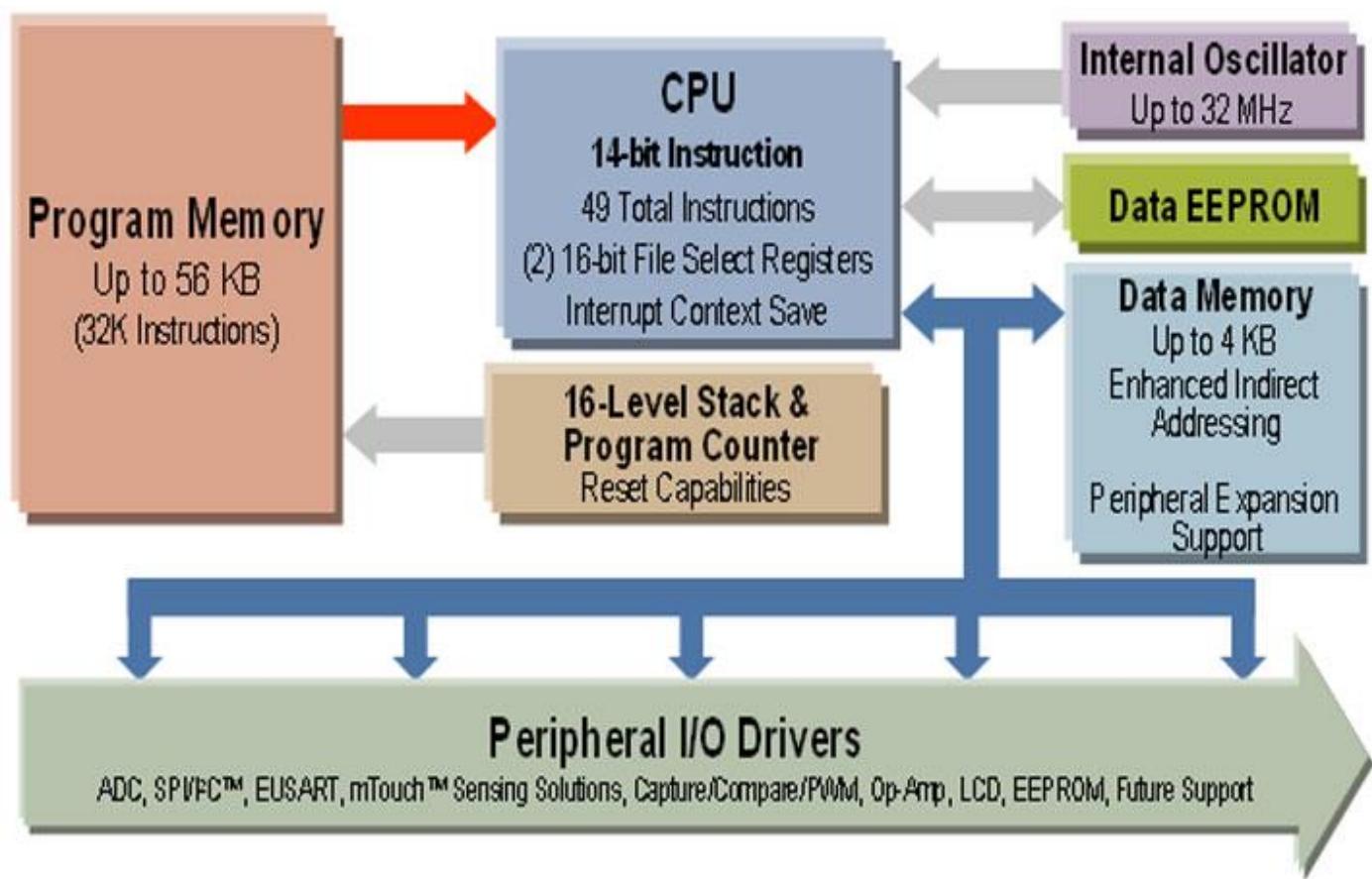


Familia de gama media mejorada	2
Arquitectura	7
Conjunto de instrucciones	17
Bits de configuración	22
Resumen de bits de configuración en C	34
Memoria de datos	36
Memoria del programa	45
Periféricos	49
ES digitales	50
Convertidor de analógico a digital	58
Celda lógica configurable	65
Herramienta de configuración de CLC	72
CLCxCON	76
CLCxSEL0-1	78
CLCxGLS1-4	81
CLCxPOL	84
Generador de forma de onda complementaria	86
Ejemplo de GTC	94
Oscilador controlado numéricamente	99
Interrupción	106
Ejemplo de interrupción	114
Ejemplos de programación	115

# Familia de gama media mejorada

**MCU PIC de 8 bits de® rango medio mejorado PIC12F1xxx, PIC16F1xxx**

- 49 (14 bits de ancho) instrucciones fáciles de aprender
- Memoria de programa direccionable de 32K word (56 KB)
- 4 KB de RAM (máx.)
- Pila de hardware de 16 niveles
- 2 registros de selección de archivos (16 bits)
- Manejo de interrupciones de hardware con guardado de contenido
- Conjunto de funciones avanzadas, comunicaciones serie múltiples y capacidad de control del motor



Haga clic en la imagen para ampliarla.

Microchip continúa invirtiendo en su línea de microcontroladores PIC de 8 bits para proporcionar una amplia cartera de productos que satisfaga las necesidades de los clientes actuales y futuros. El nuevo núcleo de gama media mejorada se basa en los mejores elementos del núcleo de gama media y proporciona un rendimiento adicional, al tiempo que mantiene la compatibilidad con los MCU PIC de gama media para una verdadera migración del producto.

## Entrenamiento a su propio ritmo

El material de estos módulos de capacitación existe en otras partes de este sitio en un formato de referencia general. Sin embargo, los módulos de capacitación lo presentan en una secuencia organizada paso a paso para ayudarlo a aprender el tema desde cero.

## Tutorial / Título de la clase

Programación de Microchip PIC16F usando MCC (Volumen 2)

Generador de forma de onda complementario en 8 bits

Introducción a la celda lógica configurable

Introducción al oscilador controlado numéricamente

Programación de Microchip PIC16F usando MCC (Volumen 1)

Introducción al temporizador PIC® MCU 0

Introducción a la arquitectura de MCU mejorada PIC16F1

## Preguntas Frecuentes

### Tema

¿Por qué SE PRUEBA INTCONbits.PEIE en código de interrupción de alta prioridad generado por MPLAB® Code Configurador PIC18F26K22?

Desactivación de detección cruzada cero (ZCD)

¿Por qué el PIC16F1619 DAC1OUT en una placa Curiosity siempre está a 0 V en modo de depuración?

¿Por qué recibo un error de sintaxis de C al compilar una instrucción de rotación a la izquierda asm("RLF aux, w");?

¿Por qué no puedo conducir mi MOSFET cuando estoy conectado a PWM?

¿Cuál es el valor de resistencia del pull-up débil interno?

¿Cuál es la protección interna proporcionada por el silicio, como los diodos de descarga?

¿Cuál es el tamaño de instrucción de un MCU PIC®?

¿Qué es Q1 en un comparador PIC16F?

¿Qué significa el sufijo "-ICD" en un MCU?

¿Cuáles son los microcontroladores PIC® base utilizados en la serie ARxxxx?

La señal de salida TX de USART muestra dos bits de parada

PIC16F870/871 - ¿Cuál es el procedimiento de reinicio después del evento de error de fotograma (FERR)?

PIC16F - No se puede cambiar la funcionalidad del pin CLKIN

¿Las tablas de pines del dispositivo muestran qué pines son tolerantes a 5V?

Probar paquetes TSOP, SSIC, QFN, DFN en una placa de prueba

¿Cómo puedo crear un amplificador de ganancia programable (PGA) utilizando una salida OP Amp y DAC de un MCU PIC®?

¿Cuál es la diferencia entre la resistencia de bytes (ED) y el número de ciclos totales de borrado/escritura antes de la actualización (TREF)?

PIC18F87xx - ¿Cómo se selecciona la funcionalidad de un pin?

Impacto de Ripple en Vdd cuando se utiliza un microcontrolador Microchip

I<sup>2</sup>C no puede escribir en SSPBUF

¿Cómo se lee la temperatura interna en el PIC18F?

¿Cómo depuro el PIC16F1503 de una placa Curiosity?

La salida PWM se está agotando mientras se configura el CCP

PIC18F25K80 - I<sup>2</sup>C deja de funcionar en un sistema controlado por interrupciones

¿Cómo se lee la memoria EEPROM de un PIC18F46J50?

¿Cómo puedo encontrar un microcontrolador compatible pin-to-pin para migrar mi proyecto?

Ejemplo de código ECAN con el modo FIFO mejorado (modo 2)

Maneje el RTCC utilizando una entrada de reloj digital externa de 32,768 kHz

Significado del sufijo de pieza personalizada

Controlador LED de corriente constante con PIC18

Calcular los valores del condensador para un oscilador de cristal

¿Puede el sensor de temperatura interno en los dispositivos PIC® medir la temperatura ambiente?

Implementación de la interfaz SMI en un PIC® de 8 bits

Inicio lento del microcontrolador PIC® en temperaturas de congelación

PIC10F - Error de memoria de calibración no válida

¿Cómo se calcula la frecuencia de muestreo para un ADC SAR en PIC16F MCU?

Cambio de la velocidad del oscilador durante el tiempo de ejecución en el PIC16F

¿Puede el RTCC en un MCU PIC® sin un VBAT funcionar con una batería?

¿Cuál es el problema más común que se encuentra al comunicarse con un módulo PIC® MCU SPI?

Necesidad de encontrar un dispositivo con más RAM

Migración a una familia PIC16 más reciente

VCAP externo en PIC16F

Los pines EUSART no funcionan en PORTA

¿Puede un PIC16F con USB ejecutarse en el oscilador interno?

¿Puede el ADC medir 5 V si el VDD es de 3,3 V?

¿Puedo reprogramar la calibración del oscilador en un dispositivo PIC®?

¿Por qué necesito usar un amplificador de ganancia de unidad para medir el voltaje de salida DAC de un PIC16F1xxx?

¿Dónde puedo encontrar ejemplos de código para un controlador PID con el acelerador matemático PIC16F161x?

¿Puedo conectar VUSB3V3 a VDD (+3,3 V) para el funcionamiento USB en el PIC16F1455?

¿Se pueden utilizar tanto UART como I²C en PIC12F1840 o PIC16F1829?

Conexión de pines NC

¿Se puede leer el CRC de un código PIC protegido?

¿Dónde puedo encontrar la referencia de voltaje ADC?

Agregar un número de serie a un PIC18

Nota de la aplicación sobre los fundamentos de la LCD y el módulo de controlador LCD para MCU PIC® de 8 bits

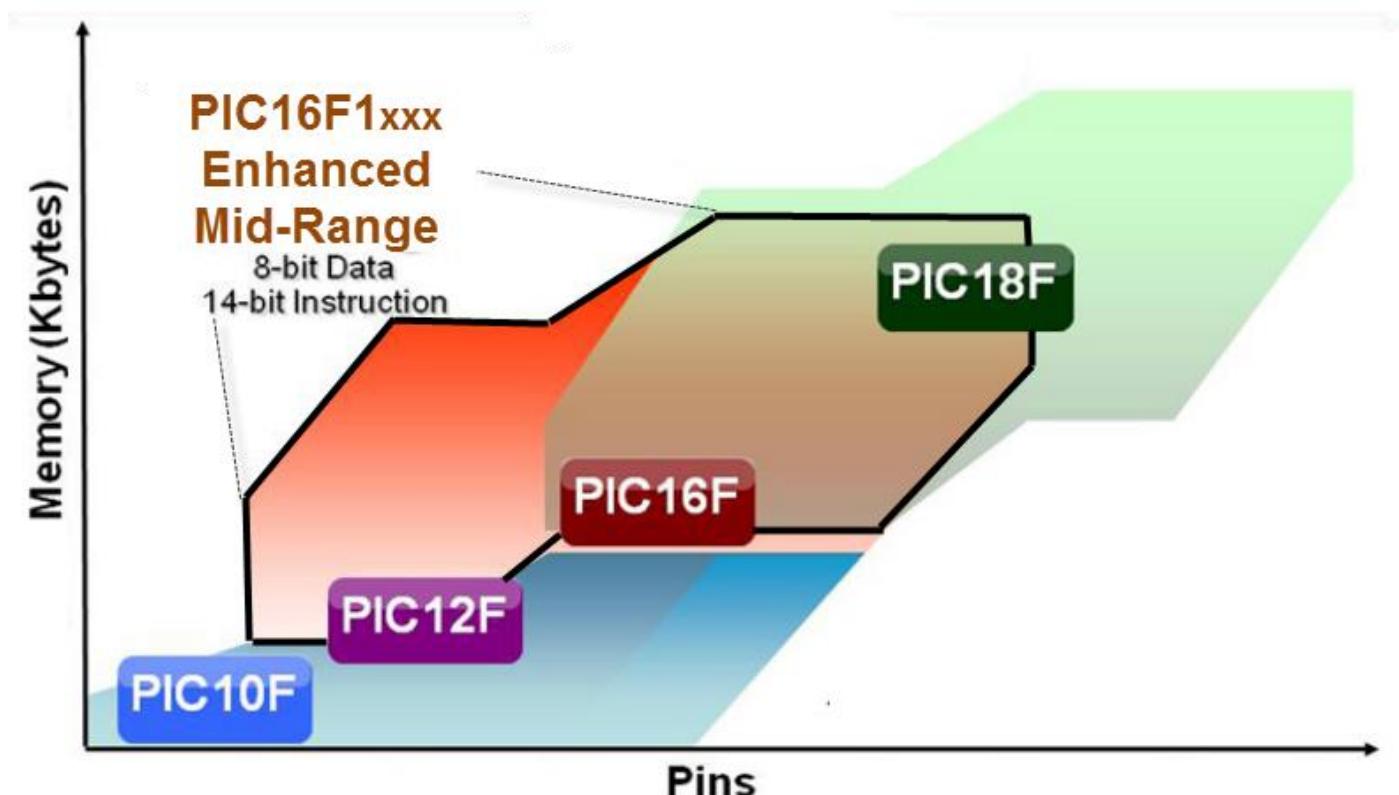
¿Por qué no puedo depurar con un PIC16F18877? ¿Por qué funciona en modo de lanzamiento pero no en modo de depuración?

¿Cuáles son las diferencias entre las subfamilias pic® MCU de 8 bits?

# Descripción general mejorada de la arquitectura PIC MCU de rango medio

## Resumen

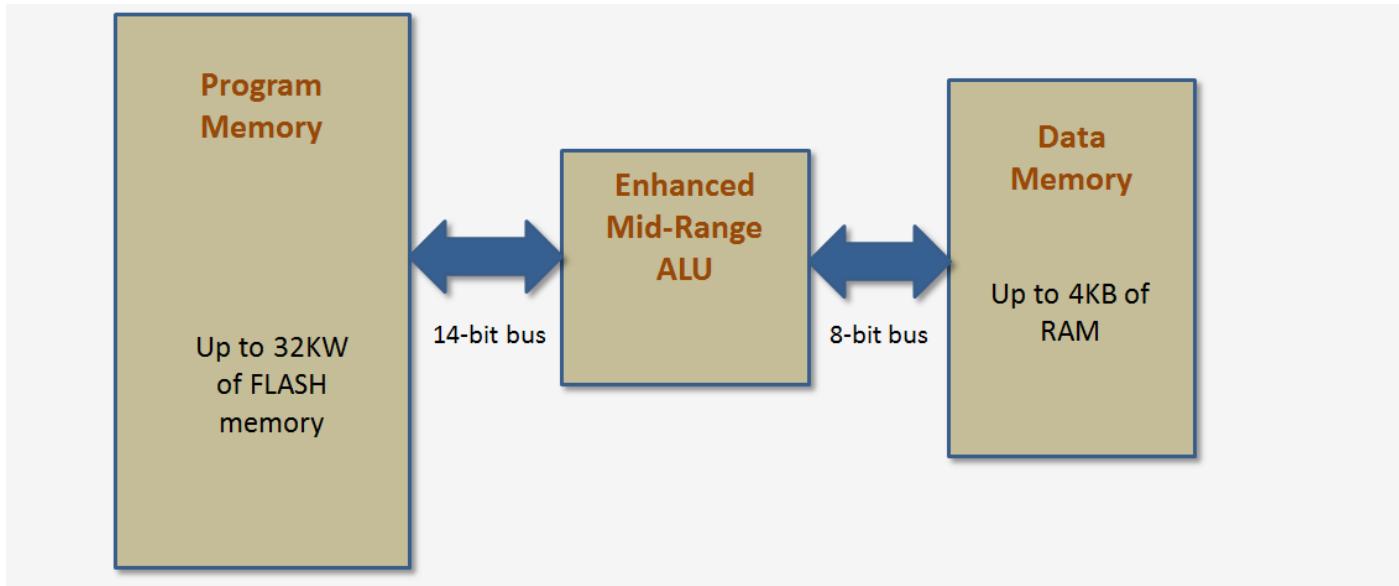
Familias de MCU de 8 bits de Microchip



La familia PICMCU de 8 bits de rango medio mejorado PIC16F1xxx abarca una amplia gama de tamaños de memoria y pines de E/S. Esta página presenta las características arquitectónicas clave de la familia pic16F1xxx de MCU. En esta página se proporcionan enlaces a los detalles técnicos necesarios para implementar aplicaciones en la familia de MCU PIC de rango medio mejorado.®

## Arquitectura de Harvard

Los MCU PIC de rango medio mejorados utilizan una arquitectura Harvard de doble bus.®



## Bus de instrucción

Las instrucciones del programa se introducen en la ALU desde la memoria del programa FLASH a través del bus de instrucciones de 14 bits. En cada ciclo de reloj de instrucciones se lee una palabra de programa de 14 bits en la ALU

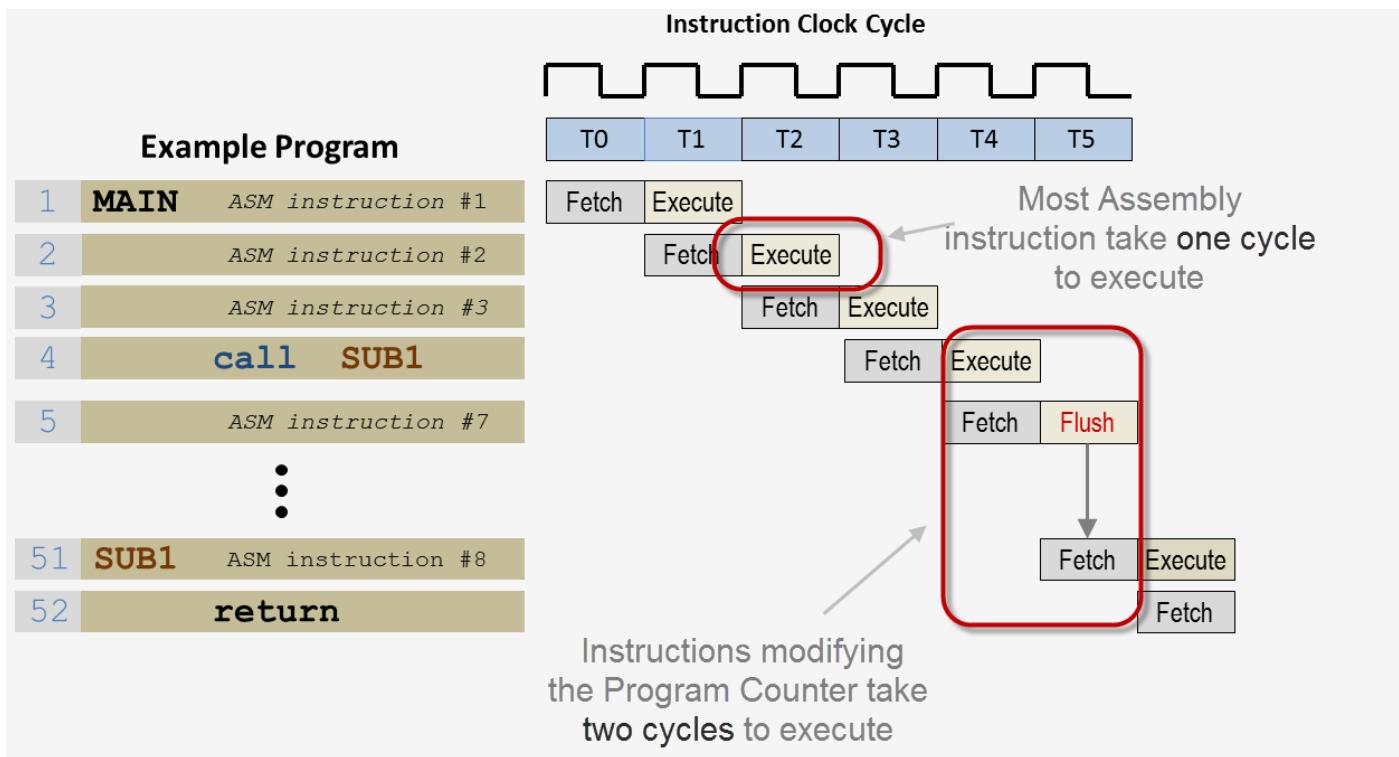
## Bus de datos

Un bus de datos de 8 bits conecta la ALU al espacio de memoria de datos. Durante cada instrucción, la ALU puede leer datos de la ubicación de la memoria de datos, modificar los datos y, a continuación, escribir los datos en la memoria.

## Instrucción Pipelining

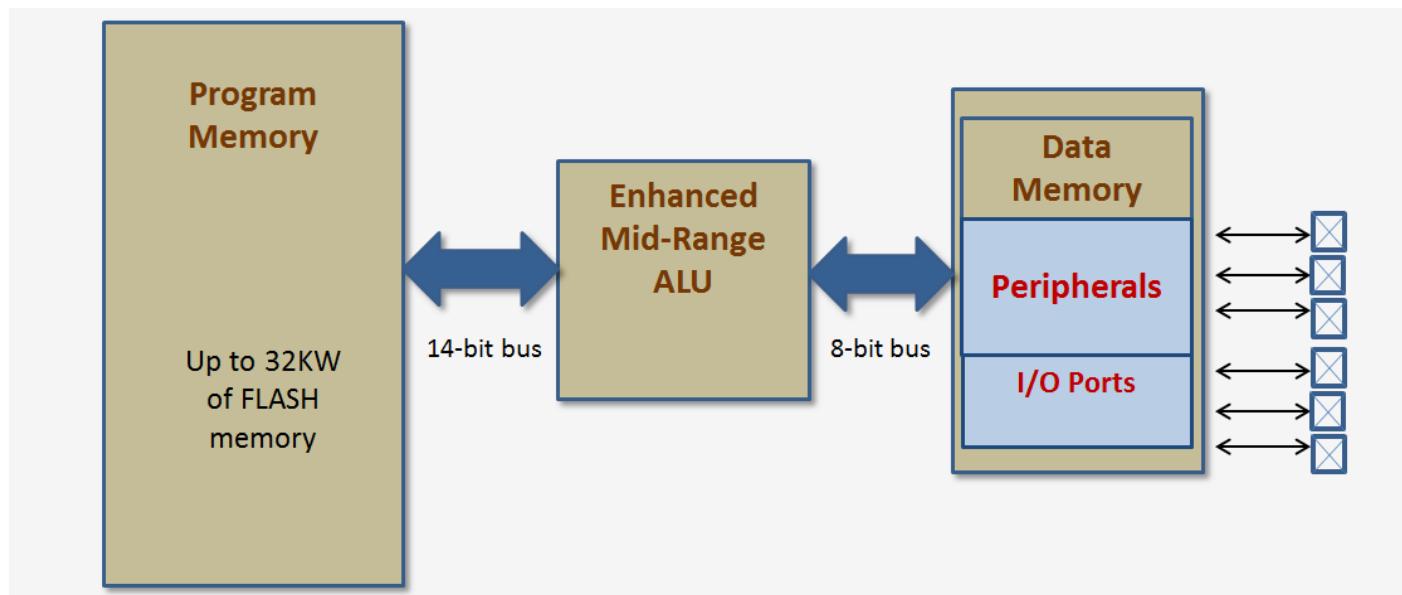
La arquitectura de doble bus mejorada del PIC de rango medio proporciona una canalización de instrucciones de dos etapas. Uno cada ciclo de reloj se ejecutan dos fases de instrucción:

1. La siguiente instrucción es "**fetched**" de la memoria del programa
2. La instrucción actual se "**ejecuta**" y lee/modifica/escribe la memoria de datos (si es necesario)



## Periféricos asignados a memoria

Una mirada más cercana a la sección Memoria de datos del PICMCU de rango medio mejorado muestra que los registros que controlan los periféricos y los puertos de E/S se accede leyendo o escribiendo en direcciones de memoria de datos específicas. Esta asignación de periféricos a la dirección de memoria simplifica enormemente el aprendizaje de cómo programar el PIC de rango medio mejorado.®



La [página memoria de datos](#) del tutorial de rango medio mejorado ofrece una descripción completa junto con ejemplos de programas para acceder a los periféricos asignados a la memoria

## Conjunto de instrucciones ortogonales

Los MCU PIC de rango medio mejorados tienen 49 instrucciones cada uno. Las instrucciones que acceden directamente a las direcciones de memoria de datos se ejecutan en un ciclo de instrucciones. Instrucciones que provocan un cambio en el contador del programa BRA, GOTO, RETURN, CALL, .. etc) tomar dos ciclos de instrucción para ejecutar.<sup>®</sup>

Al asignar los registros de E/S y periféricos a las direcciones de memoria, los MCU PIC no necesitan instrucciones especiales para las operaciones de E/S ni para establecer registros periféricos. Escribir en un puerto de E/S o configurar un periférico es una simple escritura en una ubicación de memoria. La lectura del valor de un pin de entrada, un registro de resultados de ADC o un temporizador es una lectura simple de una ubicación de memoria. Mediante el uso de un pequeño número de instrucciones ortogonales, los MCU PIC de rango medio mejorado son fáciles de programar, usan menos silicio para construir y consumen menos energía.

## Ejemplos de implementación de instrucciones

Task	C Language	Assembly
<b>WRITE</b> a value to a variable	var1 = 7 ;	<b>movlw</b> 7 <b>movf</b> var1
<b>WRITE</b> a value to a Register	T1CON = 0x72;	<b>movlw</b> 0x72 <b>movf</b> T1CON
<b>WRITE</b> to an Output Latch	LATD = 0xFF ;	<b>movlw</b> 0xFF <b>movf</b> LATD
<b>READ</b> an input PORT	var1 = PORTB;	<b>movf</b> PORTB <b>movwf</b> var1
<b>READ</b> a variable's value	var1 = var2;	<b>movf</b> var2 <b>movwf</b> var1
<b>READ</b> an SFR into another SFR	CCPR1L = ADRESH;	<b>movf</b> ADRESH <b>movwf</b> CCPR1L

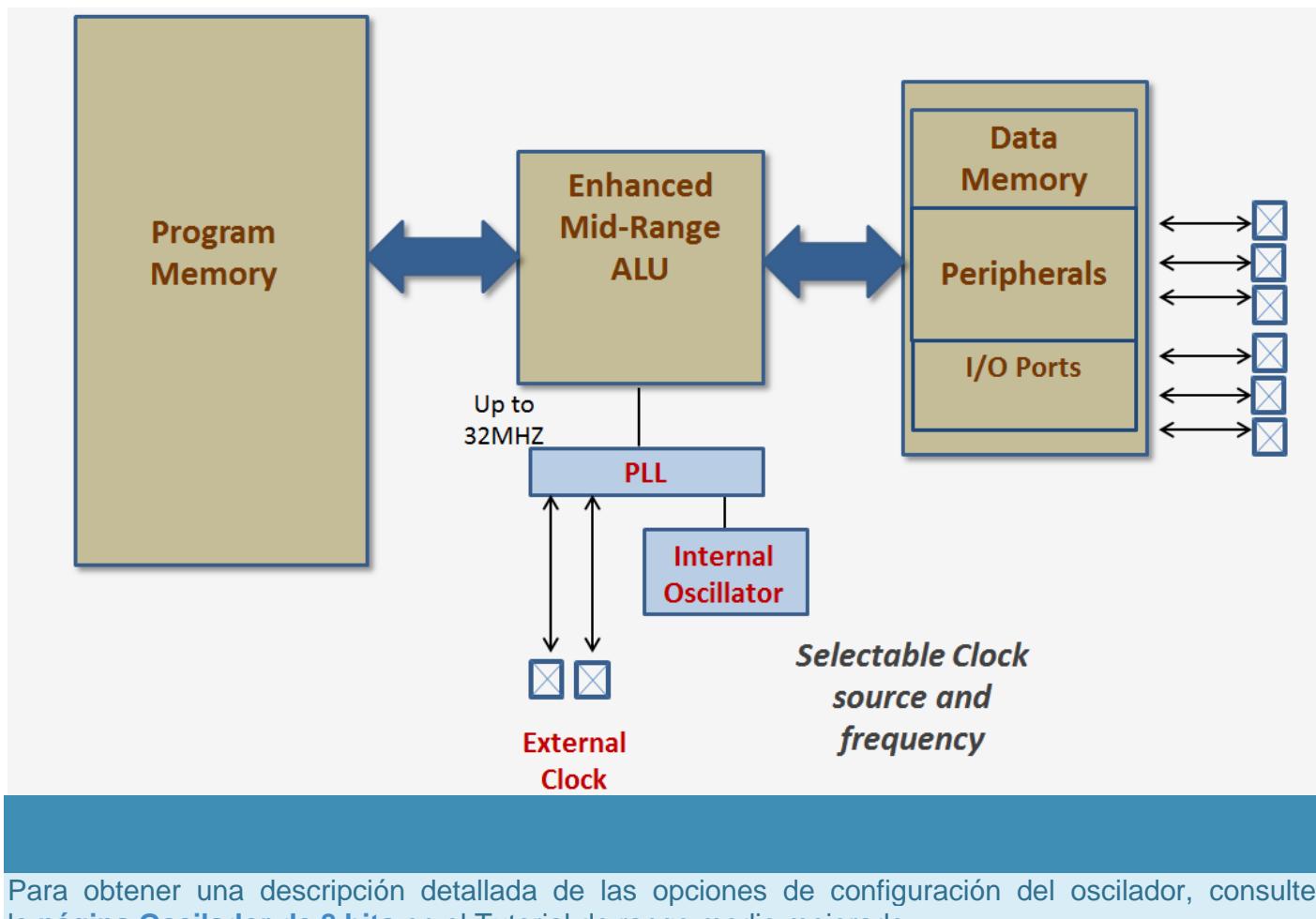
Para obtener una lista detallada del conjunto de instrucciones y una discusión completa del tiempo de instrucción, consulte la página [Conjunto de instrucciones](#) del tutorial rango medio mejorado.

## Opciones de sincronización flexibles (hasta 32 MHz)

Seleccionado por los [bits de configuración](#) del MCU PIC, el reloj del sistema tiene las siguientes propiedades:<sup>®</sup>

- Fuente opcional (**oscilador interno o circuitos externos**)
- Opciones de velocidad **flexibles de hasta 32 MHZ**
- **Arranque de dos velocidades:** permite que el sistema ejecute software de inicialización mientras el oscilador externo se estabiliza
- **Comutación de reloj:** la fuente de reloj del sistema se puede cambiar entre fuentes de reloj externas e internas a través del software.

- **Monitor de reloj a prueba de fallos:** cambia al oscilador interno en caso de una falla del reloj externo



Para obtener una descripción detallada de las opciones de configuración del oscilador, consulte la [página Oscilador de 8 bits](#) en el Tutorial de rango medio mejorado.

## E/S digitales

Casi todos los pines del MCU Enhance Mid-Range PIC se pueden utilizar como pin de entrada o salida digital. Los Pines digitales comparten estos atributos:

- Supervisar las entradas digitales
- Control de dispositivos digitales
- Pull-ups internos débiles
- Multiplexado con periféricos
- Alta capacidad de accionamiento (hasta 25 mA de fregadero/fuente en muchos pines de E/S)
- Manipulación directa de bits de un solo ciclo
- Diodos de protección ESD de 4kV

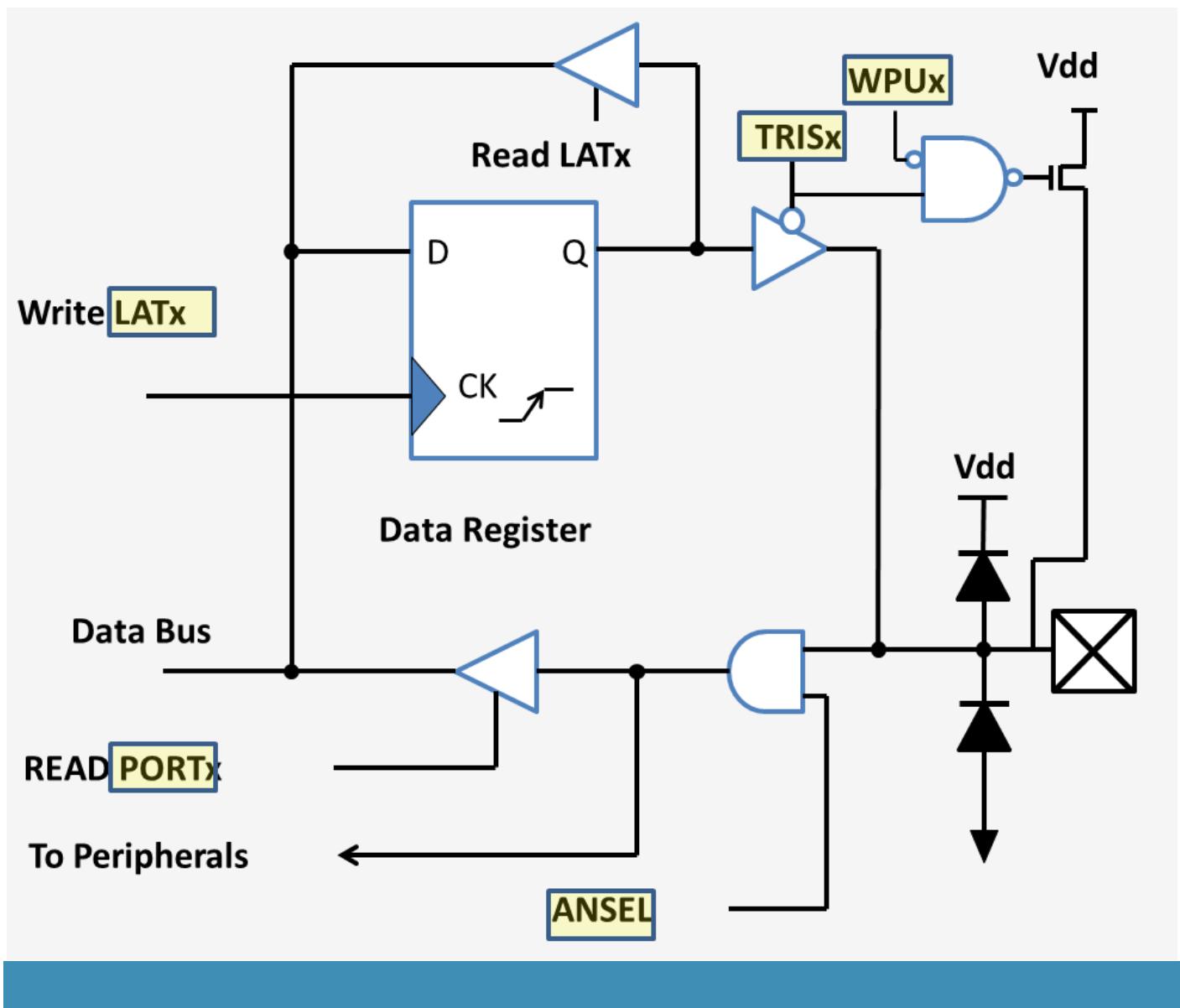
En Reset:

- Los Pines digitales vuelven a la entrada (Hi-Z)
- Los pines con capacidad analógica vuelven a ser analógicos

## Estructura típica de pines digitales

Cinco registros controlan el funcionamiento del pin digital. Estos registros de 8 bits controlan 8 pines de un PORT. Utilizando los registros **TRISx**, **PORTx**, **LATx**, **WPUx** y **ANSEL** el programa puede:

- Configurar el pin a como **trisx** de entrada o salida
- Leer un pin de entrada (o todos los pines de 8 PUERTOS) **PORTx**
- Salida de un 1 o 0 a un pin **LATx**
- Habilite o deshabilite la resistencia pull up interna **WPUx**
- Determinar si los pines con capacidad analógica funcionan en modo analógico o digital **ANSEL**



Para una discusión completa de la E/S digital PIC de rango medio mejorada, incluidos los detalles sobre la programación de operaciones digitales de entrada y salida, consulte la sección [E/S digital](#) de este tutorial de rango medio mejorado

## Pines multiplexados

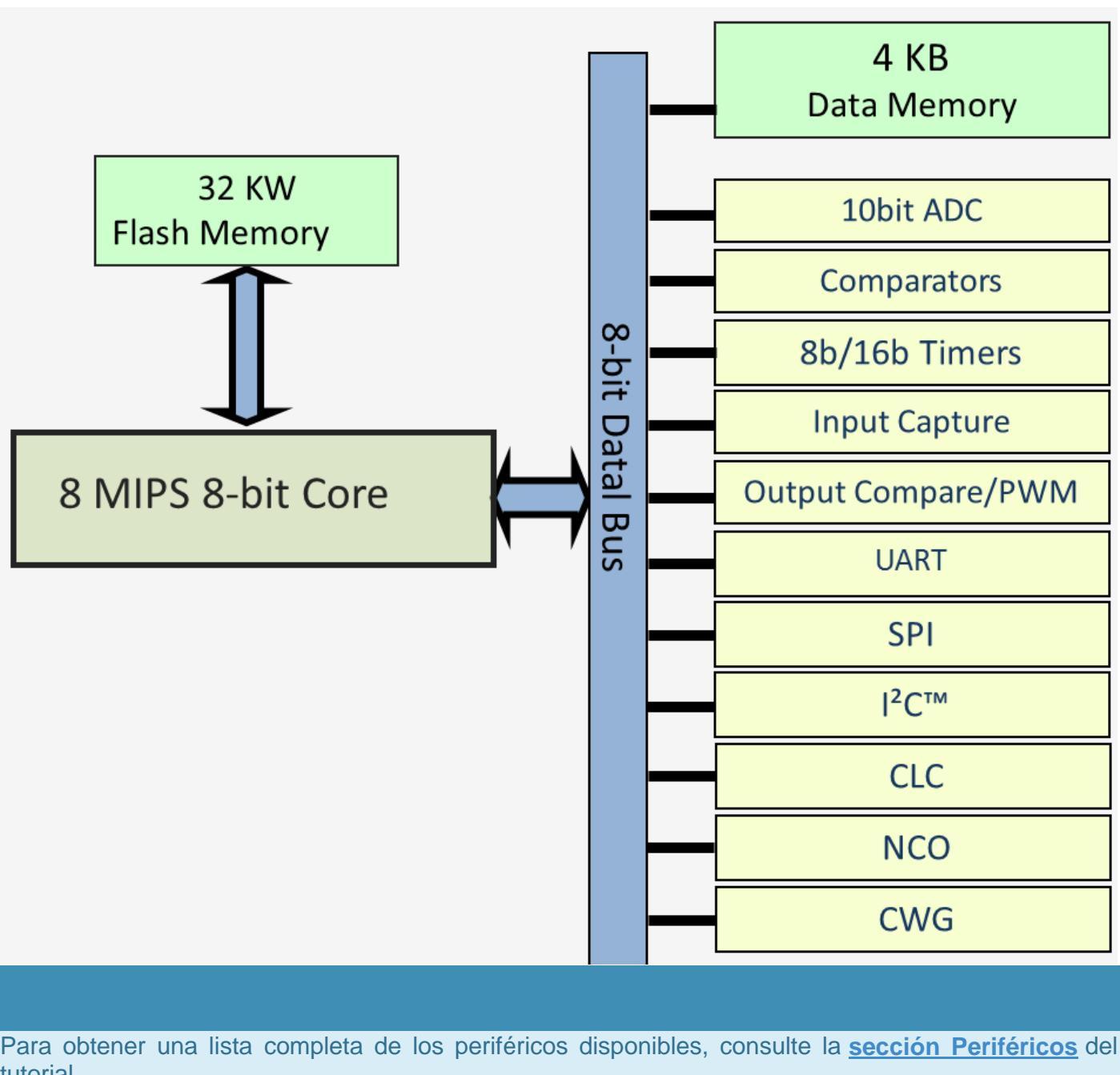
Además de configurarse como E/S digitales, los pines de los MCU PIC de rango medio mejorados pueden tener varias funciones posibles. El diagrama de pines de la hoja de datos muestra las opciones para cada pin. En el arranque el programa tiene la opción de configurar los pines.

40	<input type="checkbox"/>	↔	RB7/ICSPDAT/ICDDAT/SEG13
39	<input type="checkbox"/>	↔	RB6/ICSPCLK/ICDCLK/SEG14
38	<input type="checkbox"/>	↔	RB5/AN13/CPS5/CCP3/P#A/T1G/COM1
37	<input type="checkbox"/>	↔	RB4/AN11/CPS4/COM0
36	<input type="checkbox"/>	↔	RB3/AN9/C12IN2-/CPS3/CCP2/P2A/VLCD3
35	<input type="checkbox"/>	↔	RB2/AN8/CPS2/VLCD2
34	<input type="checkbox"/>	↔	RB1/AN10/C12IN3/CPS1/VLCD1
33	<input type="checkbox"/>	↔	RB0/AN12/CPS0/SRI/INT/SEG0

Para obtener una descripción completa de la E/S digital PIC de rango medio mejorada, incluidos los detalles sobre la configuración de los pines, consulte la [sección Periférico](#) del tutorial de rango medio mejorado

## Periféricos avanzados

Además de la E/S digital mejorada de gama media, los miembros de la familia PIC MCU tienen una variedad de periféricos avanzados. Estos periféricos incluyen periféricos para la conversión de datos, la comunicación y el acondicionamiento de señales.



