



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 ² C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	24
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 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f26k20-i-so

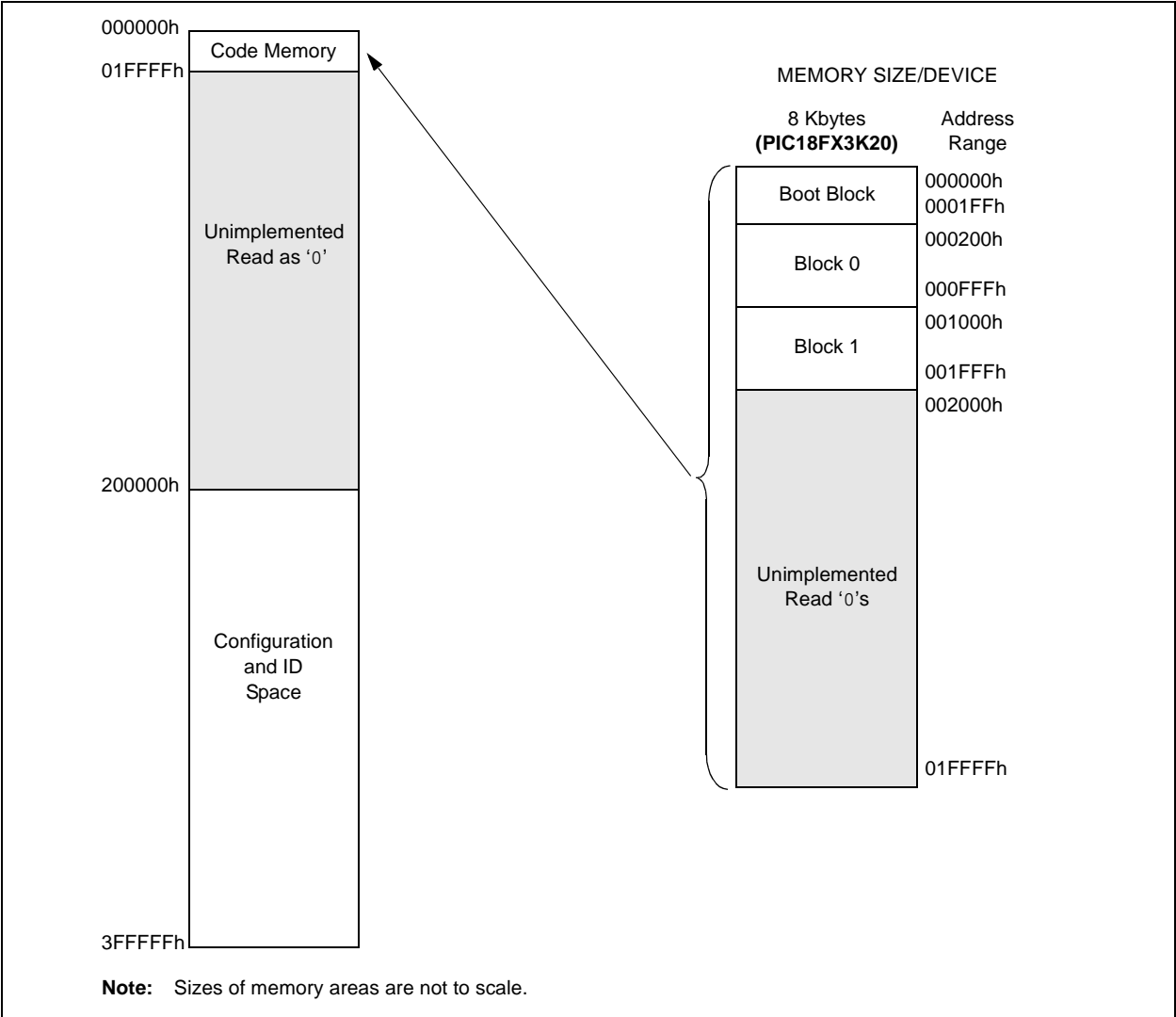
2.3 Memory Maps

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

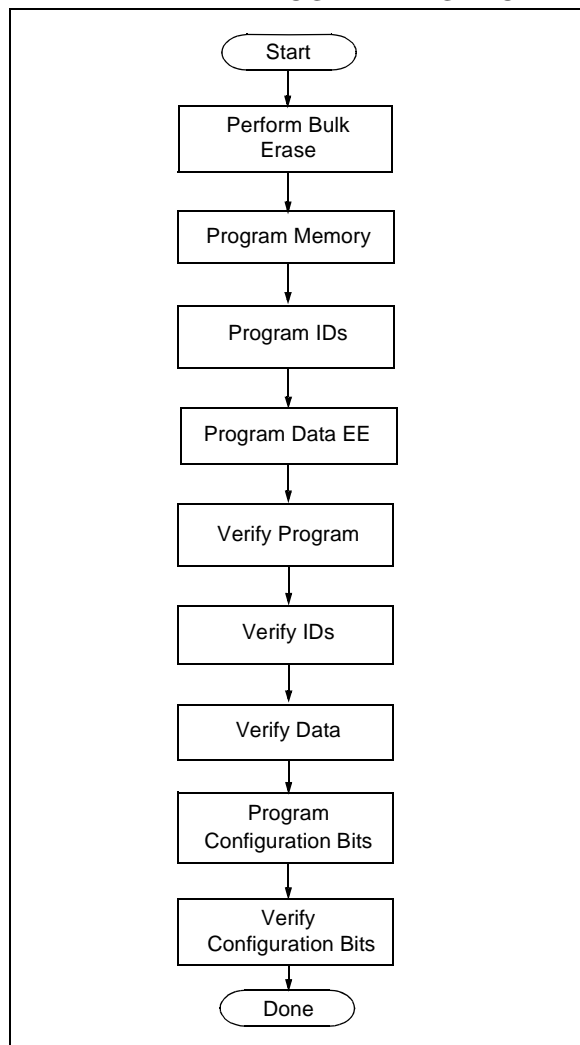


PIC18F2XK20/4XK20

