen la dirección en donde se realizará el breakpoint o punto de ruptura deteniéndose la ejecución del código. Estos bits son limpiados únicamente al realizar un POR (reset por alimentación), ningún otro reset puede limpiar estos bits.

2.10 DEPURACIÓN EN LOS MICROCONTROLADORES PIC 16F877 /A.

En la palabra de configuración, la cual esta mapeada en la dirección 0x2007, se encuentran los bits para activar y desactivar la función de debug en el micro controlador, siendo el bit BKBUG el encargado de habilitar o deshabilitar la función de debug en el dispositivo.



fiF

Fig.2.11. *Palabra de configuración.*

Para la depuración en los microcontroladores 16F877/A se hace uso de hardware para establecer conexión con un determinado software, el mas utilizado es el software MPLAB-IDE del cual obtenemos ventajas a la hora de programar y depurar código. Debido a que este software ofrece la opción de poder cargar la palabra de configuración desde su entorno activando el bit para realizar debug. En el proceso de programación se carga en la memoria del MCU el código a ejecutar, junto con un fragmento de programa que es conocido como debug executive o sistema operativo (desde el entorno del MPLAB-IDE), este fragmento de código es el encargado de realizar la depuración ocupando las últimas 256 posiciones de la memoria de programa. Debido a que este segmento de código es una aplicación utiliza ciertos recursos hardware y software del microcontrolador:

- Pin MCLR/VPP.
- Pines RB6 (PGC) y RB7 (PGD) para la transferencia serie.
- Un nivel de la pila Hardware.
- Las ultimas 256 posiciones de memoria de programa (0x1F00 – 0x1FFF).
- 15 registros RAM cuyas direcciones son: 0x70, 0xF0, 0x170, 0x1F0, 0x1E5-0x1EF.

El sistema operativo o mejor conocido como debug executive, es el encargado de establecer la comunicación entre el microcontrolador y el software MPLAB IDE a través de los pines RB6 y RB7, para transferir el contenido de los registros internos para su visualización en la PC.

Si el bit BKBUG de la palabra de configuración es programado con cero “0” (activado), el bit INBUG es cero “0” al inicio indicando que aun no se ha entrado al modo debug y se esta ejecutando el código de usuario, si una señal de parada al código de usuario ocurre, la ejecución normal del programa se para y se entra al modo on-chip debugger que se indica cuando el bit INBUG =”1” (activo).

Para detener la ejecución del programa de usuario existen diferentes maneras:

- **Una transición de alto a bajo en el pin RB6.**
  Para detener la secuencia del código de usuario se debe realizar un cambio de nivel de alto a bajo en el pin RB6, el circuito para programar generará una señal que detendrá la ejecución del programa, pasando a realizar la rutina de depuración. Esto es conocido como parada por hardware.

- **Utilizando breakpoint**.
  La parada por breakpoint es la que se realiza colocando una dirección en los 13 bits presentes en los registros ICKBUG e BIGBUG, el programa normal se detendrá cuando se de una compatibilidad entre la dirección presente en estos registros y la dirección de la instrucción que se está ejecutando. Cuando se realiza un reset por alimentación estos bits conteniendo la dirección del breakpoint quedan como ceros.

- **Activando el bit SSTEP (ejecución de un solo paso).**
  Activando el bit SSTEP se realiza la parada por ejecución de un único paso, el sistema ingresa al modo debug y genera una señal para permitir una sola instrucción.

Cuando la ejecución normal del programa de usuario se detiene, la instrucción presente será ejecutada y el programa se detendrá en la dirección de la siguiente instrucción. Entonces se realiza un ciclo de interrupción solo que el bit GIE (habilitación de interrupciones) permanece intacto. Una vez se pasa a modo debug la bandera INBUG del registro ICKBUG es activada indicando que se esta en el modo debug, el contador de programa es guardado en la cima de la pila hardware y colocado en los registros ICKBUG y BIGBUG, con esto se puede obtener el estado del contador del programa (siguiente instrucción a ser ejecutada).entonces el contador de programa se cargara con la dirección 0x2004 (que no esta en zona de código de usuario), desarrollando entonces la instrucción contenida en esta dirección , por lo que es necesario colocar una instrucción de salto como la que se muestra hacia la dirección en donde el código del debug executive inicia :

GOTO 0x1F00.

El software del debug realizará una instrucción de retorno (RETURN) para regresar al código principal que se esta ejecutando, entonces el bit INBUG será limpiado indicando que se ha salido del modo debug , en el caso que se haya utilizado el bit FREEZ para congelar los timers este se desactivará y los timers podrán trabajar nuevamente.

La secuencia completa para realizar debug esta comprendida por la etapa de programación / verificación y la etapa de depuración, en el diagrama de la figura 2.12 se observa que durante la programación serial se desarrolla la programación y verificación, después de un reset se inicia la ejecución del código programado, al detectarse una parada (HALT) se entra al modo de depuración generándose la comunicación con el software, luego regresa a la ejecución normal de código.

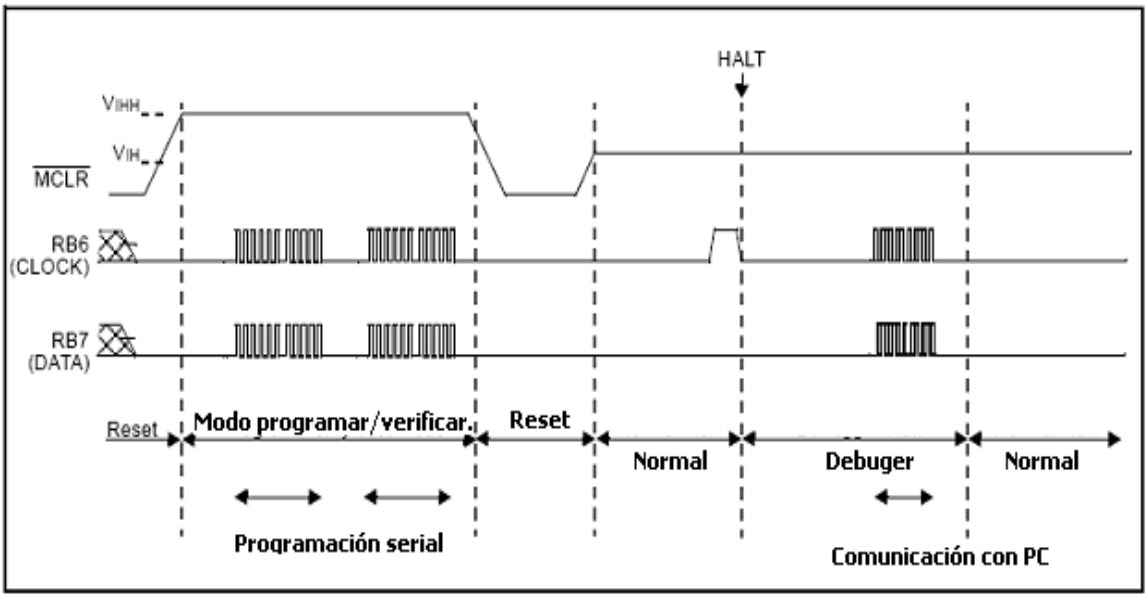


Fig.2.12 *Secuencia completa de ejecución modo debug.*

Capitulo Tres: Sistema Debug y Hardware de aplicaciones.

3.1 GENERALIDADES

El sistema para entrenamiento basado en microntroladores PIC 16F877/A , consiste en diferentes módulos para el aprovechamiento de los distintos subsistemas contenidos en los PIC 16F877/A: tarjeta debug, tarjeta principal conteniendo el MCU a depurar, tarjetas de entrada/salida digital, tarjeta de potencia para control de relés ó motor paso, tarjeta para comunicación con dispositivos y con otro microcontrolador utilizando los protocolos I²C y SPI . Por ser modular, el sistema queda abierto a desarrollar otras tarjetas para realizar otras prácticas utilizando otros periféricos. En el diagrama general de bloques de la figura 3.1 se observa cada etapa.

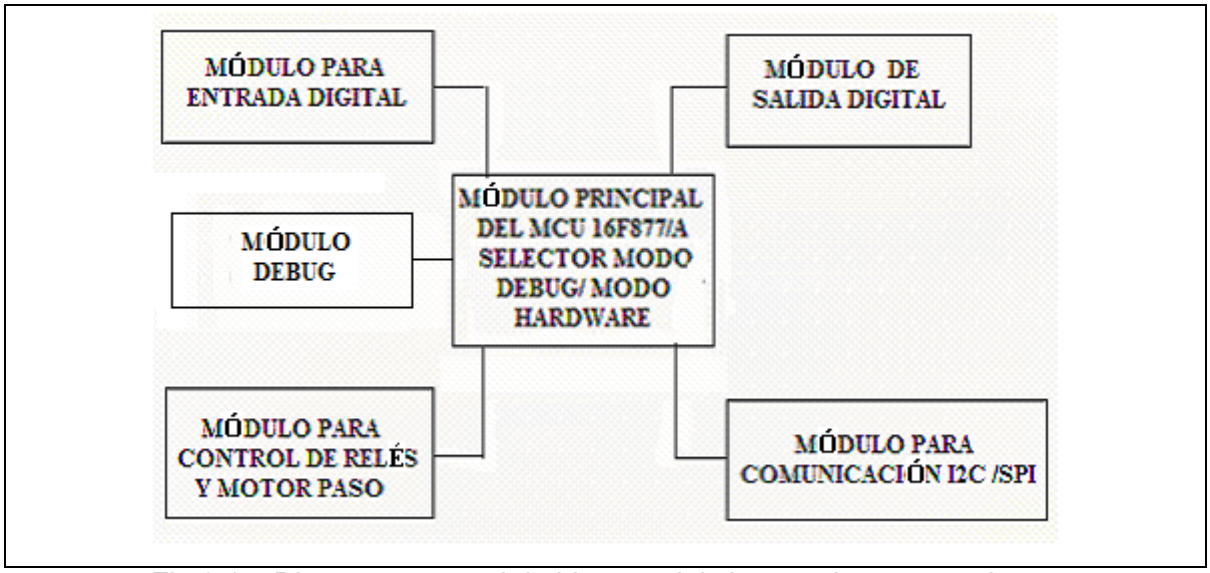


Fig.3.1 Diagrama general de bloques del sistema de entrenamiento.

3.2 GENERALIDADES SISTEMA DEBUG.

Tradicionalmente los desarrolladores de aplicaciones que utilizan controladores o microcontroladores hacen uso de emuladores (ICE emulators in-circuit) para desarrollar y depurar sus diseños antes de transferir el código al dispositivo.

La lógica de depuración que poseen los MCU 16F877/A conocida como Debug On-Chip ofrece la ventaja de realizar la depuración de manera más sencilla y práctica.

El circuito debug es el encargado de establecer conexión entre el microcontrolador a depurar y el software MPLAB, el núcleo del circuito debug radica en el MCU 16F876 el cual contiene el software que maneja las distintas etapas mostradas en el diagrama de bloques de la figura 3.2.

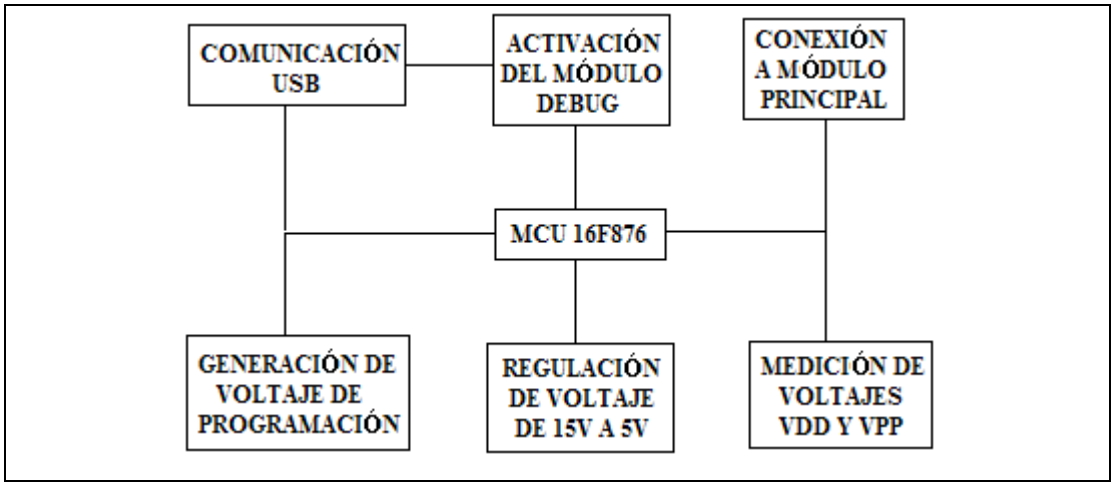


Fig.3.2 *Diagrama de bloques tarjeta debug.*

3.2.1 MICROCONTROLADOR PIC 16F876/A.

http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

El microcontrolador 16F876 posee las mismas características internas que contienen los MCU de la serie 16F877/A (memoria RAM, EEPROM, FLASH), la variación radica en la cantidad de pines disponibles (28 pines) ,por lo que se ve reducido los puertos disponibles de E/S en un circuito de aplicación, solo posee tres puertos A, B, C comparados con los cinco que poseen los de la serie 16F877/A (A, B, C, D, E) , además el PIC16F876 no posee el subsistema para puerto esclavo paralelo PSP.Para mayores detalles refiérase a la hoja de datos de los microcontroladores 16F876.

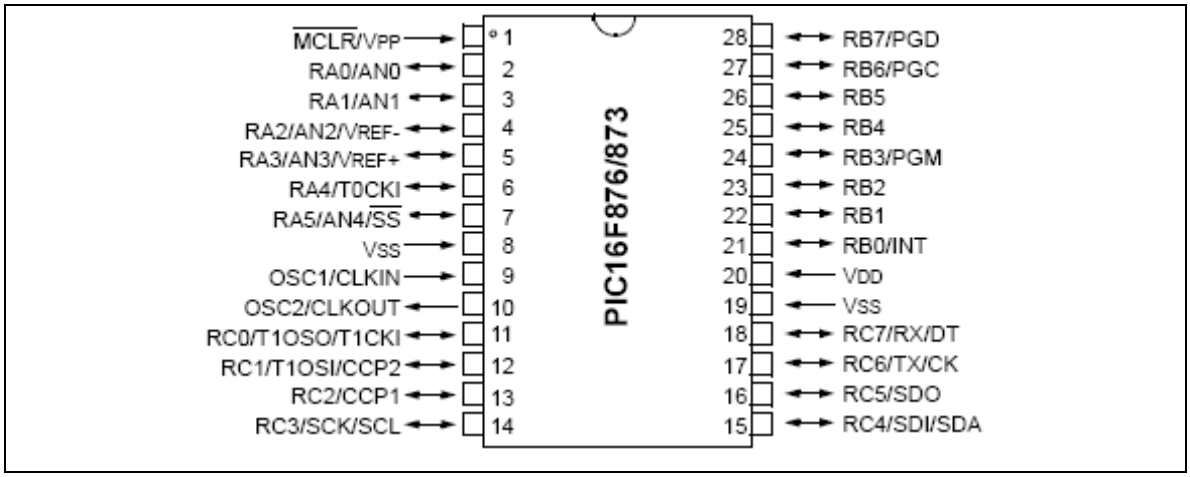


Fig.3.3 *Pines del microcontrolador 16F876.*

3.2.2 COMUNICACIÓN USB

3.2.2.1 PROTOCOLO USB.

El protocolo de comunicación USB (bus serie universal), es un bus punto a punto, el lugar de partida es un Host (PC) y el punto de llegada un periférico. En una arquitectura USB solo debe existir un host, el protocolo utilizado se basa en el paso de testigo (*Token*), el bus permite la conexión y desconexión de periféricos en cualquier momento sin necesidad de apagar el equipo.

3.2.2.2 INTERFAZ FÍSICA.

A nivel eléctrico el cable USB transmite la señal y alimentación a través de cuatro hilos como se muestra en la figura 3.4.

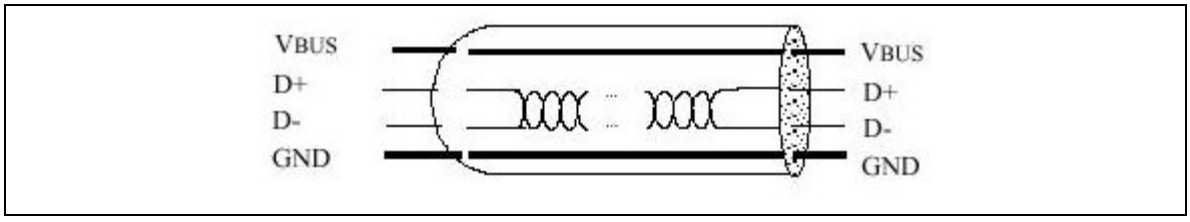


Fig.3.4 *Aspecto físico del cable USB.*

El cable proporciona la tensión nominal de 5v.Es necesario definir correctamente el diámetro del hilo con el fin de que no produzca una caída de tensión demasiado alta en el cable. El par trenzado que lleva las señales de datos contiene una impedancia típica de 90 Ohm, la velocidad a la cual se trabaja es de 12Mbps o 1.5Mbps y la sensibilidad del receptor debe ser de al menos 200mV.En la comunicación USB el reloj se transmite en el flujo de datos y la codificación es de tipo NRZI, existiendo un dispositivo que genera un bit de relleno que logra que la frecuencia del reloj permanezca constante. Una de las características es que cada paquete va precedido por un campo de sincronismo y la potencia máxima queda determinada por el host (PC), con esto se puede lograr que el dispositivo conectado pueda alimentarse desde la PC sin necesidad de fuente externa. El ordenador gestiona el consumo teniendo la capacidad de poner en reposo (*suspend*) o en marcha a un periférico USB, en reposo se reduce el consumo de potencia.

3.2.2.3 ENLACES VIRTUALES (TUBERIAS O PIPES).

Son enlaces virtuales entre el host y el dispositivo USB, este enlace configura que tipo de transferencia se va a utilizar (Control, Bulk, Isócrona o Interrup), los parámetros asociados con el ancho de banda, dirección del flujo de datos y el máximo o mínimo tamaño de los paquetes. Cada enlace esta caracterizado por su banda de paso (Token), su tipo de servicio, el número de punto terminal (End-Point) y el tamaño de los paquetes. Estos enlaces se definen y crean durante la inicialización del USB, siempre existe un enlace virtual 0 que permite tener acceso a la información de configuración del periférico USB.la norma USB define dos tipos de enlaces virtuales *Stream y Message.*

Stream: Consiste en un flujo sin formato USB definido, significa que se puede enviar cualquier tipo de dato. Este tipo de enlace soporta las transferencias Bulk, Isócrona e Interrupt, además tanto el host como el dispositivo periférico pueden controlar.

Message: Este tipo de enlace si tiene un formato USB definido y solo puede soportar la transferencia Control.

3.2.2.4 TIPOS DE TRANSFERENCIA.

Control: Modo utilizado para realizar configuraciones, existe siempre sobre el punto terminal cero (End-Point 0) por lo que todos los dispositivos USB soportan este tipo de transferencia. Los datos de control sirven para configurar el periférico al momento de conectarse al host USB.

Bulk: Este modo se utiliza para la transmisión de importantes cantidades de información , al igual que el tipo Control este no tiene perdidas de datos , este tipo de transferencia es útil cuando la tasa de transferencia no es critica por ejemplo datos desde un escáner. Solo los dispositivos de media y alta velocidad soportan este tipo de transferencia.

Interrupt: modo utilizado para las transmisiones de pequeños paquetes, rápidos y orientados a percepciones humanas como mouse .Este tipo de transferencia son orientados a dispositivos que necesiten atención periódicamente utilizados por dispositivos de baja velocidad y garantiza la transferencia de pequeñas cantidades de datos.

Isócrona: Conocida como flujo en tiempo real, utilizado para la transmisión de audio y video comprimido. Por ser en tiempo real es el modo de mayor prioridad.

3.2.2.5 ENUMERACIÓN

Cada vez que se conecta un dispositivo USB a la PC se produce el proceso de enumeración, consiste en que el host pregunta los parámetros a l dispositivo tales como:

- Consumo de energía expresada en unidades de carga.
- Número y tipos de puntos terminales.
- Clase del producto.
- Tipo de transferencia, etc.

El proceso de enumeración es inicializado por el host cuando detecta un nuevo dispositivo que ha sido adjuntado al bus USB, el host le asigna una dirección al

dispositivo y habilita su configuración permitiendo la transferencia de datos en el bus USB.

3.2.3 COMUNICACIÓN SERIAL.

3.2.3.1 CARACTERISTICAS ELÉCTRICAS.

En la comunicación serial el estándar utilizado es el RS232 creado en 1960, se caracteriza por que sus niveles de voltaje no son compatibles con los niveles TTL. Un receptor RS232 interpreta un voltaje más negativo que -3V como un 1 lógico y un voltaje más positivo que 3V como un 0 lógico. Un transmisor debe tener un voltaje más negativo que -5V como 1 lógico y un voltaje más positivo que 5V como 0 lógico. Los valores típicos para un 0 o 1 lógico están entre +-10V y +-12V. Por lo que permite mayores distancias y mayor inmunidad a ruidos.

3.2.3.2 PROTOCOLO SERIAL ASÍNCRONO ESTANDAR.

Este protocolo estandariza la técnica de comunicación serial, se establece un bit de comienzo (0) y un bit de parada (1), se contemplan las velocidades de transmisión como 75, 1200,9600 bps y otras. Diferentes parámetros pueden ser especificados cuando configuramos una comunicación serial, los más comunes son:

- Bits por caracter, usualmente de 5 a 8 bits por caracter a transferir.
- Numero de bits de Stop 1 o 2.
- Bits de paridad, usado para detectar errores en un único bit, puede ser especificado como par, impar o sin paridad.
- Velocidad de transmisión.

3.2.3.3 CONTROL DE FLUJO EN LA COMUNICACIÓN SERIAL

El control de flujo se hace necesario cuando la velocidad de transferencia del dispositivo transmisor es varias veces mayor que la velocidad a la que el dispositivo receptor procesa los datos. El control de flujo puede realizarse por hardware y software.

Control de flujo por software: denominado Xon/Xoff, usa dos caracteres ASCII uno para Xon que es el caracter 17 y otro para Xoff que es el carácter 19. El receptor envía el carácter Xoff cuando su buffer se llena indicándole al emisor que pare de enviar datos. Una vez que el receptor tiene espacio envía el carácter Xon para indicar al emisor que envíe más datos, éste control de flujo tiene la ventaja de no utilizar líneas adicionales ya que los caracteres Xon y Xoff se envían por las líneas TD/RD; como desventaja tiene la disminución de la velocidad de transferencia por el envío de estos caracteres que no forman parte del mensaje.

Control de flujo por hardware: utiliza dos líneas del cable serial RTS y CTS. Cuando el dispositivo transmisor desea enviar datos activa la línea RTS, si el receptor tiene espacio para los datos responde activando el pin CTS con lo cual se inicia la transferencia de datos. Si el buffer del receptor se llena, éste desactiva la línea CTS indicándole al emisor que pare el envío de los datos.
Este tipo de control de flujo es el utilizado en la comunicación entre el software MPLAB y la tarjeta debug.

3.2.4 CARACTERISTICAS DEL CONVERTIDOR USB –SERIAL FT232BM.

http://www.ftdichip.com/Documents/DataSheets/DS_FT232BM.pdf

El convertidor FT232BM es un chip encapsulado en un paquete LQFP que posee 32 pines, su estructura interna permite realizar las funciones de conversión USB –SERIAL. Es un dispositivo ampliamente usado en dispositivos como cables conversores USB-Serial y dispositivos que requieran comunicación USB en general, debido a los pocos componentes externos y facilidad de configuración comparados con circuitos de comunicación USB que requieren muchos componentes externos.
El diagrama de bloques se muestra en la figura 3.5.

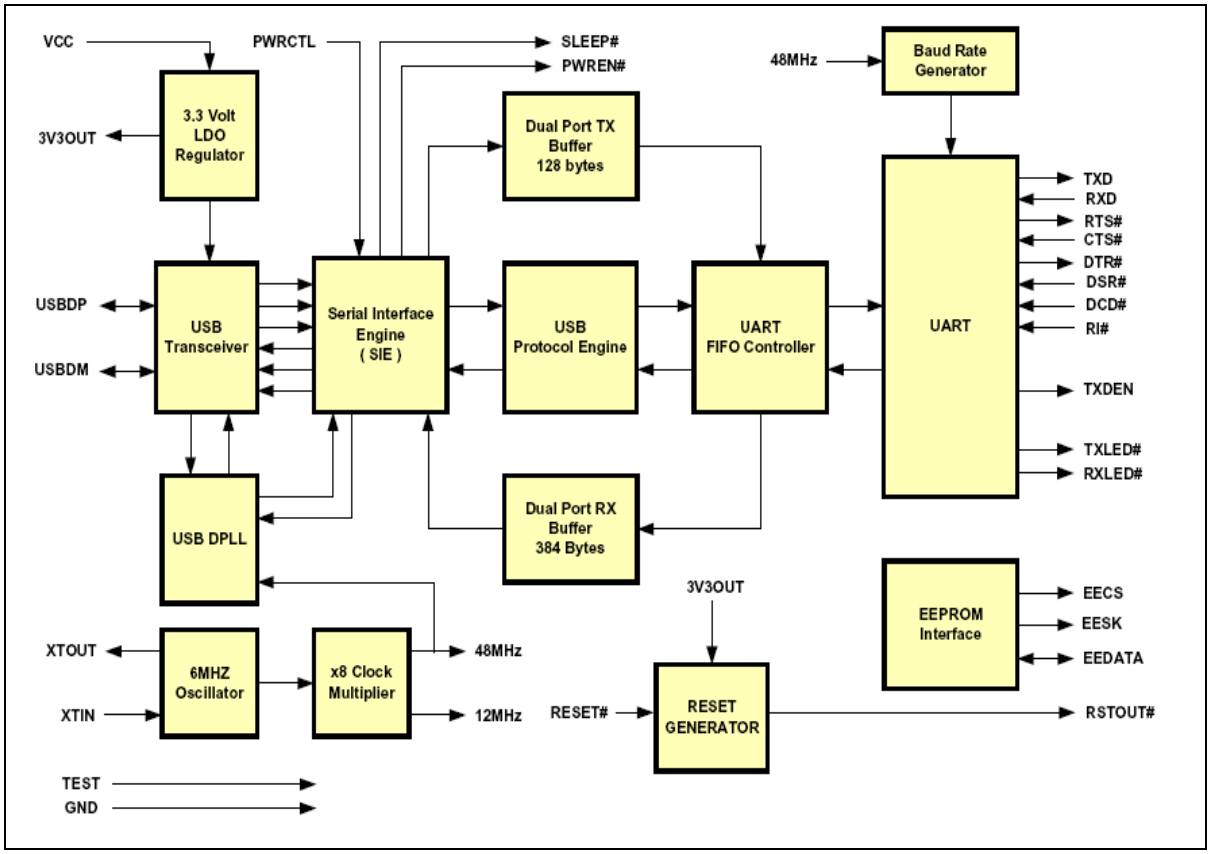


Fig.3.5 *Diagrama de bloques convertidor USB-SERIAL FT232BM.*

Características:

- Puede trabajar con lógica de 3.3V y 5V.
- Regulador interno de 3.3V para E/S USB.
- Compatible con norma USB 1.1 y USB 2.0.
- Soporta protocolos hardware y X-On/X-Off.
- Rango de voltaje de operación de 4.35V -5.25V.
- Compatible con controladores de host UHCI/OHCI/EHCI.
- PLL interno compatible con osciladores de 6Mhz - 48MHz.
- Soporta señales de interface para conexión con módems.
- Permite conexión a memoria EEPROM para ser programada vía USB.
- Buffer de recepción de 384 bytes y buffer de transmisión de 128 bytes.

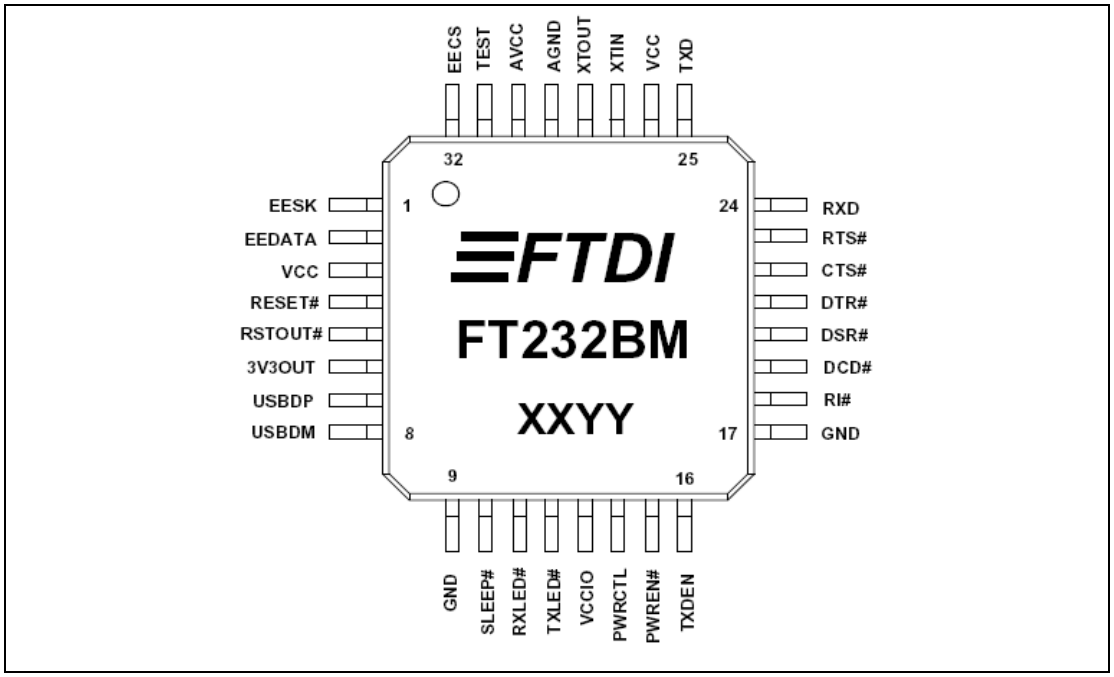


Fig.3.6 *Encapsulado FT232BM.*

3.2.4.1 CONFIGURACIONES DEL OSCILADOR.

La figura 3.7(a, b) muestra los distintos diagramas de conexión de osciladores para el funcionamiento del convertidor FT323BM.El diagrama 3.7a muestra el uso de un resonador cerámico de 3 pines, por poseer los capacitores de carga internos y una precisión de +-0.1 es la alternativa más usada en diseños de alta velocidad utilizando USB, solamente necesitando conectar una resistencia de carga (1MOhm) entre XTIN-XTOUT. La figura 3.7b detalla el uso de un resonador cerámico de 2 pines en el cual se hace uso de 2 capacitores de carga externos cuyo rango puede estar entre 22pf -27pf.

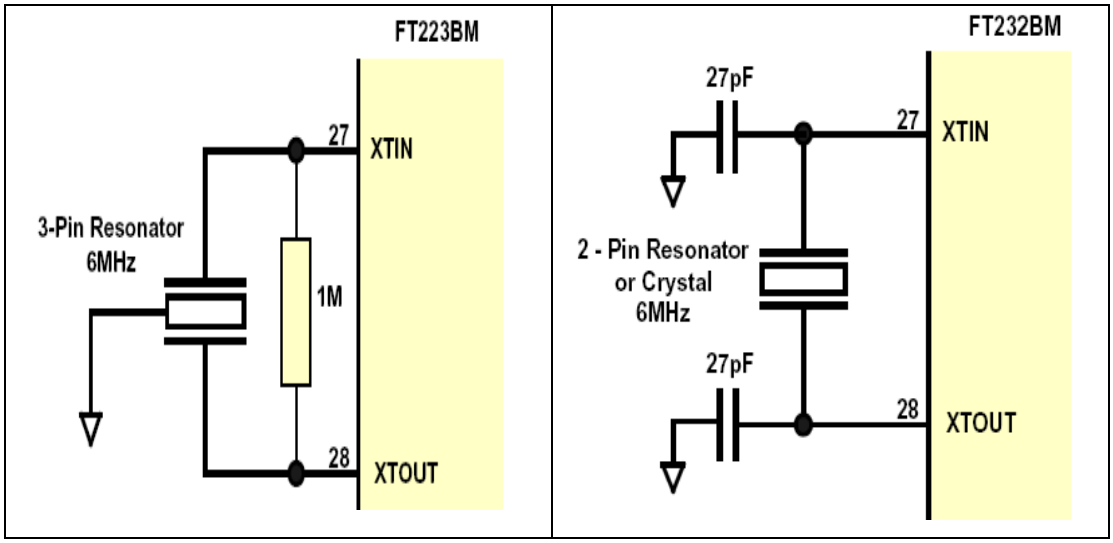


Fig.3.7 (a, b) *Diagramas de conexión del oscilador.*

3.2.4.2 DIAGRAMA DE CONEXIÓN DEL FT232BM.

El dispositivo puede funcionar a distintos voltajes (5v y 3.3V), existen diagramas específicos para cada lógica, se ha optado por el diagrama de conexión a 5V debido a que el voltaje de alimentación del sistema es de 5v ahorrándonos reguladores de voltaje. De este diagrama se derivan 2 tipos de conexión: una es que el dispositivo se alimente directamente desde el puerto USB figura 3.8a, la segunda es alimentar al dispositivo desde una fuente externa figura 3.8b.

La utilización de cualquiera de las conexiones de la figura 3.8a y 3.8b Depende de la cantidad de corriente manejada por el circuito conectado al dispositivo FT232BM, para el caso de la alimentación por USB solo puede obtenerse 100mA, si se desea mayor cantidad de corriente es preferible la conexión a fuente externa. Los capacitores de desacople mostrados en cada diagrama sirven para evitar interferencias de ruido que pueden alterar el funcionamiento del dispositivo.

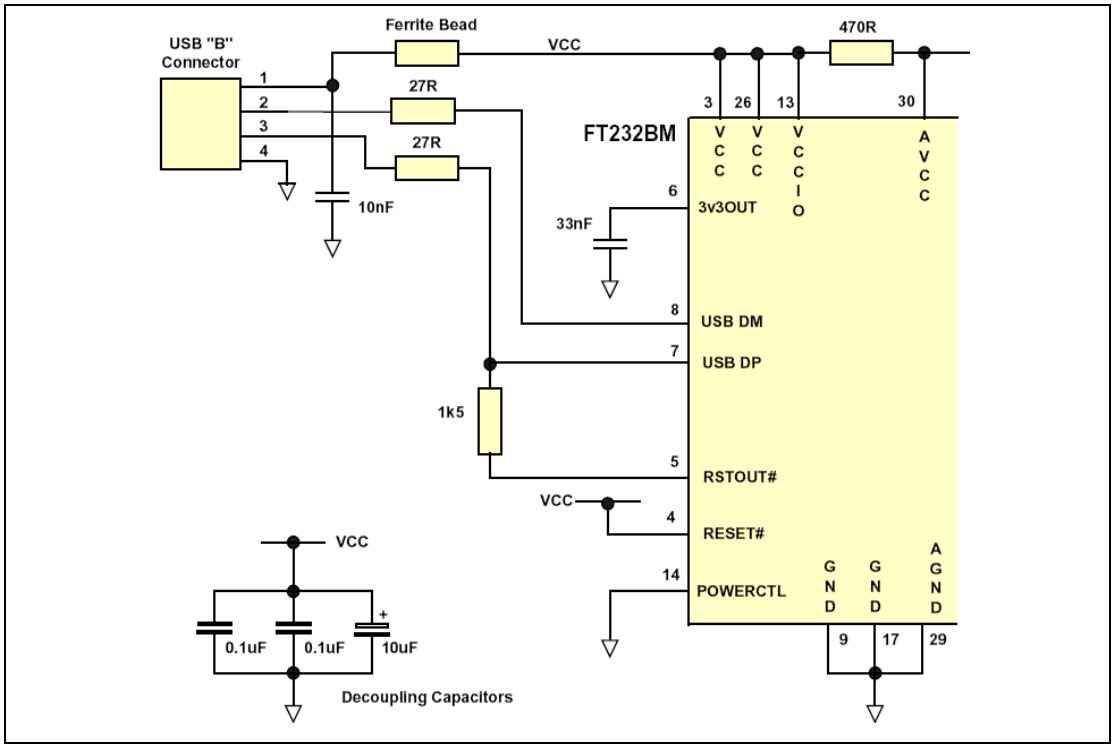


Fig.3.8a *Diagrama de conexión alimentado por el bus USB.*

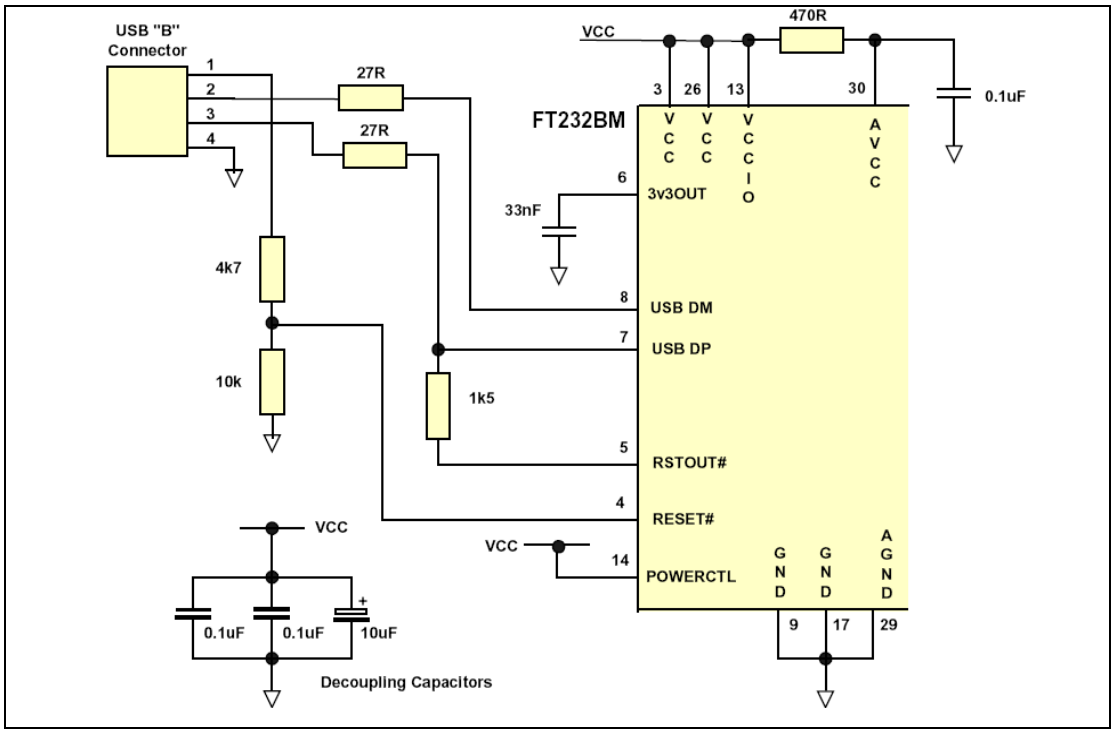


Fig.3.8b *Diagrama de conexión alimentado por fuente externa.*

3.2.5 CARACTERISTICAS DEL AMPLIFICADOR OPERACIONAL LM358.

http://www.datasheetcatalog.org/datasheet/nationalsemiconductor/DS007787.PDF

El dispositivo LM358 es un amplificador operacional de baja potencia, el encapsulado contiene dos amplificadores independientes, cada uno con alta ganancia y compensación interna de frecuencia. El dispositivo puede trabajar en un rango amplio de voltaje, es decir puede ser conectado a niveles de voltaje como 5V para trabajar con lógica TTL, sin necesidad de estar conectado específicamente a +/- 15V como sucede con los amplificadores comunes. El diagrama de pines se muestra en la figura 3.9.

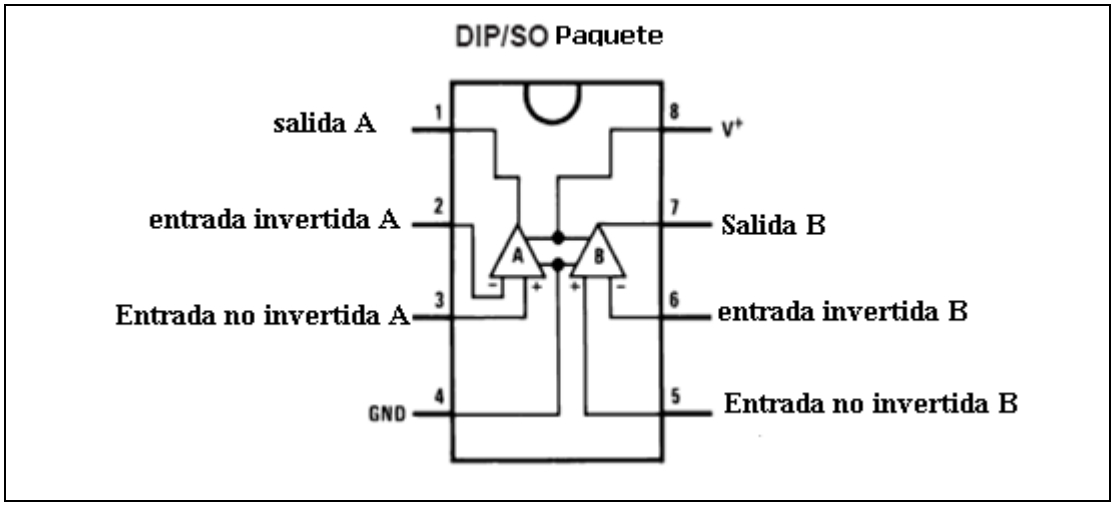


Fig.3.9 *Pines de dispositivo LM358.*

3.3 DESCRIPCIÓN DEL HARDWARE DEBUG.

El circuito debug es alimentado por una fuente externa de 15Vdc,obteniendo a traves del amplificador operacional el voltaje de programación Vpp (13.5V).El regulador 7805 mantiene los 5V necesarios para alimentar el chip FT232BM, el MCU 16F876 y el resto del circuito.

