dsPIC33E/PIC24E Flash Programming Specification for Devices with Volatile Configuration Bits

1.0 DEVICE OVERVIEW

This document defines the programming specification for the dsPIC33E/PIC24E 16-bit Digital Signal Controller (DSC) and 16-bit Microcontroller families with volatile Configuration bits. This programming specification is required only for those developing programming support for the following devices:

- dsPIC33EPXXXGP50X
- dsPIC33EPXXXMC50X
- dsPIC33EPXXXMC20X
- PIC24EPXXXMC20X
- PIC24EPXXXGP20X

Customers using only one of these devices should use development tools that already provide support for device programming.

Topics covered include:

2.0 PROGRAMMING OVERVIEW

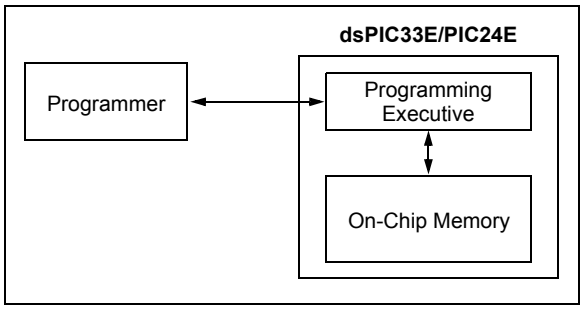
There are two methods of programming that are discussed in this programming specification:

- In-Circuit Serial Programming™ (ICSP™) programming capability
- Enhanced In-Circuit Serial Programming

The ICSP programming method is the most direct method to program the device; however, it is also the slower of the two methods. It provides native, low-level programming capability to erase, program and verify the device.

The Enhanced ICSP protocol uses a faster method that takes advantage of the programming executive, as illustrated in Figure 2-1. The programming executive provides all the necessary functionality to erase, program and verify the chip through a small command set. The command set allows the programmer to program a dsPIC33E/PIC24E device without dealing with the low-level programming protocols.

FIGURE 2-1: PROGRAMMING SYSTEM OVERVIEW FOR ENHANCED ICSP™



This programming specification is divided into two major sections that describe the programming methods independently. **Section 3.0 “Device Programming – ICSP”** describes the ICSP method. **Section 4.0 “Device Programming – Enhanced ICSP”** describes the Enhanced ICSP method.

2.1 Required Connections

These devices require specific connections for programming to take place. These connections include power, VCAP, MCLR, and one programming pair (PGEDx/PGECx). Table 2-1 describes these connections (refer to the specific device data sheet for pin descriptions and power connection requirements).

Note:	Refer to the specific device data sheet for complete pin diagrams of dsPIC33E/PIC24E devices.

TABLE 2-1: PINS USED DURING PROGRAMMING

During Programming		
Pin Name	**Pin Type**	**Pin Description**
MCLR	I	Programming Enable
VDD and AVDD(1)	P	Power Supply(1)
VSS and AVSS(1)	P	Ground(1)
VCAP	P	CPU Logic Filter Capacitor Connection
PGECx	I	Programming Pin Pair: Serial Clock
PGEDx	I/O	Programming Pin Pair: Serial Data

Legend: I = Input O = Output P = Power

Note 1: All power supply and ground pins must be connected including AVDD and AVSS.

2.2 Program Memory Write/Erase Requirements

The program Flash memory has a specific write/erase requirement that must be adhered to, for proper device operation. The rule is that any given word in memory must not be written without first erasing the page in which it is located. Thus, the easiest way to conform to this rule is to write all the data in a programming block within one write cycle. The programming methods specified in this document comply with this requirement.

Note:	A program memory word can be programmed twice before an erase, but only if (a) the same data is used in both program operations or (b) bits containing '1' are set to '0' but no '0' is set to '1'.

2.3 Memory Map

The program memory map extends from 0x0 to 0xFFFFFE. Code storage is located at the base of the memory map. The last locations of implemented program memory are reserved for the device Configuration bits.

Table 2-2 lists the program memory size, the size of the erase blocks, and the number of erase blocks present in each device variant.

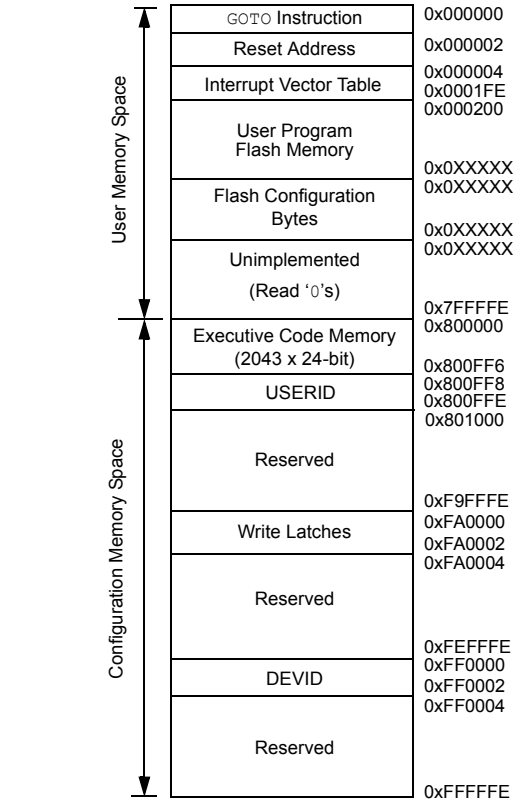
Locations 0x800000 through 0x800FFE are reserved for executive code memory. This region stores the Programming Executive (PE) and the debugging executive, which is used for device programming. This region of memory cannot be used to store user code. See **Section 6.0 "The Programming Executive"** for more information.

Locations 0xFF0000 and 0xFF0002 are reserved for the Device ID Word registers. These bits can be used by the programmer to identify which device type is being programmed. They are described in **Section 7.0 "Device ID"**. The Device ID registers read out normally, even after code protection is applied.

Locations 0x800FF8 and 0x800FFE are reserved for the User ID Word register. The User ID Words can be used for storing product information such as serial numbers, system manufacturing dates, manufacturing lot numbers and other application-specific information. They are described in **Section 2.5 "User ID Words"**.

Figure 2-2 shows a generic memory map for the devices listed in Table 2-2. See the **"Memory Organization"** chapter in the specific device data sheet for exact memory addresses.

FIGURE 2-2: PROGRAM MEMORY MAP



Note 1: Memory areas are not shown to scale.

2: Memory map is for reference only. Refer to the **"Memory Organization"** chapter in the specific device data sheet for exact memory addresses.

TABLE 2-2: CODE MEMORY SIZE

Device	User Memory Address Limit (Instruction Words)	Erase Page Size (Instruction Words)	Erase Blocks (Pages)	Executive Memory address Limit (Instruction Words)	Configuration Bit Address Range (Bytes)
dsPIC33EP32GP50X	0x0057EA (11.2K)	512	21	0x800FF6 (2K)	0x0057EC-0x0057FE (30)
dsPIC33EP32MC50X	0x0057EA (11.2K)	512	21	0x800FF6 (2K)	0x0057EC-0x0057FE (30)
dsPIC33EP32MC20X	0x0057EA (11.2K)	512	21	0x800FF6 (2K)	0x0057EC-0x0057FE (30)
PIC24EP32MC20X	0x0057EA (11.2K)	512	21	0x800FF6 (2K)	0x0057EC-0x0057FE (30)
PIC24EP32GP20X	0x0057EA (11.2K)	512	21	0x800FF6 (2K)	0x0057EC-0x0057FE (30)
dsPIC33EP64GP50X	0x00AFEA (22.5K)	1024	21	0x800FF6 (2K)	0x00AFEC-0x00AFFE (30)
dsPIC33EP64MC50X	0x00AFEA (22.5K)	1024	21	0x800FF6 (2K)	0x00AFEC-0x00AFFE (30)
dsPIC33EP64MC20X	0x00AFEA (22.5K)	1024	21	0x800FF6 (2K)	0x00AFEC-0x00AFFE (30)
PIC24EP64MC20X	0x00AFEA (22.5K)	1024	21	0x800FF6 (2K)	0x00AFEC-0x00AFFE (30)
PIC24EP64GP20X	0x00AFEA (22.5K)	1024	21	0x800FF6 (2K)	0x00AFEC-0x00AFFE (30)
dsPIC33EP128GP50X	0x0157EA (44K)	1024	42	0x800FF6 (2K)	0x0157EC-0x0157FE (30)
dsPIC33EP128MC50X	0x0157EA (44K)	1024	42	0x800FF6 (2K)	0x0157EC-0x0157FE (30)
dsPIC33EP128MC20X	0x0157EA (44K)	1024	42	0x800FF6 (2K)	0x0157EC-0x0157FE (30)
PIC24EP128MC20X	0x0157EA (44K)	1024	42	0x800FF6 (2K)	0x0157EC-0x0157FE (30)
PIC24EP128GP20X	0x0157EA (44K)	1024	42	0x800FF6 (2K)	0x0157EC-0x0157FE (30)
dsPIC33EP256GP50X	0x02AFEA (88K)	1024	85	0x800FF6 (2K)	0x02AFEC-0x02AFFE (30)
dsPIC33EP256MC50X	0x02AFEA (88K)	1024	85	0x800FF6 (2K)	0x02AFEC-0x02AFFE (30)
dsPIC33EP256MC20X	0x02AFEA (88K)	1024	85	0x800FF6 (2K)	0x02AFEC-0x02AFFE (30)
PIC24EP256MC20X	0x02AFEA (88K)	1024	85	0x800FF6 (2K)	0x02AFEC-0x02AFFE (30)
PIC24EP256GP20X	0x02AFEA (88K)	1024	85	0x800FF6 (2K)	0x02AFEC-0x02AFFE (30)
dsPIC33EP512GP50X	0x0557EA (175K)	1024	170	0x800FF6 (2K)	0x0557EC-0x0557FE (30)
dsPIC33EP512MC50X	0x0557EA (175K)	1024	170	0x800FF6 (2K)	0x0557EC-0x0557FE (30)
dsPIC33EP512MC20X	0x0557EA (175K)	1024	170	0x800FF6 (2K)	0x0557EC-0x0557FE (30)
PIC24EP512MC20X	0x0557EA (175K)	1024	170	0x800FF6 (2K)	0x0557EC-0x0557FE (30)
PIC24EP512GP20X	0x0557EA (175K)	1024	170	0x800FF6 (2K)	0x0557EC-0x0557FE (30)

2.4 Configuration Bits

2.4.1 OVERVIEW

The Configuration bits are stored in the last locations of implemented program memory. These bits can be set or cleared to select various device configurations. There are two types of Configuration bits: system operation bits, and code-protect bits. The system operation bits determine the power-on settings for system level components, such as the Oscillator and the Watchdog Timer. The code-protect bits prevent program memory from being read and written.

