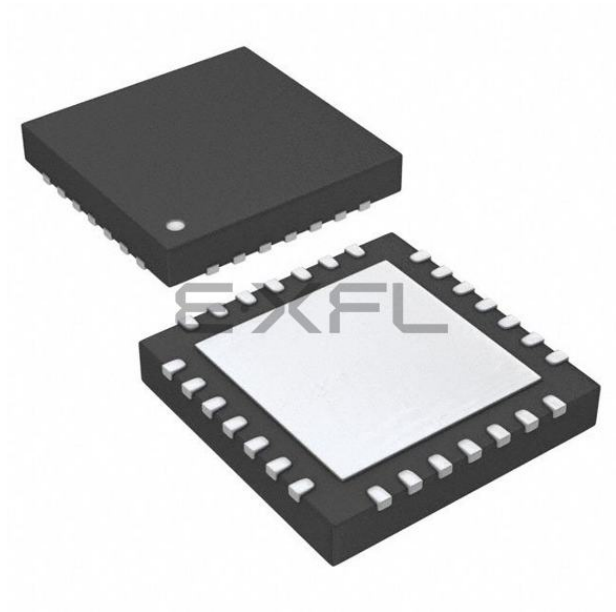


**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

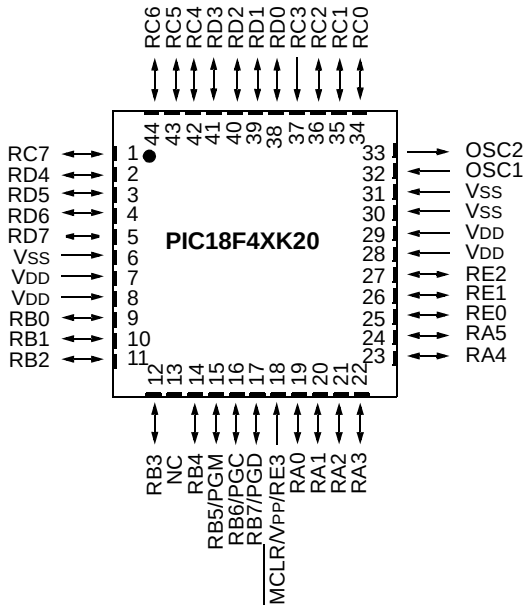
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

## Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Active |
| Core Processor | PIC |
| Core Size | 8-Bit |
| Speed | 48MHz |
| Connectivity | I²C, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, HLVD, POR, PWM, WDT |
| Number of I/O | 24 |
| Program Memory Size | 8KB (4K x 16) |
| Program Memory Type | FLASH |
| EEPROM Size | 256 x 8 |
| RAM Size | 512 x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.8V ~ 3.6V |
| Data Converters | A/D 11x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 125°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 28-VQFN Exposed Pad |
| Supplier Device Package | 28-QFN (6x6) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic18f23k20-e-ml |

**FIGURE 2-5: 44-PIN QFN PIN DIAGRAMS**

**44-PIN QFN**




**Note:** The following devices are included in 44-pin QFN parts: PIC18F43K20, PIC18F44K20, PIC18F45K20, PIC18F46K20.

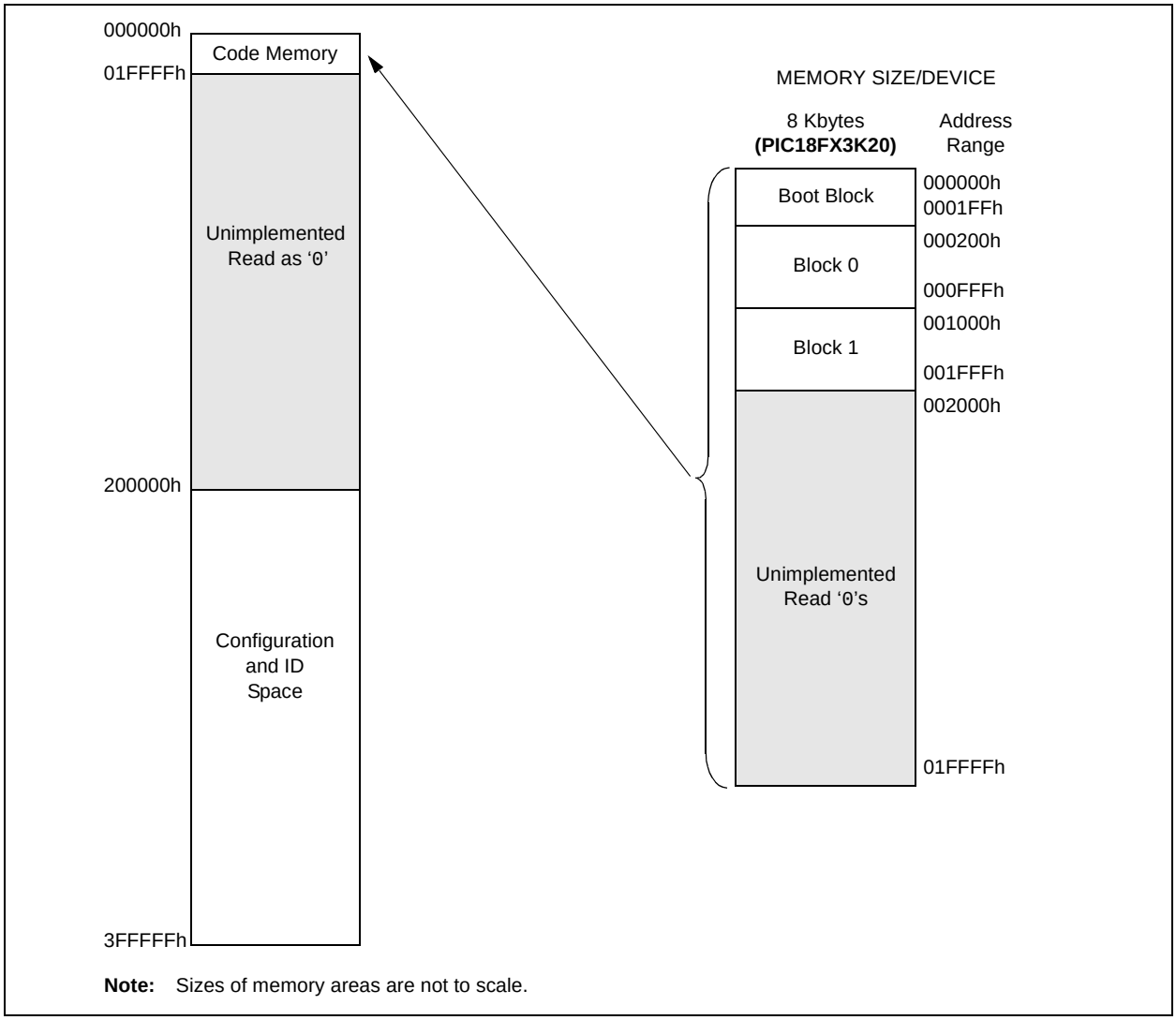
## 2.3 Memory Maps

For the PIC18FX3K20 devices, the code memory space extends from 0000h to 01FFFh (8 Kbytes) in two 4-Kbyte blocks. Addresses 0000h through 01FFh, however, define a “Boot Block” region that is treated separately from Block 0. All of these blocks define code protection boundaries within the code memory space.

**TABLE 2-2: IMPLEMENTATION OF CODE MEMORY**

| Device | Code Memory Size (Bytes) |
|---|---|
| PIC18F23K20 | 000000h-001FFFh (8K) |
| PIC18F43K20 | |

**FIGURE 2-6: MEMORY MAP AND THE CODE MEMORY SPACE FOR PIC18FX3K20 DEVICES**




**Note:** Sizes of memory areas are not to scale.

In addition to the code memory space, there are three blocks in the configuration and ID space that are accessible to the user through table reads and table writes. Their locations in the memory map are shown in Figure 2-10.

