



Welcome to [E-XFL.COM](https://www.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](#)"

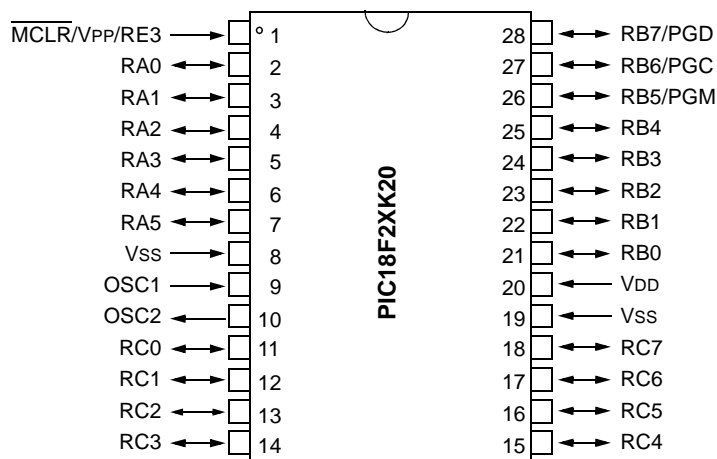
#### Details

Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	64MHz
Connectivity	I <sup>2</sup> C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	35
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.8K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 14x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-VQFN Exposed Pad
Supplier Device Package	44-QFN (8x8)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/microchip-technology/pic18f46k20-i-ml">https://www.e-xfl.com/product-detail/microchip-technology/pic18f46k20-i-ml</a>

# PIC18F2XK20/4XK20

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

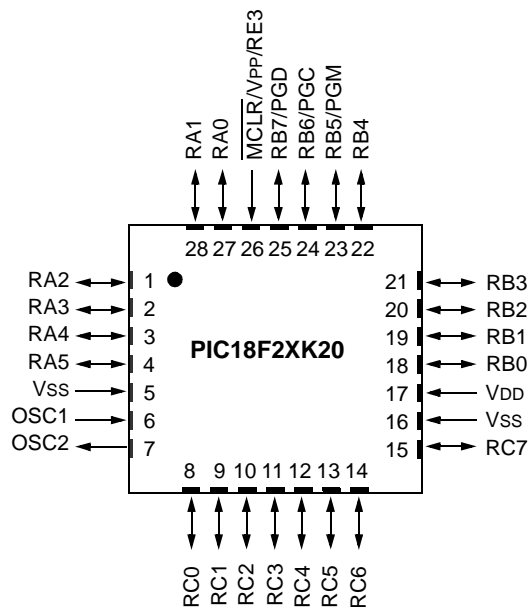
SDIP, SSOP, SOIC (300 MIL)



**Note:** The following devices are included in 28-pin SDIP, SSOP and SOIC parts: PIC18F23K20, PIC18F24K20, PIC18F25K20, PIC18F26K20.

