TECNICATURA SUPERIOR EN TELECOMUNICACIONES

ELECTRÓNICA MICROCONTROLADA

Docentes: Ing. Jorge E. Morales, Téc. Gonzalo Vera.

**Título:** Referencia de la gama media mejorada PIC.

**Grupo 1:**

❖ Birge, Adolfo Federico.

❖ Carunchio, Carlos Javier.

❖ Ferreyra, María Luciana.

❖ Gutiérrez, Emma Vilma.

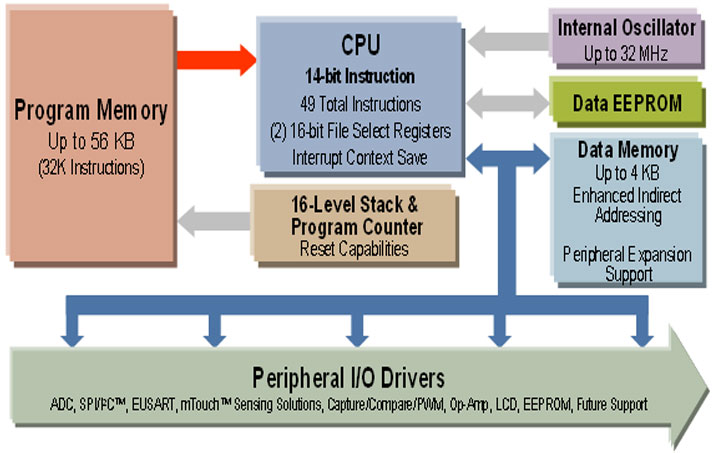
❖ Merlo, Emmanuel.

❖ Romero, Gisela de Lourdes.

Referencia de la gama media mejorada PIC

[**MCU PIC de 8 bits de® rango medio mejorado**](http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=1025&mid=10&lang=en&pageId=74)**PIC12F1xxx, PIC16F1xxx**

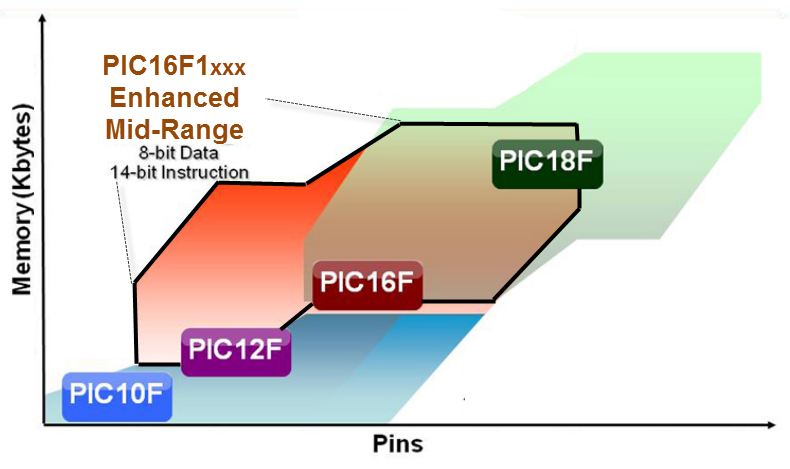
* 49 (14 bits de ancho) instrucciones fáciles de aprender
* Memoria de programa direccionable de 32K word (56 KB)
* 4 KB de RAM (máx.)
* Pila de hardware de 16 niveles
* 2 registros de selección de archivos (16 bits)
* Manejo de interrupciones de hardware con guardado de contenido
* Conjunto de funciones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

[](https://microchipdeveloper.com/local--files/8bit:emr/EnhancedMidRange.jpg)