Users may store identification information (ID) in eight ID registers. These ID registers are mapped in addresses 200000h through 200007h. The ID locations read out normally, even after code protection is applied.

Locations 300000h through 30000Dh are reserved for the Configuration bits. These bits select various device options and are described in **Section 5.0 "Configuration Word"**. These Configuration bits read out normally, even after code protection.

Locations 3FFFFEh and 3FFFFFh are reserved for the device ID bits. These bits may be used by the programmer to identify what device type is being programmed and are described in **Section 5.0 "Configuration Word"**. These device ID bits read out normally, even after code protection.

### 2.3.1 MEMORY ADDRESS POINTER

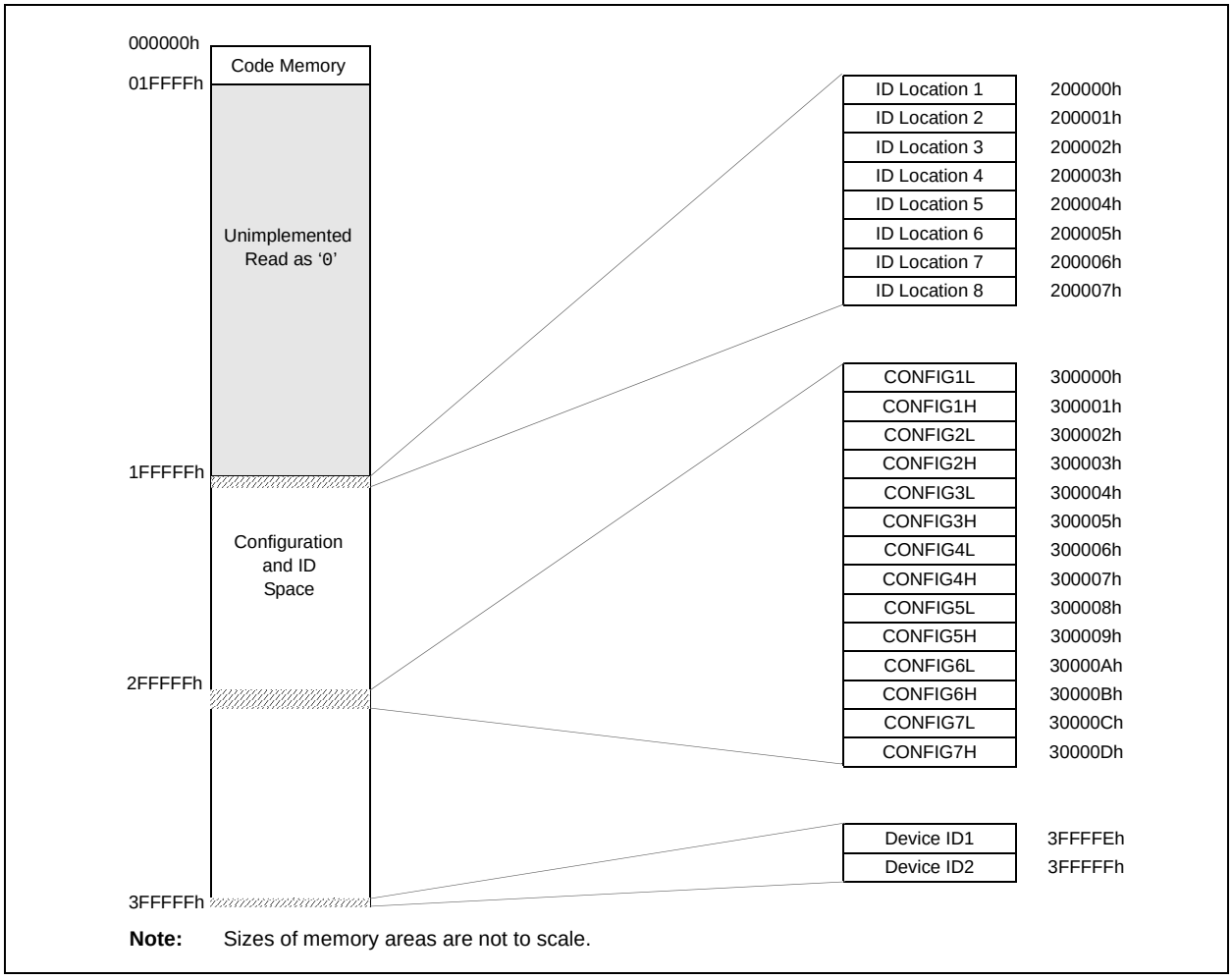
Memory in the address space, 0000000h to 3FFFFFh, is addressed via the Table Pointer register, which is comprised of three Pointer registers:

- TBLPTRU, at RAM address 0FF8h
- TBLPTRH, at RAM address 0FF7h
- TBLPTRL, at RAM address 0FF6h

| TBLPTRU | TBLPTRH | TBLPTRL |
|---|---|---|
| Addr[21:16] | Addr[15:8] | Addr[7:0] |

The 4-bit command, '0000' (core instruction), is used to load the Table Pointer prior to using any read or write operations.

**FIGURE 2-10: CONFIGURATION AND ID LOCATIONS FOR PIC18F2XK20/4XK20 DEVICES**




**Note:** Sizes of memory areas are not to scale.

## 2.6 Entering and Exiting Low-Voltage ICSP Program/Verify Mode

When the LVP Configuration bit is '1' (see **Section 5.3 "Single-Supply ICSP Programming"**), the Low-Voltage ICSP mode is enabled. As shown in Figure 2-14, Low-Voltage ICSP Program/Verify mode is entered by holding PGC and PGD low, placing a logic high on PGM and then raising MCLR/VPP/RE3 to VIH. In this mode, the RB5/PGM pin is dedicated to the programming function and ceases to be a general purpose I/O pin. Figure 2-15 shows the exit sequence.

The sequence that enters the device into the Program/Verify mode places all unused I/Os in the high-impedance state.

**FIGURE 2-14: ENTERING LOW-VOLTAGE PROGRAM/VERIFY MODE**

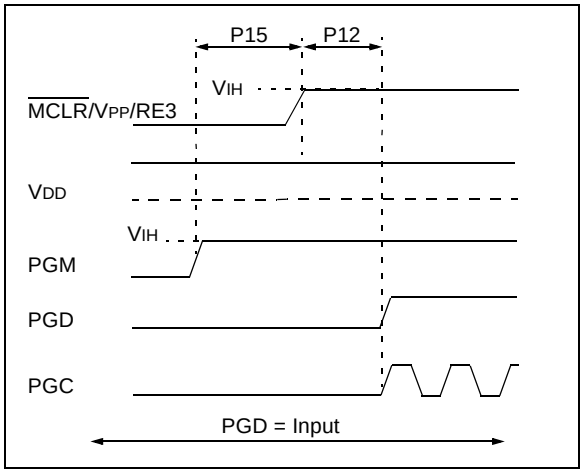
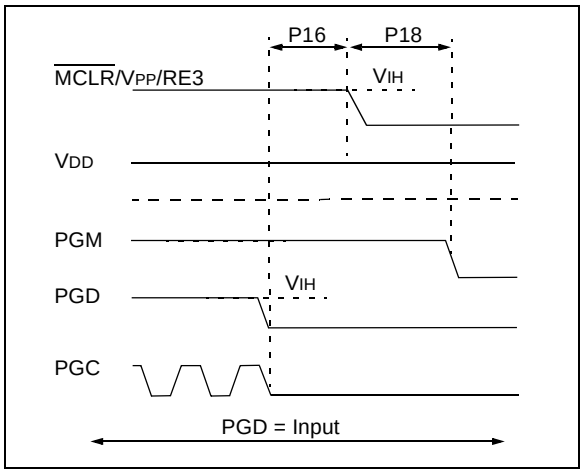



**FIGURE 2-15: EXITING LOW-VOLTAGE PROGRAM/VERIFY MODE**




## 2.7 Serial Program/Verify Operation

The PGC pin is used as a clock input pin and the PGD pin is used for entering command bits and data input/output during serial operation. Commands and data are transmitted on the rising edge of PGC, latched on the falling edge of PGC and are Least Significant bit (LSb) first.

