# Características especiales de los Microcontroladores PIC








• Estas características suelen ser los aspectos más diferenciadores entre la CPU de estos dispositivos y otros microprocesadores

• Son características pensadas para que el microcontrolador sea más "autónomo", más barato, más seguro y ocupe menos que sus hermanos mayores.

Esas características se centran en varios aspectos:

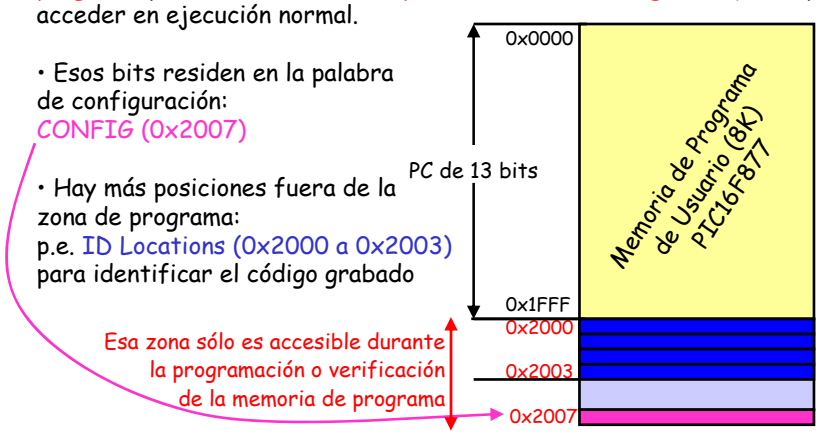
- Oscilador: más simple y con menos elementos adicionales necesarios
- *Resets* y *Watchdog*: seguridad en el arranque, reinicio y "autovigilancia"
- *Sleep*: modo de bajo consumo para aplicaciones con baterías
- Interrupciones: lógica de máscaras y eventos y posición común del PTI
- Protección de código: para evitar la "copia" de programas grabados
- ICSP e ICSP LVP: (*In-Circuit Serial Programming*) programación en serie ya en la tarjeta de la aplicación y a baja tensión (*Low Voltage Program*)
- Modo ICD: (*In-Circuit Debugger)* modo especial que permite depuración dentro del circuito por comunicación con un *Debugger*

## Configuración de las Características Especiales

• La mayor parte de las características especiales se establecen en unos bits que residen en una palabra de iguales características<que la memoria de programa pero está fuera del mapa de memoria de código a la que se puede acceder en ejecución normal.

• Esos bits residen en la palabra de configuración:
CONFIG (0x2007)

• Hay más posiciones fuera de la zona de programa:
p.e. ID Locations (0x2000 a 0x2003)
para identificar el código grabado








## BITS DE CONFIGURACION

• Todos los microcontroladores PIC tienen una posición de memoria denominada palabra de configuración, en la que cada bit tiene un significado y configura las características especiales.

• Los bits de configuración pueden ser programados (puestos a 0) o dejados sin programar (quedan a 1), con objeto de seleccionar varias configuraciones del microcontrolador: tipo de oscilador, protección o no del programa, uso ó no del watchdog, etc.

• El valor no programado de la palabra de configuración es 0x3FFF (todo "1")

• Al estar los bits de configuración en la posición 2007h de la memoria de programa (fuera del espacio de memoria de programa de usuario) es únicamente accesible durante la programación del micro y no durante la ejecución de un programa.

• Por tanto es especialmente importante cargar correctamente esos bits durante la programación para conseguir que el microcontrolador pueda funcionar

## BITS DE CONFIGURACION EN PIC16F87XA

REGISTER 12-1: CONFIGURATION WORD (ADDRESS 2007h)(1)

| CP1 | CP0 | DEBUG | — | WRT | CPD | LVP | BODEN | CP1 | CP0 | PWRTE | WDTE | FOSC1 | FOSC0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| bit13 | | | | | | | | | | | | | bit0 |

bit 13-12, bit 5-4 Protección de zonas de la memoria de programa
CP1:CP0: FLASH Program Memory Code Protection bits(2)
11 = Code protection off
10 = 1F00h to 1FFFh code protected (PIC16F877, 876)
10 = 0F00h to 0FFFh code protected (PIC16F874, 873)
01 = 1000h to 1FFFh code protected (PIC16F877, 876)
01 = 0800h to 0FFFh code protected (PIC16F874, 873)
00 = 0000h to 1FFFh code protected (PIC16F877, 876)
00 = 0000h to 0FFFh code protected (PIC16F874, 873)

bit 11 Modo debugger
DEBUG: In-Circuit Debugger Mode
1 = In-Circuit Debugger disabled, RB6 and RB7 are general purpose I/O pins
0 = In-Circuit Debugger enabled, RB6 and RB7 are dedicated to the debugger.

bit 10 Unimplemented: Read as '1'

bit 9 Activación de escritura desde programa de memoria de programa
WRT: FLASH Program Memory Write Enable
1 = Unprotected program memory may be written to by EECON control
0 = Unprotected program memory may not be written to by EECON control

bit 8 Protección de la EEPROM de datos
CPD: Data EE Memory Code Protection
1 = Code protection off
0 = Data EEPROM memory code protected

bit 7 Activación de ICSP a baja tensión
LVP: Low Voltage In-Circuit Serial Programming Enable bit
1 = RB3/PGM pin has PGM function, low voltage programming enabled
0 = RB3 is digital I/O, HV on MCLR must be used for programming

bit 6 Activación del Reset por Brown-out
BODEN: Brown-out Reset Enable bit(3)
1 = BOR enabled
0 = BOR disabled

bit 3 Activación de la temp. de encendido
PWRTE: Power-up Timer Enable bit(3)
1 = PWRT disabled
0 = PWRT enabled

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

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





## BITS DE CONFIGURACION

REGISTER 12-1: CONFIGURATION WORD FOR 16C717/770/771 DEVICE

| CP | CP | BORV1 | BORV0 | CP | CP | — | BODEN | MCLRE | PWRTE | WDTE | FOSC2 | FOSC1 | FOSC0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| bit13 | | | | | | | | | | | | | bit0 |

Los bits de la palabra de configuración dependen del micro. Dentro de la familia PIC16, hay micros que tienen opciones de configuración distintas a los PIC16F87X vistos en la diapositiva anterior.

Por ejemplo, en la figura adjunta se ve la palabra de configuración de un PIC16C717/770/771.

En este caso el oscilador puede ser interno o externo y por ello tiene mas bits para configurar el tipo de oscilador:
FOSC2:FOSC0: Oscillator Selection bits
111= EXTRC oscillator, with CLKOUT
110= EXTRC oscillator
101= INTRC oscillator, with CLKOUT
100= INTRC oscillator
011= Reserved
010= HS oscillator
001= XT oscillator
000= LP oscillator

En este caso también se puede seleccionar la tensión de brown-out
BORV1:BORV0: Brown-out Reset Voltage bits
00= VBOR set to 4.5V
01= VBOR set to 4.2V
10= VBOR set to 2.7V
11= VBOR set to 2.5V

También existen otros casos donde, por ejemplo se puede seleccionar la función del pin etiquetado como MCLR
MCLRE: MCLR Pin Function Select bit
1 = Pin's function is MCLR
0 = Pin's function is as a digital I/O. MCLR is internally tied to VDD

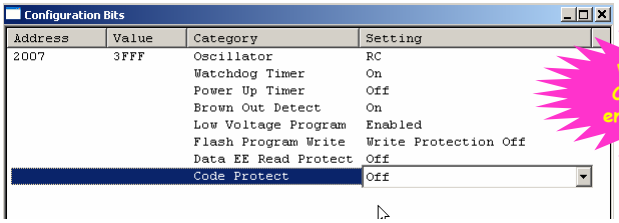
**La palabra de configuración no es idéntica para todos los microcontroladores**

## BITS DE CONFIGURACION desde MPLAB-IDE

Insistimos porque es importante: los bits de configuración únicamente pueden escribirse durante la programación del microcontrolador.

Las herramientas que ofrece MICROCHIP nos dan dos alternativas para fijar los valores de estos bits de configuración: a través del menú **Configure > Configuration_Bits** del entorno MPLAB ó mediante la inclusión de una directiva de configuración en el código del programa.