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 $V_{\text{IH}}^{\text{HH}}$ (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

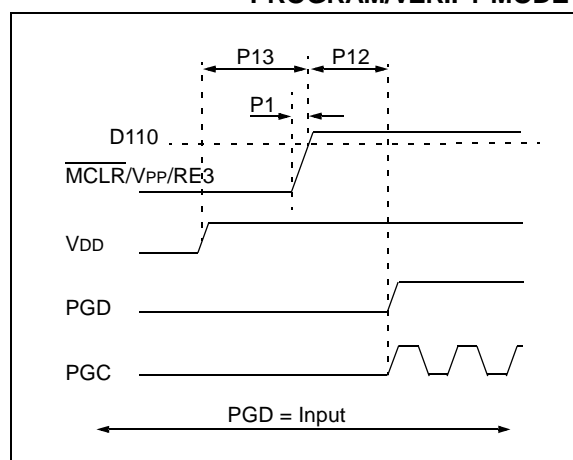
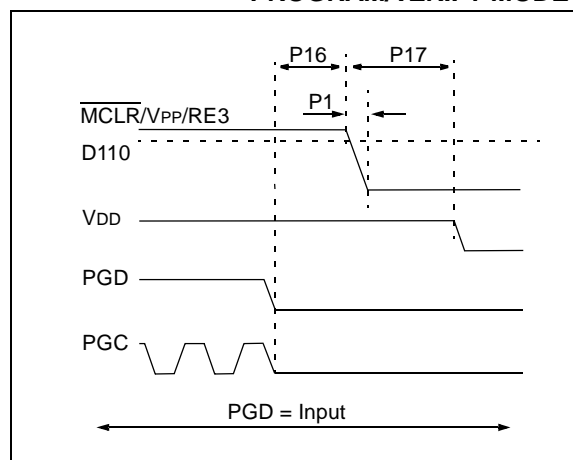


FIGURE 2-13: EXITING HIGH-VOLTAGE PROGRAM/VERIFY MODE



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

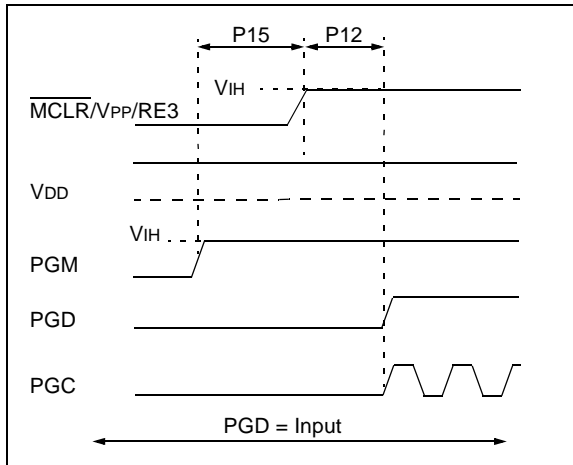
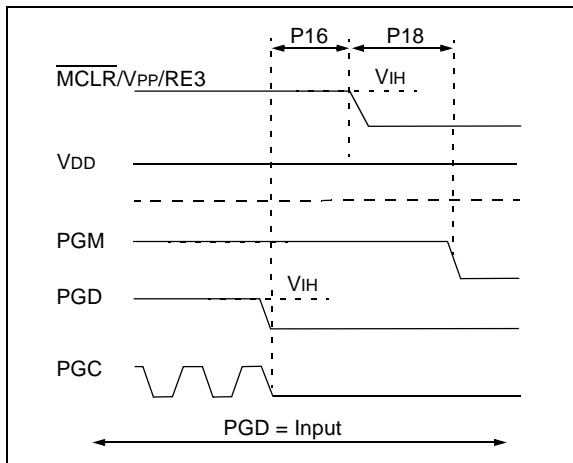