### 2.7.1 4-BIT COMMANDS

All instructions are 20 bits, consisting of a leading 4-bit command followed by a 16-bit operand, which depends on the type of command being executed. To input a command, PGC is cycled four times. The commands needed for programming and verification are shown in Table 2-6.

Depending on the 4-bit command, the 16-bit operand represents 16 bits of input data or 8 bits of input data and 8 bits of output data.

Throughout this specification, commands and data are presented as illustrated in Table 2-7. The 4-bit command is shown Most Significant bit (MSb) first. The command operand, or "Data Payload", is shown <MSB><LSB>. Figure 2-16 demonstrates how to serially present a 20-bit command/operand to the device.

### 2.7.2 CORE INSTRUCTION

The core instruction passes a 16-bit instruction to the CPU core for execution. This is needed to set up registers as appropriate for use with other commands.

**TABLE 2-6: COMMANDS FOR PROGRAMMING**

| Description | 4-Bit Command |
|---|---|
| Core Instruction (Shift in16-bit instruction) | 0000 |
| Shift out TABLAT register | 0010 |
| Table Read | 1000 |
| Table Read, post-increment | 1001 |
| Table Read, post-decrement | 1010 |
| Table Read, pre-increment | 1011 |
| Table Write | 1100 |
| Table Write, post-increment by 2 | 1101 |
| Table Write, start programming, post-increment by 2 | 1110 |
| Table Write, start programming | 1111 |

## 3.2 Code Memory Programming

Programming code memory is accomplished by first loading data into the write buffer and then initiating a programming sequence. The write and erase buffer sizes shown in Table 3-4 can be mapped to any location of the same size beginning at 000000h. The actual memory write sequence takes the contents of this buffer and programs the proper amount of code memory that contains the Table Pointer.

The programming duration is externally timed and is controlled by PGC. After a Start Programming command is issued (4-bit command, '1111'), a NOP is issued, where the 4th PGC is held high for the duration of the programming time, P9.

After PGC is brought low, the programming sequence is terminated. PGC must be held low for the time specified by parameter P10 to allow high-voltage discharge of the memory array.

The code sequence to program a PIC18F2XK20/4XK20 device is shown in Table 3-5. The flowchart shown in Figure 3-4 depicts the logic necessary to completely write a PIC18F2XK20/4XK20 device. The timing diagram that details the Start Programming command and parameters P9 and P10 is shown in Figure 3-5.

| **Note:** | The TBLPTR register must point to the same region when initiating the programming sequence as it did when the write buffers were loaded. |
|---|---|

**TABLE 3-4: WRITE AND ERASE BUFFER SIZES**

| Devices (Arranged by Family) | Write Buffer Size (bytes) | Erase Size (bytes) |
|---|---|---|
| PIC18F26K20, PIC18F46K20 | 64 | 64 |
| PIC18F24K20, PIC18F25K20, PIC18F44K20, PIC18F45K20 | 32 | 64 |
| PIC18F23K20, PIC18F43K20 | 16 | 64 |

**TABLE 3-5: WRITE CODE MEMORY CODE SEQUENCE**

| 4-bit Command | Data Payload | Core Instruction |
|---|---|---|
| Step 1: Direct access to code memory. | | |
| 0000<br>0000<br>0000 | 8E A6<br>9C A6<br>84 A6 | BSF EECON1, EEPGD<br>BCF EECON1, CFGS<br>BSF EECON1, WREN |
| Step 2: Point to row to write. | | |
| 0000<br>0000<br>0000<br>0000<br>0000<br>0000 | 0E <Addr[21:16]><br>6E F8<br>0E <Addr[15:8]><br>6E F7<br>0E <Addr[7:0]><br>6E F6 | MOVLW <Addr[21:16]><br>MOVWF TBLPTRU<br>MOVLW <Addr[15:8]><br>MOVWF TBLPTRH<br>MOVLW <Addr[7:0]><br>MOVWF TBLPTRL |
| Step 3: Load write buffer. Repeat for all but the last two bytes. | | |
| 1101 | <MSB><LSB> | Write 2 bytes and post-increment address by 2. |
| Step 4: Load write buffer for last two bytes and start programming. | | |
| 1111<br>0000 | <MSB><LSB><br>00 00 | Write 2 bytes and start programming.<br>NOP - hold PGC high for time P9 and low for time P10. |
| To continue writing data, repeat steps 2 through 4, where the Address Pointer is incremented by 2 at each iteration of the loop. | | |