### Menú Configure > Configuration_Bits




Si se utiliza la directiva, no es necesario hacer la configuración desde ese menú sino que el código máquina generado, ya incluiría el valor a grabar en la palabra de configuración y sería el valor utilizado por el programador para grabar la posición 0x2007







## BITS DE CONFIGURACION desde Directiva

El ensamblador de MICROCHIP tiene una característica que permite especificar, en el propio fichero fuente las selecciones de los bits de configuración. De esta forma, se asegura que al programar el dispositivo también se programe la configuración de acuerdo a los requisitos de la aplicación, evitando la programación equivocada de la configuración y, por tanto que el dispositivo no funcione correctamente. Esta característica consiste en el uso de la directiva CONFIG. A continuación se detalla un ejemplo y una tabla con los posibles valores para configuración.

```
LIST p = p16C77 ; List Directive,
; Revision History
;
#INCLUDE <P16C77.INC> ; Microchip Device Header File
;
#INCLUDE <MY_STD.MAC> ; File which includes my standard macros
#INCLUDE <APP.MAC> ; File which includes macros specific
; to this application
;
; Specify Device Configuration Bits
;
__CONFIG _XT_OSC & _PWRTE_ON & _BODEN_OFF & _CP_OFF &
_WDT_ON
;
org 0x00 ; Start of Program Memory
RESET_ADDR : ; First instruction to execute after a reset
end
```

Estas etiquetas están definidas en los ficheros de inclusión

| CARACTERISTICA | SIMBOLO |
|---|---|
| OSCILADOR | _RC_OSC |
| | _EXTRC_OSC |
| | _EXTRC_OSC_CLKOUT |
| | _EXTRC_OSC_NOCLKOUT |
| | _INTRC_OSC |
| | _INTRC_OSC_CLKOUT |
| | _INTRC_OSC_NOCLKOUT |
| | _LP_OSC |
| | _XT_OSC |
| | _HS_OSC |
| Watch Dog Timer | _WDT_ON |
| | _WDT_OFF |
| Power-up Timer | _PWRTE_ON |
| | _PWRTE_OFF |
| Brown-out Reset | _BODEN_ON |
| | _BODEN_OFF |
| Master Clear Enable | _MCLRE_ON |
| | _MCLRE_OFF |
| Code Protect | _CP_ALL |
| | _CP_ON |
| | _CP_75 |
| | _CP_50 |
| | _CP_OFF |
| Code Protect Data EEPROM | _DP_ON |
| | _DP_OFF |
| Code Protect Calibration Space | _CPC_ON |
| | _CPC_OFF |

Mediante el símbolo & se "enlazan" las etiquetas de configuración

## Características especiales: OSCILADOR

¿Qué necesitan los microcontroladores para funcionar?

Sólo una tensión continua estable (5V, 3.3V, 2.5V, 1.5V...) y un oscilador

Los PIC16F87X pueden funcionar con 4 modos distintos de oscilador. El usuario puede programar dos bits de configuración para seleccionar uno de estos 4 modos:

- LP Low Power Crystal (cristal de cuarzo ó resonador cerámico hasta 200KHz)
- XT Crystal/Resonator (cristal de cuarzo ó resonador cerámico hasta 4MHz)
- HS High Speed Crystal/Resonator (cristal de cuarzo entre 4MHz y 20MHz)
- RC Resistor/Capacitor (red RC externa hasta 4MHz)

Otros PIC de la familia PIC16 tienen un número mayor de modos para el oscilador y, por tanto, un número mayor de bits para seleccionar. Estos otros modos son:

- EXTRC External Resistor/Capacitor
- EXTRC External Resistor/Capacitor with CLKOUT
- INTRC Internal 4 MHz Resistor/Capacitor
- INTRC Internal 4 MHz Resistor/Capacitor with CLKOUT

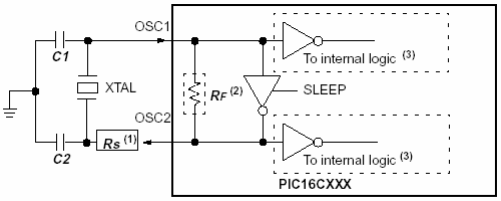
INTRC: ¡ No es necesario colocar nada fuera el reloj se genera dentro !





## CONFIGURACIONES DEL OSCILADOR

En los modos XT, LP o HS, se conecta un cristal de cuarzo o un resonador cerámico a los pines OSC1 y OSC2 tal y como se indica en la figura adjunta. Normalmente no se pone la resistencia Rs, que se sustituye por un cortocircuito




MODULOS OSCILADORES

CRISTALES CUARZO

Los valores de los condensadores C1 y C2 dependen del cristal o resonador escogido. En las tablas adjuntas se pueden ver valores típicos para estos condensadores

RESONADORES CERAMICOS

TABLE 12-1: CERAMIC RESONATORS

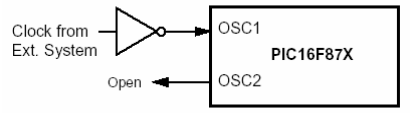
| Ranges Tested: | | | |
|---|---|---|---|
| Mode | Freq. | OSC1 | OSC2 |
| XT | 455 kHz | 68 - 100 pF | 68 - 100 pF |
| | 2.0 MHz | 15 - 68 pF | 15 - 68 pF |
| | 4.0 MHz | 15 - 68 pF | 15 - 68 pF |
| HS | 8.0 MHz | 10 - 68 pF | 10 - 68 pF |
| | 16.0 MHz | 10 - 22 pF | 10 - 22 pF |
| These values are for design guidance only. See notes following Table 12-2. | | | |
| 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. | | | |

CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

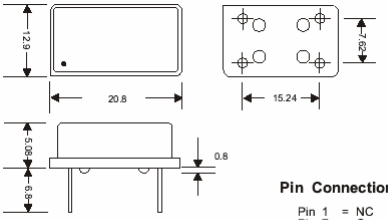
| Osc Type | Crystal Freq. | Cap. Range C1 | Cap. Range C2 |
|---|---|---|---|
| LP | 32 kHz | 33 pF | 33 pF |
| | 200 kHz | 15 pF | 15 pF |
| XT | 200 kHz | 47-68 pF | 47-68 pF |
| | 1 MHz | 15 pF | 15 pF |
| | 4 MHz | 15 pF | 15 pF |
| HS | 4 MHz | 15 pF | 15 pF |
| | 8 MHz | 15-33 pF | 15-33 pF |
| | 20 MHz | 15-33 pF | 15-33 pF |
| These values are for design guidance only. See notes following this table. | | | |
| 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 | |

## CONFIGURACIONES DEL OSCILADOR

En los modos XT, LP o HS también se puede utilizar un oscilador de cuarzo ó módulo de cristal que ya te proporciona la señal de reloj directamente sin hacer uso de la circuitería interna del microcontrolador para generar el oscilador

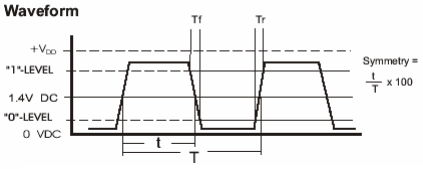







**Pin Connections**

Pin 1 = NC
Pin 7 = Ground
Pin 8 = Output
Pin 14 = +V DC

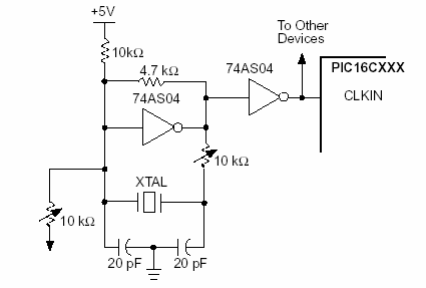


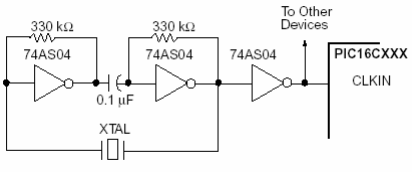





## CONFIGURACIONES DEL OSCILADOR

Algunos circuitos osciladores externos basados en cristales de cuarzo que se pueden usar en los modos XT, LP ó HS:



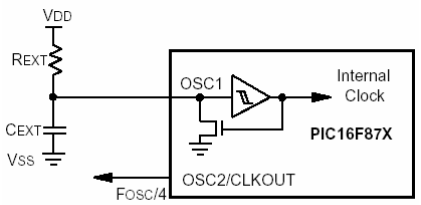



## CONFIGURACIONES DEL OSCILADOR

En aplicaciones donde la precisión del tiempo no tiene porqué ser muy elevada, la opción RC es una solución válida de bajo coste. En este caso se conectan externamente una resistencia y un condensador como se indica en la figura adjunta.

