DISEÑO, PROGRAMACIÓN E IMPLANTACIÓN DE UN SISTEMA DE CONTROL CON EL MICROCONTROLADOR PIC 16F84

INTRODUCCIÓN

En el siguiente proyecto de Microcontroladores PIC haremos una explicación teórica sobre su funcionamiento, programación y sus características principales, para así llegar al desarrollo del diseño y su implantación práctica. Para este proyecto se utilizara el

PIC 16F84 o en su defecto el PIC 16C84.

Ya que el " PIC 16F84 " es un MICROCONTROLADOR con memoria de programa tipo FLASH, lo que representa gran facilidad en el desarrollo de prototipos y en su aprendizaje ya que no se requiere de borrado con luz ultravioleta como las versiones EPROM sino, permite reprogramarlo nuevamente sin ser borrado con anterioridad. Por esta razón, lo usaremos en la mayoría de aplicaciones que se desarrollan a lo largo del estudio.

El PIC 16C84 es un microcontrolador de la familia MICROCHIP, totalmente compatible con el PIC 16F84. Su principal característica es que posee memoria "EEPROM" en lugar de memoria Flash, pero su manejo es igual. Con respecto al PIC16F84, este microcontrolador presenta dos diferencias:

- La memoria de datos tiene menor tamaño, aquí se tienen 32 registros de propósito general (el mapa de memoria de datos llega hasta 2Fh).
- En el momento de programar el microcontrolador, el fusible de selección del temporizador de arranque (Power Up Timer) trabaja de forma inversa, es decir, si en el PIC 16F84 se selecciona la opción "Low" para activarlo, en el PIC 16C84 se debe seleccionar "High".

Este microcontrolador ha sido reemplazado de forma gradual por el PIC 16F84, por lo tanto, los diseños que lo utilicen como elemento de control deben ser actualizados. Aunque, como se ve, es un proceso casi transparente.

Este microcontrolador se basa en la Arquitectura Harvard, en la cual el programa y los datos se pueden trabajar desde memorias separadas, lo que posibilita que las instrucciones y los datos posean longitudes diferentes. Esta misma estructura es la que permite la superposición de los ciclos de búsqueda y ejecución de las instrucciones, lo cual se ve reflejado en una mayor velocidad del microcontrolador.

MARCO TEÓRICO

− MEMORIA DE PROGRAMA

Es una memoria de 1 K byte de longitud con palabra de 14 bits. Como es del tipo FLASH se puede programar y borrar eléctricamente, en otras palabras, se puede programar o borrar sin necesidad de un borrador de luz ultravioleta, lo que facilita el desarrollo de programas y la experimentación. Como el PIC 16F84 tiene un contador de programa de 13 bits, tiene una capacidad de direccionamiento de 8K x 14, pero solamente tiene implementado el primer 1K x 14 (000h hasta 03FFh). Si se direccionan posiciones de memoria superiores a 3FFh se causará un solapamiento o desborde con el espacio del primer 1K.

− VECTOR DE RESET

Cuando ocurre un reset o se enciende el microcontrolador, el contador de programa se pone en ceros (000h).

Por esta razón, en la primera dirección del programa se debe escribir todo lo relacionado con la iniciación del mismo.

− VECTOR DE INTERRUPCION

Cuando el microcontrolador recibe una señal de interrupción el contador de programa apunta a la dirección 04h de la memoria de programa, por eso allí se debe escribir toda la programación necesaria para atender dicha interrupción.

− REGISTROS (Memoria RAM)

El PIC 16F84 puede direccionar 128 posiciones de memoria RAM, pero solamente tiene implementado físicamente los primeros 80 (0 a 4Fh). De estos los primeros 12 son registros que cumplen un propósito especial en el control del microcontrolador y los 68 siguientes son registros de uso general que se pueden usar para guardar los datos temporales de la tarea que se esta ejecutando. Los registros están organizados como dos bancos (paginas) de 128 posiciones de 8 bits cada una (128 x 8); todas las posiciones se pueden accesar directa o indirectamente (estas ultimas a través del registro FSR). Para seleccionar que pagina de registro se trabaja en un momento determinado se utiliza el bit RP0 del registro STATUS.

− PINES Y FUNCIONES

Los PUERTOS son el puente entre el microcontrolador y el mundo exterior. Son líneas digitales que trabajan entre cero y cinco voltios y se pueden configurar como entradas o como salidas.

El PIC 16F84 tiene dos puertos. El puerto A con 5 líneas y el puerto B con 8 líneas. Cada pin se puede configurar como entrada o como salida independiente programado por un par de registros diseñados para tal fin. En ese registro un "0" configura el pin del puerto correspondiente como salida y un "1" lo configura como entrada.

− PUERTO A

RA0 = Pin de Entrada/Salida (TTL).

RA1 = Pin de Entrada/Salida (TTL).

RA2 = Pin de Entrada/Salida (TTL).

RA3 = Pin de Entrada/Salida (TTL).

RA4/TOCKI = Pin de Entrada/Salida o entrada de Reloj Externo para el TMR0, cuando este pin se configura como salida es de tipo Open Drain (ST), cuando funciona como salida se debe conectar a Vcc (+5V) a través de una resistencia.

− PUERTO B

RB0/INT = Pin de Entrada/Salida o entrada de interrupción externa. (TTL/ST).

RB1 = Pin de Entrada/Salida (TTL).

RB2 = Pin de Entrada/Salida (TTL).

RB3 = Pin de Entrada/Salida (TTL).

RB4 = Pin de Entrada/Salida con Interrupción por cambio de Flanco (TTL).

RB5 = Pin de Entrada/Salida con Interrupción por cambio de Flanco (TTL).

RB6 = Pin de Entrada/Salida con Interrupción por cambio de Flanco (TTL/ST).

RB7 = Pin de Entrada/Salida con Interrupción por cambio de Flanco (TTL/ST).

− PINES ADICIONALES

MCLR = Pin de Reset del Microcontrolador (Master Clear). Se activa (el pic se resetea) cuando tiene un "0" lógico en su entrada.

Vss = Ground o Tierra

VDD = Fuente Positiva (+5V)

OSC2/CLKOUT = Entrada del Oscilador del Cristal. Se conecta al Cristal o Resonador en modo XT (Oscilador de Cristal). En modo RC (Resistencia−Condensador), este pin actúa como salida el cual tiene 1/4 de la frecuencia que entra por el pin OCS1/CLKIN.

OSC1/CLKIN = Entrada del Oscilador del Cristal / Entrada de reloj de una Fuente Externa.

El Puerto B tiene Internamente unas resistencias de pull−up conectadas a sus pines (sirven para fijar el pin a un nivel de cinco voltios), su uso puede ser habilitado o deshabilitado bajo control del programa. Todas las resistencias de pull−up conectan o desconectan a la vez. La resistencia de pull−up es desconectada automáticamente en un pin si este se programa como salida. El pin RB0/INT se puede configurar por software para que funcione como interrupción externa.

El pin RA4/TOCKI del puerto A puede ser configurado como un pin de entrada/salida como se mencionaba anteriormente o como entrada del temporizador/contador. Cuando este pin se programa como entrada digital, funciona como un disparador de Schmitt (Schmitt trigger, ST), esto quiere decir que puede reconocer señales un poco distorsionadas y llevarlas a niveles lógicos (cero y cinco voltios). Cuando se usa como salida digital se comporta como colector abierto, por lo tanto se debe poner una resistencia de pull−up (resistencia externa conectada a un nivel lógico de cinco voltios). Como salida, la lógica es inversa: un "0" escrito al pin del puerto entrega en el pin un "1" lógico. Además como salida no puede manejar cargas como fuente, sólo en el modo sumidero.

Como este dispositivo es de tecnología CMOS, todos los pines deben estar conectado a alguna parte, nunca dejarlos al aire por que se puede dañar el integrado. Los pines que no se estén usando se deben conectar la fuente de alimentación +5V con una resistencia de < 5 Kilo Ohmio.

La máxima capacidad de corriente de cada uno de los pines de los puertos en modo sumidero (sink) es de 25 mA y en modo fuente (source) es de 20 mA.

El consumo de corriente del microcontrolador para su funcionamiento depende del voltaje de operación, la frecuencia y de las cargas que tengan sus pines.

Por Ejemplo: Para un reloj de 4 MHz el consumo es de aproximadamente de 2mA; aunque este se puede reducir a 40 microamperios cuando está en el modo sleep (en este modo el micro se detiene y disminuye el consumo de potencia). Se sale de este estado cuando se produce alguna condición especial que veremos mas adelante.

- **PINES Y FUNCIONES (Figura)**

− EL OSCILADOR EXTERNO

Todo Microcontrolador requiere un circuito externo que le indique la velocidad a la que debe trabajar. Este circuito, que se conoce con el nombre de oscilador o reloj, es muy simple pero de vital importancia para el buen funcionamiento del sistema. El PIC 16F84 puede utilizar cuatro tipos de oscilador diferentes. Estos tipos son:

- RC. Oscilador con resistencia y condensador.
- XT. Cristal de cuarzo.
- HS. Cristal de alta velocidad.
- LP. Cristal para baja frecuencia y bajo consumo de potencia.

