Microchip Technology - PIC18F8620T-E/PT 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	25MHz
Connectivity	EBI/EMI, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	68
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.75K x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	80-TQFP
Supplier Device Package	80-TQFP (12x12)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f8620t-e-pt

In addition to the code memory space, there are three blocks in the configuration and ID space that are accessible to the user through Table Reads and Table Writes. Their locations in the memory map are shown in Figure 2-3.

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. 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. These Device ID bits read out normally, even after code protection.

2.3.1 MEMORY ADDRESS POINTER

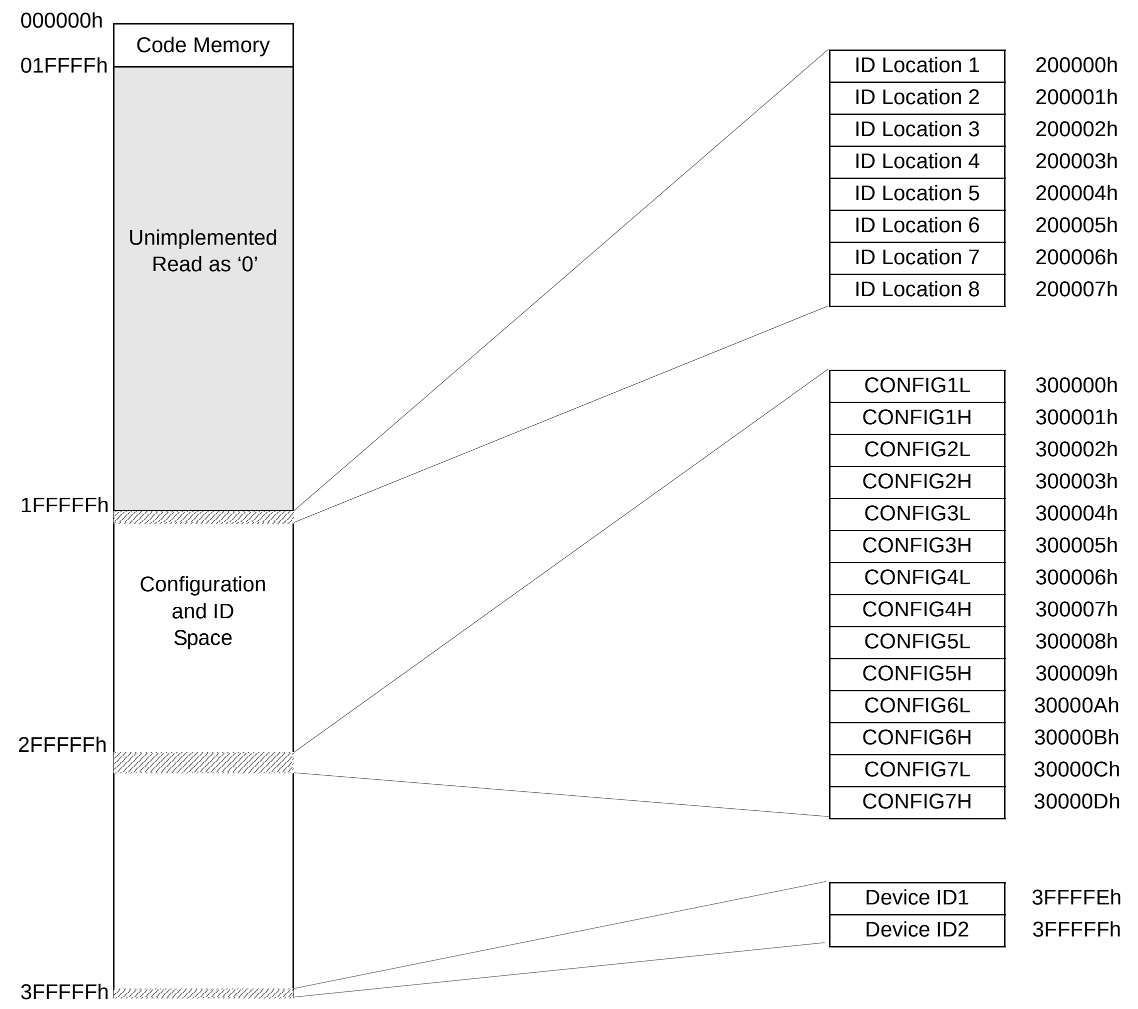
Memory in the address space 0000000h to 3FFFFFh is addressed via the Table Pointer, which is comprised of three pointer registers:

- 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.

FIGURE 2-3: CONFIGURATION AND ID LOCATIONS FOR PIC18FXX20 DEVICES



Note: Sizes of memory areas are not to scale.

TABLE 3-2: ERASE CODE MEMORY CODE SEQUENCE

4-Bit Command	Data Payload	Core Instruction
Step 1: Direct access to config memory.		

0000
0000
0000
	8E A6
	8C A6
	86 A6
		`BSF   EECON1, EEPGD`
		`BSF   EECON1, CFGS`
		`BSF   EECON1, WREN`

Step 2: Configure device for multi-panel writes.		

0000
0000
0000
0000
0000
0000
1100
	0E 3C
	6E F8
	0E 00
	6E F7
	0E 06
	6E F6
	00 40
		`MOVLW 3Ch`
		`MOVWF TBLPTRU`
		`MOVLW 00h`
		`MOVWF TBLPTRH`
		`MOVLW 06h`
		`MOVWF TBLPTRL`
		`Write 40h to 3C0006h to enable multi-panel erase.`

Step 3: Direct access to code memory and enable erase.		

0000
0000
0000
0000
0000
0000
	8E A6
	9C A6
	88 A6
	6A F8
	6A F7
	6A F6
		`BSF   EECON1, EEPGD`
		`BCF   EECON1, CFGS`
		`BSF   EECON1, FREE`
		`CLRF  TBLPTRU`
		`CLRF  TBLPTRH`
		`CLRF  TBLPTRL`

Step 4: Erase single row of all panels at an offset.		

1111
 	
0000
	<DummyLSB>
	<DummyMSB>
	00 00
		`Write 2 dummy bytes and start programming.`
		
		`NOP - hold SCLK high for time P9.`

Step 5: Repeat step 4, with Address Pointer incremented by 64 until all panels are erased.		