La frecuencia de oscilación depende de la tensión de alimentación VDD, del valor de la resistencia REXT, del valor del condensador CEXT y de la temperatura de funcionamiento. Por tanto, de una tarjeta con un micro a otra con el mismo micro y valores de REXT y CEXT, la frecuencia de oscilación podrá variar acorde a la tolerancia en los valores de REXT y CEXT.

La frecuencia del oscilador dividida por 4 se obtiene como salida en el pin OSC2



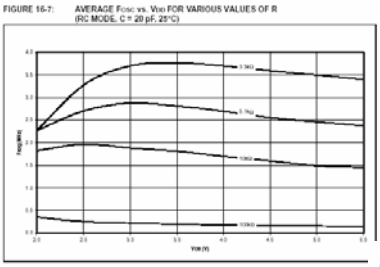

Recommended values: 3 kΩ ≤ REXT ≤ 100 kΩ
CEXT > 20pF

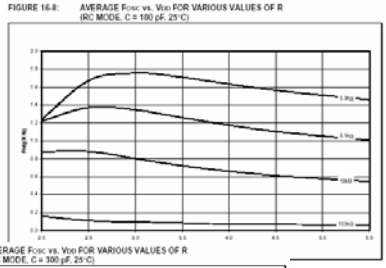


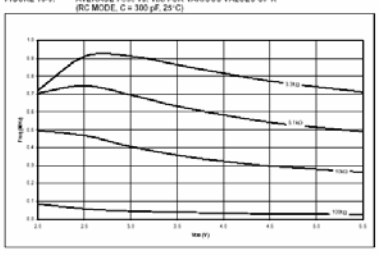


## CONFIGURACIONES DEL OSCILADOR

En las características eléctricas de cada micro se dan gráficas que te permiten escoger el valor de la resistencia para un valor del condensador, temperatura y tensión de alimentación determinadas




Cext fijo

Una curva para cada Rext

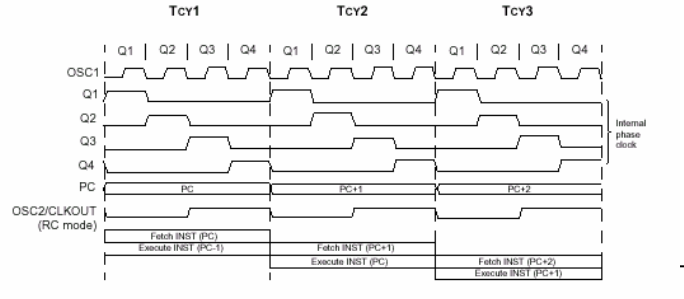
Frecuencia

VDD (tensión de alimentación)

## CICLO DE INSTRUCCION

Un ciclo de instrucción es el tiempo que se tarda en ejecutar una instrucción (salvo saltos) en el microcontrolador. En los PIC16, un ciclo de instrucción dura 4 ciclos de reloj (Q1, Q2, Q3, Q4).

- En una primera etapa, la instrucción es traída a la CPU. Esto lleva un ciclo de instrucción $T_{CY}$.
- En la segunda etapa se ejecuta la instrucción. Esto lleva otro $T_{CY}$.
- No obstante, debido al solapamiento (*pipelining* ó entubado) de traer la instrucción actual y ejecución de la instrucción previa, una instrucción se trae y otra se ejecuta cada $T_{CY}$.

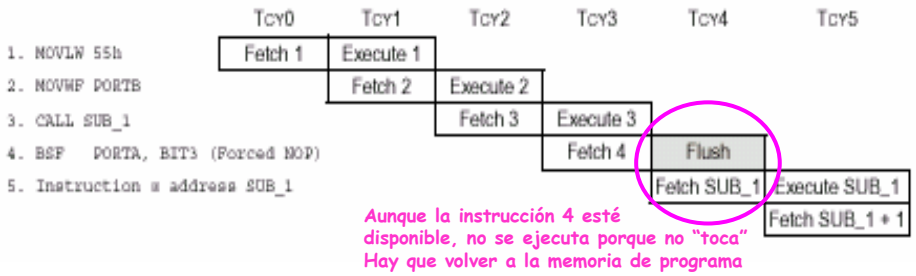







## CICLO DE INSTRUCCIÓN

Pudiera haber un ciclo de instrucción de retardo si el resultado de ejecutar la instrucción anterior modifica el contenido del Contador de Programa (Ej: GOTO ó CALL). Esto implica suspender el entubado (pipelining) de las instrucciones durante un ciclo para que la instrucción a donde se salta se traiga a la CPU.




Aunque la instrucción 4 esté disponible, no se ejecuta porque no "toca" Hay que volver a la memoria de programa

## MODO DORMIDO ("*SLEEP*")

• Los microcontroladores PIC pueden trabajar en dos modos distintos:

- ❖ Modo Normal: ejecutando las instrucciones
- ❖ Modo Dormido o de bajo consumo: se suspende la ejecución

• El consumo de un microcontrolador depende de su frecuencia de trabajo, a más frecuencia más consumo (por carga y descarga de capacidades internas y externas)

• El modo dormido supone un ahorro de consumo porque el oscilador del microcontrolador deja de oscilar, por tanto no se ejecutan instrucciones.

• Puede ser interesante para aplicaciones portátiles (alimentadas desde baterías o paneles solares) si el microcontrolador no va hacer nada durante un periodo de tiempo dado en espera de que pase algo (que lo despierten).

• En el modo "dormido" ó modo de bajo consumo se entra por software cuando se ejecuta la instrucción SLEEP.





## MODO DORMIDO ("*SLEEP*")

• Al entrar en modo dormido, el bit $\overline{PD}$ (STATUS<3>) se pone a 0 y el bit $\overline{TO}$ (STATUS<4>) se pone a 1. A continuación el oscilador deja de oscilar.

• Los pines asociados a Puertos de Entrada/Salida mantienen el valor previo a la ejecución de la instrucción SLEEP.

• Si está habilitado el WATCHDOG (en la palabra de configuración), su temporizador se pondrá a cero al ejecutar la instrucción SLEEP, pero se mantendrá "corriendo" y podrá desbordar. El Watchdog tiene un oscilador independiente del propio del microcontrolador.

• Para asegurar un consumo mínimo de corriente en este modo, se aconseja colocar los pines en los estados 1 ó 0 de forma que ningún circuito externo al microcontrolador tenga consumo de la alimentación (si es posible), apagar el conversor A/D, deshabilitar todos los relojes (ponerlos a 1 ó a 0) y deshabilitar las resistencias de pull-up del PORTB

• El pin $\overline{MCLR}$ debe estar a nivel alto para evitar un Reset externo.

## ¿COMO SE SALE DEL MODO DORMIDO?

El microcontrolador puede salir del modo de bajo consumo por alguno de los siguientes motivos:

1. RESET externo provocado en el pin MCLR.
2. Desbordamiento del WATCHDOG.
3. Interrupción o "cuasi-interrupción" provocada por algún evento de los periféricos que pueden generarlos sin la presencia del oscilador.

El $\overline{\text{MCLR}}$ RESET causará un RESET del dispositivo, pero las otras dos formas de sacar al microcontrolador del modo dormido únicamente provocan un despertar y una continuación del programa con la instrucción que sigue a SLEEP.

En el supuesto de que se haya despertado por una "cuasi-interrupción" y la máscara global de interrupciones esté a 1, tras ejecutar la instrucción que sigue a SLEEP, se continuará con la ejecución de la primera instrucción de la rutina de interrupción (posición de memoria 4).