Para obtener una lista completa de los periféricos disponibles, consulte la [sección Periféricos](#) del tutorial

## Interrumpe

Los MCU PIC de rango medio mejorados utilizan una estructura de interrupción preventiva de un solo vector.

Cada periférico del PIC es capaz de generar una solicitud de interrupción. Cuando se produce una solicitud de interrupción Y las interrupciones para el dispositivo solicitante están habilitadas, se producirá una interrupción.

El PIC de rabia media mejorado utiliza una pila de hardware de 16 niveles para almacenar el contenido actual del PC cuando se produce una interrupción. El contexto del programa se guarda en registros de sombra y el control se pasa a la dirección de memoria del programa 0x04.

## Rutina de servicio de interrupción (ISR)

El usuario es responsable de escribir el código para dar servicio a la interrupción y colocar el código en la dirección 0x04. Esta rutina de servicio de interrupción (ISR) determina el origen de la

interrupción y, a continuación, realiza la tarea necesaria para dar servicio al periférico de interrupción. La instrucción final de un ISR es la instrucción Return From Interrupt (RETFIE).

## Guardado automático de contexto

Los siguientes registros se guardan en un conjunto de registros de sombra de un solo nivel en caso de interrupción

- Registro W
- CS3
- ESTADO
- Fsr
- PCLATH

When the ISR executes the RETFIE instruction, these registers are restored to the pre-interrupt value

## Single Level Interrupt Pre-emption

Cuando se produce una interrupción, el bit de habilitación de interrupción global (GIE) en el registro de estado se deshabilita. Esto evitara que la interrupción sea adelantada por otra interrupción.

Al ejecutar un RETFIE, el estado del bit de control GIE se restaura a su valor previo a la interrupción.

Para obtener una descripción completa del proceso de interrupción y ejemplos de programación, consulte la [sección Interrupción](#) del tutorial de rango medio mejorado.

## Qué sucede en el arranque del sistema (RESET)

Hay varias fuentes de un RESET en el MCU PIC de rango medio mejorado. Las fuentes RESET comunes a casi todas las aplicaciones son el Power On Reset (POR) y el Brown Out Reset (BOR) debido a una caída del voltaje de la fuente de alimentación (es decir, brown-out). Hay varios otros métodos para restablecer el MCU, incluido el tiempo de espera de Watchdog y el acceso directo al pin MCLR.

### El contador de programas está establecido en 0x00.

Después de un RESET, la instrucción ubicada en la dirección 0 es la primera instrucción ejecutada. El desarrollador de la aplicación es responsable de colocar el código en esta dirección para 'arrancar' el PIC. El compilador MPLAB XC8 de Microchip insertará las instrucciones adecuadas para iniciar el PIC y transferir el control a [la red principal](#). Los programadores de nivel de ensamblaje tendrán que escribir el código para inicializar el PIC y saltar más allá del vector de interrupción ubicado en la dirección 0x04.<sup>®</sup>

### Todos los registros de funciones especiales se establecen en un valor predeterminado

La hoja de datos de cada MCU PIC de rango medio mejorado muestra los valores que contendrán los registros en RESET.

## Sample Register

R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W/HC-0/u	R-x/x	R/W-0/u	R/W-0/u
TMR1GE	T1GPOL	T1GTM	T1GSPM	T1GGO/ DONE	T1GVAL	T1GSS<1:0>	
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

HC = Bit is cleared by hardware

La sección [Ejemplos de programación](#) del tutorial de rango medio mejorado proporciona ejemplos de programación de cómo "arrancar" el MCU.

# Conjunto de instrucciones de rango medio enhanced

Esto se aplica a las familias PIC16F1xxx y PIC16LF1xxx de MCU PIC.®

## Operaciones orientadas a bytes

Mnemotécnico, Operandos	Descripción	Ciclos	Opcode MSb de 14 bits..... Osa	Estado afectado	Notas
ADDWF f,d	Agregar W y f	1	00 0111 dfff ffff	C, DC, Z	2
ADDWFC f,d	Añadir con Carry W y f	1	11 1101 dfff ffff	C, DC, Z	2
ANDWF f,d	Y W con f	1	00 0101 dfff ffff	Z	2
ASRF f,d	Desplazamiento aritmético a la derecha	1	11 0111 dfff ffff	C, Z	2
LSLF f,d	Desplazamiento lógico a la izquierda	1	11 0101 dfff ffff	C, Z	2
LSRF f,d	Desplazamiento lógico a la derecha	1	11 0110 dfff ffff	C, Z	2
CLRF f	Borrar f	1	00 0001 1fff ffff	Z	2
CLRW	Borrar W	1	00 0001 0000 00xx	Z	
COMF f,d	Complemento f	1	00 1001 dfff ffff	Z	2
DECF f,d	Decremento f	1	00 0011 dfff ffff	Z	2
INCF f,d	Incremento f	1	00 1010 dfff ffff	Z	2

IORWF	f,d	Inclusivo O W con f	1	00 0100 dfff ffff	Z	2
MOVF	f,d	Mover f	1	00 1000 dfff ffff	Z	2
MOVWF	f	Mover W a f	1	00 0000 1fff ffff	Ninguno	2
RLF	f,d	Gire a la izquierda f a través de Carry	1	00 1101 dfff ffff	C	2
FRR	f,d	Gire a la derecha f a través de Carry	1	00 1100 dfff ffff	C	2
SUBWF	f,d	Restar con Borrow W de f	1	11 1011 dfff ffff	C, DC, Z	2
SUBWFB	f,d	Restar W de f	1	00 0010 dfff ffff	C, DC, Z	2
SWAPF	f,d	Intercambiar mordiscos en f	1	00 1110 dfff ffff	Ninguno	
XORWF	f,d	Exclusivo OR W con f	1	00 0110 dfff ffff	Z	2

### Instrucciones de salto orientadas a bytes

Mnemotécnico, Operandos	Descripción	Ciclos	Opcode MSb de 14 bits..... Osa	Estado afectado	Notas
DECFSZ	f,d Decremento f, Omitir si 0	1(2)	00 1011 dfff ffff	Ninguno	1,2
INCFSZ	f,d Incremento f, Omitir si 0	1(2)	00 1111 dfff ffff	Ninguno	1,2

### Operaciones de registro de archivos orientadas a bits

Mnemotécnico, Operandos	Descripción	Ciclos	Opcode MSb de 14 bits..... Osa	Estado afectado	Notas
Bcf	f,b Bit Clear f	1	01 00bb bff ffff	Ninguno	2
Bsf	f,b Conjunto de bits f	1	01 01bb bfff ffff	Ninguno	2

## Operaciones de salto orientadas a bits

Mnemotécnico, Operandos	Descripción	Ciclos	Opcode MSb de 14 bits..... Osa	Estado afectado	Notas
BTFSC	f,b Prueba de bits f, omitir si está claro	1(2)	01 10bb bfff ffff	Ninguno	1,2
BTFSS	f,b Prueba de bits f, omitir si se establece	1(2)	01 11bb bfff ffff	Ninguno	1,2

## Operaciones literales

Mnemotécnico, Operandos	Descripción	Ciclos	Opcode MSb de 14 bits..... Osa	Estado afectado	Notas
ADDLW	k Agregar literal y W	1	11 1110 kkkk kkkk	C, DC, Z	
ANDLW	k Y literal con W	1	11 1001 kkkk kkkk	Z	
IORLW	k Inclusivo O literal con W	1	11 1000 kkkk kkkk	Z	
MOVLB	k Mover literal a BSR	1	00 0000 001k kkkk	Ninguno	
MOVLP	k Mover literal a PCLATH	1	11 0001 1kkk kkkk	Ninguno	

MOVLW	k	Mover literal a W	1	11 0000 kkkk kkkk	Ninguno
SUBLW	k	Restar W del literal	1	11 1100 kkkk kkkk	C, DC, Z
XORLW	k	Exclusivo O literal con W	1	11 1010 kkkk kkkk	Z

## Operaciones de control

Mnemotécnico, Operandos	Descripción		Ciclos	Opcode MSb de 14 bits..... Osa	Estado afectado	Notas
Sujetador	k	Rama relativa	2	11 001k kkkk kkkk	Ninguno	
BRW		Rama relativa con W	2	00 0000 0000 1011	Ninguno	
LLAMAR	k	Llamar subrutina	2	10 0kkk kkkk kkkk	Ninguno	
CALLW		Llamar a la subrutina con W	2	00 0000 0000 1010	Ninguno	
GOTO	k	Dirección de Goto	2	10 1kkk kkkk kkkk	Ninguno	
RETFIE	k	Retorno de la interrupción	2	00 0000 0000 1001	Ninguno	
RETLW	k	Volver, colocar literal en W	2	11 0100 kkkk kkkk	Ninguno	
DEVOLUCIÓN	k	Volver de la subrutina	2	00 0000 0000 1000	Ninguno	

## Operaciones inherentes

Mnemotécnico, Operandos	Descripción	Ciclos	Opcode MSb de 14 bits..... Osa	Estado afectado	Notas
CLRWDT	Borrar temporizador de vigilancia	1	00 0000 0110 0100	TO,PD	
Nop	Sin operación	1	00 0000 0000 0000	Ninguno	
OPCIÓN	Cargar OPTION registrarse con W	1	00 0000 0110 0010	Ninguno	
RESTABLECI MIENTO	Restablecimiento del dispositivo de software	1	00 0000 0000 0001	Ninguno	
DORMIR	Entrar en modo de espera	1	00 0000 0110 0011	PARA, PD	
TRIS	f Cargar registro TRIS	1	00 0000 0110 0fff	Ninguno	

## C-Compiler optimizado

Mnemotécnico, Operandos	Descripción	Ciclos	Opcode MSb de 14 bits..... Osa	Estado afectado	Notas
ADDFSR	Agregar literal a FSRn	1	11 0001 0nkk kkkk	Ninguno	
MOVIW	Mover FSRn indirecto a W	1	00 0000 0001 0nnn	Z	2
MOVWI	Mover W a FSRn indirecto	1	00 0000 0001 1nnnn	Z	2

## Notas

- Si se modifica el contador de programas (PC) o se realiza una prueba condicional, la instrucción requiere dos ciclos. El segundo ciclo se ejecuta como un NOP.
- Si esta instrucción se dirige a un registro INDF y se establece el MSb del FSR correspondiente, la instrucción requiere un ciclo de instrucción adicional.

# Bits de configuración PIC16F1xxx de rango medio mejorados de 8 bits

Los bits de configuración son una colección de datos binarios ubicados en la memoria flash de un microcontrolador PIC (MCU). Los bits de configuración se programan en el MCU PIC con el código de aplicación. No son código ejecutable ya que su dirección no es accesible por el contador de programas (PC). Cuando se programa en un MCU PIC, los bits de configuración completan los circuitos que habilitan o deshabilitan las características de hardware del MCU.<sup>®</sup>

Los bits de configuración se leen al salir de un restablecimiento y no se pueden modificar durante el tiempo de ejecución.

Las características especiales de la operación de MCU controlada por los bits de configuración incluyen:

1. Reloj del sistema
2. Gestión de energía
3. Seguridad del dispositivo
4. Características de funcionamiento

Los bits de configuración se generan a partir de directivas de compilador/ensamblador incluidas en los archivos de código fuente.

Esta página describe qué características están controladas por bits de configuración y cómo generarlas en el código fuente.



Los bits de configuración y los ajustes para dispositivos PIC16F1xxx individuales pueden variar. Compruebe en la hoja de datos los detalles de los bits de configuración de PIC MCU que está utilizando.

## Ubicación y formato

Los bits de configuración para la familia de MCU PIC16F1 se combinan en dos palabras de 14 bits denominadas CONFIG1 y CONFIG2. Las palabras de configuración se encuentran fuera del alcance de la PC en direcciones **0x8007** y **0x8008** en la memoria flash del MCU.

**REGISTER 10-1: CONFIGURATION WORD 1**

R/P-1/1	R/P-1/1	R/P-1/1	R/P-1/1	R/P-1/1	R/P-1/1	R/P-1/1
FCMEN	IESO	CLKOUTEN	BOREN1	BOREN0	CPD	CP
bit 13						bit 7

| R/P-1/1 |
|---------|---------|---------|---------|---------|---------|---------|
| MCLRE   | PWRTE   | WDTE1   | WDTE0   | FOSC2   | FOSC1   | FOSC0   |
| bit 6   |         |         |         |         |         | bit 0   |

## REGISTER 10-2: CONFIGURATION WORD 2

R/P-1/1	R/P-1/1	U-1	R/P-1/1	R/P-1/1	R/P-1/1	U-1
LVP	DEBUG	—	BORV	STVREN	PLLEN	—
bit 13						bit 7

U-1	R/P-1/1	R/P-1/1	U-1	U-1	R/P-1/1	R/P-1/1
—	VCAPE1	VCAPE0	—	—	WRT1	WRT0
bit 6						bit 0

Opciones de configuración de la hoja de datos de PIC16F1937

Los bits de configuración se insertan en el código fuente de la aplicación. Cuando se crea un proyecto PIC MCU, los valores de bits de configuración se cargan en el archivo de salida HEX. Los bits de configuración se programan en el PIC con el programa de aplicación.

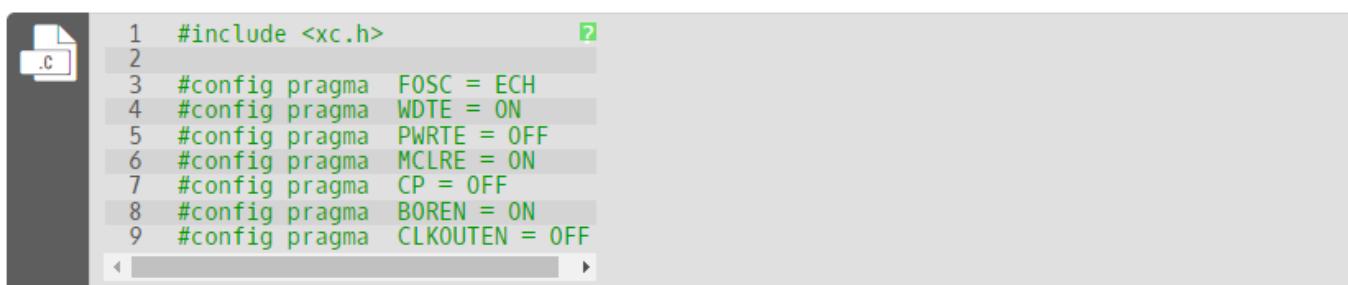
## Generación de bits de configuración en código C

El compilador MPLAB XC8 C de Microchip acepta directivas `#pragma` para establecer los bits de configuración.<sup>®</sup>



La sintaxis para generar bits de configuración:  
`#pragma config CONFIG_BIT_NAME = CONFIG_VALUE`

### Ejemplo de configuración de bits de configuración en C



```
1 #include <xc.h>
2
3 #config pragma FOSC = ECH
4 #config pragma WDTE = ON
5 #config pragma PWRTE = OFF
6 #config pragma MCLRE = ON
7 #config pragma CP = OFF
8 #config pragma BOREN = ON
9 #config pragma CLKOUTEN = OFF
```



Al establecer bits de configuración usando C, no es necesario conocer la palabra que contiene el bit que se está estableciendo. Todo lo que se necesita es el nombre del bit de configuración y el valor deseado.



Los archivos de encabezado de cada dispositivo PIC16F1xx contienen el `CONFIG_BIT_NAME` y `CONFIG_VALUE`. Puede encontrar una lista completa de los ajustes en el resumen de bits de configuración.

Una vez escritas, las directivas del compilador deben agregarse al proyecto PIC MCU de una de tres maneras:

1. En un archivo de origen C independiente que se agrega al proyecto
2. En un archivo de encabezado (`.h`) colocado en el proyecto con una instrucción `#include`
3. Colocado directamente dentro de uno de los archivos de código fuente que ya están en el proyecto



Los desarrolladores que utilizan el compilador XC8 pueden consultar la sección de bits de configuración del tutorial de MPLAB X para ver los accesos directos en la generación del código necesario para establecer los bits de configuración.

# Generación de bits de configuración en el ensamblado

Cuando se trabaja con lenguaje ensamblador, se requiere que el programador genere los valores específicos de 14 bits para cada una de las dos direcciones de configuración. Una vez que se generan los patrones, el programa realiza una llamada a la directiva `CONFIG` para establecer cada una de las palabras de configuración.

Un patrón de 14 bits para cada valor de bit de configuración individual se enumera en el archivo `.INC` para cada MCU PIC. El valor de 14 bits pasado a la directiva `CONFIG` se genera lógicamente AND-ing la configuración de bits en el archivo `.Archivo INC`.

El código para generar y cargar los bits de configuración se puede colocar en cualquier archivo de origen de ensamblado en el proyecto.

## Reloj del sistema

### Fuente de reloj (FOSC)

`FOSC` consta de tres bits de configuración individuales: `FOSC2`, `FOSC1` y `FOSC0`. El campo de bits `FOSC` se encuentra en `CONFIG1`.

`FOSC` selecciona la fuente de reloj para MCU.

Las opciones para `FOSC` son:

<b>FOSC&lt;2:0&gt;</b>	<b>Ajuste</b>	<b>Reloj Soure</b>
0 0 0	INTOSC	Oscilador interno
0 0 1	EXTRC	Oscilador RC externo
0 1 0	Hs	Oscilador de cristal externo de alta velocidad
0 1 1	Xt	Oscilador de cristal externo
1 0 0	Elepé	Oscilador de cristal externo de baja potencia
1 0 1	ECH	Reloj externo con rango de frecuencia 4 - 32 MHz
1 1 0	Ecm	Reloj externo con rango de frecuencia 0.5 - 4 MHz
1 1 1	Ecl	Reloj externo con rango de frecuencia 0 - 0.5 MHz

## Ejemplo: Selección del reloj del sistema



```
1 #include <xc.h>
2 #pragma config FOSC = HS
```

### Monitor de reloj a prueba de fallos (FCMEN)

El **FCMEN** es un único bit de configuración que reside en **CONFIG1**.

**FCMEN** controla el funcionamiento del monitor de reloj a prueba de fallos, lo que permite que el reloj cambie de externo a interno en caso de una falla de reloj externo.

Las opciones para **FCMEN** son:

<b>FCMEN</b>	<b>Ajuste</b>	<b>Función de monitor de reloj a prueba de fallos</b>
0	Apagado	Deshabilitado
1	En	Habilitado

### Ejemplo: Habilitación del monitor de reloj a prueba de fallos



```
1 #include <xc.h>
2 #pragma config FCMEN = ON // Enable Fail Safe CM
```

### Comutación intlernal/externa (IESO)

El **IESO** es un único bit de configuración que reside en **CONFIG1**.

**IESO** establece el modo para el cambio de reloj y el arranque de dos velocidades. Con **IESO** habilitado, la fuente del reloj puede ser controlada por el programa de aplicación.

Las opciones para **IESO** son:

<b>IESO</b>	<b>Ajuste</b>	<b>Función Star-up de dos velocidades</b>
0	Apagado	Deshabilitado
1	En	Habilitado



```
1 #include <xc.h>
2 #pragma config IESO = ON
```

## Habilitación de la salida de reloj (CLKOUTEN)

CLKOUTEN es un único bit de configuración que reside en CONFIG1.

CLKOUTEN permite que el pin OSCx/CLKOUT emita el reloj interno del sistema. Esto permite que el reloj del sistema PIC16F1xxx maneje otros componentes.

Las opciones para CLKOUTEN son:

CLKOUTEN	Ajuste	Función CLKOUT
0	En	Fosc se emitirá a OSCx/CLKOUT
1	Apagado	OSCx/CLKOUT será el oscilador o una función periférica

### Ejemplo: Salida del Clock del sistema interno



```
1 #include <xc.h>
2 #pragma config CLKOUT = ON
```

## Habilitación de bucle de bloqueo de fase (PLLEN)

PLLEN es un único bit de configuración que reside en CONFIG2.

El bucle de bloqueo de fase 4 X (PLL) del oscilador interno se controla mediante una combinación del bit de configuración PLLEN y el bit SPLLEN en el registro OSCON.

Las opciones para PLLEN son:

PLLEN	Ajuste	Función PLL interna
0	En	4 X PLL siempre está habilitado
1	Apagado	4 X PLL está controlado por el bit SPLLEN en OSCCON



No todos los MCU PIC16F1xxx tienen las mismas opciones para el PLL. Consulte la hoja de datos del MCU PIC que está utilizando para determinar los ajustes de configuración específicos de PLL.

### Ejemplo: Habilitación del bucle de bloqueo de fase interno



```
1 #include <xc.h>
2 #pragma config PLLEN = ON
```

# Gestión de energía

## Habilitación de restablecimiento de apagado (BOREN)

El ajuste de configuración de BOREN consta de dos bits individuales: BOREN1 y BOREN0. El campo de bits BOREN reside en CONFIG1.

BOREN permite que se produzca un MCU RESET si  $V_{DD}$  cae por debajo de un valor preestablecido. El nivel de voltaje que precipita el RESET está determinado por el bit de configuración BORV.

Hay cuatro opciones para los dos bits BOREN:

1. El restablecimiento de apagado siempre está habilitado
2. El restablecimiento de apagado siempre está deshabilitado
3. El restablecimiento de apagado está habilitado cuando se ejecuta, pero se deshabilita cuando MCU entra en modo SLEEP
4. Brown-out Reset se controla en tiempo de ejecución mediante el bit SB0REN del registro PCON.

BOREN	Ajuste	Función RESET de salida marrón
1 1	En	Siempre habilitado
0 0	Apagado	Siempre deshabilitado
1 0	NSLEEP	Habilitado mientras está ACTIVO, deshabilitado con en modo SLEEP
0 1	SBODEN	controlado por SB0REN bit de PCON Register

## Ejemplo: Desactivación de RESET de salida marrón

```
1 #include <xc.h>
2 #pragma config BOREN = OFF
```

## Nivel de voltaje marrón (BORV)

BORV es un único bit de configuración que reside en CONFIG2.

BORV solo es aplicable cuando el Brown-out Reset (controlado por el bit de configuración BOREN) está activo.

BORV selecciona uno de los dos niveles de voltaje preestablecidos como el voltaje de restablecimiento de salida marrón.

Los niveles de voltaje establecidos por BORV son:

BORV	Ajuste	Voltaje de apagado
0	L0	1,9 voltios
1	Hola	2,5 voltios



El nivel de voltaje brown-out establecido por BORV puede variar dependiendo del MCU PIC en particular. Consulte la hoja de datos para ver las opciones de voltaje de salida marrón.

## Ejemplo: habilite Brown-out RESET para que siempre esté activo y activado a 2,5 V.

```
1 #include <xc.h>
2 #pragma config BOREN = ON
3 #pragma config BORV = HI
```

Ejemplo: Habilite Brown-out RESET a 1.9 V mientras MCU está ACTIVO y deshabilite Brown-out en modo SLEEP.

```
1 #include <xc.h>
2 #pragma config BOREN = NSLEEP
3 #pragma config BORV = LO
```

## Regulador de voltaje (VCAPEN - LDO)

La configuración de VCAPEN consta de dos bits individuales: VCAPEN1 y VCAPEN0. El campo de bits VCAPEN reside en CONFIG2.

Para dispositivos con un regulador LDO interno, VCAPEN determina qué pin se asigna como  $V_{gorro}$  anclar.

La configuración de VCAPEN es:

VCAPEN	Ajuste	Función
0 0	Vió	RA6 se asigna como $V_{gorro}$
0 1	Sábado	RA5 se asigna como $V_{gorro}$
1 0	SAQ	RA0 se asigna como $V_{gorro}$
1 1	Apagado	$V_{gorro}$ está desconectado de todos los pines



No todos los MCU PIC16F1xxx tienen un regulador LDO interno. Consulte la hoja de datos del MCU PIC que está utilizando para determinar si hay un LDO y qué pines están disponibles como  $V_{gorro}$ .

## Ejemplo: Asignación de $V_{gorro}$ a RA6

```
1 #include <xc.h>
2 #pragma config VCAPEN = RA6
```

## Desactivación de $V_{gorro}$ funcionalidad

```
1 #include <xc.h>
2 #pragma config VCAPEN = OFF
```

## Programación de bajo voltaje (LVP)

LVP es un único bit de configuración que reside en CONFIG2.

El modo de entrada de programación de bajo voltaje permite programar MCU PIC16F1xxx con solo  $V_{Dd}$ . El uso de LVP elimina la necesidad de suministrar una  $V$  superior a  $V_{Dd}$  voltaje encendido MCLR/V<sub>P<sub>p</sub></sub>.

La configuración de LVP es:

LVP	Ajuste	Función de programación
1	En	Habilitado
0	Apagado	Deshabilitado

### Ejemplo: Deshabilitar las lecturas de EEPROM de fuentes externas



```
1 #include <xc.h>
2 #pragma config LVP = OFF
```

## Seguridad del dispositivo

### CPD - Protección de lectura de EEPROM de datos

CPD es un único bit de configuración que reside en CONFIG1.

La memoria EEPROM de datos internos se puede proteger de lecturas externas con CPD. Los programadores externos tienen prohibido leer EEPROM protegida. El contenido de la EEPROM todavía está disponible para lecturas de fuentes internas.

La configuración de CPD es:

Cpd	Ajuste	Función de protección de lectura EEPROM
0	En	Habilitado: EEPROM no es legible
1	Apagado	Deshabilitado - EEPROM es legible

### Ejemplo: Deshabilitar las lecturas de EEPROM de fuentes externas



```
1 #include <xc.h>
2 #pragma config CPD = ON
```

## Protección de lectura de memoria de programa (CP)

**CP** es un único bit de configuración que reside en **CONFIG1**.

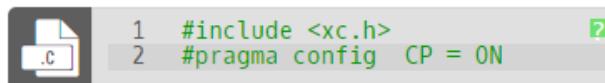
Todo el espacio de memoria del programa se puede proteger de lecturas externas con **CP**.

Verá todos los 0s cuando lea la memoria del programa protegido. El contenido de la memoria del programa todavía está disponible para lecturas de fuentes internas.

La configuración de **CP** es:

Cp	Ajuste	Función de protección de lectura
0	En	Habilitado: la memoria no es legible
1	Apagado	Deshabilitado: la memoria es legible

## Ejemplo: Deshabilitar lecturas de memoria de programa externo



```
1 #include <xc.h>
2 #pragma config CP = ON
```

## Habilitación de escritura automática de Flash (WRT)

**WRT** consta de dos bits de configuración individuales: **WRT1** y **WRT0**. El campo de bits **WRT** se encuentra en **CONFIG2**.

**WRT** establece el rango de direcciones en el que el programa de usuario puede escribir la memoria del programa.

La configuración de **WRT** es:

Wrt	Ajuste	Funcionalidad Flash Self Right
1 1	Todo	Ningún programa de usuario puede escribir en memoria
1 0	BOTA	Rango de direcciones 0 - 0x1FF protegido contra escritura 200 h - FFFn grabable
0 1	MITAD	Rango de direcciones 0 - 0x7FF protegido contra escritura 800 h - FFFn grabable
0 0	Apagado	Todas las direcciones de memoria pueden ser escritas por el programa de usuario



No todos los MCU PIC16F1xxx tienen las mismas opciones de WRT que se muestran. Consulte la hoja de datos del MCU PIC que está utilizando para determinar las opciones de WRT para el dispositivo.

## Características de funcionamiento

### Control de pines para MCLR (MCLRE)

**MCLRE** es un único bit de configuración que reside en **CONFIG1**.

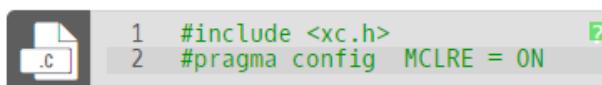
**MCLRE** controla la función de la **MCLR /V<sub>Pp</sub>** anclar.

**MCLRE** se ignora si la programación de bajo voltaje (establecida por el bit de configuración LVP) no está habilitada.

La configuración de **MCRLE** es:

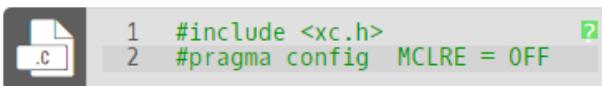
<b>MCLRE</b>	<b>Ajuste</b>	<b>Función</b>
1	En	La función pin es MCLR /V <sub>Pp</sub> con el pull-up débil interno habilitado
0	Apagado	La función Pin es una entrada digital con pull-up interno controlado por WPUx

**Ejemplo: Salir MCRL/V<sub>Pp</sub> como el pin RESET.**



```
1 #include <xc.h>
2 #pragma config MCLRE = ON
```

**Fabricación de MCRL/V<sub>Pp</sub> un pin de entrada digital.**



```
1 #include <xc.h>
2 #pragma config MCLRE = OFF
```

## Temporizador de encendido (PWRTE)

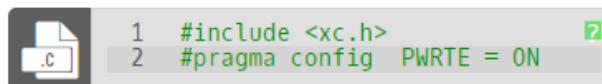
**PWRTE** es un único bit de configuración que reside en **CONFIG2**.

El tiempo de encendido proporciona un retraso nominal de 72 ms después de un restablecimiento de encendido o un restablecimiento de apagado para permitir V<sub>Dd</sub> para estabilizarse. PWRTE controla la activación o desactivación de este retraso.

Las opciones para **PWRTE** son:

<b>PWRTE</b>	<b>Ajuste</b>	<b>Función de temporizador de encendido</b>
0	En	Deshabilitado
1	Apagado	Habilitado

**Ejemplo de habilitación del temporizador de encendido**



```
1 #include <xc.h>
2 #pragma config PWRTE = ON
```

## Activación del temporizador de vigilancia (WDTE)

La configuración de WDTE consta de dos bits individuales: WDTE1 y WDTE0. El campo de bits WDTE reside en CONFIG1.

WDTE permite que se produzca un MCU RESET si el Watchdog Timer interno pasa de 0xFF a 0x00 antes de que el MCU pueda ejecutar una instrucción CLRWDT.

Hay cuatro opciones para los dos bits WDTE:

1. Watchdog Timer Reset siempre está habilitado
2. Watchdog Timer Reset siempre está deshabilitado
3. Watchdog Timer Reset está habilitado cuando se ejecuta, pero deshabilitado cuando MCU entra en modo SLEEP
4. Watchdog Timer Reset se controla en tiempo de ejecución mediante el bit SWDTEN del registro WDTCON

WDTE	Ajuste	Función de restablecimiento de Watch Dog
1 1	En	Siempre habilitado
0 0	Apagado	Siempre deshabilitado
1 0	NSLEEP	Habilitado mientras está ACTIVO, deshabilitado cuando está en modo SLEEP
0 1	SBODEN	Controlado por SBOREN bit de PCON Register

## Desbordamiento de pila de hardware (STVREN)

STVREN es un único bit de configuración que reside en CONFIG2.

El bit de configuración STVREN habilita o deshabilita un RESET en un desbordamiento o desbordamiento de pila. Un desbordamiento o desbordamiento de pila siempre establece el bit STKOVF o STKUNF en el registro PCON independientemente del valor de STVREN.

Las opciones para STVREN son:

STVREN	Ajuste	Función
1	En	Stack over/under flow RESET está HABILITADO
0	Apagado	Stack over/under flow RESET está DESHABILITADO



Los bits de configuración y los ajustes para dispositivos PIC16F1xxx individuales pueden ser diferentes. Compruebe en la hoja de datos los detalles de los bits de configuración de PIC MCU que está utilizando.



```
1 #include <xc.h>
2 #pragma config STVREN = ON
```

## DEBUG - Modo de depuración



El bit DEBUG en Configuration Word 2 es administrado automáticamente por el IDE MPLAB X. Para garantizar el correcto funcionamiento del dispositivo, este bit no debe ser alterado!

Más información:

- [Resumen de bits de configuración en lenguaje ensamblador](#)
- [Oscilador de 8 bits](#)
- [Conmutación de reloj a prueba de fallos de 8 bits](#)
- [Conmutación de reloj de 8 bits](#)
- [PLL de 8 bits](#)
- [Control de restablecimiento maestro de 8 bits](#)
- [REINICIO de salida marrón de 8 bits](#)
- [REINICIO de salida marrón de 8 bits](#)
- [Temporizador de encendido de 8 bits](#)
- [Temporizador de vigilancia MCU de 8 bits](#)
- [Desbordamiento/desbordamiento/desbordamiento de pila de 8 bits](#)

# Resumen de las opciones de bits de configuración para la MCU PIC® de rango medio mejorada mediante el compilador XC8

A continuación se muestra un resumen de las directivas de bits de configuración PIC16F1xxx aceptadas por el compilador MPLAB XC8 C de Microchip.®

Una descripción completa de estas opciones de bits de configuración se presenta en la página "[Bits de configuración](#)" del tutorial de rango medio mejorado.



Algunos MCU PIC de rango medio mejorados pueden tener un conjunto diferente de bits de configuración. Verifique la hoja de datos del MCU que está utilizando para obtener una lista completa.®



## Sintaxis

```
#pragma configuración CONFIG_BIT_NAME = CONFIG_VALUE
```

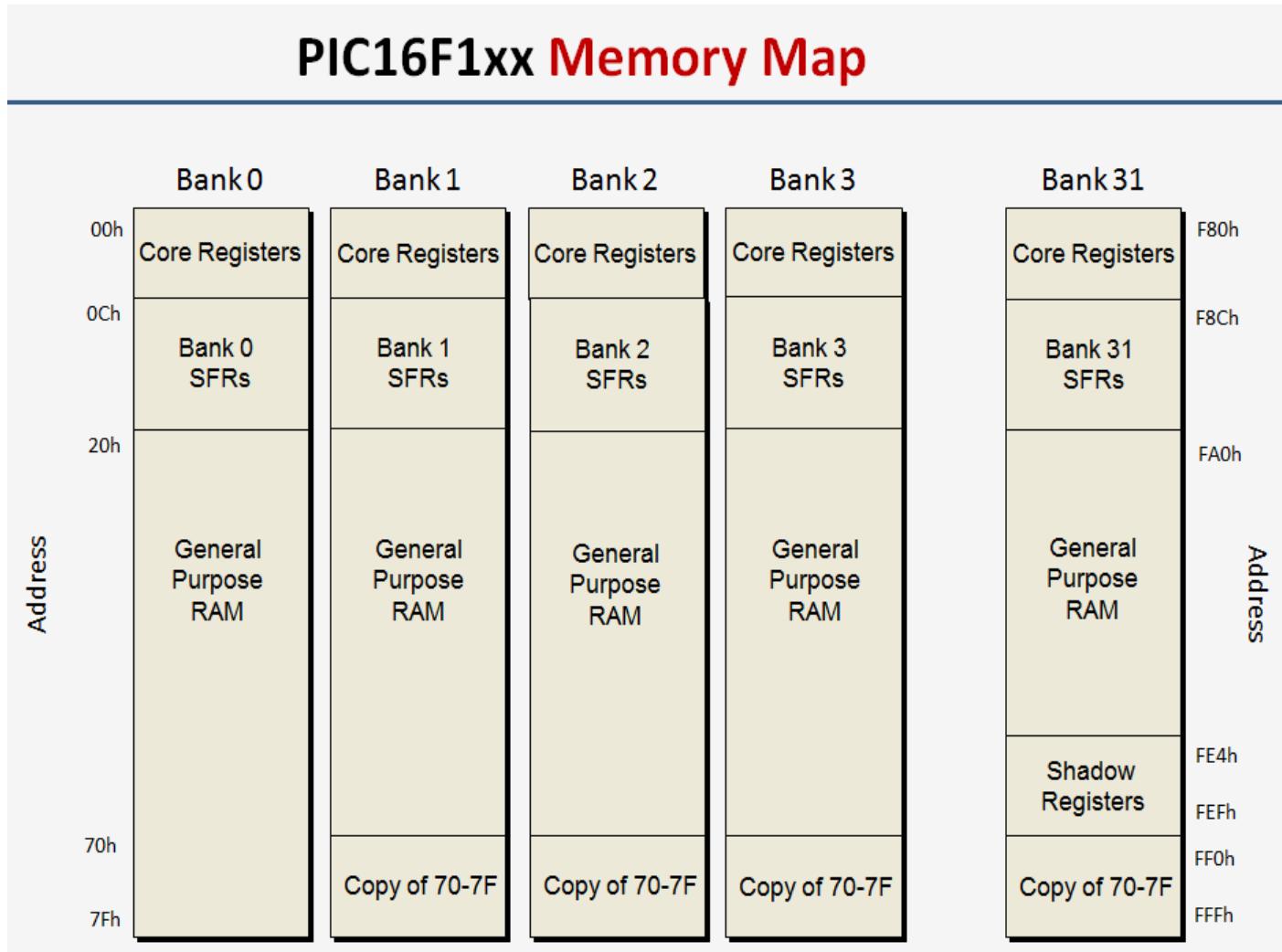
CONFIG_BIT_NAME	CONFIG_VALUE	Función de los bits de configuración
FOSC	<b>INTOSC</b> <b>EXTRC</b> <b>HS</b> <b>XT</b> <b>LP</b> <b>ECH</b> <b>ECM</b> <b>ECL</b>	Oscilador interno es la fuente de reloj Oscilador R-C externo es la fuente de reloj Cristal / Oscilador externo de alta semilla Cristal / Oscilador externo Oscilador externo de baja potencia Reloj externo con rango de frecuencia 4 - 32 MHz Reloj externo con rango de frecuencia 0.5 - 4 MHz Reloj externo con rango de frecuencia 0 - 0.5 MHz
WDTE	<b>ON</b> <b>OFF</b> <b>NSLEEP</b> <b>SWDTEN</b>	Watchdog Timer (WDT) está deshabilitado WDT está habilitado WDT está habilitado cuando se ejecuta y deshabilita cuando está en modo SLEEP WDT controlado por el bit SWDTEN en el registro WDTCON
PWRTE	<b>ENCENDIDO</b> <b>APAGADO</b>	Temporizador de encendido habilitado Temporizador de encendido desactivado
MCLRE	<b>ENCENDIDO</b> <b>APAGADO</b>	La función Pin es la función MCLR Pin es entrada digital

Cp	<b>ENCENDIDO APAGADO</b>	La protección de código está habilitada La protección de código está deshabilitada
Cpd	<b>ENCENDIDO APAGADO</b>	La protección de código de memoria de datos está habilitada La protección de código de memoria de datos está deshabilitada
BOREN	<b>ENCENDIDO APAGADO</b>	Brown-Out Reset está habilitado Brown-Out Reset está deshabilitado
CLKOUTEN	<b>ENCENDIDO APAGADO</b>	La función CLKOUT está habilitada en el pin CLKOUT La función CLKOUT está deshabilitada. Función de E/S u osc en el pin CLKOUT
IESO	<b>ENCENDIDO APAGADO</b>	El modo de conmutación de reloj interno /externo está habilitado El modo de conmutación de reloj interno /externo está deshabilitado
FCMEN	<b>ENCENDIDO APAGADO</b>	El monitor de reloj a prueba de fallos está habilitado El monitor de reloj a prueba de fallos está deshabilitado
Wrt	<b>ENCENDIDO APAGADO</b>	La protección contra escritura automática de memoria flash está habilitada La protección de autoescritura de memoria flash está deshabilitada
PLLEN	<b>ENCENDIDO APAGADO</b>	4X PLL interno está habilitado 4X PLL interno está deshabilitado
STVREN	<b>ENCENDIDO APAGADO</b>	El desbordamiento o desbordamiento de pila causará un desbordamiento de pila de restablecimiento o el desbordamiento insuficiente NO causará un restablecimiento
BORV	<b>LO HI</b>	Restablecimiento de voltaje brown-out - Punto de disparo bajo Restablecimiento de voltaje Brown-Out seleccionado - Punto de disparo alto seleccionado
LVP	<b>ENCENDIDO APAGADO</b>	Programación de bajo voltaje habilitada El alto voltaje en MCLR / VPP debe usarse para la programación

# Memoria de datos PIC16F1xxx de rango medio mejorada

## Organización de la memoria

Los MCU PIC de rango medio mejorados pueden contener hasta 4096 bytes de memoria de datos direccionable. La memoria de datos se divide en hasta 32 bancos de memoria con 128 bytes en cada banco.<sup>®</sup>



La memoria de datos PIC16F1xxx contiene cinco elementos de datos:

- Registros básicos
- Registros de funciones especiales (SFG)
- Memoria de propósito general
- Memoria común
- Registros de sombra

## Registros básicos

Las primeras 12 entradas de cada banco de memoria de datos PIC16F1xxx contienen registros denominados registros principales. Estos 12 registros se repiten en cada banco. Se puede acceder a los registros principales desde cualquier banco activo.