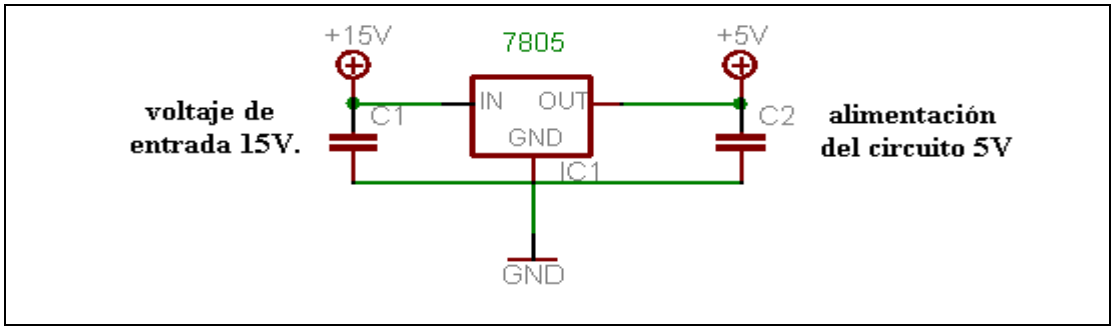


Fig.3.10 *Regulación de voltaje de alimentación.*

La comunicación se realiza a través del chip FT232BM cuyo circuito de conexión aparece en la figura 3.11.

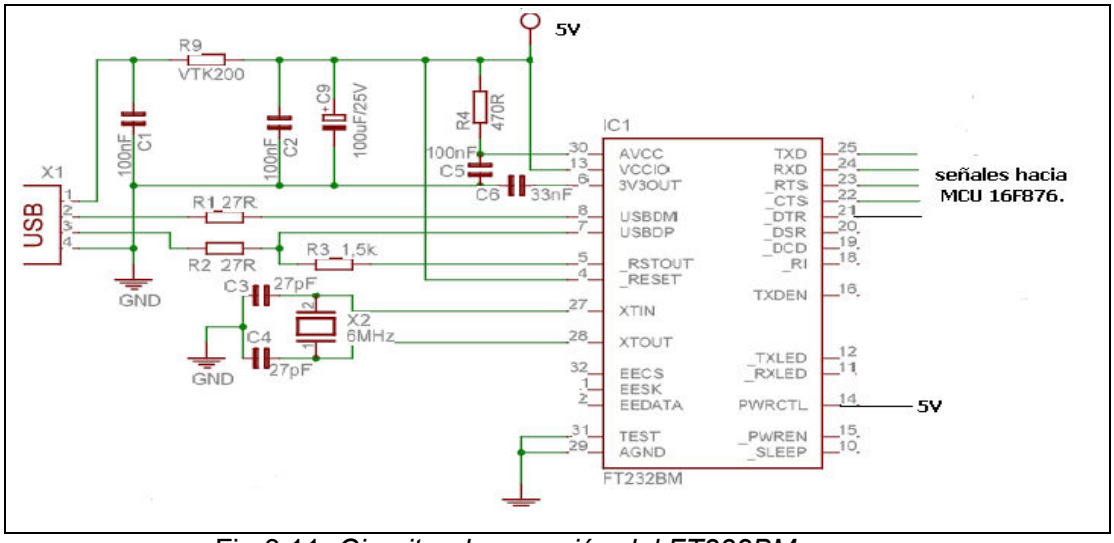


Fig.3.11 *Circuito de conexión del FT232BM.*

El software MPLAB utiliza control de flujo por hardware para manejar la tarjeta debug que es la interfaz entre MPLAB y el MCU a depurar. Para establecer conexión con la tarjeta debug se utiliza el botón conectar del ambiente grafico del software MPLAB, dicho pulso se ve reflejado en el pin DTR que controla el circuito de la figura 3.12 .El circuito consiste en un transistor de switcheo , cuando se genera la señal para conectar, el transistor recibe un pulso que logra saturarlo cambiando al estado bajo (0V) logrando con

esto un reset al MCU 16F876, luego vuelve a su estado alto (5V) iniciando la comunicación con el software MPLAB.

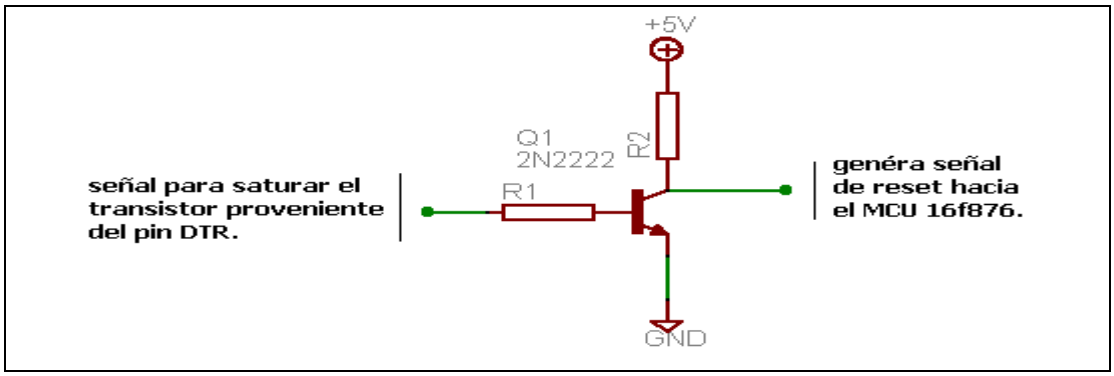


Fig.3.12 *Conexión para activar sistema debug.*

La tarjeta debug debe de ser capaz de censar periódicamente los niveles de voltaje presentes en el MCU a depurar. Para asegurar que los niveles de voltaje estén correctos se ha establecido un patrón para cada voltaje a censar como se muestra en el circuito de la figura 3.13.

El divisor de voltaje en RA1 es el encargado de medir el voltaje Vdd utilizando el patrón de 2.5V, si existe este valor en el pin RA1 el voltaje Vdd es correcto. Para el caso del voltaje Vpp se utiliza el divisor de voltaje en RA3 que es aproximadamente 2V que me indica un nivel indirecto del voltaje real Vpp, con esto se evita hardware de acomodamiento del nivel de voltaje Vpp para que sea leído por el MCU que trabaja con niveles TTL.

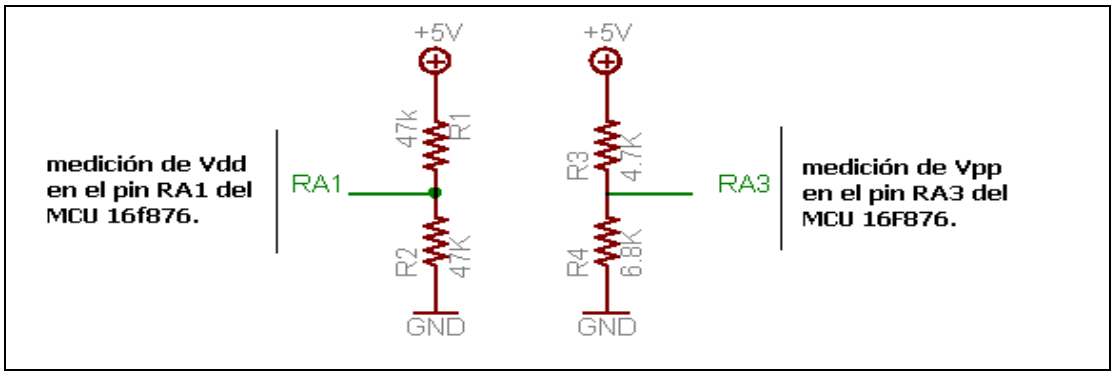


Fig.3.13 *medición de señales Vdd y Vpp.*

La tarjeta debug también es la encargada de controlar las señales para depurar el MCU 16F877/A que son PGD, PGC y MCLR/Vpp (datos, reloj y voltaje de programación en modo ICSP), éstas salidas se activan cuando se ejecuta la opción de descargar código o depurar código desde el entorno de MPLAB.

Para generar el nivel de voltaje MCLR que es de 13.5V se realiza a través del circuito de la figura 3.14, se utiliza un amplificador con el cual se establece una ganancia utilizando la resistencia variable de 50K Ohm para ajustar el voltaje deseado. El circuito se activa a través de los pines RC0 y RC2 provenientes del MCU 16F876 cuando se ejecuta la opción descargar código. Para asegurar que el voltaje de programación sea el correcto se utiliza un arreglo de resistencias que me genera un voltaje indirecto censado por el pin RA0 , además la resistencia de 33 Ohm me sirve para limitar la corriente proveniente del amplificador para evitar daños al MCU a depurar.

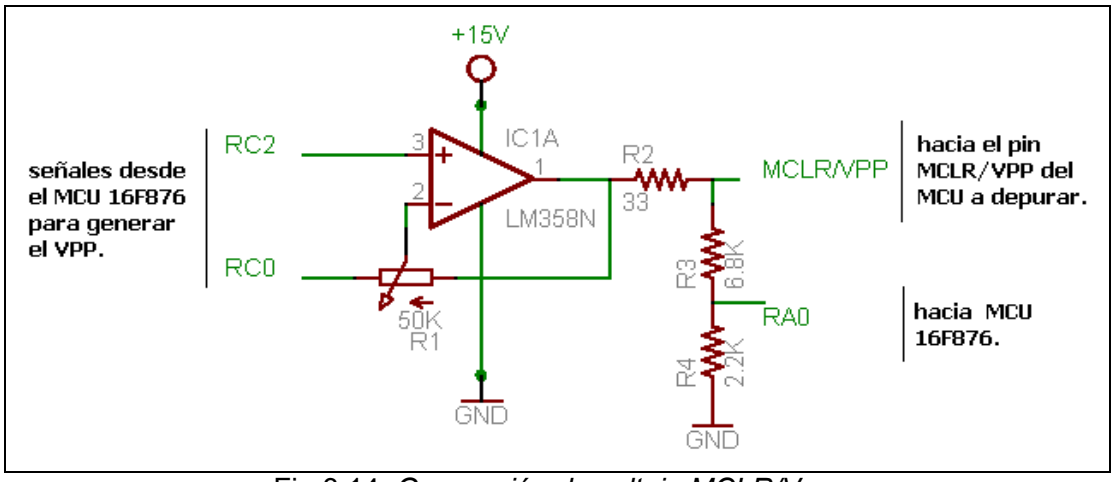


Fig.3.14 *Generación de voltaje MCLR/Vpp.*

La señales PGC (reloj) y PGD (datos) se obtienen desde el software, el MCU solamente es el encargado de llevarlas al medio físico, activando RC3 que es la salida de reloj de programación y las salidas RC4-RC5 que controlan los datos a programar.

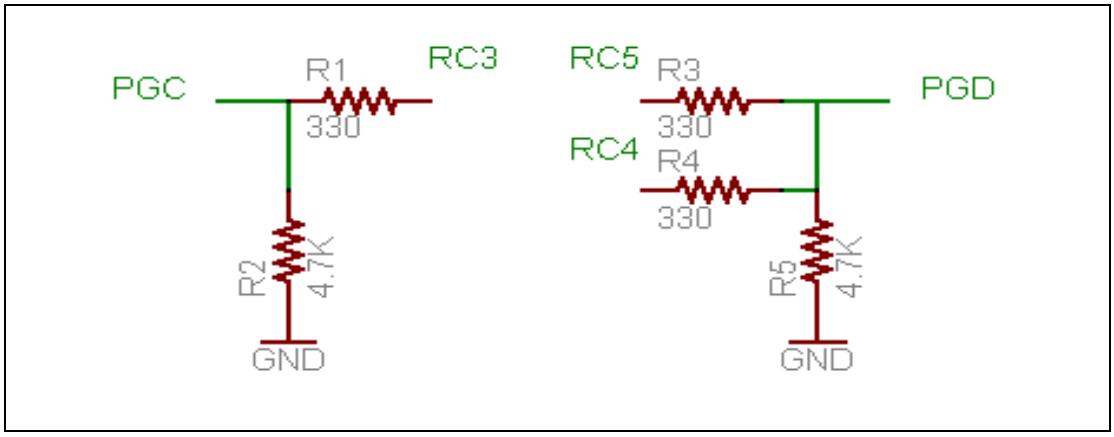


Fig. 3.15 *Salidas de señales PGC y PGD.*

3.3.1 DEPURACIÓN UTILIZANDO MPLAB Y TARJETA DEBUG.

La depuración se realiza utilizando el software MPLAB en conjunto con la tarjeta debug, las opciones de conectar, descargar, correr y depurar código se ejecutan desde el entorno grafico de MPLAB. Para desarrollar la operación de depuración de un programa determinado en los MCU 16F877/A se realiza de la siguiente manera (para detalles de conexión y especificación de parámetros refiérase a la sección del manual de usuario del sistema).

Al establecer conexión desde MPLAB el software reconoce que tipo de dispositivo se depurará, si se desea depurar el MCU 16F877 este será reconocido por MPLAB. Una vez establecida la conexión puede descargarse el código al dispositivo a través de la tarjeta utilizando el protocolo ICSP (capitulo 2), el software descargará además el sistema operativo (debug executive) que activará los registros internos de depuración del MCU (capitulo 3) y será encargado de realizar la depuración en el dispositivo utilizando registros determinados para la depuración, por ser una aplicación consume recursos dentro del dispositivo como se detalla:

Recursos reservados para depuración en los MCU 16F877/A:

- Pin MCLR/VPP – RB7 (PGD) -RB6 (PGC).
- Espacio de memoria de programa 0x1F00 – 0x1FFF.
- Registros usados 0x70 - 0xF0 - 0x170 - 0x1F0 - 0x1E5 - 0x1EF.

Es de tomar en cuenta que el código a depurar no debe utilizar estos recursos tanto de memoria como de hardware.

Al activar la opción de correr código desde el entorno de MPLAB la tarjeta se activa el dispositivo para que este inicie el desarrollo del código, los cambios en los registros internos se guardan en los registros utilizados por el sistema operativo y cuando se detiene el código estos registros son enviados al software por el pin de datos PGD para ser mostrados. Existe la opción de correr paso a paso el código, esta opción ofrece la ventaja de visualizar en tiempo real los cambios en los registros internos pero es más lenta que la anterior, ésta es muy utilizada para verificar completamente el código creado.

3.4 GENERALIDADES HARDWARE DE APLICACIONES.

El hardware de aplicaciones consiste en los periféricos necesarios para comprobar de manera práctica el uso de los subsistemas propuestos en los alcances del equipo, debido a que su fin está orientado al ámbito didáctico se dispone de módulos básicos para desarrollar distintas aplicaciones, por lo que el sistema consiste en dispositivos de entrada digital conocidos como minidips , leds visualizadores , tarjeta para manejo de relés de estado sólido, conexión para manejo de motor paso, entrada analógica , salida análoga variable y conexión para comunicación SPI e I2C con dispositivos que soporten dicha comunicación.

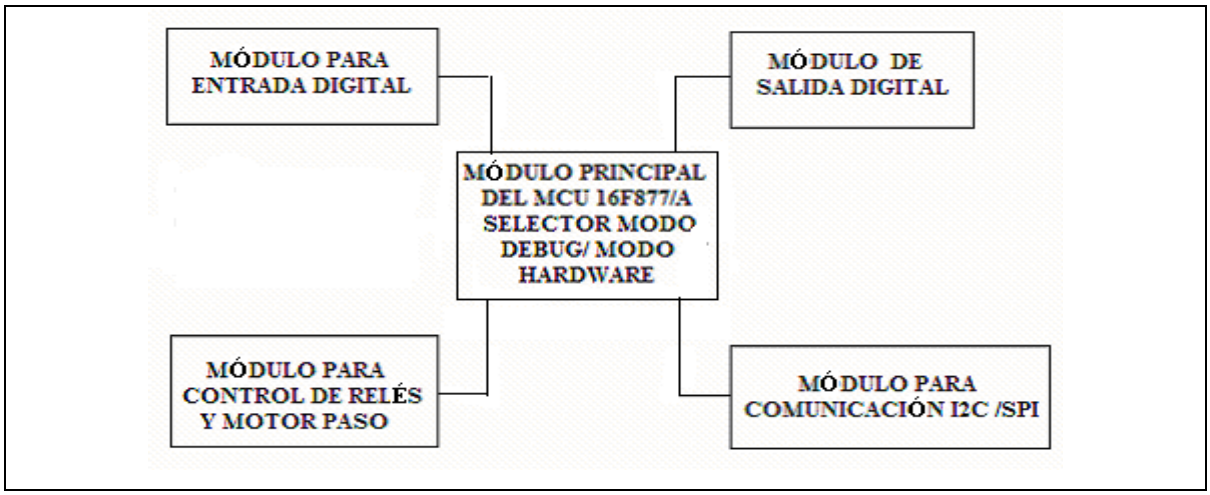


Fig. 3.16 *Diagrama de bloques hardware de aplicaciones.*

3.4.1 MÓDULO PRINCIPAL.

La tarjeta principal es la que contiene el microcontrolador a depurar, etapa conteniendo pulsadores para entrada digital, etapa de lectura de datos análogos, etapa de comunicación serial asíncrona SCI y las distintas conexiones para los módulos a utilizar. El módulo ofrece la ventaja de utilizarlo ya sea con la tarjeta debug o simplemente como tarjeta de pruebas para los PICs 16F877/A a través de un selector presente en la tarjeta. La tarjeta esta diseñada de manera que pueda soportar otros módulos creados por el usuario, siempre que estén basados en el diagrama de pines correcto, el diagrama se muestra en la figura 3.17.

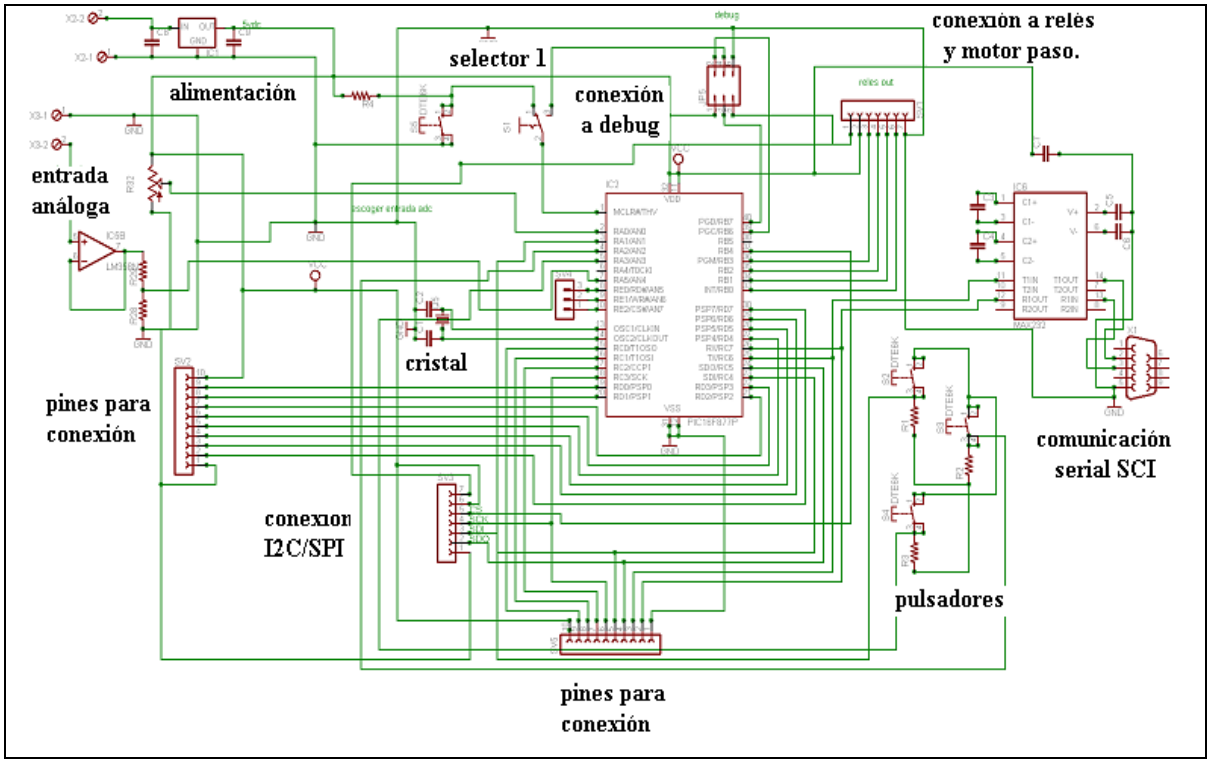


Fig. 3.17 *Diagrama tarjeta principal de aplicaciones*.

El selector 1 es el encargado de seleccionar el modo de utilización de la tarjeta principal ya sea utilizando la tarjeta debug o simplemente como tarjeta de prueba de programas.

3.4.1.1 Lectura de datos digitales

Para la lectura de datos digitales en la tarjeta principal se hace uso de pulsadores normalmente abiertos conectados al microcontrolador, estos pasan de estado bajo a alto al activarlos, la figura 3.18 muestra el diagrama de conexión.

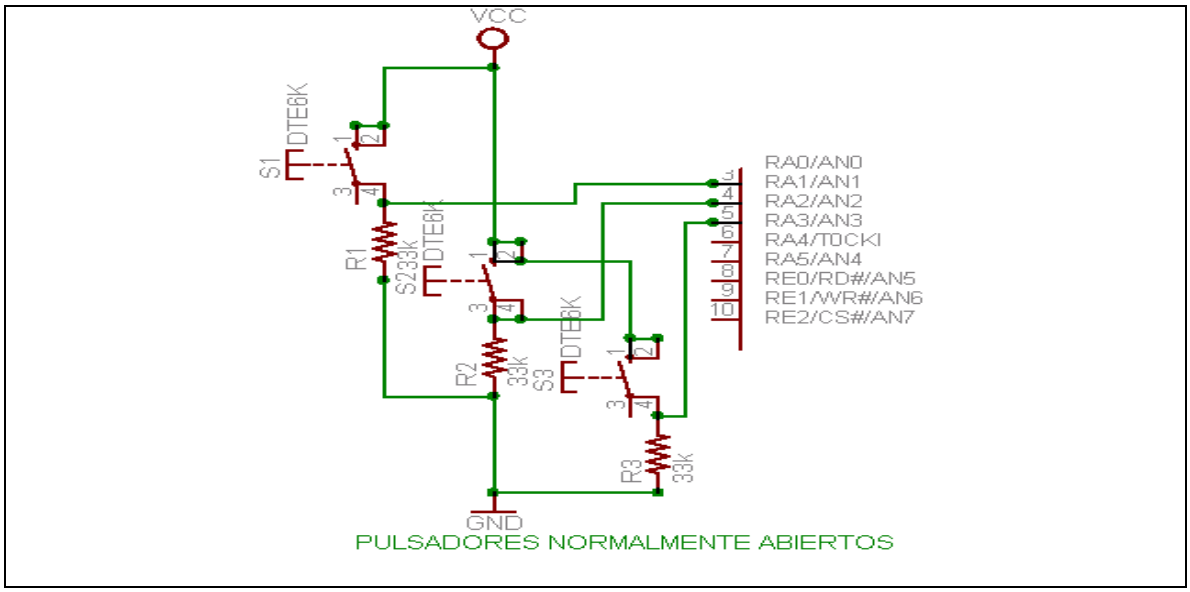


Fig. 3.18 *Etapa de pulsadores.*

3.4.1.2 Lectura de datos análogos.

Para la lectura de datos análogos se dispone del módulo de conversión análoga/digital que posee el microcontrolador, la etapa de lectura consiste en la obtención de señal de voltaje análoga desde el potenciómetro interno (0- 5Vdc) que posee la tarjeta o por medio de un dispositivo externo que puede conectarse a la tarjeta por medio de la entrada análoga , la etapa de lectura de un dispositivo externo consiste en un amplificador operacional en modo seguidor de voltaje, para proteger el circuito brindando una baja impedancia al circuito de lectura análoga ,la figura 3.19 muestra esta etapa.

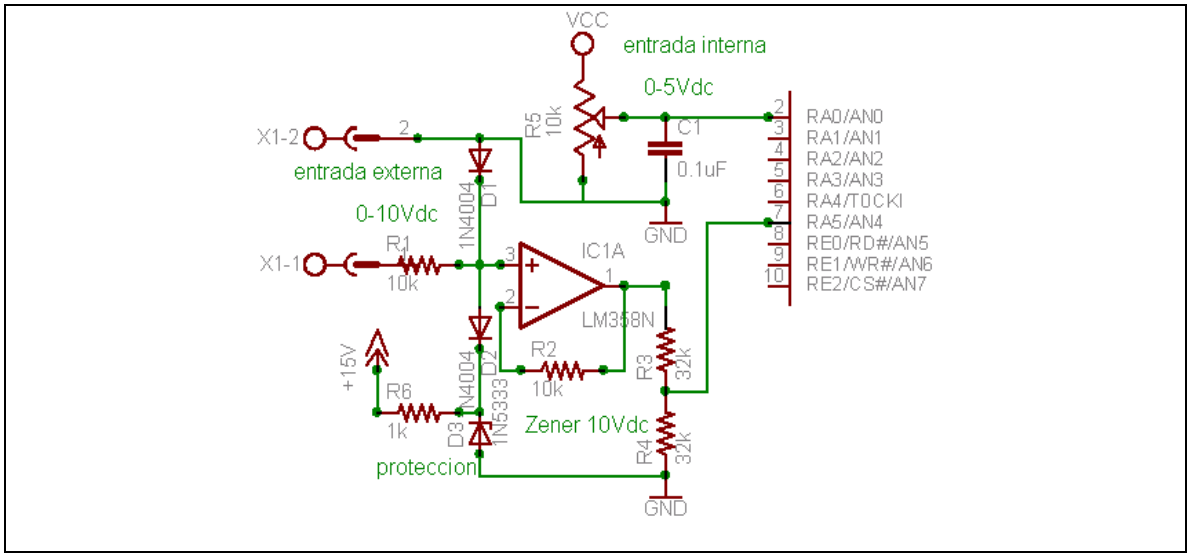


Fig. 3.19 *Etapa de lectura análoga*.

3.4.1.3 Comunicación serial SCI.

Este subsistema incluido en los MCU 16F877/A es el encargado de establecer comunicación con dispositivos como PCs y dispositivos que soporten este tipo de comunicación. Se usa el chip MAX232 que utiliza el protocolo serial RS232, los capacitores de 1uF están establecidos por el fabricante, el diagrama de conexión del chip serial y el PIC tomado de la hoja de datos del MAX232 se muestra en la figura 3.20.

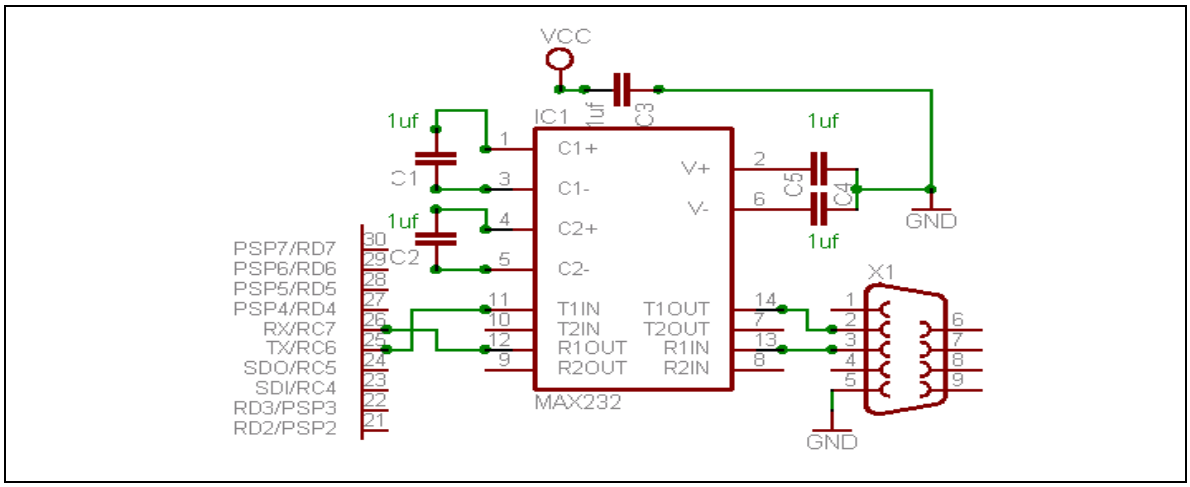


Fig. 3.20 *comunicación serial SCI.*

La etapa de conexiones generales consiste en 2 conectores de 10 pines cada uno que están situados en el puerto C y puerto D, por lo que se cuenta con dos puertos de 8 bits cada uno, también 2 conectores de 7 pines uno para la tarjeta I2C/SPI y el otro para la tarjeta de relés y motor paso, además de un conector de 3 pines para señales de control (WR/RD/CS) utilizados al conectar el PIC con un microprocesador.

3.4.2 MÓDULO PARA ENTRADA DIGITAL.

Es la tarjeta utilizada para enviar datos digitales hacia el MCU de la tarjeta principal, con este módulo puede enviarse un byte de datos a la vez (8 bits), además de conectarse al puerto C o puerto D según se desee, el diagrama se muestra en la figura 3.21 que consiste en switchs del tipo DIP, que pueden estar en estado bajo o alto según su posición y así obtener los 8 bits a enviar.

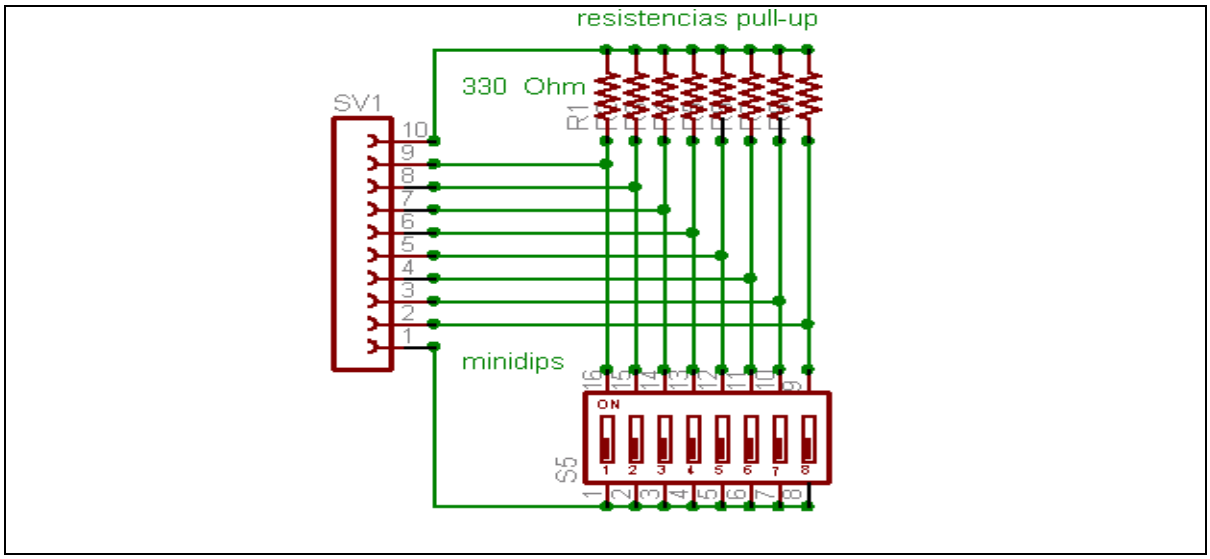


Fig. 3.21 *Diagrama Módulo de entrada digital*.

3.4.3 MÓDULO DE SALIDA DIGITAL.

El módulo de salida digital es utilizado para la visualización de datos digitales, al igual que la tarjeta de entrada digital puede ser conectada al puertoC o puertoD, consiste en 8 leds visualizadores como se muestra en la figura 3.22.

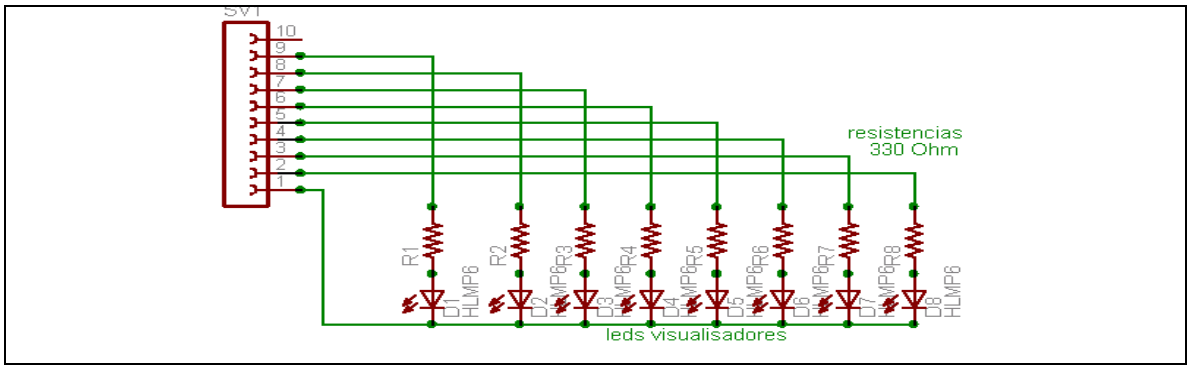


Fig. 3.22 *Diagrama módulo de salida digital.*

3.4.4 MÓDULO PARA CONTROL DE RELÉS Y MOTOR PASO.

Es la encargada de controlar la etapa de potencia que consiste en 4 relés y un motor paso. La tarjeta puede utilizarse para controlar ya sea los relés o el motor paso a través de un selector, los transistores TIP120 son utilizados como switch para controlar el motor paso, el chip ULN2003A es utilizado para controlar los relés , solo puede ser utilizada a opción a la vez , el diagrama se muestra en la figura 3.23.

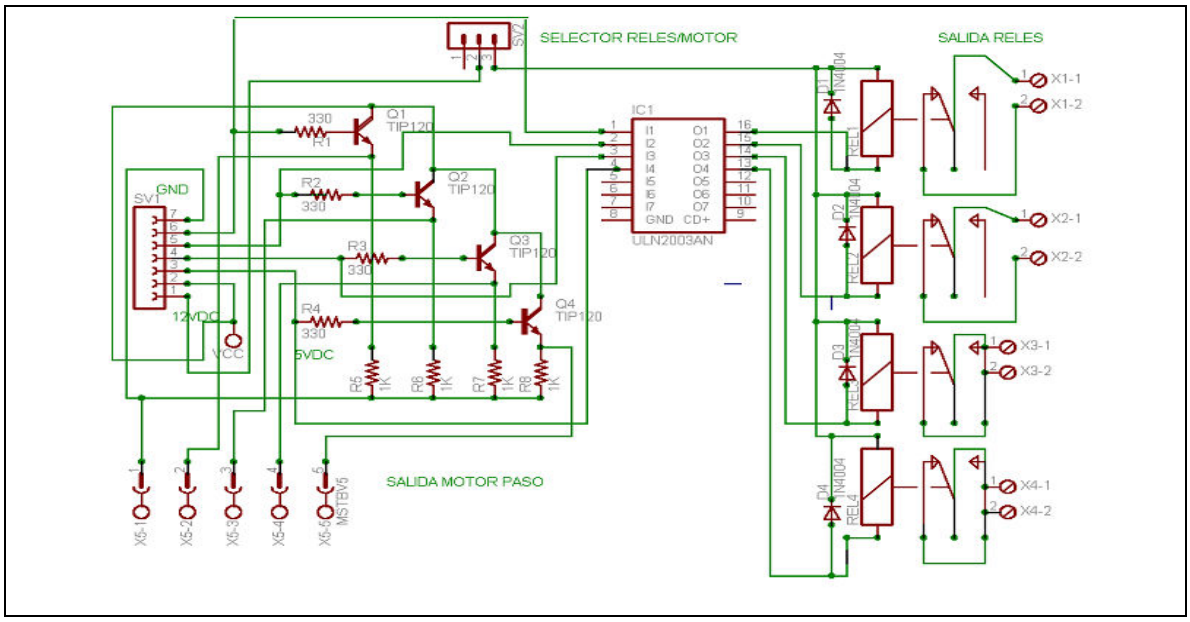


Fig. 3.23 *Diagrama de módulo para relés y motor paso.*

3.4.5 MÓDULO PARA COMUNICACIÓN I²C- SPI.

la tarjeta I²C –SPI contiene los periféricos que utilizan la comunicación I²C y SPI como: conversor digital –análogo MCP4822 (SPI), Memoria EEPROM serial (I²C) , microcontrolador 16F877A(I²C o SPI), etapa de lectura de datos análogos y puertos para lectura /escritura de datos digitales en donde pueden conectarse las tarjetas de entrada y salida digital explicadas anteriormente .El módulo I²C-SPI puede comunicarse con la tarjeta principal ya sea utilizando el protocolo I²C ó SPI, la especificación del tipo de comunicación se realiza por medio del selector , los dispositivos usados en la tarjeta I²C-SPI y su utilización son descritos a continuación. El diagrama general se muestra en la figura 3.24.

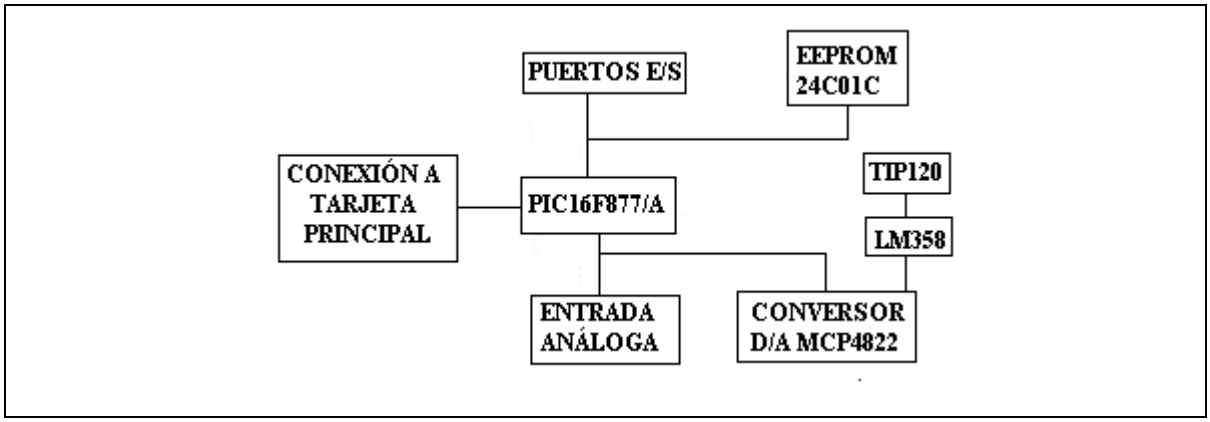


Fig.3.24 *Diagrama de bloques módulo I²C /SPI.*

Nota:

La tarjeta I²C-SPI esta diseñada para utilizarse solo en modo esclavo, la tarjeta principal es la que realiza las funciones de maestro en ambos protocolos.

3.4.5.1 SALIDA ANÁLOGA VARIABLE UTILIZANDO CHIP MCP 4822 (SPI).

Debido a que el MCU a utilizar no dispone de un módulo de conversión digital /análogo se hace uso de un chip extra que es el DAC MCP4822 el cual me realiza la conversión de los datos digitales provenientes del MCU 16F877/A, de la tarjeta principal, convirtiéndolos a voltaje análogo dentro del rango 0-10VDC y soportando 1A de corriente como máximo, el circuito de la etapa de conversión digital/ análoga se muestra en la figura 3.25.

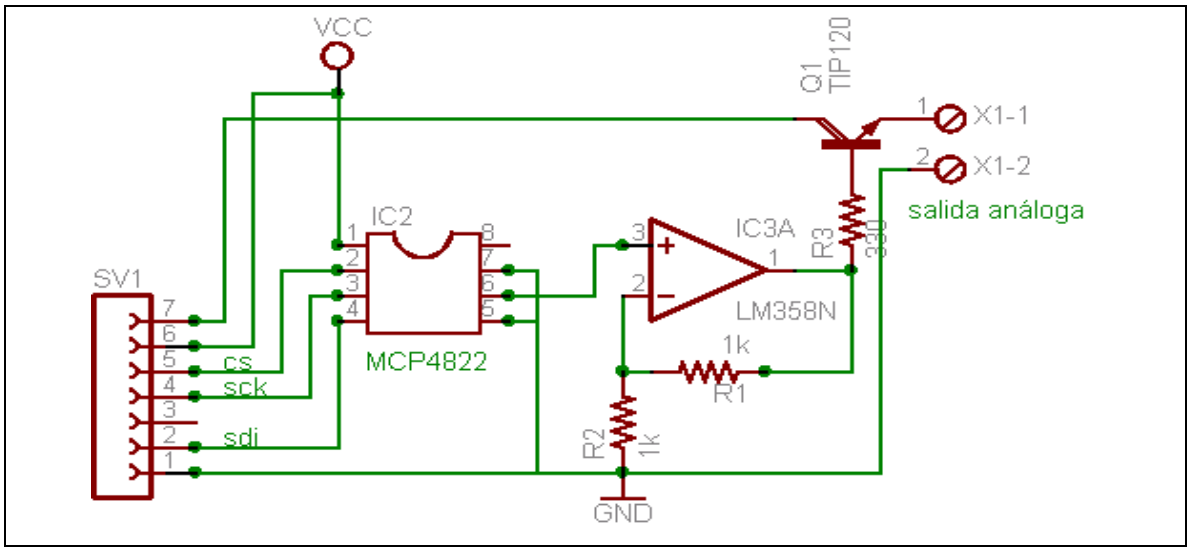


Fig.3.25 *Circuito etapa conversor digital/análogo de la tarjeta I²C-SPI.*

3.4.5.2 Chip conversor digital-análogo MCP 4822.

http://ww1.microchip.com/downloads/en/DeviceDoc/21953a.pdf

El módulo de conversión digital-análogo es el encargado de generar los niveles de voltaje DC que serán utilizados para alimentar dispositivos externos como motores DC, sensores, etc. Los MCU a utilizar no poseen este subsistema interno en el chip, por lo que se ha utilizado un chip extra que genera la conversión digital (proveniente del MCU) a análoga, la comunicación entre el conversor y el MCU se realiza a través del protocolo serial síncrono SPI (capitulo 1 Sección 1.8.2), lo que hace reducir considerablemente las conexiones (usando 3 conexiones *Clock, I/O de datos, selector de dispositivo*).

El dispositivo MCP4822 proveniente de la casa Microchip puede alimentarse en el rango de 2.7V -5.5Vdc, siendo un dispositivo de baja potencia, con una resolución de 12 bits para conversión digital-análoga. El MCP4822 posee alta precisión y bajo ruido para ser utilizado en aplicaciones industriales como calibración y compensación de señales como temperatura presión y humedad.

La arquitectura interna está basada en el uso de una cadena de resistencias, ofreciendo ventajas como el bajo error DNL, coeficiente de temperatura bajo y rápido tiempo de actuación, además incluye reset por alimentación (POR power on reset).

El diagrama de bloques interno del conversor MCP4822 se muestra en la figura 3.26.