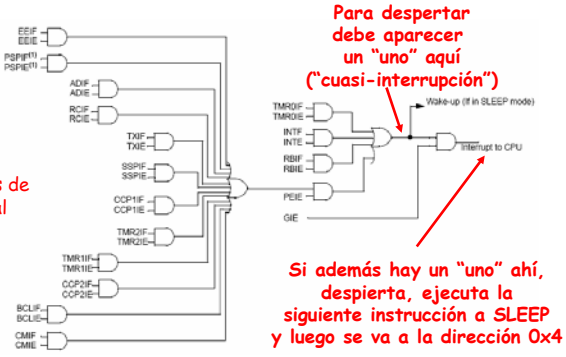
## ¿COMO SE SALE DEL MODO DORMIDO (2)?

• Mientras que la instrucción SLEEP está siendo ejecutada, la siguiente instrucción (PC+1) ya se coloca en el registro de instrucciones.

• Para "despertar" al microcontrolador a través de un evento que puede provocar una interrupción, el bit de habilitación de la interrupción correspondiente debe estar a 1 (cuasi-interrupción).

• El "despertar" del microcontrolador se produce independientemente del estado de la máscara GIE.

• Colocar una instrucción NOP después de la instrucción SLEEP suele ser habitual si no se desea hacer nada tras un SLEEP y previo el salto a la rutina de interrupción si se va a producir éste.




**Para despertar debe aparecer un "uno" aquí ("cuasi-interrupción")**

**Si además hay un "uno" ahí, despierta, ejecuta la siguiente instrucción a SLEEP y luego se va a la dirección 0x4**

## Características especiales: ARRANQUE Y REINICIOS

Los PIC16F87X tienen 6 posibles fuentes de RESET del MCU:
- Power-on Reset (POR) -> Reset de Alimentación del Microcontrolador
- MCLR Reset durante funcionamiento normal -> Activación del pin de Reset en modo normal
- MCLR Reset durante SLEEP -> Activación del pin de Reset en modo de bajo consumo
- WDT Reset (durante funcionamiento normal) -> Desbordamiento del Watchdog en modo normal
- WDT Wake-up (durante SLEEP) -> Desbordamiento del Watchdog en modo de bajo consumo
- Brown-out Reset (BOR) -> Reset por caída temporal de la alimentación

La mayoría de los registros del mapa de memoria de datos no se ven afectados por ningún tipo de RESET, su estado ó valor puede ser desconocido si se produce un POR o permanecer inalterado respecto a su último valor con otro tipo de RESET.

No obstante, hay muchos otros registros que son "reseteados" a un valor determinado si se produce un POR, un MCLR Reset ó WDT Reset durante funcionamiento normal, un MCLR Reset durante SLEEP ó un BOR. Estos registros no suelen ver afectado su valor si se produce un WDT Wake-Up, que realmente es una vuelta al funcionamiento normal desde el punto de programa donde se hubiera ejecutado el SLEEP.

Los bits $\overline{TO}$ y $\overline{PD}$ del registro STATUS y los bits $\overline{POR}$ y $\overline{BOR}$ del registro PCON dan información de cuál fue el motivo del último RESET, y pueden leerse y utilizarse luego en el programa.

TABLE 12-4: STATUS BITS AND THEIR SIGNIFICANCE

| $\overline{POR}$ | $\overline{BOR}$ | $\overline{TO}$ | $\overline{PD}$ | |
|---|---|---|---|---|
| 0 | x | 1 | 1 | Power-on Reset |
| 0 | x | 0 | x | Illegal, $\overline{TO}$ is set on $\overline{POR}$ |
| 0 | x | x | 0 | Illegal, $\overline{PD}$ is set on $\overline{POR}$ |
| 1 | 0 | 1 | 1 | Brown-out Reset |
| 1 | 1 | 0 | 1 | WDT Reset |
| 1 | 1 | 0 | 0 | WDT Wake-up |
| 1 | 1 | u | u | $\overline{MCLR}$ Reset during normal operation |
| 1 | 1 | 1 | 0 | $\overline{MCLR}$ Reset during SLEEP or interrupt wake-up from SLEEP |

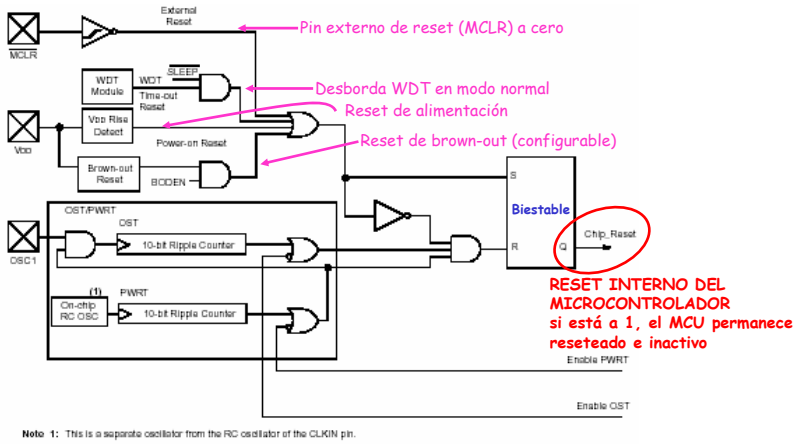
Legend: x = don't care, u = unchanged





## RESET DEL MCU Y TEMPORIZACIONES EN EL ARRANQUE

La figura muestra el diagrama de bloques simplificado de la lógica interna del RESET del microcontrolador

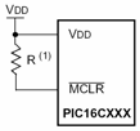



Note 1: This is a separate oscillator from the RC oscillator of the CLKIN pin.

Los PIC16F87X tienen un filtro en MCLR que detecta e ignora pequeños pulsos que se puedan producir en el pin.

## POWER-ON RESET (POR)

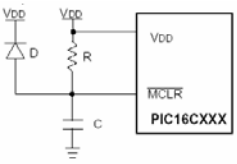
En el micro, se genera internamente un pulso de reset cada vez que se alimenta el sistema. Este pulso se produce al subir la tensión en el pin VDD y alcanzar este un valor en el rango de 1,2V a 1,7V. Esta característica permite eliminar el uso de redes RC externas al microcontrolador para generar un RESET en la puesta bajo tensión. Lo normal es unir la patilla MCLR a VDD a través de una resistencia




Cuando el dispositivo sale de la condición de RESET y comienza el funcionamiento normal, debemos asegurar que los parámetros de funcionamiento (tensión, frecuencia, temperatura, etc) estén dentro de los márgenes de funcionamiento permitidos, si no el micro podría no funcionar correctamente. Existen parámetros relativos a las características que debe tener la pendiente de subida de la tensión de alimentación para asegurar un pulso de POR.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| D004 | SVDD | VDD Rise Rate to ensure internal Power-on Reset signal | 0.05 | — | — | V/ms | See section on Power-on Reset for details |

Si la pendiente de subida de la tensión VDD es excesivamente lenta suele colocarse una red RC en la patilla MCLR para asegurar un RESET correcto. El diodo D únicamente ayuda a la descarga del condensador cuando se apaga la alimentación



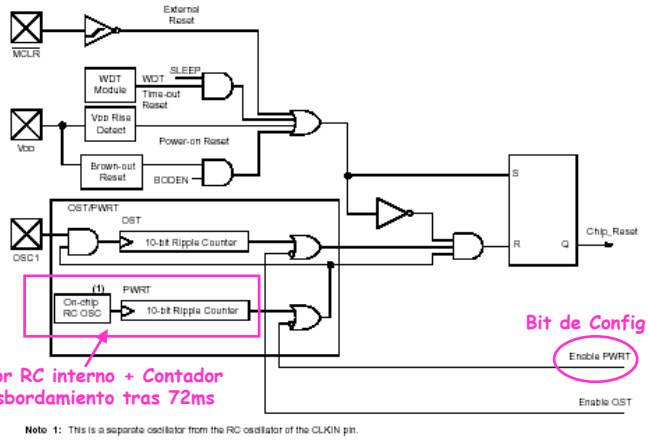

R < 40 kΩ is recommended to ensure that the voltage drop across R does not exceed 0.2V. A larger voltage drop will degrade VIH level on the MCLR/VPP pin.