FIGURE 3-4: MULTI-PANEL SINGLE ROW ERASE CODE MEMORY FLOW

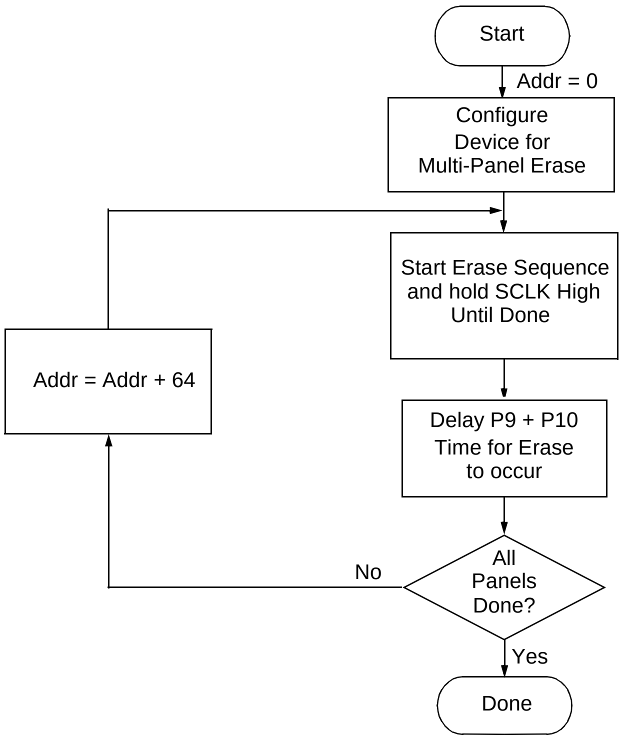
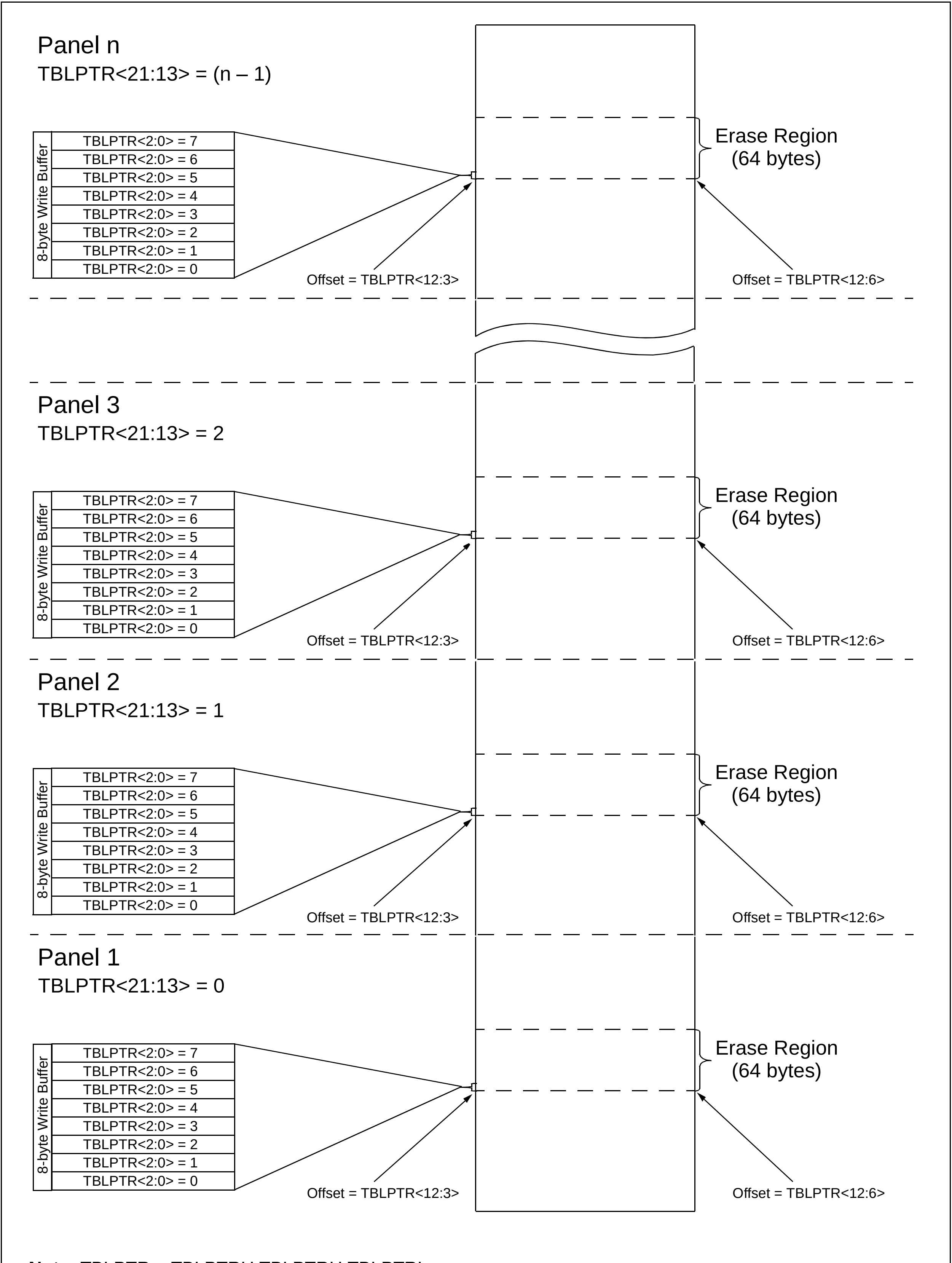

Start
Addr = 0
Configure Device for Multi-Panel Erase
Start Erase Sequence and hold SCLK High Until Done
Addr = Addr + 64
Delay P9 + P10 Time for Erase to occur
All Panels Done?
No
Yes
Done

FIGURE 3-5: ERASE AND WRITE BOUNDARIES



Note: TBLPTR = TBLPTRU:TBLPTRH:TBLPTRL.

TABLE 3-3: WRITE CODE MEMORY CODE SEQUENCE

4-Bit Command	Data Payload	Core Instruction
Step 1: Direct access to config memory.		
0000	8E A6	BSF EECON1, EEPGD
0000	8C A6	BSF EECON1, CFGS
0000	86 A6	BSF EECON1, WREN
Step 2: Configure device for multi-panel writes.		
0000	0E 3C	MOVLW 3Ch
0000	6E F8	MOVWF TBLPTRU
0000	0E 00	MOVLW 00h
0000	6E F7	MOVWF TBLPTRH
0000	0E 06	MOVLW 06h
0000	6E F6	MOVWF TBLPTRL
1100	00 40	Write 40h to 3C0006h to enable multi-panel writes.
Step 3: Direct access to code memory.		
0000	8E A6	BSF EECON1, EEPGD
0000	9C A6	BCF EECON1, CFGS
Step 4: Load write buffer for Panel 1.		
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
1101	<LSB><MSB>	Write 2 bytes and post-increment address by 2
1101	<LSB><MSB>	Write 2 bytes and post-increment address by 2
1101	<LSB><MSB>	Write 2 bytes and post-increment address by 2
1100	<LSB><MSB>	Write 2 bytes
Step 5: Repeat for Panel 2.		
Step 6: Repeat for all but the last panel (N – 1).		
Step 7: Load write buffer for last panel.		
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
1101	<LSB><MSB>	Write 2 bytes and post-increment address by 2
1101	<LSB><MSB>	Write 2 bytes and post-increment address by 2
1101	<LSB><MSB>	Write 2 bytes and post-increment address by 2
1111	<LSB><MSB>	Write 2 bytes and start programming
0000	00 00	NOP - hold SCLK high for time P9
To continue writing data, repeat steps 2 through 5, where the Address Pointer is incremented by 8 in each panel at each iteration of the loop.		