En el momento de programar o "quemar" el microcontrolador se debe especificar que tipo de oscilador se usa. Esto se hace a través de unos fusibles llamados "fusibles de configuración".

En la mayoría de las practicas que se realizan se sugiere el cristal de 4 MHz, por que garantiza una mayor precisión y un buen arranque del microcontrolador. Internamente esta frecuencia esta dividida por cuatro, lo que hace que la frecuencia efectiva de trabajo sea de 1 MHz, por lo que cada instrucción se realiza en un microsegundo (1 µS). El cristal debe ir acompañado de dos condensadores y se conecta como se muestra en la figura siguiente.

Dependiendo de la aplicación, se pueden utilizar cristales de otras frecuencias; por ejemplo se usa el cristal de 3.579545 MHz por que es muy económico, el de 32.768 KHz cuando se necesita crear bases de tiempo de un segundo muy precisas. El límite de velocidad de estos microcontroladores es de 10 MHz.

Si no se requiere mucha presición en el oscilador y se requiere economizar dinero, se puede utilizar una resistencia y un condensador, como se muestra a continuación.

Los valores recomendados para este tipo de oscilador son: 5 K**W** £ Rext £ 100 K**W** y Cext > 20 pF.

Nota: Cuando el oscilador del dispositivo esta en modo RC, no maneje el pin OSC1 con un reloj externo por que puede dañar el dispositivo.

La frecuencia del oscilador dividida por cuatro está disponible en el pin OSC2/CLKOUT, y puede ser usada para chequear propósitos o para sincronizar otra lógica.

- **RESET**

En los microcontroladores se requiere un pin de reset para reiniciar el funcionamiento del sistema cuando sea necesario, ya sea por una falla que se presente o por que así se halla diseñado el sistema. El pin de reset en los PIC es llamado "Master Clear". El PIC 16F84 admite diferentes tipos de reset:

- Al encendido (Power On Reset)
- Pulso en el pin Master Clear durante operación normal
- Pulso en el pin Master Clear durante el modo de bajo consumo (modo sleep)
- El rebase del conteo del circuito de vigilancia (watchdog) durante operación normal.
- El rebase del conteo del circuito de vigilancia (watchdog) durante el modo de bajo consumo (sleep)

El reset al encendido se consigue gracias a dos temporizadores. El primero de ellos es el OST (Oscillator Star−Up Timer: Temporizador de encendido del oscilador), orientado a mantener el microcontrolador en reset

hasta que el oscilador de cristal es estable. El segundo es el PWRT (Power−Up Timer: Temporizador de encendido), que provee un retardo fijo de 72 mS (nominal) en el encendido únicamente, diseñado para mantener el dispositivo en reset mientras la fuente se estabiliza. Para utilizar estos temporizadores, solo basta con conectar el pin Master Clear a la fuente de alimentación evitándose utilizar las tradicionales redes RC externas en el pin de reset.

− El reset por Master Clear se consigue llevando momentáneamente este pin a un estado lógico bajo, mientras que el watchdog WDT produce un reset cuando su temporizador rebasa la cuenta, o sea que pasa de 0FFh a 00H. Cuando se quiere tener control sobre el reset del sistema se puede conectar un botón como se muestra en la siguiente figura.

− Reset por Brown−Out: Un brown−out es una condición en donde la alimentación del dispositivo (Vdd) baja a un valor mínimo, pero no a cero y luego se normaliza. El dispositivo debe resetearse en caso de presentarse un brown−out. Para resetear un PIC 16F84 cuando un brown−out ocurre se debe construir un circuito de protección externo como el de la siguiente figura:

Circuito de Protección # 1.

Este circuito entrará en un reset activo cuando VDD baja por debajo de Vz + 0.7, en donde Vz = Voltaje del Zener.

Circuito de Protección # 2.

Este circuito es más económico, aunque menos eficaz. El transistor Q1 pasará a un estado de corte cuando VDD está por debajo de un cierto nivel tal que:VDD * (R1 / (R1 + R2)) = 0.7 V

- **REGISTROS (Memoria Ram)**

El PIC 16F84 puede direccionar 128 posiciones de memoria RAM, pero solamente tiene implementado físicamente los primeros 80 (0 a 4Fh). De estos los primeros 12 son registros que cumplen un propósito especial en el control del microcontrolador y los 68 siguientes son registros de uso general que se pueden usar para guardar los datos temporales de la tarea que se esta ejecutando. Los registros están organizados como dos bancos (paginas) de 128 posiciones de 8 bits cada una (128 x 8); todas las posiciones se pueden accesar directa o indirectamente (estas ultimas a través del registro FSR). Para seleccionar que pagina de registro se trabaja en un momento determinado se utiliza el bit RP0 del registro STATUS.

00h o INDO: Registro para el direccionamiento indirecto de datos. Este no es un registro disponible físicamente; utiliza el contenido del FSR y el bit RP0 del registro STATUS para seleccionar indirectamente la memoria de datos o RAM del usuario; la instrucción determinara que se debe señalar con el registro señalado.

01h o TMR0: Temporizador/contador de 8 bits. Este se puede incrementar con una señal externa aplicada al pin RA4/TOCKI o de a cuerdo a una señal interna proveniente del reloj de instrucciones del microcontrolador. La rata o tasa de incremento del registro se puede determinar por medio de un preescalador, localizado en el registro OPTION. Los anteriores microcontroladores no contaban con la generación de una interrupción cuando se rebasaba la cuenta (el paso de 0FFh a 00h).

02h o PCL: CONTADOR DE PROGRAMA. Se utiliza para direccionar las palabras de 14 bits del programa del usuario que se encuentra almacenado en la memoria ROM; este contador tiene un tamaño de 13 bits. Sobre el byte bajo, se puede escribir o leer a voluntad directamente, mientras que en el byte alto, no. El byte alto se maneja mediante el registro PCLATH (0Ah). A diferencia de los PIC de primera generación el 16F84 ante una condición de reset inicia el contador de programa con todos sus bits en "cero". Durante la ejecución normal del programa, y dado que todas las instrucciones ocupan solo una posición de memoria, el contador se

incrementa con cada instrucción, a menos que se trate de alguna instrucción de salto.

03h o STATUS: REGISTRO DE ESTADO. Contiene el estado Aritmético de la ALU, la causa de reset y los bits de preselección de pagina para la memoria de datos. En la figura se muestran los bits correspondientes a este registro. Los bits 5 y 6 (RP0 y RP1) son los bits de selección de pagina (Bank 0 y Bank 1), para el direccionamiento directo de la memoria de datos; solamente RP0 se usa en los PIC 16F84. RP1 se puede utilizar como un bit de propósito general de lectura/escritura. Los bits TO y PD no se pueden modificar por un proceso de escritura; ellos muestran la condición por la cual se ocasiono el ultimo reset.

IRP bit 7	RP1 bit 6	RP0 bit 5	T0 bit 4	PD bit 3	Z bit 2	DC bit 1	C bit 0

IRP	Selector de página para direccionamiento indirecto. Este bit no se utiliza efectivamente en el PIC 16F84, por lo que se puede utilizar como un bit de propósito general.
RP1,0	Selectores de página para un seleccionamiento directo. Solamente RP0 se utiliza en el PIC 16F84. RP1 se puede utilizar como un bit de propósito general.
T0	Time Out o bit de finalización del temporizador. Se coloca en 0 cuando el circuito de vigilancia Watchdog finaliza la temporización
PD	Power Down o bit de bajo consumo. Se coloca en 0 por la instrucción sleep.
Z	Zero o bit de cero. Se coloca en 1 cuando el resultado de una operación aritmética o lógica es cero.
DC	Digit Carry o bit de acarreo de dígito. En operaciones aritméticas se activa cuando hay un acarreo entre el bit 3 y 4, es decir cuando hay acarreo entre el nible de menor y de mayor peso.
C	Carry o bit de acarreo. En instrucciones aritméticas se activa cuando se presenta acarreo desde el bit más significativo del resultado.

04h o FSR: REGISTRO SELECTOR DE REGISTROS. En asocio con el registro IND0, se utiliza para seleccionar indirectamente los otros registros disponibles. Mientras que los antecesores del PIC 16F84 solo poseían 5 bits activos, en este microcontrolador se poseen solo 8 bits. Si en el programa no se utilizan llamadas indirectas, este registro se puede utilizar como un registro de propósito general.

05h o PORTA: PUERTO DE ENTRADA/SALIDA DE 5 BITS (RA0~ RA4). Este puerto al igual que todos sus similares en los PIC, puede leerse o escribirse como si se tratara de un registro cualquiera. El registro que controla el sentido (entrada o salida) de los pines de este puerto esta localizado en la pagina 1 (Banco 1), en la posición 85h y se llama TRISA.