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

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

**Familias de MCU de 8 bits de Microchip**

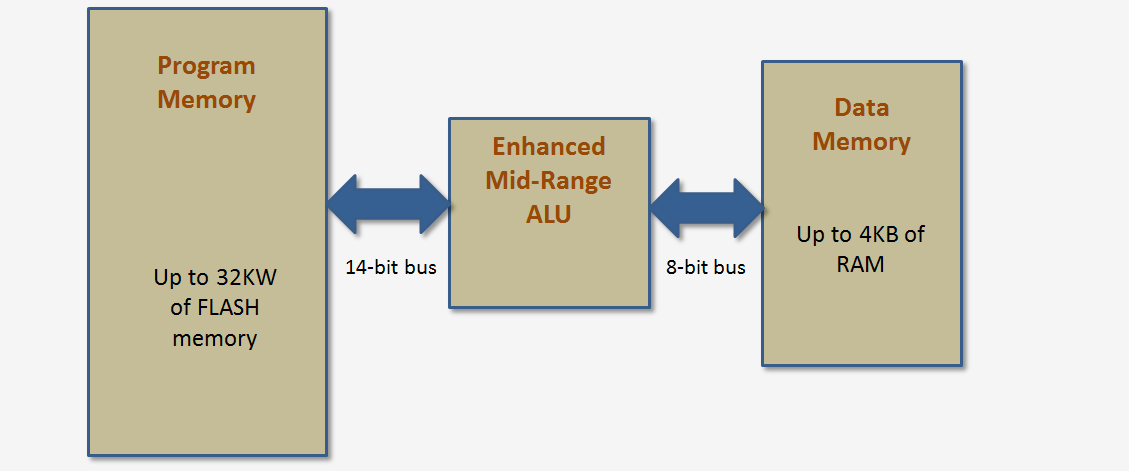


La familia PICMCU de 8 bits de rango medio mejorado PIC16F1xxx abarca una amplia gama de tamaños de memoria y pines de E/S.

Esta página presenta las características arquitectónicas clave de la familia pic16F1xxx de MCU. En esta página se proporcionan enlaces a los detalles técnicos necesarios para implementar aplicaciones en la familia de MCU PIC de rango medio mejorada.®