## POWER-UP TIMER (PWRT)

• De manera opcional (configurable) se puede esperar un tiempo (aprox. 72ms) antes de liberar el estado de reset del MCU, desde que finaliza el pulso interno de POR



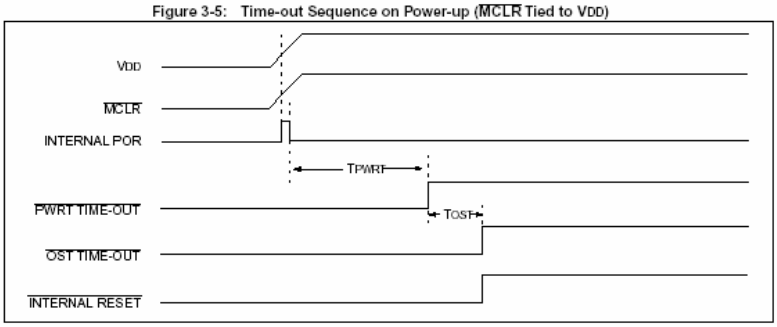

Note 1: This is a separate oscillator from the RC oscillator of the CLKIN pin.

## POWER-UP TIMER (PWRT)

El temporizador de Power-up proporciona un retardo fijo de unos 72ms desde que se produce el pulso de POR. Mientras que dura esta temporización el micro se mantiene en un estado de RESET. Este retardo PWRT permite a la tensión VDD crecer hasta un valor aceptable de alimentación para el propio micro y para el resto de circuitería que exista en la tarjeta y que se alimente desde la misma fuente de alimentación.

Existe un bit de configuración (PWRTE) que permite habilitar/deshabilitar esta temporización.

Figure 3-5: Time-out Sequence on Power-up (MCLR Tied to VDD)

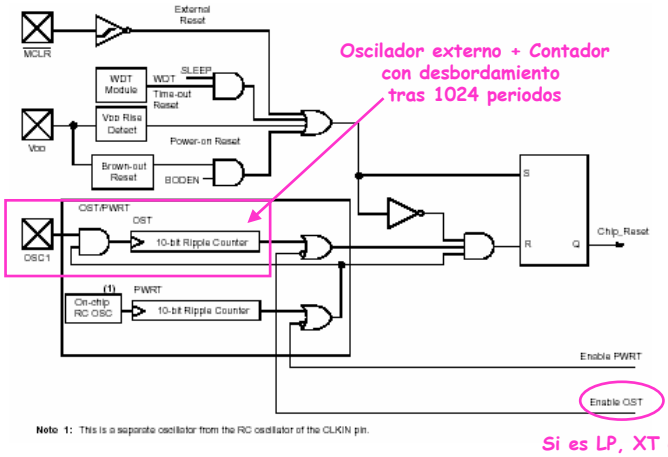







## OSCILLATOR START-UP TIMER (OST)

Aunque no sea configurable, se puede esperar después de PWRT, un tiempo adicional antes de liberar el estado de reset del MCU, para asegurar que el oscilador ha alcanzado su frecuencia estable



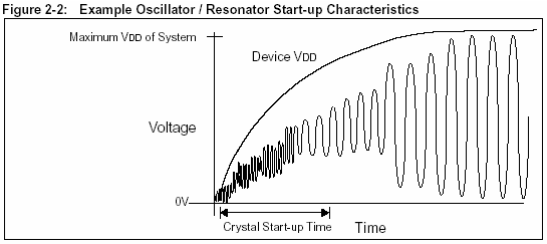

Oscilador externo + Contador con desbordamiento tras 1024 periodos

Si es LP, XT ó HS

## OSCILLATOR START-UP TIMER (OST)

La temporización de START-UP TIMER (OST) temporiza un retardo de 1024 ciclos del oscilador una vez que finaliza el retardo debido a PWRT. Esto asegura que el oscilador de cristal o resonador cerámico ha empezado a oscilar y su frecuencia es estable cuando comienza el funcionamiento del programa.

Figure 2-2: Example Oscillator / Resonator Start-up Characteristics




La temporización OST únicamente se activa si el microcontrolador está programado como modo de oscilador XT, LP ó HS. Además, únicamente se produce en algunos tipos de RESET: POR, BOR y Wake-up desde SLEEP.
La cuenta de ciclos solo comienza cuando la amplitud de la oscilación alcanza los valores umbrales de oscilación (0.3 VDD y 0.7VDD).

TABLE 12-3: TIME-OUT IN VARIOUS SITUATIONS

| Oscillator Configuration | Power-up | |
|---|---|---|
| | $\overline{PWRTE}$ = 0 | $\overline{PWRTE}$ = 1 |
| XT, HS, LP | 72 ms + 1024TOSC | 1024TOSC |
| RC | 72 ms | — |





## POWER-ON RESET (POR) EN DIVERSAS CIRCUNSTANCIAS

FIGURE 12-5: TIME-OUT SEQUENCE ON POWER-UP ($\overline{MCLR}$ TIED TO VDD)

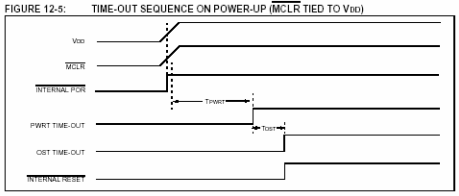



FIGURE 12-6: TIME-OUT SEQUENCE ON POWER-UP ($\overline{MCLR}$ NOT TIED TO VDD): CASE 1

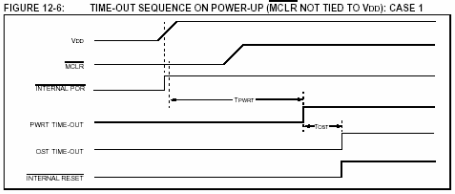



FIGURE 12-7: TIME-OUT SEQUENCE ON POWER-UP ($\overline{MCLR}$ NOT TIED TO VDD): CASE 2

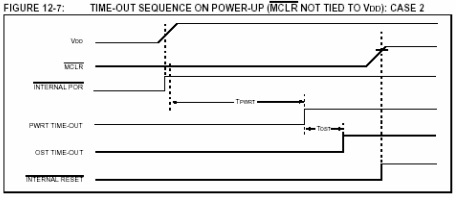
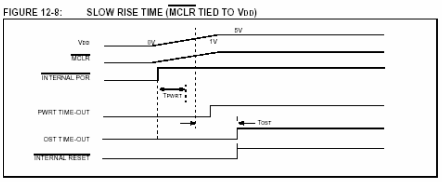



FIGURE 12-8: SLOW RISE TIME ($\overline{MCLR}$ TIED TO VDD)



## BROWN-OUT RESET

La circuitería de Brown-out disponible en el propio chip detecta si la tensión de alimentación cae por debajo de un determinado valor (BVDD) provocando un RESET del dispositivo. Esto asegura que el dispositivo no continúe con la ejecución del programa si la alimentación se sale del rango de funcionamiento válido. Brown-out RESET se utiliza principalmente en aplicaciones con alimentación desde red o desde batería donde se conmuten grandes cargas y puedan originar que la tensión de alimentación caiga temporalmente por debajo de la tensión mínima de alimentación permitida. Como hemos visto en la palabra de configuración existe un bit BODEN (o varios en otros micros) que permite habilitar (1) o deshabilitar (0) esta función. Si el BROWN-OUT está habilitado, el POWER-UP Timer también debe estarlo.

El parámetro eléctrico D005 (tipicamente 4V) es la tensión mínima permitida en la alimentación. Si la tensión de alimentación desciende por debajo de este valor durante un tiempo mayor al fijado por el parámetro 35 se producirá un RESET del microcontrolador. El RESET no está garantizado si la tensión de alimentación cae por debajo del valor D005 durante un tiempo menor del fijado por el parámetro 35.

El chip permanecerá en RESET hasta que la tensión de alimentación supere BVDD. En ese instante se inicia la temporización de POWER-UP (72 mseg) durante la que el chip se mantendrá reseteado.

