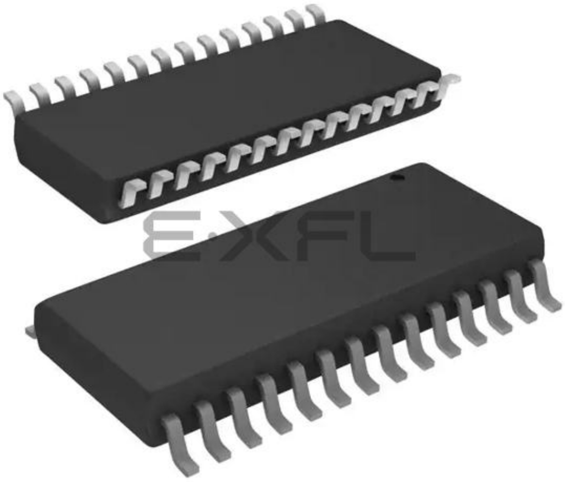
Microchip Technology - PIC18LF2510T-I/SO Datasheet



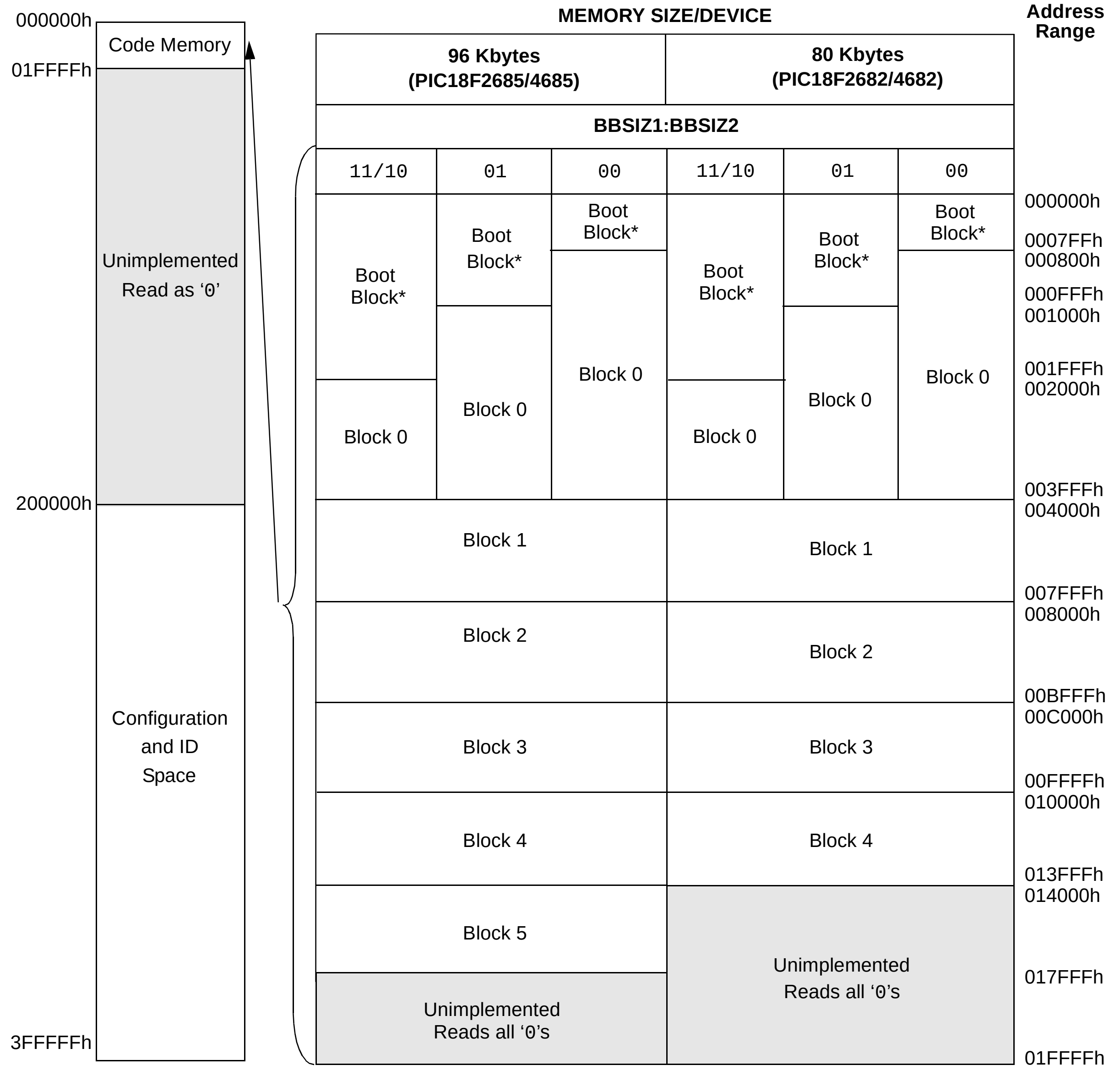
What is "Embedded - Microcontrollers"?

"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	Active
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	25
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1.5K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 5.5V
Data Converters	A/D 10x10b
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/pic18lf2510t-i-so

FIGURE 2-7: MEMORY MAP AND THE CODE MEMORY SPACE FOR PIC18F2685/4685 AND PIC18F2682/4682 DEVICES



Note: Sizes of memory areas are not to scale.