FIGURE 2-15: EXITING LOW-VOLTAGE PROGRAM/VERIFY MODE



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

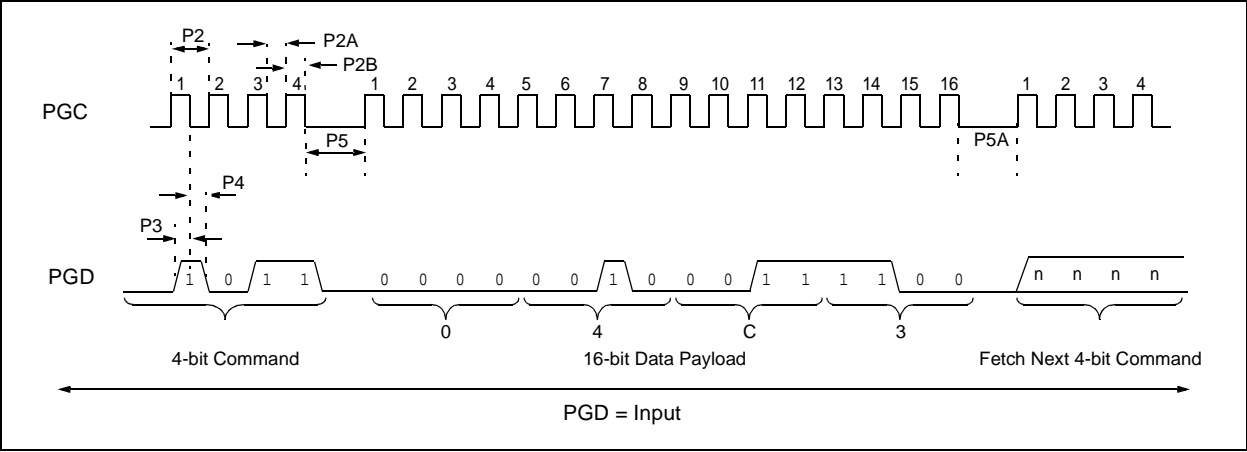
Description	4-Bit Command
Core Instruction (Shift in 16-bit instruction)	0000
Shift out TABLAT register	0010
Table Read	1000
Table Read, post-increment	1001
Table Read, post-decrement	1010
Table Read, pre-increment	1011
Table Write	1100
Table Write, post-increment by 2	1101
Table Write, start programming, post-increment by 2	1110
Table Write, start programming	1111

PIC18F2XK20/4XK20

TABLE 2-7: SAMPLE COMMAND SEQUENCE

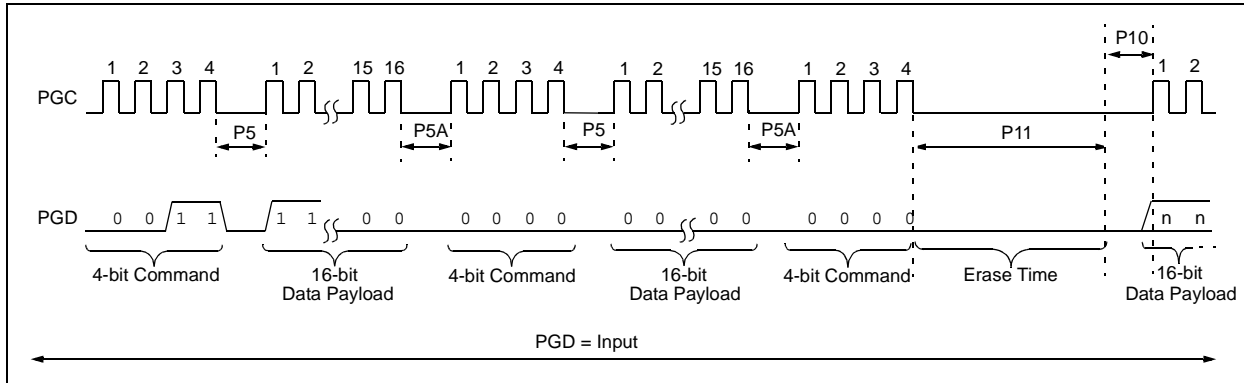
4-Bit Command	Data Payload	Core Instruction
1101	3C 40	Table Write, post-increment by 2

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



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”**).