Table 2-2 lists the configuration byte register address range for each device.

Table 2-3 is an example of a configuration byte register map for the dsPIC33EP64MC506 device. Refer to the **"Special Features"** chapter in the specific device data sheet for the full Configuration byte register description for your device.

TABLE 2-3: CONFIGURATION BYTE REGISTER MAP

Register Name	Address (see Note 4)	Bit 23-8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	00AFEC	—	—	—	—	—	—	—	—	—
Reserved	00AFEE	—	—	—	—	—	—	—	—	—
FICD	00AFF0	—	Reserved(3)	—	JTAGEN	Reserved(2)	Reserved(3)	—	ICS<1:0>	
FPOR	00AFF2	—	WDTWIN<1:0>		ALTI2C2	ALTI2C1	—	—	—	—
FWDT	00AFF4	—	FWDTEN	WINDIS	PLLKEN	WDTPRE	WDTPOST<3:0>			
FOSC	00AFF6	—	FCKSM<1:0>		IOL1WAY	—	—	OSCIOFNC	POSCMD<1:0>	
FOSCSEL	00AFF8	—	IESO	PWMLOCK(1)	—	—	—	FNOSC<2:0>		
FGS	00AFFA	—	—	—	—	—	—	—	GCP	GWRP
Reserved	00AFFC	—	—	—	—	—	—	—	—	—
Reserved	00AFFE	—	—	—	—	—	—	—	—	—

Legend: — = unimplemented, read as '1'.

Note 1: These bits are only available on dsPIC33EPXXXMC20X/50X and PIC24EPXXXMC20X devices.

2: This bit is reserved and must be programmed as '0'.

3: This bit is reserved and must be programmed as '1'.

4: Addresses shown are for 64K devices. Refer to the **"Special Features"** chapter in the specific device data sheet for the register addresses for your device.

2.4.2 CODE-PROTECT CONFIGURATION BITS

The Configuration bits control the code protection features, with two forms of code protection being provided. One form prevents code memory from being written (write protection) and the other prevents code memory from being read (i.e., read protection).

The GWRP bit (FGS<0>) controls write protection and the GCP bit (FGS<1>) controls read protection. Protection is enabled when the respective bit is '0'.

Erasing program memory sets GWRP and GCP to '1', which allows the device to be programmed.

When write protection is enabled (GWRP = 0), any programming operation to code memory will fail.

When read protection is enabled (GCP = 0), any read from code memory will cause a 0x0 to be read, regardless of the actual contents of code memory. Since the programming executive always verifies what it programs, attempting to program code memory with read protection enabled also will result in failure.

It is imperative that both GWRP and GCP are '1' while the device is being programmed and verified. Only after the device is programmed and verified should either GWRP or GCP be programmed to '0' (see **Section 2.4 "Configuration Bits"**).

Note 1: Bulk Erasing the program memory is the only way to reprogram code-protect bits from an ON state '0' to an OFF state '1'.

2: Performing a page erase operation on the last page of program memory clears the Flash Configuration words, enabling code protection as a result. Therefore, users should avoid performing page erase operations on the last page of program memory.

3: If the General Segment Code-Protect (GCP) bit FGS <1> is programmed to '0', code memory is code-protected and cannot be read. Code memory must be read and verified before enabling read protection. See **Section 2.4.2 "Code-Protect Configuration Bits"** for detailed information about code-protect Configuration bits, and **Section 3.12 "Verify Code Memory and Configuration Bits"** for details about reading and verifying code memory and Configuration Words and Bytes.

2.5 User ID Words

dsPIC33E/PIC24E devices contain four User ID Words, located at addresses 0x800FF8 through 0x800FFE. The User ID Words can be utilized by the user for storing checksum, code revisions, product information such as serial numbers, system manufacturing dates, manufacturing lot numbers and other application-specific information. These words can only be written at program time and not at run time. They can be read at run time.

The User ID Words are part of the last page of executive memory, as a result performing a page erase of the last page of executive memory will also erase the User ID Words. The Executive Memory Bulk Erase command (NVMCON = 0x400F) will also erase the User ID Words.The User ID Words register map is shown in Table 2-4.

TABLE 2-4: USER ID WORDS REGISTER MAP

Name	Address	Bit 23-16	Bit 15-0
FUID0	0x800FF8	—	UID0
FUID1	0x800FFA	—	UID1
FUID2	0x800FFC	—	UID2
FUID3	0x800FFE	—	UID3

Legend: — = unimplemented, read as '1'.

3.0 DEVICE PROGRAMMING – ICSP

ICSP mode is a special programming protocol that allows you to read and write to device memory. The ICSP mode is the most direct method used to program the device, which is accomplished by applying control codes and instructions serially to the device using the PGECx and PGEDx pins. ICSP mode also has the ability to read executive memory to determine if the programming executive is present and to write the programming executive to executive memory if Enhanced ICSP mode will be used.

In ICSP mode, the system clock is taken from the PGECx pin, regardless of the device's oscillator Configuration bits. All instructions are shifted serially into an internal buffer, then loaded into the instruction register and executed. No program fetching occurs from internal memory. Instructions are fed in 24 bits at a time. PGEDx is used to shift data in, and PGECx is used as both the serial shift clock and the CPU execution clock.

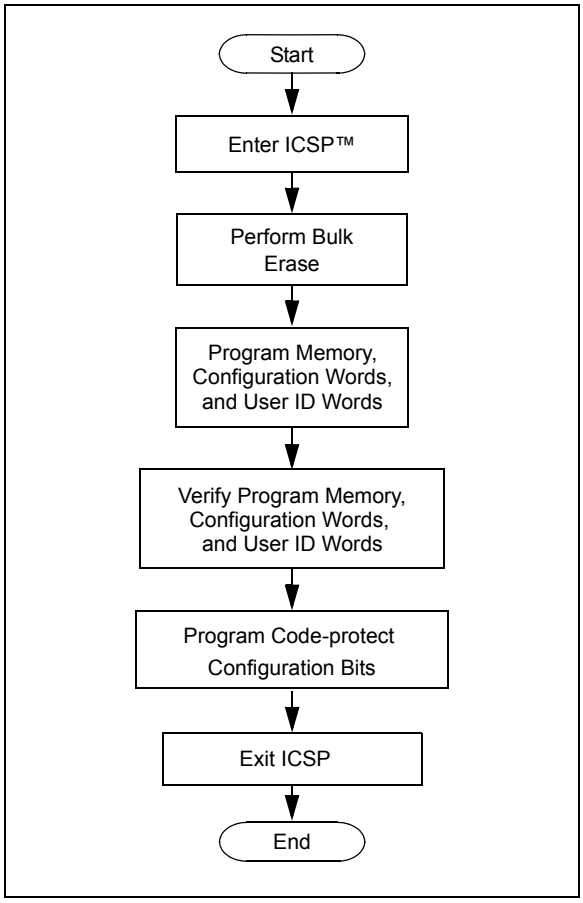
Note 1: During ICSP operation, the operating frequency of PGECx must not exceed 5 MHz.

2: ICSP mode is slower than Enhanced ICSP mode for programming.

3.1 Overview of the Programming Process

Figure 3-1 illustrates the high-level overview of the programming process. After entering ICSP mode, the first action is to Bulk Erase program memory. Next, the code memory is programmed, followed by the device Configuration bits. Code memory (including the Configuration bits) is then verified to ensure that programming was successful. Then, program the code-protect Configuration bits, if required.

FIGURE 3-1: HIGH-LEVEL ICSP™ PROGRAMMING FLOW



3.2 Entering ICSP Mode

As illustrated in Figure 3-2, entering ICSP Program/Verify mode requires three steps:

1. $\overline{MCLR}$ is briefly driven high, and then low (P21)(1).
2. A 32-bit key sequence is clocked into PGEDx.
3. $\overline{MCLR}$ is then driven high within a specified period of time 'P19' and held.

Note 1: If a capacitor is present on the $\overline{MCLR}$ pin, the high time for entering ICSP mode can vary.

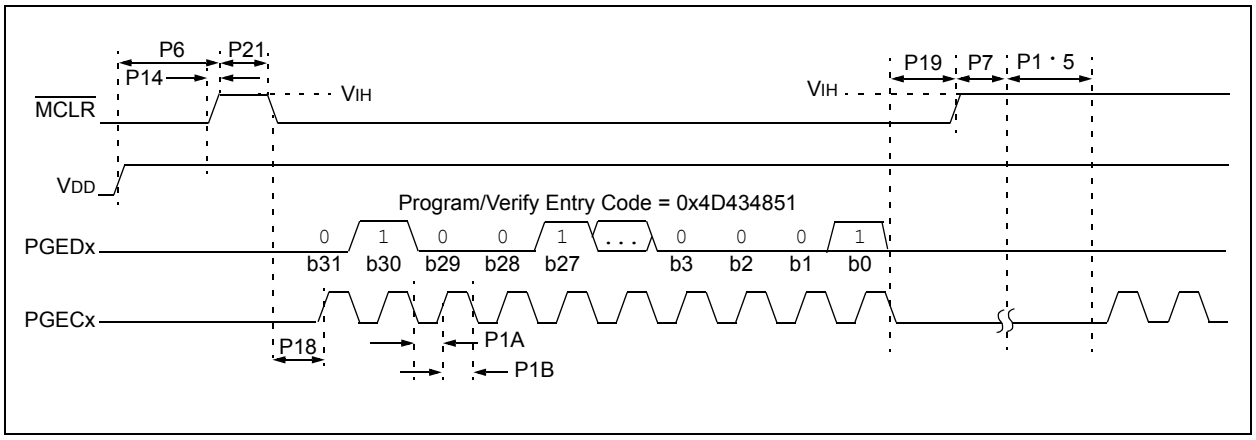
The programming voltage applied to $\overline{MCLR}$ is VIH, which is essentially VDD in the case of dsPIC33E/PIC24E devices. There is no minimum time requirement for holding at VIH. After VIH is removed, an interval of at least P18 must elapse before presenting the key sequence on PGEDx.