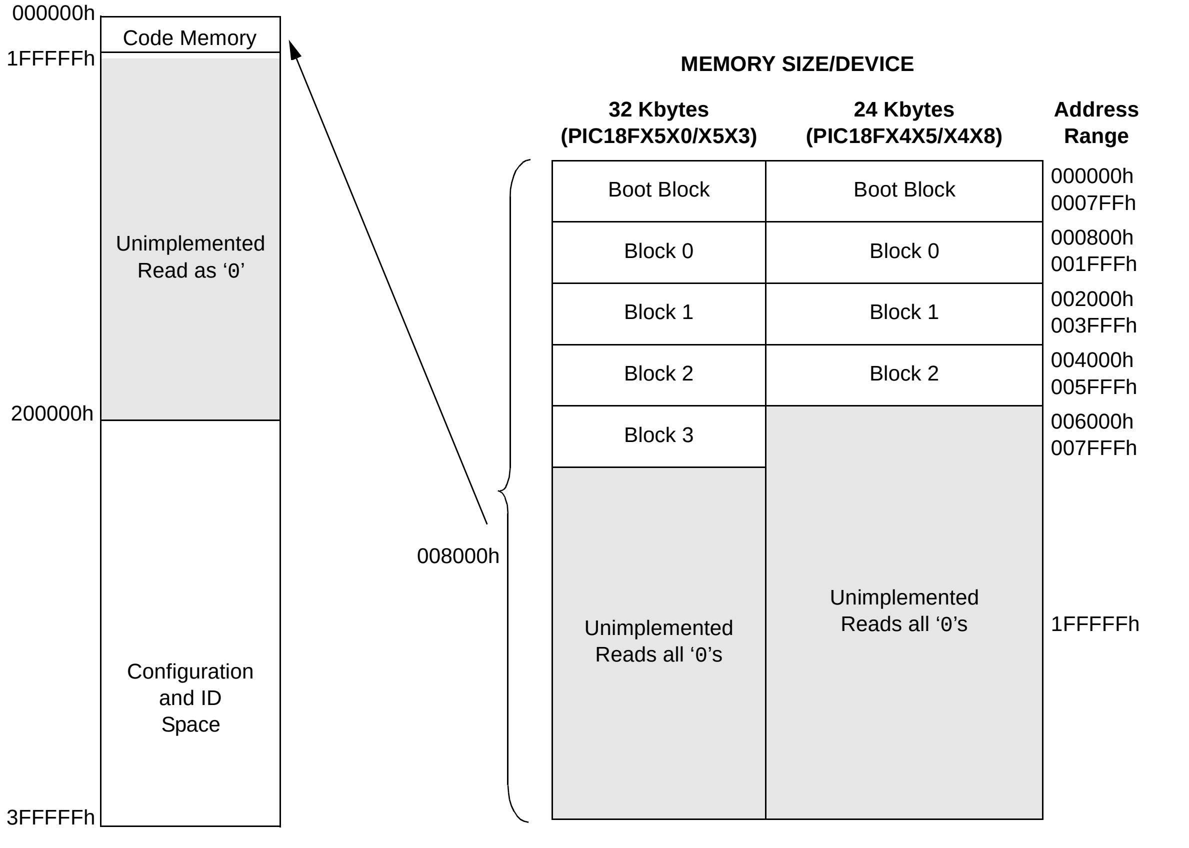
* Boot Block size is determined by the BBSIZ<1:2> bits in the CONFIG4L register.

For PIC18FX5X0/X5X3 devices, the code memory space extends from 000000h to 007FFFh (32 Kbytes) in four 8-Kbyte blocks. For PIC18FX4X5/X4X8 devices, the code memory space extends from 000000h to 005FFFh (24 Kbytes) in three 8-Kbyte blocks. Addresses, 000000h through 0007FFh, however, define a "Boot Block" region that is treated separately from Block 0. All of these blocks define code protection boundaries within the code memory space.

TABLE 2-4: 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



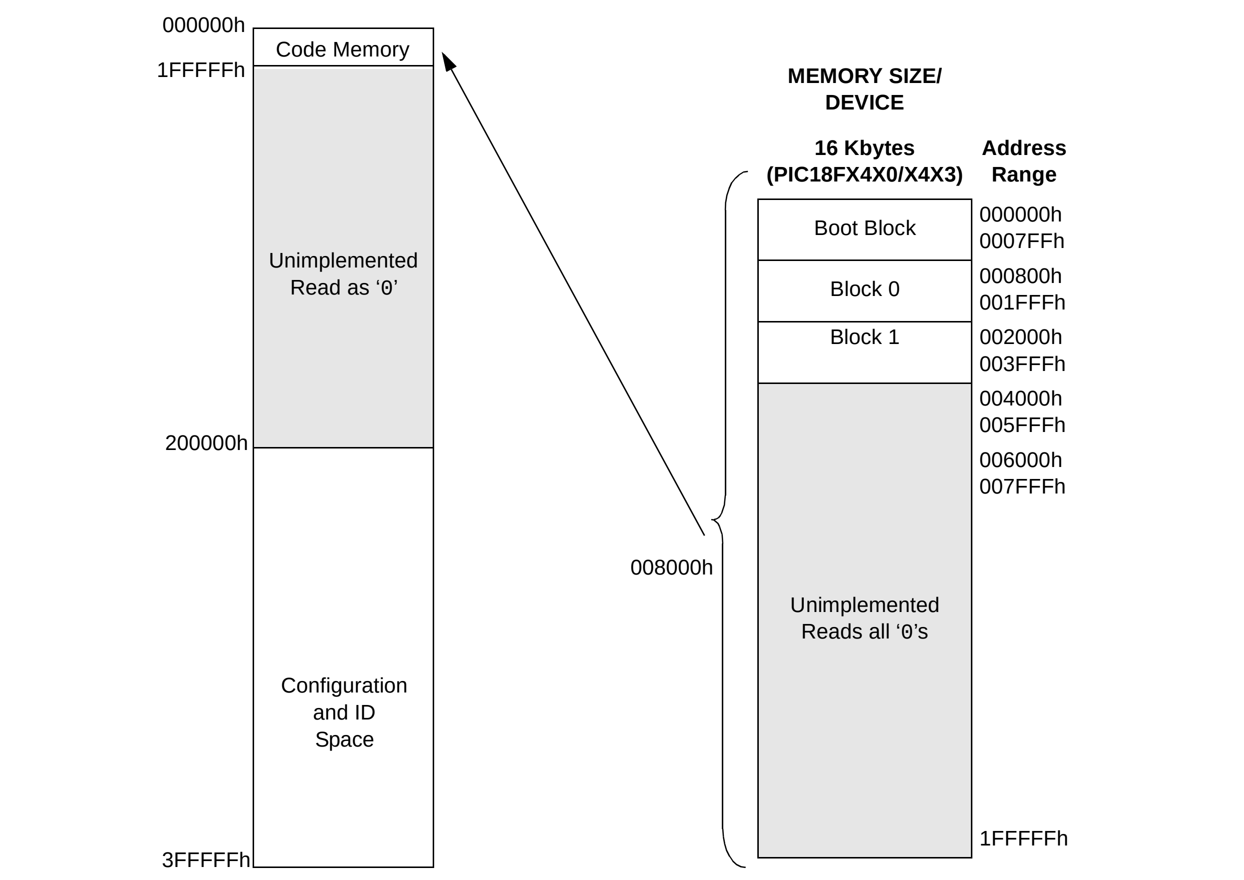
Note: Sizes of memory areas are not to scale.