06h o PORTB: PUERTO DE ENTRADA/SALIDA DE 8 BITS (RB0~RB7). Al igual que en todos los PIC, este puede leerse o escribirse como si se tratara de un registro cualquiera; algunos de sus pines tienen funciones alternas en la generación de interrupciones. El registro de control para la configuración de la función de sus pines se localiza en la pagina 1 (Banco 1), en la dirección 86h y se llama TRISB.

08h o EEDATA: REGISTRO DE DATOS DE LA EEPROM. Este registro contiene el dato que se va a escribir en la memoria EEPROM de datos o el que se leyó de ésta.

09h o EEADR: REGISTRO DE DIRECCION DE LA EEPROM. Aquí se mantiene la dirección de la EEPROM de datos que se van a trabajar, bien sea para una operación de lectura o para una de escritura

0Ah o PCLATH: REGISTRO PARA LA PARTE ALTA DE LA DIRECCION. Este registro contiene la parte alta del contador de programa y no se puede acceder directamente.

0Bh o INTCON: REGISTRO PARA EL CONTROL DE INTERRUPCIONES. Es el encargado del manejo de las interrupciones y contiene los bits que se muestran en la figura.

GIE Bit 7	EEIE bit 6	TOIE bit 5	INTE bit 4	RBIE bit 3	TOIF bit 2	INTF bit 1	RBIF bit 0
GIE	Global Interrup Enable o Habilitador general de interrupciones. 0: Deshabilita todas las interrupciones 1: Habilita las interrupciones						
EEIE	EEPROM Write Interrup Enable o Habilitación de interrupción por escritura de la EEPROM. 0: La deshabilita 1: La habilita						
T0IE	TMR0 Interrup Enable o Habilitación de interrupción del temporizador TMR0. 0: La deshabilita 1: La habilita						
INTE	INT Interrup Enable o Habilitación de la interrupción INT. 0: La deshabilita 1: La habilita						
RBIE	RBIF Interrup Enable o Habilitación de la interrupción RBIF. 0: La deshabilita 1: La habilita						
TOIF	TMR0 Overflow Interrup Flag o Bandera de la interrupción por desbordamiento del TMR0. Se coloca en 1 cuando el TMR0 pasa de 0FFh a 00h; ésta debe ser puesta a 0 por programa.						
INTF	INT Interrup Flag o Bandera de interrupción INT. Se coloca en 1 cuando la interrupción INT ocurre; ésta debe ser puesta cero por programa.						
RBIF	RB Port Change Interrup Flag o Bandera de interrupción por cambio en el puerto B. Se coloca en 1 cuando una de las entradas (RB4 a RB7) cambia; ésta debe ser puesta a 0 por programa						

81h u OPTION: REGISTRO DE CONFIGURACION MULTIPLE. Posee varios bits para configurar el preescalador, la interrupción externa, el timer y las características del Puerto B. Los bits que contiene y las funciones que realiza este registro se muestran en la figura. El preescalador es compartido entre el TMR0 y el WDT; su asignación es mutuamente excluyente ya que solamente puede uno de ellos ser preescalado a la vez.

RBPU bit 7	INTEDG bit 6	GRTS bit 5	RTE bit 4	PSA bit 3	PS2 bit 2	PS1 bit 1	PS0 bit 0

<table>
<tr><td>RBPU</td><td>Portb Pull−up Enable o Habilitación de pull−up del puerto B.

0: Habilita las pull−ups internas

1: Las deshabilita</td></tr>
<tr><td>INTEDG</td><td>INT Interrup Edge Select o Selector de flanco de la interrupción INT.

0: Flanco de bajada

1: Flanco de subida</td></tr>
<tr><td>RTS</td><td>TMR0 Signal Source o Fuente de señal del TMR0.

0: Ciclo de instrucciones interno (Temporizador)

1: Transición en el pin RA4/TOCKI (Contador)</td></tr>
<tr><td>RTE</td><td>TMR0 Signal Edge o Flanco de la señal del TMR0

0: Incremento de transición de bajo a alto

1: Incremento en transición</td></tr>
<tr><td>PSA</td><td>Preescaler Assignment o Asignación del preescalador

0: TMR0 (Contador / Temporizador)

1: WDT (Circuito de vigilancia)</td></tr>
<tr><td>PS2, 1, 0</td><td>Preescaler Value o Valores del preescalador.

<table><tr><th>Valor</th><th>TMR0</th><th>WDT</th></tr><tr><td>000</td><td>1:2</td><td>1:1</td></tr><tr><td>001</td><td>1:4</td><td>1:2</td></tr><tr><td>010</td><td>1:8</td><td>1:4</td></tr><tr><td>011</td><td>1:16</td><td>1:8</td></tr><tr><td>100</td><td>1:32</td><td>1:16</td></tr><tr><td>101</td><td>1:64</td><td>1:32</td></tr><tr><td>110</td><td>1:128</td><td>1:64</td></tr><tr><td>111</td><td>1:256</td><td>1:128</td></tr></table></td></tr>
</table>

85h o TRISA: REGISTRO DE CONFIGURACION DEL PUERTO A. Es el registro de control para el puerto A. Un "cero" en el bit correspondiente al pin lo configura como salida, mientras que un "uno" lo hace como entrada.

86h o TRISB: REGISTRO DE CONFIGURACION DEL PUERTO B. Es el registro de control para el puerto B. Un "cero" en el bit correspondiente al pin lo configura como salida, mientras que un "uno" lo hace como entrada.

88h o EECON1: REGISTRO DE PARA EL CONTROL DE LA MEMORIA EEPROM DE DATOS. Este es el registro de control para la memoria de datos y solo destina cinco bits para ello, los más bajos; los tres bits superiores permanecen sin implementar. En la figura se muestran las funciones de estos bits.

U bit 7	U bit 6	U bit 5	EEIF bit 4	WRERR bit 3	WREN bit 2	WR bit 1	RD bit 0

U	Unimplemented. No implementados.
EEIF	EEPROM Write Completion Interrup Flag o Bandera de finalización de la escritura. Se coloca en "1" cuando finaliza con éxito la escritura de la EEPROM de datos; se debe colocar en "0" por programa. El bit de habilitación correspondiente es el EEIE, localizado en el registro INTCON.
WRERR	Write Error Flag o Bandera de error de escritura. Si se coloca en "1" cuando la operación de escritura termina prematuramente, debido a cualquier condición de reset.
WREN	Write Enable o habilitación de escritura. Si se coloca en "0" no permite las operaciones de escritura; en "1" las habilita.
WR	Write Control o Control de escritura. Al colocarse en "1" inicia un ciclo de escritura. Este bit sólo es puesto a "0" por hardware, una vez la escritura termina.
RD	Read Control o Control de lectura. Al colocarse en "1" se inicia una lectura de la EEPROM de datos, la cual toma un ciclo de reloj de instrucciones. Este bit sólo se limpia (colocar en "0") por hardware, al finalizar la lectura de la posición de la EEPROM.

89h o EECON2: REGISTRO AUXILIAR PARA EL CONTROL DE LA MEMORIA EEPROM DE DATOS. Este registro no es implementado físicamente por el microcontrolador, pero que es necesario en las operaciones de escritura en la EEPROM de datos; ante cualquier intento de lectura se tendrán "ceros".

0Ch a 4Fh: REGISTRO DE PROPOSITO GENERAL. Estas 68 posiciones están implementadas en la memoria RAM estática, la cual conforma el área de trabajo del usuario; a ellas también se accede cuando en la pagina 1 (Banco 1) se direccionan las posiciones 8Ch a CFh. Esto se ha diseñado así para evitar un excesivo cambio de paginas en el manejo de la RAM del usuario, agilizando los procesos que se estén llevando a cabo y descomplicando la labor del programador.

REGISTRO DE TRABAJO W. Este es el registro de trabajo principal, se comporta de manera similar al acumulador en los microprocesadores. Este registro participa en casi todo el programa y por consiguiente en la mayoría de las instrucciones.

PILA (STACK). Estos registros no forman parte de ningún banco de memoria y no permiten el acceso por parte del usuario. Se usan para guardar el valor del contador de programa cuando se hace un llamado a una subrutina (CALL), o cuando se atiende una interrupción; luego, cuando el micro regresa a seguir ejecutando su tarea normal, el contador de programa recupera su valor leyéndolo nuevamente desde la pila. El PIC 16F84 tiene una pila de 8 niveles, esto significa que se pueden anidar 8 llamados a subrutina sin tener problema alguno.

- **CARACTERÍSTICAS ESPECIALES**

Algunos elementos que forman parte de los PIC no se encuentran en microcontroladores de otros fabricantes, o simplemente representan alguna ventaja o facilidad a la hora de hacer un diseño. A continuación una corta descripción de las más significativas.