The key sequence is a specific 32-bit pattern, '0100 1101 0100 0011 0100 1000 0101 0001' (more easily remembered as 0x4D434851 in hexadecimal). The device will enter Program/Verify mode only if the sequence is valid. The Most Significant bit of the most significant nibble must be shifted in first.

Once the key sequence is complete, VIH must be applied to $\overline{MCLR}$ and held at that level for as long as Program/Verify mode is to be maintained. An interval of at least time P19, P7, and P1 * 5 must elapse before presenting data on PGEDx. Signals appearing on PGEDx before P7 has elapsed will not be interpreted as valid.

On successful entry, the program memory can be accessed and programmed in serial fashion. While in ICSP mode, all unused I/Os are placed in a high-impedance state.

FIGURE 3-2: ENTERING ICSP™ MODE



3.3 ICSP Operation

After entering into ICSP mode, the CPU is Idle. Execution of the CPU is governed by an internal state machine. A 4-bit control code is clocked in using PGECx and PGEDx and this control code is used to command the CPU (see Table 3-1).

The SIX control code is used to send instructions to the CPU for execution and the REGOUT control code is used to read data out of the device via the VISI register.

TABLE 3-1: CPU CONTROL CODES IN ICSP™ MODE

4-bit Control Code	Mnemonic	Description
'b0000	SIX	Shift in 24-bit instruction and execute.
'b0001	REGOUT	Shift out the VISI register.
'b0010-'b1111	N/A	Reserved.

3.3.1 SIX SERIAL INSTRUCTION EXECUTION

The SIX control code allows execution of assembly instructions. When the SIX code is received, the CPU is suspended for 24 clock cycles, as the instruction is then clocked into the internal buffer. Once the instruction is shifted in, the state machine allows it to be executed over the next four clock cycles. While the received instruction is executed, the state machine simultaneously shifts in the next 4-bit command (see Figure 3-3).

Note 1: Coming out of the ICSP entry sequence, the first 4-bit control code is always forced to SIX and a forced NOP instruction is executed by the CPU. Five additional PGECx clocks are needed on start-up, thereby resulting in a 9-bit SIX command instead of the normal 4-bit SIX command. After the forced SIX is clocked in, ICSP operation resumes as normal (the next 24 clock cycles load the first instruction word to the CPU). See Figure 3-4 for details.

2: TBLRDH, TBLRDL, TBLWTH and TBLWTL instructions must be followed by a NOP instruction.

FIGURE 3-3: SIX SERIAL EXECUTION

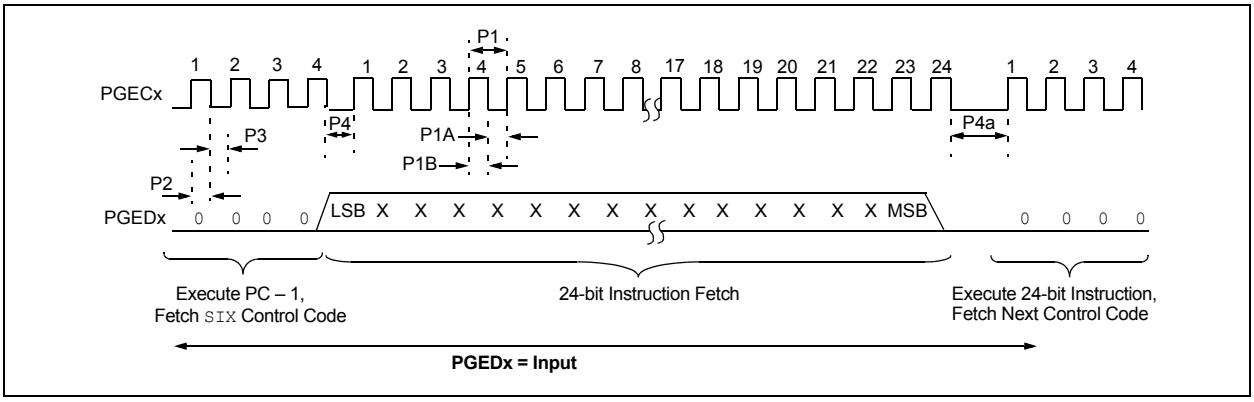
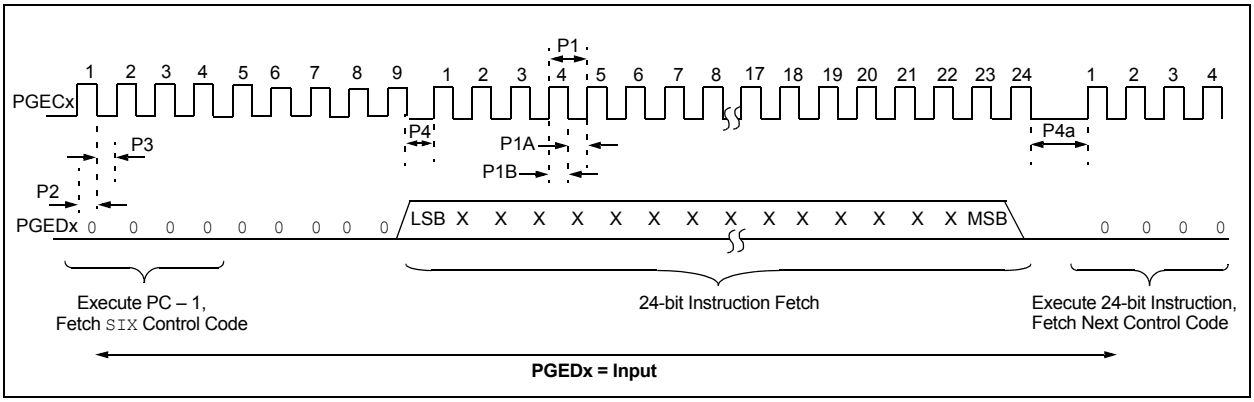


FIGURE 3-4: PROGRAM ENTRY AFTER RESET



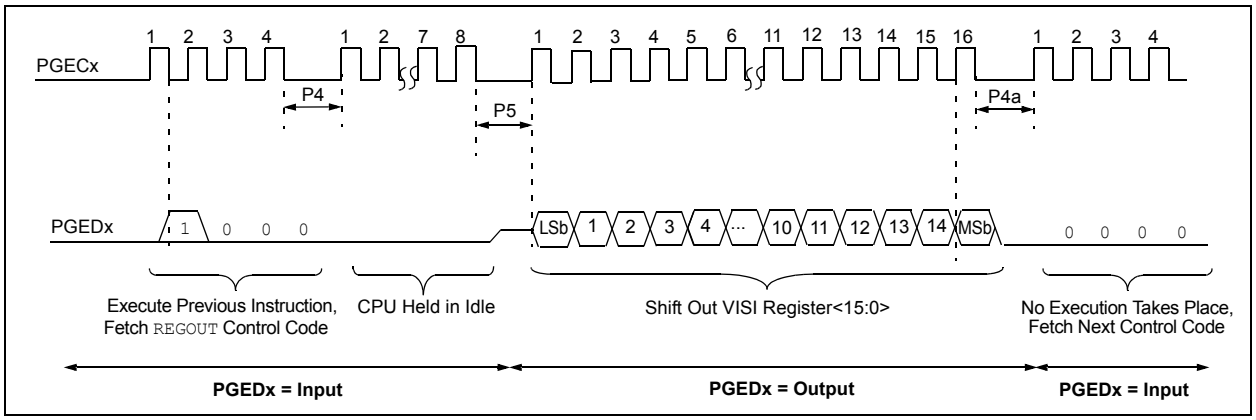
3.3.2 REGOUT SERIAL INSTRUCTION EXECUTION

The REGOUT control code allows for data to be extracted from the device in ICSP mode. It is used to clock the contents of the VISI register out of the device over the PGEDx pin. After the REGOUT control code is received, the CPU is held Idle for eight cycles. After these eight cycles, an additional 16 cycles are required to clock the data out (see Figure 3-5).

The REGOUT code is unique because the PGEDx pin is an input when the control code is transmitted to the device. However, after the control code is processed, the PGEDx pin becomes an output as the VISI register is shifted out.

Note:	The device will latch input PGEDx data on the rising edge of PGECx and will output data on the PGEDx line on the rising edge of PGECx. For all data transmissions, the Least Significant bit (LSb) is transmitted first.

FIGURE 3-5: REGOUT SERIAL EXECUTION



3.4 Flash Memory Programming in ICSP Mode

3.4.1 PROGRAMMING OPERATIONS

Flash memory write and erase operations are controlled by the NVMCON register. Programming is performed by setting NVMCON to select the type of erase operation (Table 3-2) or write operation (Table 3-3) and initiating the programming by setting the WR control bit (NVMCON<15>).

In ICSP mode, all programming operations are self-timed. There is an internal delay between the user setting the WR control bit and the automatic clearing of the WR control bit when the programming operation is complete. Refer to **Section 9.0 "AC/DC Characteristics and Timing Requirements"** for detailed information about the delays associated with various programming operations.