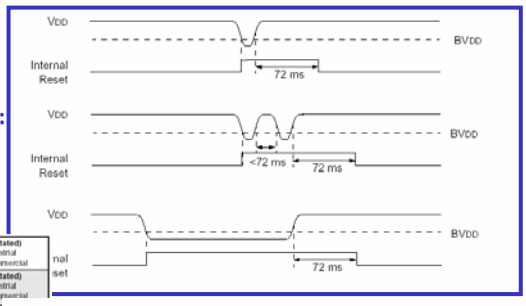
Si durante esta temporización se vuelve a producir una caida de la tensión de alimentación por debajo de BVDD, el chip volverá al estado de RESET y la temporización de Power-up volverá a arrancar desde cero cuando la tensión de alimentación vuelva a recuperarse por encima de BVDD.





## BROWN-OUT RESET

**Ejemplo de BOR y secuencia:**




| Param No. | Symbol | Characteristic/ Device | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| | | PIC16LF873/874/876/877-04 (Commercial, Industrial) | Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial 0°C ≤ TA ≤ +70°C for commercial | | | | |
| | | PIC16F873/874/876/877-04 PIC16F873/874/876/877-20 (Commercial, Industrial) | Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial 0°C ≤ TA ≤ +70°C for commercial | | | | |
| | VDD | Supply Voltage | | | | | |
| D001 | | 16LF87X | 2.0 | — | 5.5 | V | LP, XT, RC osc configuration (DC to 4 MHz) |
| D001 | | 16F87X | 4.0 | — | 5.5 | V | LP, XT, RC osc configuration |
| D001A | | | 4.5 | | 5.5 | V | HS osc configuration |
| | | | VBOR | | 5.5 | V | BOR enabled, FMAX = 5 |
| D002 | VDR | RAM Data Retention Voltage(1) | — | 1.5 | — | V | |
| D003 | VPOR | VDD Start Voltage to ensure internal Power-on Reset signal | — | Vss | — | V | See section on Power-on Reset for details |
| D004 | SVDD | VDD Rise Rate to ensure internal Power-on Reset signal | 0.05 | — | — | V/ms | See section on Power-on Reset for details |
| D005 | VBOR | Brown-out Reset Voltage | 3.7 | 4.0 | 4.35 | V | BODEN bit in configuration word enabled |

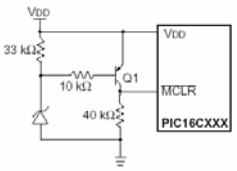
FIGURE 15-9: BROWN-OUT RESET TIMING


| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 35 | TBOR | Brown-out Reset pulse width | 100 | — | — | µs | VDD ≤ VBOR (D005) |

**Parámetros de las características eléctricas**

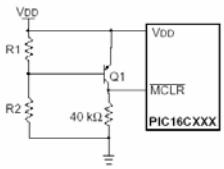
## BROWN-OUT RESET

Algunos dispositivos de la familia PIC16 no disponen de circuitería de ***BROWN-OUT***. En estos dispositivos o en aquellos en los que las características de BROWN-OUT se queden cortas (es necesario asegurar una tensión de alimentación mínima por encima de la fijada por el parámetro D005) puede resultar necesario utilizar circuitos externos al chip como los que se indican en las siguientes figuras.




This circuit will activate reset when VDD goes below (Vz + 0.7V) where Vz = Zener voltage.

Note 1: Internal Brown-out Reset circuitry should be disabled when using this circuit.
2: Resistors should be adjusted for the characteristics of the transistor.




Note 1: This brown-out circuit is less expensive, albeit less accurate. Transistor Q1 turns off when VDD is below a certain level such that:

$$V_{DD} \cdot \frac{R1}{R1 + R2} = 0.7V$$

2: Internal Brown-out Reset circuitry should be disabled when using this circuit.
3: Resistors should be adjusted for the characteristics of the transistor.





## WATCHDOG o "PERRO GUARDIAN"

• El temporizador Watchdog es un temporizador existente en el microcontrolador basado en un oscilador RC interno, independiente del oscilador del microcontrolador y que no requiere ningún componente externo. El WATCHDOG contará incluso si el reloj conectado a OSC1/CLKI y/o OSC2/CLKO está parado, por ejemplo, por la ejecución de una instrucción SLEEP ó por un defecto del cristal oscilador.

• Este oscilador RC interno no tiene nada que ver con un posible oscilador RC externo conectado a la patilla OSC1/CLKI.

• El funcionamiento o no del Watchdog se debe seleccionar en la palabra de configuración a la hora de grabar el microcontrolador: bit WDTE de la palabra de configuración a 1 -> activo (por defecto tras el borrado del microcontrolador); si está a 0 -> inactivo

• Si está activo, durante el funcionamiento normal del microcontrolador, un desbordamiento (ó time-out) del Watchdog provoca un Reset del microcontrolador (Watchdog Timer Reset). Para que no se desborde, cada cierto tiempo y antes de que llegue al límite, se debe ejecutar una instrucción CLRWDT que "limpia" el Watchdog y le hace comenzar una nueva cuenta desde cero.

• Si el dispositivo está en modo dormido, un desbordamiento del watchdog provoca que el micro despierte y continue con el funcionamiento normal (Watchdog Timer Wake-Up) con la instrucción que sigue a SLEEP (la que lo mandó a dormir)

• El bit $\overline{TO}$ del registro STATUS se pone a cero tras un desbordamiento del Watchdog y nos permite conocer tal circunstancia de desbordamiento.

## WATCHDOG o "PERRO GUARDIAN" (II)

• El time-out mínimo, tiempo que tarda en desbordar (sin postscaler) para el WATCHDOG viene dado por el siguiente parámetro:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 31* | TWDT | Watchdog Timer Time-out Period (no prescaler) | 7 | 18 | 33 | ms | VDD = 5V, -40°C to +85°C |

• Ese tiempo, como se puede comprobar, es muy variable al depender de una red RC

• Un postscaler es un divisor de frecuencia que puede hacer que se cuente el número de desbordamientos del WDT antes y hacer que el tiempo que tarda en resetear al microcontrolador sea más largo. El inconveniente es que ese divisor de frecuencia está compartido con el TMR0 y por tanto, si se usa para el TMR0 no se puede usar para el WATCHDOG y viceversa.

• El divisor de frecuencia del WATCHDOG viene definido por unos bits del registro OPTION:

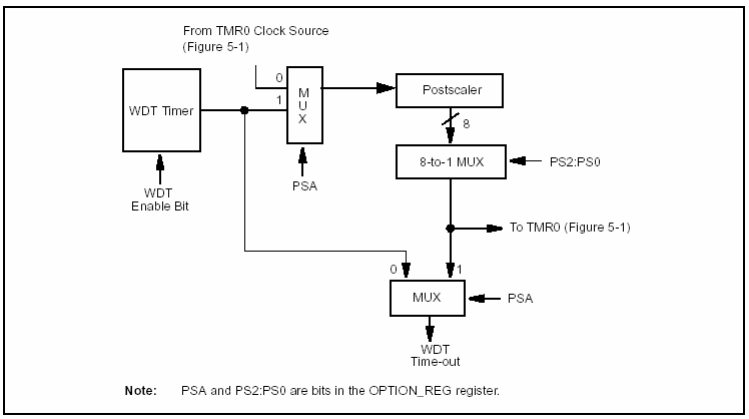
PSA: a quién se le asigna el divisor

PS2-PS1-PS0:
cuál es el factor de división de la frecuencia





## DIAGRAMA DE BLOQUES DEL WATCHDOG



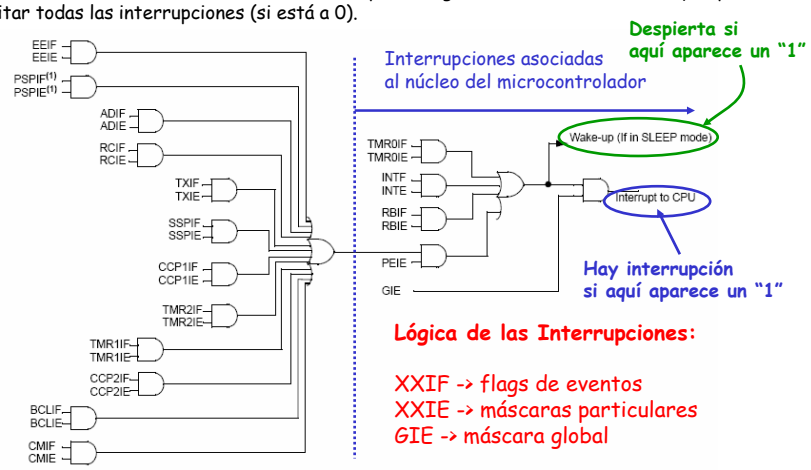

## REGISTROS ASOCIADOS AL FUNCIONAMIENTO DEL WATCHDOG

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 2007h | Config. bits | (1) | BODEN(1) | CP1 | CP0 | $\overline{\text{PWRTE}}$(1) | WDTE | FOSC1 | FOSC0 |
| 81h, 181h | OPTION_REG | RBPU | INTEDG | T0CS | T0SE | PSA | PS2 | PS1 | PS0 |

## LAS INTERRUPCIONES DEL MCU

Los PIC16F87XA tienen hasta 15 posibles fuentes de interrupción.

