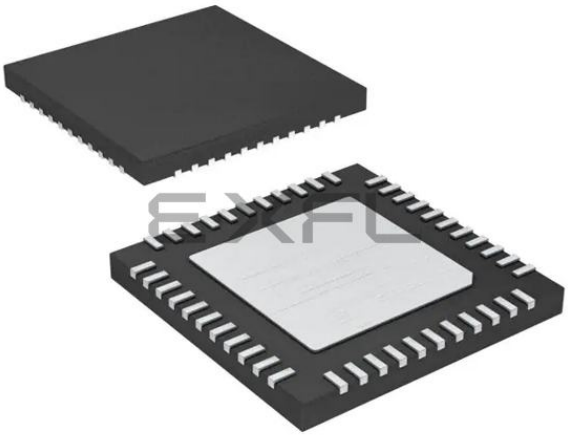
Microchip Technology - PIC18F4321T-I/ML Datasheet



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	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	40MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	36
Program Memory Size	8KB (4K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 13x10b
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	https://www.e-xfl.com/product-detail/microchip-technology/pic18f4321t-i-ml

TABLE 2-1: PIN DESCRIPTIONS (DURING PROGRAMMING): PIC18F2XXX/4XXX FAMILY

Pin Name	During Programming		
	Pin Name	Pin Type	Pin Description
$\overline{\text{MCLR}}$/VPP/RE3	VPP	P	Programming Enable
VDD[2]	VDD	P	Power Supply
VSS[2]	VSS	P	Ground
RB5	PGM	I	Low-Voltage ICSP™ Input when LVP Configuration bit equals ‘1’[1]
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.

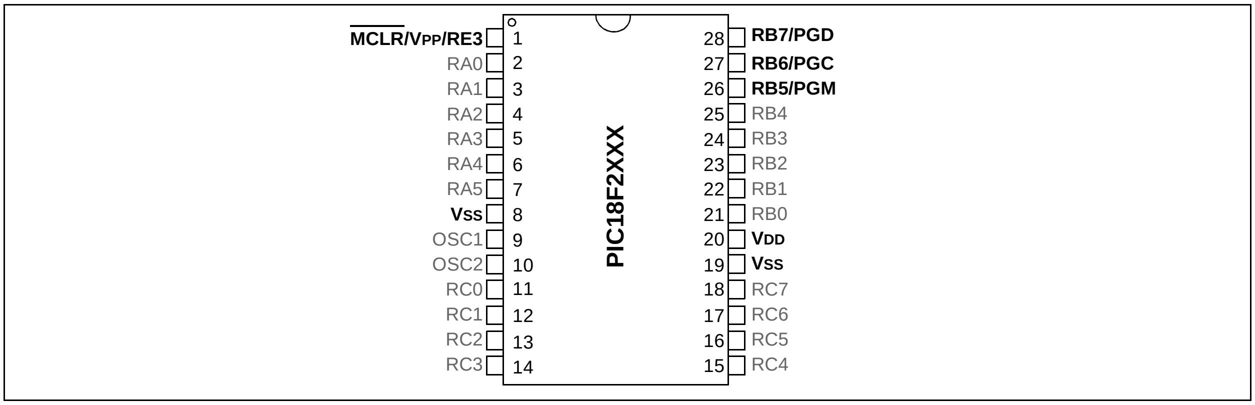
The following devices are included in 28-pin SPDIP, PDIP and SOIC parts:

- PIC18F2221
- PIC18F2321
- PIC18F2410
- PIC18F2420
- PIC18F2423
- PIC18F2450
- PIC18F2455
- PIC18F2458
- PIC18F2480
- PIC18F2510
- PIC18F2515
- PIC18F2520
- PIC18F2523
- PIC18F2525
- PIC18F2550
- PIC18F2553
- PIC18F2580
- PIC18F2585
- PIC18F2610
- PIC18F2620
- PIC18F2680
- PIC18F2682
- PIC18F2685

The following devices are included in 28-pin SSOP parts:

- PIC18F2221
- PIC18F2321

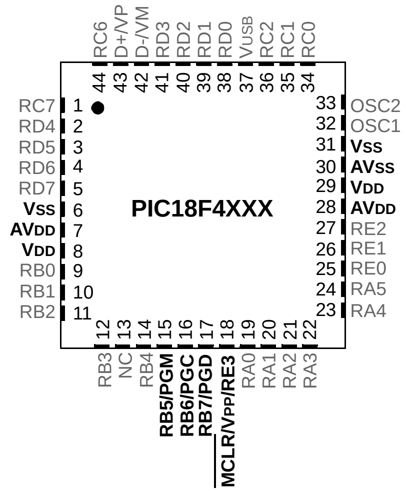
FIGURE 2-1: 28-Pin SPDIP, PDIP, SOIC,SSOP



The following devices are included in 44-pin QFN parts:

- PIC18F4221
- PIC18F4321
- PIC18F4410
- PIC18F4420
- PIC18F4423
- PIC18F4450
- PIC18F4455
- PIC18F4458
- PIC18F4480
- PIC18F4510
- PIC18F4520
- PIC18F4515
- PIC18F4523
- PIC18F4525
- PIC18F4550
- PIC18F4553
- PIC18F4580
- PIC18F4585
- PIC18F4610
- PIC18F4620
- PIC18F4680
- PIC18F4682
- PIC18F4685