TABLE 3-2: NVMCON ERASE OPERATIONS

NVMCON Value	Erase Operation
0x400F	Bulk erase user memory, executive memory, and User ID Words (does not erase Device ID registers).
0x400D	Bulk erase user memory only.
0x4003	Erase a page of program or executive memory.

TABLE 3-3: NVMCON WRITE OPERATIONS

NVMCON Value	Write Operation
0x4001	Double-word program operation.

3.4.2 STARTING AND STOPPING A PROGRAMMING CYCLE

For protection against accidental operations, the erase/write initiate sequence must be written to the NVMKEY register to allow any erase or program operation to proceed. The two instructions following the start of the programming sequence should be `NOPs`. To start a erase or write sequence the following steps must be completed:

1. Write 0x55 to the NVMKEY register.
2. Write 0xAA to the NVMKEY register.
3. Set the WR bit in the NVMCON register.
4. Execute three `NOP` instructions.

All erase and write cycles are self-timed. The WR bit should be polled to determine if the erase or write cycle has been completed.

REGISTER 3-1: NVMCON: NON-VOLATILE MEMORY (NVM) CONTROL REGISTER

R/SO-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
WR(1)	WREN(1)	WRERR(1)	NVMSIDL(2)	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	NVMOP<3:0>(1,3,4)			
bit 7							bit 0

Legend:	SO = Settable only bit		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **WR:** Write Control bit(1)
- 1 = Initiates a Flash memory program or erase operation. The operation is self-timed and the bit is cleared by hardware once operation is complete
- 0 = Program or erase operation is complete and inactive

bit 14 **WREN:** Write Enable bit(1)
- 1 = Enable Flash program/erase operations
- 0 = Inhibit Flash program/erase operations

bit 13 **WRERR:** Write Sequence Error Flag bit(1)
- 1 = An improper program or erase sequence attempt or termination has occurred (bit is set automatically on any set attempt of the WR bit)
- 0 = The program or erase operation completed normally

bit 12 **NVMSIDL:** NVM Stop-in-Idle Control bit(2)
- 1 = Discontinue primary and auxiliary Flash operation when the device enters Idle mode
- 0 = Continue primary and auxiliary Flash operation when the device enters Idle mode

bit 11-4 **Unimplemented:** Read as '0'

bit 3-0 **NVMOP<3:0>:** NVM Operation Select bits(1,3,4)
- 1111 = Bulk erase user memory, executive memory, and User ID Words
- 1110 = Reserved
- 1101 = Bulk erase user memory only
- 1100 = Reserved
- 1011 = Reserved
- 1010 = Reserved
- 0011 = Memory page erase operation
- 0010 = Reserved
- 0001 = Memory double-word program operation(5)
- 0000 = Reserved

Note 1: These bits can only be reset on a Power-on Reset (POR).
2: If this bit is set, upon exiting Idle mode, there is a delay (TNPD) before Flash memory becomes operational.
3: All other combinations of NVMOP<3:0> are unimplemented.
4: Execution of the PWRSAV instruction is ignored while any of the NVM operations are in progress.
5: Two adjacent words are programmed during execution of this operation.

3.5 Erasing Program Memory

The procedure for erasing user memory (including Configuration bits) is shown in Figure 3-6. For page erase operations, the NVMCON value should be modified suitably, according to Table 3-2.

Table 3-4 illustrates the ICSP programming process for Erasing program memory.

Note:	Program memory must be erased before writing any data to program memory.

FIGURE 3-6: ERASE FLOW

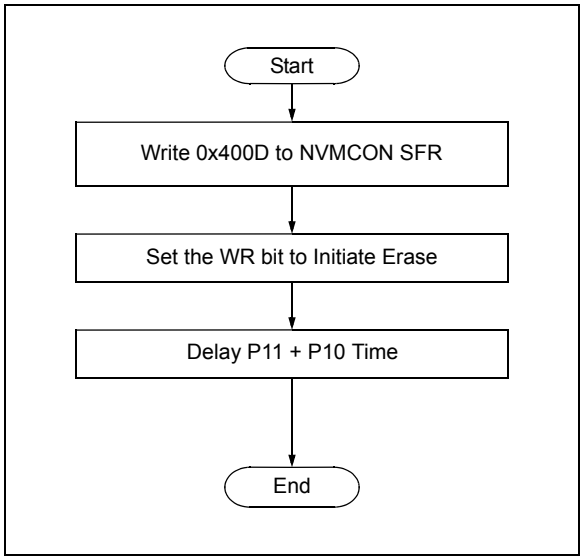


TABLE 3-4: SERIAL INSTRUCTION EXECUTION FOR ERASING CODE MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 2: Set the NVMCON to erase all program memory.		
0000	2400DA	MOV #0x400D, W10
0000	88394A	MOV W10, NVMCON
0000	000000	NOP
0000	000000	NOP
Step 3: Initiate the erase cycle.		
0000	200551	MOV #0x55, W1
0000	883971	MOV W1, NVMKEY
0000	200AA1	MOV #0xAA, W1
0000	883971	MOV W1, NVMKEY
0000	A8E729	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 4: Wait for Bulk Erase operation to complete and make sure WR bit is clear.		
—	—	Externally time ‘P11’ ms (see **Section 9.0 “AC/DC Characteristics and Timing Requirements”**) to allow sufficient time for the Bulk Erase operation to complete.

3.6 Writing Code Memory

Figure 3-7 provides a high level description of how to write to code memory. First, the device is configured for writes, two words are loaded into the write latches and the write pointer is incremented. Next, the write sequence is initiated, and finally, the WR bit is checked for the sequence to be complete. This process continues for all words to be programmed.

Table 3-5 shows the ICSP programming details for writing program memory.

To minimize programming time, the same packed data format that the programming executive uses is utilized. See **Section 6.2 "Programming Executive Commands"** for more details on the packed data format.

FIGURE 3-7: PROGRAM CODE MEMORY FLOW

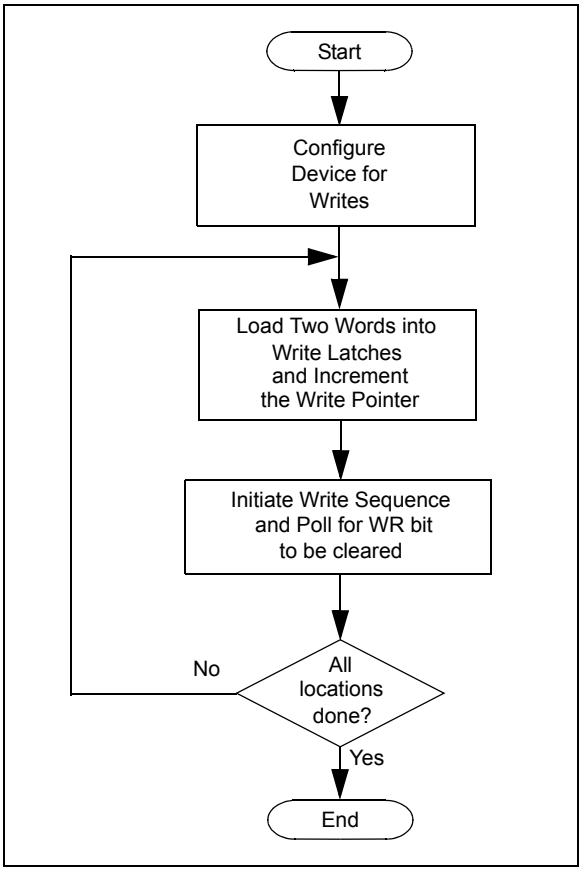


TABLE 3-5: SERIAL INSTRUCTION EXECUTION FOR WRITING CODE MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 2: Initialize the TBLPAG register for writing to latches.		
0000	200FAC	MOV #0xFA, W12
0000	8802AC	MOV W12, TBLPAG
Step 3: Load W0:W2 with the next two instruction words to program.		
0000	2xxxx0	MOV #<LSW0>, W0
0000	2xxxx1	MOV #<MSB1:MSB0>, W1
0000	2xxxx2	MOV #<LSW1>, W2

TABLE 3-5: SERIAL INSTRUCTION EXECUTION FOR WRITING CODE MEMORY (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 4: Set the read pointer (W6) and writer pointer (W7), and load the (next set of) write latches.		
0000	EB0300	CLR W6
0000	000000	NOP
0000	EB0380	CLR W7
0000	000000	NOP
0000	BB0BB6	TBLWTL[W6++], [W7]
0000	000000	NOP
0000	000000	NOP
0000	BBDBB6	TBLWTH.B[W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BBEBB6	TBLWTH.B[W6++], [++W7]
0000	000000	NOP
0000	000000	NOP
0000	BB1BB6	TBLWTL.W[W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
Step 5: Set the NVMADRU/NVMADR register-pair to point to the correct address.		
0000	2xxxx3	MOV #DestinationAddress<15:0>, W3
0000	2xxxx4	MOV #DestinationAddress<23:16>, W4
0000	883953	MOV W3, NVMADR
0000	883964	MOV W4, NVMADRU
Step 6: Set the NVMCON to program two instruction words.		
0000	24001A	MOV #0x4001, W10
0000	000000	NOP
0000	88394A	MOV W10, NVMCON
0000	000000	NOP
0000	000000	NOP
Step 7: Initiate the write cycle.		
0000	200551	MOV #0x55, W1
0000	883971	MOV W1, NVMKEY
0000	200AA1	MOV #0xAA, W1
0000	883971	MOV W1, NVMKEY
0000	A8E729	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 8: Wait for Program operation to complete and make sure WR bit is clear.		
—	—	Externally time 'P13' ms (see **Section 9.0 "AC/DC Characteristics and Timing Requirements"**) to allow sufficient time for the Program operation to complete.
0000	000000	NOP
0000	803940	MOV NVMCON, W0
0000	887C40	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
—	—	Repeat until the WR bit is clear.
Step 9: Repeat steps 3-8 until all code memory is programmed.		

3.7 Writing Configuration Bits

The procedure for writing Configuration bits is similar to the procedure for writing code memory, except that only the lowest byte of the 24-bit word is usable data, with the remaining bytes filled with '1'.