**FIGURE 3-4: PROGRAM CODE MEMORY FLOW**

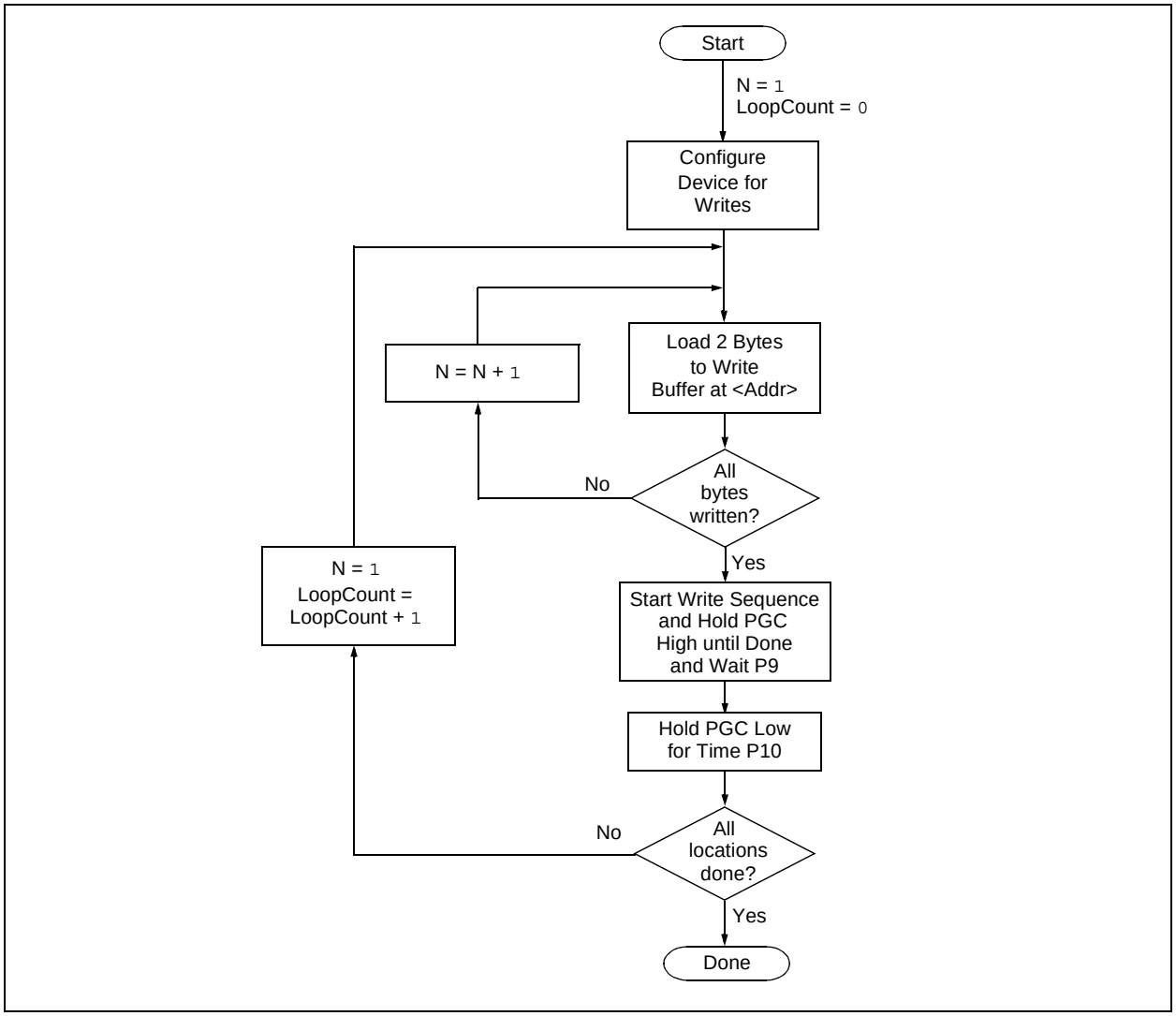
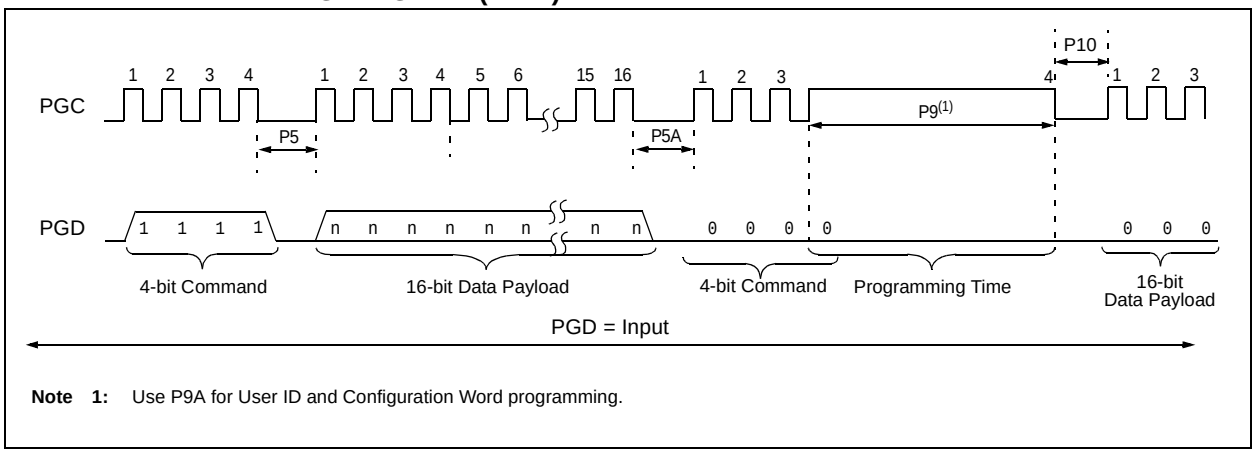



**FIGURE 3-5: TABLE WRITE AND START PROGRAMMING INSTRUCTION TIMING DIAGRAM (1111)**




**Note 1:** Use P9A for User ID and Configuration Word programming.

### 3.2.1 MODIFYING CODE MEMORY

The previous programming example assumed that the device has been Bulk Erased prior to programming (see **Section 3.1.1 "High-Voltage ICSP Bulk Erase"**). It may be the case, however, that the user wishes to modify only a section of an already programmed device.

The appropriate number of bytes required for the erase buffer must be read out of code memory (as described in **Section 4.2 "Verify Code Memory and ID Locations"**) and buffered. Modifications can be made on this buffer. Then, the block of code memory that was read out must be erased and rewritten with the modified data.

The WREN bit must be set if the WR bit in EECON1 is used to initiate a write sequence.

**TABLE 3-6: MODIFYING CODE MEMORY**

| 4-bit Command | Data Payload | Core Instruction |
|---|---|---|
| Step 1: Direct access to code memory. | | |
| 0000 | 8E A6 | BSF EECON1, EEPGD |
| 0000 | 9C A6 | BCF EECON1, CFGS |
| Step 2: Read code memory into buffer (**Section 4.1 "Read Code Memory, ID Locations and Configuration Bits"**). | | |
| Step 3: Set the Table Pointer for the block to be erased. | | |
| 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 |
| Step 4: Enable memory writes and setup an erase. | | |
| 0000 | 84 A6 | BSF EECON1, WREN |
| 0000 | 88 A6 | BSF EECON1, FREE |
| Step 5: Initiate erase. | | |
| 0000 | 88 A6 | BSF EECON1, FREE |
| 0000 | 82 A6 | BSF EECON1, WR |
| 0000 | 00 00 | NOP |
| 0000 | 00 00 | NOP Erase starts on the 4th clock of this instruction |
| Step 6: Poll WR bit. Repeat until bit is clear. | | |
| 0000 | 50 A6 | MOVF EECON1, W, 0 |
| 0000 | 6E F5 | MOVWF TABLAT |
| 0000 | 00 00 | NOP |
| 0000 | <MSB><LSB> | Shift out data(1) |
| Step 7: Load write buffer. The correct bytes will be selected based on the Table Pointer. | | |
| 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> | Write 2 bytes and post-increment address by 2. |
| • | • | |
| • | • | Repeat as many times as necessary to fill the write buffer |
| • | • | Write 2 bytes and start programming. |
| 1111 | <MSB><LSB> | NOP - hold PGC high for time P9 and low for time P10. |
| 0000 | 00 00 | |
| To continue modifying data, repeat Steps 2 through 6, where the Address Pointer is incremented by the appropriate number of bytes (see Table 3-4) at each iteration of the loop. The write cycle must be repeated enough times to completely rewrite the contents of the erase buffer. | | |
| Step 8: Disable writes. | | |
| 0000 | 94 A6 | BCF EECON1, WREN |

**TABLE 3-7: PROGRAMMING DATA MEMORY**

| 4-bit Command | Data Payload | Core Instruction |
|---|---|---|
| Step 1: Direct access to data EEPROM. | | |
| 0000 | 9E A6 | BCF EECON1, EEPGD |
| 0000 | 9C A6 | BCF EECON1, CFGS |
| Step 2: Set the data EEPROM Address Pointer. | | |
| 0000 | 0E <Addr> | MOVLW <Addr> |
| 0000 | 6E A9 | MOVWF EEADR |
| 0000 | OE <AddrH> | MOVLW <AddrH> |
| 0000 | 6E AA | MOVWF EEADRH |
| Step 3: Load the data to be written. | | |
| 0000 | 0E <Data> | MOVLW <Data> |
| 0000 | 6E A8 | MOVWF EEDATA |
| Step 4: Enable memory writes. | | |
| 0000 | 84 A6 | BSF EECON1, WREN |
| Step 5: Initiate write. | | |
| 0000 | 82 A6 | BSF EECON1, WR |
| 0000 | 00 00 | NOP |
| 0000 | 00 00 | NOP ;write starts on 4th clock of this instruction |
| Step 6: Poll WR bit, repeat until the bit is clear. | | |
| 0000 | 50 A6 | MOVF EECON1, W, 0 |
| 0000 | 6E F5 | MOVWF TABLAT |
| 0000 | 00 00 | NOP |
| 0010 | <MSB><LSB> | Shift out data(1) |
| Step 7: Hold PGC low for time P10. | | |
| Step 8: Disable writes. | | |
| 0000 | 94 A6 | BCF EECON1, WREN |
| Repeat steps 2 through 8 to write more data. | | |

**Note 1:** See Figure 4-4 for details on shift out data timing.

## 3.4 ID Location Programming

The ID locations are programmed much like the code memory. The ID registers are mapped in addresses 200000h through 200007h. These locations read out normally even after code protection.

| **Note:** | The user only needs to fill the first 8 bytes of the write buffer in order to write the ID locations. |
|---|---|

Table 3-8 demonstrates the code sequence required to write the ID locations.