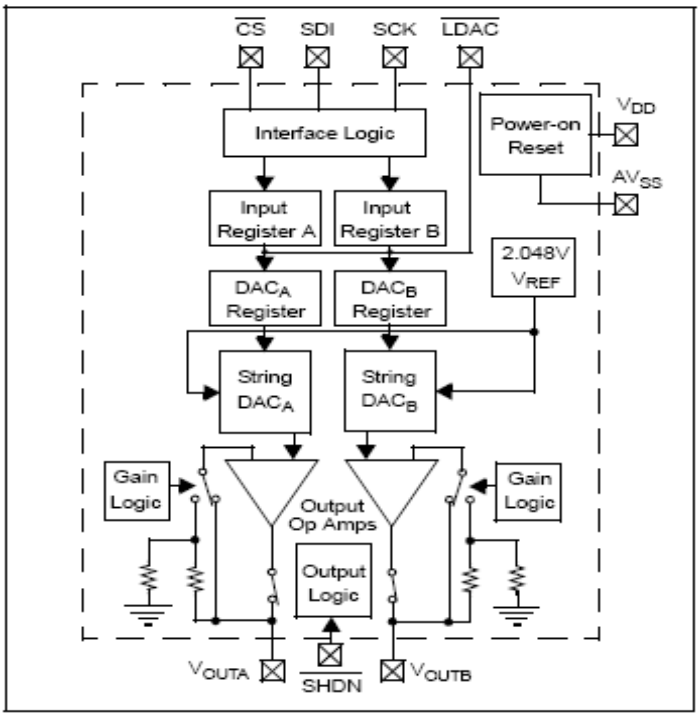


Fig. 3.26 *Diagrama de bloques MCP4822*.

3.4.5.2a Descripción de pines

Voltaje de alimentación VDD (pin 1).

El rango del voltaje de alimentación para el funcionamiento del dispositivo debe de estar entre 2.7V -5.5Vdc.

Selector de dispositivo CS (pin 2).

CS es la entrada de selección del dispositivo, que es activado en el nivel bajo, al cambiar de alto a bajo me habilita el reloj serial y las funciones de datos en el dispositivo conversor.

Reloj de entrada serial SCK (pin 3).

Entrada de reloj proveniente del dispositivo maestro (MCU).

Entrada de datos serial SDI (pin 4).

Entrada de datos provenientes del MCU.

Entrada de sincronización de salidas análogas LDAC (pin 5).

Transfiere los registros de entrada a los registros de salida cuando tiene un nivel bajo. Es utilizado para obtener conversiones simultáneas en ambas salidas del chip conversor (salidas A y B).

Entrada hardware para reposo del dispositivo SHDN (pin 6).

Si se desea colocar el dispositivo en modo de bajo consumo, este pin es utilizado, colocándole una señal de entrada que cambie de alto a bajo.

Salidas de conversión DACx (VOUTA, VOUTB pin 7).

Salidas de conversión análoga las cuales pueden ser activadas por medio de la palabra de activación en la cual se determina cual salida será utilizada si la salida A o B, ambas no pueden ser utilizadas al mismo tiempo.

Referencia de alimentación VSS (pin 8).

Conocida mejor como masa o tierra del circuito (GND).

3.4.5.2b Operación del dispositivo.

El voltaje de salida idealmente esta dado por la ecuación siguiente.

Vout= (2.048V * *G* * *Dn*) / 2^n.

Donde ***G*** es la ganancia seleccionada en la palabra de configuración que puede ser 1x ó 2x.

Dn representa el valor de entrada digital.

n representa la cantidad de bits de resolución que es de 12 bits.

La diferencia entre 2 códigos sucesivos viene dada idealmente por el bit LSB, según la ganancia utilizada así resulta el valor de la resolución como se muestra en la tabla 3.1.

GANANCIA	**RESOLUCIÓN**
1X	2.048v /4096.
2X	4.096V /4096.

Tabla 3.1. Resoluciones dadas por cada ganancia.

El comando utilizado para configurar al dispositivo consiste en una palabra (16 bits), en el cual se determina que tipo de ganancia se utilizará, la salida a utilizar y el valor digital a convertir.

Upper Half:

W-x	W-x	W-x	W-0	W-x	W-x	W-x	W-x
$\overline{A}$/B	—	$\overline{GA}$	$\overline{SHDN}$	D11	D10	D9	D8
bit 15							bit 8

Lower Half:

W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
D7	D6	D5	D4	D3	D2	D1	D0
bit 7							bit 0

bit 15 **$\overline{A}$/B:** DAC_A or DAC_B Select bit
1 = Write to DAC_B
0 = Write to DAC_A

bit 14 — Don't Care

bit 13 **$\overline{GA}$:** Output Gain Select bit
1 = 1x ($V_{OUT} = V_{REF} * D/4096$)
0 = 2x ($V_{OUT} = 2 * V_{REF} * D/4096$)

bit 12 **$\overline{SHDN}$:** Output Power-down Control bit
1 = Output Power-down Control bit
0 = Output buffer disabled, Output is high-impedance

bit 11-0 **D11:D0:** DAC Data bits
12-bit number "D" which sets the output value. Contains a value between 0 and 4095.

Fig.3.27 *Palabra de configuración del dispositivo MCP4822.*

El dispositivo MCP esta diseñado para interactuar con dispositivos que posean el protocolo SPI, soportando el modo 0,0 y modo 1,1 (explicados en la sección 1.8.2 del capitulo 1), el envío de los bits de configuración y datos hacia el dispositivo conversor se realiza a través del puerto SPI, capturando cada bit en el flanco de subida de cada ciclo de reloj .El diagrama de tiempo para la escritura hacia el dispositivo es mostrado en la figura 3.28.

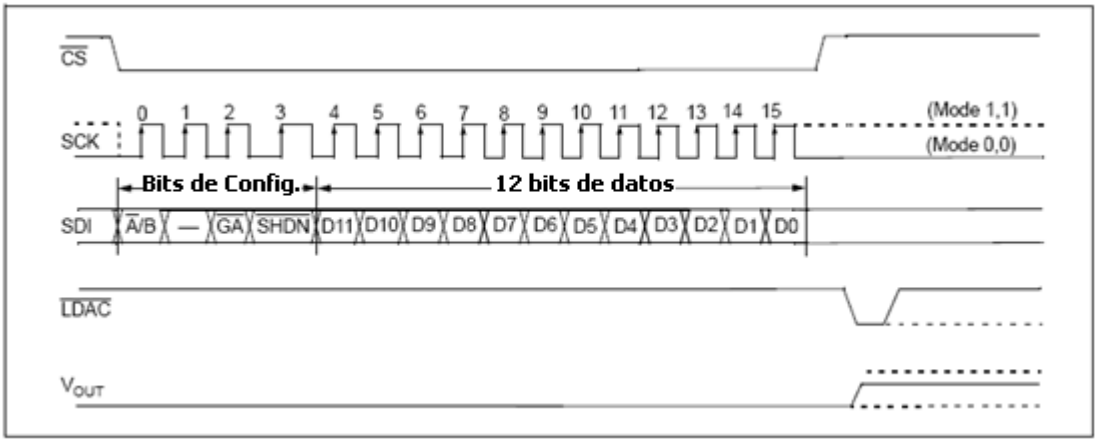


Fig.3.28 *Secuencia de escritura en el dispositivo conversor MCP4822*.

3.4.5.2c Conversión de un dato.

La secuencia de conversión se inicia al activar (con nivel bajo) el pin de entrada CS, el cual me habilita la entrada de reloj y de datos para la comunicación SPI, luego se envía la palabra de configuración que consiste en 2 bytes, iniciando con el envío del byte mas significativo que contiene los 4 bits de configuración y la parte alta del dato a convertir, el siguiente byte es el que contiene los 8 bits restantes del código a convertir. Para obtener el valor de salida debe de generarse un pulso alto en CS después de haber enviado los 16 bits, para activar las salidas y que el dato este disponible en el registro de salida. El dispositivo utiliza una estructura de salida de doble buffer permitiendo que ambas salidas (A, B) sean sincronizadas utilizando el pin LDAC.

La ganancia puede ser determinada en la palabra de configuración, la ganancia es la encargada de generar el rango de voltaje que será utilizado en las conversiones, así si se utiliza ganancia de 1 (G=1) el rango será desde 0.00V hasta un máximo voltaje de 4095/4096 *2.048V. Utilizando una ganancia de 2 (G=2) el rango estará entre 0.00V y un voltaje máximo ideal dado por 4095/4096*4.096V, cabe mencionar que el voltaje máximo para una ganancia de uno es cercano al valor de voltaje de referencia interno del dispositivo que es de 2.048V y el máximo valor con ganancia 2 es el doble del voltaje de referencia (idealmente).

3.4.5.3 LECTURA/ESCRITURA DE DATOS EN EEPROM 24C01C (I²C).

Esta etapa es la encargada de establecer conexión usando el protocolo I2C con el dispositivo de memoria EEPROM 24C01C, con este dispositivo pueden leerse ó escribirse datos desde el PIC16F877/A de la tarjeta principal, el circuito de esta etapa se muestra en la figura 3.29.

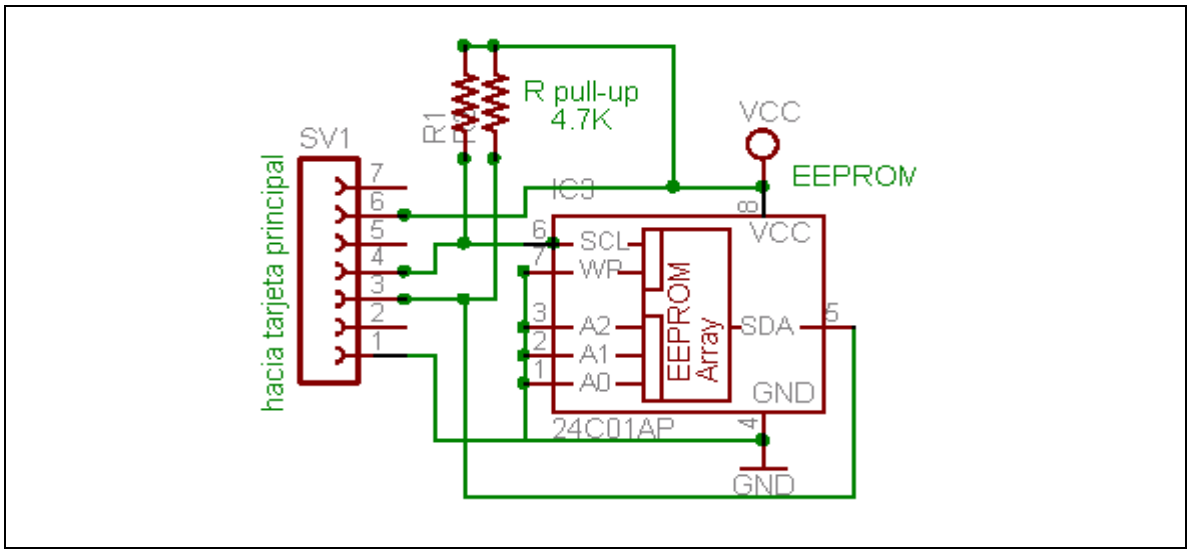


Fig.3.29 Circ*uito etapa memoria serial 24C01C de la tarjeta I²C-SPI*.

3.4.5.4 EEprom serial 24C01C.

http://ww1.microchip.com/downloads/en/DeviceDoc/21201J.pdf

La memoria de datos EEPROM es un dispositivo con capacidad de 1K bit ,es serial y eléctricamente borrable , trabaja con un rango de voltaje de 4.5V a 5.5V .La memoria esta organizada en un bloque de 128 *8 bit ,el cual es accesado por medio del protocolo I2C , solo necesitando 2 conexiones físicas. Por su bajo consumo de potencia permite operar con corrientes típicas de solo 10uA (reposo) y 1mA (lectura),su capacidad total permite guardar hasta 16 bytes de datos con una velocidad de 1ms,en el bus I2C se pueden conectar hasta 8 dispositivos de memoria logrando hasta una capacidad total de 8Kbit de memoria EEPROM continua.

CARACTERISTICAS:

- Compatible con protocolo I²C.
- Entradas del tipo Schmitt Trigger para supresión de ruido.
- Compatible con relojes de 100KHz y 400KHz.
- Buffer con capacidad de hasta 16 bytes.
- Ciclo de borrado /escritura.
- Tiempo típico de escritura de 1ms.
- Protección ESD de voltajes mayores a 4.00V.
- Permite mas de un millón de ciclos de borrado /escritura.
- Retención de datos arriba de 200 años.
- Rangos de temperatura Industrial (-40c a 85c) y en automóviles (-40c a 125c).

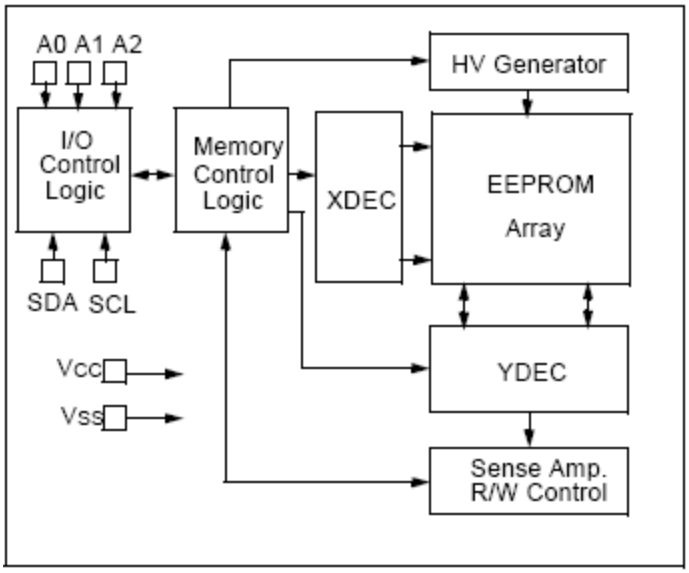


Fig.3.30 *Diagrama de bloques memoria EEPROM.*

3.4.5.4a Descripción de pines.

Entrada de datos serial SDA.

Entrada bidireccional usada para la transferencia de direcciones/datos ya sean desde la memoria o hacia la memoria. Este pin tiene un terminal de drenador abierto ,por eso el pin SDA requiere una resistencia pull-up conectada a Vcc , aproximadamente de 10K (100KHz) o 2K (400KHz).

Para transferencia normal el pin SDA permite los cambios solo en los niveles bajos de cada ciclo de reloj dados por SCL. Los niveles altos son especialmente para indicar las condiciones de START y STOP.

Reloj serial SCL.

Reloj para sincronizar la transferencia de datos desde o hacia el dispositivo de memoria EEPROM.

Entradas de direccion A0, A1, A2.

Pines para establecer la dirección que poseerá el dispositivo en el bus I2C, este valor es comparado con el valor enviado por el dispositivo maestro para establecer conexión. Hasta 8 combinaciones se pueden realizar, es decir 8 dispositivos al mismo tiempo en el bus I2C.

Test

Este pin es usado específicamente para propósitos de prueba, debe de estar ya sea en estado alto o bajo.

3.4.5.4b Operación del dispositivo.

El dispositivo soporta comunicación bidireccional utilizando solamente 2 cables, a través del protocolo I2C, el dispositivo que envía datos hacia el bus se conoce como maestro o emisor y el que recibe los datos es el receptor .El maestro controla el reloj (SCL), en el bus de comunicación, controla además el acceso al bus, genera la condición START y STOP, mientras que el dispositivo de memoria EEPROM actúa como esclavo.

Operación de escritura

Un byte de control es enviado hacia la memoria después de que el maestro ha generado el bit de START, este byte de control consiste en 4 bits de control específicos del dispositivo 24C01C que son “1010”, los bits restantes son para completar la dirección (A0, A1, A2) que distingue al dispositivo en el bus I2C,el bit restante indica si se realizara escritura (0) o lectura en el dispositivo (1) ,(figura 3.31).

Después de recibir y verificar si la dirección enviada es igual a la del dispositivo esclavo, éste genera en la línea SDA una señal de reconocimiento (ACK) en el noveno ciclo de reloj enviado por el maestro ,el esclavo debe de dejar la línea SDA en estado alto para permitir que el maestro pueda generar otro pulso de START.

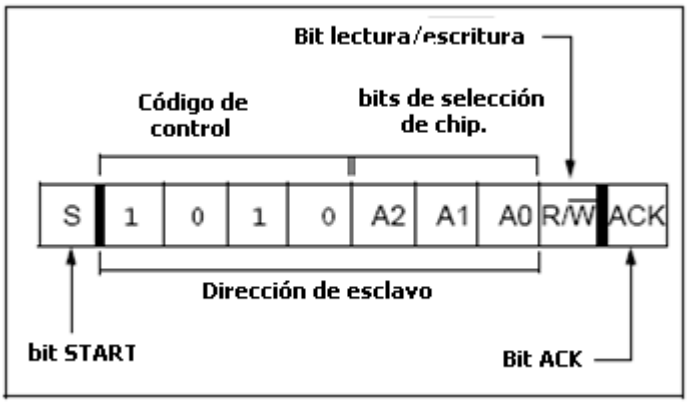


Fig.3.31 *Byte de control.*

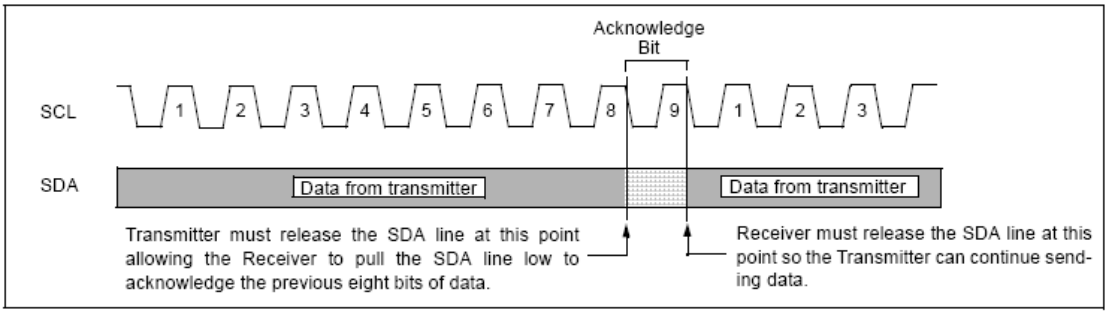


Fig.3.32 *Diagrama de tiempo de la generación de la señal ACK.*

Operación de escritura de un byte.

Después de haber recibido la condición de START y el byte de control indicando que se realizará una operación de escritura al dispositivo esclavo, éste generará una señal ACK y se quedará esperando el dato siguiente que será el byte de la dirección de memoria donde se desea guardar el dato, el byte de dirección será guardado en el puntero de dirección de la memoria EEPROM, generándose así otro bit ACK. El siguiente byte es el dato a guardar y el esclavo generará otro bit de reconocimiento ACK.

Para finalizar la secuencia de escritura el maestro envía el bit de STOP para indicar que terminó de enviar datos, el dispositivo 24C01C internamente realizará la escritura del dato en la dirección deseada, Figura 3.33.

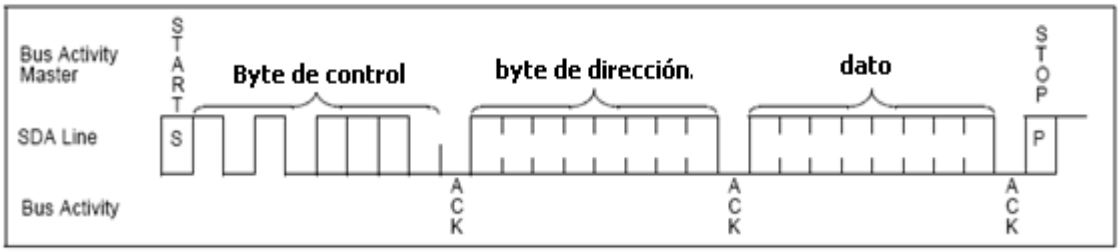


Fig.3.33 *Diagrama de escritura de un byte.*

Operación de escritura de una página.

El byte de control , el byte de direcciones y el byte del primer dato es enviado en la misma manera que en la escritura de un byte, la diferencia es que después del primer dato se envían la cantidad de datos que llenarán el espacio de pagina de memoria del dispositivo (16 bytes),el esclavo generará después de cada byte recibido la señal ACK, los 4 bits menos significativos del puntero de dirección se incrementan internamente cada vez que se reciba un byte de datos, logrando con esto cubrir todas las direcciones ,cuando se envíe el byte 15 el maestro generará el bit de STOP. Los 16 datos serán guardados en los buffer de datos de la memoria 24C01C, hasta recibir el bit de STOP se iniciará la secuencia interna de escritura a la memoria EEPROM Figura. Si el maestro envía mas de 16 datos antes de generar la condición de STOP, el contador volverá a la primera posición de memoria entonces se dará una sobre escritura de datos.

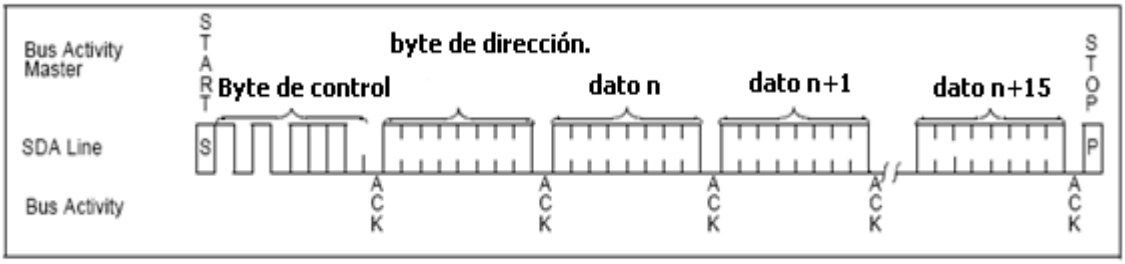


Fig.3.34 *Diagrama de escritura de una página.*

Operación de lectura.

Lectura aleatoria de una dirección de memoria.

La memoria 24C01C contiene un contador de direcciones que mantiene la dirección de la ultima palabra accesada, éste contador internamente se incrementa cada vez que se realiza una operación de lectura , si el contador estaba en la dirección n la siguiente a leer será la dirección n+1 ,al recibir la palabra de control que indica que se realizará una lectura el esclavo genera una señal ACK ,enviando los 8 bits al maestro que coloca el bit de STOP en la línea SDA indicándole al esclavo que termine la transmisión de datos, ésta secuencia de lectura se muestra en la figura 3.35.

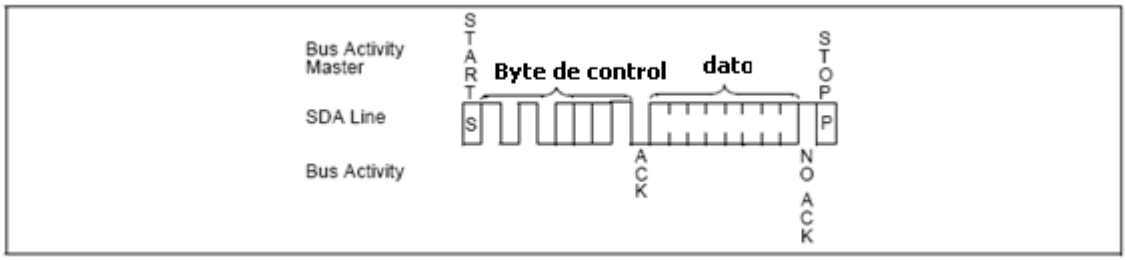


Fig.3.35 *Lectura de una dirección aleatoria.*

Lectura secuencial.

Una lectura secuencial es iniciada en la misma manera de una lectura normal, el maestro después de recibir el primer dato genera un bit ACK indicándole al esclavo que sigua enviando datos , el puntero de direcciones del esclavo se incrementa en 1 en cada operación de lectura con esto se logra realizar una lectura de la pagina de memoria en una sola operación, el puntero de dirección interna automáticamente se reinicia (00) al llegar al ultima posición (7F).Para terminar la operación de lectura el maestro debe enviar el bit de STOP para indicarle al esclavo que termine el envío de datos.

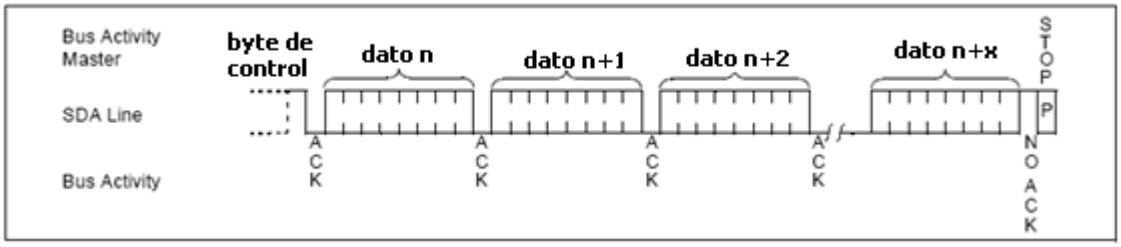


Fig.3.36 Lectura secuencial de direcciones.

Lectura especifica.

Con este tipo de lectura el maestro puede accesar a una determinada localidad de memoria para ser leída, la secuencia inicia enviando el byte de control desde el maestro hacia el esclavo indicando una operación de escritura ,después se envia un byte que me indicará la direccion que se desea leer .El maestro generará otro bit de START para enviar la palabra de control que indica que se realizará una lectura , el esclavo genera un bit ACK y envía el dato solicitado al maestro ,el maestro depués de leer el dato genera el bit de STOP indicando el fin de la secuencia. Debe tomarse en cuenta que después de esta operación el contador de direcciones se incrementara en 1 quedandose en la siguiente dirección a la leída anteriormente.

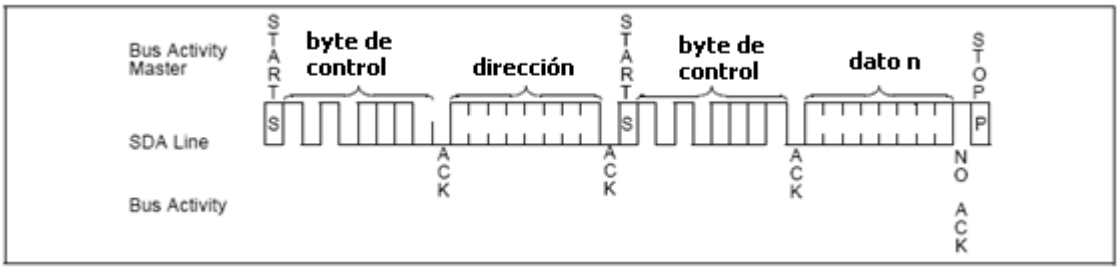


Fig.3.37 *Lectura de dirección especifica.*

3.4.5.5 Comunicación I²C o SPI entre dos microcontroladores.

Esta etapa es utilizada para conectar el PIC16F877/A de la tarjeta I2C-SPI con el PIC16F877/A de la tarjeta principal por medio del protocolo I2C ó SPI, logrando así transmitir datos análogos y digitales desde un MCU a otro MCU. Debe tenerse en cuenta que el sistema esta diseñado para que el PIC de la tarjeta principal actúe como maestro y el PIC de la tarjeta I2C-SPI como esclavo. Debe especificarse que tipo de comunicación se establecerá por medio del selector de la tarjeta I2C-SPI, el diagrama de bloques de conexión de ambas tarjetas se muestra en la figura 3.38.

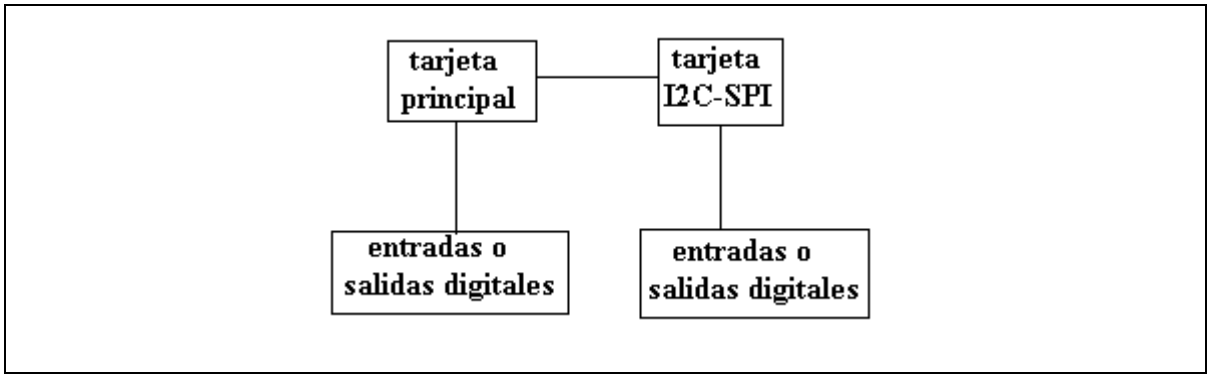


Fig.3.38 *Diagrama de bloques de conexión de ambas tarjetas*.

Ambas tarjetas pueden hacer uso de los módulos de entrada o salida digital por lo que resulta muy versátil a la hora de la comunicación enviando o recibiendo datos digitales y análogos ya que tanto la tarjeta principal como la tarjeta I²C-SPI contienen circuitos de lectura análoga.

El diagrama del circuito utilizado en esta etapa se muestra en la figura 3.39.

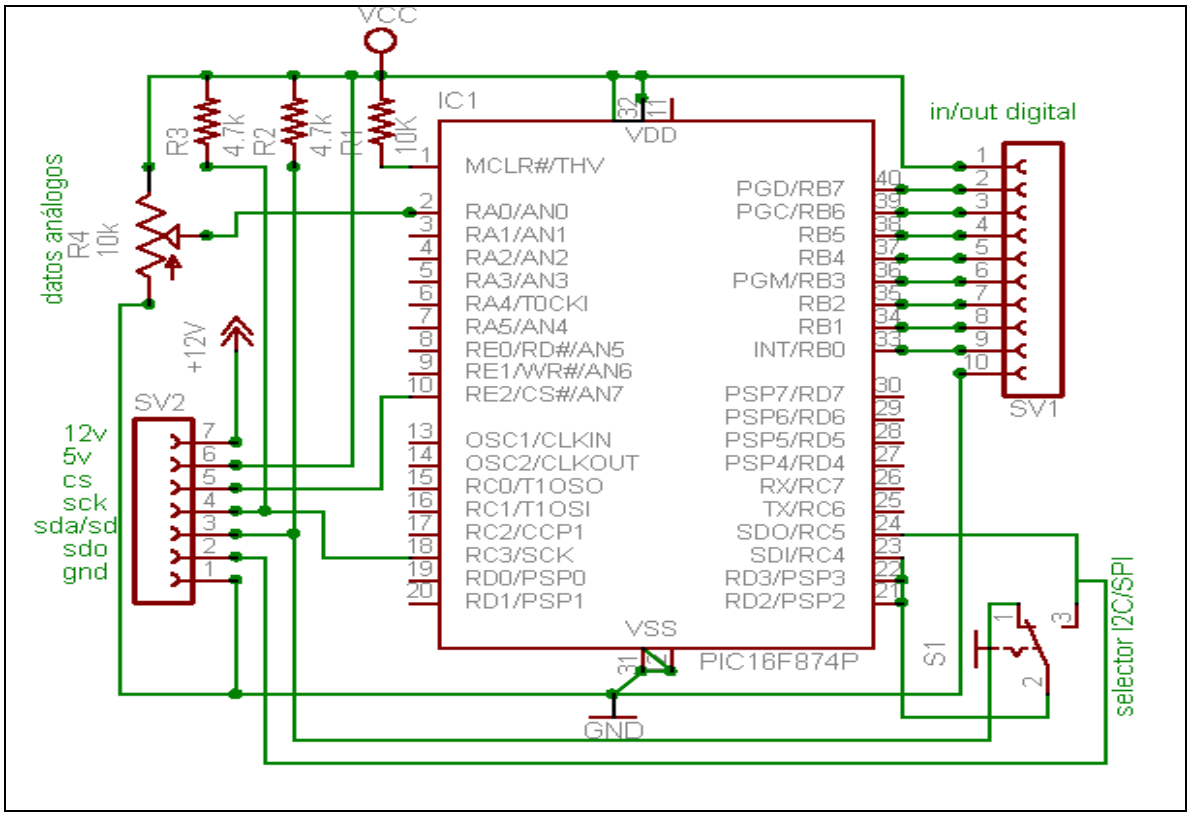


Fig. 3.39 *Diagrama conexión del PIC877/A en el módulo I²C-SPI*.

3.4.6 COSTOS DEL SISTEMA

Detalles del costo de la tarjeta debug:

Chips FTD232BM + PICs 16F876/A y Cristal de 6MHz (muestras, solo envío)…….. $5

Resistencias, capacitores varios y conector USB……$5

Bases para integrados..……$3

Cristal de 20MHz……...$1

Placa de circuito impreso……$7

mano de obra……..……..$12

Total	**$33**

Detalles costo módulo principal:

PICs 16F877/A (muestras, solo envío)……..…… $5.00

Resistencias, capacitores, diodos y conectores varios…… $4.00

Chips Max2……..……$1.50

LM358 y potenciómetro de 10KOhm……..……..……$1.00

Cristal de 4MHz……$1.00

Regulador de voltaje 7805……..…….. $0.50

Tarjeta de circuito impreso……..$7.50

Conector DB9……..……..$1.25

Switchs y conectores con tornillo……..$2.25

Total	**$24.00**

Detalles costo módulo entrada digital

Tarjeta de circuito impreso……..$1.50

Minidips y resistencias……..…… $1.50

Total	**$3.00**

Detalles costos módulo salida digital

Tarjeta de circuito impreso…….. $1.50

Leds y resistencias……..…… $1.25

Total	**$2.75**

Detalles módulo para relés y motor paso:

Relés de 12Vdc	$5.25
Transistores TIP120	$4.00
ULN2003A y Conectores con tornillo	$2.30
Resistencias varias	$1.00
Placa de circuito impreso	$5.00
Total	**$17.55**

Detalles módulo I²C-SPI:

PIC16F877/A, MCP4822, Memoria 24C01C (solo gastos de envío)	$5.00
LM358	$0.50
Cristal de 4MHz	$1.00
Potenciómetro de 10KOhm	$0.50
Resistencias, capacitores y switch	$2.00
Conectores varios	$1.00
Placa de circuito impreso	$5.00
Total	**$15.00**

Extras:

Base de acrílico	$3.50
Caja de aluminio	$3.50
Total de hardware de aplicaciones	**$69.30**

Gastos de mano de obra:

Horas trabajadas diarias	2 horas.
Precio por hora	$0.60
Total de horas por semana	10
Total de semanas	8
Total mano de obra aproximado	$48

Costo Total del sistema sin mano de obra	**$102.30**
Costo total del sistema con mano de obra	**$150.30**

El precio unitario de un depurador ICD2 en el mercado oscila entre $50 y $70, más gastos de envío; Por lo que comparado con la tarjeta depuradora hecha cuyo precio es de $33, el precio se reduce considerablemente siendo ésta una opción muy aceptable para fines didácticos o de aprendizaje.

Las tarjetas de entrenamiento básico para PIC poseen lo esencial para desarrollar pequeñas aplicaciones, en el mercado su precio oscila entre $40 y $80 http://www.microchipdirect.com/ProductSearch.aspx?Keywords=DV164007#, si se desea obtener módulos con prestaciones extras como tarjetas para control de motor o tarjetas para utilizar los tipos comunicación que posee el PIC se debe incurrir en gastos extras que elevan el costo total del sistema ya que cada módulo por separado cuesta desde los $30 o más http://www.futurlec.com/PIC16F877_Controller.shtml
http://microcontrollershop.com/product_info.php?products_id=1913
http://search.digikey.com/scripts/DkSearch/dksus.dll?vendor=0&keywords=16F877AEMBCDEVKIT-ND
Nuestro sistema ofrece diferentes módulos cuyos precios están detallados en la sección 3.4.6 de este capitulo , observamos que el precio mas elevado de uno de nuestros módulos es de $24, comparado con los módulos del mercado concluimos que su precio es menor ;más aun si se utilizan piezas como conectores o dispositivos reciclados el precio puede llegar a ser menor.
Si a sumimos que necesitamos comprar 5 módulos a $30 más la tarjeta debugger a $70 el costo del mercado es de aproximadamente $ 220 mas gastos de envió . El costo del sistema presentado es aproximadamente de un 35% menor que el costo de un sistema comprado en el mercado.Debido a que el sistema está compuesto por módulos el estudiante o quien desee realizarlo puede desarrollar solo la tarjeta debugger y el módulo principal ,con esto se logra obtener el 70% del sistema, los demás módulos son opcionales o complementarios.

Capitulo cuatro: Manual de usuario.

El manual de usuario es una herramienta especifica para utilizar el sistema implementado en conjunto con el software libre MPLAB. Esta guía está orientada a personas con conocimientos básicos de programación de microcontroladores PIC.

Basándose en esta guía y utilizando el sistema creado un docente puede crear sus propios ejemplos o utilizar ejemplos de libros ,puede comprobar dichos programas con sus alumnos creando una clase práctica más interactiva en la cual los alumnos comprendan muchas ideas que solo con la teoría no pueden comprenderse.

Desde la perspectiva del alumno ya sea guiado por un docente o trabajando de manera autodidacta ,ésta guia en conjunto con el sistema le sirve para repasar los ejemplos absorbidos en clase , desarrollar ejemplos de libros , internet o ejemplos creados por ellos mismos. Al desarrollar un programa se comprueba si en verdad el código implementado realiza lo que en teoría debería hacer ,por eso el mayor provecho que el alumno podria obtener de este sistema, es saber donde están los errores y cómo corregirlos, así no solo entiende los programas sino que es capaz de analizarlos y hacer cambios en el código o crear nuevas ideas que mejoren la idea existente.

A lo largo de este capitulo se van detallando de manera sencilla pasos e ideas para comprender mejor el sistema implementado.

4.1 SOFTWARE MPLAB-IDE

www.**uniovi**.es/ate/alberto/**Entorno**%20**MPLAB** v7xx.pdf

El software libre MPLAB desarrollado por la compañía Microchip es un sistema integrado para desarrollar aplicaciones utilizando microcontroladores PIC o DsPICS[12] , es un programa que corre bajo Windows, presenta las clásicas barras de programa, de menú, de herramientas de estado, etc. El ambiente MPLAB® posee editor de texto, compilador, simulador y la opción de conectarlo a la tarjeta debug para depuración en tiempo real. Para comenzar un programa desde cero para luego depurarlo y grabarlo al MCU se realiza lo siguiente:

[12] **DsPICS** microcontroladores para tratamiento de señales.

1. Crear un archivo con extensión .ASM.
2. Crear un Proyecto nuevo eligiendo dispositivo, nombre y ubicación.
3. Agregar el archivo .ASM como un archivo fuente (SOURCE FILE) al proyecto.
4. Compilar el código realizado comprobando errores.
5. Utilizar las herramientas ya sea de simulación, programación o depuración.

NOTA:

Antes de iniciar a utilizar el sistema se debe instalar el software MPLAB-IDE versión 8.02 brindado en la copia digital del documento, además de instalar la carpeta programas _ fínales en el directorio C: / ,disponible también en la copia digital del documento.

Una vez escrito el programa, se procede a la compilación. Para esto, desde el menú PROYECT se elige la opción BUILD ALL (construir todo), si no existen errores, devolverá un mensaje como BUILD SUCCESFULL. Los errores que muestra el compilador son del tipo sintácticos, por ejemplo, mala escritura de las instrucciones, referencia de etiquetas que no existen, etc.

También existen mensajes y advertencias; los mensajes pueden ser, por ejemplo, que se está trabajando en un banco de memoria que no es el deseado. Las advertencias tienen un poco más de peso, por ejemplo, el PIC seleccionado no es el mismo que esta definido en el programa. En ambos casos, mensajes y advertencias, la compilación termina satisfactoriamente pero hay que tener en cuenta siempre lo que nos advierten estos mensajes para prevenir errores.

Terminada la compilación el MPLAB® nos genera un archivo de extensión *.hex* el cual es completamente entendible para el PIC. Con esto, solo resta grabarlo al PIC por medio de la tarjeta debug en modo programador ó modo depurador.

MPLAB soporta la programación de diferentes familias de microcontroladores, por lo que la tarjeta debug es capaz de depurar y programar código para estas diferentes familias:

- PIC 10F/12F/16F50x.
- PIC 16F6xx /16F7xx /16F8xx/16F9xx.
- PIC 18Cxxx /18Fxxx.
- PIC18F2xxx /18F4xxx.
- PICF6xxx /18F8xxx.
- PIC 18Fxxj.
- PIC24F.
- DsPIC30F / DsPIC33F

Al utilizar el sistema por primera vez se debe conectar la tarjeta debug al puerto USB, a través del asistente para hardware nuevo instalar los drivers del chip FTD232BM que se encuentran en la carpeta **CDM 2.02.04 WHQL,** situado en la carpeta *programas usados* de la copia digital del documento**,** con esto se logra que la PC reconozca la tarjeta debug como un puerto COM virtual.

Si durante la utilización del sistema resultan fallas refiérase a la sección 4.11 de este capitulo.

4.2 COMO DEPURAR UN CÓDIGO DETERMINADO

Los pasos a seguir son:

- Conectar la tarjeta debug a la PC utilizando el puerto USB.
- Conectar la tarjeta debug con el módulo principal de aplicaciones (Fig.4.39).
- Alimentar la tarjeta debug con la fuente externa (15Vdc).
- Abrir software MPLAB versión 8.02.
- Abrir el proyecto que contiene el código a depurar con (*Project – open*), estos se encuentran en la carpeta programas _finales instalada anteriormente. Para el ejemplo 1 abrir el proyecto **ADC877A** (sección 4.5.2), para el ejemplo 2 abrir el proyecto **error** y para el ejemplo 3 abrir **SPImaster877A** (sección 4.9.3).
- especificar la tarjeta en modo debugger (depurador), (sección 4.2.1).
- Especificar dispositivo a utilizar y bits de configuración (sección 4.2.2).

- Establecer parámetros de comunicación (sección 4.2.3).
- Conectar tarjeta debug y programar MCU (sección 4.2.4).
- Realizar depuración (sección 4.2.5).

4.2.1 ESTABLECER TARJETA EN MODO DEBUG.

En la barra de menú seleccionar *debugger* > *select tool* > *MPLAB ICD2,* debido a que la tarjeta construida es reconocida como una tarjeta ICD2.como se muestra en la fig.4.1.