− CIRCUITO DE VIGILANCIA (Watchdog Timer)

Su función es restablecer el programa cuando éste se ha perdido por fallas en la programación o por alguna razón externa. Es muy útil cuando se trabaja en ambientes con mucha interferencia o ruido electromagnético. Esta conformado por un oscilador RC que se encuentra dentro del microprocesador. Este oscilador corre de manera independiente al oscilador principal. Cuando se habilita su funcionamiento, dicho circuito hace que el microcontrolador sufra un reset cada determinado tiempo (que se puede programar entre 18 mS y 2 segundos). Este reset lo puede evitar el usuario mediante una instrucción especial del microcontrolador (CLRWT: Borra el contenido del watchdog), la cual se debe ejecutar antes de que termine el periodo nominal de dicho temporizador. De esta manera si el programa se ha salido de su flujo normal, por algún ruido o interferencia externa, el sistema se reiniciará (cuando se acabe el tiempo programado y no se haya borrado el contador) y el programa puede restablecerse para continuar con su funcionamiento normal.

− TEMPORIZADOR DE ENCENDIDO (Power−up Timer)

Este proporciona un reset al microcontrolador en el momento de conectar la fuente de alimentacion, lo que garantiza un arranque correcto del sistema. En el momento de grabar el microcontrolador se debe habilitar el fusible de configuración "Power−up Timer", para ello se debe seleccionar "ON". Su tiempo de retardo es de 72 milisegundos.

− MODO DE BAJO CONSUMO (Sleep)

Esta característica permite que el microcontrolador entre en un estado pasivo donde consume muy poca potencia. Cuando se entra en este modo el oscilador principal se detiene, pero el temporizador del circuito de vigilancia (watchdog) se reinicia y empieza su conteo nuevamente. Se entra en ese estado por la ejecución de una instrucción especial (llamada SLEEP) y se sale de él cuando el microcontrolador sufre un reset por un pulso en el pin MCLR, por que el watchdog hace que se reinicie el sistema o por que ocurre una interrupción al sistema.

− INTERRUPCIONES

Este microcontrolador incluye el manejo de interrupciones, lo cual representa grandes ventajas. El PIC16F84 posee cuatro formas de interrupción que son:

- Interrupción externa en el pin RB0/INT
- Finalización del temporizador/contador TMR0
- Finalización de escritura en la EEPROM de datos
- Cambio de estado en los pines RB4 a RB7

El registro 0Bh o INTCON contiene las banderas de las interrupciones INT, cambio en el puerto B y finalización del conteo del TMR0, al igual que el control para habilitar o deshabilitar cada una de las fuentes de interrupción, incluida la de escritura de la memoria EEPROM. Sólo la bandera de finalización de la escritura reside en el registro 88h o EECON1.

Si el bit GIE (Global Interrup Enable) se coloca en 0, deshabilita todas las interrupciones. Cuando una interrupción es atendida, el bit GIE se coloca en 0 automáticamente pare evitar interferencias con otras interrupciones que se pudieran presentar, la dirección de retorno se coloca en la pila y el PIC se carga con la dirección 04h. Una vez en la rutina de servicio, la fuente de interrupción se puede determinar examinando las banderas de interrupción. La bandera respectiva se debe colocar, por software, en cero antes de regresar de la interrupción, para evitar que se vuelva a detectar nuevamente la misma interrupción. La instrucción RETFIE permite al usuario retornar de la interrupción, a la vez que habilita de nuevo las interrupciones, al colocar el bit GIE en uno. Debe tenerse presente que solamente el contador de programa es puesto en la pila al atenderse la interrupción; por lo tanto, es conveniente que el programador tenga cuidado con el registro de estados y el de trabajo, ya que se pueden introducir resultados inesperados si dentro de ella se modifican.

Interrupción Externa. Actúa sobre el pin RB0/INT y se puede configurar para activarse con el flanco de subida o el de bajada, de acuerdo al bit INTEDG (Interrup Edge Select Bit, localizado en el registro OPTION). Cuando se presenta un flanco valido en el pin INT, la bandera INTF (INTCON) se coloca en uno. La interrupción se puede deshabilitar colocando el bit de control INTE (INTCON) en cero. Cuando se atiende la interrupción, a través de la rutina de servicio, INTF se debe colocar en cero antes de regresar al programa principal. La interrupción puede reactivar al microcontrolador después de la instrucción SLEEP, si previamente el bit INTE fue habilitado

Interrupción por finalización de la temporización. La superación del conteo máximo (0FFh) en el TMR0 colocara el bit TOIF (INTCON) en uno. El bit de control respectivo es TOIE (INTCON).

Interrupción por cambio en el puerto RB. Un cambio en los pines del puerto B (RB4 a RB7) colocará en uno el bit RBIF (INTCON). El bit de control respectivo es RBIE (INTCON).

Interrupción por finalización de escritura. Cuando la escritura de un dato en la EEPROM finaliza, se coloca en 1 el bit EEIF (EECON1). El bit de control respectivo es EEIE (INTCON).

LÓGICA DE INTERRUPCIONES PARA EL PIC 16FXX

− MEMORIA DE DATOS DE LA EEPROM

El PIC 16F84 tiene una memoria EEPROM de datos de 64 posiciones (00h a 3Fh), de 8 bits cada una. Este bloque de memoria no se encuentra mapeado en ningún banco, el acceso a esas posiciones se consigue a través de dos registros de la RAM:

− El registro EEADR (posición 09), que debe contener la dirección de la posición de la EEPROM a ser accesada.

− El registro EEDATA (posición 08), que contiene el dato de 8 bits que se va a escribir o el que se obtuvo de la ultima lectura.

Adicionalmente, existen dos registros de control: el EECON1 (88h), que posee cinco bits que manejan las operaciones de lectura/escritura y el EECON2 (89h), que aunque no es un registro físico, es necesario para realizar las operaciones de escritura.

La lectura toma un ciclo de reloj de instrucciones, mientras que las escritura, por ser controlada por un temporizador incorporado, tiene un tiempo nominal de 10 milisegundos, este tiempo puede variar con la temperatura y el voltaje. Cuando se va a realizar una operación de escritura, automáticamente se hace primero la operación de borrado. El numero típico de ciclos de borrado/escritura de la EEPROM de datos es de 1.000.000.

− FUSIBLES DE CONFIGURACION

El PIC 16F84 posee cinco fusibles, cada uno de los cuales es un bit. Estos fusibles se pueden programas para seleccionar varias configuraciones del dispositivo: tipo de oscilador, protección de código, habilitación del circuito de vigilancia y el temporizador al encendido. Los bits se localizan en la posición de memoria 2007h, posición a la cual el usuario sólo tiene acceso durante la programación del microcontrolador. Cuando se programa la protección del código, el contenido de cada posición de la memoria no se puede leer completamente, de tal manera que el código del programa no se puede reconstruir. Adicionalmente, todas las posiciones de memoria del programa se protegen contra la reprogramación.

Una vez protegido el código, el fusible de protección solo puede ser borrado

(puesto a 1) si se borra toda la memoria del programa y la de datos.

− LAS PULL−UP INTERNAS

Cada uno de los pines del puerto B tiene un elemento débil pull−up interno (250 uA típico); este elemento es automáticamente desconectado cuando el pin se configura como salida. Adicionalmente, el bit RBPU (OPTION) controla todos estos elementos, los cuales están deshabilitados frente a una condición de reset. Estos elementos pull−up son especialmente útiles cuando el microcontrolador va a colocarse en el modo de bajo consumo, ya que ayudan a no tener las entradas flotantes, significado una reducción en el consumo de corriente.

− LA ARQUITECTURA

− LA ARQUITECTURA TRADICIONAL

La arquitectura tradicional de computadoras y microprocesadores se basa en el esquema propuesto por John Von Neumann, en el cual la unidad central de proceso, o CPU, esta conectada a una memoria única que contiene las instrucciones del programa y los datos (ver figura). El tamaño de la unidad de datos o instrucciones esta fijado por el ancho del bus de la memoria. Es decir que un microprocesador de 8 bits, que tiene además un bus de 8 bits que lo conecta con la memoria, deberá manejar datos e instrucciones de una o más unidades de 8 bits (bytes) de longitud. Cuando deba acceder a una instrucción o dato de más de un byte de longitud, deberá realizar más de un acceso a la memoria. Por otro lado este bus único limita la velocidad de operación del microprocesador, ya que no se puede buscar de memoria una nueva instrucción, antes de que finalicen las transferencias de datos que pudieran resultar de la instrucción anterior. Es decir que las dos principales limitaciones de esta arquitectura tradicional son :

- Que la longitud de las instrucciones esta limitada por la unidad de longitud de los datos, por lo tanto el microprocesador debe hacer varios accesos a memoria para buscar instrucciones complejas.
- Que la velocidad de operación (o ancho de banda de operación) esta limitada por el efecto de cuello de botella que significa un bus único para datos e instrucciones que impide superponer ambos tiempos de acceso.

La arquitectura von Neumann permite el diseño de programas con código automodificable, práctica bastante usada en las antiguas computadoras que solo tenían acumulador y pocos modos de direccionamiento, pero innecesaria, en las computadoras modernas.

