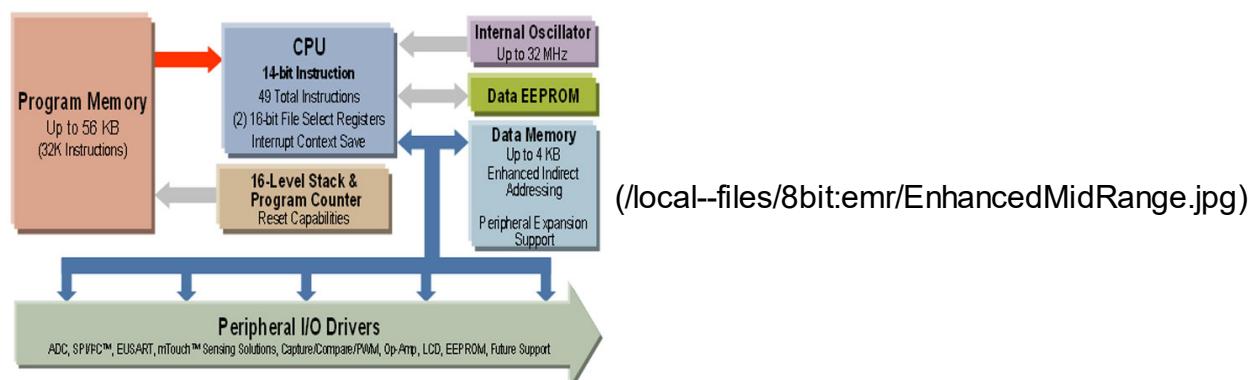


Familia de gama media mejorada

MCU PIC® de 8 bits de rango medio mejorado
(<http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=1025&mid=10&lang=en&pageId=74>)
PIC12F_1 xxx, PIC16F_1 xxx

- 49 (14 bits de ancho) instrucciones fáciles de aprender
- Memoria de programa direccionable de 32 K palabras (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, múltiples comunicaciones en serie y capacidad de control de motores



Haga clic en la imagen para ampliar.

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

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)

(<https://skills.microchip.com/programming-microchip-pic16f-using-mcc-volume-2>)

Generador de forma de onda complementario en 8 bits

(<https://skills.microchip.com/complimentary-waveform-generator-on-8-bit>)

Introducción a la celda lógica configurable (<https://skills.microchip.com/configurable-logic-cell-introduction>)

Introducción al oscilador controlado numéricamente (<https://skills.microchip.com/introduction-to-the-numerically-controlled-oscillator>)

Programación de Microchip PIC16F usando MCC (Volumen 1)

(<https://skills.microchip.com/programming-microchip-pic16f-using-mcc-volume-1>)

Introducción al PIC® MCU Temporizador 0 (<https://skills.microchip.com/introduction-to-the-8-bit-pic-mcu-timer0>)

Introducción a la arquitectura MCU mejorada PIC16F1 (<https://skills.microchip.com/introduction-to-the-pic16f1-enhanced-mcu-architecture>)

? Preguntas frecuentes

Tema

¿Por qué se prueba INTCONbits.PEIE en el código de interrupción de alta prioridad generado por MPLAB® Code Configurator en el PIC18F26K22? (/faq:3388)

Deshabilitar la detección de cruce por cero (ZCD) (/faq:155)

¿Por qué el PIC16F1619 DAC1OUT en una placa Curiosity siempre está a 0 V en modo de depuración? (/faq:335)

¿Por qué obtengo un error de sintaxis C al compilar una instrucción de rotación a la izquierda `asm("RLF aux, w");?` (/faq:276)

¿Por qué no puedo manejar mi MOSFET cuando estoy conectado a PWM? (/faq:291)

¿Cuál es el valor de resistencia del pull-up débil interno? (/faq:444)

¿Cuál es la protección interna proporcionada por el silicio, como los diodos de descarga? (/faq:1219)

¿Cuál es el tamaño de instrucción de una MCU PIC®? (/faq:1156)

¿Qué es Q1 en un comparador PIC16F? (/faq:314)

¿Qué significa el sufijo "-ICD" en una MCU? (/faq:331)

¿Cuáles son los microcontroladores PIC® básicos que se utilizan en la serie ARxxx? (/faq:381)

La señal de salida TX de USART muestra dos bits de parada (/faq:572)

PIC16F870/871 - ¿Cuál es el procedimiento de reinicio después del evento de error de marco (FERR)? (/faq:3014)

PIC16F - No se puede cambiar la funcionalidad del pin CLKIN (/faq:2705)

¿Las tablas de pines del dispositivo muestran qué pines son tolerantes a 5V? (/faq:2836)

Prueba de paquetes TSOP, SSIC, QFN, DFN en una placa de pruebas (/faq:3193)

¿Cómo puedo crear un amplificador de ganancia programable (PGA) usando un amplificador operacional y una salida DAC desde una MCU PIC®? (/faq:358)

¿Cuál es la diferencia entre la resistencia de bytes (ED) y el número total de ciclos de borrado/escritura antes de actualizar (TREF)? (/faq:2746)

PIC18F87xx - ¿Cómo se selecciona la funcionalidad de un pin? (/faq:1137)

Impacto de Ripple en Vdd cuando se usa un microcontrolador Microchip (/faq:1125)

I²C no puede escribir en SSPBUF (/faq:1773)

¿Cómo se lee la temperatura interna en el PIC18F? (/faq:2479)

¿Cómo depuro el PIC16F1503 de un Curiosity Board? (/faq:925)

La salida de PWM es baja mientras se configura el CCP (/faq:2737)

PIC18F25K80 - I²C deja de funcionar en un sistema controlado por interrupción (/faq:1892)

¿Cómo se lee la memoria EEPROM de un PIC18F46J50? (/faq:2983)

¿Cómo puedo encontrar un microcontrolador compatible pin a pin para migrar mi proyecto? (/faq:2721)

Ejemplo de código ECAN utilizando el modo FIFO mejorado (modo 2) (/faq:686)

Controle el RTCC utilizando una entrada de reloj digital externa de 32,768 kHz (/faq:2588)

Significado del sufijo de la parte personalizada (/faq:1143)

Controlador LED de corriente constante con PIC18 (/faq:1022)

Calcule los valores del condensador para un oscilador de cristal (/faq:937)

¿El sensor de temperatura interno de los dispositivos PIC® puede medir la temperatura ambiente? (/faq:1225)

Implementación de la interfaz SMI en un PIC® de 8 bits (/faq:3309)

Arranque lento del microcontrolador PIC® en temperaturas bajo cero (/faq:2828)

PIC10F - Error de memoria de calibración no válida (/faq:1689)

¿Cómo se calcula la frecuencia de muestreo para un ADC SAR en MCU PIC16F? (/faq:3332)

Cambiar la velocidad del oscilador durante el tiempo de ejecución en el PIC16F (/faq:3339)

¿Puede el RTCC en una MCU PIC® sin un VBAT funcionar con una batería? (/faq:360)

¿Cuál es el problema más común que se encuentra al comunicarse con un módulo PIC® MCU SPI? (/faq:3282)

Necesito encontrar un dispositivo con más RAM (/faq:1159)

Migración a una nueva familia PIC16 (/faq:3301)

VCAP externo en PIC16F (/faq:2686)

Los pines EUSART no funcionan en PORTA (/faq:1761)

¿Puede funcionar un PIC16F con USB en el oscilador interno? (/faq:3122)

¿Puede el ADC medir 5 V si el VDD es de 3,3 V? (/faq:1152)

¿Puedo reprogramar la calibración del oscilador en un dispositivo PIC®? (/faq:2941)

¿Por qué necesito usar un amplificador de ganancia unitaria para medir el voltaje de salida DAC de un PIC16F1xxx? (/faq:3054)

¿Dónde puedo encontrar ejemplos de código para un controlador PID usando el acelerador matemático PIC16F161x? (/faq:3116)

¿Puedo conectar VUSB3V3 a VDD (+3,3 V) para la operación USB en el PIC16F1455? (/faq:393)

¿Se pueden usar UART e I²C en PIC12F1840 o PIC16F1829? (/faq:233)

Conexión de pines NC (/faq:281)

¿Se puede leer CRC desde un PIC protegido por código? (/faq:1043)

¿Dónde puedo encontrar la referencia de voltaje ADC? (/faq:3277)

Adición de un número de serie a un PIC18 (/faq:2787)

Nota de la aplicación sobre los fundamentos de LCD y el módulo de controlador de LCD para MCU PIC® de 8 bits (/faq:3333)

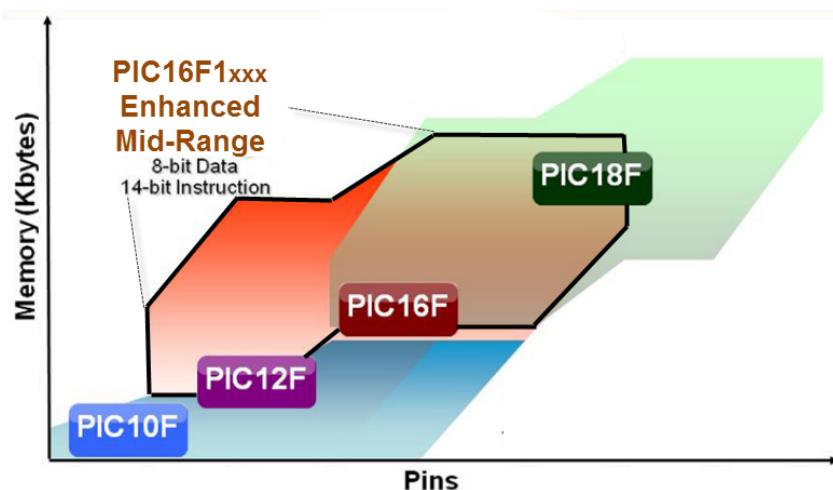
¿Por qué no puedo depurar con un PIC16F18877? ¿Por qué funciona en modo de lanzamiento pero no en modo de depuración? (/faq:3402)

¿Cuáles son las diferencias entre las subfamilias de microcontroladores PIC® de 8 bits? (/faq:70)

Descripción general de la arquitectura MCU PIC de gama media mejorada

1 Resumen

Familias de MCU de 8 bits de Microchip

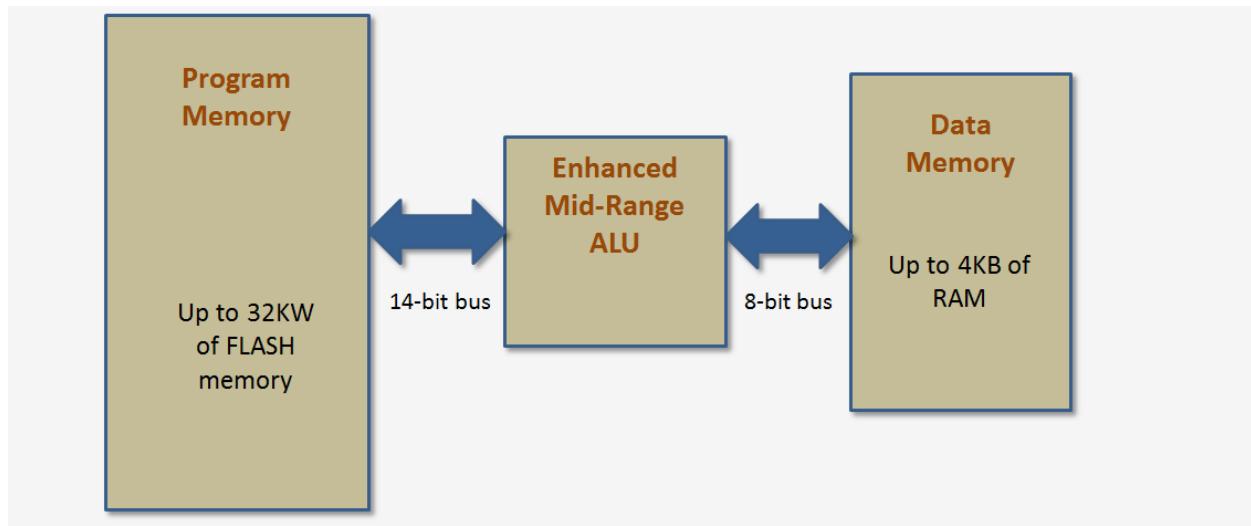


La familia de MCU PIC® de 8 bits de rango medio mejorado PIC16F1xxx abarca una amplia variedad de tamaños de memoria y pines de E/S.

Esta página presenta las características arquitectónicas clave de la familia de MCU PIC16F1xxx. En esta página se proporcionan enlaces a los detalles técnicos necesarios para implementar aplicaciones en la familia de microcontroladores PIC de gama media mejorada.

Arquitectura de Harvard

Los microcontroladores PIC® de gama media mejorados utilizan una arquitectura Harvard de doble bus.



autobús 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 instrucción, se lee una palabra de programa de 14 bits en la ALU.

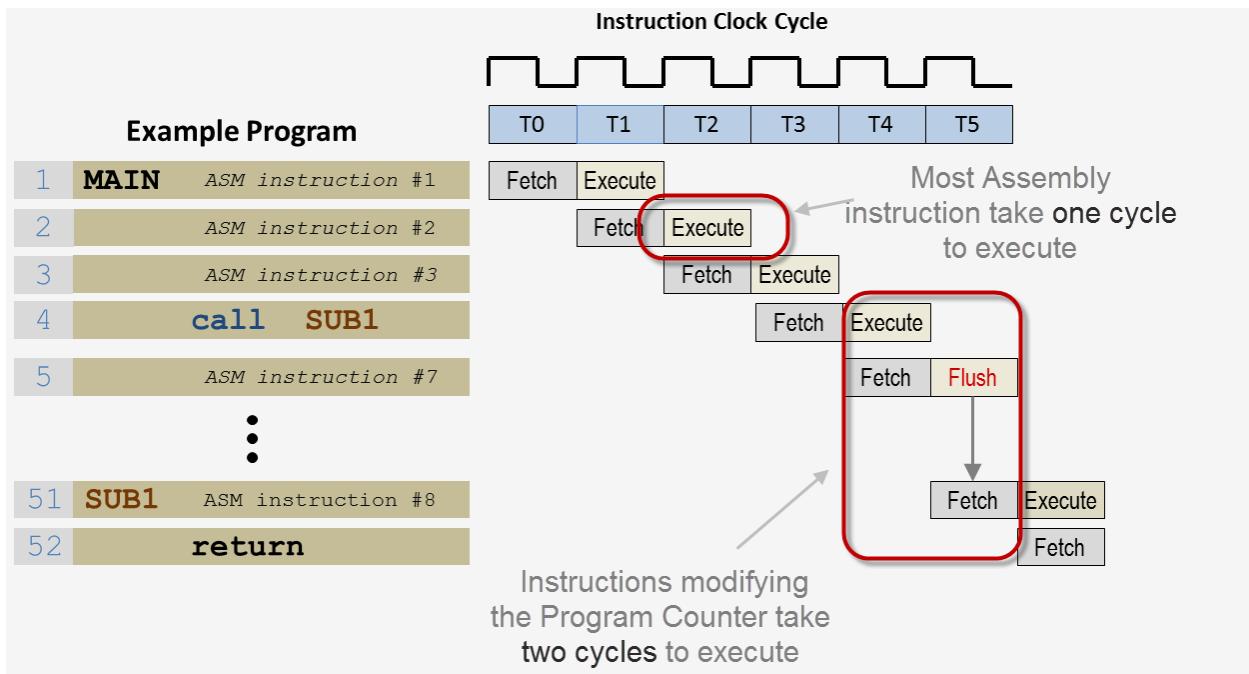
Bus de datos

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

Canalización de instrucciones

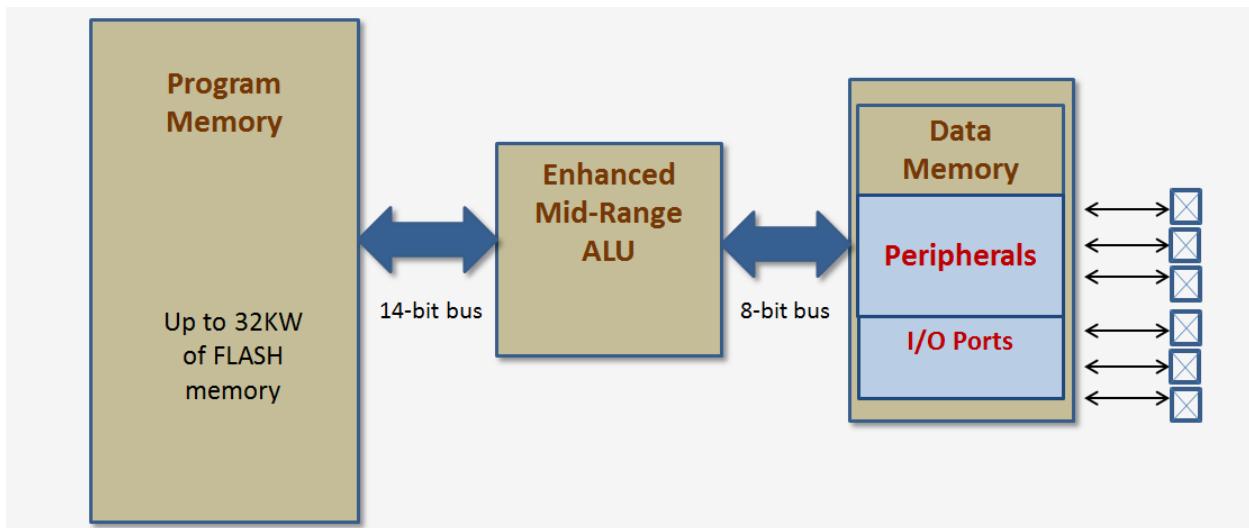
La arquitectura de doble bus del PIC de rango medio mejorado proporciona una línea de instrucción de dos etapas. Una cada ciclo de reloj ejecuta dos fases de instrucción:

1. La siguiente instrucción se **"obtiene"** 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 la memoria

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



La página de memoria ([/mcu1102:data-memory](#)) de datos del tutorial mejorado de gama media ofrece una descripción completa junto con ejemplos de programas que acceden a los periféricos asignados a la memoria.

Conjunto de instrucciones ortogonales

Cada MCU PIC® de gama media mejorada tiene 49 instrucciones. Las instrucciones que acceden directamente a las direcciones de la memoria de datos se ejecutan en un ciclo de instrucción. Las instrucciones que provocan un cambio en el contador del programa BRA, GOTO, RETURN, CALL, ..etc) tardan dos ciclos de instrucción en ejecutarse.

Al mapear los registros de E/S y periféricos a direcciones de memoria, las MCU PIC no necesitan instrucciones especiales para las operaciones de E/S o para establecer registros periféricos. Escribir en un puerto de E/S o configurar un periférico es una simple escritura en una ubicación de memoria. Leer el valor de un pin de entrada, registro de resultado ADC o temporizador es una simple lectura de una ubicación de memoria. Mediante el uso de una pequeña cantidad 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 |
|-------------------------------------|------------------|---|
| WRITE a value to a variable | var1 = 7 ; | movlw 7 movf var1 |
| WRITE a value to a Register | T1CON = 0x72; | movlw 0x72 movf T1CON |
| WRITE to an Output Latch | LATD = 0xFF ; | movlw 0xFF movf LATD |
| READ an input PORT | var1 = PORTB; | movf PORTB movwf var1 |
| READ a variable's value | var1 = var2; | movf var2 movwf var1 |
| READ an SFR into another SFR | CCPR1L = ADRESH; | movf ADRESH movwf CCPR1L |



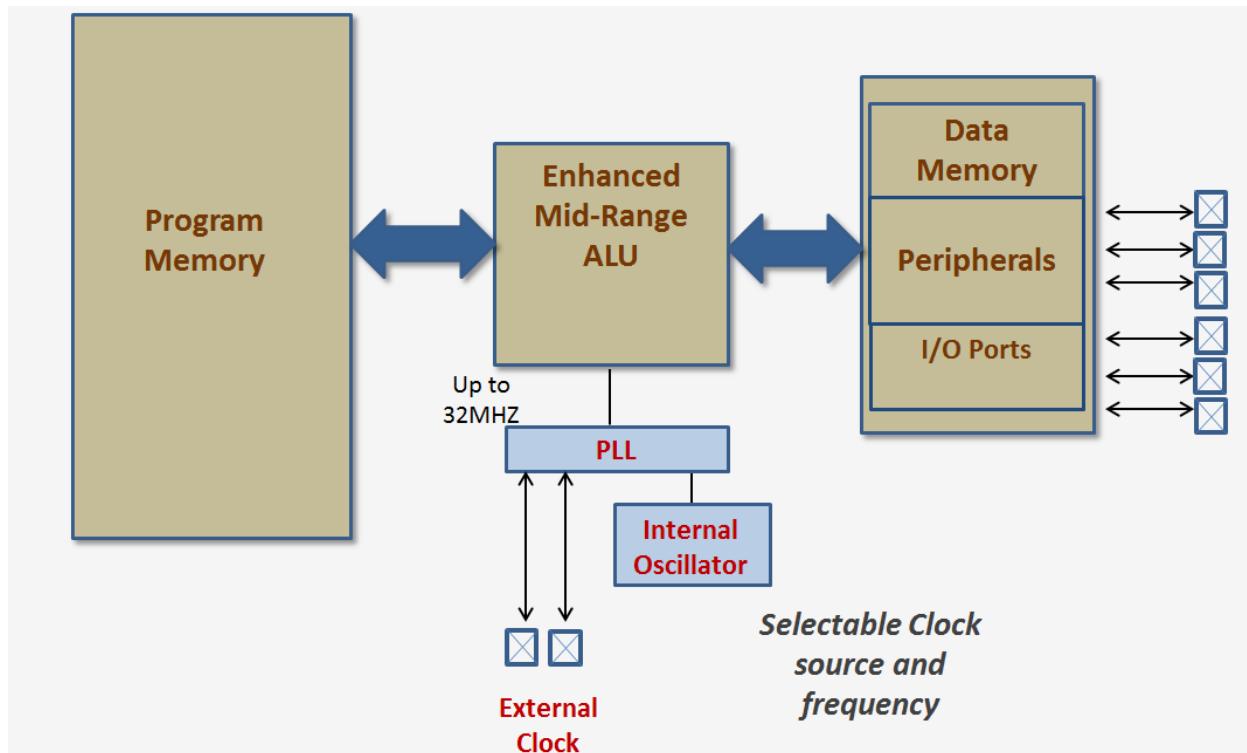
Para obtener una lista detallada del conjunto de instrucciones y una discusión completa del tiempo de instrucción, consulte la página **Conjunto (/mcu1102:instruction-set)** de instrucciones del tutorial de rango medio mejorado.

Opciones de reloj flexibles (hasta 32 MHz)

Seleccionado por los **bits de configuración (/mcu1102:configuration-bits)** del PIC® MCU, el reloj del sistema tiene las siguientes propiedades: **(/mcu1102:configuration-bits)**

- Fuente opcional (**oscilador interno** o **circuito externo**)

- Opciones de velocidad flexibles **hasta 32MHz**
- **Arranque de dos velocidades** : permite que el sistema ejecute el software de inicialización mientras el oscilador externo se estabiliza
- **Cambio** 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 fallas: cambia al oscilador interno en caso de falla del reloj externo



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

E/S digitales

Casi todos los pines de la MCU PIC Enhance de rango medio se pueden usar como pines de entrada o salida digital. Los pines digitales comparten estos atributos:

- Supervisión de entradas digitales
- Controlar dispositivos digitales
- Dominadas débiles internas
- Multiplexado con Periféricos
- Alta capacidad de accionamiento (hasta 25 mA sumidero/fuente en muchos pines de E/S)
- Manipulación directa de bit de ciclo único
- Diodos de protección ESD de 4kV

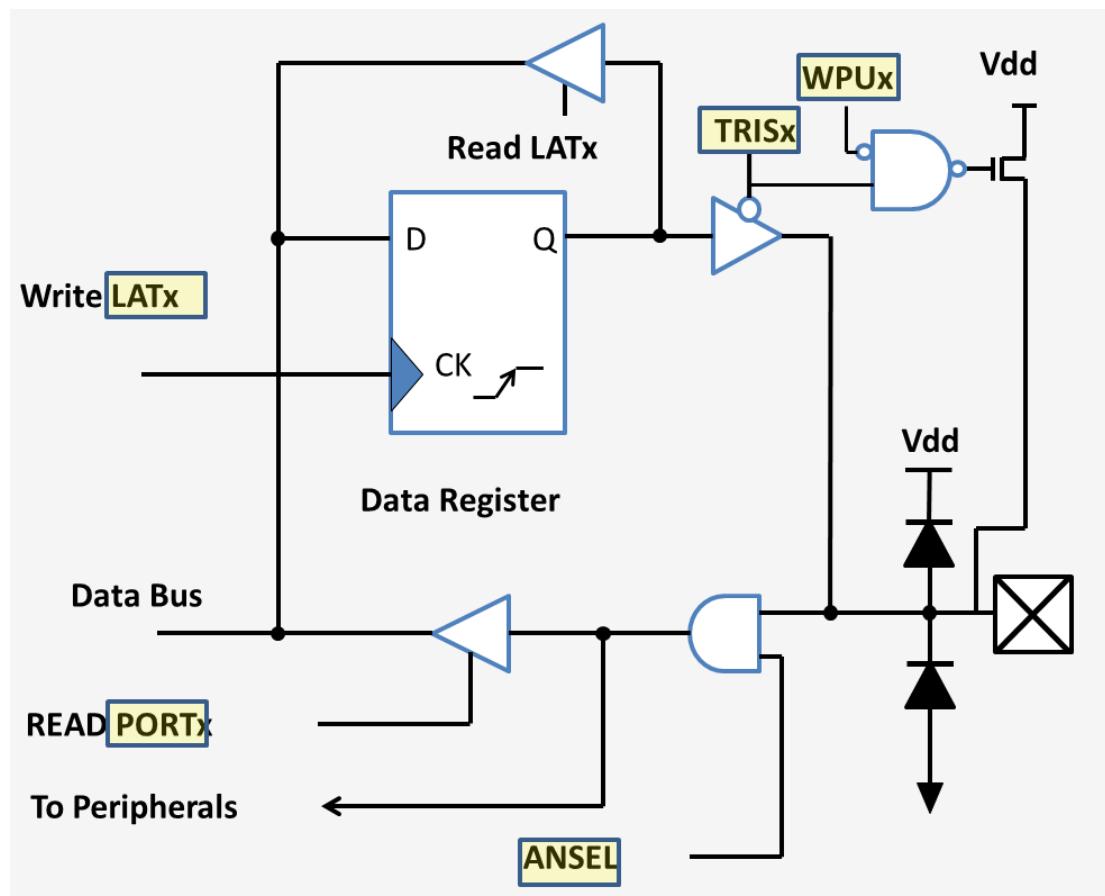
Al reiniciar:

- 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 de un pin digital. Estos registros de 8 bits controlan 8 pines de un PUERTO. Usando los registros `TRISx`, `PORTx`, `LATx`, `WPUx` y `ANSEL`, el programa puede:

- Configure el pin a como entrada o salida `TRISx`
- Leer un pin de entrada (o los 8 pines PORT) `PORTx`
- Envía un 1 o 0 a un pin `LATx`
- Habilitar o deshabilitar la resistencia pull up interna `WPUx`
- Determine si los pines con capacidad analógica funcionan en modo analógico o digital `ANSEL`



Para una discusión completa de la E/S digital PIC mejorada de rango medio, incluidos los detalles sobre la programación de las operaciones de entrada y salida digital, consulte la sección de [E \(/mcu1102:digital-io\)](#) /S digital de este tutorial mejorado de rango medio.