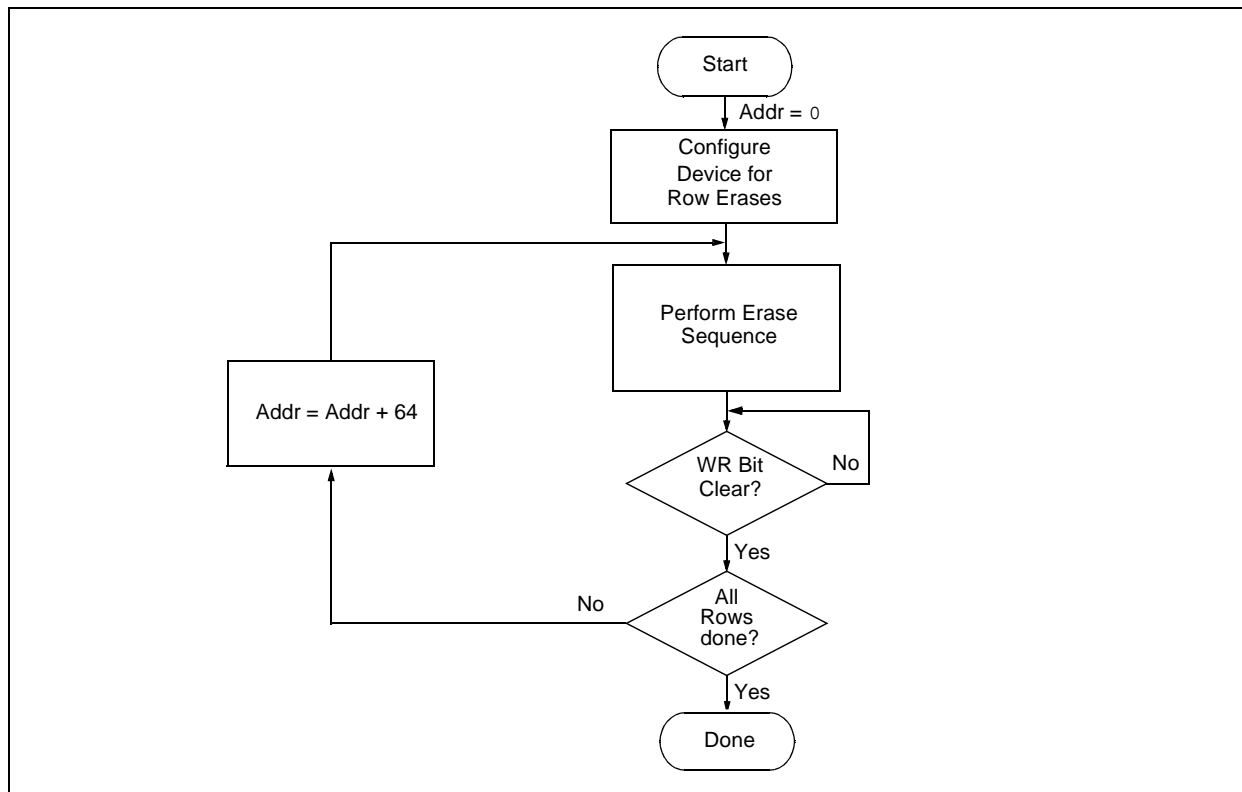
The Row Erase duration is self-timed. After the WR bit in EECON1 is set, two NOPs are issued. Erase starts upon the 4th PGC of the second NOP. It ends when the WR bit is cleared by hardware.

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.