In order to modify the ID locations, refer to the methodology described in **Section 3.2.1 "Modifying Code Memory"**. As with code memory, the ID locations must be erased before being modified.

When VDD is below the minimum for Bulk Erase operation, ID locations can be cleared with the Row Erase method described in **Section 3.1.3 "ICSP Row Erase"**.

**TABLE 3-8: WRITE ID SEQUENCE**

| 4-bit Command | Data Payload | Core Instruction |
|---|---|---|
| Step 1: Direct access to code memory. | | |
| 0000 | 8E A6 | BSF EECON1, EEPGD |
| 0000 | 9C A6 | BCF EECON1, CFGS |
| 0000 | 84 A6 | BSF EECON1, WREN |
| Step 2: Set Table Pointer to ID. Load write buffer with 8 bytes and write. | | |
| 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> | Write 2 bytes and post-increment address by 2. |
| 1101 | <MSB><LSB> | Write 2 bytes and post-increment address by 2. |
| 1101 | <MSB><LSB> | Write 2 bytes and post-increment address by 2. |
| 1111 | <MSB><LSB> | Write 2 bytes and start programming. |
| 0000 | 00 00 | NOP - hold PGC high for time P9 and low for time P10. |

## 3.5 Boot Block Programming

The code sequence detailed in Table 3-5 should be used, except that the address used in “Step 2” will be in the range of 000000h to 0007FFh.

## 3.6 Configuration Bits Programming

Unlike code memory, the Configuration bits are programmed a byte at a time. The Table Write, Begin Programming 4-bit command (‘1111’) is used, but only 8 bits of the following 16-bit payload will be written. The LSB of the payload will be written to even addresses and the MSB will be written to odd addresses. The code sequence to program two consecutive configuration locations is shown in Table 3-9. See Figure 3-5 for the timing diagram.

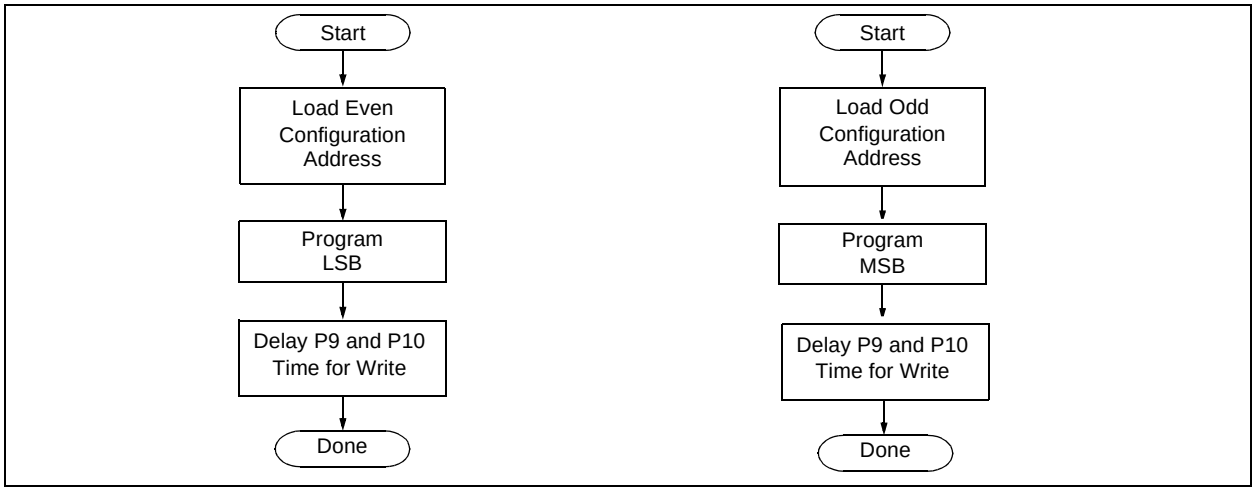
| **Note:** | The address must be explicitly written for each byte programmed. The addresses can not be incremented in this mode. |
|---|---|

**TABLE 3-9: SET ADDRESS POINTER TO CONFIGURATION LOCATION**

| 4-bit Command | Data Payload | Core Instruction |
|---|---|---|
| Step 1: Direct access to config memory. | | |
| 0000 | 8E A6 | BSF EECON1, EEPGD |
| 0000 | 8C A6 | BSF EECON1, CFGS |
| 0000 | 84 A6 | BSF EECON1, WREN |
| Step 2(1): Set Table Pointer for config byte to be written. Write even/odd addresses. | | |
| 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 ignored><LSB> | Load 2 bytes and start programming. |
| 0000 | 00 00 | NOP - hold PGC high for time P9 and low for time P10. |
| 0000 | 0E 01 | MOVLW 01h |
| 0000 | 6E F6 | MOVWF TBLPTRL |
| 1111 | <MSB><LSB ignored> | Load 2 bytes and start programming. |
| 0000 | 00 00 | NOP - hold PGC high for time P9A and low for time P10. |

**Note 1:** Enabling the write protection of Configuration bits (WRTC = 0 in CONFIG6H) will prevent further writing of Configuration bits. Always write all the Configuration bits before enabling the write protection for Configuration bits.

**FIGURE 3-8: CONFIGURATION PROGRAMMING FLOW**



## 4.0 READING THE DEVICE

### 4.1 Read Code Memory, ID Locations and Configuration Bits

Code memory is accessed one byte at a time via the 4-bit command, '1001' (table read, post-increment). The contents of memory pointed to by the Table Pointer (TBLPTRU:TBLPTRH:TBLPTRL) are serially output on PGD.

The 4-bit command is shifted in LSb first. The read is executed during the next 8 clocks, then shifted out on PGD during the last 8 clocks, LSb to MSb. A delay of P6 must be introduced after the falling edge of the 8th PGC of the operand to allow PGD to transition from an input to an output. During this time, PGC must be held low (see Figure 4-1). This operation also increments the Table Pointer by one, pointing to the next byte in code memory for the next read.

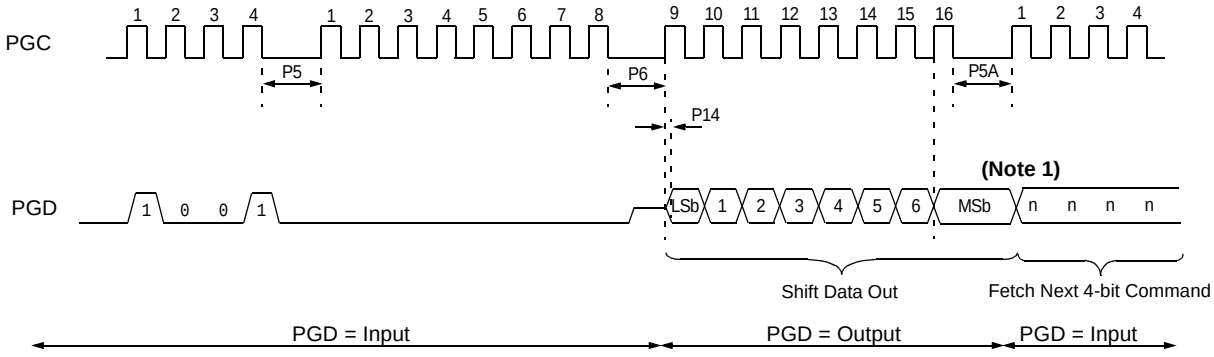
This technique will work to read any memory in the 000000h to 3FFFFFh address space, so it also applies to the reading of the ID and Configuration registers.