FIGURE 2-5: 44-PIN QFN



2.3 Memory Maps

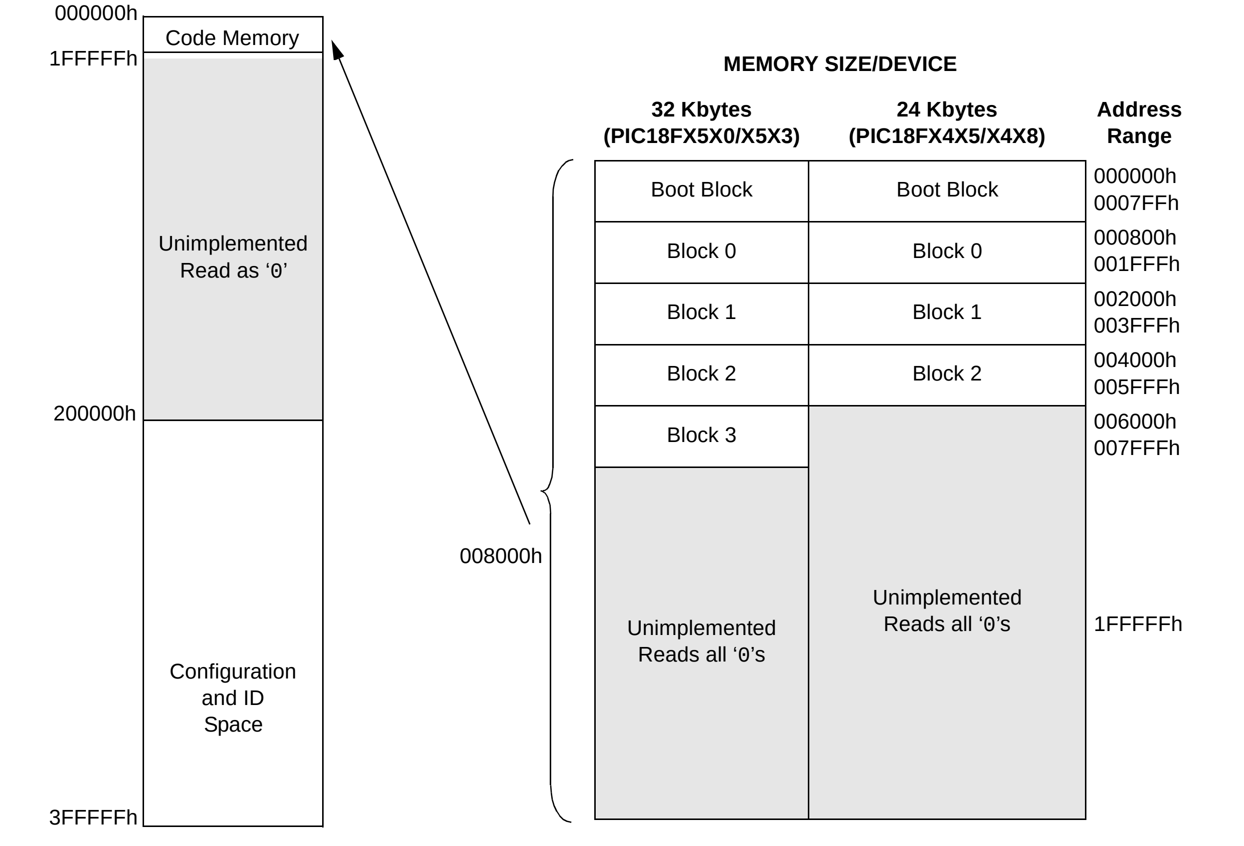
For PIC18FX6X0 devices, the code memory space extends from 0000h to 0FFFFh (64 Kbytes) in four 16-Kbyte blocks. For PIC18FX5X5 devices, the code memory space extends from 0000h to 0BFFFFh (48 Kbytes) in three 16-Kbyte blocks. Addresses, 0000h through 07FFh, however, define a “Boot Block” region that is treated separately from Block 0. All of these blocks define code protection boundaries within the code memory space.

The size of the Boot Block in PIC18F2585/2680/4585/4680 devices can be configured as 1, 2 or 4K words (see Figure 2-6). This is done through the BBSIZ<1:0> bits in the Configuration register, CONFIG4L. It is important to note that increasing the size of the Boot Block decreases the size of Block 0.

TABLE 2-4: IMPLEMENTATION OF CODE MEMORY

Device	Code Memory Size (Bytes)
PIC18F2455	000000h-005FFFh (24K)
PIC18F2458	
PIC18F4455	
PIC18F4458	
PIC18F2510	000000h-007FFFh (32K)
PIC18F2520	
PIC18F2523	
PIC18F2550	
PIC18F2553	
PIC18F4510	
PIC18F4520	
PIC18F4523	
PIC18F4550	
PIC18F4553	

