FACULTAD DE INGENIERÍA UNAM
DIVISIÓN DE EDUCACIÓN CONTINUA

## CURSOS ABIERTOS

# CURSO MICROCONTROLADORES PIC ( CA 241)

## DEL 09 AL 13 DE DICIEMBRE DE 2002

## TEMA

## EL MICROCONTROLADOR RISC PIC16/17

EXPOSITOR: ING. EDUARDO RAMÍREZ SÁNCHEZ
ING. J. SALVADOR ZAMORA ALARCÓN
PALACIO DE MINERIA
DICIEMBRE DE 2002

# INDICE

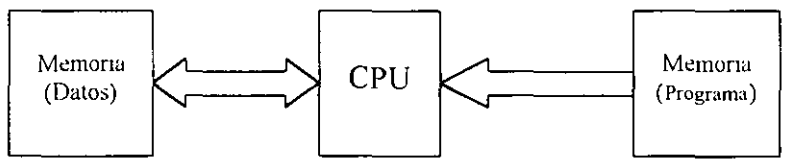
# EL MICROCONTROLADOR RISC PIC16/17

## Descripción general

La familia de microcontroladores PIC16/17 son del tipo RISC (reduced instruction set computer), y que, además de que posee un numero reducido de instrucciones, tiene las siguientes características importantes:

### Arquitectura tipo Harvard

En la arquitectura tipo Harvard se utilizan 2 espacios de memoria independientes una memoria para el programa (instrucciones) y otra para los datos. Con esto, podemos tener dos diferentes tamaños para los buses de datos y de programa y la ventaja de que se incrementa la velocidad de ejecución del programa, debido a que se leen simultáneamente tanto las instrucciones como los datos.




### Palabra larga de instrucción

Como el tamaño del bus de programa puede ser grande (12, 14 o 16 bits), entonces podemos tener instrucciones que contengan tanto el código de operación de la instrucción como el operando, permitiendo que se tengan instrucciones de una sola palabra y que se ejecuten en un solo ciclo de máquina.

Por ejemplo:

PIC16C7X : MOVLW 0X2F

| | | | |
|---|---|---|---|
| 11 | 00XX | 0010 | 1111 |

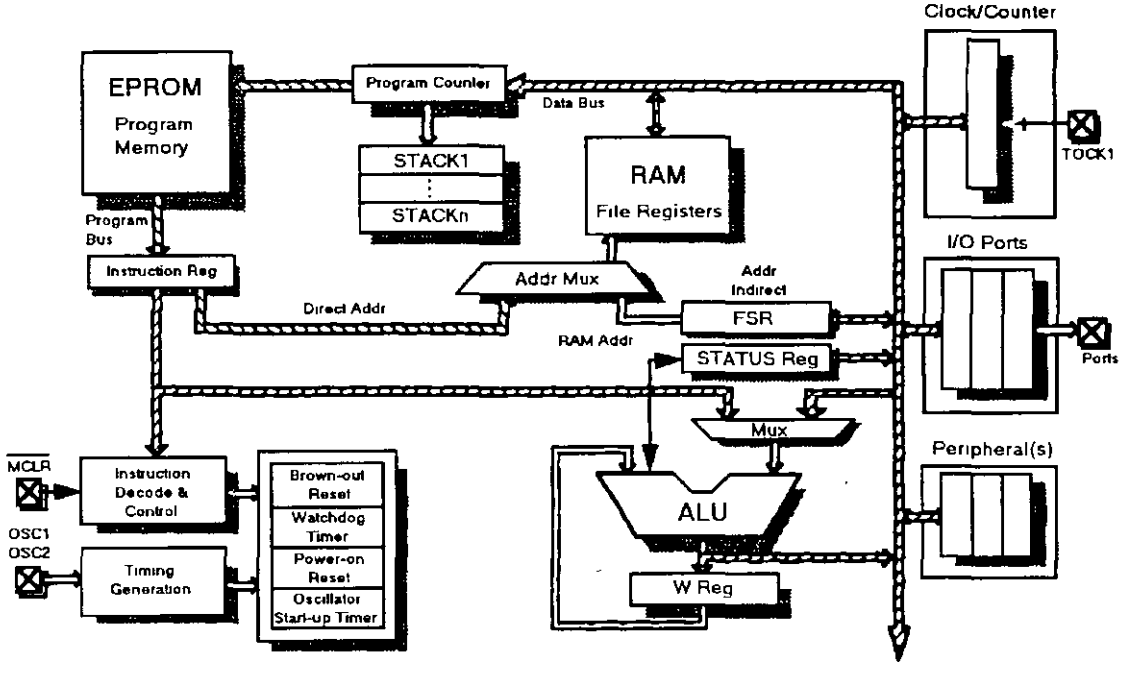
### Arquitectura Pipeline

En la mayoría de los microcontroladores, existe un ciclo de búsqueda de la instrucción y un ciclo de ejecución de la misma que se producen secuencialmente La arquitectura pipeline de los microcontroladores PIC permite que los ciclos de búsqueda y de ejecución de una instrucción se traslapen, haciendo posible que las instrucciones se ejecuten en un solo ciclo, por ejemplo:

| | Tcy0 | Tcy1 | Tcy2 | Tcy3 | Tcy4 |
|---|---|---|---|---|---|
| 1 MOVLW 0X55 | Fetch 1 | Execute 1 | | | |
| 2. MOVWF PORTB | | Fetch2 | Execute 2 | | |
| 3. CALL SUB_1 | | | Fetch 3 | Execute 3 | |
| 4. BSF PORTA,BIT3 | | | | Fetch 4 | |

**Diagrama a bloques**

En la siguiente figura se muestra el diagrama general de la arquitectura interna de los microcontroladores PIC16/17.

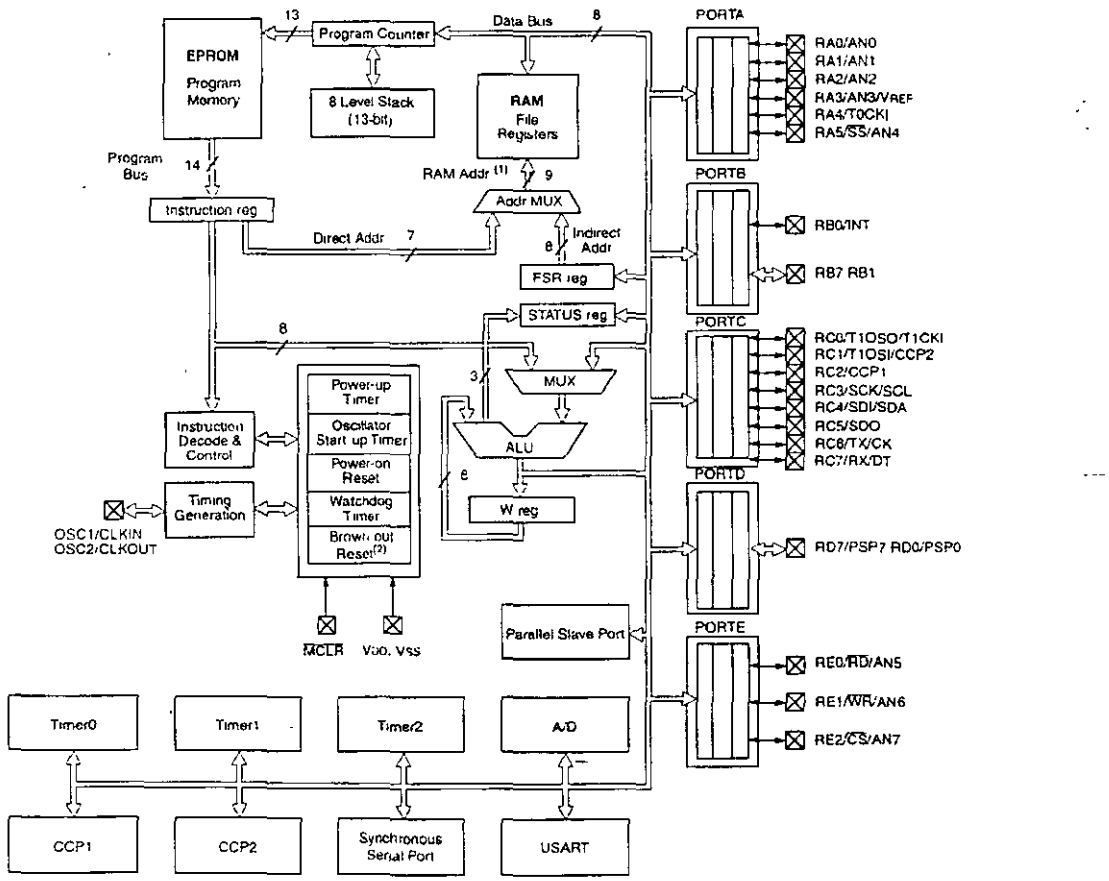



En este diagrama a bloques podemos observar claramente que se cuenta con dos tipos diferentes de memoria lo que representa la característica principal de una arquitectura tipo Harvard Podemos ver también los elementos principales de esta clase de microcontroladores:

Una unidad aritmética y lógica que es donde se realizan todas las operaciones aritméticas y lógicas
Un registro W que siempre es un operando en cualquier operación y también donde puede almacenarse el resultado de la operación.
Un registro de STATUS cuyos bits nos indican, entre otras cosas, las condiciones de la ALU después de que se efectúa una operación.
El registro contador de programa (Program Counter) que es el que nos lleva la secuencia de ejecución de nuestro programa y accesa localidades de memoria tanto de la memoria de programa como de la de registros

Dos bloques importantes de memoria: para programa y para datos y registros
Un tercer bloque de memoria que se utiliza para la pila (stack).
Un registro de instrucción que es donde se almacena la instrucción que se va a ejecutar.
Un bloque de decodificación de instrucción y control que es donde se ve cual es la función que va a realizar la instrucción y se generan todas las señales de control que se necesitan para que se lleve a cabo.
Un bloque de generación de tiempos que es donde, a partir del tipo de oscilador que se le conecte, se generan los ciclos de máquina con que trabaja el microcontrolador, genera señales de reloj externas y señales para un bloque de funciones especiales.
El bloque de funciones especiales controla lo referente a todas las formas de reinicialización ( tipos de reset) y puesta en marcha del microcontrolador.
Un registro para un direccionamiento de datos indirecto.
Multiplexores que nos permiten la transferencia de datos que proceden de diferentes buses.
El bloque reloj/contador es un temporizador que tiene la función de aceptar un reloj externo para generar tiempos, o bien, para contar eventos externos.
Una serie de puertos de entrada salida con los que podemos sacar o leer niveles TTL (0 o 5 volts).
Un bloque de periféricos en donde básicamente podemos encontrar elementos que realicen funciones tales como convertidores, temporizadores y puertos serie.

En la siguiente figura se puede observar un diagrama de bloques para una versión mas particular del microcontrolador PIC



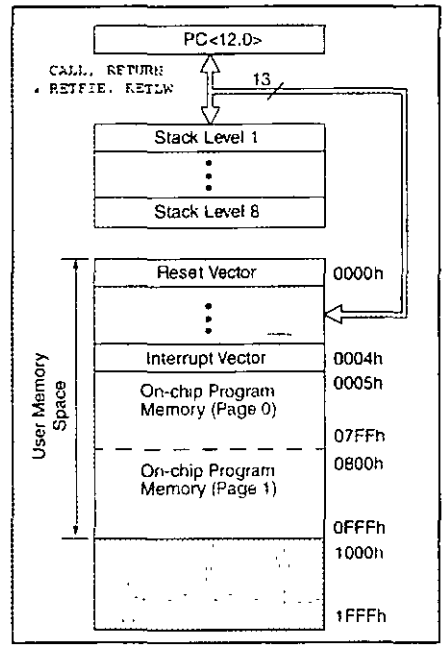
## Mapa de Memoria y Registros Internos

Para este tipo de microcontroladores tendremos dos mapas de memoria . el de programa y el de datos. En este caso describiremos los mapas de memoria para los PIC16C74/73

### Mapa de memoria de programa

El mapa de memoria de programa incluye tanto la pila ( stack) como las direcciones donde se encuentra el programa La siguiente figura muestra este mapa:

**PIC16C73/73A/74/74A**
**PROGRAM MEMORY MAP**
**AND STACK**




Dentro de esta figura podemos observar que se cuenta con un contador de programa de 13 bits (desde el bit0 hasta el bit 12) que puede direccionar desde 0000h hasta 1FFFh (8K) y con una longitud de palabra de 14 bits, pero en esta versión del PIC, solamente se tiene implementada fisicamente hasta la dirección 0FFFh (4K x 14). Esto implica que el bit 12 siempre será 0 para el contador de programa

También se observa que para estos PIC's se tienen 8 niveles en la pila (stack) y que para este caso están implementadas con memoria RAM

Aunque todo el espacio de memoria de programa está designado para el programa de los usuarios, existen dos localidades reservadas para uso particular: la dirección 0000h para el reset y la 0004h para el vector de interrupción.

## Mapa de memoria de datos

La memoria de datos esta divida en dos bancos de 128 bytes cada uno y que incluyen una parte de registros especiales y una de registros de propósito general como se muestra en la siguiente figura:

**PIC16C73/73A/74/74A**
**REGISTER FILE MAP**

| File Address | Bank 0 | Bank 1 | File Address |
|---|---|---|---|
| 00h | INDF(1) | INDF(1) | 80h |
| 01h | TMR0 | OPTION | 81h |
| 02h | PCL | PCL | 82h |
| 03h | STATUS | STATUS | 83h |
| 04h | FSR | FSR | 84h |
| 05h | PORTA | TRISA | 85h |
| 06h | PORTB | TRISB | 86h |
| 07h | PORTC | TRISC | 87h |
| 08h | PORTD(2) | TRISD(2) | 88h |
| 09h | PORTE(2) | TRISE(2) | 89h |
| 0Ah | PCLATH | PCLATH | 8Ah |
| 0Bh | INTCON | INTCON | 8Bh |
| 0Ch | PIR1 | PIE1 | 8Ch |
| 0Dh | PIR2 | PIE2 | 8Dh |
| 0Eh | TMR1L | PCON | 8Eh |
| 0Fh | TMR1H | | 8Fh |
| 10h | T1CON | | 90h |
| 11h | TMR2 | | 91h |
| 12h | T2CON | PR2 | 92h |
| 13h | SSPBUF | SSPADD | 93h |
| 14h | SSPCON | SSPSTAT | 94h |
| 15h | CCPR1L | | 95h |
| 16h | CCPR1H | | 96h |
| 17h | CCP1CON | | 97h |
| 18h | RCSTA | TXSTA | 98h |
| 19h | TXREG | SPBRG | 99h |
| 1Ah | RCREG | | 9Ah |
| 1Bh | CCPR2L | | 9Bh |
| 1Ch | CCPR2H | | 9Ch |
| 1Dh | CCP2CON | | 9Dh |
| 1Eh | ADRES | | 9Eh |
| 1Fh | ADCON0 | ADCON1 | 9Fh |
| 20h | General Purpose Register | General Purpose Register | A0h |
| 7Fh | | | FFh |

☐ Unimplemented data memory locations, read as '0'

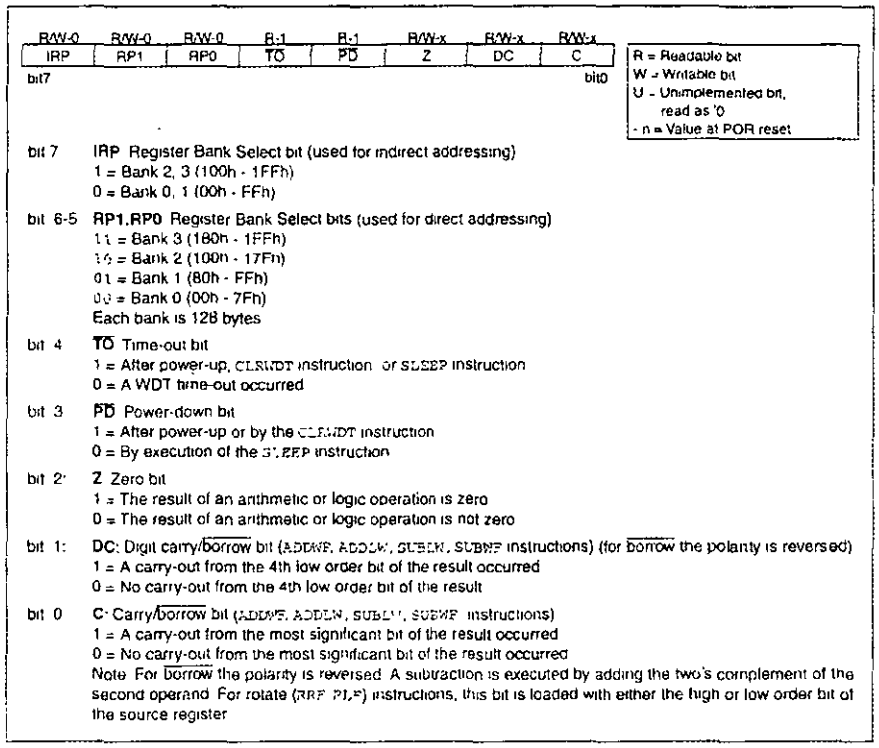
Note 1: Not a physical register
2. These registers are not physically implemented on the PIC16C73/73A, read as '0'.