| **Note:** | When table read protection is enabled, the first read access to a protected block should be discarded and the read repeated to retrieve valid data. Subsequent reads of the same block can be performed normally. |
|---|---|

**TABLE 4-1: READ CODE MEMORY SEQUENCE**

| 4-bit Command | Data Payload | Core Instruction |
|---|---|---|
| Step 1: Set Table Pointer | | |
| 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 |
| Step 2: Read memory and then shift out on PGD, LSb to MSb | | |
| 1001 | 00 00 | TBLRD *+ |

**FIGURE 4-1: TABLE READ POST-INCREMENT INSTRUCTION TIMING DIAGRAM (1001)**




**Note 1:** Magnification of the high-impedance delay between PGC and PGD is shown in Figure 4-6.

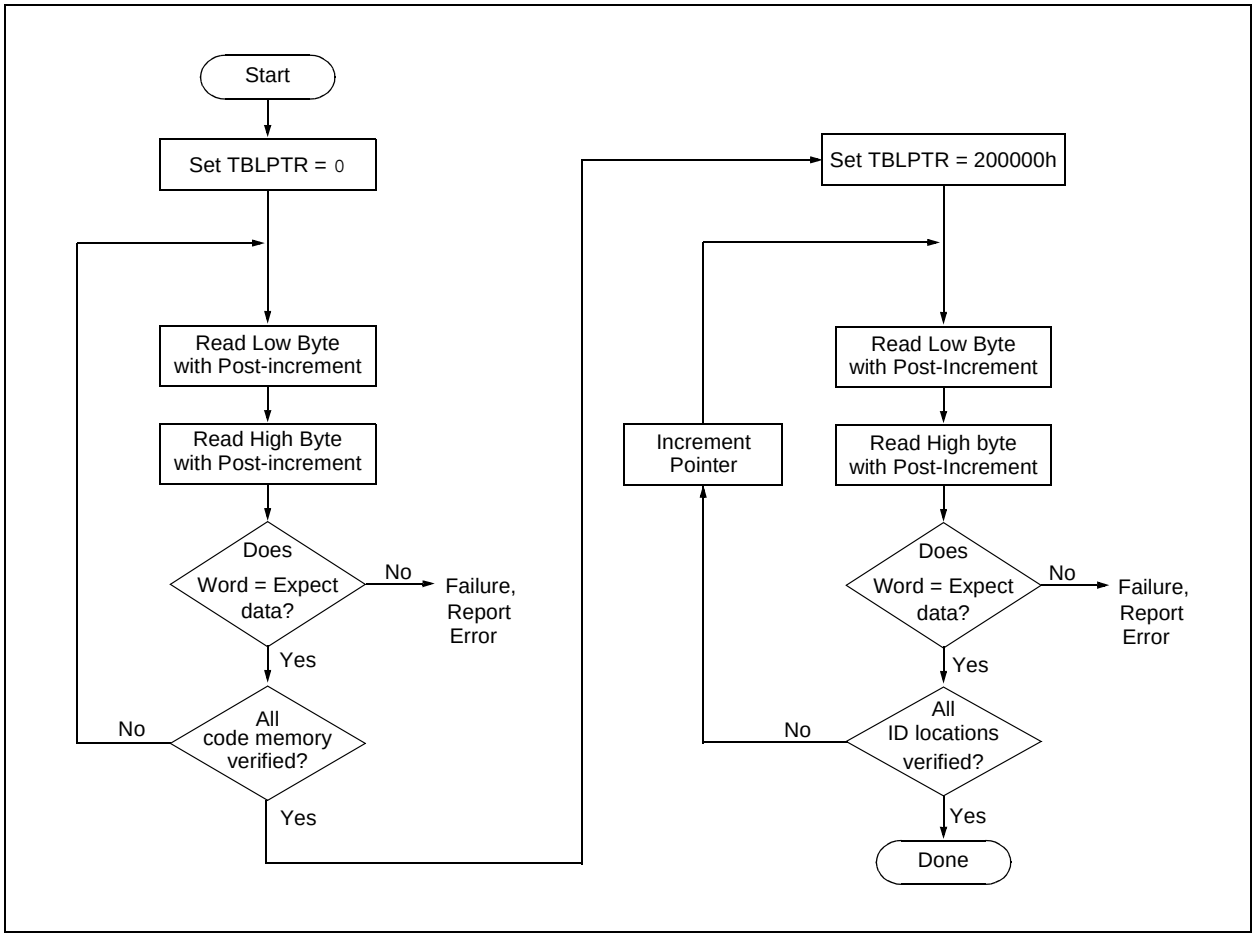
 

## 4.2 Verify Code Memory and ID Locations

The verify step involves reading back the code memory space and comparing it against the copy held in the programmer's buffer. Memory reads occur a single byte at a time, so two bytes must be read to compare against the word in the programmer's buffer. Refer to **Section 4.1 "Read Code Memory, ID Locations and Configuration Bits"** for implementation details of reading code memory.

The Table Pointer must be manually set to 200000h (base address of the ID locations) once the code memory has been verified. The post-increment feature of the table read 4-bit command can not be used to increment the Table Pointer beyond the code memory space. In a 64-Kbyte device, for example, a post-increment read of address FFFFh will wrap the Table Pointer back to 000000h, rather than point to unimplemented address 010000h.

**FIGURE 4-2: VERIFY CODE MEMORY FLOW**



## 4.3 Verify Configuration Bits

A configuration address may be read and output on PGD via the 4-bit command, '1001'. Configuration data is read and written in a byte-wise fashion, so it is not necessary to merge two bytes into a word prior to a compare. The result may then be immediately compared to the appropriate configuration data in the programmer's memory for verification. Refer to **Section 4.1 "Read Code Memory, ID Locations and Configuration Bits"** for implementation details of reading configuration data.

## 4.4 Read Data EEPROM Memory

Data EEPROM is accessed one byte at a time via an Address Pointer (register pair EEADRH:EEADR) and a data latch (EEDATA). Data EEPROM is read by loading EEADRH:EEADR with the desired memory location and initiating a memory read by appropriately configuring the EECON1 register. The data will be loaded into EEDATA, where it may be serially output on PGD via the 4-bit command, '0010' (Shift Out Data Holding register). A delay of P6 must be introduced after the falling edge of the 8th PGC of the operand to allow PGD to transition from an input to an output. During this time, PGC must be held low (see Figure 4-4).

The command sequence to read a single byte of data is shown in Table 4-2.