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

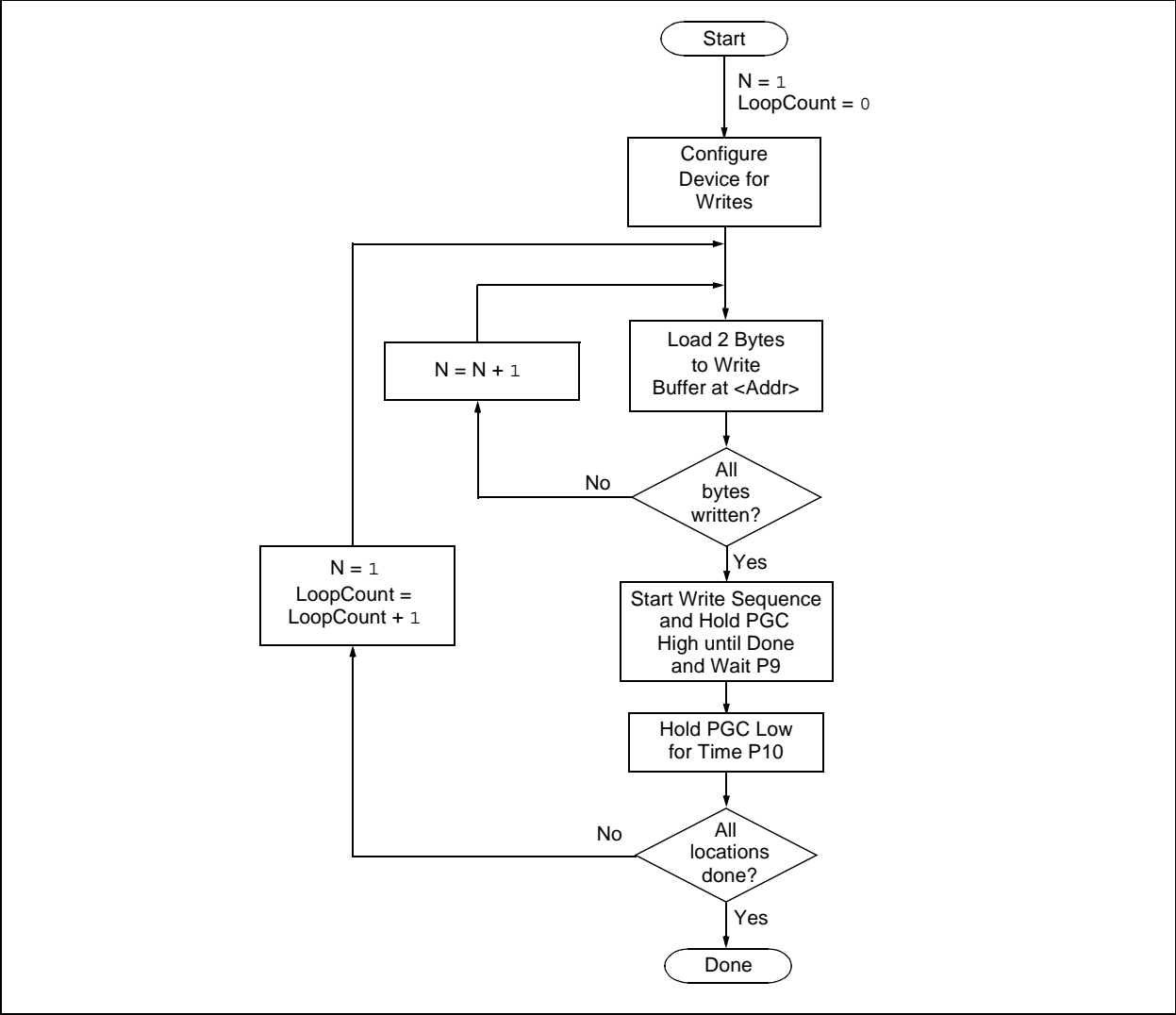
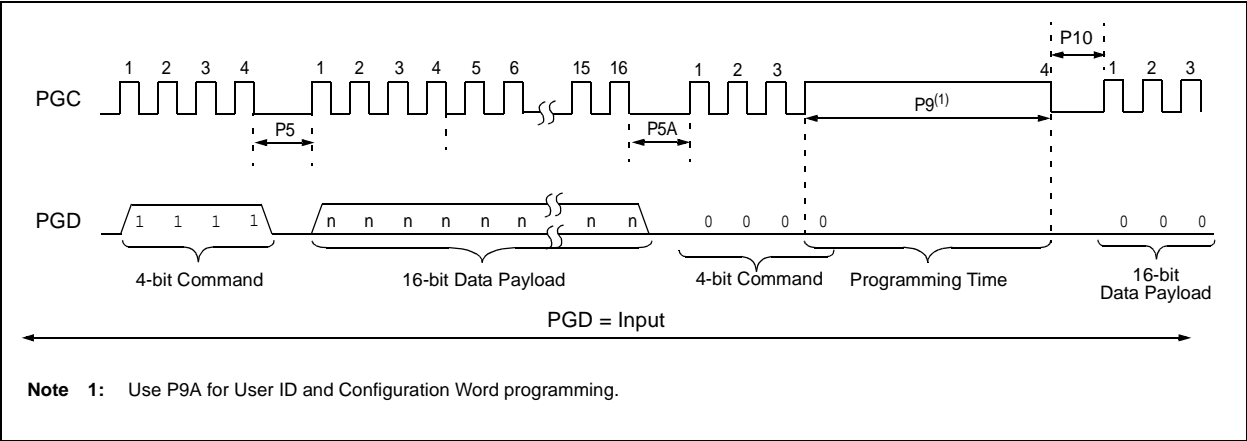


FIGURE 3-5: TABLE WRITE AND START PROGRAMMING INSTRUCTION TIMING DIAGRAM (1111)