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.

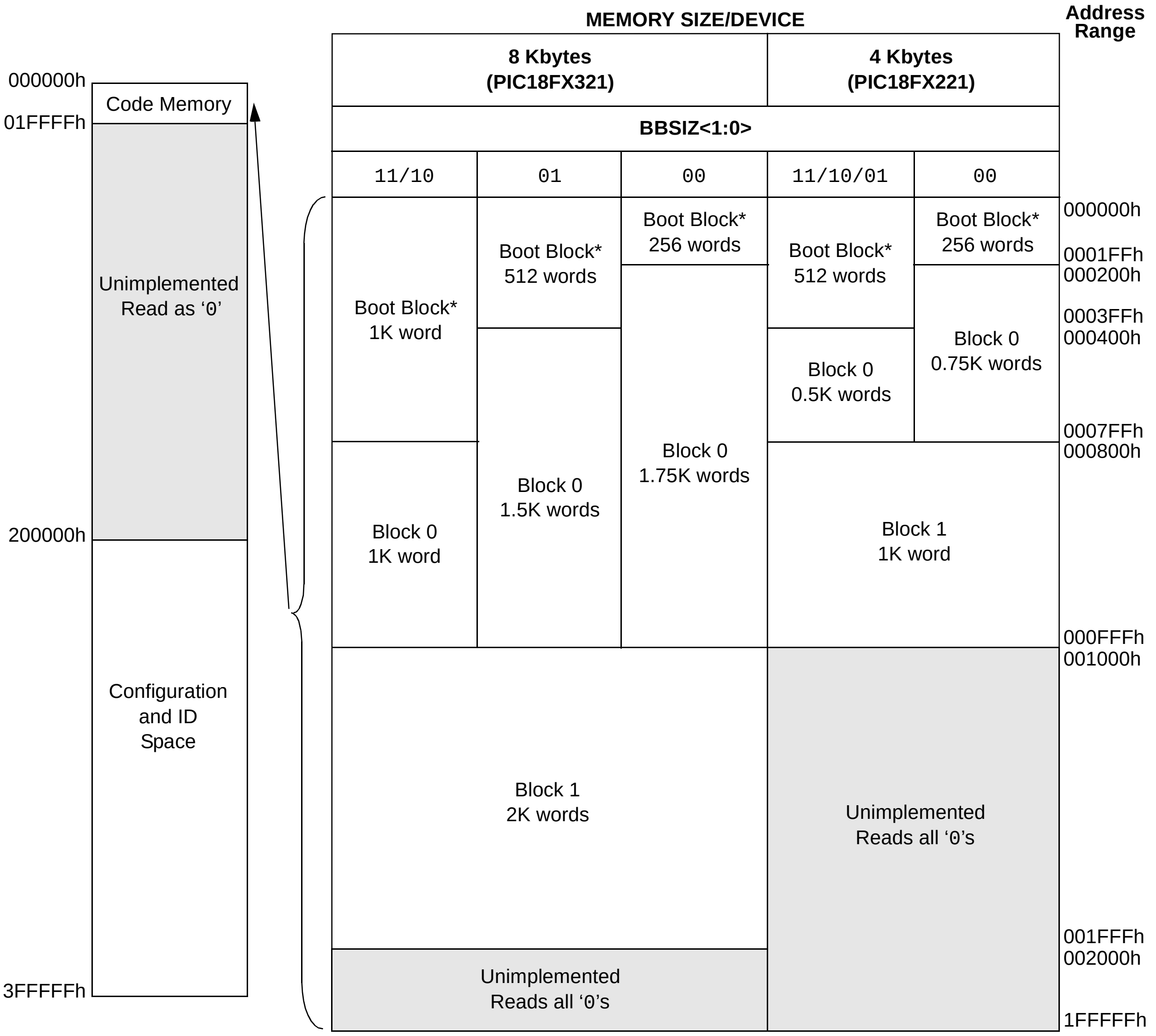
The size of the Boot Block in PIC18F2480/2580/4480/4580 devices can be configured as 1 or 2K words (see Figure 2-10). This is done through the BBSIZ<0> bit in the Configuration register, CONFIG4L. It is important to note that increasing the size of the Boot Block decreases the size of Block 0.

The size of the Boot Block in PIC18F2221/2321/4221/4321 devices can be configured as 256, 512 or 1024 words (see Figure 2-11). This is done through the BBSIZ<1:0> bits in the Configuration register, CONFIG4L (see Figure 2-11). It is important to note that increasing the size of the Boot Block decreases the size of Block 0.

TABLE 2-7: IMPLEMENTATION OF CODE MEMORY

Device	Code Memory Size (Bytes)
PIC18F2221	000000h-000FFFh (4K)
PIC18F4221	
PIC18F2321	000000h-001FFFh (8K)
PIC18F4321	

FIGURE 2-11: MEMORY MAP AND THE CODE MEMORY SPACE FOR PIC18F2221/2321/4221/4321 DEVICES



Note: Sizes of memory areas are not to scale.

* Boot Block size is determined by the BBSIZ<1:0> bits in the CONFIG4L register.