Los registros de propósito general son localidades de memoria RAM que el usuario puede utilizar para almacenar los datos que necesite durante su programa, mientras que los registros especiales tienen un fin más específico

Los registros especiales se dividen en dos grupos diferentes: los que se utilizan para programar funciones generales y válidas para todo el microcontrolador y los registros que se utilizan para las funciones propias de cada interfase.

A continuación se presentan los registros especiales que sirven para *programar* funciones generales del microcontrolador:

Registro **STATUS**:



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

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7 **IRP** Register Bank Select bit (used for indirect addressing)
1 = Bank 2, 3 (100h - 1FFh)
0 = Bank 0, 1 (00h - FFh)

bit 6-5 **RP1,RP0** Register Bank Select bits (used for direct addressing)
11 = Bank 3 (180h - 1FFh)
10 = Bank 2 (100h - 17Fh)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)
Each bank is 128 bytes

bit 4 **$\overline{TO}$** Time-out bit
1 = After power-up, CLRWDT instruction or SLEEP instruction
0 = A WDT time-out occurred

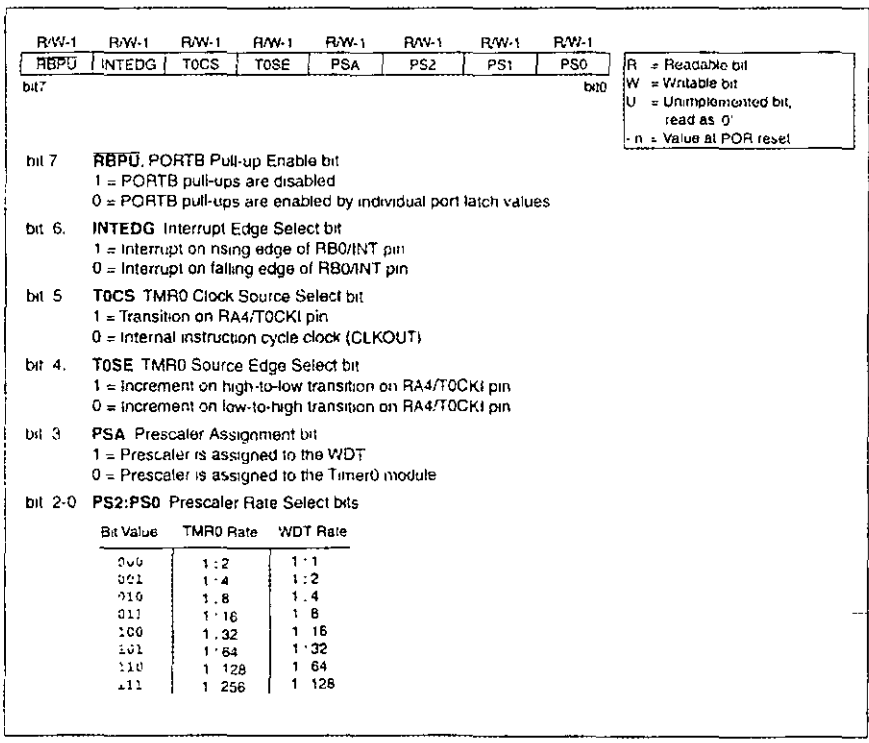
bit 3 **$\overline{PD}$** Power-down bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 2 **Z** Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1: **DC**: Digit carry/$\overline{borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{borrow}$ the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0 **C**: Carry/$\overline{borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)
1 = A carry-out from the most significant bit of the result occurred
0 = No carry-out from the most significant bit of the result occurred
Note For $\overline{borrow}$ the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low order bit of the source register.

Registro **OPTION**:



| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| $\overline{RBPU}$ | INTEDG | T0CS | T0SE | PSA | PS2 | PS1 | PS0 |
| bit7 | | | | | | | bit0 |

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7 **$\overline{RBPU}$**: PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG** Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS** TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

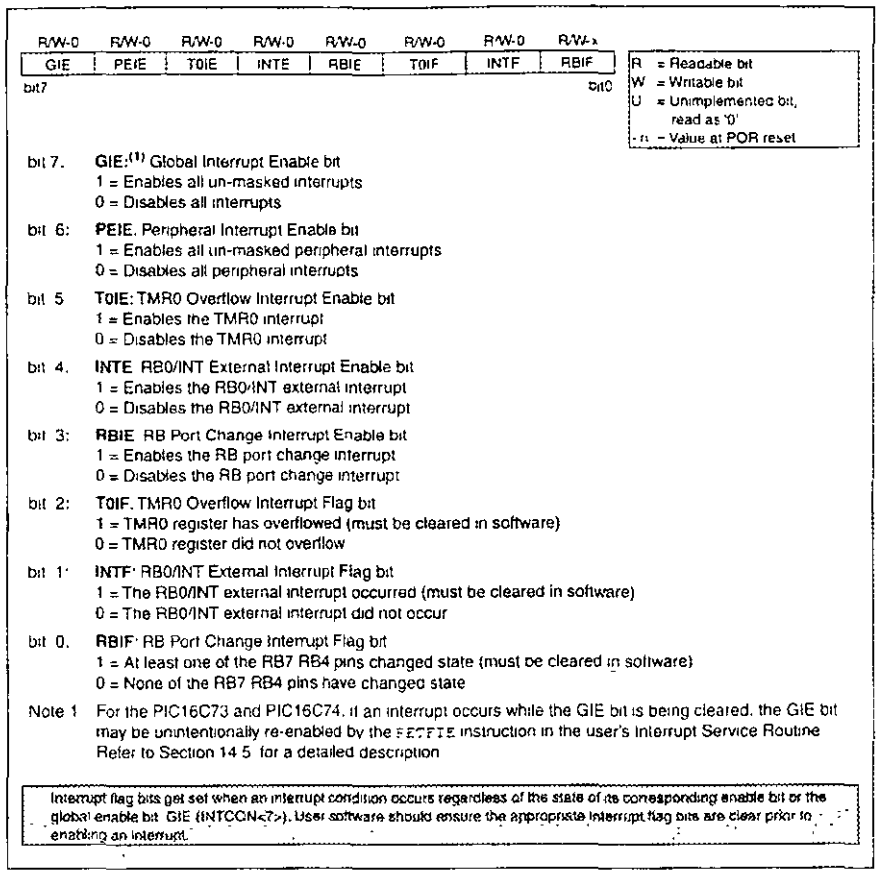
bit 4 **T0SE** TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA** Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0** Prescaler Rate Select bits

| Bit Value | TMR0 Rate | WDT Rate |
|---|---|---|
| 000 | 1 : 2 | 1 : 1 |
| 001 | 1 : 4 | 1 : 2 |
| 010 | 1 : 8 | 1 : 4 |
| 011 | 1 : 16 | 1 : 8 |
| 100 | 1 : 32 | 1 : 16 |
| 101 | 1 : 64 | 1 : 32 |
| 110 | 1 : 128 | 1 : 64 |
| 111 | 1 : 256 | 1 : 128 |

## 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 | T0IE | INTE | RBIE | T0IF | INTF | RBIF |
| bit7 | | | | | | | bit0 |

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7. **GIE:**(1) Global Interrupt Enable bit
1 = Enables all un-masked interrupts
0 = Disables all interrupts

bit 6: **PEIE**. Peripheral Interrupt Enable bit
1 = Enables all un-masked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **T0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4. **INTE** RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3: **RBIE** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2: **T0IF**. TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

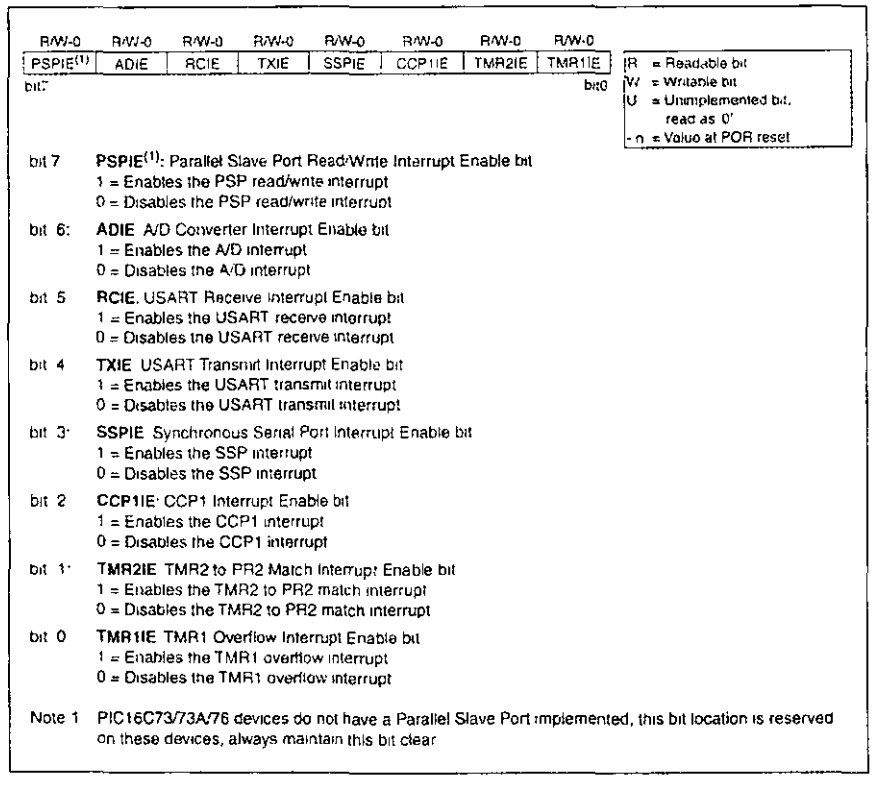
bit 1: **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0. **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Note 1 For the PIC16C73 and PIC16C74, if an interrupt occurs while the GIE bit is being cleared, the GIE bit may be unintentionally re-enabled by the `RETFIE` instruction in the user's Interrupt Service Routine. Refer to Section 14.5 for a detailed description.

Interrupt flag bits get set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

## 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 |
| bit7 | | | | | | | bit0 |

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7 **PSPIE**(1): Parallel Slave Port Read/Write Interrupt Enable bit
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

bit 6: **ADIE** A/D Converter Interrupt Enable bit
1 = Enables the A/D interrupt
0 = Disables the A/D interrupt

bit 5 **RCIE**. USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4 **TXIE** USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

bit 3: **SSPIE** Synchronous Serial Port Interrupt Enable bit
1 = Enables the SSP interrupt
0 = Disables the SSP interrupt

bit 2 **CCP1IE**: CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1: **TMR2IE** TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE** TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Note 1 PIC16C73/73A/76 devices do not have a Parallel Slave Port implemented, this bit location is reserved on these devices, always maintain this bit clear.

## 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 |
| bit7 | | | | | | | bit0 |

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7: **PSPIF(1)**: Parallel Slave Port Read/Write Interrupt Flag bit
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred

bit 6 **ADIF** A/D Converter Interrupt Flag bit
1 = An A/D conversion completed (must be cleared in software)
0 = The A/D conversion is not complete

bit 5 **RCIF**. USART Receive Interrupt Flag bit
1 = The USART receive buffer is full (cleared by reading RCREG)
0 = The USART receive buffer is empty

bit 4. **TXIF**: USART Transmit Interrupt Flag bit
1 = The USART transmit buffer is empty (cleared by writing to TXREG)
0 = The USART transmit buffer is full

bit 3: **SSPIF**: Synchronous Serial Port Interrupt Flag bit
1 = The transmission/reception is complete (must be cleared in software)
0 = Waiting to transmit/receive

bit 2 **CCP1IF**: CCP1 Interrupt Flag bit
Capture Mode
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare Mode
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM Mode
Unused in this mode

bit 1. **TMR2IF**: TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF**: TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

Note 1 PIC16C73/73A/76 devices do not have a Parallel Slave Port implemented, this bit location is reserved on these devices, always maintain this bit clear.

Interrupt flag bits get set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

## Registro **PIE2**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | CCP2IE |
| bit7 | | | | | | | bit0 |

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7-1: **Unimplemented:** Read as '0'

bit 0: **CCP2IE**: CCP2 Interrupt Enable bit
1 = Enables the CCP2 interrupt
0 = Disables the CCP2 interrupt

Registro **PIR2**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | CCP2IF |
| bit7 | | | | | | | bit0 |

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7-1: **Unimplemented:** Read as '0'

bit 0: **CCP2IF**: CCP2 Interrupt Flag bit

Capture Mode
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred

Compare Mode
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred

PWM Mode
Unused

Interrupt flag bits get set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

Registro **PCON**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-q |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | $\overline{POR}$ | $\overline{BOR}^{(1)}$ |
| bit7 | | | | | | | bit0 |

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7-2: **Unimplemented:** Read as '0'

bit 1: $\overline{POR}$, Power-on Reset Status bit
1 = No Power-on Reset occurred
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

bit 0. $\overline{BOR}^{(1)}$: Brown-out Reset Status bit
1 = No Brown-out Reset occurred
0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)

Note 1: Brown-out Reset is not implemented on the PIC16C73/74.

## Manejo de los mapas de memoria

### - Contador de Programa (Registros PCL y PCLATH)

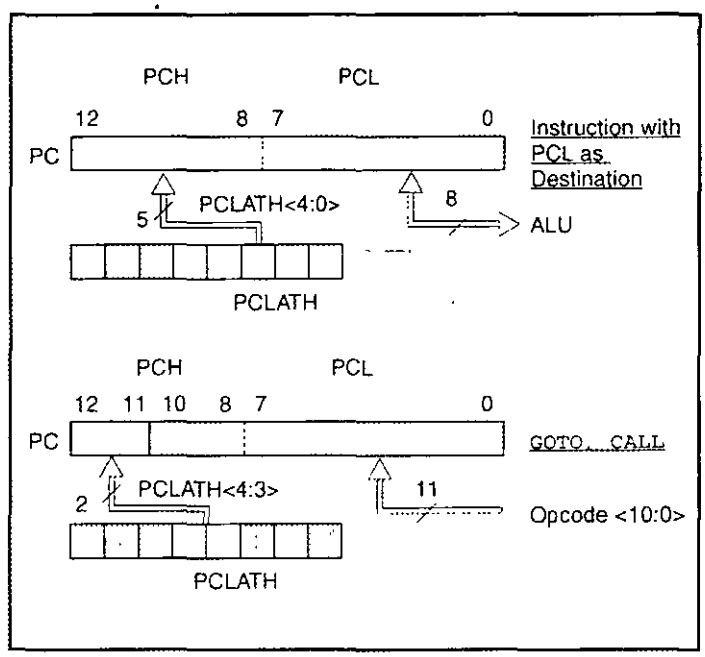
El contador de programa (PC) es un registro de 13 bits donde los 8 bits menos significativos forman el registro PCL que se puede leer o escribir. Los 5 bits más significativos (PCH) no pueden ser leídos, pero son escritos indirectamente por medio del registro PCLATH

Aun cuando con los 13 bits del PC podemos direccionar hasta 8K de memoria, en el PIC16C73/74 solamente se tienen implementadas 4Kb de localidades de memoria. Esto implica que el bit más significativo del PC siempre es cero.

El registro PCLATH nos permite trabajar en diferentes páginas del mapa de memoria. Esto es importante ya que las instrucciones de *CALL* y *GOTO* solamente utilizan 11 bits de direcciones lo que permite saltos en páginas de 2K. Cuando utilizamos estas instrucciones, el PC se forma por los 11 bits que proporcionan las instrucciones y el bit 3 del registro PCLATH, lo que nos permite hacer brincos o llamados a subrutinas de otras páginas.