Arquitectura von Neumann

− La arquitectura Harvard y sus ventajas

La arquitectura conocida como Harvard, consiste simplemente en un esquema en el que el CPU esta conectado a dos memorias por intermedio de dos buses separados. Una de las memorias contiene solamente las instrucciones del programa, y es llamada Memoria de Programa. La otra memoria solo almacena los datos y es llamada Memoria de Datos. Ambos buses son totalmente independientes y pueden ser de distintos anchos. Para un procesador de Set de Instrucciones Reducido, o RISC (Reduced Instrucción Set Computer), el set de instrucciones y el bus de la memoria de programa pueden diseñarse de manera tal que todas las instrucciones tengan una sola posición de memoria de programa de longitud. Además, como los buses son independientes, el CPU puede estar accediendo a los datos para completar la ejecución de una instrucción, y al mismo tiempo estar leyendo la próxima instrucción a ejecutar. Se puede observar claramente que las principales ventajas de esta arquitectura son:

Que el tamaño de las instrucciones no esta relacionado con el de los datos, y por lo tanto puede ser optimizado

para que cualquier instrucción ocupe una sola posición de memoria de programa, logrando así mayor velocidad y menor longitud de programa.

- Que el tiempo de acceso a las instrucciones puede superponerse con el de los datos, logrando una mayor velocidad de operación.

Una pequeña desventaja de los procesadores con arquitectura Harvard, es que deben poseer instrucciones especiales para acceder a tablas de valores constantes que pueda ser necesario incluir en los programas, ya que estas tablas se encontraran físicamente en la memoria de programa (por ejemplo en la EPROM de un microprocesador).

Arquitectura Harvard

Los microcontroladores PIC 16C5X, 16CXX y 17CXX poseen arquitectura Harvard, con una memoria de datos de 8 bits, y una memoria de programa que, según el modelo, puede ser de 12 bits para los 16C5X, 14 bits para los 16CXX y 16 bits para los 17CXX.

- **DIAGRAMA DE BLOQUE**

Diagrama de bloque del pic 16fxx

- **DIDERENCIA DE LOS TRADICIONALES MICROPROCESADORES Y LOS**

MICROCONTROLADORES PIC

La figura siguiente se representa un diagrama simplificado de la arquitectura interna del camino de los datos en el CPU de los microcontroladores PIC y los tradicionales microprocesadores. Este diagrama puede no representar con exactitud el circuito interno de estos microcontroladores, pero es exacto y claro desde la óptica del programador. La figura representa el mismo diagrama para un microprocesador ficticio de arquitectura tradicional. Se puede observar que la principal diferencia entre ambos radica en la ubicación del registro de trabajo, que para los PIC's se denomina W (Working Register), y para los tradicionales es el Acumulador(A).

En los microcontroladores tradicionales todas las operaciones se realizan sobre el acumulador. La salida del acumulador esta conectada a una de las entradas de la Unidad Aritmética y Lógica (ALU), y por lo tanto éste es siempre uno de los dos operandos de cualquier instrucción. Por convención, las instrucciones de simple operando (borrar, incrementar, decrementar, complementar), actúan sobre el acumulador. La salida de la ALU va solamente a la entrada del acumulador, por lo tanto el resultado de cualquier operación siempre quedara en este registro. Para operar sobre un dato de memoria, luego realizar la operación siempre hay que mover el acumulador a la memoria con una instrucción adicional.

En los microcontroladores PIC, la salida de la ALU va al registro W y también a la memoria de datos, por lo tanto el resultado puede guardarse en cualquiera de los dos destinos. En las instrucciones de doble operando, uno de los dos datos siempre debe estar en el registro W, como ocurría en el modelo tradicional con el acumulador. En las instrucciones de simple operando el dato en este caso se toma de la memoria (también por convención). La gran ventaja de esta arquitectura es que permite un gran ahorro de instrucciones ya que el resultado de cualquier instrucción que opere con la memoria, ya sea de simple o doble operando, puede dejarse en la misma posición de memoria o en el registro W, según se seleccione con un bit de la misma instrucción. Las operaciones con constantes provenientes de la memoria de programa (literales) se realizan solo sobre el registro W.

En la memoria de datos de los PIC's se encuentran ubicados casi todos los registros de control del

microprocesador y sus periféricos autocontenidos, y también las posiciones de memoria de usos generales. En el caso de los 16CXX, algunos registros especiales de solo escritura (TRIS y OPTION) no están accesibles dentro del bloque de memoria de datos, sino que solo se pueden cargar desde el registro W por medio de instrucciones especiales.

Contador de Programa

Este registro, normalmente denominado PC, es totalmente equivalente al de todos los microprocesadores y contiene la dirección de la próxima instrucción a ejecutar. Se incrementa automáticamente al ejecutar cada instrucción, de manera que la secuencia natural de ejecución del programa es lineal, una instrucción después de la otra. Algunas instrucciones que llamaremos de control, cambian el contenido del PC alterando la secuencia lineal de ejecución. Dentro de estas instrucciones se encuentran el GOTO y el CALL que permiten cargar en forma directa un valor constante en el PC haciendo que el programa salte a cualquier posición de la memoria. Otras instrucciones de control son los SKIP o salteos condicionales, que producen un incremento adicional del PC si se cumple una condición especifica, haciendo que el programa saltee, sin ejecutar, la instrucción siguiente.

Al resetearse el microprocesador, todos los bits del PC toman valor 1, de manera que la dirección de arranque del programa es siempre la ultima posición de memoria de programa. En esta posición se deberá poner una instrucción de salto al punto donde verdaderamente se inicia el programa.

A diferencia de la mayoría de los microprocesadores convencionales, el PC es también accesible al programador como registro de memoria interna de datos, en la posición de 02. Es decir que cualquier instrucción común que opere sobre registros puede ser utilizada para alterar el PC y desviar la ejecución del programa. El uso indiscriminado de este tipo de instrucciones complica el programa y puede ser muy peligroso, ya que puede producir comportamientos difíciles de predecir. Sin embargo, algunas de esta instrucciones utilizadas con cierto método, pueden ser muy útiles para implementar poderosas estructuras de control tales como el goto computado. Como el microprocesador opera con datos de 8 bits, y la memoria de datos es también de 8 bits, estas instrucciones solo pueden leer o modificar los bits 0 a 7 del PC.

Stack
En los microcontroladores PIC el stack es una memoria interna dedicada, de tamaño limitado, separada de las memorias de datos y de programa, inaccesible al programador, y organizada en forma de pila, que es utilizada solamente, y en forma automática, para guardar las direcciones de retorno de subrrutinas e interrupciones. Cada posición es de 11 bits y permite guardar una copia completa del PC. Como en toda memoria tipo pila, los datos son accedidos de manera tal que el primero que entra es el ultimo que sale.

En los 16C5X el stack es de solo dos posiciones, mientras que en los 16XXX es de 8 posiciones y en los 17CXX es de 16 posiciones. Esto representa, en cierta medida, una limitación de estos microcontroladores, ya que no permite hacer uso intensivo del anidamiento de subrutinas. En los 16C5X, solo se pueden anidar dos niveles de subrutinas, es decir que una subrutina que es llamada desde el programa principal, puede a su vez llamar a otra subrutina, pero esta ultima no puede llamar a una tercera, porque se desborda la capacidad del stack, que solo puede almacenar dos direcciones de retorno. Esto de hecho representa una traba para el programador y además parece impedir o dificultar la programación estructurada, sin embargo es una buena solución de compromiso ya que estos microcontroladores están diseñados para aplicaciones de alta velocidad en tiempo real, en las que el overhead (demoras adicionales) que ocasiona un excesivo anidamiento de subrutinas es inaceptable. Por otra parte existen técnicas de organización del programa que permiten mantener la claridad de la programación estructurada, sin necesidad de utilizar tantas subrutinas anidadas.

Como ya se menciono anteriormente, el stack y el puntero interno que lo direcciona, son invisibles para el programador, solo se los accede automáticamente para guardar o rescatar las direcciones de programa cuando se ejecutan las instrucciones de llamada o retorno de subrutinas, o cuando se produce una interrupción o se

ejecuta una instrucción de retorno de ella.

− Palabra de Estado del Procesador
La palabra de estado del procesador contiene los tres bits de estado de la ALU (C, DC y Z), y otros bits que por comodidad se incluyeron en este registro.

7 6 5 4 3 2 1 0

REGISTRO STATUS

El bit Z indica que el resultado de la ultima operación fue CERO. El bit C indica acarreo del bit más significativo (bit 7) del resultado de la ultima operación de suma. En el caso de la resta se comporta comporta a la inversa, C resulta 1 si no hubo pedido de préstamo. El bit DC (digit carry) indica acarreo del cuarto bit (bit 3) del resultado de la ultima operación de suma o resta, con un comportamiento análogo al del bit C, y es útil para operar en BCD (para sumar o restar números en código BCD empaquetado). El bit C es usado además en las operaciones de rotación derecha o izquierda como un paso intermedio entre el bit 0 y el bit 7.