Pines multiplexados

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

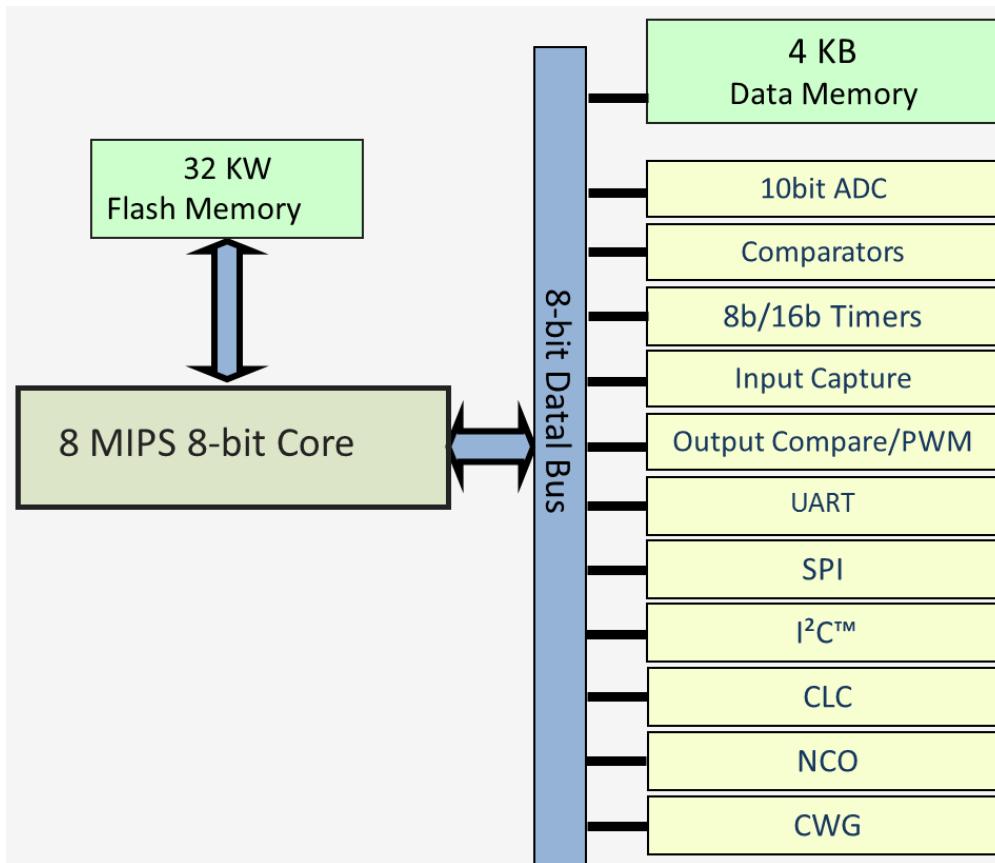
| | | | |
|----|---|---|-------------------------------------|
| 40 | □ | ↔ | RB7/ICSPDAT/ICDDAT/SEG13 |
| 39 | □ | ↔ | RB6/ICSPCLK/ICDCLK/SEG14 |
| 38 | □ | ↔ | RB5/AN13/CPS5/CCP3/P#A/T1G/COM1 |
| 37 | □ | ↔ | RB4/AN11/CPS4/COM0 |
| 36 | □ | ↔ | RB3/AN9/C12IN2-/CPS3/CCP2/P2A/VLCD3 |
| 35 | □ | ↔ | RB2/AN8/CPS2/VLCD2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



Para una discusión completa de la E/S digital PIC mejorada de rango medio, incluidos los detalles sobre la configuración de los pines, consulte la [sección Periféricos \(/mcu1102:peripherals\)](#) del tutorial de rango medio mejorado.

Periféricos avanzados

Además de las E/S digitales mejoradas, los miembros de la familia de MCU PIC de rango medio tienen una variedad de periféricos avanzados. Estos periféricos incluyen periféricos para conversión de datos, comunicación y acondicionamiento de señales.



Para obtener una lista completa de los periféricos disponibles, consulte la [sección Periféricos \(/mcu1102:peripherals\)](#) del tutorial .

Interrupciones

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

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

El PIC de rango medio mejorado utiliza una pila de hardware de 16 niveles para almacenar el contenido actual de la PC cuando ocurre una interrupción. El contexto del programa se guarda en registros 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 la fuente de la interrupción, luego realiza la tarea necesaria para dar servicio al periférico interrumpido. La instrucción final de un ISR es la instrucción Return From Interrupt (RETFIE).

Guardado automático de contexto

Los siguientes registros se guardan en un registro de sombra de un solo nivel establecido en caso de una interrupción

- registro W
- BSR
- ESTADO
- FSR
- PCLATH

Cuando el ISR ejecuta la instrucción RETFIE, estos registros se restauran al valor anterior a la interrupción.

Prioridad de interrupción de un solo nivel

Cuando ocurre una interrupción, el bit de habilitación de interrupción global (GIE) en el registro de estado está deshabilitado. Esto evitirá que una interrupción sea reemplazada por otra interrupción.

Upon executing a RETFIE the state of the GIE control bit is restored to its pre-interrupt value.



For a complete description of the interrupt process and programming examples please refer to the [Interrupt Section \(/mcu1102:interrupts\)](#) of the Enhanced Mid-range tutorial.

What happens at System Start-up (RESET)

There are several sources of a RESET on the enhanced mid-range PIC MCU. The RESET sources common to almost all applications are the Power On Reset (POR) and Brown Out Reset (BOR) due to a sagging power supply voltage (i.e. brown-out). There are several other methods for resetting the MCU including Watchdog timeout and directly accessing the MCLR pin.

Program Counter is set to 0x00.

After a RESET the instruction located at address 0 is the first instruction executed. The application developer is responsible for placing code into this address to 'boot up' the PIC. Microchip's MPLAB® XC8 compiler will insert the appropriate instructions to start-up the PIC and transfer control to `main`. Assembly level programmers will have to write the code to initialize the PIC and jump past the Interrupt Vector located at address 0x04.

All Special Function Registers are set to a Default Value

The datasheet for each enhanced mid-range PIC MCU shows the values which the registers will contain at 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



The **Programming Examples** ([/mcu1102:programming-examples](#)) section of the enhanced mid-range tutorial provides programming examples of how to 'boot-up' the MCU.

Conjunto de instrucciones de rango medio mejorado

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

Operaciones orientadas a bytes

| Mnemónicos, Operandos | Descripción | Ciclos | Código de operación de 14 bits MSb.....LSb | Estado Afectado | notas |
|-----------------------|--|--------|---|-----------------|-------|
| ADDWF f, d | Añadir W y f | 1 | 00 0111 ffff ffff | C, CC, Z | 2 |
| AÑADIRWFC f, d | Suma con Carry W y f | 1 | 11 1101 ffff ffff | C, CC, Z | 2 |
| ANDWF f, d | Y W con f | 1 | 00 0101 ffff ffff | Z | 2 |
| ASRF f, d | Desplazamiento aritmético a la derecha | 1 | 11 0111 ffff ffff | C, Z | 2 |
| LSLF f, d | Desplazamiento lógico a la izquierda | 1 | 11 0101 ffff ffff | C, Z | 2 |
| LSRF f, d | Desplazamiento lógico a la derecha | 1 | 11 0110 ffff ffff | C, Z | 2 |
| CLRF F | Borrar f | 1 | 00 0001 1fffffff | Z | 2 |
| CLRW | Borrar W | 1 | 00 0001 0000 00xx | Z | |
| COMF f, d | Complemento f | 1 | 00 1001 ffff ffff | Z | 2 |
| DECf f, d | Decremento f | 1 | 00 0011 ffff ffff | Z | 2 |
| FIN f, d | Incremento f | 1 | 00 1010 ffff ffff | Z | 2 |

| | | | | | | |
|--------|---------|-------------------------------------|---|-------------------|--------|---|
| IORWF | f, d | Inclusivo OR W con f | 1 | 00 0100 dfff ffff | Z | 2 |
| MOVF | f, d | Move f | 1 | 00 1000 dfff ffff | Z | 2 |
| MOVWF | f | Move W to f | 1 | 00 0000 1fff ffff | None | 2 |
| RLF | f,d | Rotate left f through Carry | 1 | 00 1101 dfff ffff | C | 2 |
| RRF | f,d | Rotate right f through Carry | 1 | 00 1100 dfff ffff | C | 2 |
| SUBWF | f,d | Subtract with Borrow W from f | 1 | 11 1011 dfff ffff | C,DC,Z | 2 |
| SUBWFB | f,d | Subtract W from f | 1 | 00 0010 dfff ffff | C,DC,Z | 2 |
| SWAPF | f,d | Swap nibbles in f | 1 | 00 1110 dfff ffff | None | |
| XORWF | f,d | Exclusive OR W with f | 1 | 00 0110 dfff ffff | Z | 2 |

Byte Oriented Skip Instructions

| Mnemonic, Operands | Description | | Cycles | 14-bit Opcode MSb.....LSb | Status Affected | Notes |
|-----------------------|-------------|---------------------------|--------|------------------------------|--------------------|-------|
| DECFSZ | f,d | Decrement f, Skip if 0 | 1(2) | 00 1011 dfff ffff | None | 1,2 |
| INCFSZ | f,d | Increment f, Skip if 0 | 1(2) | 00 1111 dfff ffff | None | 1,2 |

Bit Oriented File Register Operations

| Mnemonic, Operands | Description | | Cycles | 14-bit Opcode MSb.....LSb | Status Affected | Notes |
|-----------------------|-------------|-------------|--------|------------------------------|--------------------|-------|
| BCF | f,b | Bit Clear f | 1 | 01 00bb bfff ffff | None | 2 |
| BSF | f,b | Bit Set f | 1 | 01 01bb bfff ffff | None | 2 |

Bit Oriented Skip Operations

| Mnemonic, Operands | Description | | Cycles | 14-bit Opcode MSb.....LSb | Status Affected | Notes |
|-----------------------|-------------|------------------------------|--------|------------------------------|--------------------|-------|
| BTFSC | f,b | Bit Test f, Skip if Clear | 1(2) | 01 10bb bfff ffff | None | 1,2 |
| BTFSS | f,b | Bit Test f, Skip if Set | 1(2) | 01 11bb bfff ffff | None | 1,2 |

Literal Operations

| Mnemonic, Operands | | Description | Cycles | 14-bit Opcode MSb.....LSb | Status Affected | Notes |
|-----------------------|---|-----------------------------|--------|------------------------------|--------------------|-------|
| ADDLW | k | Add literal and W | 1 | 11 1110 kkkk kkkk | C,DC,Z | |
| ANDLW | k | AND literal with W | 1 | 11 1001 kkkk kkkk | Z | |
| IORLW | k | Inclusive OR literal with W | 1 | 11 1000 kkkk kkkk | Z | |
| MOVLB | k | Move literal to BSR | 1 | 00 0000 001k kkkk | None | |
| MOVLP | k | Move literal to PCLATH | 1 | 11 0001 1kkk kkkk | None | |
| MOVLW | k | Move literal to W | 1 | 11 0000 kkkk kkkk | None | |
| SUBLW | k | Subtract W from literal | 1 | 11 1100 kkkk kkkk | C,DC,Z | |
| XORLW | k | Exclusive OR literal with W | 1 | 11 1010 kkkk kkkk | Z | |

Control Operations

| Mnemonic, Operands | | Description | Cycles | 14-bit Opcode MSb.....LSb | Status Affected | Notes |
|-----------------------|---|----------------------------|--------|------------------------------|--------------------|-------|
| BRA | k | Relative Branch | 2 | 11 001k kkkk kkkk | None | |
| BRW | | Relative Branch with W | 2 | 00 0000 0000 1011 | None | |
| CALL | k | Call Subroutine | 2 | 10 0kkk kkkk kkkk | None | |
| CALLW | | Call Subroutine with W | 2 | 00 0000 0000 1010 | None | |
| GOTO | k | Goto address | 2 | 10 1kkk kkkk kkkk | None | |
| RETFIE | k | Return from interrupt | 2 | 00 0000 0000 1001 | None | |
| RETLW | k | Return, place literal in W | 2 | 11 0100 kkkk kkkk | None | |
| RETURN | k | Return from subroutine | 2 | 00 0000 0000 1000 | None | |

Inherent Operations

| Mnemonic, Operands | Description | Cycles | 14-bit Opcode | Status Affected | Notes |
|-------------------------------|-----------------------------|---------------|----------------------|----------------------------|--------------|
| | | | MSb.....LSb | | |
| CLRWDT | Clear Watchdog Timer | 1 | 00 0000 0110 0100 | TO, PD | |
| NOP | No Operation | 1 | 00 0000 0000 0000 | None | |
| OPTION | Load OPTION register with W | 1 | 00 0000 0110 0010 | None | |
| RESET | Software device Reset | 1 | 00 0000 0000 0001 | None | |
| SLEEP | Go into standby mode | 1 | 00 0000 0110 0011 | TO, PD | |
| TRIS f | Load TRIS register | 1 | 00 0000 0110 0fff | None | |

C-Compiler Optimized

| Mnemonic, Operands | Description | Cycles | 14-bit Opcode | Status Affected | Notes |
|-------------------------------|-------------------------|---------------|----------------------|----------------------------|--------------|
| | | | MSb.....LSb | | |
| ADDFSR | Add Literal to FSRn | 1 | 11 0001 0nkk kkkk | None | |
| MOVIW | Move Indirect FSRn to W | 1 | 00 0000 0001 0nnn | Z | 2 |
| MOVWI | Move W to Indirect FSRn | 1 | 00 0000 0001 1nnnn | Z | 2 |

Notes

1. If the Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.
2. If this instruction addresses an INDF register *and* the MSb of the corresponding FSR is set, the instruction requires one additional instruction cycle.

Bits de configuración PIC16F1xxx de rango medio mejorado 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 la MCU PIC con el código de la aplicación. No son código ejecutable ya que el Contador de programa (PC) no puede acceder a su dirección. Cuando se programan en una MCU PIC, los bits de configuración completan el circuito que activa o desactiva las funciones de hardware de la MCU.

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 controladas por los bits de configuración incluyen:

1. Reloj del sistema
2. Administración 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 funciones están controladas por bits de configuración y cómo generarlas en el código fuente.



Los bits de configuración y las configuraciones para dispositivos PIC16F1xxx individuales pueden variar. Consulte su hoja de datos para conocer 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 están ubicadas fuera del alcance de la PC en las direcciones `0x8007` y `0x8008` en la memoria flash de la 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 |

(/local--files/8bit:emr-configuration-bits/config1.png)

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 |
|-------|---------|---------|-----|-----|---------|---------|
| — | VCAPEN1 | VCAPEN0 | — | — | WRT1 | WRT0 |
| bit 6 | | | | | | bit 0 |

(/local--files/8bit:emr-configuration-bits/config2.png)

Ajustes de configuración de la hoja de datos de PIC16F1937

Los bits de configuración se insertan en el código fuente de la aplicación. Cuando se construye un proyecto PIC MCU, los ajustes 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

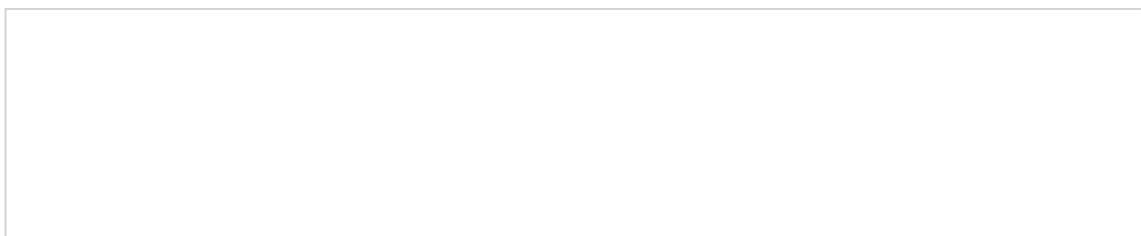
El compilador MPLAB® XC8 C de Microchip acepta directivas `#pragma` para establecer los bits de configuración.

{...}

La sintaxis para generar bits de configuración:

`#pragma config CONFIG_BIT_NAME = CONFIG_VALUE`

Ejemplo de establecimiento 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
```



Cuando se configuran los bits de configuración usando C, no es necesario conocer la palabra que contiene el bit que se está configurando. Todo lo que se necesita es el nombre del bit de configuración y el valor deseado.



Los archivos de encabezado para cada dispositivo PIC16F1xx contienen `CONFIG_BIT_NAME` y `CONFIG_VALUE`. Puede encontrar una lista completa de los ajustes en el resumen de bits de configuración ([/8bit:emr-summary-configbit-in-c](#)).

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

1. En un archivo fuente 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 ([/mplabx:view-and-set-configuration-bits](#)) del tutorial MPLAB X para ver accesos directos en la generación del código necesario para establecer los bits de configuración.

Generación de bits de configuración en ensamblaje

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.

En el archivo `.INC` de cada MCU de PIC se incluye un patrón de 14 bits para cada configuración de bit de configuración individual. El valor de 14 bits que se pasa a la directiva `CONFIG` se genera mediante la combinación lógica AND de la configuración de bits en el archivo `.INC`.

El código para generar y cargar los bits de configuración se puede colocar en cualquier archivo de origen de ensamblado del 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:

| FOSC<2:0> | Ajuste | reloj agrio |
|------------------------|---------------|---|
| 0 0 0 | INTOSC | oscilador interno |
| 0 0 1 | EXTRC | Oscilador RC externo |
| 0 1 0 | SA | Oscilador de cristal externo de alta velocidad |
| 0 1 1 | XT | Oscilador de cristal externo |
| 1 0 0 | LP | 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 fallas (FCMEN)

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

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

Las opciones para **FCMEN** son:

| FCMEN | Ajuste | Función de monitor de reloj a prueba de fallas |
|--------------|---------------|---|
| | | |

| | | |
|---|---------|-------------|
| 0 | APAGADO | Desactivado |
| 1 | EN | Activado |

Ejemplo: Habilitación del monitor de reloj a prueba de fallas



```
1 #include <xc.h>
2 #pragma config FCMEN = ON // E
```

Conmutación interna/externa (IESO)

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

IESO establece el modo de cambio de reloj y arranque de dos velocidades. Con **IESO** habilitado, la fuente del reloj puede controlarse mediante el programa de aplicación.

Las opciones para **IESO** son:

| IESO | Ajuste | Función de puesta en marcha de dos velocidades |
|-------------|---------------|---|
| 0 | APAGADO | Desactivado |
| 1 | EN | Activado |

Ejemplo: Habilitación del arranque de dos velocidades



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

Habilitación de salida de reloj (CLKOUTEN)

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

CLKOUTEN permite que el pin **OSCx/CLK0UT** emita el reloj del sistema interno. Esto permite que el reloj del sistema PIC16F1xxx controle otros componentes.

Las opciones para CLKOUTEN son:

| CLKOUTEN | Ajuste | Función CLKOUT |
|-----------------|---------------|---|
| 0 | EN | Fosc se enviará a OSCx/CLK0UT |
| 1 | APAGADO | OSCx/CLK0UT será el oscilador o una función periférica |

Ejemplo: salida del reloj 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 (PLL) de 4 X 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:

| PLÉN | 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 de la MCU PIC que está utilizando para determinar los ajustes de configuración de PLL específicos.

Ejemplo: Habilitación del bucle de bloqueo de fase interno



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

Administración de energía

Habilitación de restablecimiento de caída de tensión (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 ocurra un REINICIO DE MCU 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 B0RV .

Hay cuatro opciones para los dos bits BOREN :

1. Brown-out Reset siempre está habilitado
2. Brown-out Reset siempre está deshabilitado

3. Brown-out Reset está habilitado cuando se está ejecutando pero deshabilitado cuando MCU ingresa al modo SLEEP
4. Brown-out Reset es controlado en tiempo de ejecución por el bit **SB0REN** del registro PCON.

| ABURRIDO | Ajuste | Función de reinicio de oscurecimiento |
|-----------------|---------------|--|
| 1 1 | EN | Siempre habilitado |
| 0 0 | APAGADO | Siempre deshabilitado |
| 1 0 | NSLEEP | Habilitado mientras está ACTIVO, deshabilitado en modo SLEEP |
| 0 1 | SBODEN | controlado por el bit SB0REN del registro PCON |

Ejemplo: Desactivación de RESET de Brown-out



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

Nivel de voltaje de caída de voltaje (BORV)

B0RV es un único bit de configuración que reside en **CONFIG2** .

B0RV solo es aplicable cuando el reinicio de Brown-out (controlado por el bit de configuración **B0REN**) está activo.

B0RV selecciona uno de los dos niveles de voltaje preestablecidos como el voltaje de restablecimiento de Brown-out.

Los niveles de voltaje establecidos por **B0RV** son:

| B0RV | Ajuste | Voltaje de caída de tensión |
|-------------|---------------|------------------------------------|
| 0 | L0 | 1,9 voltios |
| 1 | HOLA | 2,5 voltios |



El nivel de voltaje de caída de voltaje establecido por **B0RV** puede variar según la MCU de PIC en particular. Consulte la hoja de datos para conocer las opciones de voltaje de reducción de voltaje.

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



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

Ejemplo: habilite el REINICIO de Brown-out a 1,9 V mientras la MCU está ACTIVA y deshabilite Brown-out en el modo SLEEP.



```
1 #include <xc.h> ?  
2 #pragma config BOREN = NSLEEP  
3 #pragma config BORV = L0 ▾
```

Regulador de Voltaje (VCAPEN - LDO)

El ajuste de 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 pin de **tapa en V**.

Los ajustes para **VCAPEN** son:

| VCAPEN | Ajuste | Función |
|--------|---------|--|
| 0 0 | RA6 | RA6 se asigna como tapa V |
| 0 1 | RA5 | RA5 se asigna como tapa V |
| 1 0 | RA0 | RA0 se asigna como tapa V |
| 1 1 | APAGADO | La tapa en V está desconectada de todos los pines |



No todas las MCU PIC16F1xxx tienen un regulador LDO interno. Consulte la hoja de datos de la MCU PIC que está utilizando para determinar si hay un LDO y qué pines están disponibles como **V_{cap}**.

Ejemplo: Asignación de **V_{cap}** a RA6



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

Deshabilitar la funcionalidad de **tapa en V**



```
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 que las MCU PIC16F1xxx se programen con solo V_{dd} . El uso de LVP elimina la necesidad de suministrar un voltaje superior a V_{dd} en $MCLR/V_{pp}$.

Los ajustes para LVP son:

| LVP | Ajuste | Función de programación |
|------------|---------------|--------------------------------|
| 1 | EN | Activado |
| 0 | APAGADO | Desactivado |

Ejemplo: deshabilitar las lecturas de EEPROM desde 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 la EEPROM protegida. El contenido de EEPROM todavía está disponible para lecturas desde fuentes internas.

Los ajustes para CPD son:

| DPC | Ajuste | Función de protección de lectura de EEPROM |
|------------|---------------|---|
| 0 | EN | Habilitado - EEPROM no es legible |
| 1 | APAGADO | Deshabilitado - EEPROM es legible |

Ejemplo: deshabilitar las lecturas de EEPROM desde fuentes externas



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

Protección de lectura de memoria de programa (\overline{CP})

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

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

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

Los ajustes para **CP** son:

| PC | Ajuste | Función de protección de lectura |
|-----------|---------------|---|
| 0 | EN | Habilitado - la memoria no es legible |
| 1 | APAGADO | Deshabilitado - la memoria es legible |

Ejemplo: deshabilitar las lecturas de la memoria del programa externo



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

Activación de autoescritura 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.

Los ajustes para **WRT** son:

| WRT | Ajuste | Funcionalidad Flash Self Right |
|------------|---------------|---|
| 1 1 | TODOS | El programa de usuario no puede escribir en la 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 | El programa de usuario puede escribir en todas las direcciones de memoria |



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

Características de funcionamiento

Control de clavijas para MCLR (MCLRE)

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

MCLRE controla la función del pin MCLR /V_{pp} .

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

Los ajustes para MCLRE son:

| MCLRE | Ajuste | Función |
|-------|---------|--|
| 1 | EN | La función de pin es MCLR /V _{pp} con pull-up débil interno habilitado |
| 0 | APAGADO | La función de pin es una entrada digital con pull-up interno controlado por WPUx |

Ejemplo: Dejar MCLR /V_{pp} como pin RESET.



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

Haciendo MCLR /V_{pp} 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 reinicio de encendido o un reinicio de apagón para permitir que V_{dd} se estabilice. La activación o desactivación de este retraso está controlada por PWRTE .

Las opciones para PWRTE son:

| PWRTE | Ajuste | Función de temporizador de encendido |
|-------|---------|--------------------------------------|
| 0 | EN | Desactivado |
| 1 | APAGADO | Activado |

Ejemplo de activación del temporizador de encendido



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

Habilitación del temporizador de vigilancia (WDTE)

El ajuste de 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 REINICIO de MCU si el temporizador de vigilancia interno pasa de **0xFF** a **0x00** antes de que la MCU pueda ejecutar una instrucción **CLRWDT**.

Hay cuatro opciones para los dos bits **WDTE**:

1. El restablecimiento del temporizador de vigilancia siempre está habilitado
2. El restablecimiento del temporizador de vigilancia siempre está deshabilitado
3. El restablecimiento del temporizador de vigilancia está habilitado cuando se está ejecutando pero deshabilitado cuando la MCU ingresa al modo SUSPENSIÓN
4. El restablecimiento del temporizador de vigilancia se controla en tiempo de ejecución mediante el bit **SWDTEN** del registro **WDTCON**

| WDTE | Ajuste | Función de reinicio del perro guardián |
|-------------|---------------|--|
| 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 bit SBOREN de registro PCON |

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 REINICIO en un desbordamiento o subdesbordamiento de la pila.

Un desbordamiento o subdesbordamiento de pila siempre establece el bit **STK0VF** o **STKUNF** en el registro **PCON** independientemente del valor de **STVREN**.

Las opciones para **STVREN** son:

| STVREN | Ajuste | Función |
|---------------|---------------|----------------|
| | | |

| | | |
|---|---------|---|
| 1 | EN | El REINICIO del flujo excesivo/insuficiente de la pila está HABILITADO |
| 0 | APAGADO | El REINICIO del flujo excesivo/insuficiente de la pila está DESHABILITADO |



Los bits de configuración y los ajustes para dispositivos PIC16F1xxx individuales pueden ser diferentes. Consulte su hoja de datos para conocer los detalles de los bits de configuración de PIC MCU que está utilizando.



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

DEPURACIÓN - Modo de depuración



El bit DEBUG en la palabra de configuración 2 es administrado automáticamente por MPLAB X IDE.

¡Para garantizar el correcto funcionamiento del dispositivo, este bit no debe modificarse!



Más información:

- Resumen de bits de configuración del lenguaje ensamblador (/8bit-i:emr-summary-configbit-in-assembly)
- Oscilador de 8 bits (/8bit:osc)
- Conmutación de reloj a prueba de fallas de 8 bits (/8bit:fscm)
- Cambio de reloj de 8 bits (/8bit:cswtch)
- PLL de 8 bits (/8bit:pll)
- Control de reinicio maestro de 8 bits (/8bit:mclr)
- REINICIO de oscurecimiento de 8 bits (/8bit:bor)
- REINICIO de oscurecimiento de 8 bits (/8bit:bor)
- Temporizador de encendido de 8 bits (/8bit:pwrt)
- Temporizador de vigilancia MCU de 8 bits (/8bit:wdt)
- Desbordamiento/subdesbordamiento de pila de 8 bits (/8bit:sof)

Resumen de los ajustes de bits de configuración para el MCU PIC® de rango medio mejorado 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.

En la página " Bits de configuración (/mcu1102:configuration-bits) " del tutorial mejorado de rango medio se presenta una descripción completa de estos ajustes de bits de configuración .