**Arquitectura de Harvard**

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



**Bus de instrucción**

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

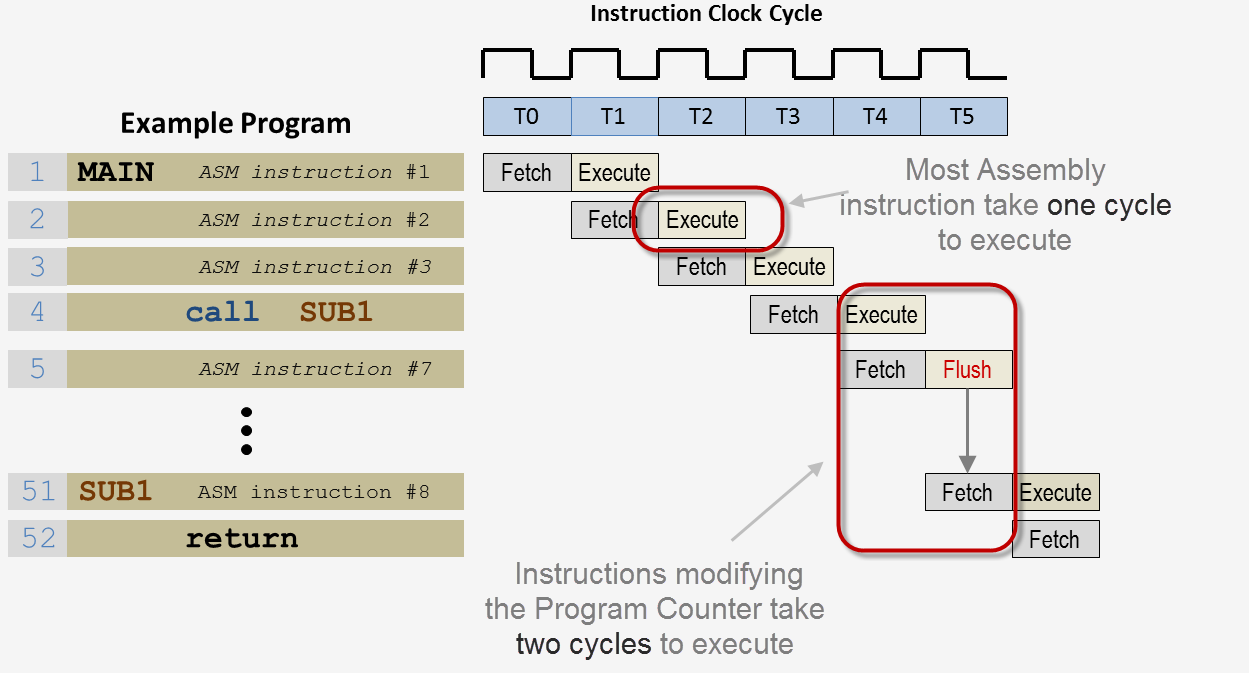
**Bus de datos**

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

**Instrucción Pipelining**

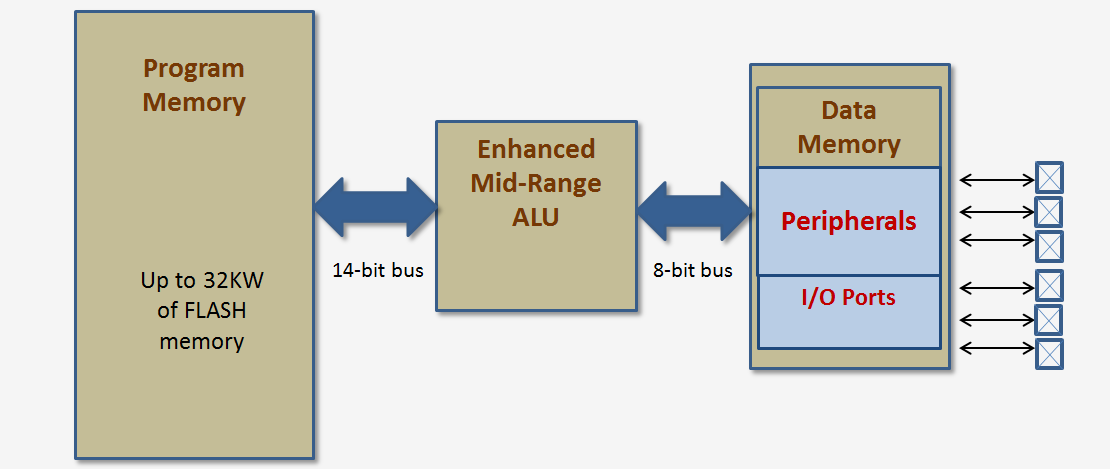
La arquitectura de doble bus mejorada del PIC de rango medio proporciona una canalización de instrucciones de dos etapas. Uno cada ciclo de reloj se ejecutan dos fases de instrucción:

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



**Periféricos asignados a memoria**

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



​

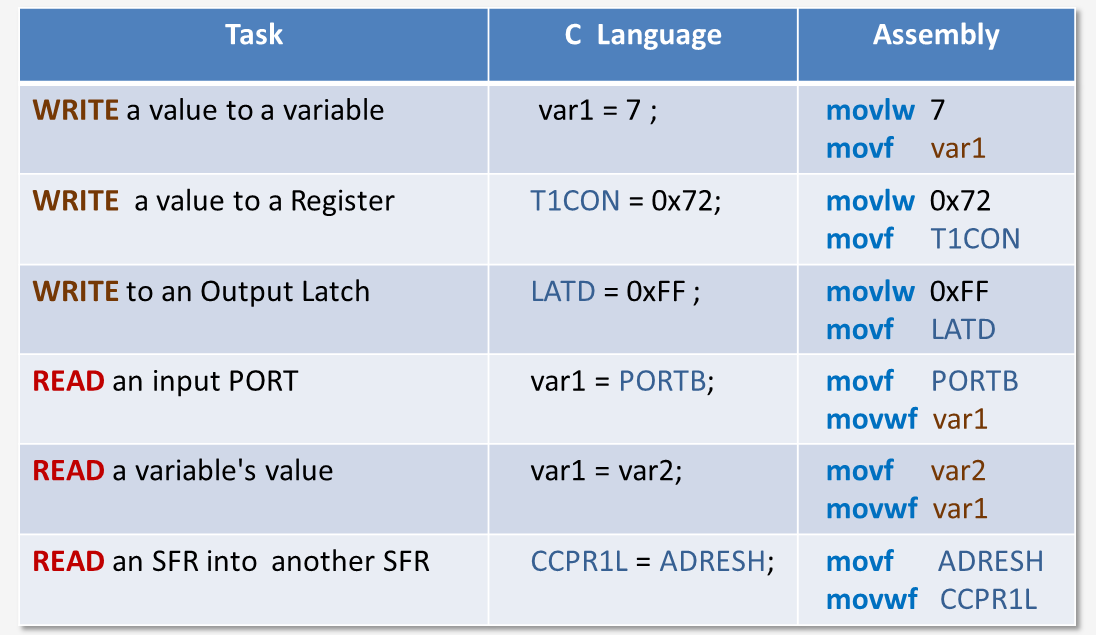
La [**página memoria de datos**](https://microchipdeveloper.com/mcu1102:data-memory) del tutorial de rango medio mejorado ofrece una descripción completa junto con ejemplos de programas para acceder a los periféricos asignados a la memoria