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 2-16, Low-Voltage ICSP Program/Verify mode is entered by holding PGC and PGD low, placing a logic high on PGM and then raising $\overline{\text{MCLR}}$/VPP/RE3 to VIH. In this mode, the RB5/PGM pin is dedicated to the programming function and ceases to be a general purpose I/O pin. Figure 2-17 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 2-16: ENTERING LOW-VOLTAGE PROGRAM/VERIFY MODE

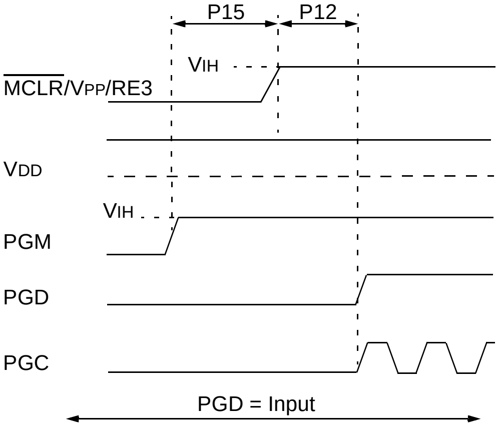
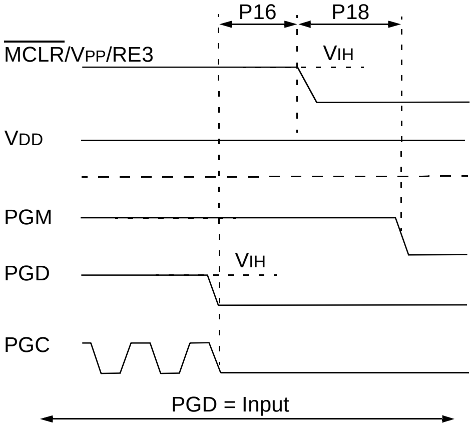


FIGURE 2-17: EXITING LOW-VOLTAGE PROGRAM/VERIFY MODE



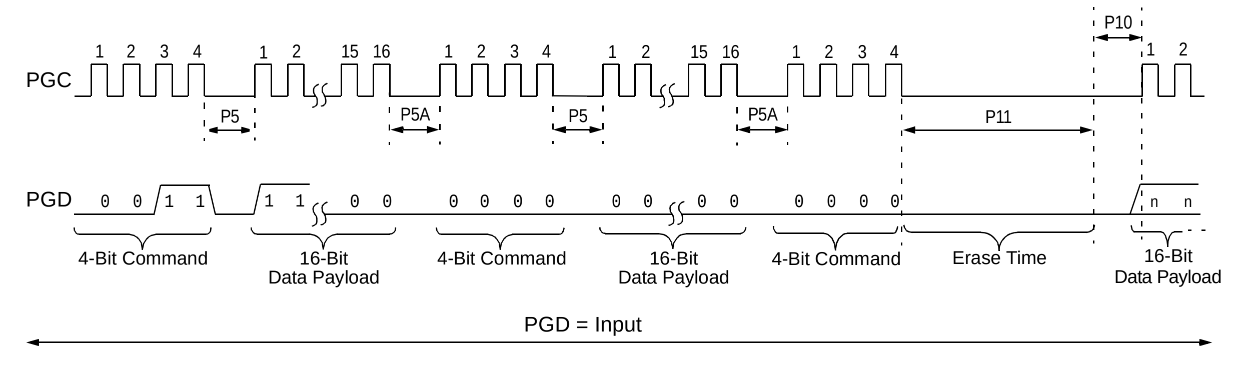
3.1.2 LOW-VOLTAGE ICSP BULK ERASE

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 (selected devices only, see **Section 3.3 "Data EEPROM Programming"**) 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.

FIGURE 3-2: BULK ERASE TIMING



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 externally timed and is controlled by PGC. After the WR bit in EECON1 is set, 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 Row Erase a PIC18F2XXX/4XXX Family device is shown in Table 3-3. The flowchart, shown in Figure 3-3, depicts the logic necessary to completely erase a PIC18F2XXX/4XXX Family 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 can point to any byte within the row intended for erase.

3.2.1 MODIFYING CODE MEMORY

The previous programming example assumed that the device had 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.		
Step 2: Read and modify code memory (see **Section 4.1 “Read Code Memory, ID Locations and Configuration Bits”**).		
0000 0000	8E A6 9C A6	BSF EECON1, EEPGD BCF EECON1, CFGS
Step 3: Set the Table Pointer for the block to be erased.		
0000 0000 0000 0000 0000 0000	0E <Addr[21:16]> 6E F8 0E <Addr[8:15]> 6E F7 0E <Addr[7:0]> 6E F6	MOVLW <Addr[21:16]> MOVWF TBLPTRU MOVLW <Addr[8:15]> MOVWF TBLPTRH MOVLW <Addr[7:0]> MOVWF TBLPTRL
Step 4: Enable memory writes and set up an erase.		
0000 0000	84 A6 88 A6	BSF EECON1, WREN BSF EECON1, FREE
Step 5: Initiate erase.		
0000 0000	82 A6 00 00	BSF EECON1, WR NOP - hold PGC high for time P9 and low for time P10.
Step 6: Load write buffer. The correct bytes will be selected based on the Table Pointer.		
0000 0000 0000 0000 0000 0000 1101 . . . 1111 0000	0E <Addr[21:16]> 6E F8 0E <Addr[8:15]> 6E F7 0E <Addr[7:0]> 6E F6 <MSB><LSB> . . . <MSB><LSB> 00 00	MOVLW <Addr[21:16]> MOVWF TBLPTRU MOVLW <Addr[8:15]> MOVWF TBLPTRH MOVLW <Addr[7:0]> MOVWF TBLPTRL 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. NOP - hold PGC high for time P9 and low for time P10.
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 7: 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 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.		