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

{...}

Sintaxis

```
#pragma config CONFIG_BIT_NAME = CONFIG_VALUE
```

CONFIG_BIT_NOMBRE CONFIG_VALUE

Función de los bits de configuración

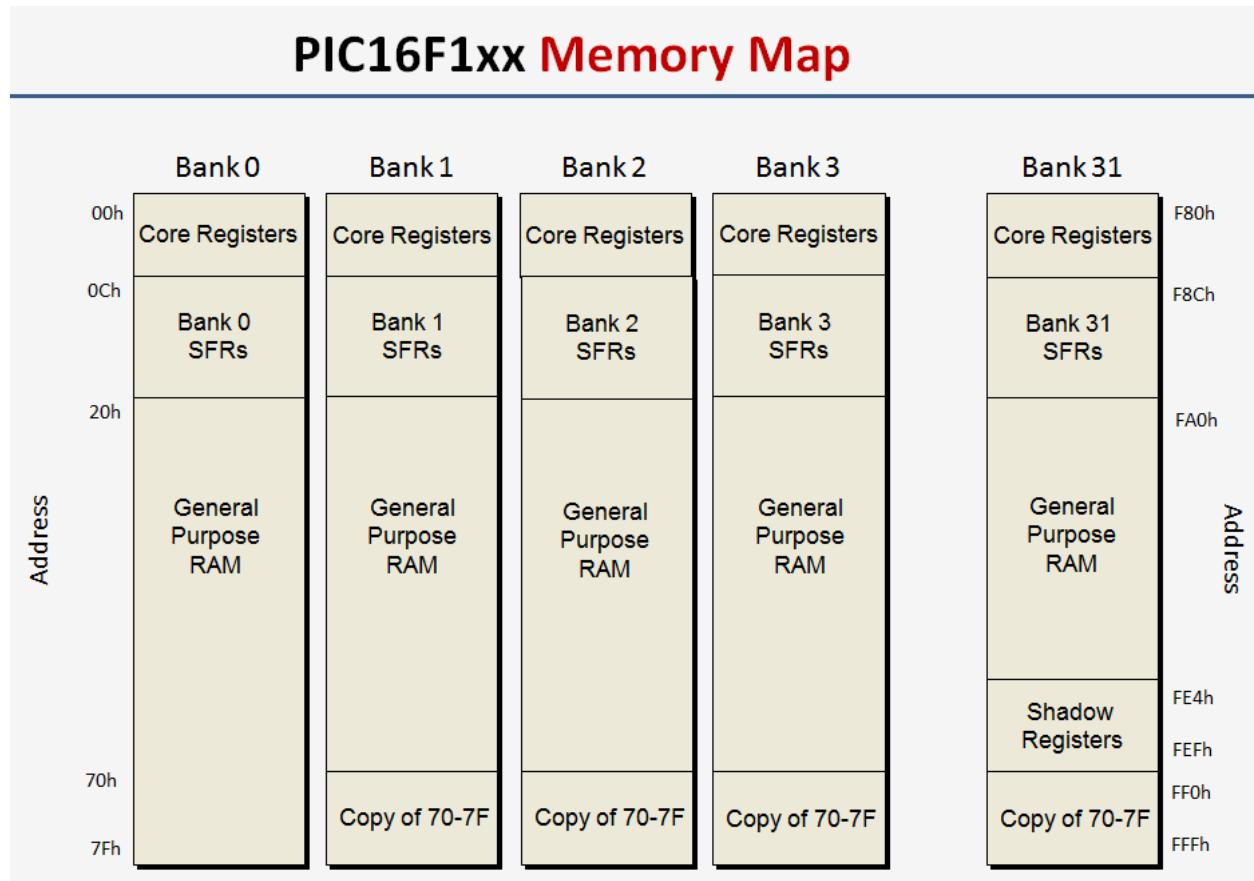
| | | |
|----------|------------------|---|
| DPC | ENCENDIDO | La protección del código de memoria de datos está activada |
| | APAGADO | La protección del código de memoria de datos está desactivada |
| ABURRIDO | ENCENDIDO | Brown-Out Reset está habilitado |
| | APAGADO | Brown-Out Reset está deshabilitado |
| CLKOUTEN | ENCENDIDO | La función CLKOUT está habilitada en el pin |
| | APAGADO | CLKOUT La función CLKOUT está deshabilitada. Función I/O u osc en el pin CLKOUT |
| IESO | ENCENDIDO | El modo de cambio de reloj interno/externo está habilitado El modo |
| | APAGADO | de cambio de reloj interno/externo está deshabilitado |
| FCMEN | ENCENDIDO | El monitor de reloj a prueba de fallas está habilitado |
| | APAGADO | El monitor de reloj a prueba de fallas está deshabilitado |
| WRT | ENCENDIDO | La protección contra escritura automática de la memoria flash está habilitada |
| | APAGADO | La protección contra escritura automática de la memoria flash está desactivada |

| | | |
|--------|------------------|--|
| PLÉN | ENCENDIDO | PLL interno 4X está habilitado |
| | APAGADO | PLL interno 4X está deshabilitado |
| STVREN | ENCENDIDO | El desbordamiento o subdesbordamiento de pila provocará un reinicio |
| | APAGADO | El desbordamiento o subdesbordamiento de pila NO provocará un reinicio |
| BORV | LO HOLA | Restablecimiento de voltaje Brown-Out - Punto de disparo bajo seleccionado |
| | | Restablecimiento de voltaje Brown-Out - Punto de disparo alto seleccionado |
| LVP | ENCENDIDO | Programación de bajo voltaje habilitada |
| | APAGADO | Se debe usar alto voltaje en MCLR/VPP para la programación |

Memoria de datos PIC16F1xxx de gama media mejorada

Organización de la memoria

Los microcontroladores PIC® de rango medio mejorado 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.



(/local--files/8bit:emr-data-memory/memory-map.png)

La memoria de datos PIC16F1xxx contiene cinco elementos de datos:

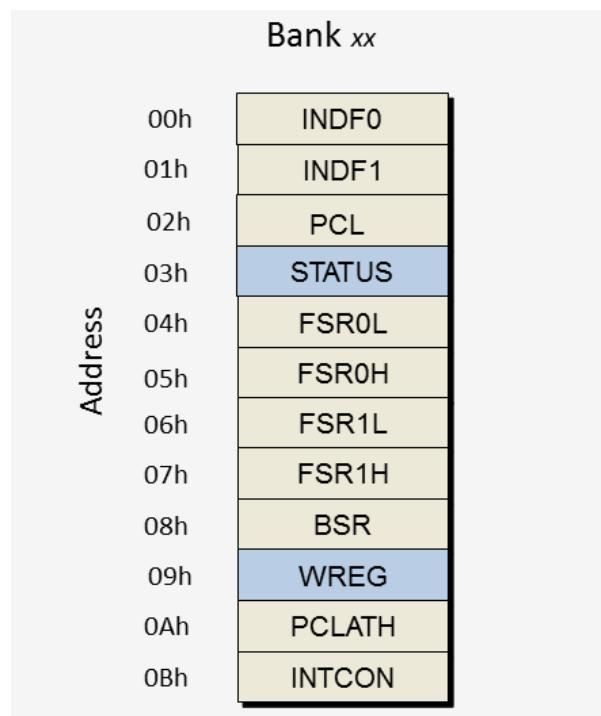
- Registros básicos
- Registros de funciones especiales (SFR)
- Memoria de propósito general
- Memoria común
- Registros de sombra

Registros principales

Las primeras 12 entradas de cada banco de memoria de datos PIC16F1xxx contienen registros denominados registros centrales. Estos 12 registros se repiten en cada banco. Se puede acceder a los registros centrales desde cualquier banco activo. Los registros básicos incluyen información para:

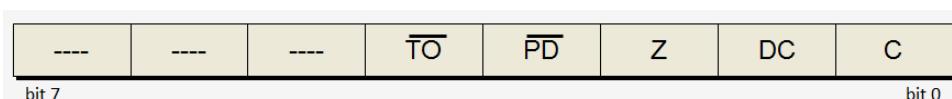
- Procesamiento general
 - Direccionamiento directo de la memoria
 - Direccionamiento indirecto de la memoria
 - control de interrupciones

Registros Generales de Tramitación



(/local--files/8bit-i:emr-core-registers/cr-processing.png)

ESTADO Registro



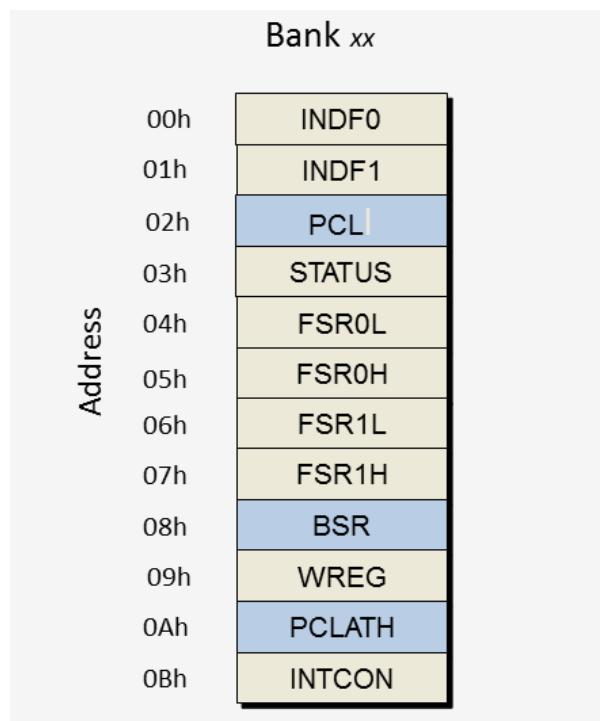
TO : indica que el temporizador de vigilancia ha expirado. **PD** : estado de la instrucción de dormir, del bit más significativo



Registro WREG

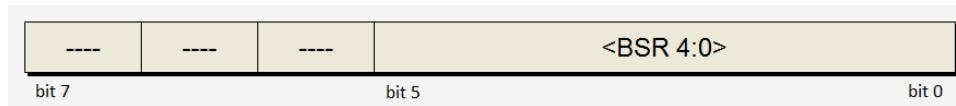
Conocido como Registro de Trabajo o registro W, WREG sirve como acumulador.

Registros de direccionamiento directo



(/local--files/8bit-i:emr-core-registers/cr-direct.png)

BSR (Registro de selección bancaria)



(/local--files/8bit-i:emr-core-registers/bsr-register.png)

Los cinco bits inferiores de BSR contienen el número de banco (0-31) del banco de datos activo. La información que explica cómo se usa el BSR se puede encontrar en la sección *"Direccionamiento directo"* (/mcu1102:direct-addressing) del tutorial PIC16F1xxx. Registros PCL y PCLATH

Estos registros se utilizan al escribir o leer del contador de programa de 15 bits. La información sobre el uso de PCL y PCLATH se explica en la sección *"Memoria (/mcu1102:program-memory) de programa"* del tutorial PIC16F1xxx

Registros de direccionamiento indirecto

| Bank <i>xx</i> | |
|----------------|--------|
| 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 |

(/local--files/8bit-i:emr-core-registers/cr-indirect.png)

INDF0, FSR0L, FSR0H,

INDF1, FSR1L, FSR1H

Estos seis registros controlan los dos canales de direccionamiento indirecto en la MCU. Los detalles del uso de estos registros se proporcionan en la sección *"Direccionamiento indirecto en la MCU PIC16F1xxx"* (/mcu1102:indirect-addressing) 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 |

(/local--files/8bit-i:emr-core-registers/cr-interrupt.png)

INTCON

| | | | | | | | |
|-------|------|--------|------|-------|--------|------|-------|
| GIE | PEIE | TMR0IE | INTE | IOCIE | TMROIF | INTF | IOCIF |
| bit 7 | | | | | | | bit 0 |

(/local--files/8bit-i:emr-core-registers/intcon-register.png)

GIE - Habilitación de interrupción global **PEIE** - Habilitación de interrupción periférica **TMR0IE** - Habilitación de interrupción de temporizador 0 **INTE** - Habilitación de interrupción externa **IOCIE** - Habilitación de interrupción por cambio **TMROIF** - Indicador de interrupción de temporizador 0 **INTF** - Indicador de interrupción externa **IOCIF** - Indicador de interrupción por 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*" (/mcu1102:interrupts) 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 están ubicados justo debajo de los registros centrales que comienzan en la dirección xxCh. **Los SFR controlan los periféricos (/mcu1102:peripherals)** PIC16F1xxx , los puertos de E/S digitales **(/mcu1102:digital-io)** y la **configuración del oscilador (/mcu1102:clocking)** .



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

| Bank 0 | | Bank 1 | | Bank 31 | |
|--------|---------|--------|------------|---------|------|
| 00Ch | PORTA | 08Ch | TRISA | F8Ch | ---- |
| 00Dh | PORTB | 08Dh | TRISB | F8Dh | ---- |
| | PORTC | | TRISC | | ---- |
| | PORTD | | TRISD | | ---- |
| | PORTE | | TRISE | | ---- |
| | PIR1 | | PIE1 | | ---- |
| | PIR2 | | PIE2 | | ---- |
| | PIR3 | | PIE3 | | ---- |
| ---- | | ---- | ---- | | ---- |
| | TMR0 | | OPTION_REG | | ---- |
| | TMR1L | | PCON | | ---- |
| | TMR1H | | WDTCON | | ---- |
| | T1CON | | OSCTUNE | | ---- |
| | T1GCON | | OSCCON | | ---- |
| | TMR2 | | OSCSTAT | | ---- |
| | PR2 | | ADRESL | | ---- |
| | T2CON | | ADRESH | | ---- |
| ---- | | | ADCON0 | | ---- |
| | CPSCON0 | | ADCON1 | | ---- |
| 01Fh | CPSCON1 | 09Fh | ---- | F9Fh | ---- |

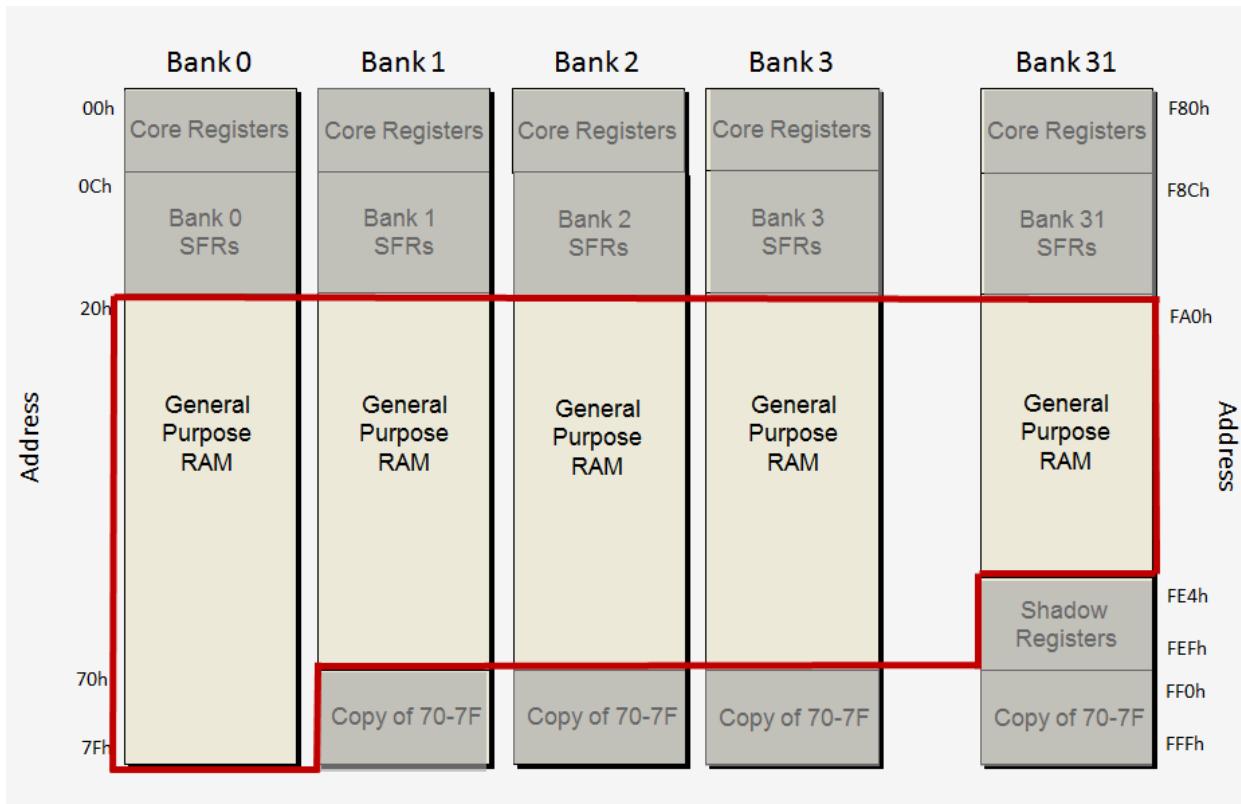
(/local--files/8bit-i:emr-special-function-registers/sfrs.png)



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

Memoria de propósito general

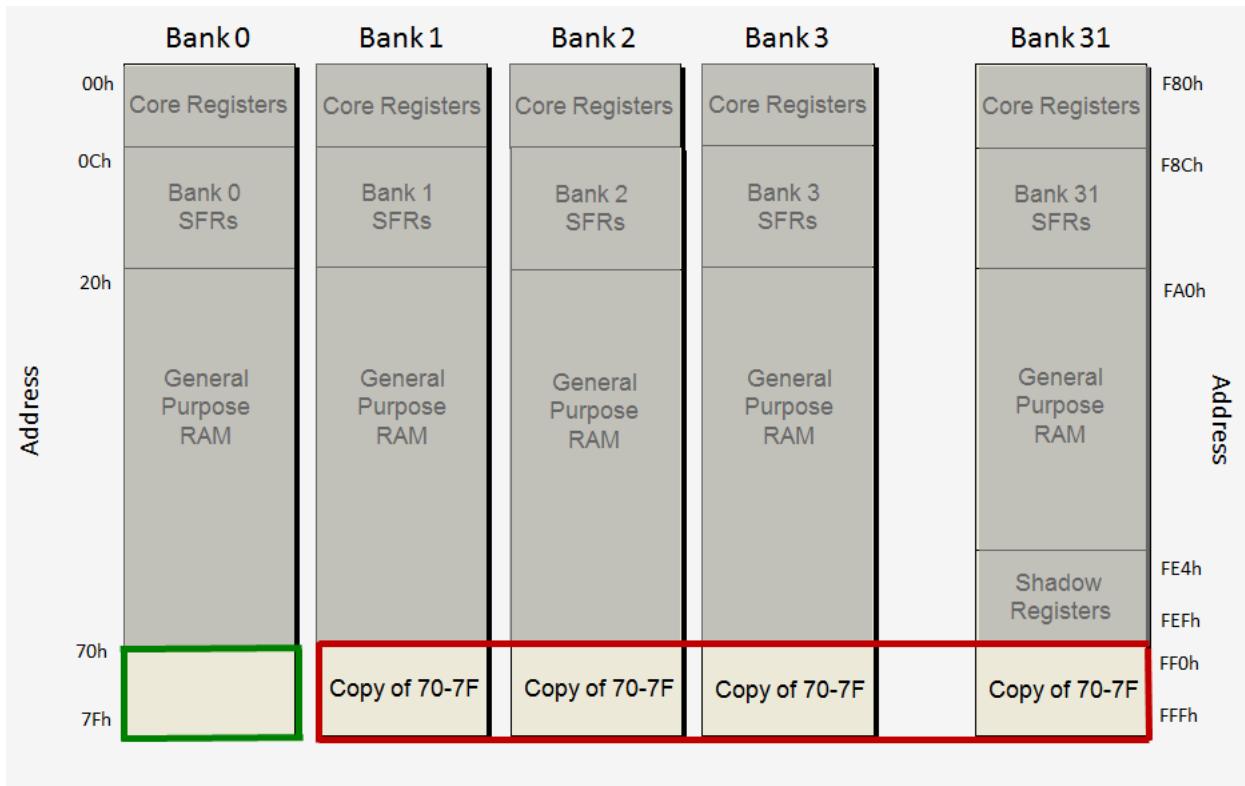
La memoria de uso general (RAM) se encuentra en cada banco de memoria justo debajo de los SFR. Esta memoria está disponible para los datos de la aplicación.



(/local--files/8bit:emr-data-memory/gp-memory.png)

Memoria común

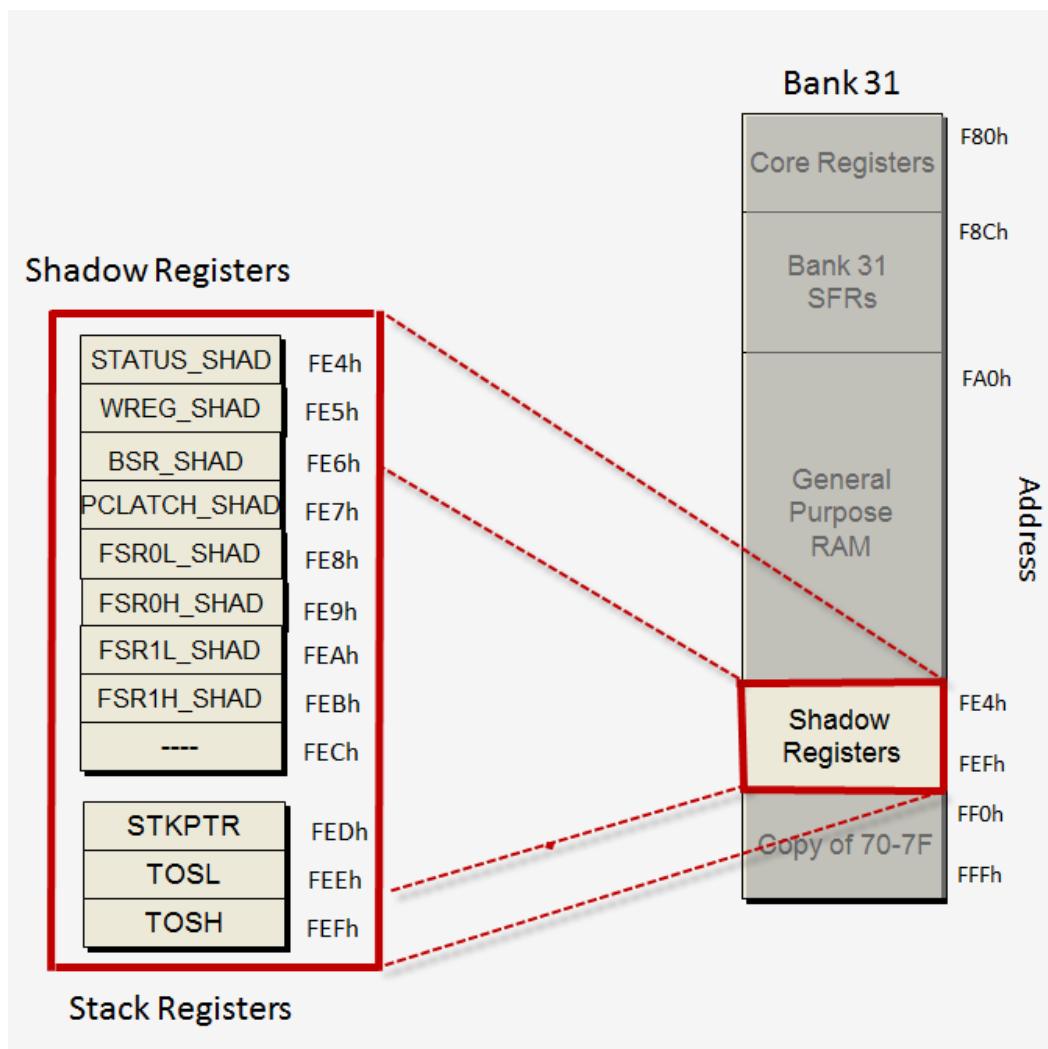
Los últimos 16 bytes del banco 0 (direcciones 70h - 7Fh) se repiten en cada banco de datos. Esto permite que los programas de aplicación accedan a las variables ubicadas en estas direcciones sin tener que configurar **BSR** .



(/local--files/8bit:emr-data-memory/common-ram.png)

Registros de sombra

En la parte inferior del banco 31 se encuentran los registros de sombra y pila PIC16F1xxx. Los registros sombra guardan el contexto del programa al capturar varios registros centrales cuando ocurre una interrupción. Los registros principales se restauran cuando se ejecuta una instrucción Return From Interrupt (RETFIE). Puede encontrar más información sobre los registros de sombra en la sección "*Interrupciones*" (/mcu1102:interrupts) del tutorial PIC16F1xxx.

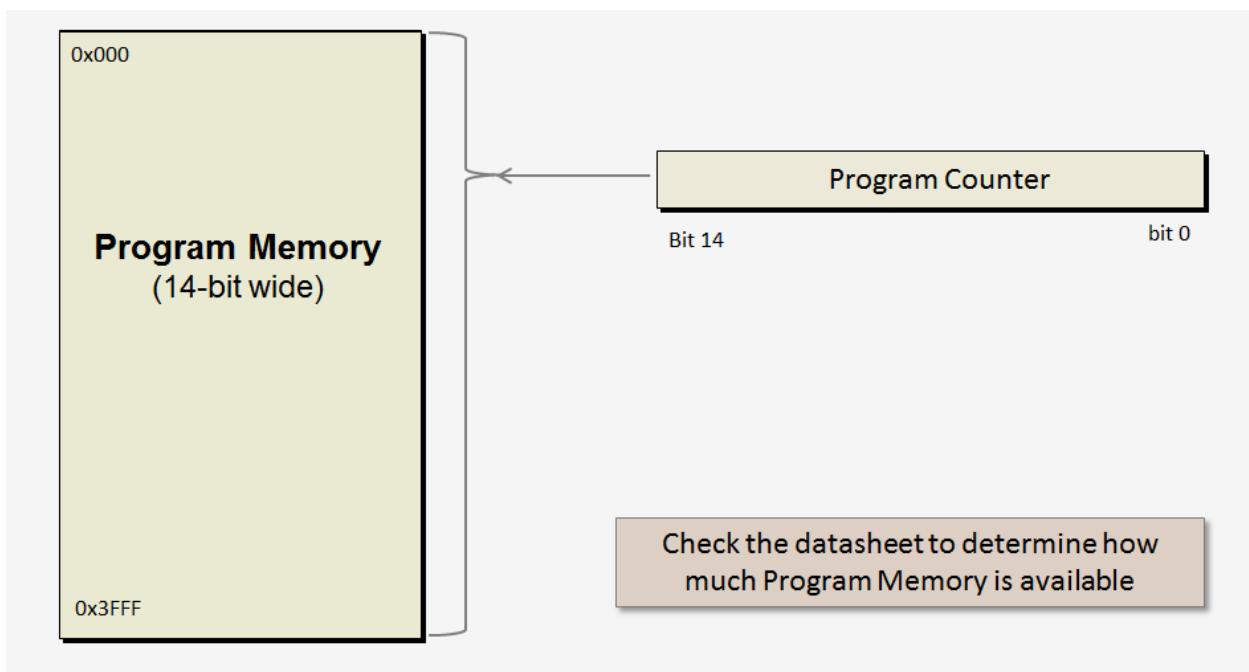


(/local--files/8bit:emr-data-memory/shadow-registers.png)

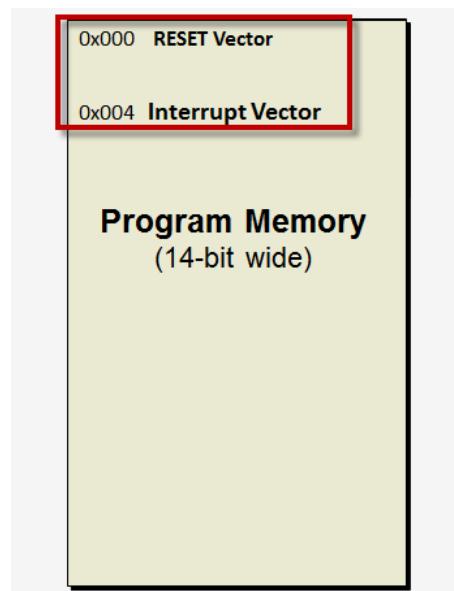
La pila se utiliza para almacenar el Contador de programa (PC) en caso de una llamada de interrupción o subrutina. La información sobre la pila y los registros asociados se puede encontrar en la sección *"Modelo del programador"* (/mcu1102:programmers-model) del tutorial PIC16F1xxx

Memoria de programa MCU PIC® de gama media mejorada

La memoria de programa en la MCU PIC® de gama media mejorada consta de hasta 32 MB de memoria Flash de 14 bits de ancho. Después de programar la MCU, la memoria del programa contiene el código de aplicación del usuario. Se accede a la memoria de programa mediante un registro de contador de programa (PC) de 15 bits.



(/local--files/8bit:emr-program-memory/pm-layout.png)



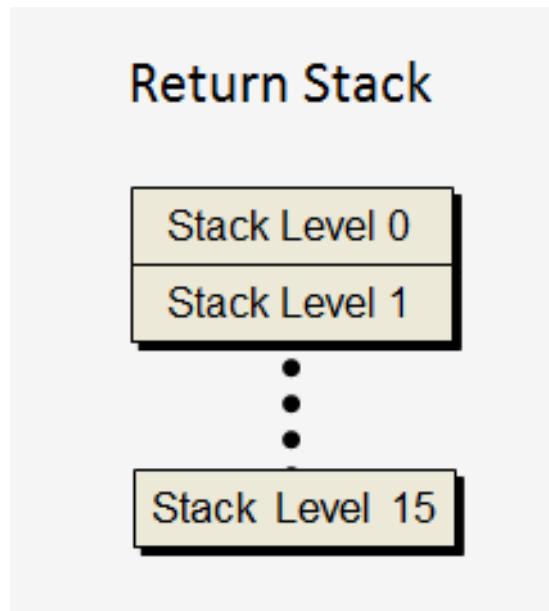
(/local--files/8bit:emr-program-memory/pm-vectors.png)

REINICIO vectorial

En RESET, el contador del programa se borra, dando como resultado todo ceros. Esto permite que la dirección de memoria de programa 0h sea la ubicación de la primera instrucción ejecutada después de una condición de REINICIO.