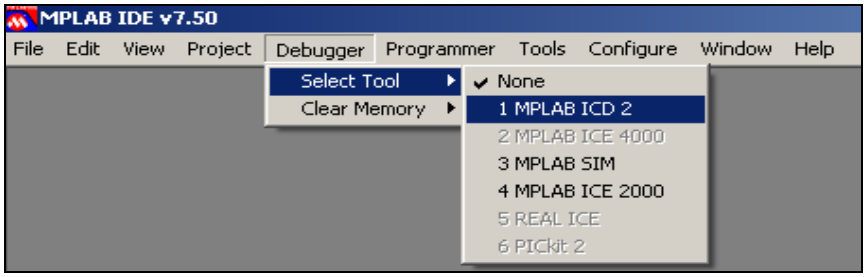


Fig.4.1 *Selección de tarjeta en modo debug.*

4.2.2 ESPECIFICACIÓN DEL DISPOSITIVO Y DE BITS DE CONFIGURACIÓN.

Para que el sistema trabaje de una manera correcta es necesario realizar configuraciones como especificar con que PIC se trabajará ya sea PIC16F877 o PIC 16F877A, para evitar errores a la hora de compilar un código determinado.

Es necesario setear los bits de configuración,en nuestro caso el bit importante será el bit *oscillator* ,el cual se configura escogiendo XT(cristal de cuarzo) ,este bit determina con que tipo de oscilador trabajará el PIC a utilizar determinando la velocidad de realización de las instrucciones.Los restantes bits deben estar desactivados o seteados como *off.*

Para escoger el dispositivo a depurar se deben seguir los siguientes pasos:

Seleccionar *configurar (configure) > seleccionar dispositivo (select device),* en la ventana aparece los distintos dispositivos soportados por la tarjeta, en nuestro caso se deberá escoger ya sea los microcontoladores PIC16F877 o PIC16F877A.

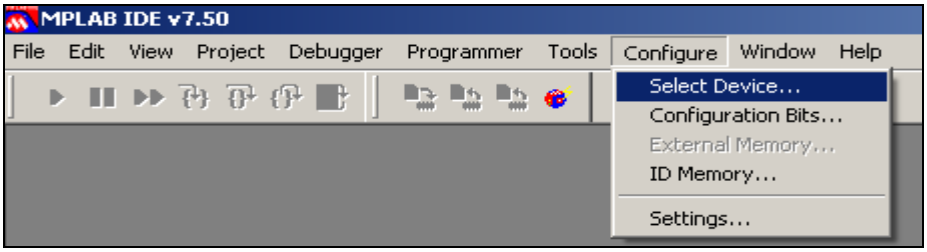


Fig.4.2 *Selección de dispositivo.*

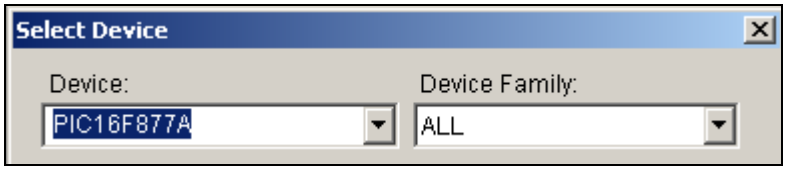


Fig.4.3 Selección de *dispositivo a utilizar.*

Seleccionar *(configure) > (configuración bits),* para establecer los bits de configuración.

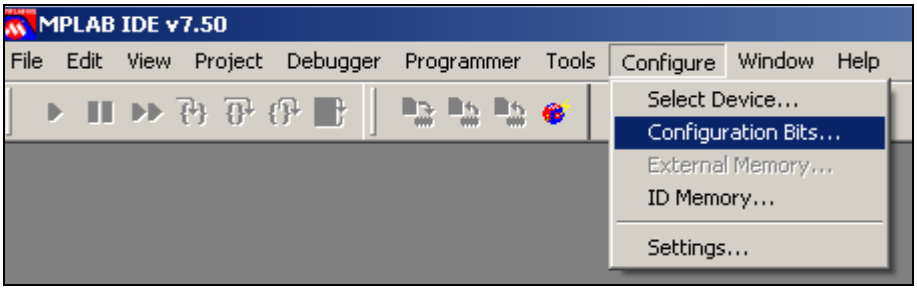


Fig.4.4 *Configuración de bits.*

Para cambiar los bits de configuración hacer click en la pestaña *configuration Bits set in code,* figura 4.6, aparece la ventana de la figura 4.5, hacer click en aceptar.

Configurar cada bit haciendo doble click en el bit a cambiar y seleccionando el modo deseado, los cambios hechos se guardan seteando la pestaña anterior y cerrando la ventana.

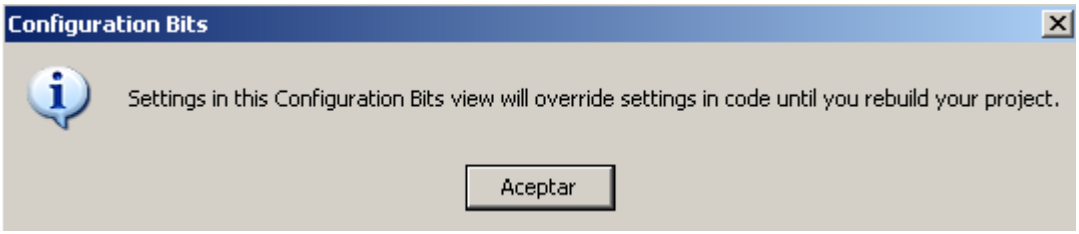


Fig.4.5 *Ventana para acceder a la palabra de configuración.*

En la figura 4.6 se muestra como deben de establecerse los bits para realizar la depuración en los MCUs 16F877/A.

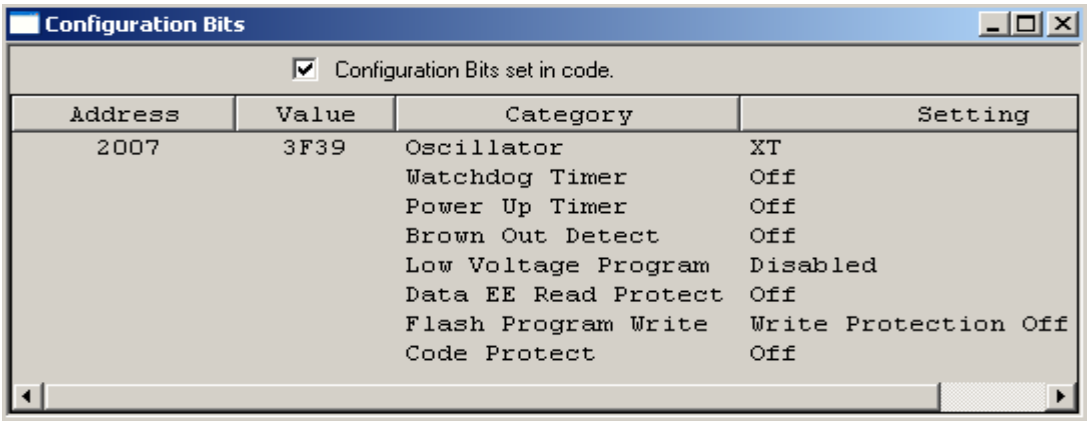


Fig.4.6 *Palabra de configuración MCU16F877/A.*

NOTA:

Con cualquier MCU a utilizar tanto la opción LVP (programación con bajo voltaje) y el watchdog timer deben de permanecer apagados (Off).

4.2.3 ESPECIFICACIÓN DE PARÁMETROS DE COMUNICACIÓN.

Verificar el puerto serie virtual en el cual se conectará la tarjeta debug en la ventana administrador de dispositivos así:

Inicio > Panel de control > sistema > Hardware > administrador de dispositivos.

Nota: El puerto COM virtual varía de computadora a computadora verificar bien donde se conectará la tarjeta debug, en el caso del ejemplo es COM2.

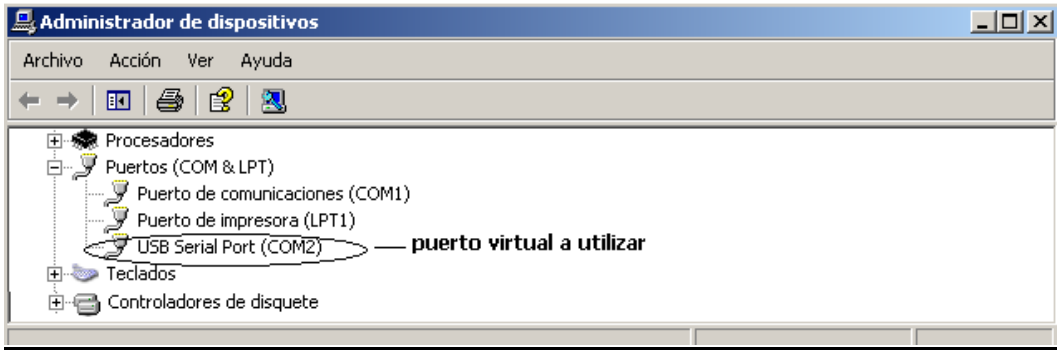


Fig. 4.7 *Ventana de administrador de dispositivos.*

Abrir la ventana de configuración del MPLAB ICD2 *(setup wizard)* Seleccionando *debugger* > *MPLAB ICD2 setup wizard,* aparecerá la ventana de configuración de parámetros en modo debug mostrada en la figura 4.8.

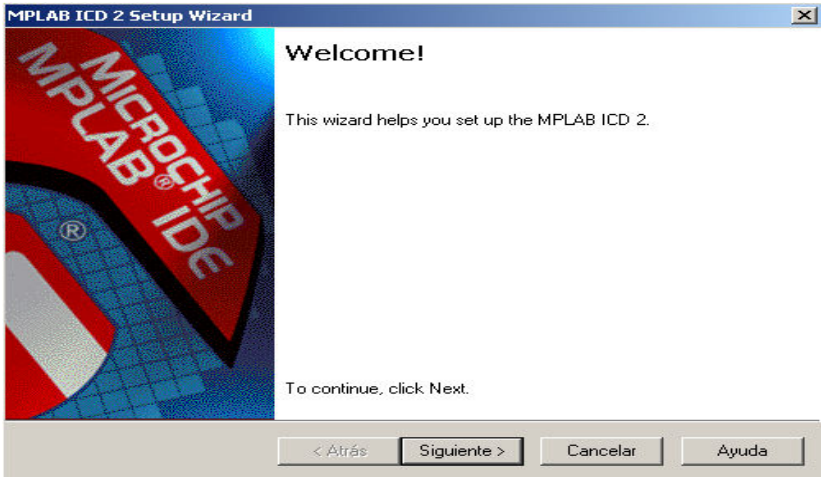


Fig.4.8 *Ventana de configuración del ICD2.*

Seleccionar *siguiente,* para establecer el tipo de comunicación a utilizar como se muestra en la figura 4.9.

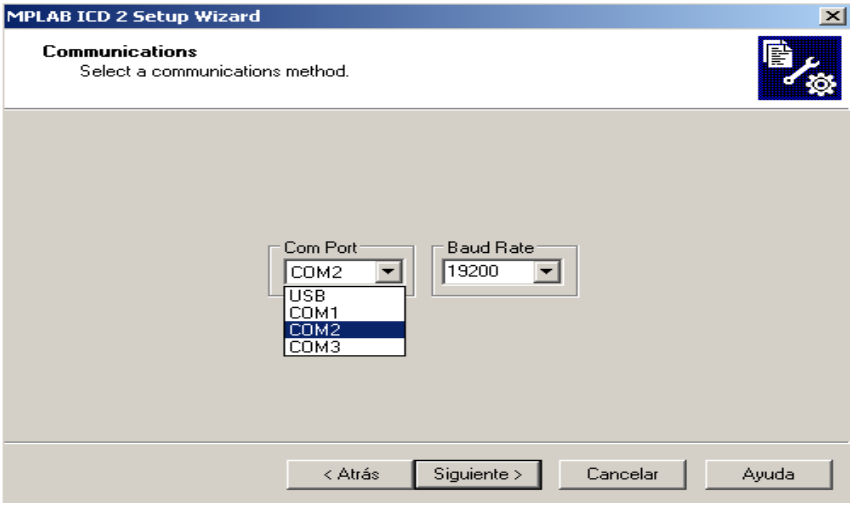


Fig.4.9 *Selección de canal de comunicación.*

Seleccionar *siguiente* para establecer si la tarjeta de aplicaciones se alimentará externamente o desde el MPLAB, es recomendable que sea con fuente externa ya que la corriente que brinda el software no es suficiente para conectarla a la tarjeta de aplicaciones a utilizar, por eso la opción *target has own power supply* debe estar activa como se muestra en la figura 4.10.

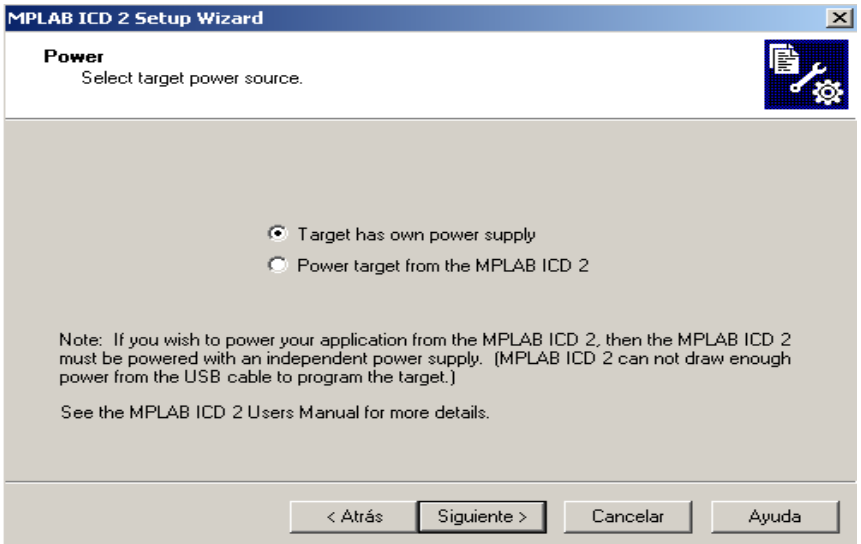


Fig.4.10 *Selección de fuente externa para la tarjeta debugger.*

Seleccionar *siguiente* y desactivar la opción de conexión automática *MPLAB IDE automatically connects to the MPLAB ICD2* ya que solo se conectará cuando el usuario lo desee por medio del botón conectar.

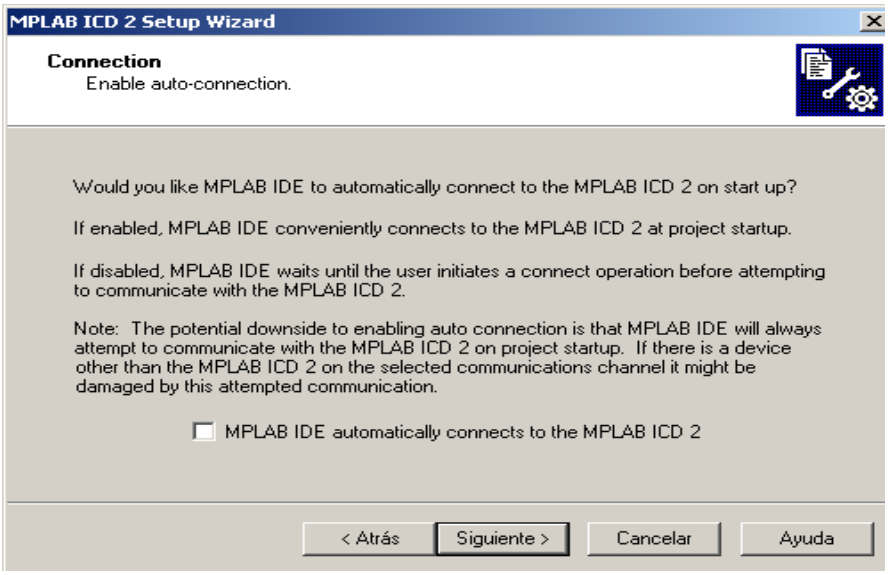


Fig.4.11 *Selección de conexión automática.*

Seleccionar *siguiente* para activar la opción de descarga de nuevo sistema operativo, *MPLAB ICD2 automatically downloads the required operating system* esta condición es utilizada cuando se desea utilizar un MCU diferente al PIC16F877/A.

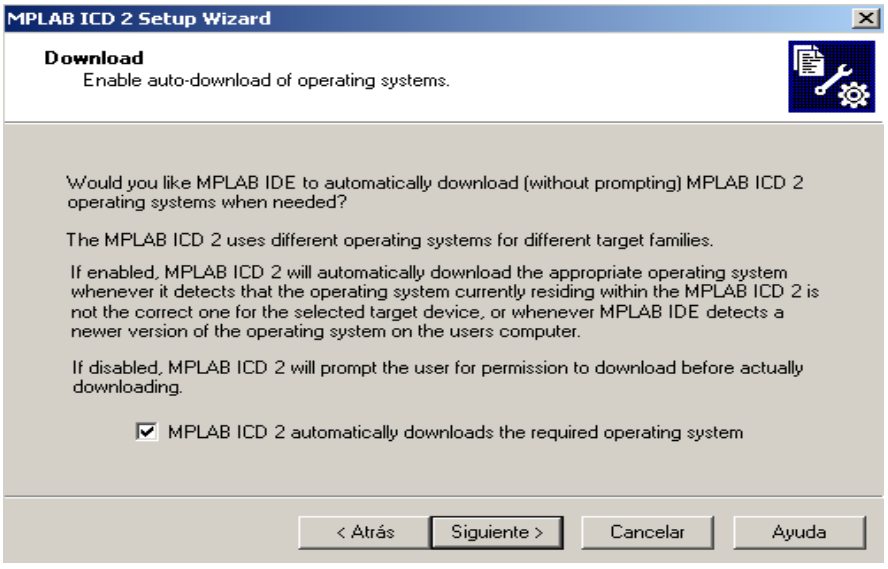


Fig.4.12 *Selección de descarga del sistema operativo.*

Seleccionar *finalizar* para activar todos los parámetros anteriormente establecidos.

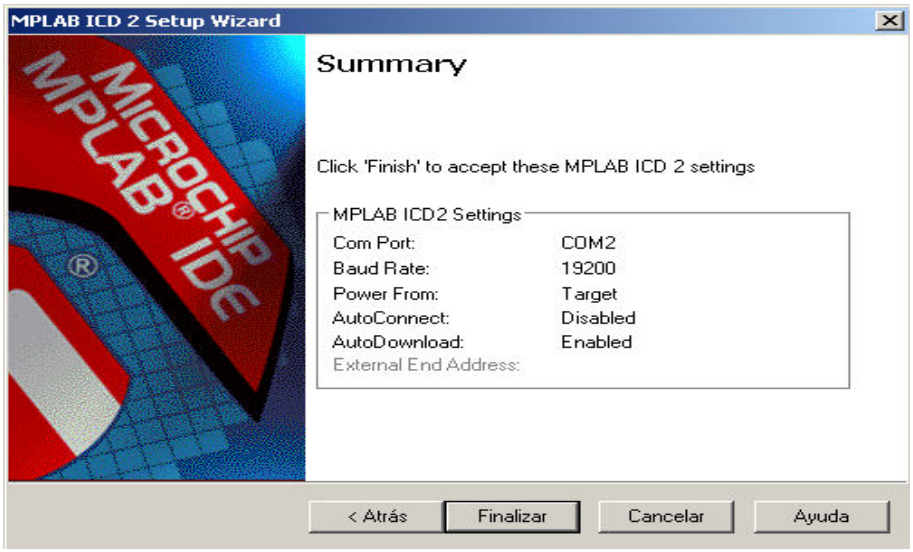


Fig.4.13 *Resumen de parámetros establecidos.*

Después de establecer los parámetros de comunicación aparece un mensaje (fig. 4.14), click en aceptar. No es necesario desactivar los buffer tipo FIFO ya que el convertidor USB-SERIE no posee, esto es necesario sólo cuando se está utilizando un puerto COM normal.

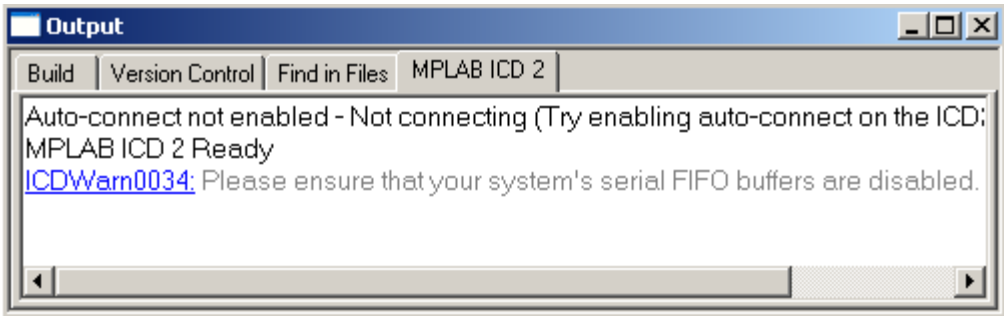


Fig.4.14 *Mensaje de desactivación buffer FIFO.*

4.2.4 CONECTAR TARJETA DEBUG Y PROGRAMAR DISPOSITIVO.

Establecer conexión entre la tarjeta y el software utilizando el botón *reset and conect to ICD*  .Cuando se establece conexión se activa la barra de herramientas fig. 4.15 y se verifica el estado de la conexión en la ventana output fig. 4.16.

Fig.4.15 *Barra de herramientas modo debug.*

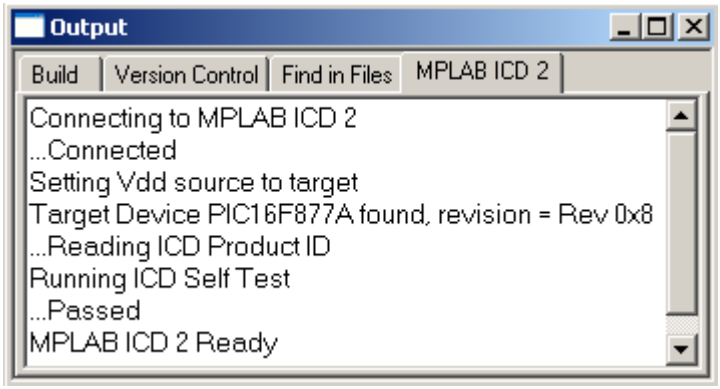


Fig.4.16 *Ventana de conexión del ICD2.*

En caso de fallas refiérase a la sección 4.11 del manual de usuario.
Para descargar el código deben de establecerse las condiciones siguientes:

En la barra de menú seleccionar debugger > settings > program , seleccionar la casilla *manually select memories and ranges* ,seleccionar además las casillas *program , ID , erase all before program* y activar el botón *full range* con esto se logra disponer de toda la memoria de programa del PIC para evitar tener residuos de otros programas depurados anteriormente, click en *aplicar y aceptar*.

Descargar el código a depurar en el MCU por medio del botón *program target device*  de la barra de herramientas en modo debug, en la ventana output puede verificarse si se realizó la descarga satisfactoriamente como se muestra en la fig. 4.17.

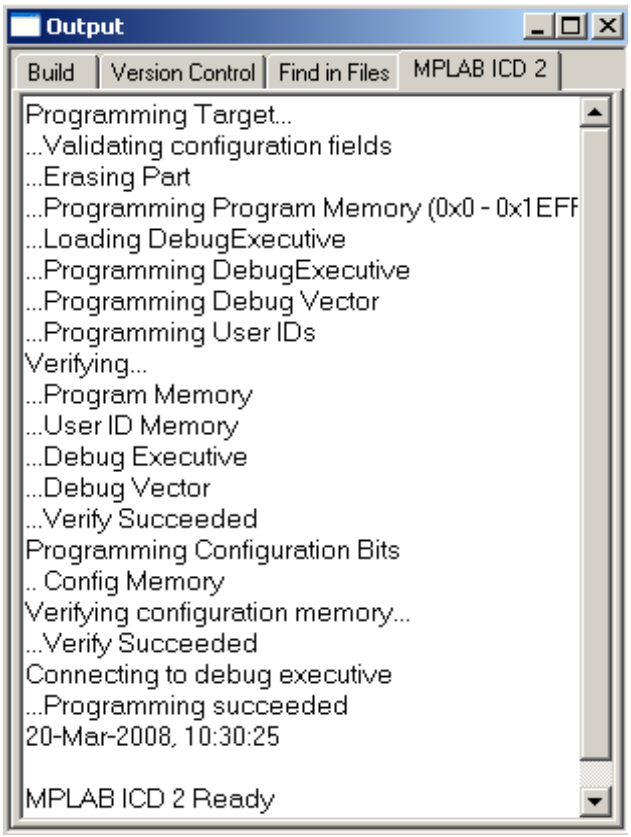


Fig.4.17 *Ventana de programación del dispositivo*.

En caso de fallas refiérase a la sección 4.11 de este capitulo.

Para poder entender de mejor manera la depuración de un código determinado se han creado 3 ejemplos ,en los cuales existen diferentes ideas con las cuales podemos obtener de manera mas concreta como este sistema puede ayudarnos a la hora de desarrollar programas utilizando el sistema implementado.

Antes de desarrollar estos ejemplos es necesario que se haya realizado los pasos descritos en las secciones 4.2, 4.2.1, 4.2.2, 4.2.3 y 4.2.4.

4.2.5 REALIZAR DEPURACIÓN UTILIZANDO EJEMPLO 1

Conversor Análogo-Digital y utilización de localidades de memoria.

Ejemplo de depuración utilizando programa para uso del módulo análogo/digital **ADC877A** (sección 4.5.2), programa que consiste en leer datos análogos y guardarlos en tres localidades de memoria RAM a partir de la dirección 0x40, al llegar a la última localidad se guarda el valor 0xFF en el registro *dato*.

Después de haber realizado la configuración, conexión y programado se realiza:

Establecer los registros a verificar seleccionando *View* > *Watch,* en dicha ventana se escogen los registros que se desean visualizar acordes al programa como se muestra en la figura 4.18.

- En la opción *Add SFR* pueden escogerse los registros específicos del dispositivo como ADCON1, TRISA, PORTC, el registro de trabajo W y más.
- En la opción *Add Symbol_* aparecen los registros de propósito general creados por el usuario, como lo es en este ejemplo el registro *dato*.

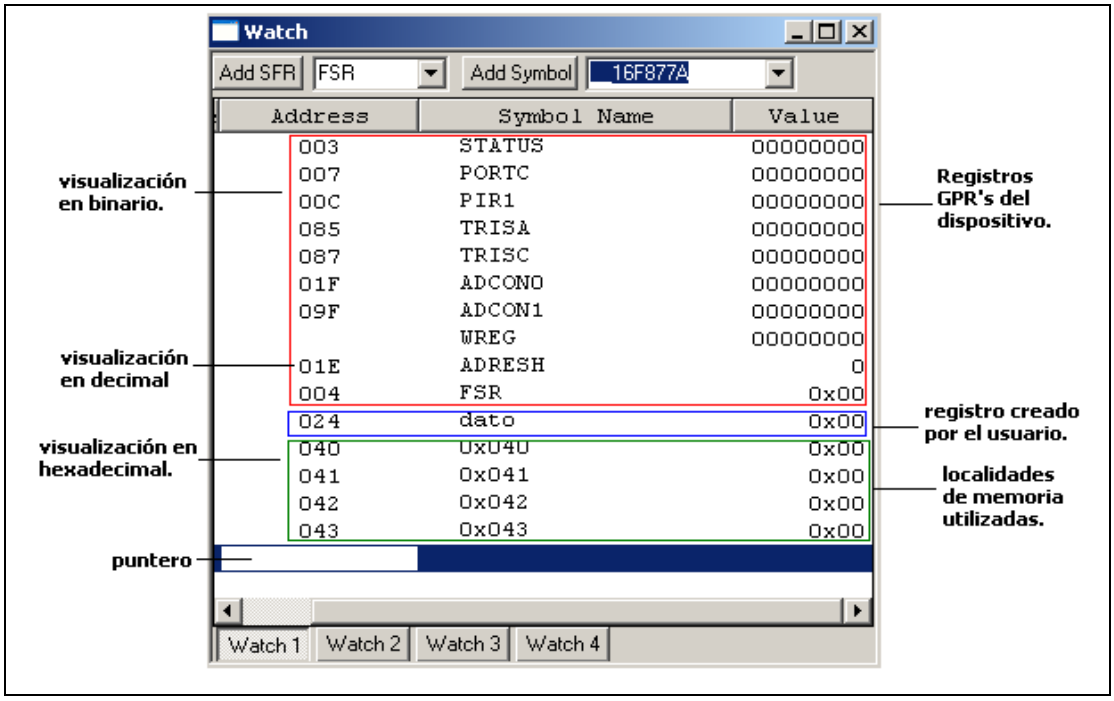


Fig. 4.18 *Registros de propósito general utilizados en el programa.*

Limpiar los registros de propósito general a utilizar para realizar la depuración de forma correcta, se realiza con *Debugger* > *Clear memory* >*GPRs,* figura 4.19.

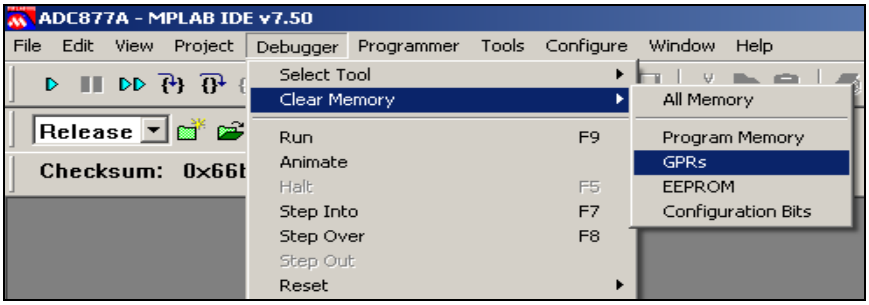


Fig.4.19 *Limpiar registros de propósito general a utilizar.*

No todos los registros se limpian o se ponen con el valor cero, ya que existen registros como STATUS, WREG y otros que poseen un valor por default después de un reset.

Para cambiar la forma de visualización de los registros, hacer click derecho en el registro a modificar, seleccionar *properties*, en el cuadro *format* escoger en que formato se visualizará el resultado ya sea binario, decimal, etc. Click en aceptar.

Para establecer direcciones de memoria a visualizar, colocar el puntero como se muestra en la figura 4.18 ingresando el valor de la dirección y presionando enter.

Para realizar un reset en el MCU a depurar se utiliza el botón *reset* , con esto se logra volver el contador del programa al inicio del código tanto en MPLAB como en el dispositivo a depurar.

En la ventana de código se observa que el puntero se sitúa en la primera instrucción del programa, se visualiza además que las instrucciones **nop** deben de colocarse en cada programa a depurar siempre al inicio del código, debido a que el depurador hace uso de las 2 primeras localidades de memoria del MCU utilizado, por lo que para evitar sobre escribir código en dichas localidades es necesario colocarlas en esta especifica posición, como se muestra en la fig.4.20.

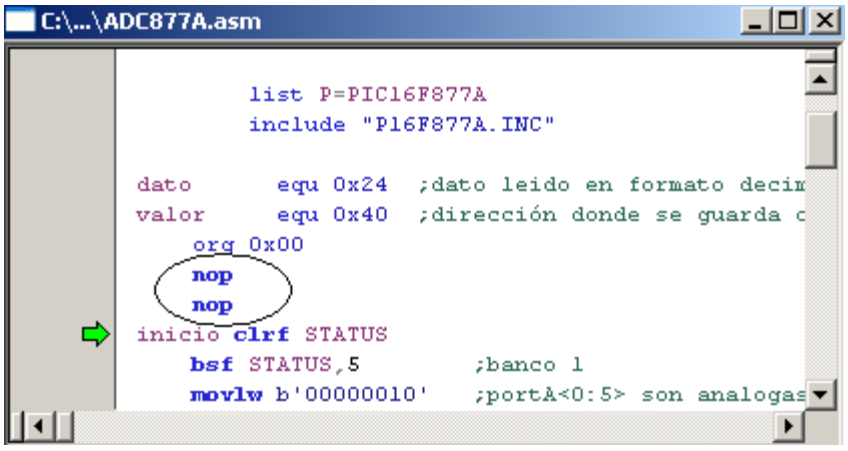


Fig.4.20 *Código ADC877A después de un reset.*

La depuración del código puede realizarse de 3 maneras:

1. De manera continua activando el botón *RUN* y verificando al final los cambios en los registros, para esto debe colocarse un breakpoint haciendo doble click izquierdo en el lugar donde se desea detener el programa, para el ejemplo 1 se coloca al final del programa como se muestra en la figura 4.21.

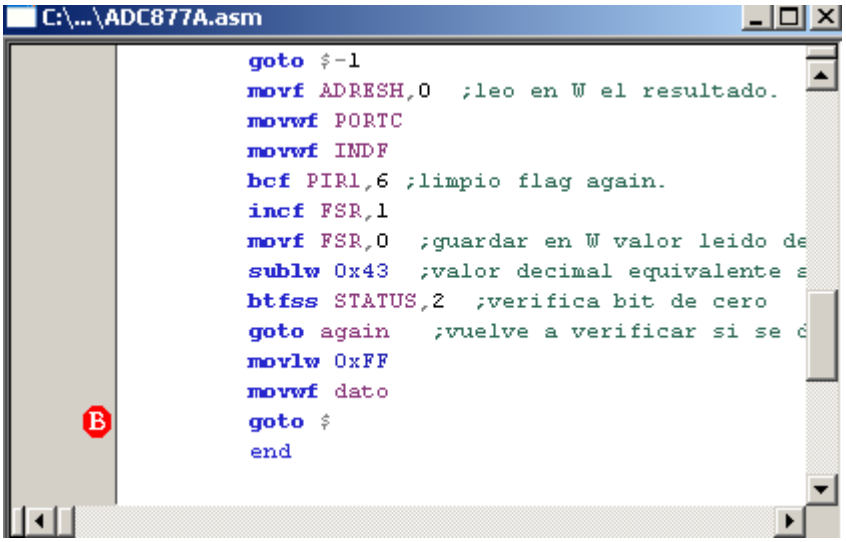


Fig.4.21 *código A utilizando un breakpoint.*

Mover el potenciómetro del módulo principal de aplicaciones a cualquier posición para obtener un valor análogo , activar el botón RUN  .Cuando se ejecuta la instrucción donde se ha colocado el breakpoint el programa se detiene y actualiza con el último dato los registros utilizados, el puntero queda en la última instrucción, como se muestra en la figura 4.22.

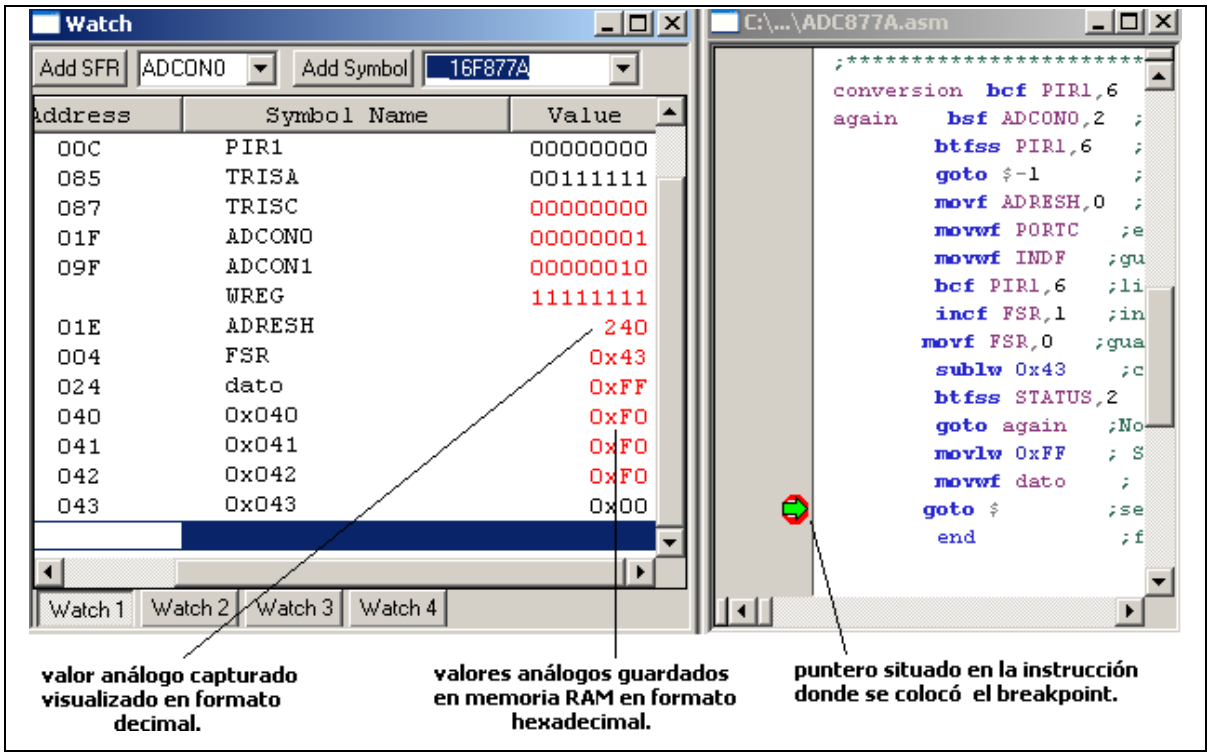


Fig.4.22 *registros actualizados y código finalizado.*

2. Corriendo el código paso a paso con el botón *ANIMATE*  y verificando en Cada instrucción los cambios en los registros utilizados.

Este proceso a diferencia del anterior es más lento ya que ejecuta instrucción por instrucción actualizando cada registro hasta terminar el programa o hasta encontrar un breakpoint.

Colocar un breakpoint en la misma posición que se colocó en la prueba anterior, limpiar los registros utilizados y realizar reset antes de la depuración paso a paso.

Activar el botón *Animate*  para iniciar la depuración paso a paso, cada vez que el puntero esté en la etiqueta *conversión* (Fig. 4.23), debe de moverse el potenciómetro a un valor determinado, con esto se logra obtener diferentes valores análogos para guardarlos en la memoria RAM.

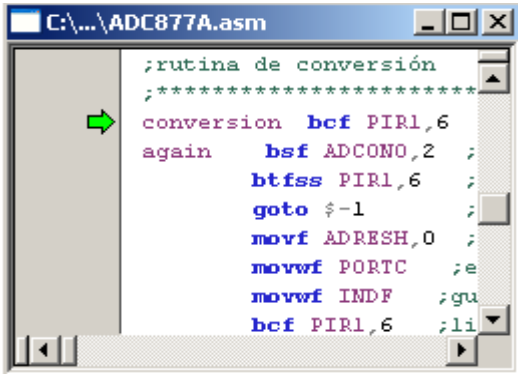


Fig.4.23 *posición del puntero en la etiqueta conversión.*

Cada vez que se ejecuta una instrucción su registro asociado se actualiza, como se observa en la figura 4.24.El programa se detendrá al llegar al breakpoint  .

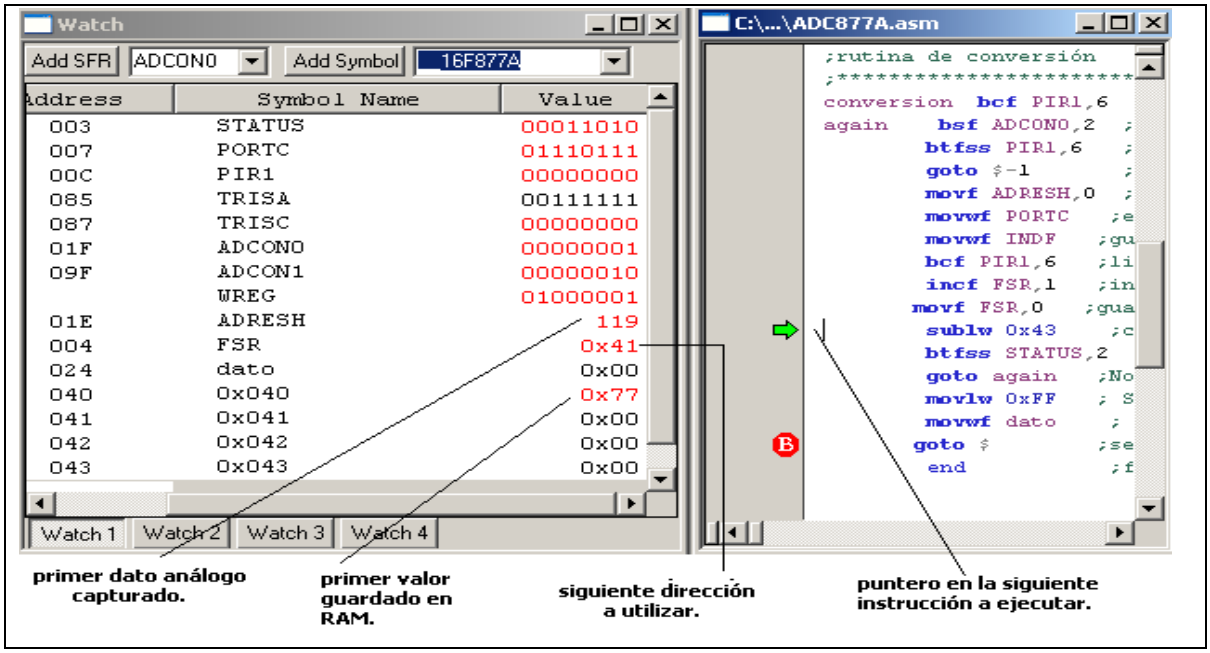


Fig.4.24 *Instrucciones ejecutadas y registros actualizados paso a paso.*

3. Por medio de un solo paso, ejecutando una instrucción que actualiza el registro asociado a dicha instrucción, usando los botones *step Into* y *step Over* .

Cada vez que se presiona uno de estos botones solo ejecuta la instrucción deseada y actualiza el registro asociado a dicha instrucción, el puntero se coloca en la siguiente instrucción a ejecutar, como se muestra en la figura 4.25.