FIGURE 3-6: PROGRAM CODE MEMORY FLOW

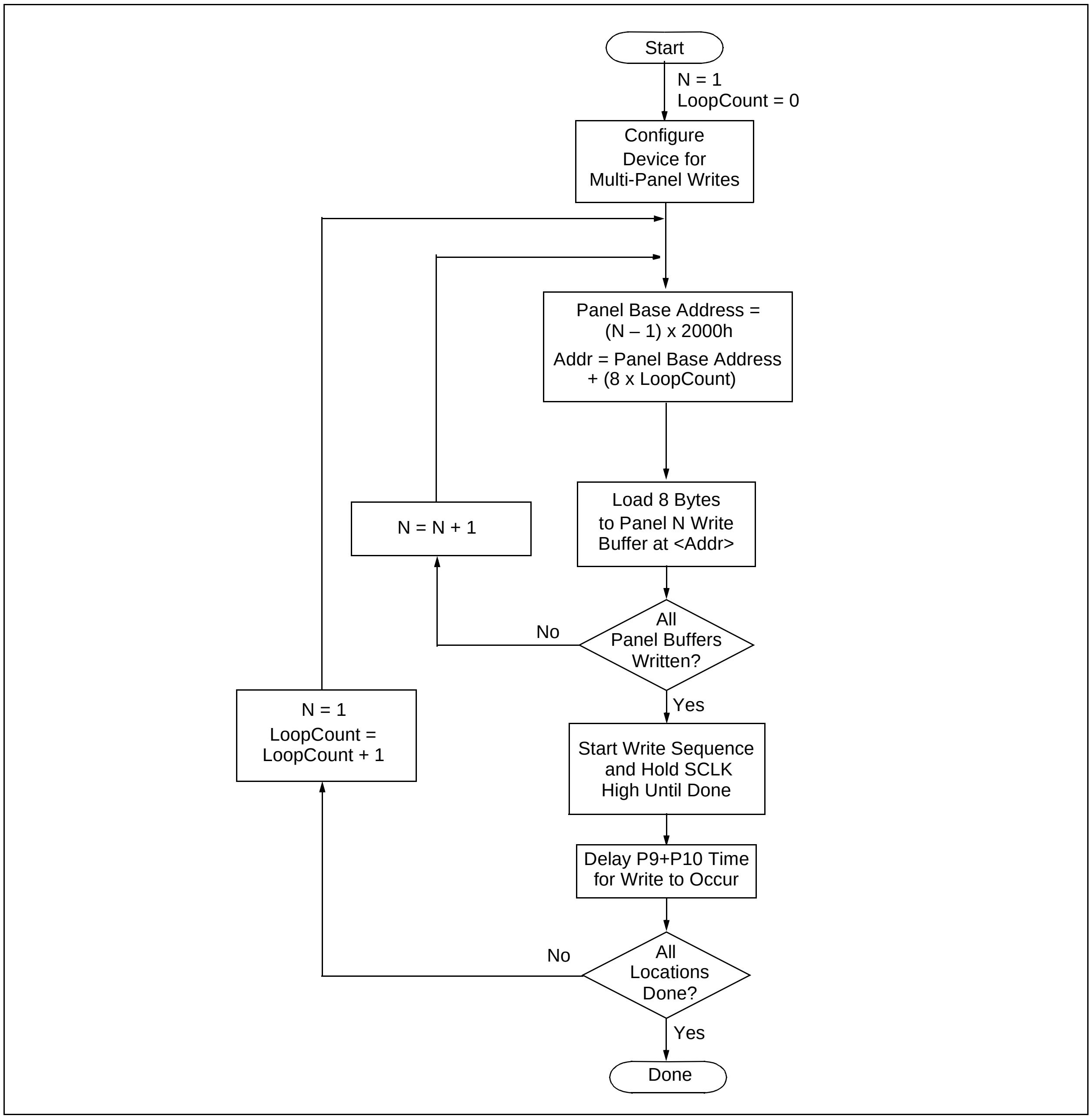
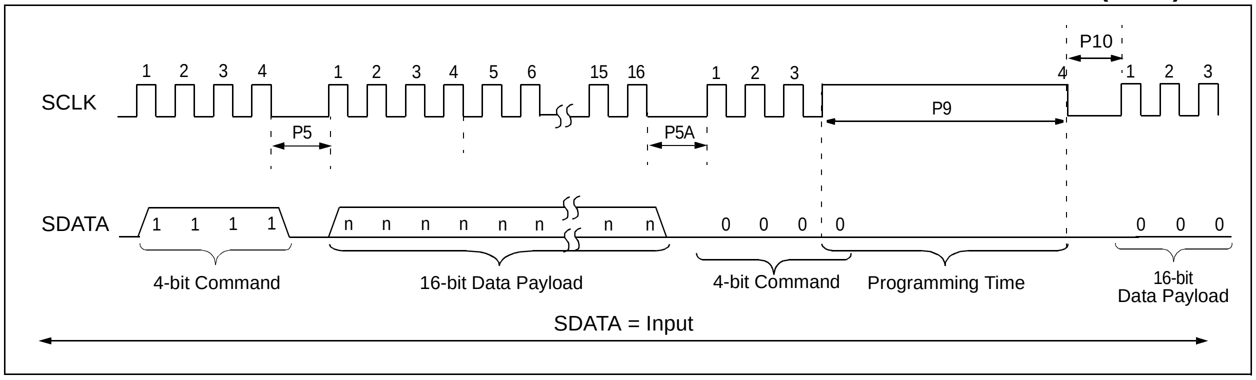

Start
N = 1
LoopCount = 0
Configure Device for Multi-Panel Writes
Panel Base Address = (N – 1) x 2000h
Addr = Panel Base Address + (8 x LoopCount)
Load 8 Bytes to Panel N Write Buffer at <Addr>
N = N + 1
All Panel Buffers Written?
No
Yes
N = 1
LoopCount = LoopCount + 1
Start Write Sequence and Hold SCLK High Until Done
Delay P9+P10 Time for Write to Occur
All Locations Done?
No
Yes
Done

FIGURE 3-7: TABLE WRITE AND START PROGRAMMING INSTRUCTION TIMING (`1111`)



3.2.1 SINGLE PANEL PROGRAMMING

The programming example presented in Section 3.2 utilizes multi-panel programming. This technique greatly decreases the total amount of time necessary to completely program a device and is the recommended method of completely programming a device.

There may be situations, however, where it is advantageous to limit writes to a single panel. In such cases, the user only needs to disable the multi-panel write feature of the device by appropriately configuring the programming control register located at 3C0006h.

The single panel that will be written will automatically be enabled based on the value of the Table Pointer.

Note:	Even though multi-panel writes are disabled, the user must still fill the 8-byte write buffer for the given panel.

3.2.2 MODIFYING CODE MEMORY

All of the programming examples up to this point have assumed that the device has been bulk erased prior to programming (see Section 3.1). It may be the case, however, that the user wishes to modify only a section of an already programmed device.

The minimum amount of data that can be written to the device is 8 bytes. This is accomplished by placing the device in Single Panel Write mode (see Section 3.2.1), loading the 8-byte write buffer for the panel, and then initiating a write sequence. In this case, however, it is assumed that the address space to be written already has data in it (i.e., it is not blank).