**Conjunto de instrucciones ortogonales**

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

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

**Ejemplos de implementación de instrucciones**



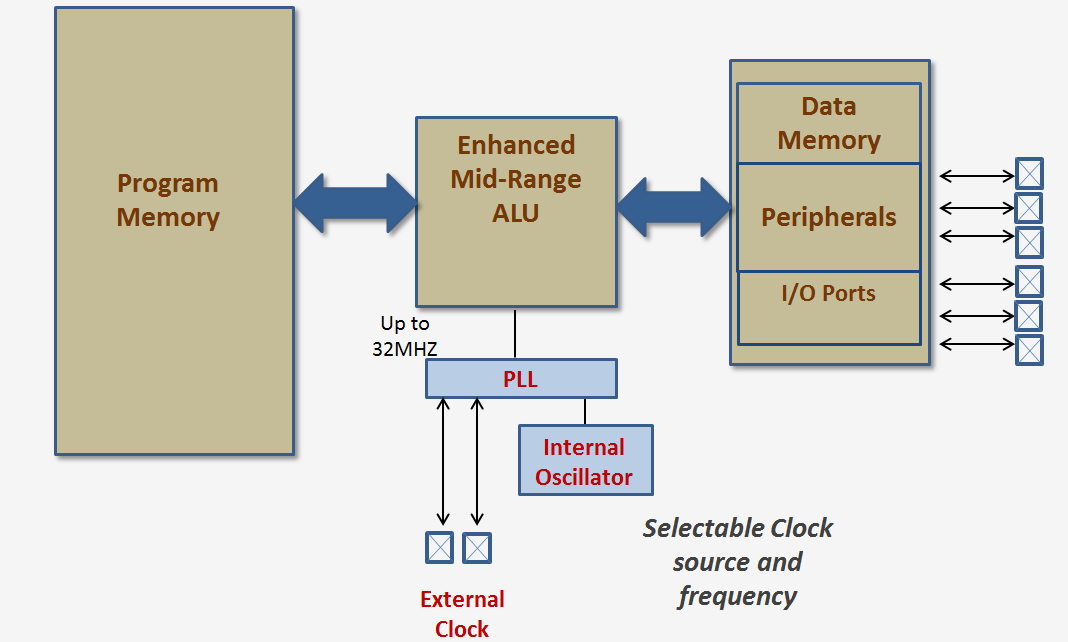
​

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

**Opciones de sincronización flexibles (hasta 32 MHz)**

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

* Fuente opcional (**oscilador interno** o **circuitos externos)**)
* Opciones de velocidad **flexibles de hasta 32 MHZ**
* **Arranque de dos velocidades:** permite que el sistema ejecute software de inicialización mientras el oscilador externo se estabiliza
* **Conmutación de reloj: la** fuente de reloj del sistema se puede cambiar entre fuentes de reloj externas e internas a través del software.
* **Monitor de reloj a prueba de** fallos: cambia al oscilador interno en caso de una falla del reloj externo



