



# PIC18F2XK20/4XK20

## Flash Memory Programming Specification

### 1.0 DEVICE OVERVIEW

This document includes the programming specifications for the following devices:

- PIC18F23K20
- PIC18F24K20
- PIC18F25K20
- PIC18F26K20
- PIC18F43K20
- PIC18F44K20
- PIC18F45K20
- PIC18F46K20

### 2.0 PROGRAMMING OVERVIEW

The PIC18F2XK20/4XK20 devices can be programmed using either the high-voltage In-Circuit Serial Programming™ (ICSP™) method or the low-voltage ICSP method. Both methods can be done with the device in the users' system. The low-voltage ICSP method is slightly different than the high-voltage method and these differences are noted where applicable. This programming specification applies to the PIC18F2XK20/4XK20 devices in all package types.

### 2.1 Hardware Requirements

In High-Voltage ICSP mode, the PIC18F2XK20/4XK20 devices require two programmable power supplies: one for VDD and one for MCLR/VPP/RE3. Both supplies should have a minimum resolution of 0.25V. Refer to **Section 6.0 “AC/DC Characteristics Timing Requirements for Program/Verify Test Mode”** for additional hardware parameters.

#### 2.1.1 LOW-VOLTAGE ICSP PROGRAMMING

In Low-Voltage ICSP mode, the PIC18F2XK20/4XK20 devices can be programmed using a single VDD source in the operating range. The MCLR/VPP/RE3 does not have to be brought to a different voltage, but can instead be left at the normal operating voltage. Refer to **Section 6.0 “AC/DC Characteristics Timing Requirements for Program/Verify Test Mode”** for additional hardware parameters.

### 2.2 Pin Diagrams

The pin diagrams for the PIC18F2XK20/4XK20 family are shown in Figure 2-3 and Figure 2-4.

**TABLE 2-1: PIN DESCRIPTIONS (DURING PROGRAMMING): PIC18F2XK20/4XK20**

| Pin Name           | During Programming |          |  |
|--------------------|--------------------|----------|--|
|                    | Pin Name           | Pin Type | Pin Description  |
| MCLR/VPP/RE3       | VPP                | P        | Programming Enable   |
| VDD <sup>(2)</sup> | VDD                | P        | Power Supply   |
| VSS <sup>(2)</sup> | VSS                | P        | Ground   |
| RB5                | PGM                | I        | Low-Voltage ICSP™ input when LVP Configuration bit equals '1' <sup>(1)</sup> |
| RB6                | PGC                | I        | Serial Clock   |
| RB7                | PGD                | I/O      | Serial Data  |

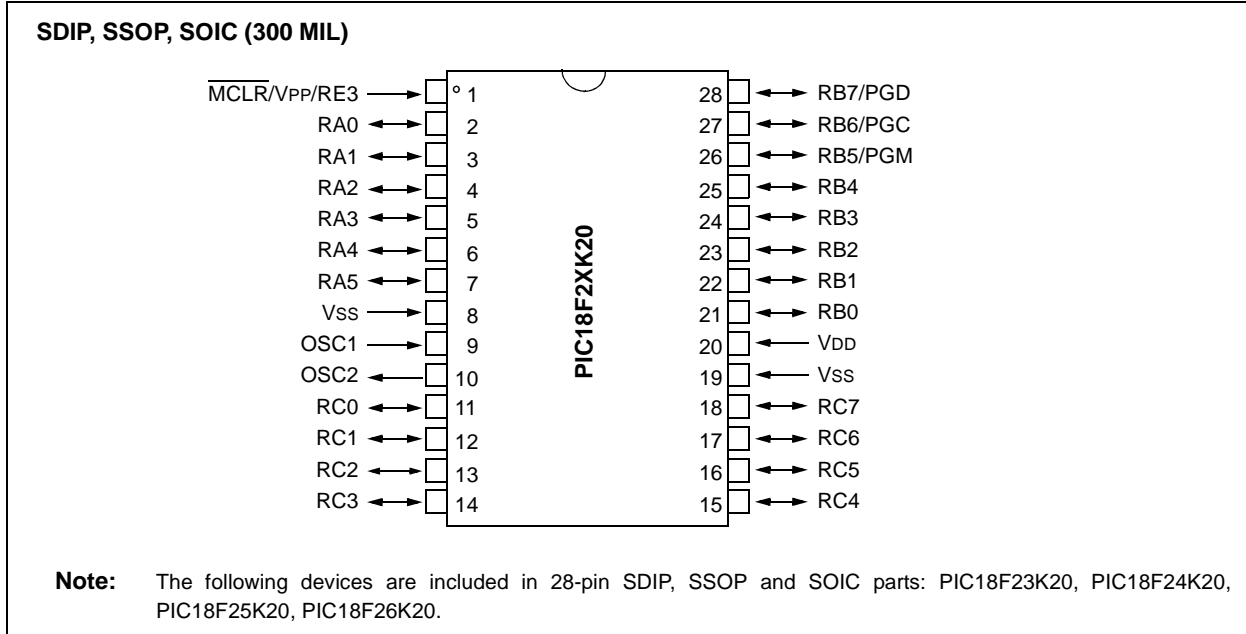
**Legend:** I = Input, O = Output, P = Power

**Note 1:** See Figure 5-1 for more information.

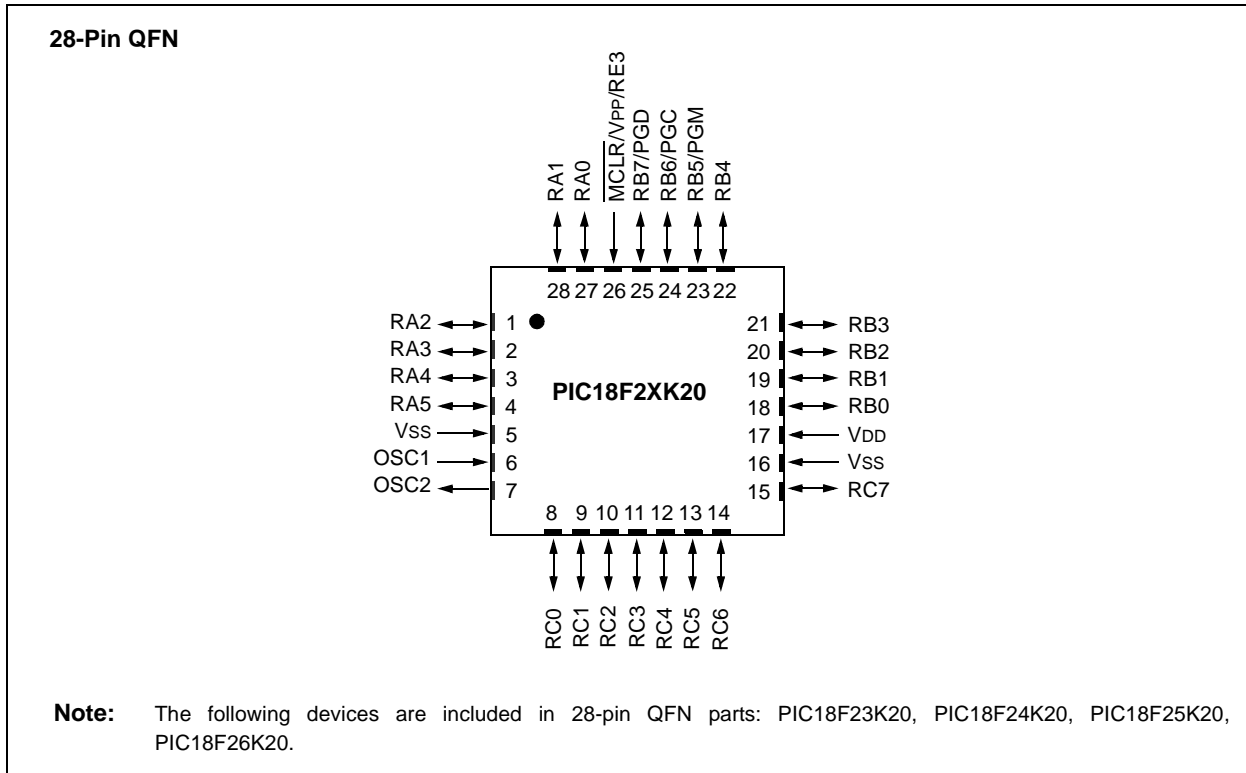
**2:** All power supply (VDD) and ground (VSS) pins must be connected.