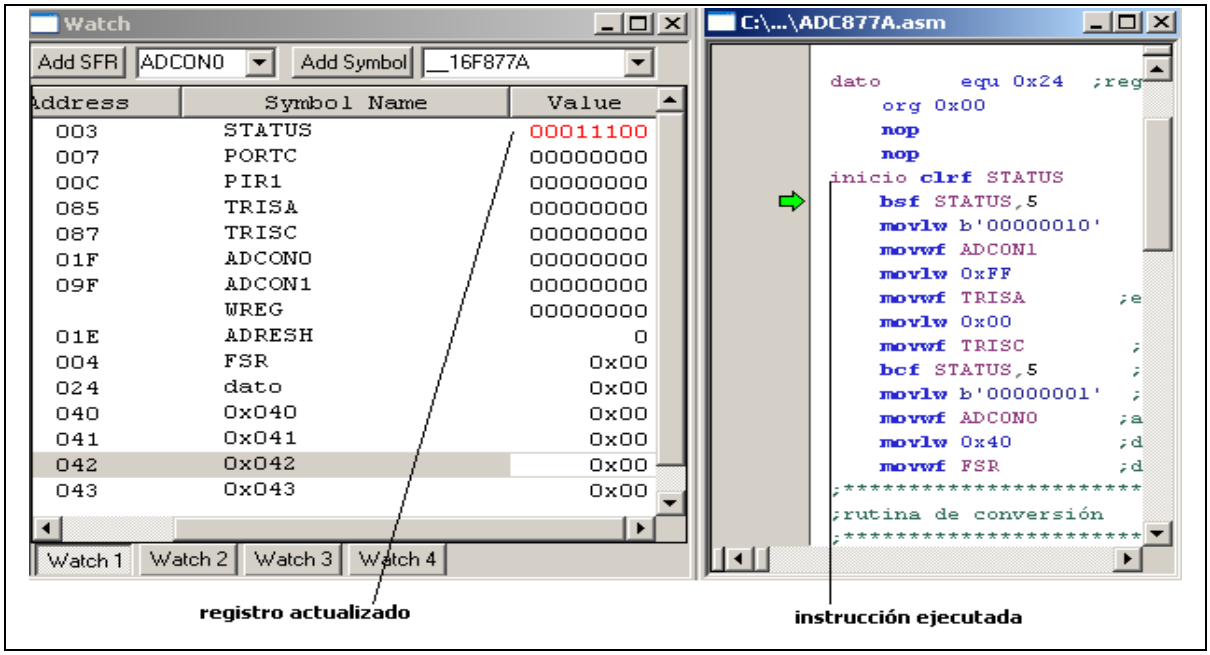


Fig.4.25 *Ejecución de una instrucción especifica.*

CONCLUSIÓN:

En la depuración continua (RUN) se corre el programa hasta finalizar ó hasta encontrar un breakpoint, mientras en la depuración paso a paso se observa que el usuario puede interactuar con el hardware a medida se ejecuta el código y observar los cambios que sufren los registros con cada instrucción ejecutada, siendo la depuración paso a paso más eficiente que la continua para verificar completamente un determinado código y encontrar errores.

4.2.6 REALIZAR DEPURACION UTILIZANDO EJEMPLO 2

Programa que contiene un error el cual se tiene que descubrir y corregir utilizando el sistema creado.

Código de programa **Error**, programa que muestra por el puerto C (leds) el código 0x0F si el dato leído es cero ó el valor 0xF0 si el dato leído es diferente de cero.
Se pretende encontrar el error lógico en el programa utilizando la herramienta de depuración. En caso de fallas refiérase a la sección 4.11 del manual de usuario.

- Realizar el circuito de conexión de la figura 4.39 y figura 4.40.
- Abrir el proyecto *error* con *Project* > *open*, el proyecto está situado en la carpeta programas_finales.
- Establecer todos los parámetros necesarios para realizar depuración.
- Programar código en el MCU del módulo principal.
- Realizar reset para iniciar depuración con el botón  .
- Colocar el dato cero en el módulo de entradas digitales (minidips).
- Correr el programa de modo continuo y verificar si el dato mostrado es 0x0F.
- con el programa ejecutándose colocar diferentes datos en los minidips y verificar si el dato mostrado es 0xF0 ó 0x0F.
- Detener el programa con el botón *Halt*  .

Como se observa, el programa solo muestra el código 0x0F, todos los datos ingresados los reconoce como ceros. Por medio de la depuración paso a paso se debe encontrar el error presente en el código.

- Realizar reset con el botón  y limpiar los registros de propósito general (*debugger* > *Clear Memory* > *GPRs*).
- Ingresar el valor de cero en los minidips.

- Colocar un breakpoint (Fig.4.26), correr paso a paso y verificar cada registro pero específicamente el bit 2 del registro STATUS (Z), el resultado se muestra en la Fig.4.26.

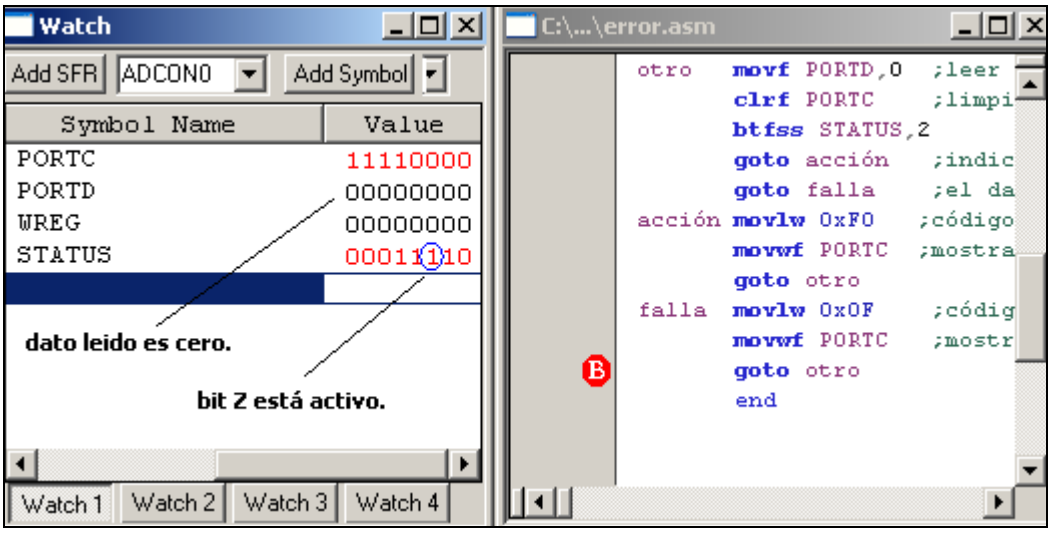


Fig.4.26 *registros después de la ejecución paso a paso.*

Puede observarse que el bit Z del registro STATUS se activa por que el dato es cero, entonces el programa muestra el código 0x0F que indica falla.

- Realizar reset con el botón e ingresar el dato 01010101 con los minidips.
- Correr paso a paso y verificar los cambios en cada registro pero específicamente el bit 2(Z) del registro STATUS, como se muestra en la figura.4.27.

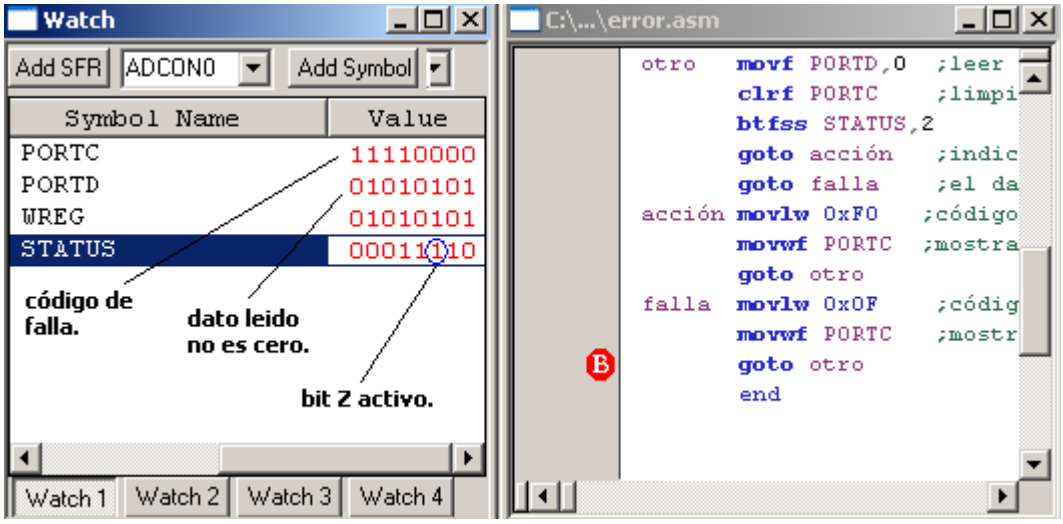


Fig.4.27 *Registros después de la ejecución paso a paso.*

Con la ejecución del código puede observarse que aunque el dato leído es diferente de cero el bit Z permanece activo, el error está descubierto pero no se conoce que lo provoca. Se recomienda verificar que instrucciones afectan el bit Z para establecer soluciones que hagan que el bit Z solo se active cuando el dato leído sea específicamente cero y permanezca inactivo con otros datos diferentes a cero.

Solución:

El bit Z se activa cada vez que se ejecutan instrucciones como CLRF, CLRFW, ADDWF, entre otras. Por lo que una instrucción de estas mantiene siempre activo el bit Z Aunque el dato no sea cero. Con la depuración paso a paso se observa que la instrucción CLRF PORTC siempre activa el bit Z.

- Borrar la instrucción CLRF PORTC del código.
- Compilar de nuevo el programa con el botón Build All .
- Grabar el programa en el MCU del módulo principal.
- Realizar reset con el botón.
- Ingresar el dato 01010101 con los minidips.

- Colocar un breakpoint (Fig.4.28), correr paso a paso, verificar que el bit Z es cero y que el dato en el puerto C es 0xF0, el resultado se muestra en la Fig.4.28.

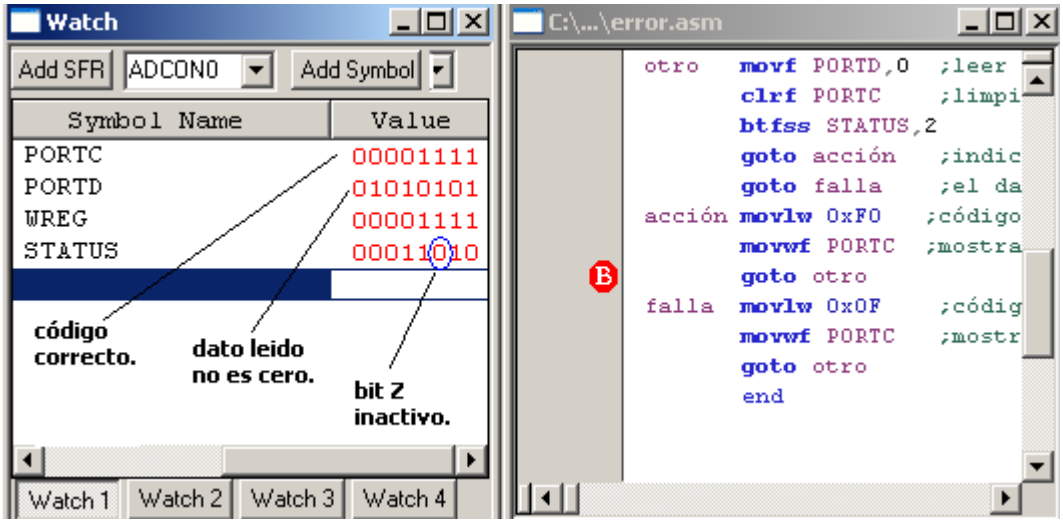


Fig.4.28 *Registros después de la depuración paso a paso del código corregido.*

- Quitar breakpoint con doble click izquierdo en el lugar donde se encuentra, realizar reset, correr el programa de modo continuo (RUN) e ingresar datos que sean cero o diferentes de cero para verificar todo el programa.

CONCLUSIÓN:

Con la herramienta de depuración no simplemente se corren programas verificando los cambios en cada registro, sino que a partir de estos cambios se determina el funcionamiento del programa que puede ser el deseado o no, ya sea por que el dato esperado no es el correcto, el bit cambia debido a otras instrucciones ó por que el registro se ve afectado por situaciones que no se han tomado en cuenta a la hora de programar , Con el sistema debug pueden determinarse errores lógicos como el mostrado en el ejemplo 2, corregirlos y verificar que el funcionamiento sea el correcto.

4.2.7 REALIZAR DEPURACION UTILIZANDO EJEMPLO 3

utilización del sistema de comunicación SPI en modo maestro

Ejemplo de depuración del programa en modo maestro **SPImaster877A** (Sección 4.9.3) para comunicación SPI, El maestro captura datos por el puerto D y los envía al PIC esclavo situado en el módulo I2C-SPI,el Esclavo los muestra por el puerto B (leds). En caso de fallas refiérase a la sección 4.11 del manual de usuario.

- Grabar en el MCU del módulo I2C-SPI el código SPIslave.
- Construir circuito de conexión de la figura 4.55.
- Cargar en MPLAB el proyecto SPImaster877A.
- Establecer los parámetros necesarios para depuración.
- establecer los registros GPR a visualizar (Fig.4.30).

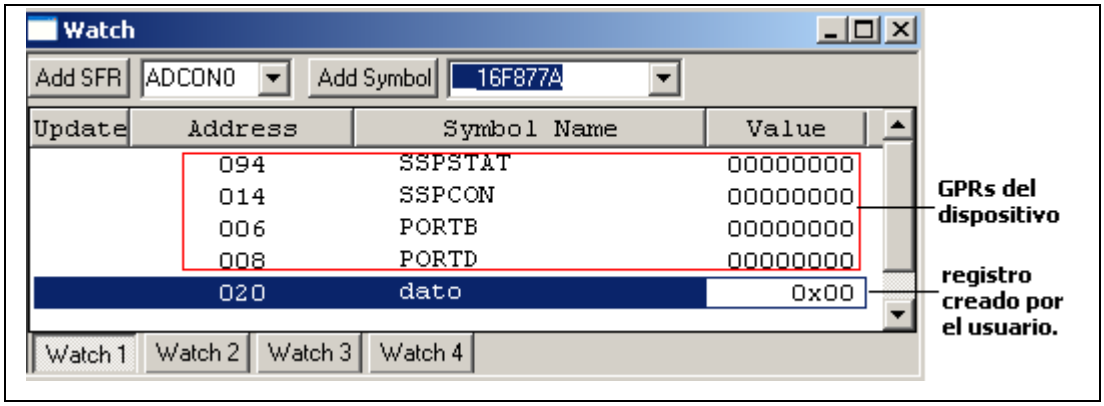


Fig.4.30 *registros utilizados en la depuración.*

- Realizar reset con el botón y reset en el módulo I2C-SPI.
- Correr de modo continuo RUN, colocar diferentes datos en los minidips y verificar los datos visualizados en el módulo de leds (puerto B de la tarjeta I2C-SPI).
- Detener el programa con el botón Halt , los registros quedan con el valor del último dato colocado en los minidips (depende del ultimo valor colocado en los minidips), el cursor queda en la ultima instrucción realizada, (puede ser cualquiera) como se muestra en la (Fig.4.31).

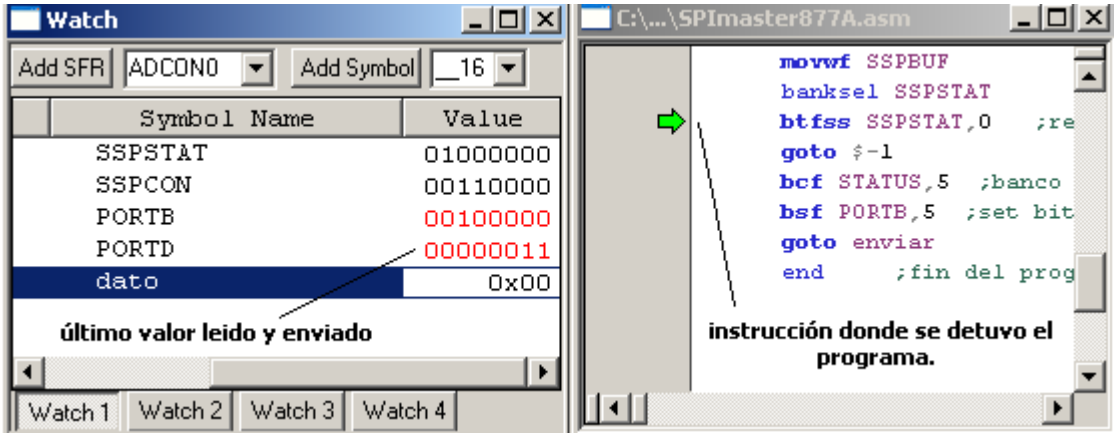


Fig.4.31 *Resultado de la depuración en modo continuo.*

- Realizar reset con el botón  y reset en la tarjeta I2C-SPI.
- Colocar un breakpoint (Fig.4.32), colocar el dato 0xFF en los minidips, correr paso a paso y verificar como se van actualizando los registros, se verifica que el dato es enviado cuando se carga el registro SSPBUF con el dato, el programa se detiene al llegar al breakpoint, el resultado se muestra en la figura 4.32.

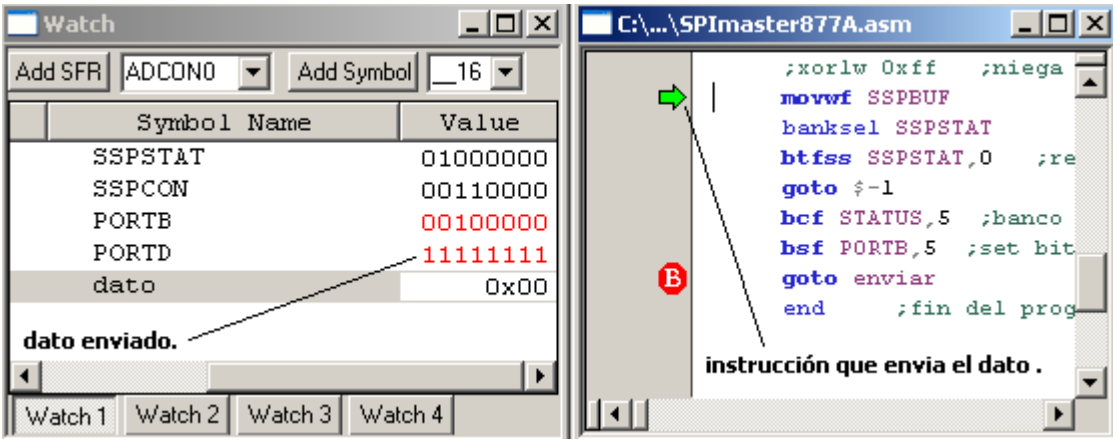


Fig.4.32 *Resultado después de la depuración paso a paso*.

- Realizar reset tanto en MPLAB-IDE como en el módulo I2C-SPI, colocar otro valor, correr paso a paso y verificar el resultado nuevamente si se desea.

CONCLUSIÓN

El ejemplo 3 muestra como la depuración en modo continuo y en modo paso a paso se puede realizar en una comunicación SPI, en este tipo de comunicación intervienen dos dispositivos uno el maestro y otro el esclavo, se ha depurado solo el código del maestro ya que este es el que decide cuando enviar los datos hacia el esclavo, pero puede observarse que existen instrucciones que dependen del resultado que envía el esclavo hacia el maestro , por tal razón pueden existir distintos tipos de errores ya sea por parte del maestro como del esclavo y ser detectados por la depuración, siendo esta herramienta muy útil para desarrollar aplicaciones no solo con un MCU sino usando 2 MCUs.

4.3 COMO PROGRAMAR UN CÓDIGO DETERMINADO.

Los pasos a seguir para realizar la programación son:

- Conectar la tarjeta debug a la PC utilizando el puerto USB.
- Conectar tarjeta debug con módulo principal de aplicaciones (Fig.4.39).
- Alimentar la tarjeta debug con la fuente externa (15Vdc).
- Abrir software MPLAB versión 8.02.
- Abrir el proyecto que contiene el código a programar (*Project – open*), Los programas se encuentran en la carpeta programas_finales.
- Especificar tarjeta en modo programador (sección 4.3.1).
- Especificar MCU a utilizar y bits de configuración (sección 4.3.2).
- Establecer parámetros de comunicación (sección 4.3.3).
- Conectar tarjeta y programar dispositivo (sección 4.3.4).
- Correr el programa desde MPLAB (sección 4.3.5).

4.3.1 ESTABLECER TARJETA EN MODO PROGRAMADOR.

Establecer la tarjeta para que trabaje en modo programador así:
Programmer > *Select programmer* > *MPLAB ICD2.*como se muestra en la Fig.4.33.

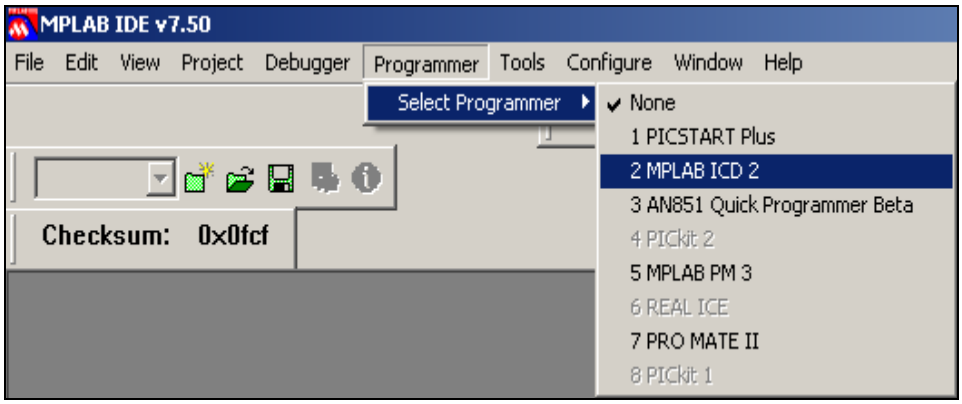


Fig.4.33 *establecer tarjeta en modo programador.*

4.3.2 ESPECIFICACIÓN DE DISPOSITIVO A UTILIZAR Y DE BITS DE CONFIGURACIÓN.

La secuencia es idéntica a la explicada en la sección 4.2.2 del manual de usuario, por lo que deben de realizarse los mismos pasos estableciendo los mismos bits de configuración.

4.3.3 ESPECIFICACIÓN DE PARÁMETROS DE COMUNICACIÓN.

Los pasos son idénticos a los explicados en la sección 4.2.3 del manual de usuario, refiérase a dicha sección y siga los pasos establecidos.

4.3.4 CONECTAR TARJETA Y PROGRAMAR DISPOSITIVO.

Establecer conexión entre la tarjeta y el software utilizando el botón *reset and conect to ICD*  . Cuando la conexión finaliza se activa la barra de herramientas fig.4.34 y el estado de conexión se visualiza en la fig.4.35.

Fig.4.34 *Barra de herramientas modo programador.*

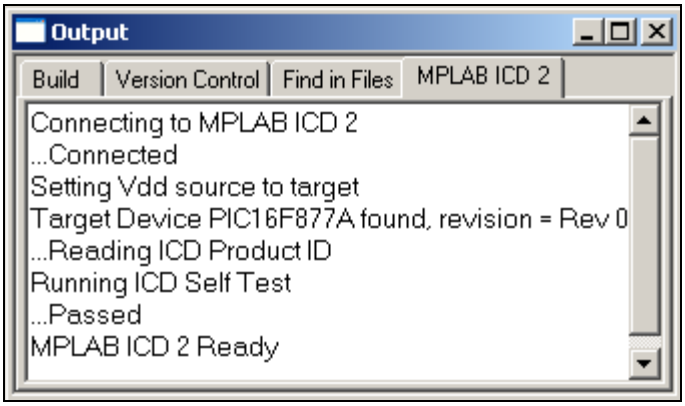


Fig.4.35 *Ventana de conexión de la tarjeta.*

Borrar dispositivo antes de programar con el botón  *Erase target device,* la verificación de dispositivo borrado se realiza con el botón  *verify target device is erased,* en la figura 4.36 se observan los mensajes cuando se realiza un borrado y en la fig.4.37 se observa la verificación.

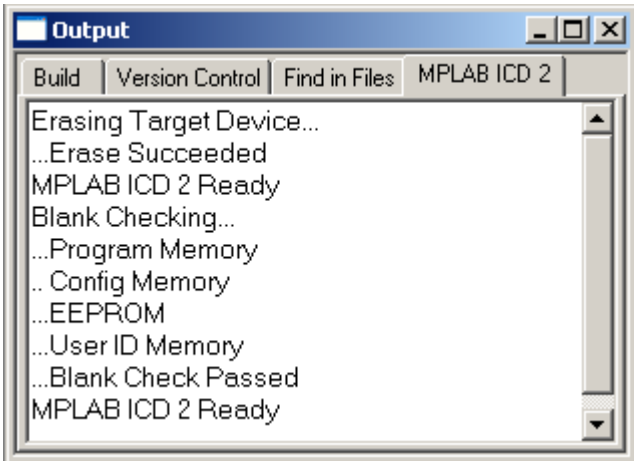


Fig.4.36 *Ventana de borrado del dispositivo.*

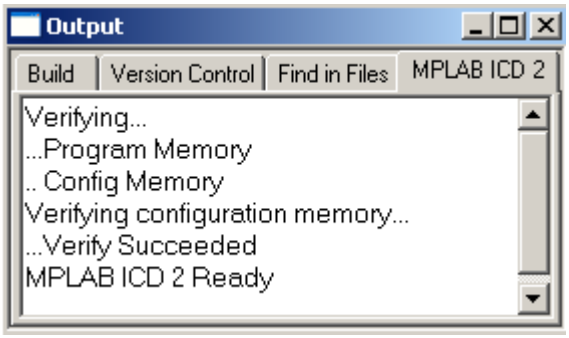


Fig.4.37 *Ventana de verificación de memoria.*

Para descargar el código en el MCU se realiza por medio del botón *program target device*  de la barra de herramientas en modo programador, en la ventana output puede verificarse si se realizó la descarga satisfactoriamente, se muestra en la figura 4.38.

NOTA. En caso de fallas refiérase a la sección 4.11 de este capitulo.

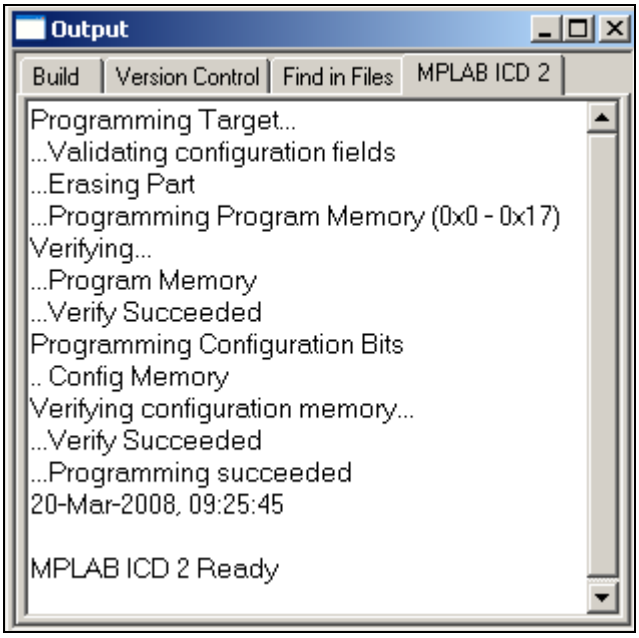


Fig. 4.38 *Ventana de programación del dispositivo*.

4.3.5 CORRER PROGRAMA DESDE MPLAB.

Una vez programado el código en el MCU, puede correrse utilizando el botón  *release from reset,* MPLAB envía un pulso alto para activar el PIC16F877/A de la tarjeta principal, con ésto se logra que el MCU programado ejecute el código en el hardware de aplicaciones.

Para realizar un reset se utiliza el botón *hold in reset*  , con esto se envía un pulso negativo para mantener en reset el PIC 16F877/A de la tarjeta principal.

Para realizar la comprobación del sistema utilizando la tarjeta debug en conjunto con el hardware de aplicaciones se propone la tutoría de ejercicios detallados a continuación, además de los diagramas de flujo de cada código y sus respectivos diagramas de conexión.El diagrama de conexión entre la tarjeta debugger y el módulo principal se detalla en la figura 4.39.

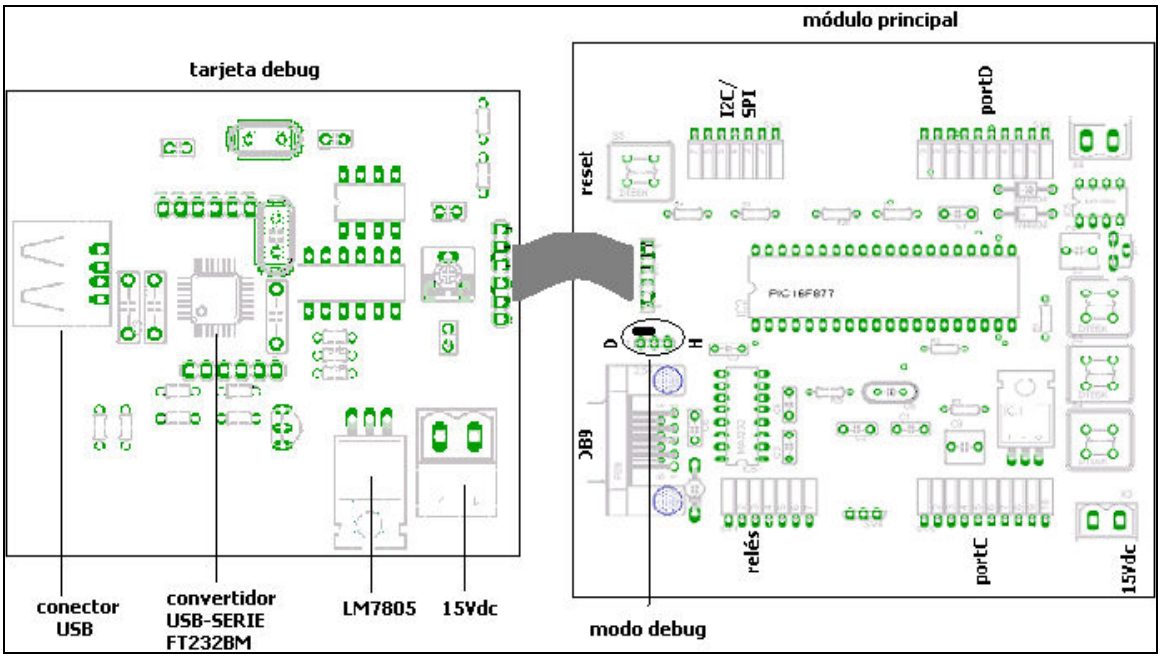


Fig. 4.39 *Diagrama de conexión tarjeta debug y módulo principal.*

TUTORÍA DE EJERCICIOS

4.4 MÓDULO DE ENTRADA Y SALIDA DIGITAL

Los MCUs a utilizar poseen el módulo de entrada /salida digital que consta de 5 puertos: Puerto A, B, C, D y E. Los registros para configurar cada puerto como entrada o como salida son respectivamente: TRISA, TRISB, TRISC, TRISD, TRISE.

Colocando "1" en cada bit de estos registros se activa el puerto correspondiente como entrada, colocando "0" en cada bit se activa el puerto como salida de datos digitales. Para mayores detalles refiérase al capitulo 1 sección 1.4.

El diagrama de conexión para utilizar los módulos de E/S digital se muestra en la figura 4.40.

4.4.1 LECTURA Y ESCRITURA DE DATOS DIGITALES

Objetivo:
Leer y escribir datos digitales usando los puertos del PIC16F8977/A.

Programa para leer datos desde el puertoD para mostrarlos por el puertoC y guardarlos en tres direcciones de memoria RAM utilizando el registro de direccionamiento indirecto FSR.

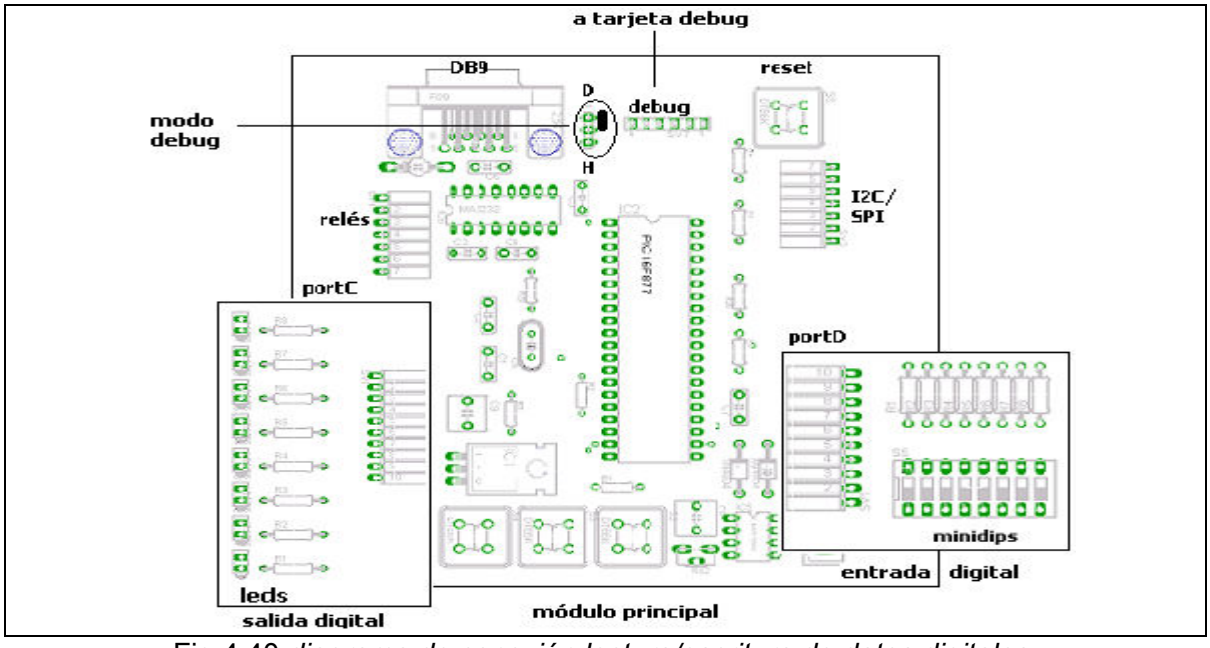
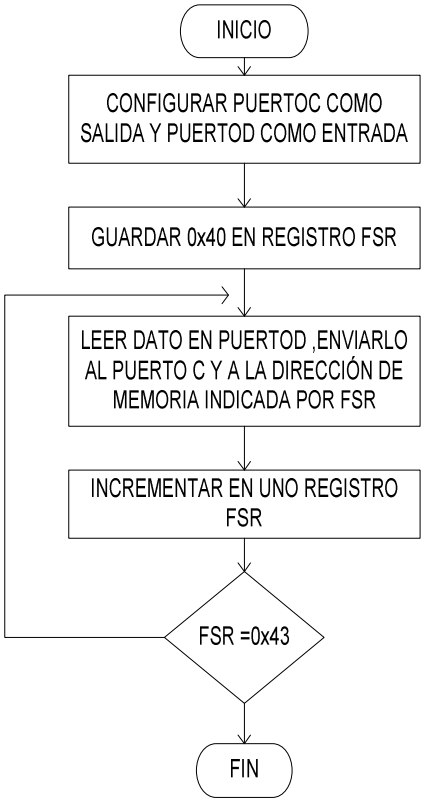


Fig.4.40 *diagrama de conexión lectura/escritura de datos digitales.*




```
        List P=PIC16F877A
        include "P16F877A.INC"

          org       0x00
          nop                 ; retardo para.
          nop                 ; depuración.
Inicio bsf STATUS, 5      ; banco 1.
        movlw 0x00
        Movwf TRISC       ; como salida.
        movlw 0xff
        Movwf TRISD       ; como entrada.
        bcf STATUS,5      ;banco 0
        Movlw 0x40        ; dirección de inicio.
        Movwf FSR         ; para guardar datos.
        clrf PORTD
        clrf PORTC
Leer  movf PORTD, 0   ; leer dato.
        Movwf PORTC       ; mostrar el dato.
        movwf INDF        ; guardar dato.
        Incf FSR, 1       ; incrementar
dirección.
        Movf FSR, 0       ; W=dirección de
FSR.
        Sublw 0x43        ; FSR es igual a
0x43.
        Btfss STATUS, 2; verifica si es igual.
        Goto leer         ; No, leer otro dato.
        Goto $            ; Si, bucle infinito.
        End
```

Fig.4.41 *Diagrama de flujo y código de programa para uso de puertos digital*

4.4.2 MANEJO DE MOTOR PASO.

Objetivo:

Controlar un motor paso utilizando los puertos A y B del PIC 877/A así:

Sentido horario (P1), paro (P2) y sentido anti horario (P3).

El diagrama de conexión para utilizar el módulo de control del motor paso en conjunto con la tarjeta principal se muestra en la figura 4.42,asegurece que la tarjeta relés/motor esté colocada en modo motor.

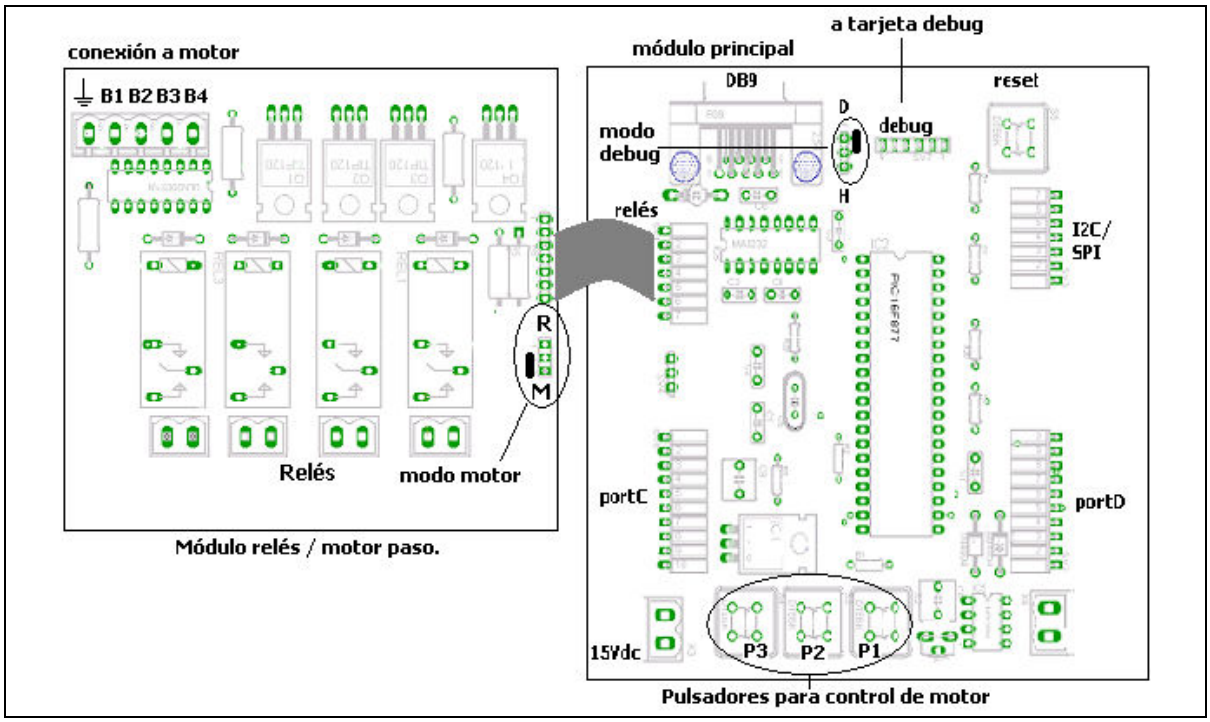


Fig.4.42 *Diagrama de conexión control de motor paso.*

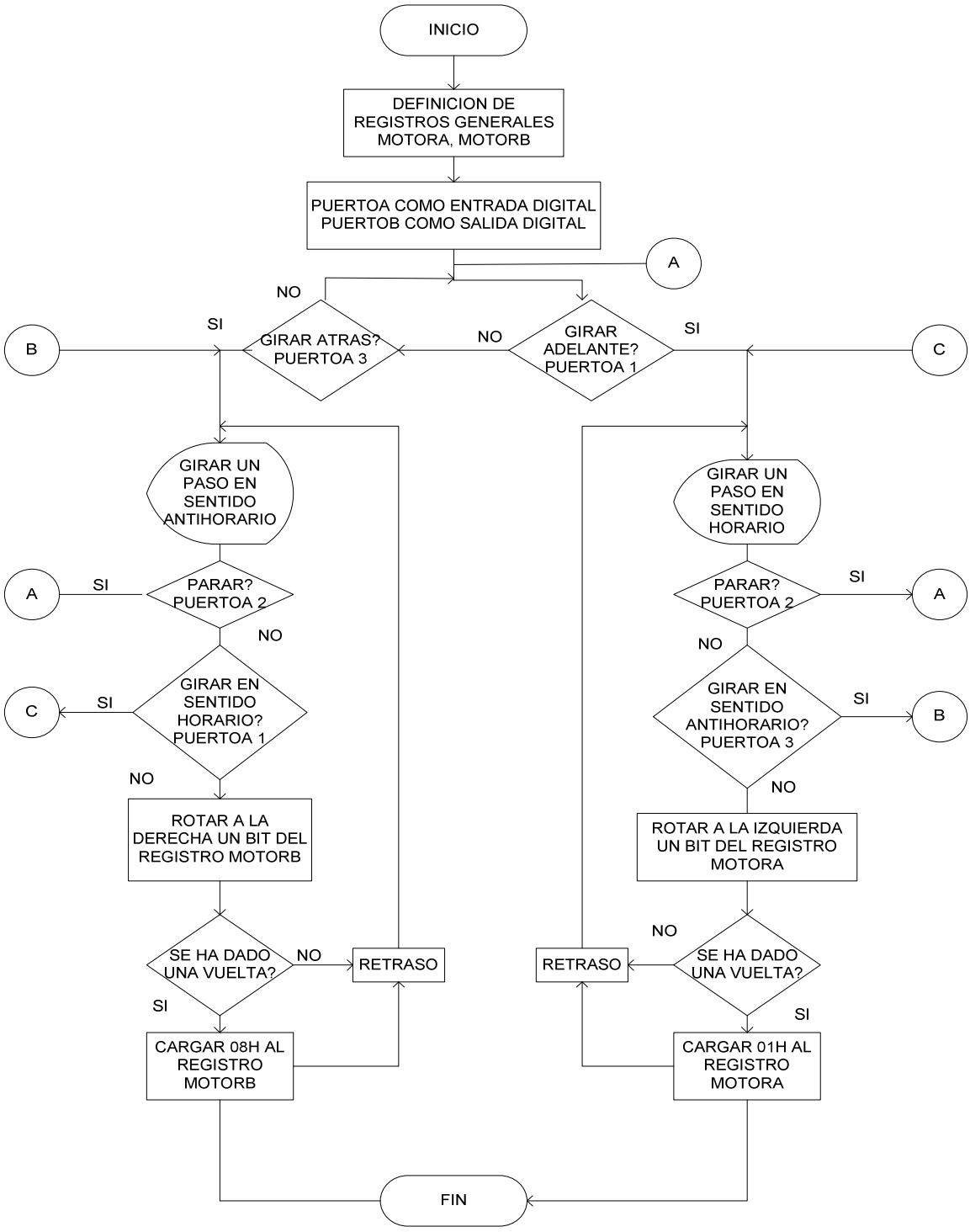


Fig.4.43 *Diagrama de flujo manejo de motor paso.*

```
List P=PIC16F877A             ; cabecera de el PIC a utilizar.
Include "P16F877A.INC"
Dato       equ 0x22          ; definicion de registros de proposito.
cont2      equ 0x41          ; general a utilizar.
Cont3      equ 0x42
Cont4      equ 0x43
MotorA     equ 0x44
MotorB     equ 0x45
    Org 0x00                 ; direccion de inicio del programa.
          nop                ; retraso para realizar debug.
          nop
Inicio     clrf STATUS
          Bsf STATUS, 5      ; banco 1
          Movlw 0x06
          Movwf ADCON1   ; configura portA y portE como puertos digitales.
          Movlw 0xf0
          Movwf TRISA      ; salidas digitales 0-3 para el motor y entrada la 4,5.
          Movlw 0xFF
          Movwf TRISE      ; puertoE como entrada.
          Bcf STATUS, 5      ; Cambio al banco 0 ----------
          Clrf PORTA
          Movlw 0x01
          Movwf motorA     ; registro de giro horario.
          Movlw 0x08
          Movwf motorB     ; registro de giro antihorario.
          Movlw d'10'
          Movwf cont2       ; contadores para realizar el retraso de 1 segundo.
          Movlw d'200'
          movwf cont3
          Movlw d'200'
          Movwf cont4
;*************************************************************
Sentido btfss PORTA, 1     ; verifica si se ha pulsado el boton adelante.
          Goto otro
          Btfsc PORTA, 1
          Goto $-1
          Goto adelante    ; ir a la rutina de giro horario.
Otro       btfss PORTA, 3    ; verifica si se ha pulsado el boton atrás.
          Goto sentido
          Btfsc PORTA, 3
          Goto $-1
          Goto atras          ; ir a rutina de giro antihorario.
;*************************************************************
; Rutinas de giro del motor.
;*************************************************************
Adelante  movf motorA, 0; rutina de  giro horario.
```

```
            Movwf PORTA    ; activar bobinas del motor.
            Call retraso       ; retraso entre pasos.
Stop        btfss PORTA, 2   ; verifica si se ha presionado el boton de paro.
            Goto  otro1
            Btfsc PORTA, 2
            Goto $-1
            Goto sentido
otro1       btfss PORTA, 3; verifica si se desea girar al sentido antihorario.
            Goto  seguir
            Btfsc PORTA, 3
            Goto $-1
            Goto atras       ; rutina de giro antihorario.
Seguir              bcf STATUS, C
            Rlf motorA, 1      ; rotar bit hacia la izquierda para dar otro paso.
            Btfss motorA, 4; verificar si se han dado cuatro pasos,una vuelta.
            Goto adelante
            Movlw 0x01      ; se vuelve a iniciar para realizar otra vuelta.
            Movwf motorA
            Goto adelante    ; realiza otra vuelta.
.************************************************************
;
Atras               movf motorB, 0; rutina de giro antihorario.
            Movwf PORTA   ; activar bobinas del motor.
            Call retraso
stop1       btfss PORTA, 2; verificar si se desea parar.
            Goto  otro2
            Btfsc PORTA, 2
            Goto $-1
            Goto sentido
otro2       btfss PORTA, 1; verificar si se desea girar en forma horaria.
            Goto seguir1
            Btfsc PORTA, 1
            Goto $-1
            Goto adelante   ; rutina de giro horario.
seguir1     bcf STATUS, C
            Rrf motorB, 1     ; rotar bit a la derecha para realizar otro paso.
            Btfss STATUS, C; verificar si se ha dado una vuelta completa.
            Goto atras
            Movlw 0x08      ; iniciar para realizar otra vuelta.
            Movwf motorB
            Goto atras       ; realizar otra vuelta.
Retraso     movlw d'5'       ; rutina de retraso de velocidad del motor.
            Movwf cont2
            Movlw d'100'
            Movwf cont3
            Movlw d'100'
            Movwf cont4
```

```
Delay     decfsz cont2, 1   ; decrementa contador 2.
          Goto delay1
          Return
delay1    decfsz cont3, 1    ; decrementa contador 3.
          Goto delay2
          Goto delay
delay2    decfsz cont4, 1     ; decrementa contador 4.
          Goto delay2
          Goto delay1
          End
```

4.4.3 MANEJO DE RELÉS.

Objetivo:

Controlar 4 relés por medio del puerto D y puerto B.