To change the values of the Configuration bits once they have been programmed, the device must be erased, as described in **Section 3.5 "Erasing Program Memory"**, and reprogrammed to the desired value. It is not possible to program a '0' to '1', but the Configuration bits may be programmed from a '1' to '0' to enable code protection.

Table 3-6 shows the ICSP programming details for writing the Configuration bits.

In order to verify the data by reading the Configuration bits after performing the write, the code protection bits should initially be programmed to a '1' to ensure that the verification can be performed properly. After verification is finished, the code protection bit can be programmed to a '0' by using a word write to the appropriate Configuration byte.

TABLE 3-6: SERIAL INSTRUCTION EXECUTION FOR WRITING CONFIGURATION BYTES

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 2: Initialize the TBLPAG register for writing to latches.		
0000	200FAC	MOV #0xFA, W12
0000	8802AC	MOV W12, TBLPAG
Step 3: Load W0:W1 with the next two Configuration bytes to program.		
0000	2FFxx0	MOV #<0xFF:Data>, W0
0000	2FFxx1	MOV #<0xFF:Data>, W1
Step 4: Set the write pointer (W3) and load the write latches.		
0000	EB0180	CLR W3
0000	000000	NOP
0000	BB1980	TBLWTL W0, [W3++]
0000	000000	NOP
0000	000000	NOP
0000	BB0981	TBLWTL W1, [W3]
0000	000000	NOP
0000	000000	NOP
Step 5: Set the NVMADRU/NVMADR register pair to point to the correct Configuration byte address.		
0000	2xxxx4	MOV #DestinationAddress<15:0>, W4
0000	2xxxx5	MOV #DestinationAddress<23:16>, W5
0000	883954	MOV W4, NVMADR
0000	883965	MOV W5, NVMADRU
Step 6: Set the NVMCON to program two instruction words.		
0000	24001A	MOV #0x4001, W10
0000	000000	NOP
0000	88394A	MOV W10, NVMCON
0000	000000	NOP
0000	000000	NOP

TABLE 3-6: SERIAL INSTRUCTION EXECUTION FOR WRITING CONFIGURATION BYTES (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 7: Initiate the write cycle.		
0000	200551	MOV #0x55, W1
0000	883971	MOV W1, NVMKEY
0000	200AA1	MOV #0xAA, W1
0000	883971	MOV W1, NVMKEY
0000	A8E729	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 8: Wait for Program operation to complete and make sure the WR bit is clear.		
—	—	Externally time 'P13' ms (see **Section 9.0 "AC/DC Characteristics and Timing Requirements"**) to allow sufficient time for the Program operation to complete.
0000	000000	NOP
0000	803940	MOV NVMCON, W0
0000	000000	NOP
0000	887C40	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
—	—	Repeat until the WR bit is clear.
Step 9: Repeat steps 3-8 until all code memory is programmed.		

3.8 Writing User ID Words

The procedure for writing the User ID Words is similar to the procedure for writing the Configuration bits. To change the values of the User ID Words after they have been programmed, the device must be erased. Because the user ID words are part of executive memory, they must be erased as described in **Section 5.2 "Erasing Executive Memory"**, and reprogrammed to the desired value. It is not possible to program a '0' to '1', but the User ID Words may be programmed from a '1' to '0'. Refer to the User ID Words Register Map (Table 2-4) for the locations of the User ID Words.

Table 3-7 shows the ICSP programming details for writing the User ID Words.

TABLE 3-7: SERIAL INSTRUCTION EXECUTION FOR WRITING USER ID WORDS

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 2: Initialize the TBLPAG register for writing to latches.		
0000	200FAC	MOV #0xFA, W12
0000	8802AC	MOV W12, TBLPAG
Step 3: Load W0:W1 with the next two User ID Words to program.		
0000	2xxxx0	MOV #<Data>, W0
0000	2xxxx1	MOV #<Data>, W1
Step 4: Set the write pointer (W3) and load the write latches.		
0000	EB0180	CLR W3
0000	000000	NOP
0000	BB1980	TBLWTL W0, [W3++]
0000	000000	NOP
0000	000000	NOP
0000	BB0981	TBLWTL W1, [W3]
0000	000000	NOP
0000	000000	NOP
Step 5: Set the NVMADRU/NVMADR register pair to point to the correct User ID Word address.		
0000	2xxxx4	MOV #DestinationAddress<15:0>, W4
0000	2xxxx5	MOV #DestinationAddress<23:16>, W5
0000	883954	MOV W4, NVMADR
0000	883965	MOV W5, NVMADRU
Step 6: Set the NVMCON to program two instruction words.		
0000	24001A	MOV #0x4001, W10
0000	000000	NOP
0000	88394A	MOV W10, NVMCON
0000	000000	NOP
0000	000000	NOP

TABLE 3-7: SERIAL INSTRUCTION EXECUTION FOR WRITING USER ID WORDS (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 7: Initiate the write cycle.		
0000	200551	MOV #0x55, W1
0000	883971	MOV W1, NVMKEY
0000	200AA1	MOV #0xAA, W1
0000	883971	MOV W1, NVMKEY
0000	A8E729	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 8: Wait for Program operation to complete and make sure the WR bit is clear.		
—	—	Externally time ‘P13’ ms (see **Section 9.0 “AC/DC Characteristics and Timing Requirements”**) to allow sufficient time for the Program operation to complete.
0000	000000	NOP
0000	803940	MOV NVMCON, W0
0000	000000	NOP
0000	887C40	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
—	—	Repeat until the WR bit is clear.
Step 9: Repeat steps 3-8 until all code memory is programmed.		

3.9 Reading Code Memory

Reading from code memory is performed by executing a series of TBLRD instructions and clocking out the data using the REGOUT command.

Table 3-8 shows the ICSP programming details for reading code memory.

To minimize reading time, the same packed data format that the programming executive uses is utilized. See **Section 6.2 "Programming Executive Commands"** for more details on the packed data format.

TABLE 3-8: SERIAL INSTRUCTION EXECUTION FOR READING CODE MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 2: Initialize TBLPAG and the read pointer (W6) for TBLRD instruction.		
0000	200xx0	MOV #<SourceAddress23:16>, W0
0000	8802A0	MOV W0, TBLPAG
0000	2xxxx6	MOV #<SourceAddress15:0>, W6

TABLE 3-8: SERIAL INSTRUCTION EXECUTION FOR READING CODE MEMORY (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 3: Initialize the write pointer (W7) and store the next four locations of code memory to W0:W5.		
0000	EB0380	CLR W7
0000	000000	NOP
0000	BA1B96	TBLRDL [W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BADBB6	TBLRDH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BADBD6	TBLRDH.B [++W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BA1BB6	TBLRDL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BA1B96	TBLRDL [W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BADBB6	TBLRDH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BADBD6	TBLRDH.B [++W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BA0BB6	TBLRDL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP

TABLE 3-8: SERIAL INSTRUCTION EXECUTION FOR READING CODE MEMORY (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 4: Output W0:W5 using the VISI register and REGOUT command.		
0000	887C40	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	000000	NOP
0000	887C41	MOV W1, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	000000	NOP
0000	887C42	MOV W2, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	000000	NOP
0000	887C43	MOV W3, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	000000	NOP
0000	887C44	MOV W4, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	000000	NOP
0000	887C45	MOV W5, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	000000	NOP
Step 5: Reset device internal PC.		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 6: Repeat steps 3-5 until all desired code memory is read.		

3.10 Reading Configuration Bits

The procedure for reading Configuration bits is similar to the procedure for reading code memory, except that a single byte is read instead of 24-bit instructions. Since there are multiple Configuration bytes, they are read one at a time.

Table 3-9 shows the ICSP programming details for reading the Configuration bits.

TABLE 3-9: SERIAL INSTRUCTION EXECUTION FOR READING CONFIGURATION BYTES

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 2: Initialize TBLPAG and the read pointer (W6) for TBLRD instruction.		
0000	200xx0	MOV #<Address23:16>, W0
0000	8802A0	MOV W0, TBLPAG
0000	2xxxx6	MOV #<Address15:0>, W6
Step 3: Clear the write pointer (W7) and store the Configuration byte.		
0000	EB0380	CLR W7
0000	000000	NOP
0000	BA5B96	TBLRDL.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 4: Output W0:W5 using the VISI register and REGOUT command.		
0000	887C40	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
Step 5: Reset device internal PC.		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 6: Repeat steps 3-5 until all Configuration bytes are read.		

3.11 Reading User ID Words

The procedure for reading User ID Words is similar to the procedure for reading configuration bits. Since there are multiple User ID Words, they are read one at a time.

Table 3-10 shows the ICSP programming details for reading the User ID Words.

TABLE 3-10: SERIAL INSTRUCTION EXECUTION FOR READING USER ID WORDS

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 2: Initialize TBLPAG and the read pointer (W6) for TBLRD instruction.		
0000	200xx0	MOV #<Address23:16>, W0
0000	8802A0	MOV W0, TBLPAG
0000	2xxxx6	MOV #<Address15:0>, W6
Step 3: Clear the write pointer (W7) and store the User ID words.		
0000	EB0380	CLR W7
0000	000000	NOP
0000	BA1BB6	TBLRDL.W [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 4: Output W0:W5 using the VISI register and REGOUT command.		
0000	887C40	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
Step 5: Reset device internal PC.		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 6: Repeat steps 3-5 until all User ID Words are read.		