# PIC18F2XK20/4XK20

**FIGURE 2-1: 28-PIN SDIP, SSOP AND SOIC PIN DIAGRAMS**

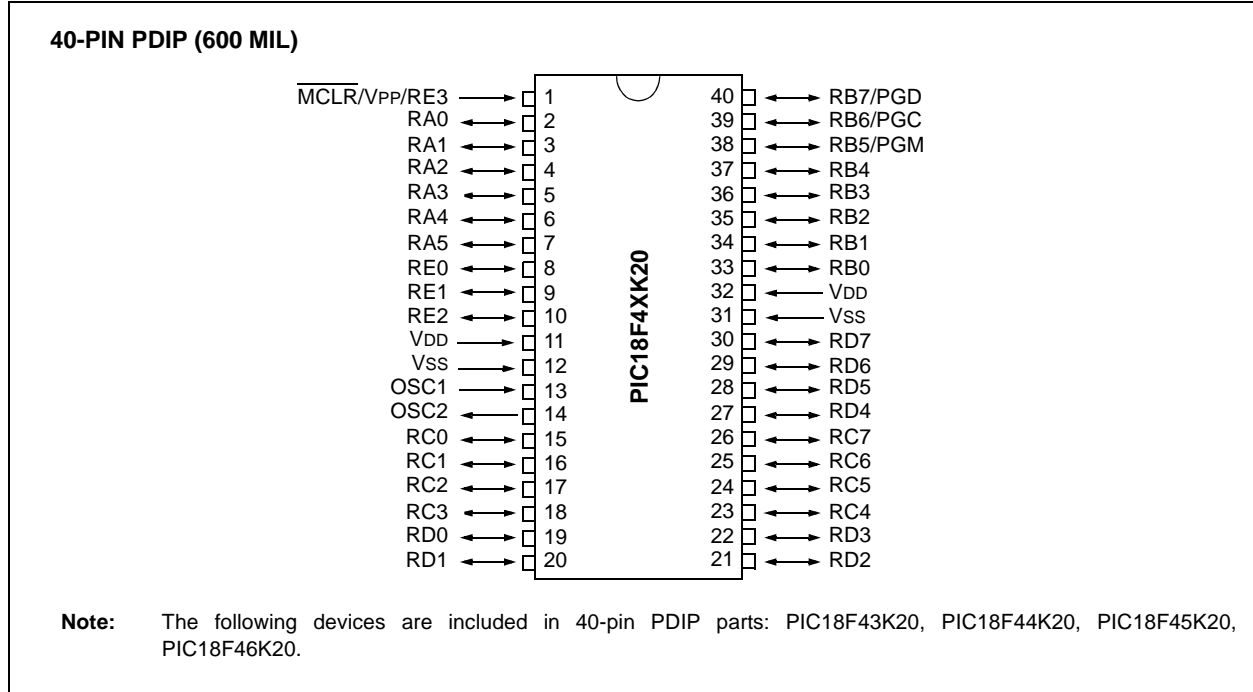


**FIGURE 2-2: 28-PIN QFN PIN DIAGRAMS**

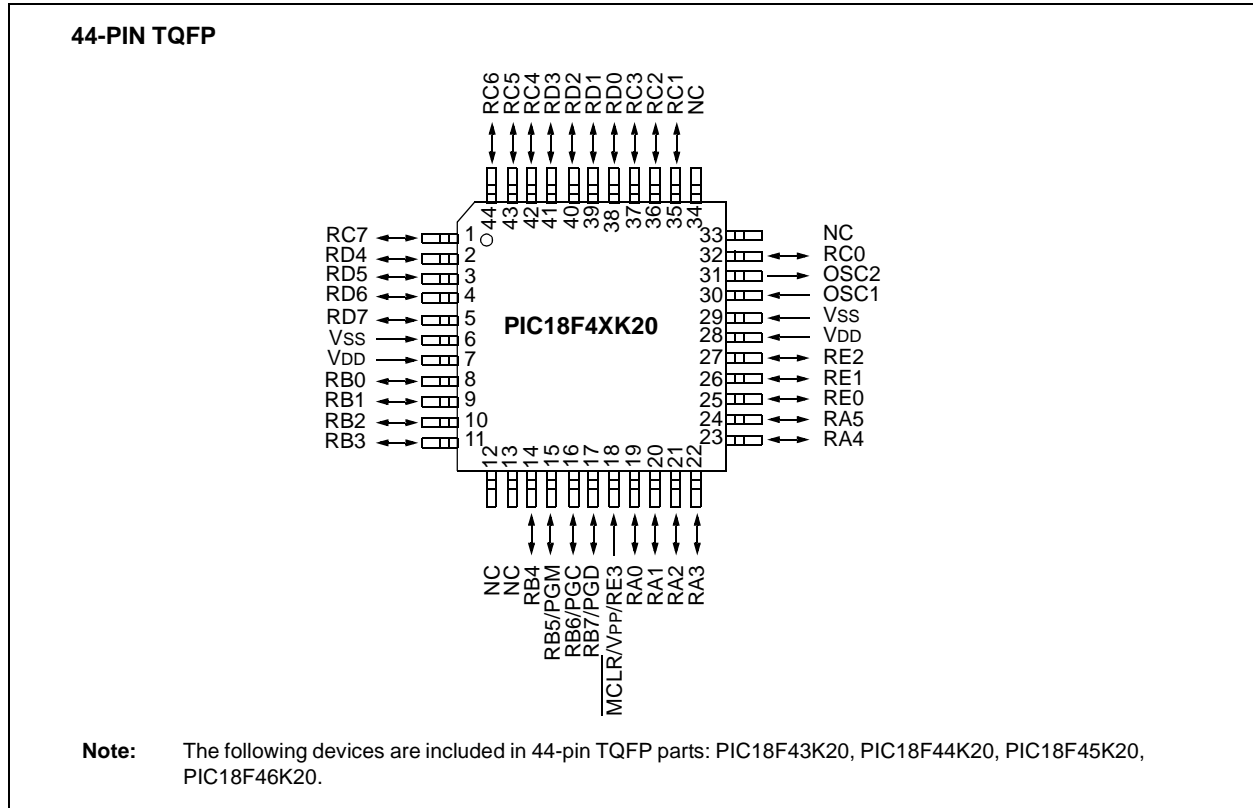


# PIC18F2XK20/4XK20

**FIGURE 2-3: 40-PIN PDIP PIN DIAGRAMS**

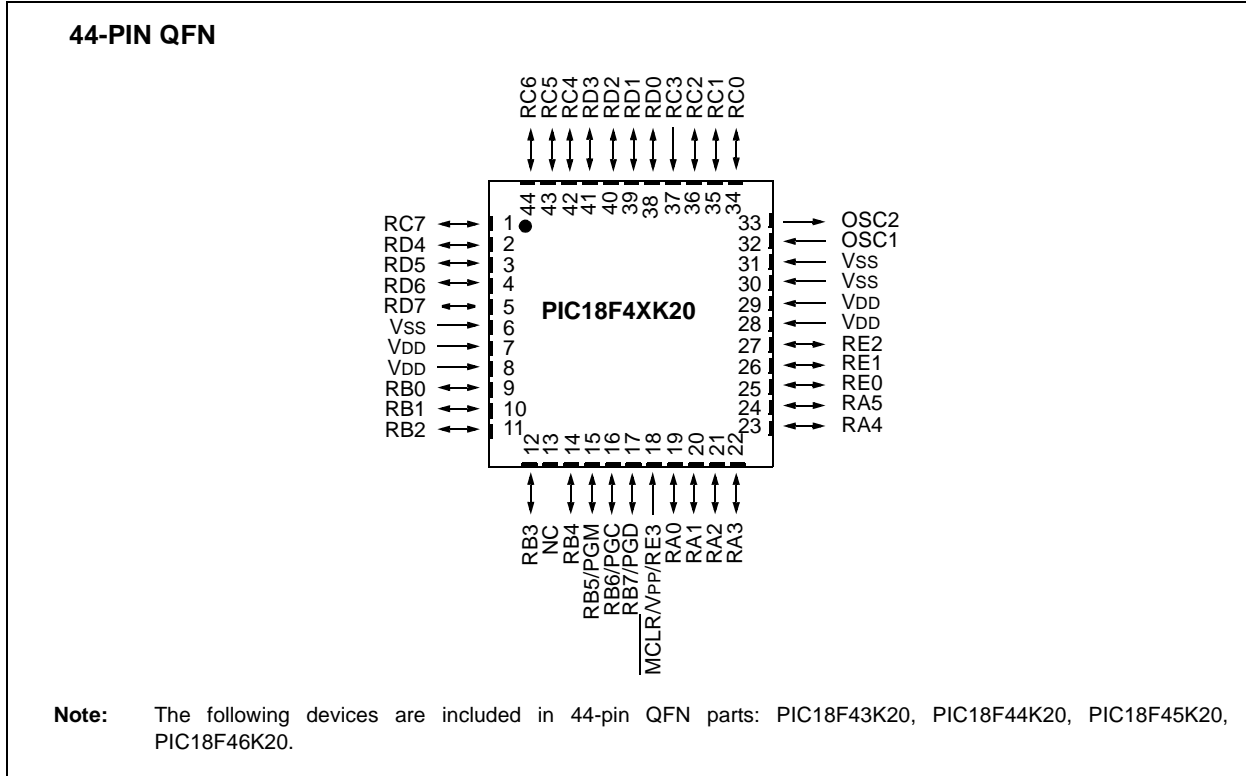


**FIGURE 2-4: 44-PIN TQFP PIN DIAGRAMS**



# PIC18F2XK20/4XK20

FIGURE 2-5: 44-PIN QFN PIN DIAGRAMS



# PIC18F2XK20/4XK20

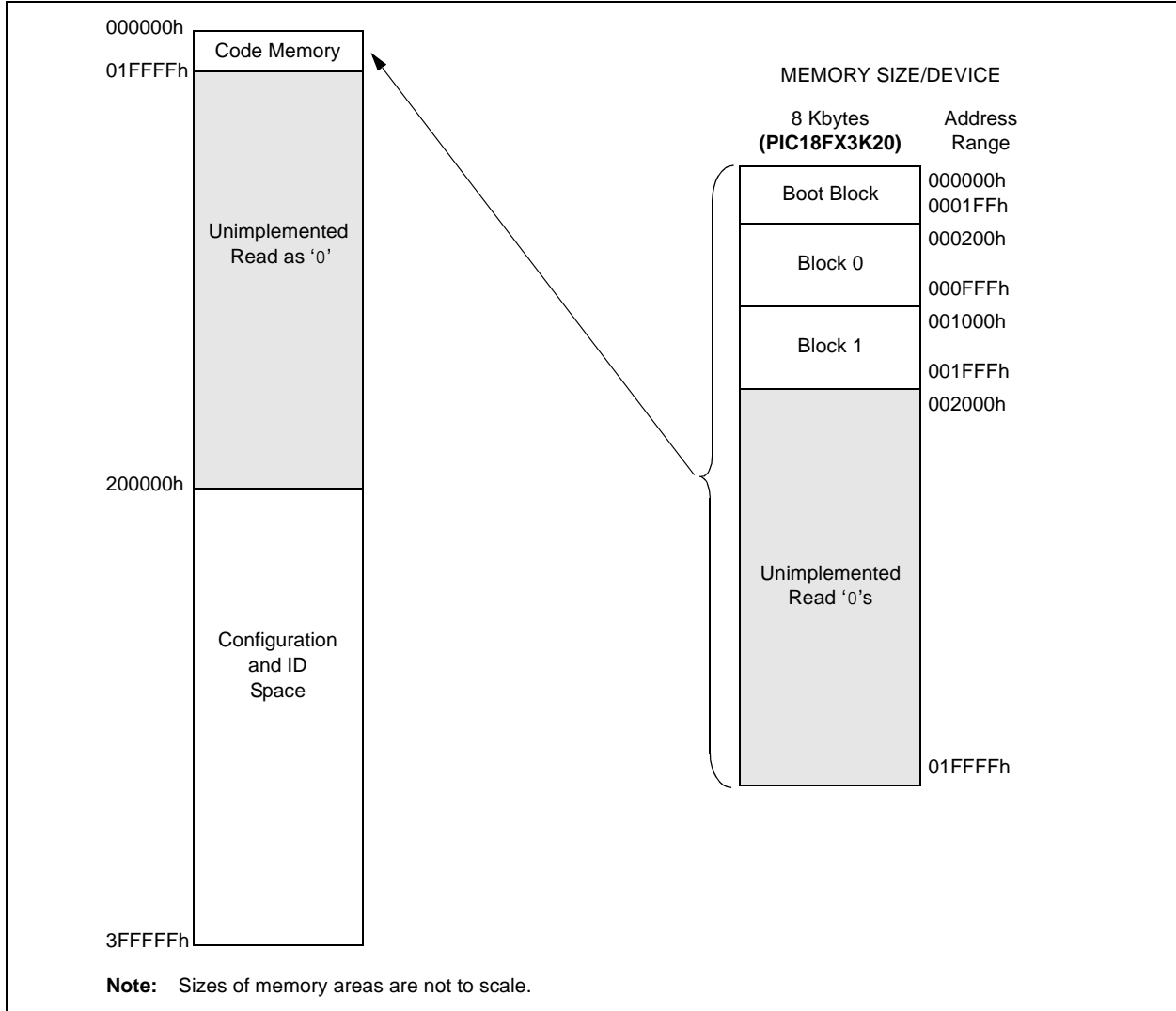
## 2.3 Memory Maps

For the PIC18FX3K20 devices, the code memory space extends from 0000h to 01FFFh (8 Kbytes) in two 4-Kbyte blocks. Addresses 0000h through 01FFFh, 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



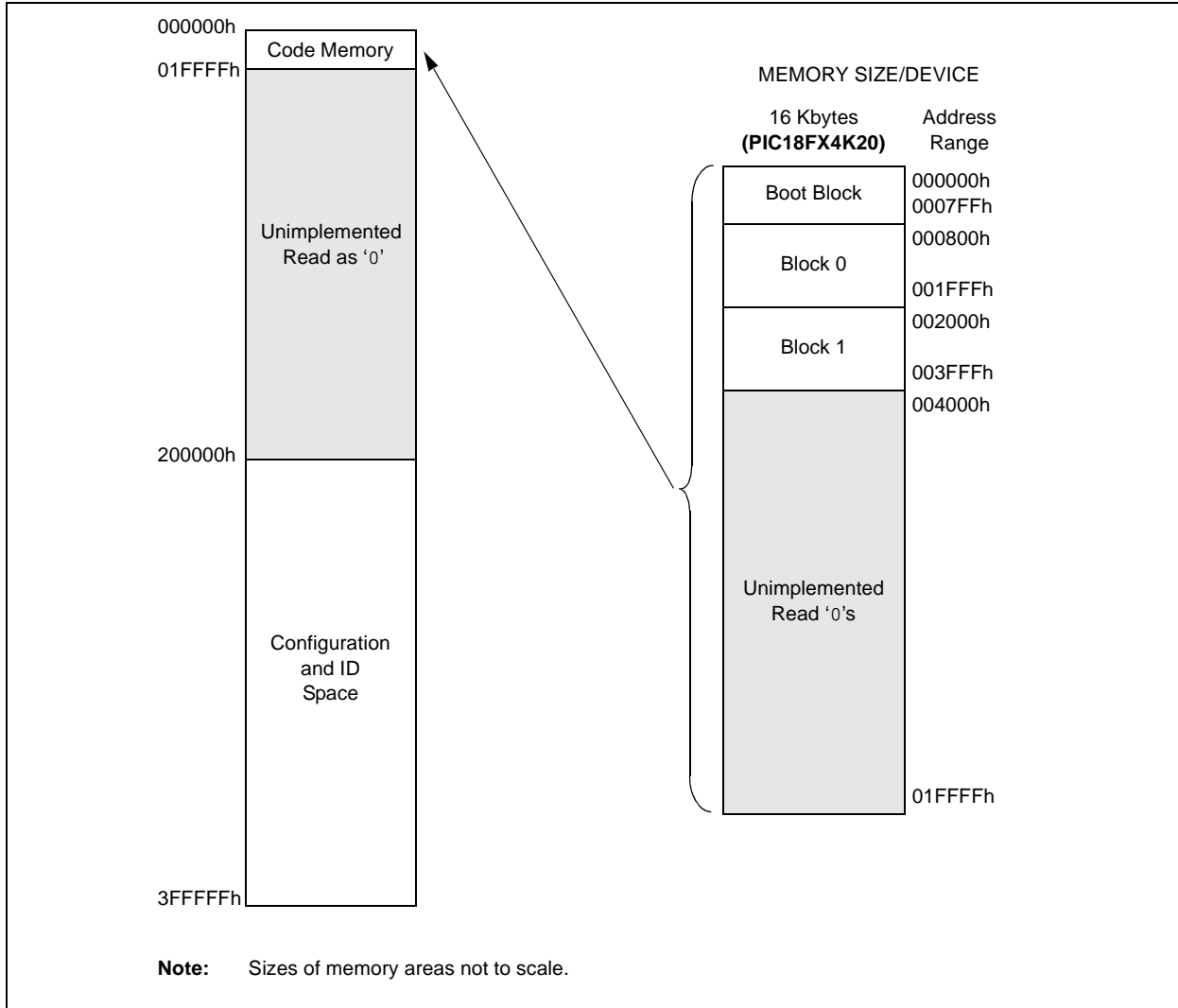
# PIC18F2XK20/4XK20

For PIC18FX4K20 devices, the code memory space extends from 000000h to 003FFFh (16 Kbytes) in two 8-Kbyte blocks. Addresses 000000h through 0007FFh, 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 2-3: IMPLEMENTATION OF CODE MEMORY**

| Device      | Code Memory Size (Bytes) |
|-------------|--------------------------|
| PIC18F24K20 | 000000h-003FFFh (16K)    |
| PIC18F44K20 |                          |

**FIGURE 2-7: MEMORY MAP AND THE CODE MEMORY SPACE FOR PIC18FX4K20 DEVICES**



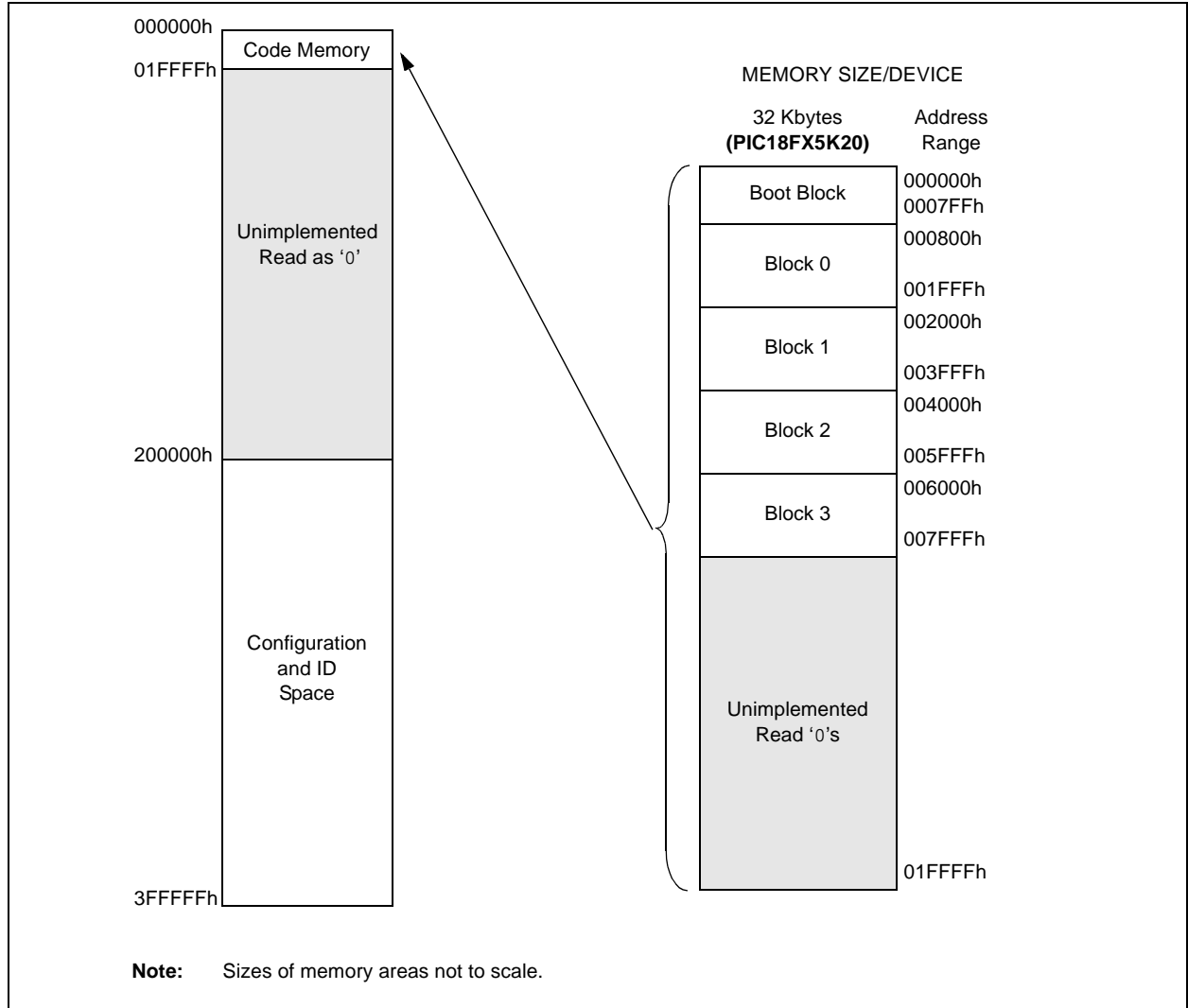
# PIC18F2XK20/4XK20

For PIC18FX5K20 devices, the code memory space extends from 000000h to 007FFFh (32 Kbytes) in four 8-Kbyte blocks. Addresses 000000h through 007FFFh, 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 2-4: IMPLEMENTATION OF CODE MEMORY**

| Device      | Code Memory Size (Bytes) |
|-------------|--------------------------|
| PIC18F25K20 | 000000h-007FFFh (32K)    |
| PIC18F45K20 |                          |