Cuando se utiliza un *CALL* o se genera una interrupción los 13 bits del registro PC se almacenan en la pila (stack). Con esto, logramos que aun cuando las rutinas estén en otras páginas, los retornos siempre serán a la página de donde fueron llamadas.

La siguiente figura muestra la carga de contador de programa (PC) para diferentes situaciones:




**- La Pila ( Stack)**

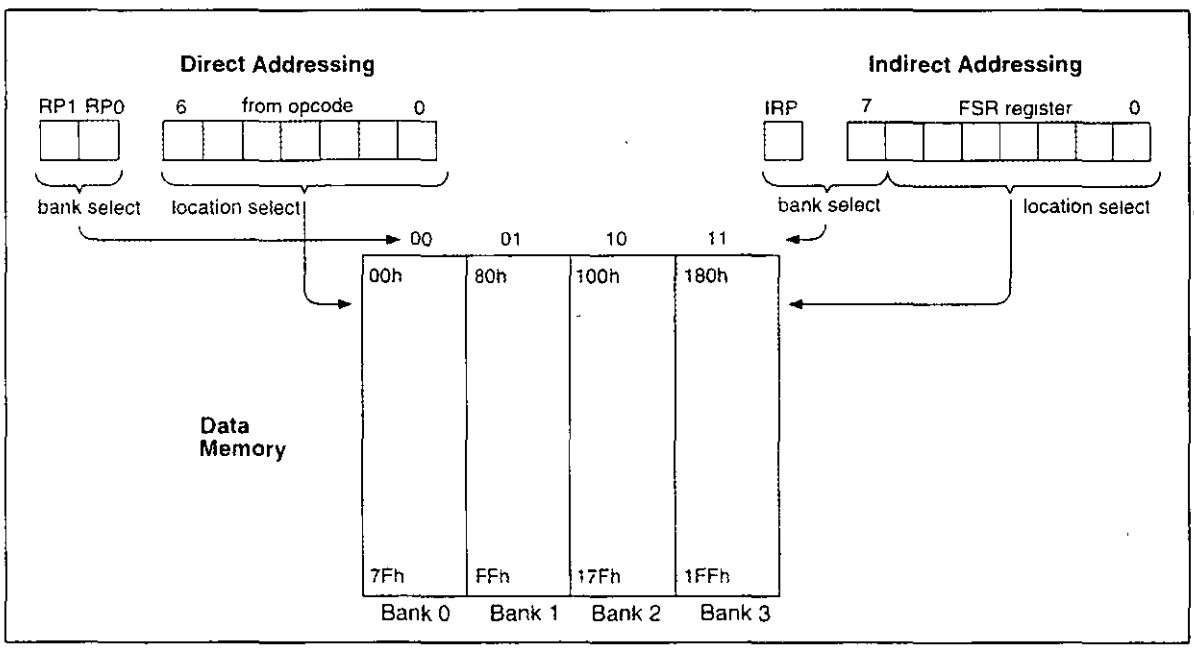
La pila o stack esta formada por 8 registros de 13 bits y cuyo apuntador no puede ser leído. En este stack se almacena el valor del contador de programa (PC) cuando se ejecuta una instrucción de *CALL* o cundo se reconoce una interrupción. El stack es circular de tal forma que cuando se hacen mas de 8 llamados rutinas, entonces se empieza a llenar nuevamente. Las instrucciones *RETURN*, *RETLW* o *RETFIE* cargan el PC con el valor almacenado en el stack.

**- Direccionamiento indirecto**

Es posible tener un direccionamiento indirecto de la memoria de datos por medio del registro INDF. El registro INDF no está físicamente implementado. Cualquier instrucción que utilice el registro INDF estará direccionando el dato apuntado por el registro de selección de archivo FSR.

Los 9 bits de la dirección del dato están formados por los 8 bits del registro FSR y el bit IRP del registro STATUS. Para los PIC16C73/74 que solamente tiene implementado dos bancos el bit IRP será siempre cero.

La siguiente figura muestra las formas de direccionar a la memoria de datos:




## Conjunto de instrucciones

Todos los PIC16CXX tienen un conjunto de 35 instrucciones de 14 bits Los 14 bits incluyen el código de operación que especifica que tipo de instrucción es y uno a más operandos. El conjunto de instrucciones está agrupado en tres categorías·

- Operaciones orientadas a byte
- Operaciones orientadas a bit
- Operaciones de control y con literales

La siguiente tabla muestra un resumen del conjunto de instrucciones y la explicación de cada instrucción se puede ver en el apéndice:

**PIC16CXX INSTRUCTION SET**

| Mnemonic, Operands | | Description | Cycles | 14-Bit Opcode MSb | LSb | Status Affected | Notes |
|---|---|---|---|---|---|---|---|
| **BYTE-ORIENTED FILE REGISTER OPERATIONS** | | | | | | | |
| ADDWF | f, d | Add W and f | 1 | 00 0111 dfff ffff | | C,DC,Z | 1,2 |
| ANDWF | f, d | AND W with f | 1 | 00 0101 dfff ffff | | Z | 1,2 |
| CLRF | f | Clear f | 1 | 00 0001 1fff ffff | | Z | 2 |
| CLRW | - | Clear W | 1 | 00 0001 0xxx xxxx | | Z | |
| COMF | f, d | Complement f | 1 | 00 1001 dfff ffff | | Z | 1,2 |
| DECF | f, d | Decrement f | 1 | 00 0011 dfff ffff | | Z | 1,2 |
| DECFSZ | f, d | Decrement f, Skip if 0 | 1(2) | 00 1011 dfff ffff | | | 1,2,3 |
| INCF | f, d | Increment f | 1 | 00 1010 dfff ffff | | Z | 1,2 |
| INCFSZ | f, d | Increment f, Skip if 0 | 1(2) | 00 1111 dfff ffff | | | 1,2,3 |
| IORWF | f, d | Inclusive OR W with f | 1 | 00 0100 dfff ffff | | Z | 1,2 |
| MOVF | f, d | Move f | 1 | 00 1000 dfff ffff | | Z | 1,2 |
| MOVWF | f | Move W to f | 1 | 00 0000 1fff ffff | | | |
| NOP | - | No Operation | 1 | 00 0000 0xx0 0000 | | | |
| RLF | f, d | Rotate Left f through Carry | 1 | 00 1101 dfff ffff | | C | 1,2 |
| RRF | f, d | Rotate Right f through Carry | 1 | 00 1100 dfff ffff | | C | 1,2 |
| SUBWF | f, d | Subtract W from f | 1 | 00 0010 dfff ffff | | C,DC,Z | 1,2 |
| SWAPF | f, d | Swap nibbles in f | 1 | 00 1110 dfff ffff | | | 1,2 |
| XORWF | f, d | Exclusive OR W with f | 1 | 00 0110 dfff ffff | | Z | 1,2 |
| **BIT-ORIENTED FILE REGISTER OPERATIONS** | | | | | | | |
| BCF | f, b | Bit Clear f | 1 | 01 00bb bfff ffff | | | 1,2 |
| BSF | f, b | Bit Set f | 1 | 01 01bb bfff ffff | | | 1,2 |
| BTFSC | f, b | Bit Test f, Skip if Clear | 1 (2) | 01 10bb bfff ffff | | | 3 |
| BTFSS | f, b | Bit Test f, Skip if Set | 1 (2) | 01 11bb bfff ffff | | | 3 |
| **LITERAL AND CONTROL OPERATIONS** | | | | | | | |
| ADDLW | k | Add literal and W | 1 | 11 111x kkkk kkkk | | C,DC,Z | |
| ANDLW | k | AND literal with W | 1 | 11 1001 kkkk kkkk | | Z | |
| CALL | k | Call subroutine | 2 | 10 0kkk kkkk kkkk | | | |
| CLRWDT | - | Clear Watchdog Timer | 1 | 00 0000 0110 0100 | | TO,PD | |
| GOTO | k | Go to address | 2 | 10 1kkk kkkk kkkk | | | |
| IORLW | k | Inclusive OR literal with W | 1 | 11 1000 kkkk kkkk | | Z | |
| MOVLW | k | Move literal to W | 1 | 11 00xx kkkk kkkk | | | |
| RETFIE | - | Return from interrupt | 2 | 00 0000 0000 1001 | | | |
| RETLW | k | Return with literal in W | 2 | 11 01xx kkkk kkkk | | | |
| RETURN | - | Return from Subroutine | 2 | 00 0000 0000 1000 | | | |
| SLEEP | - | Go into standby mode | 1 | 00 0000 0110 0011 | | TO PD | |
| SUBLW | k | Subtract W from literal | 1 | 11 110x kkkk kkkk | | C,DC,Z | |
| XORLW | k | Exclusive OR literal with W | 1 | 11 1010 kkkk kkkk | | Z | |

Note 1: When an I/O register is modified as a function of itself ( e.g., MOVF PORTB, 1 ), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 Module.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles The second cycle is executed as a NOP

## Periféricos

### Puertos de entrada/salida:

Estos microcontroladores cuentan con varios puertos de propósito general por donde podemos leer o escribir datos con niveles TTL

Son puertos programables en donde podemos decidir cual o cuales de sus bits serán salidas (para escribir datos) o cuales serán entradas (para leer datos).

Como estos puertos son programables, entonces existen dos registros asociados a cada puerto: uno para poder programar como entrada o salida los bits y otro para poder leer o escribir los datos en el puerto.

Las características de los puertos y sus registros asociados se muestran a continuación:

**- Puerto A**

PORTA es un puerto que tiene solamente 6 bits (de RA0 a RA5). El pin RA4 puede ser una entrada de tipo Schmitt trigger o una salida de colector abierto. Su programación es en el registro TRISA, donde con un '0' programamos salidas y con un '1' programamos entradas. Los registros asociados son.

| REGISTRO | DIRECCIÓN | COMENTAROS |
|---|---|---|
| PORTA | 05h | 6 BITS DE DATOS |
| TRISA | 85h | 0=SALIDA, 1=ENTRADA |

**- Puerto B**

El puerto PORTB es de 8 bits bidireccionales que pueden ser programados como entradas o salidas por medio del registro TRISB. PORTB tiene dos funciones más particulares en su modo de puerto general de entradas y salidas: por un lado, cada terminal de PORTB tiene internamente un circuito de pull-up que se puede habilitar limpiando el bit $\overline{RBPU}$ del registro OPTION y que es válido solamente para cuando el puerto esta configurado como entradas, y por otro lado, se puede generar interrupción si se presenta un cambio en cualquiera de los 4 bits más altos del puerto (RB7-RB4), siempre y cuando estén programados como entradas Los registros asociados a este puerto son.

| REGISTRO | DIRECCIÓN | COMENTAROS |
|---|---|---|
| PORTB | 05h | 8 BITS DE DATOS |
| TRISB | 86h | 0=SALIDA, 1=ENTRADA |
| OPTION | 81h | BIT $\overline{RBPU}$ |

**- Puerto C**

PORTC es un puerto bidireccional de 8 bits que pueden ser programados individualmente como salidas o entradas a través de TRISC. Los registros asociados a este puerto son.

| REGISTRO | DIRECCIÓN | COMENTAROS |
|---|---|---|
| PORTC | 07h | 8 BITS DE DATOS |
| TRISC | 87h | 0=SALIDA, 1=ENTRADA |

**- Puerto D**

El puerto PORTD es de 8 bits que se pueden configurar individualmente como salidas o entradas por medio del registro TRISD. Las entradas son de tipo Schmitt trigger y sus registros asociados son.

| REGISTRO | DIRECCIÓN | COMENTAROS |
|---|---|---|
| PORTD | 08h | 8 BITS DE DATOS |
| TRISD | 88h | 0=SALIDA, 1=ENTRADA |

Este puerto puede ser utilizado con la función de puerto paralelo esclavo. Esta función se verá más adelante.

**- Puerto E**

El puerto PORTE es de 3 bits solamente RE0, RE1 y RE2. Estos bits se pueden configurar a través del registro TRISE y las entradas son de tipo Schmitt trigger Sus registros asociados son·

| REGISTRO | DIRECCIÓN | COMENTAROS |
|---|---|---|
| PORTE | 09h | 3 BITS DE DATOS |
| TRISE | 89h | 0=SALIDA, 1=ENTRADA |

Como este puerto está multiplexado con las entradas analógicas del convertidor analógico-digital, entonces la operación de estos pines depende de la selección de los bits de control del registro ADCON1 que se verá más adelante.

Por otro lado, tanto los pines de PORTE como el registro TRISE son utilizados en la función de puerto paralelo esclavo de PORTD.

**- Puerto Paralelo Esclavo**

El puerto PORTD puede operar como un puerto paralelo esclavo de 8 bits. Esta función nos permite tener una interfase directa con un bus de datos de un microprocesador de 8 bits. En este modo tenemos una lectura y escritura asíncrona del puerto PORTD.

Para que PORTD trabaje en el modo de puerto paralelo esclavo, es necesario prender el bit *PSPMODE* del registro TRISE y programar lo pines de PORTE como entradas digitales en el mismo registro y como entradas/salidas digitales en el registro ADCON1 (que se utiliza para la función del convertidor analógico-digital).

Al habilitar este modo, los pines de PORTE se convierten en señales de control de lectura, escritura y selección:

-lectura: RE0/$\overline{RD}$
-escritura. RE1/$\overline{WR}$
-selección: RE2/$\overline{CS}$

Existen dos latches de 8 bits: uno para los datos de salida y otro para los datos de entrada. Para tener un control de la lectura y escritura de datos (dado que tienen la misma dirección) el registro TRISE cuenta con 3 bits de control.

A continuación se muestra el registro TRISE:

Registro **TRISE**

| 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 |
| bit7 | | | | | | | bit0 |

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7 : **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

bit 6: **OBF**: Output Buffer Full Status bit
1 = The output buffer still holds a previously written word
0 = The output buffer has been read

bit 5 **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

bit 4: **PSPMODE**, Parallel Slave Port Mode Select bit
1 = Parallel slave port mode
0 = General purpose I/O mode

bit 3 **Unimplemented:** Read as '0'

**PORTE Data Direction Bits**

bit 2: **Bit2**: Direction Control bit for pin RE2/$\overline{CS}$/AN7
1 = Input
0 = Output

bit 1: **Bit1**: Direction Control bit for pin RE1/$\overline{WR}$/AN6
1 = Input
0 = Output

bit 0: **Bit0**: Direction Control bit for pin RE0/$\overline{RD}$/AN5
1 = Input
0 = Output

Cuando se completa una operación de escritura o lectura se prende el bit de bandera *PSPIF* del registro PIR1 y nos puede generar una interrupción si el bit *PSPIE* del registro PIE1 está habilitado.

Los registros asociados al modo de puerto paralelo esclavos son .los que se muestran a continuación:

| Nombre | Direc. | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| PORTD | 08 | PORTD | | | | | | | |
| PORTE | 09 | | | | | | RE2 | RE1 | RE0 |
| TRISE | 89 | IBF | OBF | IBOV | PSPMODE | | TRISE2 | TRISE1 | TRISE0 |
| PIR1 | 0C | PSPIF | | | | | | | |
| PIE1 | 8C | PSPIE | | | | | | | |

**Temporizador 0 (TMR0)**

El temporizador 0 tiene la función de temporizador y de contador con las siguientes características:

- Temporizador y contador de 8 bits
- Se puede leer y escribir
- Genera interrupción cuando se da un sobreflujo de FFh a 00h
- Tiene una preescala que se puede programar pero que es compartida con el WDT
- Se puede seleccionaren el registro OPTION, el reloj interno para modo temporizador o el reloj externo para modo contador
- En modo temporizador el módulo TMR0 se incrementa cada ciclo de instrucción
- En modo contador el TMR0 se incrementa con los flancos de subida o bajada según se programen en el registro OPTION.