FIGURE 2-8: MEMORY MAP AND THE CODE MEMORY SPACE FOR PIC18FX4X5/X4X8/X5X0/X5X3 DEVICES



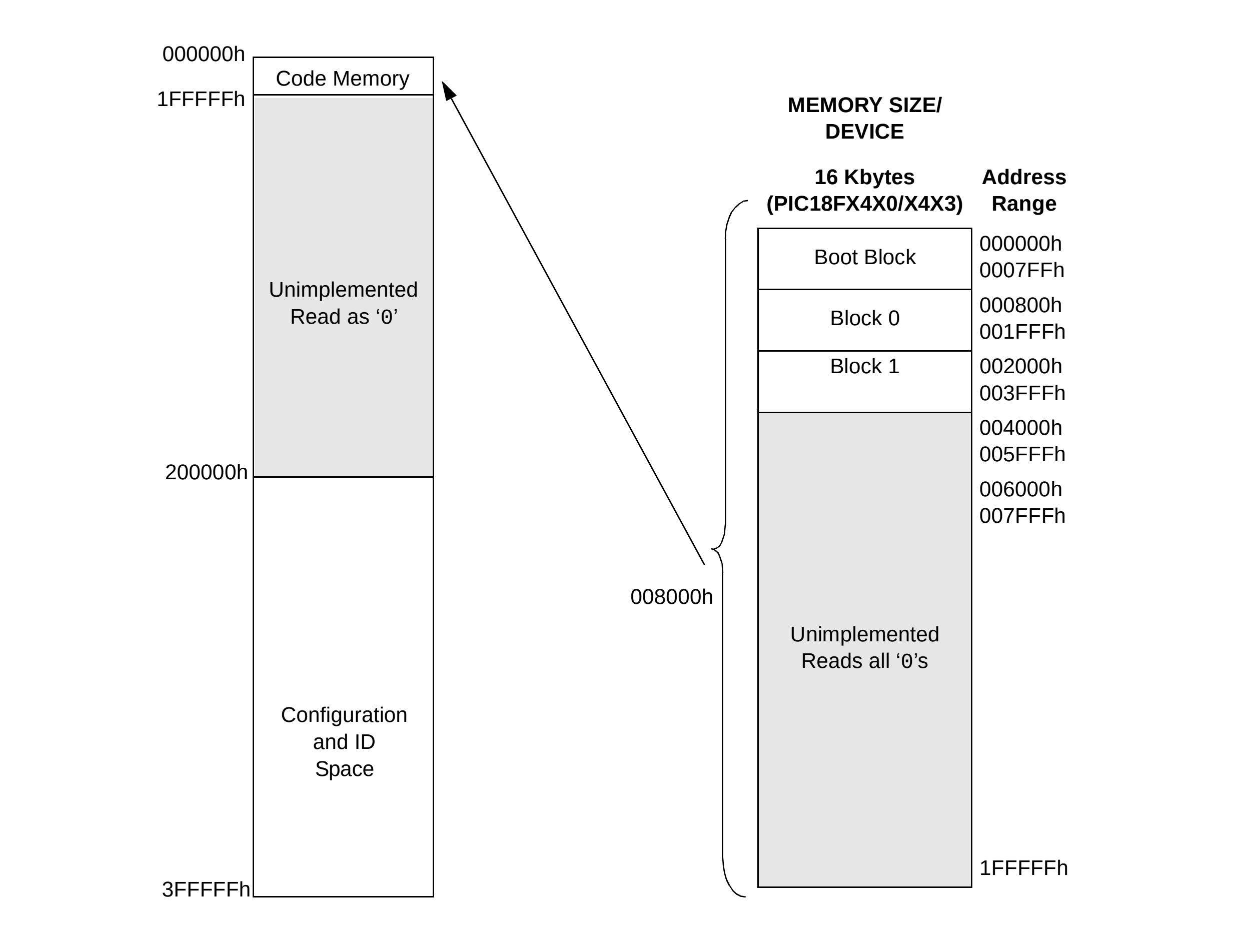
Note: Sizes of memory areas are not to scale.

For PIC18FX4X0/X4X3 devices, the code memory space extends from 000000h to 003FFFh (16 Kbytes) in two 8-Kbyte blocks. Addresses, 000000h through 0003FFh, 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)
PIC18F2410	000000h-003FFFh (16K)
PIC18F2420	
PIC18F2423	
PIC18F2450	
PIC18F4410	
PIC18F4420	
PIC18F4450	

FIGURE 2-9: MEMORY MAP AND THE CODE MEMORY SPACE FOR PIC18FX4X0/X4X3 DEVICES



Note: Sizes of memory areas are not to scale.

For PIC18F2480/4480 devices, the code memory space extends from 0000h to 03FFFh (16 Kbytes) in one 16-Kbyte block. For PIC18F2580/4580 devices, the code memory space extends from 0000h to 07FFFh (32 Kbytes) in two 16-Kbyte blocks. Addresses, 0000h through 07FFh, 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.

The size of the Boot Block in PIC18F2480/2580/4480/4580 devices can be configured as 1 or 2K words (see Figure 2-10). This is done through the BBSIZ<0> bit in the Configuration register, CONFIG4L. It is important to note that increasing the size of the Boot Block decreases the size of Block 0.

In addition to the code memory space, there are three blocks that are accessible to the user through Table Reads and Table Writes. Their locations in the memory map are shown in Figure 2-12.

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

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

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

2.3.1 MEMORY ADDRESS POINTER

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

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

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

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

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 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 being 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	3F8Fh
Erase Data EEPROM[1]	0084h
Erase Boot Block	0081h
Erase Configuration 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
Erase Code EEPROM Block 4	1080h
Erase Code EEPROM Block 5	2080h

Note 1: Selected devices only, see **Section 3.3 “Data EEPROM Programming”**.

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 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 3-5: WRITE CODE MEMORY CODE SEQUENCE

4-Bit Command	Data Payload	Core Instruction
Step 1: Direct access to code memory and enable writes.		
0000 0000	8E A6 9C A6	BSF EECON1, EEPGD BCF EECON1, CFGS
Step 2: Load write buffer.		
0000 0000 0000 0000 0000 0000	0E <Addr[21:16]> 6E F8 0E <Addr[15:8]> 6E F7 0E <Addr[7:0]> 6E F6	MOVLW <Addr[21:16]> MOVWF TBLPTRU MOVLW <Addr[15:8]> MOVWF TBLPTRH MOVLW <Addr[7:0]> MOVWF TBLPTRL
Step 3: 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.		
1111 0000	<MSB><LSB> 00 00	Write 2 bytes and start programming. 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.		

3.3 Data EEPROM Programming

Note: Data EEPROM programming is **not** available on the following devices:	
PIC18F2410	PIC18F4410
PIC18F2450	PIC18F4450
PIC18F2510	PIC18F4510
PIC18F2515	PIC18F4515
PIC18F2610	PIC18F4610

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<7:6> = 00). The WREN bit must be set (EECON1<2> = 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<1> = 1).

The write begins on the falling edge of the 4th PGC after the WR bit is set. It ends when the WR bit is cleared by hardware.