Los registros básicos incluyen información para:

- Tramitación general
- Direccionamiento directo de la memoria
- Direccionamiento indirecto de la memoria
- Control de interrupciones

## Registros Generales de Tramitación

Bank xx	
Address	
00h	INDF0
01h	INDF1
02h	PCL
03h	STATUS
04h	FSR0L
05h	FSR0H
06h	FSR1L
07h	FSR1H
08h	BSR
09h	WREG
0Ah	PCLATH
0Bh	INTCON

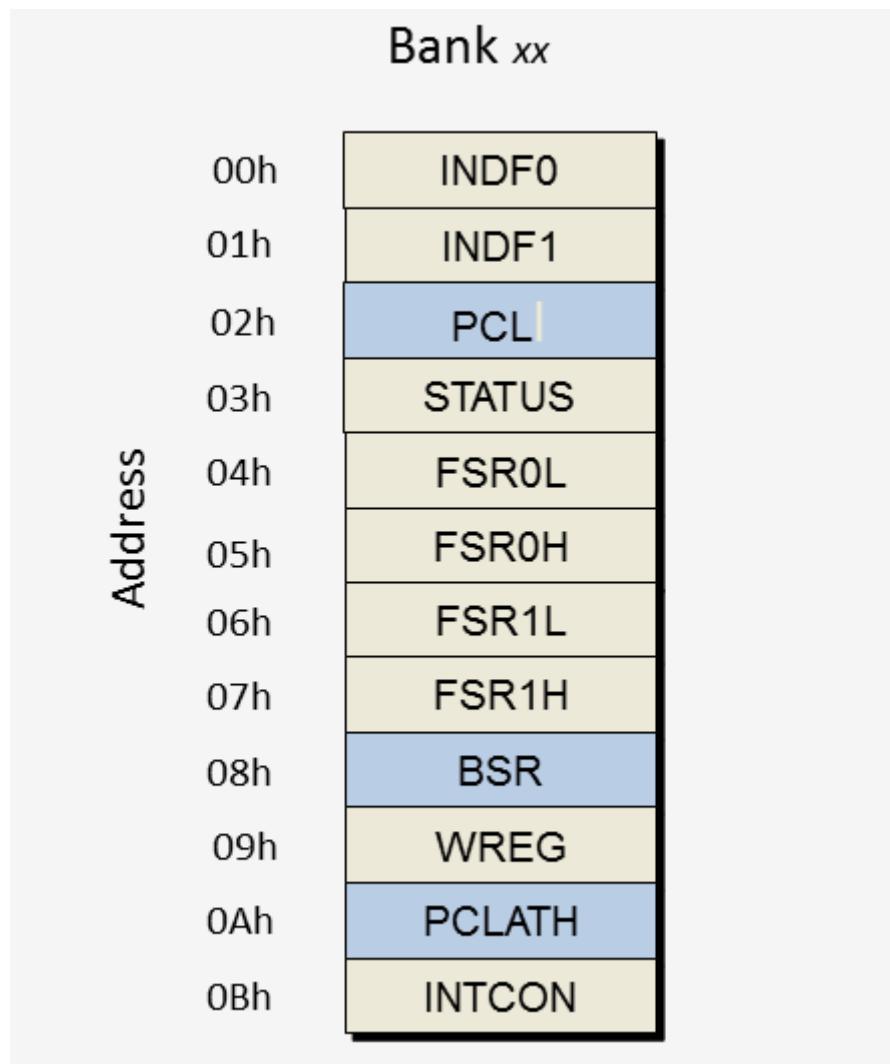
### Registro STATUS

---	---	---	TO	PD	Z	DC	C
bit 7							bit 0
<b>TO</b> -	Indica	que	Watchdog	Timer	ha		caducado
<b>PD</b> -	Estado	de	instrucción	de			suspensión
<b>Z</b> <input type="checkbox"/>	-	Indica si la última instrucción resultó en un 0					
<b>DC</b> -	la última instrucción resultó en una realización de los						4ésimo bit
C de orden bajo: <input type="checkbox"/>	la última instrucción resultó en una ejecución del bit más significativo						

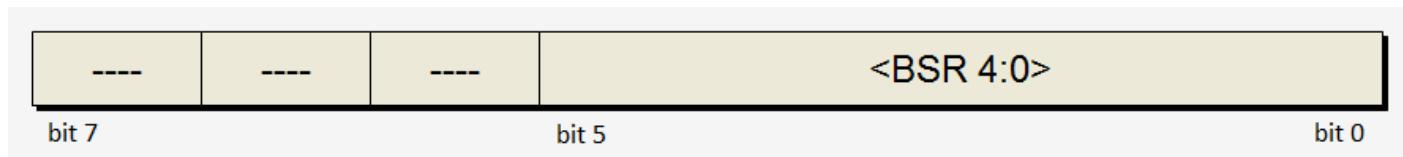
## Registro WREG

Conocido como el Registro de Trabajo o registro W, WREG sirve como acumulador.

## Registros de direccionamiento directo



## BSR (Bank Select Register)



Los cinco bits inferiores de BSR contienen el número de banco (0-31) del banco de datos activo. La información que explica cómo se utiliza el BSR se puede encontrar en la sección "[Direccionamiento directo](#)" del tutorial PIC16F1xxx.

## Registros PCL y PCLATH

Estos registros se utilizan al escribir o leer desde el contador de programas de 15 bits. La información sobre el uso de PCL y PCLATH se explica en la sección "[Memoria de programa](#)" del tutorial PIC16F1xxx.

## Registros de direcciones indirectas

Bank xx	
Address	
00h	INDF0
01h	INDF1
02h	PCL
03h	STATUS
04h	FSR0L
05h	FSR0H
06h	FSR1L
07h	FSR1H
08h	BSR
09h	WREG
0Ah	PCLATH
0Bh	INTCON

INDF0, FSR0L, FSR0H,  
INDF1, FSR1L, FSR1H

Estos seis registros controlan los dos canales de direccionamiento indirecto en el MCU. Los detalles del uso de estos registros se proporcionan en la sección "["Direccionamiento indirecto en el MCU PIC16F1xxx"](#)" del tutorial PIC16F1xxx

## Registro de control de interrupciones

Bank xx	
Address	
00h	INDF0
01h	INDF1
02h	PCL
03h	STATUS
04h	FSR0L
05h	FSR0H
06h	FSR1L
07h	FSR1H
08h	BSR
09h	WREG
0Ah	PCLATH
0Bh	INTCON

## INTCON

GIE	PEIE	TMROIE	INTE	IOCIE	TMR0IF	INTF	IOCIF
bit 7							bit 0

**GIE** - Interrupción global habilitar

**PEIE** - Interrupción periférica habilitar

**TMROIE** - Temporizador 0 interrupción habilitar

**INTE** - Interrupción externa habilitar

**IOCIE** - Interrupción en el cambio habilitar

**TMR0IF** - Temporizador 0 indicador de

interrupción **INTF** - Indicador de interrupción

externa **IOCIF** - Interrupción en el indicador de cambio

INTCON es el registro de control para las interrupciones PIC16F1xxx. Puede encontrar información sobre el uso de este registro de control en la sección "["Interrupciones"](#)" del tutorial PIC16F1xxx.

## Registros de Funciones Especiales (SFR)

En cada uno de los bancos de datos del PIC16F1xxx hay hasta 20 Registros de Funciones Especiales (SFR). Los SFR se encuentran justo debajo de los registros centrales que comienzan

en la dirección xxCh. Los SFR controlan los **periféricos** PIC16F1xxx, los **puertos de E/S digitales** y **la configuración del oscilador**.

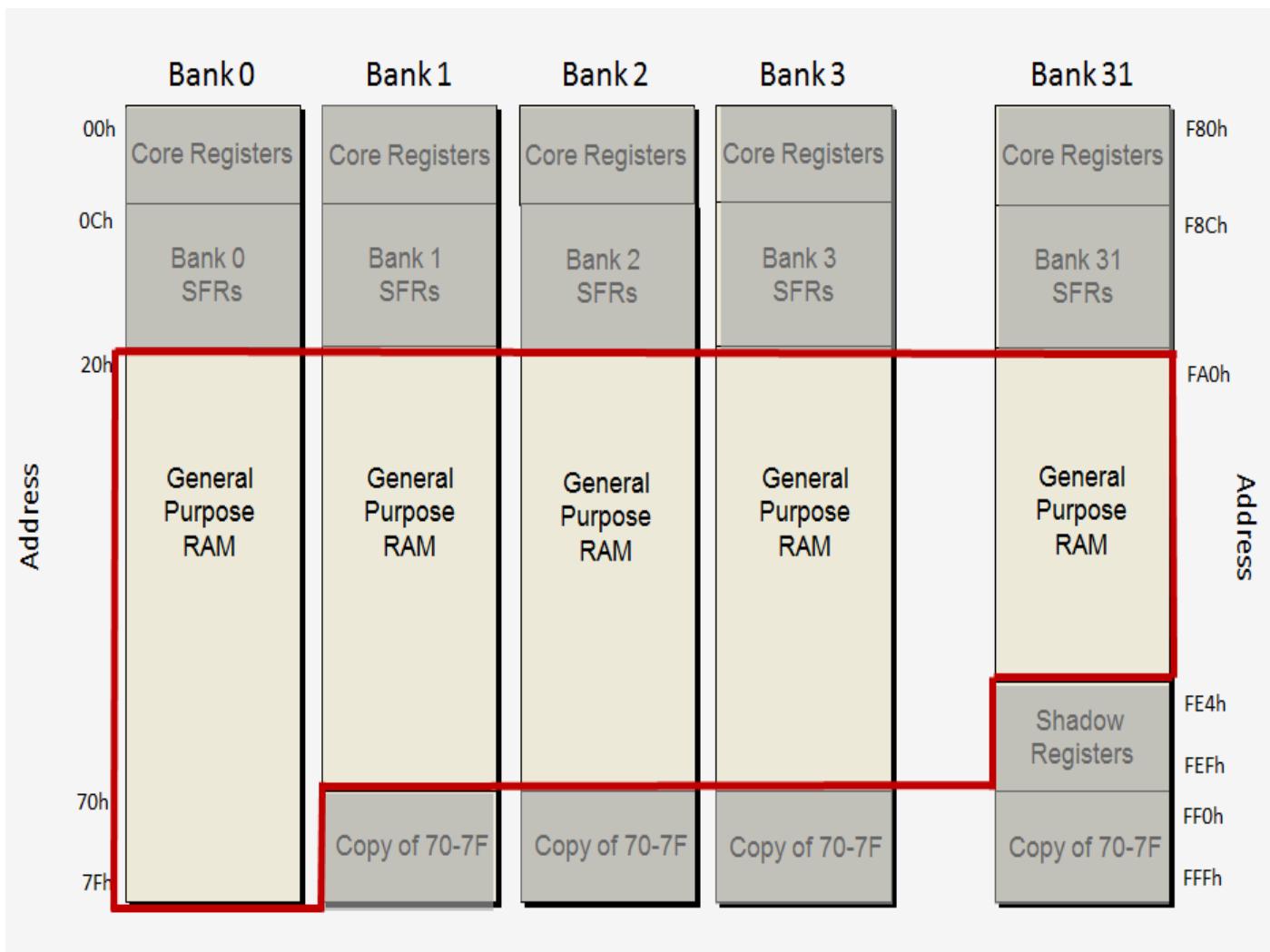
A diferencia de los registros principales, los SFR NO se duplican en cada banco. Los programas de solicitud deben asegurarse de que se haya seleccionado el banco apropiado antes de acceder a un SFR.

	Bank 0	Bank 1	Bank 31
00Ch	PORTA	TRISA	----
00Dh	PORTB	TRISB	----
	PORTC	TRISC	----
	PORTD	TRISD	----
	PORTE	TRISE	----
	PIR1	PIE1	----
	PIR2	PIE2	----
	PIR3	PIE3	----
---	---	---	----
TMR0	OPTION_REG	PCON	----
TMR1L		WDTCON	----
TMR1H		OSCTUNE	----
T1CON		OSCCON	----
T1GCON		OSCSTAT	----
TMR2		ADRESL	----
PR2		ADRESH	----
T2CON		ADCON0	----
---		ADCON1	----
CPSCON0		----	----
CPSCON1		----	----
01Fh		09Fh	F9Fh

Los SFR para cada uno **PIC MCU®** variará. Consulte la hoja de datos para conocer el nombre y la ubicación de los SFR para el MCU que está utilizando.

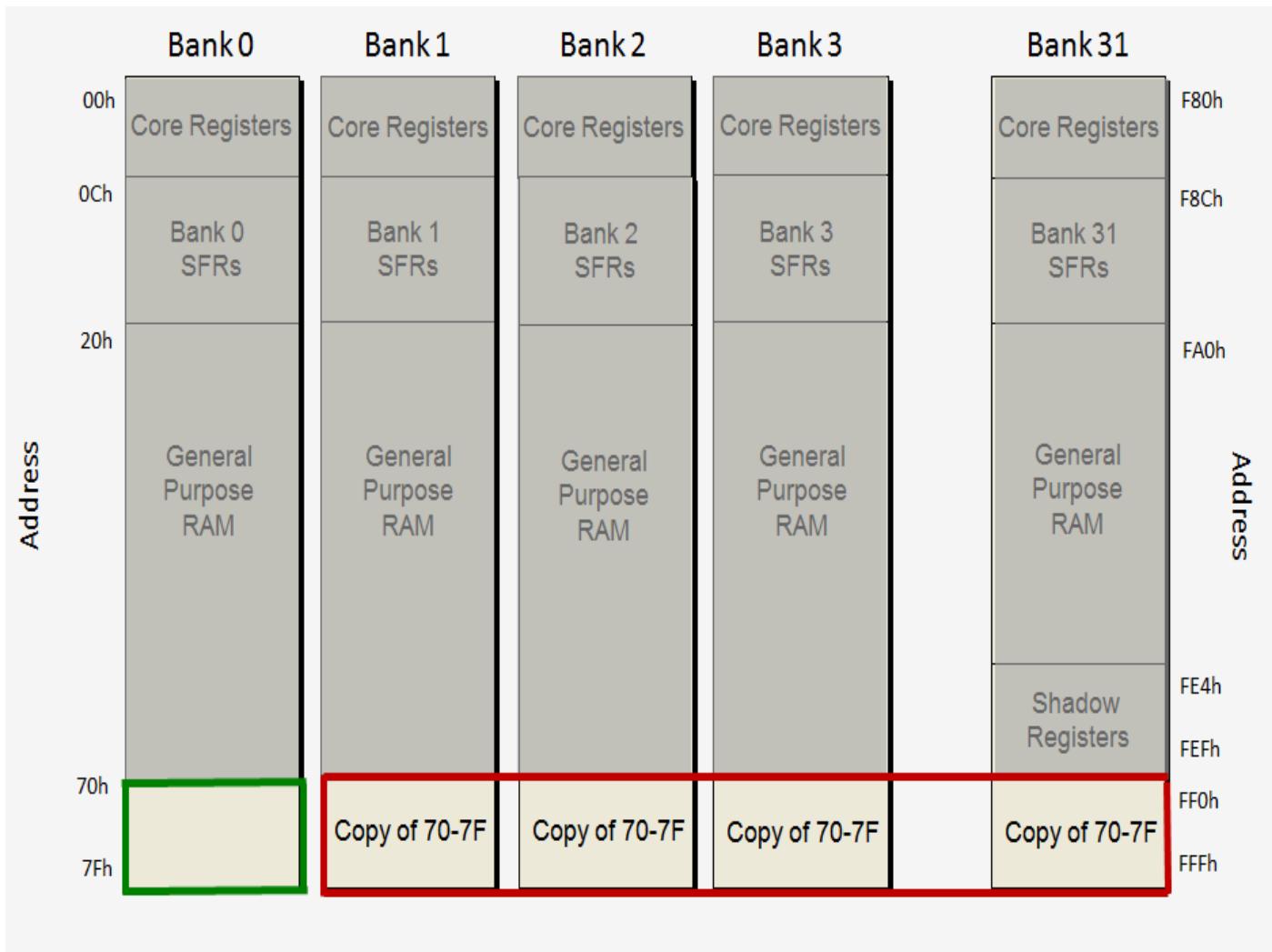
## Memoria de propósito general

La memoria de uso general (RAM) se encuentra en cada banco de memoria justo debajo de los SFR. Esta memoria está disponible para los datos de la aplicación.



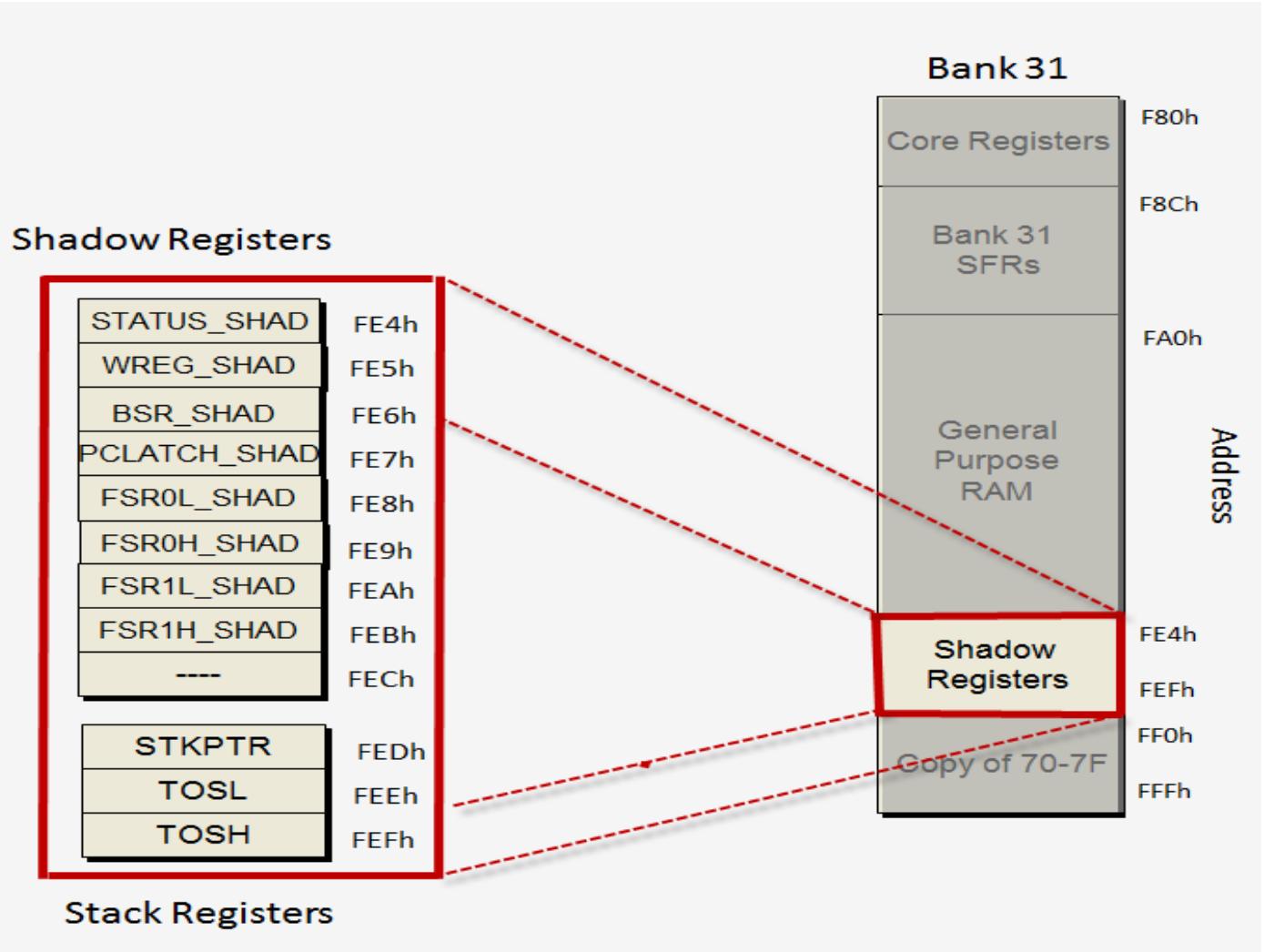
## Memoria común

Los últimos 16 bytes del banco 0 (direcciones 70h - 7Fh) se repiten en cada banco de datos. Esto permite a los programas de aplicación acceder a las variables ubicadas en estas direcciones sin tener que establecer BSR.



## Registros de sombras

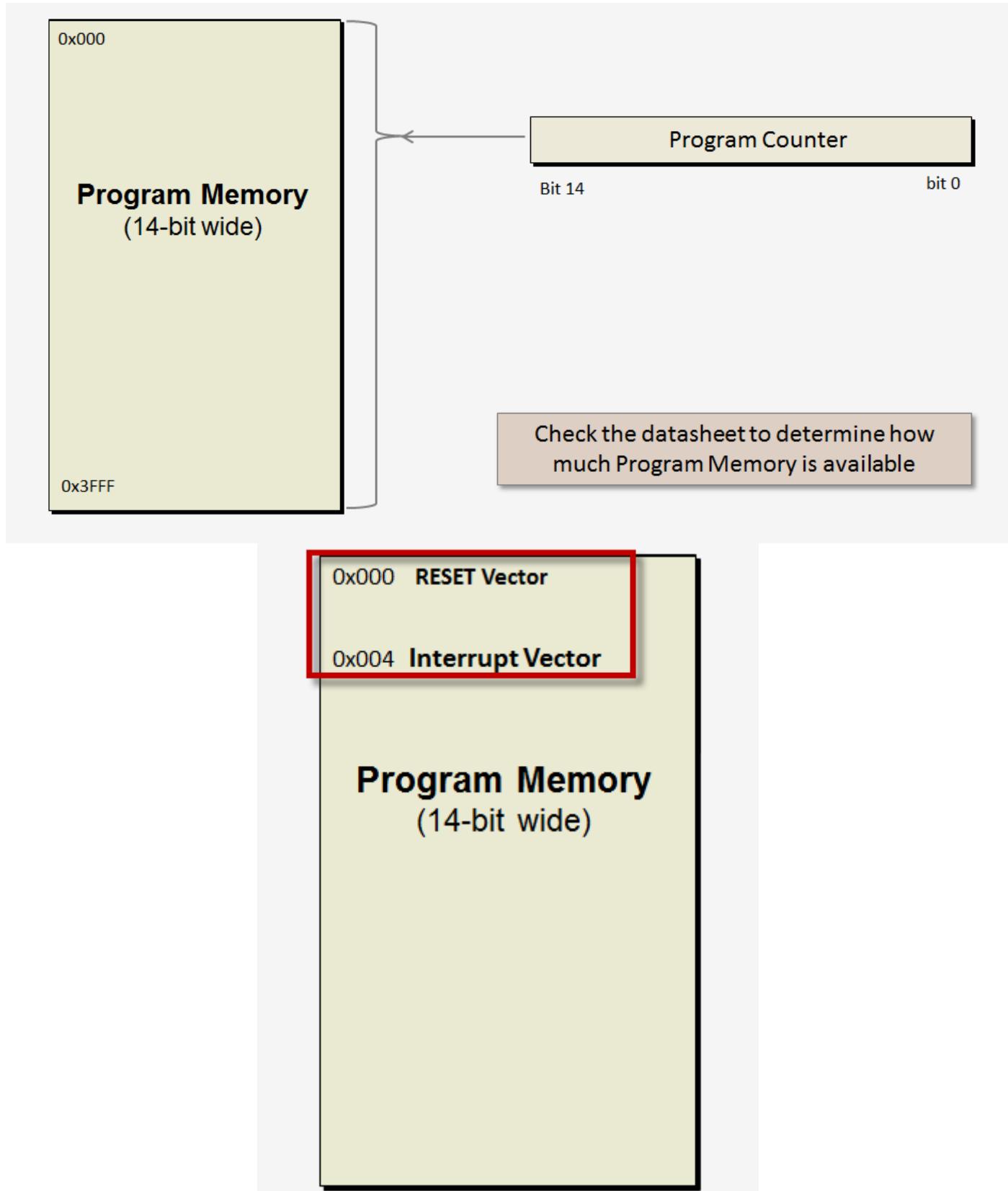
Ubicados en la parte inferior del banco 31 se encuentran los registros de sombra y pila PIC16F1xxx. Los registros de sombras guardan el contexto del programa capturando varios registros centrales cuando se produce una interrupción. Los registros principales se restauran cuando se ejecuta una instrucción de retorno desde interrupción (RETFIE). Puede encontrar más información sobre los registros de sombras en la sección "["Interrupciones"](#)" del tutorial PIC16F1xxx.



La pila se utiliza para almacenar el contador de programas (PC) en caso de una interrupción o llamada de subrutina. La información sobre la pila y los registros asociados se puede encontrar en la sección "["Modelo del programador"](#)" del tutorial PIC16F1xxx

# Memoria de programa PIC® MCU de rango medio mejorada

La memoria de programa en el MCU PIC de rango medio mejorado consta de hasta 32 MB de memoria Flash de 14 bits de ancho. Después de programar el MCU, la memoria del programa contiene el código de aplicación del usuario. Se accede a la memoria del programa mediante un registro de contador de programas (PC) de 15 bits.<sup>®</sup>

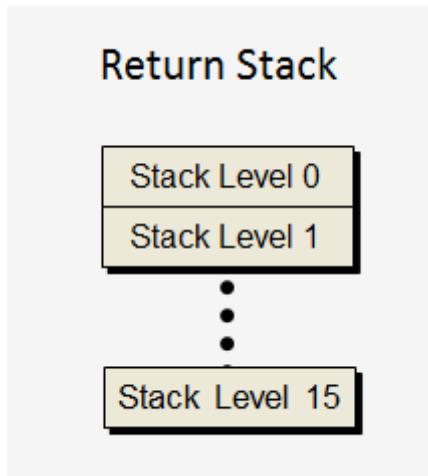


## RESET Vector

En RESET, el contador del programa se borra, lo que resulta en todos los ceros. Esto permite que la dirección de memoria del programa 0h sea la ubicación de la primera instrucción ejecutada después de una condición RESET.

## Vector de interrupción

Cuando se produce una interrupción, el control del programa se transfiere a la dirección 04h. La sección "*Interrupciones*" proporciona una descripción completa del proceso de interrupción



## Pila de retorno

Una pila de retorno de hardware de 16 entradas y 15 bits de ancho almacena el equipo en caso de interrupción o llamada a una subrutina. La pila de retorno funciona en una base de último en entrar, primero en salir.

Al ejecutar una instrucción RETURN (`RETFIE` o `RETURN`), el elemento superior de la pila se retira de la pila y se coloca en el contador del programa.



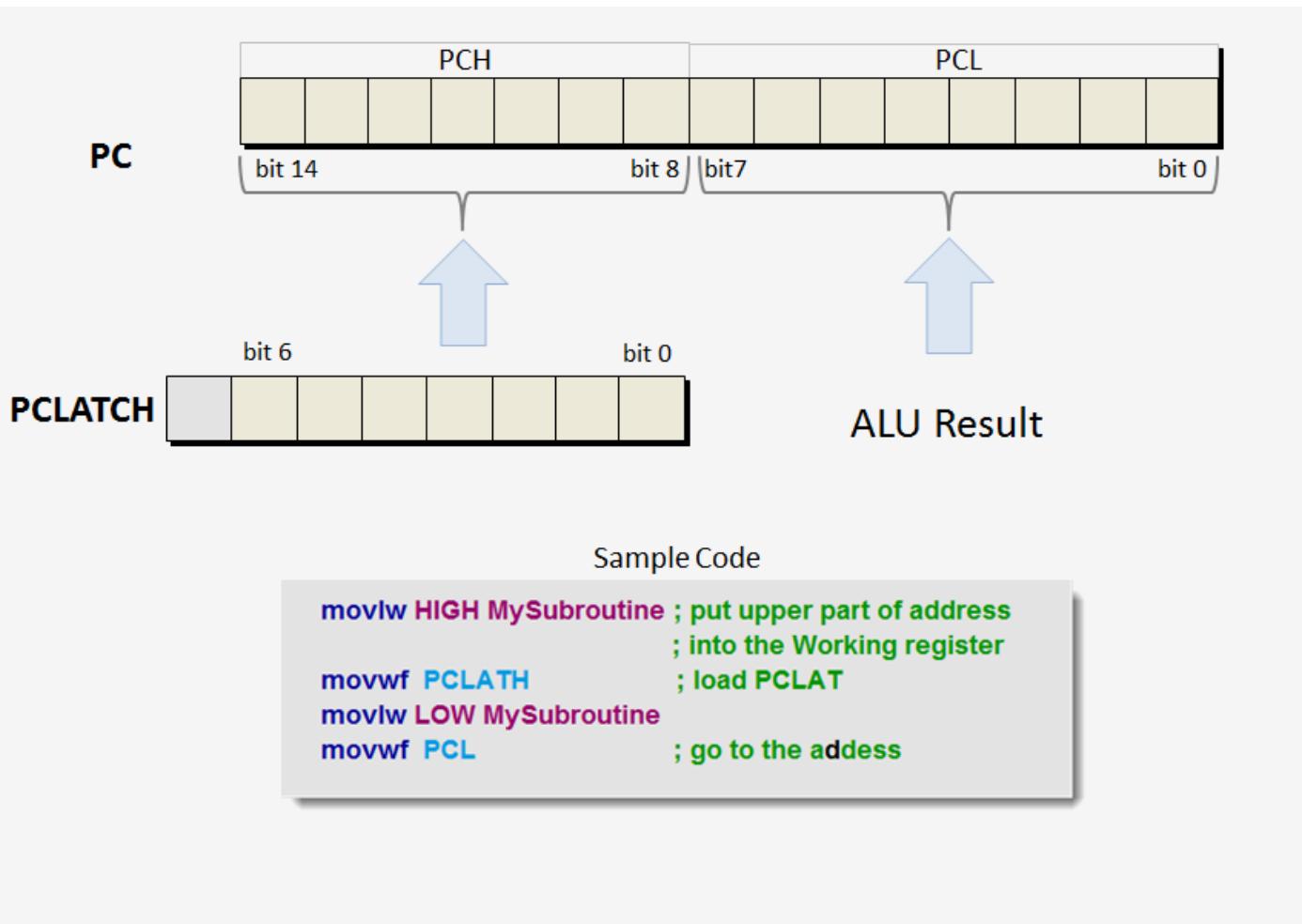
Para leer o modificar el PC de 14 bits con un MCU de 8 bits, se utilizan dos registros de funciones especiales (SFR):

- `PCL` - contiene los 8 bits inferiores del contador de programa <PC7:0>
- `PCLATH`: el contenido depende de la operación de MCU que se realiza

`PCL` y `PCLATH` se utilizan cuando el programa escribe en el PC, lee el contador del programa o ejecuta una instrucción `GOTO` o `CALL`.

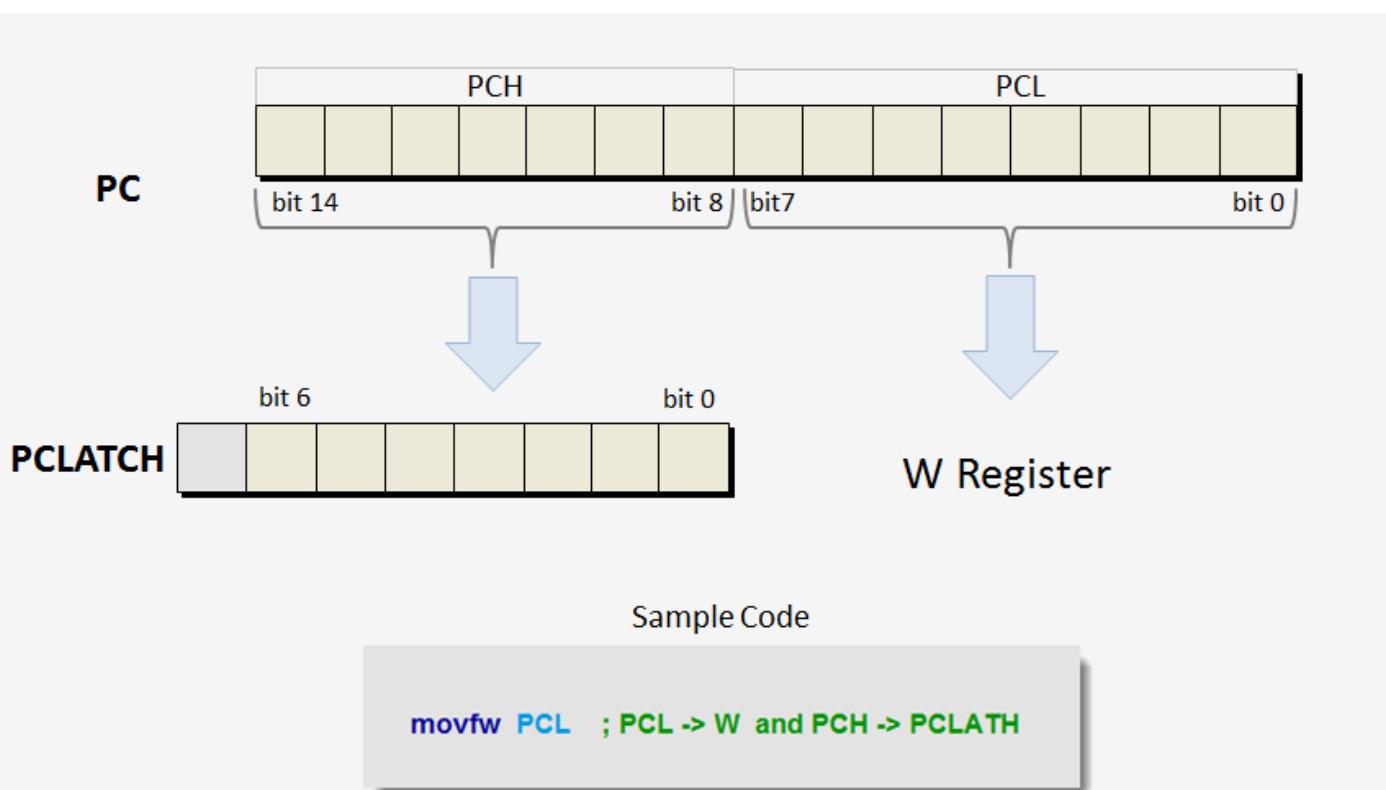
## Escribir en el PC

Cuando la aplicación escribe en `PCL`, el contenido actual de `PCLATH<5:0>` se escribirá en `PC<14:8>`. Debido a esto, el contenido de `PCLATH<5: 0>` SIEMPRE DEBE ser correcto ANTES de escribir en `PCL`.



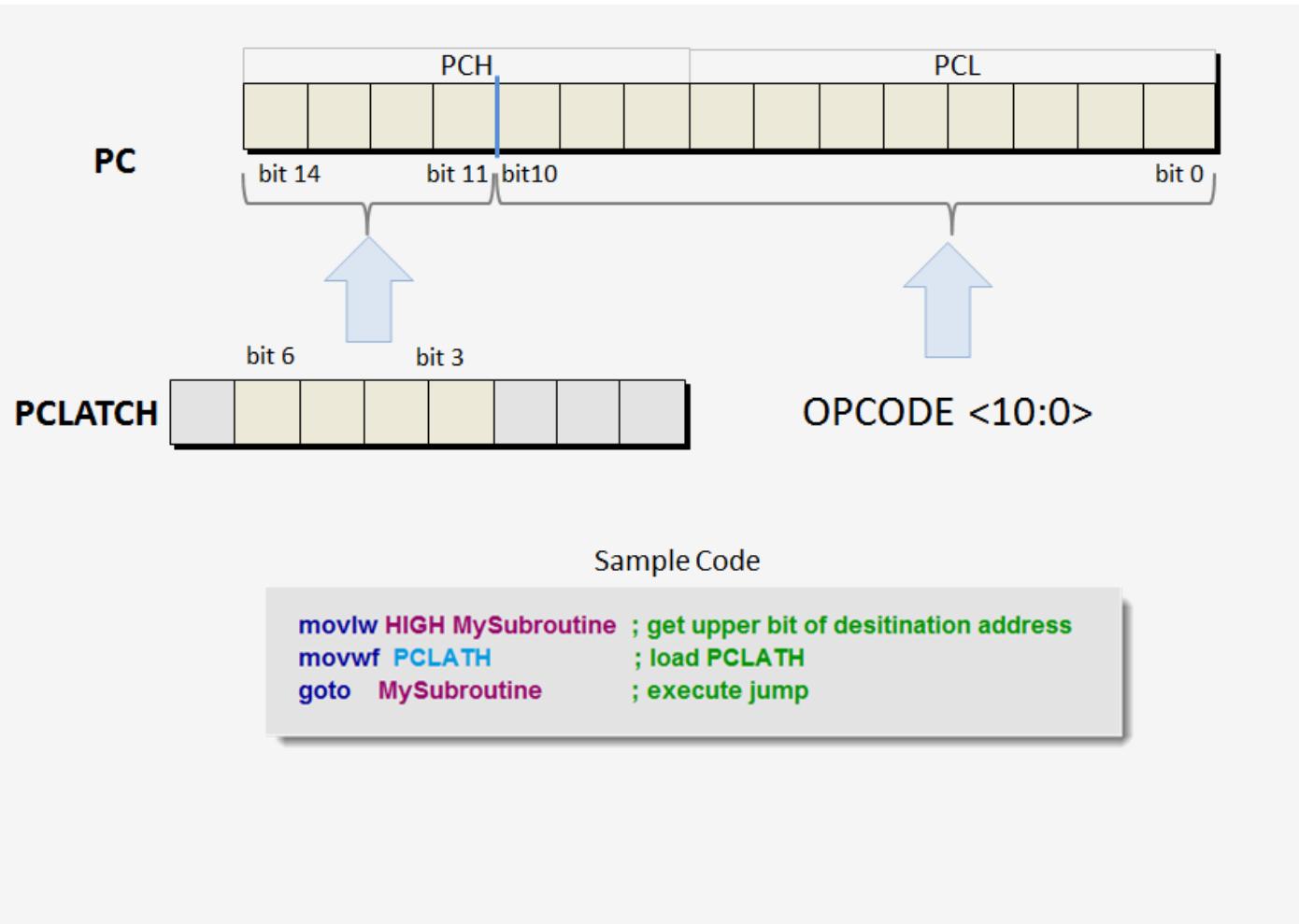
## Lectura del PC

Cuando una aplicación lee `PCL, PC<14:8>` se captura en `PCLATH`.



## Ejecución de una instrucción CALL o GOTO

Las instrucciones CALL y GOTO solo tienen 11 bits disponibles para especificar la dirección de destino. PCLATH se utiliza para extender el operando para acceder a todas las direcciones de memoria del programa. Cuando se ejecuta una LLAMADA o GOTO, la dirección de 11 bits del operando se carga en PC<10:0> y PCLATH<6:3> se carga en PC<14:11>.



# Periféricos

Periféricos de la familia de microcontroladores PIC de gama media mejorados de 8 bits®

Módulo	Nombre
<u>Adc</u>	Convertidor de analógico a digital de 10 bits
<u>CVX</u>	Celda lógica configurable
<u>CMP</u>	Comparadores analógicos
<u>Cps</u>	Módulo de detección capacitiva
<u>Dac</u>	Convertidor de digital a analógico de 5 bits
<u>Dsm</u>	Modulador de señal de datos
<u>ECCP</u>	Captura, comparación y PWM mejoradas
<u>EUSART</u>	Transmisor receptor asíncrono universal mejorado
<u>FVR</u>	Referencia de voltaje fijo
<u>MI2C</u>	Máster Circuito Interintegrado (I <sup>2</sup> C)
<u>MSSP</u>	Puerto serie síncrono maestro (SPI/I <sup>2</sup> C)
<u>Nco</u>	Oscilador controlado numéricamente
<u>Spi</u>	Interfaz periférica serie (SPI)
<u>TMR0</u>	Temporizador 0 (8 bits)
<u>TMR1</u>	Temporizador 1 (16 bits)
<u>TMR2</u>	Temporizador 2 (8 bits)
<u>UART</u>	Transmisor receptor asíncrono universal

# E/S digitales de rango medio mejoradas de 8 bits

## Estructura básica de E/S

Casi todos los pines del microcontrolador PIC de rango medio mejorado (MCU) se pueden utilizar como pin de entrada o salida digital. Los pines digitales comparten estos atributos:<sup>®</sup>

- Capacidad para monitorear entradas digitales
- Controlar las señales de salida digital
- Pull-ups internos débiles
- Multiplexado con periféricos
- Alta capacidad de accionamiento (hasta 25 mA de fregadero/fuente en muchos pines de E/S)
- Manipulación directa de bits de un solo ciclo
- Diodos de protección ESD de 4 kV

En RESET:

- Los pines digitales vuelven a la entrada (Hi-Z)
- Los pines con capacidad analógica vuelven a ser analógicos

Las E/S digitales son controladas por software en el MCU. El programa MCU configura, lee y envía los valores a los pines digitales.

## Puertos de E/S

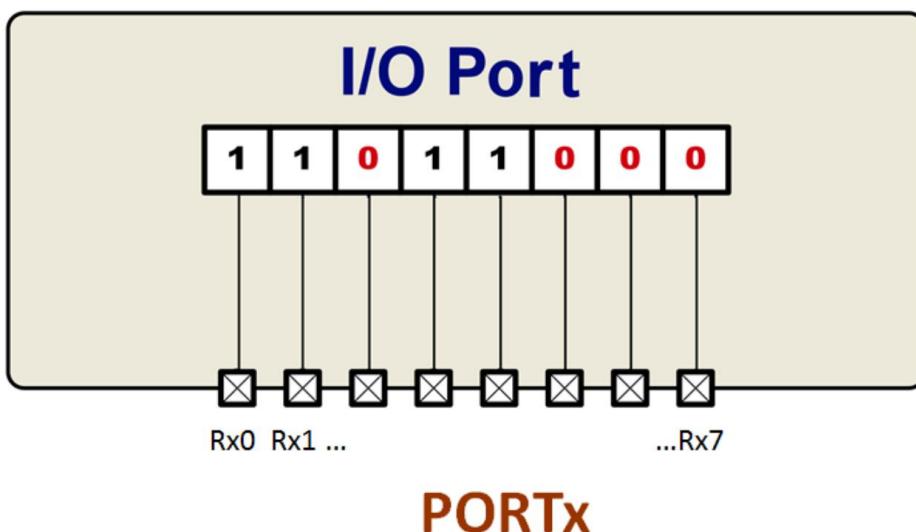


Figura 1. Puerto de E/S.

Los pines de E/S digitales individuales se combinan en grupos llamados puertos. Los puertos de E/S contienen hasta ocho pines digitales. Se puede acceder a los pines de E/S digitales individualmente pin por pin. También se puede acceder a todos los miembros de un puerto de E/S en particular en un ciclo de instrucción mediante una de las instrucciones de acceso de bytes del MCU.

Los puertos de E/S se mencionan por letra (por ejemplo, PORTA, PORTB, PORTC) El número de puertos de E/S variará según el MCU PIC que se utilice. Consulte la hoja de datos individual para determinar las asignaciones de PORT para un MCU PIC.