Los registros asociados con el TMR0 son

| Nombre | Direc. | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| TMRO | 01 | TMRO | | | | | | | |
| INTCON | 0B/8B | GIE | | TOIE | | | TOIF | | |
| OPTION | 81 | | | TOCS | TOSE | PSA | PS2 | PS1 | PS0 |
| TRISA | 85 | | | | TRISA4 | | | | |

## Temporizador 1 (TMR1)

Este temporizador y contador tiene las siguientes características:

- Temporizador y contador de 16 bits
- Se puede leer y escribir
- Genera interrupción cuando se da un sobreflujo de FFFFh a 0000h
- Tiene una preescala que se puede programar
- Se puede seleccionar el reloj interno para modo temporizador o el reloj externo para modo contador
- En modo temporizador el módulo TMR1 se incrementa cada ciclo de instrucción
- En modo contador el TMR1 se incrementa con cualquier flanco de subida
- Este temporizador (TMR1) se puede apagar o prender
- Se puede conectar un oscilador tipo LP (cristal de baja potencia) entre las terminales RC0/T1OSO/T1CKI y RC1/T1OSI/CCP2
- Se puede programar el modo contador para que trabaje en modo síncrono o asíncrono
- En modo contador síncrono, el temporizador TRM1 no se incrementa durante el estado de SLEEP.
- En modo contador asíncrono el temporizador si sigue incrementandose durante el estado de SLEEP, y puede despertarlo cuando se genera un sobleflujo en TMR1.

Algunas características importantes de TMR1 se programan en el siguiente registro:

**T1CON: TIMER1 CONTROL REGISTER (Dirección 10h)**

| U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON |
| bit7 | | | | | | | bit0 |

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7-6: **Unimplemented:** Read as '0'

bit 5-4: **T1CKPS1:T1CKPS0** Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1 4 Prescale value
01 = 1:2 Prescale value
00 = 1 1 Prescale value

bit 3: **T1OSCEN**, Timer1 Oscillator Enable Control bit
1 = Oscillator is enabled
0 = Oscillator is shut off
Note. The oscillator inverter and feedback resistor are turned off to eliminate power drain

bit 2. **$\overline{\text{T1SYNC}}$**: Timer1 External Clock Input Synchronization Control bit

TMR1CS = 1
1 = Do not synchronize external clock input
0 = Synchronize external clock input

TMR1CS = 0
This bit is ignored Timer1 uses the internal clock when TMR1CS = 0

bit 1: **TMR1CS**, Timer1 Clock Source Select bit
1 = External clock from pin RC0/T1OSO/T1CKI (on the rising edge)
0 = Internal clock (Fosc/4)

bit 0. **TMR1ON**: Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

Los registros asociados con el TMR1 son:

| Nombre | Direc. | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| TMR1L | 0E | TMR1 | (Parte | Baja) | (LSb) | | | | |
| TMR1H | 0F | TMR1 | (Parte | Alta) | (MSb) | | | | |
| T1CON | 10 | | | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON |
| INTCON | 0B/8B | GIE | PEIE | | | | | | |
| PIR1 | 0C | | | | | | | | TMR1IF |
| PIE1 | 8C | | | | | | | | TMR1IE |

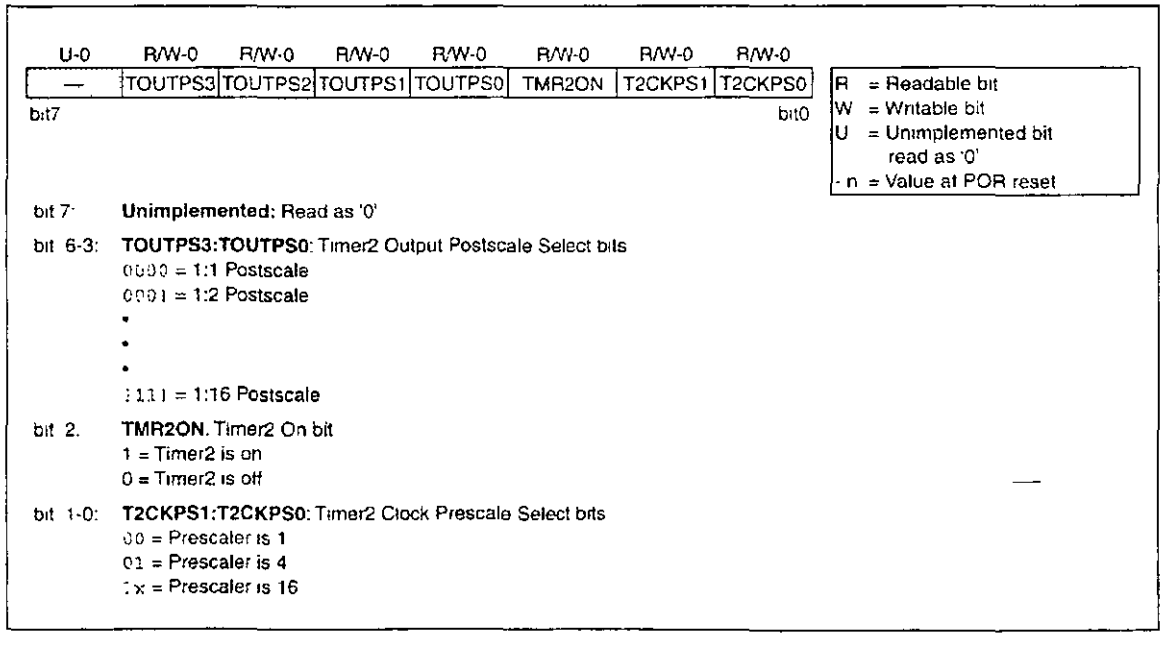
## Temporizador 2 (TMR2)

Este temporizador es muy sencillo ya que no cuenta con la opción de contador Sus principales características son:

- Temporizador de 8 bits.
- Se puede leer y escribir
- Se puede programar una preescala y una postescala.
- TMR2 se incrementa cada ciclo de instrucción.
- Cuenta con un registro de período PR2 con el que se pueden limitar los incrementos de TRM2
- Cuando se igualan las cuentas de PR2 y TMR2, entonces TMR2 se reinicializa en 00h.
- El registro PR2 se puede leer y escribir.
- La salida de sobreflujo (o igualdad) pasa por una postescala y puede generar interrupción.

La habilitación de TMR2 así como la preescala y la postescala se programan en el siguiente registro:

**T2CON: TIMER2 CONTROL REGISTER (Dirección 12h)**



Los registros asociados con el TMR1 son

| Nombre | Direc. | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| TMR2 | 11 | TMR2 | | | | | | | |
| PR2 | 92 | Timer2 | Period | Register | | | | | |
| T2CON | 12 | | TOUTPS3 | TOUTPS2 | TOUTPS1 | TOUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 |
| INTCON | 0B/8B | GIE | PEIE | | | | | | |
| PIR1 | 0C | | | | | | | TMR2IF | |
| PIE1 | 8C | | | | | | | TMR2IE | |

**Módulo de captura, comparación y PWM**

Los microcontroladores PIC16C7X tienen 2 módulos CCP (Capture/Compare/PWM) que consisten en un registro de 16 bits que puede operar como un registro de captura de 16 bits, un registro de 16 bits de comparación o una salida PWM (modulación por ancho de pulso).

El módulo CCP1 está formado por el registro CCPR1 que a su vez lo integran dos registros de 8 bits que se pueden leer o escribir la parte baja que es el registro CCPR1L y la alta dada por el registro CCPR1H.

Igualmente, el módulo CCP2 tiene un registro CCPR2 formado por los registros CCPR2L para la parte baja y CCPR2H para la parte alta.

Los módulos CCP1 y CCP2 tienen un funcionamiento similar y solo se explicará para el módulo CCP1 y se harán las distinciones cuando sea necesario.

Algunas características de funcionamiento de este módulo se pueden programar en los registros que se muestran a continuación

**CCP1CON REGISTER (Dirección 17h) / CCP2CON REGISTER (Dirección 1Dh)**

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

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n =Value at POR reset

bit 7-6: **Unimplemented:** Read as '0'

bit 5-4 **CCPxX:CCPxY** PWM Least Significant bits
Capture Mode: Unused
Compare Mode: Unused
PWM Mode: These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL

bit 3-0. **CCPxM3:CCPxM0** CCPx Mode Select bits
0000 = Capture/Compare/PWM off (resets CCPx module)
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode, set output on match (CCPxIF bit is set)
1001 = Compare mode, clear output on match (CCPxIF bit is set)
1010 = Compare mode, generate software interrupt on match (CCPxIF bit is set, CCPx pin is unaffected)
1011 = Compare mode, trigger special event (CCPxIF bit is set; CCP1 resets TMR1; CCP2 resets TMR1 and starts an A/D conversion (if A/D module is enabled))
11xx = PWM mode

**-Modo de Captura**

En este modo, los registros CCPR1H:CCPR1L capturan el valor de 16 bits de TMR1 cuando en la terminal RC2/CCP1 ocurre algún evento Este evento se puede programar en el registro CCP1CON, de tal forma que la captura se da cuando:

1. Se presenta un flanco de bajada
2. Se da un flanco de subida
3. Cada 4 flancos de subida
4. Cada 16 flancos de subida

Para poder trabajar en este modo, es necesario que TMR1 este programado para trabajar en modo temporizador (timer) o en modo contador sincronizado.

**-Modo de Comparación**

En este modo, los 16 bits del registro CCPR1 (CCPR1H:CCPR1L) son constantemente comparados con TRM1, y cuando se encuentra que los valores son iguales, entonces coloca un nivel en la terminal RC2/CCP1 El valor que presenta en la terminal es programable y puede ser

1. Coloca un nivel alto (5V)
2. Coloca un nivel bajo (0V)
3. Mantiene el nivel que tenia

Al igual que en el modo de captura, para poder trabajar en el modo de comparación es necesario que TMR1 este programado para trabajar en modo temporizador (timer) o en modo contador sincronizado

En este modo, cuando se presenta una igualdad en los registros CCPRx y TMR1, además de la acción programada para la terminal, se genera una señal especial de disparo que reinicializa en TMR1 para el caso de CCP1 o bien, para el caso de utilizar el CCP2, reinicializa el TMR1 y comienza una conversión del módulo del convertidor analógico/digital (si es que esta habilitado).

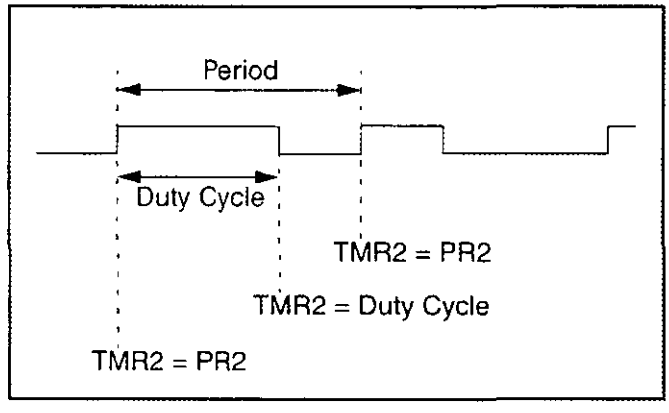
Los registros asociados con los modos de Captura y Comparación se muestran a continuación.

| Nombre | Direc. | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | 0B/8B | GIE | PEIE | | | | | | |
| PIR1 | 0C | | | | | | CCP1IF | | TMR1IF |
| PIR2 | 0D | | | | | | | | CCP2IF |
| PIE1 | 8C | | | | | | CCP1IE | | TMR1IE |
| PIE2 | 8D | | | | | | | | CCP2IE |
| TRISC | 87 | PORTC | Data | Direction | Register | | | | |
| TMR1L | 0E | Timer1 | Least | Signific. | Byte | | | | |
| TMR1H | 0F | Timer1 | Most | Signific. | Byte | | | | |
| T1CON | 10 | | | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON |
| CCPR1L | 15 | Capture/ | Compare | /PWM | Register | 1 (LSB) | | | |

| Nombre | Direc. | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| CCPR1H | 16 | Capture/ | Compare | /PWM | Register | 1 (MSB) | | | |
| CCP1CON | 17 | | | | | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 |
| CCPR2L | 1B | Capture/ | Compare | /PWM | Register | 2 (LSB) | | | |
| CCPR2H | 1C | Capture/ | Compare | /PWM | Register | 2 (MSB) | | | |
| CCP2CON | 17 | | | | | CCP2M3 | CCP2M2 | CCP2M1 | CCP2M0 |

**-Modo PWM**

Este modo nos permite tener una salida de modulación por ancho de pulso en la que podemos programar tanto el ciclo de trabajo como el periodo de la señal de salida. Se utilizan los registros CCP1CON, CCPR1L, CCPR1H, PR2, así como el TMR2. La siguiente figura muestra la salida PWM




El periodo de la salida PWM se especifica por medio del dato de 8 bits del registro PR2, y se calcula de la siguiente forma.

$$\mathbf{PWMperiod} = [(\mathbf{PR2}) + 1] \bullet 4 \bullet \mathbf{Tosc} \bullet (\mathbf{TMR2prescale\_value})$$

El ciclo de trabajo de la salida PWM se especifica con 10 bits. Los 8 bits más significativos están formados por el registro CCPR1L y los 2 menos significativos son los bits 5 y 4 del registro CCP1CON. Para calcular el ciclo de trabajo se utiliza la siguiente fórmula:

$$\mathbf{PWMduty\_cycle} = (\mathbf{CCPR1L} : \mathbf{CCP1CON} < 5 : 4 >) \bullet \mathbf{Tosc} \bullet (\mathbf{TMR2prescale\_value})$$

donde.

CCPR1L:CCP1CON<5:4>, son los 10 bits que especifican el ciclo de trabajo.

Cuando se igualan TMR2 y PR2 ocurren los siguientes eventos:

- TMR2 se inicializa en 00h
- La terminal CCP1 se pone en 1
- El ciclo de trabajo de PWM se pasa del registro CCPR1L al CCPR1H

Los pasos a seguir para operación en modo PWM son:

1- Se escribe en el registro PR2 el dato que nos genere el período de PWM requerido.
2- Se escribe en el registro CCPR1L y en los bits 5 y 4 del CCP1CON el dato para generar el ciclo de trabajo requerido.
3- Se programa la terminal CCP1 como salida por medio del bit 2 del registro TRISC.
4- Se selecciona la preescala de TMR2 y se habilita el temporizador 2 (Timer2) en el registro T2CON.
5- Se configura el módulo CCP1 en el registro CCP1CON para operar en modo PWM.

Hay que notar que se ha hecho referencia al módulo CCP1, pero que es válido para el módulo CCP2 y sus respectivos registros

Los registros asociados para trabajar en modo PWM son:

| Nombre | Direc. | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | 0B/8B | GIE | PEIE | | | | | | |
| PIR1 | 0C | | | | | | | TMR2IF | |
| PIR2 | 0D | | | | | | | | CCP2IF |
| PIE1 | 8C | | | | | | | TMR2IE | |
| PIE2 | 8D | | | | | | | | CCP2IE |
| TRISC | 87 | PORTC | Data | Direction | Register | | | | |
| TMR2 | 11 | TMR2 | | | | | | | |
| PR2 | 92 | Timer2 | Period | Register | | | | | |
| T2CON | 12 | | TOUTPS3 | TOUTPS2 | TOUTPS1 | TOUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 |
| CCPR1L | 15 | Timer2 | Duty | Cycle | Register | | | | |
| CCPR1H | 16 | Timer2 | Duty | Cycle | Register | (Slave) | | | |
| CCP1CON | 17 | | | CCP1X | CCP1Y | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 |
| CCPR2L | 1B | Timer2 | Duty | Cycle | Register | | | | |
| CCPR2H | 1C | Timer2 | Duty | Cycle | Register | (Slave) | | | |
| CCP2CON | 17 | | | CCP2X | CCP2Y | CCP2M3 | CCP2M2 | CCP2M1 | CCP2M0 |

**Puerto serial Síncrono (SSP)**

El módulo SSP es una interfase serial que puede operar en alguno de los siguientes modos:

- Interfase periférica serial (SPI)
- "Inter-Integrated Circuit" ($I^2C$)

Este puerto tiene dos registros especiales: SSPSTAT que es un registro donde vemos el estado del SSP y el registro SSPCON que es de control y sirve para programar las formas en que puede trabajar este módulo.