The minimum amount of code memory that may be erased at a given time is 64 bytes. Again, the device must be placed in Single Panel Write mode. The EECON1 register must then be used to erase the 64-byte target space prior to writing the data.

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 sequence is initiated by the setting the WR bit (EECON1<1> = 1). It is strongly recommended that the WREN bit be set only when absolutely necessary.

To help prevent inadvertent writes when using the EECON1 register, EECON2 is used to “enable” the WR bit. This register must be sequentially loaded with 55h and then AAh, immediately prior to asserting the WR bit in order for the write to occur.

The erase will begin on the falling edge of the 4th SCLK, after the WR bit is set. After the erase sequence terminates, SCLK must still be held low for the time specified by parameter #P10 to allow high voltage discharge of the memory array.

3.3 Data EEPROM Programming

Data EEPROM is accessed one byte at a time via an Address Pointer (register pair EEADR:EEADRH) and a data latch (EEDATA). Data EEPROM is written by loading EEADR:EEADRH with the desired memory location, EEDATA with the data to be written, and initiating a memory write by appropriately configuring the EECON1 and EECON2 registers. 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`). It is strongly recommended that the WREN bit be set only when absolutely necessary.

To help prevent inadvertent writes when using the EECON1 register, EECON2 is used to “enable” the WR bit. This register must be sequentially loaded with 55h and then AAh, immediately prior to asserting the WR bit in order for the write to occur.

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

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

FIGURE 3-8: PROGRAM DATA FLOW

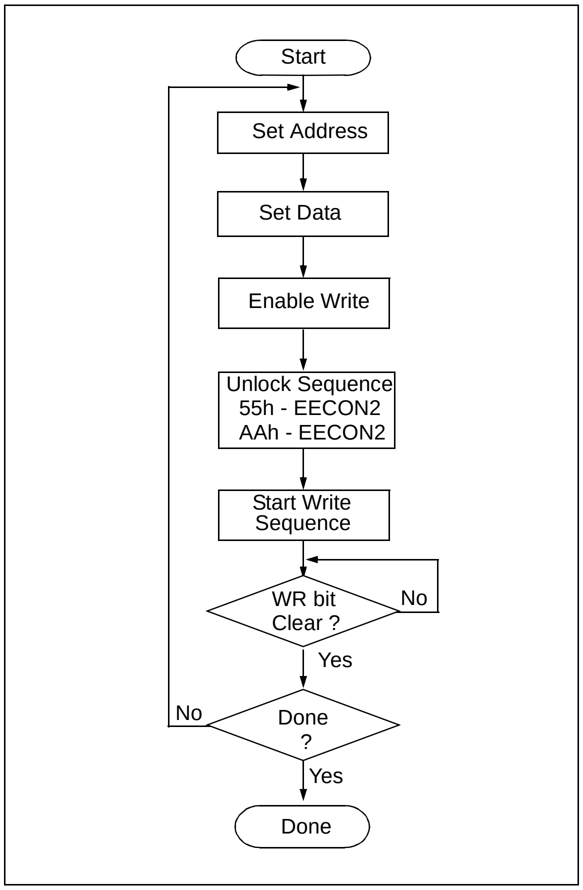


FIGURE 3-9: DATA EEPROM WRITE TIMING

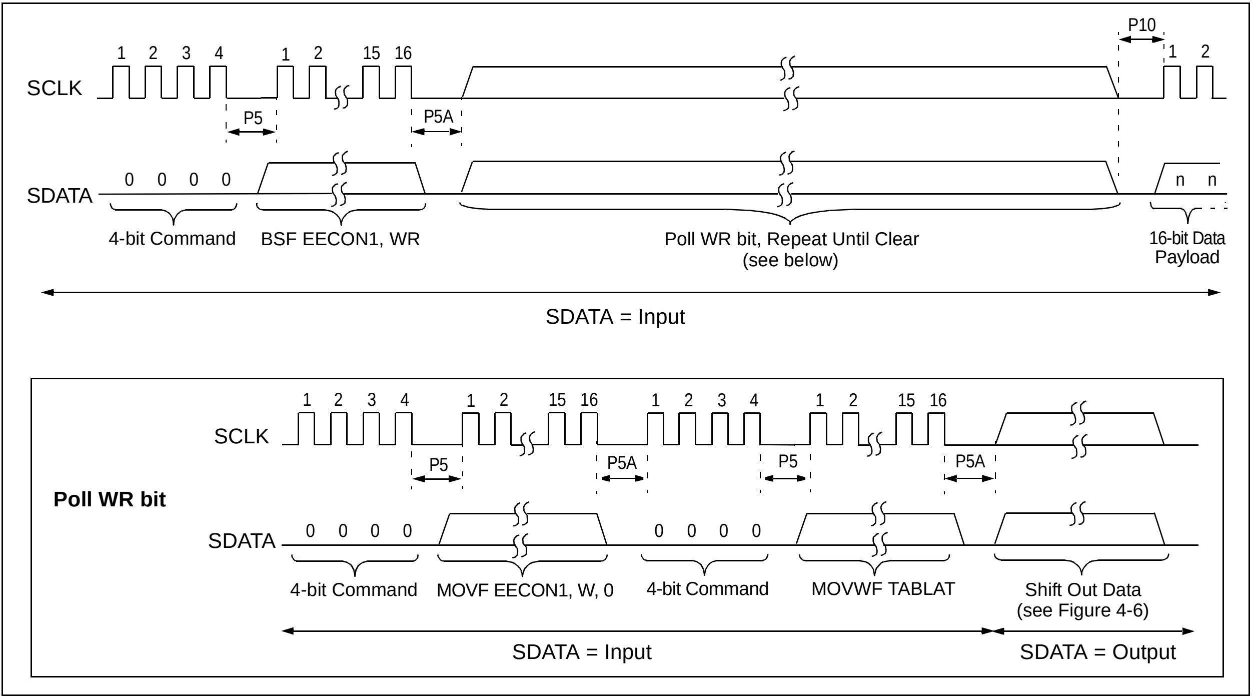


TABLE 3-5: PROGRAMMING DATA MEMORY

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: Perform required sequence.		

`0000`
`0000`
`0000`
`0000`
	`0E 55`
	`6E A7`
	`0E AA`
	`6E A7`
		`MOVLW 0X55`
		`MOVWF EECON2`
		`MOVLW 0XAA`
		`MOVWF EECON2`

Step 6: Initiate write.		
`0000`	`82 A6`	`BSF EECON1, WR`
Step 7: Poll WR bit, repeat until the bit is clear.		

`0000`
`0000`
`0010`
	`50 A6`
	`6E F5`
	`<LSB><MSB>`
		`MOVF EECON1, W, 0`
		`MOVWF TABLAT`
		`Shift out data`(1)

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, except that multi-panel writes must be disabled. The single panel that will be written will automatically be enabled, based on the value of the Table Pointer. The ID registers are mapped in addresses 200000h through 200007h. These locations read out normally, even after code protection.

Note:	Even though multi-panel writes are disabled, the user must still fill the 8-byte data buffer for the panel.

Figure 3-6 demonstrates the code sequence required to write the ID locations.

TABLE 3-6: WRITE ID SEQUENCE

4-Bit Command	Data Payload	Core Instruction
Step 1: Direct access to config memory.		