After the programming sequence terminates, PGC must still 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

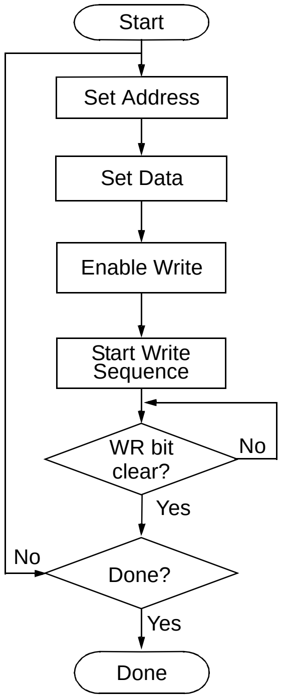


TABLE 3-7: PROGRAMMING DATA MEMORY

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

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

3.4 ID Location Programming

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

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

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

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

TABLE 3-8: WRITE ID SEQUENCE

4-Bit Command	Data Payload	Core Instruction
Step 1: Direct access to code memory and enable writes.		
0000 0000	8E A6 9C A6	BSF EECON1, EEPGD BCF EECON1, CFGS
Step 2: Load write buffer with 8 bytes and write.		
0000 0000 0000 0000 0000 0000 1101 1101 1101 1111 0000	0E 20 6E F8 0E 00 6E F7 0E 00 6E F6 <MSB><LSB> <MSB><LSB> <MSB><LSB> <MSB><LSB> 00 00	MOVLW 20h MOVWF TBLPTRU MOVLW 00h MOVWF TBLPTRH MOVLW 00h MOVWF TBLPTRL Write 2 bytes and post-increment address by 2. Write 2 bytes and post-increment address by 2. Write 2 bytes and post-increment address by 2. Write 2 bytes and start programming. 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 eight 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.

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

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

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

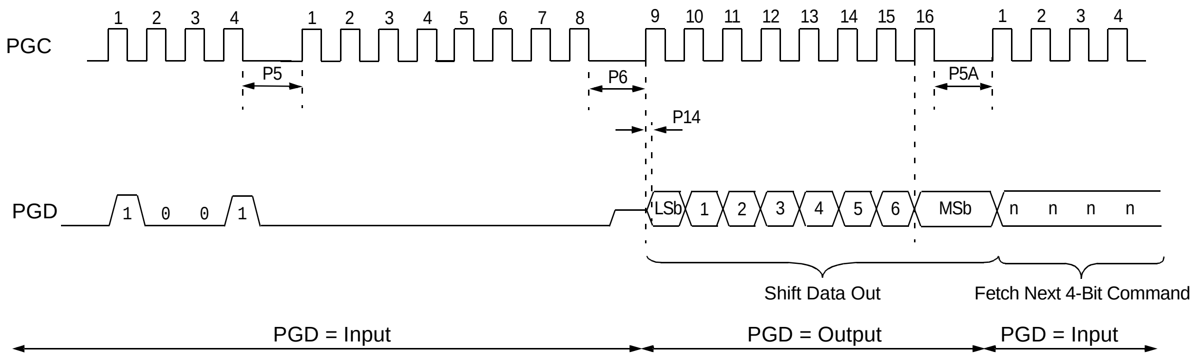
TABLE 4-1: READ CODE MEMORY SEQUENCE

4-Bit Command	Data Payload	Core Instruction
Step 1: Set Table Pointer.		

`0000`
`0000`
`0000`
`0000`
`0000`
`0000`
	`0E <Addr[21:16]>`
	`6E F8`
	`0E <Addr[15:8]>`
	`6E F7`
	`0E <Addr[7:0]>`
	`6E F6`
		`MOVLW Addr[21:16]`
		`MOVWF TBLPTRU`
		`MOVLW <Addr[15:8]>`
		`MOVWF TBLPTRH`
		`MOVLW <Addr[7:0]>`
		`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 (1001)