## SSPSTAT: SYNC SERIAL PORT STATUS REGISTER (Dirección 94h)

| U-0 | U-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| — | — | D/A | P | S | R/W | UA | BF |
| bit7 | | | | | | | bit0 |

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **D/Ā** Data/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

bit 4 **P** Stop bit ($I^2C$ mode only This bit is cleared when the SSP 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

bit 3 **S** Start bit ($I^2C$ mode only This bit is cleared when the SSP 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

bit 2 **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 valid from the address match to the next start bit, stop bit, or ACK bit.
1 = Read
0 = Write

bit 1 **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

bit 0 **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 = Transmit in progress SSPBUF is full
0 = Transmit complete, SSPBUF is empty

## SSPCON: SYNC SERIAL PORT CONTROL REGISTER (Dirección 14h)

| 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 |
| bit7 | | | | | | | bit0 |

R = Readable bit
W = Writable bit
U = Unimplemented bit read as '0'
- n = Value at POR reset

bit 7 **WCOL**: Write Collision Detect bit
1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

bit 6 **SSPOV**: Receive Overflow Detect bit

In SPI mode
1 = A new byte is received while the SSPBUF register is still holding the previous data In case of overflow, the data in SSPSR register is lost Overflow can only occur in slave mode The user must read the SSPBUF, even if only transmitting data to avoid setting overflow In master mode the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register
0 = No overflow

In $I^2C$ mode
1 = A byte is received while the SSPBUF register is still holding the previous byte SSPOV is a "don't care" in transmit mode SSPOV must be cleared in software in either mode.
0 = No overflow

bit 5 **SSPEN** Synchronous Serial Port Enable bit

In SPI mode
1 = Enables serial port and configures SCK, SDO, and SDI as serial port pins
0 = Disables serial port and configures these pins as I/O port pins

In $I^2C$ mode
1 = Enables the serial port and configures the SDA and SCL pins as serial port pins
0 = Disables serial port and configures these pins as I/O port pins
In both modes when enabled these pins must be properly configured as input or output

bit 4 **CKP**, Clock Polarity Select bit

In SPI mode
1 = Idle state for clock is a high level Transmit happens on falling edge, receive on rising edge
0 = Idle state for clock is a low level Transmit happens on rising edge receive on falling edge

In $I^2C$ mode
SCK release control
1 = Enable clock
0 = Holds clock low (clock stretch) (Used to ensure data setup time)

bit 3-0 **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 SS pin control enabled
0101 = SPI slave mode, clock = SCK pin SS pin control disabled SS can be used as I/O pin
0110 = $I^2C$ slave mode 7-bit address
0111 = $I^2C$ slave mode 10-bit address
1011 = $I^2C$ firmware controlled Master (slave idle)
1110 = $I^2C$ slave mode, 7-bit address with start and stop bit interrupts enabled
1111 = $I^2C$ slave mode 10-bit address with start and stop bit interrupts enabled

**-Modo Interfase Periférica Serial (SPI)**

En este modo podemos transmitir y recibir datos de 8 bits en forma serial, utilizando para esto tres terminales:

- Salida de dato serial (SDO) RC5/SDO
- Entrada de dato serial (SDI) RC4/SDI
- Reloj serial (SCK) RC3/SCK

Si se utiliza el modo esclavo, entonces se necesitará una terminal adicional·

- _Selección de modo esclavo($\overline{SS}$) RA5/$\overline{SS}$

Para este modo se pueden programar varias opciones de funcionamiento en el registro SSPCON Estas funciones pueden ser:

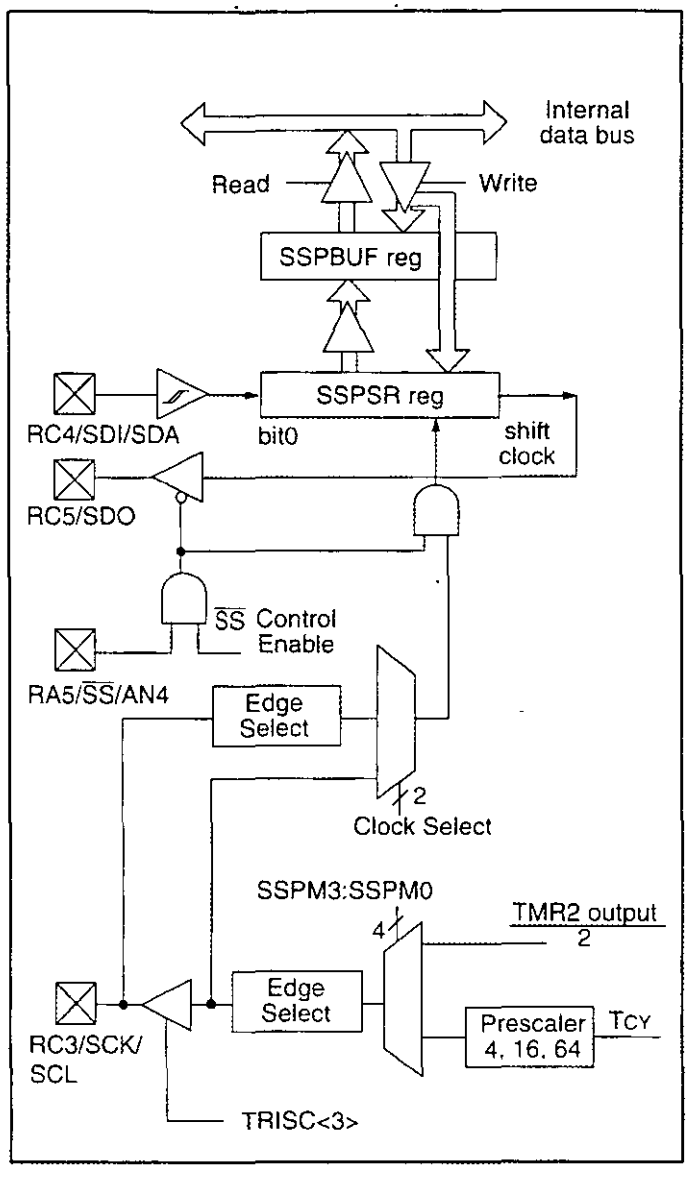
- Modo maestro (SCK es el reloj de salida)
- Modo esclavo (SCK es el reloj de entrada)
- Polaridad del reloj (la entrada o salida de datos se puede dar con el flanco de subida o de bajada del reloj SCK)
- La frecuencia del reloj (sólo en modo maestro)
- Selección de modo esclavo (sólo para modo esclavo)

El módulo SSP esta formado por un registro de corrimiento (SSPSR) para recibir o transmitir datos y un registro de buffer (SPPBUF) que es donde el microcontrolador coloca el dato a ser transmitido o lee el dato que se recibió.

Si se recibe un dato y los 8 bits ya se encuentran en el registro de buffer SSPBUF , entonces se prende el bit de bandera de buffer lleno (BF) en el registro de estado SSPSTAT. Cuando se lee el registro SSPBUF, el bit BF se limpia

Si se pretende escribir el registro SSPBUF durante un proceso de transmisión o recepción, entonces se habilita el bit de bandera de detección de colisión de escritura (WCOL) en el registro de control SSPCON.

En la siguiente figura se muestra un diagrama a bloques de este módulo en modo SPI (Interfase periférica serial)·




La terminal SS permite que se trabaje en un modo esclavo síncrono. Esto se programa en el registro SSPCON, de tal forma que cuando se coloca un nivel bajo en la terminal SS se habilita la transmisión y la recepción.

Los registros asociados con la operación de SPI son:

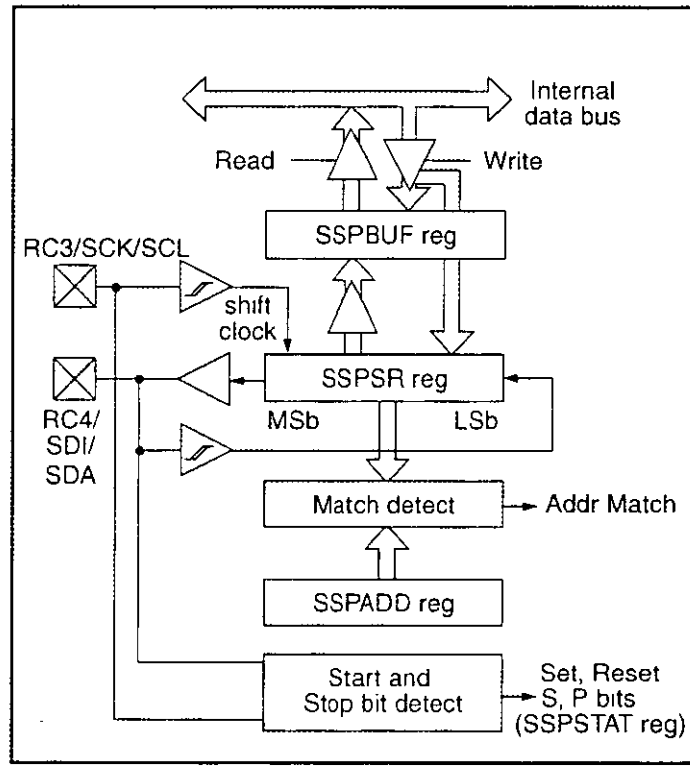
| Nombre | Direc. | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | 0B/8B | GIE | PEIE | | | | | | |
| PIR1 | 0C | | | | | SSPIF | | | |
| PIE1 | 8C | | | | | SSPIE | | | |
| SSPBUF | 13 | Buffer del SSP | | | | | | | |
| SSPCON | 14 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 |
| SSPSTAT | 94 | | | | | | | | BF |

**-Modo Inter-Integrated Circuit ($I^2C$)**

La interfase serial $I^2C$ fue desarrollada por Philips Corporation y consiste en un protocolo de comunicación que utiliza un bus de dos hilos Por medio de estos dos hilos, este protocolo puede transmitir y recibir datos de una forma segura.

El protocolo de comunicación $I^2C$ permite que exista uno o varios dispositivos trabajando como maestros y varios como esclavos. Cada dispositivo esclavo tiene una dirección, de tal forma que el maestro envía por el bus la dirección del dispositivo esclavo con el que se quiere comunicar y solamente entabla comunicación con este aun cuando todos los esclavos estén conectados al mismo bus.

En la siguiente figura se muestra el diagrama a bloques para el modo $I^2C$:




En este modo de operación la transmisión y recepción se hace por las siguientes dos terminales

- RC3/SCK/SCL: reloj
- RC4/SDI/SDA: datos

Para poder trabajar en modo $I^2C$ el módulo SSP utiliza 5 registros.

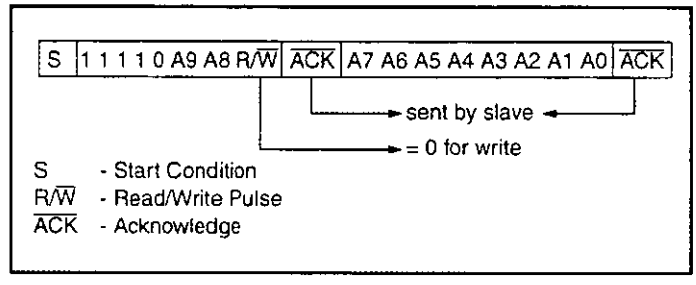
El SSPCON que es el registro de control, y permite seleccionar varios modos de funcionamiento para $I^2C$:

- $I^2C$ en modo esclavo (dirección de 7 bits)
- $I^2C$ en modo esclavo (dirección de 10 bits)
- $I^2C$ en modo esclavo (dirección de 7 bits), con soporte habilitado para modo maestro
- $I^2C$ en modo esclavo (dirección de 10 bits), con soporte habilitado para modo maestro
- $I^2C$ en modo maestro, soportando la habilitación para modo esclavo en estado ocioso

El SSPSTAT es el registro donde podemos observar el estado de la transferencia de datos. Podemos detectar los bits de inicio o paro (START o STOP), se especifica si la información recibida corresponde a una dirección o a un dato, si el siguiente byte es la terminación de la dirección de 10 bits o si el dato que se está transfiriendo es de lectura o de escritura

Los registros SSPBUF y SSPSR tienen la misma función que para el modo SPI en el registro SSPBUF se encuentra el dato que se recibió o el que se quiere transmitir a través del registro de corrimiento SSPSR.

El registro SSPADD contiene la dirección del esclavo. En el modo de 10 bits el usuario necesita escribir el byte alto de la dirección en el siguiente formato:




Después de que el byte alto ha sido reconocido, es necesario cargar la parte baja de la dirección (A7 – A0)

**- Módulo $I^2C$ en modo esclavo**

En el modo esclavo, las terminales SCL y SDA se configuran como entradas.

El funcionamiento de este periférico es como sigue:

-el SSP espera por las señal de START
-cuando se da la señal de START, entonces los 8 bits son recorridos al registro SSPSR
-el dato de SSPSR es comparado con el registro SSPADD
-si son iguales y los bits BF y SSPOV están en cero, entonces SSPSR se carga en SSPBUF
-se prende el bit de buffer lleno (BF)
-se genera el pulso de reconocimiento $\overline{ACK}$
-se prende la bandera de interrupción SSPIF y se genera una interrupción si está habilitada

A continuación se muestran los diagramas de tiempos para la recepción y la transmisión en modo esclavo:

**DIAGRAMA DE TIEMPOS PARA LA RECEPCIÓN (Dirección de 7 bits)**

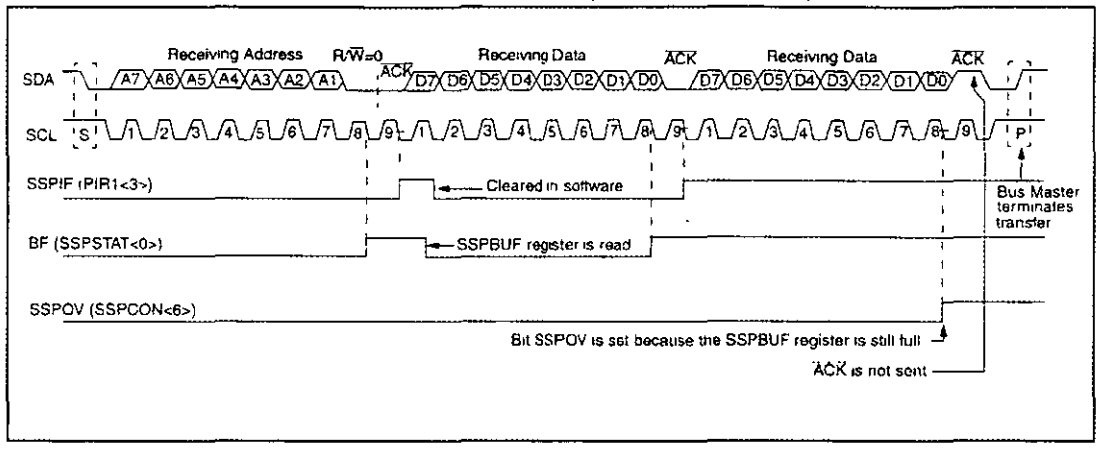
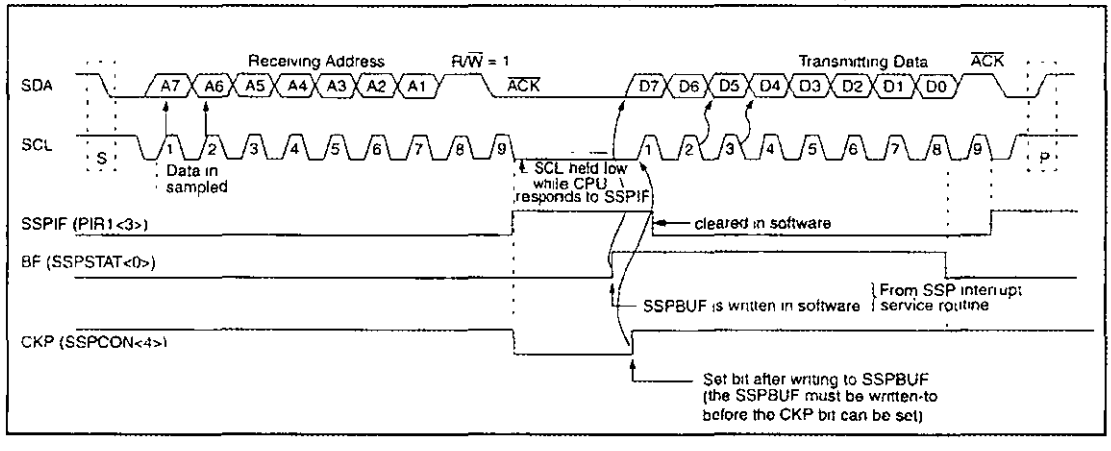



**DIAGRAMA DE TIEMPOS PARA LA TRANSMISIÓN (Dirección de 7 bits)**




## - Módulo $I^2C$ en modo maestro

El módulo $I^2C$ no puede trabajar completamente en modo maestro, ya que en determinado momento el puerto puede mandar datos (modo maestro) pero también puede recibirlos (modo esclavo) Es por esto que en el registro SSPCON sólo se pueden programar el modo maestro junto con el modo esclavo ocioso o bien el modo maestro habilitado con el modo esclavo activo

Para este modo de operación las líneas SDA y SCL deben ser programadas como salidas en el registro TRISC.