## Pin de E/S digital típico

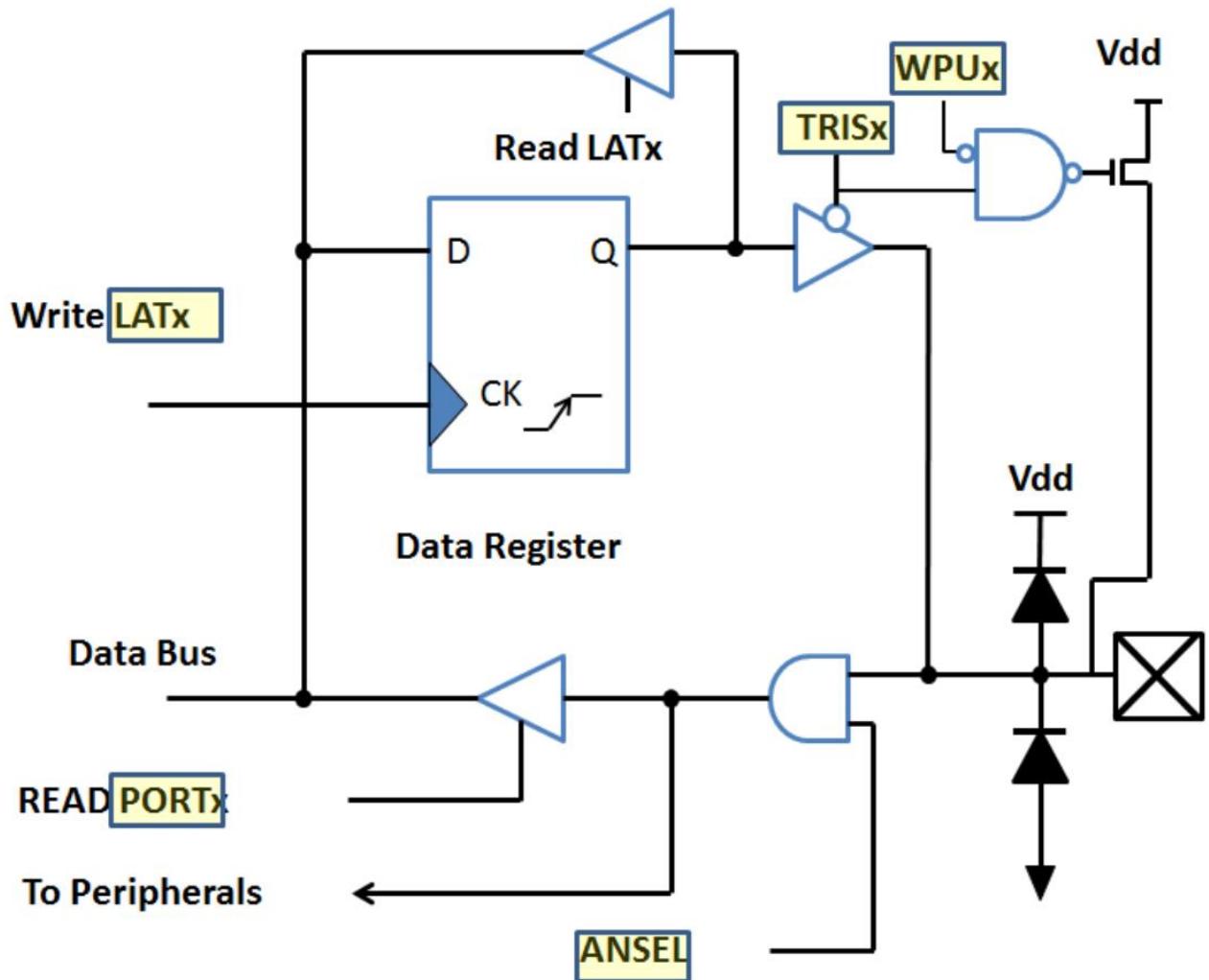


Figura 2. E/S de pin digital.

Hay cinco registros disponibles para configurar y controlar los pines de E/S digitales.

1. TRISx - Establece la dirección como entrada o salida.
2. ANSELx: determina si un pin con capacidad analógica funciona como entrada analógica o E/S digital.
3. LATx- Se utiliza para generar valores para un pin digital.
4. PORTx- Lee el valor de entrada de un pin digital.
5. WPUx- Habilita el pull up débil interno.



Hay cinco registros de control de E/S para cada puerto.

- Para el puerto A los registros de control son: TRISA, ANSELA, LATA, PORTA y WPUA.
- Para el puerto B los registros de control son: TRISB, ANSELB, LATB, PORTB y WPUB.

A lo largo de esta página se proporcionan ejemplos de programas que muestran cómo utilizar los registros de control de E/S.

## Identificación de los pines de E/S en la hoja de datos

Las E/S digitales pueden compartir pines con otros periféricos y líneas de control MCU. Algunos pines de E/S digitales son analógicos y se pueden configurar para funcionar como pines de entrada analógicos. Consulte el diagrama de pines de la hoja de datos del dispositivo para determinar qué pines están disponibles como E/S digitales.

Los pines digitales se identifican mediante tres identificadores secuenciales:

- El primer identificador para un pin digital es la letra R.
- El segundo identificador es una letra del PORT en la que está asociado el pin (como A para PORTA, B para PORTB, etc.).
- El identificador final es un número del 0 al 7 que indica la posición en el PORT que ocupa el Pin.

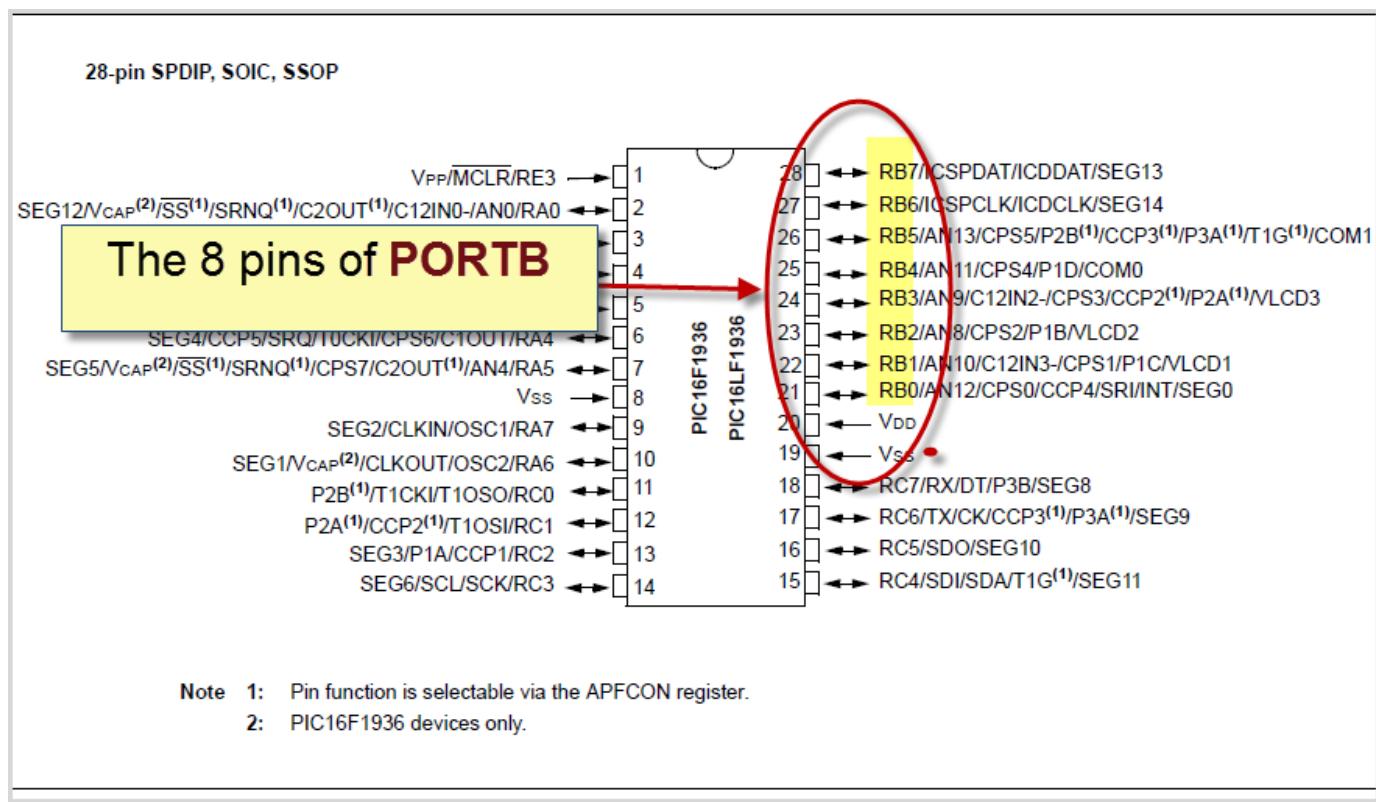


Figura 3. Todos los pines digitales asociados con PORTB están resaltados en amarillo.

Los miembros de PORTB van desde RB0 - RB7.

Para los otros pines en el MCU:

- Los pines con capacidad analógica se designan con las dos letras AN seguidas de un número.
- Los pines periféricos y de control de MCU se designan por su nombre en la hoja de datos.

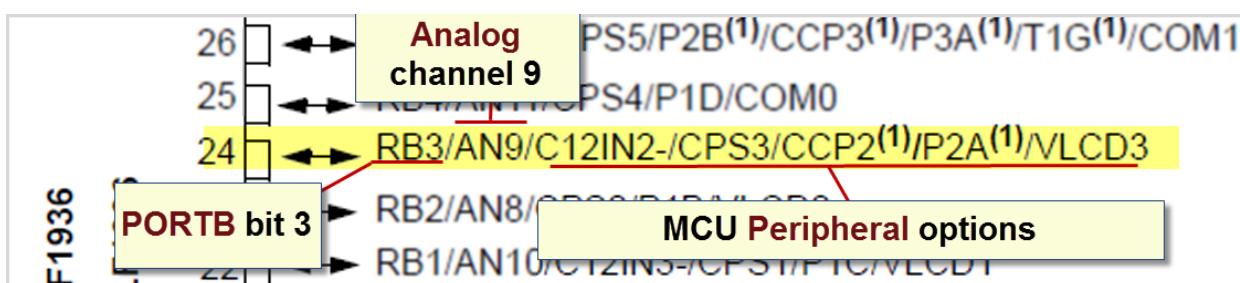


Figura 4. En el primer plano del RB3/pin 24, se muestran las opciones para el PIC16L/F1936.

El pin 24 se puede configurar como pin digital PORTB bit 3, canal analógico 9 o uno de varios periféricos



Para que un pin funcione como un pin digital, todos los periféricos asociados con el pin NO deben estar habilitados.

Si se va a utilizar un pin con capacidad analógica como pin digital, además de no habilitar el periférico, el pin debe configurarse específicamente como un pin digital.

## Configuración analógica vs digital

Dependiendo de qué MCU PIC de rango medio mejorado se utilice, se pueden configurar hasta 30 pines digitales para que sean pines analógicos. Los registros ANSELx se utilizan para configurar el modo de los pines analógicos. (ANSELA controla el modo de todos los pines con capacidad analógica en PORTA, ANSELB controla el modo para PORTB, ANSELD para PORTD, etc.).

Los pines con capacidad analógica en el puerto se asignan a bits individuales en un registro ANSELx. Un valor de 1 en un bit de un registro ANSELx habilitará el modo analógico del pin de puerto correspondiente. Un valor de 0 configura el pin para que sea digital.

En RESET todos los pines con capacidad analógica vuelven al modo analógico. El MCU establece todos los bits ANSELx relevantes en 1.

### Ejemplo:

El pin 24, en la figura 4, se puede configurar como canal analógico 9 o como pin digital RB3. Para utilizar este pin como pin digital, el bit 3 de ANSELB debe borrarse.

```
1 banksel ANSELB ; select bank  
2 bcf ANSELB, 3 ; designate bit 3 as digital
```

```
1 ANSELBbits.ANSB3 = 0 ;  
2  
3 // -- or --  
4  
5 ANSELB3 = 0 ;
```

Cuando trabaje con pines digitales con capacidad analógica, recuerde:



- Si se intenta una conversión de analógico a digital (ADC) en un pin con capacidad analógica configurado como digital, la conversión devolverá un valor invariable que no refleja el voltaje en el pin.
- Si un pin está configurado como un pin analógico, cualquier valor digital leído del pin siempre será 1 independientemente del voltaje en la almohadilla de entrada.
- Si se escribe un pin configurado analógico como si fuera un pin digital, el nivel de salida del pin no cambiará.

## Salidas digitales

El registro TRISx controla la dirección de los datos en cada bit de un puerto. Cada pin de un puerto se asigna a un bit en un registro TRIS. La dirección de datos para cada pin se puede establecer escribiendo un valor de 8 bits en el registro TRIS individualmente, configurando/borrando un bit de registro TRIS o la dirección de todos los bits en un puerto.

En RESET todos los bits asociados con los pines en los registros TRIS se establecen en 1 haciendo que todos los pines sean entradas High-Z.

## Convertir un pin en un pin de salida digital

Para configurar un pin como una salida digital, 0s debe colocarse en los bits de registro TRISx correspondientes.

## Escribir en un Pin Digital

El valor de salida para cada puerto se puede cargar escribiendo en el registro LAT del puerto. (Al igual que todos los demás registros de control de puertos, los nombres de los registros LATx están escritos. Los registros LATx comienzan con LATA y pasan por LATB, LATC...)

Escribir de 1 a un bit en el registro LAT conducirá el pin a Vdd. Un 0 en un bit LAT tirará del pin a Vss.



Escribir en el registro POTRTx también impulsará la señal de salida al igual que escribir en el registro LATx.

Sin embargo, bajo cargas altas o a alta frecuencia, si los comandos de modificación de varios bits (BSF, BCF) se escriben secuencialmente en un registro PORT de salida, es posible que la última instrucción de manipulación de bits sobrescriba una instrucción anterior, lo que resulta en un valor incorrecto en el puerto de salida. Para evitar la posibilidad de que esto ocurra, es muy recomendable que la salida se realice siempre al registro LATx.

## Código de ejemplo:

El código que se muestra aquí es un ejemplo de configuración de todos los pines en PORTB como salidas digitales. Una vez configurados como pines de salida, los cuatro bits inferiores del puerto se impulsan alto, mientras que los cuatro bits superiores se establecen en 0.



```
1 void main(void)
2 {
3     ANSELB = 0 ;          // set PORTB as digital
4     LATB = 0 ;           // set output latch to known value
5     TRISB = 0 ;          // make PORTB an output port
6     LATB = 0x0F          // set lower four bits of PORTB
7     while(1);
8 }
```



```
1 banksel ANSELB
2 clr ANSELB           ; make all pins digital
3 banksel LATB
4 clr LATB             ; set latch to known value
5 banksel TRISB
6 clr TRISB            ; set all PORTB pins outputs
7 banksel LATB
8 movlw 0x0F           ; set lower 4 pins of PORTB
9 movwf LATB           ; loop forever
10 goto $
```



¿Por qué se borró el registro LAT antes de que se estableciera el registro TRIS?  
En RESET se desconoce el contenido de los registros LATx. Se recomienda que el valor de todos los bits de registro LAT de salida se establezca en un valor conocido (y seguro) antes de habilitar los pines de salida. Esto evitará cualquier pulso de salida espurio e involuntario.

En este ejemplo, solo se cambia la configuración de RB3 a una salida digital. Una vez configurado, RB3 se impulsa alto. Todos los demás pines (y sus respectivos bits de registro de control) se dejan sin cambios.

1 void main(void)  
2 {  
3 ANSELbits.ANSB3 = 0 ; // set RB3 as digital  
4 LATBbits.LATB3 = 0 ; // set output latch to known value  
5 TRISBbits.TRISB3 = 0 ; // make RB3 an output pin  
6 LATBbits.LATB3 = 1 ; // set RB3 high  
7 while(1);  
8 }

1 banksel ANSELB  
2 bcf ANSELB,3 ; make all RB3 digital  
3 banksel LATB  
4 bcf LATB,3 ; set latch to known value  
5 banksel TRISB  
6 bcf TRISB,3 ; set RB3 as an output pin  
7 banksel LATB  
8 bsf LATB,3 ; set RB3 hight  
9 goto \$ ; loop forever

## Entradas digitales

### Hacer que un pin sea un pin de entrada digital:

Para configurar un pin como entrada digital, 1s debe colocarse en los bits de registro TRISx correspondientes.

### Lectura de un pin digital

El valor de cada pin de entrada se puede observar leyendo el registro PORTx correspondiente. Si el nivel de voltaje en un pin en particular está por encima del umbral de entrada, el bit en el registro PORTx asociado con el pin contendrá un 1. Los voltajes por debajo del umbral contendrán un 0.

Escribir en un bit de registro LATx de un pin configurado como entrada no afectará al valor del pin.

### Código de ejemplo:

Este programa de ejemplo configura RA3 (`porta` bit 3) como un pin de entrada digital. Luego, el programa monitorea continuamente el valor de RA3. Cuando RA3 se eleva, el control del programa se transfiere a una rutina de excepción.



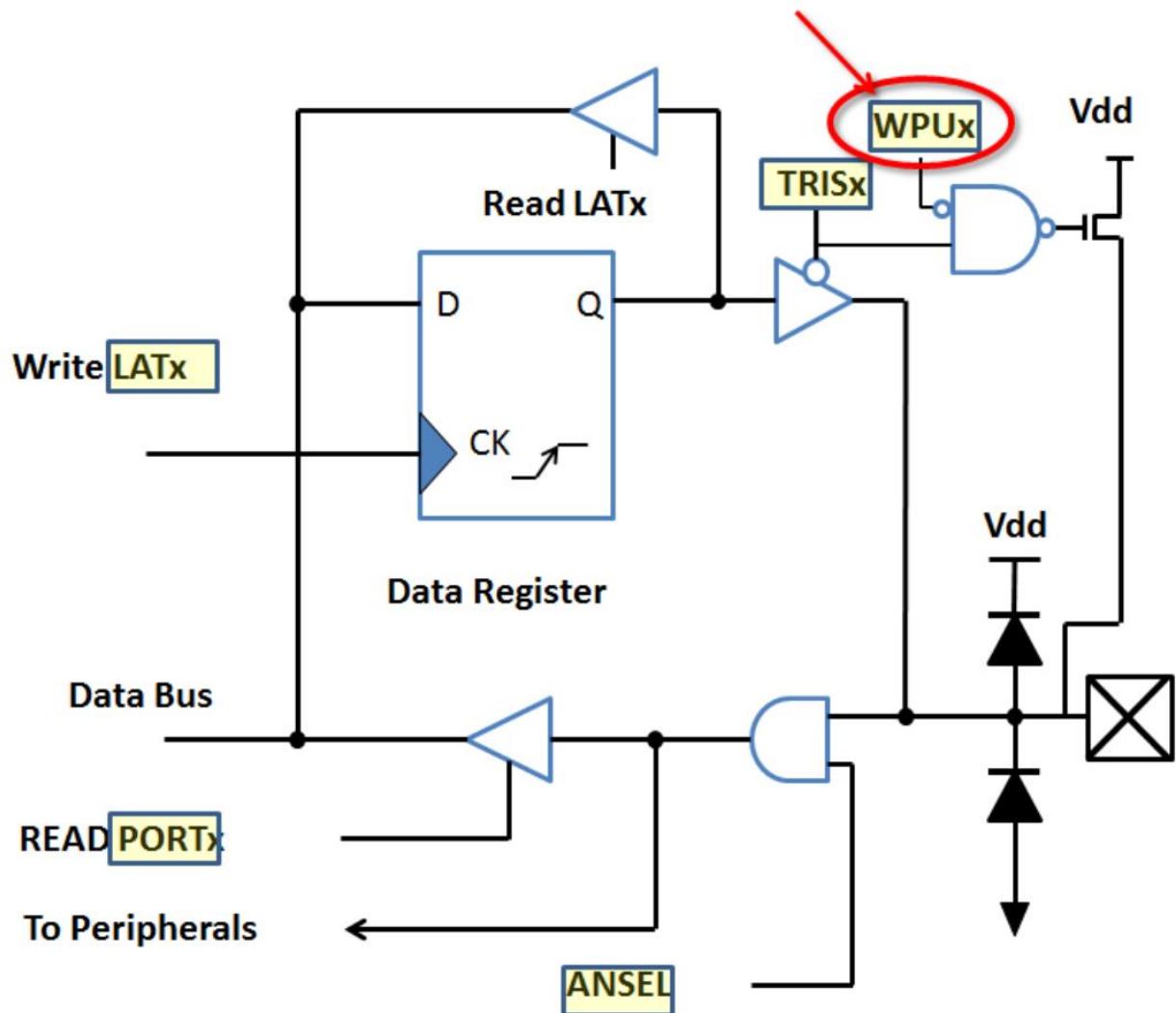
```
1 #define input_pin PORTAbits.RA3
2 void main(void)
3 {
4 ANSELAbits.ANSA3 = 0; // make RA3 a digital pin
5 TRISAbits.TRISA3 = 0; // set RA3 as an output
6 while(1)
7     if (input_pin) exception_routine();
8 }
```



```
1 Start:
2 banksel ANSELA
3 bcf ANSELA,3 ; set RA3 as digital
4 banksel TRISA
5 bsf TRISA3 ; set RA3 as an input pin
6 Loop:
7 banksel PORTA
8 btfsc PORTA,3 ; "test" RA3
9 goto Exception_Routnine ; if 1 do this
10 goto Loop
11
12 Exception_Routine
13 ; exception code goes here
```

## Pull-ups internos débiles

Las resistencias de extracción débiles internas están habilitadas para cada almohadilla de E/S utilizando el registro WP<sub>Ux</sub>.



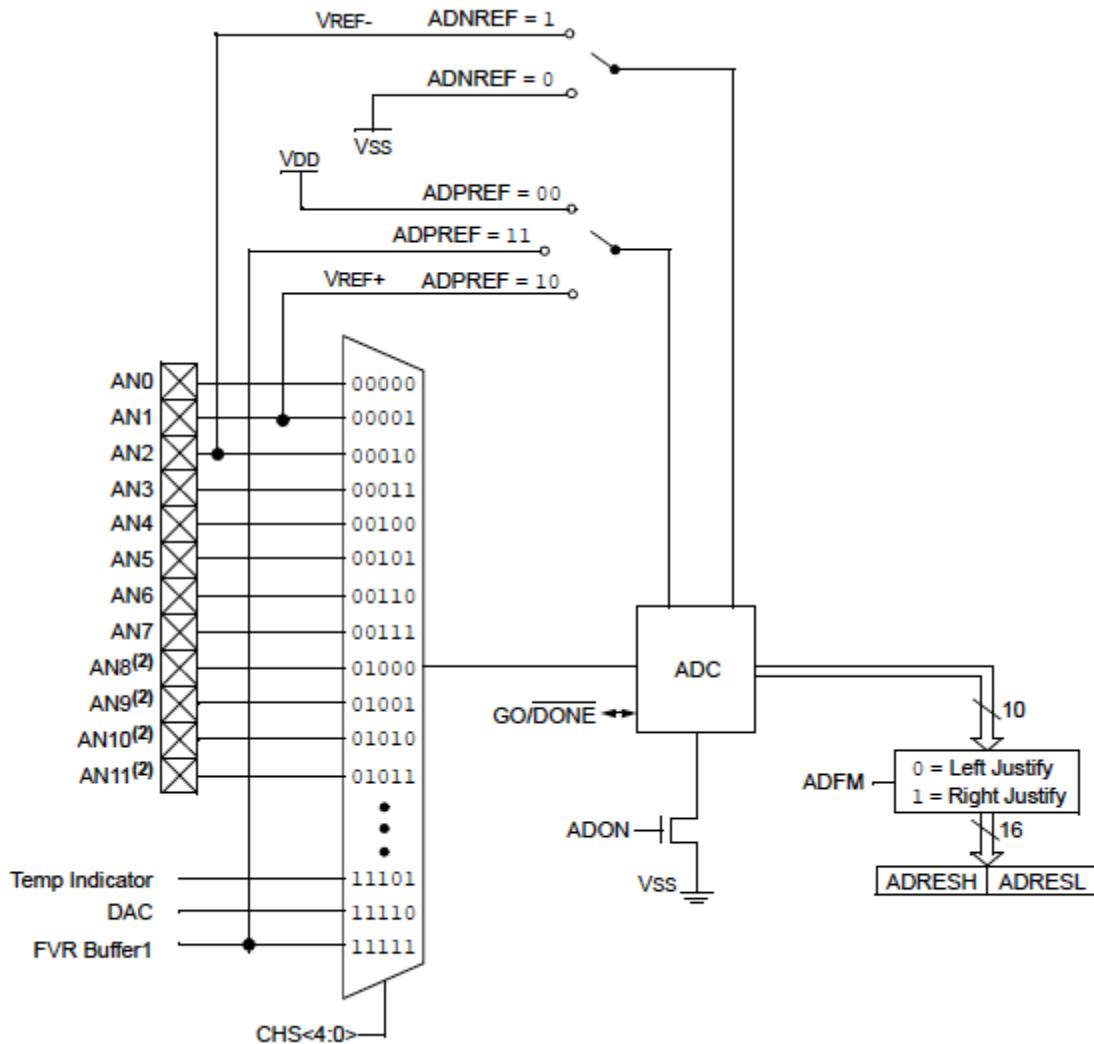
El pull-up se desactivará automáticamente cuando TRIS se establezca en una salida o el pin se configure como una entrada analógica. Estos cambios en TRIS y ANSEL anularán la configuración de WPU.

## Habilitación de Pull-ups

El pull-up interno para un pin de entrada digital se habilita escribiendo un 1 en el registro WPUx apropiado. Escribir un 0 en el registro WPUx deshabilita el pull-up.

# Convertidor de analógico a digital (ADC)

El convertidor de analógico a digital (ADC) puede convertir una señal de entrada analógica en una representación digital binaria de 10 bits de esa señal. Los microcontroladores Microchip introducen entradas analógicas, que se multiplexan en una sola muestra y circuito de retención. La salida de la muestra y la retención están conectadas a la entrada del ADC. El ADC genera el resultado binario de 10 bits a través de una aproximación sucesiva y almacena el resultado de la conversión en los registros de resultados del ADC.



El ADC utiliza una referencia de voltaje que se puede seleccionar por software para ser generada internamente o suministrada externamente.

El ADC también puede generar una interrupción al finalizar una conversión. Esta interrupción se puede utilizar para activar el dispositivo desde SLEEP.

## Configuración de ADC

Cuando el ADC se configura por primera vez, debe tener habilitadas varias opciones de configuración. Estos incluyen:

- Configuración del puerto ADC
- Selección de canales ADC



bit 6-2

### CHS<4:0>: Analog Channel Select bits

00000 = AN0

00001 = AN1

00010 = AN2

00011 = AN3

00100 = AN4

00101 = AN5

00110 = AN6

00111 = AN7

01000 = AN8<sup>(1)</sup>

01001 = AN9<sup>(1)</sup>

01010 = AN10<sup>(1)</sup>

01011 = AN11<sup>(1)</sup>

01100 = Reserved. No channel connected.

•

•

•

11100 = Reserved. No channel connected.

11101 = Temperature Indicator<sup>(4)</sup>

11110 = DAC output<sup>(2)</sup>

11111 = FVR (Fixed Voltage Reference) Buffer 1 Output<sup>(3)</sup>

Algunos dispositivos pueden tener menos canales

- Consulte la página ["Módulo de convertidor de digital a analógico \(DAC\) \(MCU de 8 bits\)"](#) para obtener más información.
- Consulte la página ["Referencia de voltaje fijo \(FVR\)"](#) para obtener más información.
- Consulte la página ["Indicador de temperatura"](#) para obtener más información.

---

## Selección de referencia de voltaje ADC

El ADC puede utilizar varias fuentes de referencia de voltaje como base para las mediciones de voltaje analógico.

**Valor digital = [Voltaje analógico / (V<sub>Ref+</sub> - V<sub>Ref-</sub>) ] \* 1024**

Los bits ADPREF del registro ADCON1 proporcionan control de la referencia de voltaje positivo. La referencia de voltaje positivo puede ser:

- V<sub>Ref+</sub>
- V<sub>Dd</sub>
- Referencia de voltaje fijo (FVR)

Los bits ADNREF del registro ADCON1 proporcionan control de la referencia de voltaje negativo. La referencia de voltaje negativo puede ser:

- V<sub>Ref-</sub>
- V<sub>Ss</sub>

V<sub>Dd</sub> y V<sub>Ss</sub> son las conexiones al bus de voltaje que alimenta el dispositivo.

$V_{Ref+}$  y  $V_{Ref-}$  son pines de E/S específicos en el dispositivo. Una referencia de voltaje externo está conectada a estos pines.

FVR es una característica en muchos dispositivos PIC, aunque no en todos. Puede incluir un solo voltaje o, a veces, hay más de un nivel de voltaje disponible.<sup>®</sup>

Los bits de selección de referencia de voltaje se encuentran en el registro ADCON1 y las opciones de selección se muestran a continuación.

#### REGISTER 16-2: ADCON1: A/D CONTROL REGISTER 1

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
ADFM	ADCS<2:0>		—	ADNREF	ADPREF<1:0>		
bit 7							bit 0

bit 2            **ADNREF: A/D Negative Voltage Reference Configuration bit**

0 =  $V_{REF-}$  is connected to  $V_{SS}$

1 =  $V_{REF-}$  is connected to external  $V_{REF-}$  pin<sup>(1)</sup>

bit 1-0          **ADPREF<1:0>: A/D Positive Voltage Reference Configuration bits**

00 =  $V_{REF+}$  is connected to  $V_{DD}$

01 = Reserved

10 =  $V_{REF+}$  is connected to external  $V_{REF+}$  pin<sup>(1)</sup>

11 =  $V_{REF+}$  is connected to internal Fixed Voltage Reference (FVR) module<sup>(1)</sup>

## Origen del reloj de conversión de ADC

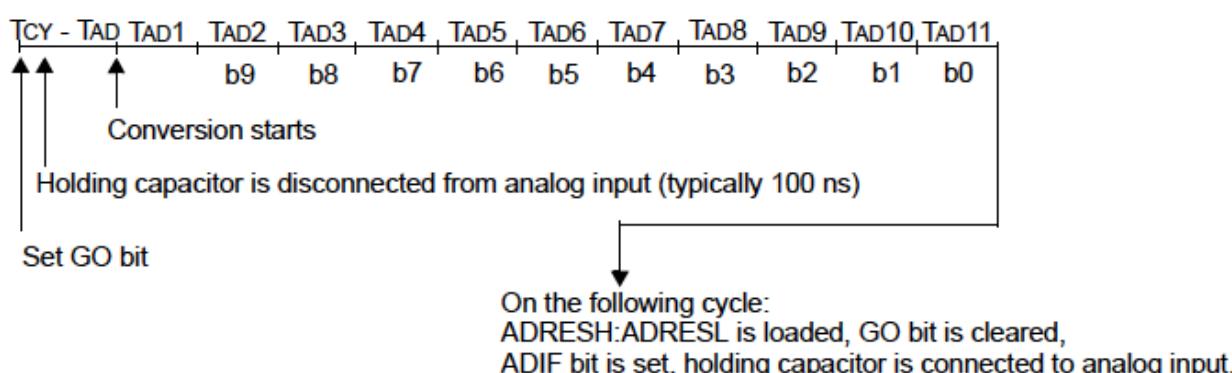
La fuente del reloj de conversión es seleccionable por software a través de los bits ADCS del registro ADCON1. Hay hasta siete opciones de reloj posibles dependiendo del dispositivo que se utilice:

- FOSC/2
- FOSC/4
- FOSC/8
- FOSC/16
- FOSC/32
- FOSC/64
- FRC (oscilador interno dedicado)

$F_{osc}$  es el oscilador del sistema que ejecuta el reloj de instrucciones del dispositivo.

El reloj es fundamental para producir la conversión analógica a digital más rápida pero también precisa.

El tiempo para completar la conversión de un bit se define como  $T_{Anuncio}$ . Una conversión completa de 10 bits requiere 11,5  $T_{Anuncio}$  períodos como se muestra aquí:



Para una conversión correcta, la T<sub>Anuncio</sub> se deben cumplir las especificaciones. Un reloj ADC se puede seleccionar fácilmente en el gráfico a continuación. Un gráfico similar aparece en la hoja de datos del dispositivo. Los valores que son los mejores aparecen en el centro del gráfico en un fondo blanco.

ADC Clock Period (T <sub>AD</sub> )		Device Frequency (Fosc)					
ADC Clock Source	ADCS<2:0>	32 MHz	20 MHz	16 MHz	8 MHz	4 MHz	1 MHz
Fosc/2	000	62.5 ns <sup>(2)</sup>	100 ns <sup>(2)</sup>	125 ns <sup>(2)</sup>	250 ns <sup>(2)</sup>	500 ns <sup>(2)</sup>	2.0 µs
Fosc/4	100	125 ns <sup>(2)</sup>	200 ns <sup>(2)</sup>	250 ns <sup>(2)</sup>	500 ns <sup>(2)</sup>	1.0 µs	4.0 µs
Fosc/8	001	0.5 µs <sup>(2)</sup>	400 ns <sup>(2)</sup>	0.5 µs <sup>(2)</sup>	1.0 µs	2.0 µs	8.0 µs <sup>(3)</sup>
Fosc/16	101	800 ns	800 ns	1.0 µs	2.0 µs	4.0 µs	16.0 µs <sup>(3)</sup>
Fosc/32	010	1.0 µs	1.6 µs	2.0 µs	4.0 µs	8.0 µs <sup>(3)</sup>	32.0 µs <sup>(3)</sup>
Fosc/64	110	2.0 µs	3.2 µs	4.0 µs	8.0 µs <sup>(3)</sup>	16.0 µs <sup>(3)</sup>	64.0 µs <sup>(3)</sup>
FRC	x11	1.0-6.0 µs <sup>(1,4)</sup>					

**Legend:** Shaded cells are outside of recommended range.

**Note 1:** The FRC source has a typical T<sub>AD</sub> time of 1.6 µs for V<sub>DD</sub>.

**2:** These values violate the minimum required T<sub>AD</sub> time.

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

**4:** The ADC clock period (T<sub>AD</sub>) and total ADC conversion time can be minimized when the ADC clock is derived from the system clock Fosc. However, the FRC clock source must be used when conversions are to be performed with the device in Sleep mode.

La selección FRC del oscilador interno será una conversión más lenta pero garantizará la T<sub>Anuncio</sub> se cumplen los requisitos. El FRC también se puede utilizar en modo de suspensión para ejecutar mediciones de ADC.

## Control de interrupciones

El módulo ADC tiene la capacidad de generar una interrupción al completar una conversión de analógico a digital. Esta interrupción también se puede generar mientras el dispositivo está funcionando o mientras está en reposo. Si el dispositivo está en SUSPENSIÓN, la interrupción activará el dispositivo y, a continuación, procesará la rutina de servicio de interrupción (ISR) siempre que los bits de interrupción estén habilitados.

Esos bits de interrupción incluyen:

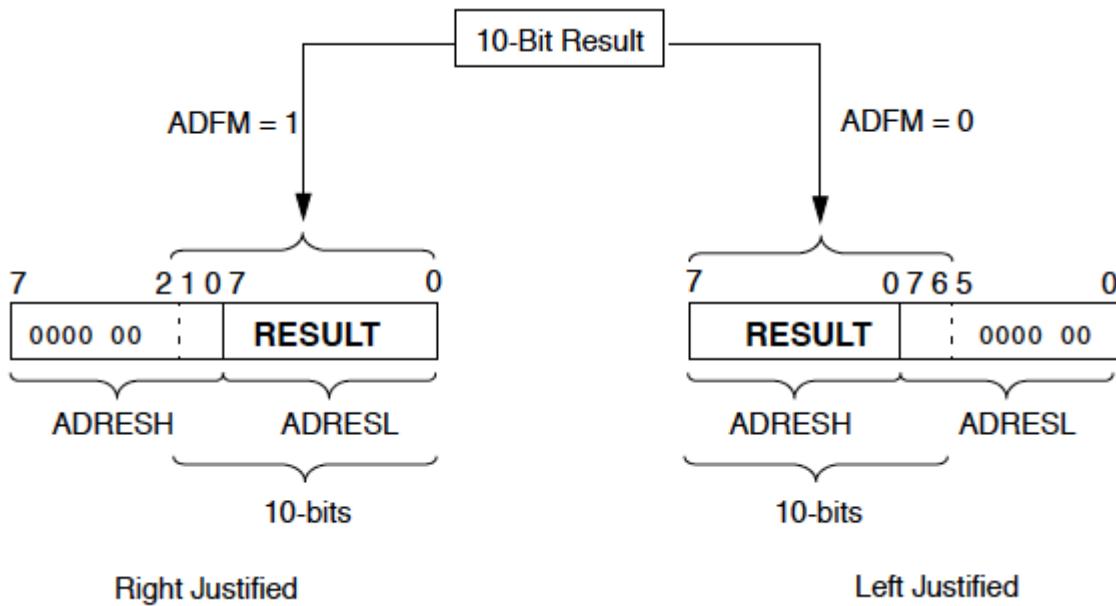
- El indicador de interrupción de ADC es el bit ADIF en el registro de interrupción periférica (PIR1) 1.
- La habilitación de interrupciones de ADC es el bit ADIE en el registro de habilitación de interrupciones periféricas (PIE1).
- También deben habilitarse el bit Global Interrupt Enable (GIE) y el peripheral Interrupt Enable (PEIE) en el registro INTCON.

Después de que se ejecuta una interrupción del modo SLEEP y se completan el ISR y el ADC, el bit ADIF debe borrarse en el software.

## Formato de resultado de ADC

El resultado de la conversión de ADC se almacena en dos registros de 8 bits de ancho; ADRESH y ADRESL. Este par de registros tiene un ancho de 16 bits, por lo que el módulo ADC tiene la

flexibilidad de justificar a la izquierda o a la derecha el resultado de 10 bits en el registro de resultados de 16 bits. El bit de selección de formato ADC (ADFM) del registro de ADCON1 controla esta justificación. Los bits adicionales en los registros ADRESH y ADRESL se cargan con '0's.



bit 7

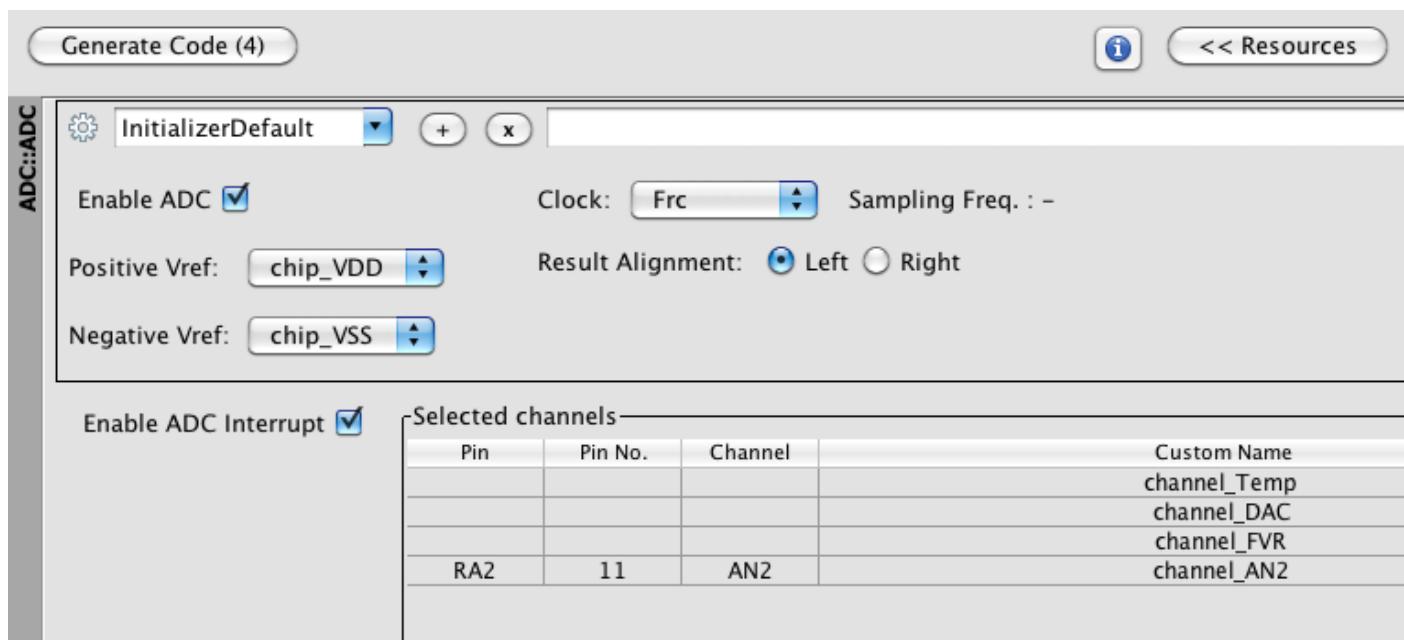
**ADFM: A/D Result Format Select bit**

- 1 = Right justified. Six Most Significant bits of ADRESH are set to '0' when the conversion result is loaded.
- 0 = Left justified. Six Least Significant bits of ADRESL are set to '0' when the conversion result is loaded.

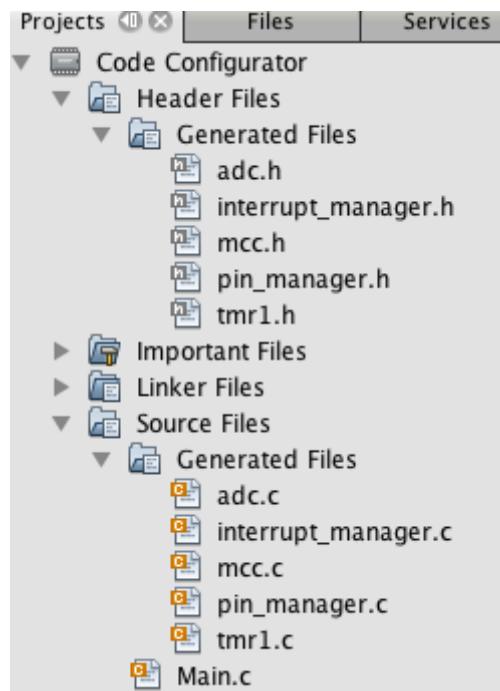
El resultado puede copiarse en una variable o usarse en una ecuación para implementar una función basada en el resultado de ADC.