RD0 controla relé 1.
RD1 controla relé 2.
RD2 controla relé 3.
RD3 controla relé 4.

Para desarrollar este ejercicio debe de utilizarse el diagrama de conexión de la tarjeta de control de relés y el módulo principal como se muestra en la figura 4.44,asegurese de colocar la tarjeta relés/motor en modo relés.

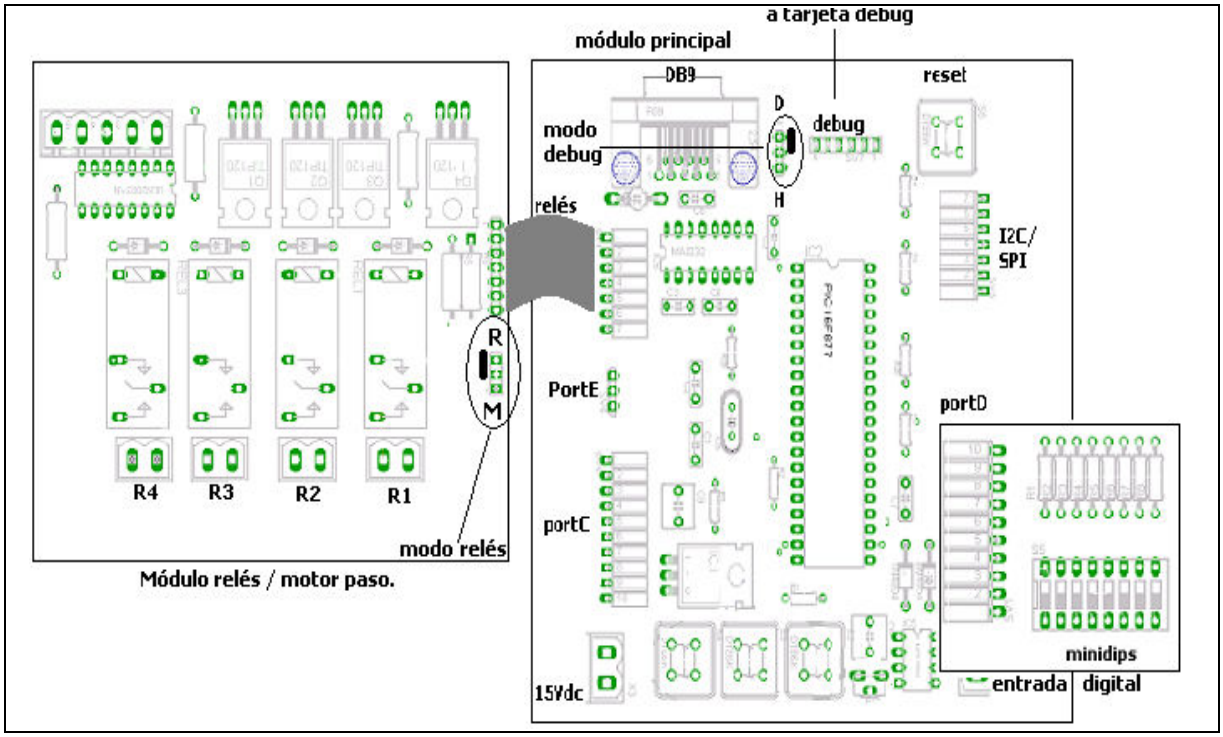


Fig.4.44 *Diagrama de conexión manejo de relés.*

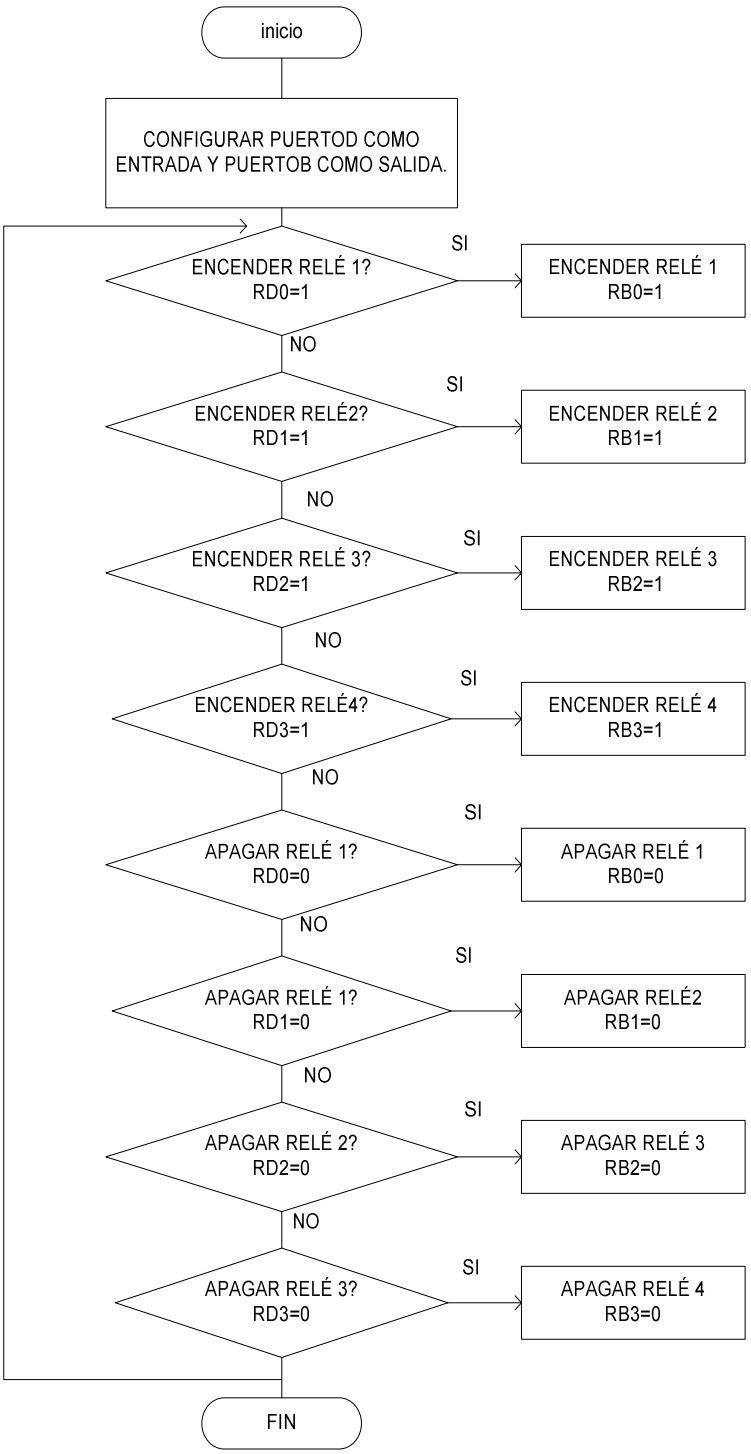


Fig.4.45 *Diagrama de flujo manejo de relés de 12Vdc.*

```
    List P=PIC16F877A
    Include "P16F877A.INC"

   Org 0x00
            Nop
            Nop
Inicio      clrf STATUS
            Bsf STATUS, 5           ; banco 1
            Movlw 0x00              ; salidas digitales.
            Movwf TRISB         ; para manejo de relés.
            Movlw 0xff
            Movwf TRISD         ; puerto D como entradas digitales.
            Bcf STATUS, 5       ; banco 0
            Clrf PORTD              ; limpia puerto D y B
            Clrf PORTB
encender1   btfss PORTD, 0      ; verifica si se desea activar el relé1.
            Goto encender2          ; sino verifica la próxima entrada.
            Bsf PORTB, 0        ; activa relé 1.
encender2   btfss PORTD, 1      ; verifica si se desea activar el relé2.
            Goto encender3      ; sino verifica la próxima entrada.
            Bsf PORTB, 1        ; activa relé 2.
encender3   btfss PORTD, 2          ; verifica si se desea activar el relé3.
            Goto encender4      ; sino verifica la próxima entrada.
            Bsf PORTB, 2        ; activa relé 3.
encender4   btfss PORTD, 3      ; verifica si se desea activar el relé4.
            Goto apagar1        ; sino verifica la próxima entrada.
            Bsf PORTB, 3        ; activa relé 4.
apagar1     bits PORTD, 0       ; se desea apagar relé 4.
            Goto apagar2
            Bcf PORTB, 0        ; apaga relé 1.
apagar2     bits PORTD, 1       ; se desea apagar relé 4.
            Goto apagar3
            Bcf PORTB, 1        ; apaga relé 2.
apagar3     bits PORTD, 2       ; se desea apagar relé 4.
            Goto apagar4
            Bcf PORTB, 2        ; apaga relé 3.
apagar4     bits PORTD, 3       ; se desea apagar relé 4.
            Goto encender1
            Bcf PORTB, 3        ; apaga relé 4.
            Goto encender1
            End
```

4.5 CONVERTIDOR ANÁLOGO-DIGITAL.

http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

Los registros asociados a este subsistema son:

Registros de control ADCON1 - ADCON0 –ADRESH – ADRESL.
Registros de definición de pines de entrada TRISA – PORTA – TRISE -PORTE.
Registros de interrupciones INTCON – PIR1 – PIE1.

Registro ADCON1

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

ADFM selección de formato de resultado.
"1" ajuste a la derecha. "0" ajuste a la izquierda.

PCFG3:PCFG0 configuración de las entradas ya sea digitales o análogas.

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C/R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2/1
011x	D	D	D	D	D	D	D	D	—	—	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1/2

Registro ADCON0

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

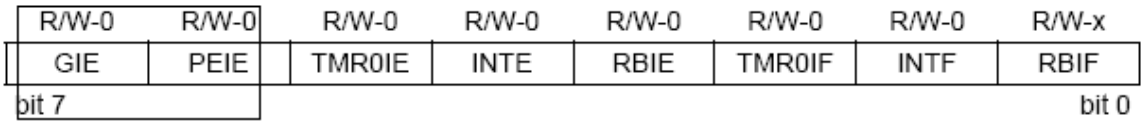
ADCS1:ADCS0 selección de reloj para la conversión A/D.
"00"=Fosc/2 "01"=Fosc/8 "10"=Fosc/32 "11"=Frc.

CHS0:CHS2 selección del canal de conversión.
000= canal 0 001= canal 1 010= canal 2 011= canal 3
100= canal 4 101= canal 5 110= canal 6 111= canal 7

GO/DONE estado de la conversión.
Si ADON = 1 "1" conversión en progreso - "0" conversión finalizada.

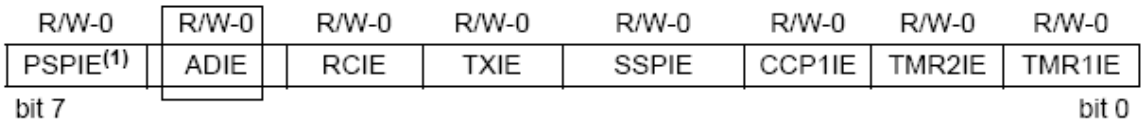
ADON bit de encendido del conversor A/D.
"1" modulo encendido "0" modulo apagado.

Registro INTCON



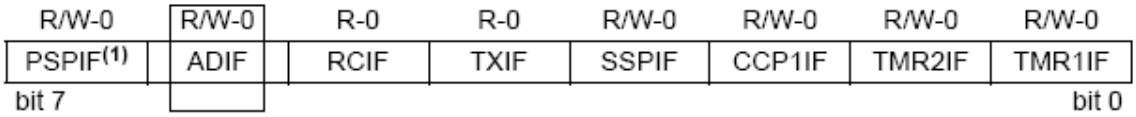
GIE habilitación global de interrupciones.
PEIE habilitación de interrupciones de periféricos.

Registro PIE1



ADIE habilitación de la interrupción del conversor A/D.

Registro PIR1



ADIF bandera de interrupción del convertidor A/D.
"1" conversión completada "0" conversión no completada.

4.5.1 PASOS PARA REALIZAR UNA CONVERSION EN EL MÓDULO A/D.

Configuración del modulo A/D.

- Definir entradas análogas y tensión de referencia utilizando ADCON1.
- Definir canal (es) de la conversión utilizando ADCON0.
- Seleccionar reloj de conversión utilizando ADCON0.
- Encender el modulo A/D utilizando bit ADON de ADCON0.

Configuración de interrupciones (opcional).

- Colocar "0" en el bit ADIF del registro PIR1.
- Habilitar la interrupción del conversor A/D en el registro PIE1.
- Habilitar las interrupciones de los periféricos en INTCON.
- Habilitar la máscara global de interrupciones en INTCON.

Tiempo de espera de conversión.

- Es el tiempo que se tarda en capturar el valor análogo a convertir en el canal establecido, típicamente es de 20useg.

Iniciar la conversión.

- Colocar a “1” el bit GO/DONE del registro ADCON0.

Esperar a que se complete la conversión.

- Verificar cuando el bit GO/DONE se pone a “0”.
- Esperando a que se de la interrupción por conversión.

Resultado de la conversión.

- Leer el dato obtenido en los registros ADRESH: ADRESL.
- Poner a “0” el bit ADIF si se utilizó interrupción.

4.5.2 PROGRAMA PARA USO DEL CONVERTIDOR ANÁLOGO/DIGITAL.

Conectar el módulo de salida digital al módulo principal como se muestra en la figura 4.46, visualizar la entrada interna y la entrada externa de voltaje en el módulo principal.

Objetivo:

Leer datos usando el convertidor análogo / digital (código A).
Programa para leer datos análogos desde la entrada interna (potenciómetro en canal 0) ó desde la entrada externa (canal 4),mostrar los datos por el módulo de salida digital conectado al puertoC y guardarlos en tres localidades de memoria RAM a partir de la dirección 0x40 , luego grabar el valor 0xff en el registro dato.
El convertidor está configurado así: Vref +5V interno, Oscilador XT, Canal 0 ó Canal4.

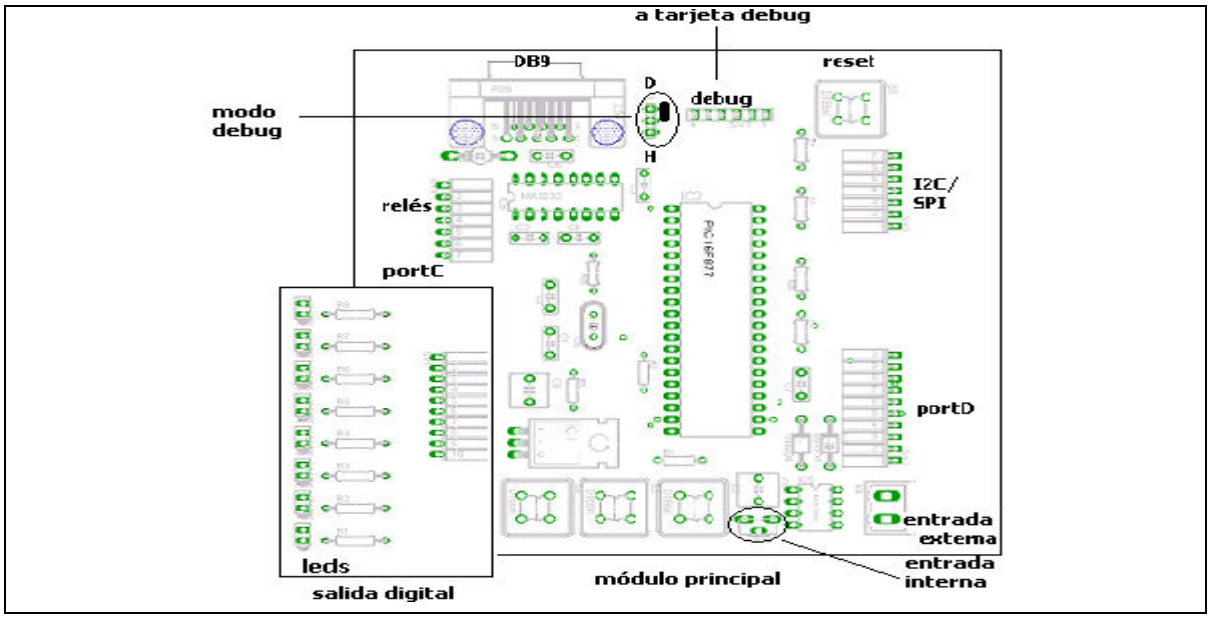


Fig.4.46 *Diagrama de conexión uso de módulo análogo-digital.*

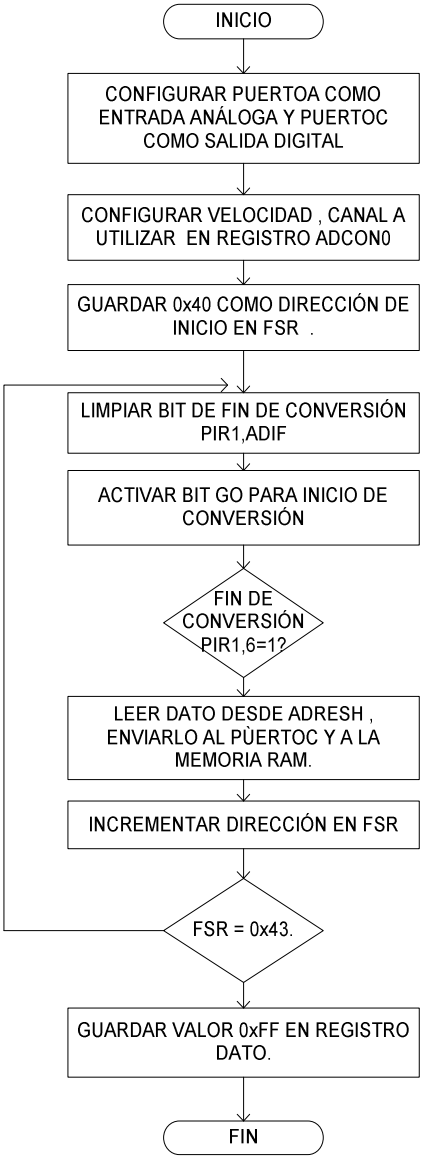


Fig.4.47 *Diagrama de flujo conversion análoga/digital.*

```
        List P=PIC16F877A
        Include "P16F877A.INC"

                Dato    equ 0x24        ; registro creado por el usuario.
                Org     0x00
                Nop
                Nop
Inicio          clrf STATUS
                Bsf STATUS, 5           ; banco 1
```

```
                Movlw b'00000010'   ; portA<0:5> como entradas análogas.
                Movwf ADCON1
                Movlw 0xFF
                Movwf TRISA         ; entradas análogas.
                Movlw 0x00
                Movwf TRISC         ; salidas  digitales.
                Bcf STATUS, 5       ; banco 0
                Movlw b'00000001'   ; canal 0 AN0
                Movwf ADCON0        ; activamos registro para convertidor A/D .Fosc/2
                Movlw 0x40          ; dirección de inicio de
                Movwf FSR           ; datos guardados.
;*********************************************************************
; Rutina de conversión
;*********************************************************************
Conversión      bcf PIR1, 6         ; limpio bandera ADIF de dato convertido.
Otro            bsf ADCON0, 2       ; activamos bit GO de inicio de conversión.
                Btfss PIR1, 6               ; conversión finalizada?
                Goto $-1            ; no, espera entonces.
                Movf ADRESH, 0      ; leo en W el resultado.
                Movwf PORTC         ; envío al puertoC para visualizarlo.
                Movwf INDF          ; guardar dato en dirección de memoria.
                Bcf PIR1, 6         ; limpio bandera de fin de conversión.
                Incf FSR, 1         ; incrementa dirección.
                Movf FSR, 0         ; guardar en W dirección en FSR.
                Sublw 0x43          ; compara con valor 0x43.
                Btfss STATUS, 2     ; se llego a la dirección 0x43?
                Goto otro           ; No, vuelve a leer otro dato.
                Movlw 0xFF          ; Si, se guarda el valor 0xFF.
                Movwf dato          ; en el registro dato.
                Goto $              ; se queda en un bucle infinito.
                End                 ; fin del programa.
```

4.6 MÓDULO DE MEMORIA EEPROM Y FLASH

http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

Registros Asociados a este subsistema son:
EECON1 – EECON2 – EEDATA – EEDATH - EEADR –EEDRH.

REGISTRO EECON1

R/W-x	U-0	U-0	U-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	—	—	—	WRERR	WREN	WR	RD
bit 7							bit 0

EEPGD Seleccionar memoria de programa “1” o memoria de datos “0”**.**
WRERR Bit indicador de escritura: “1” error - “0” escritura finalizada normalmente.

WREN Bit de activación de escritura: “1” permite escribir - “0” no permite escritura.
WR Inicia un ciclo de escritura al activarlo con “1”.
RD Inicia un ciclo de lectura al activarlo con “1”.

REGISTRO EECON2 El registro EECON2 no está físicamente implementado, éste registro sólo se utiliza en la secuencia de escritura, por lo que al realizar una lectura solo se leen ceros.

REGISTRO EEDATA y EEDATH registros utilizados para guardar los datos a escribir hacia la memoria o los datos leídos desde la memoria, siendo EEDATA la parte baja y EEDATH la parte alta para operaciones en memoria flash y EEDATA para operaciones en memoria EEPROM.

REGISTROS EEDR y EEDRH Registros usados para guardar la dirección a utilizar en una escritura o lectura, siendo EEADR la parte baja y EEADRH la parte alta para operaciones en memoria flash y EEADR para operaciones en memoria EEPROM.

4.6.1 PASOS PARA REALIZAR UNA ESCRITURA EN LA MEMORIA EEPROM

1 Guardar el valor de la dirección en el registro EEADR.
2 Guardar el valor del dato en el registro EEDATA.
3 Seleccionar tipo de memoria EEPROM, bit EEPGD en EECON1.
4 Activar bit de permiso de escritura, bit WREN en EECON1.
5 Desactivar las todas interrupciones para evitar errores de escritura.
6 guardar en registro W el valor de 55h, luego transmitirlo a EECON2.
7 guardar en registro W el valor de AAh, luego transmitirlo a EECON2.
8 Iniciar ciclo de escritura activando bit WR en EECON1.
9 Desactivar permiso de escritura en WREN para evitar escrituras parásitas.
10 si se desea volver al paso 1.

4.6.2 PASOS PARA REALIZAR UNA LECTURA A MEMORIA EEPROM.

1 Guardar el valor de la dirección a leer en el registro EEADR.
2 Seleccionar tipo de memoria EEPROM, bit EEPGD en EECON1.
3 Activar bit de lectura RD en EECON1.
4 Leer dato que queda guardado en registro EEDATA.
5 Volver al paso 1 si se desea.

4.6.3 PROGRAMA DE ESCRITURA A MEMORIA EEPROM.

Para este ejercicio solo se utiliza el módulo principal como se muestra en la figura 4.48.

Objetivo:
Escribir una cadena de datos en la memoria EEPROM del PIC 16F877/A.

*Programa que escribe la cadena de datos "*UNIVERSIDAD DON BOSCO 2010*" en la memoria EEPROM iniciando en la dirección 0x00.*

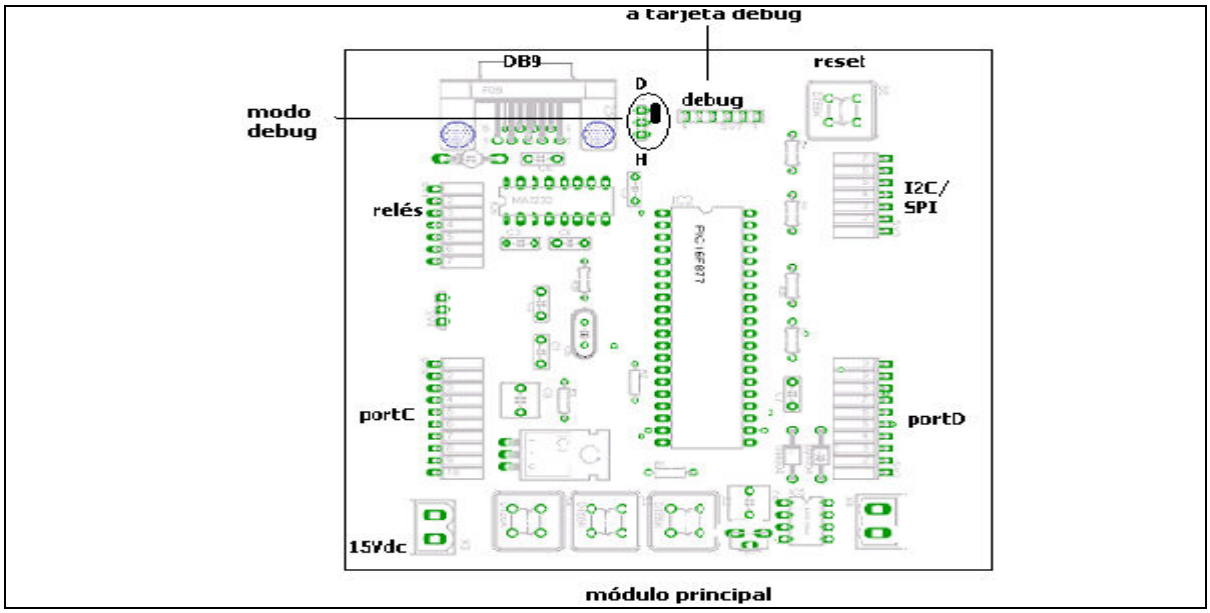


Fig.4.48 *Diagrama de conexión escritura a memoria EEPROM.*

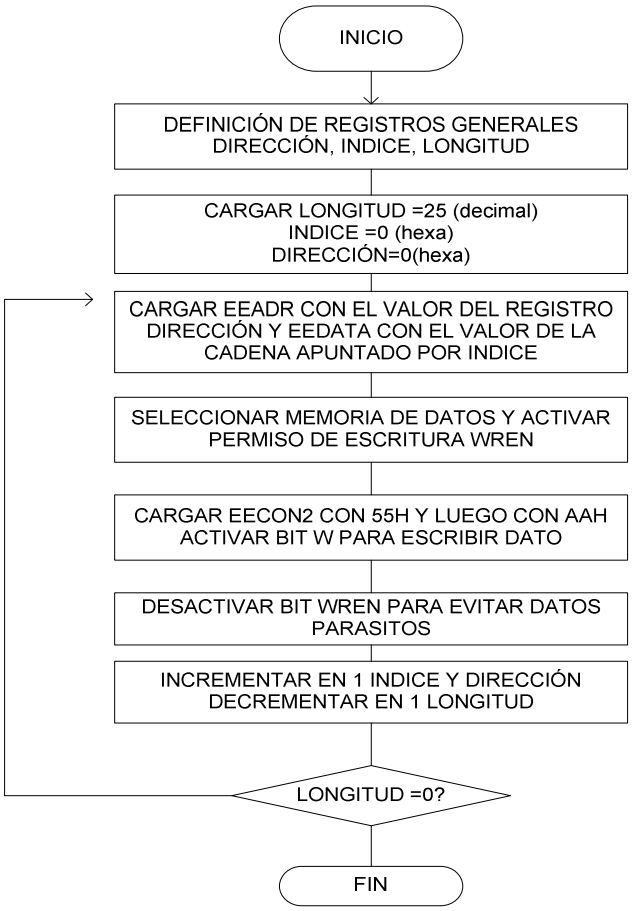


Fig.4.49 *Diagrama de flujo escritura de datos en EEPROM.*

```
        List P=PIC16F877
        Include "P16F877.INC"

Índice                  equ 0x40
Addresslow      equ 0x42                ; contador de direcciones.
Longitud        equ 0x44                ; longitud de la cadena.
Addressalta     equ 0x46

    Org  0x00
         Nop
         Nop
Inicio   clrf STATUS
         Movlw d'25'
         Movwf longitud         ; longitud de la cadena a guardar.
         Movlw 0x00             ; dirección  baja inicial EEPROM
         Movwf addresslow       ; registro  que contiene las direcciones a utilizar.
         Movlw 0x00
```

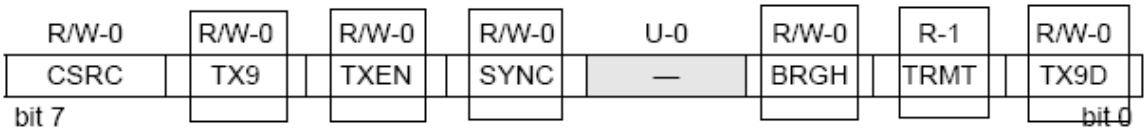
```
        Movwf índice                ; índice de cada dato a grabar.
;*****************************************
Otro    movf addresslow, 0  ; coloca dirección a guardar en W.
        Bsf STATUS, 6               ; banco 2
        Movwf EEADR                 ; se guarda la dirección a escribir en EEADR.
        Bcf STATUS, 6               ; banco 0
        Movf indice, 0              ; coloca en W el índice para capturar.
        Call cadena         ; el primer valor de la cadena.
        Bsf STATUS, 6               ; banco 2
        Movwf EEDATA        ; captura datos a guardar en la eeprom.
        Bsf STATUS, 5               ; banco 3 rutinas de escritura de datos a la eeprom.
WRITE  bcf EECON1, 7        ; selecciona con EEPGD la memoria de datos  EEPROM.
        Bsf EECON1,2        ;WREN activa permiso de escritura,.
        Movlw 0x55                  ; rutina para realizar una escritura a memoria.
        Movwf EECON2
        Movlw 0xAA                  ; está dada por el fabricante.
        Movwf EECON2
        Bsf EECON1, 1       ; activa WR para escritura de datos.
        Nop
        Nop
        Bcf EECON1, 2       ; desactiva WREN para evitar escrituras parásitas.
        Bcf STATUS, 5               ; banco 0
        Bcf STATUS, 6
        Incf addresslow, 1  ; incrementa dirección.
        Incf indice, 1      ; incrementa   índice de la cadena.
        Decfsz longitud, 1  ; decrementa longitud para conocer cuando
        Goto otro           ; se termino de escribir la cadena.
        Goto $
Cadena addwf        PCL, 1          ; cadena a guardar en la eeprom.
      DT "UNIVERSIDAD DON BOSCO 2010"
        End
```

4.7 MÓDULO SERIAL SCI.

http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

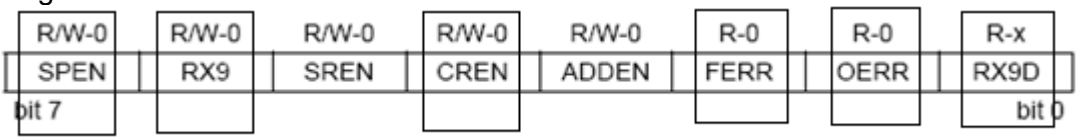
Los registros asociados con este registro son:
Registros de control y estado TXSTA – RCSTA.
Registro de relación de baudios SPBRG.
Registro de datos TXREG (transmisión) – RCREG (recepción).
Registros de interrupciones PIR1 - PIE1.

Registro TXSTA.



SYNC "1" modo síncrono "0" modo asíncrono.
BRGH "1" velocidad alta "0" velocidad baja
TXEN "1" activar transmisión "0" transmisión inactiva
TRMT "1" registro TSR vacio "0" registro TSR lleno
TX9 "1" transmitir 9 bits "0" transmitir 8 bits
TX9D se coloca el noveno bit a transmitir.

Registro RCSTA



SPEN "1" activar puerto serial "0" puerto serial inactivo.
CREN "1" recepción continua "0" recepción inactiva.
FERR bandera indicadora de error de framing activa en "1"
OERR bandera indicadora de error de sobre escritura activa en "1".

Registro SPBRG

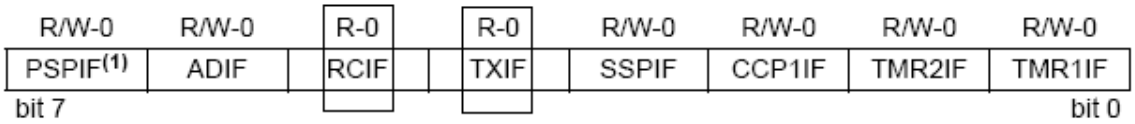
La generación de baudios queda determinado por:

BRGH = 1 (Velocidad alta) : Baudios = Fosc / (16* (X+1))

BRGH = 0 (Velocidad baja): Baudios = Fosc / (64 * (X+1))

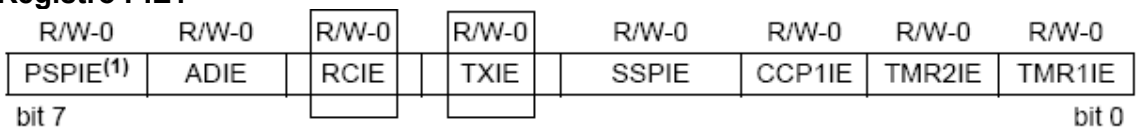
X es el valor hexadecimal cargado en el registro SPBRG.

Registro PIR1



RCIF interrupción que indica que se ha recibido un dato.
TXIF interrupción que indica que se ha transmitido un dato.

Registro PIE1



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
bit 7							bit 0

RCIE activar interrupción de recepción activo en “1”.
TXIE activar interrupción de transmisión activo en “1”.

4.7.1 PASOS PARA REALIZAR UNA TRANSMISION SERIE ASINCRONA.

- Cargar SPBRG para establecer los baudios de transmisión con velocidad baja o alta (BRGH).
- Activar modulo SCI (SPEN=1) definiéndolo como asíncrono SYNC=0.
- Si se desea interrupción de transmisión TXIE se activa con “1”.
- Si se desea transmitir 9 bits configurar TX9=”1”.
- Activar transmisión TXEN=”1”.
- Si se selecciono 9 bits cargar el noveno bit en TX9D.
- Cargar el dato en TXREG para iniciar la transmisión (TXIF=0 indica que el registro TXREG tiene un dato y TXIF=1 si TXREG está vacío, si se activó interrupción).
- Leer bit TMRT (TXSTA) para verificar si se transmitió el dato (indica que TSR esta vacio).

4.7.2 PASOS PARA REALIZAR UNA RECEPCION SERIE ASINCRONA.

- Cargar SPBRG para establecer los baudios de transmisión con velocidad baja o alta (BRGH).
- Activar modulo SCI (SPEN=1) definiéndolo como asíncrono SYNC=0.
- Si se desea interrupción por recepción activar RCIE=”1”.
- Si se desea recibir 9 bits activar RX9=”1”.
- Activar recepción con RCEN=”1”.
- Al completarse una recepción y se ha activado la interrupción se activará RCIF=”1”.
- En caso de recibir 9 bits leer el noveno bit en RX9D.
- Detectar si ha existido un error indicado por las banderas de error.
- Leer el dato en el registro RCREG.
- Si se dio un error resetear con RCEN=0.

4.7.3 PROGRAMA PARA TRANSMISION SERIE ASÍNCRONA.

Utilizar el diagrama de conexión que se muestra en la figura 4.50,conectar la tarjeta principal a la PC usando el cable serie.

Objetivo:
Utilizar módulo serial SCI de los PIC 16F877/A para enviar datos hacia una PC.

Programa para envío de una cadena de texto hacia la PC utilizando el módulo serial asíncrono SCI, utilizando velocidad de transmisión de 9600Kbps.

