 **PIC18F2XXX/4XXX FAMILY**

Especificación de programación del microcontrolador Flash

1. DESCRIPCIÓN GENERAL DEL DISPOSITIVO

Este documento incluye las especificaciones de programación para los siguientes dispositivos:



**2.0 RESUMEN DE PROGRAMACIÓN**

Los dispositivos de la familia PIC18F2XXX / 4XXX se pueden programar utilizando la programación en serie de alto voltaje en circuito ™ (ICSP ™) o el método ICSP de bajo voltaje. Ambos métodos se pueden hacer con el dispositivo en el sistema del usuario. El método ICSP de bajo voltaje es ligeramente diferente del método de alto voltaje y estas diferencias se observan donde es aplicable.

Esta especificación de programación se aplica a los dispositivos de la familia PIC18F2XXX / 4XXX en todos los tipos de paquetes.

**2.1 Requisitos de hardware**

En el modo ICSP de alto voltaje, los dispositivos de la familia PIC18F2XXX / 4XXX requieren dos fuentes de alimentación programables: una para VDD y otra para MCLR / VPP / RE3. Ambos suministros deben tener una resolución mínima de 0.25V. Consulte la Sección 6.0 “Requisitos de sincronización de las características de CA / CC para el modo de prueba de programa / verificación” para obtener información sobre parámetros de hardware adicionales.

2.1.1 LOW-VOLTAGE ICSP PROGRAMMING

En el modo ICSP de bajo voltaje, los dispositivos de la familia PIC18F2XXX / 4XXX se pueden programar utilizando una fuente VDD en el rango operativo. El MCLR / VPP / RE3 no tiene que llevarse a un voltaje diferente, sino que puede dejarse en el voltaje de funcionamiento normal. Consulte la Sección 6.0 “Requisitos de sincronización de las características de CA / CC para el modo de prueba de programa / verificación” para obtener información sobre parámetros de hardware adicionales.

**2.2 Diagramas de pines**

Los diagramas de pines para la familia PIC18F2XXX / 4XXX se muestran en la Figura 2-1, Figura 2-2, Figura 2-3, Figura 2-4, Figura 2-5.

**Tabla 2-1: DESCRIPCIONES DE LOS PINES (DURANTE LA PROGRAMACIÓN): FAMILIA PIC18F2XXX / 4XXX**

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre del pin | Durante la programación | | |
| Nombre | Tipo | Descripción |
| MCLR/VPP/RE3 | VPP | P | Habilitación de programación |
| VDD**(2)** | VDD | P | Fuente de alimentación |
| VSS(2) | VSS | P | Tierra |
| RB5 | PGM | I | Entrada ICSP ™ de bajo voltaje cuando el bit de configuración LVP es igual a '1' (1) |
| RB6 | PGC | I | Relog |
| RB7 | PGD | I/O | Datos |

**Leyenda**: I = Entrada, O = Salida, P = Potencia

Nota 1: Consulte la Figura 5-1 para obtener más información.

2: Todos los pines de la fuente de alimentación (VDD) y de tierra (VSS) deben estar conectados

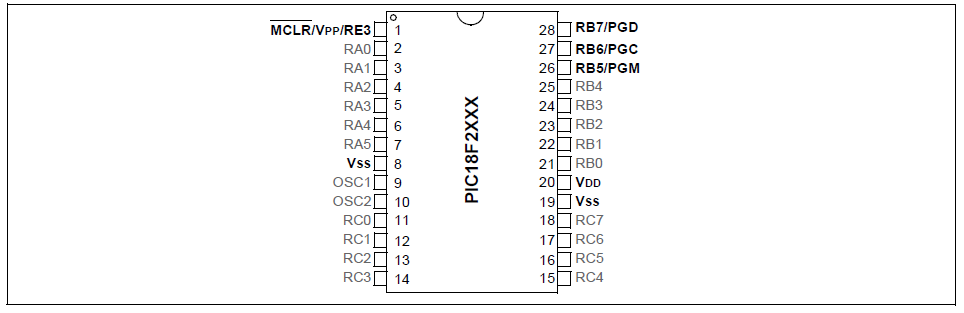
Los siguientes dispositivos están incluidos en encapsulados SPDIP, PDIP y SOIC de 28 pines:



Los siguientes dispositivos están incluidos en encapsulados SSOP de 28 pines



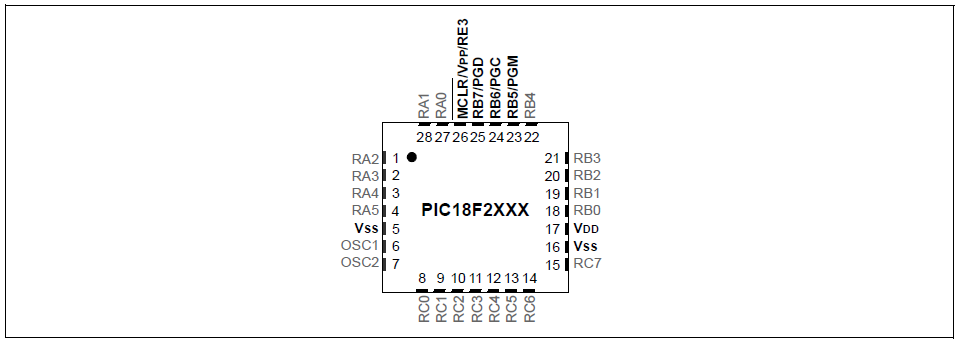
**Figura 2-1: 28-Pin SPDIP, PDIP, SOIC,SSOP**



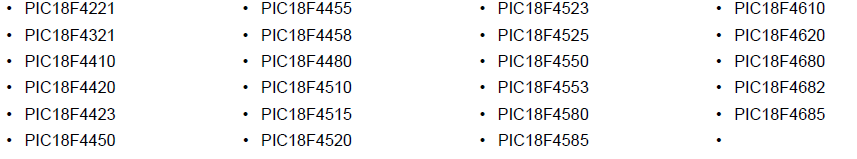
Los siguientes dispositivos están incluidos en encapsulado QFN de 28 pines



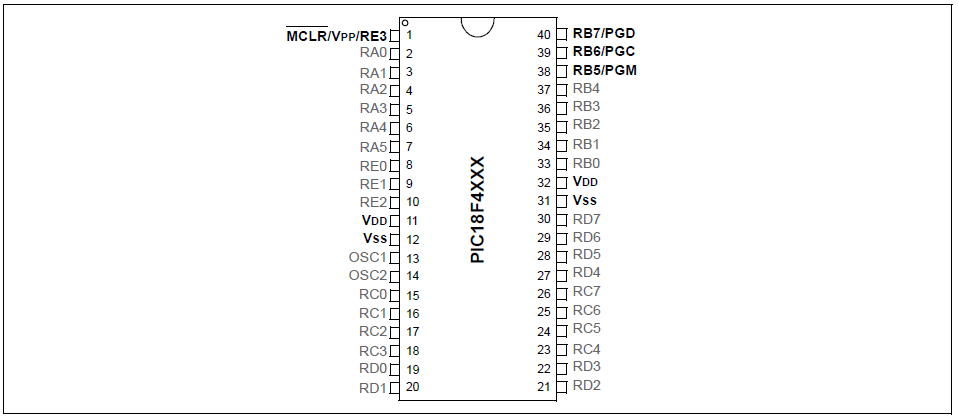
**Figura 2-2: 28-Pin QFN**



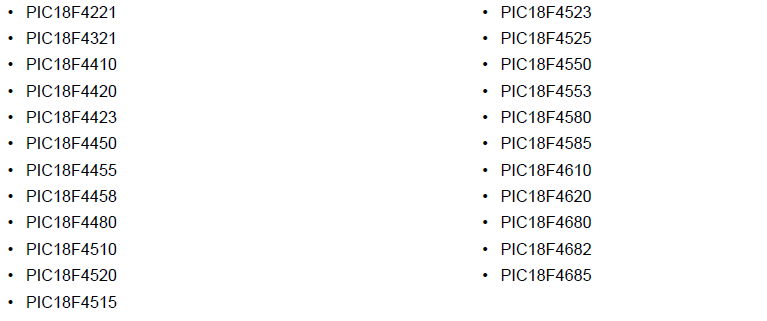
Los siguientes dispositivos están incluidos en encapsulado PDIP de 40 pines:



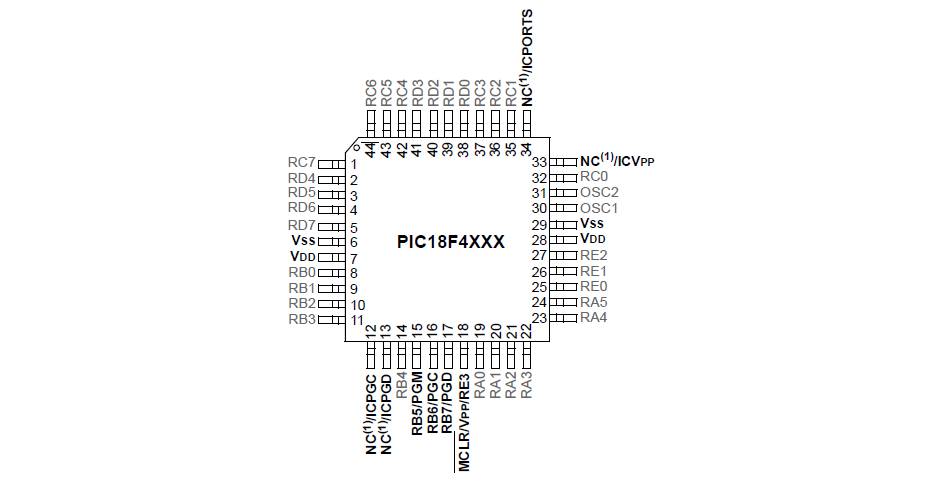
**Figura 2-3: 40-Pin PDIP**



Los siguientes dispositivos están incluidos en encapsulado TQFP de 44 pines:



**Figura 2-4: 44-PIN TQFP**

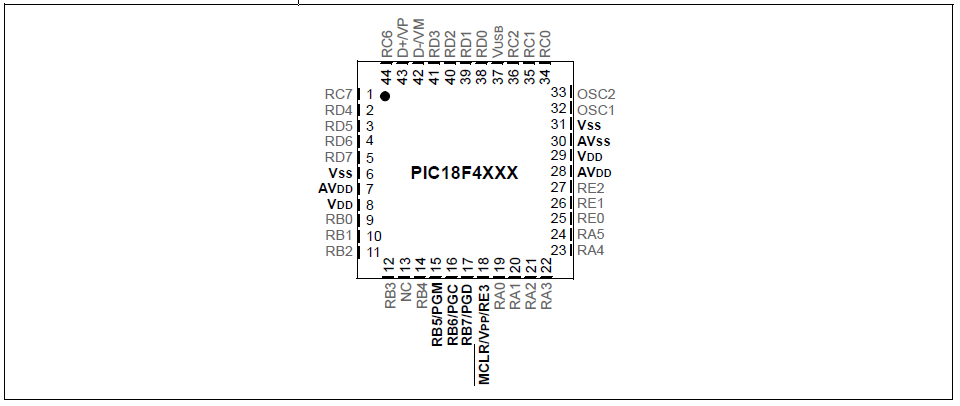


Nota 1: estos pines son NC (sin conexión) para todos los dispositivos enumerados anteriormente, con la excepción de los dispositivos PIC18F4450, PIC18F4455, PIC18F4458 y PIC18F4553 (consulte la Sección 2.8 "Puerto ICSP / ICD dedicado (solo TQFP de 44 pines)" para obtener más información sobre la programación de estos pines en estos dispositivos).

Los siguientes dispositivos están incluidos en encapsulado QFN de 44 pines:



**Figura 2-5: 44-PIN QFN**

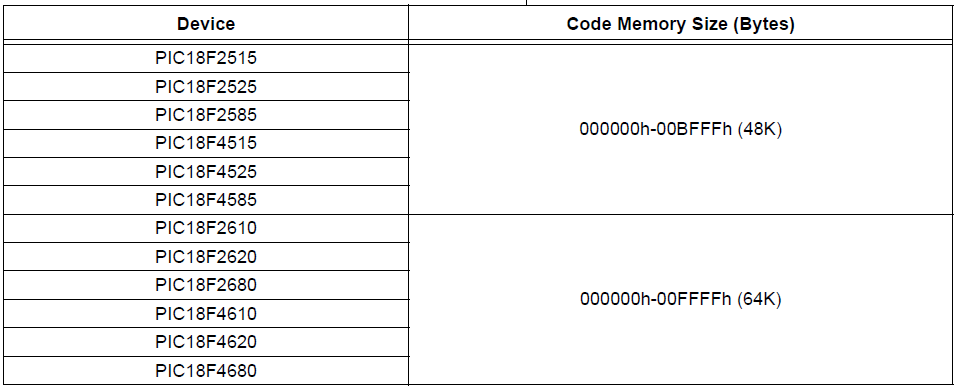


**2.3 Mapas de memoria**

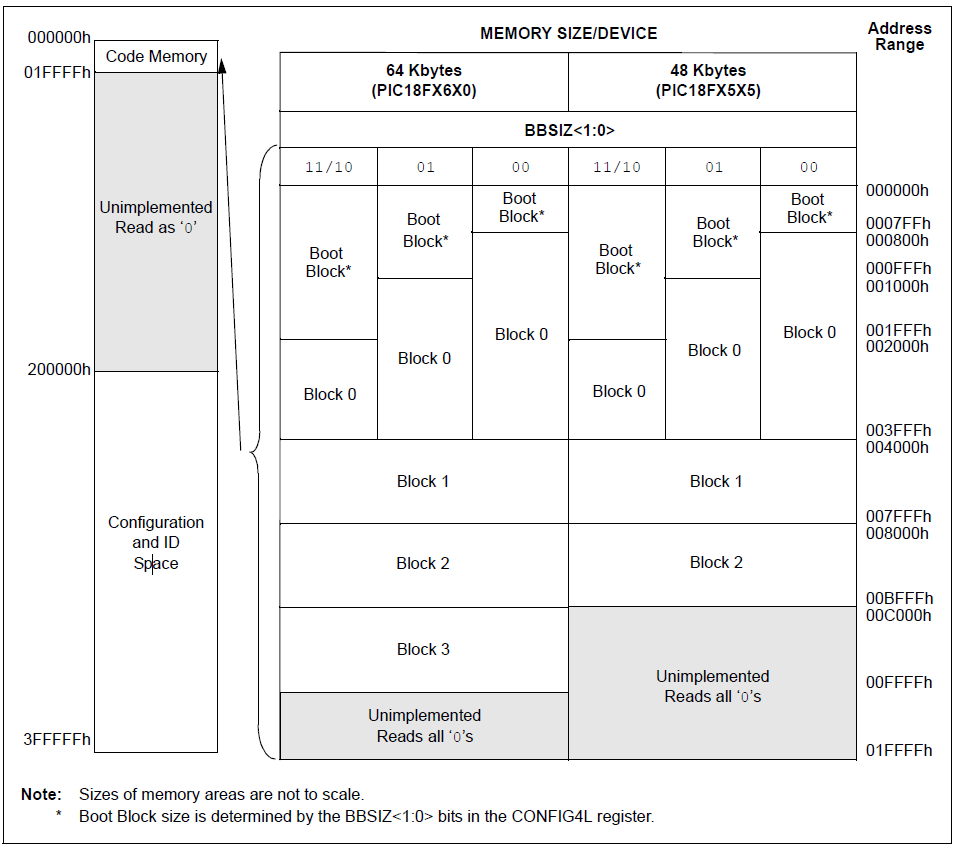
Para los dispositivos PIC18FX6X0, el espacio de memoria de código se extiende desde 0000h a 0FFFFh (64 Kbytes) en cuatro bloques de 16 Kbytes. Para los dispositivos PIC18FX5X5, el espacio de memoria de código se extiende desde 0000h a 0BFFFFh (48 Kbytes) en tres bloques de 16 Kbytes. Las direcciones, 0000h a 07FFh, sin embargo, definen una región de "Bloque de arranque" que se trata por separado del Bloque 0. Todos estos bloques definen límites de protección de código dentro del espacio de memoria de código.

El tamaño del Bloque de arranque en los dispositivos PIC18F2585 / 2680/4585/4680 se puede configurar como 1, 2 o 4K palabras (consulte la Figura 2-6). Esto se realiza a través de los bits BBSIZ <1: 0> en el registro de configuración, CONFIG4L. Es importante tener en cuenta que al aumentar el tamaño del Bloque de arranque disminuye el tamaño del Bloque 0.

**Tabla 2-2: IMPLEMENTACIÓN DE LA MEMORIA DE CÓDIGO**



**Figura 2-6:** **MAPA DE MEMORIA Y CÓDIGO ESPACIO DE MEMORIA PARA DISPOSITIVOS PIC18FX5X5 / X6X0**



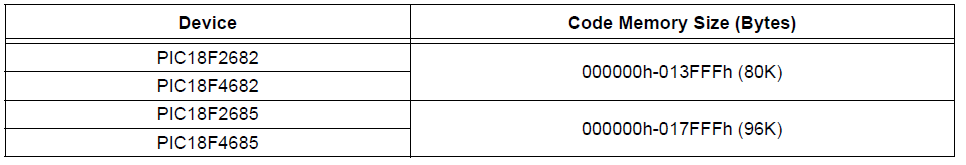
Nota: Los tamaños de las áreas de memoria no están a escala.

\* El tamaño del bloque de inicio está determinado por los bits BBSIZ <1: 0> en el registro CONFIG4L.

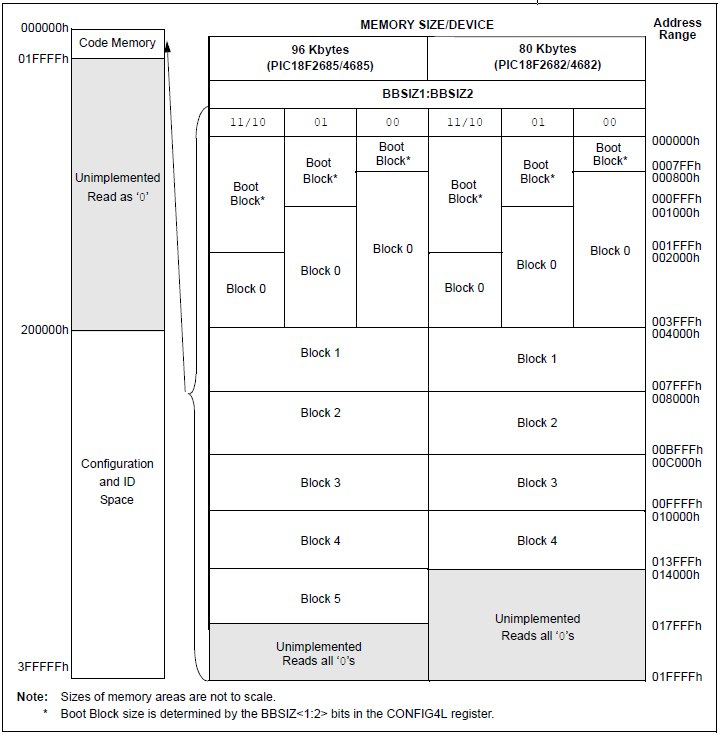
Para los dispositivos PIC18F2685 / 4685, el espacio de memoria de código se extiende desde 0000h a 017FFFh (96 Kbytes) en cinco bloques de 16 Kbytes. Para los dispositivos PIC18F2682 / 4682, el espacio de memoria de código se extiende desde 0000h a 0013FFFh (80 Kbytes) en cuatro bloques de 16 Kbytes. Las direcciones, 0000h a 0FFFh, sin embargo, definen una región de "Bloque de arranque" que se trata por separado del Bloque 0. Todos estos bloques definen límites de protección de código dentro del espacio de memoria de código.