Vector de interrupción

Cuando ocurre una interrupción, el control del programa se transfiere a la dirección 04h. La sección "*Interrupciones*" (/mcu1102:interrupts) proporciona una descripción completa del proceso de interrupción



(/local--files/8bit:emr-program-memory/return-stack.png)

Pila de retorno

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

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



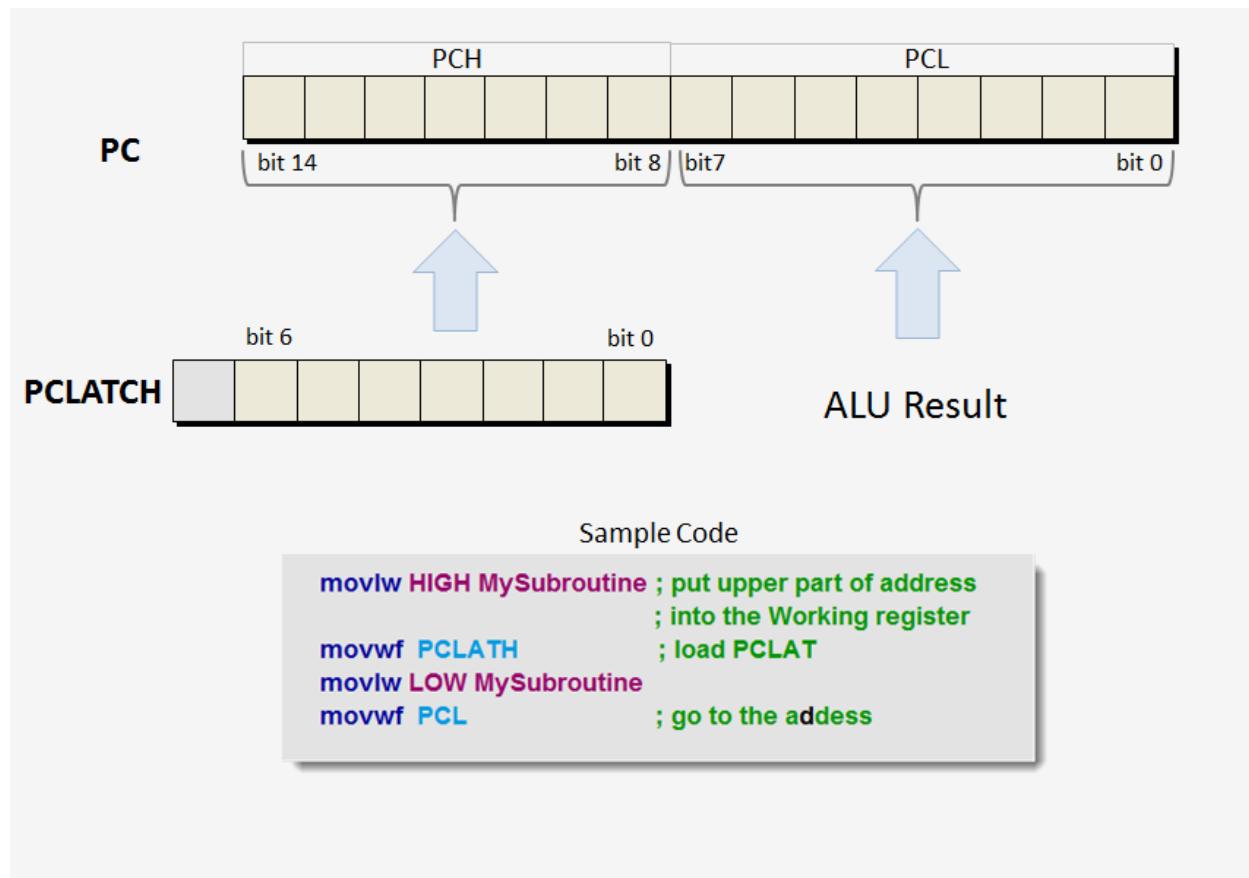
Para leer o modificar la PC de 14 bits con una MCU de 8 bits, se utilizan dos registros de funciones especiales (SFR):

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

PCL y PCLATH se utilizan cuando el programa escribe en la PC, lee el contador del programa o ejecuta una instrucción GOT0 o CALL .

Escribir en la PC

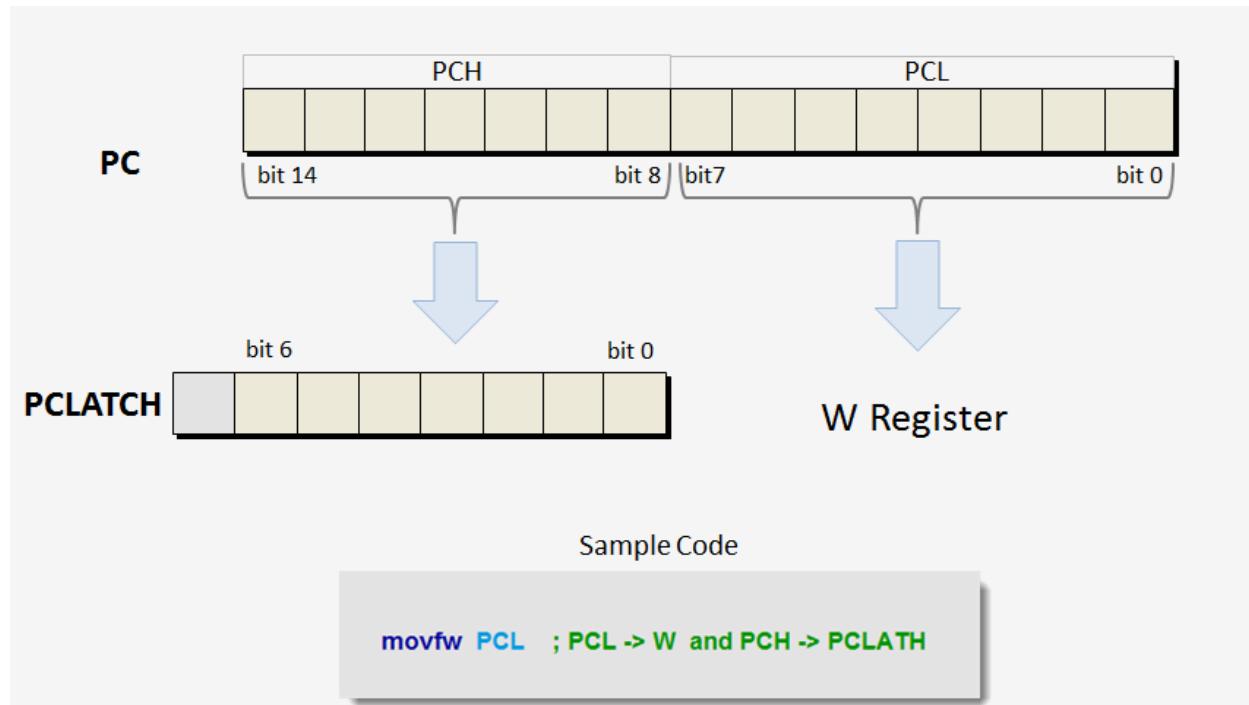
Cuando la aplicación escribe en PCL , el contenido actual de PCLATH<5:0> se escribirá en PC<14:8> . Debido a esto, el contenido de PCLATH<5:0> siempre DEBE ser correcto ANTES de escribir en PCL .



(/local--files/8bit:emr-program-memory/pcl-write.png)

leyendo la computadora

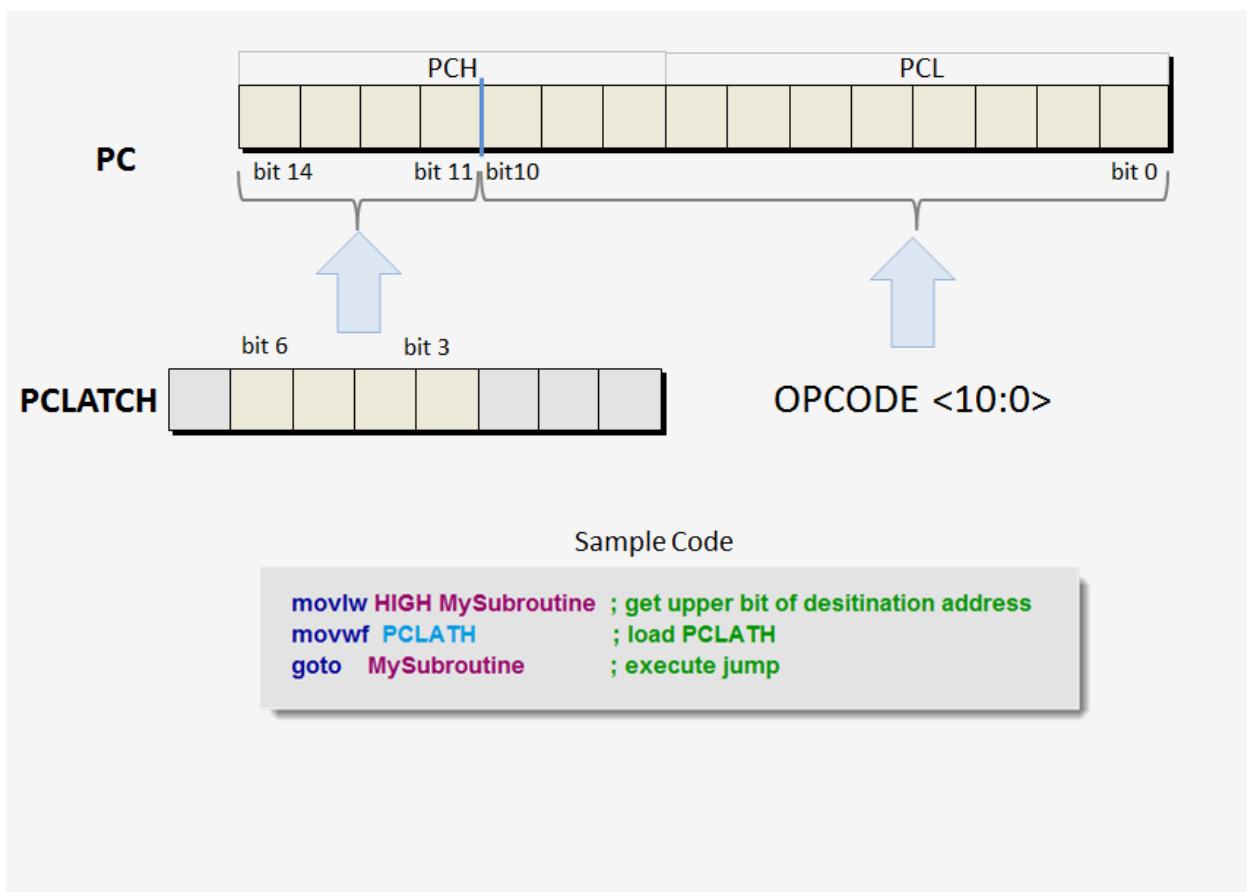
Cuando una aplicación lee `PCL` , `PC<14:8>` se captura en `PCLATH` .



(/local--files/8bit:emr-program-memory/pcl-read.png)

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 usa para extender el operando para acceder a todas las direcciones de memoria del programa. Cuando se ejecuta `CALL` 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>` .



(/local--files/8bit:emr-program-memory/goto.png)

Periféricos

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

| Módulo | Nombre |
|-----------------------------------|--|
| ADC | Convertidor analógico a digital de 10 bits |
| CVX (/8bit:emr-clc) | Celda lógica configurable |
| CMP | Comparadores analógicos |
| CPS | Módulo de detección capacitiva |
| CAD | Convertidor digital a analógico de 5 bits |
| DSM | Modulador de señal de datos |
| ECCP | Captura, comparación y PWM mejorados |
| EUSART | Transmisor receptor asíncrono universal mejorado |
| RVF | Referencia de voltaje fijo |
| MI2C | Circuito Interintegrado Maestro (I ² C) |
| MSSP | Puerto serie síncrono maestro (SPI/I ² C) |
| suboficial (/8bit:emr-nco) | Oscilador controlado numéricamente |
| SPI | Interfaz de periféricos en serie (SPI) |
| TMR0 | Temporizador 0 (8 bits) |
| TMR1 | Temporizador 1 (16 bits) |
| TMR2 | Temporizador 2 (8 bits) |
| UART | Transmisor receptor asíncrono universal |

...E/S digital de gama media mejorada de 8 bits

Estructura básica de E/S

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

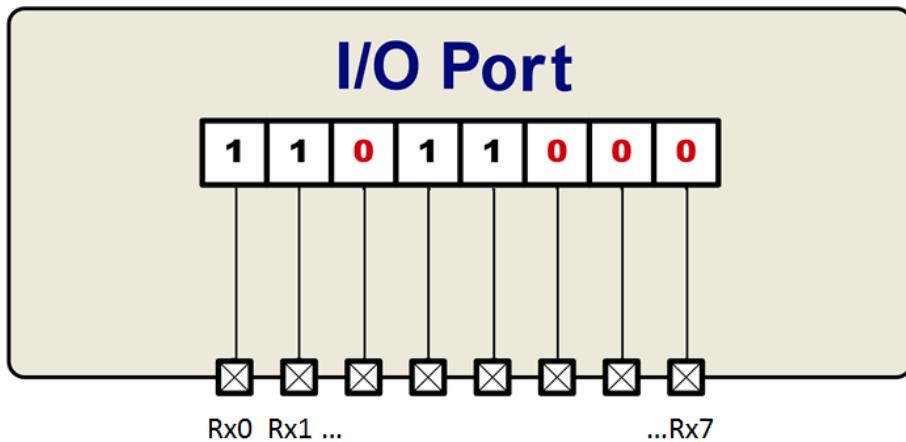
- Capacidad para monitorear entradas digitales
- Señales de salida digital de control
- Dominadas débiles internas
- Multiplexado con periféricos
- Alta capacidad de accionamiento (hasta 25 mA disipador/fuente en muchos pines de E/S)
- Manipulación directa de bit de ciclo único
- Diodos de protección ESD de 4 kV

En REINICIO:

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

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

Puertos de E/S



PORT_x

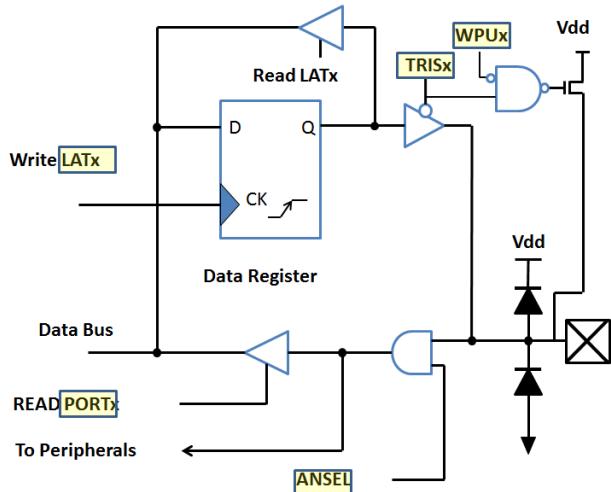
(https://microchipdeveloper-com.translate.goog/local--files/8bit:emr-digital-io/ports.svg?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

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 digital individualmente pin por pin. También se puede acceder a todos los miembros de un puerto de E/S particular en un ciclo de instrucción mediante una de las instrucciones de acceso a bytes de la MCU.

Los puertos de E/S se denominan por letras (p. ej., PORTA, PORTB, PORTC). El número de puertos de E/S variará según el MCU de PIC que se utilice. Consulte la hoja de datos individual para determinar las asignaciones de PUERTO para una MCU PIC.

Pin típico de E/S digital



(https://microchipdeveloper-com.translate.goog/local--files/8bit:emr-digital-io/basic-digital-io.svg?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

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.

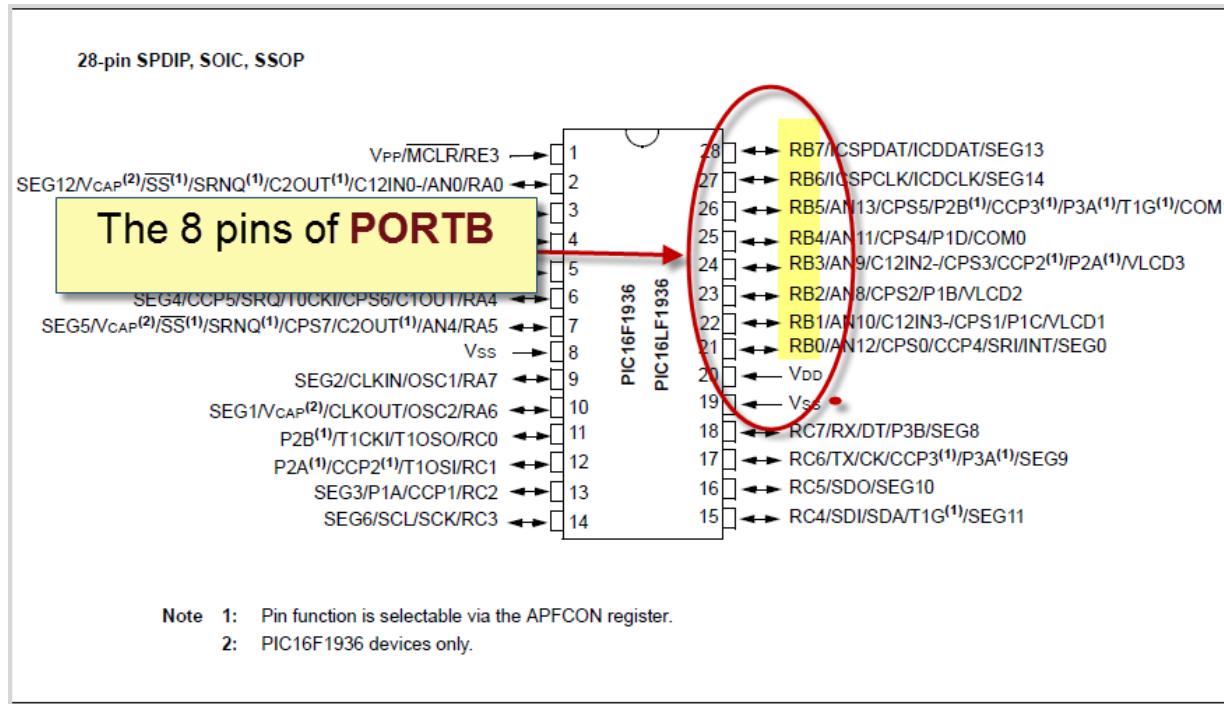
A lo largo de esta página se proporcionan ejemplos de programas que muestran cómo utilizar los registros de control de E/S.

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 de MCU. Algunos pines de E/S digitales tienen capacidad analógica y se pueden configurar para operar como pines de entrada analógica. Consulte el diagrama de pines de la hoja de datos del dispositivo para determinar qué pines están disponibles como E/S digital.

Los pines digitales se identifican mediante tres identificadores secuenciales:

- El primer identificador de un pin digital es la letra R.
- El segundo identificador es una letra del PUERTO en el 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 PUERTO que ocupa el Pin.



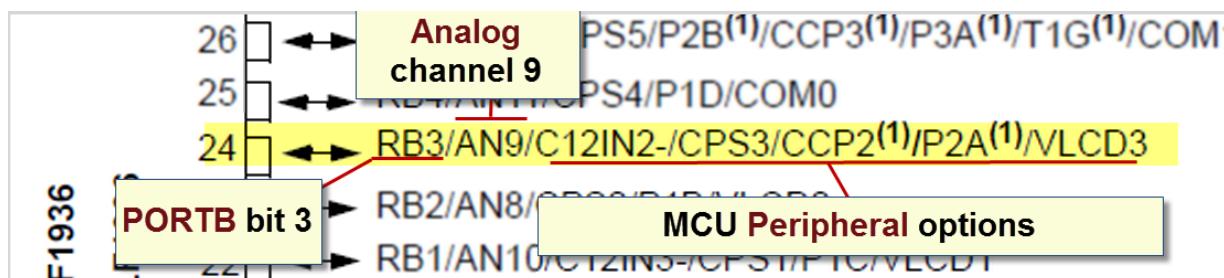
(https://microchipdeveloper-com.translate.goog/local--files/8bit:emr-digital-io/portb.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

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 la 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 están designados por nombre en la hoja de datos.



(https://microchipdeveloper-com.translate.goog/local--files/8bit:emr-digital-io/pin.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

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 **P0RTB** 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 con capacidad analógica. (ANSELA controla el modo de todos los pines con capacidad analógica en **P0RTA** , ANSELB controla el modo para **P0RTB** , ANSELD para **P0RTD** , 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 del 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. La MCU establece todos los bits ANSELx relevantes en 1.

Ejemplo:

El pin 24, en la Figura 4, puede configurarse como canal analógico 9 o como pin digital RB3.

Para usar este pin como pin digital, se debe borrar el bit 3 de ANSELB.



```
1  banksel  ANSELB      ; se?
2  bcf      ANSELB,3      ; de
```



```
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 el panel de entrada.
- Si se escribe un pin analógico configurado 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 en un puerto se asigna a un bit en un registro TRIS. La dirección de datos para cada pin se puede configurar 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, lo que hace que todos los pines sean entradas High-Z.

Convertir un pin en un pin de salida digital

Para configurar un pin como salida digital, se deben colocar 0 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 con letras. Los registros LATx comienzan con LATA y

continúan con LATB, LATC...)

Escribir un 1 a un bit en el registro LAT conducirá el pin a Vdd. Un 0 en un bit LAT llevará el 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 alta frecuencia, si se escriben secuencialmente varios comandos de modificación de bits (BSF, BCF) en un registro de PUERTO de salida, es posible que la última instrucción de manipulación de bits sobrescriba una instrucción anterior, dando como resultado un valor incorrecto en el puerto de salida. Para evitar la posibilidad de que esto ocurra, se recomienda encarecidamente que la salida siempre se realice en el 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 elevan mientras que los cuatro bits superiores se establecen en 0.



```
1 void main(void) ?  
2 {  
3     ANSELB = 0 ; // set PORTB  
4     LATB = 0 ; // set output  
5     TRISB = 0 ; // make PORTB  
6     LATB = 0x0F // set lower  
7     while(1);  
8 }
```



```
1 banksel ANSELB ?  
2 clr ANSELB ; make  
3 banksel LATB ;  
4 clr LATB ; set 1  
5 banksel TRISB ;  
6 clr TRISB ; set :  
7 banksel LATB ;  
8 movlw 0x0F ;  
9 movwf LATB ;  
10 goto $ .100
```



¿Por qué se borró el registro LAT antes de configurar 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 pulsos de salida espurios e involuntarios.

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



```
1 void main(void) ?  
2 {  
3     ANSELBbits.ANSB3 = 0 ; //  
4     LATBbits.LATB3 = 0 ; //  
5     TRISBbits.TRISB3 = 0 ; //  
6     LATBbits.LATB3 = 1 ; //  
7     while(1);  
8 }
```



```
1 banksel ANSELB ?  
2 bcf ANSELB,3 ; mal  
3 banksel LATB  
4 bcf LATB,3 ; set  
5 banksel TRISB  
6 bcf TRISB,3 ; set 1  
7 banksel LATB  
8 bsf LATB,3 ; set  
9 goto $ ; loop
```

Entradas digitales

Convertir un pin en un pin de entrada digital:

Para configurar un pin como entrada digital, se deben colocar 1 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 el valor del pin.

Código de ejemplo:

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



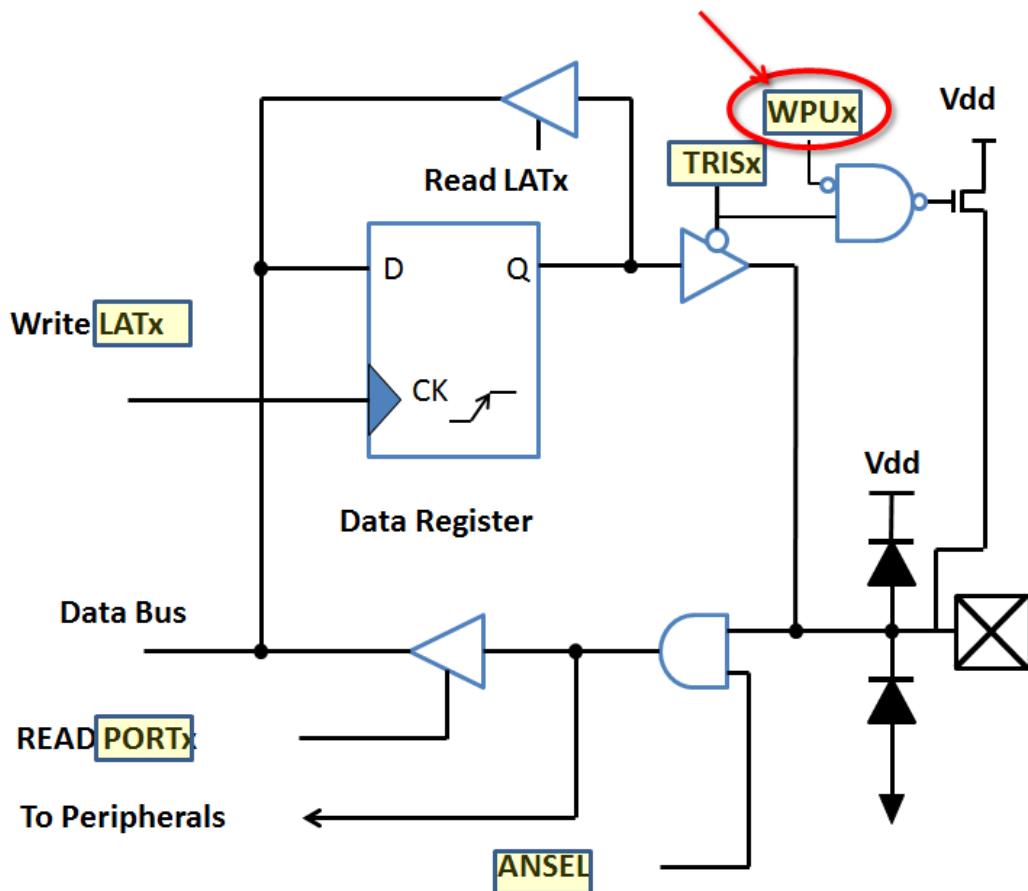
```
1  #define input_pin PORTAbits.R3
2  void main(void)
3  {
4  ANSELAbits.ANSA3 = 0; // make
5  TRISAbits.TRISA3 = 0; // set
6  while(1)
7  if (input_pin) exception_routine();
8 }
```