0000
0000
	8E A6
	8C A6
		`BSF EECON1, EEPGD`
		`BSF EECON1, CFGS`

Step 2: Configure device for single panel writes.		

0000
0000
0000
0000
0000
0000
1100
	0E 3C
	6E F8
	0E 00
	6E F7
	0E 06
	6E F6
	00 00
		`MOVLW 3Ch`
		`MOVWF TBLPTRU`
		`MOVLW 00h`
		`MOVWF TBLPTRH`
		`MOVLW 06h`
		`MOVWF TBLPTRL`
		`Write 00h to 3C0006h to enable single panel writes.`

Step 3: Direct access to code memory.		

0000
0000
	8E A6
	9C A6
		`BSF EECON1, EEPGD`
		`BCF EECON1, CFGS`

Step 4: Load write buffer. Panel will be automatically determined by address.		

0000
0000
0000
0000
0000
0000
1101
1101
1101
1111
0000
	0E 20
	6E F8
	0E 00
	6E F7
	0E 00
	6E F6
	<LSB><MSB>
	<LSB><MSB>
	<LSB><MSB>
	<LSB><MSB>
	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 SCLK high for time P9`

In order to modify the ID locations, refer to the methodology described in Section 3.2.2, “Modifying Code Memory”. As with code memory, the ID locations must be erased before modified.

3.5 Boot Block Programming

The Boot Block segment is programmed in exactly the same manner as the ID locations (see Section 3.4). Multi-panel writes must be disabled so that only addresses in the range 0000h to 01FFh will be written.

The code sequence detailed in Figure 3-6 should be used, except that the address data used in “Step 2” will be in the range 000000h to 0001FFh.

3.6 Configuration Bits Programming

Unlike code memory, the configuration bits are programmed a byte at a time. The “Table Write, Begin Programming” 4-bit command (`1111`) is used, but only 8 bits of the following 16-bit payload will be written. The LSB of the payload will be written to even addresses, and the MSB will be written to odd addresses. The code sequence to program two consecutive configuration locations is shown in Figure 3-7.

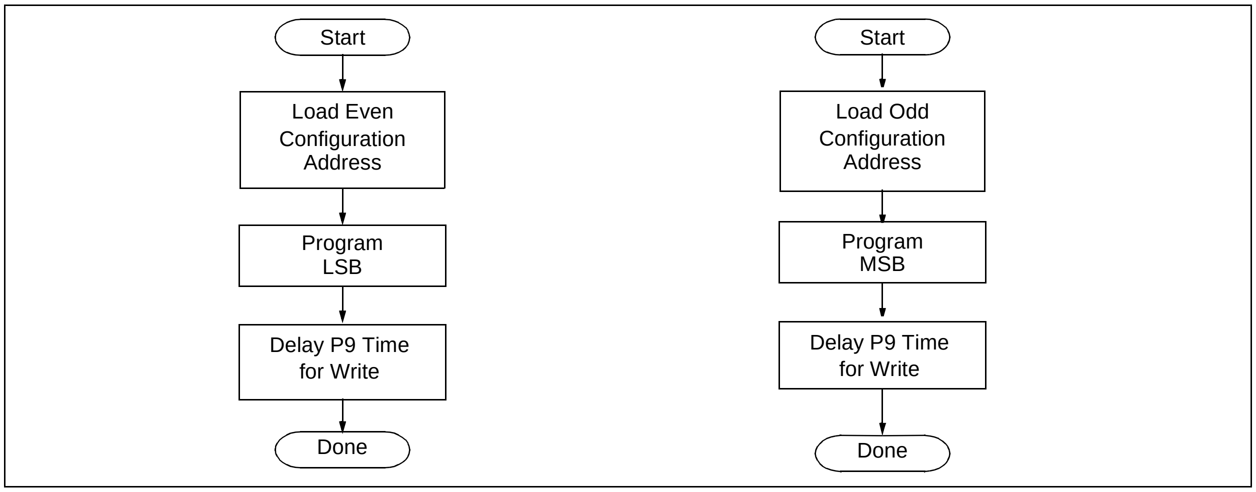
TABLE 3-7: SET ADDRESS POINTER TO CONFIGURATION LOCATION

4-Bit Command	Data Payload	Core Instruction
Step 1: Direct access to config memory.		
0000 0000	8E A6 8C A6	BSF EECON1, EEPGD BSF EECON1, CFGS
Step 2: Position the program counter(1).		
0000 0000	EF 00 F8 00	GOTO 100000h
Step 3(2): Set Table Pointer for config byte to be written. Write even/odd addresses.		
0000 0000 0000 0000 0000 0000 1111 0000 0000 1111 0000	0E 30 6E F8 0E 00 6E F7 0E 00 6E F6 <LSB><MSB ignored> 00 00 2A F6 <LSB ignored><MSB> 00 00	MOVLW 30h MOVWF TBLPTRU MOVLW 00h MOVWF TBLPRTH MOVLW 00h MOVWF TBLPTRL Load 2 bytes and start programming NOP - hold SCLK high for time P9 INCF TBLPTRL Load 2 bytes and start programming NOP - hold SCLK high for time P9

Note 1: If the code protection bits are programmed while the program counter resides in the same block, then the interaction of code protection logic may prevent further table write. To avoid this situation, move the program counter outside the code protection area (e.g., `GOTO 100000h`).