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 (Section 4.1 “Read Code Memory, ID Locations and Configuration Bits”).		
Step 3: Set the Table Pointer for the block to be erased.		
0000	0E <Addr[21:16]>	MOVLW <Addr[21:16]>
0000	6E F8	MOVWF TBLPTRU
0000	0E <Addr[8:15]>	MOVLW <Addr[8:15]>
0000	6E F7	MOVWF TBLPTRH
0000	0E <Addr[7:0]>	MOVLW <Addr[7:0]>
0000	6E F6	MOVWF TBLPTRL
Step 4: Enable memory writes and setup an erase.		
0000	84 A6	BSF EECON1, WREN
0000	88 A6	BSF EECON1, FREE
Step 5: Initiate erase.		
0000	88 A6	BSF EECON1, FREE
0000	82 A6	BSF EECON1, WR
0000	00 00	NOP
0000	00 00	NOP Erase starts on the 4th clock of this instruction
Step 6: Poll WR bit. Repeat until bit is clear.		
0000	50 A6	MOVF EECON1, W, 0
0000	6E F5	MOVWF TABLAT
0000	00 00	NOP
0000	<MSB><LSB>	Shift out data ⁽¹⁾
Step 7: Load write buffer. The correct bytes will be selected based on the Table Pointer.		
0000	0E <Addr[21:16]>	MOVLW <Addr[21:16]>
0000	6E F8	MOVWF TBLPTRU
0000	0E <Addr[8:15]>	MOVLW <Addr[8:15]>
0000	6E F7	MOVWF TBLPTRH
0000	0E <Addr[7:0]>	MOVLW <Addr[7:0]>
0000	6E F6	MOVWF TBLPTRL
1101	<MSB><LSB>	Write 2 bytes and post-increment address by 2.
.	.	
.	.	Repeat as many times as necessary to fill the write buffer
.	.	Write 2 bytes and start programming.
1111	<MSB><LSB>	NOP - hold PGC high for time P9 and low for time P10.
0000	00 00	
To continue modifying data, repeat Steps 2 through 6, where the Address Pointer is incremented by the appropriate number of bytes (see Table 3-4) at each iteration of the loop. The write cycle must be repeated enough times to completely rewrite the contents of the erase buffer.		
Step 8: Disable writes.		
0000	94 A6	BCF EECON1, WREN

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 ($\text{EECON1}\langle 7:6 \rangle = 00$). The WREN bit must be set ($\text{EECON1}\langle 2 \rangle = 1$) 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 ($\text{EECON1}\langle 1 \rangle = 1$).