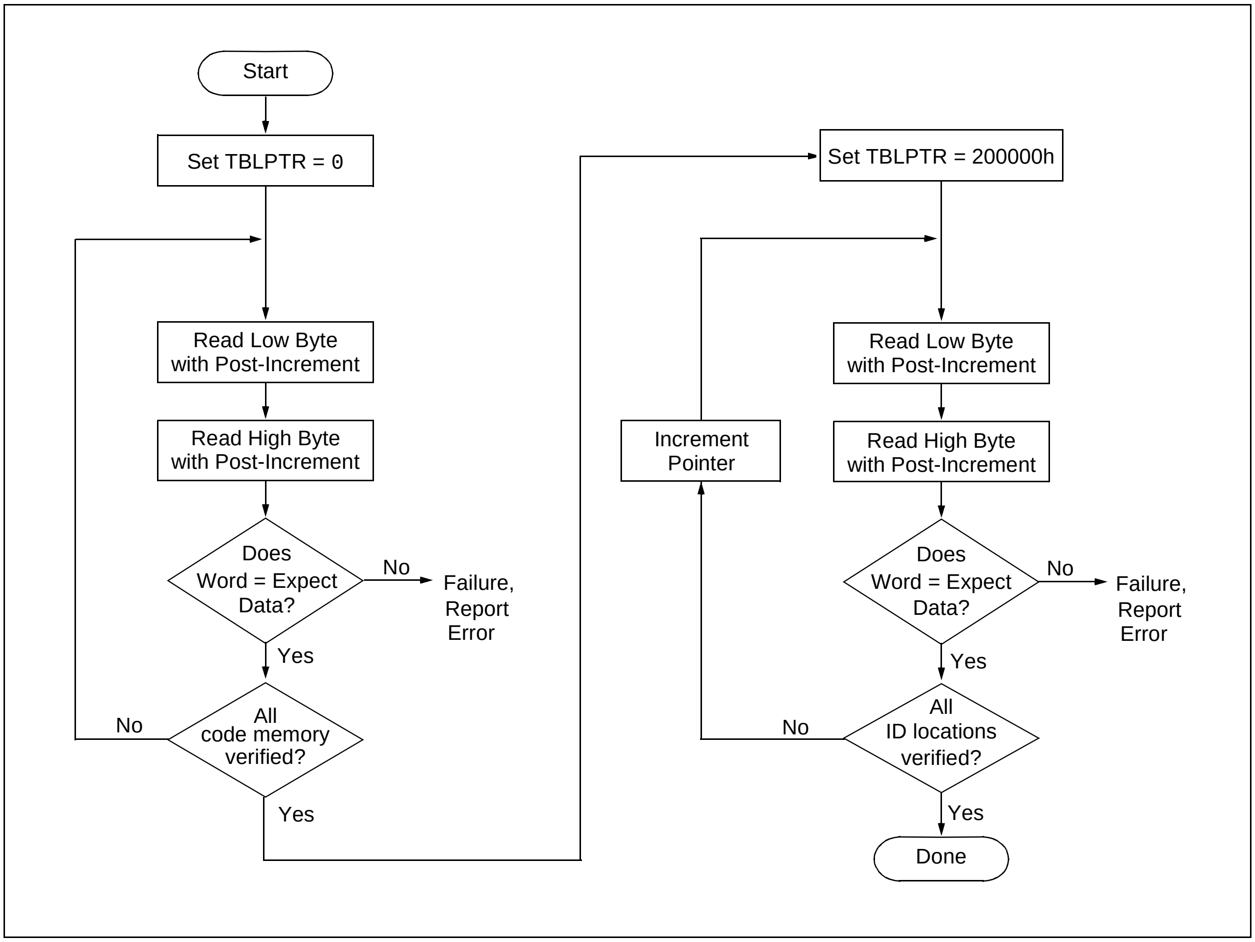


4.2 Verify Code Memory and ID Locations

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 may 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 the unimplemented address, 010000h.

FIGURE 4-2: VERIFY CODE MEMORY FLOW


Start
Set TBLPTR = 0
Read Low Byte with Post-Increment
Read High Byte with Post-Increment
Does Word = Expect Data?
No
Failure, Report Error
Yes
All code memory verified?
No
Yes
Set TBLPTR = 200000h
Increment Pointer
Read Low Byte with Post-Increment
Read High Byte with Post-Increment
Does Word = Expect Data?
No
Failure, Report Error
Yes
All ID locations verified?
No
Yes
Done

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

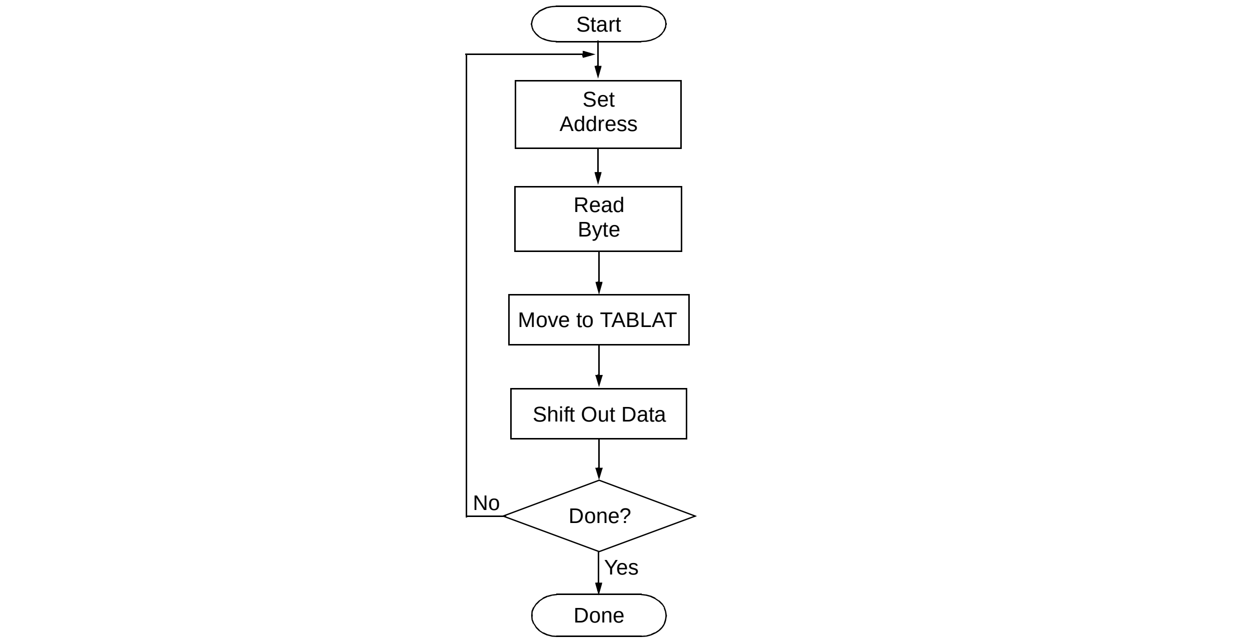
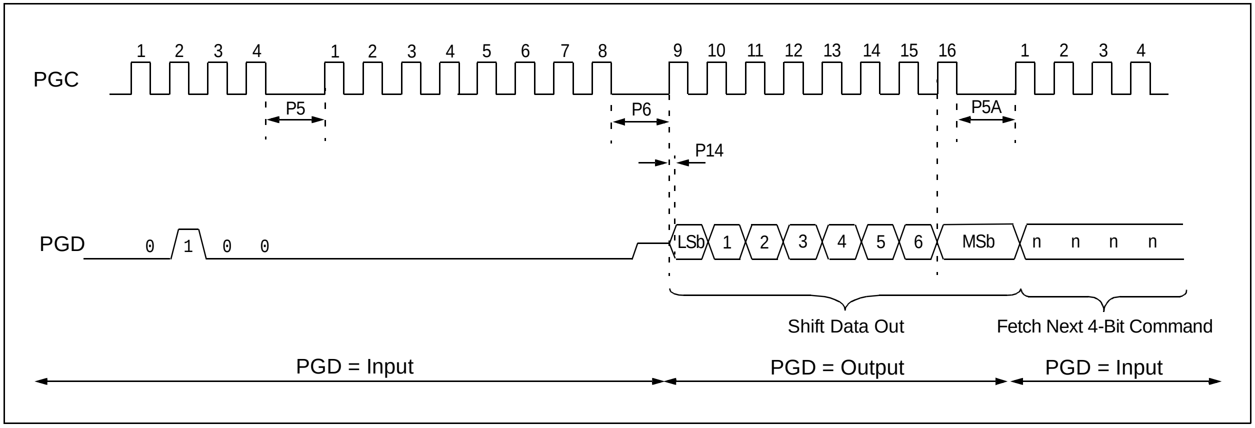