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

FIGURE 3-10: CONFIGURATION PROGRAMMING FLOW

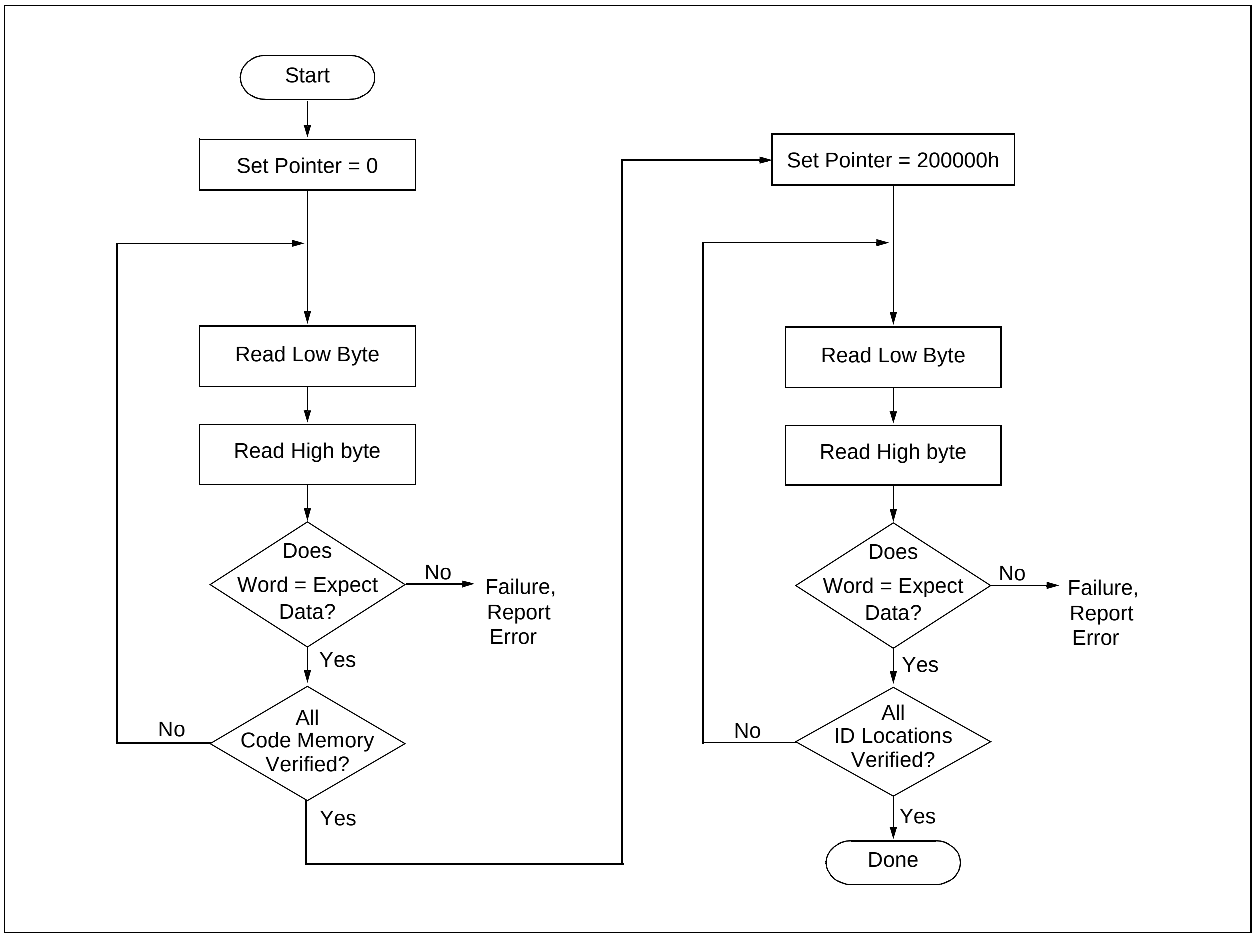


4.2 Verify Code Memory and ID locations

The verify step involves reading back the code memory space and comparing 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 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 32-Kbyte device, for example, a post-increment read of address 7FFFh will wrap the Table Pointer back to 0000h, rather than point to unimplemented address 8000h.

FIGURE 4-2: VERIFY CODE MEMORY FLOW



4.3 Verify Configuration Bits

A configuration address may be read and output on SDATA 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 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 EEADR:EEADRH) and a data latch (EEDATA). Data EEPROM is read by loading EEADR:EEADRH 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 SDATA 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 SCLK of the operand to allow SDATA to transition from an input to an output. During this time, SCLK must be held low (see Figure 4-4).

The command sequence to read a single byte of data is shown in Figure 4-2.

FIGURE 4-3: READ DATA EEPROM FLOW

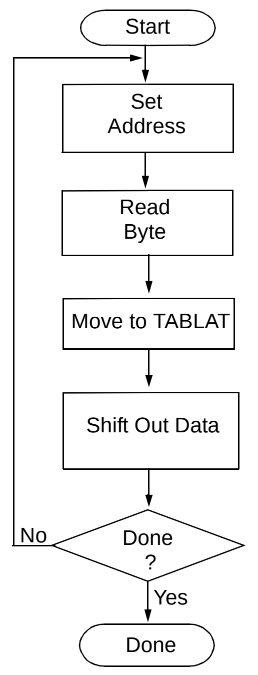


TABLE 4-2: READ DATA EEPROM MEMORY

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.		

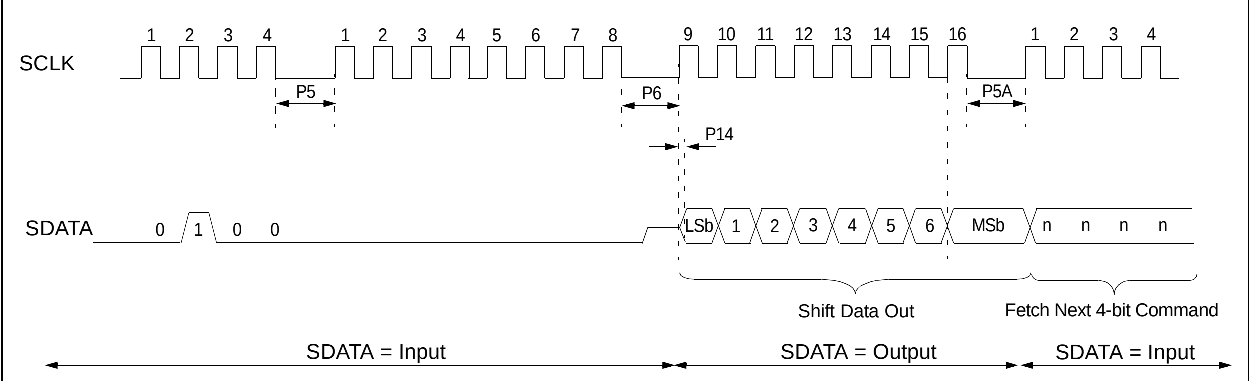
`0000`
`0000`
`0000`
`0000`
	`0E <Addr>`
	`6E A9`
	`OE <AddrH>`
	`6E AA`
		`MOVLW <Addr>`
		`MOVWF EEADR`
		`MOVLW <AddrH>`
		`MOVWF EEADRH`

Step 3: Initiate a memory read.		
`0000`	`80 A6`	`BSF EECON1, RD`
Step 4: Load data into the Serial Data Holding register.		

`0000`
`0000`
`0010`
	`50 A8`
	`6E F5`
	`<LSB><MSB>`
		`MOVF EEDATA, W, 0`
		`MOVWF TABLAT`
		`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`)



5.0 CONFIGURATION WORD

The PIC18FXX20 devices have several configuration words. These bits can be set or cleared to select various device configurations. All other memory areas should be programmed and verified prior to setting configuration words. These bits may be read out normally, even after read or code protection.

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 0Fh. In doing so, if the user code inadvertently tries to execute from the ID space, the ID data will execute as NOP.

5.2 Device ID Word

The device ID word for the PIC18FXX20 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.

5.3 Low Voltage Programming (LVP) Bit

The LVP bit in Configuration register, CONFIG4L, enables low voltage ICSP programming. The LVP bit defaults to a ‘1’ from the factory.

If Low Voltage 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 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 normal ICSP mode is always available, regardless of the state of the LVP bit, by applying VIHH to the $\overline{\text{MCLR}}$/VPP pin.

2: While in low voltage ICSP mode, the RB5 pin can no longer be used as a general purpose I/O.