Los registros asociados con la operación del módulo SSP son:

| Nombre | Direc. | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | 0B/8B | GIE | PEIE | | | | | | |
| PIR1 | 0C | | | | | SSPIF | | | |
| PIE1 | 8C | | | | | SSPIE | | | |
| SSPBUF | 13 | Buffer | | | | | | | |
| SSPADD | 93 | Register | Address | ($I^2C$ mode) | | | | | |
| SSPCON | 14 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 |
| SSPSTAT | 94 | | | $D/\overline{A}$ | P | S | $R/\overline{W}$ | UA | BF |

**Transmisor y receptor universal síncrono y asíncrono (USART)**

Este módulo permite la comunicación serial con dispositivos tales como terminales, computadoras personales, circuitos integrados como convertidores A/D o D/A, memorias seriales EEPROM's y otros dispositivos más.

Este módulo USART puede configurarse para trabajar en los siguientes modos:

- asíncrono (full duplex)
- síncrono como maestro (half duplex)
- síncrono como esclavo- (half duplex)

la transmisión y recepción de datos se realiza por las terminales:

-RC6/TX/CK
-RC7/RX/DT

En este módulo se utilizan dos registros para control y comprobar el estado. Estos registro son.

## TXSTA: TRANSMIT STATUS AND CONTROL REGISTER

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

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7. **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)

bit 6 **TX9** 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5. **TXEN**, Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled
Note SREN/CREN overrides TXEN in SYNC mode

bit 4 **SYNC** USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **Unimplemented**, Read as '0'

bit 2 **BRGH** High Baud Rate Select bit

Asynchronous mode
1 = High speed

> **Note:** For the PIC16C73/73A/74/74A, the asynchronous high speed mode (BRGH = 1) may experience a high rate of receive errors. It is recommended that BRGH = 0. If you desire a higher baud rate than BRGH = 0 can support, refer to the device errata for additional information, or use the PIC16C76/77.

0 = Low speed

Synchronous mode
Unused in this mode

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

bit 0. **TX9D** 9th bit of transmit data. Can be parity bit.

## RCSTA: RECEIVE STATUS AND CONTROL REGISTER

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | R-0 | R-0 | R-x |
|---|---|---|---|---|---|---|---|
| SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D |
| bit7 | | | | | | | bit0 |

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7 **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

bit 6. **RX9** 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **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
Unused in this mode

bit 4 **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

bit 3. **Unimplemented:** Read as '0'

bit 2 **FERR**, Framing Error bit
1 = Framing error (Can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

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

bit 0 **RX9D**: 9th bit of received data (Can be parity bit)

Además de estos registros, tiene dos registros de datos el TXREG que es donde se carga el dato que se va a transmitir y el RCREG que es donde se almacena el dato recibido Asimismo, existe un registro SPBRG que se utiliza para programar el baudaje de trabajo

**-USART en modo asíncrono**

En este modo de operación se utiliza el formato NRZ (no retorno por cero) con un bit de inicio, 8 o 9 bits de datos y un bit de paro. La comunicación se realiza por medio de las terminales·

- TX para transmisión
- RX para recepción

Existe un generador de baudaje que puede producir un baudaje estándar a partir del oscilador del microcontrolador y que se puede programar de acuerdo a la siguiente fórmula:

| SYNC | BRGH=0 (Low Speed) | BRGH=1 (High Speed) |
|---|---|---|
| 0 | Baud Rate = Fosc/(64(SPBRG+1)) | Baud Rate = Fosc/(16(SPBRG +1)) |

Las siguientes tablas muestran algunos valores del baudaje para varios valores de frecuencia del oscilador y del registro SPBRG.

**BAUDAJE PARA EL MODO ASÍNCRONO (BRGH=0)**

| BAUD RATE (K) | FOSC = 20 MHz | | | 16 MHz | | | 10 MHz | | | 7 15909 MHz | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KBAUD | % ERROR | SPBRG value (decimal) | KBAUD | % ERROR | SPBRG value (decimal) | KBAUD | % ERROR | SPBRG value (decimal) | KBAUD | % ERROR | SPBRG value (decimal) |
| 0 3 | NA | - | - | NA | - | - | NA | - | - | NA | - | - |
| 1 2 | 1 221 | +1 73 | 255 | 1 202 | +0 16 | 207 | 1 202 | +0.16 | 129 | 1.203 | +0 23 | 92 |
| 2 4 | 2.404 | +0.16 | 129 | 2 404 | +0 16 | 103 | 2.404 | +0.16 | 64 | 2.380 | -0.83 | 46 |
| 9 6 | 9.469 | -1 36 | 32 | 9 615 | +0 16 | 25 | 9 766 | +1 73 | 15 | 9 322 | -2 90 | 11 |
| 19 2 | 19 53 | +1 73 | 15 | 19.23 | +0 16 | 12 | 19.53 | +1 73 | 7 | 18 64 | -2 90 | 5 |
| 76.8 | 78 13 | +1 73 | 3 | 83.33 | +8 51 | 2 | 78 13 | +1 73 | 1 | NA | - | - |
| 96 | 104 2 | +8 51 | 2 | NA | - | - | NA | - | - | NA | - | - |
| 300 | 312 5 | +4 17 | 0 | NA | - | - | NA | - | - | NA | - | - |
| 500 | NA | - | - | NA | - | - | NA | - | - | NA | - | - |
| HIGH | 312 5 | - | 0 | 250 | - | 0 | 156.3 | - | 0 | 111.9 | - | 0 |
| LOW | 1 221 | - | 255 | 0 977 | - | 255 | 0.6104 | - | 255 | 0 437 | - | 255 |

| BAUD RATE (K) | FOSC = 5 0688 MHz | | | 4 MHz | | | 3.579545 MHz | | | 1 MHz | | | 32.768 kHz | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KBAUD | % ERROR | SPBRG value (decimal) | KBAUD | % ERROR | SPBRG value (decimal) | KBAUD | % ERROR | SPBRG value (decimal) | KBAUD | % ERROR | SPBRG value (decimal) | KBAUD | % ERROR | SPBRG value (decimal) |
| 0.3 | 0 31 | +3 13 | 255 | 0 3005 | -0 17 | 207 | 0 301 | +0 23 | 185 | 0 300 | +0.16 | 51 | 0,256 | -14 67 | 1 |
| 1 2 | 1 2 | 0 | 65 | 1 202 | +1 67 | 51 | 1 190 | -0 83 | 46 | 1 202 | +0 16 | 12 | NA | - | - |
| 2 4 | 2 4 | 0 | 32 | 2.404 | +1.67 | 25 | 2 432 | +1 32 | 22 | 2 232 | -6.99 | 6 | NA | - | - |
| 9.6 | 9.9 | +3 13 | 7 | NA | - | - | 9 322 | -2 90 | 5 | NA | - | - | NA | - | - |
| 19 2 | 19 8 | +3.13 | 3 | NA | - | - | 18 64 | -2 90 | 2 | NA | - | - | NA | - | - |
| 76 8 | 79 2 | +3 13 | 0 | NA | - | - | NA | - | - | NA | - | - | NA | - | - |
| 96 | NA | - | - | NA | - | - | NA | - | - | NA | - | - | NA | - | - |
| 300 | NA | - | - | NA | - | - | NA | - | - | NA | - | - | NA | - | - |
| 500 | NA | - | - | NA | - | - | NA | - | - | NA | - | - | NA | - | - |
| HIGH | 79 2 | - | 0 | 62 500 | - | 0 | 55.93 | - | 0 | 15 63 | - | 0 | 0.512 | - | 0 |
| LOW | 0 3094 | - | 255 | 3.906 | - | 255 | 0 2185 | - | 255 | 0 0610 | - | 255 | 0.0020 | - | 255 |

**BAUDAJE PARA EL MODO ASÍNCRONO (BRGH=1)**

| BAUD RATE (K) | FOSC = 20 MHz KBAUD | % ERROR | SPBRG value (decimal) | 16 MHz KBAUD | % ERROR | SPBRG value (decimal) | 10 MHz KBAUD | % ERROR | SPBRG value (decimal) | 7.16 MHz KBAUD | % ERROR | SPBRG value (decimal) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 9 6 | 9 615 | +0 16 | 129 | 9 615 | +0 16 | 103 | 9 615 | +0.16 | 64 | 9.520 | -0 83 | 46 |
| 19 2 | 19.230 | +0 16 | 64 | 19.230 | +0.16 | 51 | 18.939 | -1.36 | 32 | 19 454 | +1 32 | 22 |
| 38 4 | 37 878 | -1.36 | 32 | 38 461 | +0.16 | 25 | 39.062 | +1.7 | 15 | 37.286 | -2.90 | 11 |
| 57 6 | 56.818 | -1 36 | 21 | 58 823 | +2 12 | 16 | 56.818 | -1 36 | 10 | 55.930 | -2 90 | 7 |
| 115 2 | 113 636 | -1 36 | 10 | 111 111 | -3 55 | 8 | 125 | +8 51 | 4 | 111.860 | -2 90 | 3 |
| 250 | 250 | 0 | 4 | 250 | 0 | 3 | NA | - | - | NA | - | - |
| 625 | 625 | 0 | 1 | NA | - | - | 625 | 0 | 0 | NA | - | - |
| 1250 | 1250 | 0 | 0 | NA | - | - | NA | - | - | NA | - | - |

| BAUD RATE (K) | FOSC = 5.068 MHz KBAUD | % ERROR | SPBRG value (decimal) | 4 MHz KBAUD | % ERROR | SPBRG value (decimal) | 3 579 MHz KBAUD | % ERROR | SPBRG value (decimal) | 1 MHz KBAUD | % ERROR | SPBRG value (decimal) | 32 768 kHz KBAUD | % ERROR | SPBRG value (decimal) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 9.6 | 9 6 | 0 | 32 | NA | - | - | 9 727 | +1.32 | 22 | 8 928 | -6 99 | 6 | NA | - | - |
| 19 2 | 18.645 | -2 94 | 16 | 1.202 | +0 17 | 207 | 18 643 | -2 90 | 11 | 20 833 | +8 51 | 2 | NA | - | - |
| 38 4 | 39 6 | +3 12 | 7 | 2.403 | +0 13 | 103 | 37 286 | -2 90 | 5 | 31 25 | -18 61 | 1 | NA | - | - |
| 57.6 | 52 8 | -8.33 | 5 | 9 615 | +0 16 | 25 | 55 930 | -2 90 | 3 | 62.5 | +8 51 | 0 | NA | - | - |
| 115.2 | 105 6 | -8 33 | 2 | 19 231 | +0 16 | 12 | 111.860 | -2 90 | 1 | NA | - | - | NA | - | - |
| 250 | NA | - | - | NA | - | - | 223 721 | -10 51 | 0 | NA | - | - | NA | - | - |
| 625 | NA | - | - | NA | - | - | NA | - | - | NA | - | - | NA | - | - |
| 1250 | NA | - | - | NA | - | - | NA | - | - | NA | - | - | NA | - | - |

-Transmisión en modo asíncrono

Los pasos para programar el USART como transmisor asíncrono son:

1.- Inicializar el registro SPBRG con el valor apropiado para el baudaje seleccionado Si se escoge un baudaje de alta velocidad, entonces se debe poner el bit BRGH=1.
2 - Habilitar el puerto serial asíncrono, configurando los bits SYNC=0 y SPEN=1
3.- Se debe prender el bit TXIE si se decide utilizar interrupciones
4 - Si se decide utilizar 9 datos de transmisión, entonces se debe prender el bit TX9
5.- Habilitamos la transmisión prendiendo el bit TXEN
6.- Si se seleccionó 9 bits de datos de transmisión, entonces se debe cargar el noveno bit en el bit TX9D del registro TXSTA
7 - Cargar el dato que se transmitirá en el registro TXREG Esto inicia la transmisión

Los registros asociados para la transmisión asíncrona son

| Nombre | Direc. | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| PIR1 | 0C | | | | TXIF | | | | |
| RCSTA | 18 | SPEN | | | | | | | |
| TXREG | 19 | TX7 | TX6 | TX5 | TX4 | TX3 | TX2 | TX1 | TX0 |
| PIE1 | 8C | | | | TXIE | | | | |
| TXSTA | 98 | | TX9 | TXEN | SYNC | | BRGH | TRMT | TX9D |
| SPBRG | 99 | Registro | de | baudaje | | | | | |

-Recepción en modo asíncrono

Los pasos a seguir para programar el USART como receptor asíncrono son.

1 - Inicializar el registro SPBRG con el valor apropiado para el baudaje seleccionado. Si se escoge un baudaje de alta velocidad, entonces se debe poner el bit BRGH=1
2 - Habilitar el puerto serial asíncrono, configurando los bits SYNC=0 y SPEN=1
3 - Se debe prender el bit RCIE si se decide utilizar interrupciones
4.- Si se decide utilizar 9 datos de recepción, entonces se debe prender el bit RX9
5.- Habilitamos la recepción prendiendo el bit CREN
6.- Cuando se completa una recepción se prende el bit de bandera RCIF y se puede generar una interrupción si está habilitado el bit RCIE
7 - Si está habilitada la recepción de 9 bits, entonces se toma el noveno bit de datos que queda en el bit RX9D del registro RCSTA y se determina si ocurrió un error durante la recepción
8.- Se leen del registro RCREG los 8 bits de datos recibidos
9 - Si ocurrió algún error, entonces se limpia el bit CREN

Los registros asociados para la recepción asíncrona son.

| Nombre | Direc. | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| PIR1 | 0C | | | RCIF | | | | | |
| RCSTA | 18 | SPEN | RX9 | | CREN | | FERR | OERR | RX9D |
| RCREG | 1A | RX7 | RX6 | RX5 | RX4 | RX3 | RX2 | RX1 | RX0 |
| PIE1 | 8C | | RCIE | | | | | | |
| TXSTA | 98 | | | | SYNC | | BRGH | | |
| SPBRG | 99 | Registro | de | baudaje | | | | | |

**-USART en modo maestro síncrono**

En este modo la transmisión y la recepción no se puede realizar al mismo tiempo. Esto es, cuando se está transmitiendo la recepción se inhibe y viceversa. Se dice que este tipo de comunicación se realiza de manera half-duplex

La comunicación se realiza a través de las terminales.