TABLE 4-2: READ DATA EEPROM MEMORY

4-Bit Command	Data Payload	Core Instruction
Step 1: Direct access to data EEPROM.		
0000 0000	9E A6 9C A6	BCF EECON1, EEPGD BCF EECON1, CFGS
Step 2: Set the data EEPROM Address Pointer.		
0000 0000 0000 0000	0E <Addr> 6E A9 0E <AddrH> 6E AA	MOVLW <Addr> MOVWF EEADR MOVLW <AddrH> 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 0000 0000 0010	50 A8 6E F5 00 00 <MSB><LSB>	MOVF EEDATA, W, 0 MOVWF TABLAT NOP Shift Out Data[1]

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

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



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 (3FFFFEh:3FFFFFh) should be ignored.

A "blank" or "erased" memory cell will read as '`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 Figure 4-5 for blank configuration expect data for the various PIC18F2XXX/4XXX Family 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 4-5: BLANK CHECK FLOW

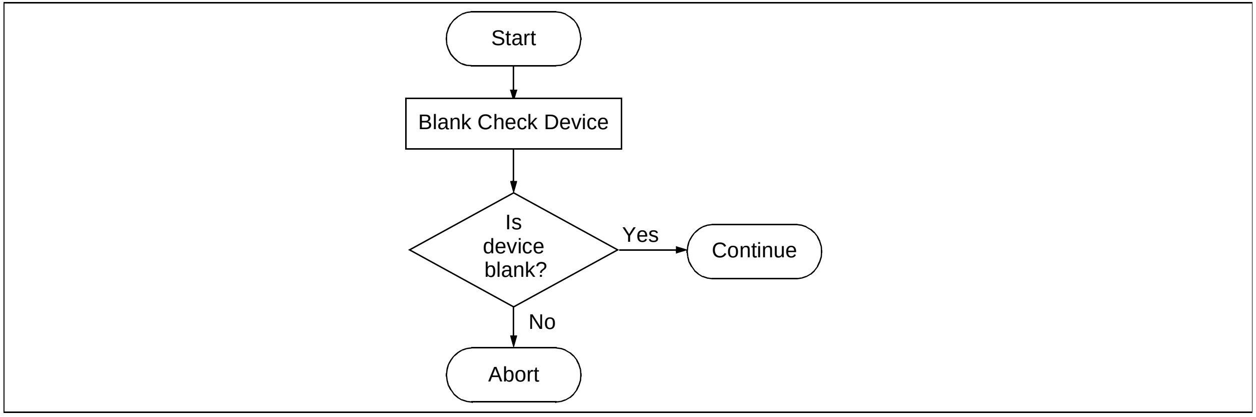


TABLE 5-1: CONFIGURATION BITS AND DEVICE IDS

File Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Default/ Unprogrammed Value
300000h[1,8]	CONFIG1L	—	—	USBDIV	CPUDIV1	CPUDIV0	PLLDIV2	PLLDIV1	PLLDIV0	`--00 0000`
300001h	CONFIG1H	IESO	FCMEN	—	—	FOSC3	FOSC2	FOSC1	FOSC0	`00-- 0111`
										`00-- 0101`[1,8]
300002h	CONFIG2L	—	—	—	BORV1	BORV0	BOREN1	BOREN0	$\overline{\text{PWRTEN}}$	`---1 1111`
				VREGEN[1,8]						`--01 1111`[1,8]
300003h	CONFIG2H	—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN	`---1 1111`
300005h	CONFIG3H	MCLRE	—	—	—	—	LPT1OSC	PBADEN	CCP2MX[7]	`1--- -011`[7]
									—	`1--- -01-`
300006h	CONFIG4L	$\overline{\text{DEBUG}}$	XINST	ICPRT[1]	—	—	LVP	—	STVREN	`100- -1-1`[1]
				BBSIZ1	BBSIZ0	—				`1000 -1-1`
				—	BBSIZ[3]	—				`10-0 -1-1`[3]
				ICPRT[8]	—	BBSIZ[8]				`100- 01-1`[8]
				BBSIZ1[2]	BBSIZ2[2]	—				`1000 -1-1`[2]
300008h	CONFIG5L	—	—	CP5[10]	CP4[9]	CP3[4]	CP2[4]	CP1	CP0	`--11 1111`
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—	`11-- ----`
30000Ah	CONFIG6L	—	—	WRT5[10]	WRT4[9]	WRT3[4]	WRT2[4]	WRT1	WRT0	`--11 1111`
30000Bh	CONFIG6H	WRTD	WRTB	WRTC[5]	—	—	—	—	—	`111- ----`
30000Ch	CONFIG7L	—	—	EBTR5[10]	EBTR4[9]	EBTR3[4]	EBTR2[4]	EBTR1	EBTR0	`--11 1111`
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—	`-1-- ----`
3FFFFEh	DEVID1[6]	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	See Table 5-2
3FFFFFh	DEVID2[6]	DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	See Table 5-2

Legend: - = unimplemented. Shaded cells are unimplemented, read as ‘0’.

Note 1: Implemented only on PIC18F2455/2550/4455/4550 and PIC18F2458/2553/4458/4553 devices.