El bit PD (POWER DOWN) sirve para detectar si la alimentación fue apagada y encendida nuevamente, tiene que ver con la secuencia de inicialización, el watch dog timer y la instrucción sleep, y su uso se detallara en la sección referida al modo POWER DOWN. El bit TO (TIME−OUT) sirve para detectar si una condición de reset fue producida por el watch dog timer, esta relacionado con los mismos elementos que el bit anterior y su uso se detallara en la sección referida al WATCH DOG TIMER. Los bits de selección de pagina PA0/PA1/PA2 se utilizan en las instrucciones de salto GOTO y CALL, y se explicaran con detalle en la sección referida a las instrucciones de control, y a la organización de la memoria de programa. En realidad en el 16C54 estos bits no se usan y sirven para propósitos generales. En el 16C57 el PA0 si se usa pero los otros dos no. En el 16C55 se utilizan PA0 y PA1. PA2 esta reservado para uso futuro y en cualquiera de los PIC 16C5X sirve para propósitos generales.

OTROS REGISTROS ESPECIALES

Las ocho primeras posiciones del área de datos están reservadas para alojar registros de propósito especial, quedando las restantes libres para contener los datos u operandos que se desee (registros de propósito general).

El registro INDF que ocupa la posición 0 no está implementando físicamente y, como se ha explicado, se le referencia en el direccionamiento indirecto de datos aunque se utiliza el contenido de FSR.

En la dirección esta el registro TAR0 (Temporizador) que puede ser leído y escrito como cualquier otro registro. Puede incrementar su valor con una señal externa aplicada al pin T0CKI o mediante el oscilador interno.

El PC ocupa la posición 2 del área de datos en donde se halla el registro PCL al que se añaden 3 bits auxiliares y se conectan con los dos niveles de la Pila en las instrucciones CALL y RETLW.

El registro de Estado ocupa la posición 3 y entre sus bits se encuentran los señalizadores C, DC y Z y los bits PA1 y PA0 que seleccionan la página en la memoria de programa. El bit 7 (PA2) no está implementando en los PIC de la gama baja.

FRS se ubica en la dirección 4 y puede usarse para contener las dirección del dato en las instrucciones con direccionamiento indirecto y también para guardar operandos en sus 5 bits de menos peso.

Los registros que ocupan las posiciones 5 ,6 y 7 soportan los Puertos A, B y C de E/S. Pueden ser leídos y

escritos como cualquier otro registro y manejan los valores de los bits que entran y salen por los pines de E/S del microcontrolador.

- **PROGRAMACIÓN EN LENGUAJE ENSAMBLADOR**

El programa fuente esta compuesto por una sucesión de líneas de programa. Cada línea de programa esta compuesta por 4 campos separados por uno o más espacios o tabulaciones. Estos campos son:

[Etiqueta] Comando [Operando(s)] [;Comentario]

La etiqueta es opcional. El comando puede ser un mnemónico del conjunto de instrucciones. El operando esta asociado al comando, si no hay comando no hay operando, e inclusive algunos comandos no llevan operando. El comentario es opcional para el compilador aunque es buena práctica considerarlo obligatorio para el programador.

La etiqueta, es el campo que empieza en la primer posición de la línea. No se pueden insertar espacios o tabulaciones antes de la etiqueta sino será considerado comando. Identifica la línea de programa haciendo que el compilador le asigne un valor automáticamente. Si se trata de una línea cuyo comando es una instrucción de programa del microcontrolador, se le asigna el valor de la dirección de memoria correspondiente a dicha instrucción (location counter). En otros casos se le asigna un valor de una constante, o la dirección de una variable, o será el nombre de una macroinstrucción, etc.

El comando puede ser un código mnemónico de instrucción del microcontrolador, o una directiva o pseudoinstrucción para el compilador. En el primer caso será directamente traducido a código de maquina, en el segundo caso será interpretado por el compilador y realizara alguna acción en tiempo de compilación como ser asignar un valor a una etiqueta, etc.

El campo de parámetros puede contener uno o más parámetros separados por comas. Los parámetros dependen de la instrucción o directiva. Pueden ser números o literales que representen constantes o direcciones.

El campo de comentario debe comenzar con un caracter punto y coma. No necesita tener espacios o tabulaciones separándolo del campo anterior, e incluso puede empezar en la primer posición de la línea. El compilador ignora todo el texto que contenga la línea después de un caracter punto y coma. De esta manera pueden incluirse líneas que contengan solo comentarios, y es muy buena práctica hacer uso y abuso de esta posibilidad para que los programas resulten autodocumentados.

− CONJUNTO DE INSTRUCCIONES

El conjunto de instrucciones de los microprocesadores PIC 16CXX consiste en un pequeño repertorio de solo 33 instrucciones de 12 bits, que pueden ser agrupadas para su estudio en tres a cinco grupos. En este curso se ha optado por clasificarlas, desde el punto de vista del programador, en cinco categorías bien definidas de acuerdo con la función y el tipo de operandos involucrados. En primer lugar se agrupan las instrucciones que operan con bytes y que involucran algún registro de la memoria interna. En segundo lugar se analizaran las instrucciones que operan solo sobre el registro W y que permiten cargarle una constante implícita o incluida literalmente en la instrucción (literales). En tercer lugar se agrupan las instrucciones que operan sobre bits individuales de los registros de la memoria interna. En cuarto lugar se clasifican las instrucciones de control de flujo del programa, es decir las que permiten alterar la secuencia lineal de ejecución de las instrucciones. Por último se agrupan unas pocas instrucciones que llamaremos especiales, cuyas funciones o tipos de operandos son muy específicos y no encajan en ninguna de las clasificaciones anteriores.

Instrucciones de Byte que operan con Registros

Estas instrucciones pueden ser de simple o doble operando de origen. El primer operando de origen será siempre el registro seleccionado en la instrucción, el segundo, en caso de existir, será el registro W. El destino, es decir donde se guardara el resultado, será el registro seleccionado o el W, según se seleccione con un bit de la instrucción.

El formato genérico de estas instrucciones es el siguiente :
11 10 9 8 7 6 5 4 3 2 1 0

Los bits 0 a 4 (5 bits), denominados f permiten seleccionar uno de 32 registros de la memoria interna. El bit 5, denominado d, permite especificar el destino del resultado. Si d = 1 el resultado se guardara en el registro seleccionado. Si d = 0 el resultado se guardara en W. Los bits 6 a 11 identifican la instrucción especifica a realizar.

Las instrucciones siguientes son las tres operaciones lógicas de doble operando :

ANDWF f,d ;operación AND lógica, destino = W Ù f
IORWF f,d ;operación OR lógica, destino = W Ú f
XORWF f,d ;operación XOR lógica, destino = W Å f

Los nombres mnemónicos de estas instrucciones provienen de : AND W con F, Inclusive OR W con F y XOR W con F.
Las que siguen son las cuatro operaciones aritméticas y lógicas sencillas de simple operando :

MOVF f,d ;movimiento de datos, destino = f
COMF f,d ;complemento lógico, destino = NOT f
INCF f,d ;incremento aritmético, destino = f + 1
DECF f,d ;decremento aritmético, destino = f − 1

Los mnemónicos de estas instrucciones provienen de : MOVe File, COMplement File, NCrement File y DECrement File.

En las siete instrucciones anteriores el único bit afectado de la palabra de estado del procesador es el Z, que se pone en 1 si el resultado de la operación es 00000000, y se pone en 0 si el resultado tiene cualquier otro valor.

A continuación siguen las dos instrucciones de rotación de bits a través del CARRY :

RLF f,d ;rotación a la izquierda, destino = f ROT ¬
RRF f,d ;rotación a la derecha, destino = f ROT ®

En estas operaciones (Rotate Left File y Rotate Right File) los bits son desplazados de cada posición a la siguiente, en sentido derecho o izquierdo. El desplazamiento es cerrado, formando un anillo, con el bit C (CARRY) de la palabra de estado.

En estas dos instrucciones, el único bit afectado de la palabra de estado del procesador es el bit C, que tomará el valor que tenia el bit 7 o el bit 0, según sea el sentido del desplazamiento.

Estas instrucciones son muy útiles para la manipulación de bits, y además para realizar operaciones aritméticas, ya que en numeración binaria, desplazar un número a la izquierda es equivalente a multiplicarlo por 2, y hacia la derecha, a dividirlo por 2.

La instrucción siguiente realiza el intercambio de posiciones entre los cuatro bits menos significativos y los cuatro más significativos (nibble bajo y nibble alto).

SWAPF f,d ;intercambia nibbles, destino = SWAP f

Esta instrucción (SWAP File) no afecta ninguno de los bits de la palabra de estado del procesador.

Esta instrucción es muy útil para el manipuleo de números BCD empaquetados, en los que en un solo byte se guardan dos dígitos BCD (uno en cada nibble).