- CK como línea de reloj
- DT como línea de datos

Al igual que en el modo asíncrono, el baudaje de trabajo también es generado a partir de la frecuencia de oscilación del microcontrolador y se puede calcular de acuerdo con la siguiente fórmula:

| SYNC | BRGH=* |
|---|---|
| 1 | Baud Rate = Fosc/(4(SPBRG+1)) |

La siguiente tabla muestran algunos valores del baudaje para varios valores de frecuencia del oscilador y del registro SPBRG:

| BAUD RATE (K) | FOSC = 20 MHz KBAUD | % ERROR | SPBRG value (decimal) | 16 MHz KBAUD | % ERROR | SPBRG value (decimal) | 10 MHz KBAUD | % ERROR | SPBRG value (decimal) | 7 15909 MHz KBAUD | % ERROR | SPBRG value (decimal) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 3 | NA | - | - | NA | - | - | NA | - | - | NA | - | - |
| 1.2 | NA | - | - | NA | - | - | NA | - | - | NA | - | - |
| 2.4 | NA | - | - | NA | - | - | NA | - | - | NA | - | - |
| 9 6 | NA | - | - | NA | - | - | 9.766 | +1 73 | 255 | 9.622 | +0 23 | 185 |
| 19 2 | 19 53 | +1 73 | 255 | 19.23 | +0 16 | 207 | 19.23 | +0 16 | 129 | 19.24 | +0.23 | 92 |
| 76 8 | 76 92 | +0 16 | 64 | 76 92 | +0.16 | 51 | 75.76 | -1 36 | 32 | 77.82 | +1.32 | 22 |
| 96 | 96.15 | +0.16 | 51 | 95.24 | -0 79 | 41 | 96.15 | +0.16 | 25 | 94.20 | -1.88 | 16 |
| 300 | 294.1 | -1.96 | 16 | 307.69 | +2 56 | 12 | 312 5 | +4 17 | 7 | 298.3 | -0.57 | 5 |
| 500 | 500 | 0 | 9 | 500 | 0 | 7 | 500 | 0 | 4 | NA | - | - |
| HIGH | 5000 | - | 0 | 4000 | - | 0 | 2500 | - | 0 | 1789.8 | - | 0 |
| LOW | 19 53 | - | 255 | 15.625 | - | 255 | 9 766 | - | 255 | 6 991 | - | 255 |

| BAUD RATE (K) | FOSC = 5.0688 MHz KBAUD | % ERROR | SPBRG value (decimal) | 4 MHz KBAUD | % ERROR | SPBRG value (decimal) | 3 579545 MHz KBAUD | % ERROR | SPBRG value (decimal) | 1 MHz KBAUD | % ERROR | SPBRG value (decimal) | 32.768 kHz KBAUD | % ERROR | SPBRG value (decimal) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 3 | NA | - | - | NA | - | - | NA | - | - | NA | - | - | 0 303 | +1.14 | 26 |
| 1 2 | NA | - | - | NA | - | - | NA | - | - | 1.202 | +0.16 | 207 | 1.170 | -2.48 | 6 |
| 2 4 | NA | - | - | NA | - | - | NA | - | - | 2 404 | +0.16 | 103 | NA | - | - |
| 9 6 | 9.6 | 0 | 131 | 9.615 | +0.16 | 103 | 9.622 | +0.23 | 92 | 9.615 | +0.16 | 25 | NA | - | - |
| 19 2 | 19 2 | 0 | 65 | 19 231 | +0 16 | 51 | 19 04 | -0.83 | 46 | 19.24 | +0 16 | 12 | NA | - | - |
| 76.8 | 79.2 | -3.13 | 15 | 76.923 | +0.16 | 12 | 74.57 | -2.90 | 11 | 83 34 | +8 51 | 2 | NA | - | - |
| 96 | 97.48 | +1 54 | 12 | 1000 | +4.17 | 9 | 99.43 | +3.57 | 8 | NA | - | - | NA | - | - |
| 300 | 316 8 | +5 60 | 3 | NA | - | - | 298.3 | -0 57 | 2 | NA | - | - | NA | - | - |
| 500 | NA | - | - | NA | - | - | NA | - | - | NA | - | - | NA | - | - |
| HIGH | 1267 | - | 0 | 100 | - | 0 | 894 9 | - | 0 | 250 | - | 0 | 8 192 | - | 0 |
| LOW | 4 950 | - | 255 | 3.906 | - | 255 | 3 496 | - | 255 | 0.9766 | - | 255 | 0 032 | - | 255 |

-Transmisión en modo maestro síncrono

Los pasos a seguir para programar el USART como transmisor maestro síncrono son:

1 - Inicializar el registro SPBRG con el valor apropiado para el baudaje seleccionado
2 - Habilitar el puerto serial en modo maestro síncrono, configurando los bits SYNC=1, SPEN=1 y CSRC=1
3.- Se debe prender el bit TXIE si se decide utilizar interrupciones
4.- Si se decide utilizar 9 datos de transmisión, entonces se debe prender el bit TX9
5.- Habilitamos la transmisión prendiendo el bit TXEN
6.- Si se seleccionó 9 bits de datos de transmisión, entonces se debe cargar el noveno bit que será transmitido en el bit TX9D del registrò TXSTA
7 - Cargar el dato que se transmitirá en el registro TXREG. Esto inicia la transmisión

Los registros asociados para la transmisión en modo maestro síncrono son:

| Nombre | Direc. | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| PIR1 | 0C | | | | TXIF | | | | |
| RCSTA | 18 | SPEN | | | | | | | |
| TXREG | 19 | TX7 | TX6 | TX5 | TX4 | TX3 | TX2 | TX1 | TX0 |
| PIE1 | 8C | | | | TXIE | | | | |
| TXSTA | 98 | CSRC | TX9 | TXEN | SYNC | | | TRMT | TX9D |

| SPBRG | 99 | Registro de baudaje |
|---|---|---|

-Recepción en modo maestro síncrono

Los pasos a seguir para programar el USART como receptor maestro síncrono son:

1.- Inicializar el registro SPBRG con el valor apropiado para el baudaje seleccionado
2 - Habilitar el puerto serial en modo maestro síncrono, configurando los bits SYNC=1, SPEN=1 y CSRC=1
3.- Se deben limpiar los bits CREN y SREN
4.- Se debe prender el bit RCIE si se decide utilizar interrupciones
5.- Si se decide utilizar 9 datos de recepción, entonces se debe prender el bit RX9
6.- Si se requiere una sola recepción se debe prender el bit SREN. Para recepción continua se prende el bit CREN
7 - Cuando se completa una recepción se prende el bit de bandera RCIF y se puede generar una interrupción si está habilitado el bit RCIE
8 - Si está habilitada la recepción de 9 bits, entonces se toma el noveno bit de datos que queda en el bit RX9D del registro RCSTA y se determina si ocurrió un error durante la recepción
9.- Se leen del registro RCREG los 8 bits de datos recibidos
10.-Si ocurrió algún error, entonces se limpia el bit CREN

Los registros asociados para la recepción en modo maestro síncrono son:

| Nombre | Direc | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| PIR1 | 0C | | | RCIF | | | | | |
| RCSTA | 18 | SPEN | RX9 | SREN | CREN | | FERR | OERR | RX9D |
| RCREG | 1A | RX7 | RX6 | RX5 | RX4 | RX3 | RX2 | RX1 | RX0 |
| PIE1 | 8C | | RCIE | | | | | | |
| TXSTA | 98 | CSRC | | | SYNC | | | TRMT | |
| SPBRG | 99 | Registro | de | baudaje | | | | | |

**-USART en modo esclavo síncrono**

El modo esclavo síncrono es similar al modo maestro excepto por que en este caso el reloj se genera externamente en lugar de generarse en forma interna. Otra diferencia es que en este modo si se pueden transmitir y recibir durante el modo de SLEEP.

-Transmisión en modo esclavo síncrono

La transmisión en modo esclavo es similar al modo maestro solo que en este caso, cuando se termina de transmitir un dato y el registro TXREG carga un nuevo dato en el registro de corrimiento TSR para transmitirlo, se puede generar una interrupción (si es que están habilitadas) que despierta al microcontrolador del estado de SLEEP.

Los pasos a seguir para programar el USART como transmisor esclavo síncrono son.

1.- Habilitar el puerto serial en modo esclavo síncrono, configurando los bits SYNC=1, SPEN=1 y CSRC=0
2.- Limpiar los bits de bandera CREN y SREN
3.- Si se decide utilizar interrupciones se debe prender el bit TXIE
4.- Si se decide utilizar 9 datos de transmisión, entonces se debe prender el bit TX9
5.- Habilitamos la transmisión prendiendo el bit TXEN
6.- Si se seleccionó 9 bits de datos de transmisión, entonces se debe cargar el noveno bit a ser transmitido en el bit TX9D del registro TXSTA
7 - Cargar el dato que se transmitirá en el registro TXREG. Esto inicia la transmisión

Los registros asociados para la transmisión en modo esclavo síncrono son:

| Nombre | Direc | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| PIR1 | 0C | | | | TXIF | | | | |
| RCSTA | 18 | SPEN | | | | | | | |
| TXREG | 19 | TX7 | TX6 | TX5 | TX4 | TX3 | TX2 | TX1 | TX0 |
| PIE1 | 8C | | | | TXIE | | | | |
| TXSTA | 98 | CSRC | TX9 | TXEN | SYNC | | | TRMT | TX9D |
| SPBRG | 99 | Registro | de | baudaje | | | | | |

-Recepción en modo esclavo síncrono

La recepción en modo esclavo también es similar al modo maestro, sólo que aquí si el receptor se habilita antes de la instrucción de SLEEP, entonces cuando se recibe un dato se puede generar una interrupción que puede despertar al microcontrolador y sacarlo de sus estado de SLEEP y ejecutar la rutina de interrupción.

Los pasos a seguir para programar el USART como receptor esclavo síncrono son

1.- Habilitar el puerto serial en modo esclavo síncrono, configurando los bits SYNC=1, SPEN=1 y CSRC=0
2.- Se debe prender el bit RCIE si se decide utilizar interrupciones
3.- Si se decide utilizar 9 bits de datos de recepción, entonces se debe prender el bit RX9.
4.- Se habilita la recepción prendiendo el bit CREN
5.- Cuando se completa una recepción se prende el bit de bandera RCIF y se puede generar una interrupción si está habilitado el bit RCIE
6.- Si está habilitada la recepción de 9 bits, entonces se toma el noveno bit de datos que queda en el bit RX9D del registro RCSTA y se determina si ocurrió un error durante la recepción
7.- Se leen los 8 bits de datos recibidos en el registro RCREG
10.-Si ocurrió algún error, entonces se limpia el bit CREN

Los registros asociados para la recepción en modo esclavo síncrono son

| Nombre | Direc. | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| PIR1 | 0C | | | RCIF | | | | | |
| RCSTA | 18 | SPEN | RX9 | SREN | CREN | | | OERR | RX9D |
| RCREG | 1A | RX7 | RX6 | RX5 | RX4 | RX3 | RX2 | RX1 | RX0 |
| PIE1 | 8C | | RCIE | | | | | | |
| TXSTA | 98 | CSRC | | | SYNC | | | | |
| SPBRG | 99 | Registro | de | baudaje | | | | | |

**Convertidor Analógico/Digital (ADC)**

El convertidor analógico/digital (ADC) que tiene el PIC16/17 nos permite leer señales analógicas que estén variando de 0V a 5V máximo y convertirlas a señales digitales. Este convertidor es de aproximaciones sucesivas y con una resolución de 8 bits, esto significa que para una señal analógica de entrada tendremos resultados digitales que variaran desde 00h hasta FFh.

Dependiendo del número del microcontrolador PIC que se seleccione, puede no tener el módulo ADC o tener uno que nos permita tener 5 o bien 8 entradas analógicas.

La referencia de voltaje máximo de las señales analógicas de entrada se puede seleccionar por software de tal forma que puede ser la tensión de $V_{DD}$ o la de $V_{REF}$ cuyo valor máximo es de $V_{DD}$ + 0.3V.

Para las entradas analógicas se utilizan 5 de las terminales del puerto A y las tres terminales del puerto E

Para la programación de este módulo se utilizan 3 registros:

- Registro 0 de control del ADC (ADCON0)
- Registro 1 de control del ADC (ADCON1).
- Registro de resultado del ADC (ADRES).

En el registro ADCON0 podemos ver algunas condiciones de estado del módulo ADC o programar ciertas características de control de este módulo. El registro ADCON1 nos permite configurar las funciones de las terminales de los puertos y el registro ADRES contiene el resultado de 8 bits de la conversión.

A continuación se muestran los registros ADCON0 y ADCON1:

**ADCON0 REGISTER (Dirección 9Fh)**

| 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 |
| bit7 | | | | | | | bit0 |

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7-6. **ADCS1:ADCS0:** A/D Conversion Clock Select bits
00 = Fosc/2
01 = Fosc/8
10 = Fosc/32
11 = FRC (clock derived from an internal RC oscillator)

bit 5-3. **CHS2:CHS0.** Analog Channel Select bits
000 = channel 0, (RA0/AN0)
001 = channel 1, (RA1/AN1)
010 = channel 2, (RA2/AN2)
011 = channel 3, (RA3/AN3)
100 = channel 4, (RA5/AN4)
101 = channel 5, (RE0/AN5)(1)
110 = channel 6, (RE1/AN6)(1)
111 = channel 7, (RE2/AN7)(1)

bit 2: **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
If ADON = 1
1 = A/D conversion in progress (setting this bit starts the A/D conversion)
0 = A/D conversion not in progress (This bit is automatically cleared by hardware when the A/D conversion is complete)

bit 1: **Unimplemented.** Read as '0'

bit 0. **ADON:** A/D On bit
1 = A/D converter module is operating
0 = A/D converter module is shutoff and consumes no operating current

**Note 1:** A/D channels 5, 6, and 7 are implemented on the PIC16C74/74A/77 only.

**ADCON1 REGISTER (Dirección 1Fh)**

| U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | PCFG2 | PCFG1 | PCFG0 |
| bit7 | | | | | | | bit0 |

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7-3 **Unimplemented:** Read as '0'

bit 2-0: **PCFG2:PCFG0.** A/D Port Configuration Control bits

| PCFG2:PCFG0 | RA0 | RA1 | RA2 | RA5 | RA3 | RE0(1) | RE1(1) | RE2(1) | VREF |
|---|---|---|---|---|---|---|---|---|---|
| 000 | A | A | A | A | A | A | A | A | VDD |
| 001 | A | A | A | A | VREF | A | A | A | RA3 |
| 010 | A | A | A | A | A | D | D | D | VDD |
| 011 | A | A | A | A | VREF | D | D | D | RA3 |
| 100 | A | A | D | D | A | D | D | D | VDD |
| 101 | A | A | D | D | VREF | D | D | D | RA3 |
| 11x | D | D | D | D | D | D | D | D | — |

A = Analog input
D = Digital I/O

**Note 1:** RE0, RE1, and RE2 are implemented on the PIC16C74/74A/77 only.

La fuente de reloj que se utiliza para el ADC se puede programar y seleccionar de las cuatro opciones que se muestran en la siguiente tabla. Para asegurar que se tenga una conversión adecuada se debe seleccionar el tipo de reloj que garantice un tiempo mínimo de muestreo (TAD) de 1 6 us. Por otro lado, el tiempo de conversión es de 10 TAD.

**TAD VS. FRECUENCIA DE OPERACIÓN DEL DISPOSITIVO**

| AD Clock Source (TAD) | | Device Frequency | | | |
|---|---|---|---|---|---|
| **Operation** | **ADCS1:ADCS0** | **20 MHz** | **5 MHz** | **1.25 MHz** | **333.33 kHz** |
| 2Tosc | 00 | 100 ns(2) | 400 ns(2) | 1.6 µs | 6 µs |
| 8Tosc | 01 | 400 ns(2) | 1.6 µs | 6.4 µs | 24 us(3) |
| 32Tosc | 10 | 1.6 µs | 6.4 µs | 25.6 µs(3) | 96 us(3) |
| RC(5) | 11 | 2 - 6 µs(1,4) | 2 - 6 µs(1,4) | 2 - 6 µs(1,4) | 2 - 6 µs(1) |

Legend. Shaded cells are outside of recommended range.

Note 1: The RC source has a typical TAD time of 4 µs.

2. These values violate the minimum required TAD time.

3: For faster conversion times, the selection of another clock source is recommended.

4. When device frequency is greater than 1 MHz, the RC A/D conversion clock source is recommended for sleep operation only.

5. For extended voltage devices (LC), please refer to Electrical Specifications section.

El módulo ADC puede estar trabajando y generando conversiones mientras el microcontrolador se encuentra en el estado de SLEEP. Si están habilitadas las interrupciones, entonces cuando se completa una conversión la interrupción despierta al microprocesador.

Utilizando el módulo CCP2 en el modo de disparo de evento especial que inicia una conversión cuando se presenta una igualación de los registros CCPR y el TMR1, podemos hacer conversiones a una frecuencia constante.

Los pasos para utilizar el ADC son los siguientes:

1.- Configurar el módulo ADC:
- Configurar las terminales analógicas, el voltaje de referencia y las entradas y salidas digitales en el registro ADCON1
- Seleccionar el canal de entrada A/D en el registro ADCON0.
- Seleccionar el reloj de conversión en el registro ADCON0.
- Prender el módulo ADC en el registro ADCON0.

2.- Si se van a utilizar interrupción es necesario configurarla:
- Limpiar el bit ADIF.
- Prender los bits ADIE y GIE

3.-Esperar el tiempo de adquisición requerido

4.-Iniciar la conversión prendiendo el bit $GO/\overline{DONE}$ en el registro ADCON0.

5.- Esperar el fin de conversión o conversión completa por el método de poleo. checando que el bit $GO/\overline{DONE}$ este limpio o esperando la interrupción del ADC

6.- Leer el resultado en el registro ADRES y limpiar la bandera ADIF si se requiere.

7 - Para la siguiente conversión se puede ir al paso 1 o 2 según se requiera Es importante dejar un tiempo de espera de por lo menos 2 TAD antes de iniciar la siguiente adquisición.