**FIGURE 2-8: MEMORY MAP AND THE CODE MEMORY SPACE FOR PIC18FX5K20 DEVICES**



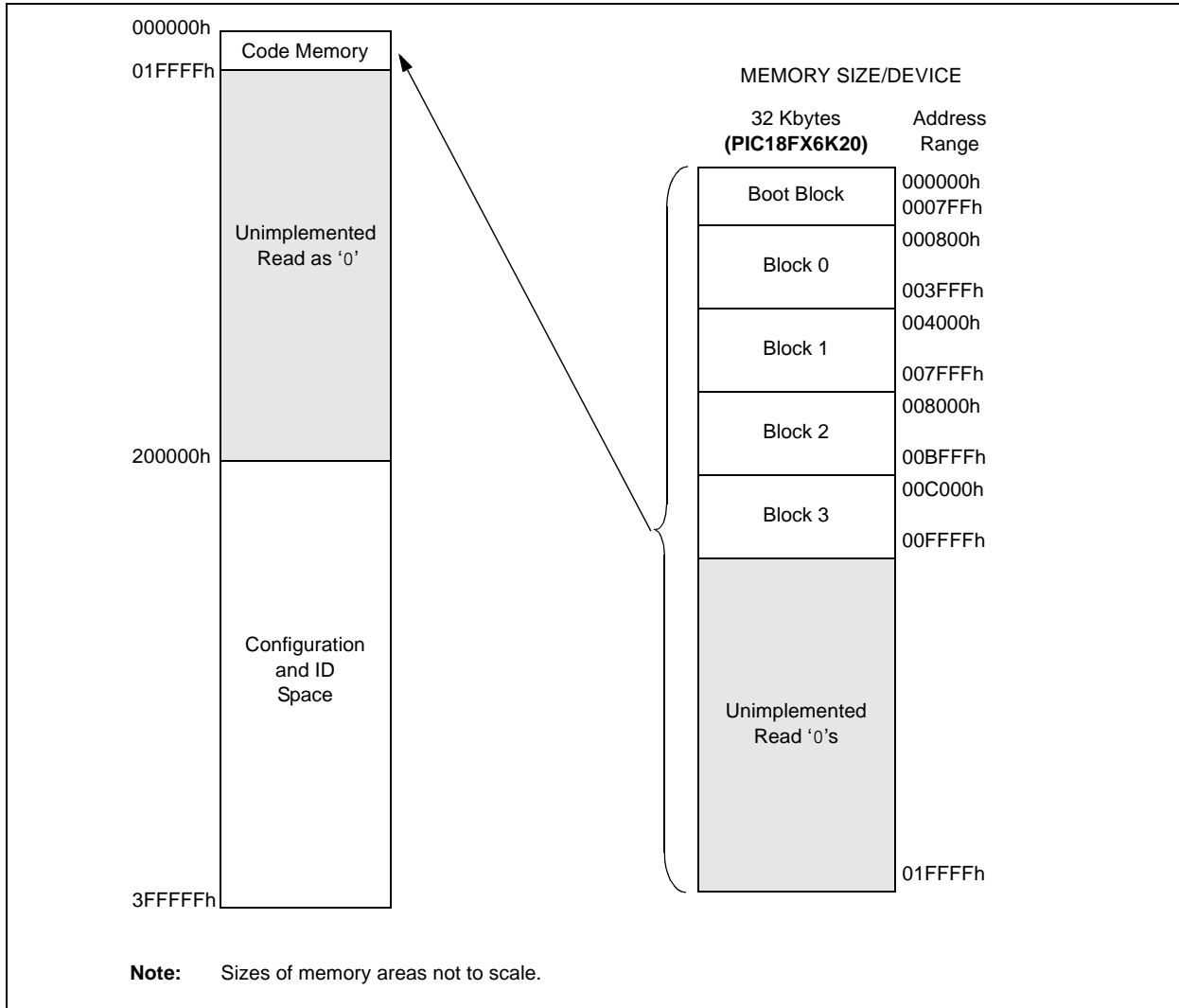
# PIC18F2XK20/4XK20

For PIC18FX6K20 devices, the code memory space extends from 000000h to 00FFFFh (64 Kbytes) in four 16-Kbyte blocks. Addresses 000000h through 0007FFh, 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 2-5: IMPLEMENTATION OF CODE MEMORY**

| Device      | Code Memory Size (Bytes) |
|-------------|--------------------------|
| PIC18F26K20 | 000000h-00FFFFh (64K)    |
| PIC18F46K20 |                          |

**FIGURE 2-9: MEMORY MAP AND THE CODE MEMORY SPACE FOR PIC18FX6K20 DEVICES**





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 3FFFEh and 3FFFFh 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

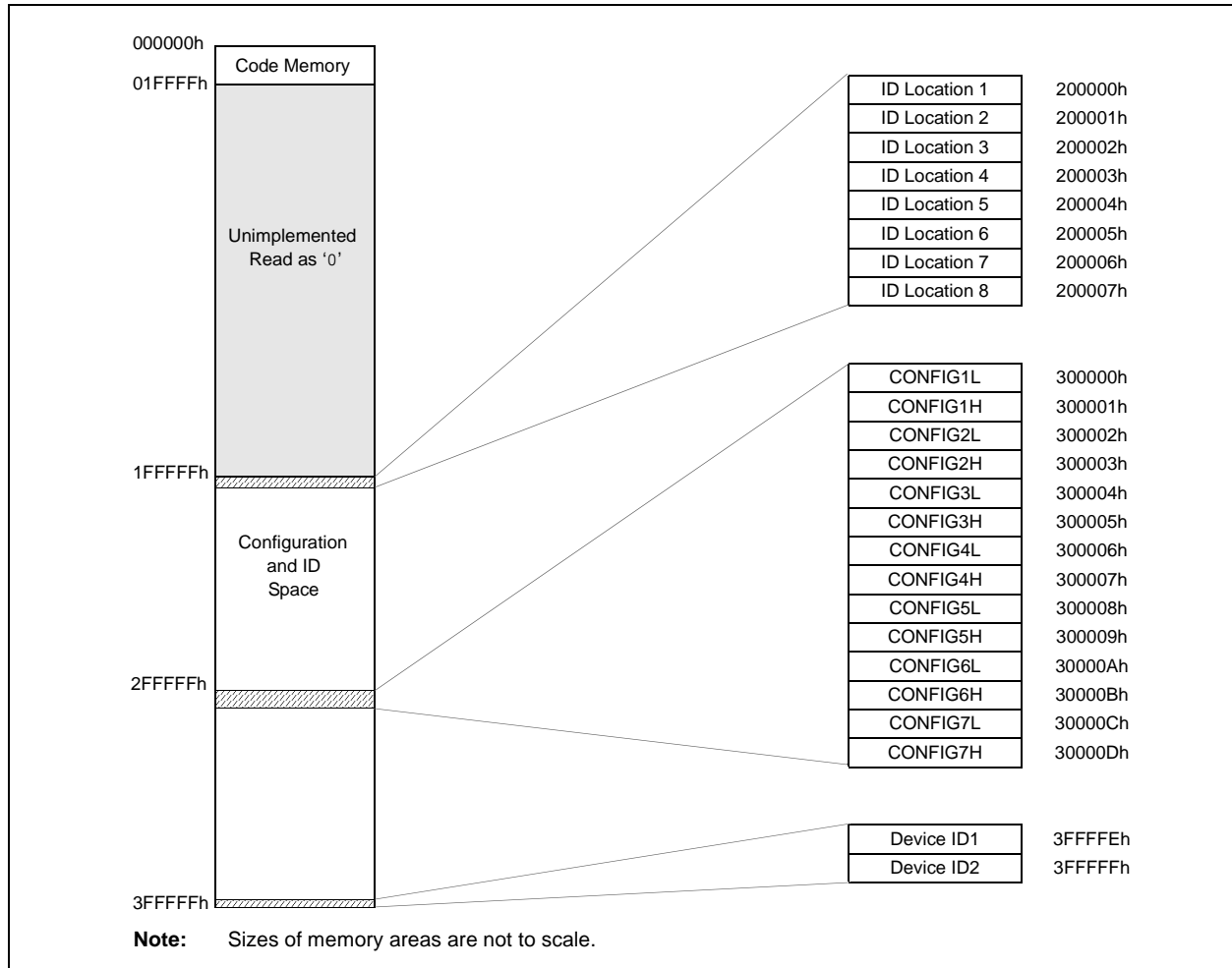
Memory in the address space, 000000h to 3FFFFFFh, is addressed via the Table Pointer register, which is comprised of three Pointer registers:

- 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**

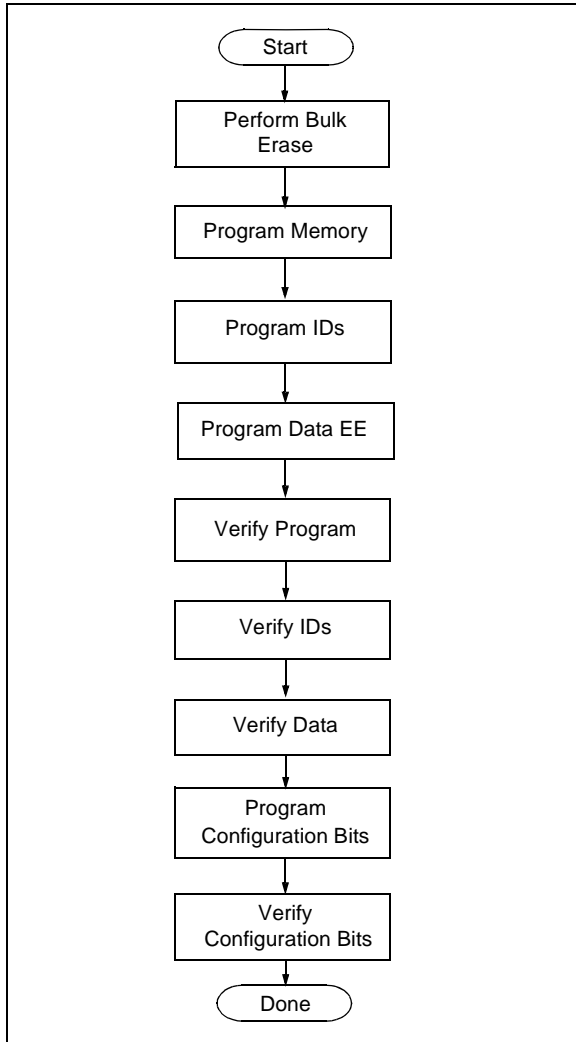


# PIC18F2XK20/4XK20

## 2.4 High-Level Overview of the Programming Process

Figure 2-11 shows the high-level overview of the programming process. First, a Bulk Erase is performed. Next, the code memory, ID locations and data EEPROM are programmed. These memories are then verified to ensure that programming was successful. If no errors are detected, the Configuration bits are then programmed and verified.

**FIGURE 2-11: HIGH-LEVEL PROGRAMMING FLOW**

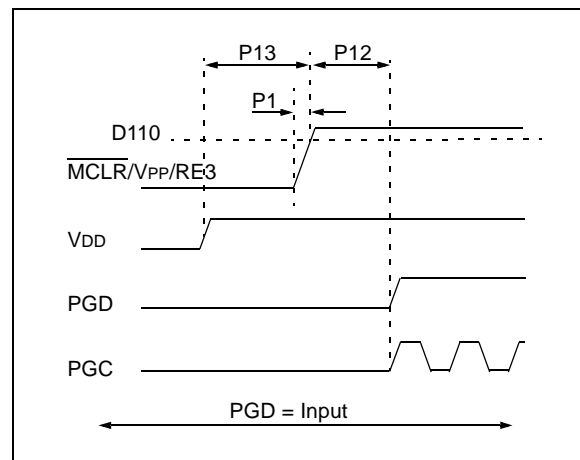


## 2.5 Entering and Exiting High-Voltage ICSP Program/Verify Mode

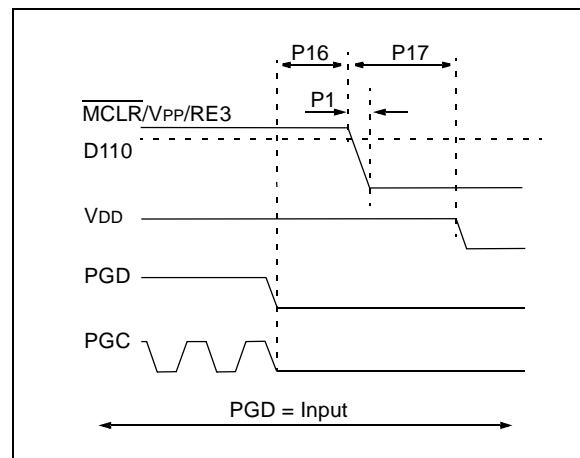
As shown in Figure 2-12, the High-Voltage ICSP Program/Verify mode is entered by holding PGC and PGD low and then raising  $\overline{\text{MCLR}}/\text{VPP}/\text{RE3}$  to  $V_{\text{IH}}\text{H}$  (high voltage). Once in this mode, the code memory, data EEPROM, ID locations and Configuration bits can be accessed and programmed in serial fashion. Figure 2-13 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 2-12: ENTERING HIGH-VOLTAGE PROGRAM/VERIFY MODE**