```
1 Start:                      ?
2 banksel ANSELA
3 bcf ANSELA,3 ; set RA3
4 banksel TRISA
5 bsf TRISA3 ; set
6 Loop:
7 banksel PORTA
8 btfsc PORTA,3 ; "test" RA3
9 goto Exception_Routine;
10 goto Loop;
```

Dominadas débiles internas

Las resistencias pull-up débiles internas están habilitadas para cada pad de E/S mediante el registro WPUx.



(https://microchipdeveloper-com.translate.goog/local--files/8bit:emr-digital-io/wpu.svg?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)



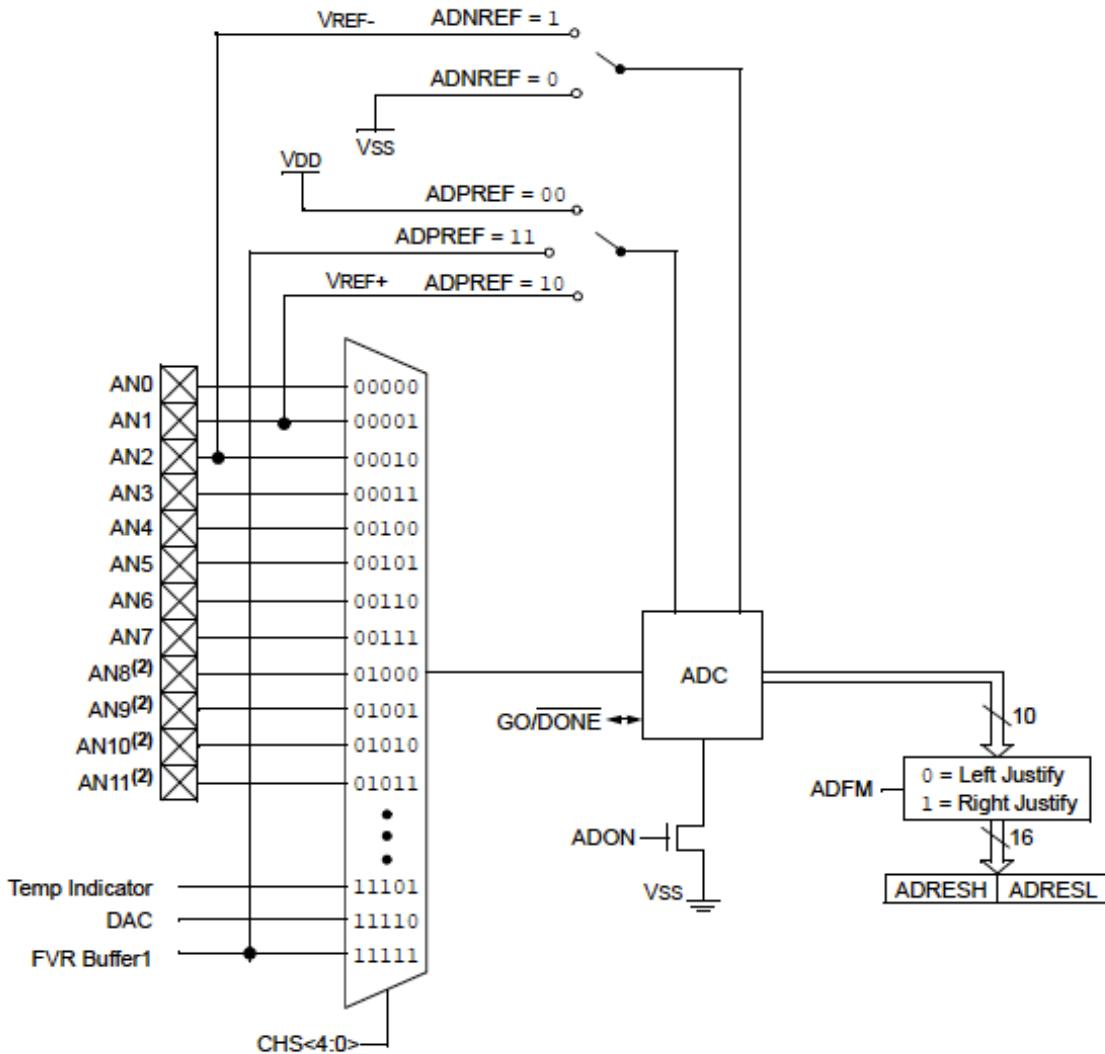
El pull-up se desactivará automáticamente cuando TRIS se configure como una salida o el pin se configure como una entrada analógica. Estos cambios en TRIS y ANSEL anularán la configuración de WPU

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.

...Conversor analógico a digital

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. Las entradas analógicas de los microcontroladores de Microchip, que se multiplexan en un solo circuito de muestra y retención. La salida de la muestra y retención está conectada a la entrada del ADC. El ADC genera el resultado binario de 10 bits a través de aproximaciones sucesivas y almacena el resultado de la conversión en los registros de resultados del ADC.



(https://microchipdeveloper.com.translate.goog/local--files/8bit:adc/adcblock.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

El ADC utiliza una referencia de voltaje que es seleccionable por software para generarse internamente o suministrarse externamente.

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

Configuración ADC

Cuando el ADC se configura por primera vez, debe tener varios ajustes de configuración habilitados. Éstos incluyen:

- Configuración del puerto ADC
- Selección de canal ADC
- Selección de referencia de voltaje ADC

- Fuente de reloj de conversión ADC
- control de interrupción ADC
- formato de resultado ADC

Veremos paso a paso cada uno de estos a continuación.

Configuración del puerto ADC

El primer ajuste de configuración es la configuración de pines de E/S. La mayoría de los pines de E/S del ADC se pueden usar como entrada analógica o entrada digital. Al convertir señales analógicas usando el ADC, el pin de E/S debe configurarse para entrada analógica configurando los bits asociados en el registro TRIS y el registro ANSEL.

El registro TRIS para el pin de E/S debe tener su bit asociado establecido en 1 para convertirlo en una entrada. Si el pin de E/S es parte del bloque PORTA, el registro TRISA contiene el bit.

REGISTER 6-4: TRISA: PORTA TRI-STATE REGISTER

| R/W-1/1 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 |
| bit 7 | | | | | | | bit 0 |

(https://microchipdeveloper.com.translate.goog/local--files/8bit:adc/trisa.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

El siguiente paso es establecer el bit en el registro ANSEL para el pin de E/S y establecer el bit en 1 para habilitar el ADC en ese pin. Si el pin de E/S es parte del bloque PORTA, el registro ANSELA contiene el bit.

REGISTER 6-5: ANSELA: PORTA ANALOG SELECT REGISTER

| U-0 | U-0 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|-------|-----|---------|---------|---------|---------|---------|---------|
| — | — | ANSA5 | ANSA4 | ANSA3 | ANSA2 | ANSA1 | ANSA0 |
| bit 7 | | | | | | | bit 0 |

(https://microchipdeveloper.com.translate.goog/local--files/8bit:adc/ansela.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

Selección de canal ADC

El multiplexor ADC debe conectarse al pin de E/S antes de iniciar el proceso de muestreo y retención. Esto se hace con un conjunto de bits en el registro ADCON0.

Antes de que se solicite una muestra de ADC, estos bits de selección de canal se configuran para conectarse al pin de E/S deseado. Solo se puede conectar un pin al ADC a la vez. Una vez que se completa el proceso, los bits de selección se pueden cambiar para conectarse al siguiente pin y el proceso ADC comienza de nuevo.

REGISTER 11-1: ADCON0: A/D CONTROL REGISTER 0

| U-0 | R/W-0/0 |
|-------|---------|---------|---------|---------|---------|---------|---------|
| — | CHS4 | CHS3 | CHS2 | CHS1 | CHS0 | GO/DONE | ADON |
| bit 7 | | | | | | | bit 0 |

(https://microchipdeveloper.com.translate.goog/local--files/8bit:adc/adcon0.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

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⁽¹⁾
 01001 = AN9⁽¹⁾
 01010 = AN10⁽¹⁾
 01011 = AN11⁽¹⁾
 01100 = Reserved. No channel connected.
 .
 .
 .
 11100 = Reserved. No channel connected.

11101 = Temperature Indicator⁽⁴⁾
 11110 = DAC output⁽²⁾
 11111 = FVR (Fixed Voltage Reference) Buffer 1 Output⁽³⁾

(https://microchipdeveloper.com.translate.goog/local--files/8bit:adc/adcselect.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

Algunos dispositivos pueden tener menos canales

- Consulte la página "Módulo convertidor de digital a analógico (DAC) (MCU de 8 bits)" (https://microchipdeveloper.com.translate.goog/8bit:dac?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc) para obtener más información.
- Consulte la página "Referencia de voltaje fijo (FVR)" (https://microchipdeveloper.com.translate.goog/8bit:fvr?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc) para obtener más información.
- Consulte la página "Indicador de temperatura" (https://microchipdeveloper.com.translate.goog/8bit:temp?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc) para obtener más información.

419&_x_tr_pto=sc) para obtener más información.

Selección de referencia de voltaje ADC

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

Valor digital = [Tensión analógica / (V_{REF} + - V_{REF} -)] * 1024

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

- V_{REF} +
- V_{DD} -
- Referencia de voltaje fijo (FVR)

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

- REF - V-
- V_{SS}

V_{DD} y V_{SS} 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 función en muchos dispositivos PIC®, aunque no en todos. Puede incluir un solo voltaje o, a veces, más de un nivel de voltaje está disponible.

Los bits de selección de referencia de voltaje están 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 |

(https://microchipdeveloper-com.translate.goog/local--files/8bit:adc/adcon1.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

| | |
|---------|---|
| bit 2 | ADNREF: A/D Negative Voltage Reference Configuration bit 0 = VREF- is connected to Vss 1 = VREF- is connected to external VREF- pin ⁽¹⁾ |
| bit 1-0 | ADPREF<1:0>: A/D Positive Voltage Reference Configuration bits 00 = VREF+ is connected to VDD 01 = Reserved 10 = VREF+ is connected to external VREF+ pin ⁽¹⁾ 11 = VREF+ is connected to internal Fixed Voltage Reference (FVR) module ⁽¹⁾ |

(https://microchipdeveloper.com.translate.goog/local--files/8bit:adc/vref.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

Fuente de reloj de conversión ADC

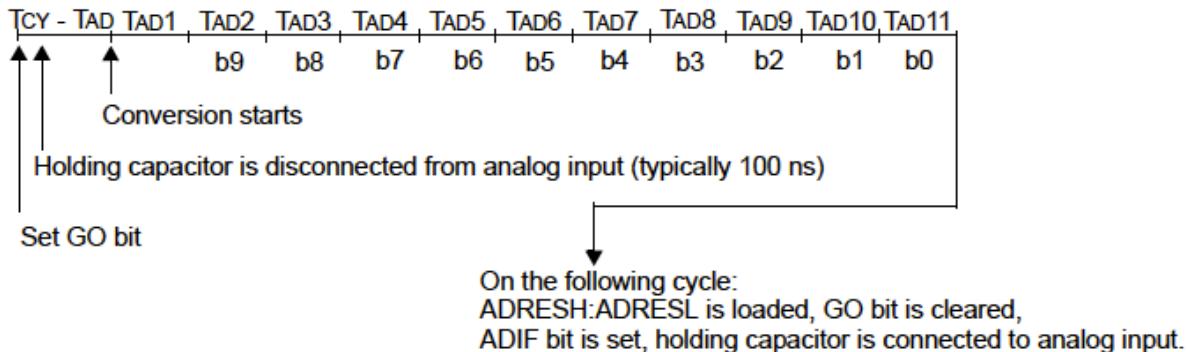
La fuente del reloj de conversión es seleccionable por software a través de los bits ADCS del registro ADCON1. Hay hasta siete opciones de reloj posibles según el dispositivo que se utilice:

- 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_{AD}. Una conversión completa de 10 bits requiere períodos de 11,5 T_{AD}, como se muestra aquí:



(https://microchipdeveloper.com.translate.goog/local--files/8bit:adc/tad.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

Para una conversión correcta, se debe cumplir con la especificación T_{AD} adecuada. Un reloj ADC se puede seleccionar fácilmente de la tabla a continuación. Aparece un gráfico similar en la hoja de datos del dispositivo. Los mejores valores se muestran en el medio del gráfico con un fondo blanco.

| ADC Clock Period (T_{AD}) | | 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 ⁽²⁾ | 100 ns ⁽²⁾ | 125 ns ⁽²⁾ | 250 ns ⁽²⁾ | 500 ns ⁽²⁾ | 2.0 μ s |
| Fosc/4 | 100 | 125 ns ⁽²⁾ | 200 ns ⁽²⁾ | 250 ns ⁽²⁾ | 500 ns ⁽²⁾ | 1.0 μ s | 4.0 μ s |
| Fosc/8 | 001 | 0.5 μ s ⁽²⁾ | 400 ns ⁽²⁾ | 0.5 μ s ⁽²⁾ | 1.0 μ s | 2.0 μ s | 8.0 μ s ⁽³⁾ |
| Fosc/16 | 101 | 800 ns | 800 ns | 1.0 μ s | 2.0 μ s | 4.0 μ s | 16.0 μ s ⁽³⁾ |
| Fosc/32 | 010 | 1.0 μ s | 1.6 μ s | 2.0 μ s | 4.0 μ s | 8.0 μ s ⁽³⁾ | 32.0 μ s ⁽³⁾ |
| Fosc/64 | 110 | 2.0 μ s | 3.2 μ s | 4.0 μ s | 8.0 μ s ⁽³⁾ | 16.0 μ s ⁽³⁾ | 64.0 μ s ⁽³⁾ |
| FRC | x11 | 1.0-6.0 μ s ^(1,4) |

Legend: Shaded cells are outside of recommended range.

Note 1: The FRC source has a typical T_{AD} time of 1.6 μ s for VDD.

2: These values violate the minimum required T_{AD} time.

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

4: The ADC clock period (T_{AD}) 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.

(https://microchipdeveloper-com.translate.goog/local--files/8bit:adc/tadchart.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

La selección de FRC del oscilador interno será una conversión más lenta pero garantizará que se cumplan los requisitos de T_{AD} . El FRC también se puede usar en modo de suspensión para ejecutar mediciones de ADC.

Control de interrupciones

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

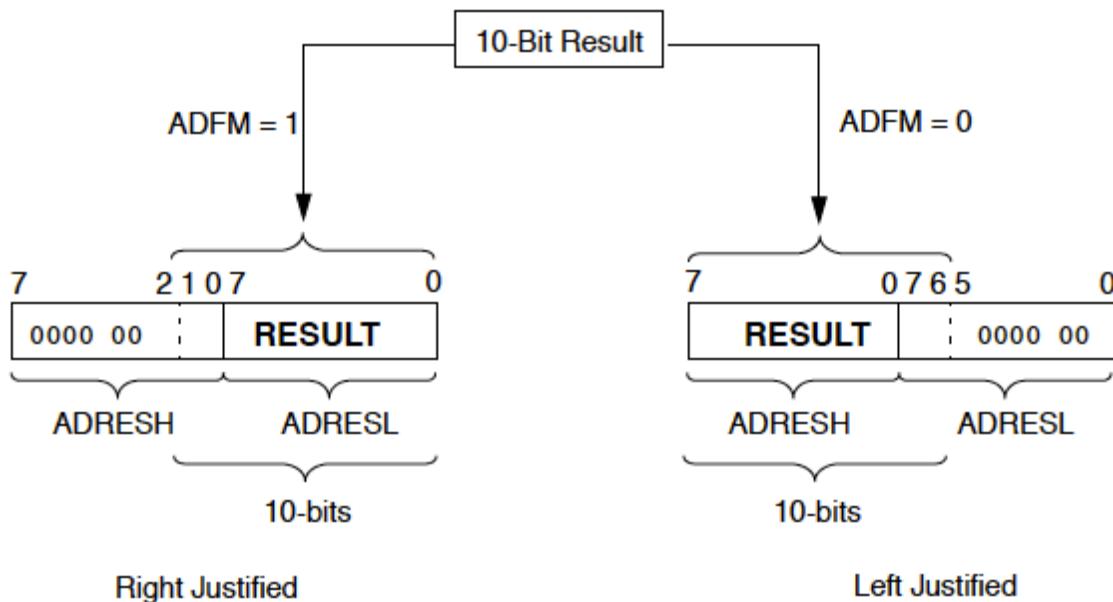
Esos bits de interrupción incluyen:

- El indicador de interrupción ADC es el bit ADIF en el registro 1 de interrupción periférica (PIR1).
- La habilitación de interrupción ADC es el bit ADIE en el registro de habilitación de interrupción periférica (PIE1).
- El bit de habilitación de interrupción global (GIE) y los bits de habilitación de interrupción periférica (PEIE) en el registro INTCON también deben estar habilitados.

Después de ejecutar una interrupción del modo SLEEP y completar ISR y ADC, el bit ADIF debe borrarse en el software.

Formato de resultado ADC

El resultado de la conversión ADC se almacena en dos registros de 8 bits de ancho; ADRESH y ADRESL. Este par de registros tiene 16 bits de ancho, 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) en el registro ADCON1 controla esta justificación. Los bits adicionales en los registros ADRESH y ADRESL se cargan con '0'.



(https://microchipdeveloper-com.translate.goog/local--files/8bit:adc/justified.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

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.

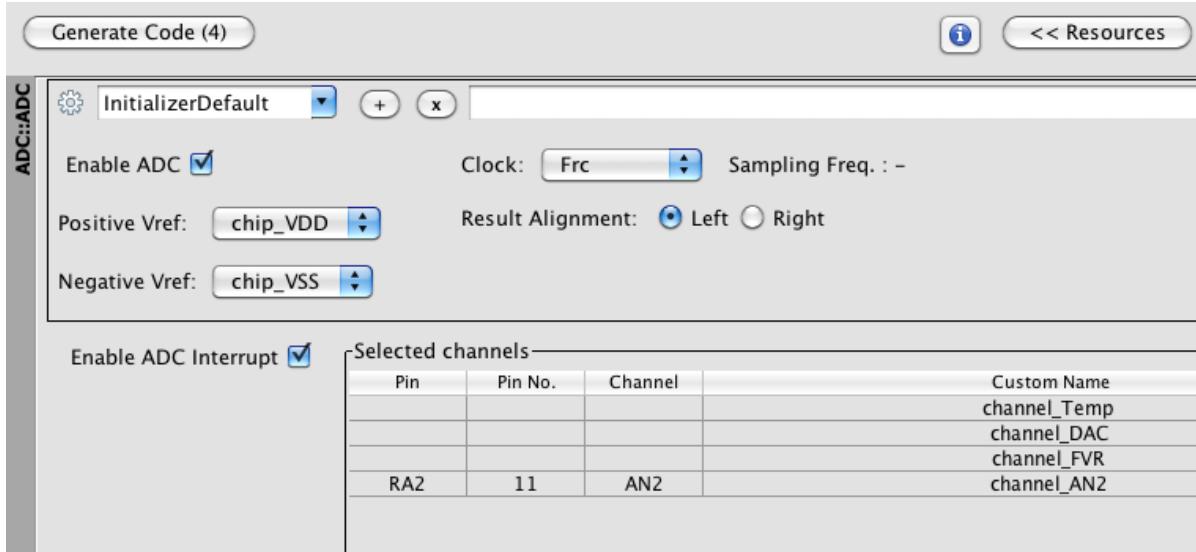
(https://microchipdeveloper-com.translate.goog/local--files/8bit:adc/adfmbit.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

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

Configurador de código MPLAB® - Configuración

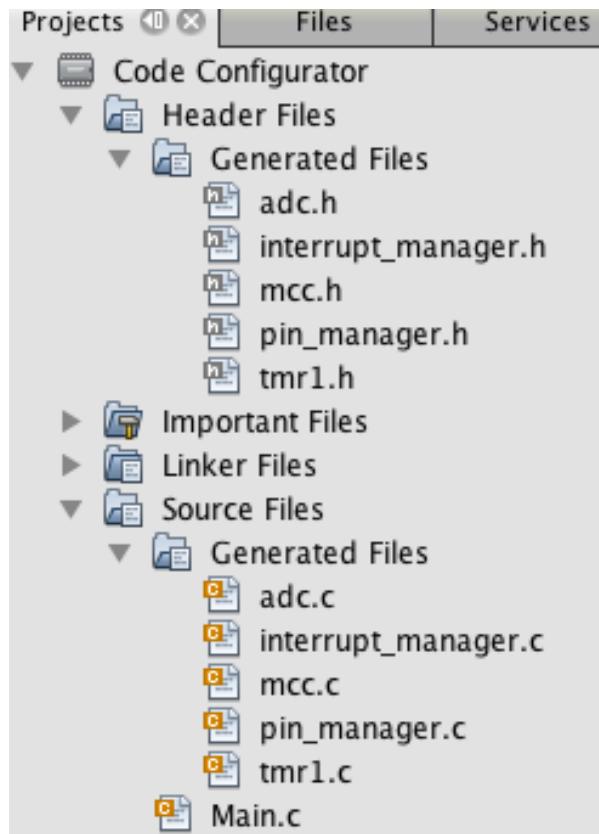
ADC

MPLAB® Code Configurator (MCC) (https://microchipdeveloper-com.translate.goog/mcc:overview?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc) facilita la configuración del código ADC. Todos los ajustes de configuración descritos anteriormente se pueden configurar en una pantalla GUI simple dentro de MPLAB X IDE. La pantalla ADC se muestra aquí:



(https://microchipdeveloper-com.translate.goog/local--files/8bit:adc/mc2.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

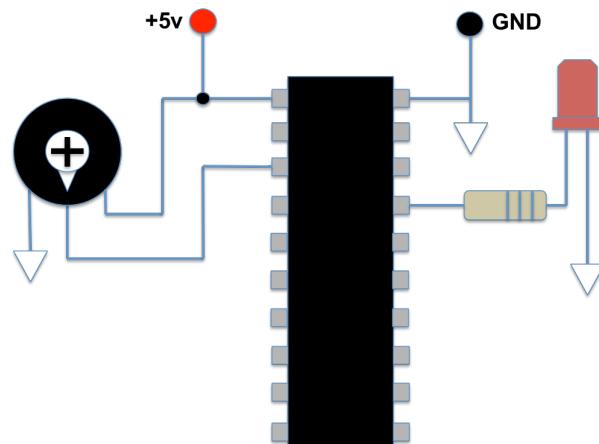
Cada opción de configuración se selecciona como una casilla de verificación o desde un menú desplegable. Una vez realizadas las selecciones, el código se genera y se coloca en el proyecto. Se crean dos archivos llamados `adc.h` y `adc.c`. Contienen código de configuración de ADC y también funciones personalizadas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



(https://microchipdeveloper-com.translate.goog/projects:mcu1101-project-3?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

Ejemplo de proyecto ADC

Aquí hay un proyecto de ejemplo paso a paso para configurar el ADC usando MCC (https://microchipdeveloper-com.translate.goog/projects:mcu1101-project-3?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc).



(https://microchipdeveloper-com.translate.goog/projects:mcu1101-project-3?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

project-3?_x_tr_sl=en&_x_tr_tl=es&
_x_tr_hl=es-419&_x_tr_pto=sc)

...Celda lógica configurable

La celda lógica configurable (CLC) proporciona una lógica programable que opera fuera de las limitaciones de velocidad de ejecución del software. La celda lógica acepta hasta 16 señales de entrada y, mediante el uso de puertas configurables, reduce las 16 entradas en cuatro líneas lógicas que controlan 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
- Bits de registro

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

Las posibles configuraciones incluyen:

- Lógica combinatoria
 - Y
 - NAND
 - Y-O
 - Y-O-INVERTIR
 - OR-XOR
 - O-XNOR
 - Pestillos
 - RS
 - Reloj D con Set y Reset
 - D transparente con Set y Reset
 - JK cronometrado con reinicio
-

Videotutorial de CLC

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

Microchip Self Paced Training - Configurable Logic Cell



Configuración de CLC

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

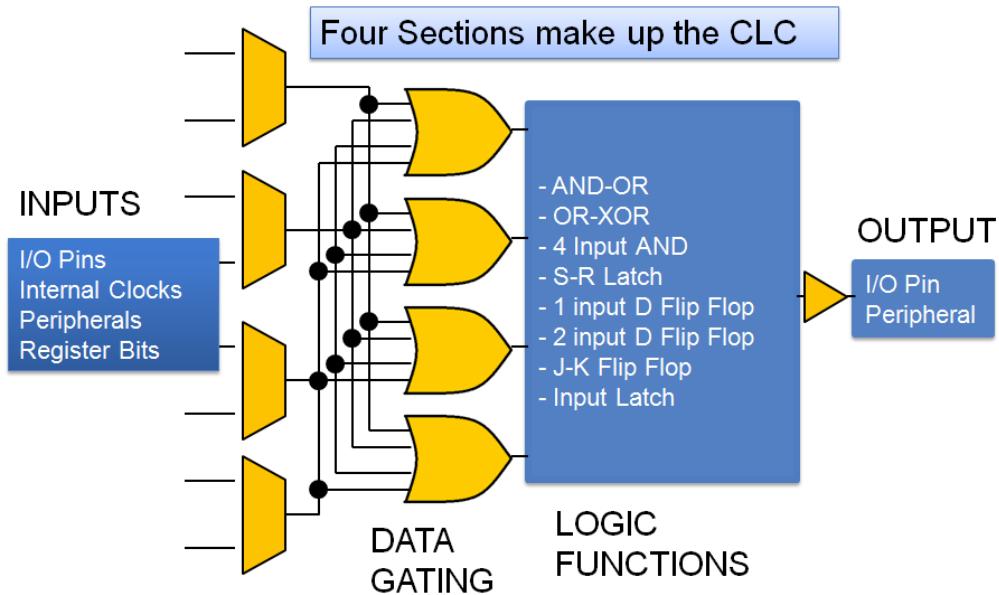
Incluyen:

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

Un dispositivo **PIC®** 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 representan el número CLC (por ejemplo, CLC1 usa el registro CLC1CON).

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

- Entradas
- puerta de datos
- función lógica
- Configuración de salida



(https://microchipdeveloper.com.translate.goog/local--files/8bit:clc/clcsetup.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

Entradas

Las entradas pueden provenir de 8 a 16 fuentes diferentes, según el dispositivo PIC, y de esta lista, se pueden elegir hasta cuatro para alimentar la sección de activación de datos.

Pueden incluir:

- pines de E/S
- Salidas de reloj internas
- Salidas de periféricos
- Bits de registro

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



(https://microchipdeveloper-com.translate.goog/local--files/8bit:clc/CLCselregisters.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

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 |

(https://microchipdeveloper-com.translate.goog/local--files/8bit:clc/CLCinputs.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es)

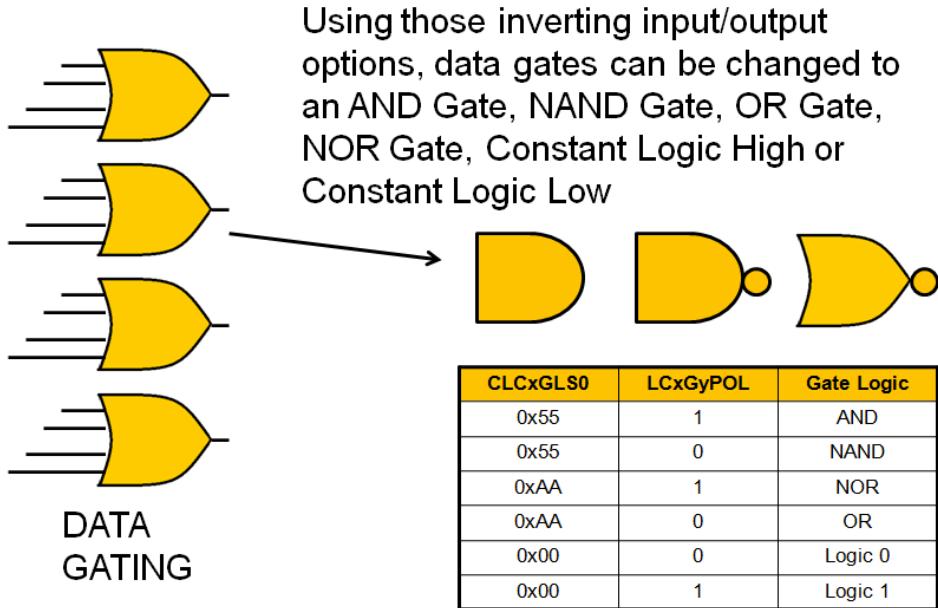
puerta de datos

La sección de puerta de datos tiene cuatro puertas lógicas que deben configurarse. Esto requiere que se configuren 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 puede invertirse o no invertirse individualmente.

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



(https://microchipdeveloper.com.translate.google/local--files/8bit:clc/datagating.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

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 el bit T está establecido, la entrada no está invertida. Si se establece el

bit N, la entrada se invierte. Si ambos se ponen a cero, entonces la entrada no está conectada a la puerta.

CLC1POL

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

CLC1GLS0

| | | | | | | | |
|----------|----------|----------|----------|----------|----------|----------|----------|
| LC1G1D4T | LC1G1D4N | LC1G1D3T | LC1G1D3N | LC1G1D2T | LC1G1D2N | LC1G1D1T | LC1G1D1N |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

Input 4

Input 3

Input 2

Input 1

(https://microchipdeveloper-com.translate.goog/local--files/8bit:clc/clcglsrcsregisters.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

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

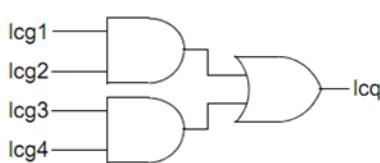
0 - no invertido

1 - invertido

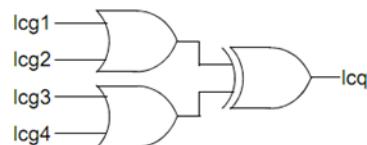
función lógica

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

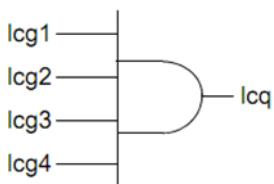
000 = AND – OR



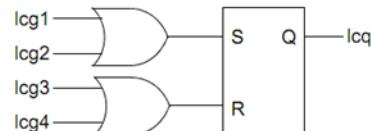
001 = OR – XOR



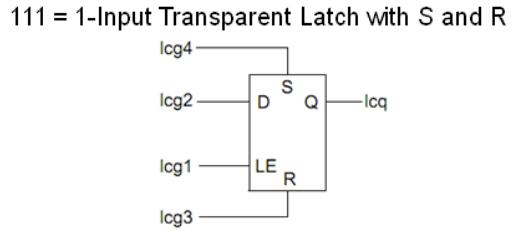
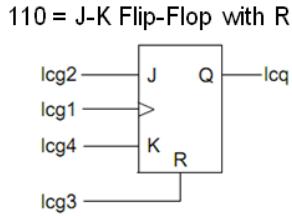
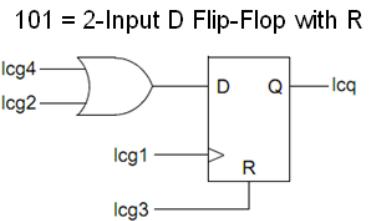
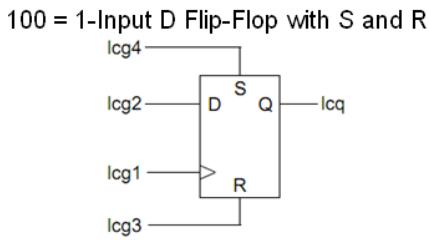
010 = 4-Input AND



011 = S-R Latch



(https://microchipdeveloper-com.translate.goog/local--files/8bit:clc/logic1.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)



(https://microchipdeveloper.com.translate.goog/local--files/8bit:clc/logic2.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

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

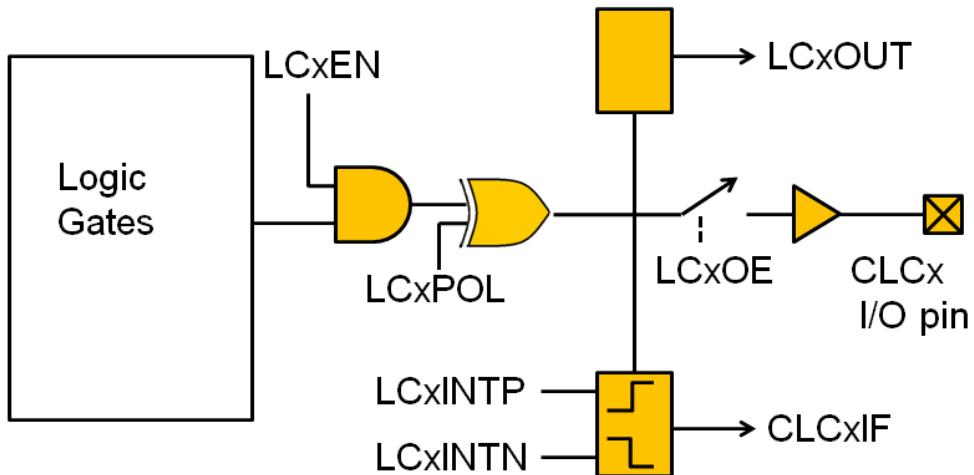


(https://microchipdeveloper.com.translate.goog/local--files/8bit:clc/clcccon.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

Producción

Todas las secciones CLC se reducen a una sola salida que puede controlar un pin de E/S, alimentar otro módulo CLC o periférico interno, o también puede activar 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



(https://microchipdeveloper.com.translate.goog/local--files/8bit:clc/output.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

Los bits en el registro CLCxCON controlan la configuración de salida.

LCxEN: bit de habilitación del módulo CLC (1 - CLC activado, 0 - desactivado)

LCxOE: bit de habilitación de salida (1: habilitar, 0: deshabilitar)

LCxOUT: monitorizar internamente la salida a través del software (bit de solo lectura)

LCxINTP: habilitación de interrupción de borde ascendente (1 - CLCxIF configurado en el flanco ascendente)

LCxINTN: habilitación de la interrupción del flanco descendente (1 - CLCxIF configurado en el flanco descendente)

CLCxCON

| LCxEN | LCxOE | LCxOUT | LCxINTP | LCxINTN | LCxMODE2 | LCxMODE1 | LCxMODE0 |
|-------|-------|--------|---------|---------|----------|----------|----------|
| 0 | 1 | X | 1 | 0 | — | — | — |

P - Rising Edge Interrupt
N - Falling Edge Interrupt

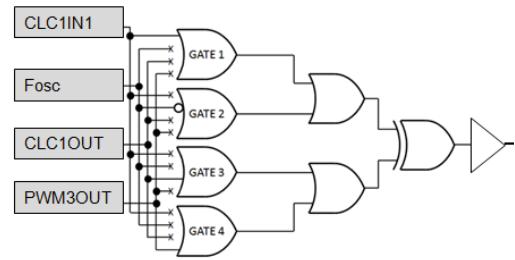
(https://microchipdeveloper.com.translate.goog/local--files/8bit:clc/clcconoutput.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

Ejemplo de CVX

Aquí hay un ejemplo simple que muestra los ocho registros configurados en el software para crear la configuración 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
```