Se dispone de un bit de habilitación de interrupciones global GIE (INTCON<7>) que permite deshabilitar todas las interrupciones (si está a 0).




**Lógica de las Interrupciones:**

XXIF -> flags de eventos
XXIE -> máscaras particulares
GIE -> máscara global





## ¿QUE INTERRUPCIONES DE PERIFERICOS PUEDEN "DESPERTAR" AL MICROCONTROLADOR?

- Escritura o lectura del puerto esclavo paralelo (PSPIF)
- Desbordamiento del TMR1, siempre que el timer 1 este contando pulsos externos en modo contador asíncrono (TMR1IF).
- Captura de un módulo CCP (CCPxIF).
- Comparacion del modulo CCP en modo "disparo de evento especial". El TMR1 debe contar pulsos externos (CCPxIF).
- Módulo SSP al detectar un bit de START ó STOP ó una colisión en el bus (SSPIF ó BCLIF)
- Módulo SSP al transmitir o recibir en modo esclavo en SPI/I2C(SSPIF).
- Módulo USART al RX o TX (modo síncrono) (RCIF ó TXIF).
- Al finalizar una conversión A/D siempre que el reloj de la conversión sea el RC interno(ADIF).
- Al completar una escritura en EEPROM (EEIF).
- Al modificarse el estado de salida de alguno de los comparadores (CMIF).
- Interrupcion externa por flanco en el pin RB0/INT (INTF).
- Interrupcion por cambio en los valores de los pines RB4 a RB7 del PORTB (RBIF).

Los otros periféricos no pueden generar interrupciones ya que para su funcionamiento necesitan el reloj del sistema y este se detiene en el modo "dormido".

## LAS INTERRUPCIONES DEL MCU

• Cuando el bit GIE está a 1, si una interrupción tiene su flag a 1 y sus bits de habilitación a 1, el microcontrolador terminará la instrucción que se está ejecutando en ese instante y, a continuación, pasará a ejecutar la posición 4 de la memoria de programa que corresponde a la posición del vector de interrupción y **que es el mismo para todas las interrupciones**, por software se debe detectar el origen y se debe decidir la prioridad

• Las fuentes de interrupción pueden deshabilitarse individualmente utilizando sus máscaras o bits de habilitación (bits acabados en "E").

• El bit GIE se pone a 0 tras un RESET, al principio las interrupciones están desactivadas.

• Al producirse el salto a la rutina o programa de tratamiento de la interrupción, el bit GIE se pone a 0 deshabilitando el resto de interrupciones, salvo que por software se vuelva a poner a 1 ese bit GIE. El retorno de programa de tratamiento de interrupción (RETFIE) coloca en la máscara global GIE el valor 1, además de recuperar el PC de la pila hardware.

• Los bits de flags (XXXF) pueden ponerse a 1 independientemente de que sus bits de habilitación (XXXE) estén o no a 1, ya que indican eventos.





## LAS INTERRUPCIONES DEL MCU

• Las interrupciones por flanco en el pin RB0/INT, por cambio en los pines RB4 a RB7 del PORTB y por OVERFLOW del TIMER0 tienen sus bits de flags y habilitación en el propio registro INTCON (son interrupciones asociadas al núcleo del microcontrolador).

• Para que estas posibles fuentes soliciten interrupción únicamente necesitan tener su correspondiente máscara bit de habilitación a 1 y que el GIE esté a 1.

• El resto de posibles fuentes de interrupción tienen sus bits de flags en registros de funciones especiales denominados PIR1 y PIR2. Sus bits de habilitación están en los registros PIE1 y PIE2. Para que puedan solicitar interrupción, aparte de que su bit correspondiente de habilitación esté a 1 deben pasar un filtro adicional a los que tienen las asociadas al núcleo. Debe cumplirse que los bits GIE (máscara global) y PEIE (máscara de periféricos) estén a 1.

• Los bits de flag de las interrupciones (con la excepción de algunos de los relacionados con los módulos de comunicación serie) deben ponerse a 0 por software dentro del programa de tratamiento de interrupción y antes del retorno del mismo. Si no fuera así, al ejecutarse el RETFIE (GIE a 1) se volvería a entrar de forma recursiva en la rutina de interrupción y no se saldría de ella. Volvería a aparecer un "1" en la salida de la lógica de interrupción

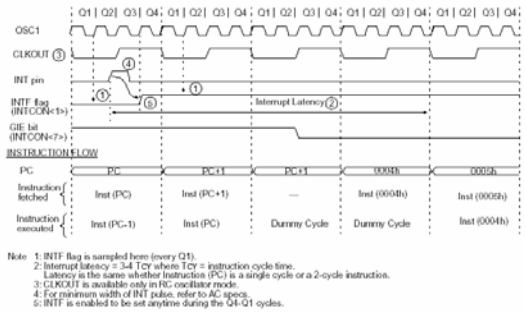
• Los flags se ponen a 1 cuando se produce el evento pero se deben poner a 0 por software (instrucción del tipo bcf REG?,xxIF).

## LAS INTERRUPCIONES DEL MCU: LATENCIA DE INTERRUPCION

• Se define latencia de interrupción como el tiempo que pasa desde que se produce el evento que provoca la interrupción (flanco en INT, desbordamiento de TMR0, etc) hasta que se inicia la ejecución de la instrucción de la posición de memoria de programa 0004h.

• Para las interrupciones síncronas (internas típicamente como por ejemplo el desbordamiento de TMR0), la latencia es de 3TCY.

• Para interrupciones asíncronas (tipicamente externas como por ejemplo un flanco en INT ó PORTB), la latencia estará entre 3 – 3.75 TCY dependiendop del momento del ciclo de instrucción en el que se produce el evento que provoca la interrupción. La latencia es la misma tanto si el evento se produce en medio de una instrucción de un solo ciclo como en una instrucción de 2 ciclos (saltos).




Note 1: INTF flag is sampled here (every Q1).
2: Interrupt latency = 3-4 TCY where TCY = instruction cycle time. Latency is the same whether Instruction (PC) is a single cycle or a 2-cycle instruction.
3: CLKOUT is available only in RC oscillator mode.
4: For minimum width of INT pulse, refer to AC specs.
5: INTF is enabled to be set anytime during the Q4-Q1 cycles.





## LAS INTERRUPCIONES DEL MCU: Salvar el Contexto del Programa Principal

• Cuando se produce una interrupción **solo se guarda en la pila hardware interna el valor del PC**.

• Normalmente, se **deberán salvar algunos otros registros** para no perder su contenido después de regresar de la rutina de interrupción, máxime cuando se ignora cuándo se va a producir el salto a ese programa de tratamiento.

• Típicamente, estos registros son al menos el **W y el STATUS** (imaginemos que se produce el salto a la rutina de interrupción en un punto donde se iba a testear un bit del registro STATUS ó se iba a escribir en un registro el contenido de W).

• También puede resultar interesante guardar el registro PCLATH, especialmente si en la rutina de interrupción se cambia de página de memoria de programa.