El tamaño del Bloque de arranque en los dispositivos PIC18F2685 / 4685 y PIC18F2682 / 4682 se puede configurar como 1, 2 o 4K palabras (consulte la Figura 2-7). Esto se realiza a través de los bits BBSIZ <2: 1> en el registro de configuración, CONFIG4L. Es importante tener en cuenta que al aumentar el tamaño del Bloque de arranque disminuye el tamaño del Bloque 0.

**Tabla 2-3: IMPLEMENTACIÓN DE LA MEMORIA DEL CÓDIGO**



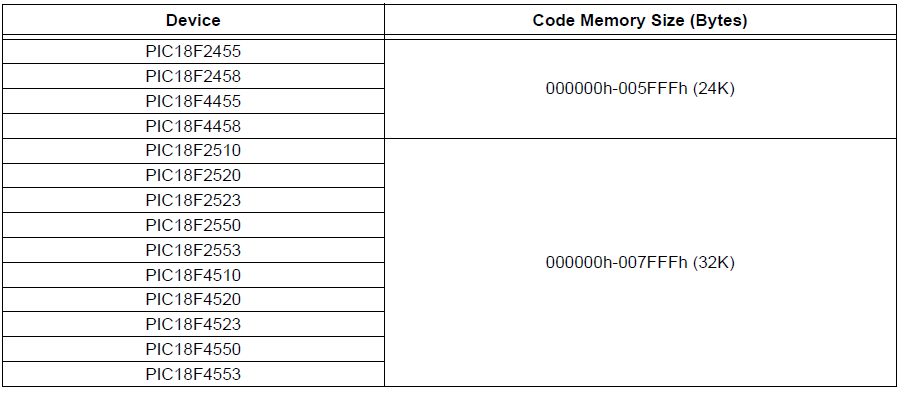
**Figura 2-7: MAPA DE MEMORIA Y CÓDIGO ESPACIO DE MEMORIA PARA DISPOSITIVOS PIC18F2685 / 4685 Y PIC18F2682 / 4682**



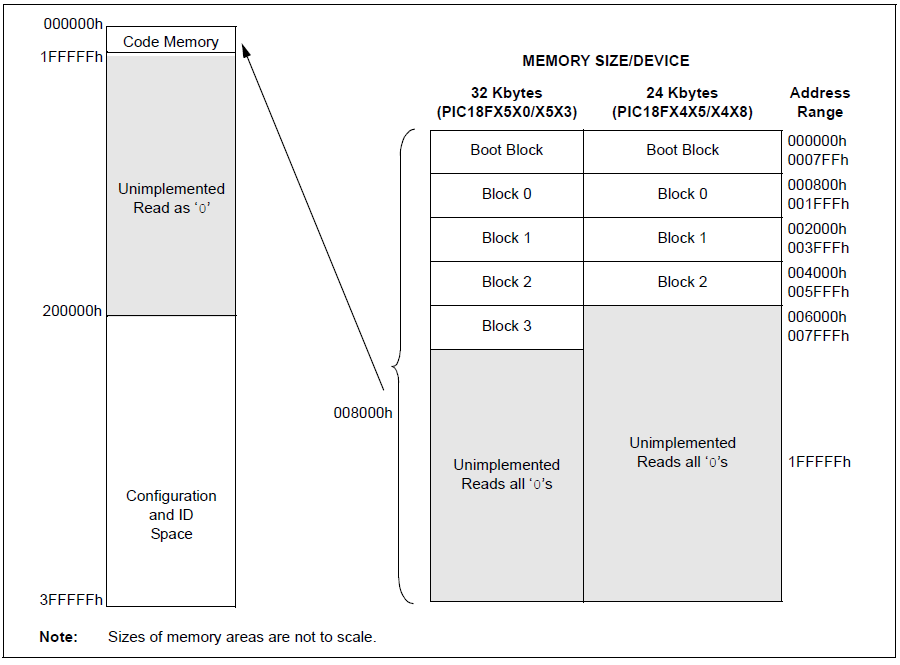
**Nota:** Los tamaños de las áreas de memoria no están a escala. \* El tamaño del bloque de arranque está determinado por los bits BBSIZ <1: 2> en el registro CONFIG4L

Para los dispositivos PIC18FX5X0 / X5X3, el espacio de memoria de código se extiende desde 000000h a 007FFFh (32 Kbytes) en cuatro bloques de 8 Kbytes. Para los dispositivos PIC18FX4X5 / X4X8, el espacio de memoria de código se extiende desde 000000h a 005FFFh (24 Kbytes) en tres bloques de 8 Kbytes. Sin embargo, las direcciones de 000000h a 0007FFh definen una región de "Bloque de arranque" que se trata por separado del Bloque 0. Todos estos bloques definen límites de protección de código dentro del espacio de memoria de código.

**TABLA 2-4: IMPLEMENTACIÓN DE LA MEMORIA DEL CÓDIGO**

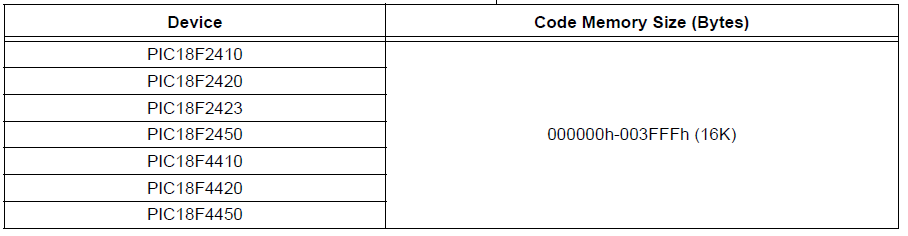


**FIGURA 2-8: MAPA DE MEMORIA Y CÓDIGO ESPACIO DE MEMORIA PARA DISPOSITIVOS PIC18FX4X5 / X4X8 / X5X0 / X5X3**

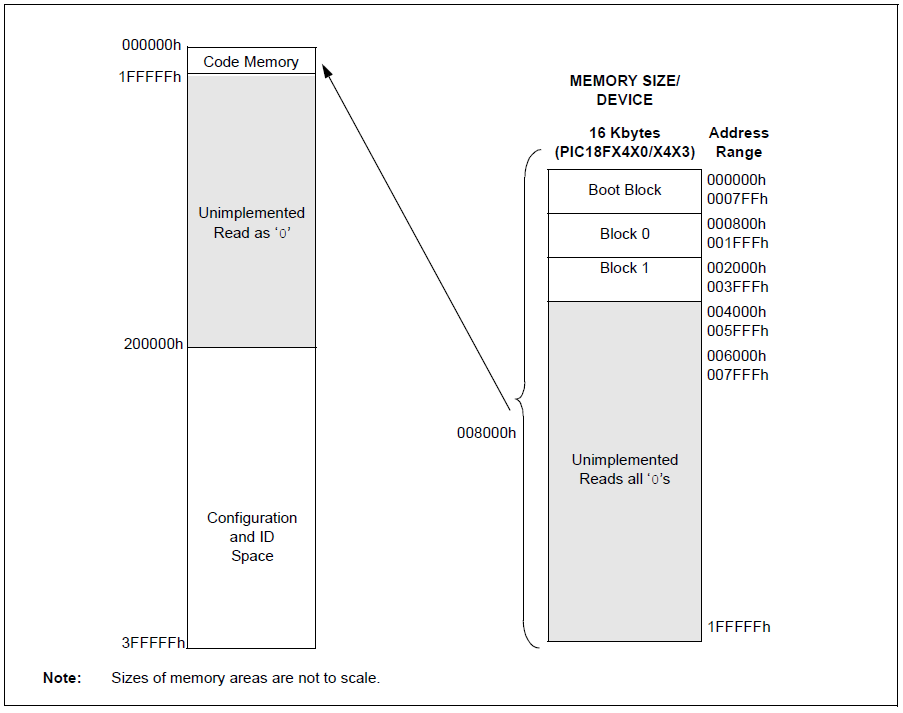


Para los dispositivos PIC18FX4X0 / X4X3, el espacio de memoria de código se extiende desde 000000h a 003FFFh (16 Kbytes) en dos bloques de 8 Kbytes. Sin embargo, las direcciones de 000000h a 0003FFh definen una región de "Bloque de arranque" que se trata por separado del Bloque 0. Todos estos bloques definen límites de protección de código dentro del espacio de memoria de código.

**TABLA 2-5: IMPLEMENTACIÓN DE LA MEMORIA DEL CÓDIGO**



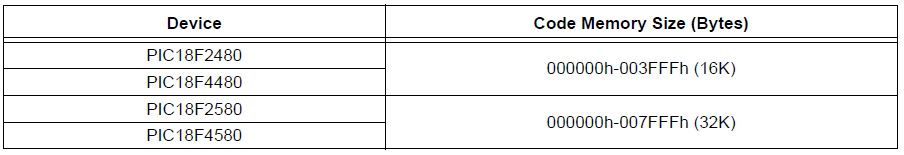
**FIGURA 2-9: MAPA DE MEMORIA Y CÓDIGO ESPACIO DE MEMORIA  PARA DISPOSITIVOS PIC18FX4X0 / X4X3**



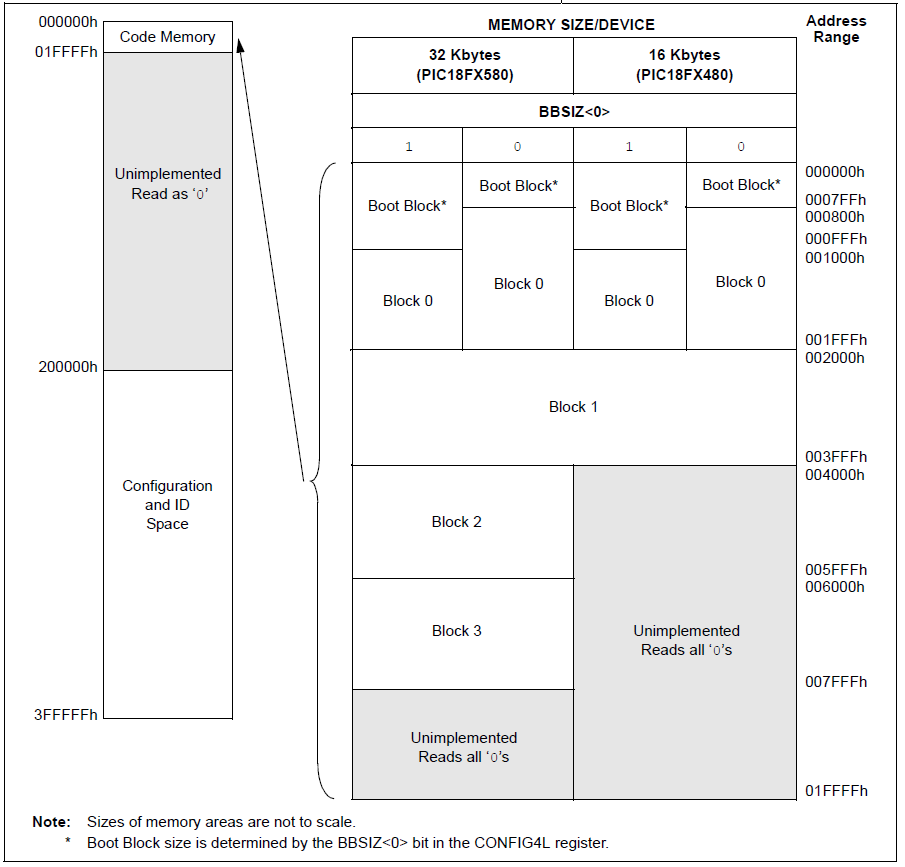
Para los dispositivos PIC18F2480 / 4480, el espacio de memoria de código se extiende desde 0000h a 03FFFh (16 Kbytes) en un bloque de 16 Kbytes. Para los dispositivos PIC18F2580 / 4580, el espacio de memoria de código se extiende desde 0000h a 07FFFh (32 Kbytes) en dos bloques de 16 Kbytes. Las direcciones, 0000h a 07FFh, sin embargo, definen una región de "Bloque de arranque" que se trata por separado del Bloque 0. Todos estos bloques definen límites de protección de código dentro del espacio de memoria de código.

El tamaño del Bloque de arranque en los dispositivos PIC18F2480 / 2580/4480/4580 se puede configurar como 1 o 2K palabras (consulte la Figura 2-10). Esto se realiza a través del bit BBSIZ <0> en el registro de configuración, CONFIG4L. Es importante tener en cuenta que al aumentar el tamaño del Bloque de arranque disminuye el tamaño del Bloque 0.

**TABLA 2-6: IMPLEMENTACIÓN DE LA MEMORIA DEL CÓDIGO**



**FIGURA 2-10: MAPA DE MEMORIA Y CÓDIGO ESPACIO DE MEMORIA PARA DISPOSITIVOS PIC18F2480 / 2580/4480/4580**

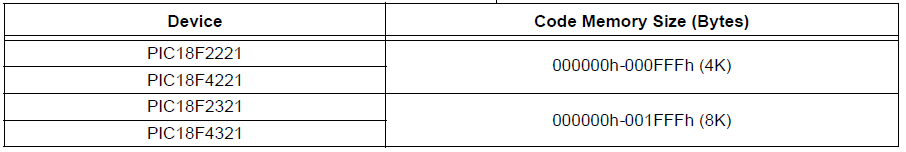


**Nota: Los tamaños de las áreas de memoria no están a escala. \* El tamaño del bloque de arranque está determinado por el bit BBSIZ <0> en el registro CONFIG4L.**

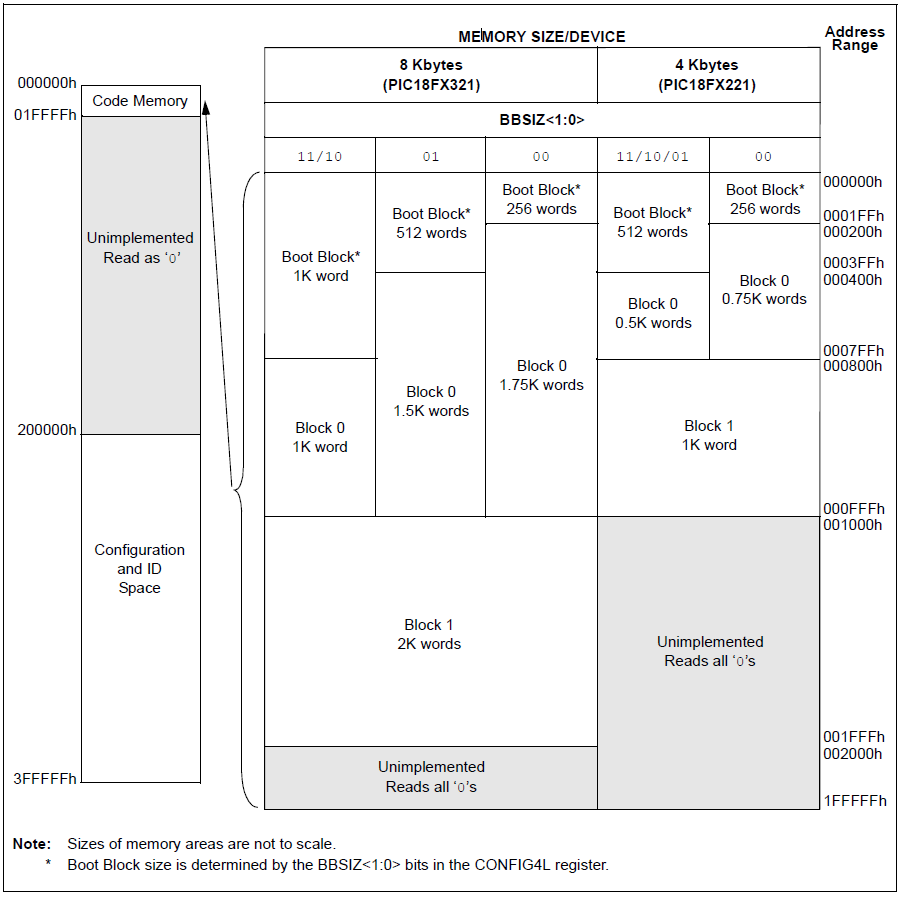
Para los dispositivos PIC18F2221 / 4221, el espacio de memoria de código se extiende desde 0000h a 00FFFh (4 Kbytes) en un bloque de 4 Kbytes. Para los dispositivos PIC18F2321 / 4321, el espacio de memoria de código se extiende desde 0000h a 01FFFh (8 Kbytes) en dos bloques de 4 Kbytes. Sin embargo, las direcciones de 0000h a 07FFh definen una región variable de "Bloque de arranque" que se trata por separado del Bloque 0. Todos estos bloques definen límites de protección de código dentro del espacio de memoria de código.

El tamaño del Bloque de arranque en los dispositivos PIC18F2221 / 2321/4221/4321 se puede configurar como 256, 512 o 1024 palabras (consulte la Figura 2-11). Esto se realiza a través de los bits BBSIZ <1: 0> en el registro de configuración, CONFIG4L (ver Figura 2-11). Es importante tener en cuenta que al aumentar el tamaño del Bloque de arranque disminuye el tamaño del Bloque 0.

**TABLA 2-7: IMPLEMENTACIÓN DE LA MEMORIA DEL CÓDIGO**



**FIGURA 2-11: MAPA DE MEMORIA Y CÓDIGO ESPACIO DE MEMORIA PARA DISPOSITIVOS PIC18F2221 / 2321/4221/4321**



**Nota: Los tamaños de las áreas de memoria no están a escala. \* El tamaño del bloque de arranque está determinado por los bits BBSIZ <1: 0> en el registro CONFIG4L.**

Además del espacio de memoria de código, hay tres bloques que son accesibles para el usuario a través de Lecturas de tabla y Escrituras de tabla. Sus ubicaciones en el mapa de memoria se muestran en la Figura 2-12.

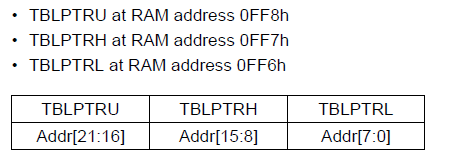
Los usuarios pueden almacenar información de identificación (ID) en ocho registros de ID. Estos registros de ID se asignan en direcciones, 200000h a 200007h. Las ubicaciones de ID se leen normalmente, incluso después de aplicar la protección de código.

Las ubicaciones, 300000h a 30000Dh, están reservadas para los bits de configuración. Estos bits seleccionan varias opciones de dispositivo y se describen en la Sección 5.0 “Palabra de configuración”. Estos bits de configuración se leen normalmente, incluso después de la protección del código.

Las ubicaciones, 3FFFFEh y 3FFFFFh, están reservadas para los bits de ID de dispositivo. El programador puede utilizar estos bits para identificar qué tipo de dispositivo se está programando y se describen en la Sección 5.0 “Palabra de configuración”. Estos bits de ID de dispositivo se leen normalmente, incluso después de la protección del código.