​

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

**E/S digitales**

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

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

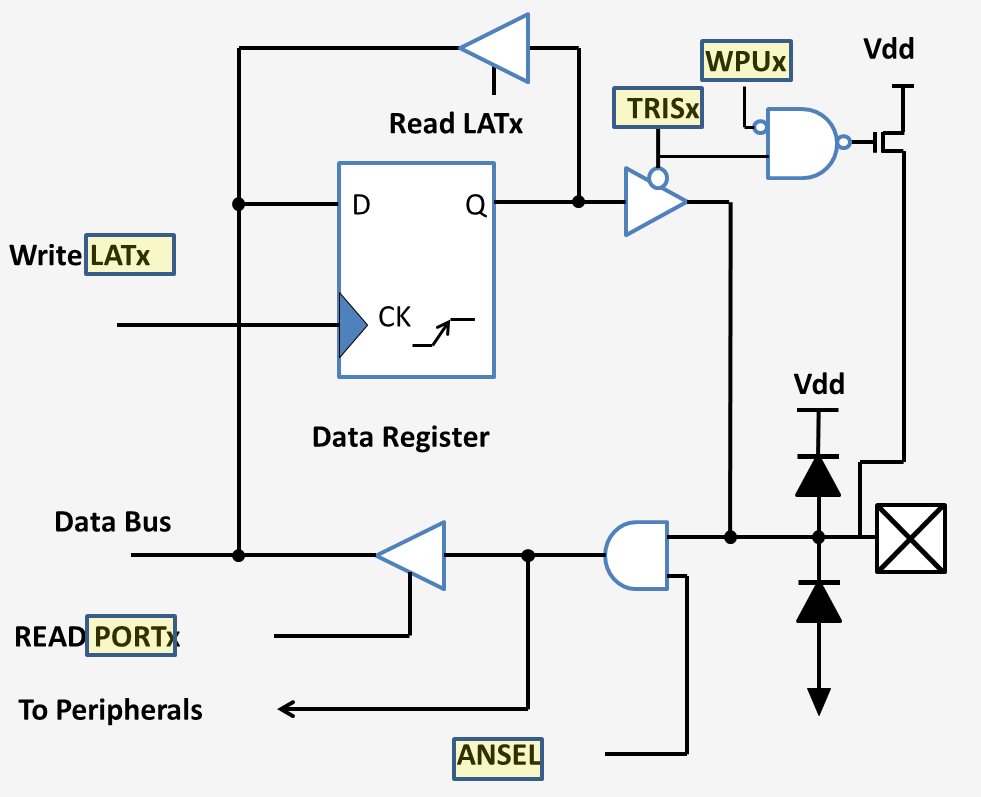
En Reset:

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

**Estructura típica de pines digitales**

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

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

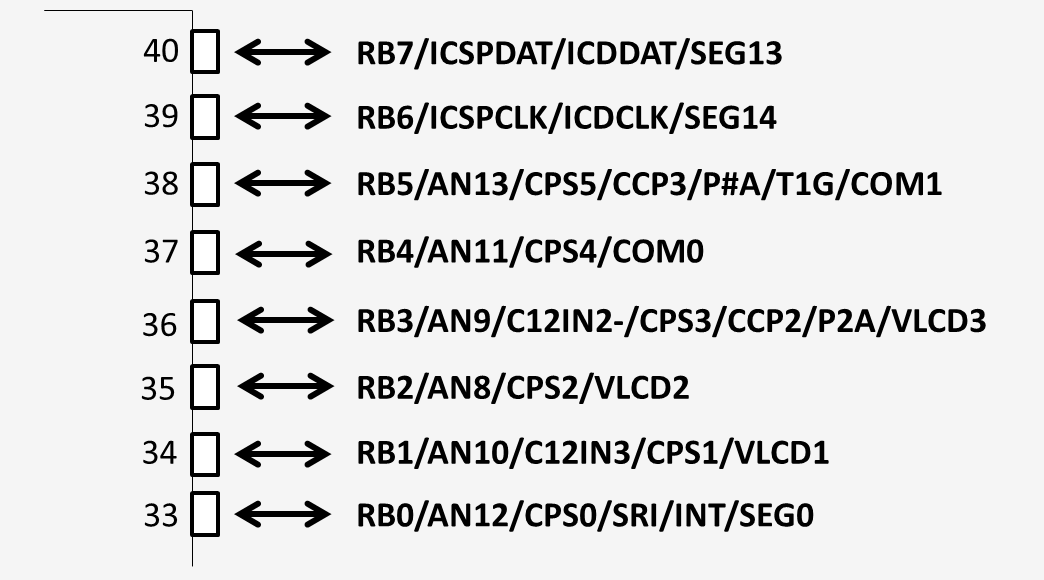


​

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

**Pines multiplexados**

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

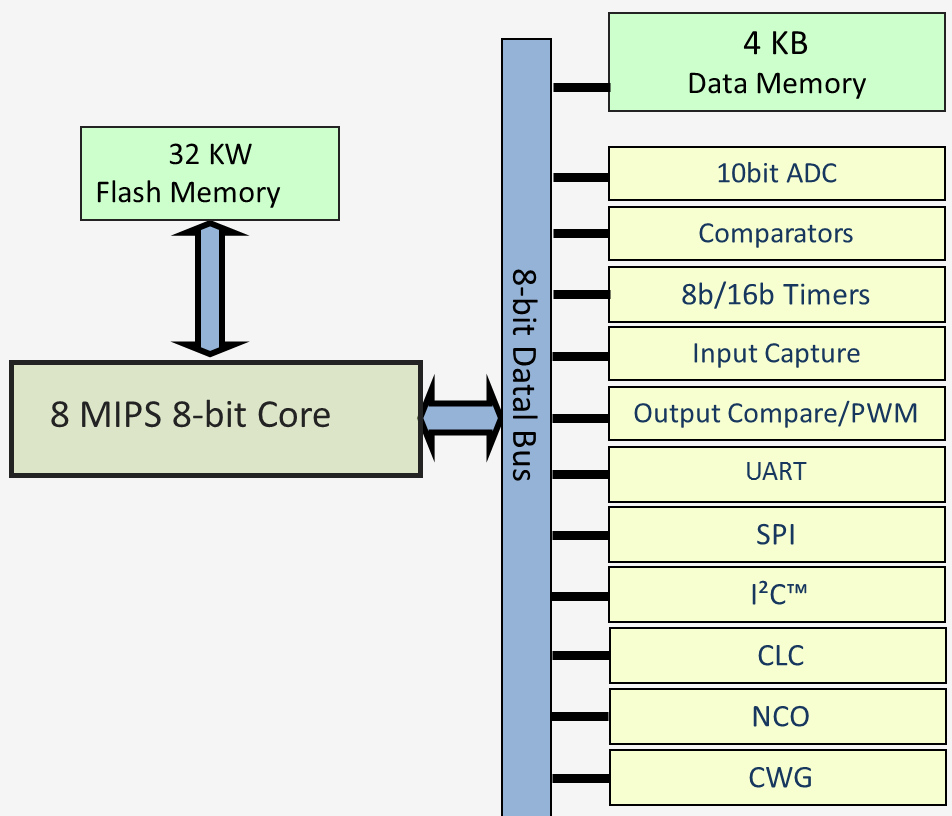


​

Para obtener una descripción completa de la E/S digital PIC de rango medio mejorada, incluidos los detalles sobre la configuración de los pines, consulte la [**sección Periférico**](https://microchipdeveloper.com/mcu1102:peripherals) del tutorial de rango medio mejorado

**Periféricos avanzados**

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



​

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

**Interrumpe**

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

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

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

**Rutina de servicio de interrupción (ISR)**

El usuario es responsable de escribir el código para dar servicio a la interrupción y colocar el código en la dirección 0x04. Esta rutina de servicio de interrupción (ISR) determina el origen de la interrupción y, a continuación, realiza la tarea necesaria para dar servicio al periférico de interrupción. La instrucción final de un ISR es la instrucción Return From Interrupt (RETFIE).

**Guardado automático de contexto**

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

* Registro W
* BSR
* ESTADO
* Fsr
* PCLATH

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

**Preferencia de interrupción de un solo nivel**

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

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

​

Para obtener una descripción completa del proceso de interrupción y ejemplos de programación, consulte la [**sección Interrupción**](https://microchipdeveloper.com/mcu1102:interrupts) del tutorial de rango medio mejorado.