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

28-Pin QFN



**Note:** The following devices are included in 28-pin QFN parts: PIC18F23K20, PIC18F24K20, PIC18F25K20, PIC18F26K20.

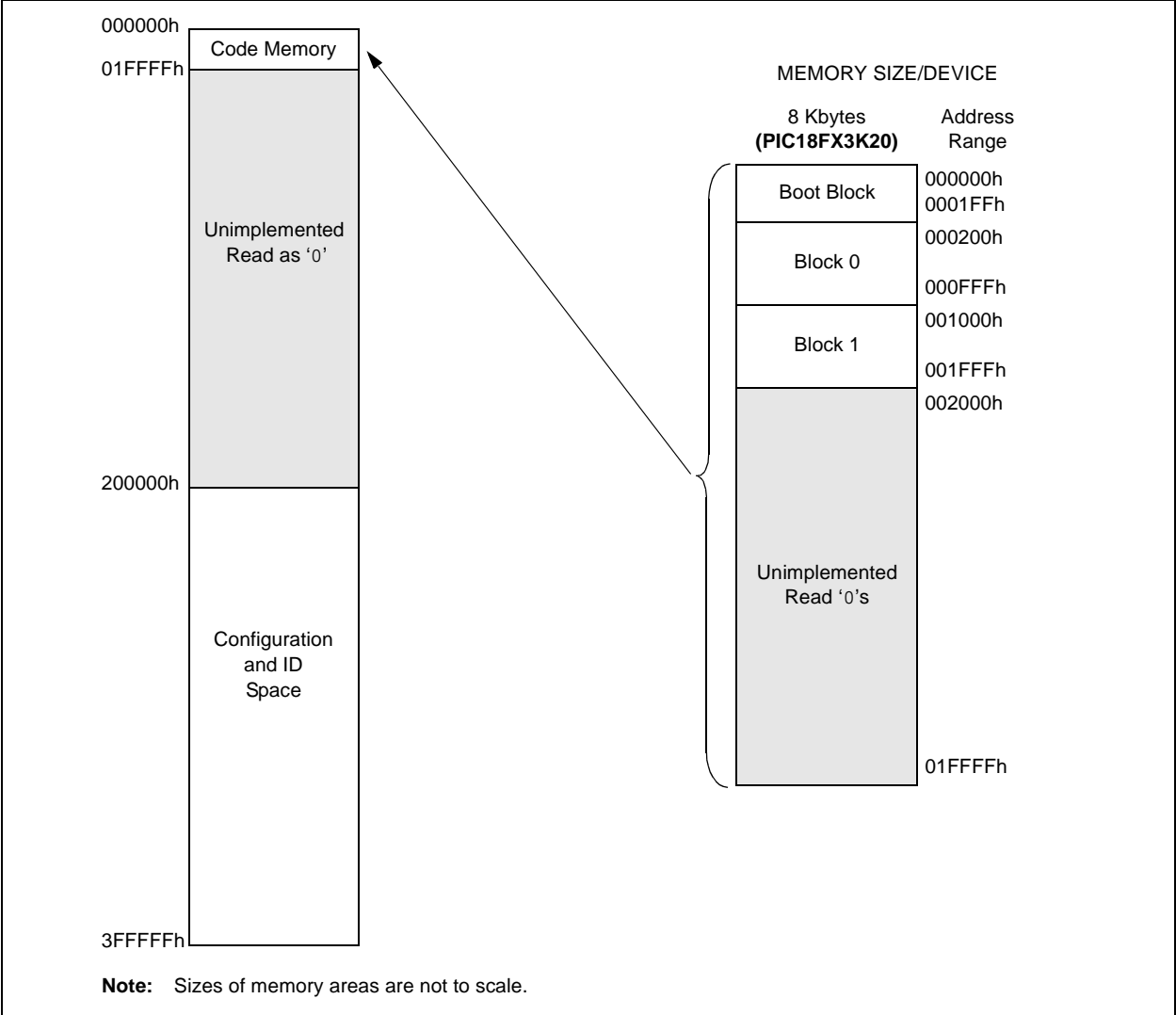
2.3 Memory Maps

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