**FIGURE 2-13: EXITING HIGH-VOLTAGE PROGRAM/VERIFY MODE**

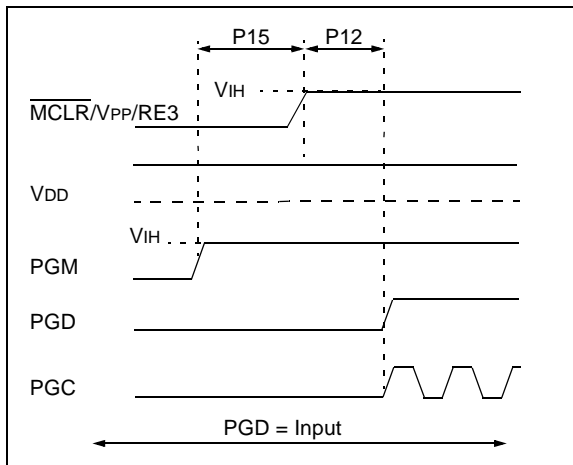


## 2.6 Entering and Exiting Low-Voltage ICSP Program/Verify Mode

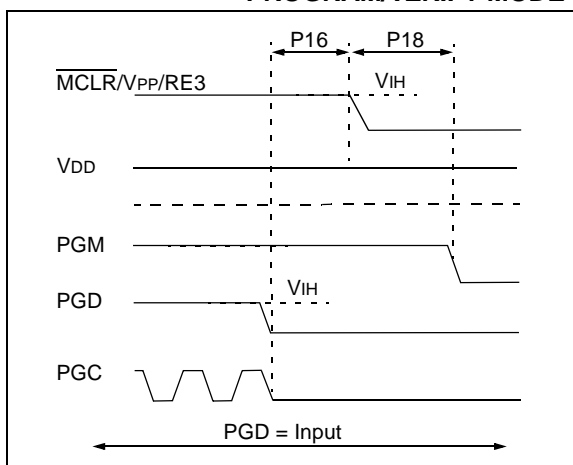
When the LVP Configuration bit is '1' (see **Section 5.3 "Single-Supply ICSP Programming"**), the Low-Voltage ICSP mode is enabled. As shown in Figure 2-14, Low-Voltage ICSP Program/Verify mode is entered by holding PGC and PGD low, placing a logic high on PGM and then raising MCLR/VPP/RE3 to  $V_{IH}$ . 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**

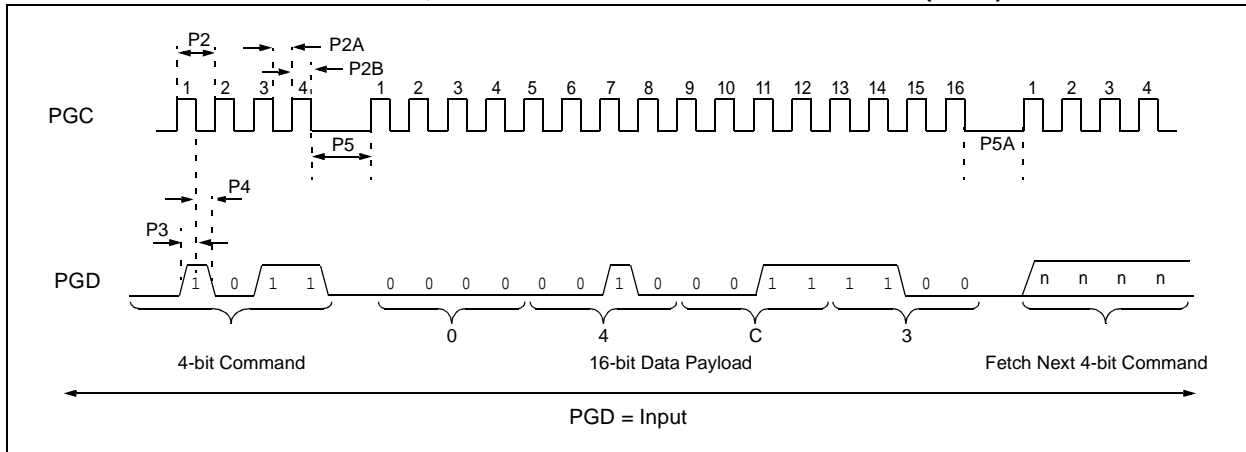
| Description   | 4-Bit Command |
|---|---------------|
| Core Instruction (Shift in 16-bit instruction)      | 0000          |
| Shift out TABLAT register                           | 0010          |
| Table Read  | 1000          |
| Table Read, post-increment                          | 1001          |
| 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          |

# PIC18F2XK20/4XK20

**TABLE 2-7: SAMPLE COMMAND SEQUENCE**

| 4-Bit Command | Data Payload | Core Instruction                 |
|---------------|--------------|----------------------------------|
| 1101          | 3C 40        | Table Write, post-increment by 2 |

**FIGURE 2-16: TABLE WRITE, POST-INCREMENT TIMING DIAGRAM (1101)**



## 3.0 DEVICE PROGRAMMING

Programming includes the ability to erase or write the various memory regions within the device.

In all cases, except high-voltage ICSP Bulk Erase, the EECON1 register must be configured in order to operate on a particular memory region.

When using the EECON1 register to act on code memory, the EEPGD bit must be set (EECON1<7> = 1) and the CFGS bit must be cleared (EECON1<6> = 0). The WREN bit must be set (EECON1<2> = 1) to enable writes of any sort (e.g., erases) and this must be done prior to initiating a write sequence. The FREE bit must be set (EECON1<4> = 1) in order to erase the program space being pointed to by the Table Pointer. The erase or write sequence is initiated by setting the WR bit (EECON1<1> = 1). It is strongly recommended that the WREN bit only be set immediately prior to a program or erase.

### 3.1 ICSP Erase

#### 3.1.1 HIGH-VOLTAGE ICSP BULK ERASE

Erasing code or data EEPROM is accomplished by configuring two Bulk Erase Control registers located at 3C0004h and 3C0005h. Code memory may be erased portions at a time, or the user may erase the entire device in one action. Bulk Erase operations will also clear any code-protect settings associated with the memory block erased. Erase options are detailed in Table 3-1. If data EEPROM is code-protected (CPD = 0), the user must request an erase of data EEPROM (e.g., 0084h as shown in Table 3-1).

**TABLE 3-1: BULK ERASE OPTIONS**

| Description               | Data (3C0005h:3C0004h) |
|---------------------------|------------------------|
| Chip Erase                | 0F8Fh                  |
| Erase User ID             | 0088h                  |
| Erase Data EEPROM         | 0084h                  |
| Erase Boot Block          | 0081h                  |
| Erase Config Bits         | 0082h                  |
| Erase Code EEPROM Block 0 | 0180h                  |
| Erase Code EEPROM Block 1 | 0280h                  |
| Erase Code EEPROM Block 2 | 0480h                  |
| Erase Code EEPROM Block 3 | 0880h                  |

The actual Bulk Erase function is a self-timed operation. Once the erase has started (falling edge of the 4th PGC after the NOP command), serial execution will cease until the erase completes (parameter P11). During this time, PGC may continue to toggle but PGD must be held low.

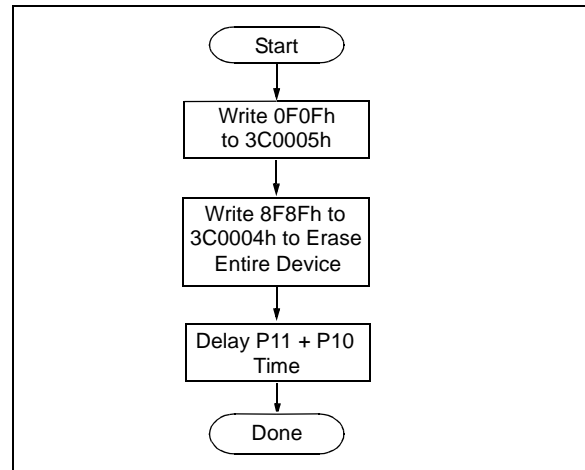
The code sequence to erase the entire device is shown in Table 3-2 and the flowchart is shown in Figure 3-1.

**Note:** A Bulk Erase is the only way to reprogram code-protect bits from an “on” state to an “off” state.

**TABLE 3-2: BULK ERASE COMMAND SEQUENCE**

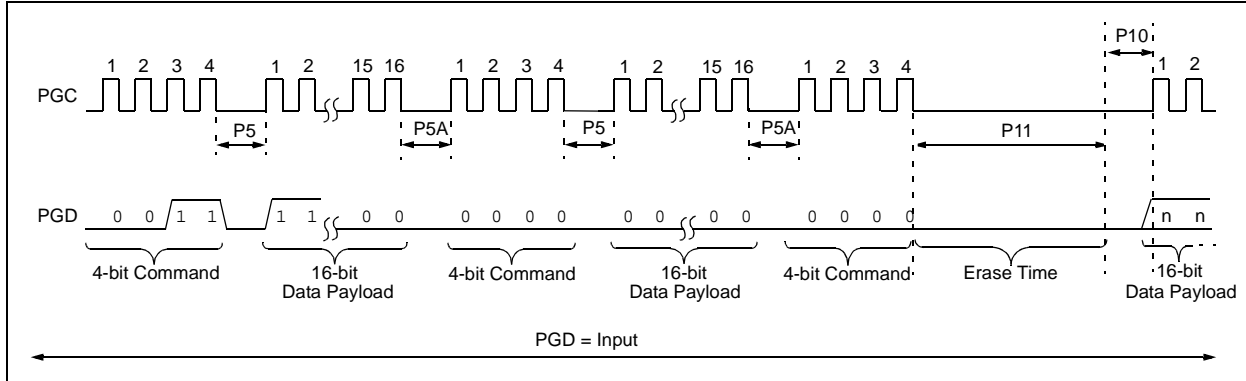
| 4-Bit Command | Data Payload | Core Instruction                               |
|---------------|--------------|--|
| 0000          | 0E 3C        | MOVLW 3Ch                                      |
| 0000          | 6E F8        | MOVWF TBLPTRU                                  |
| 0000          | 0E 00        | MOVLW 00h                                      |
| 0000          | 6E F7        | MOVWF TBLPTRH                                  |
| 0000          | 0E 05        | MOVLW 05h                                      |
| 0000          | 6E F6        | MOVWF TBLPTRL                                  |
| 1100          | 0F 0F        | Write 0Fh to 3C0005h                           |
| 0000          | 0E 3C        | MOVLW 3Ch                                      |
| 0000          | 6E F8        | MOVWF TBLPTRU                                  |
| 0000          | 0E 00        | MOVLW 00h                                      |
| 0000          | 6E F7        | MOVWF TBLPTRH                                  |
| 0000          | 0E 04        | MOVLW 04h                                      |
| 0000          | 6E F6        | MOVWF TBLPTRL                                  |
| 1100          | 8F 8F        | Write 8F8Fh TO 3C0004h to erase entire device. |
| 0000          | 00 00        | NOP  |
| 0000          | 00 00        | Hold PGD low until erase completes.            |

**FIGURE 3-1: BULK ERASE FLOW**



# PIC18F2XK20/4XK20

**FIGURE 3-2: BULK ERASE TIMING DIAGRAM**



### 3.1.2 LOW-VOLTAGE ICSP BULK ERASE

When using low-voltage ICSP, the part must be supplied by the voltage specified in parameter D111 if a Bulk Erase is to be executed. All other Bulk Erase details as described above apply.

If it is determined that a program memory erase must be performed at a supply voltage below the Bulk Erase limit, refer to the erase methodology described in **Section 3.1.3 “ICSP Row Erase”** and **Section 3.2.1 “Modifying Code Memory”**.

If it is determined that a data EEPROM erase must be performed at a supply voltage below the Bulk Erase limit, follow the methodology described in **Section 3.3 “Data EEPROM Programming”** and write ‘1’s to the array.

### 3.1.3 ICSP ROW ERASE

Regardless of whether high or low-voltage ICSP is used, it is possible to erase one row (64 bytes of data), provided the block is not code or write-protected. Rows are located at static boundaries beginning at program memory address 000000h, extending to the internal program memory limit (see **Section 2.3 “Memory Maps”**).

The Row Erase duration is self-timed. After the WR bit in EECON1 is set, two NOPs are issued. Erase starts upon the 4th PGC of the second NOP. It ends when the WR bit is cleared by hardware.

The code sequence to Row Erase a PIC18F2XK20/4XK20 device is shown in Table 3-3. The flowchart shown in Figure 3-3 depicts the logic necessary to completely erase a PIC18F2XK20/4XK20 device. The timing diagram for Row Erase is identical to the data EEPROM write timing shown in Figure 3-7.

**Note:** The TBLPTR register can point at any byte within the row intended for erase.

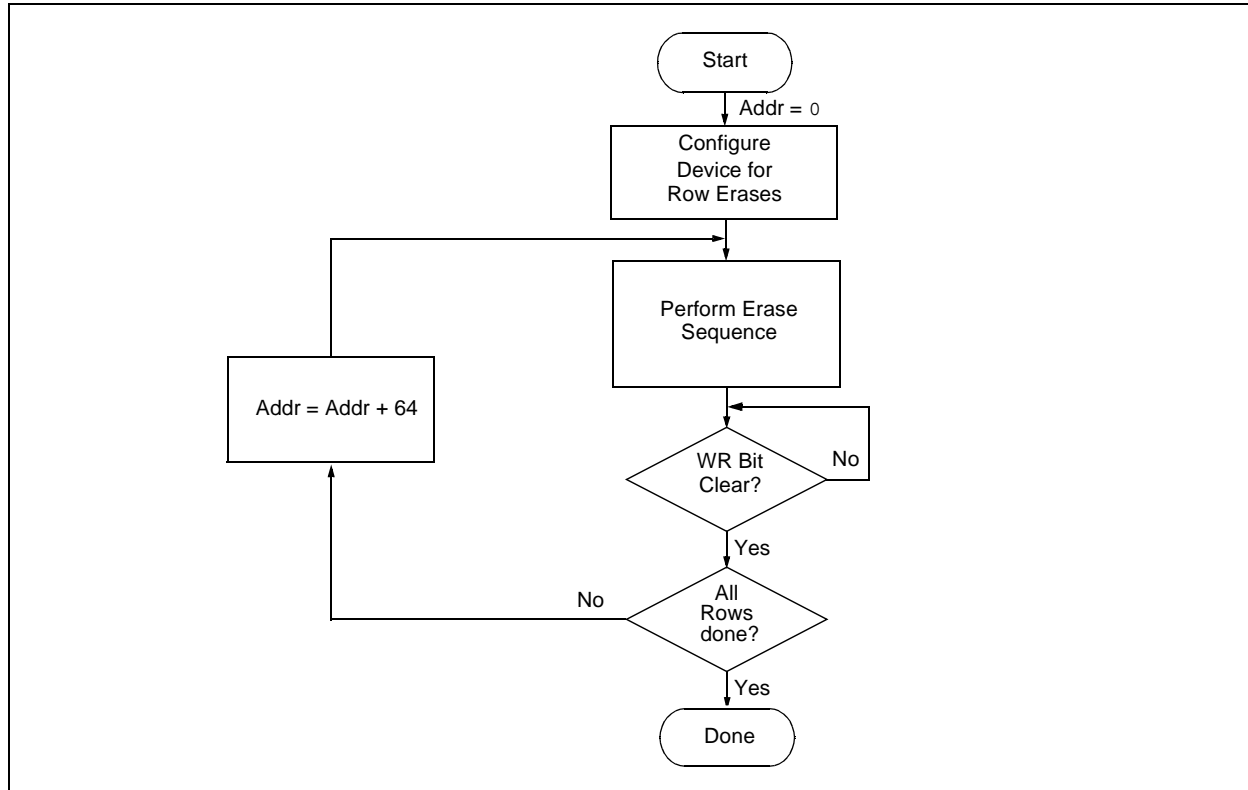
**TABLE 3-3: ERASE CODE MEMORY CODE SEQUENCE**

| 4-bit Command   | Data Payload | Core Instruction                                      |
|---|--------------|---|
| Step 1: Direct access to code memory and enable writes.                                 |              |   |
| 0000  | 8E A6        | BSF EECON1, EEPGD                                     |
| 0000  | 9C A6        | BCF EECON1, CFGS                                      |
| 0000  | 84 A6        | BSF EECON1, WREN                                      |
| Step 2: Point to first row in code memory.  |              |   |
| 0000  | 6A F8        | CLRF TBLPTRU  |
| 0000  | 6A F7        | CLRF TBLPTRH  |
| 0000  | 6A F6        | CLRF TBLPTRL  |
| Step 3: Enable erase and erase single row.  |              |   |
| 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 4: Poll WR bit. Repeat until 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 <sup>(1)</sup>                         |
| Step 5: Hold PGC low for time P10.  |              |   |
| Step 6: Repeat step 3 with Address Pointer incremented by 64 until all rows are erased. |              |   |
| Step 7: Disable writes.   |              |   |
| 0000  | 94 A6        | BCF EECON1, WREN                                      |