Las dos operaciones que siguen son la suma y la resta aritméticas :

ADDWF f,d ;suma aritmética, destino = f + W
SUBWF f,d ;resta aritmética, destino = f − W

Estas operaciones (ADD W a F y SUBstract W de F) afectan a los tres bits de estado C, DC y Z.

El bit Z se pone en 1 si el resultado de la operación es 00000000, y se pone en 0 si el resultado tiene cualquier otro valor.

La suma se realiza en aritmética binaria pura sin signo. Si hay un acarreo del bit 7, es decir que el resultado es mayor que 255, el bit C (carry) resulta 1, en caso contrario resulta 0. Si hay un acarreo del bit 3, es decir que la suma de las dos mitades (nibbles) menos significativas (bits 0 a 3) resulta mayor que 15, se pone en 1 el bit DC (digit carry), en caso contrario se pone en 0.

Ejemplos :
1010 0010 1101 0000
+ 0100 1111 **C DC Z** + 0110 1111 **C DC Z**
1111 0001 0 1 0 0011 1111 1 0 0

La resta se realiza sumando, en binario puro sin signo, el registro f más el complemento a dos de W (el complemento a 1, o complemento lógico, más 1)

Ejemplos :

f 0100 0100 0010 1000
W − 0010 1000 **C DC Z** − 0100 0100 **C DC Z**
0001 1100 1 0 0 1110 0100 0 1 0

equivalente a :
f 0100 0100 0010 1000
cmp.2 W + 1101 1000 **C DC Z** + 1011 1100 **C DC Z**
0001 1100 1 0 0 1110 0100 0 1 0

Los bits de estado C y DC toman el valor normal correspondiente a la suma de f con el complemento a 2 de W. De esta manera el significado para la operación de resta resulta invertido, es decir que C (carry) es 1 si no hubo desborde en la resta, o dicho de otra manera, si el contenido de W es menor que el de f. El bit DC se comporta de manera similar, es decir que DC es 1 si no hubo desborde en la mitad menos significativa, lo que equivale a decir que el nibble bajo del contenido de W es menor que el del registrof.

Las instrucciones que siguen son de simple operando, pero son casos especiales ya que el destino es siempre el registro seleccionado :

CLRF f ;borrado de contenido, f = 0

MOVWF f ;copia contenido W ® f, f = W

La instrucción CLRF (CLeaR File) afecta solo al bit Z que resulta siempre 0.
La instrucción MOVWF (MOVe W a F) no afecta ningún bit de la palabra de estado.

- **Instrucciones de Byte que operan sobre W y Literales**

Estas instrucciones se refieren todas al registro W, es decir que uno de los operandos de origen y el operando de destino son siempre el registro W. En las instrucciones de este grupo que tienen un segundo operando de origen, este es siempre una constante de programa literalmente incluida en la instrucción, llamada constante literal o simplemente literal.

El formato genérico de estas instrucciones es el siguiente :

11 10 9 8 7 6 5 4 3 2 1 0

Los bits 0 a 7 especifican la constante literal de 8 bits que se utilizara en la operación.

Las tres instrucciones que siguen son las operaciones lógicas tradicionales, similares a las que ya vimos anteriormente, pero realizadas entre una constante de programa y el registro W :

IORLW k ; operación OR lógica, W = W Ú k
ANDLW k ; operación AND lógica, W = W Ù k
XORLW k ; operación XOR lógica, W = W Å k

En estas tres instrucciones (Inclusive OR Literal W, AND Literal W y XOR Literal W) el único bit afectado de la palabra de estado del procesador es el Z, que se pone en 1 si el resultado de la operación es 00000000, y se pone en 0 si el resultado tiene cualquier otro valor.

La instrucción que sigue sirve para cargar una constante de programa en el registro W :

MOVLW k ;carga constante en W, W = K

Esta (MOVe Literal W) instrucción no afecta ninguno de los bits de estado del procesador.

La instrucción que sigue (CLeaR W) no correspondería incluirla en este grupo, y pertenece en realidad al primero, el de las instrucciones que operan sobre registros, ya que se trata de un caso especial de la instrucción CLRF, con destino W, y f = 0. La incluimos aquí porque como se le ha asignado un mnemónico particular referido específicamente al registro W, creemos que, desde el punto de vista del programador, es más útil verla dentro del grupo de instrucciones referidas a W.

CLRW ;borra el contenido de W, W = 0

Al igual que en la instrucción CLRF, el único bit de estado afectado es el Z que resulta1

− Instrucciones de Bit
El formato genérico de estas instrucciones es el siguiente :

11 10 9 8 7 6 5 4 3 2 1 0

Los bits 0 a 4 (5 bits), denominados f, permiten seleccionar uno de 32 registros de la memoria interna. Los bits 5 a 7, denominados b, permiten especificar el numero de bit (0 a 7) sobre el que se operara. Estas

instrucciones operan solamente sobre el bit especificado, el resto de los bits del registro no son alterados. Estas instrucciones no tienen especificación de destino, ya que el mismo es siempre el registro seleccionado.

BCF f,b ;borra el bit b de f ;bit f(b) = 0
BSF f,b ;coloca en uno el bit b de f ;bit f(b) = 1

Estas instrucciones (Bit Clear File y Bit Set File) no afectan ningún bit de la palabra de estado del procesador.

− **Instrucciones de Control**

GOTO k ;salto a la posición k (9 bits) del programa

Esta es la típica instrucción de salto incondicional a cualquier posición de la memoria de programa (que en la mayoría de los microprocesadores convencionales se llama JUMP). La constante literal k es la dirección de destino del salto, es decir la nueva dirección de memoria de programa a partir de la cual comenzarán a leerse las instrucciones después de ejecutar la instrucción GOTO. Esta instrucción simplemente carga la constante k en el registro PC (contador de programa). La única complicación de esta instrucción es que la constante k es de solo 9 bits, mientras que el registro PC es de 11 bits, ya que en el 16C57 debe permitir direccionar una memoria de programa de 2 K. Los dos bits faltantes, bit 9 y 10 del PC, son tomados respectivamente de los bits de selección de página PA0 y PA1 de la palabra de estado.

Este comportamiento particular hace que la memoria de programa aparezca como dividida en paginas de 512 posiciones como se vera más adelante. El programador debe tener en cuenta que antes de ejecutar una instrucción GOTO es posible que haya que programar los bits PA0 y PA1.

La que sigue es la instrucción de llamado a subrutina:

CALL k ;salto a la subrutina en la posición k (8 bits)

Su comportamiento es muy similar al de la instrucción GOTO, salvo que además de saltar guarda en el stack la dirección de retorno de la subrutina (para la instrucción RETLW). Esto lo hace simplemente guardando en el stack una copia del PC incrementado, antes de que el mismo sea cargado con la nueva dirección k. La única diferencia con la instrucción GOTO respecto de la forma en la que se realiza el salto, es que en la instrucción CALL la constante k tiene solo 8 bits en vez de 9. En este caso también se utilizan PA0 y PA1 para cargar los bits 9 y 10 del PC, pero además el bit 8 del PC es cargado siempre con 0. Esto hace que los saltos a subrutina solo puedan realizarse a posiciones que estén en las primeras mitades de las paginas mencionadas. El programador debe tener en cuenta este comportamiento y asegurarse de ubicar las posiciones de inicio de las subrutinas en las primeras mitades de las paginas.

La instrucción que aparece a continuación es la de retorno de subrutina:

RETLW k ;retorno de subrutina con constante k, W = k

Esta (RETurn con Literal in W) instrucción produce el retorno de subrutina con una constante literal k en el registro W. La operación que realiza consiste simplemente en sacar del stack un valor y cargarlo en el PC. Ese valor es el PC incrementado antes de realizar el salto, de la ultima instrucción CALL ejecutada, por lo tanto es la dirección de la instrucción siguiente a dicho CALL.. Dado que el stack es de 11 bits, el valor cargado en el PC es una dirección completa, y por lo tanto se puede retornar a cualquier posición de la memoria de programa, sin importar como estén los bits de selección de pagina. Esta instrucción además carga siempre una constante literal en el registro W. Ya que esta es la única instrucción de retorno de subrutina de los PIC16CXX, no hay en estos microprocesadores forma de retornar de una subrutina sin alterar el registro W.

Por otro lado, y con una metodología especial de programación, un conjunto de sucesivas instrucciones RETLW puede ser usado como una tabla de valores constantes incluida en el programa (Ej. : tablas BCD/7 seg., hexa/ASCII, etc.).

A continuación se presentan las dos únicas instrucciones de salteo (skip) condicional. Estas instrucciones son los únicos medios para implementar bifurcaciones condicionales en un programa. Son muy generales y muy poderosas ya que permiten al programa tomar decisiones en función de cualquier bit de cualquier posición de la memoria interna de datos, y eso incluye a los registros de periféricos, los puertos de entrada/salida e incluso la palabra de estado del procesador. Estas dos instrucciones reemplazan y superan a todo el conjunto de instrucciones de salto condicional que poseen los microprocesadores sencillos convencionales (salto por cero, por no cero, por carry, etc.).