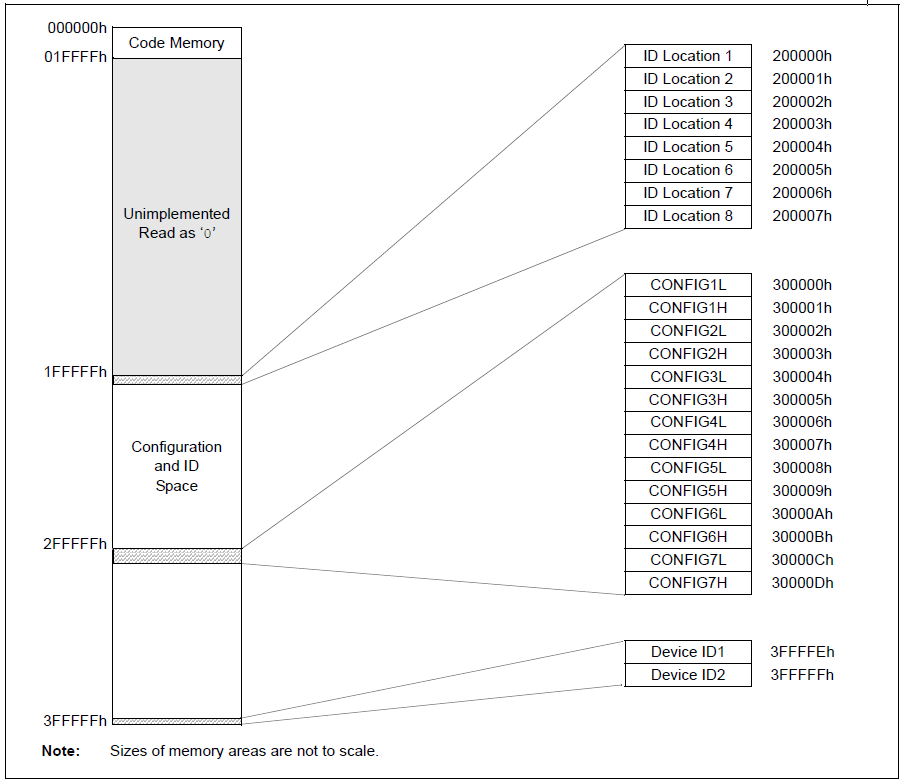
2.3.1 PUNTERO DE DIRECCIÓN DE MEMORIA

La memoria en el espacio de direcciones, 0000000h a 3FFFFFh, se direcciona a través del registro de puntero de tabla, que consta de tres registros de puntero:



El comando de 4 bits, "0000" (instrucción principal), se usa para cargar el puntero de tabla antes de usar muchas operaciones de lectura o escritura.

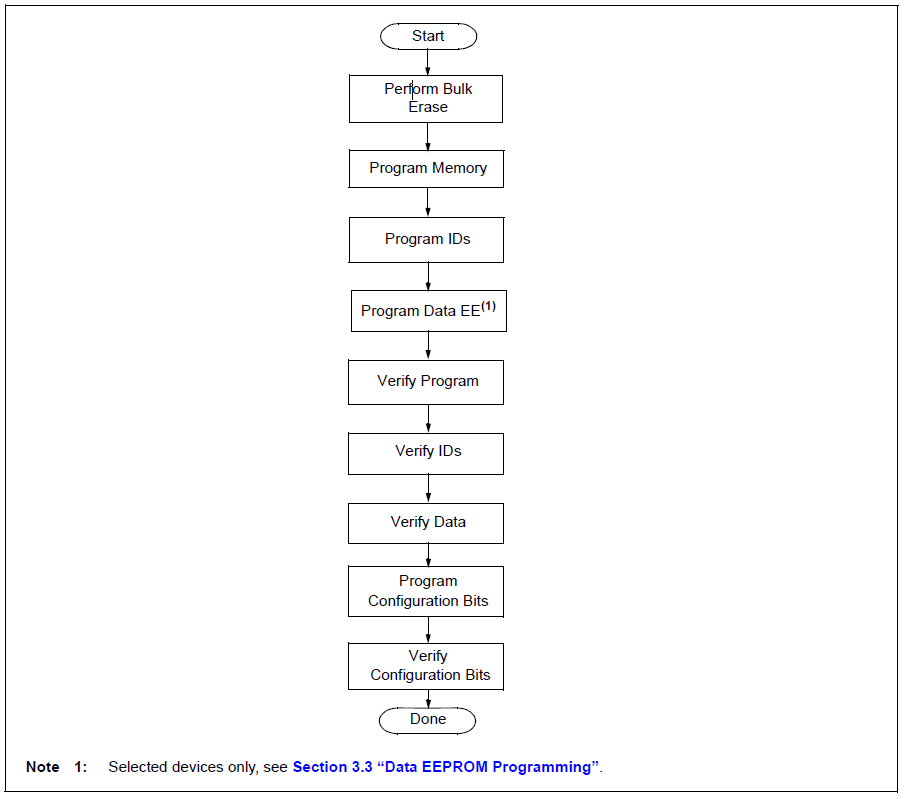
**FIGURA 2-12: UBICACIONES DE CONFIGURACIÓN E ID PARA LOS DISPOSITIVOS FAMILIARES PIC18F2XXX / 4XXX**



**2.4 Descripción general de alto nivel del proceso de programación**

La Figura 2-13 muestra la descripción general de alto nivel del proceso de programación. Primero, se realiza un borrado masivo. A continuación, se programan la memoria de código, las ubicaciones de ID y la EEPROM de datos (solo dispositivos seleccionados, consulte la Sección 3.3 “Programación de EEPROM de datos”). Estas memorias luego se verifican para asegurar que la programación fue exitosa. Si no se detectan errores, los bits de configuración se programan y verifican.

**FIGURA 2-13: FLUJO DE PROGRAMACIÓN DE ALTO NIVEL**



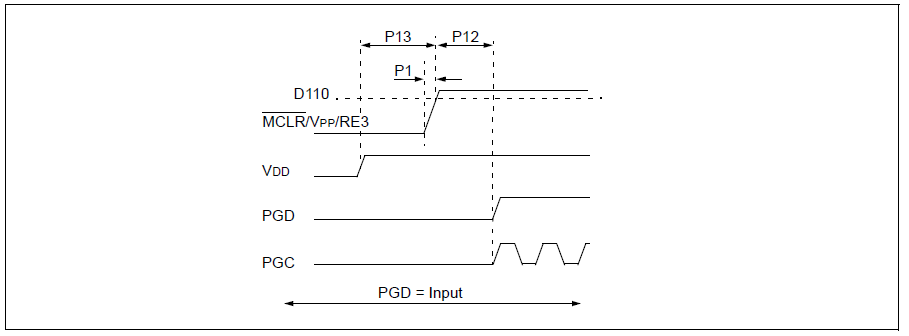
**Nota 1: Solo dispositivos seleccionados, consulte la Sección 3.3 “Programación de EEPROM de datos”.**

**2.5 Entrar y salir del programa ICSP de alto voltaje / modo de verificación**

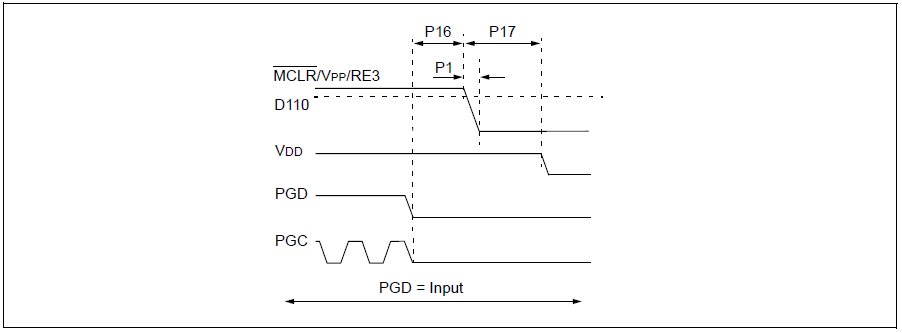
Como se muestra en la Figura 2-14, se ingresa al modo Programa / Verificación ICSP de alto voltaje manteniendo PGC y PGD bajos y luego elevando MCLR / VPP / RE3 a VIHH (alto voltaje). Una vez en este modo, la memoria de código, la EEPROM de datos (solo dispositivos seleccionados, consulte la Sección 3.3 “Programación de EEPROM de datos”), las ubicaciones de ID y los bits de configuración se pueden acceder y programar en serie. La Figura 2-15 muestra la secuencia de salida.

La secuencia que ingresa al dispositivo en el modo Programa / Verificación coloca todas las E / S no utilizadas en el estado de alta impedancia.

**FIGURA 2-14: INGRESO AL PROGRAMA DE ALTO VOLTAJE / MODO DE VERIFICACIÓN**



**FIGURA 2-15: SALIR DEL PROGRAMA DE ALTO VOLTAJE / MODO DE VERIFICACIÓN**

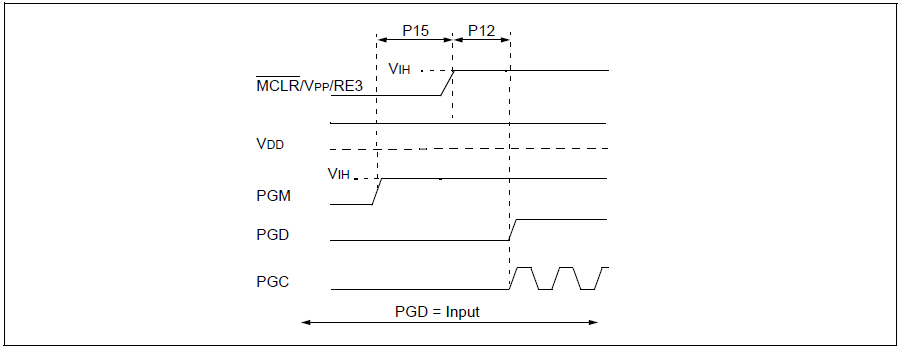


**2.6 Entrar y salir del programa ICSP de bajo voltaje / modo de verificación**

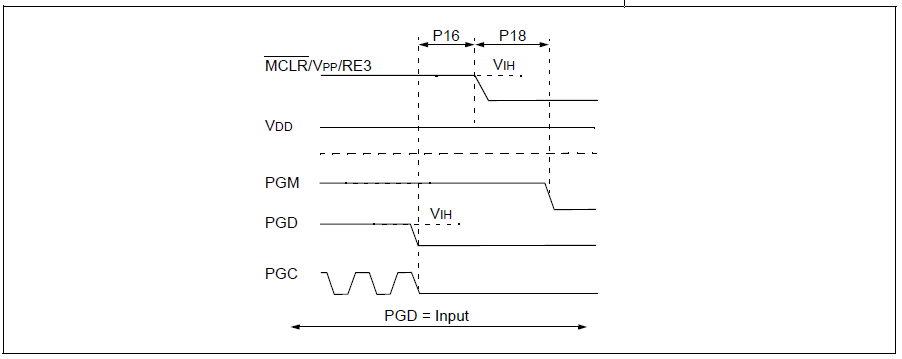
Cuando el bit de configuración de LVP es ‘1’ (consulte la Sección 5.3 “Programación de ICSP de suministro único”), se habilita el modo ICSP de bajo voltaje. Como se muestra en la Figura 2-16, se ingresa al modo Programa / Verificación ICSP de bajo voltaje manteniendo PGC y PGD bajos, colocando una lógica alta en PGM y luego elevando MCLR / VPP / RE3 a VIH. En este modo, el pin RB5 / PGM está dedicado a la función de programación y deja de ser un pin de E / S de propósito general. La Figura 2-17 muestra la secuencia de salida.

La secuencia que ingresa al dispositivo en el modo Programa / Verificación coloca todas las E / S no utilizadas en el estado de alta impedancia.

**FIGURA 2-16: INGRESO AL PROGRAMA DE BAJO VOLTAJE / MODO DE VERIFICACIÓN**



**FIGURA 2-17: SALIR DEL PROGRAMA DE BAJO VOLTAJE / MODO DE VERIFICACIÓN**



**2.7 Programa en serie / Operación de verificación**

El pin PGC se usa como un pin de entrada de reloj y el pin PGD se usa para ingresar bits de comando y entrada / salida de datos durante la operación en serie. Los comandos y los datos se transmiten en el borde ascendente de PGC, enganchados en el borde descendente de PGC y son los bits menos significativos (LSb) primero.

2.7.1 COMANDOS DE 4 BITS

Todas las instrucciones son de 20 bits, que consisten en un comando principal de 4 bits seguido de un operando de 16 bits, que depende del tipo de comando que se ejecute. Para ingresar un comando, PGC se cicla cuatro veces. Los comandos necesarios para la programación y la verificación se muestran en la Tabla 2-8.

Dependiendo del comando de 4 bits, el operando de 16 bits representa 16 bits de datos de entrada u 8 bits de datos de entrada y 8 bits de datos de salida.

A lo largo de esta especificación, los comandos y los datos se presentan como se ilustra en la Tabla 2-9. El comando de 4 bits se muestra primero como el bit más significativo (MSb). El operando de comando, o "Carga de datos", se muestra como <MSB> <LSB>. La Figura 2-18 muestra cómo presentar en serie un comando / operando de 20 bits al dispositivo.

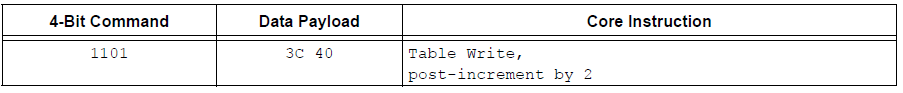
2.7.2 INSTRUCCIÓN BÁSICA

La instrucción central pasa una instrucción de 16 bits al núcleo de la CPU para su ejecución. Esto es necesario para configurar registros según corresponda para su uso con otros comandos.

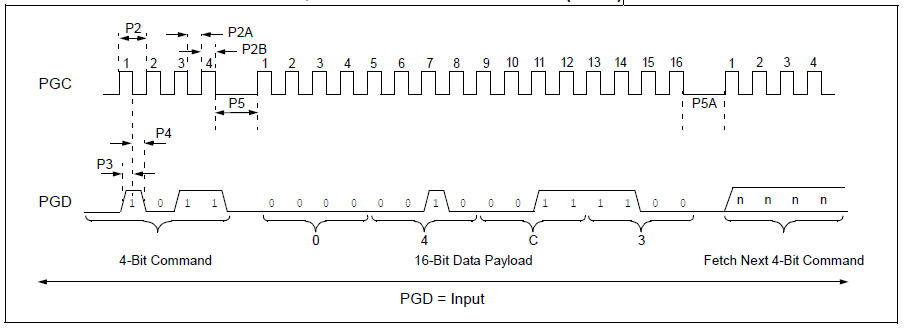
**TABLA 2-8: MANDOS PARA PROGRAMACIÓN**

|  |  |  |
| --- | --- | --- |
| **Descripción** | Traducción | Comando de 4 bits |
| Core Instruction (Shift in16-bit instruction) | Instrucción principal (cambio en la instrucción de 16 bits) | 0000 |
| Shift Out TABLAT Register | Shift Out Registro TABLAT | 0010 |
| Table Read | Tabla leída | 1000 |
| Table Read, Post-Increment | Tabla Leída, Post-Incremento | 1001 |
| Table Read, Post-Decrement | Tabla leída, posterior a la disminución | 1010 |
| Table Read, Pre-Increment | Lectura de tabla, pre-incremento | 1011 |
| Table Write | Escribir tabla | 1100 |
| Table Write, Post-Increment by 2 | Escritura de tabla, post-incremento en 2 | 1101 |
| Table Write, Start Programming, Post-Increment by 2 | Escritura de tabla, inicio de programación, Post-incremento en 2 | 1110 |
| Table Write, Start Programming | Escribir en tabla, comenzar a programar | 1111 |

**TABLA 2-9: SECUENCIA DE COMANDO DE MUESTRA**



**FIGURA 2-18: ESCRITURA DE TABLA, TIEMPO POSTERIOR AL INCREMENTO (1101)**



**2.8 Puerto ICSP / ICD dedicado (solo TQFP de 44 pines)**

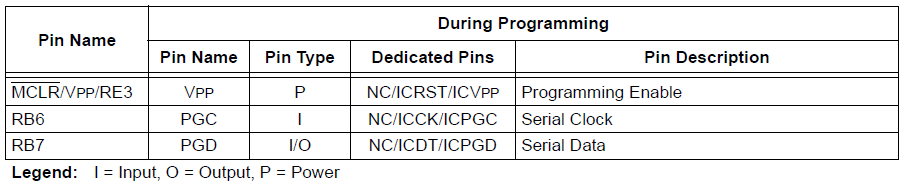
Los dispositivos TQFP PIC18F4455 / 4458/4550/4553 de 44 pines están diseñados para admitir una entrada de programación alternativa: el puerto ICSP / ICD dedicado. El propósito principal de este puerto es proporcionar una opción alternativa de depuración en circuito (ICD) y liberar los pines (RB6, RB7 y MCLR) que normalmente se usarían para depurar la aplicación. Sin embargo, junto con la capacidad de ICD, el puerto dedicado ICSP / ICD también proporciona un puerto alternativo para ICSP.

Establecer el bit de configuración ICPRT habilita el puerto dedicado ICSP / ICD. El puerto ICSP / ICD dedicado funciona igual que el puerto ICSP / ICD predeterminado; sin embargo, se utilizan pines alternativos en lugar de los pines predeterminados. La Tabla 2-10 identifica los pines funcionalmente equivalentes para fines de ICSP:

El puerto dedicado ICSP / ICD es un puerto alternativo. Por lo tanto, ICSP todavía está disponible a través del puerto predeterminado aunque el bit de configuración ICPRT esté configurado. Cuando se ve el VIH en el pin MCLR / VPP / RE3 antes de aplicar VIH al pin ICRST / ICVPP, se ignora el estado del pin ICRST / ICVPP. Del mismo modo, cuando se ve el VIH en ICRST / ICVPP antes de aplicar el VIH a MCLR / VPP / RE3, se ignora el estado del pin MCLR / VPP / RE3.

**Nota: El bit de configuración ICPRT solo se puede programar a través del puerto ICSP predeterminado. Las funciones de borrado de chip a través del puerto ICSP / ICD dedicado no afectan este bit. Cuando se establece el bit de configuración ICPRT (puerto dedicado ICSP / ICD habilitado), el pin NC / ICPORTS debe estar vinculado a VDD o VSS. El bit de configuración ICPRT debe mantenerse libre para todos los dispositivos de 28 y 40 pines; de lo contrario, puede ocurrir una operación inesperada**

**TABLA 2-10: PINES EQUIVALENTES ICSP ™**



**3.0 PROGRAMACIÓN DE DISPOSITIVOS**

La programación incluye la capacidad de borrar o escribir las diversas regiones de memoria dentro del dispositivo.

En todos los casos, excepto el borrado masivo ICSP de alto voltaje, el registro EECON1 debe configurarse para operar en una región de memoria particular.

Cuando se utiliza el registro EECON1 para actuar en la memoria de código, se debe establecer el bit EEPGD (EECON1 <7> = 1) y se debe borrar el bit CFGS (EECON1 <6> = 0). El bit WREN debe establecerse (EECON1 <2> = 1) para permitir escrituras de cualquier tipo (por ejemplo, borrados) y esto debe hacerse antes de iniciar una secuencia de escritura. El bit FREE debe establecerse (EECON1 <4> = 1) para borrar el espacio del programa al que apunta el puntero de tabla. La secuencia de borrado o escritura se inicia configurando el bit WR (EECON1 <1> = 1). Se recomienda encarecidamente que el bit WREN solo se establezca inmediatamente antes del borrado de un programa.

**3.1 Borrar ICSP**

3.1.1 BORRADO A GRANEL ICSP DE ALTO VOLTAJE

El borrado de código o datos EEPROM se logra mediante la configuración de dos registros de control de borrado masivo ubicados en 3C0004h y 3C0005h. La memoria de código puede borrarse, porciones a la vez, o el usuario puede borrar todo el dispositivo en una sola acción. Las operaciones de borrado masivo también borrarán cualquier configuración de protección de código asociada con el bloque de memoria que se borra. Las opciones de borrado se detallan en la Tabla 3-1. Si la EEPROM de datos está protegida por código (CPD = 0), el usuario debe solicitar un borrado de EEPROM de datos (por ejemplo, 0084h como se muestra en la Tabla 3-1).