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

# PIC18F2XK20/4XK20

FIGURE 3-3: SINGLE ROW ERASE CODE MEMORY FLOW





## 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**

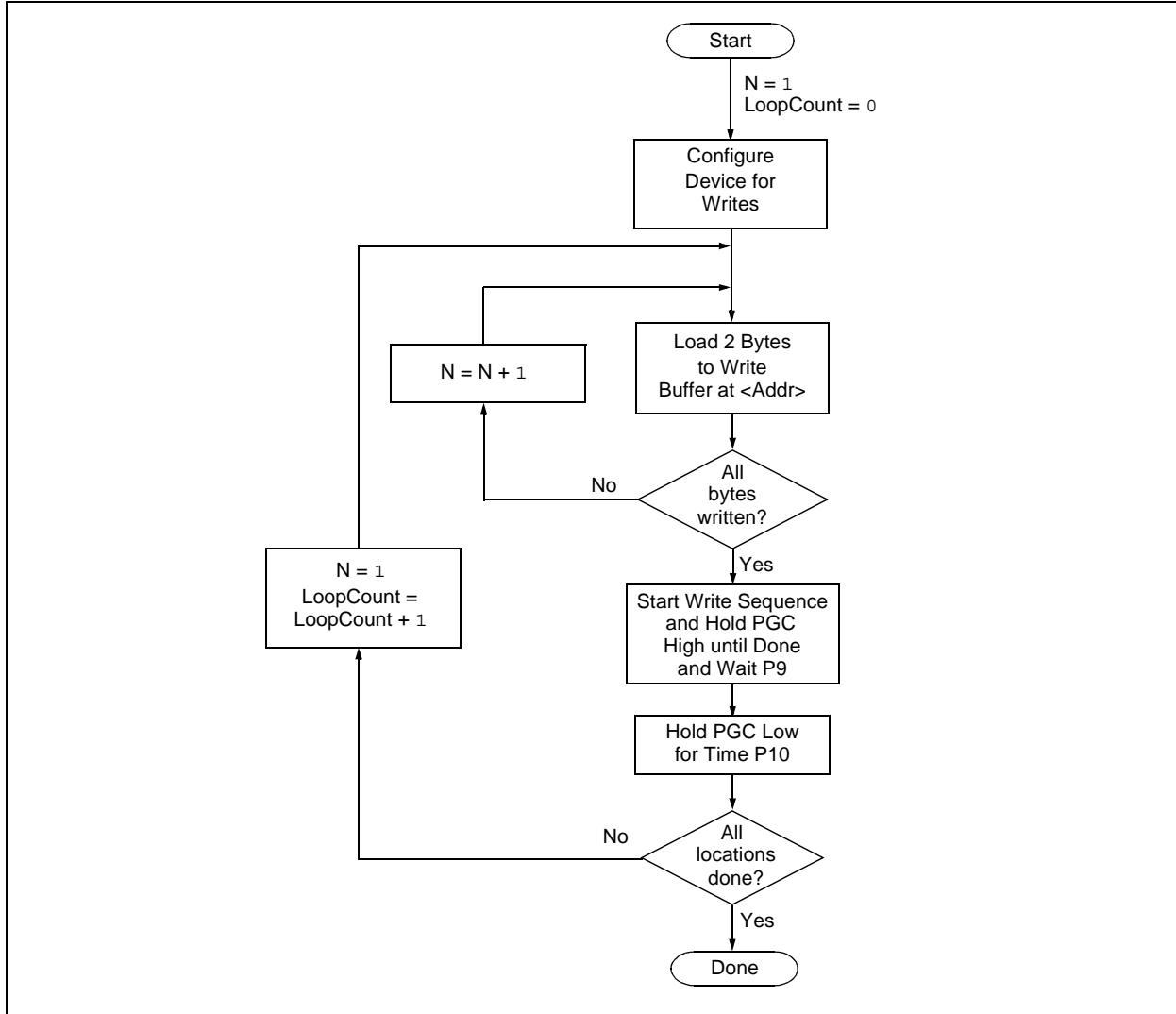
| Devices (Arranged by Family)                       | Write Buffer Size (bytes) | Erase Size (bytes) |
|--|---------------------------|--------------------|
| PIC18F26K20, PIC18F46K20                           | 64                        | 64                 |
| PIC18F24K20, PIC18F25K20, PIC18F44K20,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

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

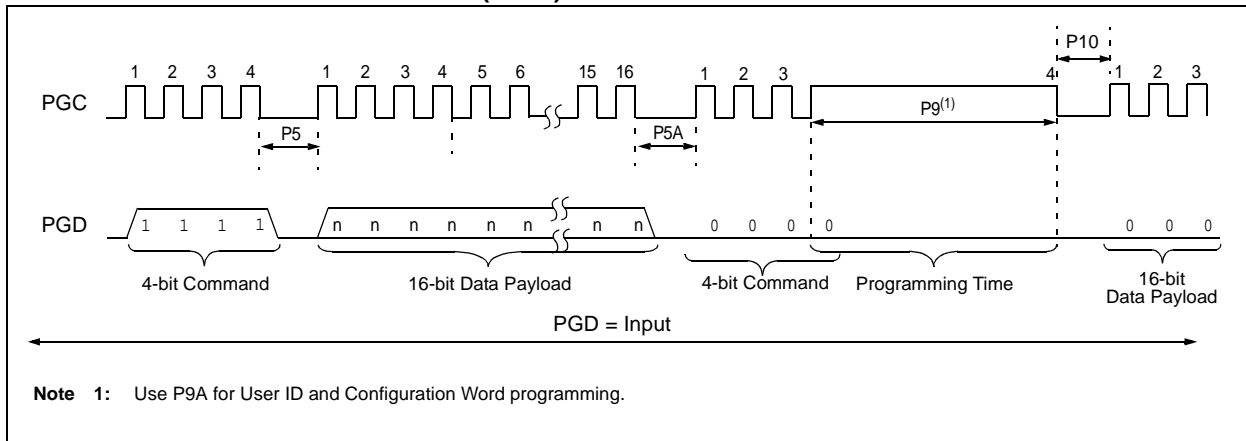
| 4-bit Command  | Data Payload     | Core Instruction                                      |
|--|------------------|---|
| Step 1: Direct access to code memory.  |                  |   |
| 0000   | 8E A6            | BSF EECON1, EEPGD                                     |
| 0000   | 9C A6            | BCF EECON1, CFGS                                      |
| 0000   | 84 A6            | BSF EECON1, WREN                                      |
| Step 2: Point to row to write.   |                  |   |
| 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 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   | <MSB><LSB>       | Write 2 bytes and start programming.                  |
| 0000   | 00 00            | 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. |                  |   |

# PIC18F2XK20/4XK20

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



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



## 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 ( <b>Section 4.1 “Read Code Memory, ID Locations and Configuration Bits”</b> ).   |                  |  |
| 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 <sup>(1)</sup>                              |
| 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   |

# PIC18F2XK20/4XK20

## 3.3 Data EEPROM Programming

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 written by loading EEADRH:EEADR with the desired memory location, EEDATA with the data to be written and initiating a memory write by appropriately configuring the EECON1 register. A byte write automatically erases the location and writes the new data (erase-before-write).

When using the EECON1 register to perform a data EEPROM write, both the EEPGD and CFGS bits must be cleared ( $EECON1\langle 7:6 \rangle = 00$ ). The WREN bit must be set ( $EECON1\langle 2 \rangle = 1$ ) to enable writes of any sort and this must be done prior to initiating a write sequence. The write sequence is initiated by setting the WR bit ( $EECON1\langle 1 \rangle = 1$ ).

The write begins on the falling edge of the 24th PGC after the WR bit is set. It ends when the WR bit is cleared by hardware.

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

FIGURE 3-6: PROGRAM DATA FLOW

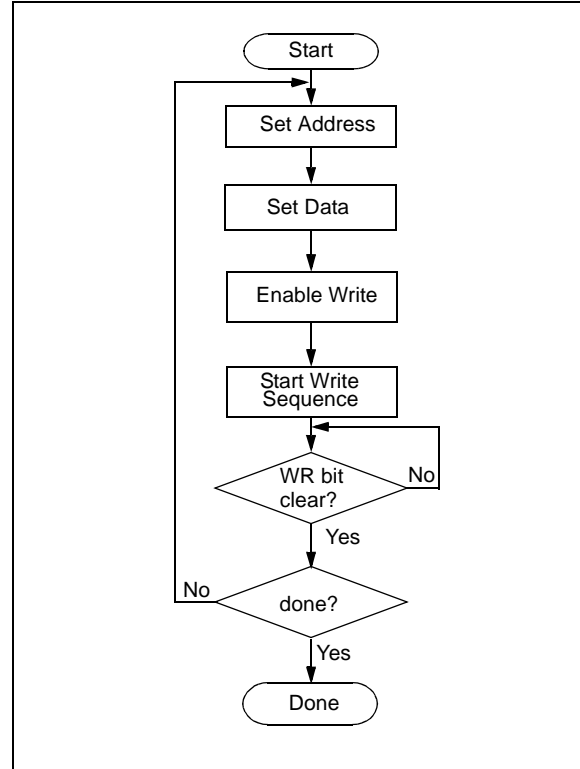
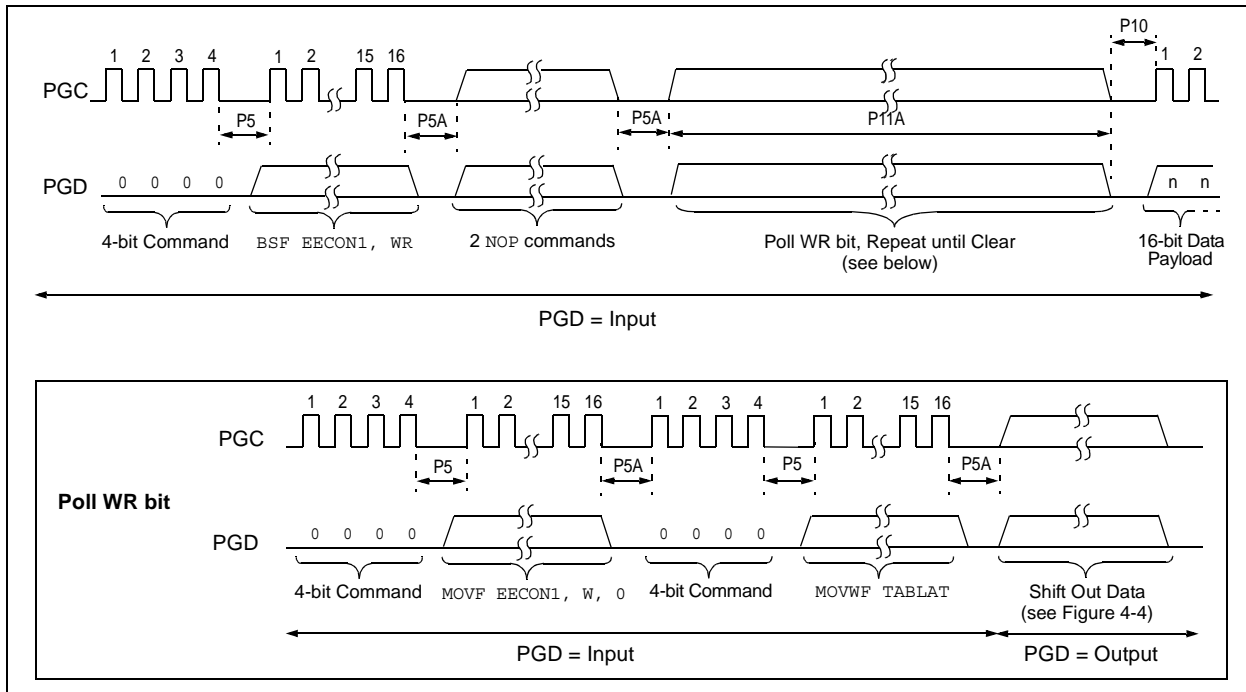


FIGURE 3-7: DATA EEPROM WRITE TIMING DIAGRAM



**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  | 0E <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 <sup>(1)</sup>                      |
| 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.

# PIC18F2XK20/4XK20

## 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.

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 demonstrates the code sequence required to write the ID locations.

**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.

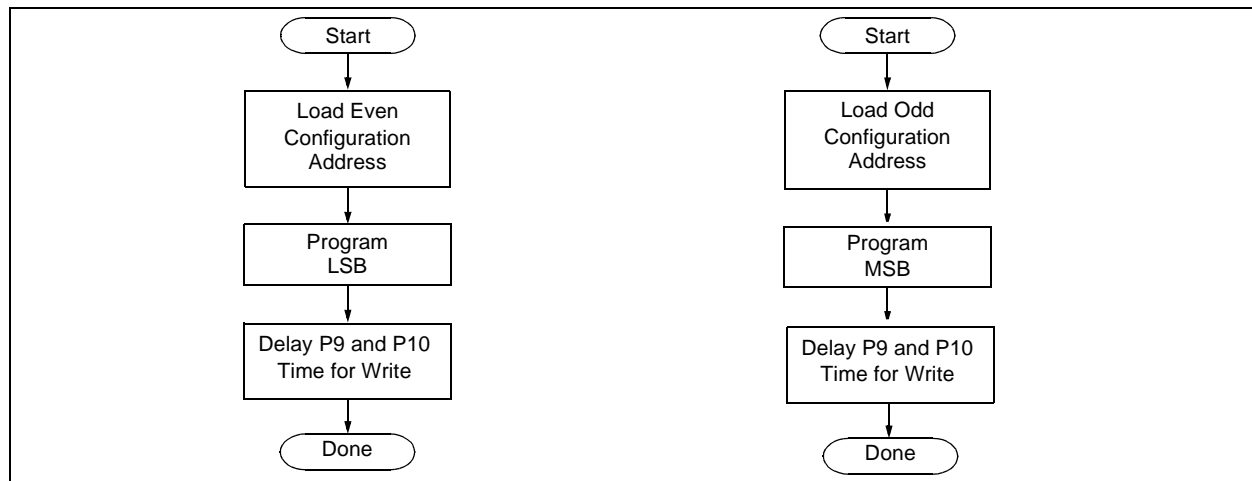
**Note:** The address must be explicitly written for each byte programmed. The addresses can not be incremented in this mode.