Los registros asociados al módulo del convertidor analógico/digital son:

| Nombre | Direc. | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | 0B/8B | GIE | PEIE | | | | | | |
| PIR1 | 0C | | ADIF | | | | CCP1IF | | |
| PIE1 | 8C | | ADIE | | | | CCP1IE | | |
| ADRES | 1E | A/D | Result | Reg. | | | | | |
| ADCON0 | 1F | ADCS1 | ADCS0 | CHS2 | CHS1 | CHS0 | GO/$\overline{DONE}$ | | ADON |
| ADCON1 | 9F | | | | | | PCFG2 | PCFG1 | PCFG0 |
| PORTA | 05 | | | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 |
| TRISA | 85 | | | PORTA | Data | Direc. | Reg. | | |

## Características Especiales

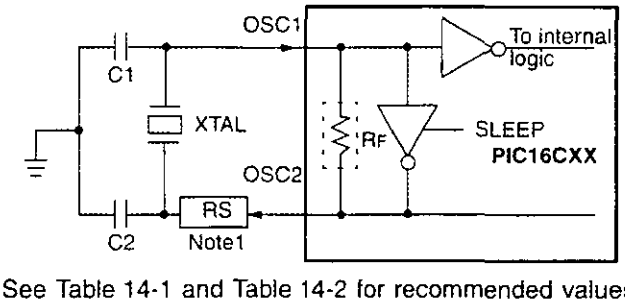
### Configuraciones del oscilador

Este microcontrolador puede operar con cuatro diferentes opciones de oscilador:

- Cristal de baja potencia LP
- Cristal resonante XT
- Cristal resonante de alta frecuencia HS
- Circuito resonante RC

Las siguientes figuras muestran los diagramas para las opciones LP, XT y HS:

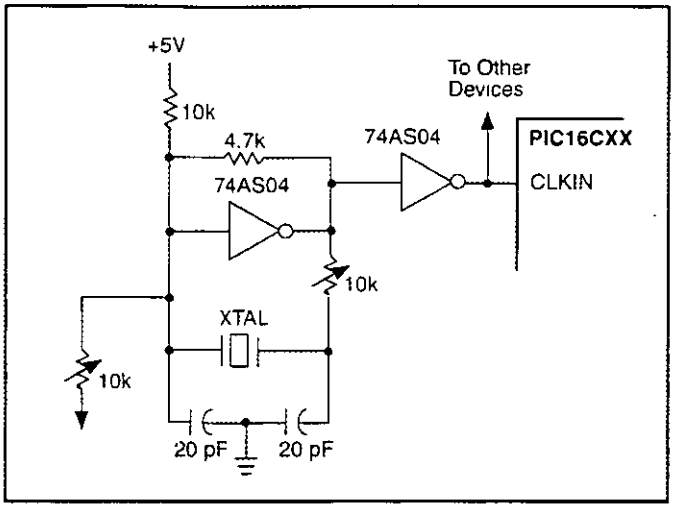
**CRYSTAL/CERAMIC RESONATOR OPERATION (HS,XT, OR LP OSC CONFIGUTATION)**



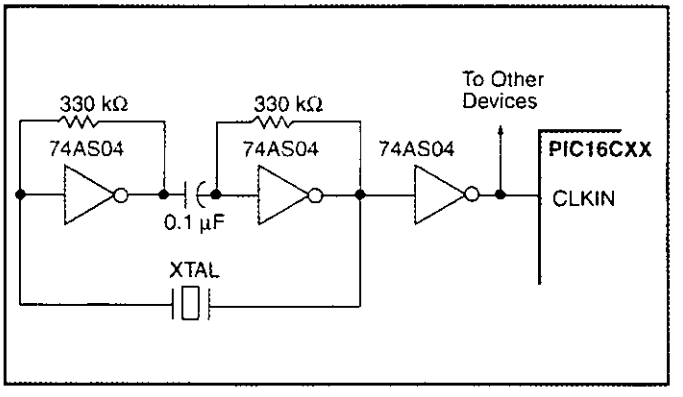

See Table 14-1 and Table 14-2 for recommended values of C1 and C2.

Note 1: A series resistor may be required for AT strip cut crystals.

**EXTERNAL PARALLEL RESONANT CRYSTAL OSCILLATOR CIRCUIT**




**EXTERNAL SERIES RESONANT CRYSTAL OSCILLATOR CIRCUIT**




Las siguientes tablas muestran las frecuencias y los valores de los capacitores para circuitos resonantes cerámicos y de cristal·

**CERAMIC**

| Ranges Tested: | | | |
|---|---|---|---|
| **Mode** | **Freq** | **OSC1** | **OSC2** |
| XT | 455 kHz | 68 - 100 pF | 68 - 100 pF |
| | 2.0 MHz | 15 - 68 pF | 15 - 68 pF |
| | 4.0 MHz | 15 - 68 pF | 15 - 68 pF |
| HS | 8.0 MHz | 10 - 68 pF | 10 - 68 pF |
| | 16.0 MHz | 10 - 22 pF | 10 - 22 pF |
| **These values are for design guidance only.** See notes at bottom of page | | | |
| **Resonators Used:** | | | |
| 455 kHz | Panasonic EFO-A455K04B | ± 0.3% | |
| 2.0 MHz | Murata Erie CSA2 00MG | ± 0.5% | |
| 4.0 MHz | Murata Erie CSA4.00MG | ± 0.5% | |
| 8.0 MHz | Murata Erie CSA8.00MT | ± 0.5% | |
| 16.0 MHz | Murata Erie CSA16.00MX | ± 0.5% | |
| All resonators used did not have built-in capacitors. | | | |

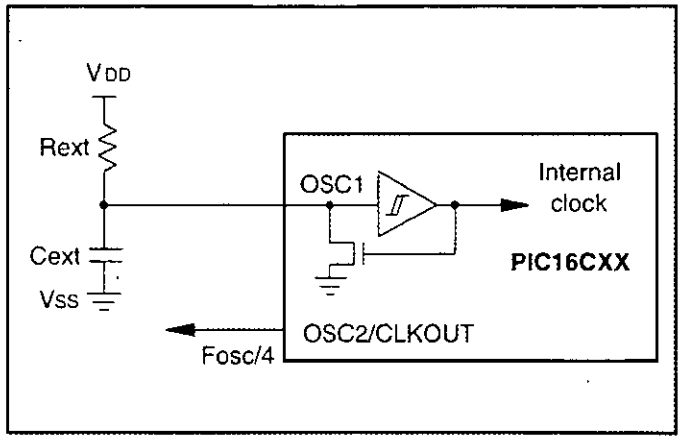
**CRYSTAL**

| Osc Type | Crystal Freq | Cap. Range C1 | Cap. Range C2 |
|---|---|---|---|
| LP | 32 kHz | 33 pF | 33 pF |
| | 200 kHz | 15 pF | 15 pF |
| XT | 200 kHz | 47-68 pF | 47-68 pF |
| | 1 MHz | 15 pF | 15 pF |
| | 4 MHz | 15 pF | 15 pF |
| HS | 4 MHz | 15 pF | 15 pF |
| | 8 MHz | 15-33 pF | 15-33 pF |
| | 20 MHz | 15-33 pF | 15-33 pF |
| **These values are for design guidance only.** See notes at bottom of page. | | | |
| **Crystals Used** | | | |
| 32 kHz | Epson C-001R32.768K-A | | ± 20 PPM |
| 200 kHz | STD XTL 200.000KHz | | ± 20 PPM |
| 1 MHz | ECS ECS-10-13-1 | | ± 50 PPM |
| 4 MHz | ECS ECS-40-20-1 | | ± 50 PPM |
| 8 MHz | EPSON CA-301 8.000M-C | | ± 30 PPM |
| 20 MHz | EPSON CA-301 20.000M-C | | ± 30 PPM |

Note 1: Recommended values of C1 and C2 are identical to the ranges tested (Table 14-1).
2: Higher capacitance increases the stability of oscillator but also increases the start-up time.
3: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.
4: Rs may be required in HS mode as well as XT mode to avoid overdriving crystals with low drive level specification.

El circuito oscilador RC presenta la ventaja de que puede ser más económico, pero la frecuencia que genera presenta muchas variaciones debidas a factores externos tales como temperatura, humedad, ruido, etc. El circuito RC se muestra a continuación:

**RC OSCILLATOR MODE**



Para el circuito RC se recomienda que los valores de Rext estén entre 3Kohms y 100kohms. Asimismo, por razones de estabilidad y ruido los valores de Cext deben ser menores de 20pF

## Reset

Existen varias formas de reinicializar al microcontrolador. La siguiente tabla muestra los tipos de reinicialización así como los efectos en dos de los registros más importantes·

**CONDICIONES DE RESET PARA REGISTROS ESPECIALES**

| Condición | Contador de Programa (registro PCL) | **Registro STATUS** |
|---|---|---|
| Power-on Reset | 000h | 0001 1xxx |
| $\overline{MCRL}$ durante operación normal | 000h | 000u uuuu |
| $\overline{MCRL}$ durante el estado de SLEEP | 000h | 0001 0uuu |
| WDT durante operación normal | 000h | 0000 1uuu |
| WDT durante el estado de SLEEP | PC + 1 | uuu0 0uuu |
| Interrupción durante el estado de SLEEP | PC + 1(1) | uuu1 0uuu |

u= sin cambio, x= no se sabe

(1) Cuando despiertan al microcontrolador por medio de una interrupción y está habilitado el bit de interrupción general GIE, entonces el contador de programa PC se carga con el vector de interrupción (0004h)

## Power-on Reset

La condición de Power-on Reset se presenta cuando se conecta el voltaje de alimentación. Cuando se detecta que el voltaje se encuentra en el rango de 1.5V a 2.1V, entonces se genera internamente un pulso de reset que permite que el microcontrolador no empiece a trabajar hasta que se estabilice el nivel de voltaje de alimentación. Adicional al Power-on Reset existe la función Power-Up Timer que se puede habilitar (en un registro de configuración CONFIG) y que genera un tiempo de espera de 72 ms nominalmente. Este tiempo se genera por medio de un circuito interno RC.

Por otro lado, existe también la función Oscillator Star-Up Timer que provee un retardo de 1024 ciclos para garantizar que el oscilador ya esta trabajando en forma estable.

**Watchdog Timer (WDT)**

La base de tiempo del WDT es un oscilador libre realizado con un circuito RC interno. El WDT se puede habilitar o deshabilitar permanentemente por medio del registro CONFIG.

Si durante la operación normal el WDT genera un tiempo fuera entonces provoca un reset que hace que el contador de programa se reinicialize a partir de la dirección 000h. Si el tiempo fuera del WDT se genera durante el estado de SLEEP, entonces provoca que el dispositivo se despierte y continúe con su operación normal Para evitar que el WDT reinicialize al microcontrolador o lo despierte de su estado de SLEEP es necesario limpiarlo con la instrucción *CLRWDT*.

El WDT tiene un tiempo fuera con un periodo de 18ms sin la preescala. Si se requieren periodos más largos, entonces en necesario utilizar la preescala que se selecciona en el registro OPTION pudiéndose lograr periodos de hasta 2.3 segundos. Cuando se genera un tiempo fuera se apaga el bit $\overline{TO}$ del registro STATUS.

Las instrucciones de *CLRWDT* y de *SLEEP* limpian el WDT y el contador de la preescala, pero la preescala seleccionada se le vuelve a signar al WDT.

**Modo de SLEEP**

La instrucción de SLEEP coloca al microcontrolador en un estado de apagado o desconectado (power-down) y de bajo consumo de corriente Para esto, todas las terminales de entrada y salida las mantiene a VDD o a VSS, detiene el sistema del oscilador deshabilitando el reloj externo y el WDT sigue corriendo si es que está habilitado

Los siguientes eventos pueden despertar al microcontrolador y sacarlo de su estado de SLEEP:

- Reset externo dado en la terminal $\overline{MCRL}$
- El tiempo fuera del WDT si es que está habilitado
- Cuando ocurre una interrupción.

En el primer caso el contador de programa se reinicializa a partir de la dirección 000h, mientras que en los otros casos continua con la ejecución del programa.

## Interrupciones

Existen varias fuentes que pueden generar interrupción:

- Externa en la terminal RB0/INT.
- Sobreflujo de TMR0.
- Por cambio en los bits 4 a 7 de PORTB.
- Fin de conversión del módulo ADC.
- Sobreflujo de TMR1
- TMR2.
- CCP1
- CCP2
- Transmisión y recepción asíncrona con el periférico USART.
- SSP.

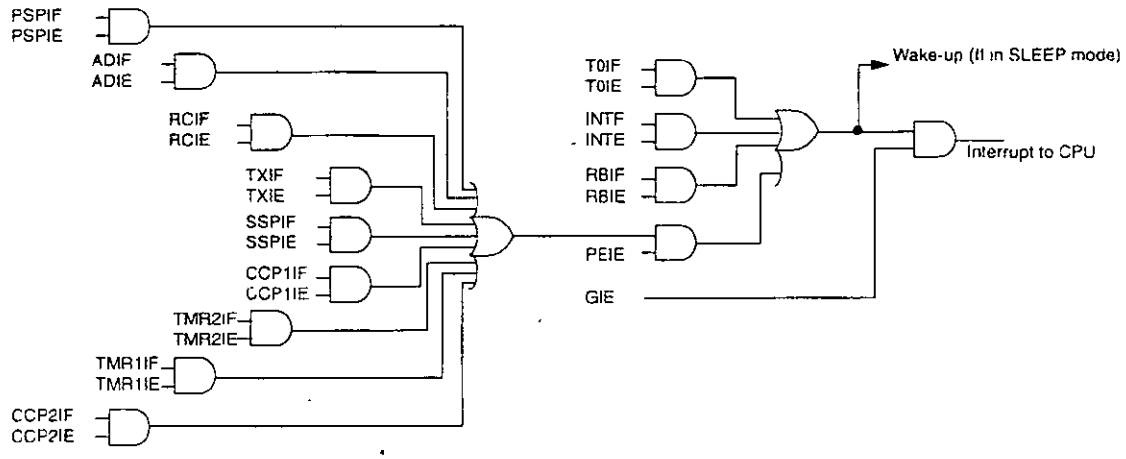
El bit GIE del registro INTCON habilita o deshabilita todas las interrupciones mascarables. Cada dispositivo tiene un bit para habilitar o deshabilitar la posibilidad de que pueda generar una interrupción y un bit de bandera que se prende cundo se genera. Los bits de habilitación y banderas se encuentran en los registros INTCON, PIE1, PIR1, PIE2 y PIR2.

Cuando se responde a una interrupción, se limpia el bit GIE para deshabilitar futuras interrupciones. La dirección a la que se va a retornar se almacena en la pila (stack) y el contador de programa se carga con la dirección del vector de interrupción (0004h) En esta dirección comienza la rutina de servicio de interrupción y es aquí donde se debe determinar, por medio de poleo de los bits de bandera, cual de los dispositivos fue el que generó la interrupción. Los bits de bandera de interrupción deben ser limpiados por software antes de que se vuelvan a habilitar las interrupciones para evitar un requerimiento infinito de interrupciones

Para salir de una rutina de servicio de interrupción se utiliza la instrucción *RETFIE* que además vuelve a prender el bit GIE para habilitar nuevamente la posibilidad de interrupción.

La siguiente figura nos muestra un diagrama lógico para las interrupciones:

**LÓGICA DE INTERRUPCIONES**




The following table shows which devices have which interrupts.

| Device | T0IF | INTF | RBIF | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | CCP2IF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PIC16C72 | Yes | Yes | Yes | - | Yes | - | - | Yes | Yes | Yes | Yes | - |
| PIC16C73 | Yes | Yes | Yes | - | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| PIC16C73A | Yes | Yes | Yes | - | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| PIC16C74 | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| PIC16C74A | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| PIC16C76 | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| PIC16C77 | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |

## Bibliografía:


"PIC16C7X, Pin 8-Bit CMOS Microcontrollers with A/D Converter", DS30390E, Microchip Technology Inc., 1999.

"PIC 16/17 MICROCONTROLLER DATA BOOK", Microchip Technology Inc., 1995/1996, 2-517-2-722.

"MPLAB. IDE, SIMULATOR, EDITOR,USER'S GUIDE", DS51025D, Microchip Technology Inc., 2000.