**TABLA 3-1: OPCIONES DE BORRADO A GRANEL**

|  |  |  |
| --- | --- | --- |
| Descripción | Traducción | Datos  **(3C0005h:3C0004h)** |
| Chip Erase | Borrado del chip | 3F8Fh |
| Erase Data EEPROM**(1)** | Borrar datos EEPROM (1) | 0084h |
| Erase Boot Block | Borrar bloque de arranque | 0081h |
| Erase Configuration Bits | Borrar bits de configuración | 0082h |
| Erase Code EEPROM Block 0 | Borrar código EEPROM Bloque 0 | 0180h |
| Erase Code EEPROM Block 1 | Borrar Código EEPROM Bloque 1 | 0280h |
| Erase Code EEPROM Block 2 | Borrar Código EEPROM Bloque 2 | 0480h |
| Erase Code EEPROM Block 3 | Borrar Código EEPROM Bloque 3 | 0880h |
| Erase Code EEPROM Block 4 | Borrar Código EEPROM Bloque 4 | 1080h |
| Erase Code EEPROM Block 5 | Borrar Código EEPROM Bloque 5 | 2080h |

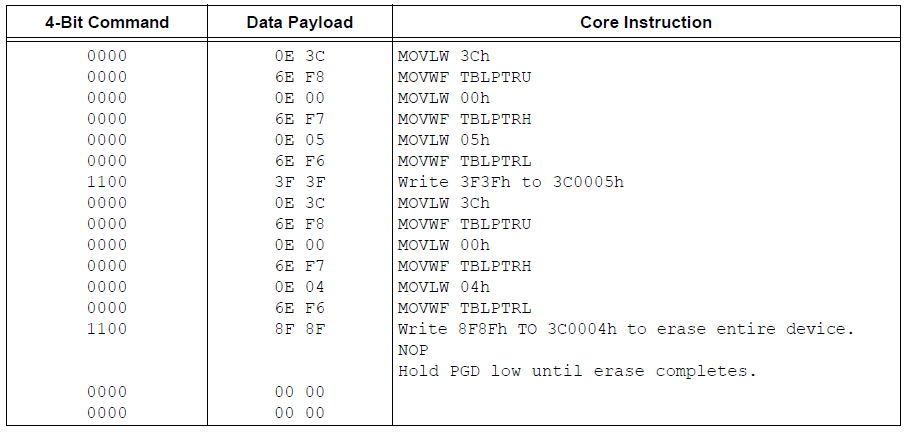
**Nota 1: Solo dispositivos seleccionados, consulte la Sección 3.3 “Programación de EEPROM de datos”.**

La función de borrado masivo real es una operación temporizada. Una vez que ha comenzado el borrado (borde descendente del 4º PGC después del comando NOP), la ejecución en serie cesará hasta que se complete el borrado (Parámetro P11). Durante este tiempo, PGC puede continuar alternando pero PGD debe mantenerse bajo.

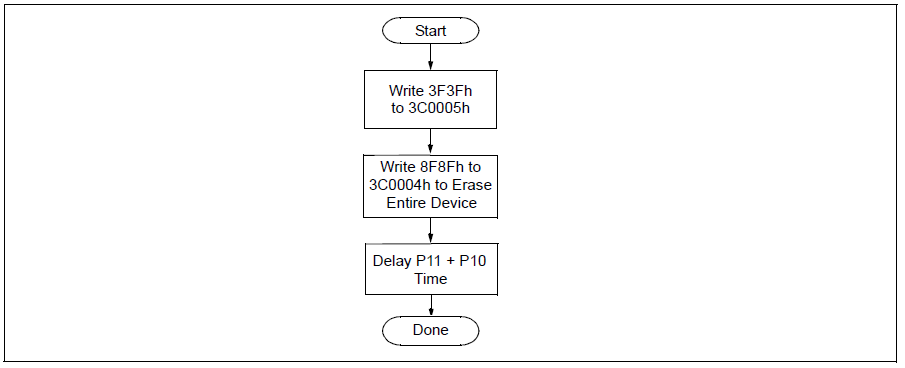
La secuencia de códigos para borrar todo el dispositivo se muestra en la Tabla y el diagrama de flujo se muestra en la Figura 3-1.

**Nota: Un borrado masivo es la única forma de reprogramar los bits de protección de código de un estado ON a un estado OFF.**

**TABLA 3-2: SECUENCIA DE COMANDO DE BORRADO A GRANEL**



**FIGURA 3-1: FLUJO DE BORRADO A GRANEL**



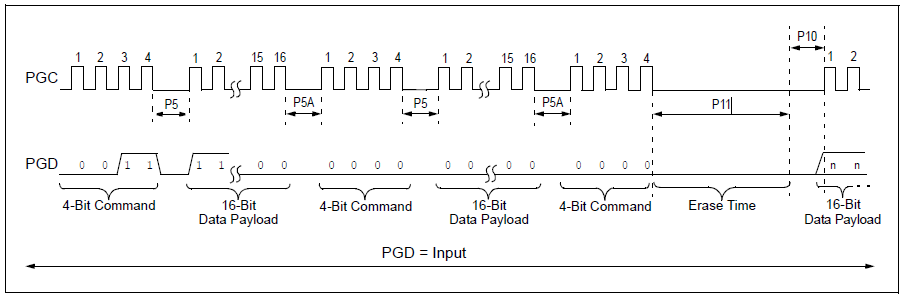
3.1.2 BORRADO A GRANEL ICSP DE BAJO VOLTAJE

Cuando se utiliza ICSP de bajo voltaje, la parte debe ser alimentada por el voltaje especificado en el parámetro D111 si se va a ejecutar un borrado masivo. Se aplican todos los demás detalles de borrado masivo, como se describió anteriormente.

Si se determina que un borrado de la memoria del programa debe realizarse a un voltaje de suministro por debajo del límite de borrado masivo, consulte la metodología de borrado descrita en la Sección 3.1.3 “Borrado de filas ICSP” y la Sección 3.2.1 “Modificación de la memoria de código”.

Si se determina que un borrado de EEPROM de datos (solo dispositivos seleccionados, consulte la Sección 3.3 “Programación de EEPROM de datos”) debe realizarse a un voltaje de suministro por debajo del límite de Borrado a granel, siga la metodología descrita en la Sección 3.3 “Programación de EEPROM de datos” y escriba '1's a la matriz.

**FIGURA 3-2: TIEMPO DE BORRADO A GRANEL**



3.1.3 BORRADO DE FILAS ICSP

Independientemente de si se utiliza ICSP de alto o bajo voltaje, es posible borrar una fila (64 bytes de datos), siempre que el bloque no esté protegido contra códigos o escritura. Las filas están ubicadas en límites estáticos, comenzando en la dirección de memoria del programa, 000000h, extendiéndose hasta el límite interno de la memoria del programa (consulte la Sección 2.3 “Mapas de memoria”).

La duración del borrado de fila está programada externamente y está controlada por PGC. Después de establecer el bit WR en EECON1, se emite un NOP, donde el 4º PGC se mantiene alto durante el tiempo de programación, P9

Después de que PGC se baja, la secuencia de programación finaliza. PGC debe mantenerse bajo durante el tiempo especificado por Parámetro P10 para permitir la descarga de alto voltaje de la matriz de memoria.

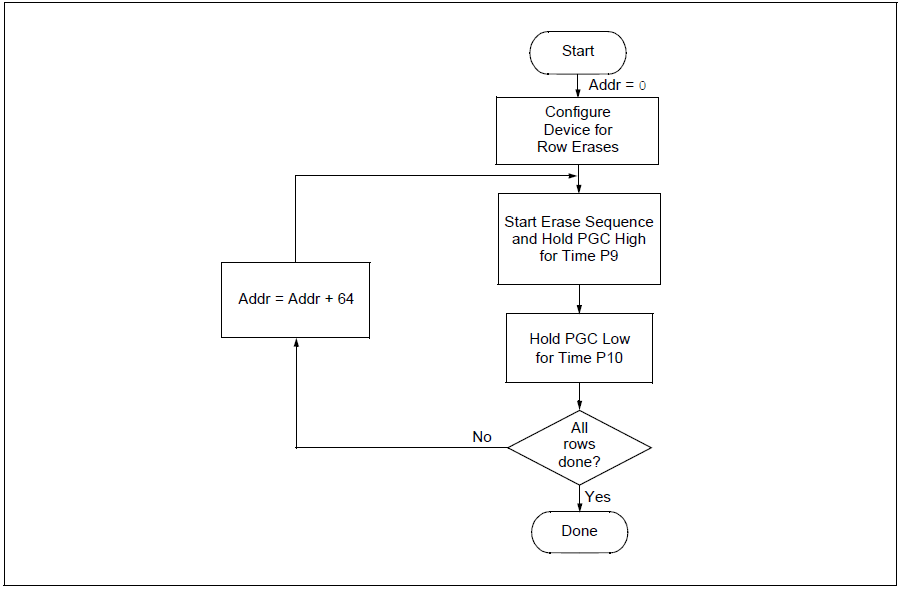
La secuencia de código para borrar un dispositivo de la familia PIC18F2XXX / 4XXX se muestra en la Tabla 3-3. El diagrama de flujo, que se muestra en la Figura 3-3, muestra la lógica necesaria para borrar completamente un dispositivo de la familia PIC18F2XXX / 4XXX. El diagrama de tiempos que detalla el comando Iniciar programación y los Parámetros P9 y P10 se muestra en la Figura 3-5.

**Nota: El registro TBLPTR puede apuntar a cualquier byte dentro de la fila destinada para borrar.**

**TABLA 3-3: BORRAR SECUENCIA DE CÓDIGO DE MEMORIA DEL CÓDIGO**

|  |  |  |
| --- | --- | --- |
| Comando 4 bits | Carga de datos | Instrucción central |
| Paso 1: acceso directo a la memoria de código y habilitar escrituras. | | |
| 0000 | 8E A6 | BSF EECON1, EEPGD |
| 0000 | 9C A6 | BCF EECON1, CFGS |
| 0000 | 84 A6 | BSF EECON1, WREN |
| Paso 2: apunte a la primera fila en la memoria de código. | | |
| 0000 | 6A F8 | CLRF TBLPTRU |
| 0000 | 6A F7 | CLRF TBLPTRH |
| 0000 | 6A F6 | CLRF TBLPTRL |
| Paso 3: habilite borrar y borrar una sola fila. | | |
| 0000 | 88 A6 | BSF EECON1, FREE |
| 0000 | 82 A6 | BSF EECON1, WR |
| 0000 | 00 00 | NOP – Mantenga PGC alto para el tiempo P9 y bajo para el tiempo P10. |
| Paso 4: repita el paso 3, con el puntero de dirección incrementado en 64 hasta que se borren todas las filas. | | |

**FIGURA 3-3: FLUJO PARA BORRADO DE UN REGISTRO DE MEMORIA DE CODIGO**



**3.2 Programación de memoria de código**

La memoria del código de programación se realiza cargando primero los datos en el búfer de escritura y luego iniciando una secuencia de programación. Los tamaños de búfer de escritura y borrado, que se muestran en la Tabla 3-4, se pueden asignar a cualquier ubicación del mismo tamaño, comenzando a las 000000h. La secuencia de escritura de memoria real toma el contenido de este búfer y programa la cantidad adecuada de memoria de código que contiene el puntero de tabla

La duración de la programación es temporizada externamente y está controlada por PGC. Después de que se emite un comando de Inicio de programación (comando de 4 bits, "1111"), se emite un NOP, donde el 4º PGC se mantiene alto durante el tiempo de programación, P9.

Después de que PGC se baja, la secuencia de programación finaliza. PGC debe mantenerse bajo durante el tiempo especificado por el parámetro P10 para permitir la descarga de alto voltaje de la matriz de memoria

La secuencia de códigos para programar un dispositivo de la familia PIC18F2XXX / 4XXX se muestra en la Tabla 3-5. El diagrama de flujo, que se muestra en la Figura 3-4, muestra la lógica necesaria para escribir completamente un dispositivo de la familia PIC18F2XXX / 4XXX. El diagrama de tiempos que detalla el comando Iniciar programación y los Parámetros P9 y P10 se muestra en la Figura 3-5.

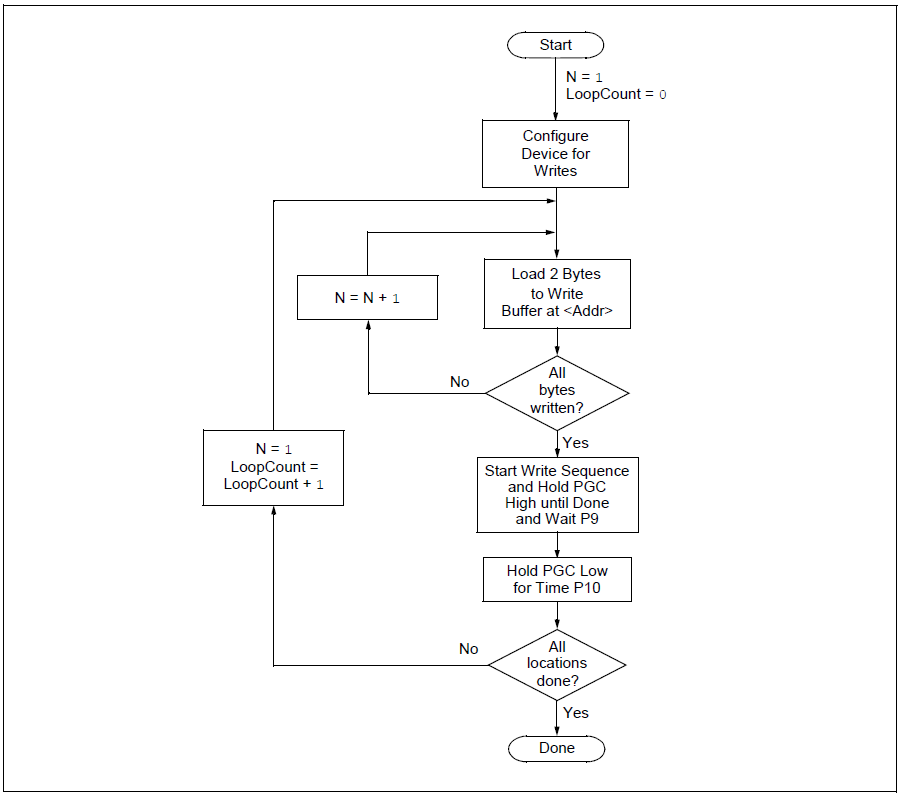
**TABLA 3-4: ESCRIBIR Y BORRAR TAMAÑOS DE BUFFER**

|  |  |  |
| --- | --- | --- |
| Dispositivos (organizados por familia) | Tamaño del búfer de escritura (bytes) | Tamaño del búfer de borrado (bytes) |
| PIC18F2221, PIC18F2321, PIC18F4221, PIC18F4321 | 8 | 64 |
| PIC18F2450, PIC18F4450 | 16 | 64 |
| PIC18F2410, PIC18F2510, PIC18F4410, PIC18F4510 | 32 | 64 |
| PIC18F2420, PIC18F2520, PIC18F4420, PIC18F4520 |
| PIC18F2423, PIC18F2523, PIC18F4423, PIC18F4523 |
| PIC18F2480, PIC18F2580, PIC18F4480, PIC18F4580 |
| PIC18F2455, **PIC18F2550**, PIC18F4455, PIC18F4550 |
| PIC18F2458, PIC18F2553, PIC18F4458, PIC18F4553 |
| PIC18F2515, PIC18F2610, PIC18F4515, PIC18F4610 | 64 | 64 |
| PIC18F2525, PIC18F2620, PIC18F4525, PIC18F4620 |
| PIC18F2585, PIC18F2680, PIC18F4585, PIC18F4680 |
| PIC18F2682, PIC18F2685, PIC18F4682, PIC18F4685 |

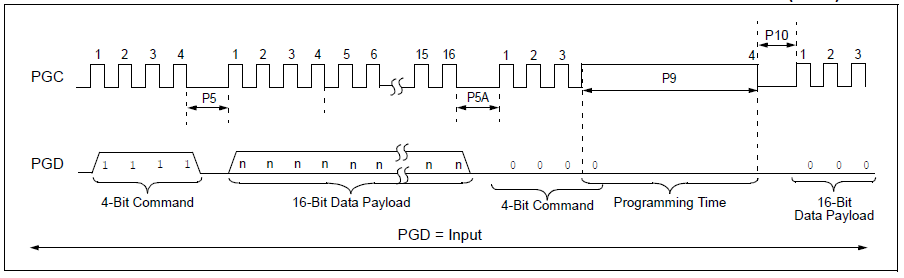
**TABLA 3-5: ESCRIBA LA SECUENCIA DEL CÓDIGO DE MEMORIA DEL CÓDIGO**

|  |  |  |
| --- | --- | --- |
| Comando 4 bits | Carga de datos | Instrucción central |
| Paso 1: acceso directo a la memoria de código y habilitar escrituras. | | |
| 0000 | 8E A6 | BSF EECON1, EEPGD |
| 0000 | 9C A6 | BCF EECON1, CFGS |
| Paso 2: cargar el búfer de escritura. | | |
| 0000 | 0E <Addr[21:16]> | MOVLW <Addr[21:16]> |
| 0000 | 6E F8 | MOVWF TBLPTRU |
| 0000 | 0E <Addr[15:8]> | MOVLW <Addr[15:8]> |
| 0000 | 6E F7 | MOVWF TBLPTRH |
| 0000 | 0E <Addr[7:0]> | MOVLW <Addr[7:0]> |
| 0000 | 6E F6 | MOVWF TBLPTRL |
| Paso 3: repita para todos menos los dos últimos bytes | | |
| 1101 | <MSB><LSB> | Escribe 2 bytes y pos-incrementa la dirección en 2. |
| Paso 4: Cargue el búfer de escritura para los últimos dos bytes. | | |
| 1111 | <MSB><LSB> | Escribe 2 bytes y comienza a programar |
| 0000 | 00 00 | NOP - Mantenga PGC alto para el tiempo P9 y bajo para el tiempo P10. |
| Para continuar escribiendo datos, repita los pasos 2 a 4, donde el puntero de dirección se incrementa en 2 en cada iteración del bucle. | | |

**FIGURA 3-4: FLUJO DE MEMORIA DEL CÓDIGO DE PROGRAMA**



**FIGURA 3-5: TABLA ESCRIBA Y COMIENCE LA PROGRAMACIÓN DE PROGRAMACIÓN DE TIEMPO (1111)**



3.2.1 MODIFICANDO LA MEMORIA DEL CÓDIGO

El ejemplo de programación anterior suponía que el dispositivo había sido borrado a granel antes de la programación (consulte la Sección 3.1.1 "Borrado a granel ICSP de alto voltaje"). Sin embargo, puede darse el caso de que el usuario desee modificar solo una sección de un dispositivo ya programado.

El número apropiado de bytes necesarios para el búfer de borrado debe leerse de la memoria de código (como se describe en la Sección 4.2 "Verificar la memoria de código y las ubicaciones de ID") y almacenarse en el búfer. Se pueden hacer modificaciones en este búfer. Luego, el bloque de memoria de código que se leyó debe borrarse y reescribirse con los datos modificados.

El bit WREN debe establecerse si el bit WR en EECON1 se usa para iniciar una secuencia de escritura.