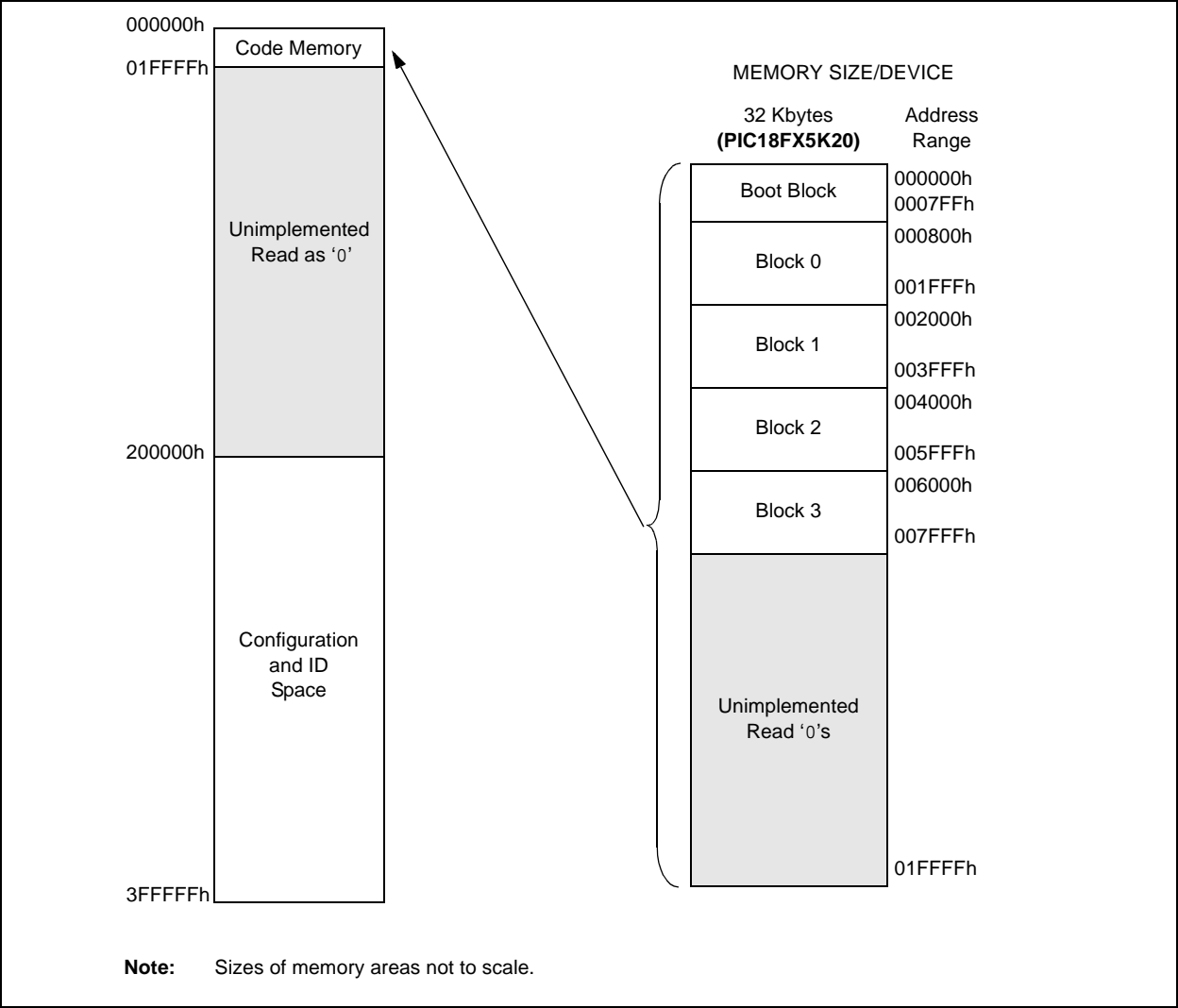
# PIC18F2XK20/4XK20

For 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



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

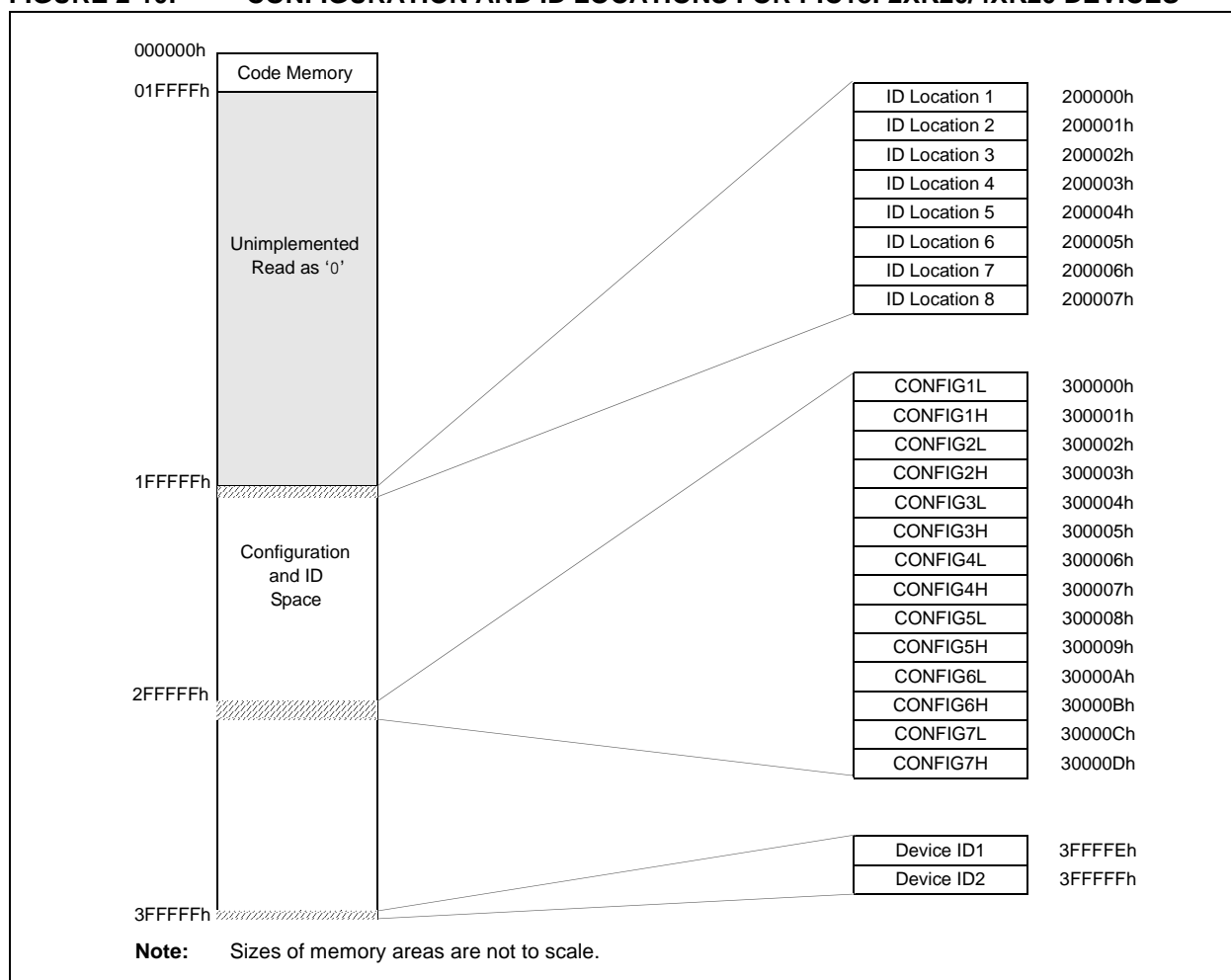
Memory in the address space, 000000h to 3FFFFFFh, is addressed via the Table Pointer register, which is comprised of three Pointer registers:

- 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**



## 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.

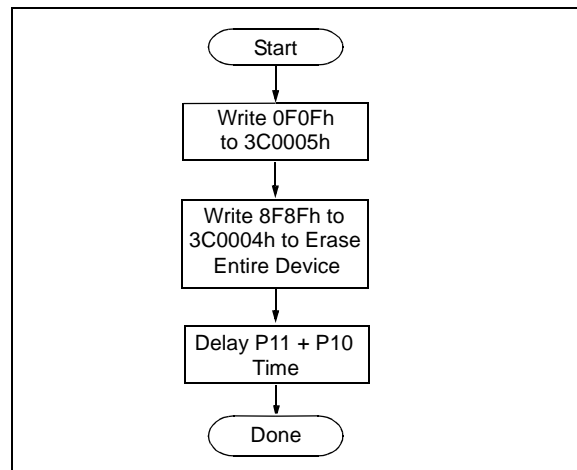
The code sequence to erase the entire device is shown in Table 3-2 and the flowchart is shown in Figure 3-1.