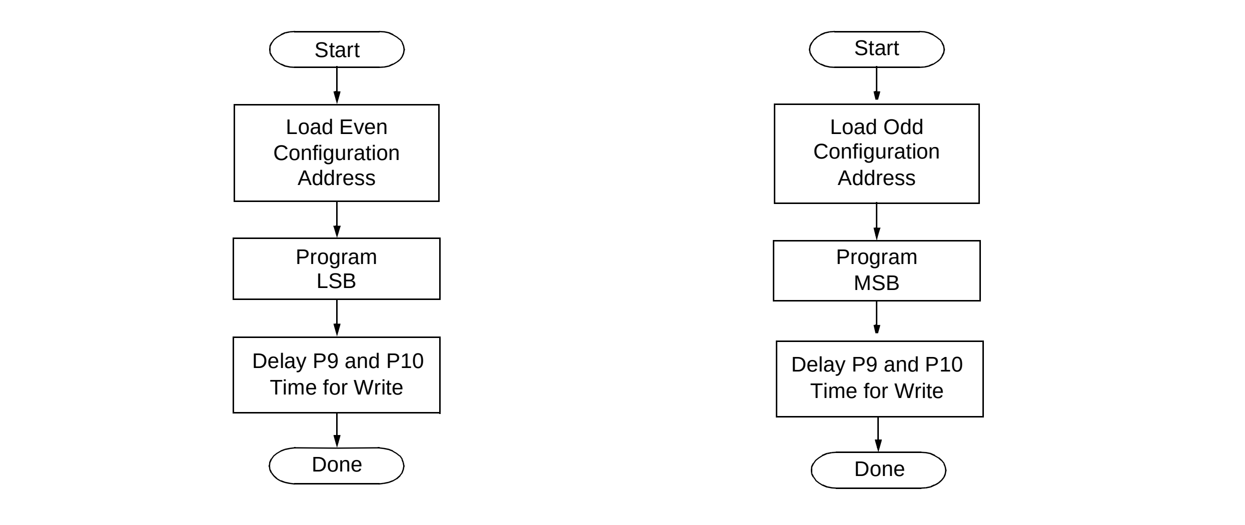
Note 1: See Figure 4-4 for details on shift out data timing.

TABLE 3-9: SET ADDRESS POINTER TO CONFIGURATION LOCATION

4-Bit Command	Data Payload	Core Instruction
Step 1: Enable writes and direct access to configuration memory.		
0000 0000	8E A6 8C A6	BSF EECON1, EEPGD BSF EECON1, CFGS
Step 2: Set Table Pointer for configuration byte to be written. Write even/odd addresses.[1]		
0000 0000 0000 0000 0000 0000 1111 0000 0000 0000 1111 0000	0E 30 6E F8 0E 00 6E F7 0E 00 6E F6 <MSB ignored><LSB> 00 00 0E 01 6E F6 <MSB><LSB ignored> 00 00	MOVLW 30h MOVWF TBLPTRU MOVLW 00h MOVWF TBLPRTH MOVLW 00h MOVWF TBLPTRL Load 2 bytes and start programming. NOP - hold PGC high for time P9 and low for time P10. MOVLW 01h MOVWF TBLPTRL Load 2 bytes and start programming. NOP - hold PGC high for time P9 and low for time P10.

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

FIGURE 3-8: CONFIGURATION PROGRAMMING FLOW

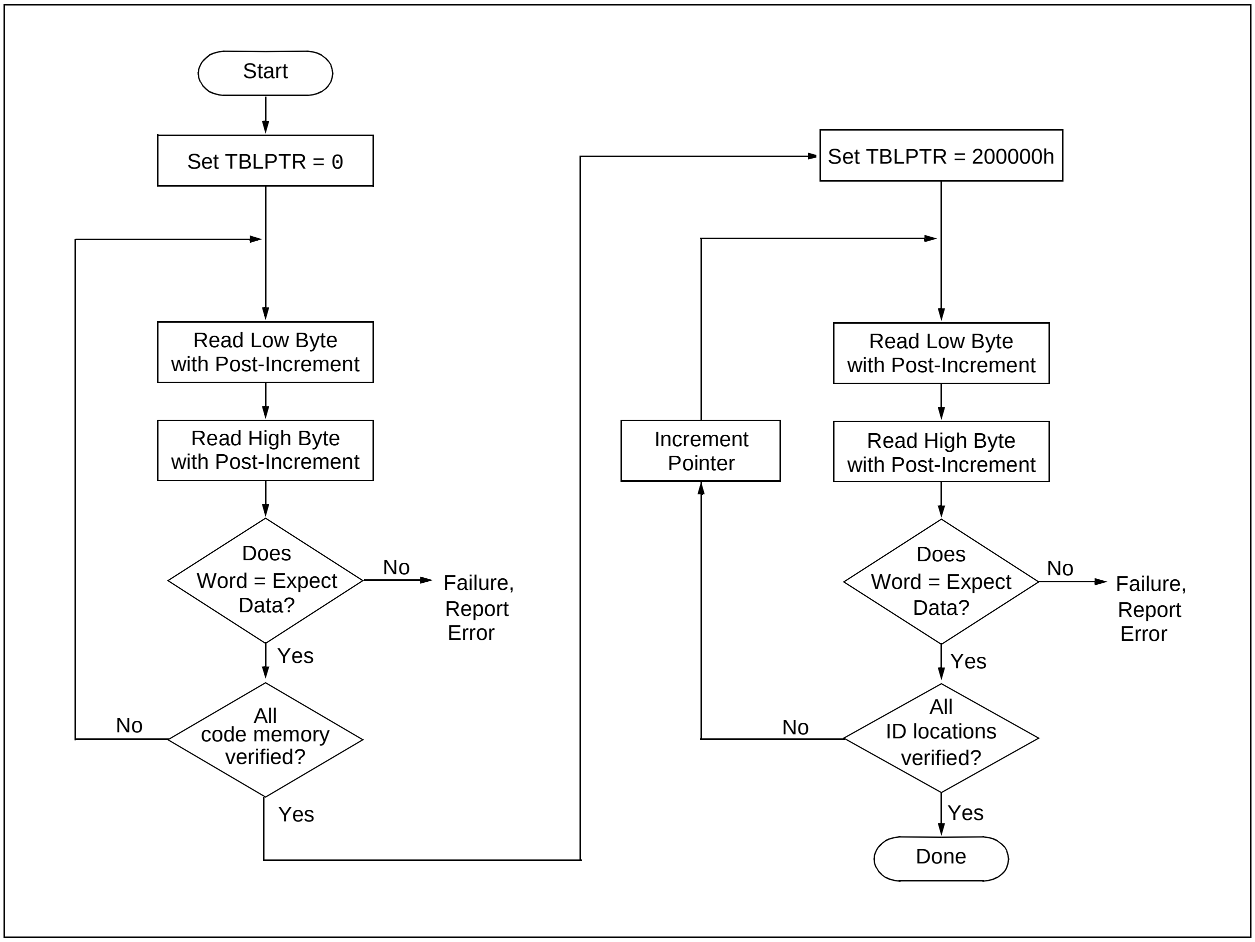

Start
Load Even Configuration Address
Program LSB
Delay P9 and P10 Time for Write
Done
Start
Load Odd Configuration Address
Program MSB
Delay P9 and P10 Time for Write
Done