**TABLA 3-6: MODIFICANDO LA MEMORIA DEL CÓDIGO**

|  |  |  |
| --- | --- | --- |
| Comando 4 bits | Carga de datos | Instrucción central |
| Paso 1: acceso directo a la memoria de código. | | |
| Paso 2: Lea y modifique la memoria de código (consulte la Sección 4.1 “Lectura de memoria de código, ubicaciones de ID y bits de configuración”). | | |
| 0000 | 8E A6 | BSF EECON1, EEPGD |
| 0000 | 9C A6 | BCF EECON1, CFGS |
| Paso 3: establezca el puntero de tabla para que se borre el bloque. | | |
| 0000 | 0E <Addr[21:16]> | MOVLW <Addr[21:16]> |
| 0000 | 6E F8 | MOVWF TBLPTRU |
| 0000 | 0E <Addr[8:15]> | MOVLW <Addr[8:15]> |
| 0000 | 6E F7 | MOVWF TBLPTRH |
| 0000 | 0E <Addr[7:0]> | MOVLW <Addr[7:0]> |
| 0000 | 6E F6 | MOVWF TBLPTRL |
| Paso 4: habilite las escrituras de memoria y configure un borrado. | | |
| 0000 | 84 A6 | BSF EECON1, WREN |
| 0000 | 88 A6 | BSF EECON1, FREE |
| Paso 5: Inicia el borrado. | | |
| 0000 | 82 A6 | BSF EECON1, WR |
| 0000 | 00 00 | NOP - Mantenga PGC alto para el tiempo P9 y bajo para el tiempo P10. |
| Paso 6: Cargue el búfer de escritura. Se seleccionarán los bytes correctos en función del puntero de tabla | | |
| 0000 | 0E <Addr[21:16]> | MOVLW <Addr[21:16]> |
| 0000 | 6E F8 | MOVWF TBLPTRU |
| 0000 | 0E <Addr[8:15]> | MOVLW <Addr[8:15]> |
| 0000 | 6E F7 | MOVWF TBLPTRH |
| 0000 | 0E <Addr[7:0]> | MOVLW <Addr[7:0]> |
| 0000 | 6E F6 | MOVWF TBLPTRL |
| 1101 | <MSB><LSB> | Escribe 2 bytes y pos- incrementa la dirección en 2. |
| . | . | Repita tantas veces como sea necesario para llenar el búfer de escritura |
| . | . |
| . | . |
| 1111 | <MSB><LSB> | Escribe 2 bytes y comienza a programar |
| 0000 | 00 00 | NOP - Mantenga PGC alto para el tiempo P9 y bajo para el tiempo P10 |
| Para continuar modificando datos, repita los pasos 2 a 6, donde el puntero de dirección se incrementa en el número apropiado de bytes (ver Tabla 3-4) en cada iteración del bucle. El ciclo de escritura debe repetirse suficientes veces para reescribir completamente el contenido del búfer de borrado. | | |
| Paso 7: deshabilita las escrituras. | | |
| 0000 | 94 A6 | BCF EECON1, WREN |

**3.3 Programación de datos EEPROM**

|  |  |
| --- | --- |
| Nota: La programación de EEPROM de datos no está disponible en los siguientes dispositivos: | |
| PIC18F2410 | PIC18F4410 |
| PIC18F2450 | PIC18F4450 |
| PIC18F2510 | PIC18F4510 |
| PIC18F2515 | PIC18F4515 |
| PIC18F2610 | PIC18F4610 |

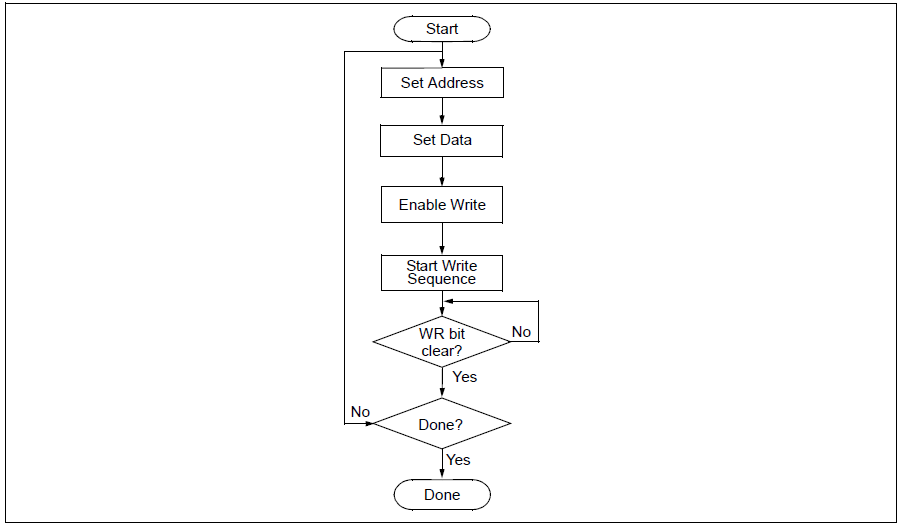
Se accede a la EEPROM de datos un byte a la vez a través de un puntero de dirección (par de registro: EEADRH: EEADR) y un pestillo de datos (EEDATA). Los datos EEPROM se escriben cargando EEADRH: EEADR con la ubicación de memoria deseada, EEDATA, con los datos que se escribirán e iniciando una escritura de memoria configurando adecuadamente el registro EECON1. Una escritura de byte borra automáticamente la ubicación y escribe los nuevos datos (borrar antes de escribir).

Cuando se utiliza el registro EECON1 para realizar una escritura EEPROM de datos, se deben borrar los bits EEPGD y CFGS (EECON1 <7: 6> = 00). El bit WREN debe establecerse (EECON1 <2> = 1) para habilitar escrituras de cualquier tipo y esto debe hacerse antes de iniciar una secuencia de escritura. La secuencia de escritura se inicia configurando el bit WR (EECON1 <1> = 1).

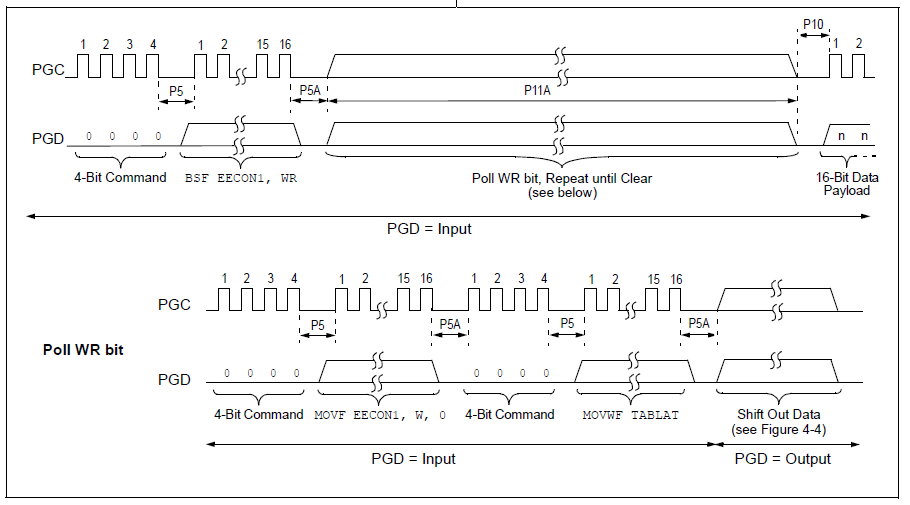
La escritura comienza en el flanco descendente del 4º PGC después de establecer el bit WR. Finaliza cuando el hardware borra el bit WR.

Después de que termina la secuencia de programación, PGC debe mantenerse bajo durante el tiempo especificado por el parámetro P10 para permitir la descarga de alto voltaje de la matriz de memoria.

**FIGURA 3-6: FLUJO DE DATOS DEL PROGRAMA**



**FIGURA 3-7: TIEMPO DE ESCRITURA DE EEPROM DE DATOS**



**TABLA 3-7: MEMORIA DE DATOS DE PROGRAMACIÓN**

|  |  |  |
| --- | --- | --- |
| Comando 4 bits | Carga de datos | Instrucción central |
| Paso 1: acceso directo a datos EEPROM | | |
| 0000 | 9E A6 | BCF EECON1, EEPGD |
| 0000 | 9C A6 | BCF EECON1, CFGS |
| Paso 2: establezca el puntero de dirección EEPROM de datos | | |
| 0000 | 0E <Addr> | MOVLW <Addr> |
| 0000 | 6E A9 | MOVWF EEADR |
| 0000 | OE <AddrH> | MOVLW <AddrH> |
| 0000 | 6E AA | MOVWF EEADRH |
| Paso 3: cargue los datos que se escribirán | | |
| 0000 | 0E <Data> | MOVLW <Data> |
| 0000 | 6E A8 | MOVWF EEDATA |
| Paso 4: Habilite las escrituras en memoria. | | |
| 0000 | 84 A6 | BSF EECON1, WREN |
| Paso 5: Inicia la escritura. | | |
| 0000 | 82 A6 | BSF EECON1, WR |
| Paso 6: sondear el bit WR, repetir hasta que el bit esté claro. | | |
| 0000 | 50 A6 | MOVF EECON1, W, 0 |
| 0000 | 6E F5 | MOVWF TABLAT |
| 0000 | 00 00 | NOP |
| 0000 | <MSB><LSB> | Desplazar datos (1) |
| Paso 7: Mantenga PGC bajo por el tiempo P10. | | |
| Paso 8: deshabilita las escrituras | | |
| 0000 | 94 A6 | BCF EECON1, WREN |
| Repita los pasos 2 a 8 para escribir más datos. | | |

Nota 1: Consulte la Figura 4-4 para obtener detalles sobre el cambio de temporización de datos.

**3.4 Programación de ubicación de ID**

Las ubicaciones de ID se programan de manera muy similar a la memoria de código. Los registros de ID se asignan en direcciones, 200000h a 200007h. Estas ubicaciones se leen normalmente incluso después de la protección del código.

**Nota: El usuario solo necesita completar los primeros 8 bytes del búfer de escritura para escribir las ubicaciones de ID.**

La Tabla 3-8 muestra la secuencia de código requerida para escribir las ubicaciones de ID.

Para modificar las ubicaciones de ID, consulte la metodología descrita en la Sección 3.2.1 "Modificación de la memoria de código". Al igual que con la memoria de código, las ubicaciones de ID deben borrarse antes de modificarse.

**TABLA 3-8: SECUENCIA DE IDENTIFICACIÓN DE ESCRITURA**

|  |  |  |
| --- | --- | --- |
| Comando 4 bits | Carga de datos | Instrucción central |
| Paso 1: acceso directo a la memoria de código y habilitar escrituras. | | |
| 0000 | 8E A6 | BSF EECON1, EEPGD |
| 0000 | 9C A6 | BCF EECON1, CFGS |
| Paso 2: Cargue el búfer de escritura con 8 bytes y escriba. | | |
| 0000 | 0E 20 | MOVLW 20h |
| 0000 | 6E F8 | MOVWF TBLPTRU |
| 0000 | 0E 00 | MOVLW 00h |
| 0000 | 6E F7 | MOVWF TBLPTRH |
| 0000 | 0E 00 | MOVLW 00h |
| 0000 | 6E F6 | MOVWF TBLPTRL |
| 1101 | <MSB><LSB> | Escribe 2 bytes y pos-incrementa la dirección en 2. |
| 1101 | <MSB><LSB> | Escribe 2 bytes y pos-incrementa la dirección en 2. |
| 1101 | <MSB><LSB> | Escribe 2 bytes y pos-incrementa la dirección en 2. |
| 1111 | <MSB><LSB> | Escribe 2 bytes y comienza a programar |
| 0000 | 00 00 | NOP - Mantenga PGC alto para el tiempo P9 y bajo para el tiempo P10. |

**3.5 Programación de bloque de arranque**

Se debe utilizar la secuencia de código detallada en la Tabla 3-5, excepto que la dirección utilizada en el "Paso 2" estará en el rango de 000000h a 0007FFh

**3.6 Programación de bits de configuración**

A diferencia de la memoria de código, los bits de configuración se programan un byte a la vez. Se usa el comando Table Write, Begin Programming de 4 bits ('1111'), pero solo se escribirán ocho bits de la siguiente carga útil de 16 bits. El LSB de la carga útil se escribirá en direcciones pares y el MSB se escribirá en direcciones impares. La secuencia de código para programar dos ubicaciones de configuración consecutivas se muestra en la Tabla 3-9.

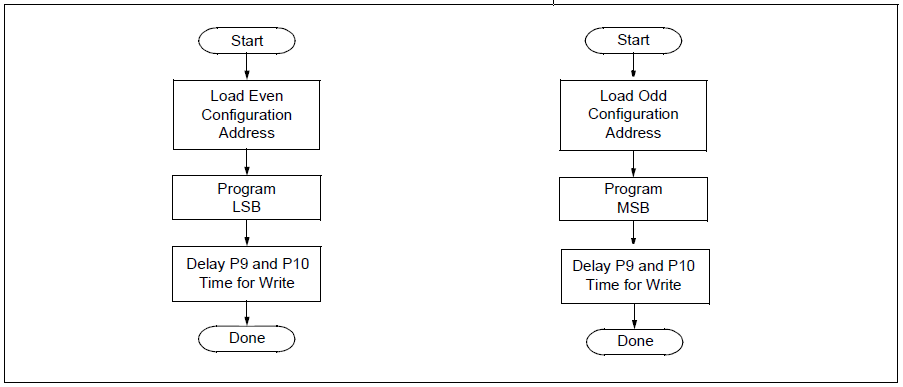
**Nota: La dirección debe escribirse explícitamente para cada byte programado. Las direcciones no se pueden incrementar en este modo.**

**TABLA 3-9: ESTABLECER EL PUNTERO DE DIRECCIÓN A LA UBICACIÓN DE CONFIGURACIÓN**

|  |  |  |
| --- | --- | --- |
| Comando 4 bits | Carga de datos | Instrucción central |
| Paso 1: habilite las escrituras y el acceso directo a la memoria de configuración | | |
| 0000 | 8E A6 | BSF EECON1, EEPGD |
| 0000 | 8C A6 | BSF EECON1, CFGS |
| Paso 2: Configure el puntero de tabla para que se escriba el byte de configuración. Escriba direcciones pares / impares. (1) | | |
| 0000 | 0E 30 | MOVLW 30h |
| 0000 | 6E F8 | MOVWF TBLPTRU |
| 0000 | 0E 00 | MOVLW 00h |
| 0000 | 6E F7 | MOVWF TBLPRTH |
| 0000 | 0E 00 | MOVLW 00h |
| 0000 | 6E F6 | MOVWF TBLPTRL |
| 1111 | <MSB ignorado><LSB> | Carga 2 bytes y comience a programar |
| 0000 | 00 00 | NOP - Mantenga PGC alto para el tiempo P9 y bajo para el tiempo P10. |
| 0000 | 0E 01 | MOVLW 01h |
| 0000 | 6E F6 | MOVWF TBLPTRL |
| 1111 | <MSB><LSB ignorado> | Carga 2 bytes y comience a programar |
| 0000 | 00 00 | NOP - Mantenga PGC alto para el tiempo P9 y bajo para el tiempo P10 |

**Nota 1: Habilitar la protección contra escritura de los bits de configuración (WRTC = 0 en CONFIG6H) evitará una mayor escritura de los bits de configuración. Siempre escriba todos los bits de configuración antes de habilitar la protección contra escritura para los bits de configuración.**

**FIGURA 3-8: FLUJO DE PROGRAMACIÓN DE CONFIGURACIÓN**



**4.0 LECTURA DEL DISPOSITIVO**

**4.1 Memoria de código de lectura, ubicaciones de ID y bits de configuración**

Se accede a la memoria de código, un byte a la vez, a través del comando de 4 bits, "1001" (Tabla Leída, post-incremento). Los contenidos de memoria señalados por el puntero de tabla (TBLPTRU: TBLPTRH: TBLPTRL) se emiten en serie en PGD.

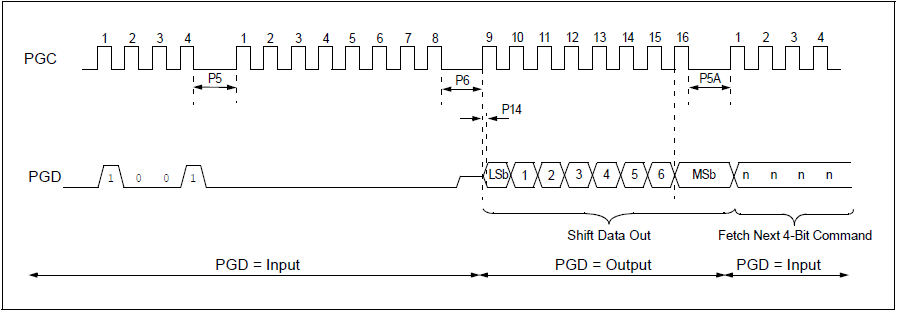
El comando de 4 bits se desplaza, LSb primero. La lectura se ejecuta durante los siguientes ocho relojes, luego se desplaza en PGD durante los últimos ocho relojes, LSb a MSb. Se debe introducir un retraso de P6 después del borde descendente del octavo PGC del operando para permitir que PGD pase de una entrada a una salida. Durante este tiempo, el PGC debe mantenerse bajo (consulte la Figura 4-1). Esta operación también incrementa el puntero de tabla en uno, apuntando al siguiente byte en la memoria de código para la próxima lectura.

Esta técnica funcionará para leer cualquier memoria en el espacio de direcciones 000000h a 3FFFFFh, por lo que también se aplica a la lectura de la ID y los registros de configuración

**TABLA 4-1: LEA LA SECUENCIA DE MEMORIA DEL CÓDIGO**

|  |  |  |
| --- | --- | --- |
| Comando 4 bits | Carga de datos | Instrucción central |
| Paso 1: Establecer el puntero de tabla. | | |
| 0000 | 0E <Addr[21:16]> | MOVLW Addr[21:16] |
| 0000 | 6E F8 | MOVWF TBLPTRU |
| 0000 | 0E <Addr[15:8]> | MOVLW <Addr[15:8]> |
| 0000 | 6E F7 | MOVWF TBLPTRH |
| 0000 | 0E <Addr[7:0]> | MOVLW <Addr[7:0]> |
| 0000 | 6E F6 | MOVWF TBLPTRL |
| Paso 2: Lea la memoria y luego cambie a PGD, LSb a MSb. | | |
| 1001 | 00 00 | TBLRD \*+ |

**FIGURA 4-1: TABLA LEA EL TIEMPO DE INSTRUCCIÓN POSTERIOR AL INCREMENTO (1001)**

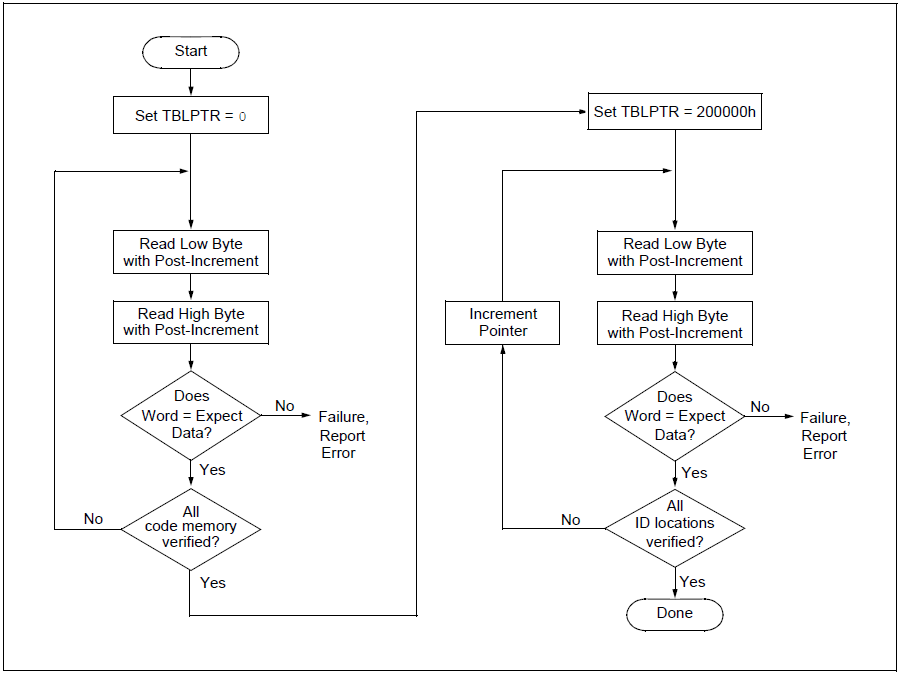


**4.2 Verificar memoria de código y ubicaciones de identificación**

El paso de verificación implica volver a leer el espacio de memoria del código y compararlo con la copia contenida en el búfer del programador. Las lecturas de memoria ocurren un solo byte a la vez, por lo que se deben leer dos bytes para comparar con la palabra en el búfer del programador. Consulte la Sección 4.1 “Memoria de código de lectura, ubicaciones de ID y bits de configuración” para obtener detalles de implementación de la memoria de código de lectura.