## MpLAB Code Configurator - Configuración de ADC®

[MPLAB® Code Configurator \(MCC\)](#) hace que la configuración del código ADC sea bastante fácil. Todas las opciones de configuración descritas anteriormente se pueden configurar en una pantalla GUI simple dentro de MPLAB X IDE. La pantalla ADC se muestra aquí:

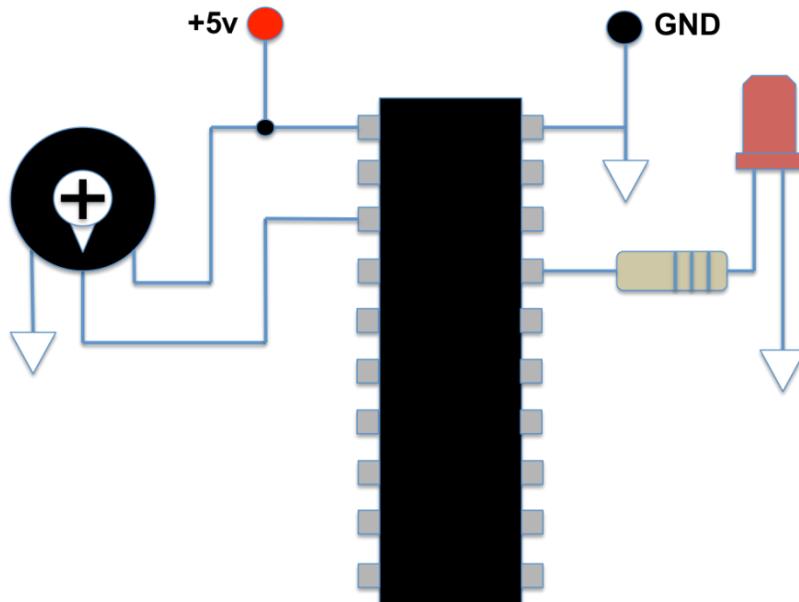


Cada opción de configuración se selecciona como una casilla de verificación o desde un menú desplegable. Después de realizar las selecciones, el código se genera y se coloca en el proyecto. Se crean dos archivos denominados `adc.h` y `adc.c`. Contienen código de configuración de ADC y también funciones personalizadas para usar el ADC dentro de los archivos de proyecto principales.



## Ejemplo de proyecto de ADC

Aquí hay un proyecto de ejemplo paso a paso para configurar el [ADC mediante MCC](#).



# Celda lógica configurable

La celda lógica configurable (CLC) proporciona lógica programable que funciona fuera de las limitaciones de velocidad de la ejecución de software. La celda lógica toma hasta 16 señales de entrada y, mediante el uso de puertas configurables, reduce las 16 entradas en cuatro líneas lógicas que impulsan una de las ocho funciones lógicas de salida única seleccionables.

Las fuentes de entrada son una combinación de lo siguiente:

- Pines de E/S
- Relojes internos
- Periféricos
- Registrar bits

La salida se puede dirigir internamente a periféricos y a un pin de salida.

Las configuraciones posibles incluyen:

- Lógica combinatoria
  - Y
  - NAND
  - Y-O
  - Y-O-INVERTIR
  - OR-XOR
  - OR-XNOR
- Cierres
  - S-R
  - Reloj D con Set y Reset
  - D transparente con Set y Reset
  - J-K con reloj con reset

---

## CLC Video Tutorial

<https://youtu.be/rNE1sI21x5M>

Este vídeo presenta la celda lógica configurable (CLC) para dispositivos MCU de 8 bits de Microchip y muestra cómo usarla.

---

## Configuración de CLC

El periférico CLC tiene cuatro secciones que deben configurarse antes de poder utilizarse. Esto implica configurar ocho registros en su programa de software. Una vez que se configuran estos registros, el CLC se ejecutará independientemente del control del software hasta que los registros se cambien a través del software.

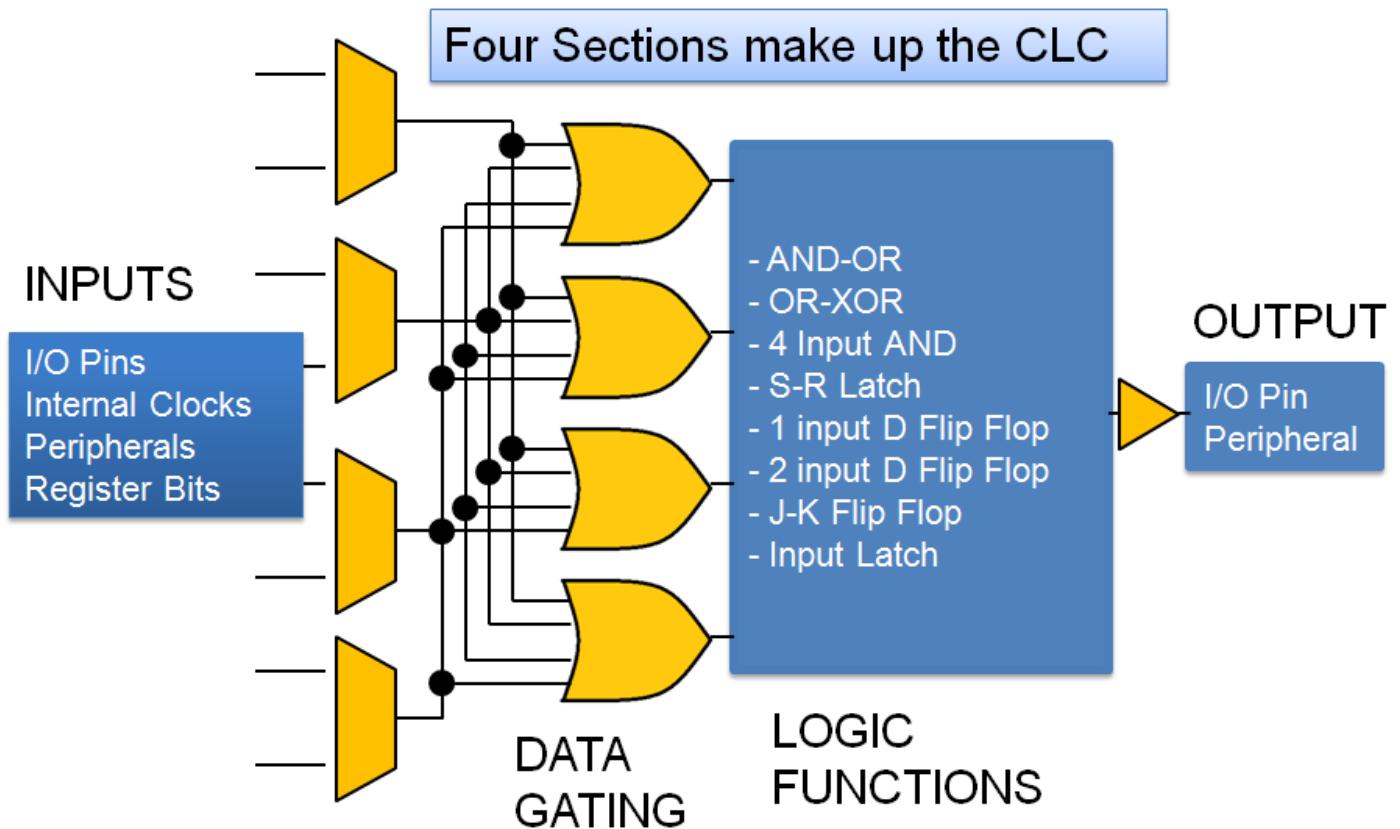
Entre ellos se incluyen:

- CLCxCON
- CLCxSEL0
- CLCxSEL1
- CLCxGLS0
- CLCxGLS1
- CLCxGLS2
- CLCxGLS3
- CLCxPOL

Un **Pic®** El dispositivo puede tener varios CLC, por lo que cada módulo CLC tiene su propio conjunto de ocho registros. La x en los nombres de registro anteriores representa el número CLC (por ejemplo, CLC1 utiliza el registro CLC1CON).

Para simplificar la configuración, el CLC se puede dividir en cuatro secciones que deben configurarse. Entre ellos se incluyen:

- Entradas
- Cierre de datos
- Función lógica
- Configuración de salida



## Entradas

Las entradas pueden provenir de 8-16 fuentes diferentes, dependiendo del dispositivo PIC, y de esta lista, se pueden elegir hasta cuatro para alimentar la sección de cierre de datos. Pueden incluir:

- Pines de E/S
- Salidas de reloj internas

- Salidas de periféricos
- Registrar bits

Las entradas se seleccionan por bits en los registros CLCxSEL0 y CLCxSEL1.



Cada entrada tiene un código de 3 bits asociado que se coloca en los registros CLCxSEL para habilitar la entrada.

CLC 1 Input	Source
CLC1IN[0]	CLC1IN0 PIN
CLC1IN[1]	CLC1IN1 PIN
CLC1IN[2]	SYNCC1OUT
CLC1IN[3]	SYNCC2OUT
CLC1IN[4]	Fosc
CLC1IN[5]	TMR0IF
CLC1IN[6]	TMR1IF
CLC1IN[7]	TMR2 = PR2
CLC1IN[8]	Ic1_out
CLC1IN[9]	Ic2_out
CLC1IN[10]	Ic3_out
CLC1IN[11]	Ic4_out
CLC1IN[12]	NCO1OUT
CLC1IN[13]	HFINTOSC
CLC1IN[14]	PWM3OUT
CLC1IN[15]	PWM4OUT

## Cierre de datos

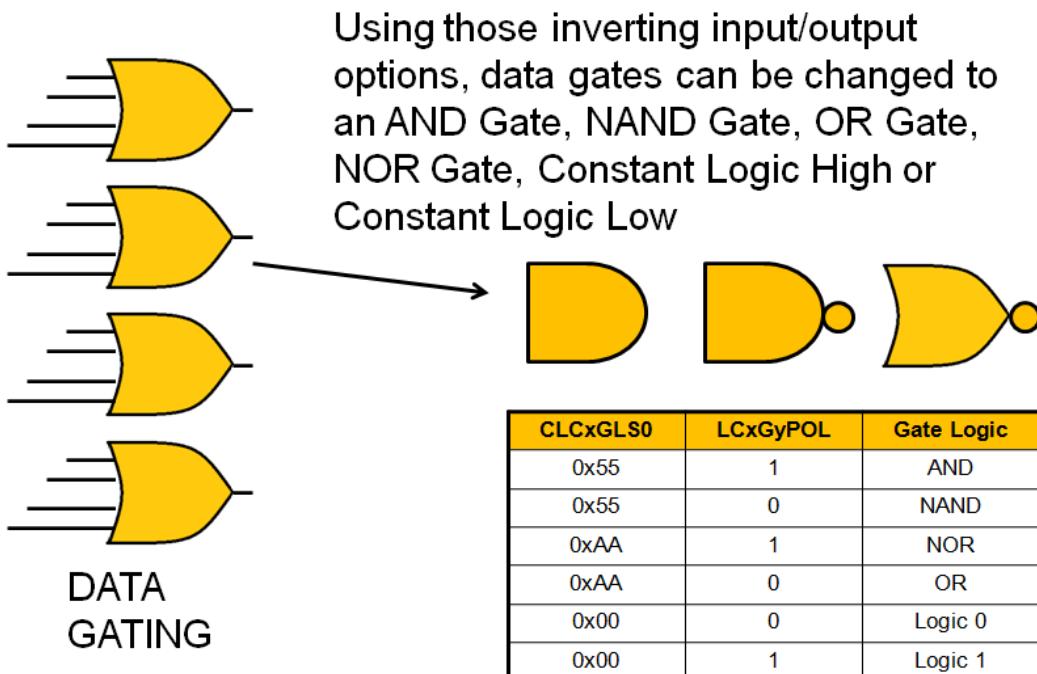
La sección Data Gating tiene cuatro puertas lógicas que deben configurarse. Esto requiere que se establezcan cinco registros separados. Configuran la conexión invertida o no invertida desde las entradas que controlan el periférico CLC. Los cinco registros incluyen:

- CLCxGLS0

- CLCxGLS1
- CLCxGLS2
- CLCxGLS3
- CLCxPOL

Cada puerta comienza como una puerta OR base, pero cada entrada y salida se puede invertir individualmente o no invertir.

Esto permite crear puertas AND, NAND, OR y NOR. Las puertas también se pueden configurar para impulsar un nivel lógico constante de 1 o 0.



Cada entrada a una puerta de datos tiene un par de bits en uno de los registros CLCxGLSx. Los dos bits incluyen un bit no invertido (T) y un bit invertido (N) que debe configurarse. Si se establece el bit T, entonces la entrada no está invertida. Si se establece el bit N, la entrada se invierte. Si ambos se establecen en cero, entonces la entrada no está conectada a la puerta.

### CLC1POL

LC1POL	-	-	-	LC1G4POL	LC1G3POL	LC1G2POL	LC1G1POL 0
--------	---	---	---	----------	----------	----------	---------------

### CLC1GLS0

LC1G1D4T 0	LC1G1D4N 1	LC1G1D3T 0	LC1G1D3N 1	LC1G1D2T 0	LC1G1D2N 1	LC1G1D1T 0	LC1G1D1N 1
Input 4		Input 3		Input 2		Input 1	

El bit de registro CLCxPOL, bit LCxGxPOL, invertirá o no invertirá la salida de la puerta.

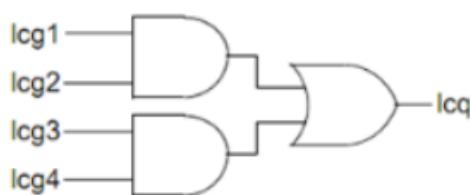
0 - no invertido

1 - invertido

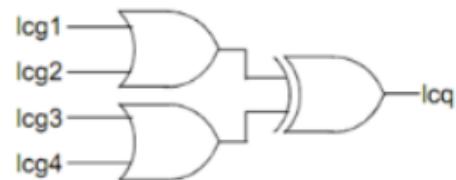
## Función lógica

La función lógica tiene ocho opciones para elegir. Se selecciona en el registro CLCxCON. Cada función lógica tiene un código de 3 bits asociado.

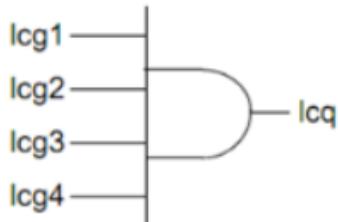
**000 = AND – OR**



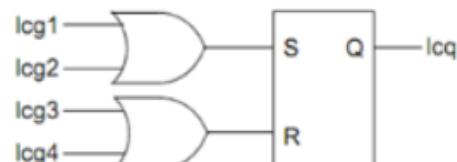
**001 = OR – XOR**



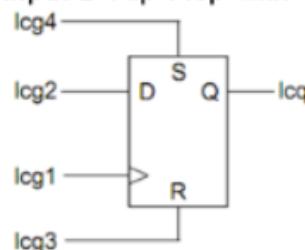
**010 = 4-Input AND**



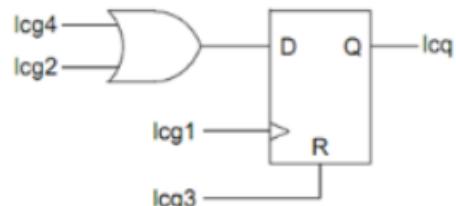
**011 = S-R Latch**



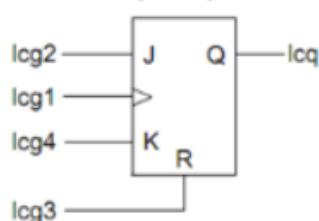
**100 = 1-Input D Flip-Flop with S and R**



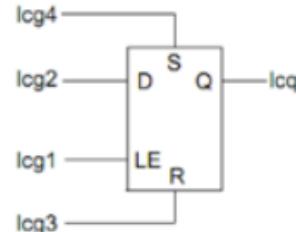
**101 = 2-Input D Flip-Flop with R**



**110 = J-K Flip-Flop with R**



**111 = 1-Input Transparent Latch with S and R**



El código de 3 bits se establece en los bits LCxMODE del registro CLCxCON para habilitar la función lógica seleccionada.

## CLCxCON

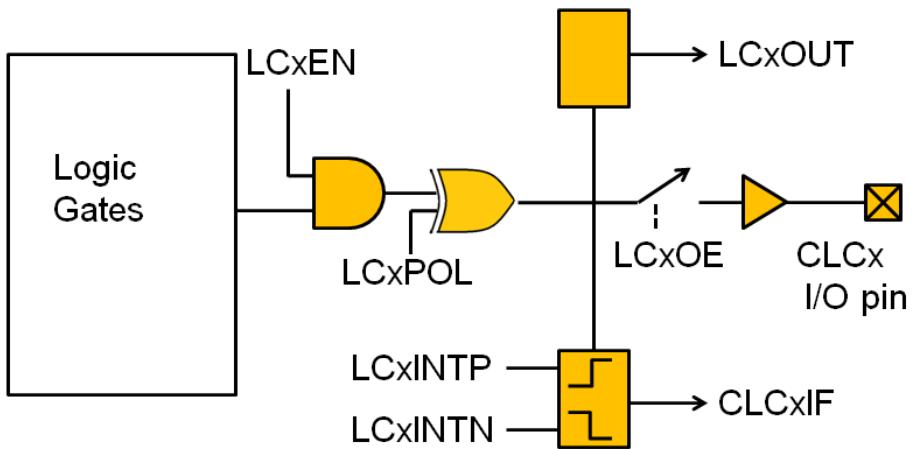
LCxMODE < 2 : 0 >

-	-	-	-	-	1	1	0
---	---	---	---	---	---	---	---

## Salida

Todas las secciones de CLC se reducen a una sola salida que puede manejar un pin de E/S, alimentar otro módulo CLC o periférico interno, o también puede desencadenar una interrupción de borde ascendente o descendente. Estas diversas opciones se configuran en los registros CLCxCON y CLCxPOL.

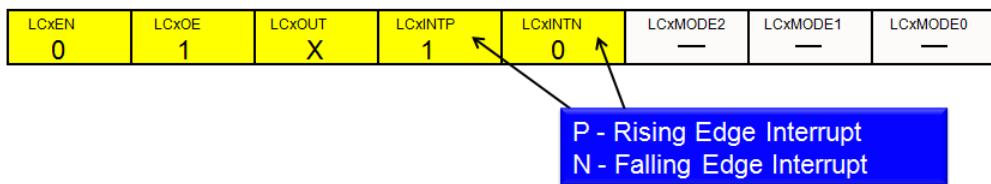
There are multiple bits that control the output from the CLC module



Los bits del registro CLCxCON controlan la configuración de salida.

- LCxEN – Bit de habilitación del módulo CLC (1 - CLC On, 0 - Off)
- LCxOE – Bit de habilitación de salida (1 – Enable, 0 – Disable)
- LCxOUT - Monitoree internamente la salida a través de software (Read Only Bit)
- LCxINTP – Rising edge interrupt enable (1-CLCxIF set on Rising Edge)
- LCxINTN – Falling edge interrupt enable (1-CLCxIF set on Falling Edge)

CLCxCON

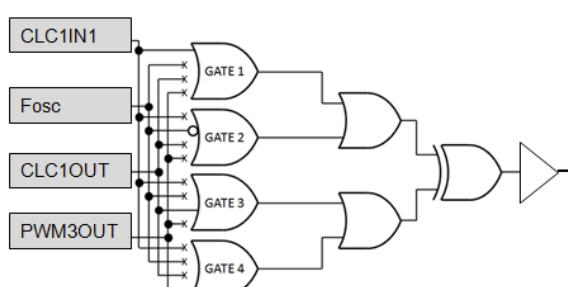


## Ejemplo de CLC

Aquí hay un ejemplo simple que muestra los ocho registros configurados en el software para crear la configuración de CLC que se muestra en la imagen.

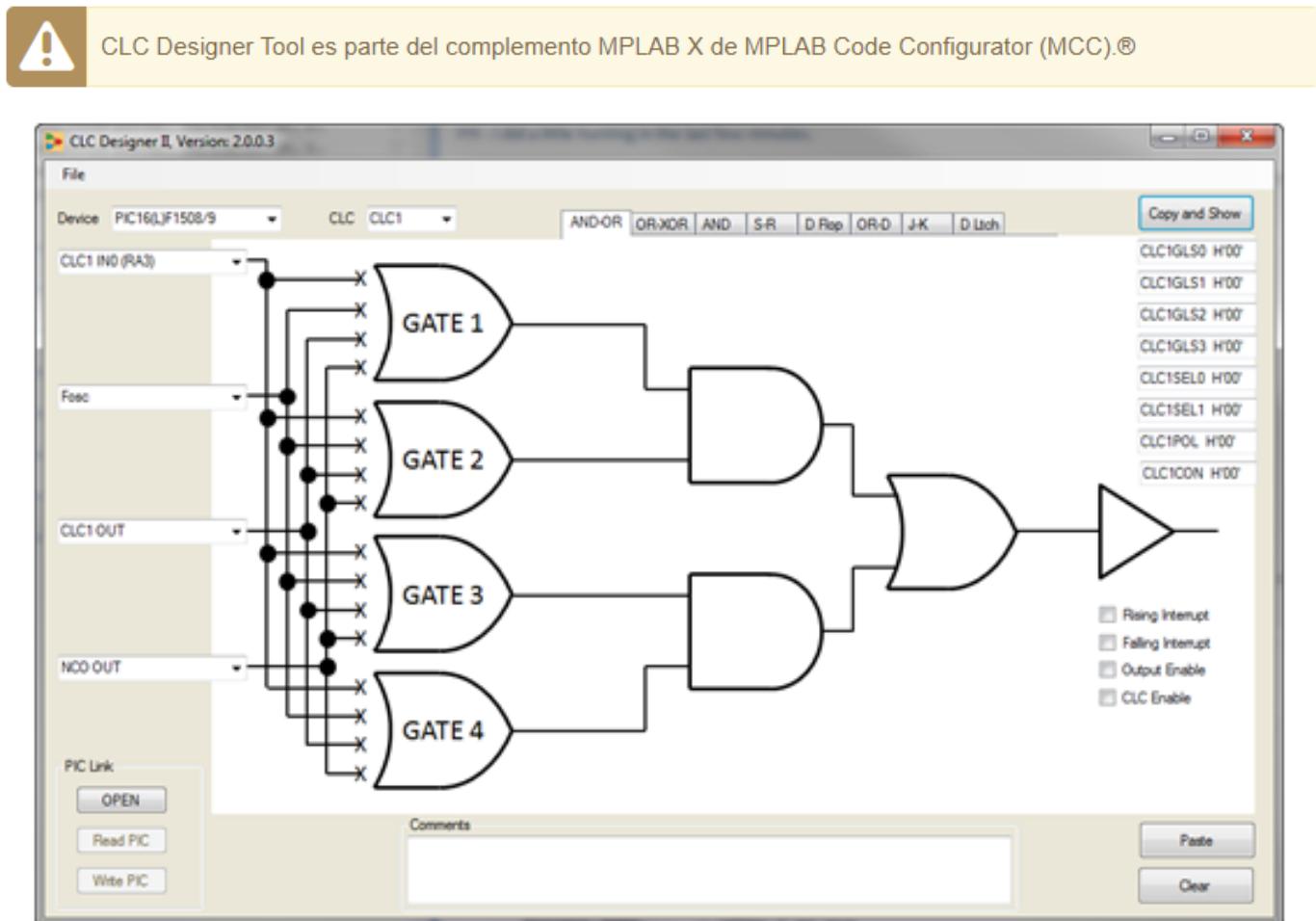
The example below shows a setup for the CLC1 module.  
The eight register settings are shown for this example in a format for the XC8 compiler.

```
// CLC 1 Setup
CLC1SEL0 = 0x01; // CLC1IN1 Pin, Fosc Inputs
CLC1SEL1 = 0x02; // CLC1OUT, PWM3OUT Inputs
CLC1GLS0 = 0x01; // Input 1 not inverted
CLC1GLS1 = 0x04; // Input 2 inverted
CLC1GLS2 = 0x20; // Input 3 not inverted
CLC1GLS3 = 0x80; // Input 4 not inverted
CLC1POL = 0x00; // Output of CLC1 is not inverted
CLC1CON = 0xD2; // Enable OR-XOR, Rising Edge Interrupt, Output Pin Enabled, CLC enabled
```



# Herramienta CLC Designer

CLC Designer Tool es una herramienta basada en GUI que facilita mucho la creación de la estructura de CLC. A través de una serie de opciones de configuración, la herramienta generará automáticamente los ocho ajustes de registro para que pueda incluirlo en su **MPLAB X®** proyecto.



# Herramienta de configuración de CLC

<https://youtu.be/OT34K3v4OTI>

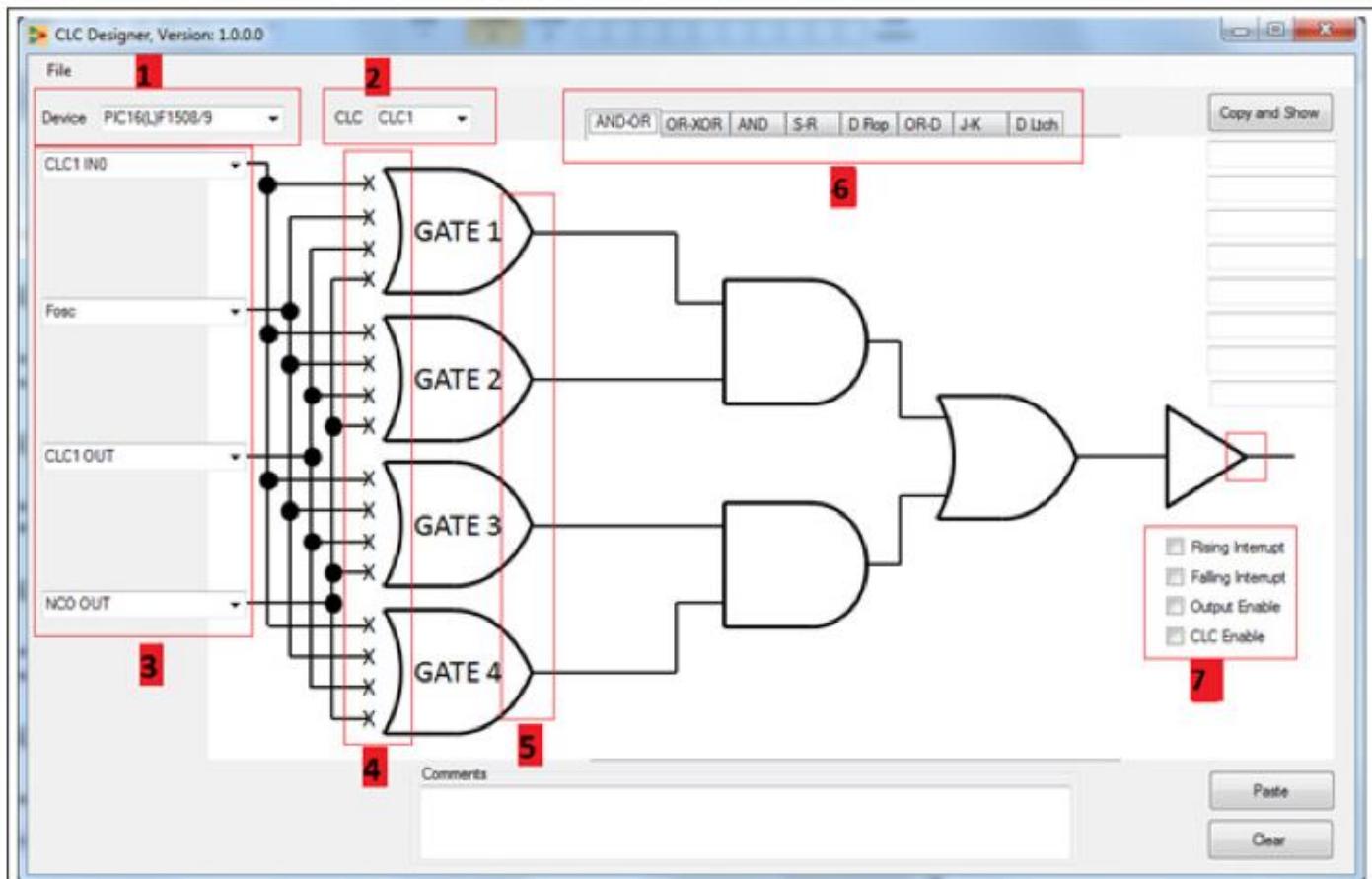
Configurable Logic Cell (CLC) proporciona lógica programable que opera fuera de las limitaciones de velocidad de la ejecución de software. La celda lógica toma hasta 16 señales de entrada. Mediante el uso de puertas configurables, reduce las 16 entradas a cuatro líneas lógicas que impulsan una de las ocho funciones lógicas de salida única seleccionables. El CLC puede tener su funcionalidad preprogramada o programada dinámicamente. Esto proporciona una mayor flexibilidad y potencial en los diseños integrados.

El propósito de la herramienta de configuración de CLC es agilizar el proceso de configuración del módulo CLC simulando la funcionalidad de los registros en una interfaz gráfica de usuario (GUI). El resultado final del uso de la herramienta será un código fuente generado, que se puede colocar en un **MPLAB X®** archivo de proyecto. El ejemplo de código creado se genera de forma personalizada, en función de las entradas y preferencias, como el lenguaje de programación.

La herramienta CLC forma parte de la herramienta **MPLAB Code Configurator (MCC)**.

El video en la parte superior de la página muestra un ejemplo paso a paso del uso de la herramienta de configuración de CLC. A continuación se muestra una guía escrita paso a paso.

## Pasos de configuración de la GUI de CLC



**1**

## Seleccione el dispositivo

Aquí es donde se seleccionará el dispositivo, como el [PIC16F1508](#). Cuando se selecciona un dispositivo, el programa se configurará automáticamente en ese dispositivo específico, como las entradas de datos y el número de salidas CLC disponibles.

**2**

## Seleccione el número de CLC

Un dispositivo puede tener varios CLC, por lo que deberá seleccionar el número de CLC que desea configurar. Algunos dispositivos, como el [PIC10F320](#), solo tendrán un módulo CLC disponible en el dispositivo seleccionado. La "X" en cada registro de CLC será reemplazada por cualquier módulo de CLC que se utilice.

**3**

## Seleccione las entradas de datos

Hay cuatro grupos de selección de entrada. Cada grupo consta de ocho selecciones. Para un dispositivo con solo ocho entradas, las ocho entradas están disponibles en cada grupo. Para un dispositivo con 16 entradas, solo ocho de las 16 están disponibles en cada grupo. Sin embargo, estas entradas se distribuyen de una manera que minimiza la exclusión de algunas combinaciones de selección de entradas. Ninguna entrada aparecerá dos veces en el mismo grupo, pero aparecerá como una entrada en otros grupos.

**4**

## Entradas de puerta

Una vez que se seleccionan las entradas de datos, se pueden mapear en cada una de las cuatro puertas. La salida de cada puerta diferirá según la función lógica seleccionada. Para seleccionar la entrada en una puerta, simplemente coloque el cursor sobre la "X" deseada y haga clic en ella una vez. La flecha del cursor habrá cambiado al puntero y aparecerá una línea que extiende la entrada hacia la puerta. Para invertir la señal, haga clic de nuevo donde estaba la "X" y ahora debería aparecer una burbuja, que indica una inversión. Si se hace clic una vez más, la burbuja y la línea deberían desaparecer y volver de forma predeterminada al estado original no conectado.

**5**

## Salidas de compuerta

Cada una de las salidas de la puerta se puede invertir. Para hacer esto, simplemente haga clic una vez en la salida de una puerta individual para que aparezca una burbuja. La salida ahora está invertida. Para deshacer esto, haga clic en la burbuja nuevamente para que desaparezca. Es importante tener en cuenta que cualquier puerta sin entradas seleccionadas tendrá su salida predeterminada en el estado "apagado", lógica 0. Si se desea una lógica constante 1, invierta la lógica predeterminada 0 haciendo clic en la salida de la burbuja inversora.

**6**

## Seleccione el bloque lógico

Hay ocho funciones lógicas disponibles seleccionadas por las pestañas de la herramienta CLC. Los bloques lógicos no se pueden configurar para nada que no sea lo que se muestra. Solo se puede utilizar una función lógica a la vez para cada módulo CLC.

**7**

## Control de salida

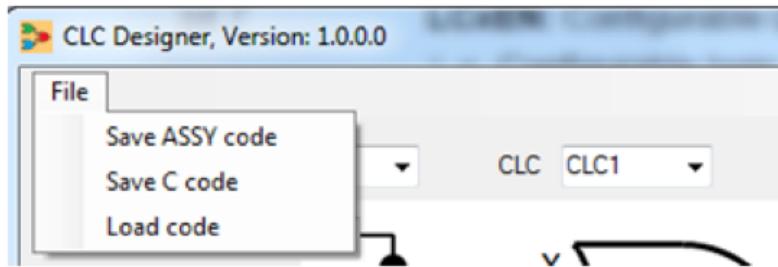
La salida del bloque lógico se alimenta a la última etapa del CLC, la puerta de inversión. Para invertir la salida, haga clic en el pin de salida del búfer una vez para que aparezca una burbuja.

Desde aquí, la salida se puede enrutar a otros periféricos, un pin de salida o de vuelta a la entrada CLC. Se puede habilitar una interrupción en el borde ascendente y/o descendente de la salida CLC. Estas funciones se seleccionan marcando las casillas en la esquina inferior derecha de la pantalla de la herramienta CLC.

## 8

## Guardar/Cargar

La herramienta de configuración de CLC proporciona un método conveniente para guardar el diseño (o cargar un diseño anterior). Cuando el diseño esté concluido y listo para ser implementado en el software, haga clic en el menú desplegable **Archivo** en la esquina superior izquierda del cuadro de diálogo.



A continuación, haga clic en **Guardar código ASSY** o **Guardar código C**, dependiendo del idioma de salida deseado. El código para todos los CLC configurados del dispositivo seleccionado se incluirá en el archivo de salida. El archivo resultante tendrá una extensión .inc.

Ambas piezas de código producen el mismo efecto. La Asamblea es más larga debido a la naturaleza del idioma. El código ahora se puede incluir fácilmente como un archivo de biblioteca o copiar y pegar en un programa existente.

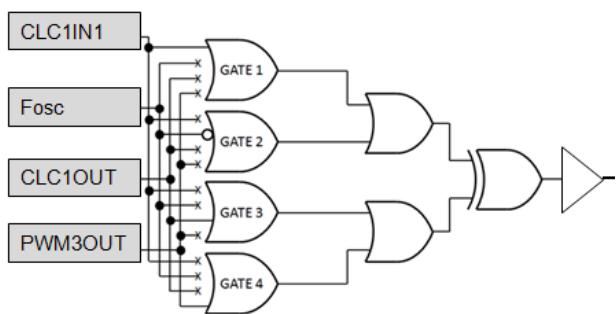
- **Comentarios del proyecto:** Los comentarios también se pueden guardar y cargar dentro del archivo de salida. Para hacerlo, simplemente complete el área de texto de entrada de comentarios como se ve en la parte inferior de la pantalla de la Herramienta de configuración de CLC antes de guardar.
- **Carga de un archivo existente:** Para cargar código guardado previamente desde la herramienta CLC, haga clic en **Archivo > Cargar código >**. Si se importa correctamente, la herramienta habrá llenado la GUI con los valores apropiados correspondientes a los registros en el código cargado.

## Ejemplo

Aquí hay un ejemplo simple creado con la herramienta de configuración de CLC. Muestra que los ocho registros se han configurado en software para crear la configuración de CLC que se muestra en la imagen.

The example below shows a setup for the CLC1 module. The eight register settings are shown for this example in a format for the XC8 compiler.

```
// CLC 1 Setup
CLC1SEL0 = 0x01; // CLC1IN1 Pin, Fosc Inputs
CLC1SEL1 = 0x02; // CLC1OUT, PWM3OUT Inputs
CLC1GLS0 = 0x01; // Input 1 not inverted
CLC1GLS1 = 0x04; // Input 2 inverted
CLC1GLS2 = 0x20; // Input 3 not inverted
CLC1GLS3 = 0x80; // Input 4 not inverted
CLC1POL = 0x00; // Output of CLC1 is not inverted
CLC1CON = 0xD2; // Enable OR-XOR, Rising Edge Interrupt, Output Pin Enabled, CLC enabled
```



## Descarga/Documentación

La herramienta de configuración de CLC forma parte del configurador de código mpLAB. Puede obtener la Herramienta de configuración de CLC, el Manual del usuario y la Guía de consejos y trucos de CLC directamente en los siguientes enlaces:



La herramienta CLC ahora está incorporada en la herramienta MPLAB Code Configurator (MCC). Puede encontrar información sobre cómo instalar MCC Tool visitando la página [MPLAB Code Configurator \(MCC\)](#).

- ["Guía del usuario de la herramienta de configuración configurable Logic Cell \(CLC\)"](#)
- ["Configurable Logic Cell Tips 'n Tricks"](#)

# Registro CLCxCON

La sección Salida y la sección Lógica de la [celda lógica configurable \(CLC\)](#) están controladas por el Registro CLCCON.

## CLCxCON: Registro de control de celda lógica configurable

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
LCxEN	LCxOE	LCxOUT	LCxINTP	LCxINTN	LCxMODE2	LCxMODE1	LCxMODE0

bit 7

bit 0

### Leyenda

R = Bit legible

W = Bit grabable

U = Bit no implementado, leído como '0'

u = Bit no ha cambiado

x = bit es desconocido

-n/n = Valor en POR y BOR/Valor en todos los demás reinicios

'1' = bit está establecido

'0' = bit está borrado

-n = Valor en el restablecimiento de POR

- bit 7      **LCxEN:** CLC Habilitar bit  
1 = CLC está habilitado y la mezcla de señales de entrada 0 = CLC está deshabilitada y tiene salida lógica cero
- bit 6      **LCxOE:** Bit de habilitación de salida CLC  
1 = Salida del pin del puerto CLC habilitada  
0 = Salida del pin del puerto CLCI deshabilitada
- bit 5      **LCxOUT:** Bit de salida de datos CLC  
Solo lectura: datos de salida de celda lógica, después de LCxPOL; muestreado de alambre lcx\_out
- bit 4      **LCxINTP:** CLC Positive Edge Going Interrupt Enable bit  
1 = LCxF se establecerá cuando se produzca una arista ascendente en lcx\_out  
0 = LCxF no se establecerá
- bit 3      **LCxINTN:** CLC Negative Edge Going Interrupt Enable bit  
1 = LCxF se establecerá cuando se produzca una caída de la arista en lcx\_out  
0 = LCxF no se establecerá
- bit 2-0     **LCxMODE<2:0>:** Bits del modo funcional de CLC  
111 = La celda es un pestillo transparente de 1 entrada con S y R  
110 = La celda es J-K Flip-Flop con R  
101 = La celda es 2-input D Flip-Flop con R  
100 = La celda es 1-input D Flip-Flop con S y R  
011 = La celda es S-R latch  
010 = La celda es de 4 entradas Y  
001 = La celda es OR-XOR  
000 = La celda es AND-OR

De la [hoja de datos de PIC16F1507](#).

---

El bit CLC Enable habilitará o deshabilitará el módulo CLC. Un 1 lo habilita y un 0 lo deshabilita.

bit 7      **LCxEN:** Configurable Logic Cell Enable bit

1 = Configurable logic cell is enabled and mixing input signals

0 = Configurable logic cell is disabled and has logic zero output

---

El bit CLC Output Enable habilitará o deshabilitará la salida del módulo CLC. Un 1 lo habilita y un 0 lo deshabilita.

bit 6	<b>LCxOE:</b> Configurable Logic Cell Output Enable bit 1 = Configurable logic cell port pin output enabled 0 = Configurable logic cell port pin output disabled
-------	--

---

El bit LCOUT de CLC es un indicador que se puede monitorear en software para determinar el estado de la salida de CLC.

bit 5	<b>LCxOUT:</b> Configurable Logic Cell Data Output bit Read-only: logic cell output data, after LCxPOL; sampled from lcx_out wire.
-------	---

---

El bit LCINTP de CLC permite la interrupción de borde ascendente en el CLC. Cuando está habilitado (establecido en 1), el CLC activará una interrupción cuando la salida del CLC aumente de un estado bajo a alto.

bit 4	<b>LCxINTP:</b> Configurable Logic Cell Positive Edge Going Interrupt Enable bit 1 = LCxIF will be set when a rising edge occurs on lcx_out 0 = LCxIF will not be set
-------	---

---

El bit LCINTN de CLC permite la interrupción del borde descendente en el CLC. Cuando está habilitado (establecido en un 1), el CLC activará una interrupción cuando la salida del CLC caiga de un estado alto a uno bajo.

bit 3	<b>LCxINTN:</b> Configurable Logic Cell Negative Edge Going Interrupt Enable bit 1 = LCxIF will be set when a falling edge occurs on lcx_out 0 = LCxIF will not be set
-------	--

---

Los bits LCMODE de CLC seleccionan la función lógica de CLC utilizando tres bits (0-2).

bit 2-0	<b>LCxMODE&lt;2:0&gt;:</b> Configurable Logic Cell Functional Mode bits 111 = Cell is 1-input transparent latch with S and R 110 = Cell is J-K Flip-Flop with R 101 = Cell is 2-input D Flip-Flop with R 100 = Cell is 1-input D Flip-Flop with S and R 011 = Cell is S-R latch 010 = Cell is 4-input AND 001 = Cell is OR-XOR 000 = Cell is AND-OR
---------	---

# Registros CLCxSELn

Los registros CLCxSEL, contenidos en la [celda lógica configurable \(CLC\)](#), controlan qué entradas se utilizan con la CLC.

## Fuentes de entrada de CLC

El CLC tendrá múltiples entradas para seleccionar y cada una tendrá un código de 3 bits asociado, como se muestra en la tabla a continuación. Cada entrada se puede conectar a una de las dos puertas de datos de entrada a través de un multiplexor controlado por los registros CLCxSEL0 y CLCxSEL1.