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

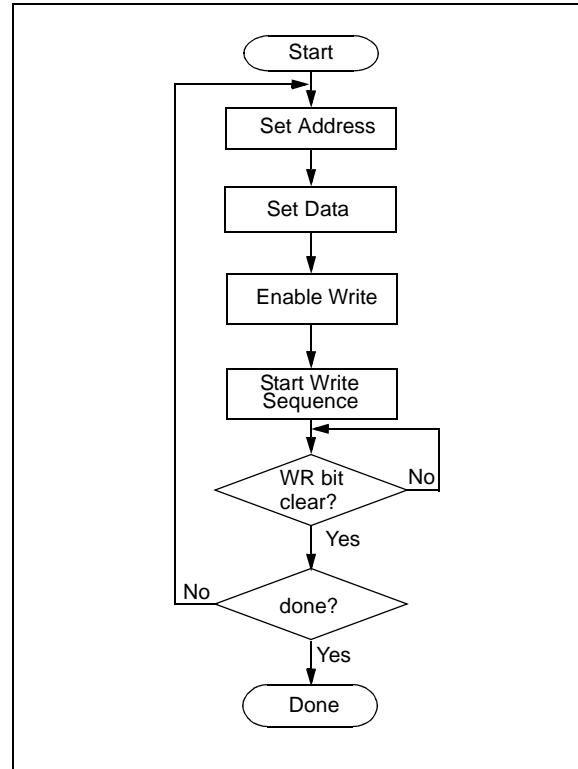


FIGURE 3-7: DATA EEPROM WRITE TIMING DIAGRAM

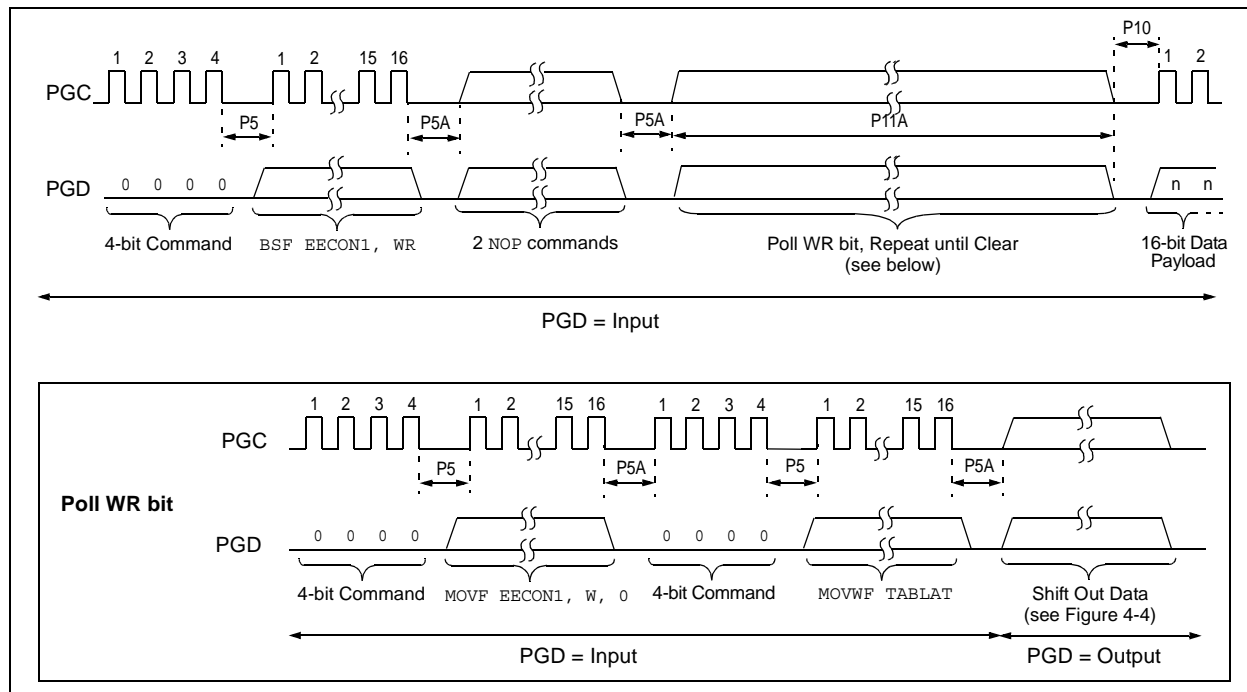


TABLE 3-7: PROGRAMMING DATA MEMORY

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 ⁽¹⁾
Step 7: Hold PGC low for time P10.		
Step 8: Disable writes.		
0000	94 A6	BCF EECON1, WREN
Repeat steps 2 through 8 to write more data.		

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

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.