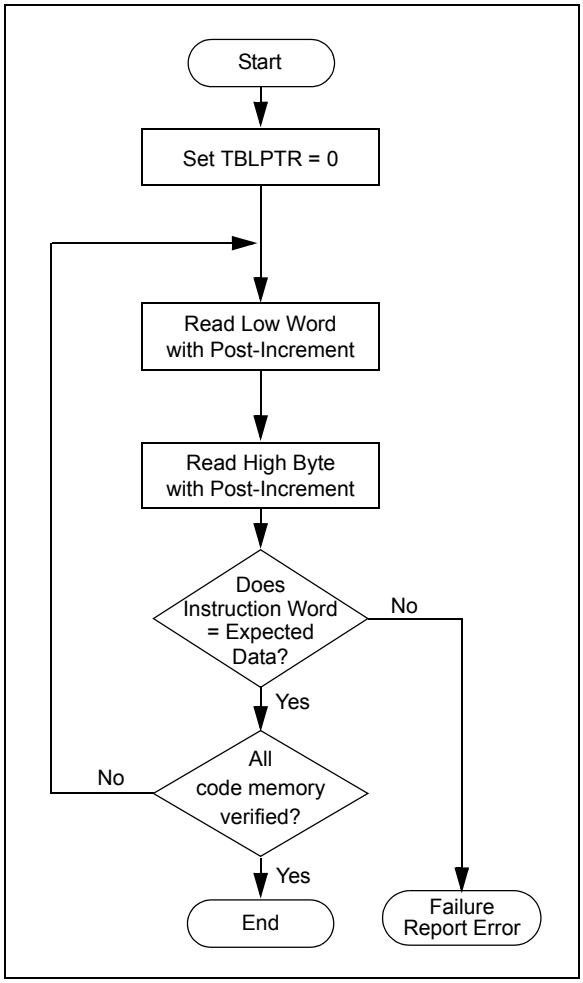
3.12 Verify Code Memory and Configuration Bits

The verify step involves reading back the code memory space and comparing it against the copy held in the programmer's buffer. The Configuration words are verified with the rest of the code.

The verify process is illustrated in Figure 3-8. The lower word of the instruction is read, and then the lower byte of the upper word is read and compared against the instruction stored in the programmer's buffer. Refer to **Section 3.9 "Reading Code Memory"** for implementation details of reading code memory.

Note:	Because the Configuration bytes include the device code protection bit, code memory should be verified immediately after writing, if the code protection is to be enabled. This is because the device will not be readable or verifiable if a device Reset occurs after the code-protect bit has been cleared.

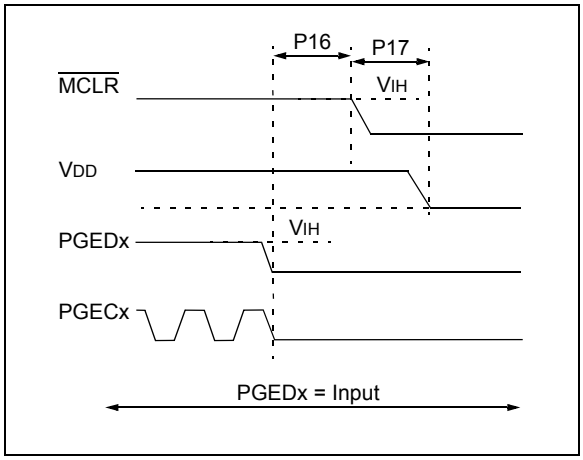
FIGURE 3-8: VERIFY CODE MEMORY FLOW



3.13 Exiting ICSP Mode

Exiting Program/Verify mode is done by removing VIH from MCLR, as illustrated in Figure 3-9. The only requirement for exit is that an interval P16 should elapse between the last clock and program signals on PGECx and PGEDx before removing VIH.

FIGURE 3-9: EXITING ICSP™ MODE



4.0 DEVICE PROGRAMMING – ENHANCED ICSP

This section discusses programming the device through Enhanced ICSP and the programming executive. The programming executive resides in executive memory (separate from code memory) and is executed when Enhanced ICSP programming mode is entered. The programming executive provides the mechanism for the programmer (host device) to program and verify the dsPIC33E/PIC24E devices using a simple command set and communication protocol. There are several basic functions provided by the programming executive:

- Read Memory
- Erase Memory
- Program Memory
- Blank Check

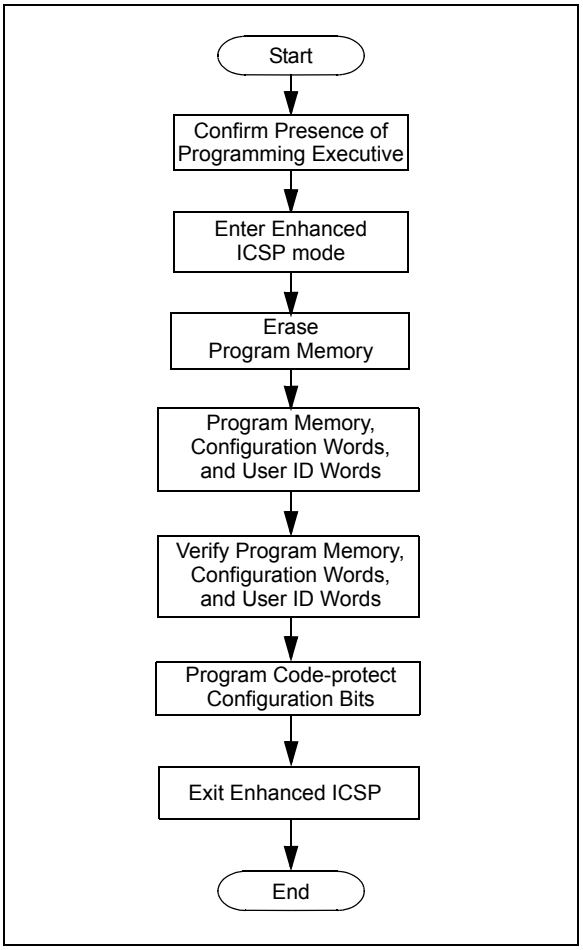
The programming executive performs the low-level tasks required for erasing, programming and verifying a device. This allows the programmer to program the device by issuing the appropriate commands and data. A detailed description for each command is provided in **Section 6.2 "Programming Executive Commands"**.

Note:	The programming executive uses the device's data RAM for variable storage and program execution. After running the programming executive, no assumptions should be made about the contents of data RAM.

4.1 Overview of the Programming Process

Figure 4-1 shows the high-level overview of the programming process. First, it must be determined if the programming executive is present in executive memory, and then Enhanced ICSP mode is entered. The program memory is then erased, and the program memory and Configuration words are programmed and verified. Last, the code-protect Configuration bits are programmed (if required) and Enhanced ICSP mode is exited.

FIGURE 4-1: HIGH-LEVEL ENHANCED ICSP™ PROGRAMMING FLOW



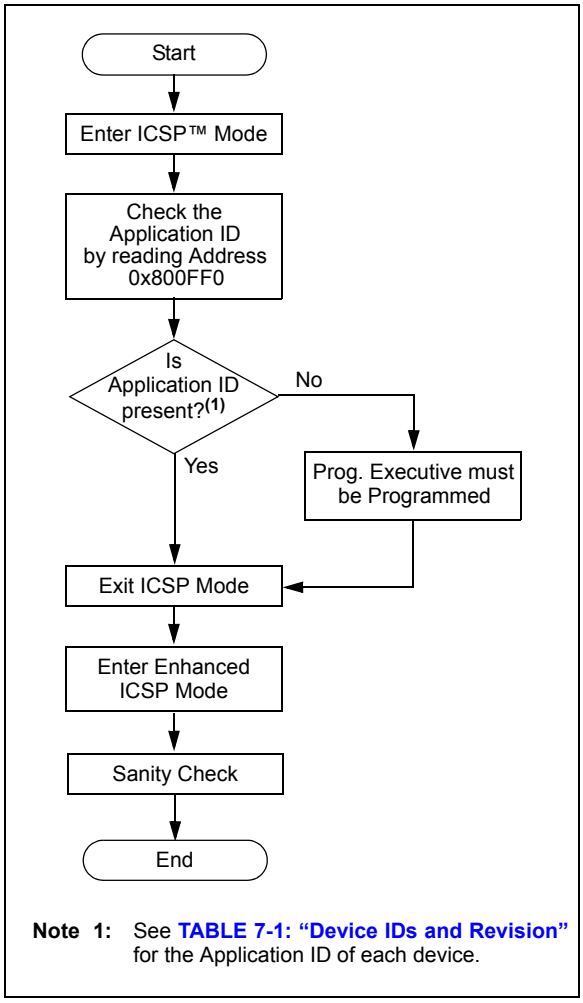
4.2 Confirming the Presence of the Programming Executive

Before programming, the programmer must confirm that the programming executive is stored in executive memory. The procedure for this task is illustrated in Figure 4-2.

First, ICSP mode is entered. Then, the unique Application ID Word stored in executive memory is read. If the programming executive is resident, the correct Application ID Word is read and programming can resume as normal. However, if the Application ID Word is not present, the programming executive must be programmed to executive code memory using the method described in **Section 5.0 "Programming the Programming Executive to Memory"**. See Table 7-1 for the Application ID of each device.

Section 3.0 "Device Programming – ICSP" describes the ICSP programming method. **Section 4.3 "Reading the Application ID Word"** describes the procedure for reading the Application ID Word in ICSP mode.

FIGURE 4-2: CONFIRMING PRESENCE OF PROGRAMMING EXECUTIVE



Note 1: See **TABLE 7-1: "Device IDs and Revision"** for the Application ID of each device.

4.3 Reading the Application ID Word

The Application ID Word is stored at address 0x800FF0 in executive code memory. To read this memory location, you must use the SIX control code to move this program memory location to the VISI register. Then, the REGOUT control code must be used to clock the contents of the VISI register out of the device. The corresponding control and instruction codes that must be serially transmitted to the device to perform this operation are shown in Table 4-1.

After the programmer has clocked out the Application ID Word, it must be inspected. If the application ID has the value shown in **TABLE 7-1: "Device IDs and Revision"**, the programming executive is resident in memory and the device can be programmed using the mechanism described in **Section 4.0 "Device Programming – Enhanced ICSP"**. However, if the application ID has any other value, the programming executive is not resident in memory; it must be loaded to memory before the device can be programmed. The procedure for loading the programming executive to memory is described in **Section 5.0 "Programming the Programming Executive to Memory"**.