**FIGURE 4-3: READ DATA EEPROM FLOW**

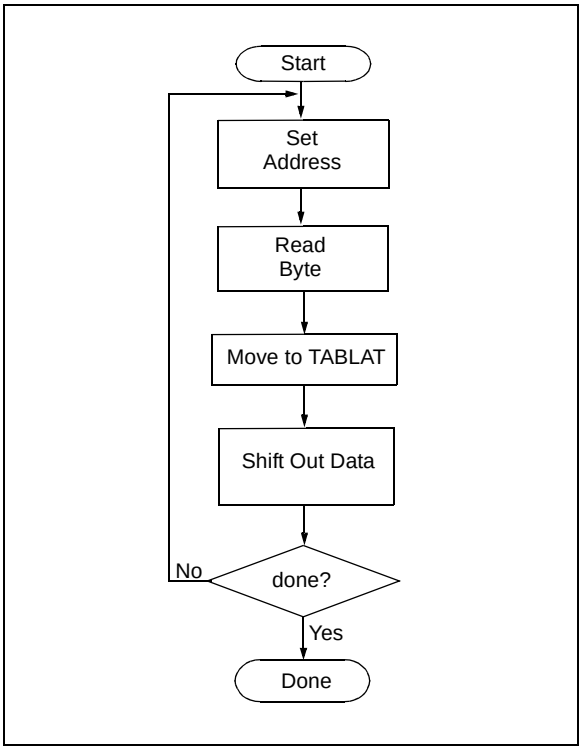



**TABLE 4-2: READ DATA EEPROM MEMORY**

| 4-bit Command | Data Payload | Core Instruction |
|---|---|---|
| Step 1: Direct access to data EEPROM. | | |
| 0000 | 9E A6 | BCF EECON1, EEPGD |
| 0000 | 9C A6 | BCF EECON1, CFGS |
| Step 2: Set the data EEPROM Address Pointer. | | |
| 0000 | 0E <Addr> | MOVLW <Addr> |
| 0000 | 6E A9 | MOVWF EEADR |
| 0000 | OE <AddrH> | MOVLW <AddrH> |
| 0000 | 6E AA | MOVWF EEADRH |
| Step 3: Initiate a memory read. | | |
| 0000 | 80 A6 | BSF EECON1, RD |
| Step 4: Load data into the Serial Data Holding register. | | |
| 0000 | 50 A8 | MOVF EEDATA, W, 0 |
| 0000 | 6E F5 | MOVWF TABLAT |
| 0000 | 00 00 | NOP |
| 0010 | <MSB><LSB> | Shift Out Data(1) |

**Note 1:** The <LSB> is undefined. The <MSB> is the data.

**TABLE 5-2: DEVICE ID VALUE**

| Device | Device ID Value | |
|---|---|---|
| | DEVID2 | DEVID1 |
| PIC18F23K20 | 20h | 111x xxxx |
| PIC18F24K20 | 20h | 101x xxxx |
| PIC18F25K20 | 20h | 011x xxxx |
| PIC18F26K20 | 20h | 001x xxxx |
| PIC18F43K20 | 20h | 110x xxxx |
| PIC18F44K20 | 20h | 100x xxxx |
| PIC18F45K20 | 20h | 010x xxxx |
| PIC18F46K20 | 20h | 000x xxxx |

**Note:** The ‘x’s in DEVID1 contain the device revision code.

**TABLE 5-3: PIC18F2XK20/4XK20 BIT DESCRIPTIONS (CONTINUED)**

| Bit Name | Configuration Words | Description |
|---|---|---|
| CP3 | CONFIG5L | Code Protection bits (Block 3 code memory area)<br>1 = Block 3 is not code-protected<br>0 = Block 3 is code-protected |
| CP2 | CONFIG5L | Code Protection bits (Block 2 code memory area)<br>1 = Block 2 is not code-protected<br>0 = Block 2 is code-protected |
| CP1 | CONFIG5L | Code Protection bits (Block 1 code memory area)<br>1 = Block 1 is not code-protected<br>0 = Block 1 is code-protected |
| CP0 | CONFIG5L | Code Protection bits (Block 0 code memory area)<br>1 = Block 0 is not code-protected<br>0 = Block 0 is code-protected |
| CPD | CONFIG5H | Code Protection bits (Data EEPROM)<br>1 = Data EEPROM is not code-protected<br>0 = Data EEPROM is code-protected |
| CPB | CONFIG5H | Code Protection bits (Boot Block memory area)<br>1 = Boot Block is not code-protected<br>0 = Boot Block is code-protected |
| WRT3 | CONFIG6L | Write Protection bits (Block 3 code memory area)<br>1 = Block 3 is not write-protected<br>0 = Block 3 is write-protected |
| WRT2 | CONFIG6L | Write Protection bits (Block 2 code memory area)<br>1 = Block 2 is not write-protected<br>0 = Block 2 is write-protected |
| WRT1 | CONFIG6L | Write Protection bits (Block 1 code memory area)<br>1 = Block 1 is not write-protected<br>0 = Block 1 is write-protected |
| WRT0 | CONFIG6L | Write Protection bits (Block 0 code memory area)<br>1 = Block 0 is not write-protected<br>0 = Block 0 is write-protected |
| WRTD | CONFIG6H | Write Protection bit (Data EEPROM)<br>1 = Data EEPROM is not write-protected<br>0 = Data EEPROM is write-protected |
| WRTB | CONFIG6H | Write Protection bit (Boot Block memory area)<br>1 = Boot Block is not write-protected<br>0 = Boot Block is write-protected |
| WRTC | CONFIG6H | Write Protection bit (Configuration registers)<br>1 = Configuration registers are not write-protected<br>0 = Configuration registers are write-protected |

.

**TABLE 5-3: PIC18F2XK20/4XK20 BIT DESCRIPTIONS (CONTINUED)**

| Bit Name | Configuration Words | Description |
|---|---|---|
| EBTR3 | CONFIG7L | Table Read Protection bit (Block 3 code memory area)<br>1 = Block 3 is not protected from table reads executed in other blocks<br>0 = Block 3 is protected from table reads executed in other blocks |
| EBTR2 | CONFIG7L | Table Read Protection bit (Block 2 code memory area)<br>1 = Block 2 is not protected from table reads executed in other blocks<br>0 = Block 2 is protected from table reads executed in other blocks |
| EBTR1 | CONFIG7L | Table Read Protection bit (Block 1 code memory area)<br>1 = Block 1 is not protected from table reads executed in other blocks<br>0 = Block 1 is protected from table reads executed in other blocks |
| EBTR0 | CONFIG7L | Table Read Protection bit (Block 0 code memory area)<br>1 = Block 0 is not protected from table reads executed in other blocks<br>0 = Block 0 is protected from table reads executed in other blocks |
| EBTRB | CONFIG7H | Table Read Protection bit (Boot Block memory area)<br>1 = Boot Block is not protected from table reads executed in other blocks<br>0 = Boot Block is protected from table reads executed in other blocks |
| DEV<10:3> | DEVID2 | Device ID bits<br>These bits are used with the DEV<2:0> bits in the DEVID1 register to identify part number. |
| DEV<2:0> | DEVID1 | Device ID bits<br>These bits are used with the DEV<10:3> bits in the DEVID2 register to identify part number. |
| REV<4:0> | DEVID1 | Revision ID bits<br>These bits are used to indicate the revision of the device. |

.