Data Input	Icx <sub>d1</sub> D1S	Icx <sub>d2</sub> D2S	Icx <sub>d3</sub> D3S	Icx <sub>d4</sub> D4S	CLC 1	CLC 2
CLCxIN[0]	000	—	—	100	CLC1IN0	CLC2IN0
CLCxIN[1]	001	—	—	101	CLC1IN1	CLC2IN1
CLCxIN[2]	010	—	—	110	Reserved	Reserved
CLCxIN[3]	011	—	—	111	Reserved	Reserved
CLCxIN[4]	100	000	—	—	Fosc	Fosc
CLCxIN[5]	101	001	—	—	TMR0IF	TMR0IF
CLCxIN[6]	110	010	—	—	TMR1IF	TMR1IF
CLCxIN[7]	111	011	—	—	TMR2 = PR2	TMR2 = PR2
CLCxIN[8]	—	100	000	—	Icx1_out	Icx1_out
CLCxIN[9]	—	101	001	—	Icx2_out	Icx2_out
CLCxIN[10]	—	110	010	—	Icx3_out	Icx3_out
CLCxIN[11]	—	111	011	—	Icx4_out	Icx4_out
CLCxIN[12]	—	—	100	000	NCO1OUT	LFINTOSC
CLCxIN[13]	—	—	101	001	HFINTOSC	ADCFRC
CLCxIN[14]	—	—	110	010	PWM3OUT	PWM1OUT
CLCxIN[15]	—	—	111	011	PWM4OUT	PWM2OUT

De la [hoja de datos de PIC16F1507](#).

Las selecciones de entrada se controlan mediante los registros CLCxSEL0 y CLCxSEL1 estableciendo el código de entrada de 3 bits.

- El registro CLCxSEL0 controla las puertas de entrada de datos 1 y 2. Los bits 0-2 controlan la entrada 1 y los bits 4-6 controlan la entrada 2.

## CLCxSEL0: MULTIPLEXER DATA 1 Y 2 SELECT REGISTER

U-0	R/W-x/u	R/W-x/u	R/W-x/u	U-0	R/W-x/u	R/W-x/u	R/W-x/u
—		LCxD2S<2:0>	—		LCxD1S<2:0>		
bit 7							bit 0

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

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

bit 6-4      **LCxD2S<2:0>:** Input Data 2 Selection Control bits<sup>(1)</sup>

111 = CLCxIN[11] is selected for lcdn2

110 = CLCxIN[10] is selected for lcdn2

101 = CLCxIN[9] is selected for lcdn2

100 = CLCxIN[8] is selected for lcdn2

011 = CLCxIN[7] is selected for lcdn2

010 = CLCxIN[6] is selected for lcdn2

001 = CLCxIN[5] is selected for lcdn2

000 = CLCxIN[4] is selected for lcdn2

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

bit 2-0      **LCxD1S<2:0>:** Input Data 1 Selection Control bits<sup>(1)</sup>

111 = CLCxIN[7] is selected for lcdn1

110 = CLCxIN[6] is selected for lcdn1

101 = CLCxIN[5] is selected for lcdn1

100 = CLCxIN[4] is selected for lcdn1

011 = CLCxIN[3] is selected for lcdn1

010 = CLCxIN[2] is selected for lcdn1

001 = CLCxIN[1] is selected for lcdn1

000 = CLCxIN[0] is selected for lcdn1

De la [hoja de datos de PIC16F1507](#).

- El registro CLCxSEL1 controla las puertas de entrada de datos 3 y 4. Los bits 0-2 controlan la entrada 3 y los bits 4-6 controlan la entrada 4.

## CLCxSEL1: MULTIPLEXER DATA 3 Y 4 SELECT REGISTER

U-0	R/W-x/u	R/W-x/u	R/W-x/u	U-0	R/W-x/u	R/W-x/u	R/W-x/u
—	LCxD4S<2:0>			—	LCxD3S<2:0>		
bit 7	bit 0						

**Legend:**

R = Readable bit

u = Bit is unchanged

'1' = Bit is set

W = Writable bit

x = Bit is unknown

'0' = Bit is cleared

U = Unimplemented bit, read as '0'

-n/n = Value at POR and BOR/Value at all other Resets

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

bit 6-4       **LCxD4S<2:0>:** Input Data 4 Selection Control bits<sup>(1)</sup>

111 = CLCxIN[3] is selected for lcxd4

110 = CLCxIN[2] is selected for lcxd4

101 = CLCxIN[1] is selected for lcxd4

100 = CLCxIN[0] is selected for lcxd4

011 = CLCxIN[15] is selected for lcxd4

010 = CLCxIN[14] is selected for lcxd4

001 = CLCxIN[13] is selected for lcxd4

000 = CLCxIN[12] is selected for lcxd4

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

bit 2-0       **LCxD3S<2:0>:** Input Data 3 Selection Control bits<sup>(1)</sup>

111 = CLCxIN[15] is selected for lcxd3

110 = CLCxIN[14] is selected for lcxd3

101 = CLCxIN[13] is selected for lcxd3

100 = CLCxIN[12] is selected for lcxd3

011 = CLCxIN[11] is selected for lcxd3

010 = CLCxIN[10] is selected for lcxd3

001 = CLCxIN[9] is selected for lcxd3

000 = CLCxIN[8] is selected for lcxd3

De la [hoja de datos de PIC16F1507](#).

# Registros CLCxGLSn

Los registros CLCxGLSn, contenidos en la [celda lógica configurable \(CLC\)](#), controlan la polaridad de las entradas CLC seleccionadas.

---

## Gating de datos de CLC

Las salidas de los multiplexores de entrada se dirigen a la entrada de la función lógica deseada a través de la etapa de cierre de datos. Cada puerta de datos puede dirigir cualquier combinación de cuatro entradas seleccionadas. La compuerta se puede configurar para dirigir cada señal de entrada como datos invertidos o no invertidos. Las señales dirigidas se ORed juntas en cada puerta. La salida de cada puerta también se puede invertir antes de pasar a la etapa de función lógica, pero eso está controlado por el registro [CLCxPOL](#).

La sección Data Gating está controlada por uno de los cuatro registros. Cada puerta tiene un registro separado. Cada entrada tiene un bit "N" y un bit "P". La configuración del bit "N" invierte la entrada y la configuración del bit "P" hace que no se invierta. Si ninguno de los dos está configurado, la puerta tendrá un nivel lógico constante alto o bajo dependiendo de la configuración de polaridad de salida en:

- CLCxGLS0
- CLCxGLS1
- CLCxGLS2
- CLCxGLS3

CLCxGLS0 se muestra en la Figura 1

| R/W-x/u  |
|----------|----------|----------|----------|----------|----------|----------|----------|
| LCxG1D4T | LCxG1D4N | LCxG1D3T | LCxG1D3N | LCxG1D2T | LCxG1D2N | LCxG1D1T | LCxG1D1N |
| bit 7    | bit 0    |          |          |          |          |          |          |

**Legend:**

R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7	<b>LCxG1D4T:</b> Gate 1 Data 4 True (non-inverted) bit 1 = lcxd4T is gated into lcxg1 0 = lcxd4T is not gated into lcxg1
bit 6	<b>LCxG1D4N:</b> Gate 1 Data 4 Negated (inverted) bit 1 = lcxd4N is gated into lcxg1 0 = lcxd4N is not gated into lcxg1
bit 5	<b>LCxG1D3T:</b> Gate 1 Data 3 True (non-inverted) bit 1 = lcxd3T is gated into lcxg1 0 = lcxd3T is not gated into lcxg1
bit 4	<b>LCxG1D3N:</b> Gate 1 Data 3 Negated (inverted) bit 1 = lcxd3N is gated into lcxg1 0 = lcxd3N is not gated into lcxg1
bit 3	<b>LCxG1D2T:</b> Gate 1 Data 2 True (non-inverted) bit 1 = lcxd2T is gated into lcxg1 0 = lcxd2T is not gated into lcxg1
bit 2	<b>LCxG1D2N:</b> Gate 1 Data 2 Negated (inverted) bit 1 = lcxd2N is gated into lcxg1 0 = lcxd2N is not gated into lcxg1
bit 1	<b>LCxG1D1T:</b> Gate 1 Data 1 True (non-inverted) bit 1 = lcxd1T is gated into lcxg1 0 = lcxd1T is not gated into lcxg1
bit 0	<b>LCxG1D1N:</b> Gate 1 Data 1 Negated (inverted) bit 1 = lcxd1N is gated into lcxg1 0 = lcxd1N is not gated into lcxg1

Figura 1

De la [hoja de datos de PIC16F1507](#).

## Creación de varias puertas

Cada puerta de datos es, en esencia, una puerta de entrada de 1 a 4 Y / NAND / OR / NOR dependiendo de la configuración de inversión / no inversión. Cuando cada entrada se invierte y la salida se invierte, la puerta es un NOR de todas las entradas de datos habilitadas. Cuando las entradas y salidas no están invertidas, la puerta es un OR de todas las entradas habilitadas.

La tabla que se muestra en la Figura 2 resume la lógica básica que se puede obtener en una puerta mediante el uso de los bits de selección de lógica de puerta.

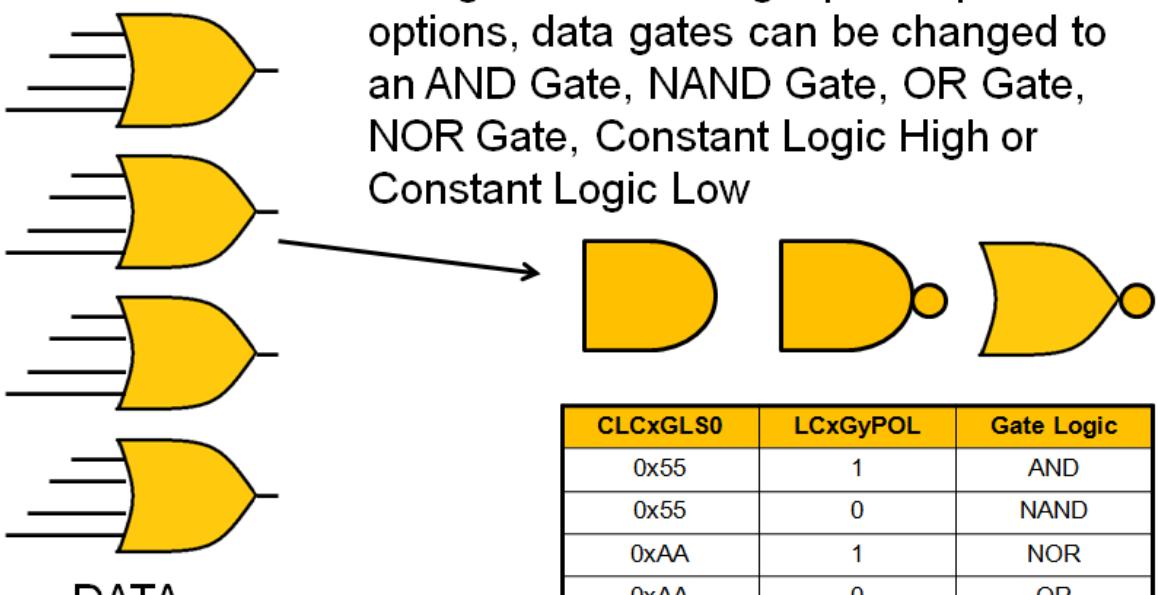


Figura 2

La tabla muestra la lógica de cuatro variables de entrada. Sin embargo, cada puerta se puede configurar para usar menos de cuatro. Si no se selecciona ninguna entrada, la salida será 0 o 1, dependiendo del bit de polaridad de salida de la puerta. La polaridad de salida está controlada por el registro.

## CLC1POL

LC1POL	-	-	-	LC1G4POL	LC1G3POL	LC1G2POL	LC1G1POL 0
--------	---	---	---	----------	----------	----------	---------------

## CLC1GLS0

LC1G1D4T 0	LC1G1D4N 1	LC1G1D3T 0	LC1G1D3N 1	LC1G1D2T 0	LC1G1D2N 1	LC1G1D1T 0	LC1G1D1N 1
Input 4		Input 3		Input 2		Input 1	

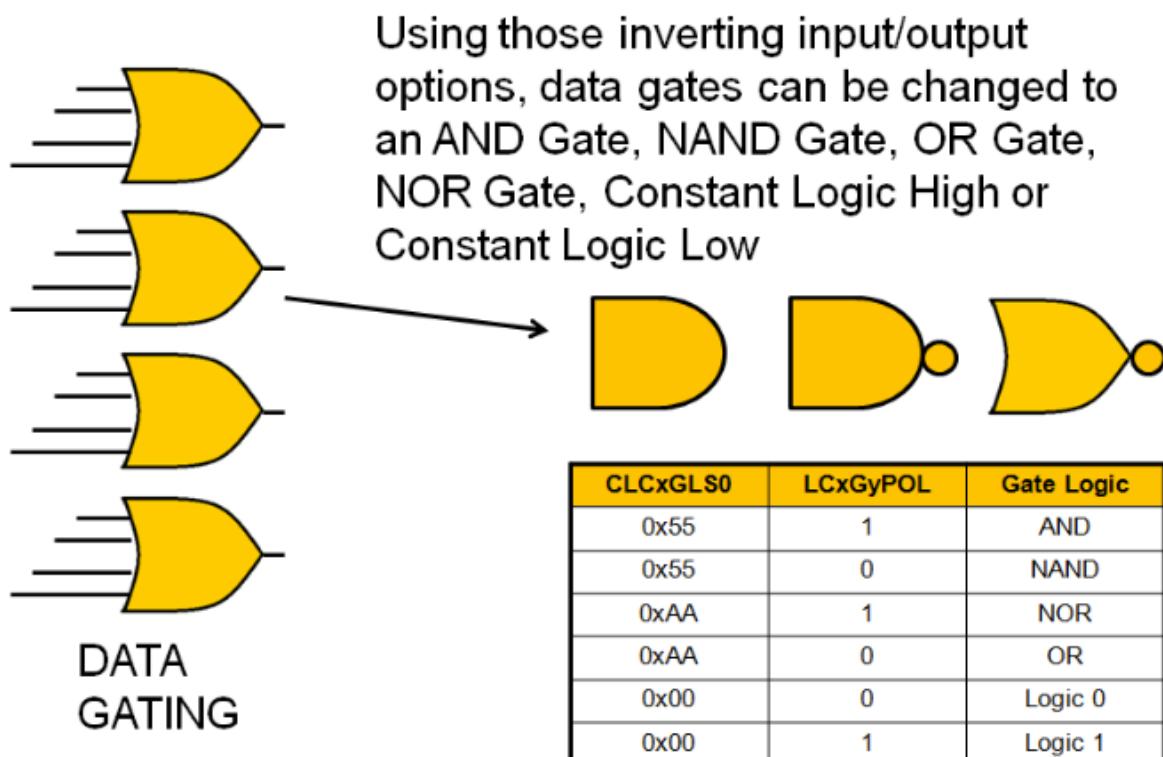
Figura 3

# Registro CLCxPOL

El registro CLCxPOL, contenido en la [celda lógica configurable \(CLC\)](#), controla la polaridad de las salidas de cierre de datos y también la polaridad de la salida CLC.

## Salidas de puerta de datos CLC

La siguiente imagen resume la lógica básica que se puede obtener en una puerta mediante el uso de los bits de selección de lógica de puerta.



La tabla muestra la lógica de cuatro variables de entrada, pero cada puerta se puede configurar para usar menos de cuatro. Si no se selecciona ninguna entrada, la salida será 0 o 1, dependiendo del bit de polaridad de salida de la puerta. La polaridad de salida está controlada por el registro CLCxPOL.

### CLC1POL

LC1POL	-	-	-	LC1G4POL	LC1G3POL	LC1G2POL	LC1G1POL 0
--------	---	---	---	----------	----------	----------	---------------

### CLC1GLS0

LC1G1D4T 0	LC1G1D4N 1	LC1G1D3T 0	LC1G1D3N 1	LC1G1D2T 0	LC1G1D2N 1	LC1G1D1T 0	LC1G1D1N 1
Input 4		Input 3		Input 2		Input 1	

## Control de salida CLC

La salida de todo el CLC también se puede invertir en el registro CLCxPOL estableciendo el séptimo bit en el registro. Borrar el bit hará que la salida CLC no esté invertida.

**CLC1POL**

LC1POL	-	-	-	LC1G4POL	LC1G3POL	LC1G2POL	LC1G1POL	
1	-		Invertir		la			salida

CLC 0 - No invertir la salida CLC

# Generador de forma de onda complementaria (CWG)

El generador de forma de onda complementaria (CWG) produce una forma de onda complementaria con el retardo de banda muerta de una selección de fuentes de entrada.

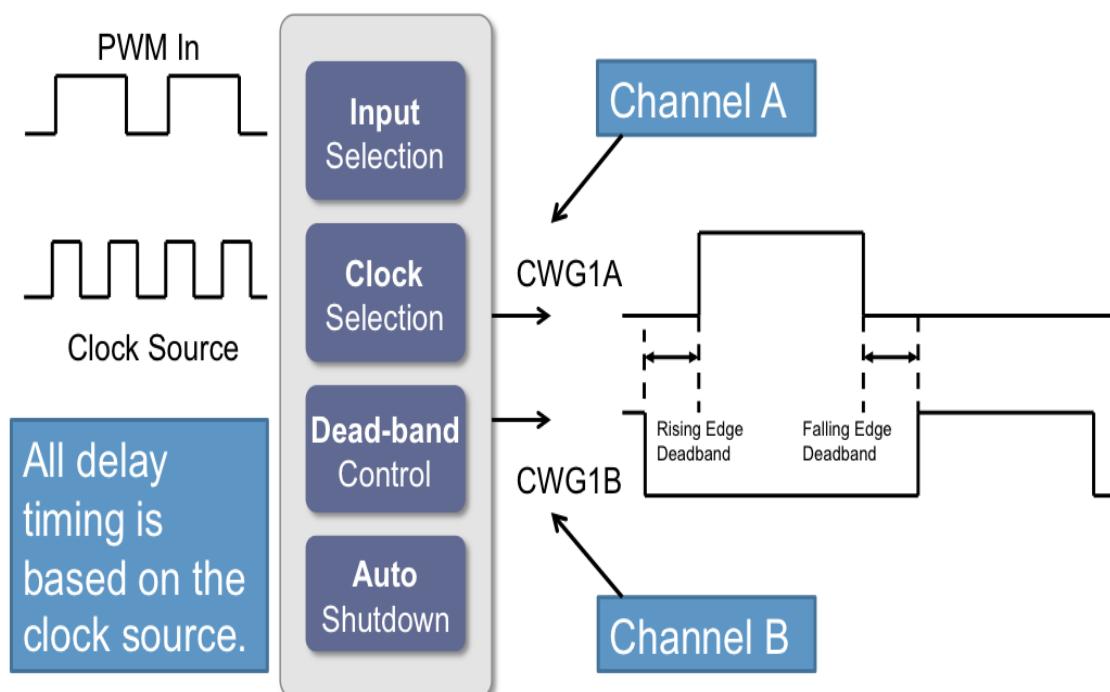
El módulo CWG tiene las siguientes características:

- Control de fuente de reloj de banda muerta seleccionable
- Fuentes de entrada seleccionables
- Control de habilitación de salida
- Control de polaridad de salida
- Control de banda muerta con contadores independientes de banda muerta de borde ascendente y descendente de 6 bits
- Control de apagado automático con:
  - Fuentes de apagado seleccionables
  - Habilitación de reinicio automático
  - Control de anulación del pin de apagado automático

El CWG genera una forma de onda complementaria de dos salidas a partir de una de varias fuentes de entrada seleccionables. La transición de apagado a encendido de cada salida puede retrasarse de la transición de encendido a apagado de la otra salida, creando así un retraso de tiempo inmediatamente donde no se controla ninguna salida. Esto se conoce como tiempo muerto o banda muerta y se cubre en la sección a continuación titulada "Control de banda muerta".

Puede ser necesario protegerse contra la posibilidad de fallas en el circuito. En este caso, la unidad activa puede terminarse antes de que la condición de falla cause daños. Esto se conoce como apagado automático y se cubre en la sección a continuación titulada "Control de apagado automático".

Complimentary Waveform Generator creates a set of complementary waveforms from one input source.



El GTC requiere que se establezcan cinco secciones:

- Entrada
- Reloj
- Banda muerta
- Apagado
- Control de salida

## CWG Video Tutorial

Este video presenta el Generador de Forma de Onda Complementaria (CWG) para dispositivos MCU de 8 bits de Microchip y muestra cómo usarlo.

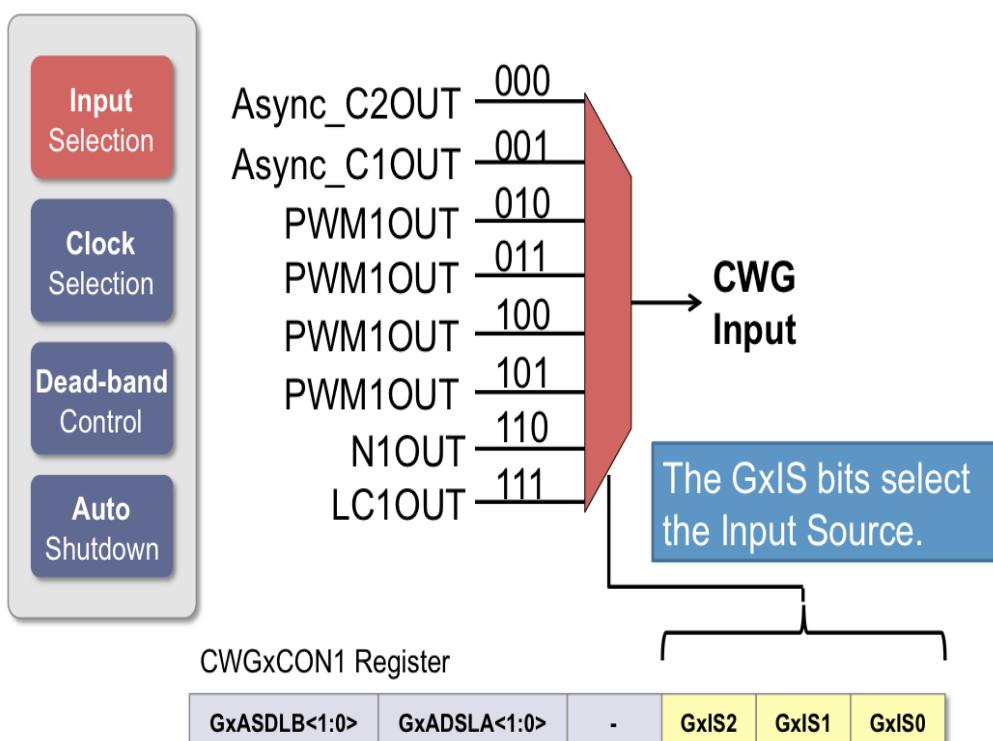
## Fuente de entrada

El GTC ofrece varias fuentes de entrada para generar la forma de onda complementaria. Esto puede variar de un dispositivo a otro. La siguiente lista es del dispositivo PIC16F1507. La lista incluye:

- PWM1 (Salida modulada por ancho de pulso 1)
- PWM2 (Salida modulada por ancho de pulso 2)
- PWM3 (Salida modulada por ancho de pulso 3)
- PWM4 (Salida modulada por ancho de pulso 4)
- N1OUT (Salida de oscilador controlada numéricamente)
- LC1OUT (Salida de celda lógica configurable)

La fuente de entrada se selecciona utilizando los bits GxIS<2:0> en el registro CWGxCON1.

Algunos dispositivos también incluyen la salida de un módulo de comparación como entrada al CWG. Es mejor consultar la hoja de datos del dispositivo que está utilizando para obtener la lista actualizada de opciones de entrada seleccionadas.

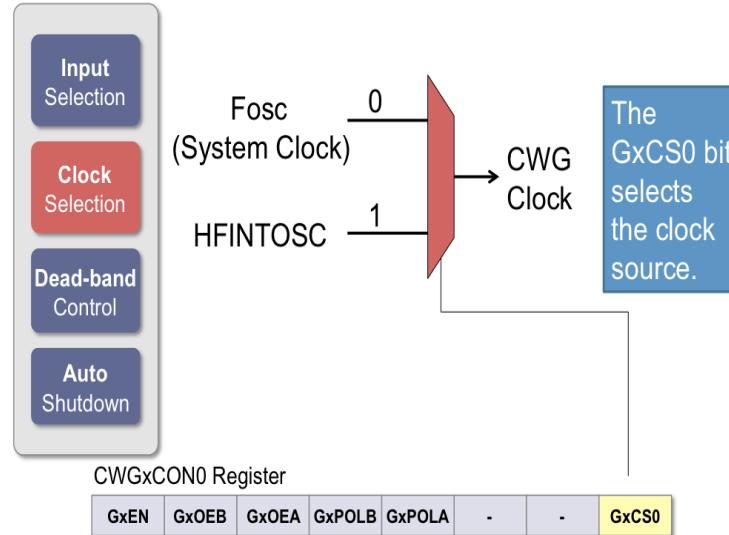


# Fuente del reloj

El módulo CWG permite seleccionar una de las dos fuentes de reloj:

- $F_{osc}$  (reloj del sistema)
- HFINTOSC (solo 16 MHz)

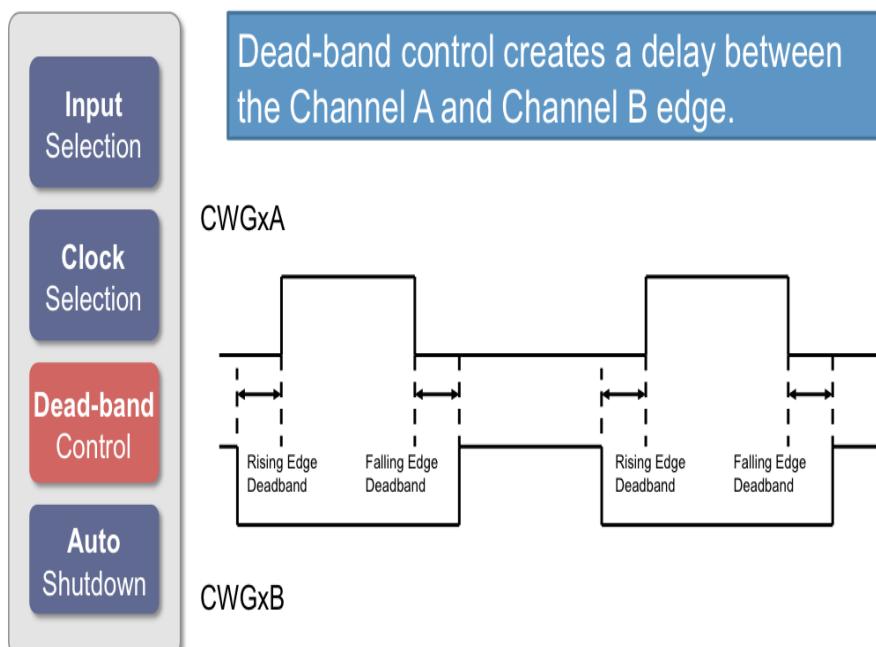
Las fuentes de reloj se seleccionan utilizando el bit GxCS0 del registro CWGxCON0.



# Control de banda muerta

El control de banda muerta proporciona señales de salida no superpuestas, para evitar la corriente de disparo en los interruptores de potencia. El CWG contiene dos contadores de banda muerta de 6 bits (registros CWGxDBR y CWGxDBF). Un contador de banda muerta se utiliza para el borde ascendente del control de fuente de entrada, mientras que el otro se utiliza para el borde descendente del control de fuente de entrada.

La banda muerta se cronometra contando los períodos de reloj CWG desde cero hasta el valor en los registros de contador de banda muerta ascendente o descendente.

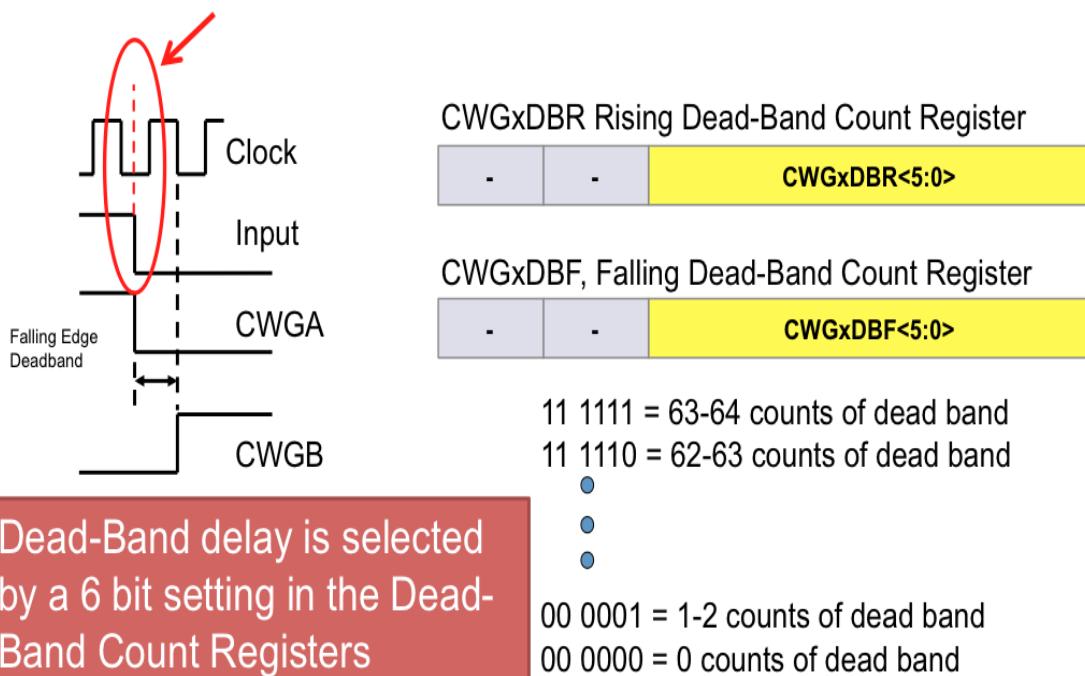


## Control de borde ascendente

La banda muerta de borde ascendente retrasa el encendido de la salida CWGxA desde que se apaga la salida CWGxB. El tiempo de banda muerta del borde ascendente comienza cuando el borde ascendente de la señal de la fuente de entrada se hace realidad. Cuando esto sucede, la salida CWGxB se apaga inmediatamente y comienza el tiempo de retardo de banda muerta del borde ascendente. Cuando se alcanza el tiempo de retardo de banda muerta del borde ascendente, se activa la salida CWGxA.

El registro CWGxDBR establece la duración del intervalo de banda muerta en el borde ascendente de la señal de fuente de entrada. Esta duración es de 0 a 64 recuentos de banda muerta. La banda muerta siempre se cuenta fuera del borde en la señal de la fuente de entrada. Un recuento de cero (0), indica que no hay banda muerta presente. Si la señal de la fuente de entrada no está presente durante el tiempo suficiente para que se complete el recuento, no se verá ninguna salida en la salida respectiva.

A change in state of the input signal may not occur in sync with the clock signal. This can cause longer dead band delays.

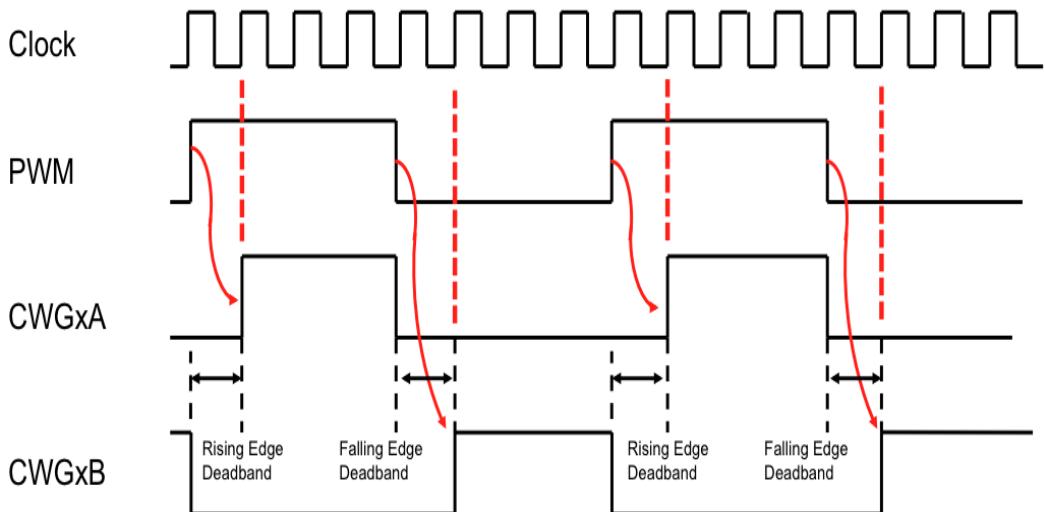


## Control de borde descendente

La banda muerta del borde descendente retrasa el encendido de la salida CWGxB desde que se apaga la salida CWGxA. El tiempo de banda muerta del borde descendente comienza cuando el borde descendente de la fuente de entrada se hace realidad. Cuando esto sucede, la salida CWGxA se apaga inmediatamente y comienza el tiempo de retardo de banda muerta del borde descendente. Cuando se alcanza el tiempo de retardo de banda muerta del borde descendente, se activa la salida CWGxB.

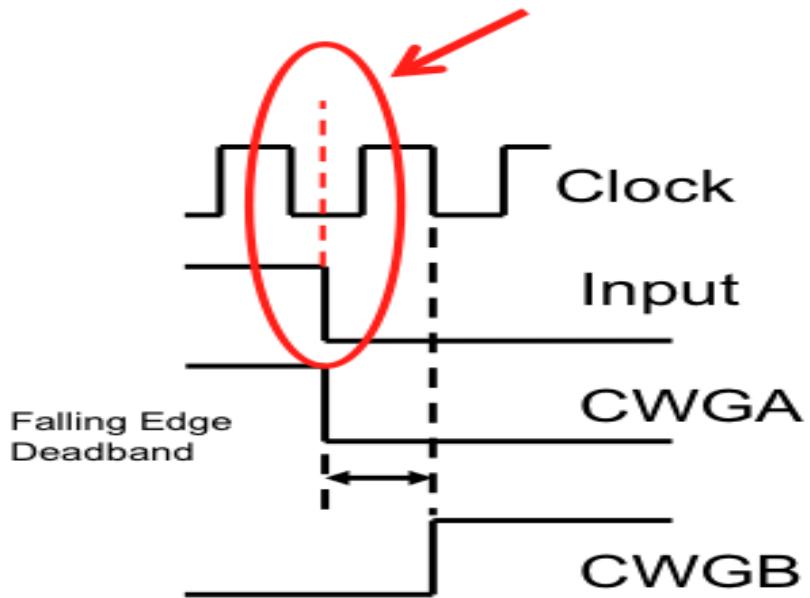
El registro CWGxDBF establece la duración del intervalo de banda muerta en el borde descendente de la señal de fuente de entrada. Esta duración es de 0 a 64 recuentos de banda muerta. La banda muerta siempre se cuenta fuera del borde en la señal de la fuente de entrada. Un recuento de cero (0), indica que no hay banda muerta presente. Si la señal de la fuente de entrada no está presente durante el tiempo suficiente para que se complete el recuento, no se verá ninguna salida en la salida respectiva.

Using a one clock pulse delay for rising and falling edge Dead-Band control, the PWM signal would produce two waveforms similar to what is shown here.



## Incertidumbre de banda muerta

Cuando los bordes ascendentes y descendentes de la fuente de entrada activan los contadores de banda muerta, la entrada puede ser asincrónica a la entrada de reloj. Esto creará cierta incertidumbre en el retraso del tiempo muerto. La incertidumbre máxima es igual a un período de reloj CWG.



## Control de apagado automático

El apagado automático es un método para anular inmediatamente los niveles de salida CWG con configuraciones específicas que permiten el apagado seguro del circuito. El estado de apagado se puede borrar automáticamente o mantenerse hasta que el software lo borre.

El estado de apagado se puede introducir mediante cualquiera de los dos métodos siguientes:

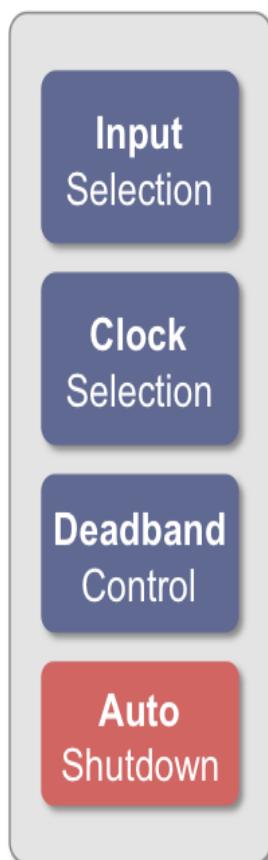
- Software generado
- Entrada externa

## Software generado

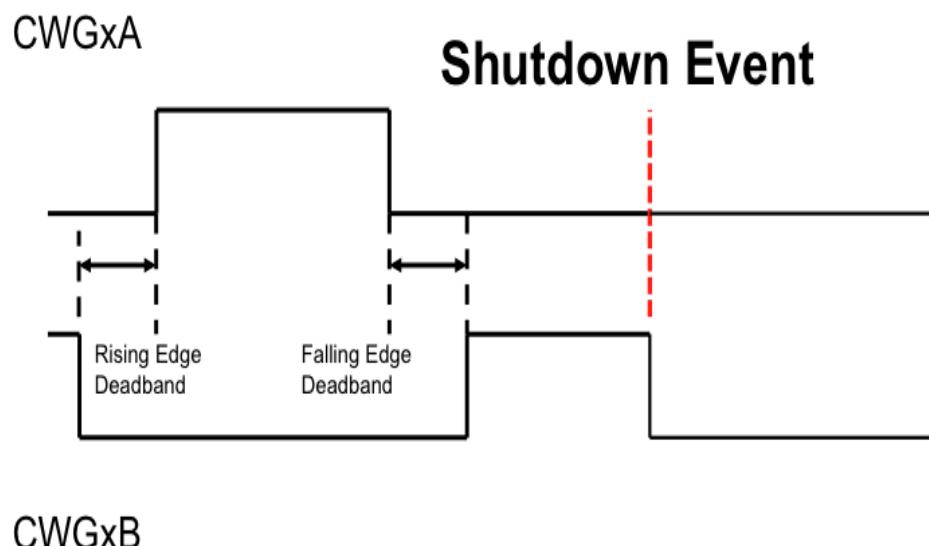
Establecer el bit GxASE del registro CWGxCON2 forzará al CWG al estado de apagado. Cuando el reinicio automático está deshabilitado, el estado de apagado persistirá mientras se establezca el bit GxASE. Cuando el reinicio automático está habilitado, el bit GxASE se borrará automáticamente y reanudará la operación en el siguiente evento de borde ascendente.

## Entrada externa

Las señales de apagado externas proporcionan la forma más rápida de suspender de forma segura la operación de CWG en caso de una condición de falla. Cuando cualquiera de las señales de apagado seleccionadas se activa, las salidas CWG pasarán inmediatamente a los niveles de anulación seleccionados sin demora en el software. Se puede seleccionar cualquier combinación de dos señales de apagado para causar una condición de apagado. Las señales de apagado ofrecidas pueden variar con el dispositivo que se está utilizando, pero esas fuentes de apagado son seleccionadas por los bits GxASDS0 y GxASDS1 del registro CWGxCON2.



Auto Shutdown will disable both of the CWG outputs through an external trigger or internal software request .



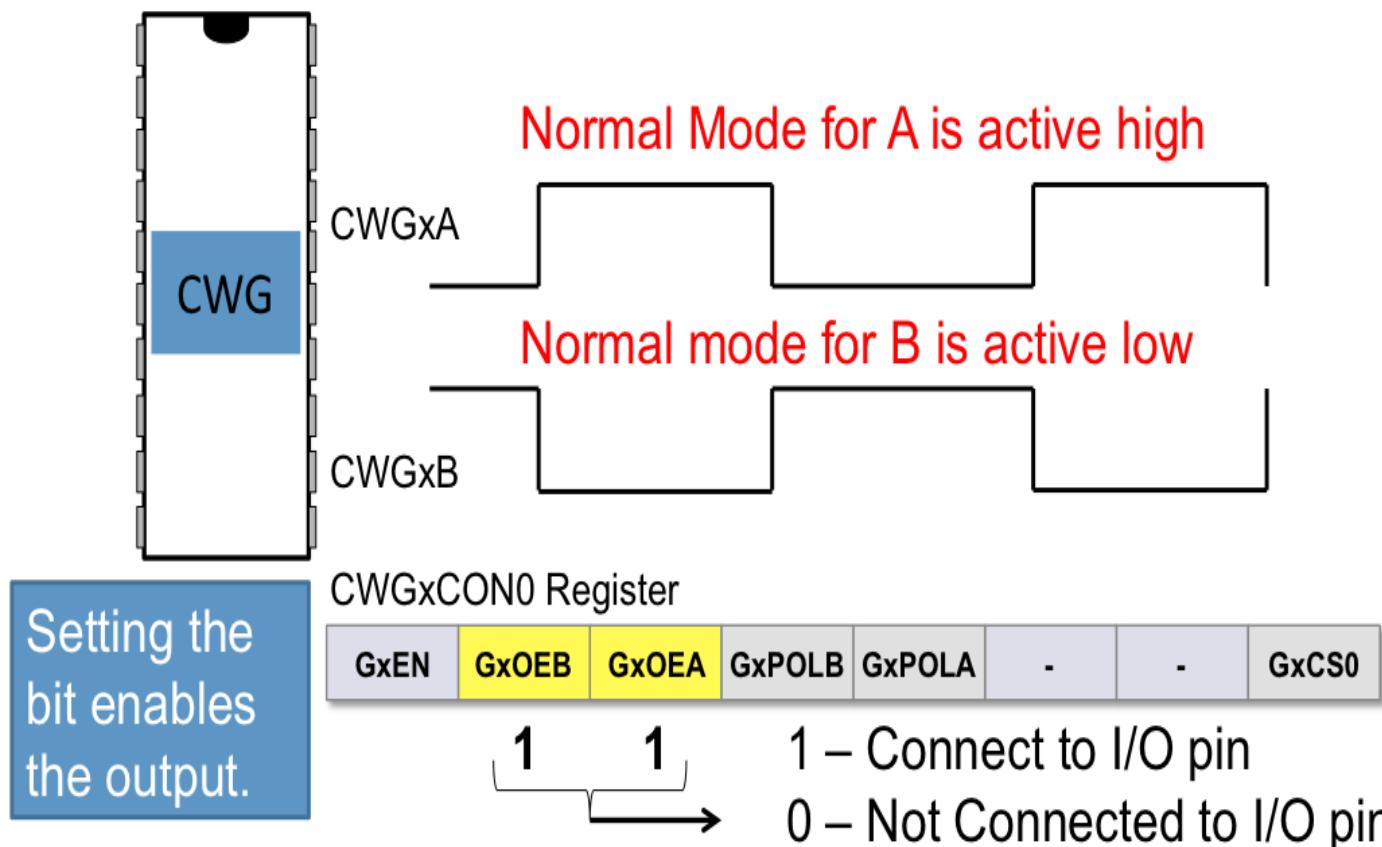
# Control de salida

## Habilitar pin de salida

Cada pin de salida CWG tiene un control de habilitación de pin de salida individual. Las habilitaciones de salida se seleccionan con los bits GxOEA y GxOEB del registro CWGxCON0. Cuando se borra una habilitación de pin de salida, el CWG no tiene conexión con el pin de salida. Cuando se establece la habilitación de salida, el valor de anulación o la forma de onda PWM activa se aplica al pin según la selección de prioridad de puerto interno.