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

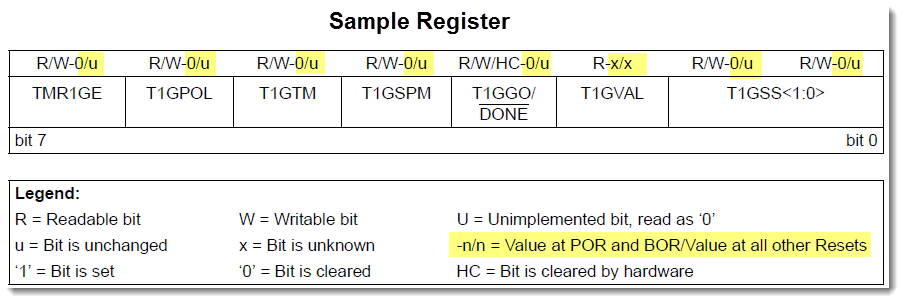
Hay varias fuentes de un RESET en el MCU PIC de rango medio mejorado. Las fuentes RESET comunes a casi todas las aplicaciones son el Power On Reset (POR) y el Brown Out Reset (BOR) debido a una caída del voltaje de la fuente de alimentación (es decir, brown-out). Hay varios otros métodos para restablecer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

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

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

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

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



**Conjunto de instrucciones de rango medio enhnanced**

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Operaciones orientadas a bytes** | | | | | | |
| **Mnemotécnico, Operandos** | | **Descripción** | **Ciclos** | **Opcode MSb de 14 bits...... Osa** | **Estado afectado** | **Notas** |
| ADDWF | f,d | Agregar W y f | 1 | 00 0111 dfff ffff | C, DC, Z | 2 |
| ADDWFC | f,d | Añadir con Carry W y f | 1 | 11 1101 dfff ffff | C, DC, Z | 2 |
| ANDWF | f,d | Y W con f | 1 | 00 0101 dfff ffff | Z | 2 |
| ASRF | f,d | Desplazamiento aritmético a la derecha | 1 | 11 0111 dfff ffff | C, Z | 2 |
| LSLF | f,d | Desplazamiento lógico a la izquierda | 1 | 11 0101 dfff ffff | C, Z | 2 |
| LSRF | f,d | Desplazamiento lógico a la derecha | 1 | 11 0110 dfff ffff | C, Z | 2 |
| CLRF | f | Borrar f | 1 | 00 0001 1fff ffff | Z | 2 |
| CLRW |  | Borrar W | 1 | 00 0001 0000 00xx | Z |  |
| COMF | f,d | Complemento f | 1 | 00 1001 dfff ffff | Z | 2 |
| DECF | f,d | Decremento f | 1 | 00 0011 dfff ffff | Z | 2 |
| INCF | f,d | Incremento f | 1 | 00 1010 dfff ffff | Z | 2 |
| IORWF | f,d | Inclusivo O W con f | 1 | 00 0100 dfff ffff | Z | 2 |
| MOVF | f,d | Mover f | 1 | 00 1000 dfff ffff | Z | 2 |
| MOVWF | f | Mover W a f | 1 | 00 0000 1fff ffff | Ninguno | 2 |
| RLF | f,d | Gire a la izquierda f a través de Carry | 1 | 00 1101 dfff ffff | C | 2 |
| FRR | f,d | Gire a la derecha f a través de Carry | 1 | 00 1100 dfff ffff | C | 2 |
| SUBWF | f,d | Restar con Borrow W de f | 1 | 11 1011 dfff ffff | C, DC, Z | 2 |
| SUBWFB | f,d | Restar W de f | 1 | 00 0010 dfff ffff | C, DC, Z | 2 |
| SWAPF | f,d | Intercambiar mordiscos en f | 1 | 00 1110 dfff ffff | Ninguno |  |
| XORWF | f,d | Exclusivo OR W con f | 1 | 00 0110 dfff ffff | Z | 2 |
| **Instrucciones de salto orientadas a bytes** | | | | | | |
| **Mnemotécnico, Operandos** | | **Descripción** | **Ciclos** | **Opcode MSb de 14 bits...... Osa** | **Estado afectado** | **Notas** |
| DECFSZ | f,d | Decremento f, Omitir si 0 | 1(2) | 00 1011 dfff ffff | Ninguno | 1,2 |
| INCFSZ | f,d | Incremento f, Omitir si 0 | 1(2) | 00 1111 dfff ffff | Ninguno | 1,2 |