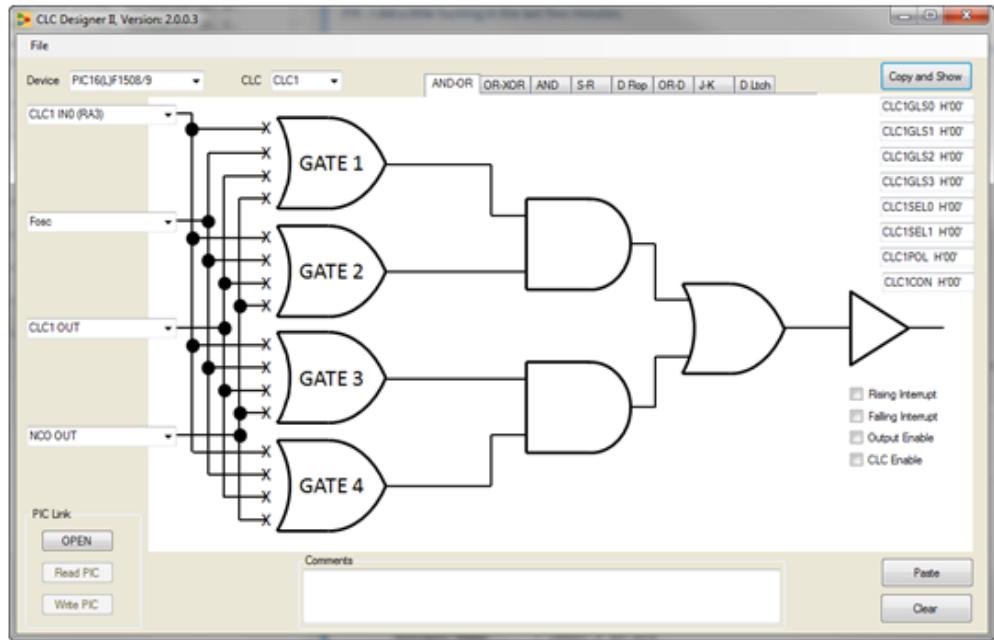
(https://microchipdeveloper-com.translate.goog/local--files/8bit:clc/clcexample.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

Herramienta de diseño de CLC

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



La herramienta CLC Designer es parte del complemento MPLAB Code Configurator (MCC) MPLAB® X.



(https://microchipdeveloper.com.translate.goog/local--files/8bit:clc/clcgui.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hi=es-419&_x_tr_pto=sc)

...Herramienta de configuración de CLC

Configurable Logic Cell (CLC)



La celda lógica configurable (CLC) proporciona lógica programable que opera fuera de las limitaciones de velocidad de ejecución del software. La celda lógica acepta hasta 16 señales de entrada. Mediante el uso de puertas configurables, reduce las 16 entradas a cuatro líneas lógicas que controlan 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.

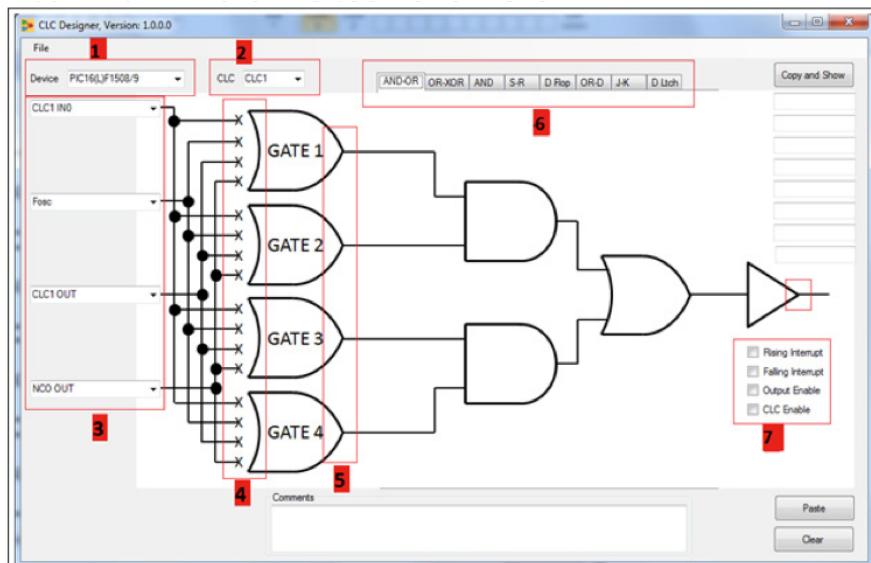
El propósito de la herramienta de configuración CLC es optimizar el proceso de configuración del módulo CLC simulando la funcionalidad de los registros en una

interfaz gráfica de usuario (GUI). El resultado final del uso de la herramienta será un código fuente generado, que se puede colocar en un archivo de proyecto **MPLAB® X existente**. El ejemplo de código creado se genera de forma personalizada, según sus entradas y preferencias, como el lenguaje de programación.

La herramienta CLC es 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 CLC. A continuación se incluye una guía escrita paso a paso.

Pasos de configuración de la GUI de CLC



(https://microchipdeveloper-com.translate.goog/local--files/8bit:clcgui/CLCGUI_Highlight.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

1 Seleccione el dispositivo

Aquí es donde se seleccionará el dispositivo, como el PIC16F1508 (<https://translate.google.com/website?sl=en&tl=es&hl=es-419&prev=search&u=https://www.microchip.com/wwwproducts/en/PIC16F1508>) . Cuando se selecciona un dispositivo, el programa se configurará automáticamente para ese dispositivo específico, como las entradas de datos y la cantidad de salidas CLC disponibles.

2 Seleccione el Número CLC

Un dispositivo puede tener varios CLC, por lo que deberá seleccionar el número de CLC que desea configurar. Algunos dispositivos, como el PIC10F320 (<https://translate.google.com/website?sl=en&tl=es&hl=es-419&prev=search&u=https://www.microchip.com/wwwproducts/en/PIC10F320>) , solo tendrán un módulo CLC disponible en el dispositivo seleccionado. La "X" en cada registro CLC será reemplazada por el módulo CLC que se utilice.

3 Seleccionar 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 todos los grupos. 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 entrada. 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 desplace el cursor sobre la "X" deseada y haga clic 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 nuevamente donde estaba la "X" y ahora debería aparecer una burbuja, indicando una inversión. Si se hace clic una vez más, la burbuja y la línea deberían desaparecer y regresar por defecto al estado original desconectado.

5 Salidas de puerta

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, vuelva a hacer clic en la burbuja para que desaparezca. Es importante tener en cuenta que cualquier puerta sin entradas seleccionadas tendrá su salida predeterminada en el estado "apagado", 0 lógico. Si se desea un 1 lógico constante, invierta el 0 lógico predeterminado haciendo clic en la salida de la burbuja inversora.

6 Seleccionar el bloque lógico

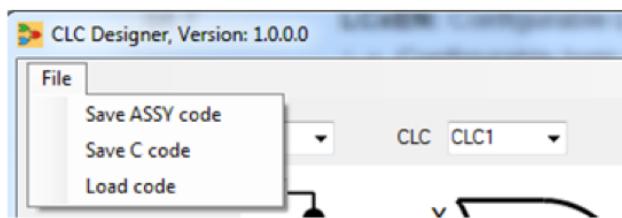
Hay ocho funciones lógicas disponibles seleccionadas por las pestañas de la herramienta CLC. Los bloques lógicos no se pueden configurar para nada que no sea lo que se muestra. Solo se puede usar una función lógica a la vez para cada módulo CLC.

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 una vez en el pin de salida del búfer para que aparezca una burbuja. Desde aquí, la salida se puede enrutar a otros periféricos, un pin de salida o volver a la entrada CLC. Se puede habilitar una interrupción en el flanco 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 CLC proporciona un método conveniente para guardar el diseño (o cargar un diseño anterior). Cuando el diseño haya concluido y esté listo para implementarse en el software, haga clic en el menú desplegable **Archivo** en la esquina superior izquierda del cuadro de diálogo.



(https://microchipdeveloper-com.translate.goog/local--files/8bit:clcgui/CLCGUI_Output.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

Luego, haga clic en **Guardar código ASSY** o **Guardar código C**, según el 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 lenguaje. 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.

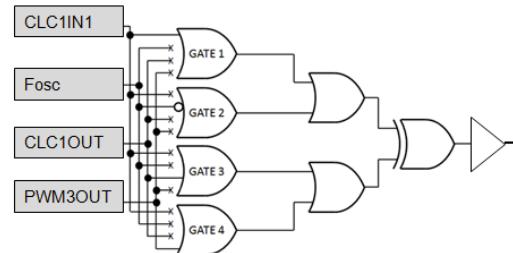
- **Cargar un archivo existente:** para cargar código previamente guardado desde la herramienta CLC, haga clic en **Archivo > Cargar > Código** . Si se importó correctamente, la herramienta habrá llenado la GUI con los valores apropiados correspondientes a los registros en el código cargado.