**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**

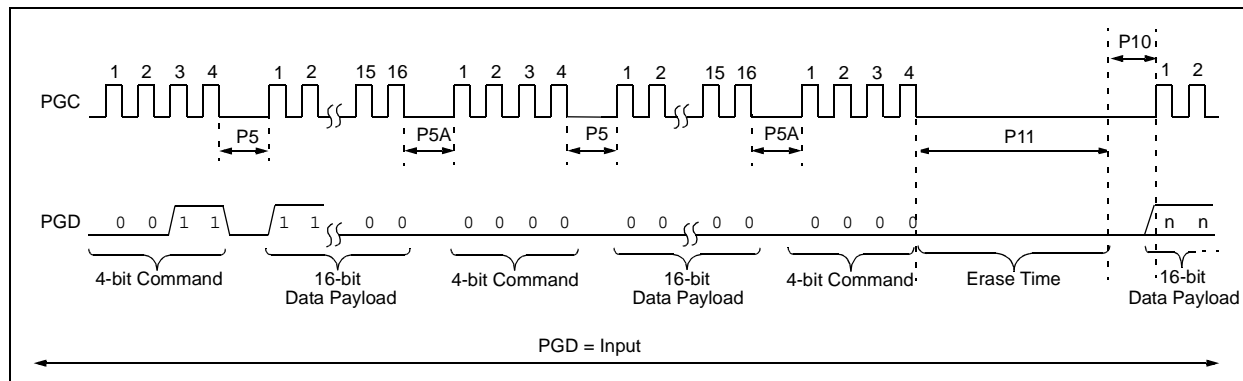
4-Bit Command	Data Payload	Core Instruction
0000	0E 3C	MOVLW 3Ch
0000	6E F8	MOVWF TBLPTRU
0000	0E 00	MOVLW 00h
0000	6E F7	MOVWF TBLPTRH
0000	0E 05	MOVLW 05h
0000	6E F6	MOVWF TBLPTRL
1100	0F 0F	Write 0Fh to 3C0005h
0000	0E 3C	MOVLW 3Ch
0000	6E F8	MOVWF TBLPTRU
0000	0E 00	MOVLW 00h
0000	6E F7	MOVWF TBLPTRH
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**

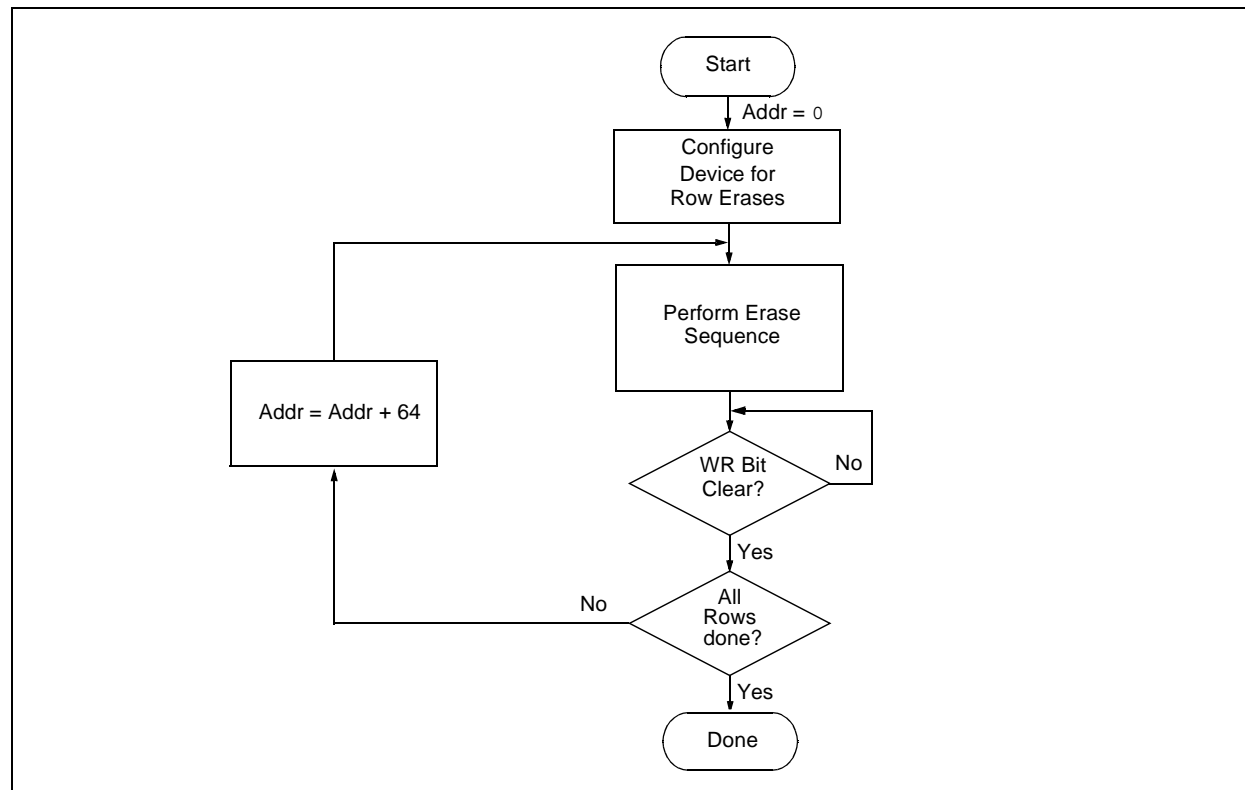
4-bit Command	Data Payload	Core Instruction
Step 1: Direct access to code memory and enable writes.		
0000	8E A6	BSF EECON1, EEPGD
0000	9C A6	BCF EECON1, CFGS
0000	84 A6	BSF EECON1, WREN
Step 2: Point to first row in code memory.		
0000	6A F8	CLRF TBLPTRU
0000	6A F7	CLRF TBLPTRH
0000	6A F6	CLRF TBLPTRL
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.		