• **Como no hay pila en RAM**, ni instrucciones "PUSH" y "POP", hay que reservar posiciones de memoria en RAM que habitualmente denominaremos W_TEMP, STATUS_TEMP y PCLATH_TEMP donde se guardan los valores de W, STATUS y PCLATH al entrar en la rutina de interrupción.

## LAS INTERRUPCIONES DEL MCU: Salvando el Contexto

• Cuando se produce una interrupción, al ser imprevisible su aparición no resulta posible conocer "a priori" el banco de RAM en el que nos encontramos cuando entra el programa de tratamiento de la interrupción (PTI). La información sobre el "banco" actual está en el registro STATUS que debemos salvar para poder recuperar el banco al final del PTI. Pero para guardarlo necesitamos usar el registro W como intermediario, luego el primer paso será salvar W.

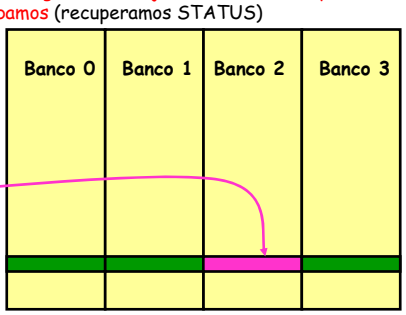
• La posición W_TEMP (de propósito general) puede estar en cualquiera de los 4 bancos de RAM, aunque se le haya asignado una dirección de 9 bits, sólo los 7 bits más bajos (posición relativa en un banco de RAM) pueden asegurarse. Eso no genera ningún error si justo antes de recuperar W nos volvemos a situar en el banco en el que estábamos (recuperamos STATUS)

Si guardamos en el banco 2, recuperamos del banco 2, si era en el banco 0, recuperamos del banco 0, etc.

```
W_TEMP EQU 0x20
...

PTI  movwf W_TEMP
```

Si estaba en el banco 2 al saltar interrupción




• Luego deben respetarse las posiciones "gemelas" en otros bancos o bien usar una posición que esté direccionada en todos los bancos de RAM





## LAS INTERRUPCIONES DEL MCU: Salvando el Contexto del Programa Principal

• Para los PIC16F876/877, las 16 últimas posiciones de cada banco son comunes (se accede a la 0x70 a la 0x7F desde cualquier banco), resulta interesante colocar el registro W_TEMP en alguna de esas posiciones, de forma que se reserva un único registro físico y no los 4 que exigiría el preservar posiciones equivalentes de bancos.

• Tras salvar W, se debe proceder a salvar el registro STATUS. Pero no se puede hacer con la instrucción:

```
movf    STATUS,W
```

ya que la propia instrucción modifica STATUS antes de guardarlo en W.
Por tal motivo se debe utilizar:

```
swapf STATUS,W
```

lo que supone salvar STATUS pero con los nibbles intercambiados, esto no tendrá trascendencia si al recuperarlo tenemos la precaución de hacer lo mismo.

• Una vez a salvo STATUS (aunque "girado") en W ya podemos situarnos en el banco de RAM en el que se quiera trabajar dentro del PTI y seguir salvando los registros pero ahora conociendo perfectamente el banco en el que se guardan.

## LAS INTERRUPCIONES DEL MCU:
## Secuencia a seguir para salvar el Contexto del Programa Principal

```
;Para salvar el contexto no podemos emplear la instrucción MOVF ya que afecta
;al registro STATUS,para evitarlo empleamos la instrucción SWAPF
        movwf W_tmp         ;Salvamos el registro W
        swapf STATUS,W      ; y el registro STATUS "girado" en W
        bcf STATUS,RP0      ;Aseguramos el paso al banco 0
        bcf STATUS,RP1      ;poniendo a 0 los dos bits de selección de banco
        movwf STATUS_tmp ;Guardamos en el banco 0 STATUS girado
        movf PCLATH,W       ;Salvamos también PCLATH en W
        movwf PCLATH_tmp ;y ahora en una posición auxiliar del banco 0
```

Siempre se debe hacer así o de una manera similar (si se guardan más registros) al principio de un PTI

Sugerencia: se puede definir una Macro denominada SALVAR o PUSH con todas esas instrucciones





## LAS INTERRUPCIONES DEL MCU:
## Recuperando el Contexto del Programa Principal

· Para recuperar los registros almacenados y restaurarlos antes de volver al programa principal se debe seguir la secuencia "inversa" a la de almacenamiento y en un determinado orden.

· El registro PCLATH será el primero en ser recuperado, asegurándonos primero que nos ubicamos en el banco en el que lo guardamos (el 0 en el caso anterior)

```
movf    PCLATH_TEMP,W
movwf   PCLATH
```

· A continuación debemos recuperar STATUS, pero recordando que lo guardamos "girado". A la hora de recuperarlo le "metemos" otro giro más en los *nibbles* y ya lo tenemos en W como estaba antes de entrar en el programa de tratamiento. Acto seguido se lo pasamos a STATUS con una instrucción MOVWF ya que ésta no altera para nada el registro STATUS

```
swapf   STATUS_TEMP,W
movwf   STATUS
```

· Estamos en el banco donde estábamos al entrar en el PTI, luego de la posición dónde guardamos W lo sacamos, con la precaución de no usar la instrucción MOVF ya que afectaría a STATUS. El truco consiste en usar dos instrucciones SWAPF que no alteran STATUS

```
swapf   W_TEMP,F
swapf   W_TEMP,W
```

LAS INTERRUPCIONES DEL MCU:
Secuencia completa a seguir para recuperar el Contexto del Programa Principal

```
;Para recuperar los registros salvados no podemos usar MOVF porque modifica a STATUS,
;para evitarlo usamos la instrucción SWAPF
        movf    PCLATH_tmp,W    ;Recuperamos PCLATH
        movwf   PCLATH          ;directamente
        swapf   STATUS_tmp,W    ;Recuperamos el registro STATUS con un SWAPF
        movwf   STATUS          ;ahora estamos en el banco de partida
        swapf   W_tmp,F         ;Recuperamos también el W con dos SWAPF
        swapf   W_tmp,W         ;para evitar la instrucción MOVF

        retfie                  ;Ahora ya podemos retornar del PTI
```

Siempre se debe hacer así o de una manera similar (si se guardan más registros) al final de un PTI

Sugerencia: se puede definir una Macro denominada RECUPERAR o POP con todas esas instrucciones





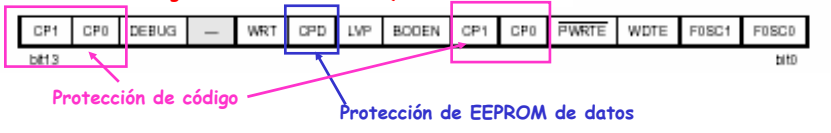
Protección de la Memoria

- Tanto la memoria de programa como la EEPROM de datos del microcontrolador pueden ser protegidas mediante hardware para que no puedan ser leídas externamente

- Estas protecciones son configurables por el usuario en la palabra de configuración por tanto no son modificables en tiempo de ejecución.

- La protección puede ser parcial o total y depende del tipo de microcontrolador

Palabra de configuración en PIC16F87x (CONFIG : 0x2007




Hay dos "pares" de bits CP1-CP0, hay que grabar los dos pares por igual para lograr la protección del código de una manera segura, si se hace desde el entorno MPLAB con la ventana de Configuración, lo grabación de la palabra CONFIG se hará cargando los dos pares de bits

## Protección de la Memoria

· Cuando se está grabando el código, resulta posible una primera lectura (verificación) antes de hacer efectiva la protección. Primero se graba la memoria de programa, luego se lee para verificar la correcta escritura y finalmente se graban los bits de protección. Si activaran la protección, ya no serían posible posteriores verificaciones.

· Una vez que se ha activado la protección, no resulta posible desproteger el código modificando exclusivamente los bits de CONFIG:

- En los microcontroladores con memoria FLASH se debería borrar totalmente la memoria de programa para poder modificar de nuevo CONFIG, pero si se ha borrado toda la memoria… ¿qué queremos proteger?

- En los microcontroladores con EPROM y ventana para borrado por luz UV, la palabra de configuración se puede borrar igual que el resto de la memoria de programa, salvo los bits de protección de código,