TABLE 5-1: DEVICE ID VALUES

Device	Device ID Value	
	DEVID2	DEVID1
PIC18F6520	0Bh	001x xxxx
PIC18F6620	06h	011x xxxx
PIC18F6720	06h	001x xxxx
PIC18F8520	0Bh	000x xxxx
PIC18F8620	06h	010x xxxx
PIC18F8720	06h	000x xxxx

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

TABLE 5-3: PIC18FXX20 CONFIGURATION BIT DESCRIPTIONS (CONTINUED)

Bit Name	Configuration Words	Description
EBTR0	CONFIG7L	Table Read Protection bit (Block 0) 1 = Code memory not protected from table reads executed in other blocks 0 = Code memory protected from table reads executed in other blocks
EBTR1	CONFIG7L	Table Read Protection bit (Block 1) 1 = Code memory not protected from Table Reads executed in other blocks 0 = Code memory protected from Table Reads executed in other blocks
EBTR2	CONFIG7L	Table Read Protection bit (Block 2) 1 = Code memory not protected from Table Reads executed in other blocks 0 = Code memory protected from Table Reads executed in other blocks
EBTR3	CONFIG7L	Table Read Protection bit (Block 3) 1 = Code memory not protected from Table Reads executed in other blocks 0 = Code memory protected from Table Reads executed in other blocks
EBTR4(2)	CONFIG7L	Table Read Protection bit (Block 4) 1 = Code memory not protected from Table Reads executed in other blocks 0 = Code memory protected from Table Reads executed in other blocks
EBTR5(2)	CONFIG7L	Table Read Protection bit (Block 5) 1 = Code memory not protected from Table Reads executed in other blocks 0 = Code memory protected from Table Reads executed in other blocks
EBTR6(2)	CONFIG7L	Table Read Protection bit (Block 6) 1 = Code memory not protected from Table Reads executed in other blocks 0 = Code memory protected from Table Reads executed in other blocks
EBTR7(2)	CONFIG7L	Table Read Protection bit (Block 7) 1 = Code memory not protected from Table Reads executed in other blocks 0 = Code memory protected from Table Reads executed in other blocks
EBTRB	CONFIG7H	Table Read Protection bit (Boot Block) 1 = Boot block not protected from Table Reads executed in other blocks 0 = Boot block protected from Table Reads executed in other blocks
DEV10:DEV3	DEVID2	Device ID bits These bits are used with the DEV2:DEV0 bits in the DEVID1 register to identify part number.
DEV2:DEV0	DEVID1	Device ID bits These bits are used with the DEV10:DEV3 bits in the DEVID2 register to identify part number.
REV4:REV0	DEVID1	These bits are used to indicate the revision of the device.

Note 1: Unimplemented in PIC18F6X20 (64-pin) devices; maintain this bit set.

2: Unimplemented in PIC18FX620 devices; maintain this bit set.

3: PIC18F8520/8620 devices only.

TABLE 5-4: CHECKSUM COMPUTATION

Device	Code Protect	Checksum	Blank Value	0xAA at 0 and Max Address
PIC18F6520	None	SUM(0000:07FF)+SUM(0800:1FFF)+SUM(2000:3FFF)+ SUM(4000:5FFF)+SUM(6000:7FFF)+(CFGW1L & 0000)+ (CFGW1H & 0027)+(CFGW2L & 000F)+(CFGW2H & 000F)+ (CFGW3L & 0000)+(CFGW3H & 0002)+(CFGW4L & 0085)+ (CFGW4H & 0000)+(CFGW5L & 00FF)+(CFGW5H & 00C0)+ (CFGW6L & 00FF)+(CFGW6H & 00E0)+(CFGW7L & 00FF)+ (CFGW7H & 0040)	05A8	04FE
	Boot Block	SUM(0800:1FFF)+SUM(2000:3FFF)+SUM(4000:5FFF)+ SUM(6000:7FFF)+(CFGW1L & 0000)+(CFGW1H & 0027)+ (CFGW2L & 000F)+(CFGW2H & 000F)+(CFGW3L & 0000)+ (CFGW3H & 0002)+(CFGW4L & 0085)+(CFGW4H & 0000)+ (CFGW5L & 00FF)+(CFGW5H & 00C0)+(CFGW6L & 00FF)+ (CFGW6H & 00E0)+(CFGW7L & 00FF)+(CFGW7H & 0040)+ SUM(IDs)	077F	734
	Boot/ Block1/ Block2	SUM(4000:5FFF)+SUM(6000:7FFF)+(CFGW1L & 0000)+ (CFGW1H & 0027)+(CFGW2L & 000F)+(CFGW2H & 000F)+ (CFGW3L & 0000)+(CFGW3H & 0002)+(CFGW4L & 0085)+ (CFGW4H & 0000)+(CFGW5L & 00FF)+(CFGW5H & 00C0)+ (CFGW6L & 00FF)+(CFGW6H & 00E0)+(CFGW7L & 00FF)+ (CFGW7H & 0040)+SUM(IDs)	857C	8531
	All	(CFGW1L & 0000)+(CFGW1H & 0027)+(CFGW2L & 000F)+ (CFGW2H & 000F)+(CFGW3L & 0000)+(CFGW3H & 0002)+ (CFGW4L & 0085)+(CFGW4H & 0000)+(CFGW5L & 00FF)+ (CFGW5H & 00C0)+(CFGW6L & 00FF)+(CFGW6H & 00E0)+ (CFGW7L & 00FF)+(CFGW7H & 0040)+SUM(IDs)	480	048A

Legend: Item		Description
CFGW	=	Configuration Word
SUM[a:b]	=	Sum of locations, a to b inclusive
SUM_ID	=	Byte-wise sum of lower four bits of all customer ID locations
+	=	Addition
&	=	Bit-wise AND

TABLE 5-4: CHECKSUM COMPUTATION (CONTINUED)

Device	Code Protect	Checksum	Blank Value	0xAA at 0 and Max Address
PIC18F6720	None	SUM(0000:01FF)+SUM(0200:3FFF)+SUM(4000:7FFF)+ SUM(8000:BFFF)+SUM(C000:FFFF)+SUM(10000:13FFF)+ SUM(14000:17FFF)+SUM(18000:1BFFF)+SUM(1C000:1FFFF)+ (CFGW1L & 0000)+(CFGW1H & 0027)+(CFGW2L & 000F)+ (CFGW2H & 000F)+(CFGW3L & 0000)+(CFGW3H & 0001)+ (CFGW4L & 0085)+(CFGW4H & 0000)+(CFGW5L & 00FF)+ (CFGW5H & 00C0)+(CFGW6L & 00FF)+(CFGW6H & 00E0)+ (CFGW7L & 00FF)+(CFGW7H & 0040)	05A8	04FE
	Boot Block	SUM(0200:3FFF)+SUM(4000:7FFF)+SUM(8000:BFFF)+ SUM(C000:FFFF)+SUM(10000:13FFF)+SUM(14000:17FFF)+ SUM(18000:1BFFF)+SUM(1C000:1FFFF)+(CFGW1L & 0000)+ (CFGW1H & 0027)+(CFGW2L & 000F)+(CFGW2H & 000F)+ (CFGW3L & 0000)+(CFGW3H & 0001)+(CFGW4L & 0085)+ (CFGW4H & 0000)+(CFGW5L & 00FF)+(CFGW5H & 00C0)+ (CFGW6L & 00FF)+(CFGW6H & 00E0)+(CFGW7L & 00FF)+ (CFGW7H & 0040)+SUM(IDs)	077F	0734
	Boot/ Block1/ Block2	SUM(8000:BFFF)+SUM(C000:FFFF)+SUM(10000:13FFF)+ SUM(14000:17FFF)+SUM(18000:1BFFF)+SUM(1C000:1FFFF)+ (CFGW1L & 0000)+(CFGW1H & 0027)+(CFGW2L & 000F)+ (CFGW2H & 000F)+(CFGW3L & 0000)+(CFGW3H & 0001)+ (CFGW4L & 0085)+(CFGW4H & 0000)+(CFGW5L & 00FF)+ (CFGW5H & 00C0)+(CFGW6L & 00FF)+(CFGW6H & 00E0)+ (CFGW7L & 00FF)+(CFGW7H & 0040)+SUM(IDs)	857C	8531
	All	(CFGW1L & 0000)+(CFGW1H & 0027)+(CFGW2L & 000F)+ (CFGW2H & 000F)+(CFGW3L & 0000)+(CFGW3H & 0001)+ (CFGW4L & 0085)+(CFGW4H & 0000)+(CFGW5L & 00FF)+ (CFGW5H & 00C0)+(CFGW6L & 00FF)+(CFGW6H & 00E0)+ (CFGW7L & 00FF)+(CFGW7H & 0040)+SUM(IDs)	480	048A