El puntero de tabla debe establecerse manualmente en 200000 h (dirección base de las ubicaciones de ID) una vez que se haya verificado la memoria de código. La función de incremento posterior del comando Table Read de 4 bits no se puede usar para incrementar el puntero de tabla más allá del espacio de memoria de código. En un dispositivo de 64 Kbytes, por ejemplo, una lectura de dirección de incremento posterior, FFFFh, ajustará el puntero de la tabla a 000000h, en lugar de apuntar a la dirección no implementada, 010000h.

**FIGURA 4-2: VERIFICAR EL FLUJO DE MEMORIA DEL CÓDIGO**



**4.3 Verificar bits de configuración**

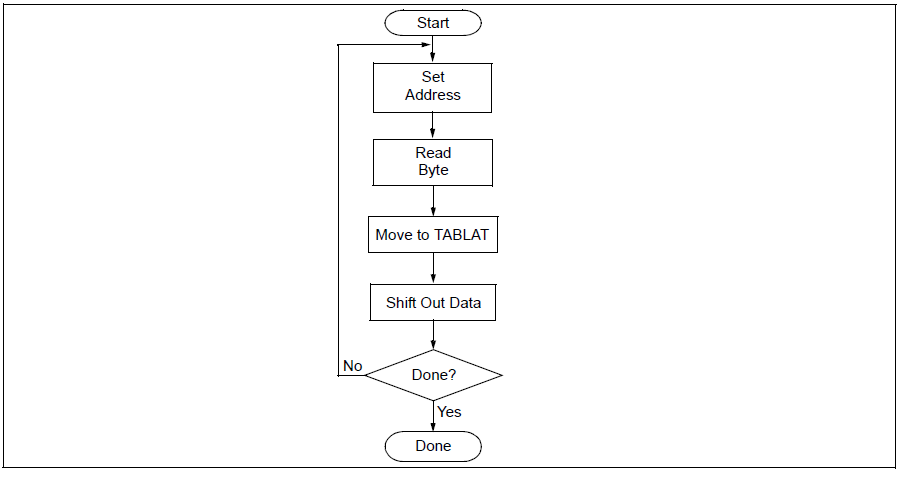
Se puede leer y emitir una dirección de configuración en PGD mediante el comando de 4 bits, "1001". Los datos de configuración se leen y escriben en forma de byte, por lo que no es necesario fusionar dos bytes en una palabra antes de una comparación. El resultado puede compararse inmediatamente con los datos de configuración apropiados en la memoria del programador para su verificación. Consulte la Sección 4.1 “Memoria de código de lectura, ubicaciones de ID y bits de configuración” para obtener detalles de implementación de lectura de datos de configuración.

**4.4 Leer datos de memoria EEPROM**

Se accede a la EEPROM de datos, un byte a la vez, a través de un puntero de dirección (par de registro: EEADRH: EEADR) y un pestillo de datos (EEDATA). Los datos EEPROM se leen cargando EEADRH: EEADR con la ubicación de memoria deseada e iniciando una lectura de memoria configurando adecuadamente el registro EECON1. Los datos se cargarán en EEDATA, donde se pueden emitir en serie en PGD a través del comando de 4 bits, "0010" (registro Shift Out Data Holding). Se debe introducir un retraso de P6 después del borde descendente del octavo PGC del operando para permitir que PGD pase de una entrada a una salida. Durante este tiempo, el PGC debe mantenerse bajo (consulte la Figura 4-4).

La secuencia de comandos para leer un solo byte de datos se muestra en la Tabla 4-2.

**FIGURA 4-3: LEER FLUJO DE EEPROM DE DATOS**

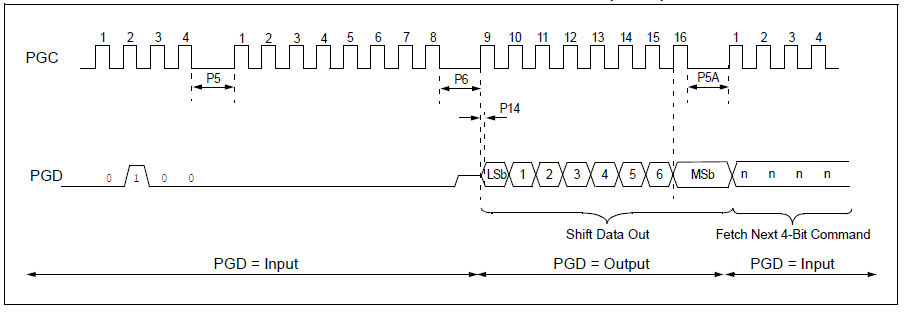


**TABLA 4-2: LEER DATOS MEMORIA EEPROM**

|  |  |  |
| --- | --- | --- |
| Comando 4 bits | Carga de datos | Instrucción central |
| Paso 1: acceso directo a datos EEPROM. | | |
| 0000 | 9E A6 | BCF EECON1, EEPGD |
| 0000 | 9C A6 | BCF EECON1, CFGS |
| Paso 2: Configure el puntero de dirección EEPROM de datos. | | |
| 0000 | 0E <Addr> | MOVLW <Addr> |
| 0000 | 6E A9 | MOVWF EEADR |
| 0000 | OE <AddrH> | MOVLW <AddrH> |
| 0000 | 6E AA | MOVWF EEADRH |
| Paso 3: iniciar una lectura de memoria. | | |
| 0000 | 80 A6 | BSF EECON1, RD |
| Paso 4: cargar datos en el registro de retención de datos en serie | | |
| 0000 | 50 A8 | MOVF EEDATA, W, 0 |
| 0000 | 6E F5 | MOVWF TABLAT |
| 0000 | 00 00 | NOP |
| 0010 | <MSB><LSB> | Shift Out Data**(1)** |

**Nota 1: <LSB> no está definido. El <MSB> son los datos.**

**FIGURA 4-4: CAMBIAR EL TIEMPO DE REGISTRO DE RETENCIÓN DE DATOS (0010)**



**4.5 Verificar datos EEPROM**

Una dirección EEPROM de datos puede leerse mediante una secuencia de instrucciones básicas (comando de 4 bits, "0000") y luego salir en PGD a través del comando de 4 bits, "0010" (registro TABLAT). El resultado puede compararse inmediatamente con los datos apropiados en la memoria del programador para su verificación. Consulte la Sección 4.4 “Leer la memoria EEPROM de datos” para obtener detalles sobre la implementación de la lectura de datos EEPROM.

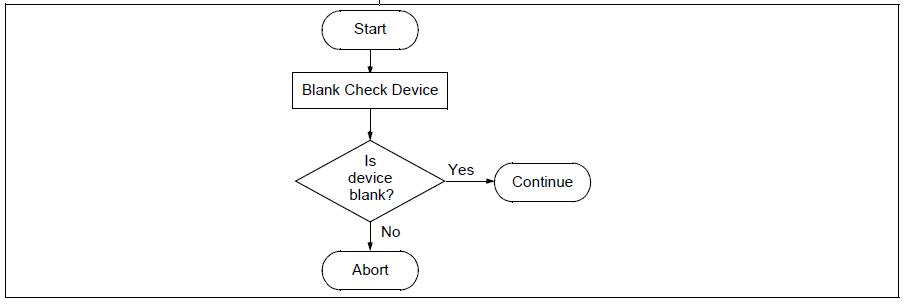
**4.6 Cheque en blanco**

El término Verificación en blanco significa verificar que el dispositivo no tiene celdas de memoria programadas. Todas las memorias deben ser verificadas: memoria de código, EEPROM de datos, ubicaciones de ID y bits de configuración. Los registros de ID de dispositivo (3FFFFEh: 3FFFFFh) deben ignorarse

Una celda de memoria "en blanco" o "borrada" se leerá como "1". Por lo tanto, la verificación en blanco de un dispositivo simplemente significa verificar que todos los bytes se leen como FFh, excepto los bits de configuración. Los bits de configuración no utilizados (reservados) leerán '0' (programado). Consulte la Figura 4-5 para ver los datos esperados de la configuración en blanco para los diversos dispositivos de la familia PIC18F2XXX / 4XXX.

Dado que la verificación en blanco es simplemente verificación de EEPROM de código y datos con datos esperados de FFh, consulte la Sección 4.4 “Leer la memoria EEPROM de datos” y la Sección 4.2 “Verificar la memoria de código y las ubicaciones de ID” para obtener detalles sobre la implementación.

**FIGURA 4-5: FLUJO DE CONTROL EN BLANCO**



**5.0 PALABRA DE CONFIGURACIÓN**

Los dispositivos de la familia PIC18F2XXX / 4XXX tienen varias palabras de configuración. Estos bits se pueden establecer o borrar para seleccionar varias configuraciones de dispositivo. Todas las demás áreas de memoria deben programarse y verificarse antes de configurar las palabras de configuración. Estos bits pueden leerse normalmente, incluso después de la protección de lectura o código. Consulte la Tabla 5-1 para obtener una lista de bits de configuración e ID de dispositivo, y la Tabla 5-3 para ver las descripciones de los bits de configuración.

**5.1 Ubicaciones de ID**

Un usuario puede almacenar información de identificación (ID) en ocho ubicaciones de ID, mapeadas en 200000h: 200007h. Se recomienda que el mordisco más significativo de cada ID sea Fh. Al hacerlo, si el código de usuario intenta inadvertidamente ejecutarse desde el espacio de ID, los datos de ID se ejecutarán como un NOP.

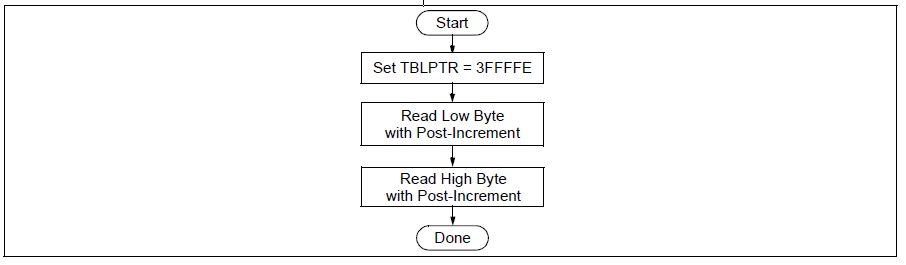
**5.2 Palabra de ID del dispositivo**

La palabra de ID de dispositivo para los dispositivos de la familia PIC18F2XXX / 4XXX se encuentra en 3FFFFEh: 3FFFFFh. El programador puede utilizar estos bits para identificar qué tipo de dispositivo se está programando y leer normalmente, incluso después de la protección de código o lectura

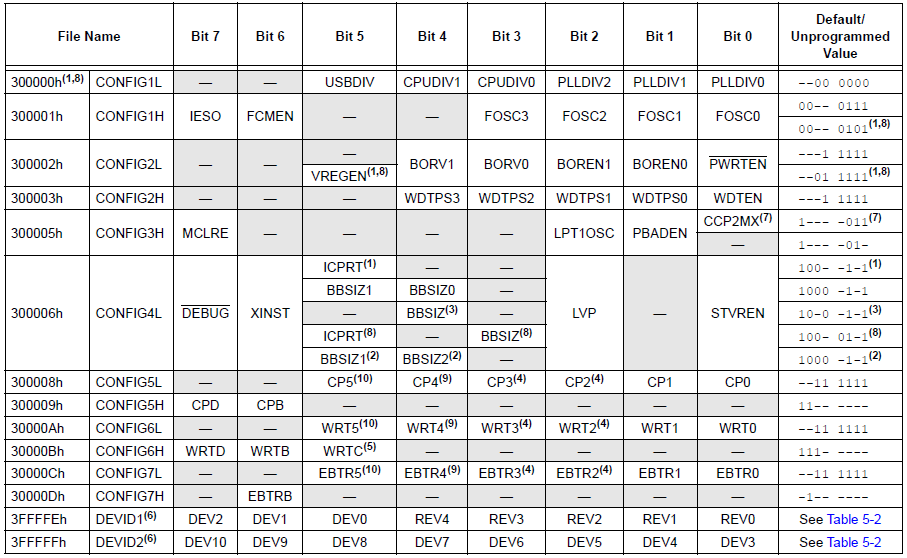
En algunos casos, los dispositivos pueden compartir los mismos valores DEVID. En tales casos, el bit más significativo de la revisión del dispositivo, REV4 (DEVID1 <4>), deberá examinarse para determinar completamente el dispositivo al que se accede.

Consulte la Tabla 5-2 para obtener una lista completa de los valores de ID de dispositivo.

**FIGURA 5-1: LEA EL FLUJO DE LA PALABRA DE ID DE DISPOSITIVO**



**TABLA 5-1: BITS DE CONFIGURACIÓN E ID DE DISPOSITIVO**



**Leyenda**: - = no implementado. Las celdas sombreadas no están implementadas, se leen como "0".

**Nota** 1: Implementado solo en dispositivos PIC18F2455 / 2550/4455/4550 y PIC18F2458 / 2553/4458/4553.

  2: Implementado solo en dispositivos PIC18F2585 / 2680/4585/4680, PIC18F2682 / 2685 y PIC18F4682 / 4685.

  3: Implementado solo en dispositivos PIC18F2480 / 2580/4480/4580.

  4: Estos bits solo se implementan en dispositivos específicos basados en la memoria disponible. Consulte la Sección 2.3 “Mapas de memoria”.

5: En los dispositivos PIC18F2480 / 2580/4480/4580, este bit es de solo lectura en modo de ejecución normal; solo se puede escribir en modo Programa.

6: Los registros DEVID son de solo lectura y el usuario no puede programarlos.

7: Implementado en todos los dispositivos con la excepción de los dispositivos PIC18FXX8X y PIC18F2450 / 4450.

8: Implementado solo en dispositivos PIC18F2450 / 4450.

9: Implementado solo en dispositivos PIC18F2682 / 2685 y PIC18F4682 / 4685.

10: Implementado solo en dispositivos PIC18F2685 / 4685

**TABLA 5-2: VALORES DE ID DE DISPOSITIVO**

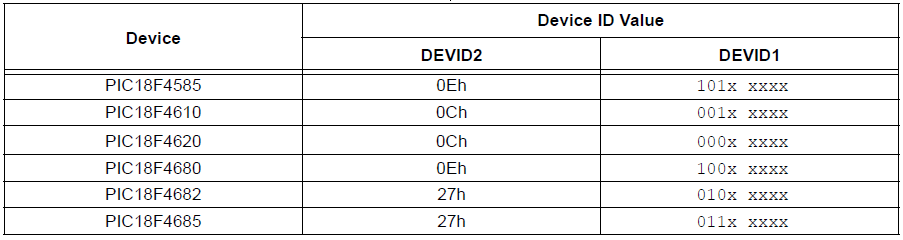
|  |  |  |
| --- | --- | --- |
| Dispositivo | Valor de ID del dispositivo | |
| **DEVID2** | **DEVID1** |
| PIC18F2221 | 21h | 011x xxxx |
| PIC18F2321 | 21h | 001x xxxx |
| PIC18F2410 | 11h | 011x xxxx |
| PIC18F2420 | 11h | 010x xxxx**(1)** |
| PIC18F2423 | 11h | 010x xxxx**(2)** |
| PIC18F2450 | 24h | 001x xxxx |
| PIC18F2455 | 12h | 011x xxxx |
| PIC18F2458 | 2Ah | 011x xxxx |
| PIC18F2480 | 1Ah | 111x xxxx |
| PIC18F2510 | 11h | 001x xxxx |
| PIC18F2515 | 0Ch | 111x xxxx |
| PIC18F2520 | 11h | 000x xxxx**(1)** |
| PIC18F2523 | 11h | 000x xxxx**(2)** |
| PIC18F2525 | 0Ch | 110x xxxx |
| PIC18F2550 | 12h | 010x xxxx |
| PIC18F2553 | 2Ah | 010x xxxx |
| PIC18F2580 | 1Ah | 110x xxxx |
| PIC18F2585 | 0Eh | 111x xxxx |
| PIC18F2610 | 0Ch | 101x xxxx |
| PIC18F2620 | 0Ch | 100x xxxx |
| PIC18F2680 | 0Eh | 110x xxxx |
| PIC18F2682 | 27h | 000x xxxx |
| PIC18F2685 | 27h | 001x xxxx |
| PIC18F4221 | 21h | 010x xxxx |
| PIC18F4321 | 21h | 000x xxxx |
| PIC18F4410 | 10h | 111x xxxx |
| PIC18F4420 | 10h | 110x xxxx**(1)** |
| PIC18F4423 | 10h | 110x xxxx**(2)** |
| PIC18F4450 | 24h | 000x xxxx |
| PIC18F4455 | 12h | 001x xxxx |
| PIC18F4458 | 2Ah | 001x xxxx |
| PIC18F4480 | 1Ah | 101x xxxx |
| PIC18F4510 | 10h | 101x xxxx |
| PIC18F4515 | 0Ch | 011x xxxx |
| PIC18F4520 | 10h | 100x xxxx**(1)** |
| PIC18F4523 | 10h | 100x xxxx**(2)** |
| PIC18F4525 | 0Ch | 010x xxxx |
| PIC18F4550 | 12h | 000x xxxx |
| PIC18F4553 | 2Ah | 000x xxxx |
| PIC18F4580 | 1Ah | 100x xxxx |