## 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

**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.

## 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.

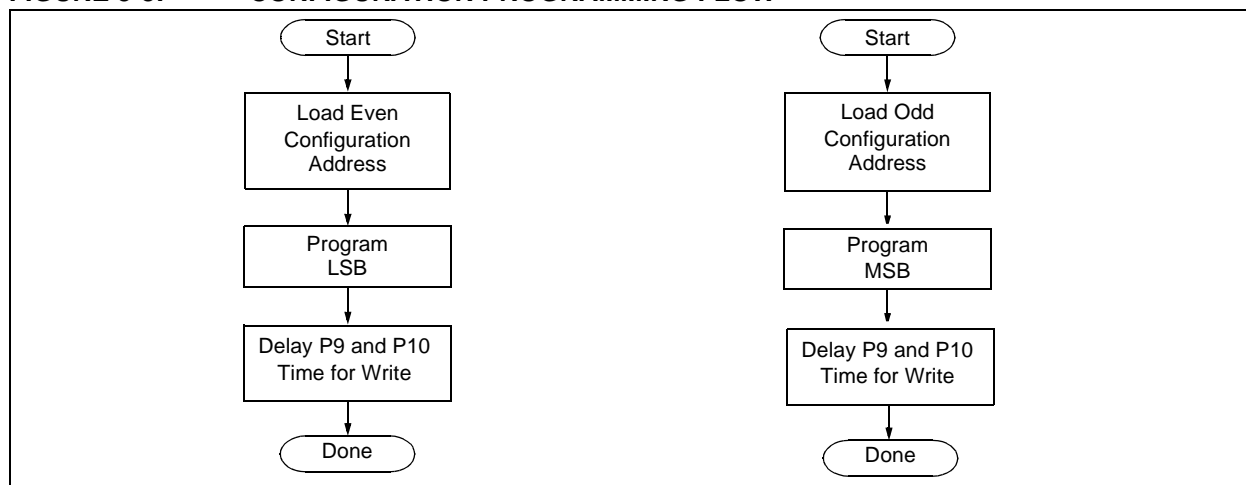
**Note:** The address must be explicitly written for each byte programmed. The addresses can not be incremented in this mode.