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

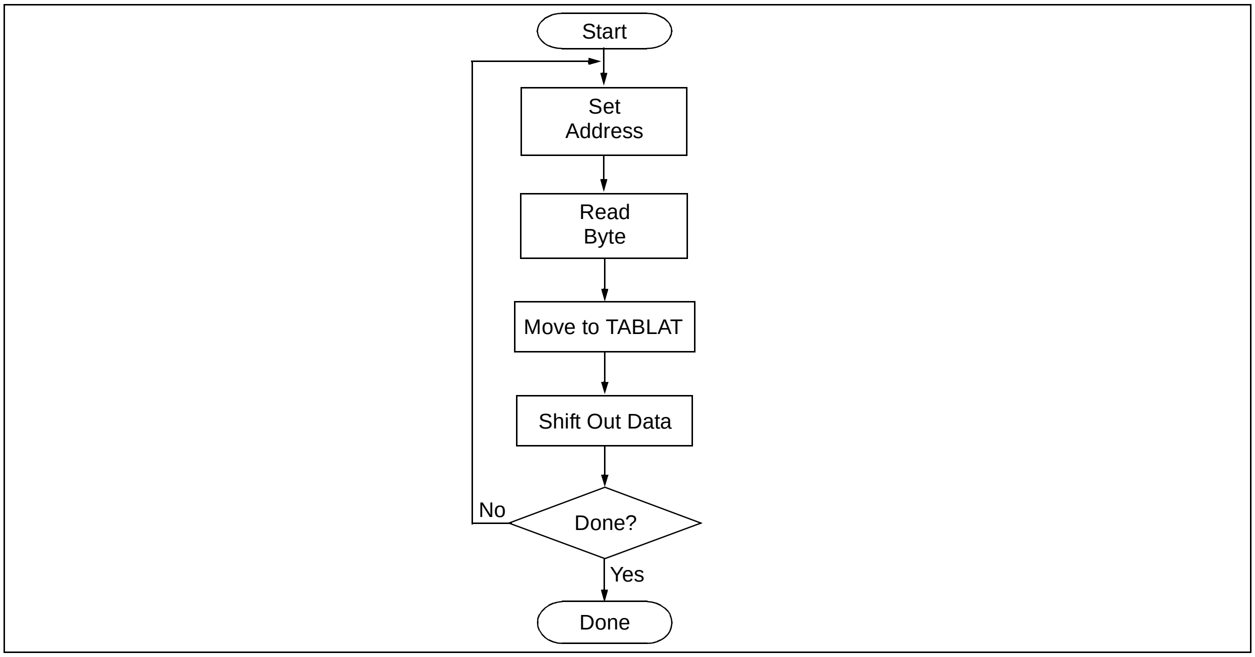
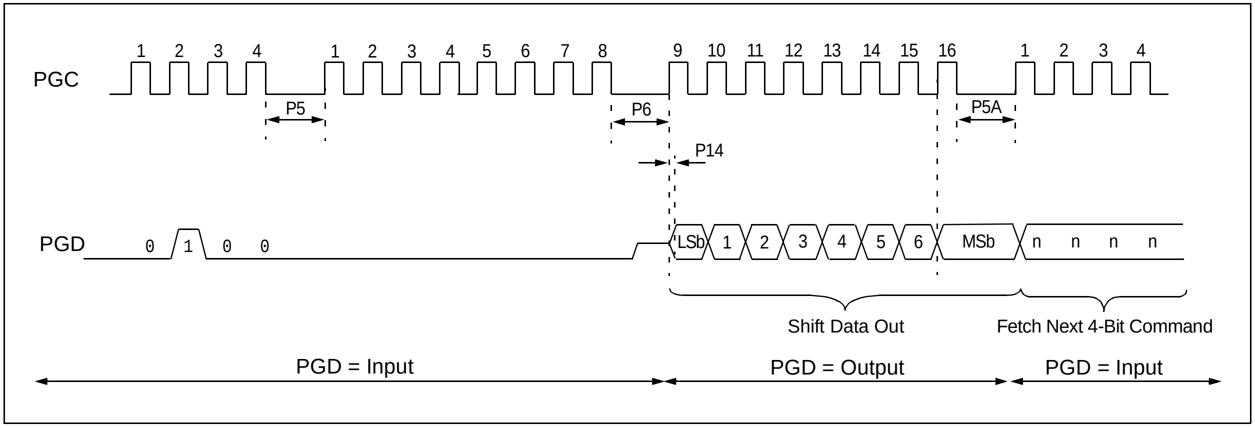


TABLE 4-2: READ DATA EEPROM MEMORY

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.