**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 <sup>(1)</sup> : 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**



# PIC18F2XK20/4XK20

## 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.

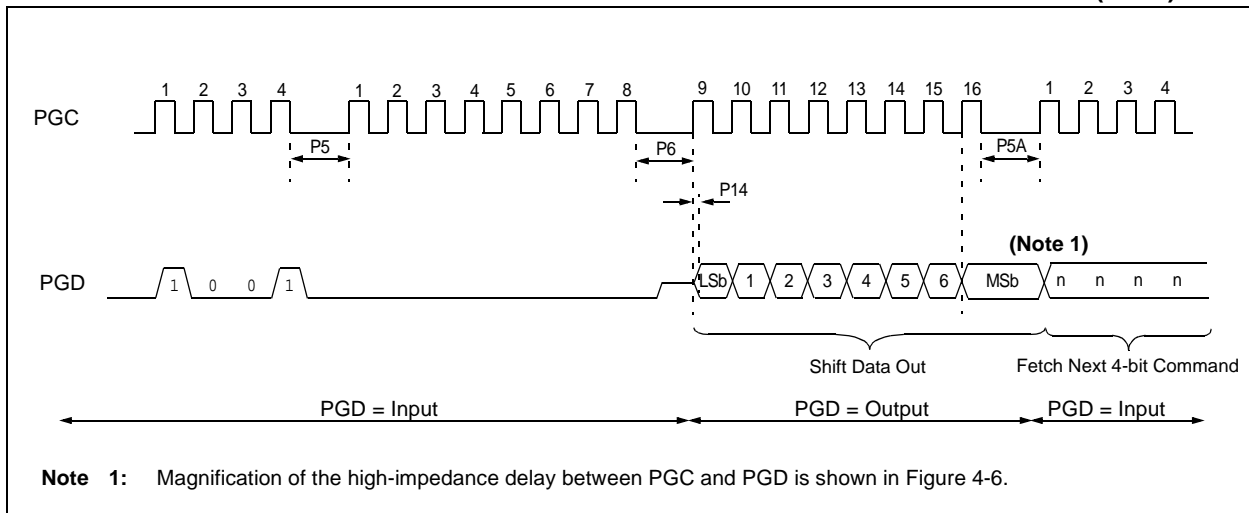
This technique will work to read any memory in the 000000h to 3FFFFFFh address space, so it also applies to the reading of the ID and Configuration registers.

**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)**



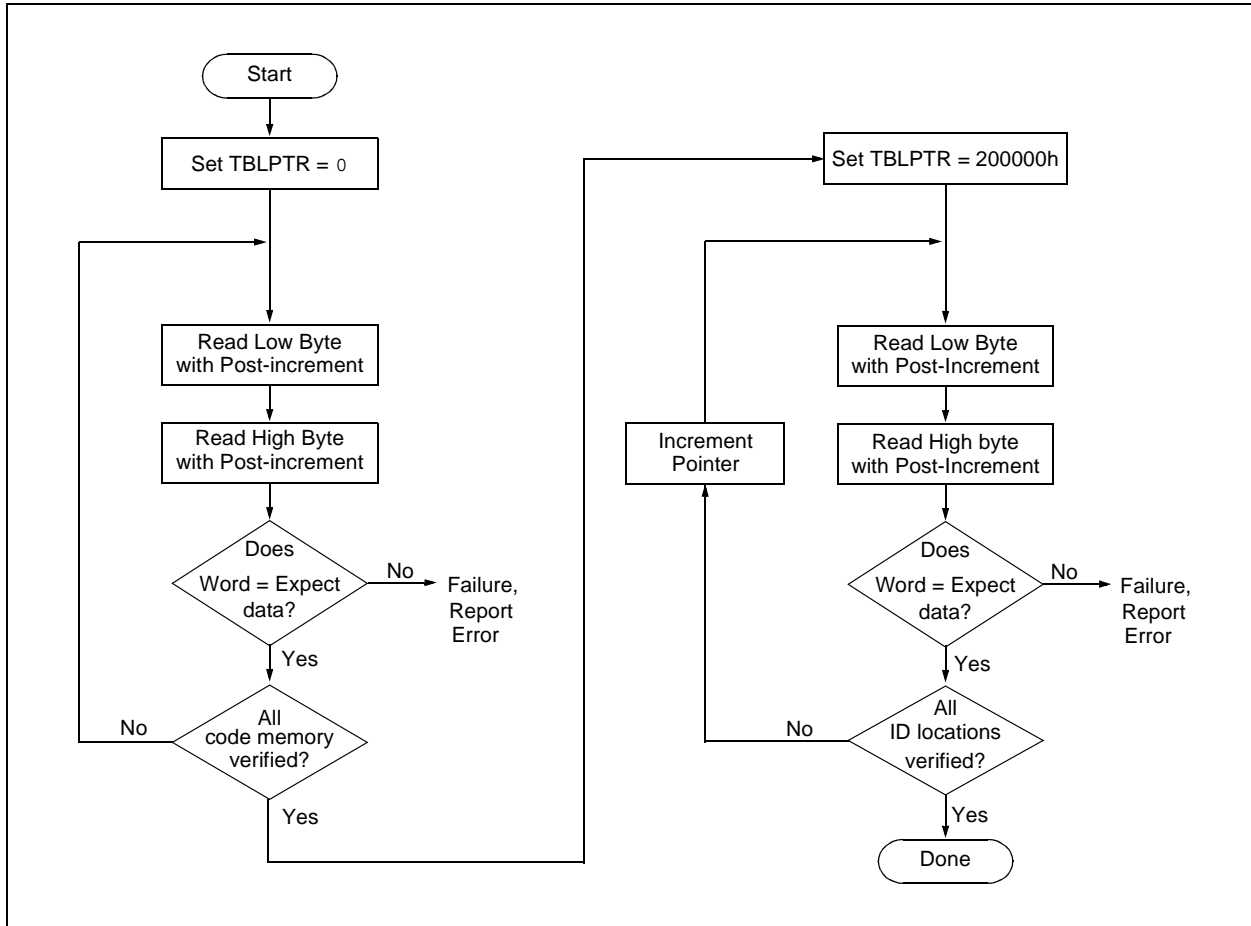


## 4.2 Verify Code Memory and ID Locations

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**



# PIC18F2XK20/4XK20

## 4.3 Verify Configuration Bits

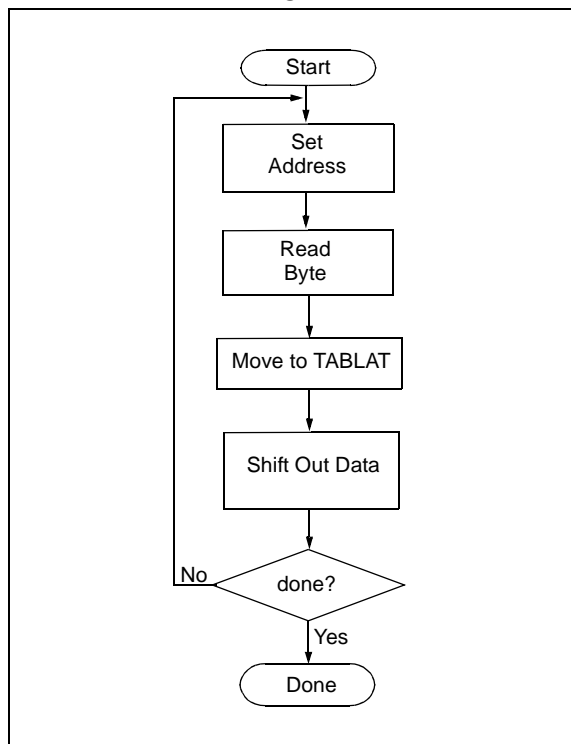
A configuration address may be read and output on PGD via the 4-bit command, '1001'. Configuration data is read and written in a byte-wise fashion, so it is not necessary to merge two bytes into a word prior to a compare. The result may then be immediately compared to the appropriate configuration data 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**

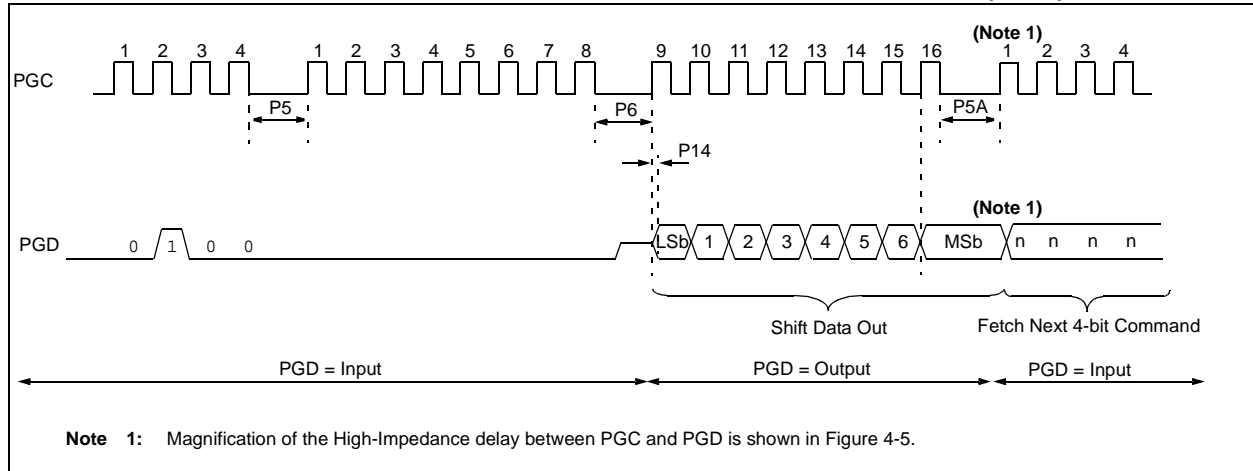


**TABLE 4-2: READ DATA EEPROM MEMORY**

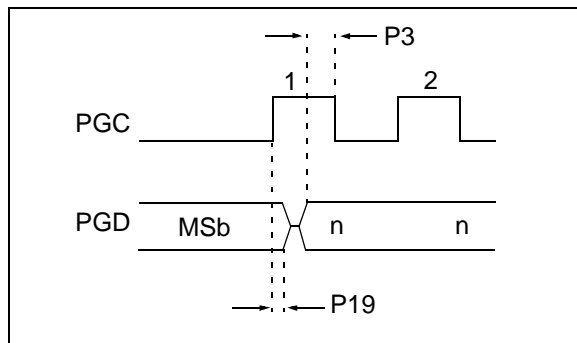
| 4-bit Command  | Data Payload | Core Instruction              |
|--|--------------|-------------------------------|
| Step 1: Direct access to data EEPROM.                    |              |                               |
| 0000   | 9E A6        | BCF EECON1, EEPGD             |
| 0000   | 9C A6        | BCF EECON1, CFGS              |
| Step 2: Set the data EEPROM Address Pointer.             |              |                               |
| 0000   | 0E <Addr>    | MOVLW <Addr>                  |
| 0000   | 6E A9        | MOVWF EEADR                   |
| 0000   | 0E <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 <sup>(1)</sup> |

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

**FIGURE 4-4: SHIFT OUT DATA HOLDING REGISTER TIMING DIAGRAM (0010)**



**FIGURE 4-5: HIGH-IMPEDANCE DELAY**



## 4.5 Verify Data EEPROM

A data EEPROM address may be read via a sequence of core instructions (4-bit command, '0000') and then output on PGD via the 4-bit command, '0010' (TABLAT register). The result may then be immediately compared to the appropriate data in the programmer's memory for verification. Refer to **Section 4.4 "Read Data EEPROM Memory"** for implementation details of reading data EEPROM.

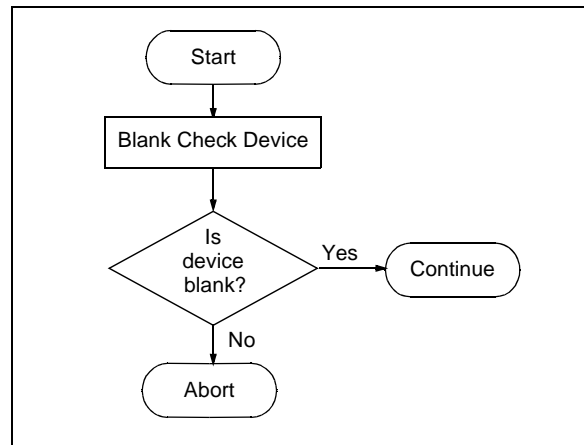
## 4.6 Blank Check

The term "Blank Check" means to verify that the device has no programmed memory cells. All memories must be verified: code memory, data EEPROM, ID locations and Configuration bits. The device ID registers (3FFFEh:3FFFFh) should be ignored.

A "blank" or "erased" memory cell will read as a '1'. Therefore, Blank Checking a device merely means to verify that all bytes read as FFh except the Configuration bits. Unused (reserved) Configuration bits will read '0' (programmed). Refer to Table 5-1 for blank configuration expect data for the various PIC18F2XK20/4XK20 devices.

Given that Blank Checking is merely code and data EEPROM verification with FFh expect data, refer to **Section 4.4 "Read Data EEPROM Memory"** and **Section 4.2 "Verify Code Memory and ID Locations"** for implementation details.

**FIGURE 4-6: BLANK CHECK FLOW**



# PIC18F2XK20/4XK20

## 5.0 CONFIGURATION WORD

The PIC18F2XK20/4XK20 devices have several Configuration Words. These bits can be set or cleared to select various device configurations. All other memory areas should be programmed and verified prior to setting Configuration Words. These bits may be read out normally, even after read or code protection. See Table 5-1 for a list of Configuration bits and device IDs and Table 5-3 for the Configuration bit descriptions.

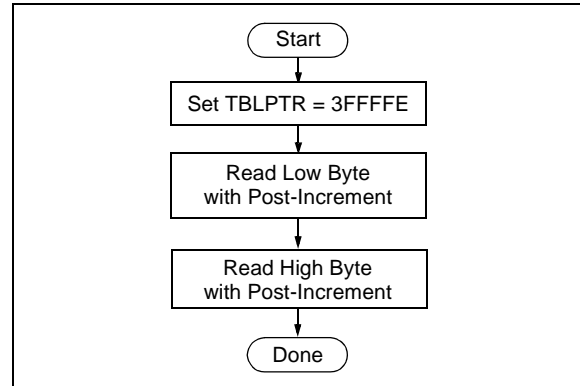
### 5.1 User ID Locations

A user may store identification information (ID) in eight ID locations mapped in 200000h:200007h. It is recommended that the Most Significant nibble of each ID be Fh. In doing so, if the user code inadvertently tries to execute from the ID space, the ID data will execute as a NOP.