Ejemplo

Aquí hay un ejemplo simple creado con la herramienta de configuración CLC. Muestra que los ocho registros se han configurado 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
```



(https://microchipdeveloper.com.translate.goog/local--files/8bit:clc/clcexample.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc)

Descarga/Documentación

La herramienta de configuración CLC es parte del configurador de código MPLAB. Puede obtener la Herramienta de configuración de CLC, el Manual del usuario y la Guía de trucos y sugerencias de CLC directamente desde los siguientes enlaces:



La herramienta CLC ahora está incorporada en la herramienta MPLAB Code Configurator (MCC). Puede encontrar información sobre cómo instalar la herramienta MCC visitando la página MPLAB Code Configurator (MCC) (https://microchipdeveloper-com.translate.goog/mcc:overview?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc) .

- *"Guía del usuario de la herramienta de configuración de celdas lógicas configurables (CLC)"* (<https://translate.google.com/website?sl=en&tl=es&hl=es-419&prev=search&u=http://www.microchip.com/mymicrochip/filehandler.aspx?ddocname%3Den555291>)
- *"Sugerencias y trucos de celdas lógicas configurables"* (<https://translate.google.com/website?sl=en&tl=es&hl=es-419&prev=search&u=http://www.microchip.com/mymicrochip/filehandler.aspx?ddocname%3Den557835>)

Registro CLCxCON

La sección de salida y la sección lógica de la celda lógica configurable (CLC) (<https://microchip-dev.wikidot.com/8bit:clc>) están controladas por el registro CLCCON.

CLCxCON: Registro de control de celda lógica configurable

| L/E- 0/0 | L/E- 0/0 | L/E-0/0 | L/E-0/0 | L/E-0/0 | L/E-0/0 | L/E-0/0 | L/E-0/0 |
|-------------|-------------|-----------|---------|---------|----------|----------|----------|
| LCxES | LCxOE | LCxSALIDA | LCxINTP | LCxINTN | LCxMODE2 | LCxMODE1 | LCxMODE0 |

bit 7

bit 0

| Leyenda | | |
|-------------------------------|---------------------------|--|
| R = Bit legible | W = bit grabable | U = bit no implementado, leído como '0' |
| u = El bit no cambia | x = el bit es desconocido | -n/n = Valor en POR y BOR/Valor en todos los demás reinicios |
| '1' = el bit está establecido | '0' = el bit se borra | -n = Valor en el reinicio de POR |

bit 7 **LCxEN:** bit de habilitación de CLC

1 = CLC está habilitado y mezcla señales de entrada
0 = CLC está deshabilitado y tiene salida lógica cero

bit 6 **LCxEOE:** bit de habilitación de salida CLC

1 = Salida de pin de puerto CLC habilitada
0 = Salida de pin de puerto CLCI deshabilitada

bit 5 **LCxOUT:** bit de salida de datos CLC

Solo lectura: datos de salida de celda lógica, después de LCxPOL; muestreado del cable lcx_out

bit 4 **LCxINTP:** bit de habilitación de interrupción de avance de flanco positivo CLC

1 = LCxIF se establecerá cuando se produzca un flanco ascendente en lcx_out
0 = LCxIF no se establecerá

bit 3 **LCxINTN:** bit de habilitación de interrupción de flanco negativo CLC

1 = LCxIF se establecerá cuando se produzca un flanco descendente en lcx_out

$0 = \overline{LCxIF}$ no se establecerá

bit 2-0 **LCxMODE<2:0>**: Bits de modo funcional CLC

111 = La celda es un pestillo transparente de 1 entrada con S y R

110 = La celda es un Flip-Flop JK con R

101 = La celda es un Flip-Flop D de 2 entradas con R

100 = La celda es un Flip-Flop D de 1 entrada con S y R

011 = La celda es SR latch

010 = La celda es AND de 4 entradas

001 = La celda es OR-XOR

000 = La celda es AND-OR

De la hoja de datos **PIC16F1507**

(<http://ww1.microchip.com/downloads/en/DeviceDoc/40001586D.pdf>) .

El bit de habilitación de CLC 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

(/local--files/8bit:clccon/LCEN.png)

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

bit 6 **LCxOE**: Configurable Logic Cell Output Enable bit

1 = Configurable logic cell port pin output enabled

0 = Configurable logic cell port pin output disabled

(/local--files/8bit:clccon/LCOE.png)

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

bit 5 **LCxOUT**: Configurable Logic Cell Data Output bit

Read-only: logic cell output data, after LCxPOL; sampled from lcx_out wire.

(/local--files/8bit:clccon/LCOUT.png)

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

bit 4 **LCxINTP**: 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

(/local--files/8bit:clccon/LCINTP.png)

El bit CLC LCINTN habilita la interrupción de flanco descendente en el CLC. Cuando está habilitado (establecido en 1), el CLC activará una interrupción cuando la salida del CLC caiga del estado alto al bajo.

bit 3 **LCxINTN:** 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

(/local--files/8bit:clccon/LCINTN.png)

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

bit 2-0 **LCxMODE<2:0>:** 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

(/local--files/8bit:clccon/LCMODE.png)

Registros CLCxSELn

Los registros CLCxSEL, contenidos en la celda lógica configurable (CLC) (<https://microchip-dev.wikidot.com/8bit:clc>) , controlan qué entradas se usan con el CLC.

Fuentes de entrada CLC

El CLC tendrá múltiples entradas para seleccionar y cada una tendrá un código de 3 bits asociado, como se muestra en la siguiente tabla. 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 | lcxd1 D1S | lcxd2 D2S | lcxd3 D3S | lcxd4 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 | — | lcx1_out | lcx1_out |
| CLCxIN[9] | — | 101 | 001 | — | lcx2_out | lcx2_out |
| CLCxIN[10] | — | 110 | 010 | — | lcx3_out | lcx3_out |
| CLCxIN[11] | — | 111 | 011 | — | lcx4_out | lcx4_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 |

(/local--files/8bit:clcsel/CLCInputs.png)

De la hoja de datos PIC16F1507

(<http://ww1.microchip.com/downloads/en/DeviceDoc/40001586D.pdf>) .

Las selecciones de entrada están controladas por los registros CLCxSEL0 y CLCxSEL1 configurando 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: MULTIPLEXOR DATOS 1 Y 2 SELECCIONAR REGISTRO

| 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
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 | LCxD2S<2:0>: Input Data 2 Selection Control bits ⁽¹⁾ 111 = CLCxIN[11] is selected for lcx2 110 = CLCxIN[10] is selected for lcx2 101 = CLCxIN[9] is selected for lcx2 100 = CLCxIN[8] is selected for lcx2 011 = CLCxIN[7] is selected for lcx2 010 = CLCxIN[6] is selected for lcx2 001 = CLCxIN[5] is selected for lcx2 000 = CLCxIN[4] is selected for lcx2 |
| bit 3 | Unimplemented: Read as '0' |
| bit 2-0 | LCxD1S<2:0>: Input Data 1 Selection Control bits ⁽¹⁾ 111 = CLCxIN[7] is selected for lcx1 110 = CLCxIN[6] is selected for lcx1 101 = CLCxIN[5] is selected for lcx1 100 = CLCxIN[4] is selected for lcx1 011 = CLCxIN[3] is selected for lcx1 010 = CLCxIN[2] is selected for lcx1 001 = CLCxIN[1] is selected for lcx1 000 = CLCxIN[0] is selected for lcx1 |

(/local--files/8bit:clcsel/CLCSEL0.png)

De la hoja de datos PIC16F1507

(<http://ww1.microchip.com/downloads/en/DeviceDoc/40001586D.pdf>) .

-
- 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: MULTIPLEXOR DATOS 3 Y 4 SELECCIONAR REGISTRO

| 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 | 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 | LCxD4S<2:0>: Input Data 4 Selection Control bits ⁽¹⁾ |
| | 111 = CLCxIN[3] is selected for lcx4 |
| | 110 = CLCxIN[2] is selected for lcx4 |
| | 101 = CLCxIN[1] is selected for lcx4 |
| | 100 = CLCxIN[0] is selected for lcx4 |
| | 011 = CLCxIN[15] is selected for lcx4 |
| | 010 = CLCxIN[14] is selected for lcx4 |
| | 001 = CLCxIN[13] is selected for lcx4 |
| | 000 = CLCxIN[12] is selected for lcx4 |
| bit 3 | Unimplemented: Read as '0' |
| bit 2-0 | LCxD3S<2:0>: Input Data 3 Selection Control bits ⁽¹⁾ |
| | 111 = CLCxIN[15] is selected for lcx3 |
| | 110 = CLCxIN[14] is selected for lcx3 |
| | 101 = CLCxIN[13] is selected for lcx3 |
| | 100 = CLCxIN[12] is selected for lcx3 |
| | 011 = CLCxIN[11] is selected for lcx3 |
| | 010 = CLCxIN[10] is selected for lcx3 |
| | 001 = CLCxIN[9] is selected for lcx3 |
| | 000 = CLCxIN[8] is selected for lcx3 |

(/local--files/8bit:clcsel/CLCSEL1.png)

De la hoja de datos PIC16F1507

(<http://ww1.microchip.com/downloads/en/DeviceDoc/40001586D.pdf>).

Registros CLCxGLSn

Los registros CLCxGLSn, contenidos en la celda lógica configurable (CLC) (<https://microchip-dev.wikidot.com/8bit:clc>) , controlan la polaridad de las entradas CLC seleccionadas.

Puerta de datos CLC

Las salidas de los multiplexores de entrada se dirigen a la entrada de función lógica deseada a través de la etapa de puerta de datos. Cada puerta de datos puede dirigir cualquier combinación de cuatro entradas seleccionadas. La puerta se puede configurar para dirigir cada señal de entrada como datos invertidos o no invertidos. Las señales dirigidas se combinan en OR 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 (<https://microchip-dev.wikidot.com/8bit:clcpol>) .

La sección de puerta de datos está controlada por uno de los cuatro registros. Cada puerta tiene un registro separado. Cada entrada tiene un bit "N" y un bit "P". Establecer el bit "N" invierte la entrada y establecer el bit "P" hace que no se invierta. Si no se establece ninguno, 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 | LCxG1D4T: Gate 1 Data 4 True (non-inverted) bit 1 = lcxd4T is gated into lcxg1 0 = lcxd4T is not gated into lcxg1 |
| bit 6 | LCxG1D4N: Gate 1 Data 4 Negated (inverted) bit 1 = lcxd4N is gated into lcxg1 0 = lcxd4N is not gated into lcxg1 |
| bit 5 | LCxG1D3T: Gate 1 Data 3 True (non-inverted) bit 1 = lcxd3T is gated into lcxg1 0 = lcxd3T is not gated into lcxg1 |
| bit 4 | LCxG1D3N: Gate 1 Data 3 Negated (inverted) bit 1 = lcxd3N is gated into lcxg1 0 = lcxd3N is not gated into lcxg1 |
| bit 3 | LCxG1D2T: Gate 1 Data 2 True (non-inverted) bit 1 = lcxd2T is gated into lcxg1 0 = lcxd2T is not gated into lcxg1 |
| bit 2 | LCxG1D2N: Gate 1 Data 2 Negated (inverted) bit 1 = lcxd2N is gated into lcxg1 0 = lcxd2N is not gated into lcxg1 |
| bit 1 | LCxG1D1T: Gate 1 Data 1 True (non-inverted) bit 1 = lcxd1T is gated into lcxg1 0 = lcxd1T is not gated into lcxg1 |
| bit 0 | LCxG1D1N: Gate 1 Data 1 Negated (inverted) bit 1 = lcxd1N is gated into lcxg1 0 = lcxd1N is not gated into lcxg1 |

(/local--files/8bit:clcglis/CLCGGLS.png)

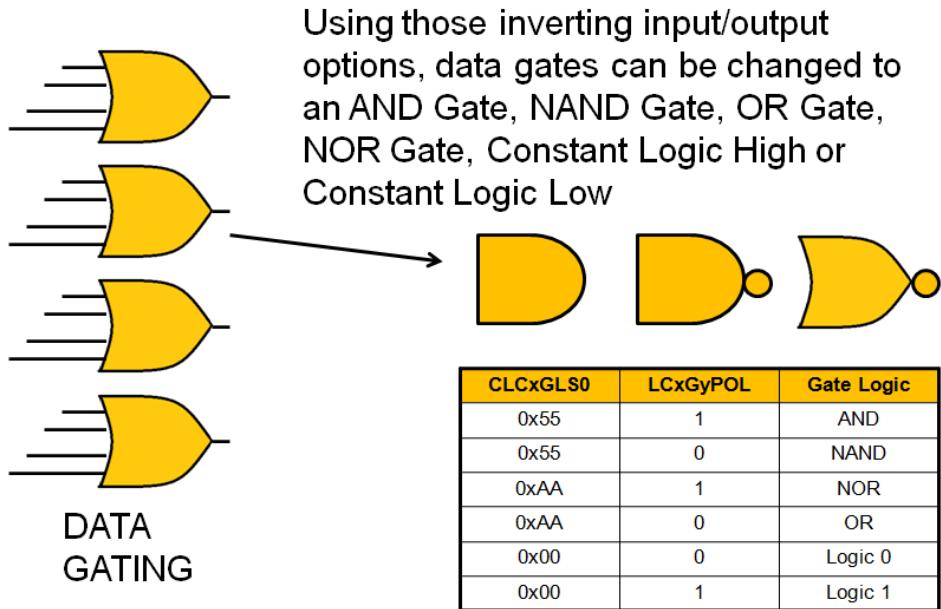
Figura 1

De la hoja de datos **PIC16F1507** (<http://ww1.microchip.com/downloads/en/DeviceDoc/40001586D.pdf>) .

Crear varias puertas

Cada compuerta de datos es, en esencia, una compuerta AND/NAND/OR/NOR de 1 a 4 entradas según 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 usando los bits de selección de lógica de puerta.



(/local--files/8bit:clc/datagating.png)

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 seleccionan entradas, la salida será 0 o 1, según el bit de polaridad de salida de la puerta. La polaridad de salida es controlada por el registro.

CLC1POL

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

CLC1GLS0

| | | | | | | | |
|----------|----------|----------|----------|----------|----------|----------|----------|
| LC1G1D4T | LC1G1D4N | LC1G1D3T | LC1G1D3N | LC1G1D2T | LC1G1D2N | LC1G1D1T | LC1G1D1N |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

Input 4

Input 3

Input 2

Input 1

(/local--files/8bit:clc/clcglsrcisters.png)

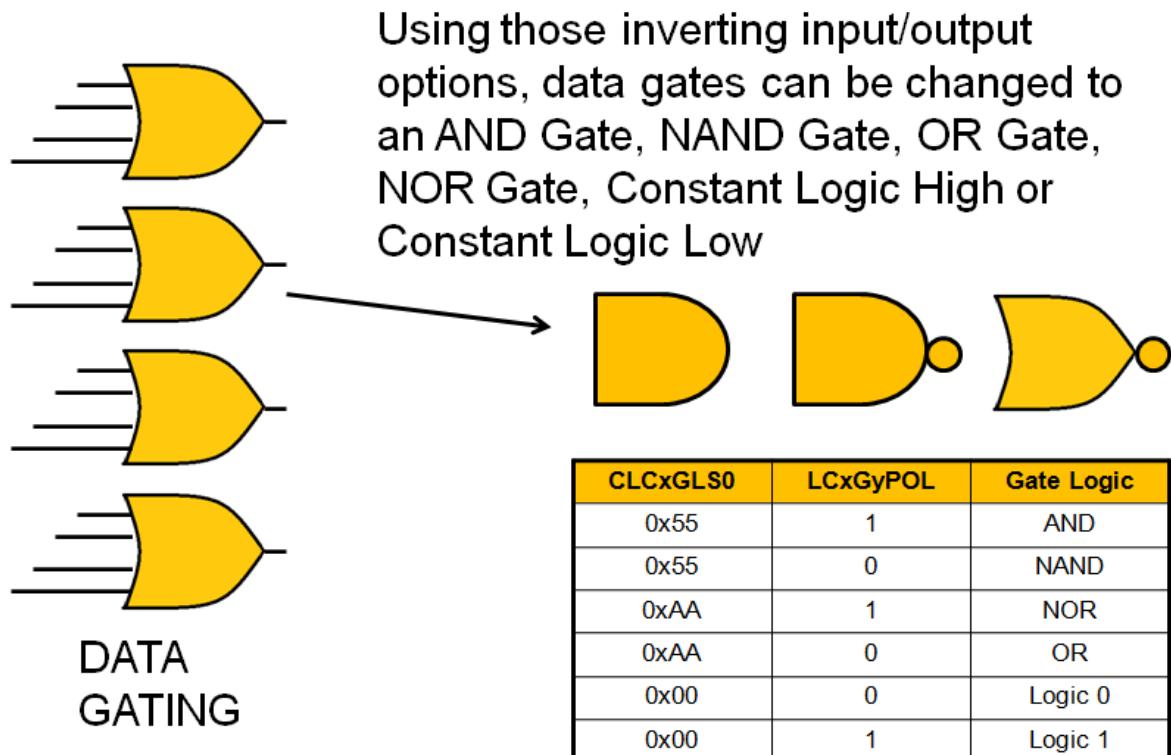
figura 3

Registro CLCxPOL

El registro CLCxPOL, contenido en la celda lógica configurable (CLC) (<https://microchip-dev.wikidot.com/8bit:clc>) , controla la polaridad de las salidas de puerta 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 usando los bits de selección de lógica de puerta.



(/link)

La tabla muestra la lógica de cuatro variables de entrada, pero cada puerta se puede configurar para usar menos de cuatro. Si no se seleccionan entradas, 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

(/link)

Control de salida CLC

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

CLC1POL

| | | | | | | | |
|--------|---|---|---|----------|----------|----------|----------|
| LC1POL | - | - | - | LC1G4POL | LC1G3POL | LC1G2POL | LC1G1POL |
|--------|---|---|---|----------|----------|----------|----------|

(/link)

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 a partir de una selección de fuentes de entrada.

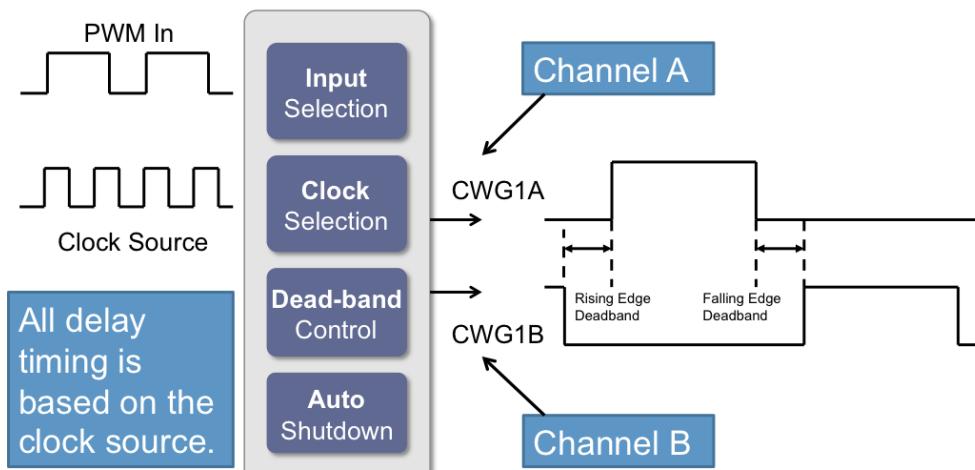
El módulo CWG tiene las siguientes características:

- 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 flanco ascendente y descendente de 6 bits
- Control de apagado automático con:
 - Fuentes de apagado seleccionables
 - Habilitar reinicio automático
 - Control de anulación de 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 se puede retrasar con respecto a la transición de encendido a apagado de la otra salida, creando así un retraso de tiempo inmediato en el que no se activa ninguna salida. Esto se conoce como tiempo muerto o banda muerta y se trata 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, el variador activo puede terminarse antes de que la condición de falla cause daño. 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.



(/local--files/8bit:cwg/cwg.png)

El CWG requiere que se establezcan cinco secciones:

- Aporte
- Reloj
- banda muerta
- Cerrar
- Control de salida

Videotutorial de CWG

Este video presenta el generador de formas de onda complementarias (CWG) para dispositivos MCU de 8 bits de Microchip y muestra cómo usarlo.

Microchip Self Paced Training - Complimentary Waveform Gene...



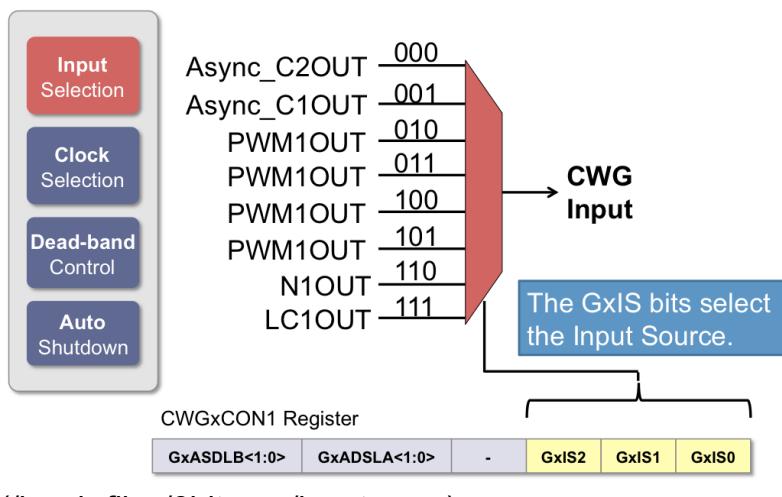
Fuente de entrada

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

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

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

Algunos dispositivos también incluyen la salida de un módulo comparador como entrada al CWG. Lo mejor es consultar la hoja de datos del dispositivo que está utilizando para ver la lista actualizada de las opciones de entrada seleccionadas.



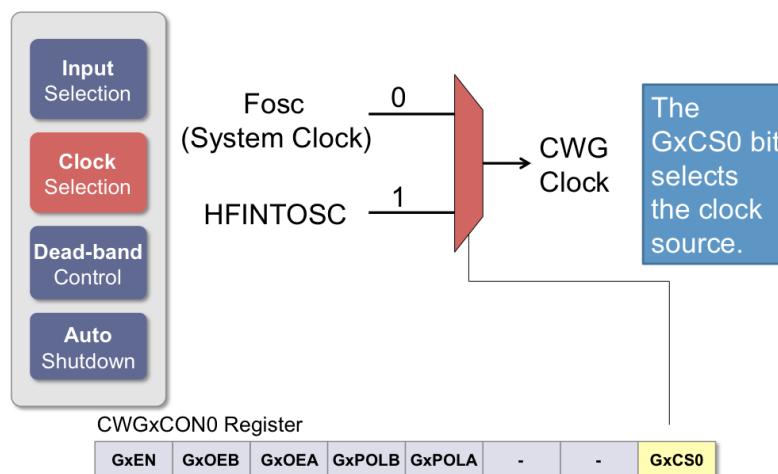
(/local--files/8bit:cwg/inputs.png)

Fuente de reloj

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

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

Las fuentes de reloj se seleccionan utilizando el bit G1CS0 del registro CWGxCON0.

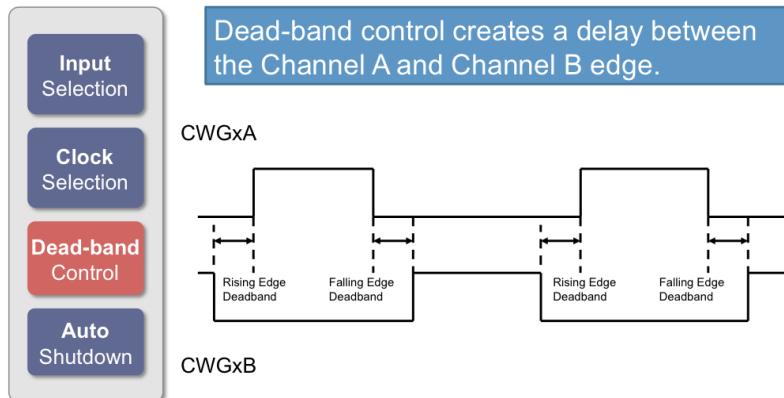


(/local--files/8bit:cwg/clocksource.png)

Control de banda muerta

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

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



(/local--files/8bit:cwg/deadband.png)

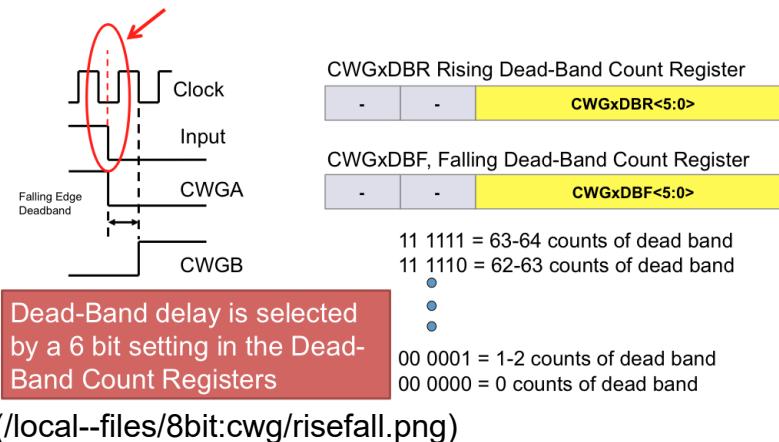
Control de borde ascendente

La banda muerta del flanco ascendente retrasa el encendido de la salida CWGxA desde que se apaga la salida CWGxB. El tiempo de banda muerta del flanco ascendente comienza cuando el flanco ascendente de la señal de la fuente de entrada se vuelve verdadero. Cuando esto sucede, la salida CWGxB se apaga inmediatamente

y comienza el tiempo de retardo de banda muerta de flanco ascendente. Cuando se alcanza el tiempo de retardo de banda muerta del flanco ascendente, se activa la salida CWGxA.

El registro CWGxDBR establece la duración del intervalo de banda muerta en el flanco ascendente de la señal de fuente de entrada. Esta duración es de 0 a 64 conteos de banda muerta. La banda muerta siempre se cuenta desde el borde de la señal de la fuente de entrada. Un conteo 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 conteo, 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.

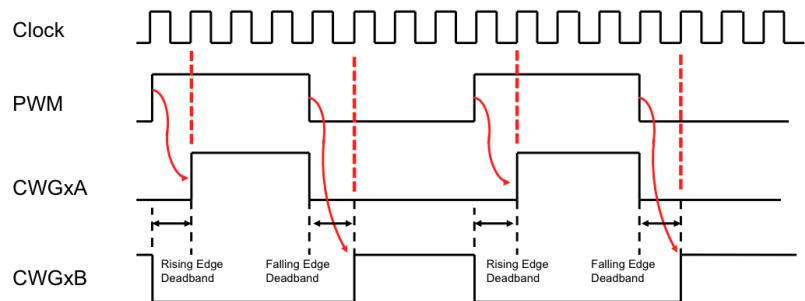


Control de borde descendente

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

El registro CWGxDBF establece la duración del intervalo de banda muerta en el flanco descendente de la señal de fuente de entrada. Esta duración es de 0 a 64 conteos de banda muerta. La banda muerta siempre se cuenta desde el borde de la señal de la fuente de entrada. Un conteo 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 conteo, no se verá ninguna salida en la salida respectiva.

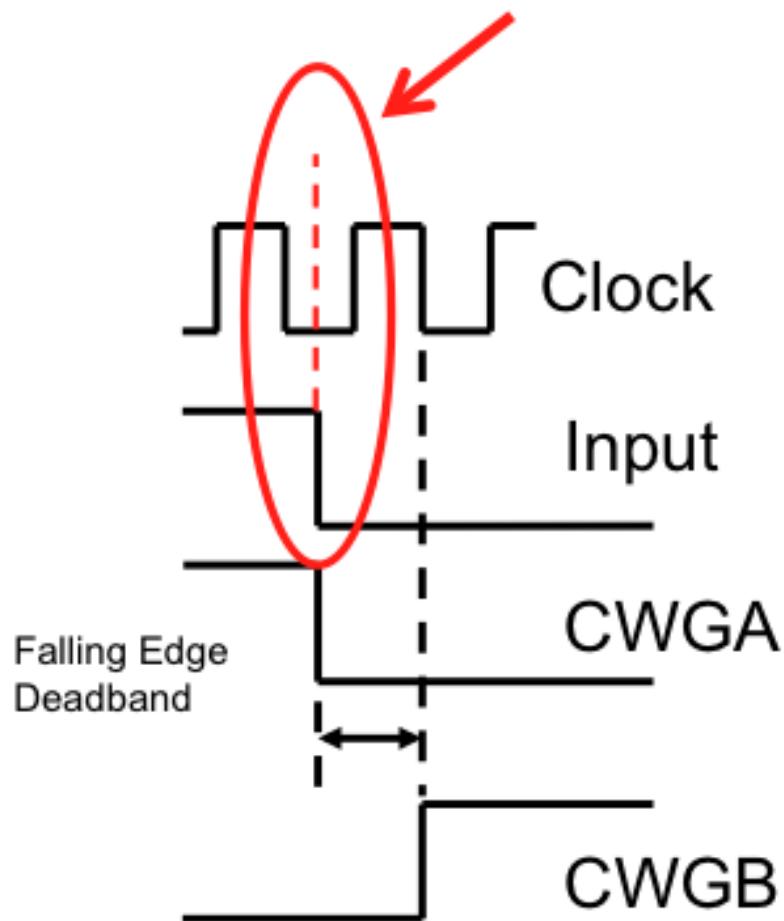
Using a one clock pulse delay for rising and falling edge Dead-Band control, the PWM signal would produce two waveforms similar to what is shown here.



(/local--files/8bit:cwg/dbexample.png)

Incertidumbre de banda muerta

Cuando los flancos ascendente y descendente de la fuente de entrada activan los contadores de banda muerta, la entrada puede ser asíncrona con la entrada del reloj. Esto creará cierta incertidumbre en el retardo de tiempo de banda muerta. La incertidumbre máxima es igual a un período de reloj CWG.



(/local--files/8bit:cwg/uncertainty.png)

Control de apagado automático

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

Se puede ingresar al estado de apagado mediante cualquiera de los dos métodos siguientes:

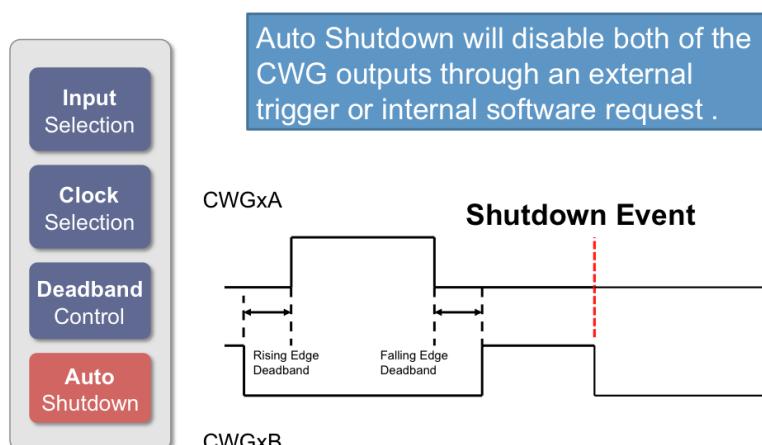
- software generado
- Entrada externa

Software generado

Establecer el bit GxASE del registro CWGxCON2 obligará al CWG a entrar en el estado de apagado. Cuando el reinicio automático está deshabilitado, el estado de apagado persistirá mientras el bit GxASE esté establecido. Cuando el reinicio automático está habilitado, el bit GxASE se borrará automáticamente y reanudará la operación en el próximo evento de flanco ascendente.

Entrada externa

Las señales de apagado externas brindan la forma más rápida de suspender de manera segura el funcionamiento del CWG en caso de una condición de falla. Cuando cualquiera de las señales de apagado seleccionadas se activa, las salidas CWG irán inmediatamente a los niveles de anulación seleccionados sin demora de software. Se puede seleccionar cualquier combinación de dos señales de apagado para provocar una condición de apagado. Las señales de apagado que se ofrecen pueden variar según el dispositivo que se utilice, pero esas fuentes de apagado se seleccionan mediante los bits GxASDS0 y GxASDS1 del registro CWGxCON2.



(/local--files/8bit:cwg/shutdown.png)

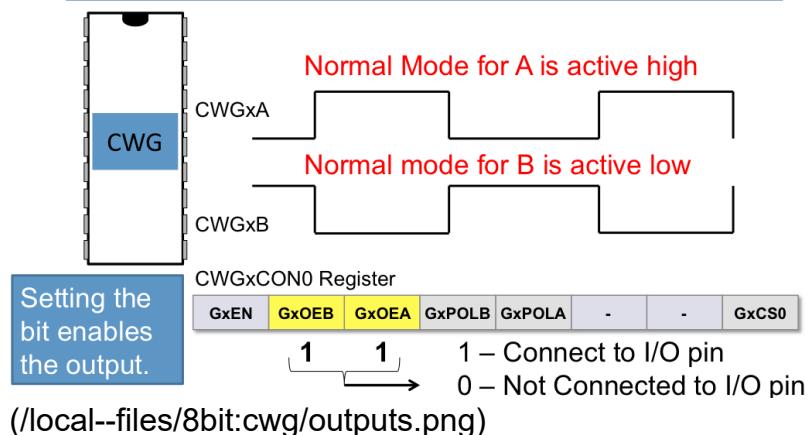
Control de salida

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 la habilitación de un 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 del puerto interno.

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

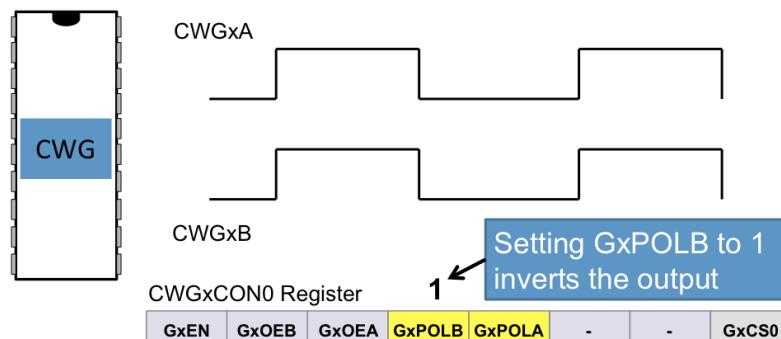
The CWGA and CWGB outputs can be connected to the I/O pin through the CWGCON0 Register setting.



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 está activa alta. Borrar el bit de polaridad de salida configura la salida correspondiente como baja activa. Sin embargo, la polaridad no afecta los niveles de anulación. La polaridad de salida se selecciona con los bits GxPOLB y GxPOLA 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 funcionando durante la suspensión, siempre que el reloj y las fuentes de entrada seleccionadas permanezcan activas. 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 esté seleccionado 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 de reloj del CWG, cuando el CWG esté habilitado y la fuente de entrada esté activa, la CPU quedará inactiva durante la suspensión, pero el CWG seguirá funcionando y el HFINTOSC permanecerá activo. Esto tendrá un efecto directo en la corriente del modo de reposo.

Ejemplo de CWG

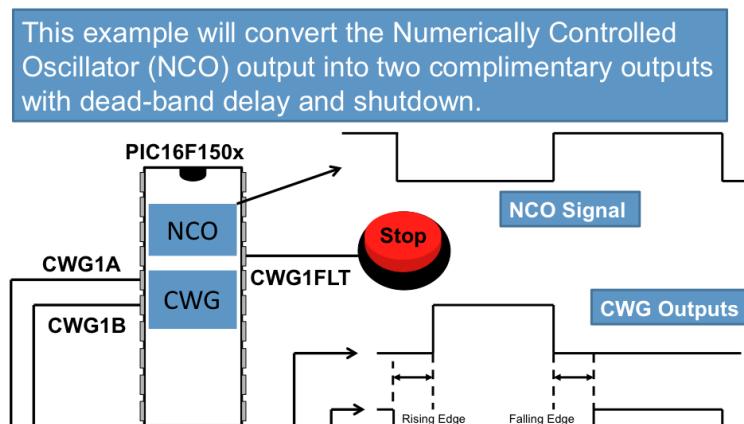
A veces es útil pasar a través de un ejemplo. Visite este ejemplo de CWG (/8bit:cwgexample) para ver instrucciones paso a paso para el módulo CWG.

Ejemplo de generador de forma de onda complementaria

Esta página es un ejemplo paso a paso que utiliza el generador de formas de onda complementarias (CWG).

Este ejemplo utiliza la señal de salida del oscilador controlado numéricamente (NCO) (funcionando 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 interruptor externo como control de apagado, como se muestra en el siguiente diagrama de bloques.

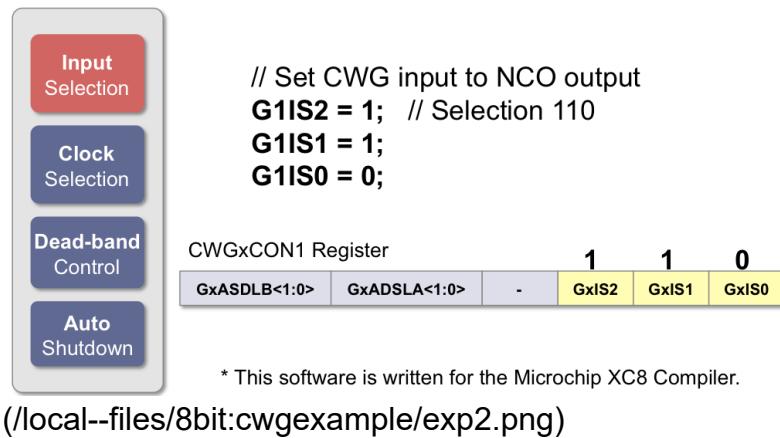


(/local--files/8bit:cwgexample/exp1.png)

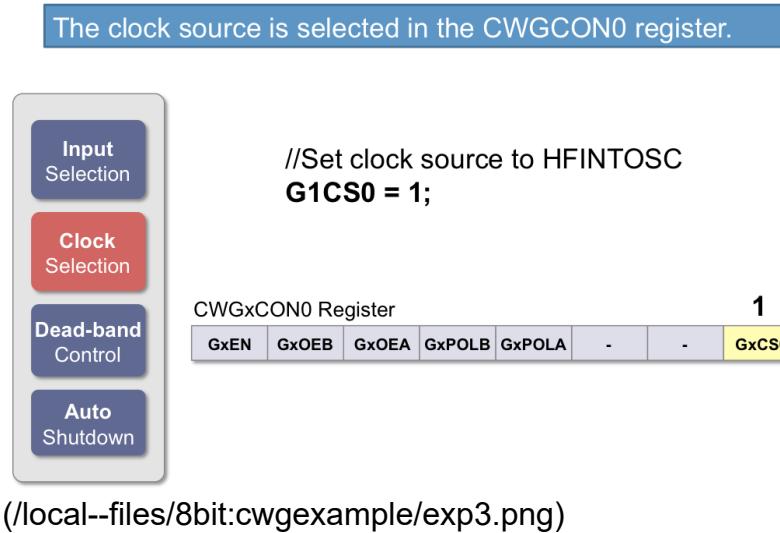
Configuración de ejemplo de CWG

- 1 El **NCO** se selecciona como entrada para el módulo CWG.

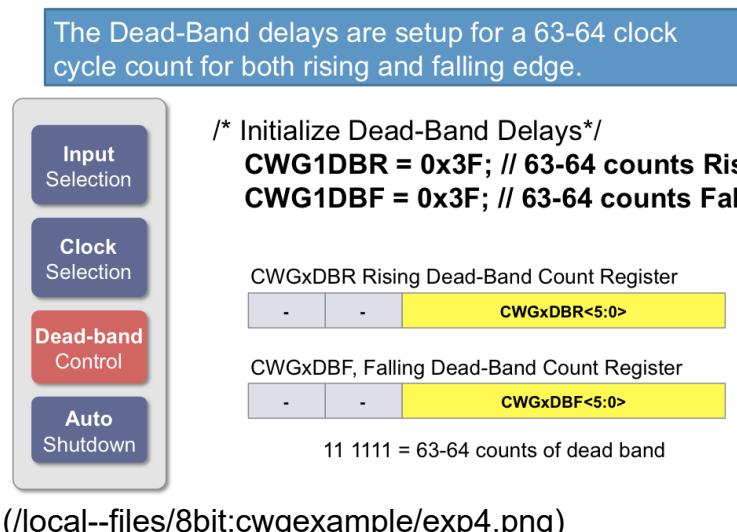
The NCO output is selected as the input signal for the CWG module in the CWGCON1 register.



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

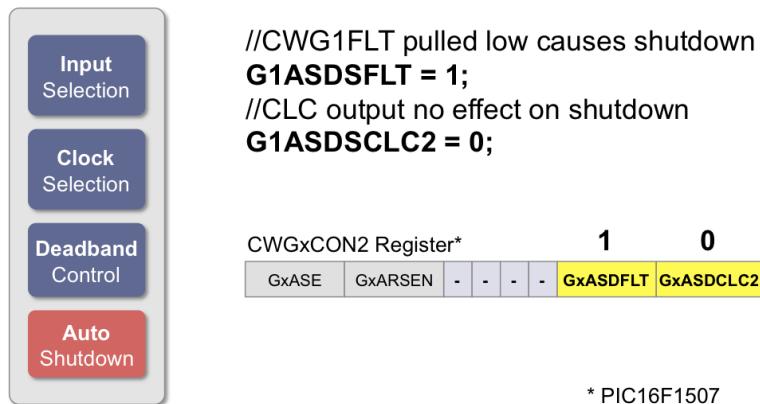


3 Los retrasos de banda muerta se establecen en 63-64 conteos de reloj .



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

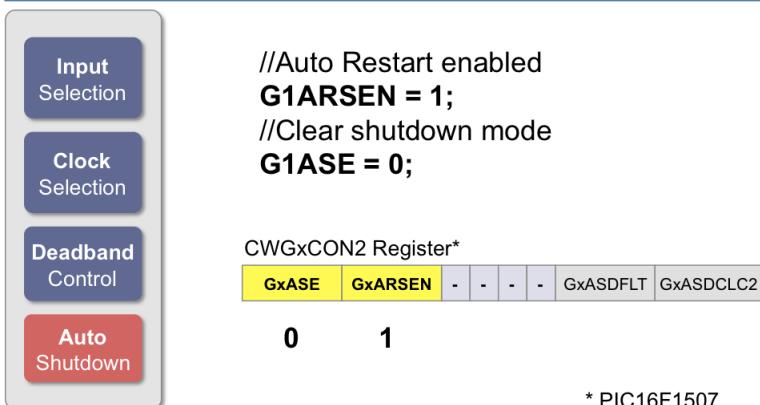
The Shutdown control is set to the CWG1FLT I/O pin.



(/local--files/8bit:cwgexample/exp5.png)

- 5 El **reinicio automático** está habilitado y el **bit de apagado** se borra para comenzar en modo de ejecución.

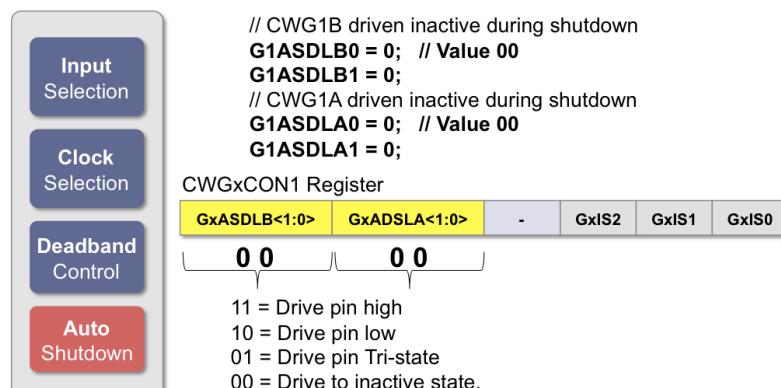
Auto restart is enabled.
The shutdown bit is cleared for proper operation.



(/local--files/8bit:cwgexample/exp6.png)

- 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.



(/local--files/8bit:cwgexample/exp7.png)

- 7 Las **salidas CWG** están conectadas a los pines de E/S y la polaridad de salida está configurada en modo normal.

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