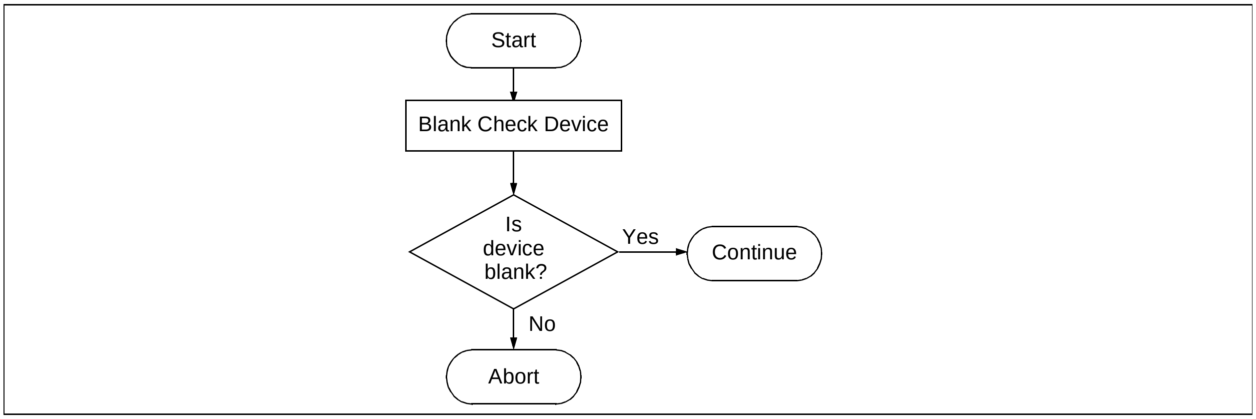
4.6 Blank Check

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



5.0 CONFIGURATION WORD

The PIC18F2XXX/4XXX Family devices have several Configuration Words. These bits can be set or cleared to select various device configurations. All other memory areas should be programmed and verified prior to setting the Configuration Words. These bits may be read out normally, even after read or code protection. See Table 5-1 for a list of Configuration bits and Device IDs, and Table 5-3 for the Configuration bit descriptions.

5.1 ID Locations

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

5.2 Device ID Word

The Device ID Word for the PIC18F2XXX/4XXX Family devices is located at 3FFFFEh:3FFFFFh. These bits may be used by the programmer to identify what device type is being programmed and read out normally, even after code or read protection.

In some cases, devices may share the same DEVID values. In such cases, the Most Significant bit of the device revision, REV4 (DEVID1<4>), will need to be examined to completely determine the device being accessed.

See Table 5-2 for a complete list of Device ID values.

FIGURE 5-1: READ DEVICE ID WORD FLOW

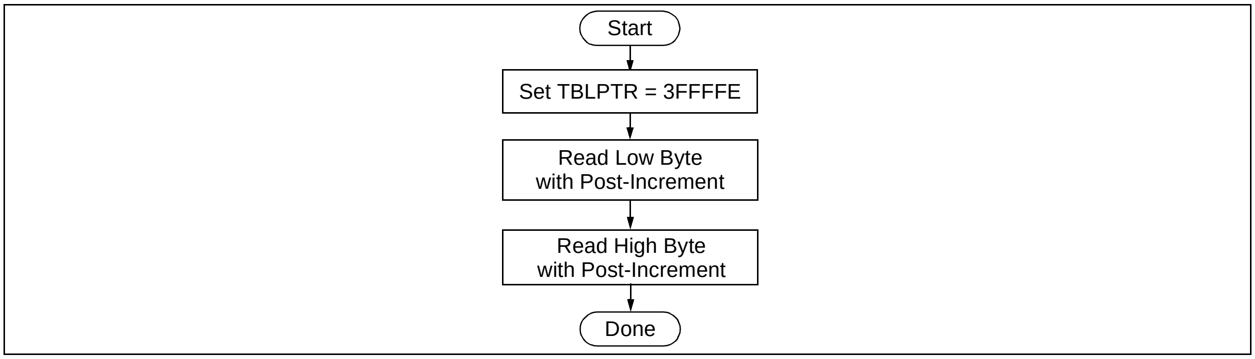

Start
Set TBLPTR = 3FFFFE
Read Low Byte with Post-Increment
Read High Byte with Post-Increment
Done

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	—	—	— VREGEN(1,8)	BORV1	BORV0	BOREN1	BOREN0	$\overline{\text{PWRTEN}}$	`---1 1111` `--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)
				BBSIZ1
				—
				ICPRT(8)
				BBSIZ1(2)
					—
					BBSIZ0
					BBSIZ(3)
					—
					BBSIZ2(2)
						—
						—
						—
						BBSIZ(8)
						—
							LVP	—	STVREN	`100- -1-1`(1)
										`1000 -1-1`
										`10-0 -1-1`(3)
										`100- 01-1`(8)
										`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

Device	Device ID Value	
	DEVID2	DEVID1
PIC18F2221	21h	`011x xxxx`
PIC18F2321	21h	`001x xxxx`
PIC18F2410	11h	`011x xxxx`
PIC18F2420	11h	`010x xxxx`(1)
PIC18F2423	11h	`010x xxxx`(2)
PIC18F2450	24h	`001x xxxx`
PIC18F2455	12h	`011x xxxx`
PIC18F2458	2Ah	`011x xxxx`
PIC18F2480	1Ah	`111x xxxx`
PIC18F2510	11h	`001x xxxx`
PIC18F2515	0Ch	`111x xxxx`
PIC18F2520	11h	`000x xxxx`(1)
PIC18F2523	11h	`000x xxxx`(2)
PIC18F2525	0Ch	`110x xxxx`
PIC18F2550	12h	`010x xxxx`
PIC18F2553	2Ah	`010x xxxx`
PIC18F2580	1Ah	`110x xxxx`
PIC18F2585	0Eh	`111x xxxx`
PIC18F2610	0Ch	`101x xxxx`
PIC18F2620	0Ch	`100x xxxx`
PIC18F2680	0Eh	`110x xxxx`
PIC18F2682	27h	`000x xxxx`
PIC18F2685	27h	`001x xxxx`
PIC18F4221	21h	`010x xxxx`
PIC18F4321	21h	`000x xxxx`
PIC18F4410	10h	`111x xxxx`
PIC18F4420	10h	`110x xxxx`(1)
PIC18F4423	10h	`110x xxxx`(2)
PIC18F4450	24h	`000x xxxx`
PIC18F4455	12h	`001x xxxx`
PIC18F4458	2Ah	`001x xxxx`
PIC18F4480	1Ah	`101x xxxx`
PIC18F4510	10h	`101x xxxx`
PIC18F4515	0Ch	`011x xxxx`
PIC18F4520	10h	`100x xxxx`(1)
PIC18F4523	10h	`100x xxxx`(2)
PIC18F4525	0Ch	`010x xxxx`
PIC18F4550	12h	`000x xxxx`
PIC18F4553	2Ah	`000x xxxx`
PIC18F4580	1Ah	`100x 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 5-2: 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 (CONTINUED)