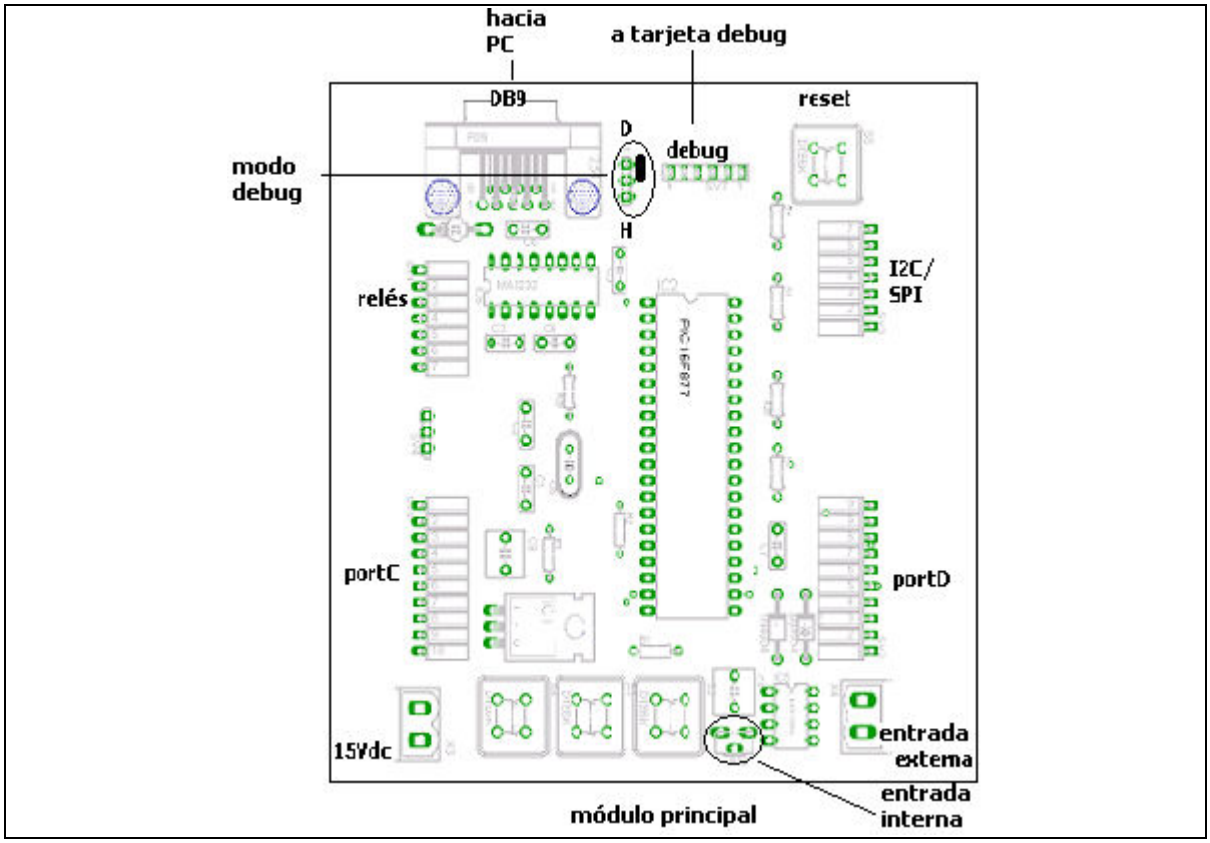


Fig. 4.50 *Diagrama de conexión módulo serial para transmisión y recepción.*

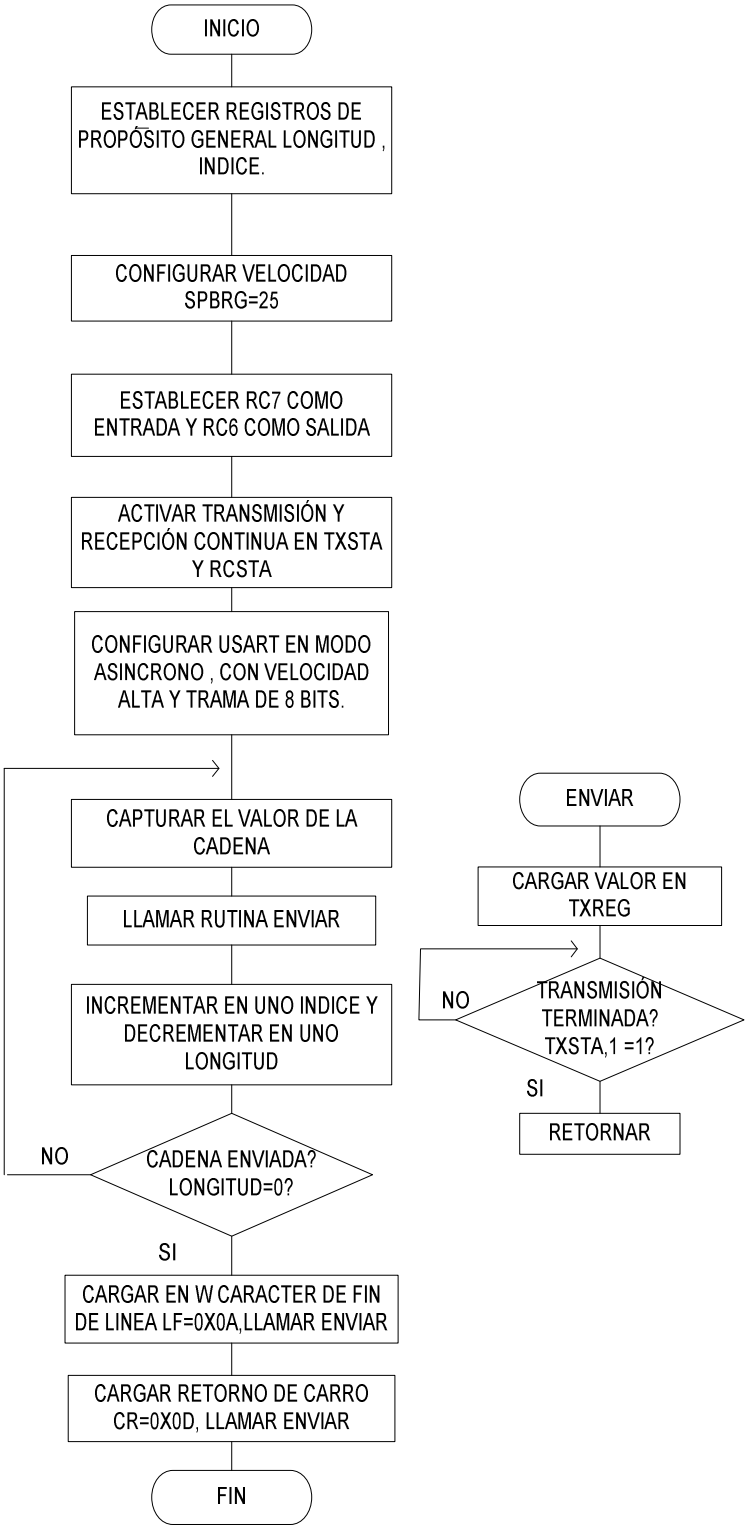


Fig.4.51 *Diagrama de bloques transmisión asíncrona*

```
      List P=PIC16F877A
      Include "P16F877A.INC"
Índice   equ 0x20              ; puntero para cada caracter de la cadena
Longitud equ 0x21              ; longitud de la cadena.
         Org     0x00
         Nop
         Nop                    ; retraso para realizar debug.
Inicio   clrf STATUS
         Bsf STATUS, 5                 ; banco 1
         Movlw d'25'           ; Cargamos el registro SPBRG con d'25' para
         Movwf SPBRG           ; conseguir 9600 baudios con un oscilador de 4MHz
         Bsf TXSTA, 2                  ; poner a 1 el bit BRGH
         Bcf TXSTA, 4                  ; limpiar bit SYNC
         Movlw b'10000000'     ; 7 entrada y 6 salida.
         Movwf TRISC           ; configurar para RX y TX.
         Bcf STATUS, 5                 ; banco 0
         Movlw b'10010000'      ; recepción continua y tamaño 8 bits
         Movwf RCSTA                   ; en el registro RCSTA
         Bsf STATUS, 5                 ; USART en modo asíncrono y velocidad alta
         Bsf TXSTA, 5          ; activamos TXEN y tramas de 8 bits en registro TXSTA
         Bcf STATUS, 5                 ; Cambio al banco 0 ----------
         Clrf TXREG            ; limpiar registro de transmisión.
         Movlw d'0'
         Movwf índice
         Movlw d'26'
         Movwf longitud
TIRA     addwf  PCL, 1
      DT "UNIVERSIDAD DON BOSCO 2010"
Otro     movf indice, 0
         Call TIRA             ; llamar rutina donde se encuentra la cadena a enviar.
         Call enviar
         Incf indice, 1                ; incrementa puntero de cadena.
         Decfsz longitud, 1    ; decrementa la longitud hasta llegar al ultimo caracter.
         Goto otro             ; envía otro dato si aun no se ha llegado al ultimo caracter.
         Movlw 0x0A
         Call enviar
         Movlw 0x0D                    ; caracter retorno de carro
         Call enviar
         Goto $                         ; bucle infinito.
Enviar           movwf TXREG           ; colocar cada caracter en el registro de transmisión.
         Bsf     STATUS, 5             ; Cambio al banco 1
         Btfss TXSTA, 1                ; comprueba si acabó de transmitir.
         Goto $-1
         Bcf STATUS, 5                 ; banco 0
         Return
         End                   ; si se termino de enviar la cadena termina el programa.
```

4.7.4 PROGRAMA PARA RECEPCIÓN SERIE ASÍNCRONA.

Objetivo:
Recibir datos desde la PC usando el módulo SCI de los PICs 16F877/A
; Programa para recepción de datos desde la PC, se utiliza el hyperterminal.

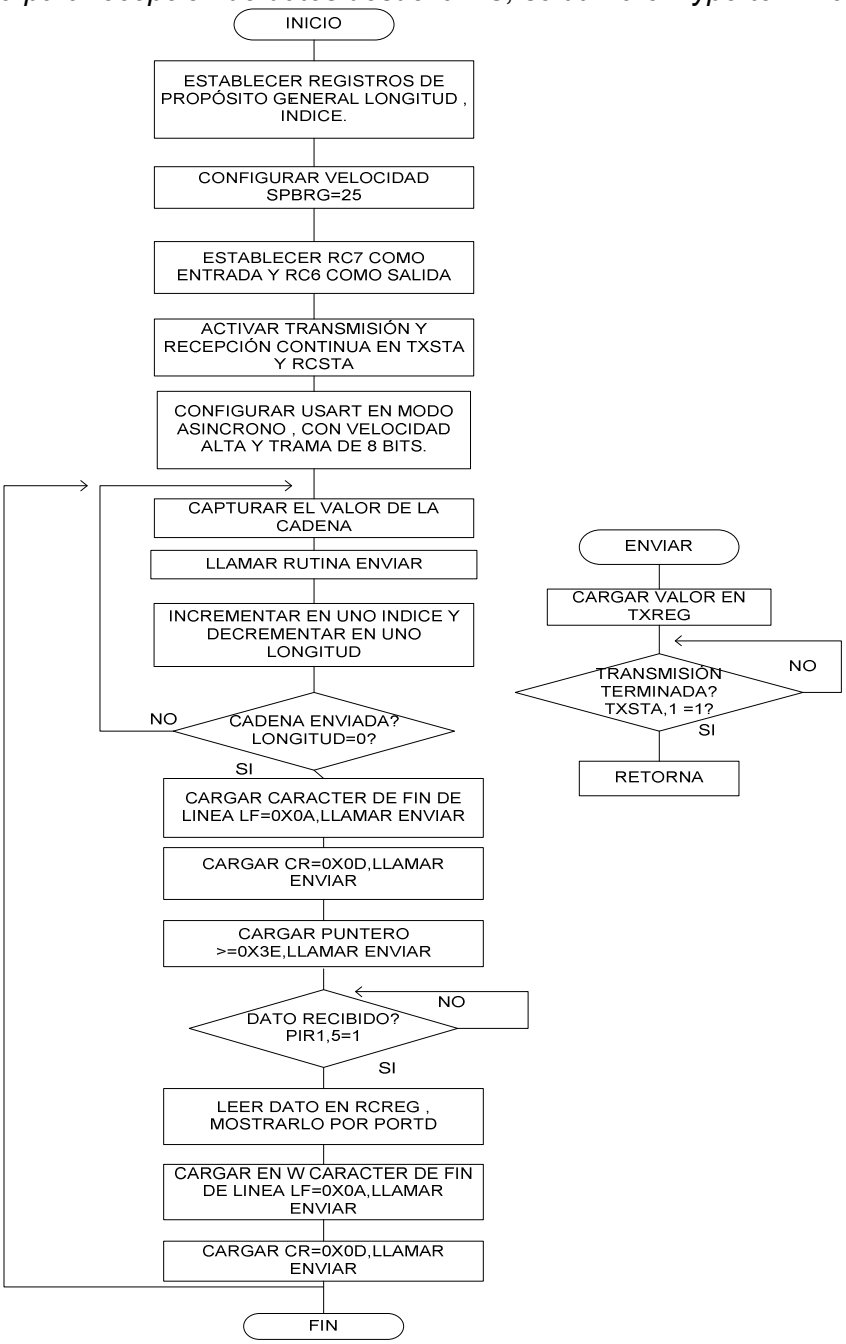


Fig.4.52 *Diagrama de bloques recepción serie asíncrona.*

```
List P=PIC16F877A
      Include "P16F877A.INC"

Índice   equ 0x20              ; puntero para cada caracter de la cadena
Longitud equ 0x21              ; longitud de la cadena.
    Org 0x00
    Nop
    Nop                        ; retraso para realizar debug.
Inicio   clrf STATUS
         Bsf STATUS, 5                 ; banco 1
         Movlw 0x00
         Movwf PORTD                   ; PORTD como salida de datos.
         Movlw d'25'           ; Cargamos el registro SPBRG con d'25' para
         Movwf SPBRG                   ; conseguir 9600 baudios con un oscilador de 4MHz
                               ; (4MHz/ (9600*16))-1 = 25,041
         Bsf TXSTA, 2                  ; poner a 1 el bit BRGH
         Bcf TXSTA, 4                  ; limpiar bit SYNC
         Movlw b'10000000'     ; 7 entrada y 6 salida.
         Movwf TRISC           ; configurar para RX y TX.
         Bcf STATUS, 5                 ; banco 0
         Clrf RCSTA            ; limpiar registro de recepción.
         Movlw b'10010000'       ; recepción continua y tamaño 8 bits
         Movwf RCSTA                   ; en el registro RCSTA
         Bsf STATUS, 5         ; Configuramos USART en modo asíncrono y velocidad alta
         Bsf TXSTA, 5          ; activamos TXEN y tramas de 8 bits en registro TXSTA
         Bcf      STATUS, 5    ; Cambio al banco 0 ----------
Otro     clrf TXREG            ; limpiar registro de transmisión.
         Movlw d'0'
         Movwf índice
         Movlw d'12'
         Movwf longitud
TIRA     addwf  PCL, 1
       DT "INGRESE DATO"
Caracter movf indice, 0
         Call TIRA             ; llamar rutina onde se encuentra la cadena a enviar.
         Call enviar
         Incf indice, 1                ; incrementa puntero de cadena.
         Decfsz longitud, 1    ; decrementa la longitud hasta llegar al ultimo caracter.
         Goto caracter         ; envía otro dato si aun no se ha llegado al ultimo caracter
         Movlw 0x0A
         Call enviar
        Movlw 0x0D             ; caracter retorno de carro CR.
         Call enviar
         Movlw 0x3E            ; puntero para ingresar dato.
         Call enviar
Leer     btfss PIR1, 5           ; esperamos a tener un dato.
```

```
        Goto $-1
        Bcf  PIR1, 5            ; limpiamos el indicador.
        Movf RCREG, 0             ; leemos dato.
        Movwf PORTD
        Movlw 0x0A              ; carácter de fin de línea.
        Call enviar
        Movlw 0x0D               ; carácter retorno de carro.
        Call enviar
        Goto otro                 ; vuelve a la solicitar otro dato.
Enviar          movwf TXREG             ; colocar cada caracter en el registro de
transmisión.
        Bsf     STATUS, 5       ; Cambio al banco 1
        Btfss TXSTA, 1                  ; comprueba si acabó de transmitir.
        Goto $-1
        Bcf STATUS, 5                   ; banco 0
        Return
        End                     ; Fin del programa.
```

4.8 MÓDULO MSSP PARA COMUNICACIÓN I²C.

http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

Registros asociados con el módulo MSSP para comunicación I2C.
Registros de control SSPCON - SSPCON2.
Registro de estado SSPSTAT.
Registro de dirección SSPADD.
Buffer de transmisión /recepción SSPBUF.
Registro de desplazamiento serie (no accesible) SSPSR.
Registros de interrupción INTCON - PIR1 - PIE1 - PIR2 - PIE2.

Puede actuar en modo maestro y esclavo.

Registro SSPCON

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

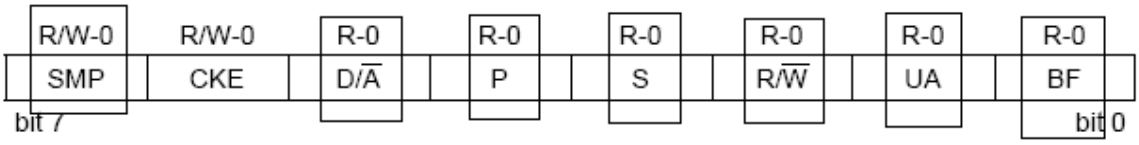
SSPEN "1" activa puerto serial "0" puerto serial inactivo.
SSPM3:SSPM0 bits de selección de modo síncrono I2C.

1111 = I²C Slave mode, 10-bit address with Start and Stop bit interrupts enabled
1110 = I²C Slave mode, 7-bit address with Start and Stop bit interrupts enabled
1011 = I²C Firmware Controlled Master mode (Slave Idle)
1000 = I²C Master mode, clock = Fosc/(4 * (SSPADD + 1))
0111 = I²C Slave mode, 10-bit address
0110 = I²C Slave mode, 7-bit address

Registro SSPCON2

GCEN "1" activa interrupción de llamada general de dirección.
"0" modo de llamada general es inactivo.
ACKSTAT "1" indica que se recibió bit de reconocimiento desde el esclavo.
"0" no se recibió bit de reconocimiento desde el esclavo.
ACKDT "1" genera bit de reconocimiento en modo receptor.
"0" no genera bit de reconocimiento.
ACKEN "1" inicia transmisión del bit de reconocimiento.
"0" no envía bit de reconocimiento.
RCEN "1" activa modo de recepción "0" modo recepción inactivo.
PEN "1" activa señal de STOP "0" no genera señal STOP.
RSEN "1" activa repetición de START "0" no genera repetición.
SEN "1" activa señal START "0" no genera señal START.

Registro SSPSTAT

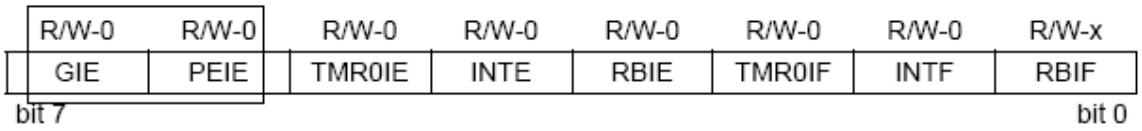


SMP "1" velocidad 100KHz -1Mhz "0" velocidad de 400KHz.
D/A bit indicador que se recibió dirección o dato.
P bandera indicadora de señal de STOP.
S bandera indicadora de señal de START.
R/W bit indicador de lectura /escritura.
UA bit indicador de actualización de datos.
BF "1" indica que el buffer ha recibido un dato.
"0" indica que el buffer esta vacio.

Registro SSPADD
Se carga con la dirección hexadecimal del dispositivo que actuará en modo esclavo.

Registro SSPBUF
Registro que contiene los datos recibidos y datos a enviar por el bus I2C.

Registro INTCON

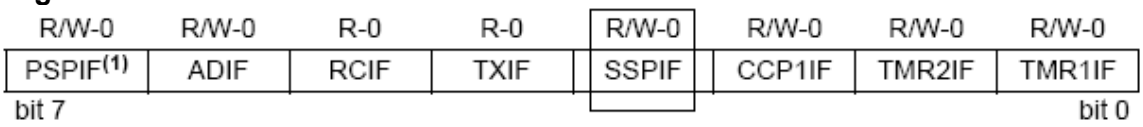


R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
bit 7							bit 0

GIE "1" activa máscara de interrupciones globales.
"0" máscara de interrupciones inactiva.

PEIE "1" activa interrupciones de periféricos.
"0" interrupciones de periféricos inactiva.

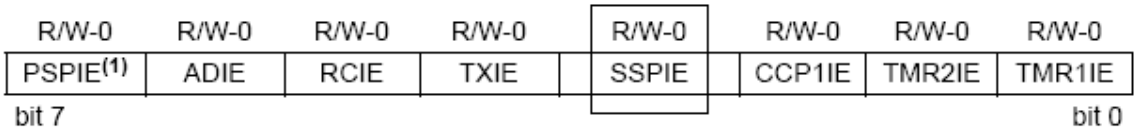
Registro PIR1



R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

SSPIF "1" bit indicador que ha ocurrido una interrupción en el modulo MSSP.
"0" no ha ocurrido interrupción en el modulo MSSP.

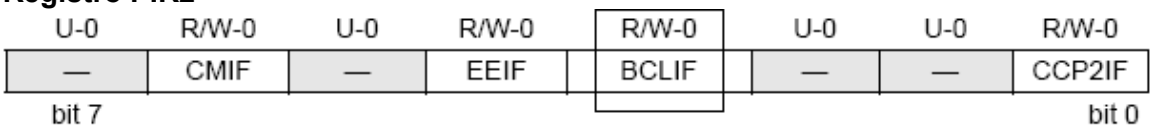
Registro PIE1



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
bit 7							bit 0

SSPIE "1" activa interrupción en módulo MSSP.
"0" interrupción en modulo MSSP inactiva.

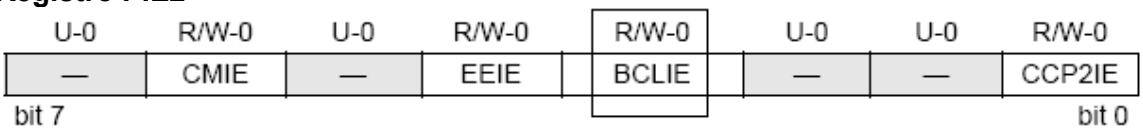
Registro PIR2



U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	CMIF	—	EEIF	BCLIF	—	—	CCP2IF
bit 7							bit 0

BCLIF "1" bit indicador de colisión en el bus I2C.
"0" no ha ocurrido colisión en el bus.

Registro PIE2



U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	CMIE	—	EEIE	BCLIE	—	—	CCP2IE
bit 7							bit 0

BCLIE "1" activa interrupción por colisión en el bus.
"0" interrupción por colisión inactiva.

4.8.1 PASOS PARA REALIZAR UNA TRANSMISIÓN EN MODO MAESTRO

- Activar puerto serial síncrono con SSPEN=1.
- Configurar velocidad SMP=1 (100KHz).
- Cargar SSPADD para obtener la velocidad deseada.
- Configurar niveles de entrada I2C CKE=0.
- Configurar dispositivo como maestro SSPM3:SSPM0 dirección de 7 bits.
- Activar bit de interrupción de modulo MSSP (SSPIE=1 - PIE1).
- Generar señal de START colocando bit SEN=1.
- Esperar señal de START (leer bit SSPIF =1 (PIR1)).
- Cargar dirección en SSPBUF y colocar cero en R/W.
- Leer bit de reconocimiento ACK, colocarlo en ACKSTAT (SSPCON2<6>).
- Leer si se envió la dirección SSPIF=1.
- Limpiar bit SSPIF=0.
- Cargar en SSPBUF el dato a enviar (8 bits).
- Leer bit de reconocimiento ACK, colocarlo en ACKSTAT (SSPCON2<6>).
- Verificar si se envió el dato leyendo SSPIF=1.
- Limpiar bit SSPIF=0.
- Generar la condición de STOP PEN=1.
- Leer bit SSPIF para verificar condición de STOP

4.8.2 PROGRAMA PARA TRANSMISIÓN I²C MODO MAESTRO.

Conectar el módulo de salida digital a la tarjeta de comunicación SPI/ I²C. Conectar el módulo de comunicación SPI/ I²C con la tarjeta principal como se muestra en la figura 4.53. Asegurarse que la tarjeta de comunicación esté seteada en modo I²C.

Objetivo:

Programa que lee datos análogos en el canal 0 del dispositivo maestro y envía el resultado digital hacia el esclavo por medio del protocolo I²C, el esclavo muestra el dato por el puerto B.

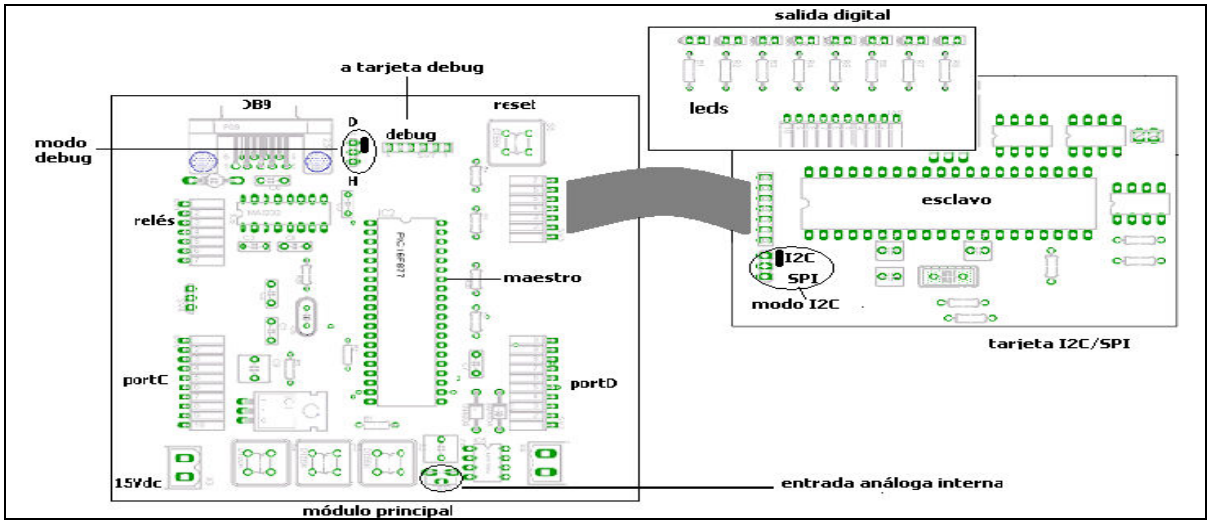


Fig.4.53 *Diagrama de conexión para comunicación I²C.*

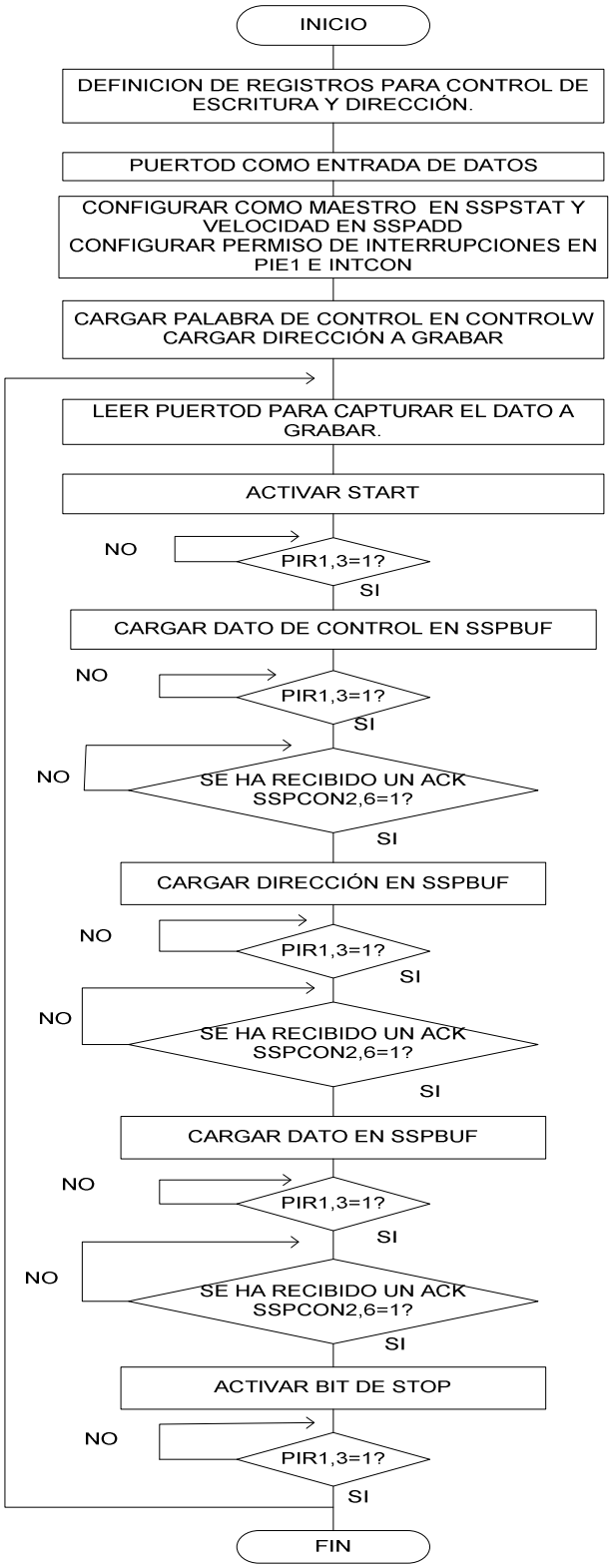

INICIO
DEFINICION DE REGISTROS PARA CONTROL DE ESCRITURA Y DIRECCIÓN.
PUERTOD COMO ENTRADA DE DATOS
CONFIGURAR COMO MAESTRO EN SSPSTAT Y VELOCIDAD EN SSPADD
CONFIGURAR PERMISO DE INTERRUPCIONES EN PIE1 E INTCON
CARGAR PALABRA DE CONTROL EN CONTROLW
CARGAR DIRECCIÓN A GRABAR
LEER PUERTOD PARA CAPTURAR EL DATO A GRABAR.
ACTIVAR START
NO
PIR1,3=1?
SI
CARGAR DATO DE CONTROL EN SSPBUF
NO
PIR1,3=1?
SI
NO
SE HA RECIBIDO UN ACK SSPCON2,6=1?
SI
CARGAR DIRECCIÓN EN SSPBUF
NO
PIR1,3=1?
SI
NO
SE HA RECIBIDO UN ACK SSPCON2,6=1?
SI
CARGAR DATO EN SSPBUF
NO
PIR1,3=1?
SI
NO
SE HA RECIBIDO UN ACK SSPCON2,6=1?
SI
ACTIVAR BIT DE STOP
NO
PIR1,3=1?
SI
FIN

Fig.4.54 *Diagrama de bloques transmisión I²C modo maestro.*

```
;*************************************************************
; Programa para transmisión mediante el I2C
; Este programa es para que el pic funcione como maestro
;*************************************************************
            list p=PIC16F877A
            include "P16F877A.INC"

        cblock 0x20
        Mensaje     ;Contendrá el dato a enviar por I2C al slave
        DirSlave    ;Dirección del esclavo
        BkStatus    ;Backup del registro STATUS
        BkW         ;Backup del registro W
        Pausa       ;Pausa en centésima de seg Usada por subrutina "Hacer Tiempo"
        endc        ;Fin de definiciones

        org     0
        nop                         ;instrucciones de retrazo
        nop                         ;para depurar código.
INICIO      bsf STATUS,5        ;Apunta a banco 1
            movlw 0xFF          ;puerto D como entrada.
            movwf TRISD
            bcf STATUS,5        ;Apunta a banco 0
            clrf    PORTB               ;Limpia puerto B
            call    configuración ;Configuración para uso de i2c
            clrf    Mensaje
            movlw d'10'         ;Pausa de 10 centésimas de segundo para que en...
            movwf Pausa         ;...el arranque de tiempo a los esclavos a quedar..
            movlw b'01111000'   ;Establece dirección de envío..
            Movwf DirSlave      ;..es decir, dirección del esclavo.
            movf PORTD,0                ;leer puerto D.
            movwf Mensaje       ;"Mensaje" para envío por I2C
            call    Enviar      ;Envía el dato de "Mensaje" al esclavo.
            bcf SSPCON,5            ;desactiva modulo MSSP
            goto    INICIO
;*****************************************************************************
; SUBRUTINAS
;*****************************************************************************
Configuración movwf BkW         ;Hace copia de W
            movf    STATUS,0    ;Hace copia de registro de estado
            banksel         PORTA
            movwf BkStatus
            banksel TRISC               ;Configuramos I2C Pasamos a direccionar
Banco 1
            movlw b'00011000'   ; Establece líneas SDA y SCL como entradas...
            iorwf   TRISC,1     ;..respetando los valores para otras líneas.
```

```
                movlw d'9'              ; Establece velocidad I2C según...
                movwf SSPADD            ; ...valor de ClockValue para velocidad
                bcf   SSPSTAT,6         ; Establece niveles de entrada I2C
                bcf   SSPSTAT,7         ; Habilita slew rate
                banksel SSPCON          ; Pasamos a direccionar Banco 0
                movlw b'00111000'       ; Master mode, SSP enable, velocidad según...
                movwf SSPCON            ; ... Fosc/(4x(SSPADD+1))
                bcf   PIR1,3            ; Limpia flag de eventos SSP
                bcf   PIR1,7            ; Limpia bit. Mandatorio por Datasheet
                movf  BkStatus,0        ;Restaura las copias de registros
                movwf STATUS                    ;registro de estado
                movf  BkW,0             ;registro W
                return
; -------------------------------------------------------------------------------
        ;Envía un mensaje (comando) almacenado en "MensajeOut" al Slave cuya
dirección
        ;se ha de encontrarse en la variable "DirSlave"
;-------------------------------------------------------------------------------
Enviar          movwf BkW               ;Hace copia de W
                movf  STATUS,0          ;Hace copia de registro de estado
                banksel         PORTA
                movwf BkStatus
                bcf   STATUS,5          ;banco 0
enviarstart     call Start              ;Envía condición de inicio
                call  Checkope          ;Espera fin evento
                banksel         DirSlave
                movf  DirSlave,0        ;Dirección esclavo
                call  Byte              ;Envía dirección y orden de escritura
                call  Checkope          ;Espera fin evento
                call  AckTest                   ;Verifica llegada ACK
                banksel SSPCON2
                bcf   SSPCON2,6         ;limpia bandera ACK.
                xorlw 1
                btfsc STATUS,2          ;Chequea si llegó ACK
                goto  enviarstart       ;No. Reintentamos envío
                banksel Mensaje         ;Si. Seguimos con envío dato
                movf  Mensaje,0         ;Lo deja en W para que la subrutina Byte lo envíe
                call  Byte              ;envía por i2c
                call  Checkope          ;Espera fin evento
                call  Stop              ;Envía condición de parada
                call  Checkope          ;Espera fin evento
                movf  BkStatus,0        ;Restaura las copias de registros
                movwf STATUS                    ;registro de estado
                movf  BkW,0             ;registro W
                return
; -------------------------------------------------------------------------------
```

```
        ;Envía condición de start
;-------------------------------------------------------------------------------
Start:
                banksel SSPCON2
                bsf     SSPCON2,0   ; Envía Start
                return
; ------------------------------------------------------------------------------
        ;Envía condición de stop
;-------------------------------------------------------------------------------
Stop:
                banksel SSPCON2
                bsf     SSPCON2,PEN         ;Activa secuencia de stop
                return
; ------------------------------------------------------------------------------
        ;Envía el contenido de W por i2c
;-------------------------------------------------------------------------------
Byte:
                banksel SSPBUF      ; Cambia a banco 0
                movwf SSPBUF        ; inicia condición de escritura
                return
; ------------------------------------------------------------------------------
        ;Chequea que la operación anterior termino y se puede proceder con
        ;el siguiente evento SSP
;-------------------------------------------------------------------------------
Checkope:
                banksel  PIR1                   ;banco 0
                btfss PIR1,3
                goto $-1
                bcf     PIR1,3          ; Limpiamos flag
                return
;-------------------------------------------------------------------------------
        ;Chequea ack tras envío de dirección o dato
        ;Devuelve en W 0 o 1 dependiendo de si llegó (0) o no (1) ACK
;-------------------------------------------------------------------------------
AckTest:
                banksel SSPCON2    ; Cambia a banco 1
                btfss   SSPCON2,6  ;Chequea llegada ACK desde slave
                Retlw   0               ;llegó ACK
                Retlw   1               ;no llegó ACK
                 end
```

4.8.3 PASOS PARA REALIZAR UNA RECEPCION EN MODO ESCLAVO I²C.

- Configurar R3 y R4 como entradas en TRISC
- Activar modulo MSSP SSPEN=1
- Configurar velocidad SMP=1 (100KHz)
- Cargar SSPADD con el valor para la velocidad deseada (baudios)
- Configurar niveles de entrada I2C con CKE=0
- Configurar dispositivo en modo esclavo SSMP3:SSMP0 (0110)
- Activar polaridad del reloj CKP =1 (reloj activo)
- Activar bit de interrupción del modulo MSSP SSPIE=1 (PIE1)
- Esperar condición de START ,leer bit SSPIF=1
- Leer bit BF=1 para verificar que ocurrió una coincidencia de dirección
- Limpiar BF
- Leer BF=1 si se dio la recepción de datos
- Leer SSPBUF que contiene el dato (opcional)
- Limpiar BF

4.8.4 PROGRAMA PARA RECEPCIÓN POR I²C MODO ESCLAVO.

Objetivo:
Verificar el uso del módulo MSSP configurado para comunicación I²C en modo esclavo.

Programa para recepción de datos desde el PIC maestro del módulo principal mediante el bus I²C, este programa es para que el pic funcione como esclavo.

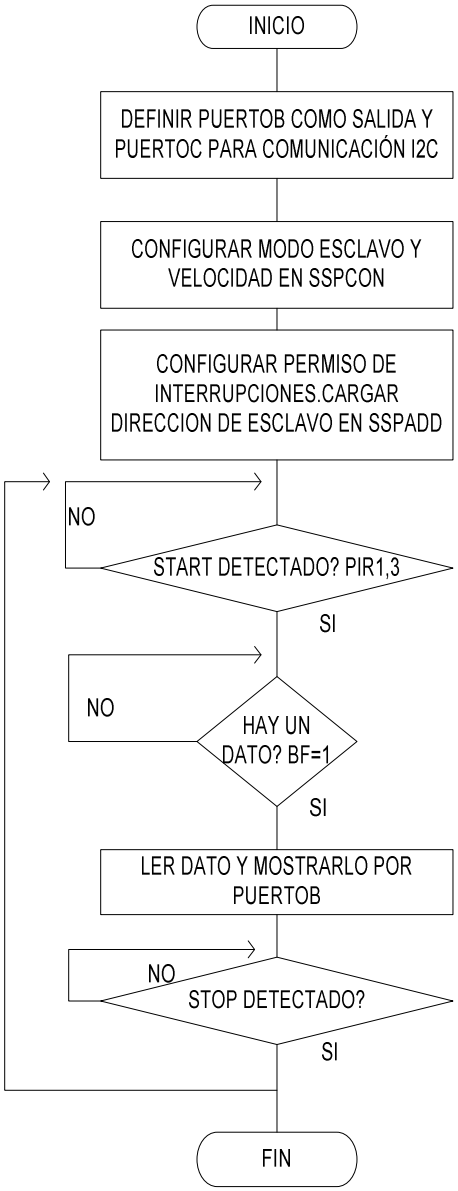


Fig.4.55 *Diagrama de bloques recepción I2C modo esclavo.*

```
;*****************************************************************
;
; Programa para  recepción de datos mediante el bus I2C.
; Este programa es para que el pic funcione como esclavo.
;*****************************************************************
;
    list p=PIC16F877A
    include"P16F877A.INC"

        cblock 0x20
```

```
        MensajeIn       ;Contendrá el dato recibido por I2C del máster
        BkStatus        ;Backup del registro STATUS
        BkW             ;Backup W
        Temp            ;Variable Temporal usada para evaluación de eventos I2C
        endc            ;Fin de definiciones

                                org     0
                                goto    INICIO
;--------------------------------------------------------------------------------
                                org     4
Interrupción    movwf BkW               ;Hace copia de W
                movf    STATUS,0        ;Hace copia de registro de estado
                banksel         PORTA
                movwf BkStatus
                banksel PIR1
                btfss   PIR1,3          ;Ha ocurrido un evento SSP? (I2C)
                goto    $-1             ;No. entonces será por otra cosa. Saltamos.
                call    eventoI2C       ;Si. Procesamos el evento.
                banksel PIR1
                bcf     PIR1,3          ;Limpiamos el flag
                movf    BkStatus,0      ;Restaura las copias de registros
                movwf STATUS                    ;registro de estado
                movf    BkW,0           ;registro W
                retfie
;--------------------------------------------------------------------------------
INICIO          bsf STATUS,5            ;;Inicio del cuerpo del programa ,Apunta a banco 1
                movlw b'00000000'       ;Salida (Leds)
                movwf TRISB             ;
                banksel         PORTB                   ;Apunta a banco 0
                clrf    PORTB                   ;Limpia puerto B
                call    configuración ;Configuración para uso de i2c
                 banksel INTCON
                bsf     INTCON,7        ;Activamos las interrupciones
                banksel MensajeIn
lazo            movf    MensajeIn,0     ;Muestra por port B (leds)...
                movwf PORTB                     ;...el ultimo valor enviado por el Máster
                goto    lazo
;*******************************************************************************
; SUBRUTINAS
;*******************************************************************************
configuración   movwf BkW               ;Hace copia de W
                movf    STATUS,0        ;Hace copia de registro de estado
        banksel         PORTA
        movwf BkStatus
        banksel TRISC                   ;Configuramos I2C,Pasamos a  Banco 1
        movlw b'00011000'   ; Establece líneas SDA y SCL como entradas...
```

```
        iorwf   TRISC,1         ;..respetando los valores para otras líneas.
        bcf     SSPSTAT,6       ; Establece I2C input levels
        bcf     SSPSTAT,7       ; Habilita slew rate
        bsf     SSPCON2,7       ; Habilita direccionamiento global
        movlw b'01111000'       ; Dirección esclavo
        movwf SSPADD
        banksel SSPCON          ; Pasamos a direccionar Banco 0
        movlw b'00110110'       ; Slave mode, SSP enable, velocidad según...
        movwf SSPCON            ; ... Fosc/(4x(SSPADD+1))
        bcf     PIR1,3          ; Limpia flag de eventos SSP
        bcf     PIR1,7          ; Limpia bit. Mandatorio por Datasheet
        banksel PIE1
        bsf     PIE1,3
        bsf     INTCON,6
        movf    BkStatus,0      ;Restaura las copias de registros
        movwf STATUS                    ;registro de estado
        movf    BkW,0           ;registro W
        return
; ------------------------------------------------------------------------------
eventoI2C:
        banksel SSPSTAT
        movf    SSPSTAT,0       ; Obtiene el valor de SSPSTAT
        andlw  b'00101101'      ; elimina los bits no importantes SSPSTAT.
        banksel         Temp
        movwf Temp              ; para chequeo posterior.
        movlw b'00101001'       ; de datos, el buffer está lleno.
        banksel         Temp
        xorwf   Temp,0
        btfss   STATUS,2        ; Estamos en el segundo estado?
        goto    $-1             ; NO, checkeamos siguiente estado
        call    ReadI2C         ; SI, Tomamos el byte del SSP.
        movwf MensajeIn         ;valor recibido desde el maestro.
        ;movwf PORTB
        return
;---------------------------------------------------------------------
ReadI2C                 ;Usada para escribir datos en bus I2C
;---------------------------------------------------------------------
        banksel SSPBUF
        movf    SSPBUF,0        ; Toma el byte y lo guarda en W
        return
        end
```

4.8.5 ESCRITURA DE DATOS A MEMORIA EEPROM 24C01C

conectar el módulo principal con la tarjeta de comunicación SPI/I²C para enviar información a la memoria EEPROM como se muestra en la figura 4.56.

Objetivo:

Guardar datos desde un PIC16F877/A hacia una memoria EEPROM 24C01C utilizando el bus I²C.

La memoria 24C01C soporta comunicación I²C y se encuentra en la tarjeta I²C /SPI, la explicación del funcionamiento se encuentra en la sección 3.4.5.3 del capitulo 3.