**Leyenda: las 'x' en DEVID1 contienen el código de revisión del dispositivo**

**Nota 1: DEVID1 bit 4 se utiliza para determinar el tipo de dispositivo (REV4 = 0).**

**2: DEVID1 bit 4 se utiliza para determinar el tipo de dispositivo (REV4 = 1).**

**TABLA 5-2: VALORES DE ID DE DISPOSITIVO (CONTINUACIÓN)**



**Leyenda: las 'x' en DEVID1 contienen el código de revisión del dispositivo**

**Nota 1: DEVID1 bit 4 se utiliza para determinar el tipo de dispositivo (REV4 = 0).**

**2: DEVID1 bit 4 se utiliza para determinar el tipo de dispositivo (REV4 = 1).**

**TABLA 5-3: PIC18F2XXX / 4XXX DESCRIPCIONES DE BITS FAMILIARES**

|  |  |  |
| --- | --- | --- |
| Nombre del Bit | Palabras de configuración | Descripción |
| IESO | CONFIG1H | Bit Interno Interno Externo  1 = El modo de cambio interno externo está habilitado  0 = El modo de cambio interno externo está deshabilitado |
| FCMEN | CONFIG1H | Bit de habilitación de monitor de reloj a prueba de fallos  1 = El monitor de reloj a prueba de fallas está habilitado  0 = El monitor de reloj a prueba de fallas está deshabilitado |
| FOSC<3:0> | CONFIG1H | Bits de selección de oscilador  11xx = Oscilador RC externo, función CLKO en RA6  101x = oscilador RC externo, función CLKO en RA6  1001 = Oscilador interno RC, función CLKO en RA6, función de puerto en RA7  1000 = Oscilador interno RC, función de puerto en RA6, función de puerto en RA7  0111 = Oscilador externo RC, función de puerto en RA6  0110 = oscilador HS, PLL está habilitado (frecuencia de reloj = 4 x FOSC1)  0101 = oscilador EC, función de puerto en RA6  0100 = oscilador EC, función CLKO en RA6  0011 = Oscilador RC externo, función CLKO en RA6  0010 = oscilador HS  0001 = oscilador XT  0000 = oscilador LP |
| FOSC<3:0> | CONFIG1H | Bits de selección de oscilador  (PIC18F2455 / 2550/4455/4550, PIC18F2458 / 2553/4458/4553 y Solo dispositivos PIC18F2450 / 4450)  111x = oscilador HS, PLL está habilitado, HS es utilizado por USB  110x = oscilador HS, HS es utilizado por USB  1011 = Oscilador interno, HS es utilizado por USB  1010 = Oscilador interno, XT es utilizado por USB  1001 = Oscilador interno, función CLKO en RA6, EC es utilizado por USB  1000 = Oscilador interno, función de puerto en RA6, EC es utilizado por USB  0111 = EC oscilador, PLL está habilitado, función CLKO en RA6, EC es usado por USB  0110 = EC oscilador, PLL está habilitado, función de puerto en RA6, EC es usado por USB  0101 = oscilador EC, función CLKO en RA6, EC es usado por USB  0100 = oscilador EC, función de puerto en RA6, EC es usado por USB  001x = oscilador XT, PLL está habilitado, XT es utilizado por USB  000x = oscilador XT, XT es utilizado por USB |
| USBDIV | CONFIG1L | Bit de selección de reloj USB  (PIC18F2455 / 2550/4455/4550, PIC18F2458 / 2553/4458/4553 y Solo dispositivos PIC18F2450 / 4450)  Selecciona la fuente del reloj para la operación USB a toda velocidad:  1 = la fuente del reloj USB proviene del PLL de 96 MHz dividido por 2  0 = la fuente del reloj USB proviene directamente del bloque oscilador OSC1 / OSC2; sin división |
| CPUDIV<1:0> | CONFIG1L | Bits de selección de reloj del sistema de la CPU  (PIC18F2455 / 2550/4455/4550, PIC18F2458 / 2553/4458/4553 y Solo dispositivos PIC18F2450 / 4450)  11 = reloj del sistema de la CPU dividido por 4  10 = reloj del sistema de la CPU dividido por 3  01 = reloj del sistema de la CPU dividido por 2  00 = Sin división del reloj del sistema de la CPU |

**Nota 1: Los bits BBSIZ, BBSIZ <1: 0> y BBSIZ <2: 1>, no se pueden cambiar una vez que se habilita cualquiera de los siguientes bits de protección de código: CPB o CP0, WRTB o WRT0, EBTRB o EBTR0**

**2: No disponible en dispositivos PIC18FXX8X y PIC18F2450 / 4450**

**TABLA 5-3: PIC18F2XXX / 4XXX DESCRIPCIONES DE BITS FAMILIARES (CONTINUACIÓN)**

|  |  |  |
| --- | --- | --- |
| Nombre del Bit | Palabras de configuración | Descripción |
| PLLDIV<2:0> | CONFIG1L | Bits de selección de oscilador   (PIC18F2455 / 2550/4455/4550, PIC18F2458 / 2553/4458/4553 y Solo dispositivos PIC18F2450 / 4450)  El divisor debe seleccionarse para proporcionar una entrada de 4 MHz en el PLL de 96 MHz:  111 = Oscilador dividido por 12 (entrada de 48 MHz)  110 = Oscilador dividido por 10 (entrada de 40 MHz)  101 = Oscilador dividido por 6 (entrada de 24 MHz)  100 = Oscilador dividido por 5 (entrada de 20 MHz)  011 = Oscilador dividido por 4 (entrada de 16 MHz)  010 = Oscilador dividido por 3 (entrada de 12 MHz)  001 = Oscilador dividido por 2 (entrada de 8 MHz)  000 = Sin división - oscilador utilizado directamente (entrada de 4 MHz) |
| VREGEN | CONFIG2L | Bit de habilitación del regulador de voltaje USB  (Solo dispositivos PIC18F2455 / 2550/4455/4550, PIC18F2458 / 2553/4458/4553 y PIC18F2450 / 4450)  1 = el regulador de voltaje USB está habilitado  0 = el regulador de voltaje USB está desactivado |
| BORV<1:0> | CONFIG2L | Bits de voltaje de restablecimiento de disminución de voltaje  11 = VBOR está configurado a 2.0V  10 = VBOR está configurado en 2.7V  01 = VBOR está configurado en 4.2V  00 = VBOR está configurado en 4.5V |
| BOREN<1:0> | CONFIG2L | Bits de habilitación de restablecimiento de disminución (reset)  11 = El restablecimiento de apagado está habilitado solo en hardware (SBOREN está deshabilitado)  10 = El restablecimiento de apagado está habilitado solo en hardware y deshabilitado en el modo de suspensión SBOREN está deshabilitado  01 = El restablecimiento de desvanecimiento está habilitado y controlado por software (SBOREN está habilitado)  00 = Restablecimiento de deshabilitación está deshabilitado en hardware y software |
| PWRTEN | CONFIG2L | Bit de activación del temporizador de encendido  1 = PWRT está deshabilitado  0 = PWRT está habilitado |
| WDPS<3:0> | CONFIG2H | Watchdog Timer Postscaler Seleccionar bits  1111 = 1: 32,768  1110 = 1: 16,384  1101 = 1: 8.192  1100 = 1: 4.096  1011 = 1: 2.048  1010 = 1: 1.024  1001 = 1: 512  1000 = 1: 256  0111 = 1: 128  0110 = 1:64  0101 = 1:32  0100 = 1:16  0011 = 1: 8  0010 = 1: 4  0001 = 1: 2  0000 = 1: 1 |

**Nota 1: Los bits BBSIZ, BBSIZ <1: 0> y BBSIZ <2: 1>, no se pueden cambiar una vez que se habilita cualquiera de los siguientes bits de protección de código: CPB o CP0, WRTB o WRT0, EBTRB o EBTR0.**

**2: No disponible en dispositivos PIC18FXX8X y PIC18F2450 / 4450.**

**TABLA 5-3: PIC18F2XXX / 4XXX DESCRIPCIONES DE BITS FAMILIARES (CONTINUACIÓN)**

|  |  |  |
| --- | --- | --- |
| Nombre del Bit | Palabras de configuración | Descripción |
| WDTEN | CONFIG2H | Bit de activación del temporizador del perro guardián  1 = WDT está habilitado  0 = WDT está deshabilitado (el control se coloca en el bit SWDTEN) |
| MCLRE | CONFIG3H | Bit de habilitación de pin MCLR  1 = el pin MCLR está habilitado, el pin de entrada RE3 está deshabilitado  0 = el pin de entrada RE3 está habilitado, el pin MCLR está deshabilitado |
| LPT1OSC | CONFIG3H | Bit de activación del oscilador del temporizador de baja potencia 1  1 = Timer1 está configurado para operación de baja potencia  0 = Timer1 está configurado para operación de alta potencia |
| PBADEN | CONFIG3H | PORTB A / D Habilitar bit  1 = los pines PORTB A / D <4: 0> están configurados como canales de entrada analógica en Reset  0 = los pines PORTB A / D <4: 0> están configurados como E / S digitales al reiniciar |
| PBADEN | CONFIG3H | Bit de habilitación A / D PORTB (solo dispositivos PIC18FXX8X)  1 = los pines PORTB A / D <4: 0> y PORTB A / D <1: 0> se configuran como canales de entrada analógica en Reset  0 = los pines PORTB A / D <4: 0> están configurados como E / S digitales al reiniciar |
| CCP2MX | CONFIG3H | Bit CCP2 MUX  1 = La entrada / salida CCP2 se multiplexa con RC1 (2)  0 = la entrada / salida CCP2 se multiplexa con RB3 |
| \_\_\_\_\_\_\_  DEBUG | CONFIG4L | Bit de habilitación del depurador en segundo plano  1 = El depurador en segundo plano está deshabilitado, RB6 y RB7 están configurados como pines de E / S de uso general  0 = El depurador en segundo plano está habilitado, RB6 y RB7 están dedicados a la depuración en circuito |
| XINST | CONFIG4L | Bit de habilitación de conjunto de instrucciones extendido  1 = La extensión del conjunto de instrucciones y el modo de direccionamiento indexado están habilitados  0 = La extensión del conjunto de instrucciones y el modo de direccionamiento indexado están deshabilitados (modo heredado) |
| ICPRT | CONFIG4L | Bit de habilitación de puerto dedicado en circuito (ICD / ICSP ™) (solo dispositivos PIC18F2455 / 2550/4455/4550, PIC18F2458 / 2553/4458/4553 y PIC18F2450 / 4450)  1 = ICPORT está habilitado  0 = ICPORT está deshabilitado |
| BBSIZ<1:0>**(1)** | CONFIG4L | Bits de selección de tamaño de bloque de arranque (solo dispositivos PIC18F2585 / 2680/4585/4680)  11 = bloque de arranque de palabras de 4K (8 Kbytes)  10 = bloque de arranque de palabras de 4K (8 Kbytes)  01 = 2K palabras (4 Kbytes) Bloque de arranque  00 = bloque de arranque de 1K palabra (2 Kbytes) |
| BBSIZ<2:1>**(1)** | CONFIG4L | Bits de selección de tamaño de bloque de arranque (solo dispositivos PIC18F2682 / 2685/4582/4685)  11 = bloque de arranque de palabras de 4K (8 Kbytes)  10 = bloque de arranque de palabras de 4K (8 Kbytes)  01 = 2K palabras (4 Kbytes) Bloque de arranque  00 = bloque de arranque de 1K palabra (2 Kbytes) |

Nota 1: Los bits BBSIZ, BBSIZ <1: 0> y BBSIZ <2: 1>, no se pueden cambiar una vez que se habilita cualquiera de los siguientes bits de protección de código: CPB o CP0, WRTB o WRT0, EBTRB o EBTR0.

2: No disponible en dispositivos PIC18FXX8X y PIC18F2450 / 4450.

**TABLA 5-3: PIC18F2XXX / 4XXX DESCRIPCIONES DE BITS FAMILIARES (CONTINUACIÓN)**

|  |  |  |
| --- | --- | --- |
| Nombre del Bit | Palabras de configuración | Descripción |
| BBSIZ<1:0>**(1)** | CONFIG4L | Bits de selección de tamaño de bloque de arranque (solo dispositivos PIC18F2321 / 4321)  11 = bloque de arranque de 1K palabra (2 Kbytes)  10 = bloque de arranque de 1K palabra (2 Kbytes)  01 = 512 palabras (1 Kbyte) Bloque de arranque  00 = 256 palabras (512 bytes) Bloque de arranque  Bits de selección de tamaño de bloque de arranque (solo dispositivos PIC18F2221 / 4221)  11 = 512 palabras (1 Kbyte) Bloque de arranque  10 = 512 palabras (1 Kbyte) Bloque de arranque  01 = 512 palabras (1 Kbyte) Bloque de arranque  00 = 256 palabras (512 bytes) Bloque de arranque |
| BBSIZ**(1)** | CONFIG4L | Bits de selección de tamaño de bloque de arranque (solo dispositivos PIC18F2480 / 2580/4480/4580 y PIC18F2450 / 4450)  1 = bloque de arranque de 2K palabras (4 Kbytes)  0 = bloque de arranque de 1K palabra (2 Kbytes) |
| LVP | CONFIG4L | Bit de habilitación de programación de bajo voltaje  1 = La programación de bajo voltaje está habilitada, RB5 es el pin PGM  0 = La programación de bajo voltaje está desactivada, RB5 es un pin de E / S |
| STVREN | CONFIG4L | Bit de habilitación de reinicio de desbordamiento / desbordamiento de pila  1 = Restablecer en desbordamiento / desbordamiento de pila está habilitado  0 = Restablecer en desbordamiento / desbordamiento de pila deshabilitado |
| CP5 | CONFIG5L | Bit de protección de código (área de memoria de código del bloque 5)  (Solo dispositivos PIC18F2685 y PIC18F4685)  1 = El bloque 5 no está protegido por código  0 = El bloque 5 está protegido por código |
| CP4 | CONFIG5L | Bit de protección de código (área de memoria de código del bloque 4)  (Solo dispositivos PIC18F2682 / 2685 y PIC18F4682 / 4685)  1 = El bloque 4 no está protegido por código  0 = el bloque 4 está protegido por código |
| CP3 | CONFIG5L | Bit de protección de código (área de memoria de código del bloque 3)  1 = El bloque 3 no está protegido por código  0 = el bloque 3 está protegido por código |
| CP2 | CONFIG5L | Bit de protección de código (área de memoria de código del bloque 2)  1 = El bloque 2 no está protegido por código  0 = el bloque 2 está protegido por código |
| CP1 | CONFIG5L | Bit de protección de código (área de memoria de código del bloque 1)  1 = El bloque 1 no está protegido por código  0 = El bloque 1 está protegido por código |
| CP0 | CONFIG5L | Bit de protección de código (área de memoria de código del bloque 0)  1 = El bloque 0 no está protegido por código  0 = el bloque 0 está protegido por código |
| CPD | CONFIG5H | Bit de protección de código (EEPROM de datos)  1 = EEPROM de datos no está protegido por código  0 = EEPROM de datos está protegido por código |
| CPB | CONFIG5H | Bit de protección de código (área de memoria del bloque de arranque)  1 = El bloque de arranque no está protegido por código  0 = Boot Block está protegido por código |

Nota 1: Los bits BBSIZ, BBSIZ <1: 0> y BBSIZ <2: 1>, no se pueden cambiar una vez que se habilita cualquiera de los siguientes bits de protección de código: CPB o CP0, WRTB o WRT0, EBTRB o EBTR0.

2: No disponible en dispositivos PIC18FXX8X y PIC18F2450 / 4450

**TABLA 5-3: PIC18F2XXX / 4XXX DESCRIPCIONES DE BITS FAMILIARES (CONTINUACIÓN)**

|  |  |  |
| --- | --- | --- |
| Nombre del Bit | Palabras de configuración | Descripción |
| WRT5 | CONFIG6L | Bit de protección contra escritura (área de memoria de código del bloque 5) (solo dispositivos PIC18F2685 y PIC18F4685)  1 = El bloque 5 no está protegido contra escritura  0 = el bloque 5 está protegido contra escritura |
| WRT4 | CONFIG6L | Bit de protección contra escritura (área de memoria de código del bloque 4) (solo dispositivos PIC18F2682 / 2685 y PIC18F4682 / 4685)  1 = El bloque 4 no está protegido contra escritura  0 = el bloque 4 está protegido contra escritura |
| WRT3 | CONFIG6L | Bit de protección contra escritura (área de memoria de código del bloque 3)  1 = El bloque 3 no está protegido contra escritura  0 = el bloque 3 está protegido contra escritura |
| WRT2 | CONFIG6L | Bit de protección contra escritura (área de memoria de código del bloque 2)  1 = El bloque 2 no está protegido contra escritura  0 = el bloque 2 está protegido contra escritura |
| WRT1 | CONFIG6L | Bit de protección contra escritura (área de memoria del código del bloque 1)  1 = El bloque 1 no está protegido contra escritura  0 = el bloque 1 está protegido contra escritura |
| WRT0 | CONFIG6L | Bit de protección contra escritura (área de memoria del código del bloque 0)  1 = El bloque 0 no está protegido contra escritura  0 = el bloque 0 está protegido contra escritura |
| WRTD | CONFIG6H | Bit de protección contra escritura (EEPROM de datos)  1 = EEPROM de datos no está protegido contra escritura  0 = EEPROM de datos está protegido contra escritura |
| WRTB | CONFIG6H | Bit de protección contra escritura (área de memoria del bloque de arranque)  1 = El bloque de arranque no está protegido contra escritura  0 = El bloque de arranque está protegido contra escritura |
| WRTC | CONFIG6H | Bit de protección contra escritura (registros de configuración)  1 = Los registros de configuración no están protegidos contra escritura  0 = Los registros de configuración están protegidos contra escritura |
| EBTR5 | CONFIG7L | Bit de protección de lectura de tabla (área de memoria de código del bloque 5) (solo dispositivos PIC18F2685 y PIC18F4685)  1 = El bloque 5 no está protegido de las lecturas de tabla ejecutadas en otros bloques  0 = el bloque 5 está protegido de las lecturas de tabla ejecutadas en otros bloques |
| EBTR4 | CONFIG7L | Bit de protección de lectura de tabla (área de memoria de código del bloque 4)  (Solo dispositivos PIC18F2682 / 2685 y PIC18F4682 / 4685)  1 = El bloque 4 no está protegido de las lecturas de tabla ejecutadas en otros bloques  0 = el bloque 4 está protegido de las lecturas de tabla ejecutadas en otros bloques |
| EBTR3 | CONFIG7L | Bit de protección de lectura de tabla (área de memoria de código del bloque 3)  1 = el bloque 3 no está protegido de las lecturas de tabla ejecutadas en otros bloques  0 = el bloque 3 está protegido de las lecturas de tabla ejecutadas en otros bloques |
| EBTR2 | CONFIG7L | Bit de protección de lectura de tabla (área de memoria de código del bloque 2)  1 = El bloque 2 no está protegido de las lecturas de tabla ejecutadas en otros bloques  0 = el bloque 2 está protegido de las lecturas de tabla ejecutadas en otros bloques |
| EBTR1 | CONFIG7L | Bit de protección de lectura de tabla (área de memoria de código del bloque 1)  1 = El bloque 1 no está protegido de las lecturas de tabla ejecutadas en otros bloques  0 = El bloque 1 está protegido de las lecturas de tabla ejecutadas en otros bloques |

**Nota 1: Los bits BBSIZ, BBSIZ <1: 0> y BBSIZ <2: 1>, no se pueden cambiar una vez que se habilita cualquiera de los siguientes bits de protección de código: CPB o CP0, WRTB o WRT0, EBTRB o EBTR0.**

**2: No disponible en dispositivos PIC18FXX8X y PIC18F2450 / 4450**

**TABLA 5-3: PIC18F2XXX / 4XXX DESCRIPCIONES DE BITS FAMILIARES (CONTINUACIÓN)**

|  |  |  |
| --- | --- | --- |
| Nombre del Bit | Palabras de configuración | Descripción |
| EBTR0 | CONFIG7L | Bit de protección de lectura de tabla (área de memoria del código del bloque 0)  1 = El bloque 0 no está protegido de las lecturas de tabla ejecutadas en otros bloques  0 = El bloque 0 está protegido de las lecturas de tabla ejecutadas en otros bloques |
| EBTRB | CONFIG7H | Bit de protección de lectura de tabla (área de memoria del bloque de arranque)  1 = El bloque de arranque no está protegido de las lecturas de tabla ejecutadas en otros bloques  0 = El bloque de arranque está protegido de las lecturas de tabla ejecutadas en otros bloques |
| DEV<10:3> | DEVID2 | Bits de ID del dispositivo  Estos bits se utilizan con los bits DEV <2: 0> en el registro DEVID1 para identificar  número de pieza |
| DEV<2:0> | DEVID1 | Bits de ID del dispositivo  Estos bits se utilizan con los bits DEV <10: 3> en el registro DEVID2 para identificar  número de pieza |
| REV<4:0> | DEVID1 | Bits de ID de revisión  Estos bits se utilizan para indicar la revisión del dispositivo. El bit REV4 es   a veces se usa para especificar completamente el tipo de dispositivo. |

**Nota 1: Los bits BBSIZ, BBSIZ <1: 0> y BBSIZ <2: 1>, no se pueden cambiar una vez que se habilita cualquiera de los siguientes bits de protección de código: CPB o CP0, WRTB o WRT0, EBTRB o EBTR0.**

**2: No disponible en dispositivos PIC18FXX8X y PIC18F2450 / 4450.**

**5.3 Programación de ICSP de suministro único**

El bit LVP en el registro de configuración, CONFIG4L, permite la programación ICSP de suministro único (bajo voltaje). El bit LVP tiene por defecto un '1' (habilitado) de fábrica.