2: Implemented on PIC18F2585/2680/4585/4680, PIC18F2682/2685 and PIC18F4682/4685 devices only.

3: Implemented on PIC18F2480/2580/4480/4580 devices only.

4: These bits are only implemented on specific devices based on available memory. Refer to **Section 2.3 “Memory Maps”**.

5: In PIC18F2480/2580/4480/4580 devices, this bit is read-only in Normal Execution mode; it can be written only in Program mode.

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

7: Implemented on all devices with the exception of the PIC18FXX8X and PIC18F2450/4450 devices.

8: Implemented on PIC18F2450/4450 devices only.

9: Implemented on PIC18F2682/2685 and PIC18F4682/4685 devices only.

10: Implemented on PIC18F2685/4685 devices only.

TABLE 5-2: DEVICE ID VALUES (CONTINUED)

Device	Device ID Value	
	DEVID2	DEVID1
PIC18F4585	0Eh	`101x xxxx`
PIC18F4610	0Ch	`001x xxxx`
PIC18F4620	0Ch	`000x xxxx`
PIC18F4680	0Eh	`100x xxxx`
PIC18F4682	27h	`010x xxxx`
PIC18F4685	27h	`011x xxxx`

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

Note 1: DEVID1 bit 4 is used to determine the device type (REV4 = `0`).

2: DEVID1 bit 4 is used to determine the device type (REV4 = `1`).

TABLE 5-3: PIC18F2XXX/4XXX FAMILY BIT DESCRIPTIONS

Bit Name	Configuration Words	Description
IESO	CONFIG1H	Internal External Switchover bit 1 = Internal External Switchover mode is enabled 0 = Internal External Switchover mode is disabled
FCMEN	CONFIG1H	Fail-Safe Clock Monitor Enable bit 1 = Fail-Safe Clock Monitor is enabled 0 = Fail-Safe Clock Monitor is disabled
FOSC<3:0>	CONFIG1H	Oscillator Selection bits 11xx = External RC oscillator, CLKO function on RA6 101x = External RC oscillator, CLKO function on RA6 1001 = Internal RC oscillator, CLKO function on RA6, port function on RA7 1000 = Internal RC oscillator, port function on RA6, port function on RA7 0111 = External RC oscillator, port function on RA6 0110 = HS oscillator, PLL is enabled (Clock Frequency = 4 x FOSC1) 0101 = EC oscillator, port function on RA6 0100 = EC oscillator, CLKO function on RA6 0011 = External RC oscillator, CLKO function on RA6 0010 = HS oscillator 0001 = XT oscillator 0000 = LP oscillator
FOSC<3:0>	CONFIG1H	Oscillator Selection bits **(PIC18F2455/2550/4455/4550, PIC18F2458/2553/4458/4553 and PIC18F2450/4450 devices only)** 111x = HS oscillator, PLL is enabled, HS is used by USB 110x = HS oscillator, HS is used by USB 1011 = Internal oscillator, HS is used by USB 1010 = Internal oscillator, XT is used by USB 1001 = Internal oscillator, CLKO function on RA6, EC is used by USB 1000 = Internal oscillator, port function on RA6, EC is used by USB 0111 = EC oscillator, PLL is enabled, CLKO function on RA6, EC is used by USB 0110 = EC oscillator, PLL is enabled, port function on RA6, EC is used by USB 0101 = EC oscillator, CLKO function on RA6, EC is used by USB 0100 = EC oscillator, port function on RA6, EC is used by USB 001x = XT oscillator, PLL is enabled, XT is used by USB 000x = XT oscillator, XT is used by USB
USBDIV	CONFIG1L	USB Clock Selection bit **(PIC18F2455/2550/4455/4550, PIC18F2458/2553/4458/4553 and PIC18F2450/4450 devices only)** Selects the clock source for full-speed USB operation: 1 = USB clock source comes from the 96 MHz PLL divided by 2 0 = USB clock source comes directly from the OSC1/OSC2 oscillator block; no divide
CPUDIV<1:0>	CONFIG1L	CPU System Clock Selection bits **(PIC18F2455/2550/4455/4550, PIC18F2458/2553/4458/4553 and PIC18F2450/4450 devices only)** 11 = CPU system clock divided by 4 10 = CPU system clock divided by 3 01 = CPU system clock divided by 2 00 = No CPU system clock divide

Note 1: The BBSIZ bits, BBSIZ<1:0> and BBSIZ<2:1> bits, cannot be changed once any of the following code-protect bits are enabled: CPB or CP0, WRTB or WRT0, EBTRB or EBTR0.

2: Not available in PIC18FXX8X and PIC18F2450/4450 devices.

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 $\overline{\text{MCLR}}$/VPP/RE3 is raised to VIHH. 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 VIHH to the $\overline{\text{MCLR}}$/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 PIC18F2XXX/4XXX Family 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 PIC18F2XXX/4XXX Family 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 Words, appropriately masked
- ID locations (if any block is code-protected)

The Least Significant 16 bits of this sum is the checksum. The contents of the data EEPROM are not used.

5.6.1 PROGRAM MEMORY

When program memory contents are summed, each 16-bit word is added to the checksum. The contents of program memory, from 000000h to the end of the last program memory block, are used for this calculation. Overflows from bit 15 may be ignored.

5.6.2 CONFIGURATION WORDS

For checksum calculations, unimplemented bits in Configuration Words should be ignored as such bits always read back as '1's. Each 8-bit Configuration Word is ANDed with a corresponding mask to prevent unused bits from affecting checksum calculations.

The mask contains a '0' in unimplemented bit positions, or a '1' where a choice can be made. When ANDed with the value read out of a Configuration Word, only implemented bits remain. A list of suitable masks is provided in Table 5-5.