TABLE 4-1: SERIAL INSTRUCTION EXECUTION FOR READING THE APPLICATION ID WORD

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 2: Initialize TBLPAG and the read pointer (W0) for TBLRD instruction.		
0000	200800	MOV #0x80, W0
0000	8802A0	MOV W0, TBLPAG
0000	20FF00	MOV #0xFF0, W0
0000	20F881	MOV #VISI, W1
0000	000000	NOP
0000	BA0890	TBLRDL [W0], [W1]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 3: Output the VISI register using the REGOUT command.		
0001	<VISI>	Clock out contents of the VISI register.

4.4 Entering Enhanced ICSP Mode

As illustrated in Figure 4-3, entering Enhanced ICSP Program/Verify mode requires three steps:

1. The MCLR pin is briefly driven high, and then low.
2. A 32-bit key sequence is clocked into PGEDx.
3. MCLR is then driven high within a specified period of time and held.

The programming voltage applied to MCLR is VIH, which is essentially VDD in dsPIC33E/PIC24E devices. There is no minimum time requirement for holding at VIH. After VIH is removed, an interval of at least P18 must elapse before presenting the key sequence on PGEDx.

The key sequence is a specific 32-bit pattern, '0100 1101 0100 0011 0100 1000 0101 0000' (more easily remembered as 0x4D434850 in hexadecimal format). The device will enter Program/Verify mode only if the key sequence is valid. The Most Significant bit (MSb) of the most significant nibble must be shifted in first.

Once the key sequence is complete, VIH must be applied to MCLR and held at that level for as long as Program/Verify mode is to be maintained. An interval time of at least P19, P7, and P1 * 5 must elapse before presenting data on PGEDx. Signals appearing on PGEDx before P7 has elapsed will not be interpreted as valid.

On successful entry, the program memory can be accessed and programmed in serial fashion. While in the Program/Verify mode, all unused I/Os are placed in the high-impedance state.

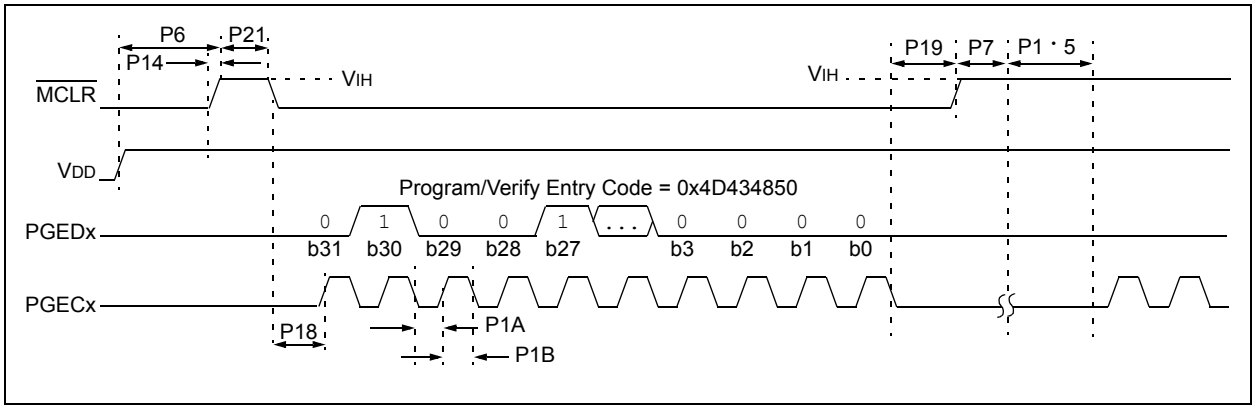
4.5 Blank Check

The term "Blank Check" implies verifying that the device has been successfully erased and has no programmed memory locations. A blank or erased memory location is always read as '1'.

The Device ID registers (0xFF0000:0xFF0002) can be ignored by the Blank Check since this region stores device information that cannot be erased. Additionally, all unimplemented memory space should be ignored from the Blank Check.

The QBLANK command is used for the Blank Check. It determines if the code memory is erased by testing these memory regions. A 'BLANK' or 'NOT BLANK' response is returned. If it is determined that the device is not blank, it must be erased before attempting to program the chip.

FIGURE 4-3: ENTERING ENHANCED ICSP™ MODE



4.6 Code Memory Programming

4.6.1 PROGRAMMING METHODOLOGY

There are two commands that can be used for programming code memory when utilizing the Programming Executive. The PROG2W command programs and verifies two 24-bit instruction words into the program memory starting at the address specified. The second and faster command PROGP, allows up to 64 24-bit instruction words to be programmed and verified into program memory starting at the address specified. See **Section 6.0 "The Programming Executive"** for a full description for each of these commands.

Figure 4-4 and Figure 4-5 show the programming methodology for the PROG2W and PROGP commands. In both instances, 22K instruction words of the dsPIC33EP64MC206 device are programmed.

Note:	If a bootloader needs to be programmed, the bootloader code must not be programmed into the first page of code memory. For example, if a bootloader located at address 0x200 attempts to erase the first page, it would inadvertently erase itself. Instead, program the bootloader into the second page (e.g., 0x400).

FIGURE 4-4: FLOWCHART FOR DOUBLE WORD PROGRAMMING

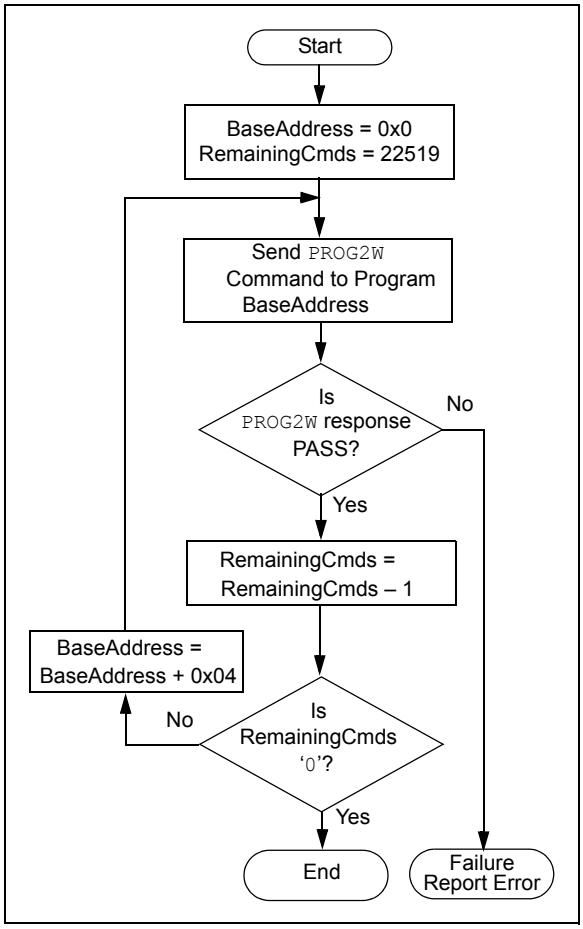
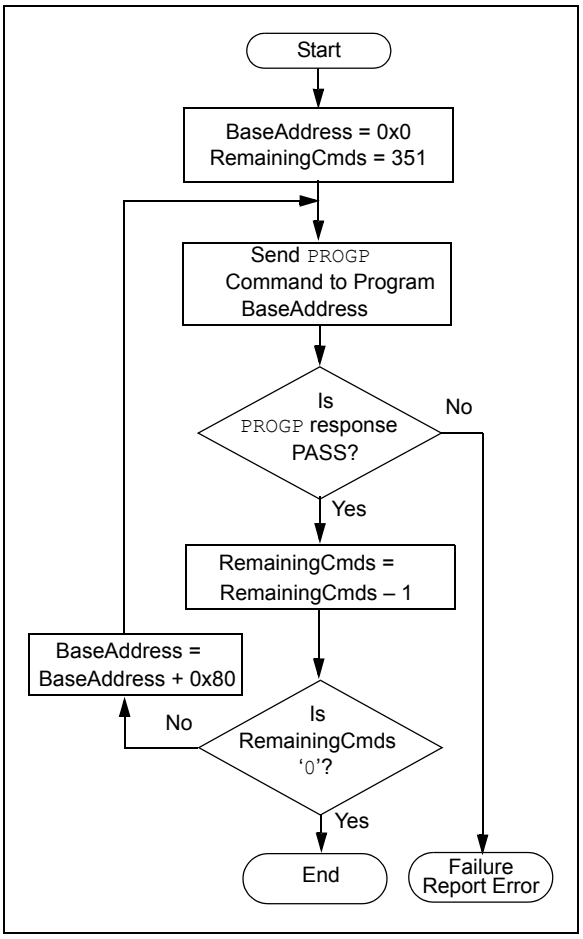


FIGURE 4-5: FLOWCHART FOR MULTIPLE WORD PROGRAMMING

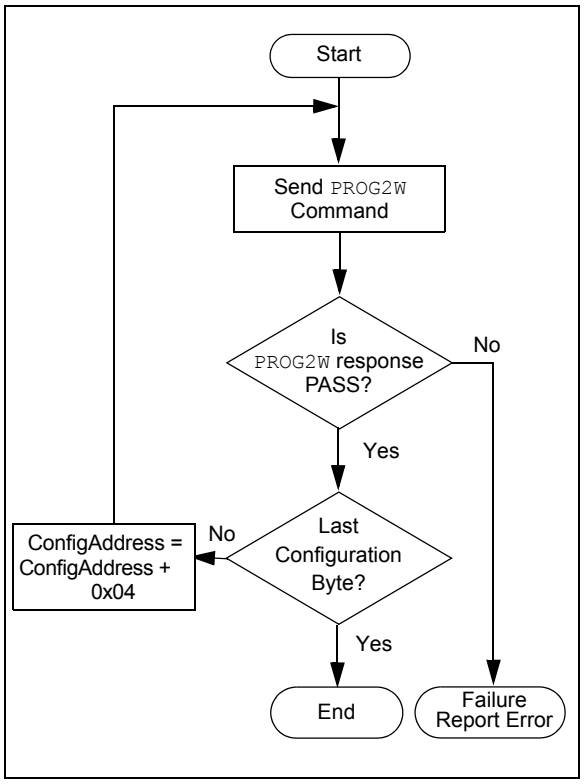


4.7 Configuration Bit Programming

Configuration bits are programmed one at a time using the PROG2W command. This command specifies the configuration data and address. When Configuration bits are programmed, any unimplemented bits must be programmed with a '1'.