## 5.2 Device ID Word

The device ID word for the PIC18F2XK20/4XK20 devices is located at 3FFFFEh:3FFFFFh. These bits may be used by the programmer to identify what device type is being programmed and read out normally, even after code or read protection. See Table 5-2 for a complete list of device ID values.

**FIGURE 5-1: READ DEVICE ID WORD FLOW**



**TABLE 5-1: CONFIGURATION BITS AND DEVICE IDs**

| File Name | Bit 7                 | Bit 6 | Bit 5 | Bit 4 | Bit 3  | Bit 2                | Bit 1                | Bit 0  | Default/<br>Unprogrammed Value |               |
|-----------|-----------------------|-------|-------|-------|--------|----------------------|----------------------|--------|--------------------------------|---------------|
| 300001h   | CONFIG1H              | IESO  | FCMEN | —     | —      | FOSC3                | FOSC2                | FOSC1  | FOSC0                          | 00-- 0111     |
| 300002h   | CONFIG2L              | —     | —     | —     | BORV1  | BORV0                | BOREN1               | BOREN0 | PWRTEN                         | ---1 1111     |
| 300003h   | CONFIG2H              | —     | —     | —     | WDTPS3 | WDTPS2               | WDTPS1               | WDTPS0 | WDTEN                          | ---1 1111     |
| 300005h   | CONFIG3H              | MCLRE | —     | —     | —      | HFOFST               | LPT1OSC              | PBADEN | CCP2MX                         | 1--- 1011     |
| 300006h   | CONFIG4L              | DEBUG | XINST | —     | —      | —                    | LVP                  | —      | STVREN                         | 10-- -1-1     |
| 300008h   | CONFIG5L              | —     | —     | —     | —      | CP3 <sup>(1)</sup>   | CP2 <sup>(1)</sup>   | CP1    | CP0                            | ---- 1111     |
| 300009h   | CONFIG5H              | CPD   | CPB   | —     | —      | —                    | —                    | —      | —                              | 11-- ----     |
| 30000Ah   | CONFIG6L              | —     | —     | —     | —      | WRT3 <sup>(1)</sup>  | WRT2 <sup>(1)</sup>  | WRT1   | WRT0                           | ---- 1111     |
| 30000Bh   | CONFIG6H              | WRD   | WRTB  | WRTC  | —      | —                    | —                    | —      | —                              | 111- ----     |
| 30000Ch   | CONFIG7L              | —     | —     | —     | —      | EBTR3 <sup>(1)</sup> | EBTR2 <sup>(1)</sup> | EBTR1  | EBTR0                          | ---- 1111     |
| 30000Dh   | CONFIG7H              | —     | EBTRB | —     | —      | —                    | —                    | —      | —                              | -1-- ----     |
| 3FFFFEh   | DEVID1 <sup>(2)</sup> | DEV2  | DEV1  | DEV0  | REV4   | REV3                 | REV2                 | REV1   | REV0                           | See Table 5-2 |
| 3FFFFFh   | DEVID2 <sup>(2)</sup> | DEV10 | DEV9  | DEV8  | DEV7   | DEV6                 | DEV5                 | DEV4   | DEV3                           | See Table 5-2 |

**Legend:** x = unknown, u = unchanged, — = unimplemented. Shaded cells are unimplemented, read as '0'.

**Note 1:** These bits are only implemented on specific devices. Refer to **Section 2.3 “Memory Maps”** to determine which bits apply based on available memory.

**2:** DEVID registers are read-only and cannot be programmed by the user.

# PIC18F2XK20/4XK20

**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.

# PIC18F2XK20/4XK20

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

| Bit Name   | Configuration Words | Description  |
|------------|---------------------|--|
| IESO       | CONFIG1H            | Internal External Switchover bit<br>1 = Internal External Switchover mode enabled<br>0 = Internal External Switchover mode disabled  |
| FCMEN      | CONFIG1H            | Fail-Safe Clock Monitor Enable bit<br>1 = Fail-Safe Clock Monitor enabled<br>0 = Fail-Safe Clock Monitor disabled  |
| FOSC<3:0>  | CONFIG1H            | Oscillator Selection bits<br>11xx = External RC oscillator, CLKOUT function on RA6<br>101x = External RC oscillator, CLKOUT function on RA6<br>1001 = HFINTOSC, CLKOUT function on RA6, port function on RA7<br>1000 = HFINTOSC, port function on RA6, port function on RA7<br>0111 = External RC oscillator, port function on RA6<br>0110 = HS oscillator, PLL enabled (clock frequency = 4 x FOSC1)<br>0101 = EC oscillator, port function on RA6<br>0100 = EC oscillator, CLKOUT function on RA6<br>0011 = External RC oscillator, CLKOUT function on RA6<br>0010 = HS oscillator<br>0001 = XT oscillator<br>0000 = LP oscillator |
| BORV<1:0>  | CONFIG2L            | Brown-out Reset Voltage bits<br>11 = VBOR set to 1.8V<br>10 = VBOR set to 2.2V<br>01 = VBOR set to 2.7V<br>00 = VBOR set to 3.0V   |
| BOREN<1:0> | CONFIG2L            | Brown-out Reset Enable bits<br>11 = Brown-out Reset enabled in hardware only (SBOREN is disabled)<br>10 = Brown-out Reset enabled in hardware only and disabled in Sleep mode (SBOREN is disabled)<br>01 = Brown-out Reset enabled and controlled by software (SBOREN is enabled)<br>00 = Brown-out Reset disabled in hardware and software  |
| PWRTEN     | CONFIG2L            | Power-up Timer Enable bit<br>1 = PWRT disabled<br>0 = PWRT enabled   |
| WDPS<3:0>  | CONFIG2H            | Watchdog Timer Postscaler Select bits<br>1111 = 1:32,768<br>1110 = 1:16,384<br>1101 = 1:8,192<br>1100 = 1:4,096<br>1011 = 1:2,048<br>1010 = 1:1,024<br>1001 = 1:512<br>1000 = 1:256<br>0111 = 1:128<br>0110 = 1:64<br>0101 = 1:32<br>0100 = 1:16<br>0011 = 1:8<br>0010 = 1:4<br>0001 = 1:2<br>0000 = 1:1   |

# PIC18F2XK20/4XK20

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

| Bit Name | Configuration Words | Description  |
|----------|---------------------|--|
| WDTEN    | CONFIG2H            | Watchdog Timer Enable bit<br>1 = WDT enabled<br>0 = WDT disabled (control is placed on SWDTEN bit)   |
| MCLRE    | CONFIG3H            | MCLR Pin Enable bit<br>1 = MCLR pin enabled, RE3 input pin disabled<br>0 = RE3 input pin enabled, MCLR pin disabled  |
| HFOFST   | CONFIG3H            | HFINTOSC Fast Start<br>1 = HFINTOSC output is not delayed<br>0 = HFINTOSC output is delayed until oscillator is stable (IOFS = 1)  |
| LPT1OSC  | CONFIG3H            | Low-Power Timer1 Oscillator Enable bit<br>1 = Timer1 configured for low-power operation<br>0 = Timer1 configured for higher power operation  |
| PBADEN   | CONFIG3H            | PORTB A/D Enable bit<br>1 = PORTB A/D<4:0> pins are configured as analog input channels on Reset<br>0 = PORTB A/D<4:0> pins are configured as digital I/O on Reset                                       |
| CCP2MX   | CONFIG3H            | CCP2 MUX bit<br>1 = CCP2 input/output is multiplexed with RC1<br>0 = CCP2 input/output is multiplexed with RB3   |
| DEBUG    | CONFIG4L            | Background Debugger Enable bit<br>1 = Background debugger disabled, RB6 and RB7 configured as general purpose I/O pins<br>0 = Background debugger enabled, RB6 and RB7 are dedicated to In-Circuit Debug |
| XINST    | CONFIG4L            | Extended Instruction Set Enable bit<br>1 = Instruction set extension and Indexed Addressing mode enabled<br>0 = Instruction set extension and Indexed Addressing mode disabled (Legacy mode)             |
| LVP      | CONFIG4L            | Low-Voltage Programming Enable bit<br>1 = Low-Voltage Programming enabled, RB5 is the PGM pin<br>0 = Low-Voltage Programming disabled, RB5 is an I/O pin   |
| STVREN   | CONFIG4L            | Stack Overflow/Underflow Reset Enable bit<br>1 = Reset on stack overflow/underflow enabled<br>0 = Reset on stack overflow/underflow disabled   |

# PIC18F2XK20/4XK20

**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 |



# PIC18F2XK20/4XK20

**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.  |

# PIC18F2XK20/4XK20

## 5.3 Single-Supply ICSP Programming

The LVP bit in Configuration register, CONFIG4L, enables Single-Supply (Low-Voltage) ICSP Programming. The LVP bit defaults to a '1' (enabled) from the factory.

If Single-Supply Programming mode is not used, the LVP bit can be programmed to a '0' and RB5/PGM becomes a digital I/O pin. However, the LVP bit may only be programmed by entering the High-Voltage ICSP mode, where MCLR/VPP/RE3 is raised to  $V_{IH}$ . Once the LVP bit is programmed to a '0', only the High-Voltage ICSP mode is available and only the High-Voltage ICSP mode can be used to program the device.

**Note 1:** The High-Voltage ICSP mode is always available, regardless of the state of the LVP bit, by applying  $V_{IH}$  to the MCLR/VPP/RE3 pin.

**2:** While in Low-Voltage ICSP mode, the RB5 pin can no longer be used as a general purpose I/O.

## 5.4 Embedding Configuration Word Information in the HEX File

To allow portability of code, a PIC18F2XK20/4XK20 programmer is required to read the Configuration Word locations from the hex file. If Configuration Word information is not present in the hex file, then a simple warning message should be issued. Similarly, while saving a hex file, all Configuration Word information must be included. An option to not include the Configuration Word information may be provided. When embedding Configuration Word information in the hex file, it should start at address 300000h.

Microchip Technology Inc. feels strongly that this feature is important for the benefit of the end customer.

## 5.5 Embedding Data EEPROM Information In the HEX File

To allow portability of code, a PIC18F2XK20/4XK20 programmer is required to read the data EEPROM information from the hex file. If data EEPROM information is not present, a simple warning message should be issued. Similarly, when saving a hex file, all data EEPROM information must be included. An option to not include the data EEPROM information may be provided. When embedding data EEPROM information in the hex file, it should start at address F00000h.

Microchip Technology Inc. believes that this feature is important for the benefit of the end customer.

## 5.6 Checksum Computation

The checksum is calculated by summing the following:

- The contents of all code memory locations
- The Configuration Word, appropriately masked
- ID locations (Only if any portion of program memory is code-protected)

The Least Significant 16 bits of this sum are the checksum.

Code protection limits access to program memory by both external programmer (code-protect) and code execution (table read protect). The ID locations, when included in a code protected checksum, contain the checksum of an unprotected part. The unprotected checksum is distributed: one nibble per ID location. Each nibble is right justified.

Table 5-4 describes how to calculate the checksum for each device.

**Note:** The checksum calculation differs depending on the code-protect setting. Since the code memory locations read out differently depending on the code-protect setting, the table describes how to manipulate the actual code memory values to simulate the values that would be read from a protected device. When calculating a checksum by reading a device, the entire code memory can simply be read and summed. The Configuration Word and ID locations can always be read.

# PIC18F2XK20/4XK20

**TABLE 5-4: CHECKSUM COMPUTATION**

| Device      | Code-Protect | Checksum  | Blank Value | 0xAA at 0 and Max Address |
|-------------|--------------|---|-------------|---------------------------|
| PIC18FX3K20 | None         | SUM[0000:01FF]+SUM[0200:0FFF]+SUM[1000:1FFF]+(CONFIG1L & 00h)+(CONFIG1H & CFh)+(CONFIG2L & 1Fh)+(CONFIG2H & 1F)+(CONFIG3L & 00h)+(CONFIG3H & 8Fh)+(CONFIG4L & C5h)+(CONFIG4H & 00h)+(CONFIG5L & 03h)+(CONFIG5H & C0h)+(CONFIG6L & 03h)+(CONFIG6H & E0h)+(CONFIG7L & 03h)+(CONFIG7H & 40h) | E33Eh       | E294h                     |
|             | Boot Block   | SUM[0200:0FFF]+SUM[1000:1FFF]+(CONFIG1L & 00h)+(CONFIG1H & CFh)+(CONFIG2L & 1Fh)+(CONFIG2H & 1F)+(CONFIG3L & 00h)+(CONFIG3H & 8Fh)+(CONFIG4L & C5h)+(CONFIG4H & 00h)+(CONFIG5L & 03h)+(CONFIG5H & C0h)+(CONFIG6L & 03h)+(CONFIG6H & E0h)+(CONFIG7L & 03h)+(CONFIG7H & 40h)+SUM_ID         | E520h       | E4C6h                     |
|             | Boot/Block 0 | SUM[1000:1FFF]+(CONFIG1L & 00h)+(CONFIG1H & CFh)+(CONFIG2L & 1Fh)+(CONFIG2H & 1F)+(CONFIG3L & 00h)+(CONFIG3H & 8Fh)+(CONFIG4L & C5h)+(CONFIG4H & 00h)+(CONFIG5L & 03h)+(CONFIG5H & C0h)+(CONFIG6L & 03h)+(CONFIG6H & E0h)+(CONFIG7L & 03h)+(CONFIG7H & 40h)+SUM_ID                        | F31Fh       | F2C5h                     |
|             | All          | (CONFIG1L & 00h)+(CONFIG1H & CFh)+(CONFIG2L & 1Fh)+(CONFIG2H & 1F)+(CONFIG3L & 00h)+(CONFIG3H & 8Fh)+(CONFIG4L & C5h)+(CONFIG4H & 00h)+(CONFIG5L & 03h)+(CONFIG5H & C0h)+(CONFIG6L & 03h)+(CONFIG6H & E0h)+(CONFIG7L & 03h)+(CONFIG7H & 40h)+SUM_ID                                       | 031Dh       | 0318h                     |
| PIC18FX4K20 | None         | SUM[0000:07FF]+SUM[0800:1FFF]+SUM[2000:3FFF]+(CONFIG1L & 00h)+(CONFIG1H & CFh)+(CONFIG2L & 1Fh)+(CONFIG2H & 1F)+(CONFIG3L & 00h)+(CONFIG3H & 8Fh)+(CONFIG4L & C5h)+(CONFIG4H & 00h)+(CONFIG5L & 03h)+(CONFIG5H & C0h)+(CONFIG6L & 03h)+(CONFIG6H & E0h)+(CONFIG7L & 03h)+(CONFIG7H & 40h) | C33Eh       | C294h                     |
|             | Boot Block   | SUM[0800:1FFF]+SUM[2000:3FFF]+(CONFIG1L & 00h)+(CONFIG1H & CFh)+(CONFIG2L & 1Fh)+(CONFIG2H & 1F)+(CONFIG3L & 00h)+(CONFIG3H & 8Fh)+(CONFIG4L & C5h)+(CONFIG4H & 00h)+(CONFIG5L & 03h)+(CONFIG5H & C0h)+(CONFIG6L & 03h)+(CONFIG6H & E0h)+(CONFIG7L & 03h)+(CONFIG7H & 40h)+SUM_ID         | CB1Eh       | CAC4h                     |
|             | Boot/Block 0 | SUM[2000:3FFF]+(CONFIG1L & 00h)+(CONFIG1H & CFh)+(CONFIG2L & 1Fh)+(CONFIG2H & 1F)+(CONFIG3L & 00h)+(CONFIG3H & 8Fh)+(CONFIG4L & C5h)+(CONFIG4H & 00h)+(CONFIG5L & 03h)+(CONFIG5H & C0h)+(CONFIG6L & 03h)+(CONFIG6H & E0h)+(CONFIG7L & 03h)+(CONFIG7H & 40h)+SUM_ID                        | E31Dh       | E2C3h                     |
|             | All          | (CONFIG1L & 00h)+(CONFIG1H & CFh)+(CONFIG2L & 1Fh)+(CONFIG2H & 1F)+(CONFIG3L & 00h)+(CONFIG3H & 8Fh)+(CONFIG4L & C5h)+(CONFIG4H & 00h)+(CONFIG5L & 03h)+(CONFIG5H & C0h)+(CONFIG6L & 03h)+(CONFIG6H & E0h)+(CONFIG7L & 03h)+(CONFIG7H & 40h)+SUM_ID                                       | 031Bh       | 0316h                     |