5.6.3 ID LOCATIONS

Normally, the contents of these locations are defined by the user, but MPLAB® IDE provides the option of writing the device's unprotected 16-bit checksum in the 16 Most Significant bits of the ID locations (see MPLAB IDE Configure/ID Memory" menu). The lower 16 bits are not used and remain clear. This is the sum of all program memory contents and Configuration Words (appropriately masked) before any code protection is enabled.

If the user elects to define the contents of the ID locations, nothing about protected blocks can be known. If the user uses the preprotected checksum, provided by MPLAB IDE, an indirect characteristic of the programmed code is provided.

5.6.4 CODE PROTECTION

Blocks that are code-protected read back as all '0's and have no effect on checksum calculations. If any block is code-protected, then the contents of the ID locations are included in the checksum calculation.

All Configuration Words and the ID locations can always be read out normally, even when the device is fully code-protected. Checking the code protection settings in Configuration Words can direct which, if any, of the program memory blocks can be read, and if the ID locations should be used for checksum calculations.

TABLE 5-5: CONFIGURATION WORD MASKS FOR COMPUTING CHECKSUMS

Device	Configuration Word (CONFIGxx)													
	1L	1H	2L	2H	3L	3H	4L	4H	5L	5H	6L	6H	7L	7H
	Address (30000xh)													
	0h	1h	2h	3h	4h	5h	6h	7h	8h	9h	Ah	Bh	Ch	Dh
PIC18F2221	00	CF	1F	1F	00	87	F5	00	03	C0	03	E0	03	40
PIC18F2321	00	CF	1F	1F	00	87	F5	00	03	C0	03	E0	03	40
PIC18F2410	00	CF	1F	1F	00	87	C5	00	03	C0	03	E0	03	40
PIC18F2420	00	CF	1F	1F	00	87	C5	00	03	C0	03	E0	03	40
PIC18F2423	00	CF	1F	1F	00	87	C5	00	03	C0	03	E0	03	40
PIC18F2450	3F	CF	3F	1F	00	86	ED	00	03	40	03	60	03	40
PIC18F2455	3F	CF	3F	1F	00	87	E5	00	07	C0	07	E0	07	40
PIC18F2458	3F	CF	3F	1F	00	87	E5	00	07	C0	07	E0	07	40
PIC18F2480	00	CF	1F	1F	00	86	D5	00	03	C0	03	E0	03	40
PIC18F2510	00	1F	1F	1F	00	87	C5	00	0F	C0	0F	E0	0F	40
PIC18F2515	00	CF	1F	1F	00	87	C5	00	0F	C0	0F	E0	0F	40
PIC18F2520	00	CF	1F	1F	00	87	C5	00	0F	C0	0F	E0	0F	40
PIC18F2523	00	CF	1F	1F	00	87	C5	00	0F	C0	0F	E0	0F	40
PIC18F2525	00	CF	1F	1F	00	87	C5	00	0F	C0	0F	E0	0F	40
PIC18F2550	3F	CF	3F	1F	00	87	E5	00	0F	C0	0F	E0	0F	40
PIC18F2553	3F	CF	3F	1F	00	87	E5	00	0F	C0	0F	E0	0F	40
PIC18F2580	00	CF	1F	1F	00	86	D5	00	0F	C0	0F	E0	0F	40
PIC18F2585	00	CF	1F	1F	00	86	C5	00	0F	C0	0F	E0	0F	40
PIC18F2610	00	CF	1F	1F	00	87	C5	00	0F	C0	0F	E0	0F	40
PIC18F2620	00	CF	1F	1F	00	87	C5	00	0F	C0	0F	E0	0F	40
PIC18F2680	00	CF	1F	1F	00	86	C5	00	0F	C0	0F	E0	0F	40
PIC18F2682	00	CF	1F	1F	00	86	C5	00	3F	C0	3F	E0	3F	40
PIC18F2685	00	CF	1F	1F	00	86	C5	00	3F	C0	3F	E0	3F	40
PIC18F4221	00	CF	1F	1F	00	87	F5	00	03	C0	03	E0	03	40
PIC18F4321	00	CF	1F	1F	00	87	F5	00	03	C0	03	E0	03	40
PIC18F4410	00	CF	1F	1F	00	87	C5	00	03	C0	03	E0	03	40
PIC18F4420	00	CF	1F	1F	00	87	C5	00	03	C0	03	E0	03	40
PIC18F4423	00	CF	1F	1F	00	87	C5	00	03	C0	03	E0	03	40
PIC18F4450	3F	CF	3F	1F	00	86	ED	00	03	40	03	60	03	40
PIC18F4455	3F	CF	3F	1F	00	87	E5	00	07	C0	07	E0	07	40
PIC18F4458	3F	CF	3F	1F	00	87	E5	00	07	C0	07	E0	07	40
PIC18F4480	00	CF	1F	1F	00	86	D5	00	03	C0	03	E0	03	40
PIC18F4510	00	CF	1F	1F	00	87	C5	00	0F	C0	0F	E0	0F	40
PIC18F4515	00	CF	1F	1F	00	87	C5	00	0F	C0	0F	E0	0F	40
PIC18F4520	00	CF	1F	1F	00	87	C5	00	0F	C0	0F	E0	0F	40
PIC18F4523	00	CF	1F	1F	00	87	C5	00	0F	C0	0F	E0	0F	40
PIC18F4525	00	CF	1F	1F	00	87	C5	00	0F	C0	0F	E0	0F	40
PIC18F4550	3F	CF	3F	1F	00	87	E5	00	0F	C0	0F	E0	0F	40
PIC18F4553	3F	CF	3F	1F	00	87	E5	00	0F	C0	0F	E0	0F	40
PIC18F4580	00	CF	1F	1F	00	86	D5	00	0F	C0	0F	E0	0F	40
PIC18F4585	00	CF	1F	1F	00	86	C5	00	0F	C0	0F	E0	0F	40
PIC18F4610	00	CF	1F	1F	00	87	C5	00	0F	C0	0F	E0	0F	40

Legend: Shaded cells are unimplemented.