Si no se utiliza el modo de Programación de suministro único, el bit LVP se puede programar a "0" y RB5 / PGM se convierte en un pin de E / S digital. Sin embargo, el bit LVP solo se puede programar entrando en el modo ICSP de alto voltaje, donde MCLR / VPP / RE3 se eleva a VIHH. Una vez que el bit LVP se programa en "0", solo el modo ICSP de alto voltaje está disponible y solo se puede usar el modo ICSP de alto voltaje para programar el dispositivo

**Nota 1: El modo ICSP de alto voltaje siempre está disponible, independientemente del estado del bit LVP, aplicando VIHH al pin MCLR / VPP / RE3.**

**2: Mientras está en modo ICSP de bajo voltaje, el pin RB5 ya no se puede usar como E / S de propósito general.**

**5.4 Incrustar información de la palabra de configuración en el archivo HEX**

Para permitir la portabilidad del código, se requiere un programador de la familia PIC18F2XXX / 4XXX para leer las ubicaciones de Word de configuración del archivo hexadecimal. Si la información de la Palabra de configuración no está presente en el archivo hexadecimal, se debe emitir un simple mensaje de advertencia. Del mismo modo, al guardar un archivo hexadecimal, se debe incluir toda la información de Word de configuración. Se puede proporcionar una opción para no incluir la información de la Palabra de configuración. Al incrustar información de Word de configuración en el archivo hexadecimal, debe comenzar en la dirección, 300000h.

Microchip Technology Inc. cree firmemente que esta característica es importante para el beneficio del cliente final.

**5.5 Incrustar información EEPROM de datos en el archivo HEX**

Para permitir la portabilidad del código, se requiere un programador de la familia PIC18F2XXX / 4XXX para leer la información EEPROM de datos del archivo hexadecimal. Si la información EEPROM de datos no está presente, se debe emitir un simple mensaje de advertencia. Del mismo modo, al guardar un archivo hexadecimal, se debe incluir toda la información EEPROM de datos. Se puede proporcionar una opción para no incluir la información EEPROM de datos. Al incrustar información EEPROM de datos en el archivo hexadecimal, debe comenzar en la dirección, F00000h.

Microchip Technology Inc. cree que esta característica es importante para el beneficio del cliente final.

**5.6 Cálculo de suma de verificación**

La suma de comprobación se calcula sumando lo siguiente

• El contenido de todas las ubicaciones de memoria de código

• Las palabras de configuración, debidamente enmascaradas

• Ubicaciones de ID (si algún bloque está protegido por código)

Los 16 bits menos significativos de esta suma son la suma de verificación. No se utilizan los contenidos de la EEPROM de datos.

5.6.1 MEMORIA DEL PROGRAMA

Cuando se suman los contenidos de la memoria del programa, cada palabra de 16 bits se agrega a la suma de verificación. Los contenidos de la memoria del programa, desde 000000h hasta el final del último bloque de memoria del programa, se utilizan para este cálculo. Los desbordamientos del bit 15 pueden ignorarse.

5.6.2 PALABRAS DE CONFIGURACIÓN

Para los cálculos de suma de verificación, los bits no implementados en las palabras de configuración deben ignorarse ya que dichos bits siempre se leen como '1'. Cada palabra de configuración de 8 bits se AND con una máscara correspondiente para evitar que los bits no utilizados afecten los cálculos de suma de verificación.

La máscara contiene un "0" en posiciones de bits sin implementar, o un "1" donde se puede elegir. Cuando AND con el valor leído de una palabra de configuración, solo quedan los bits implementados. En la Tabla 5-5 se proporciona una lista de máscaras adecuadas.

5.6.3 UBICACIONES DE ID

Normalmente, el contenido de estas ubicaciones está definido por el usuario, pero MPLAB® IDE ofrece la opción de escribir la suma de verificación de 16 bits desprotegida del dispositivo en los 16 bits más significativos de las ubicaciones de identificación (consulte el menú Configurar MPLAB IDE / Memoria de ID ") . Los 16 bits inferiores no se usan y permanecen claros. Esta es la suma de todos los contenidos de la memoria del programa y las palabras de configuración (enmascaradas adecuadamente) antes de habilitar cualquier protección de código.

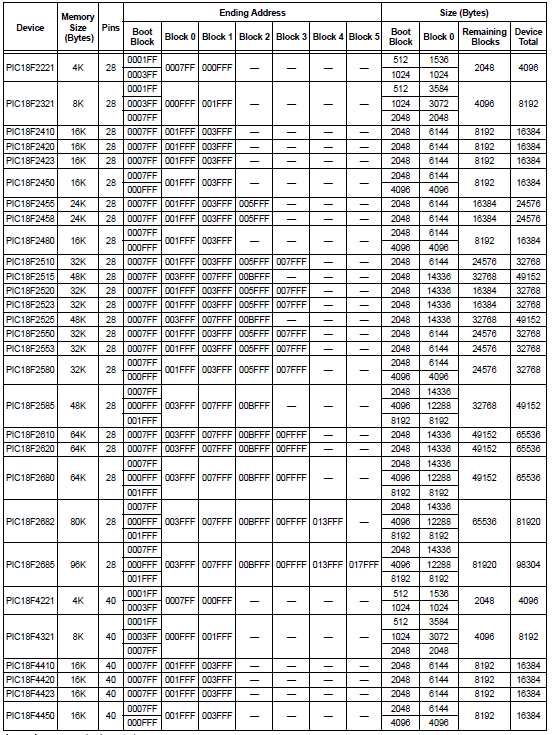
Si el usuario elige definir el contenido de las ubicaciones de ID, no se sabrá nada sobre los bloques protegidos. Si el usuario usa la suma de verificación preprotegida, proporcionada por MPLAB IDE, se proporciona una característica indirecta del código programado

5.6.4 PROTECCIÓN DEL CÓDIGO

Los bloques que están protegidos por código se vuelven a leer como todos '0' y no tienen efecto en los cálculos de suma de verificación. Si algún bloque está protegido por código, el contenido de las ubicaciones de ID se incluye en el cálculo de la suma de verificación

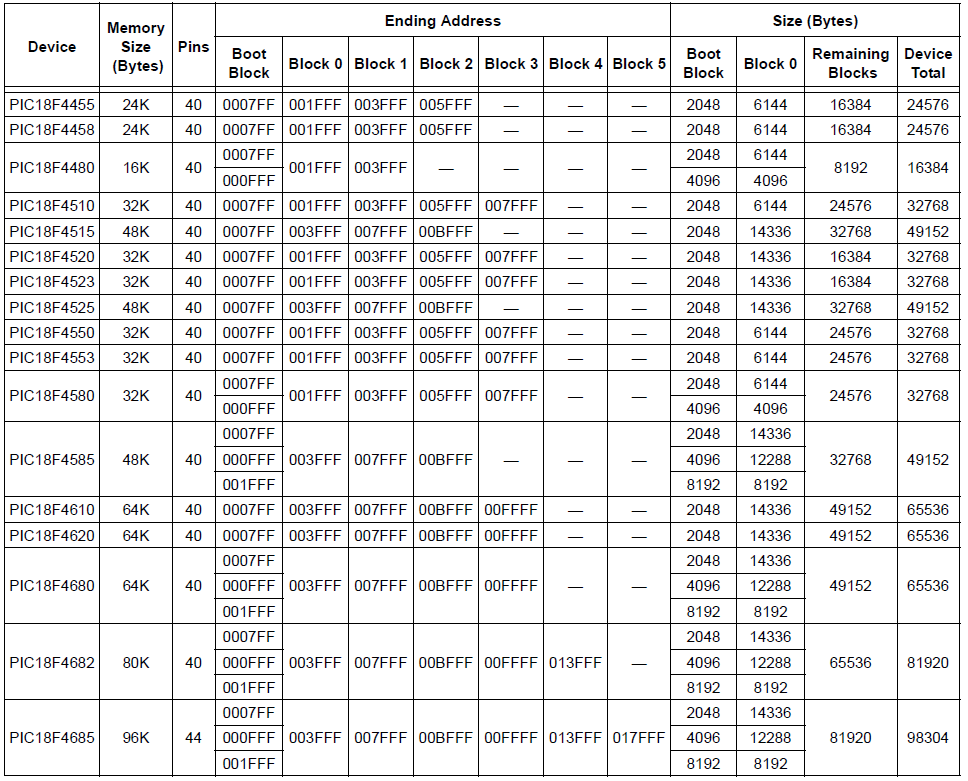
Todas las palabras de configuración y las ubicaciones de ID siempre se pueden leer normalmente, incluso cuando el dispositivo está completamente protegido por código. La comprobación de la configuración de protección de código en las palabras de configuración puede indicar cuál de los bloques de memoria del programa, si lo hay, puede leerse, y si las ubicaciones de ID deben usarse para los cálculos de suma de comprobación

**TABLA 5-4: UBICACIONES Y TAMAÑOS DE BLOQUES DE DISPOSITIVOS**



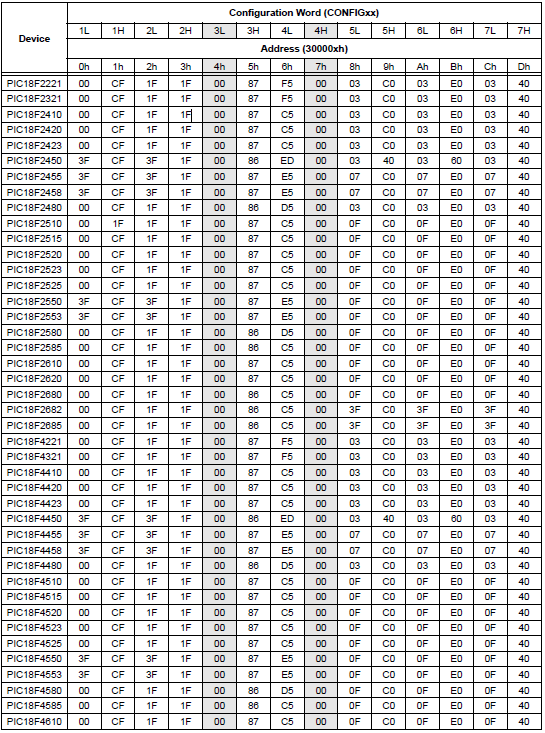
**Leyenda: - = no implementado.**

**TABLA 5-4: UBICACIONES Y TAMAÑOS DE BLOQUES DE DISPOSITIVOS (CONTINUACIÓN)**



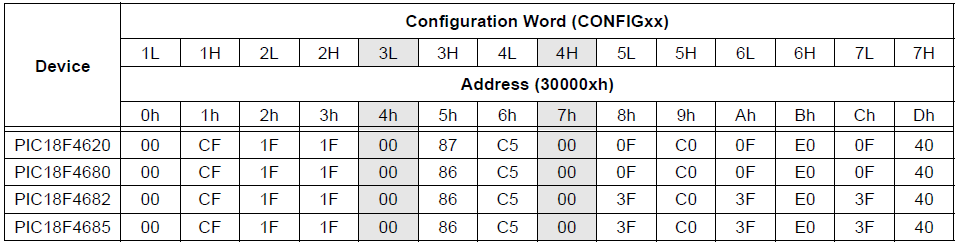
**Leyenda: - = no implementado.**

**TABLE 5-5: CONFIGURATION WORD MASKS FOR COMPUTING CHECKSUMS**



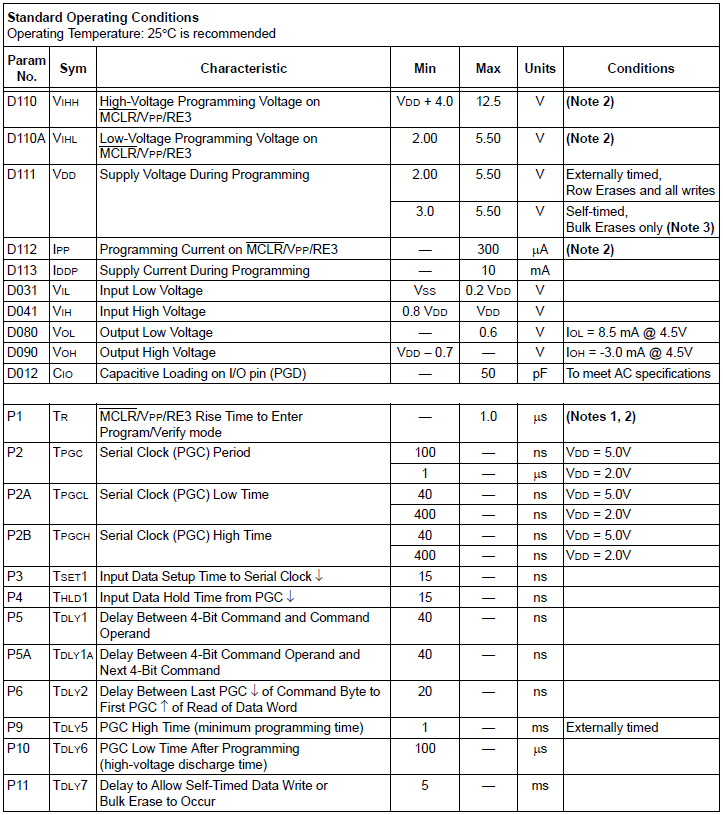
**Leyenda: las celdas sombreadas no están implementadas.**

**TABLA 5-5: MÁSCARAS DE PALABRAS DE CONFIGURACIÓN PARA COMPROBACIÓN DE COMPROBACIONES (CONTINUACIÓN)**



**Leyenda: las celdas sombreadas no están implementadas.**

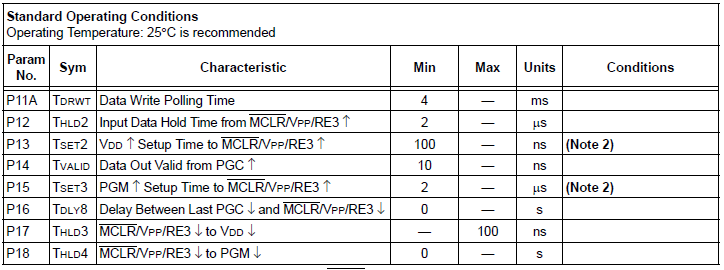
**6.0 REQUISITOS DE TEMPORIZACIÓN DE CARACTERÍSTICAS AC / DC  PARA EL PROGRAMA / MODO DE VERIFICACIÓN DE PRUEBA**



**Nota 1: No permita exceso de tiempo al hacer la transición de MCLR entre VIL y VIHH. Esto puede provocar ejecuciones espurias de programas. El tiempo de transición máximo es: 1 TCY + TPWRT (si está habilitado) + 1024 TOSC (solo para los modos LP, HS, HS / PLL y XT) + 2 ms (solo para el modo HS / PLL) + 1.5 micro segundos (para el modo EC solo)  donde TCY es el tiempo del ciclo de instrucciones, TPWRT es el período del temporizador de encendido y TOSC es el período del oscilador. Para valores específicos, consulte la sección Características eléctricas de la hoja de datos del dispositivo para el dispositivo en particular.**

**2: Cuando ICPRT = 1, esta especificación también se aplica a ICVPP. 3: a 0 ° C-50 ° C**

**6.0 REQUISITOS DE TEMPORIZACIÓN DE CARACTERÍSTICAS AC / DC PARA EL PROGRAMA / MODO DE VERIFICACIÓN DE LA PRUEBA (CONTINUACIÓN)**



**Nota 1: No permita exceso de tiempo al hacer la transición de MCLR entre VIL y VIHH. Esto puede provocar ejecuciones espurias de programas. El tiempo de transición máximo es: 1 TCY + TPWRT (si está habilitado) + 1024 TOSC (solo para los modos LP, HS, HS / PLL y XT) +  2 ms (solo para el modo HS / PLL) + 1.5 micro segundos (para Modo EC solamente) donde TCY es el tiempo del ciclo de instrucción, TPWRT es el período del temporizador de encendido y TOSC es el período del oscilador. Para valores específicos, consulte la sección Características eléctricas de la hoja de datos del dispositivo para el dispositivo en particular.**

**2: Cuando ICPRT = 1, esta especificación también se aplica a ICVPP.**

**3:** At 0°C-50°C.

**Tenga en cuenta los siguientes detalles de la función de protección de código en dispositivos Microchip:**

• Los productos de Microchip cumplen con las especificaciones contenidas en su hoja de datos de Microchip en particular.

• Microchip cree que su familia de productos es una de las familias más seguras de su tipo en el mercado hoy en día, cuando se usa de la manera prevista y en condiciones normales.

• Existen métodos deshonestos y posiblemente ilegales para violar la función de protección del código. Todos nuestros métodos, a nuestro entender, requieren el uso de los productos Microchip de una manera que esté fuera de las especificaciones operativas contenidas en las Hojas de datos de Microchip. Lo más probable es que la persona que lo hace esté involucrada en el robo de propiedad intelectual.

• Microchip está dispuesto a trabajar con el cliente que está preocupado por la integridad de su código.

• Ni Microchip ni ningún otro fabricante de semiconductores pueden garantizar la seguridad de su código. La protección del código no significa que garantizamos que el producto sea "irrompible".

Proteccion del codigo esta en constante evolucion. En Microchip estamos comprometidos a mejorar continuamente las características de protección de código de nuestros productos. Los intentos de romper la función de protección de código de Microchip pueden ser una violación de la Ley de Derechos de Autor Digital Millennium. Si tales actos permiten el acceso no autorizado a su software u otro trabajo protegido por derechos de autor, es posible que tenga derecho a demandar por alivio bajo esa Ley.

La información contenida en esta publicación sobre aplicaciones de dispositivos y similares se proporciona solo para su conveniencia y puede ser reemplazada por actualizaciones. Es su responsabilidad asegurarse de que su aplicación cumpla con sus especificaciones. MICROCHIP NO OFRECE REPRESENTACIONES NI GARANTÍAS DE NINGÚN TIPO, EXPLÍCITAS O IMPLÍCITAS, ESCRITAS U ORALES, ESTATUTARIAS O DE OTRA MANERA, RELACIONADAS CON LA INFORMACIÓN, INCLUIDAS, PERO SIN LIMITARSE A SU CONDICIÓN, CALIDAD, RENDIMIENTO, COMERCIABILIDAD O IDONEIDAD. Microchip niega toda responsabilidad derivada de esta información y su uso. El uso de dispositivos Microchip en aplicaciones de soporte vital y / o seguridad es totalmente bajo el riesgo del comprador, y el comprador acepta defender, indemnizar y eximir de responsabilidad a Microchip de todos los daños, reclamos, demandas o gastos resultantes de dicho uso. No se transmiten licencias, implícita o de otro modo, bajo ningún derecho de propiedad intelectual de Microchip a menos que se indique lo contrario.