BTFSC f,b ;salteo si bit = 0, bit = f(0) Þ saltea
BTFSS f,b ;salteo si bit = 1, bit = f(1) Þ saltea

BTFSC (Bit Test File and Skip if Clear) saltea la próxima instrucción si el bit b del registro f es cero. La instrucción BTFSS (Bit Test File and Skip if Set) saltea si el bit es 1. Estas instrucciones pueden usarse para realizar o no una acción según sea el estado de un bit, o, en combinación con GOTO, para realizar una bifurcación condicional.

Ejemplo 1 : Ejemplo 2:

Las instrucciones que siguen son casos especiales de las de incremento y decremento vistas anteriormente. Estas instrucciones podrían categorizarse dentro del grupo de instrucciones orientadas a byte sobre registros (primer grupo), ya que efectivamente operan sobre los mismos, y el formato del código de la instrucción responde al de ese grupo, pero, a diferencia de las otras, pueden además alterar el flujo lineal del programa y por eso se les incluyó en este grupo.

DECFSZ f,d ;decrementa y saltea sí 0, destino= f − 1, = 0 Þ saltea
INCFSZ f,d ;incrementa y saltea sí 0, destino= f + 1, = 0 Þ saltea

Estas dos instrucciones (DECrement File and Skip if Zero, e INCrement File and Skip if Zero) se comportan de manera similar a DECF e INCF, salvo que no afectan a ningún bit de la palabra de estado. Una vez realizado el incremento o decremento, si el resultado es 00000000, el microprocesador salteara la próxima instrucción del programa. Estas instrucciones se utilizan generalmente en combinación con una instrucción de salto (GOTO), para el diseño de ciclos o lazos (loops) de instrucciones que deben repetirse una cantidad determinada de veces.

Ejemplo:
clrf 10 ; pongo cero en la posición 10 de la memoria interna
loop ; lo que sigue se ejecutará 256 veces

incfsz 10,1; incremento la posición 10 hasta que llegue a 0
goto loop ; si no llego a cero voy a repetir la secuencia
cuando llegue a cero salteo el goto y sigue la continuación del programa

- **Instrucciones Especiales**

En este grupo se reunieron las instrucciones que controlan funciones específicas del microprocesador o que actúan sobre registros especiales no direccionados como memoria interna normal.

La instrucción que sigue es la típica NO OPERATION, existente en casi todos los microprocesadores.

NOP ;no hace nada, consume tiempo

Esta instrucción solo sirve para introducir una demora en el programa, equivalente al tiempo de ejecución de una instrucción. No afecta ningún bit de la palabra de estado.

La siguiente es una instrucción especifica de control de los puertos de entrada/salida.

TRIS f ;carga el tristate control, TRISf = W

Esta instrucción (TRIState) carga el registro de control de los buffers tristate de un puerto de entrada salida (data dirección register), con el valor contenido en W. El parámetro f debe ser la dirección de memoria interna del puerto, aunque el valor W no será cargado en el puerto sino en el registro de tristate del mismo. Los valores validos para f son 4 y 5 en los 16C54/56 y 4, 5 y 6 en los 16C55/57. Esta instrucción no afecta ningún bit de la palabra de estado.

La siguiente instrucción sirve para programar el registro OPTION que controla el RTCC y prescaler

OPTION ;carga el registro OPTION, OPTION = W

El registro OPTION no es accesible como memoria interna y solo se lo puede programar con esta instrucción. Esta instrucción no afecta ningún bit de la palabra de estado.

La instrucción que sigue borra el contador del watch dog timer. Este registro tampoco esta accesible como memoria, y esta es la única instrucción que lo modifica.

CLRWDT ;borra el watch dog timer, WDT = 0

Esta instrucción, además, coloca en uno los bits PD (power down) y TO (time−out) de la palabra de estado.

La siguiente es una instrucción especial de control del microcontrolador que lo pone en el modo power down. En este modo el microprocesador se detiene, el oscilador se apaga, los registros y puertos conservan su estado, y el consumo se reduce al mínimo. La única forma de salir de este estado es por medio de un reset o por time−out del watch dog timer.

SLEEP ;coloca el µC en modo sleep, WDT = 0

Esta instrucción, además, borra el bit PD (power down) y setea el bit TO (time−out) de la palabra de estado.

- **Direccionamiento de la memoria de datos (RAM)**

La memoria interna se direcciona en forma directa por medio de los 5 bits f contenidos en las instrucciones que operan sobre registros. De esta manera se puede direccionar cualquier posición desde la 00 a la 1F. Como se vió en el capítulo correspondiente a los mapas de memoria, las direcciones 10 a 1F corresponden a los bancos de registros, por lo tanto, en los microcontroladores que tengan más de un banco, antes de acceder a alguna variable que se encuentre en esta zona, el programador deberá asegurarse de haber programado los bits de selección de banco en el registro FSR.

Los registros especiales y de uso general de la posición 00 a la 0f están presentes en todos los PIC16CXX, al igual que el banco 0 de registros. Los bancos 1, 2 y 3 de registros están presentes solo en el 16C57.

El registro FSR, además de servir para seleccionar el banco activo, sirve como puntero para direccionamiento indirecto. La posición 00 del mapa de RAM es la llamada dirección indirecta. Sí en cualquier instrucción se

opera con la dirección 00, en realidad se estará operando con la dirección a donde apunte el contenido del FSR. Por ejemplo si el FSR contiene el valor 14, una instrucción que opere sobre la dirección 0, operara en realidad sobre la dirección 14. Se puede decir en este ejemplo que la posición 14 de memoria fue direccionada en forma indirecta a través del puntero FSR.

Ejemplo :

Esta porción de programa borra 5 posiciones de memoria a partir de la dirección 12

FSR equ 04 ; (definición al comienzo del programa)

```
movlw 5 ;prepara para repetir 5 veces
movwf 08 ;(el registro 08 es el contador del loop)
movlw 12h ;apunta a la dirección 12h
movwf FSR ;
loop:
clrf 0 ;borra una posición de memoria
incf FSR ;apunta a la siguiente
decfsz 08 ;si todavía no borra todas
goto loop ;sige borrando
```

El direccionamiento indirecto es muy útil para el procesamiento de posiciones consecutivas de memoria, como en el ejemplo, o para el direccionamiento de datos en subrutinas.

− Direccionamiento de la memoria de programa (EPROM, OTP)
La instrucción GOTO dispone solo de 9 bits en el código de operación para especificar la dirección de destino del salto. Al ejecutar una instrucción GOTO el microprocesador toma los dos bits que restan para completar la dirección de 11 bits, de los bits 5 y 6 de la palabra de estado. Estos últimos son llamados bits de selección de página (PA0 y PA1). El programador deberá asegurarse de que estos dos bits tengan el valor correcto antes de toda instrucción GOTO.

FIG. Direccionamiento directo con instrucción GOTO

Deberá tenerse en cuenta además que es posible avanzar de una página a otra en forma automática cuando el PC se incrementa. Esto ocurre si el programa empieza en una página y sigue en la siguiente.

Sin embargo, al incrementarse el PC desde la última posición de una página a la primera de la siguiente, **los bits PA0 y PA1 no se modifican**, y por lo tanto sí se ejecuta una instrucción GOTO, CALL o alguna que actúe sobre el PC, esta producirá un salto a la página anterior, a menos que el programador tenga la precaución de actualizar el valor de dichos bits. Por este motivo es conveniente dividir el programa en módulos o rutinas que estén confinados a una página.

En el caso de la instrucción CALL, el direccionamiento se complica un poco más, ya que la misma solo dispone de 8 bits para especificar la dirección de destino salto. En este caso también se utilizan los mismos bits de selección de página para completar los bits décimo y decimoprimero de la dirección, pero falta el noveno bit. En estas instrucciones este bit se carga siempre con 0, lo que implica que solo se pueden realizar saltos a subrutina a las mitades inferiores de cada página. En este caso también el programador deberá asegurarse que el estado de los bits PA0 y PA1 sea el correcto al momento de ejecutarse la instrucción.

FIG.Direccionamiento directo con instrucción CALL

Las instrucciones que operan sobre el PC como registro y alteran su contenido provocando un salto,

responden a un mecanismo muy similar al de las instrucciones CALL para la formación de la dirección de destino. En este caso los bits 0 a 7 son el resultado de la instrucción, el bit 8 es 0 y los bits restantes se toman de PA0 y PA1.

Este mecanismo se llama paginado, y a pesar de que representa una complicación bastante molesta para el programador, resulta muy útil ya que permite ampliar la capacidad de direccionamiento de memoria de programa para las instrucciones de salto.

- **PROYECTO**
- DIAGRAMA DE FLUJO
- ESQUEMA ELÉCTRICO
- PROGRAMACIÓN EN LENGUAJE ENSAMBLADOR