**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**

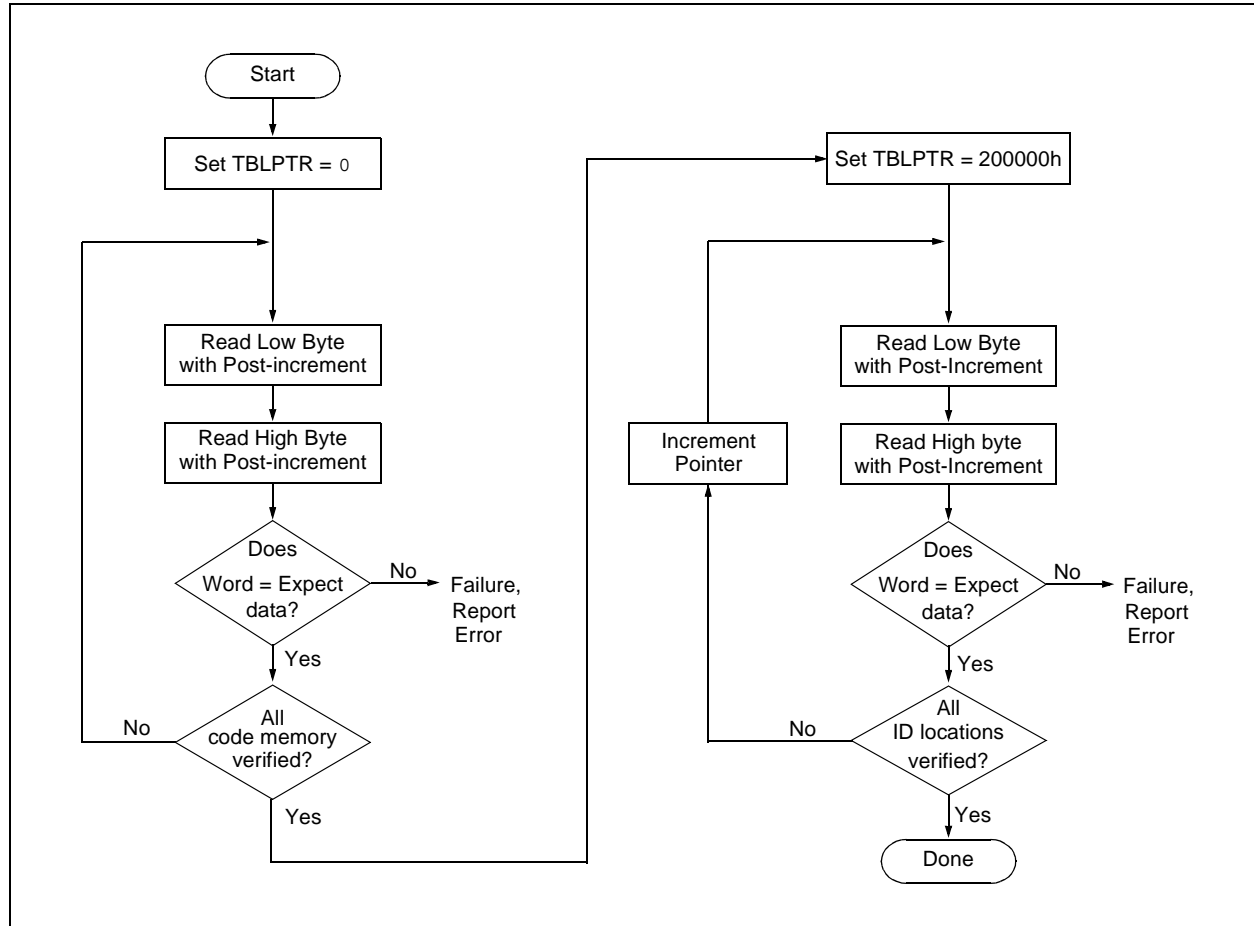


## 4.2 Verify Code Memory and ID Locations

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

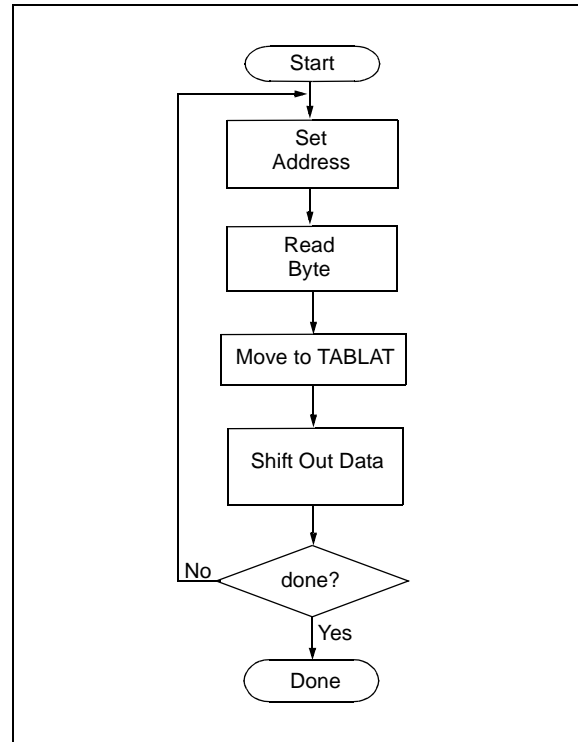
A configuration address may be read and output on PGD via the 4-bit command, '1001'. Configuration data is read and written in a byte-wise fashion, so it is not necessary to merge two bytes into a word prior to a compare. The result may then be immediately compared to the appropriate configuration data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

# PIC18F2XK20/4XK20

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

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



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

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

## 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

**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

---

**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, rPIC 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.

*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.*