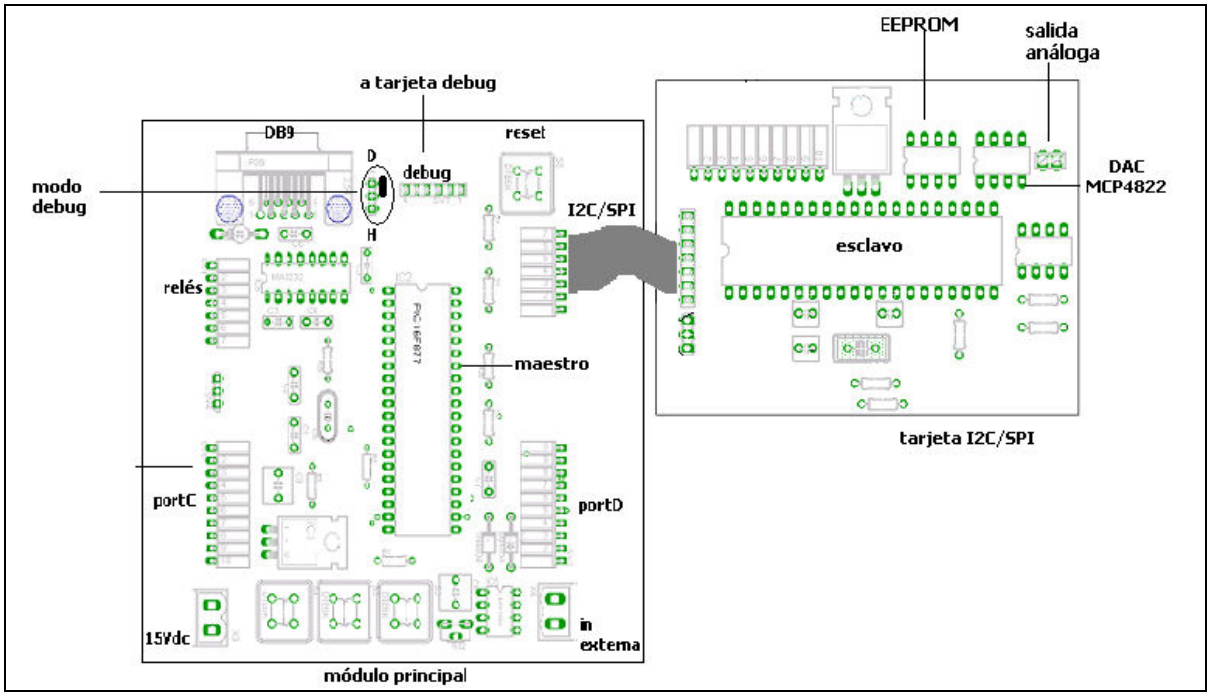


Fig.4.56 *Diagrama de conexión entre PIC y memoria EEPROM 24C01C.*

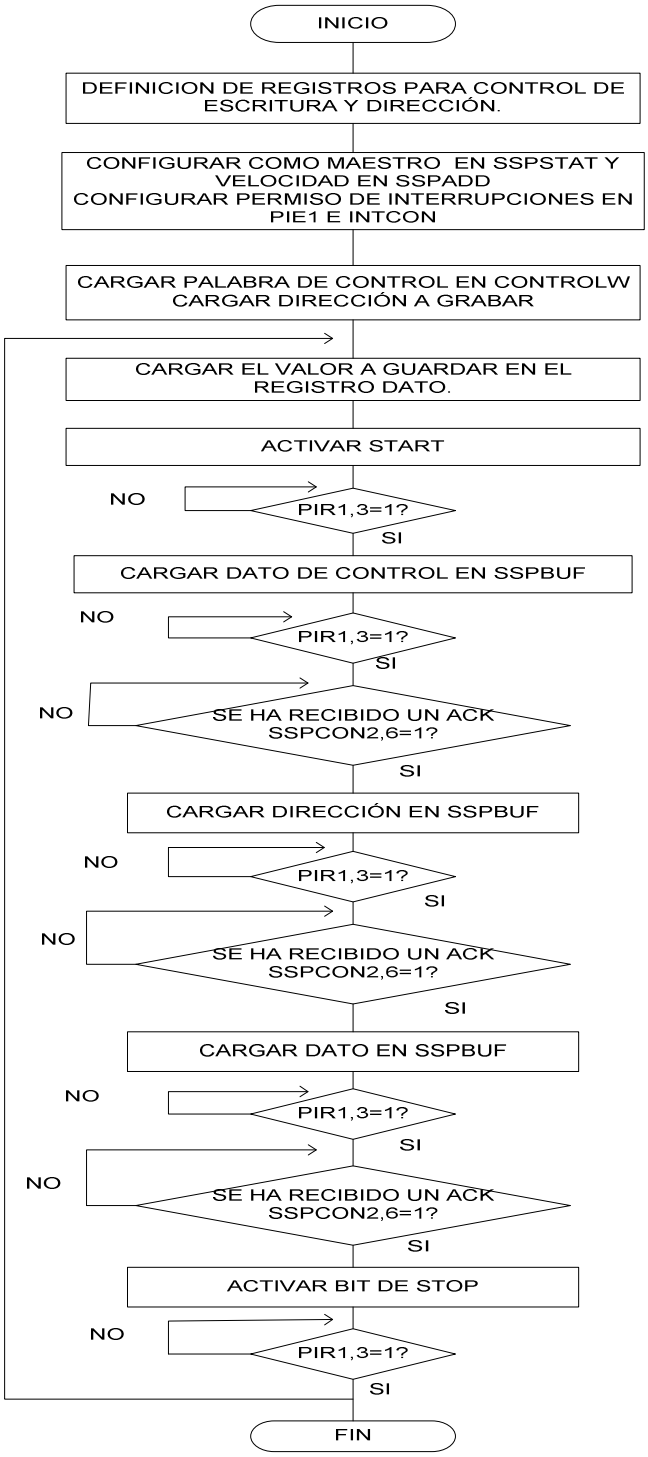

INICIO
DEFINICION DE REGISTROS PARA CONTROL DE ESCRITURA Y DIRECCIÓN.
CONFIGURAR COMO MAESTRO EN SSPSTAT Y VELOCIDAD EN SSPADD
CONFIGURAR PERMISO DE INTERRUPCIONES EN PIE1 E INTCON
CARGAR PALABRA DE CONTROL EN CONTROLW
CARGAR DIRECCIÓN A GRABAR
CARGAR EL VALOR A GUARDAR EN EL REGISTRO DATO.
ACTIVAR START
NO
PIR1,3=1?
SI
CARGAR DATO DE CONTROL EN SSPBUF
NO
PIR1,3=1?
SI
NO
SE HA RECIBIDO UN ACK SSPCON2,6=1?
SI
CARGAR DIRECCIÓN EN SSPBUF
NO
PIR1,3=1?
SI
NO
SE HA RECIBIDO UN ACK SSPCON2,6=1?
SI
CARGAR DATO EN SSPBUF
NO
PIR1,3=1?
SI
NO
SE HA RECIBIDO UN ACK SSPCON2,6=1?
SI
ACTIVAR BIT DE STOP
NO
PIR1,3=1?
SI
FIN

Fig.4.57 *Diagrama de bloques escritura a memoria 24C01C*.

```
List P=PIC16F877A
Include "P16F877A.INC"

   Controlw     equ 0x20
   Controlr     equ 0x21
   Dirección    equ 0x22
   Dato         equ 0x23

Org 0h
   Nop                  ;retraso para realizar debug.
   Nop
   clrf STATUS
   bsf STATUS,5                 ;banco 1
   movlw 0x00
   movwf PORTD                  ;salidas
   movlw b'00011000'
   movwf PORTC                  ;entradas en bits 3 y 4
   banksel SSPCON
   movlw b'00101000'    ; activar modo I2C.
   movwf SSPCON
   banksel SSPSTAT
   movlw b'10000000'
   movwf SSPSTAT
   banksel SSPADD
   movlw d'9'           ;configurar velocidad de transferencia.
   movwf SSPADD
   bcf STATUS,5                 ;banco 0
   clrf PORTD
   movlw b'10100000'    ;dato para indicar a la memoria
   movwf controlw       ;que se desea escribir un dato en ella.
   movlw b'10100001'    ;dato de control para leer datos desde
   movwf controlr       ;la memoria 24C01C.
   movlw 0x14           ;dirección en la cual se guardará el dato.
   movwf  dirección
   movlw b'10101010'    ;valor del dato a escribir en la memoria..
   movwf dato
   banksel SSPCON2
   bsf SSPCON2,0        ;activar bit de start
   bcf STATUS,5                 ;banco 0
   call espera          ;rutina de verificación de fin de acción.
   bsf PORTD,7                  ;verifica el start
   movf controlw,0
   movwf SSPBUF         ;cargar el dato de control para escritura.
   call espera
```

```
        bsf PORTD,6         ;indica que el dato se envió.
        banksel SSPCON2
aquí    btfsc SSPCON2,6     ;verificar si se recibió un ACK desde la memoria.
        goto aquí
        bcf STATUS,5                ;banco 0
        bsf PORTD,5         ;fue recibido ACK
        movf dirección,0
        movwf SSPBUF        ;cargar valor de la dirección a grabar.
        call espera
        bsf PORTD,4                 ;envió dirección.
        banksel SSPCON2
aqui1   btfsc SSPCON2,6     ;verificar si se recibió un ACK desde la memoria.
        goto aqui1
        bcf STATUS,5                ;banco 0
        bsf PORTD,3                 ;recibió ACK
        movf dato,0
        movwf SSPBUF        ;cargar dato a escribir.
        call espera
        bsf PORTD,2                 ;envió dato
        banksel SSPCON2
aqui2   btfsc SSPCON2,6     ;verificar si se recibió un ACK desde la memoria.
        goto aqui2
        bcf STATUS,5                ;banco 0
        bsf PORTD,1                 ;recibió ACK
        banksel SSPCON2
        bsf SSPCON2,2       ;activo bit de stop
        bcf STATUS,5                ;banco 0
        call espera
        bsf PORTD,0         ;se dio un stop
        goto $
;*********************************************************************************
;rutina de espera a que termine una acción hecha por el maestro start, stop,etc.
;*********************************************************************************
espera          btfss PIR1,3            ;verifica bit de interrupción por evento I2C.
        goto espera
        bcf PIR1,3
        return              ;regresa a la instrucción donde se quedó.
        end                 ;fin del programa.
```

4.9 MÓDULO MSSP PARA COMUNICACIÓN SPI

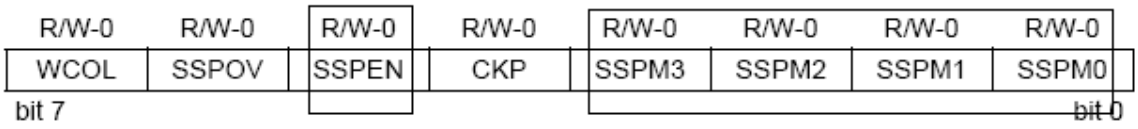
http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

Registros asociados con el módulo MSSP para comunicación SPI.
Registros de control SSPCON.
Registro de estado SSPSTAT.
Buffer de transmisión /recepción SSPBUF.

Registros de interrupción INTCON - PIR1 - PIE1.

Puede actuar en modo maestro y esclavo.

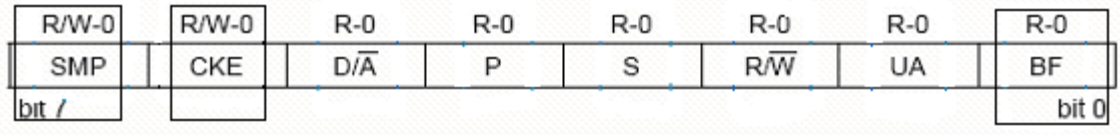
Registro SSPCON



SSPEN "1" activa puerto serial "0" puerto serial inactivo.
SSPM3:SSPM0 bits de selección de modo síncrono I2C.

0000 = SPI Master mode, clock = Fosc/4
0001 = SPI Master mode, clock = Fosc/16
0010 = SPI Master mode, clock = Fosc/64
0011 = SPI Master mode, clock = TMR2 output/2
0100 = SPI Slave mode, clock = SCK pin. $\overline{SS}$ pin control enabled.
0101 = SPI Slave mode, clock = SCK pin. $\overline{SS}$ pin control disabled. $\overline{SS}$ can be used as I/O pin.

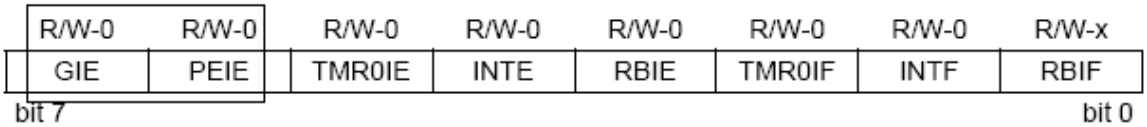
Registro SSPSTAT



SMP "1" muestrea los datos al final del tiempo de salida
"0" muestrea los datos en medio del tiempo de salida.
CKE selector de flanco de reloj**.**
CKP=0 "1" transmitir datos en flancos positivos del reloj.
"0" transmitir datos en flancos negativos del reloj.
CKP=1 "1" transmitir datos en flancos negativos del reloj.
"0" transmitir datos en flancos positivos del reloj.
BF "1" indica que el buffer ha recibido un dato.
"0" indica que el buffer esta vacio.

Registro SSPBUF
Registro que contiene los datos recibidos y datos a enviar por el bus I2C.

Registro INTCON



GIE "1" activa máscara de interrupciones globales.
"0" máscara de interrupciones inactiva.
PEIE "1" activa interrupciones de periféricos.

"0" interrupciones de periféricos inactiva.

Registro PIR1

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

SSPIF "1" bit indicador que ha ocurrido una interrupción en el modulo MSSP.
"0" no ha ocurrido interrupción en el modulo MSSP.

Registro PIE1

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
bit 7							bit 0

SSPIE "1" activa interrupción en modulo MSSP.
"0" interrupción en modulo MSSP inactiva.

Nota:
Para realizar este ejercicio el MCU del módulo I2C-SPI deberá contener el código para funcionar como esclavo SPI (SPIslave).

4.9.1 PASOS PARA REALIZAR UNA TRANSMISION MODO MAESTRO SPI.

- Determinar el pin RC5 como salida y el pin RC4 como entrada.
- Determinar en bit CKP de SSPCON el tipo de flanco se usara.
- Determinar la velocidad del reloj bits SSPM3:SSPM0 (SSPCON).
- Determinar en SSPSTAT <CKE> en cual flanco se transmitirá.
- Activar módulo serial MSSP con bit SSPEN del registro SSPCON.
- Enviar cero al pin SS (RA5) del PIC esclavo.
- Cargar dato a enviar en registro SSPBUF.
- Leer si el bit BF=1 de SSPSTAT para verificar si se envio el dato.
- Enviar uno al bit SS (RA5) del esclavo para evitar leer datos no deseados.
- Repetir el paso 6 si se desea.

4.9.2 PASOS PARA REALIZAR UNA RECEPCION MODO ESCLAVO SPI

- Determinar el pin RC5 como salida y el pin RC4 como entrada.
- Activar interrupciones globales en registro INTCON.
- Determinar en bit CKP de SSPCON el tipo de flanco se usara.
- Determinar la velocidad del reloj bits SSPM3:SSPM0 (SSPCON).
- Activar el pin SS para control de esclavo bits SSPM3:SSPM0.
- Determinar en SSPSTAT <CKE> en cual flanco se recibirán los datos.
- Activar modulo serial MSSP con bit SSPEN del registro SSPCON.
- verificar pin SS (RA5) para saber si el maestro desea transmitir.
- Verificar si ha existido una interrupción leyendo registro PIR1 (SSPBIF).

- Limpiar bit indicador de dato recibido SSPBIF del registro PIR1.
- Leer dato guardado en SSPBUF.
- Mostrar dato o guardarlo en una dirección de memoria deseada.
- Repetir el paso 8 si se desea.

4.9.3 PROGRAMA PARA TRANSMISIÓN SPI MODO MAESTRO.

Conectar la tarjeta principal con el módulo de comunicación ,así como lo muestra la figura 4.58. Asegurarse que la tarjeta de comunicación esté seteada en modo SPI.

Objetivo:

Utilizar el módulo MSSP de los PIC 16F877/A en modo SPI configurado para trabajar como maestro.

Programa para enviar datos desde el PIC maestro hacia el PIC esclavo, los datos son leídos usando el puertoD del PIC maestro y enviados por medio del bus SPI.

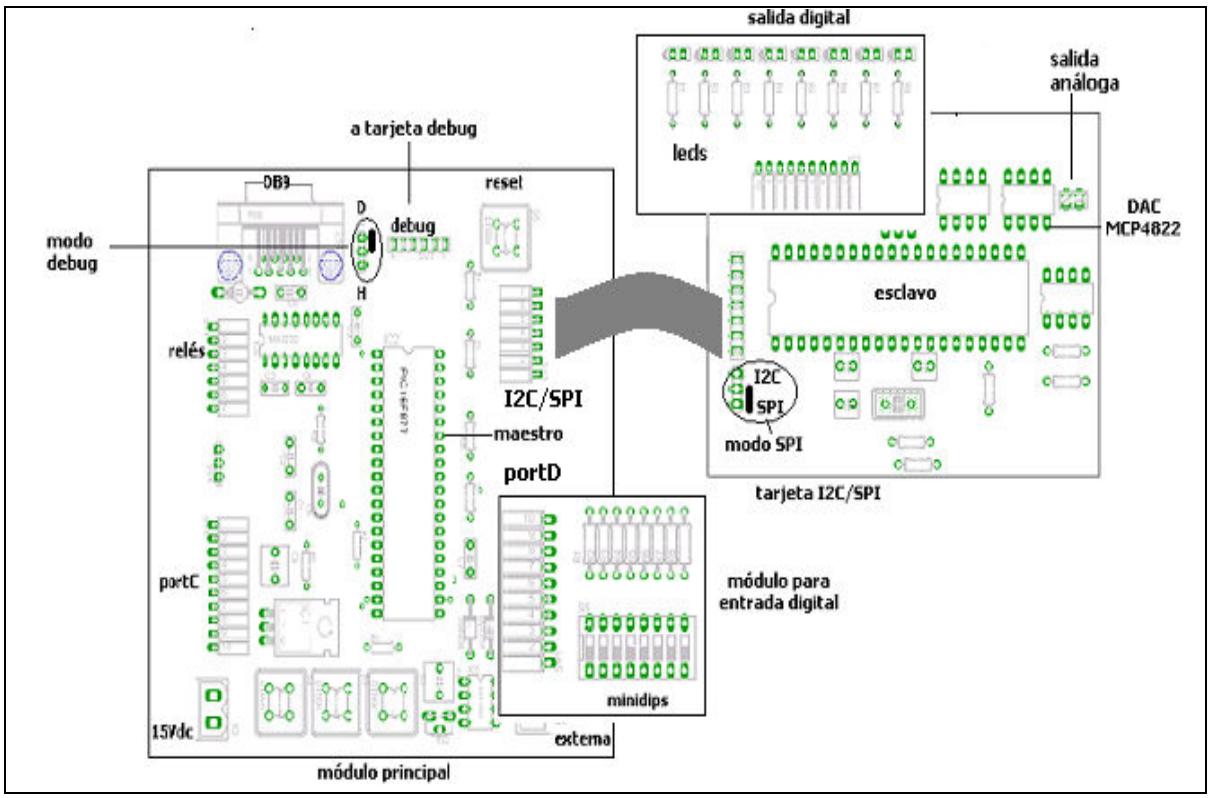
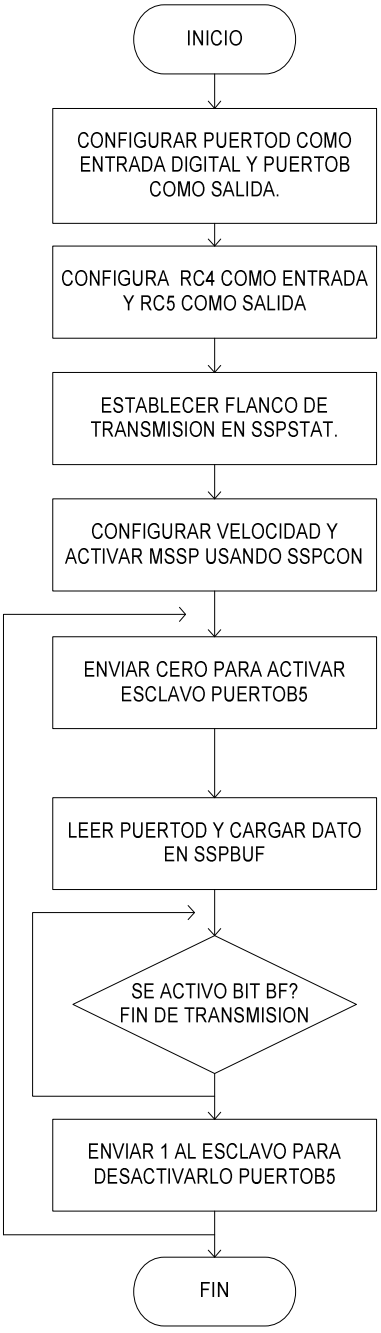


Fig.4.58 *Diagrama de conexión para comunicación SPI*.



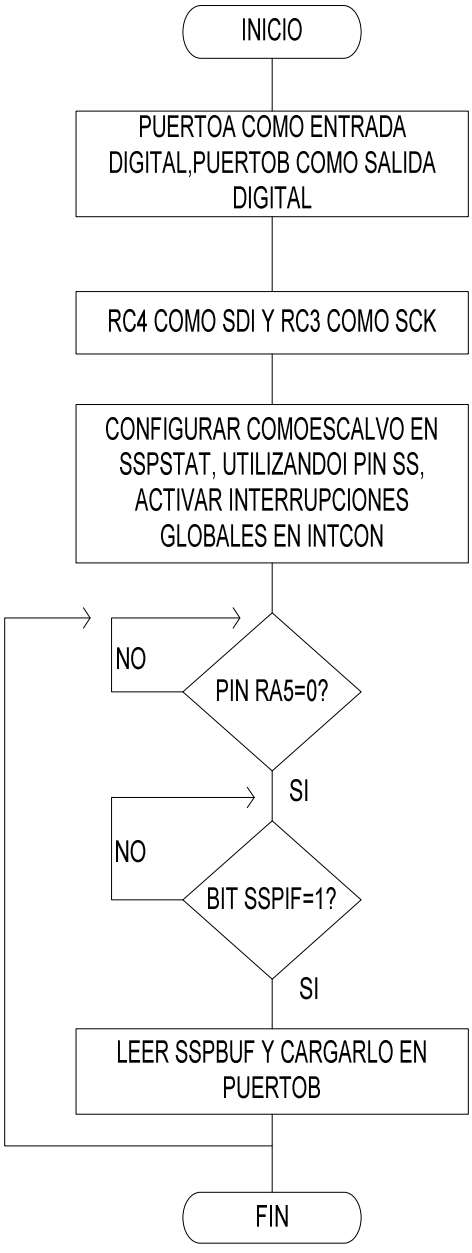

```
        list P=PIC16F877A
        include "P16F877A.INC"

      ORG  0x00     ;dirección de inicio del
programa
        nop           ;retraso para realizar debug.
        nop
inicio   clrf STATUS
        clrf PORTB
        bsf STATUS,5         ;banco 1
        movlw b'00010000'  ;pin RC5 es SDO.
        movwf TRISC          ;entrada RC4
        movlw 0xff
        movwf TRISD      ;puertoD entrada de
datos.
        movlw 0x40
        movwf SSPSTAT  ;bit CKE flanco
negativo.
        bcf STATUS,5      ;banco 0
        movlw 0x30
        movwf SSPCON   ;activa bits
SSPEN,CKP.
        clrf SSPBUF        ;limpiar registro
SSPBUF.
enviar  bsf PORTB,5
        bcf PORTB,5       ; activar el otro PIC.
        movf PORTD,0     ;colocar contador en
W.
        movwf SSPBUF
        banksel SSPSTAT  ; banco de
SSPSTAT.
        btfss SSPSTAT,0    ;revisar bit BF
        goto $-1
        bcf STATUS,5       ;banco 0
        bsf PORTB,5        ;desactiva el otro PIC.
        goto enviar
        end                ;final del programa.
```

Fig.4.59 *Diagrama de bloques y programa SPI maestro.*

4.9.4 PROGRAMA PARA RECEPCIÓN MODO ESCLAVO SPI.

Objetivo:

Utilizar el módulo MSSP de los PIC16F877/A configurado como receptor esclavo en modo SPI.Programa para recibir datos desde el MCU maestro.




```
        list P=PIC16F877A
        include "P16F877A.INC"

org     0x00    ; inicio del programa
        nop     ;retraso para  debug.
        nop

inicio  clrf STATUS
        bsf STATUS,5            ;banco 1
        movlw b'00000110'    ; PORTA I/O.
        movwf ADCON1         ;digital
        movlw b'11111111' ;porta 4 input
        movwf TRISA       ;in  digitales
        clrf TRISB           ;salida
        movlw b'00011000' ;bit 3 y 4 input
        movwf TRISC
        movlw 0x40
        movwf SSPSTAT
        bcf STATUS,5        ;banco 0
        movlw b'11000000'  ;activa
        movwf INTCON      ; interrupciones
        movlw 0x34 ;
        movwf SSPCON     ;SS pin enabled
        clrf  PORTB
otro    btfsc PORTA,5     ;señal de
activación.
        goto $-1          ;desde el maestro.
        btfss PIR1,3       ;revisar bit
SSPBIF.
        goto $-1
        bcf PIR1,3         ; limpiar bit
SSPIF.
        movf  SSPBUF,0    capturar dato e
        movwf PORTB       ;mostrar el dato.
        goto otro         ;otro dato.
        end               ;fin del programa.
```

Fig.4.60 *Diagrama de bloques y programa SPI esclavo.*

4.10 CONVERTIDOR DIGITAL-ANÁLOGO

Los microcontroladores a utilizar no poseen el módulo digital/análogo en sus subsistemas, por lo que en la tarjeta I²C/SPI está colocado el chip conversor digital-análogo MCP 4822, el cual se comunica con la tarjeta principal por medio del protocolo SPI, la descripción funcional del dispositivo está detallada en la sección 3.4.5.2.

4.10.1 PROGRAMA CONVERTIDOR DIGITAL-ANÁLOGO.

El diagrama de conexión para salida análoga aparece en la figura 4.61.

Objetivo:

Leer datos análogos desde el PIC y Enviar la conversión digital hacia convertidor MCP4822 para convertirlos a datos análogos nuevamente, usando comunicación SPI.

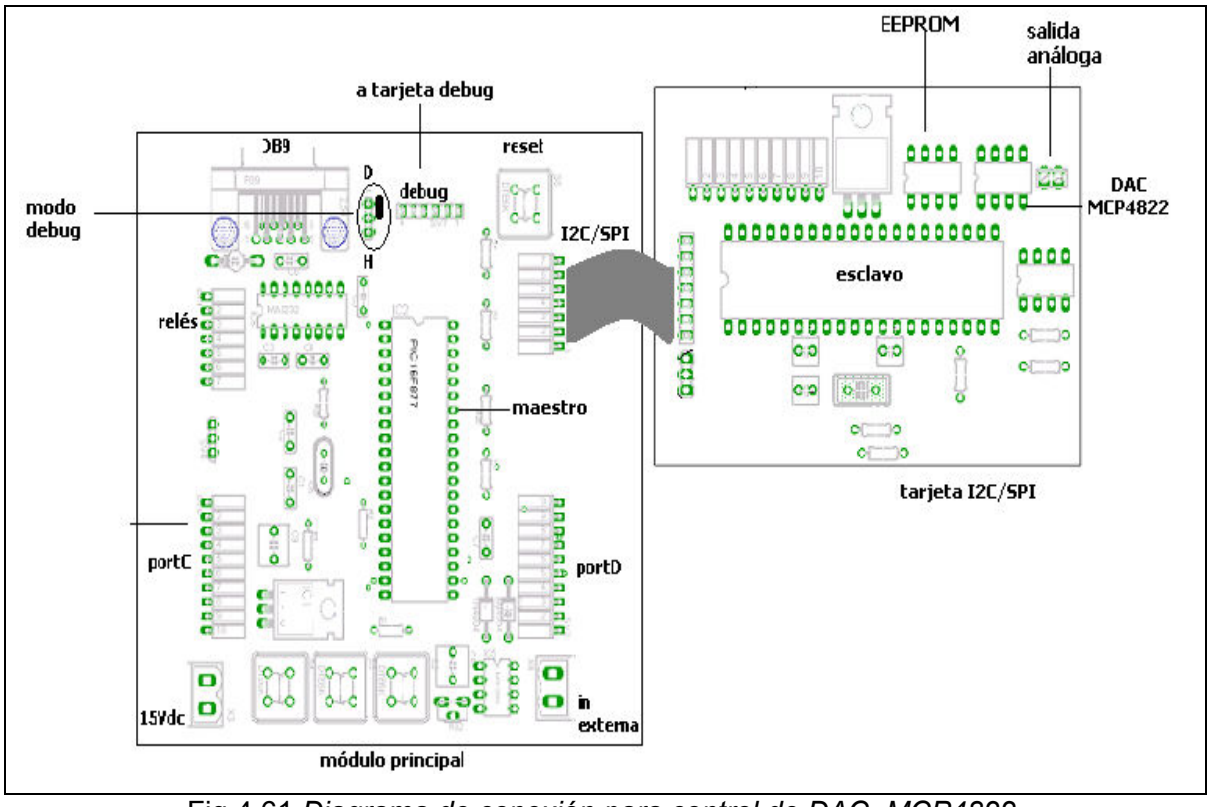


Fig.4.61 *Diagrama de conexión para control de DAC MCP4822.*

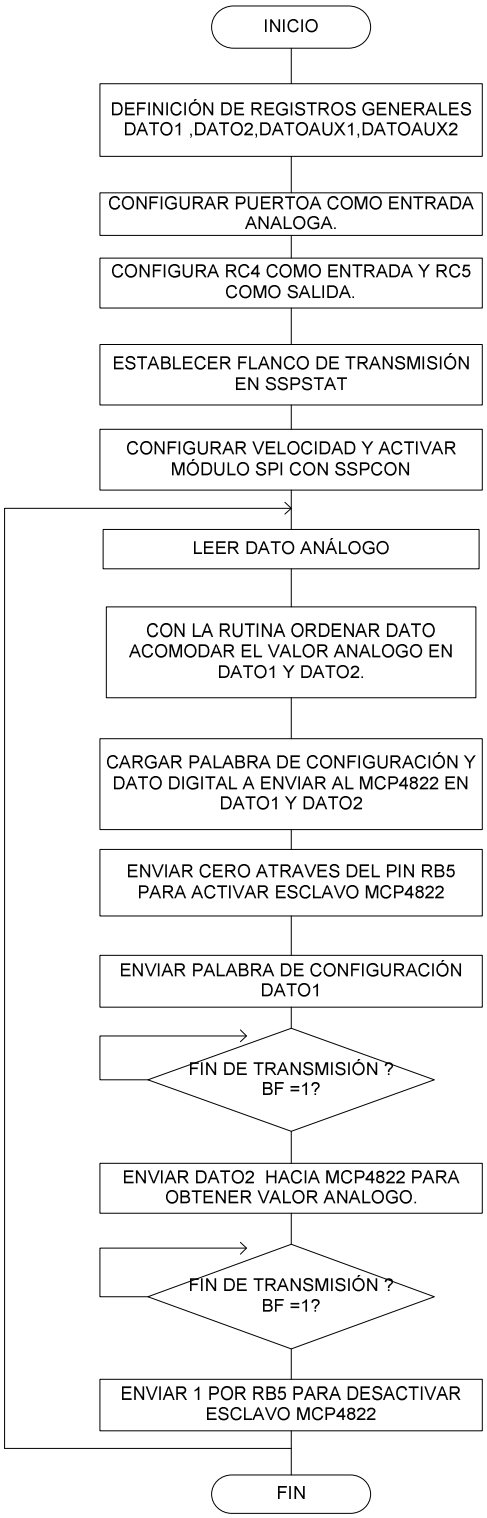


Fig.4.62 *Diagrama de bloques conversión digital /Análoga.*

```
        list P=PIC16F877A
        include "P16F877A.INC"

        dato1       equ 0x20
        dato2       equ 0x21
        datoaux1    equ 0x22
        datoaux2    equ 0x23
        Dly0        equ 0x24
        Dly1        equ 0x25
        dato        equ 0x26

                org 0
                clrf STATUS
                bsf STATUS,5                ;banco 1
                movlw b'00000010'   ;puerto A<0:5> son entradas análogas.
                movwf ADCON1        ;justificado a la izquierda usando ADRESH.
                movlw 0xFF
                movwf TRISA                 ;entradas análogas.
                clrf PORTB          ;salida para activar convertidor.
                movlw 0x10
                movwf TRISC                 ; RC4 como entrada , las otras como salidas.
                movlw 0x00
                movwf SSPSTAT       ;activa bit CKE con flanco positivo.
                bcf STATUS,5                ;banco 0
                movlw 0x31
                movwf SSPCON        ;activa bits SSPEN,CKP y Fosc/16
                clrf PORTB
                movlw b'10010000'   ;valor de programación 1001 ,con GA de 2x.
                movwf dato1         ;iniciar contador a cero.
                movlw b'00001111'   ;inicializar dato2 con cero.
                movwf dato2
                movlw b'00000001'   ;canal 4 AN4 o canal 0.
                movwf ADCON0        ;activamos registro para conversor A/D .Fosc/2
conversión      bcf PIR1,6          ;limpio bandera ADIF de dato convertido.
otro            bsf ADCON0,2                ;activamos bit GO de inicio de conversión.
                btfss PIR1,6
                goto $-1
                movf ADRESH,0
                movwf dato          ;valor análogo leído.
                movwf datoaux1
                movwf datoaux2
                swapf datoaux1,1
                swapf datoaux2,1
                movlw 0x0F
                andwf datoaux1,1
                movlw 0xF0
```

```
                andwf datoaux2,1
                movf datoaux1,0
                iorwf dato1,1
                movf datoaux2,0
                iorwf dato2,1
                bsf PORTB,5             ;desactivar convertidor MCP4822.
enviar          bcf PORTB,5             ;mandar uno "1"  para activar el MCP4822.
                movf dato1,0        ;colocar contador en W.
                movwf SSPBUF
                banksel SSPSTAT
lazo            btfss SSPSTAT,0     ;revisar bit BF.
                goto lazo
                bcf STATUS,5            ;banco 0
                movf dato2,0
                movwf SSPBUF
                banksel SSPSTAT
lazo1           btfss SSPSTAT,0     ;verificar bit BF.
                goto lazo1
                bcf STATUS,5            ;banco 0
                bsf PORTB,5             ;set bit SS
                clrf ADRESH         ;limpiar registro de dato análogo.
                bcf PIR1,6              ;limpiar bit de dato convertido.
                goto otro           ;leer otro dato.
                end                 ;fin del programa.
```

4.11 FALLAS COMUNES

Fallos Durante la utilización de la tarjeta debug y del hardware de aplicaciones:

No se reconoce dispositivo USB.
Verificar si se han instalado o se han dañado los drivers para utilizar el convertidor USB-SERIE FTD232BM.

Perdida de comunicación entre MPLAB y la tarjeta debug.
Desconectar tarjeta debug y volver a conectar al puerto USB y al software MPLAB.

No se reconoce dispositivo.
Verificar el cable de conexión entre tarjeta debug y módulo principal.
Verificar si el MCU de la tarjeta principal está energizado, verificar voltajes.
Verificar si el MCU utilizado está dañado, si es así cambiarlo.

Falla 083: el sistema no puede realizar debug.
Desconectar y conectar nuevamente al puerto USB.
Verificar el cable de conexión entre tarjeta debug y módulo principal.
Verificar si el MCU de la tarjeta principal está energizado.
Verificar si el cristal del módulo principal está funcionando.
Verificar si el MCU utilizado está dañado.
Reiniciar software MPLAB-IDE.

CONCLUSIONES GENERALES

- El sistema debug ha sido desarrollado para ser utilizado ya sea en desktop o laptop que soporten el software libre MPLAB-IDE de la compañía Microchip. Para las pruebas se utilizaron las versiones 6.5 y 8.02 (pero se puede utilizar otra versión) de MPLAB que fueron instalados en PCs con las características siguientes:
  Microsoft Windows XP, procesador 2.80GHz, 448MB de RAM y conector USB.

- La tarjeta debug es acoplable tanto a la tarjeta de aplicaciones presentada como a otro hardware que utilice otra familia de microcontroladores aceptada por el software MPLAB ,para detalles refiérase a la sección 4.1 del capitulo cuatro.

- El hardware de aplicaciones ha sido desarrollado para ser utilizado en conjunto con la tarjeta debug o para funcionar como hardware de pruebas con MCUs ya programados, está hecho de manera modular para obtener el mayor provecho al realizar diferentes combinaciones de los subsistemas.

- El módulo principal está desarrollado para trabajar con los módulos realizados: módulo para conexión I^2C/SPI, módulo para entrada de datos, módulo para salida de datos, módulo de potencia para control de un motor paso y cuatro relés.

- El manual de usuario ofrece los pasos necesarios para poder utilizar el software MPLAB en conjunto con la tarjeta debug y el hardware de aplicaciones, orientando al usuario a obtener el mayor provecho de aprendizaje basándose en los ejemplos detallados.

- El tutorial de ejercicios ha sido desarrollado acorde a los subsistemas propuestos y cubiertos por el hardware de aplicaciones, ofreciendo un ejemplo para cada subsistema.

- El hardware ha sido probado con cada uno de los ejemplos desarrollados y con cada uno de los ejercicios propuestos en el manual de usuario, verificando con esto el funcionamiento del equipo tanto en modo depurador como en modo hardware de pruebas.

- Los ejemplos propuestos han sido desarrollados con el fin que el usuario pueda familiarizarse con el entorno de MPLAB y con el hardware, y con esto implementar sus propios ejemplos apoyados en la herramienta de depuración.

- El convertidor análogo-digital se utilizó con una frecuencia de entrada mucho menor a la máxima permitida (Fmax=50KHz) en un rango de 0-10Vdc, obteniendo resultados aceptables.

- Para el uso del convertidor digital-análogo se determinó que la frecuencia de la señal de salida es determinada por el tiempo de respuesta del convertidor MCP4822 (10us >100KHz), se comprobó con una frecuencia mucho menor en un rango de 0-10V obteniendo resultados aceptables.

- La depuración de código que incluya interrupciones no es posible realizarlas, ya que el hardware debug solo depura códigos de bloques continuos de memoria, los errores serán de tipo lógicos si se utilizan interrupciones.

RECOMENDACIONES

- Para proteger la tarjeta debug no alimentar la tarjeta con voltajes mayores al indicado (15Vdc).

- Establecer bien como se utilizará el módulo principal, colocando el jumper en la respectiva posición, para usar el módulo en conjunto con la tarjeta debug (debug) o simplemente como hardware de prueba (hardware).

- Seleccionar en el módulo I²C/SPI con que tipo de comunicación se trabajará (I²C /SPI), así como en el módulo de potencia se debe especificar si se utilizará salida a motor paso o salida a relés.

- Para protección del sistema no debe excederse el voltaje de entrada al convertidor análogo-digital por encima de 10Vdc, ni demandandar una corriente mayor a 1A a la salida del convertidor digital-análogo, aunque el sistema está protegido se sugiere no arriesgar el equipo.

- Para mejor visualización del uso de los relés se recomienda colocar leds en cada uno de ellos que indiquen cuando cada dispositivo está encendido o pagado.

- Si se desea desarrollar otros módulos siguiendo la respectiva secuencia de pines el hardware queda abierto para realizarlo.

- Si se desea ahondar en el uso del software MPLAB se ofrecen tutoriales en la copia digital del documento u obtenerlos de la página principal de Microchip.

- No debe incluirse código con instrucciones para realizar interrupciones, el sistema operativo de la tarjeta debug no lo soporta y por ende ocurrirá un error lógico.

BIBLIOGRAFIA

- Microchip Technology Inc.December 1997.“PIC Micro Mid Range MCU Family reference Manual”. Sections 5,6,7,9,10,17,18,23.
  Available In:
  http://ww1.microchip.com/downloads/en/DeviceDoc/33023a.pdf

- Microchip Technology Inc.2001.“PIC16F87x/A Data Sheet 28/40-Pin 8-Bit CMOS FLASH Microcontrollers”. PDF Document.Sections 1 (pag.5),section 2 (pag.11),section 3(Pag.29),section 4 (pag.41),section 9 (pag.65),section 10(pag. 95) and section 11(pag.111).
  Available In :
  http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

- Microchip Technology Inc.2005.“12 Bit-DACs with internal Vref and SPI Interface” Section 1 (pag.1),section 3(Pag.14),section 4(pag.15),section 5(pag.17).
  Available In :
  http://ww1.microchip.com/downloads/en/DeviceDoc/21953a.pdf

- Microchip Technology Inc.2008.“1K 5.0V I²C serial EEPROM”. Section 1 (pag.2),section 2(Pag.5),section 3(pag.6),section 4(pag.6),section 5(pag.8),section 6(pag.9),section 8(pag.11).
  Available In :
  http://ww1.microchip.com/downloads/en/DeviceDoc/21201J.pdf

- Microchip Technology Inc.2003.“In-circuit serial programming ICSP guide”. Programming Specification to PIC16F8xx,Pag 181.
  Available In :
  http://ww1.microchip.com/downloads/en/DeviceDoc/30277d.pdf

  http://ww1.microchip.com/downloads/en/AppNotes/00656b.pdf

- Microchip Technology Inc.2001.“On-Chip Debbuger Specification”.
  Section 1.3 (pag.5),section 1.4(pag.6).
  Chapter 2 (pag.7)
  Available In :
  http://ww1.microchip.com/downloads/en/DeviceDoc/51242a.pdf

- Future technology devices International Ltd.2005.”FT232BM USB UART IC Data Sheet” .Version 1.8.
  Available in:
  http://www.ftdichip.com/Documents/DataSheets/DS_FT232BM.pdf

- manual transistor 2N2222/A.
  http://www.datasheetcatalog.org/datasheet/philips/2N2222.pdf

  http://www.datasheetcatalog.org/datasheet/SGSThomsonMicroelectronics/mXyzzyw.pdf

- Manual transistor TIP120/121/122.
  www.fairchildsemi.com/ds/TI%2FTIP122.pdf

- Manual del amplificador operacional LM358.
  http://www.datasheetcatalog.org/datasheet/nationalsemiconductor/DS007787.PDF

- ATE.Universidad de Oviedo.Fernando Nuño García.”Módulo SSP-Interface I²C”.
  Documento PDF.
  http://www.ate.uniovi.es/fernando/Doc2003/SED/SSP_I2C.pdf

- ATE.Universidad de Oviedo.Fernando Nuño García.”Módulo SSP_SPI”.
  Documento PDF.
  www.ate.**uniovi**.es/fernando/Doc2003/SED/**SSP_SPI**_BREVE.pdf

- ATE.Universidad de Oviedo.”El Entorno MPLAB”.
  Documento PDF.
  www.**uniovi**.es/ate/alberto/**Entorno**%20**MPLAB**_v7xx.pdf