La función CWG se puede desactivar completamente borrando el pin GxEN en el registro CWGxCON0.

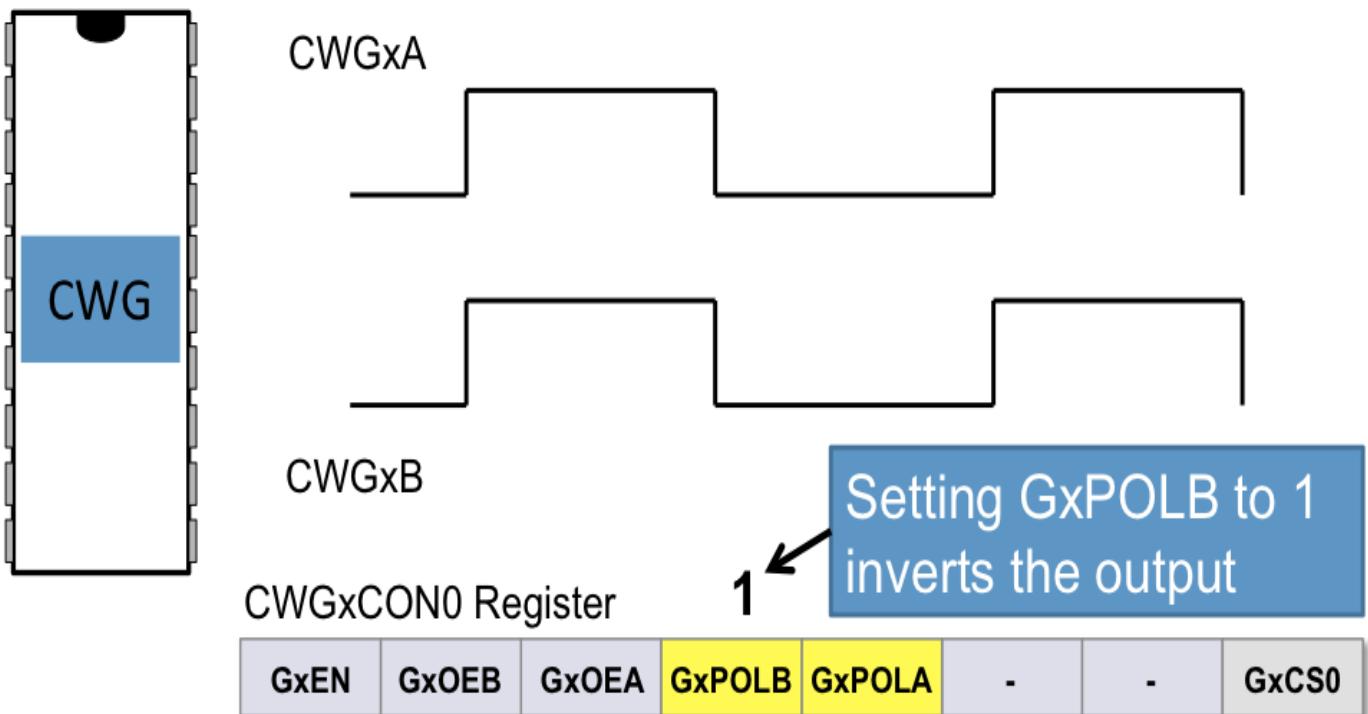
The CWGA and CWGB outputs can be connected to the I/O pin through the CWGCON0 Register setting.



## Control de polaridad

La polaridad de cada salida CWG se puede seleccionar de forma independiente. Cuando se establece el bit de polaridad de salida, la salida correspondiente es alta activa. Al borrar el bit de polaridad de salida, se configura la salida correspondiente como activa baja. Sin embargo, la polaridad no afecta a los niveles de anulación. La polaridad de salida se selecciona con los bits GxPOLA y GxPOLB del registro CWGxCON0.

Polarity of the outputs can be set to invert signal. This would allow Channel A and Channel B to output the same exact signal.



## Funcionamiento durante el modo de suspensión

El módulo CWG funciona independientemente del reloj del sistema y continuará ejecutándose durante la suspensión, siempre que el reloj y las fuentes de entrada seleccionadas permanezcan activos. El oscilador interno de alta frecuencia (HFINTOSC) permanece activo durante la suspensión, siempre que el módulo CWG esté habilitado, la fuente de entrada esté activa y el HFINTOSC se seleccione como fuente de reloj, independientemente de la fuente de reloj del sistema seleccionada.

En otras palabras: si el HFINTOSC se selecciona simultáneamente como el reloj del sistema y la fuente del reloj CWG, entonces cuando el CWG esté habilitado y la fuente de entrada esté activa, la CPU permanecerá inactiva durante la suspensión, pero el CWG continuará funcionando y el HFINTOSC permanecerá activo. Esto tendrá un efecto directo en la corriente del modo de suspensión.

## Ejemplo de GTC

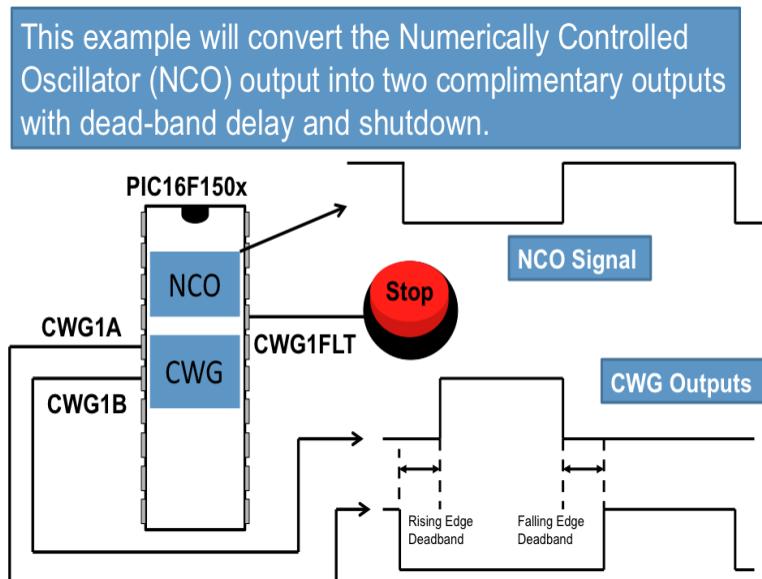
A veces es útil repasar un ejemplo. Visite este [ejemplo de CWG](#) para ver instrucciones paso a paso para el módulo CWG.

# Ejemplo de generador de forma de onda complementaria

Esta página es un ejemplo paso a paso que utiliza el Generador Complementario de Forma de Onda (CWG).

En este ejemplo se utiliza la señal de salida del oscilador controlado numéricamente (NCO) (que funciona al 50 % del ciclo de trabajo) como entrada y el oscilador interno de alta frecuencia (HFINTOSC) como reloj.

Este ejemplo también produce dos salidas complementarias y utiliza un conmutador externo como control de apagado, como se muestra en el diagrama de bloques siguiente.

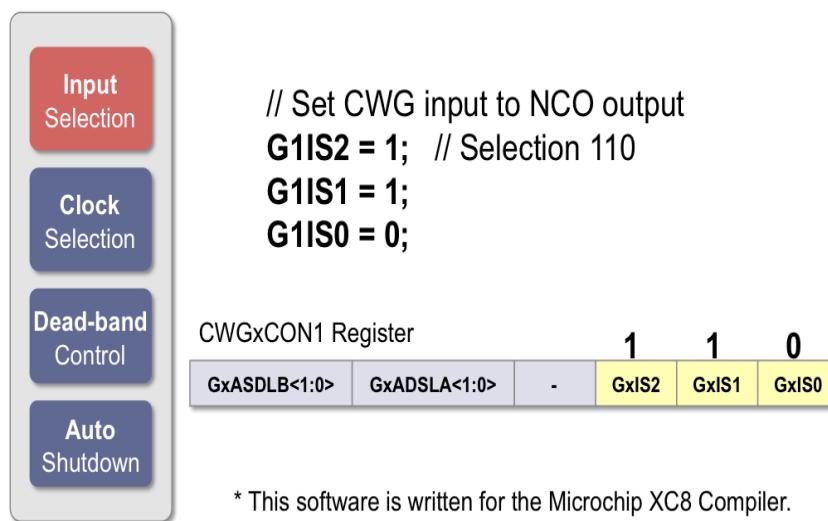


## Configuración de ejemplo de CWG

1

El **suboficial** se selecciona como entrada para el módulo CWG.

The NCO output is selected as the input signal for the CWG module in the CWGCON1 register.



2

El oscilador interno de alta frecuencia se selecciona como fuente de reloj.

The clock source is selected in the CWGCON0 register.



//Set clock source to HFINTOSC  
**G1CS0 = 1;**

CWGxCON0 Register 1

GxEN	GxOEB	GxOEA	GxPOLB	GxPOLA	-	-	<b>GxCS0</b>
------	-------	-------	--------	--------	---	---	--------------

3

Los retrasos de banda muerta se establecen en **recuentos de reloj de 63-64**.

The Dead-Band delays are setup for a 63-64 clock cycle count for both rising and falling edge.



*/\* Initialize Dead-Band Delays\*/*

**CWG1DBR = 0x3F; // 63-64 counts Rising**  
**CWG1DBF = 0x3F; // 63-64 counts Falling**

CWGxDBR Rising Dead-Band Count Register

-	-	<b>CWGxDBR&lt;5:0&gt;</b>
---	---	---------------------------

CWGxDBF, Falling Dead-Band Count Register

-	-	<b>CWGxDBF&lt;5:0&gt;</b>
---	---	---------------------------

11 1111 = 63-64 counts of dead band

4

El pin **CWGFLT1** está habilitado como una señal de apagado que está conectada a un interruptor momentáneo normalmente abierto. Cuando se presiona el interruptor, aparece una señal baja en el pin CWGFLT1 y activa el apagado.

The Shutdown control is set to the CWG1FLT I/O pin.



//CWG1FLT pulled low causes shutdown  
**G1ASDSFLT = 1;**  
//CLC output no effect on shutdown  
**G1ASDCLC2 = 0;**

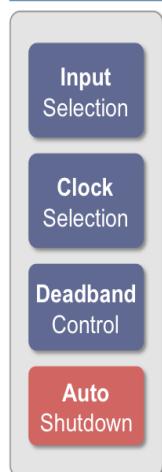
CWGxCON2 Register*					
GxASE	GxARSEN	-	-	-	GxASDFLT GxASDCLC2

\* PIC16F1507

5

El reinicio automático está habilitado y el bit de apagado se borra para iniciarse en modo de ejecución.

Auto restart is enabled.  
The shutdown bit is cleared for proper operation.



//Auto Restart enabled  
**G1ARSEN = 1;**  
//Clear shutdown mode  
**G1ASE = 0;**

CWGxCON2 Register*					
GxASE	GxARSEN	-	-	-	GxASDFLT GxASDCLC2

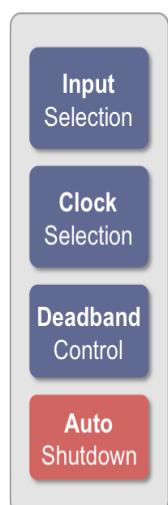
0      1

\* PIC16F1507

6

Ambas salidas están configuradas para conducir a su respectivo modo inactivo cuando se produce la señal de apagado.

The Shutdown state of the outputs are set to inactive.



// CWG1B driven inactive during shutdown  
**G1ASDLB0 = 0; // Value 00**  
**G1ASDLB1 = 0;**  
// CWG1A driven inactive during shutdown  
**G1ASDLA0 = 0; // Value 00**  
**G1ASDLA1 = 0;**

CWGxCON1 Register

GxASDLB<1:0>	GxASDLA<1:0>	-	GxIS2	GxIS1	GxIS0
0 0	0 0				

11 = Drive pin high  
10 = Drive pin low  
01 = Drive pin Tri-state  
00 = Drive to inactive state.

7

Las **salidas CWG** están conectadas a los pines de E/S y la polaridad de salida se establece en modo normal.

The CWG output signals are connected to the CWG I/O pins through the CWGCON0 register.

```
G1OE A = 1; //Output signal on CWG1A pin  
G1OE B = 1; //Output signal on CWG1B pin  
G1POL A = 0; //Output is normal polarity  
G1POL B = 0; //Output is normal polarity
```

CWGxCON0 Register

GxEN	GxOEB	GxOEA	GxPOLB	GxPOLA	-	-	GxCs0
1	1	0	0				

8

A continuación, se habilita el **CWG**. Como referencia, también se muestra el código de configuración del suboficial que produce la señal de entrada.

The CWG module is then enabled as the last step.

```
G1EN = 1; //Enable CWG module
```

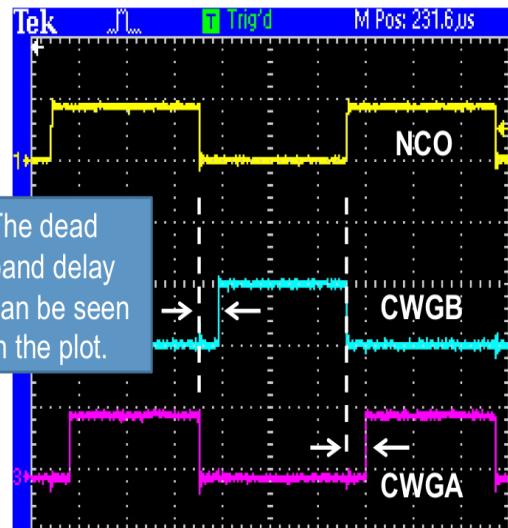
The NCO module is setup to produce a 50% duty cycle square wave input to the CWG.

```
NCO1INCH = 8; // load NCO increment high register  
NCO1INCL = 65;  
NCO1CLK = 0b00000000; // Select HF internal OSC = 16 Mhz  
NCO1CON = 0b11010000; // Enable NCO, Enable output,active hi, 50% dc mode
```

## Operación de ejemplo de CWG

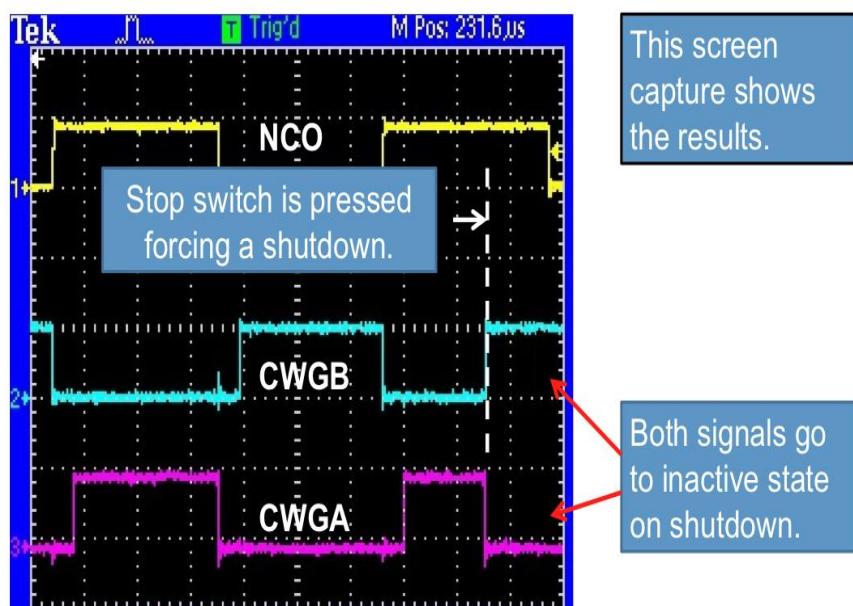
El ejemplo de CWG se está ejecutando y la captura de pantalla a continuación muestra los resultados.

La señal de NCO se puede ver en la parte superior de la captura de pantalla junto con las señales CWGA y CWGB. También puede ver los retrasos de banda muerta que se activan por los bordes de la señal de entrada del suboficial.



This screen capture shows the results.

El CWG se ve obligado a entrar en modo de apagado. Cuando se presiona el interruptor momentáneo, las salidas CWG se conducen a su estado inactivo mostrando que se ha producido el apagado.



This screen capture shows the results.

Both signals go to inactive state on shutdown.

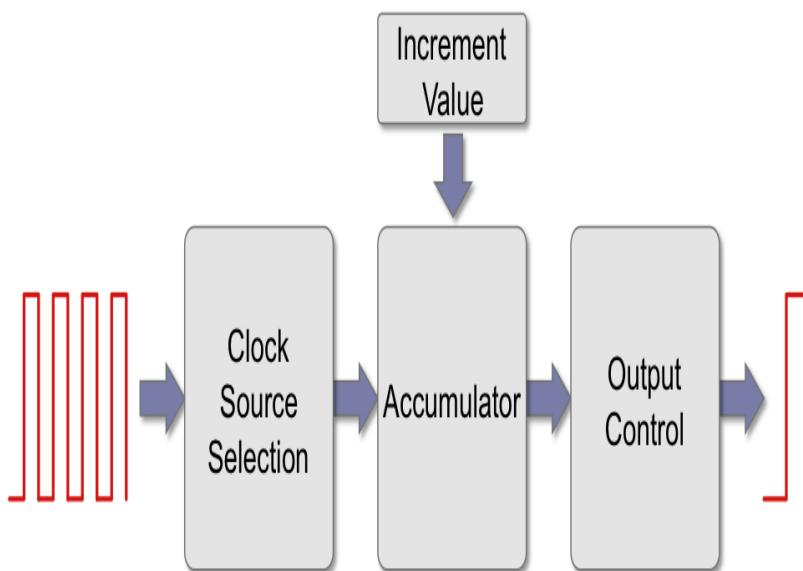
# Oscilador controlado numéricamente

El módulo **oscilador controlado numéricamente (NCOx)** es un temporizador que utiliza el desbordamiento de un acumulador para crear una señal de salida. El desbordamiento del acumulador se controla mediante un valor de incremento ajustable en lugar de un solo pulso de reloj o un incremento de postescalador. Esto ofrece una ventaja sobre un contador simple impulsado por temporizador en que la resolución de la división no varía con el valor algo limitado del divisor preescalador / postescalador. El NCOx es más útil para aplicaciones que requieren precisión de frecuencia y resolución fina en un ciclo de trabajo fijo.

Las características del NCOx incluyen:

- Función de incremento de 16 bits
- Modo de ciclo de trabajo fijo (FDC)
- Modo de frecuencia de pulso (PF)
- Control de ancho de pulso de salida
- Múltiples fuentes de entrada de reloj
- Control de polaridad de salida
- Capacidad de interrupción

El NCOx funciona agregando repetidamente un valor fijo a un acumulador. Las adiciones se producen a la velocidad de reloj de entrada. El acumulador se desbordará con un acarreo periódicamente, que es la salida DE NCOx en bruto. Esto reduce efectivamente el reloj de entrada en la relación entre el valor agregado y el valor máximo del acumulador.



La salida de NCOx se puede modificar aún más estirando el pulso o alternando un flip-flop. La salida NCOx modificada se distribuye internamente a otros periféricos y, opcionalmente, a un pin de E/S. El desbordamiento del acumulador también puede generar una interrupción. El período NCOx cambia en pasos discretos para crear una frecuencia media. Esta salida depende de la capacidad del circuito receptor (es decir, CWG o circuito de convertidor resonante externo) para promediar la salida de NCOx para reducir la incertidumbre.

El desbordamiento del módulo NCO se basa en la siguiente fórmula:

The NCO output is based on the formula below

$$\frac{\text{Accumulator Overflow}}{\text{Rate}} = \frac{\text{Accumulator Overflow Value}}{\text{Input Clock Frequency} \times \text{Increment Value}}$$

## Video Tutorial de Suboficiales

Este video presenta el oscilador controlado numéricamente (NCO) para dispositivos MCU de 8 bits de Microchip y muestra cómo usarlo.

## Modos de suboficiales

El módulo NCO puede emitir una señal en uno de los dos modos.

NCO has Two modes of operation:

- Fixed 50% duty cycle (FDC)



- Pulse Frequency Modulation (PFM)



El registro NCOCON controla la configuración de modo para el suboficial.

## NCOxCON Register

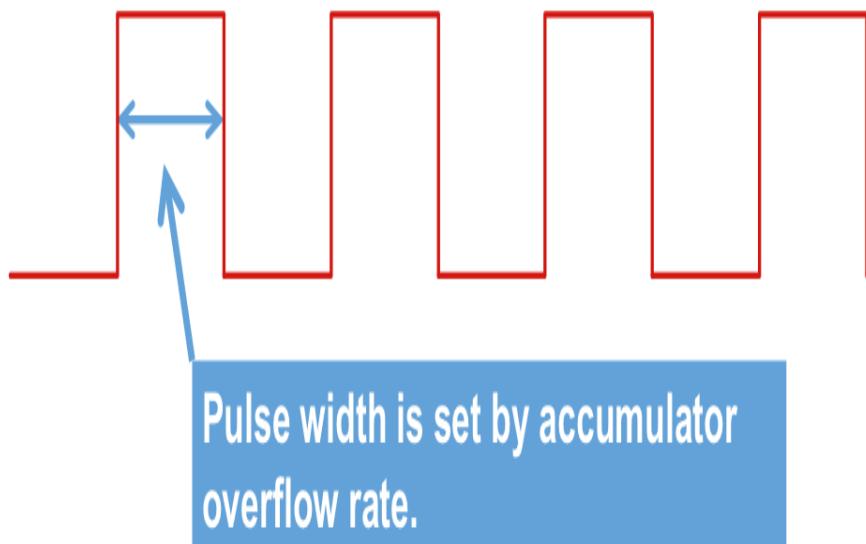
NxEN	NxOE	NxOUT	NxPOL	-	-	-	NxPFM
------	------	-------	-------	---	---	---	-------

### NxPFM NCOx Pulse Frequency Mode Bit

- 1 – NCOx operates in Pulse Frequency Mode
- 0 – NCOx operates in Fixed Duty Cycle Mode

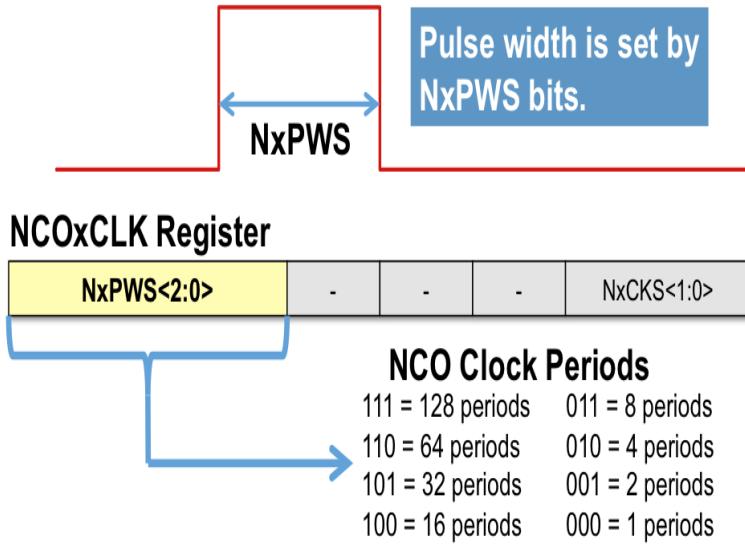
#### Ciclo de trabajo fijo

El modo Ciclo de trabajo fijo alterna la salida en cada desbordamiento del acumulador. Mientras el valor del sumador y el reloj no cambien, esto dará como resultado una salida del ciclo de trabajo del 50%.



#### Modulación de frecuencia de pulso

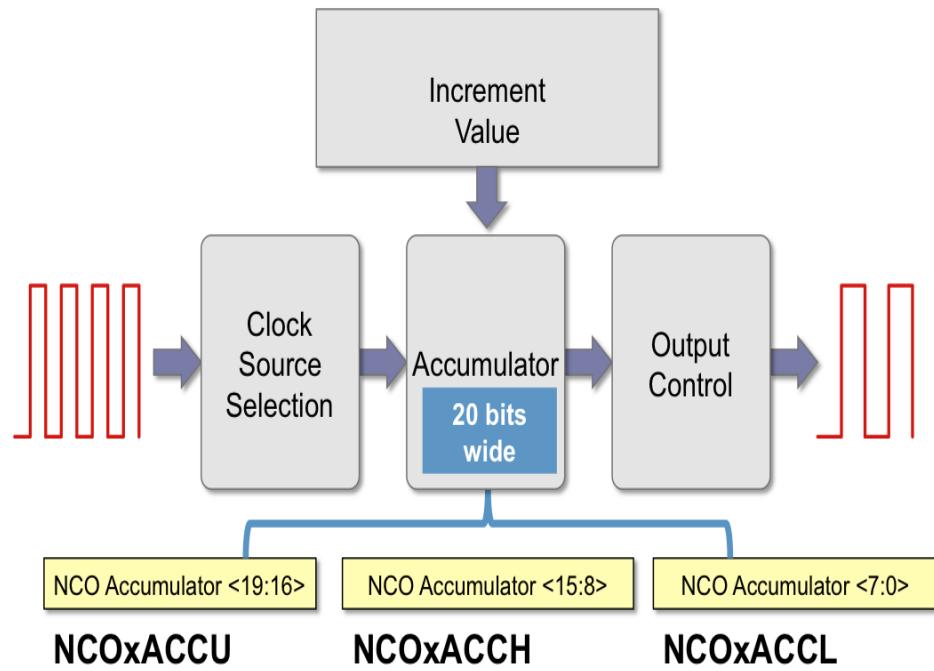
El modo de modulación de frecuencia de pulso activará un pulso en cada desbordamiento del acumulador durante un período establecido por tres bits en el registro NCOCON.



## Acumulador

El acumulador es un registro de 20 bits con un valor máximo de 1.048.575. El acceso de lectura y escritura al acumulador está disponible a través de tres registros:

- NCOxACCL
- NCOxACCH
- NCOxACCU



Cuando el acumulador se desborda, la salida del módulo NCO cambiará de estado.

## Víbora

El sumador NCOx es un sumador completo, que funciona independientemente del reloj del sistema. Agrega el valor del valor de incremento al acumulador en cada pulso de reloj de suboficiales. El sumador toma el valor en el acumulador y luego agrega el valor de incremento. El resultado se

vuelve a colocar en el acumulador. El valor del acumulador se revertirá y cualquier valor más allá de los 1.048.575 se colocará como valor inicial en el acumulador.

Esto se puede restablecer si se desea escribiendo un cero en el acumulador. Esto se hace normalmente dentro de la rutina de servicio de interrupción si está habilitado en el módulo NCO.

## Accumulator Overflow

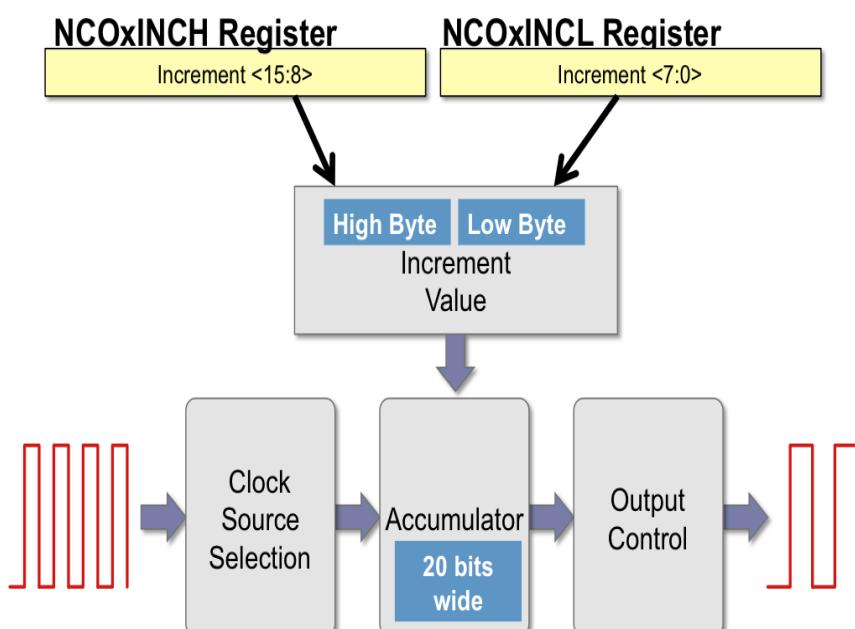
If the addition of the final increment value forces the accumulator to exceed the 1,048,575 maximum value, then any remainder will carry over to the start of the next accumulator count.



## Registros de incremento

El valor de incremento se almacena en dos registros de 8 bits que componen un valor de incremento de 16 bits. Los 8 bits inferiores están en el registro NCOxINCL y los 8 bits superiores están en el registro NCOxINCH.

- NCOxINCL
- NCOxINCH



Ambos registros son legibles y esribibles. Los registros de incremento tienen doble búfer para permitir que se realicen cambios de valor sin deshabilitar primero el módulo NCOx. Las cargas del búfer son inmediatas cuando el módulo está deshabilitado. Es necesario escribir primero en el

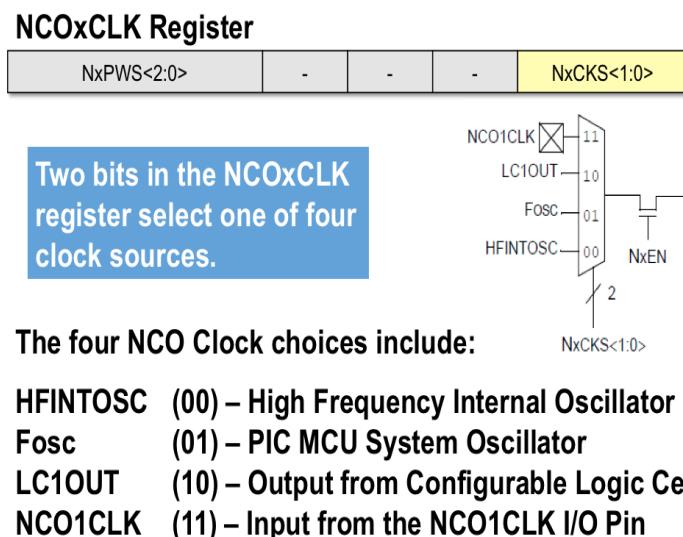
registro NCOxINCH porque luego el búfer se carga sincrónicamente con la operación NCOx después de ejecutar la escritura en el registro NCOxINCL.

## Fuentes de reloj

Las fuentes de reloj disponibles para el NCOx incluyen:

- HFINTOSC
- $F_{osc}$
- LCxOUT
- Pin CLKIN

La fuente de reloj NCOx se selecciona configurando los bits NxCKS<2:0> en el registro NCOxCLK.

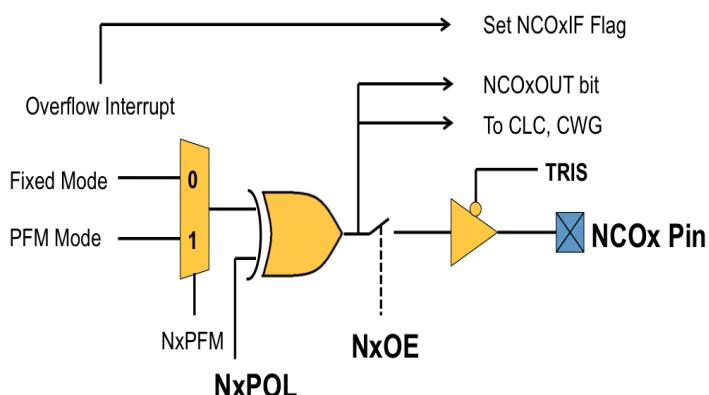


La selección de HFINTOSC continuará ejecutándose incluso si el dispositivo se pone en modo de suspensión.

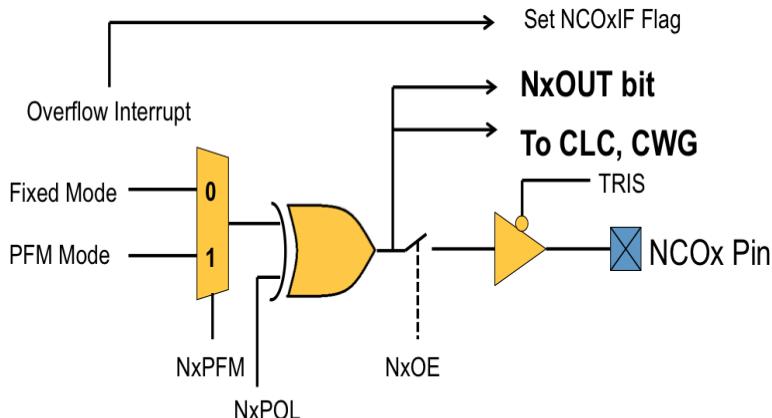
## Salida

La salida del suboficial tiene varias opciones que se pueden establecer en el registro NCOCON.

La salida se puede habilitar o deshabilitar (bit NxOE) y también invertir (bit NxPOL) configurando o borrando bits en el registro NCOCON.



La salida también se puede monitorear en software leyendo el estado del bit NxOUT en el registro NCOCON.



**NCOxCON Register**

NxEN	NxOE	<b>NxOUT</b>	NxPOL	-	-	-	NxPFM
------	------	--------------	-------	---	---	---	-------

El bit NxEN puede deshabilitar todo el módulo NCO. Una configuración '1' habilita el módulo y una configuración '0' lo deshabilita.

## Interruptor

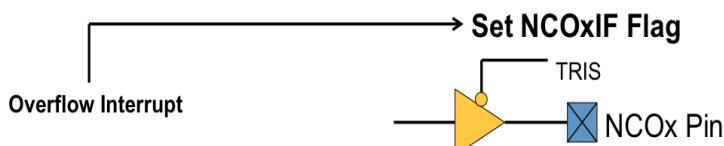
La salida de NCO puede desencadenar una interrupción interna cuando el acumulador se desborda. Esto es manejado por tres bits (GIE, PEIE, NCOxIE) en el conjunto de registros que se muestran a continuación. Esto permitirá al suboficial controlar las acciones del software a través de la rutina de servicio de interrupción en el código de la aplicación, al tiempo que emite una señal a un pin de E/S.

GIE es el bit de habilitación de interrupción global. PEIE es el bit de habilitación de interrupción periférica. NCOxIE es el bit de habilitación de interrupción de suboficiales. Puede haber múltiples módulos de suboficiales. La "x" representa el número de suboficial.

El bit NCOxIF es el indicador de interrupción. Esto se puede monitorear en el software para ver si se ha producido una interrupción. Esto debe borrarse en la rutina de servicio de interrupción o en la rutina de software que lee el bit.

The NCO overflow can also trigger an internal interrupt.

To setup the interrupt, four bits must be set.



**NCOxCON - NCO Control Register**

NxEN	<b>NxOE</b>	NxOUT	NxPOL	-	-	-	NxPFM
------	-------------	-------	-------	---	---	---	-------

**INTCON - Interrupt Control Register**

<b>GIE</b>	<b>PEIE</b>	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF
------------	-------------	--------	------	-------	--------	------	-------

**PIE2 - Peripheral Interrupt Enable Register**

-	C2IE	C1IE	-	BCLIE	<b>NCOxIE</b>	-	-
---	------	------	---	-------	---------------	---	---

**PIR2 - Peripheral Interrupt Request Register**

-	C2IF	C1IF	-	BCL1IF	<b>NCOxIF</b>	-	-
---	------	------	---	--------	---------------	---	---

# Interrupciones de rango medio mejoradas

## Resumen

Las interrupciones son eventos detectados por el MCU que hacen que el flujo normal del programa se anticipe. Las interrupciones pausan el programa actual y transfieren el control a una rutina de firmware escrita por el usuario especificada denominada Rutina de servicio de interrupción (ISR). El ISR procesa el evento de interrupción y, a continuación, reanuda el flujo normal del programa.

En este artículo se muestra cómo habilitar y procesar interrupciones en la familia PIC16F1xxx de PIC de rango medio mejorado.<sup>®</sup>

---

## Descripción general del proceso de interrupción

1

### Programar MCU para reaccionar a las interrupciones

El MCU debe programarse para permitir que se produzcan interrupciones. La configuración de la habilitación de interrupción global (GIE) y, en muchos casos, la habilitación de interrupción periférica (PEIE) permite que la MCU reciba interrupciones. GIE y PEIE se encuentran en el registro de funciones especiales de Control de interrupciones (INTCON).

2

### Habilitar interrupciones de periféricos seleccionados

Cada periférico en el MCU tiene un bit de habilitación individual. El bit de habilitación de interrupción individual de un periférico debe establecerse, además de GIE/PEIE, antes de que el periférico pueda generar una interrupción. Los bits de habilitación de interrupciones individuales se encuentran en INTCON, PIE1, PIE2 y PIE3.

3

### Periférico afirma una solicitud de interrupción

Cuando un periférico alcanza un estado en el que se necesita la intervención del programa, el periférico establece un indicador de solicitud de interrupción (xxIF). Estos indicadores de interrupción se establecen independientemente del estado de los bits de habilitación de interrupciones GIE, PEIE e individuales. Los indicadores de interrupción se encuentran en INTCON, PIR1, PIR2 y PIR3.

Los indicadores de solicitud de interrupción se bloquean alto cuando se establecen y deben ser borrados por el ISR escrito por el usuario.

4

### Se produce una interrupción

Cuando se establece un indicador de solicitud de interrupción y la interrupción está habilitada correctamente, comienza el proceso de interrupción:

- Las interrupciones globales se desactivan desactivando GIE a 0.

- El contexto actual del programa se guarda en los registros de sombra.
- El valor del contador de programa se almacena en la pila de retorno.
- El control del programa se transfiere al vector de interrupción en la dirección 04h.

**5**

## IsR se ejecuta

El ISR es una función escrita por el usuario y colocada en la dirección 04h. El ISR hace lo siguiente:

1. Comprueba los periféricos habilitados para interrupciones en busca del origen de la solicitud de interrupción.
2. Realiza las tareas periféricas necesarias.
3. Borra el indicador de solicitud de interrupción adecuado.
4. Ejecuta la instrucción Return From Interrupt (RETFIE) como la instrucción ISR final.

**6**

## El control se devuelve al programa principal

Cuando se ejecuta RETFIE:

1. Las interrupciones globales están habilitadas (GIE=1).
2. El contexto del programa se restaura desde los registros de sombras.
3. La dirección de retorno de la pila se carga en el contador de programas.
4. La ejecución se reanuda desde el punto en el que se interrumpió.

# Registros utilizados para procesar interrupciones

## Registro de control de interrupciones

Registro INTCON

GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF
bit 7				bit 0			
GIE - Global Interrupt Enable							
PEIE - Peripheral Interrupt Enable							
TMR0IE - Timer0 Interrupt Enable							
INTE - External Interrupt Enable							
IOCIE - Interrupt on Change Enable							
TMR0IF - Timer0 Interrupt flag							
INTF - Indicador de interrupción externa				IOCIF - Indicador de interrupción en el indicador de cambio			

INTCON contiene indicadores de habilitación de interrupciones globales y periféricas, así como los indicadores de solicitud de interrupción individuales y los indicadores de habilitación de interrupciones para tres de las interrupciones PIC16F1xxxx.

## Registros de habilitación de interrupciones

### Registro PIE1

TMR1GIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMRI1E
bit 7							
bit 0							
<b>TMR1GIE</b> - Timer1 Gate Interrupt Enable							
<b>ADIE</b> - Analog-to-Digital Converter (ADC) Interrupt Enable							
<b>RCIE</b> - Universal Synchronous Asynchronous Receiver Transmitter (USART) Receive Interrupt Enable							
<b>TXIE</b> - USART Transmit Interrupt Enable							
<b>SSPIE</b> - Synchronous Serial Port (MSSP) Interrupt Enable							
<b>CCP1IE</b> - CCP1 Interrupt Enable							
<b>TMR2IE</b> - Timer2 Interrupt Enable							
<b>TMRI1E</b> - Timer1 Interrupt Enable							

### Registro PIE2

OSFIE	C2IE	C1IE	EEIE	BCLIE	LCDIE	---	CCP2IE
bit 7							
bit 0							
<b>OSFIE</b> - Interruptor de falla del oscilador Habilitar							
<b>C2IE</b> - Comparador C2 Interrupción Habilitar							
<b>C1IE</b> - Comparador Interrupción C1 Habilitar							
<b>EEIE</b> - EEPROM Interrupción de finalización de escritura Habilitar							
<b>BCLIE</b> - Interrupción de colisión de bus MSSP Habilitar							
<b>LCDIE</b> - Interrupción del módulo LCD Habilitar							
--- - No implementado, leído como 0							
<b>CCP2IE</b> - CCP2 Interrupt Enable							

### Registro PIE3

---	CCP5IE	CCP4IE	CCP3IE	TMR6IE	---	TMR4IE	---
bit 7							
bit 0							
--- - Lectura no implementada como 0							
<b>CCP5IE</b> - CCP5 Interrupt Enable							
<b>CCP4IE</b> - CCP4 Interrupt Enable							
<b>CCP3IE</b> - CCP3 Interrupt Enable							
<b>TMR6IE</b> - Timer6 Interrupt Enable							
--- - Unimplemented, read as 0							
<b>TMR4IE</b> - Timer4 Interrupt Enable							
--- - Sin implementar, leído como 0							

PIE1, PIE2 y PIE3 contienen los indicadores de habilitación de interrupciones individuales para los periféricos del MCU.