Bit Name	Configuration Words	Description
WRT5	CONFIG6L	Write Protection bit (Block 5 code memory area) **(PIC18F2685 and PIC18F4685 devices only)** 1 = Block 5 is not write-protected 0 = Block 5 is write-protected
WRT4	CONFIG6L	Write Protection bit (Block 4 code memory area) **(PIC18F2682/2685 and PIC18F4682/4685 devices only)** 1 = Block 4 is not write-protected 0 = Block 4 is write-protected
WRT3	CONFIG6L	Write Protection bit (Block 3 code memory area) 1 = Block 3 is not write-protected 0 = Block 3 is write-protected
WRT2	CONFIG6L	Write Protection bit (Block 2 code memory area) 1 = Block 2 is not write-protected 0 = Block 2 is write-protected
WRT1	CONFIG6L	Write Protection bit (Block 1 code memory area) 1 = Block 1 is not write-protected 0 = Block 1 is write-protected
WRT0	CONFIG6L	Write Protection bit (Block 0 code memory area) 1 = Block 0 is not write-protected 0 = Block 0 is write-protected
WRTD	CONFIG6H	Write Protection bit (Data EEPROM) 1 = Data EEPROM is not write-protected 0 = Data EEPROM is write-protected
WRTB	CONFIG6H	Write Protection bit (Boot Block memory area) 1 = Boot Block is not write-protected 0 = Boot Block is write-protected
WRTC	CONFIG6H	Write Protection bit (Configuration registers) 1 = Configuration registers are not write-protected 0 = Configuration registers are write-protected
EBTR5	CONFIG7L	Table Read Protection bit (Block 5 code memory area) **(PIC18F2685 and PIC18F4685 devices only)** 1 = Block 5 is not protected from Table Reads executed in other blocks 0 = Block 5 is protected from Table Reads executed in other blocks
EBTR4	CONFIG7L	Table Read Protection bit (Block 4 code memory area) **(PIC18F2682/2685 and PIC18F4682/4685 devices only)** 1 = Block 4 is not protected from Table Reads executed in other blocks 0 = Block 4 is protected from Table Reads executed in other blocks
EBTR3	CONFIG7L	Table Read Protection bit (Block 3 code memory area) 1 = Block 3 is not protected from Table Reads executed in other blocks 0 = Block 3 is protected from Table Reads executed in other blocks
EBTR2	CONFIG7L	Table Read Protection bit (Block 2 code memory area) 1 = Block 2 is not protected from Table Reads executed in other blocks 0 = Block 2 is protected from Table Reads executed in other blocks
EBTR1	CONFIG7L	Table Read Protection bit (Block 1 code memory area) 1 = Block 1 is not protected from Table Reads executed in other blocks 0 = Block 1 is protected from Table Reads executed in other blocks

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.

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 $\overline{\text{MCLR}}$/VPP/RE3	VDD + 4.0	12.5	V	**(Note 2)**
D110A	VIHL	Low-Voltage Programming Voltage on $\overline{\text{MCLR}}$/VPP/RE3	2.00	5.50	V	**(Note 2)**
D111	VDD	Supply Voltage During Programming	2.00	5.50	V	Externally timed, Row Erases and all writes
			3.0	5.50	V	Self-timed, Bulk Erases only **(Note 3)**
D112	IPP	Programming Current on $\overline{\text{MCLR}}$/VPP/RE3	—	300	μA	**(Note 2)**
D113	IDDP	Supply Current During Programming	—	10	mA	
D031	VIL	Input Low Voltage	VSS	0.2 VDD	V	
D041	VIH	Input High Voltage	0.8 VDD	VDD	V	
D080	VOL	Output Low Voltage	—	0.6	V	IOL = 8.5 mA @ 4.5V
D090	VOH	Output High Voltage	VDD – 0.7	—	V	IOH = -3.0 mA @ 4.5V
D012	CIO	Capacitive Loading on I/O pin (PGD)	—	50	pF	To meet AC specifications
P1	TR	$\overline{\text{MCLR}}$/VPP/RE3 Rise Time to Enter Program/Verify mode	—	1.0	μs	**(Notes 1, 2)**
P2	TPGC	Serial Clock (PGC) Period	100	—	ns	VDD = 5.0V
			1	—	μs	VDD = 2.0V
P2A	TPGCL	Serial Clock (PGC) Low Time	40	—	ns	VDD = 5.0V
			400	—	ns	VDD = 2.0V
P2B	TPGCH	Serial Clock (PGC) High Time	40	—	ns	VDD = 5.0V
			400	—	ns	VDD = 2.0V
P3	TSET1	Input Data Setup Time to Serial Clock ↓	15	—	ns	
P4	THLD1	Input Data Hold Time from PGC ↓	15	—	ns	
P5	TDLY1	Delay Between 4-Bit Command and Command Operand	40	—	ns	
P5A	TDLY1A	Delay Between 4-Bit Command Operand and Next 4-Bit Command	40	—	ns	
P6	TDLY2	Delay Between Last PGC ↓ of Command Byte to First PGC ↑ of Read of Data Word	20	—	ns	
P9	TDLY5	PGC High Time (minimum programming time)	1	—	ms	Externally timed
P10	TDLY6	PGC Low Time After Programming (high-voltage discharge time)	100	—	μs	
P11	TDLY7	Delay to Allow Self-Timed Data Write or Bulk Erase to Occur	5	—	ms	

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

2: When ICPRT = 1, this specification also applies to ICVPP.

3: At 0°C-50°C.

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 unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, dsPIC, FlashFlex, flexPWR, JukeBlox, KEELOQ, KEELOQ logo, Kleer, LANCheck, MediaLB, MOST, MOST logo, MPLAB, OptoLyzer, PIC, PICSTART, PIC^{32} logo, RightTouch, SpyNIC, SST, SST Logo, SuperFlash and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

The Embedded Control Solutions Company and mTouch are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, BodyCom, chipKIT, chipKIT logo, CodeGuard, dsPICDEM, dsPICDEM.net, ECAN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, KleerNet, KleerNet logo, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, RightTouch logo, REAL ICE, SQI, Serial Quad I/O, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, 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.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

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

ISBN: 978-1-63277-856-7

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Microchip received ISO/TS-16949:2009 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.