Legend:

Item		Description
CFGW	=	Configuration Word
SUM[a:b]	=	Sum of locations, a to b inclusive
SUM_ID	=	Byte-wise sum of lower four bits of all customer ID locations
+	=	Addition
&	=	Bit-wise AND

TABLE 5-4: CHECKSUM COMPUTATION (CONTINUED)

Device	Code Protect	Checksum	Blank Value	0xAA at 0 and Max Address
PIC18F8520	None	SUM(0000:07FF)+SUM(0800:1FFF)+SUM(2000:3FFF)+ SUM(4000:5FFF)+SUM(6000:7FFF)+(CFGW1L & 0000)+ (CFGW1H & 0027)+(CFGW2L & 000F)+(CFGW2H & 000F)+ (CFGW3L & 0000)+(CFGW3H & 0002)+(CFGW4L & 0085)+ (CFGW4H & 0000)+(CFGW5L & 00FF)+(CFGW5H & 00C0)+ (CFGW6L & 00FF)+(CFGW6H & 00E0)+(CFGW7L & 00FF)+ (CFGW7H & 0040)	05AA	500
	Boot Block	SUM(0800:1FFF)+SUM(2000:3FFF)+SUM(4000:5FFF)+ SUM(6000:7FFF)+(CFGW1L & 0000)+(CFGW1H & 0027)+ (CFGW2L & 000F)+(CFGW2H & 000F)+(CFGW3L & 0000)+ (CFGW3H & 0002)+(CFGW4L & 0085)+(CFGW4H & 0000)+ (CFGW5L & 00FF)+(CFGW5H & 00C0)+(CFGW6L & 00FF)+ (CFGW6H & 00E0)+(CFGW7L & 00FF)+(CFGW7H & 0040)+ SUM(IDs)	783	071A
	Boot/ Block1/ Block2	SUM(4000:5FFF)+SUM(6000:7FFF)+(CFGW1L & 0000)+ (CFGW1H & 0027)+(CFGW2L & 000F)+(CFGW2H & 000F)+ (CFGW3L & 0000)+(CFGW3H & 0002)+(CFGW4L & 0085)+ (CFGW4H & 0000)+(CFGW5L & 00FF)+(CFGW5H & 00C0)+ (CFGW6L & 00FF)+(CFGW6H & 00E0)+(CFGW7L & 00FF)+ (CFGW7H & 0040)+SUM(IDs)	8580	8517
	All	(CFGW1L & 0000)+(CFGW1H & 0027)+(CFGW2L & 000F)+ (CFGW2H & 000F)+(CFGW3L & 0000)+(CFGW3H & 0002)+ (CFGW4L & 0085)+(CFGW4H & 0000)+(CFGW5L & 00FF)+ (CFGW5H & 00C0)+(CFGW6L & 00FF)+(CFGW6H & 00E0)+ (CFGW7L & 00FF)+(CFGW7H & 0040)+SUM(IDs)	484	470

Legend: Item		Description
CFGW	=	Configuration Word
SUM[a:b]	=	Sum of locations, a to b inclusive
SUM_ID	=	Byte-wise sum of lower four bits of all customer ID locations
+	=	Addition
&	=	Bit-wise AND

TABLE 5-4: CHECKSUM COMPUTATION (CONTINUED)

Device	Code Protect	Checksum	Blank Value	0xAA at 0 and Max Address
PIC18F8620	None	SUM(0000:01FF)+SUM(0200:3FFF)+SUM(4000:7FFF)+ SUM(8000:BFFF)+SUM(C000:FFFF)+(CFGW1L & 0000)+ (CFGW1H & 0027)+(CFGW2L & 000F)+(CFGW2H & 000F)+ (CFGW3L & 0083)+(CFGW3H & 0001)+(CFGW4L & 0085)+ (CFGW4H & 0000)+(CFGW5L & 000F)+(CFGW5H & 00C0)+ (CFGW6L & 000F)+(CFGW6H & 00E0)+(CFGW7L & 000F)+ (CFGW7H & 0040)	035B	02B1
	Boot Block	SUM(0200:3FFF)+SUM(4000:7FFF)+SUM(8000:BFFF)+ SUM(C000:FFFF)+(CFGW1L & 0000)+(CFGW1H & 0027)+ (CFGW2L & 000F)+(CFGW2H & 000F)+(CFGW3L & 0083)+ (CFGW3H & 0001)+(CFGW4L & 0085)+(CFGW4H & 0000)+ (CFGW5L & 000F)+(CFGW5H & 00C0)+(CFGW6L & 000F)+ (CFGW6H & 00E0)+(CFGW7L & 000F)+(CFGW7H & 0040)+ SUM(IDs)	052E	04D4
	Boot/ Block1/ Block2	SUM(8000:BFFF)+SUM(C000:FFFF)+(CFGW1L & 0000)+ (CFGW1H & 0027)+(CFGW2L & 000F)+(CFGW2H & 000F)+ (CFGW3L & 0083)+(CFGW3H & 0001)+(CFGW4L & 0085)+ (CFGW4H & 0000)+(CFGW5L & 000F)+(CFGW5H & 00C0)+ (CFGW6L & 000F)+(CFGW6H & 00E0)+(CFGW7L & 000F)+ (CFGW7H & 0040)+SUM(IDs)	832B	82D1
	All	(CFGW1L & 0000)+(CFGW1H & 0027)+(CFGW2L & 000F)+ (CFGW2H & 000F)+(CFGW3L & 0083)+(CFGW3H & 0001)+ (CFGW4L & 0085)+(CFGW4H & 0000)+(CFGW5L & 000F)+ (CFGW5H & 00C0)+(CFGW6L & 000F)+(CFGW6H & 00E0)+ (CFGW7L & 000F)+(CFGW7H & 0040)+SUM(IDs)	031F	031A

Legend:	Item		Description
	CFGW	=	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