**TABLE 5-4: CHECKSUM COMPUTATION (CONTINUED)**

| Device | Code-Protect | Checksum | Blank Value | 0xAA at 0 and Max Address |
|---|---|---|---|---|
| PIC18FX6K20 | None | SUM[0000:07FF]+SUM[0800:3FFF]+SUM[4000:7FFF]+ SUM[8000:BFFF]+SUM[C000:FFFF]+(CONFIG1L & 00h)+ (CONFIG1H & CFh)+(CONFIG2L & 1Fh)+(CONFIG2H & 1F)+ (CONFIG3L & 00h)+(CONFIG3H & 8Fh)+(CONFIG4L & C5h)+ (CONFIG4H & 00h)+(CONFIG5L & 0Fh)+(CONFIG5H & C0h)+ (CONFIG6L & 0Fh)+(CONFIG6H & E0h)+(CONFIG7L & 0Fh)+ (CONFIG7H & 40h) | 0362h | 02B8h |
| | Boot Block | SUM[0800:3FFF]+SUM[4000:7FFF]+SUM[8000:BFFF]+SUM[C000:FFFF]+ (CONFIG1L & 00h)+(CONFIG1H & CFh)+(CONFIG2L & 1Fh)+ (CONFIG2H & 1F)+(CONFIG3L & 00h)+(CONFIG3H & 8Fh)+ (CONFIG4L & C5h)+(CONFIG4H & 00h)+(CONFIG5L & 0Fh)+ (CONFIG5H & C0h)+(CONFIG6L & 0Fh)+(CONFIG6H & E0h)+ (CONFIG7L & 0Fh)+(CONFIG7H & 40h)+SUM_ID | 0B2Dh | 0AE2h |
| | Boot/ Block 0/ Block 1 | SUM[3000:BFFF]+SUM[C000:FFFF]+(CONFIG1L & 00h)+ (CONFIG1H & CFh)+(CONFIG2L & 1Fh)+(CONFIG2H & 1F)+ (CONFIG3L & 00h)+(CONFIG3H & 8Fh)+(CONFIG4L & C5h)+ (CONFIG4H & 00h)+(CONFIG5L & 0Fh)+(CONFIG5H & C0h)+ (CONFIG6L & 0Fh)+(CONFIG6H & E0h)+(CONFIG7L & 0Fh)+ (CONFIG7H & 40h)+SUM_ID | 832Ah | 82DFh |
| | All | (CONFIG1L & 00h)+(CONFIG1H & CFh)+(CONFIG2L & 1Fh)+ (CONFIG2H & 1F)+(CONFIG3L & 00h)+(CONFIG3H & 8Fh)+ (CONFIG4L & C5h)+(CONFIG4H & 00h)+(CONFIG5L & 0Fh)+ (CONFIG5H & C0h)+(CONFIG6L & 0Fh)+(CONFIG6H & E0h)+ (CONFIG7L & 0Fh)+(CONFIG7H & 40h)+SUM_ID | 031Eh | 0328h |

**Legend:**

| Item | | Description |
|---|---|---|
| CONFIGx | = | Configuration Word |
| SUM[a:b] | = | Sum of locations, a to b inclusive |
| SUM_ID | = | Byte-wise sum of lower four bits of all customer ID locations |
| + | = | Addition |
| & | = | Bit-wise AND |

## 6.0 AC/DC CHARACTERISTICS TIMING REQUIREMENTS FOR PROGRAM/ VERIFY TEST MODE

**Standard Operating Conditions**
Operating Temperature: 25°C is recommended

| Param No. | Sym. | Characteristic | Min. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| D110 | VIHH | High-Voltage Programming Voltage on MCLR/VPP/RE3 | VDD + 4.5 | 9 | V | |
| D110A | VIHL | Low-Voltage Programming Voltage on MCLR/VPP/RE3 | 1.80 | 3.60 | V | |
| D111 | VDD | Supply Voltage During Programming | 1.80 | 3.60 | V | Row Erase/Write |
| | | | 2.7 | 3.60 | V | Bulk Erase operations |
| D112 | IPP | Programming Current on MCLR/VPP/RE3 | — | 300 | μA | |
| D113 | IDDP | Supply Current During Programming | — | 10 | mA | |
| D031 | VIL | Input Low Voltage | VSS | 0.2 VDD | V | |
| D041 | VIH | Input High Voltage | 0.8 VDD | VDD | V | |
| D080 | VOL | Output Low Voltage | — | 0.6 | V | IOL = X.X mA @ 2.7V |
| D090 | VOH | Output High Voltage | VDD – 0.7 | — | V | IOH = -Y.Y mA @ 2.7V |
| D012 | CIO | Capacitive Loading on I/O pin (PGD) | — | 50 | pF | To meet AC specifications |
| | | | | | | |
| P1 | TR | MCLR/VPP/RE3 Rise Time to enter Program/Verify mode | — | 1.0 | μs | **(Note 1)** |
| P2 | TPGC | Serial Clock (PGC) Period | 100 | — | ns | VDD = 3.6V |
| | | | 1 | — | μs | VDD = 1.8V |
| P2A | TPGCL | Serial Clock (PGC) Low Time | 40 | — | ns | VDD = 3.6V |
| | | | 400 | — | ns | VDD = 1.8V |
| P2B | TPGCH | Serial Clock (PGC) High Time | 40 | — | ns | VDD = 3.6V |
| | | | 400 | — | ns | VDD = 1.8V |
| P3 | TSET1 | Input Data Setup Time to Serial Clock ↓ | 15 | — | ns | |
| P4 | THLD1 | Input Data Hold Time from PGC ↓ | 15 | — | ns | |
| P5 | TDLY1 | Delay between 4-bit Command and Command Operand | 40 | — | ns | |
| P5A | TDLY1A | Delay between 4-bit Command Operand and next 4-bit Command | 40 | — | ns | |
| P6 | TDLY2 | Delay between Last PGC ↓ of Command Byte to First PGC ↑ of Read of Data Word | 20 | — | ns | |
| P9 | TDLY5 | PGC High Time (minimum programming time) | 1 | — | ms | Externally Timed |
| P9A | TDLY5A | PGC High Time | 5 | | ms | Configuration Word programming time |
| P10 | TDLY6 | PGC Low Time after Programming (high-voltage discharge time) | 200 | — | μs | |
| P11 | TDLY7 | Delay to allow Self-Timed Data Write or Bulk Erase to occur | 5 | — | ms | |
| P11A | TDRWT | Data Write Polling Time | 4 | — | ms | |

**Note 1:** Do not allow excess time when transitioning MCLR between VIL and VIHH; this can cause spurious program executions to occur. The maximum transition time is:
1 TCY + TPWRT (if enabled) + 1024 TOSC (for LP, HS, HS/PLL and XT modes only) + 2 ms (for HS/PLL mode only) + 1.5 μs (for EC mode only) where TCY is the instruction cycle time, TPWRT is the Power-up Timer period and TOSC is the oscillator period. For specific values, refer to the Electrical Characteristics section of the device data sheet for the particular device.

## 6.0 AC/DC CHARACTERISTICS TIMING REQUIREMENTS FOR PROGRAM/ VERIFY TEST MODE (CONTINUED)

**Standard Operating Conditions**
Operating Temperature: 25°C is recommended

| Param No. | Sym. | Characteristic | Min. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| P12 | THLD2 | Input Data Hold Time from $\overline{MCLR}$/VPP/RE3 ↑ | 2 | — | µs | |
| P13 | TSET2 | VDD ↑ Setup Time to $\overline{MCLR}$/VPP/RE3 ↑ | 100 | — | ns | |
| P14 | TVALID | Data Out Valid from PGC ↑ | 10 | — | ns | |
| P15 | TSET3 | PGM ↑ Setup Time to $\overline{MCLR}$/VPP/RE3 ↑ | 2 | — | µs | |
| P16 | TDLY8 | Delay between Last PGC ↓ and $\overline{MCLR}$/VPP/RE3 ↓ | 0 | — | s | |
| P17 | THLD3 | $\overline{MCLR}$/VPP/RE3 ↓ to VDD ↓ | — | 100 | ns | |
| P18 | THLD4 | $\overline{MCLR}$/VPP/RE3 ↓ to PGM ↓ | 0 | — | s | |
| P19 | THIZ | Delay from PGC ↑ to PGD High-Z | 3 | 10 | nS | |
| P20 | TPPDP | Hold time after VPP changes | 5 | — | µs | |

**Note 1:** Do not allow excess time when transitioning $\overline{MCLR}$ between VIL and VIHH; this can cause spurious program executions to occur. The maximum transition time is:
1 TCY + TPWRT (if enabled) + 1024 TOSC (for LP, HS, HS/PLL and XT modes only) + 2 ms (for HS/PLL mode only) + 1.5 µs (for EC mode only) where TCY is the instruction cycle time, TPWRT is the Power-up Timer period and TOSC is the oscillator period. For specific values, refer to the Electrical Characteristics section of the device data sheet for the particular device.

**Note the following details of the code protection feature on Microchip devices:**

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.



**Trademarks**



*Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*