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

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

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

TABLE 3-8: WRITE ID SEQUENCE

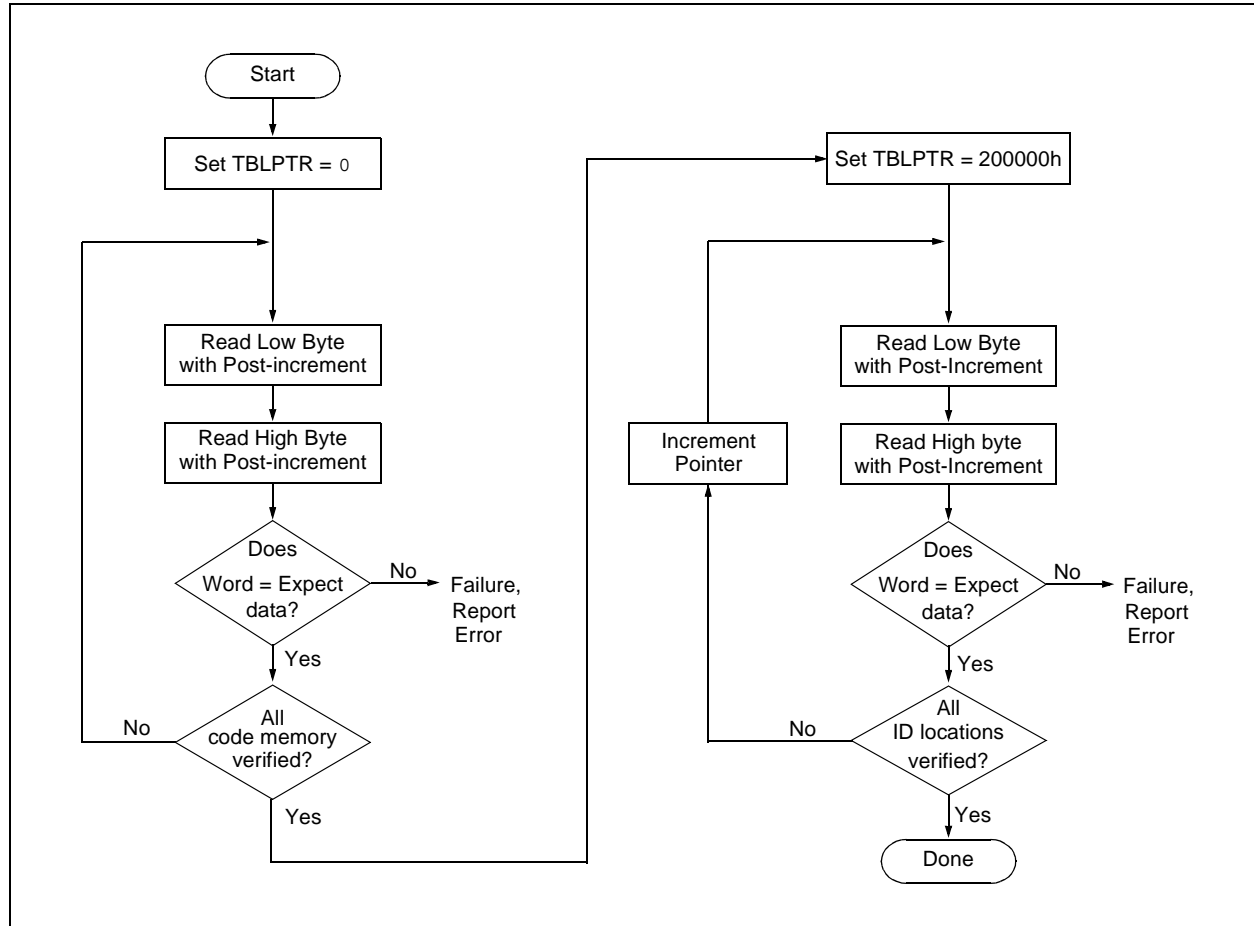
4-bit Command	Data Payload	Core Instruction
Step 1: Direct access to code memory.		
0000	8E A6	BSF EECON1, EEPGD
0000	9C A6	BCF EECON1, CFGS
0000	84 A6	BSF EECON1, WREN
Step 2: Set Table Pointer to ID. Load write buffer with 8 bytes and write.		
0000	0E 20	MOVLW 20h
0000	6E F8	MOVWF TBLPTRU
0000	0E 00	MOVLW 00h
0000	6E F7	MOVWF TBLPTRH
0000	0E 00	MOVLW 00h
0000	6E F6	MOVWF TBLPTRL
1101	<MSB><LSB>	Write 2 bytes and post-increment address by 2.
1101	<MSB><LSB>	Write 2 bytes and post-increment address by 2.
1101	<MSB><LSB>	Write 2 bytes and post-increment address by 2.
1111	<MSB><LSB>	Write 2 bytes and start programming.
0000	00 00	NOP - hold PGC high for time P9 and low for time P10.

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

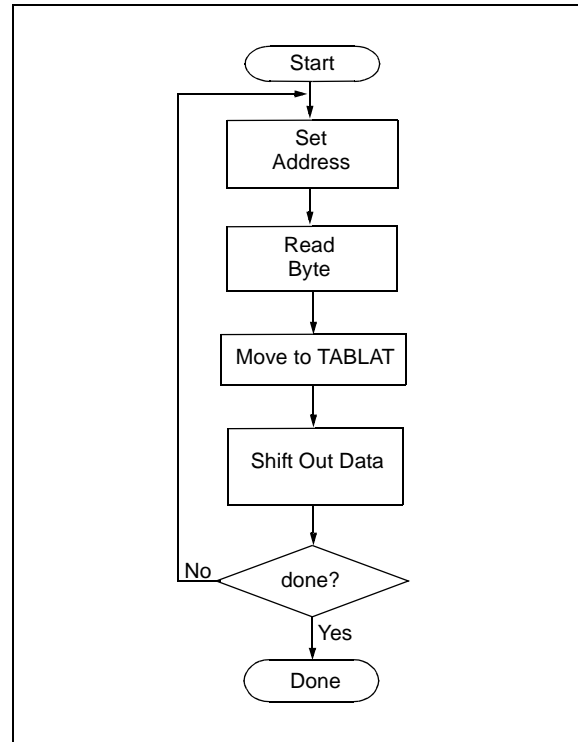


TABLE 4-2: READ DATA EEPROM MEMORY

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 ⁽¹⁾

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

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

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

Note 1: Do not allow excess time when transitioning $\overline{\text{MCLR}}$ between V_{IL} and V_{IHH}; this can cause spurious program executions to occur. The maximum transition time is:
 1 T_{CY} + T_{PWRT} (if enabled) + 1024 T_{OSC} (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_{CY} is the instruction cycle time, T_{PWRT} 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.