| **Operaciones de registro de archivos orientadas a bits** | | | | | | |
| **Mnemotécnico, Operandos** | | **Descripción** | **Ciclos** | **Opcode MSb de 14 bits...... Osa** | **Estado afectado** | **Notas** |
| Bcf | f,b | Bit Clear f | 1 | 01 00bb bff ffff | Ninguno | 2 |
| BSF | f,b | 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 | Mover literal a BSR | 1 | 00 0000 001k kkkk | Ninguno |  |
| MOVLP | k | Mover literal a PCLATH | 1 | 11 0001 1kkk kkkk | Ninguno |  |
| MOVLW | k | Mover literal a W | 1 | 11 0000 kkkk kkkk | Ninguno |  |
| SUBLW | k | Restar W del literal | 1 | 11 1100 kkkk kkkk | C, DC, Z |  |
| XORLW | k | Exclusivo O literal con W | 1 | 11 1010 kkkk kkkk | Z |  |
| **Operaciones de control** | | | | | | |
| **Mnemotécnico, Operandos** | | **Descripción** | **Ciclos** | **Opcode MSb de 14 bits...... Osa** | **Estado afectado** | **Notas** |
| Sujetador | k | Rama relativa | 2 | 11 001k kkkk kkkk | Ninguno |  |
| BRW |  | Rama relativa con W | 2 | 00 0000 0000 1011 | Ninguno |  |
| LLAMAR | k | Llamar subrutina | 2 | 10 0kkk kkkk kkkk | Ninguno |  |
| CALLW |  | Llamar a la subrutina con W | 2 | 00 0000 0000 1010 | Ninguno |  |
| GOTO | k | Dirección de Goto | 2 | 10 1kkk kkkk kkkk | Ninguno |  |
| RETFIE | k | Retorno de la interrupción | 2 | 00 0000 0000 1001 | Ninguno |  |
| RETLW | k | Volver, colocar literal en W | 2 | 11 0100 kkkk kkkk | Ninguno |  |
| DEVOLUCIÓN | k | Volver de la subrutina | 2 | 00 0000 0000 1000 | Ninguno |  |
| **Operaciones inherentes** | | | | | | |
| **Mnemotécnico, Operandos** | | **Descripción** | **Ciclos** | **Opcode MSb de 14 bits...... Osa** | **Estado afectado** | **Notas** |
| CLRWDT |  | Borrar temporizador de vigilancia | 1 | 00 0000 0110 0100 | TO,PD |  |
| Nop |  | Sin operación | 1 | 00 0000 0000 0000 | Ninguno |  |
| OPCIÓN |  | Cargar OPTION registrarse con W | 1 | 00 0000 0110 0010 | Ninguno |  |
| RESTABLECIMIENTO |  | Restablecimiento del dispositivo de software | 1 | 00 0000 0000 0001 | Ninguno |  |
| DORMIR |  | Entrar en modo de espera | 1 | 00 0000 0110 0011 | PARA, PD |  |
| TRIS | f | Cargar registro TRIS | 1 | 00 0000 0110 0fff | Ninguno |  |
| **C-Compiler optimizado** | | | | | | |
| **Mnemotécnico, Operandos** | | **Descripción** | **Ciclos** | **Opcode MSb de 14 bits...... Osa** | **Estado afectado** | **Notas** |
| ADDFSR |  | Agregar literal a FSRn | 1 | 11 0001 0nkk kkkk | Ninguno |  |
| MOVIW |  | Mover FSRn indirecto a W | 1 | 00 0000 0001 0nnn | Z | 2 |
| MOVWI |  | Mover W a FSRn indirecto | 1 | 00 0000 0001 1nnnn | Z | 2 |

**Notas**

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

Bits de configuración

Resumen de bits de configuración en C

Memoria de datos (desarrolar carpeta)

Memoria del programa (desarrolar carpeta)

Periféricos (desarrolar carpeta)

Interrumpe(desarrolar carpeta)