## Registros de solicitudes de interrupción

### Registro PIR1

TMR1GIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMRI1F
bit 7					bit 0		

TMR1GIF - Indicador de interrupción de puerta ADIF - Indicador

de interrupción de ADC RCIF - Indicador de interrupción de recepción

USART TXIF - Indicador de interrupción de transmisión USART

SSPIF - Indicador de interrupción MSSP CCP1IF - Indicador de

interrupción CCP1 TMR2IF - Indicador de interrupción del temporizador2

TMRI1F - Indicador de interrupción de Timer1

### Registro PIR2

OSFIF	C2IF	C1IF	EEIF	BCLIF	LCDIF	---	CCP2IF
bit 7					bit 0		

OSFIF - Indicador

de interrupción de fallo del oscilador C2IF - Indicador de interrupción

del comparador C2 C1IF - Indicador de interrupción

del comparador C1 EEIF - Indicador de interrupción

de finalización de escritura EEPROM BCLIF - Indicador

EEIF de interrupción de colisión del bus MSSP LCDIF - Indicador de interrupción del módulo LCD --- - No implementado, leído como 0

CCP2IF - CCP2 Interrupt Flag

### Registro PIR3

---	CCP5IF	CCP4IF	CCP3IF	TMR6IF	---	TMRI4IF	---
bit 7					bit 0		

--- - Sin implementar, leído como 0

CCP5IF - CCP5 Indicador

de interrupción CCP4IF - CCP4 Indicador

de interrupción CCP3IF - Indicador

de interrupción CCP3 TMR6IF - Indicador

de interrupción del temporizador6 --- - No implementado, leído como 0

TMRI4IF - Indicador

de interrupción del temporizador4 --- - No implementado, léase como 0

PIR1, PIR2 y PIR3 contienen los indicadores de solicitud de interrupción individuales para los periféricos del MCU.

## OPTION\_REG

### OPTION\_REG

WPUEN	INTEDG	TMR0CS	TMR0SE	PSA	PS<2:0>
-------	--------	--------	--------	-----	---------

bit 7

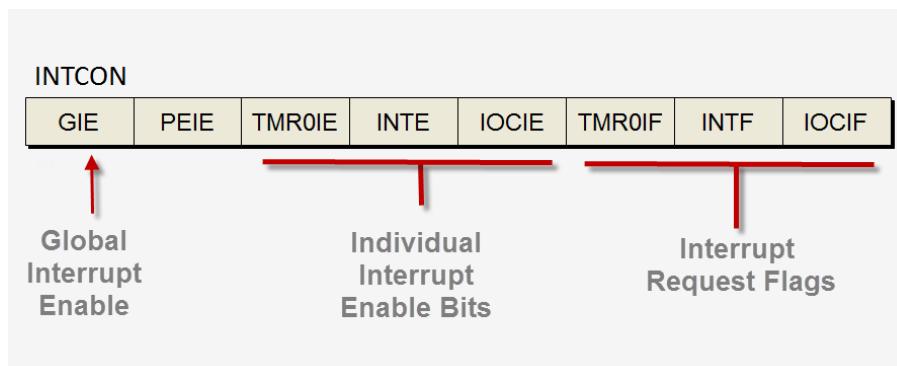
bit 0

El indicador INTEDG en OPTION\_REG se utiliza para establecer una arista ascendente o defectuosa en el pin INT como desencadenador de una interrupción INTE.

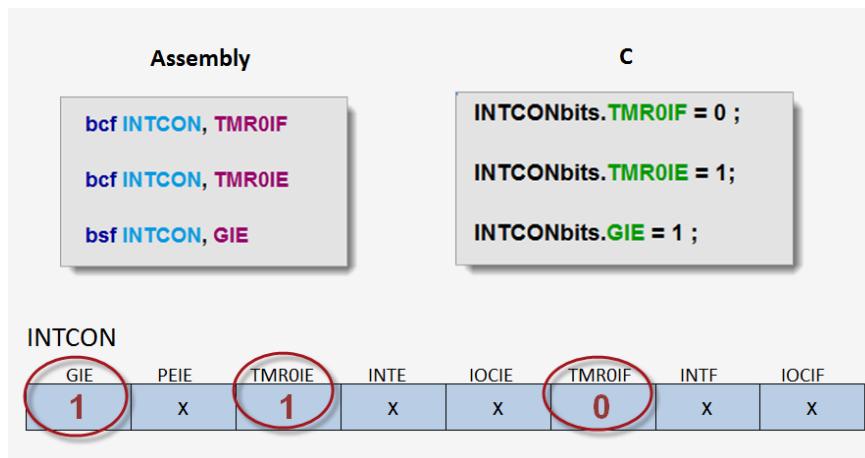
## Habilitación de interrupciones

### Interrupciones principales

Tres fuentes de interrupción (Timer0, Interrupción externa e Interrupción al cambiar) tienen bits de habilitación de interrupciones ubicados en INTCON. Estas interrupciones se denominan interrupciones centrales.



Para habilitar una de las interrupciones principales, solo es necesario configurar el bit de activación de interrupción individual y GIE.



Borrar un indicador de solicitud de interrupción antes de establecer el indicador de activación de interrupción evita que las solicitudes de interrupción pendientes desencadenen una interrupción inmediata.

### Interrupciones periféricas

Los periféricos PIC16F1xxx, capaces de generar solicitudes de interrupción cada uno, tienen sus indicadores de habilitación de interrupciones en uno de los tres registros PIE. Para habilitar una interrupción periférica, el indicador de interrupción individual, GIE y PEIE deben estar todos configurados.

Assembly	C
<pre> banksel PIR1 bsf PIR1, SSPIF banksel PIE1 bsf PIE1, SSPIE  bsf INTCON, PEIE bsf INTCON, GIE </pre>	<pre> PIR1bits.SSP1IF = 0 ; PIE1bits.SSP1IE = 1; INTCONbits.PEIE =1; INTCONbits.GIE = 1 ; </pre>

	GIE	PEIE	TMROIE	INTE	IOCIE	TMROIF	INTF	IOCIF
INTCON	1	1	X	X	X	X	X	X
	TMR1GIE	ADIE	RCIE	TXIE	SSP1IE	CCPIE	TMR2IE	TMR1IE
PIE1	X	X	X	X	1	X	X	X
	TMR1GIF	ADIF	RCIF	TXIF	SSP1IF	CCPIF	TMR2IF	TMR1IF
PIR1	X	X	X	X	0	X	X	X

## Mantenimiento de una interrupción

### Isr

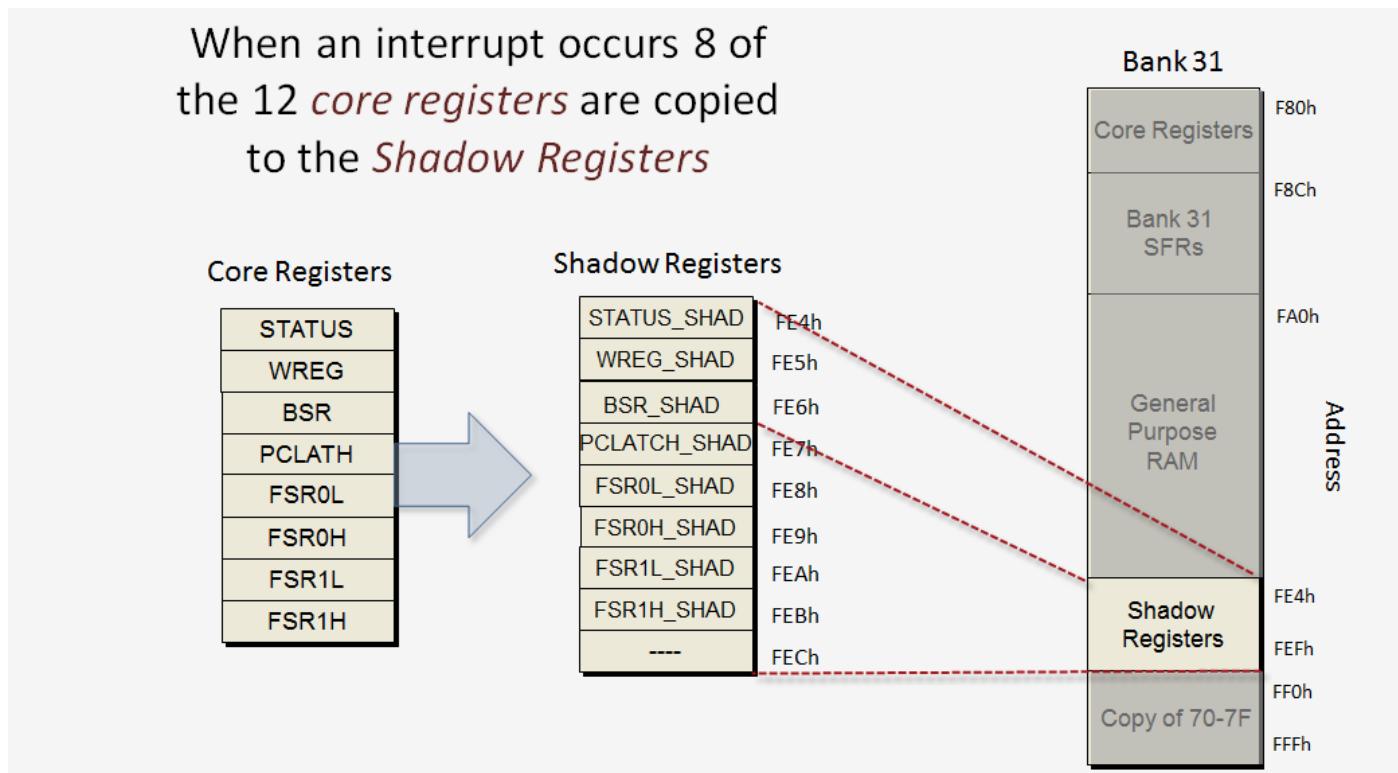
El ISR es un programa escrito por el usuario que realiza las tareas necesarias cuando se produce una interrupción. El usuario es responsable de escribir el ISR y colocarlo en la dirección 04h. La última instrucción ejecutada por el ISR debe ser la instrucción RETFIE. Los ISR se pueden escribir en C o en lenguaje ensamblador.

### ISR en Asamblea (izquierda) e ISR en C (derecha)

<pre> ORG 0      ; tells MPASM to put code at address 0 goto Start  ; bypass ISR on startup/reset  ORG 4      ; tells MPASM to put code at 04h  my_ISR:    ; ISR  ;  ; ISR code goes here ;  retfie  Start: goto \$; application code goes here </pre>	<pre> void interrupt my_ISR (void) {     // ISR code goes here } </pre> <p>Using the XC8 keyword <code>interrupt</code> places the function at address 04h and inserts the RETFIE instruction</p>
--	---

## Ahorro de contexto

El hardware del MCU invocará el mecanismo de guardado de contexto cuando se produzca una interrupción.



## Verificar el origen de la interrupción

Hay un ISR que atiende todas las interrupciones de la aplicación. El ISR necesita verificar secuencialmente los indicadores de solicitud de interrupción individuales para determinar el origen de la interrupción.

## Borrar el indicador de solicitud de interrupción

Los indicadores de solicitud de interrupción se bloquean cuando los establece el periférico. Deben ser autorizados por ISR. Si el ISR no puede restablecer un indicador de solicitud, se producirá otra interrupción inmediatamente después de que el ISR devuelva el control al programa principal.

```
void interrupt sample_isr (void)
{
    if (TMR2IF == 1)
    {
        // perform TMR2 Interrupt task
        TMR2IF = 0 ;
    }

    if (INTE == 1)
    {
        // External Interrupt tasks
        INTF = 0 ;
    }

    if (CCP1IF == 1)
        // CCP tasks
        CCP1IF = 0;
}
```

# Código de interrupción de ejemplo

## Procesamiento de una interrupción desde Timer2

La siguiente animación muestra el código para configurar y procesar una interrupción de Timer2 en el MCU de rango medio mejorado PIC16F1xxx. Los iconos de control en la parte inferior de la animación permiten que la pantalla se detenga y se escalone un solo paso.

Puede encontrar más información sobre las interrupciones PIC16F1xxx en la página [Interrupciones](#).

La página [Timer2/4/6](#) proporciona los detalles sobre la configuración y el funcionamiento de Timer2.

# Ejemplo de procesamiento de una interrupción en un MCU PIC16F1xxx

## Procesamiento de una interrupción desde Timer2

La siguiente animación muestra el código para configurar y procesar una interrupción de Timer2 en el MCU de rango medio mejorado PIC16F1xxx. Los iconos de control en la parte inferior de la animación permiten que la pantalla se detenga y se escalone un solo paso.

Puede encontrar más información sobre las interrupciones PIC16F1xxx en la página [Interrupciones](#).

La página [Timer2/4/6](#) proporciona los detalles sobre la configuración y el funcionamiento de Timer2.

# Ejemplos de programación con el MCU PIC16F1xxx

## Resumen

Esta página contiene ejemplos de cuatro tareas de programa que utilizan un microcontrolador PIC (MCU) de rango medio mejorado. Cada página de ejemplo muestra el código de lenguaje C necesario para completar la tarea.<sup>®</sup>

Estos ejemplos se crearon con MPLAB X IDE. A continuación se proporciona información sobre el IDE, el proyecto y las opciones de bits de configuración para cada uno de estos ejemplos.<sup>®</sup>

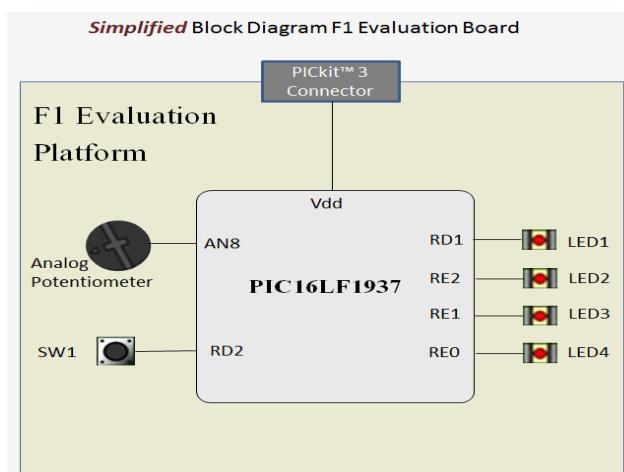
## Recursos utilizados en programas de ejemplo

### - Plataforma de Desarrollo

Los programas de ejemplo fueron diseñados para ser ejecutados en el [kit de evaluación F1](#) de Microchip (número de pieza DV164132).

Este kit incluye:

- El kit de evaluación F1 (número de pieza DM164130-1).
- El programador/[depurador PICkit™ 3](#) (número de pieza PG164130).



El diagrama de bloques parciales del kit de evaluación F1 ilustra las conexiones utilizadas por los programas de ejemplo:

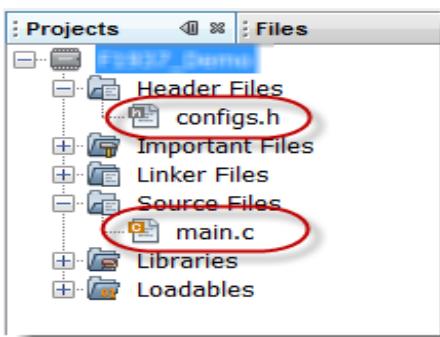
Anclar	Tipo	Dirección	Función
BIT1 PORTD (RD1)	Digital	Salida	unidades LED1
PORTE Bit2 (RE2)	Digital	Salida	unidades LED2
PORTE Bit1 (RE1)	Digital	Salida	unidades LED3
PORTE Bit0 (RE0)	Digital	Salida	unidades LED4
AN8	Analógico	Entrada	potenciómetro
BIT2 PORTD (RD2)	Digital	Entrada	comutador 1 (SW1)



V<sub>Dd</sub> se suministra a través del conector PICkit 3. Para ejecutar el código de ejemplo, debe configurar el PICkit 3 para suministrar la energía al destino.

### - Configuración de MPLAB X IDE

Los proyectos IDE de MPLABX utilizados en los programas de ejemplo comparten una estructura similar.<sup>®</sup>



## Archivos de proyecto

Los proyectos de ejemplo son aplicaciones sencillas que demuestran una característica de la arquitectura MCU.

La versión en lenguaje C de los ejemplos tiene dos archivos:

- `config.h` que contiene los valores de bits de configuración
- `main.c` contiene todo el código

Los ejemplos en MPASM™ solo tienen un archivo de origen (`main.asm`) incluido en el proyecto.

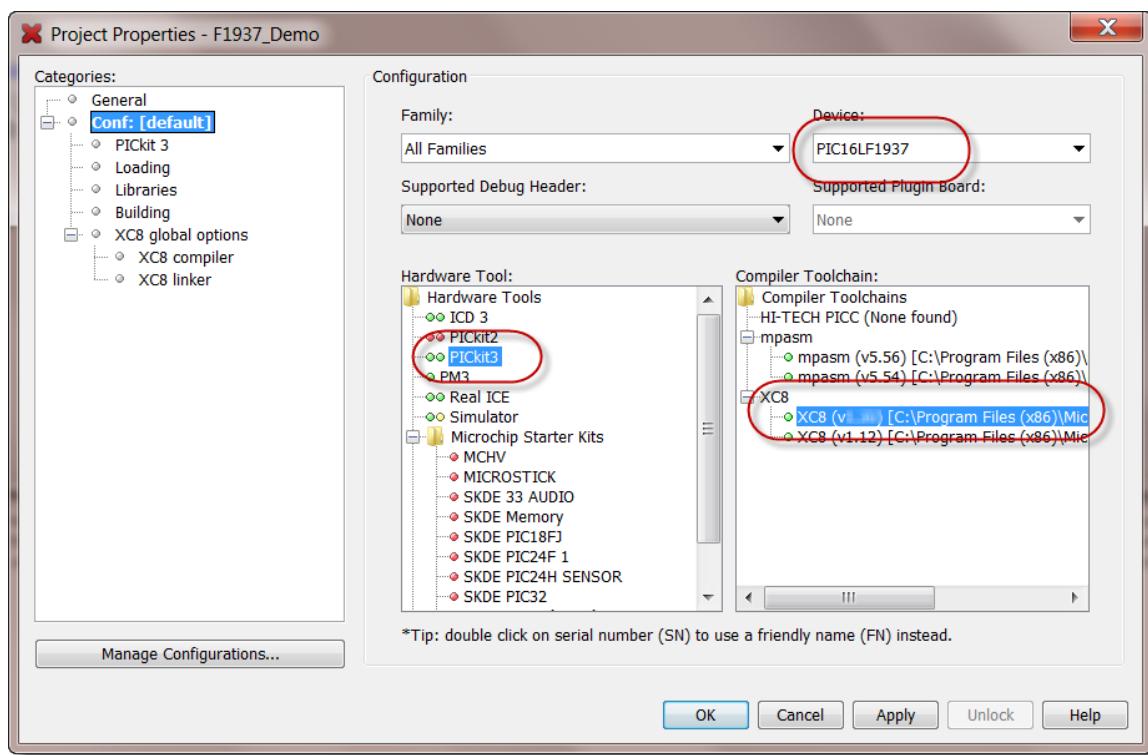
## Configuración de propiedades del proyecto

Todos los proyectos utilizados en los programas de ejemplo utilizan:

- PIC16LF1937 como procesador
- PICkit™ 3 como herramienta de hardware

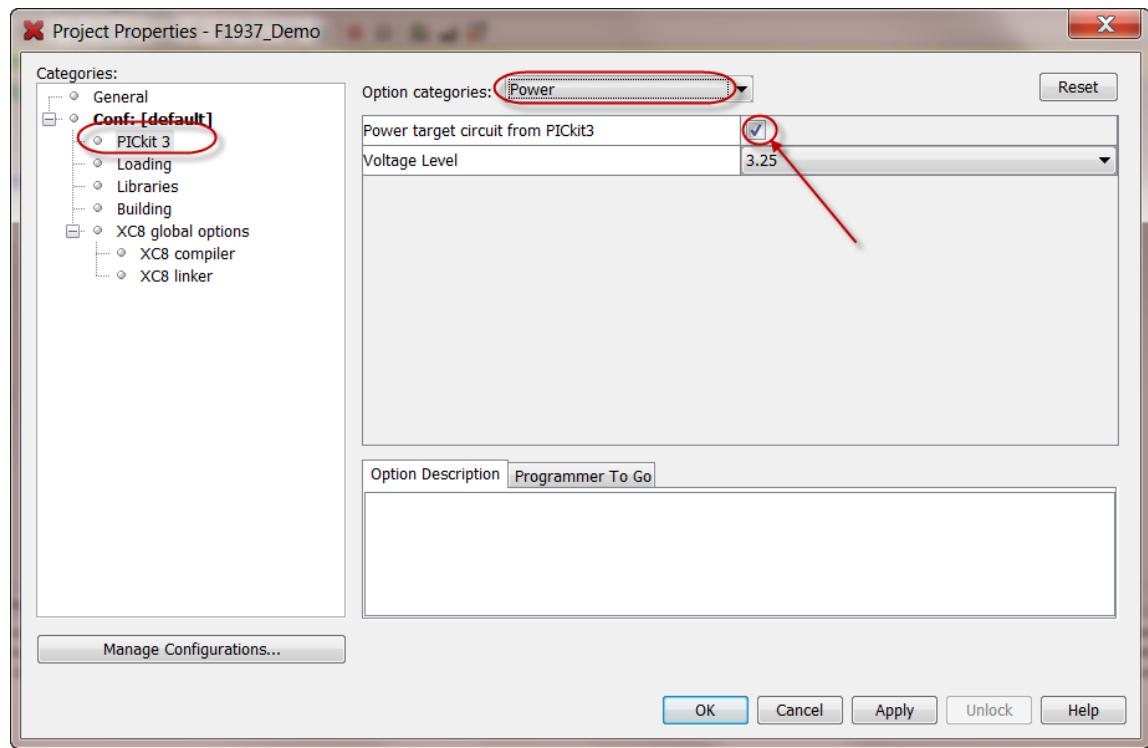
Los proyectos en C utilizan el compilador MPLAB XC8 (v1.21 o superior).

Los proyectos de MPASM utilizan la última versión de MPASM como herramienta de software.



## Configuración de PICkit 3

El proyecto requiere configurar el PICkit 3 para suministrar V<sub>Dd</sub> al tablero de destino. Esto elimina la necesidad de una fuente de alimentación externa.



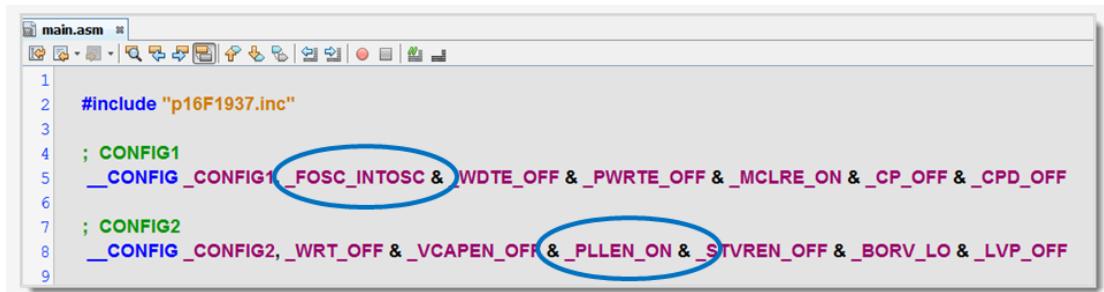
### - Descripción de los bits de configuración

Una preocupación principal al establecer los bits de configuración para cualquier aplicación es la fuente de reloj. Los programas de ejemplo se escribieron para una placa de desarrollo que no tiene una fuente de reloj externa para el MCU. Como resultado, los bits de configuración dirigen al PIC16LF1937 para usar el oscilador RC interno combinado con el 4x PLL como fuente de reloj.

Los programas de ejemplo no necesitan otras características habilitadas para la configuración.

## Lenguaje ensamblador

Para los programas de ejemplo escritos en lenguaje ensamblador, los bits de configuración se codifican en línea utilizando la directiva **\_CONFIG**:



```
main.asm
1 #include "p16F1937.inc"
2
3 ; CONFIG1
4 _CONFIG _CONFIG1 _FOSC_INTOSC & _WDTE_OFF & _PWRTE_OFF & _MCLRE_ON & _CP_OFF & _CPD_OFF
5
6 ; CONFIG2
7 _CONFIG _CONFIG2, _WRT_OFF & _VCAPEN_OFF & _PLLEN_ON & _STVREN_OFF & _BORV_LO & _LVP_OFF
8
9
```

## Lenguaje C

Por ejemplo, en los programas escritos en C, los valores de bits de configuración se proporcionan en un archivo denominado `configs.h`.

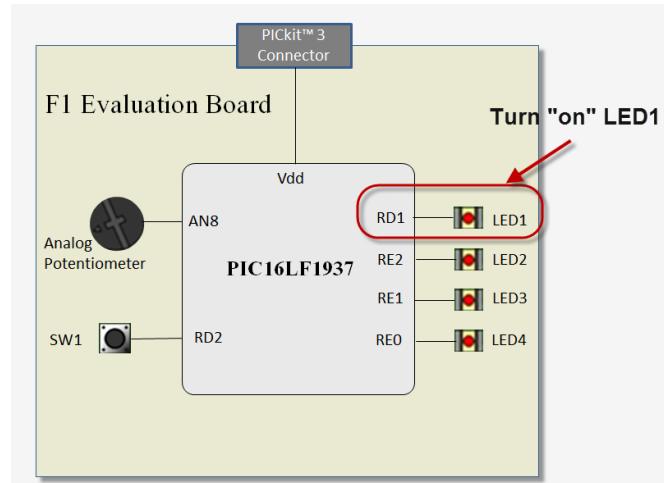
El archivo `configs.h` se encuentra en el directorio del proyecto y se carga en el proyecto con una **directiva #include** de `main.c`



Puede encontrar más información sobre los ajustes de bits de configuración en el [Tutorial de rango medio mejorado](#).

## Ejemplos de programas

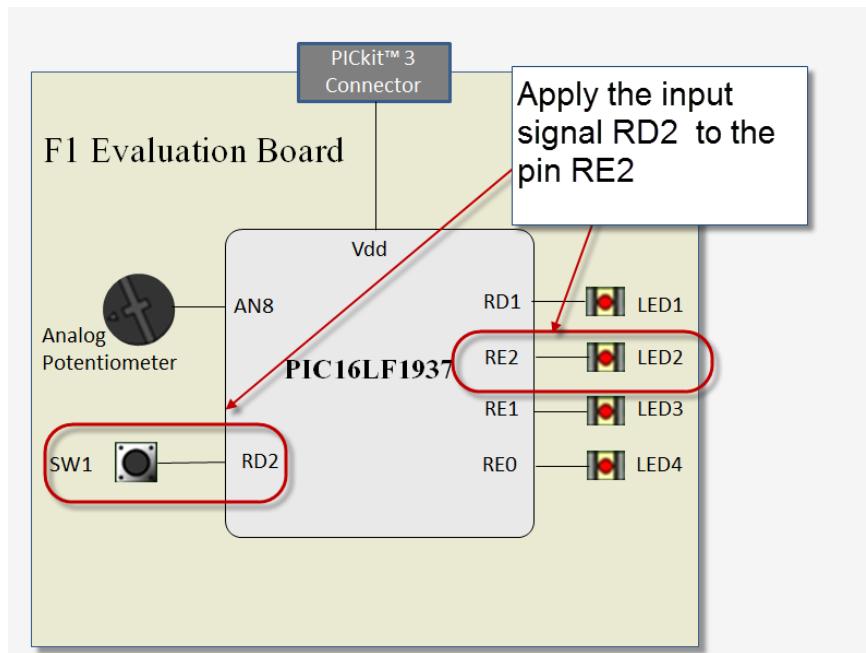
- Salida digital



Este programa de ejemplo activa LED1 emitiendo un 1 lógico a RD1 (es decir, PUERTO D, bit 1).

C code	Assembly code
<pre> 1 #include "configs.h"      // configuration bits 2 #include &lt;xc.h&gt;          // MPLAB XC8 header file 3 4 5 int main(void) { 6     TRISD = 0 ;           // all bits on PORTD as outputs 7     while(1) 8     { 9         LATDbits.LATD1 = 1 ; //turn on RD1 10    } 11 } </pre>	<pre> 1 #include "p16F1937.inc" 2 ; CONFIG1 3 __CONFIG__CONFIG1, _FOSC_INTOSC 4 5 ; CONFIG2 6 __CONFIG__CONFIG2, _LVP_OFF 7 8 9 ORG 0 10 goto start ; bypass interrupt vector 11 12 13 ORG 4 14 retfie ; interrupt vector 15 16 17 start: 18 banksel TRISD 19 clrf TRISD ; make all PORTD pins output pins 20 banksel LATD 21 moviw 0x02 22 movwf LATD ; turn on RD1 23 goto \$ 24 25 end </pre>

### - Entrada digital



Este programa de ejemplo toma la entrada de RD2 (PUERTO D, bit2) y la envía en el pin de salida RE2 (PUERTO E, bit2).

```

1 #include "configs.h"      // configuration bits
2 #include <xc.h>          // MPLAB XC8 header file
3
4
5 int main(void) {
6     TRISEbits.TRISE2 = 0 ; // make RE2 an output
7     ANSELDbits.ANSD2 = 0 ; // make RD2 a Digital Pin
8     TRISD = 0xFF ;        // ensure all PORTD pins are input
9     while(1)
10    {
11        LATEbits.LATE2 = PORTDbits.RD2 ;
12    }
13 }

```

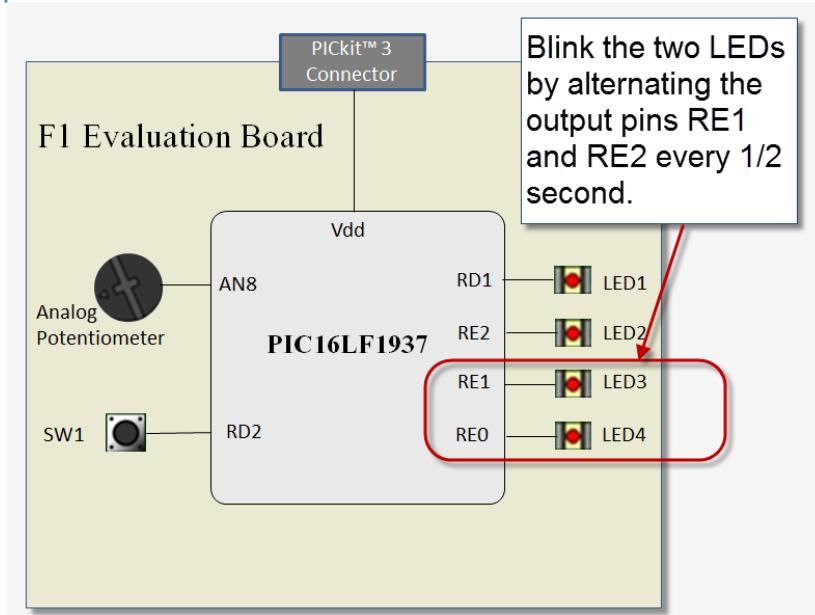
Set RE2 as an output pin

Set RD2 as a digital pin

Ensure all PORTD pins are inputs

Takes input from RD2 then outputs it to RE2

## - Uso de una interrupción



## Procesamiento de una interrupción

En este ejemplo se utiliza una interrupción Timer2 para alternar un patrón en dos LED. Para realizar esta tarea, la aplicación hace lo siguiente:

- Inicializar puertos de E/S
- Inicializar Timer2 para generar una interrupción periódica
- Escriba una rutina de servicio de interrupciones para alternar los LED
- Habilitar interrupciones de Timer2

```
main.c
1 #include "configs.h"           // configuration bits
2 #include <xc.h>                // MPLAB XC8 header file
3
4 void initialize_tmr2(void)
5 {
6     TMR2ON = 1;
7     T2CONbits.T2CKPS = 3;        // set pre-scaler to 64
8     T2CONbits.T2OUTPS1 = 1;      // post scaler = 3
9     PR2 = 255;
10}
11
12 void interrupt ISR(void)
13 {
14     TMR2IF = 0;                 // Clear IF flag
15     LATE ^= 0x03;               // toggle RE0 and RE1
16 }
17
18 int main(void)
19 {
20     LATE = 0x01;                // RE0 = 1 , RE1 = 0
21     TRISE = 0xFC;               // make RE0 and RE1 outputs
22     initialize_tmr2();          // initialize Timer2
23
24     TMR2IE = 1;                 // enable Timer2 interrupt
25     PEIE = 1;                   // enable peripheral interrupts
26     GIE = 1;                    // enable global interrupts
27
28     while(1);
29 }
30
```

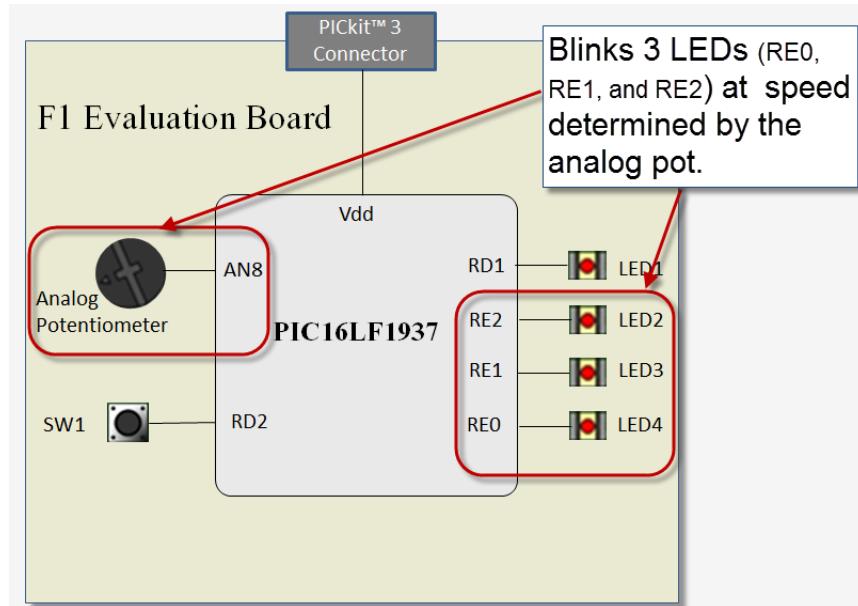
Initializes Timer2 to interrupt every 500ms

Interrupt Service Routine:  
Clears interrupt request flag then toggles RE0 and RE1

Initialize I/O ports and Timer2

Enable Interrupts

## - Múltiples interrupciones

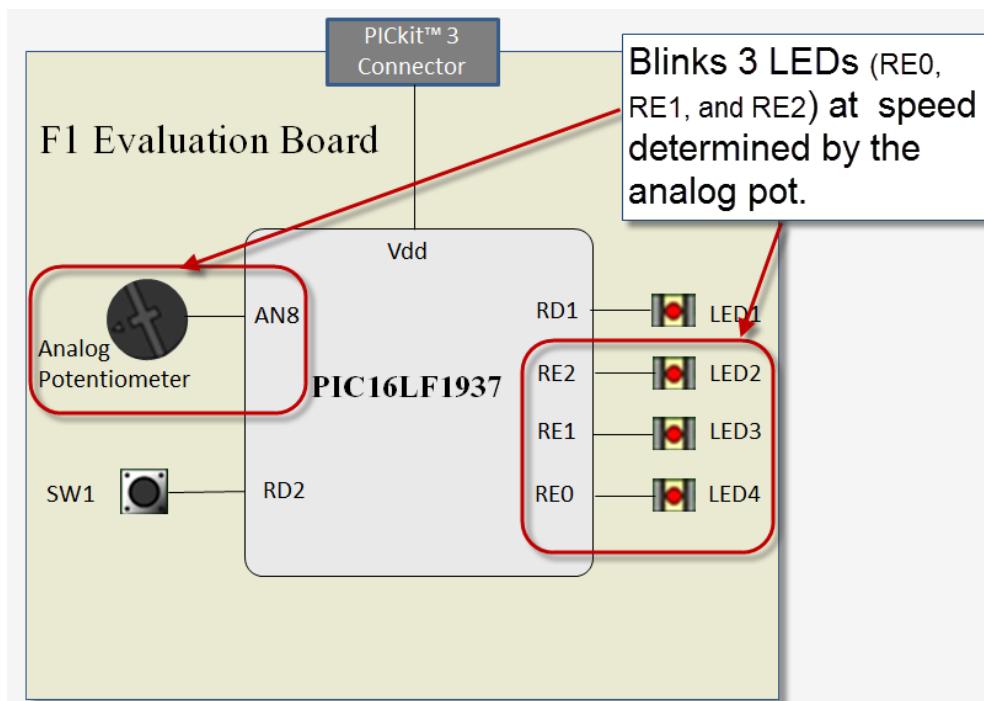


## Procesamiento de múltiples interrupciones

En este ejemplo se utiliza una interrupción Timer2 para parpadear los LED. El intervalo de tiempo utilizado por Timer2 está determinado por el valor del potencímetro analógico conectado a AN8. La rutina de conversión que recopila el valor en AN8 se desencadena mediante una interrupción. Este programa de ejemplo hace lo siguiente:

- Inicializa los puertos de E/S
- Internaliza Timer2 y los módulos ADC
- Implementa rutinas de servicio de interrupción de Timer2 y ADC
- Habilita interrupciones

## Funciones de soporte



## main() y rutina de servicio de interrupción

```
35 int Last_ADC_Result ;
36
37 int main(void)
38 {
39     initialize_tmr2();           // initialize Timer2
40     enable_interrupts();
41     initialize_adc();
42     initialize_io();
43
44     while(1) ;                  // main function
45 }
46
47
48 void interrupt ISR(void)
49 {
50     if( TMR2IF )
51     {
52         TMR2IF = 0 ;            // Clear IF flag
53         PR2 = Last_ADC_Result; // update Timer2 period
54         LATE ^= 0x07 ;         // toggle RE0 and RE1
55         ADGO = 1 ;             // start an ADC conversion
56     }
57
58     if ( ADIF )
59     {
60         Last_ADC_Result = ADRESH ;
61         ADIF = 0 ;
62     }
63 }
```

Initializes system then goes into `while(1)` loop

### Timer2 ISR:

- 1) Clears Interrupt Flag
- 2) Updates Timer Period Reg.
- 3) Toggle LED outputs
- 4) Starts next ADC

### ADC ISR:

- 1) Records ADC result
- 2) Clears interrupt flag



**PREGUNTA:** En lugar de que el TIMER2 inicie la próxima conversión de analógico a digital, ¿no tendría más sentido que el ISR de ADC iniciara una conversión?

**RESPUESTA:** En este ejemplo, el período Timer2 es tal que podrían producirse varios miles de conversiones A-D entre interrupciones del temporizador. Dado que solo el último resultado de ADC se coloca en PR2, no se utilizarían todos los demás valores de ADC capturados en un período. Hacer que el ISR del temporizador reinicie el ADC reduce el número de interrupciones.

### - Descargar

### Archivos de laboratorio

Los archivos de los laboratorios se pueden descargar a continuación:

Archivo	Descargar			Instrucciones de instalación
	Windows	Linux	Mac OS X	
Archivos de proyecto y de origen	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>

### Software requerido

Para ejecutar los programas de demostración, deberá tener MPLAB X IDE y MPLAB XC8 C Compiler instalados en su computadora. Los proyectos se pueden ejecutar en la placa de evaluación F1 con el PICkit 3 o utilizando el simulador de software de MPLAB X IDE.



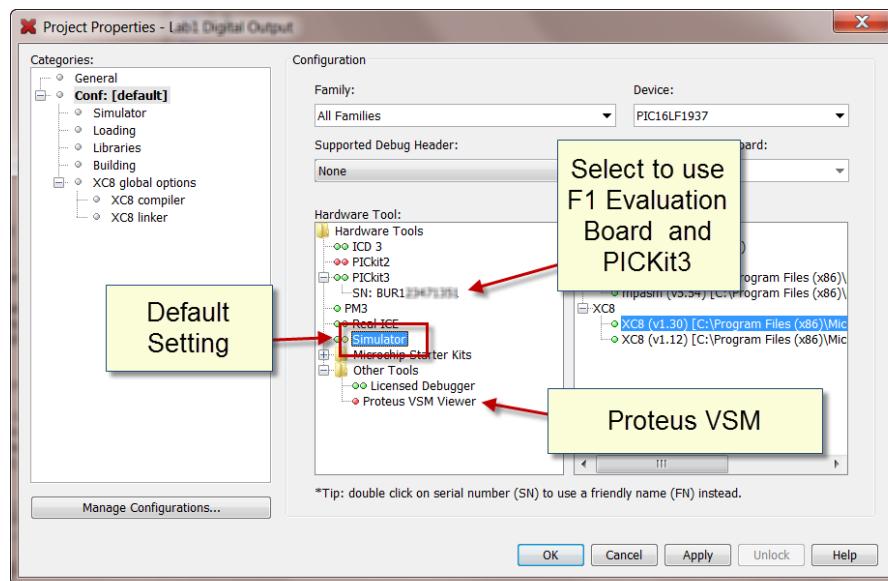
## Instalación

Para instalar los archivos de laboratorio, descomprima el archivo descargado en cualquier directorio de su computadora. El documento "Introducción" en los archivos de proyecto descargados muestra cómo instalar el compilador y MPLAB X IDE.

## Configuración de los programas de demostración

Cuando se instalan, los archivos de laboratorio contienen proyectos MPLAB X para cada demostración.

Todos los proyectos de demostración se han configurado para utilizar el simulador interno de MPLAB X IDE como herramienta de hardware. Tiene la opción de seleccionar el Kit de evaluación de F1 como herramienta de simulación. El cambio de la configuración predeterminada se puede realizar desde la ventana **Propiedades del proyecto** para cada uno de los proyectos.



## Ejecución de los programas de demostración

Puede ejecutar el proyecto de demostración utilizando MPLAB X IDE. Deberá abrir el proyecto y ejecutarlos en **modo de depuración**. Si está utilizando el kit de evaluación F1 con un PICkit 3, tiene la opción de presionar el botón **Proyecto de destino del programa** para ver el proyecto ejecutarse en el hardware de destino.