**Legend:**

|             |   |
|-------------|---|
| <u>Item</u> | <u>Description</u>  |
| 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  |

# PIC18F2XK20/4XK20

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

| Device      | Code-Protect         | Checksum  | Blank Value | 0xAA at 0 and Max Address |
|-------------|----------------------|---|-------------|---------------------------|
| PIC18FX5K20 | None                 | SUM[0000:07FF]+SUM[0800:1FFF]+SUM[2000:3FFF]+SUM[4000:5FFF]+SUM[6000:7FFF]+(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) | 8362h       | 82B8h                     |
|             | Boot Block           | SUM[0800:1FFF]+SUM[2000:3FFF]+SUM[4000:5FFF]+SUM[6000:7FFF]+(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         | 8B35h       | 8AEAh                     |
|             | Boot/Block 0/Block 1 | SUM[4000:5FFF]+SUM[6000:7FFF]+(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                                       | C332h       | C2E7h                     |
|             | 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   | 0326h       | 0330h                     |

- 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  |

# PIC18F2XK20/4XK20

**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:FFF F]+(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

# PIC18F2XK20/4XK20

## 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 $\overline{\text{MCLR}}/\text{VPP}/\text{RE3}$    | $V_{DD} + 4.5$        | 9                   | V     |                                     |
| D110A                                      | VIHL               | Low-Voltage Programming Voltage on $\overline{\text{MCLR}}/\text{VPP}/\text{RE3}$     | 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                                       | I <sub>PP</sub>    | Programming Current on $\overline{\text{MCLR}}/\text{VPP}/\text{RE3}$                 | —                     | 300                 | μA    |                                     |
| D113                                       | I <sub>DDP</sub>   | Supply Current During Programming   | —                     | 10                  | mA    |                                     |
| D031                                       | V <sub>IL</sub>    | Input Low Voltage   | V <sub>SS</sub>       | 0.2 V <sub>DD</sub> | V     |                                     |
| D041                                       | V <sub>IH</sub>    | Input High Voltage  | 0.8 V <sub>DD</sub>   | V <sub>DD</sub>     | V     |                                     |
| D080                                       | V <sub>OL</sub>    | Output Low Voltage  | —                     | 0.6                 | V     | I <sub>OL</sub> = X.X mA @ 2.7V     |
| D090                                       | V <sub>OH</sub>    | Output High Voltage   | V <sub>DD</sub> - 0.7 | —                   | V     | I <sub>OH</sub> = -Y.Y mA @ 2.7V    |
| D012                                       | C <sub>IO</sub>    | Capacitive Loading on I/O pin (PGD)   | —                     | 50                  | pF    | To meet AC specifications           |
| P1   | T <sub>R</sub>     | $\overline{\text{MCLR}}/\text{VPP}/\text{RE3}$ Rise Time to enter Program/Verify mode | —                     | 1.0                 | μs    | <b>(Note 1)</b>                     |
| P2   | T <sub>PGC</sub>   | Serial Clock (PGC) Period   | 100                   | —                   | ns    | V <sub>DD</sub> = 3.6V              |
|  |                    |   | 1                     | —                   | μs    | V <sub>DD</sub> = 1.8V              |
| P2A  | T <sub>PGCL</sub>  | Serial Clock (PGC) Low Time   | 40                    | —                   | ns    | V <sub>DD</sub> = 3.6V              |
|  |                    |   | 400                   | —                   | ns    | V <sub>DD</sub> = 1.8V              |
| P2B  | T <sub>PGCH</sub>  | Serial Clock (PGC) High Time  | 40                    | —                   | ns    | V <sub>DD</sub> = 3.6V              |
|  |                    |   | 400                   | —                   | ns    | V <sub>DD</sub> = 1.8V              |
| P3   | T <sub>SET1</sub>  | Input Data Setup Time to Serial Clock ↓   | 15                    | —                   | ns    |                                     |
| P4   | T <sub>HLD1</sub>  | Input Data Hold Time from PGC ↓   | 15                    | —                   | ns    |                                     |
| P5   | T <sub>DLY1</sub>  | Delay between 4-bit Command and Command Operand                                       | 40                    | —                   | ns    |                                     |
| P5A  | T <sub>DLY1A</sub> | Delay between 4-bit Command Operand and next 4-bit Command                            | 40                    | —                   | ns    |                                     |
| P6   | T <sub>DLY2</sub>  | Delay between Last PGC ↓ of Command Byte to First PGC ↑ of Read of Data Word          | 20                    | —                   | ns    |                                     |
| P9   | T <sub>DLY5</sub>  | PGC High Time (minimum programming time)  | 1                     | —                   | ms    | Externally Timed                    |
| P9A  | T <sub>DLY5A</sub> | PGC High Time   | 5                     | —                   | ms    | Configuration Word programming time |
| P10  | T <sub>DLY6</sub>  | PGC Low Time after Programming (high-voltage discharge time)                          | 200                   | —                   | μs    |                                     |
| P11  | T <sub>DLY7</sub>  | Delay to allow Self-Timed Data Write or Bulk Erase to occur                           | 5                     | —                   | ms    |                                     |
| P11A                                       | T <sub>DRWT</sub>  | Data Write Polling Time   | 4                     | —                   | ms    |                                     |

**Note 1:** Do not allow excess time when transitioning  $\overline{\text{MCLR}}$  between V<sub>IL</sub> and V<sub>IHH</sub>; this can cause spurious program executions to occur. The maximum transition time is:  
 $1 T_{CY} + T_{PWRT}$  (if enabled) + 1024 T<sub>Osc</sub> (for LP, HS, HS/PLL and XT modes only) + 2 ms (for HS/PLL mode only) + 1.5 μs (for EC mode only) where T<sub>CY</sub> is the instruction cycle time, T<sub>PWRT</sub> is the Power-up Timer period and T<sub>Osc</sub> 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 $\overline{\text{MCLR}}/\text{VPP}/\text{RE3} \uparrow$                 | 2    | —    | $\mu\text{s}$ |            |
| P13  | TSET2  | $\text{VDD} \uparrow$ Setup Time to $\overline{\text{MCLR}}/\text{VPP}/\text{RE3} \uparrow$       | 100  | —    | ns            |            |
| P14  | TVALID | Data Out Valid from PGC $\uparrow$  | 10   | —    | ns            |            |
| P15  | TSET3  | PGM $\uparrow$ Setup Time to $\overline{\text{MCLR}}/\text{VPP}/\text{RE3} \uparrow$              | 2    | —    | $\mu\text{s}$ |            |
| P16  | TDLY8  | Delay between Last PGC $\downarrow$ and $\overline{\text{MCLR}}/\text{VPP}/\text{RE3} \downarrow$ | 0    | —    | s             |            |
| P17  | THLD3  | $\overline{\text{MCLR}}/\text{VPP}/\text{RE3} \downarrow$ to $\text{VDD} \downarrow$              | —    | 100  | ns            |            |
| P18  | THLD4  | $\overline{\text{MCLR}}/\text{VPP}/\text{RE3} \downarrow$ to PGM $\downarrow$                     | 0    | —    | s             |            |
| P19  | THIZ   | Delay from PGC $\uparrow$ to PGD High-Z   | 3    | 10   | nS            |            |
| P20  | TPPDP  | Hold time after VPP changes   | 5    | —    | $\mu\text{s}$ |            |

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

# PIC18F2XK20/4XK20

---

NOTES:



---

**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.

---

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

**Trademarks**

The Microchip name and logo, the Microchip logo, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, rfPIC and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Octopus, Omniscient Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICtail, PIC<sup>32</sup> logo, REAL ICE, rLAB, Select Mode, Total Endurance, TSHARC, UniWinDriver, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2009, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

**QUALITY MANAGEMENT SYSTEM**  
**CERTIFIED BY DNV**  
**== ISO/TS 16949:2002 ==**

*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.*



---

---

## WORLDWIDE SALES AND SERVICE

---

---

### AMERICAS

**Corporate Office**  
2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 480-792-7200  
Fax: 480-792-7277  
Technical Support:  
<http://support.microchip.com>  
Web Address:  
[www.microchip.com](http://www.microchip.com)

**Atlanta**  
Duluth, GA  
Tel: 678-957-9614  
Fax: 678-957-1455

**Boston**  
Westborough, MA  
Tel: 774-760-0087  
Fax: 774-760-0088

**Chicago**  
Itasca, IL  
Tel: 630-285-0071  
Fax: 630-285-0075

**Cleveland**  
Independence, OH  
Tel: 216-447-0464  
Fax: 216-447-0643

**Dallas**  
Addison, TX  
Tel: 972-818-7423  
Fax: 972-818-2924

**Detroit**  
Farmington Hills, MI  
Tel: 248-538-2250  
Fax: 248-538-2260

**Kokomo**  
Kokomo, IN  
Tel: 765-864-8360  
Fax: 765-864-8387

**Los Angeles**  
Mission Viejo, CA  
Tel: 949-462-9523  
Fax: 949-462-9608

**Santa Clara**  
Santa Clara, CA  
Tel: 408-961-6444  
Fax: 408-961-6445

**Toronto**  
Mississauga, Ontario,  
Canada  
Tel: 905-673-0699  
Fax: 905-673-6509

### ASIA/PACIFIC

**Asia Pacific Office**  
Suites 3707-14, 37th Floor  
Tower 6, The Gateway  
Harbour City, Kowloon  
Hong Kong  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**Australia - Sydney**  
Tel: 61-2-9868-6733  
Fax: 61-2-9868-6755

**China - Beijing**  
Tel: 86-10-8528-2100  
Fax: 86-10-8528-2104

**China - Chengdu**  
Tel: 86-28-8665-5511  
Fax: 86-28-8665-7889

**China - Hong Kong SAR**  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**China - Nanjing**  
Tel: 86-25-8473-2460  
Fax: 86-25-8473-2470

**China - Qingdao**  
Tel: 86-532-8502-7355  
Fax: 86-532-8502-7205

**China - Shanghai**  
Tel: 86-21-5407-5533  
Fax: 86-21-5407-5066

**China - Shenyang**  
Tel: 86-24-2334-2829  
Fax: 86-24-2334-2393

**China - Shenzhen**  
Tel: 86-755-8203-2660  
Fax: 86-755-8203-1760

**China - Wuhan**  
Tel: 86-27-5980-5300  
Fax: 86-27-5980-5118

**China - Xiamen**  
Tel: 86-592-2388138  
Fax: 86-592-2388130

**China - Xian**  
Tel: 86-29-8833-7252  
Fax: 86-29-8833-7256

**China - Zhuhai**  
Tel: 86-756-3210040  
Fax: 86-756-3210049

### ASIA/PACIFIC

**India - Bangalore**  
Tel: 91-80-3090-4444  
Fax: 91-80-3090-4080

**India - New Delhi**  
Tel: 91-11-4160-8631  
Fax: 91-11-4160-8632

**India - Pune**  
Tel: 91-20-2566-1512  
Fax: 91-20-2566-1513

**Japan - Yokohama**  
Tel: 81-45-471- 6166  
Fax: 81-45-471-6122

**Korea - Daegu**  
Tel: 82-53-744-4301  
Fax: 82-53-744-4302

**Korea - Seoul**  
Tel: 82-2-554-7200  
Fax: 82-2-558-5932 or  
82-2-558-5934

**Malaysia - Kuala Lumpur**  
Tel: 60-3-6201-9857  
Fax: 60-3-6201-9859

**Malaysia - Penang**  
Tel: 60-4-227-8870  
Fax: 60-4-227-4068

**Philippines - Manila**  
Tel: 63-2-634-9065  
Fax: 63-2-634-9069

**Singapore**  
Tel: 65-6334-8870  
Fax: 65-6334-8850

**Taiwan - Hsin Chu**  
Tel: 886-3-6578-300  
Fax: 886-3-6578-370

**Taiwan - Kaohsiung**  
Tel: 886-7-536-4818  
Fax: 886-7-536-4803

**Taiwan - Taipei**  
Tel: 886-2-2500-6610  
Fax: 886-2-2508-0102

**Thailand - Bangkok**  
Tel: 66-2-694-1351  
Fax: 66-2-694-1350

### EUROPE

**Austria - Wels**  
Tel: 43-7242-2244-39  
Fax: 43-7242-2244-393

**Denmark - Copenhagen**  
Tel: 45-4450-2828  
Fax: 45-4485-2829

**France - Paris**  
Tel: 33-1-69-53-63-20  
Fax: 33-1-69-30-90-79

**Germany - Munich**  
Tel: 49-89-627-144-0  
Fax: 49-89-627-144-44

**Italy - Milan**  
Tel: 39-0331-742611  
Fax: 39-0331-466781

**Netherlands - Drunen**  
Tel: 31-416-690399  
Fax: 31-416-690340

**Spain - Madrid**  
Tel: 34-91-708-08-90  
Fax: 34-91-708-08-91

**UK - Wokingham**  
Tel: 44-118-921-5869  
Fax: 44-118-921-5820

03/26/09