```
G1OEA = 1; //Output signal on CWG1A pin
G1OEB = 1; //Output signal on CWG1B pin
G1POLA = 0; //Output is normal polarity
G1POLB = 0; //Output is normal polarity
```

CWGXCON0 Register

| GxEN | GxOEB | GxOEA | GxPOLB | GxPOLA | - | - | GxCS0 |
|------|-------|-------|--------|--------|---|---|-------|
| 1 | 1 | 0 | 0 | | | | |

(/local--files/8bit:cwgexample/exp8.png)

- 8 A continuación, se habilita el **CWG**. Como referencia, también se muestra el código de configuración NCO 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.

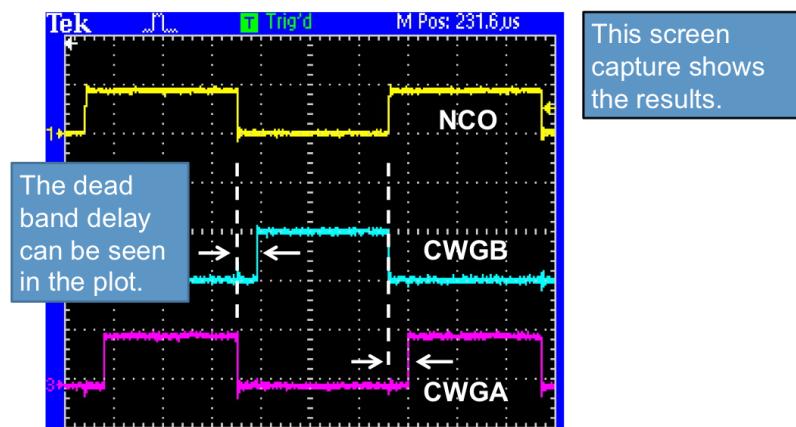
```
NCO1INCH = 8; // load NCO increment high register
NCO1INCL = 65;
NCO1CLK = 0b00000000; // Select HF internal OSC = 16 Mhz
NCO1CON = 0b11010000; // Enable NCO, Enable output,active hi, 50% dc mode
```

(/local--files/8bit:cwgexample/exp9.png)

Operación de ejemplo de CWG

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

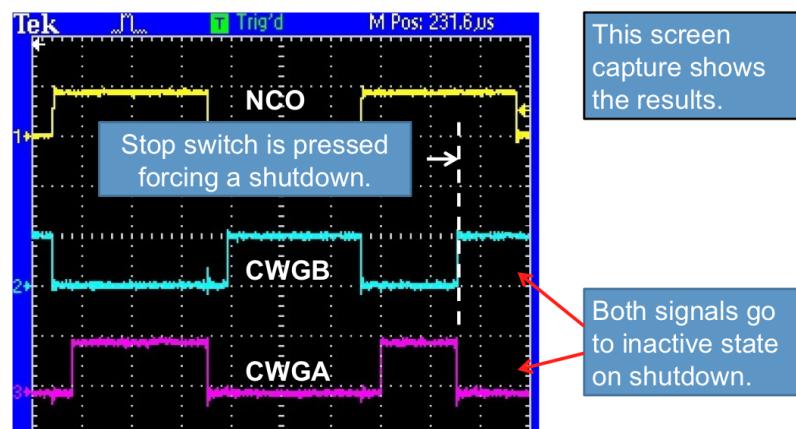
La señal NCO se puede ver en la parte superior de la captura de pantalla junto con las señales CWGA y CWGB. También puede ver los retrasos de banda muerta que se desencadenan por los bordes de la señal de entrada NCO.



(/local--files/8bit:cwgexample/exp10.png)

El CWG se fuerza al modo de apagado.

Cuando se presiona el interruptor momentáneo, las salidas CWG pasan a su estado inactivo, lo que indica que se ha producido el apagado.



(/local--files/8bit:cwgexample/exp11.png)

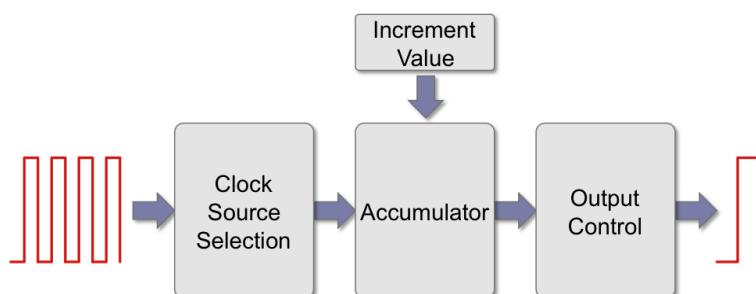
Oscilador controlado numéricamente

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 posterior al escalador. Esto ofrece una ventaja sobre un simple contador accionado por temporizador en el sentido de que la resolución de la división no varía con el valor del divisor del preescalador/postescalador algo limitado. 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 opera agregando repetidamente un valor fijo a un acumulador. Las adiciones ocurren a la velocidad del reloj de entrada. El acumulador se desbordará con un acarreo periódicamente, que es la salida de NCOx sin procesar. Esto reduce efectivamente el reloj de entrada en la proporción del valor agregado al valor máximo del acumulador.



(/local--files/8bit:nco/NCO.png)

La salida 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, se envía 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 promedio. Esta salida depende de la capacidad del circuito receptor (es decir, CWG o circuito convertidor resonante externo) para promediar la salida 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

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

(/local--files/8bit:nco/NCO_formula.png)

Videotutorial para suboficiales

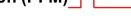
Este video presenta el oscilador controlado numéricamente (NCO) para dispositivos MCU de 8 bits de Microchip y muestra cómo usarlo.

Microchip Self-Paced Training – Numerically Controlled Oscillat...



Modos NCO

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) 

(/local--
 files/8bit:nco/NCO_modes.png)

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

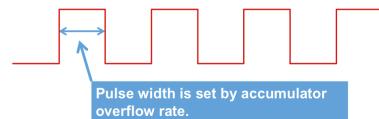
NCOxCON Register

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

(/local--
 files/8bit:nco/ncocon.png)

Ciclo de trabajo fijo

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



(/local--
 files/8bit:nco/NCO_duty.png)

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.

NCOxCLK Register

 Pulse width is set by NxPWS bits.
 NxPWS<2:0> → NCO Clock Periods
 111 = 128 periods 011 = 8 periods
 110 = 64 periods 010 = 4 periods
 101 = 32 periods 001 = 2 periods
 100 = 16 periods 000 = 1 periods

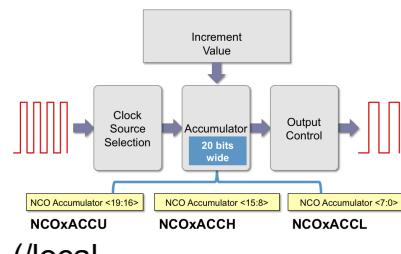
(/local--
 files/8bit:nco/NCO_pulse.png)

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



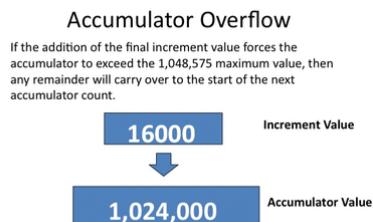
(/local--
files/8bit:nco/NCO_accumulator.png)

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

Sumador

El sumador NCOx es un sumador completo que opera independientemente del reloj del sistema. Agrega el valor del valor de incremento al acumulador en cada pulso de reloj NCO. El sumador toma el valor en el acumulador y luego suma el valor del incremento. El resultado se vuelve a colocar en el acumulador. El valor del acumulador se acumulará y cualquier valor más allá de 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 normalmente se hace dentro de la rutina de servicio de interrupción si está habilitado en el módulo NCO.

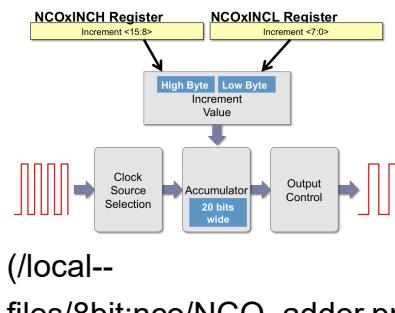


(/local--
files/8bit:nco/NCO_400_animated.gif)

Registros de incremento

El valor de incremento se almacena en dos registros de 8 bits que forman 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
- NCOxPULGADAS



(/local--
files/8bit:nco/NCO_adder.png)

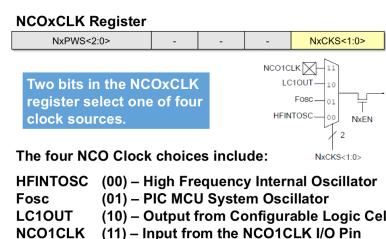
Ambos registros son de lectura y escritura. Los registros de incremento tienen doble búfer para permitir que se realicen cambios de valor sin deshabilitar primero el módulo NCOx. Las cargas del búfer son inmediatas cuando el módulo está deshabilitado. Primero es necesario escribir en el registro NCOxINCH porque luego el búfer se carga sincrónicamente con la operación NCOx después de que se ejecuta la escritura en el registro NCOxINCL.

Fuentes de reloj

Las fuentes de reloj disponibles para el NCOx incluyen:

- HFINTOSC
- F_{osc}
- LCxSALIDA
- pasador CLKIN

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



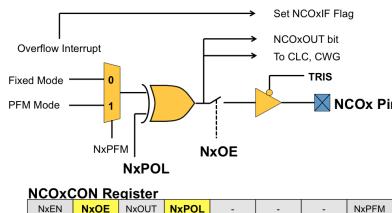
(/local--
files/8bit:nco/NCO_clock.png)

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

Producción

La salida del NCO tiene varias opciones que se pueden configurar en el registro NCOCON.

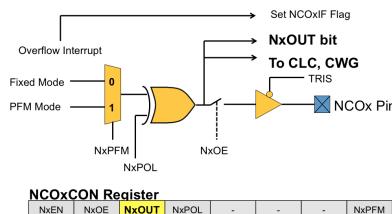
La salida se puede habilitar o deshabilitar (bit NxOE) y también invertir (bit NxPOL) configurando o borrando bits en el registro NCOCON.



(/local--

files/8bit:nco/NCO_outstate.png)

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



(/local--

files/8bit:nco/NCO_output.png)

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

Interrumpir

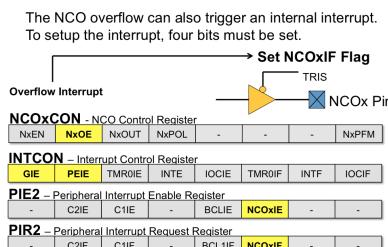
La salida NCO puede disparar una interrupción interna cuando el acumulador se desborda. Esto es manejado por tres bits (GIE, PEIE, NCOxIE) en el conjunto de registros que se muestra a continuación. Esto permitirá que el NCO controle las acciones del software a través de la rutina de servicio de interrupción en el código de la aplicación y al mismo tiempo emita 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 NCO. Puede haber múltiples módulos NCO. La "x" representa el número de NCO.

El bit NCOxIF es la bandera indicadora 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.



(/local--

files/8bit:nco/NCO_interrupt.png)

Interrupciones de rango medio mejoradas

1 Resumen

Las interrupciones son eventos detectados por la MCU que hacen que se anule el flujo normal del programa. Las interrupciones pausan el programa actual y transfieren el control a una rutina de firmware específica escrita por el usuario llamada Rutina de servicio de interrupción (ISR). El ISR procesa el evento de interrupción y luego reanuda el flujo normal del programa. Este artículo muestra cómo habilitar y procesar interrupciones en la familia PIC16F1xxx de PIC® de rango medio ^{mejorado}.

Descripción general del proceso de interrupción

1 Programa MCU para reaccionar a las interrupciones

La MCU debe programarse para permitir que ocurran interrupciones. La configuración de Global Interrupt Enable (GIE) y, en muchos casos, Peripheral Interrupt Enable (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 la MCU tiene un bit de habilitación individual. Se debe establecer el bit de activación de interrupción individual de un periférico, además de GIE/PEIE, antes de que el periférico pueda generar una interrupción. Los bits de habilitación de interrupción 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 GIE, PEIE y los bits de activación de interrupción individuales. Los indicadores de interrupción se encuentran en INTCON, PIR1, PIR2 y PIR3.

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

4 Ocurre 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 borrando GIE a 0.
- El contexto del programa actual 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 Ejecuciones de ISR

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

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

6 El control vuelve 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 sombra.
3. La dirección de retorno de la pila se carga en el contador de programa.
4. La ejecución se reanuda desde el punto en 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 | | | |

(/local--files/8bit:emr-interrupts/intcon-reg.png)

GIE - Habilitación de interrupción global **PEIE** - Habilitación de interrupción periférica **TMR0IE** - Habilitación de interrupción del temporizador 0 **INTE** - Habilitación de interrupción externa **IOCIE** - Habilitación de interrupción por cambio **TMR0IF** - Indicador de interrupción del temporizador 0 **INTF** - Indicador de interrupción externa **IOCIF** - Indicador de interrupción por cambio



INTCON contiene banderas de habilitación de interrupción global y periférica, así como banderas de solicitud de interrupción individuales y banderas de habilitación de interrupción para tres de las interrupciones PIC16F1xxxx.

Registros de habilitación de interrupción

registro PIE1

| | | | | | | | |
|---------|------|------|------|-------|--------|--------|---------|
| TMR1GIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMRI1IE |
| bit 7 | | | | bit 0 | | | |

(/local--files/8bit:emr-interrupts/pie1-reg.png)

TMR1GIE - Habilitación de interrupción de puerta del temporizador 1 **ADIE** - Habilitación de interrupción de convertidor analógico a digital (ADC) **RCIE** - Habilitación de interrupción de recepción de transmisor receptor asíncrono síncrono universal (USART) **TXIE** - Habilitación de interrupción de transmisión USART **SSPIE** - Habilitación de interrupción de puerto serie síncrono (MSSP) **CCP1IE** - Habilitación de interrupción CCP1 **TMR2IE** - Habilitación de interrupción del temporizador 2 **TMRI1IE** - Habilitación de interrupción





registro PIE2

| | | | | | | | |
|-------|------|------|------|-------|-------|-----|--------|
| OSFIE | C2IE | C1IE | EEIE | BCLIE | LCDIE | --- | CCP2IE |
| bit 7 | | | | bit 0 | | | |

(/local--files/8bit:emr-interrupts/pie2-reg.png)

OSFIE - Habilitación de interrupción por falla del oscilador **C2IE** - Habilitación de interrupción del comparador C2 **C1IE** - Habilitación de interrupción del comparador C1 **EEIE** - Habilitación de interrupción de finalización de escritura de EEPROM **BCLIE** - Habilitación de interrupción de colisión de bus MSSP **LCDIE** - Habilitación de interrupción del módulo LCD **---** - Sin implementar, se lee como 0 **CCP2IE** - CCP2 Habilitar interrupción



registro PIE3

| | | | | | | | |
|-------|--------|--------|--------|--------|-----|--------|-----|
| --- | CCP5IE | CCP4IE | CCP3IE | TMR6IE | --- | TMR4IE | --- |
| bit 7 | | | | bit 0 | | | |

(/local--files/8bit:emr-interrupts/pie3-reg.png)

--- - No implementado leído como 0 **CCP5IE** - CCP5 Interrupt Enable **CCP4IE** - CCP4 Interrupt Enable **CCP3IE** - CCP3 Interrupt Enable **TMR6IE** - Timer6 Interrupt Enable **---** - No implementado, leído como 0 **TMR4IE** - Timer4 Interrupt Enable **---** - No implementado, leído como 0



PIE1 , **PIE2** y **PIE3** contienen los indicadores de activación de interrupción individuales para los periféricos de la MCU.

Registros de solicitud de interrupción

registro PIR1

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

(/local--files/8bit:emr-interrupts/pir1-reg.png)

TMR1GIF - Indicador de interrupción de puerta del temporizador 1 **ADIF** - Indicador de interrupción de ADC **RCIF** - Indicador de interrupción de recepción de USART **TXIF** - Indicador de interrupción de transmisión de USART **SSPIF** - Indicador de interrupción de MSSP **CCP1IF** - Indicador de interrupción de CCP1 **TMR2IF** - Indicador de interrupción de temporizador 2 **TMRI1F** - Indicador de interrupción



registro PIR2

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

(/local--files/8bit:emr-interrupts/pir2-reg.png)

OSFIF - Indicador de interrupción por falla del oscilador **C2IF** - Indicador de interrupción C2 del comparador **C1IF** - Indicador de interrupción C1 del comparador **EEIF** - Indicador de interrupción de finalización de escritura de EEPROM **BCLIF** - Indicador de interrupción de colisión del bus MSSP **LCDIF** - Indicador de interrupción del módulo LCD **---** - Sin implementar, se lee como 0 **CCP2IF** - CCP2 Indicador de interrupción



registro PIR3



(/local--files/8bit:emr-interrupts/pir3-reg.png)

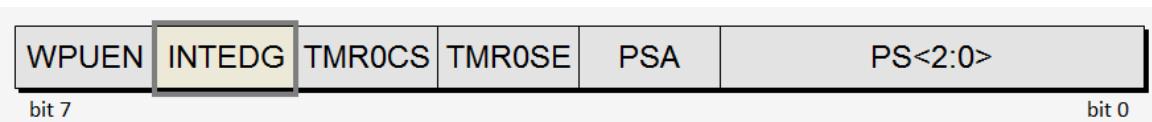
--- - Sin implementar, leer como 0 CCP5IF - Bandera de interrupción de CCP5 CCP4IF - Bandera de interrupción de CCP4 CCP3IF - Bandera de interrupción de CCP3 TMR6IF - Bandera de interrupción de Timer6 --- - No implementada, leer como 0 TMR4IF - Bandera de interrupción de Timer4 --- - No implementada, leer como 0



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

OPCIÓN_REG

OPCIÓN_REG



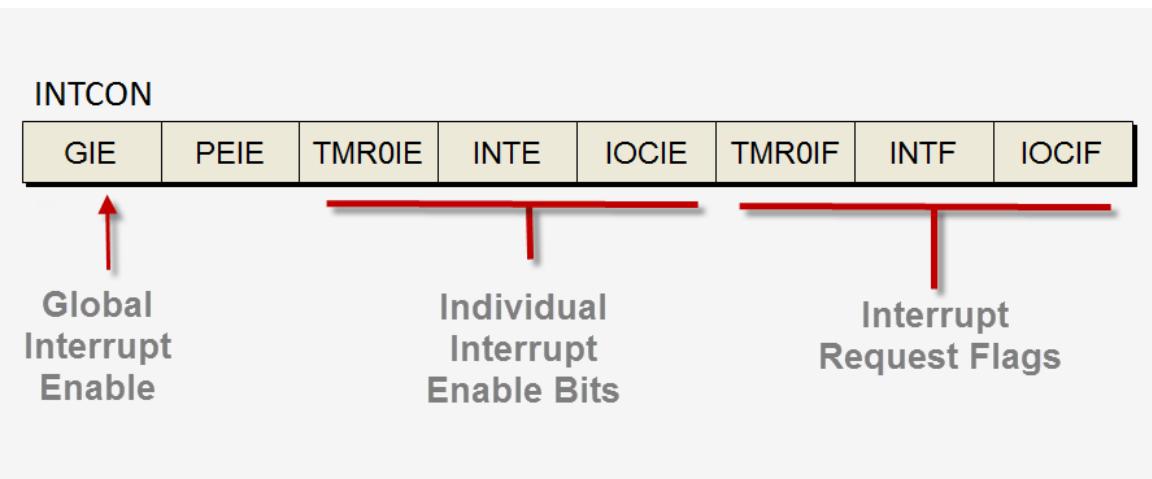
(/local--files/8bit:emr-interrupts/option-reg.png)

El indicador INTEDG en OPTION_REG se usa para establecer un flanco ascendente o descendente en el pin INT como desencadenante de una interrupción INTE.

Habilitación de interrupciones

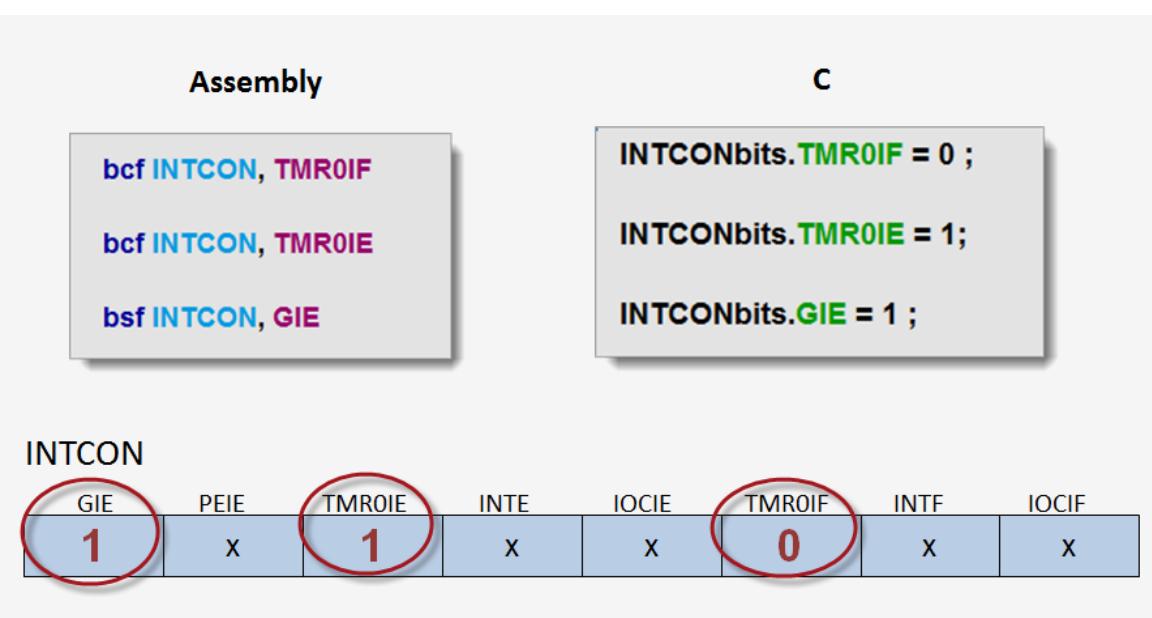
Interrupciones del núcleo

Tres fuentes de interrupción (Timer0, External Interrupt e Interrupt on Change) tienen bits de habilitación de interrupción ubicados en INTCON. Estas interrupciones se denominan interrupciones centrales.



(/local--files/8bit:emr-interrupts/core-enable-reg.png)

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



(/local--files/8bit:emr-interrupts/core-enable-code.png)



Borrar un indicador de solicitud de interrupción antes de establecer el indicador de activación de interrupción evita que cualquier solicitud de interrupción pendiente active una interrupción inmediata.

Interrupciones Periféricas

Los periféricos PIC16F1xxx, capaces de generar solicitudes de interrupción cada uno, tienen sus indicadores de activación de interrupción 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, SSP1IF banksel PIE1 bsf PIE1, SSP1IE 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 |

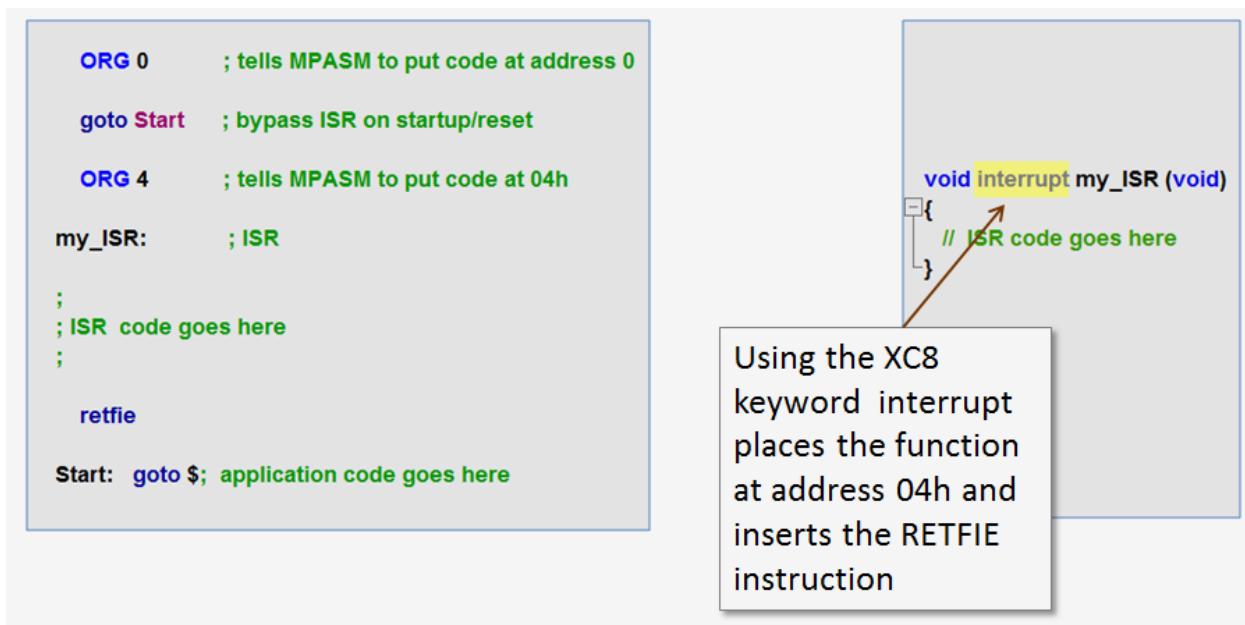
(/local--files/8bit:emr-interrupts/peie-code.png)

Dar servicio a una interrupción

ISR

El ISR es un programa escrito por el usuario que realiza las tareas necesarias cuando ocurre 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 lenguaje C o ensamblador.

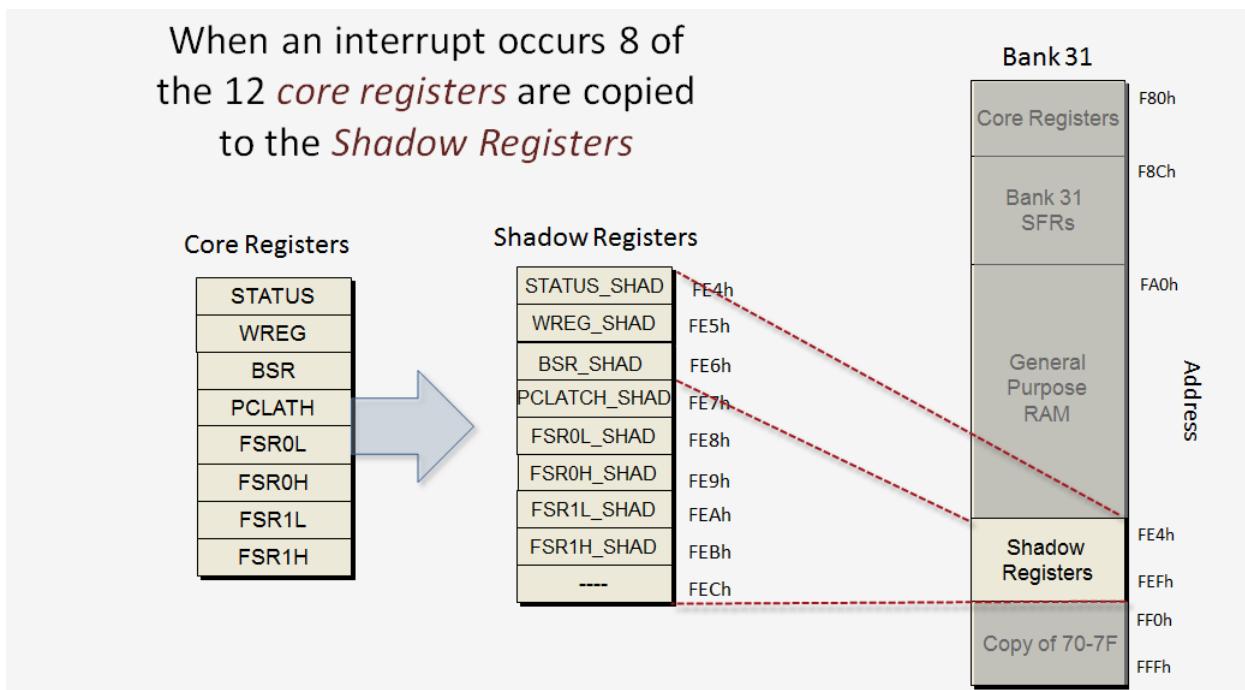
ISR en ensamblaje (izquierda) e ISR en C (derecha)



(/local--files/8bit:emr-interrupts/isr-location.png)

Guardar contexto

El hardware de la MCU invocará el mecanismo de ahorro de contexto cuando ocurra una interrupción.



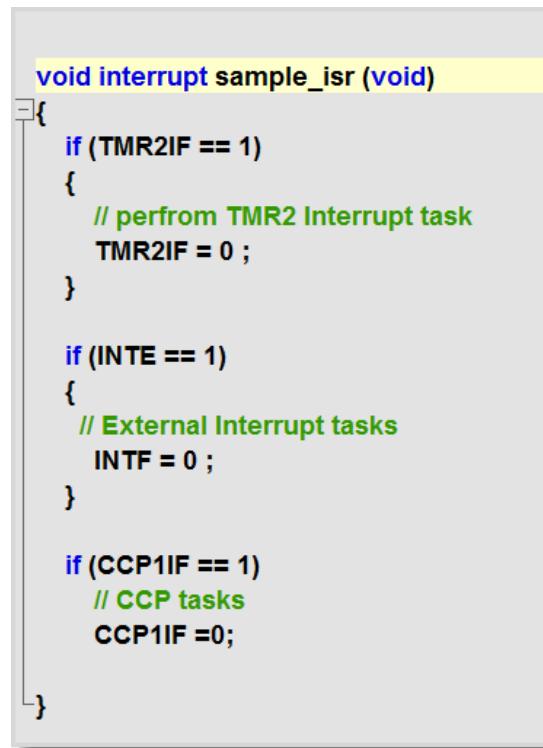
(/local--files/8bit:emr-interrupts/context-saving.png)

Verificar fuente de la interrupción

Hay un ISR que da servicio a todas las interrupciones de la aplicación. El ISR necesita verificar secuencialmente los indicadores de solicitud de interrupción individuales para determinar la fuente de la interrupción.

Borrado del 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 restablece 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 (INTF == 1)
    {
        // External Interrupt tasks
        INTF = 0 ;
    }

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

```

(/local--files/8bit:emr-interrupts/sample-isr.png)

Ejemplo de código de interrupción

Procesando una interrupción del Timer2

La siguiente animación muestra el código para configurar y procesar una interrupción del Timer2 en el MCU de rango medio mejorado PIC16F1xxx. Los íconos de control en la parte inferior de la animación permiten pausar la visualización y realizar un solo paso.

 [Imagen de presentación de diapositivas](#)

◀ ▶ ▶ 1 / 0

Se puede encontrar una explicación más detallada de las interrupciones PIC16F1xxx en la página Interrupciones (/mcu1102:interrupts) .

La página Timer2/4/6 (/8bit:timer2) proporciona los detalles sobre la configuración y el funcionamiento de Timer2.

Ejemplo de procesamiento de una interrupción en una MCU PIC16F1xxx

Procesando una interrupción del Timer2

La siguiente animación muestra el código para configurar y procesar una interrupción del Timer2 en el MCU de rango medio mejorado PIC16F1xxx. Los íconos de control en la parte inferior de la animación permiten pausar la visualización y realizar un solo paso.

 [Imagen de presentación de diapositivas](#)

◀ ▶ ▶ 1 / 0

Se puede encontrar una explicación más detallada de las interrupciones PIC16F1xxx en la página Interrupciones (/mcu1102:interrupts) .

La página Timer2/4/6 (/8bit:timer2) proporciona los detalles sobre la configuración y el funcionamiento de Timer2.