Multiple PROG2W commands are required to program all Configuration bits. A flowchart for Configuration bit programming is shown in Figure 4-6.

FIGURE 4-6: CONFIGURATION BIT PROGRAMMING FLOW



4.8 Programming Verification

After code memory is programmed, the contents of memory can be verified to ensure that programming was successful. Verification requires code memory to be read back and compared against the copy held in the programmer's buffer.

The READP command can be used to read back all the programmed code memory and Configuration words.

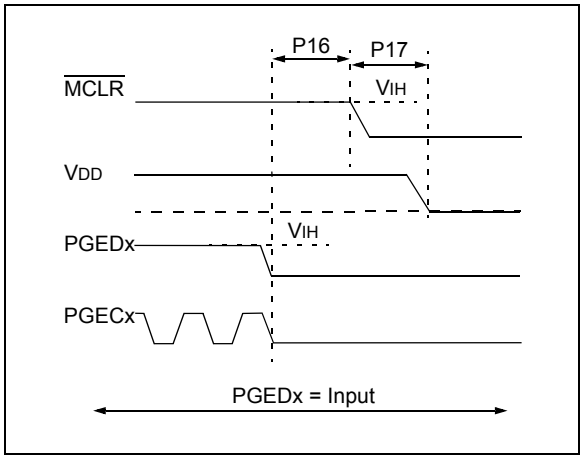
Alternatively, you can have the programmer perform the verification after the entire device is programmed, using a checksum computation.

See **Section 8.0 "Checksum Computation"** for more information on calculating the checksum.

4.9 Exiting Enhanced ICSP Mode

Exiting Program/Verify mode is done by removing VIH from MCLR, as illustrated in Figure 4-7. The only requirement for exit is that an interval P16 should elapse between the last clock and program signals on PGECx and PGEDx before removing VIH.

FIGURE 4-7: EXITING ENHANCED ICSP™ MODE



5.0 PROGRAMMING THE PROGRAMMING EXECUTIVE TO MEMORY

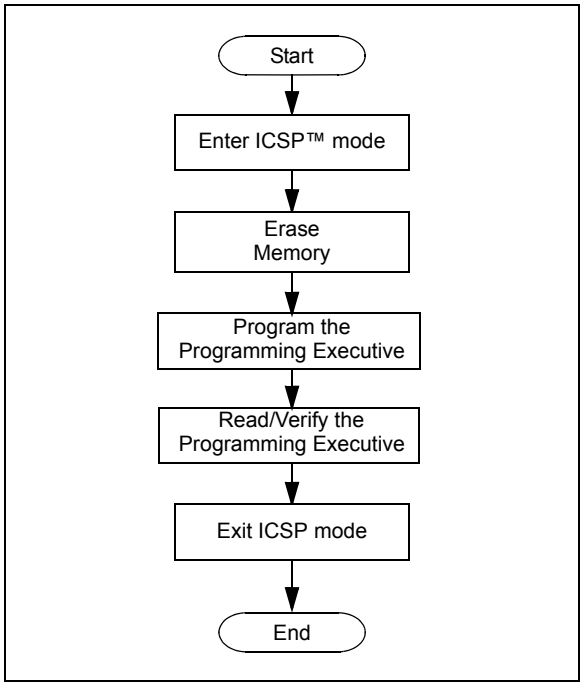
Note: The Programming Executive (PE) can be located within the following folder within your installation of MPLAB® IDE: `...\Microchip\MPLAB IDE\REAL ICE`, and then selecting the Hex PE file: `RIPE_10a_xxxxxx.hex`

5.1 Overview

If it is determined that the programming executive is not present in executive memory (as described in **Section 4.2 "Confirming the Presence of the Programming Executive"**), the programming executive must be programmed to executive memory.

Figure 5-1 shows the high level process of programming the programming executive into executive memory. First, ICSP mode must be entered and executive memory and user memory are erased, and then the programming executive is programmed and verified. Finally, ICSP mode is exited.

FIGURE 5-1: HIGH-LEVEL PROGRAMMING EXECUTIVE PROGRAMMING FLOW



5.2 Erasing Executive Memory

The procedure for erasing executive memory is similar to that of erasing program memory and is shown in Figure 5-2. It consists of setting NVMCON to 0x400F, and then executing the programming cycle. Note that program memory is also erased with this operation.

Table 5-1 illustrates the ICSP programming process for Bulk Erasing memory.

Note: The programming executive must always be erased before it is programmed, as described in Figure 5-1.

FIGURE 5-2: BULK ERASE FLOW

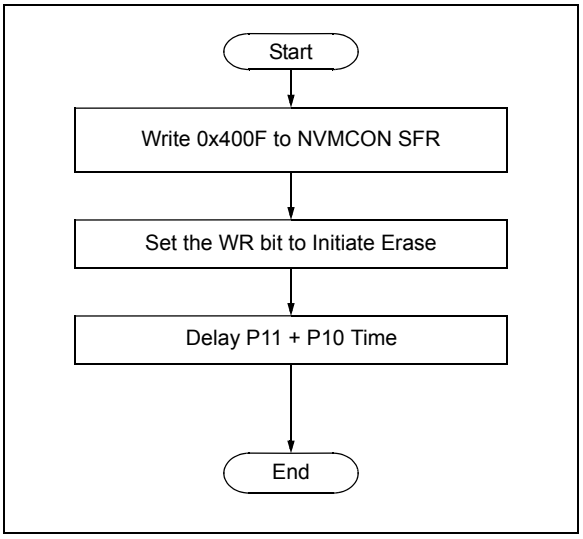


TABLE 5-1: SERIAL INSTRUCTION EXECUTION FOR BULK ERASING CODE MEMORY AND EXECUTIVE MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 2: Set the NVMCON to erase all memory.		
0000	2400FA	MOV #0x400F, W10
0000	88394A	MOV W10, NVMCON
0000	000000	NOP
0000	000000	NOP
Step 3: Initiate the erase cycle.		
0000	200551	MOV #0x55, W1
0000	883971	MOV W1, NVMKEY
0000	200AA1	MOV #0xAA, W1
0000	883971	MOV W1, NVMKEY
0000	A8E729	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 4: Wait for Bulk Erase operation to complete and make sure WR bit is clear.		
—	—	Externally time ‘P11’ ms (see **Section 9.0 “AC/DC Characteristics and Timing Requirements”**) to allow sufficient time for the Bulk Erase operation to complete.

5.3 Program the Programming Executive

Storing the programming executive to executive memory is similar to normal programming of code memory. The executive memory must first be erased, and then the programming executive must be programmed a double word at a time. This control flow is summarized in Figure 5-3.

Table 5-2 illustrates the ICSP programming process for programming executive memory. To minimize programming time, the same packed data format that the programming executive uses is utilized. See **Section 6.2 "Programming Executive Commands"** for more details on the packed data format.

FIGURE 5-3: PROGRAM CODE MEMORY FLOW

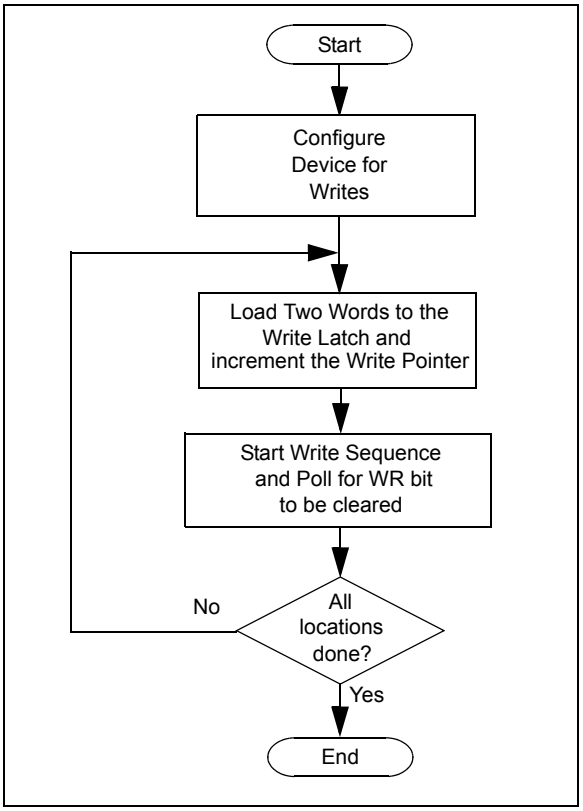


TABLE 5-2: PROGRAMMING THE PROGRAMMING EXECUTIVE

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 2: Initialize the TBLPAG register for writing to latches.		
0000	200FAC	MOV #0xFA, W12
0000	8802AC	MOV W12, TBLPAG
Step 3: Load W0:W2 with the next two instruction words to program.		
0000	2xxxx0	MOV #<LSW0>, W0
0000	2xxxx1	MOV #<MSB1:MSB0>, W1
0000	2xxxx2	MOV #<LSW1>, W2

TABLE 5-2: PROGRAMMING THE PROGRAMMING EXECUTIVE (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 4: Set the read pointer (W6) and the write pointer (W7), and load the write latches.		
0000	EB0300	CLR W6
0000	000000	NOP
0000	EB0380	CLR W7
0000	000000	NOP
0000	BB0BB6	TBLWTL[W6++], [W7]
0000	000000	NOP
0000	000000	NOP
0000	BBDBB6	TBLWTH.B[W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BBEBB6	TBLWTH.B[W6++], [++W7]
0000	000000	NOP
0000	000000	NOP
0000	BB1BB6	TBLWTL.W[W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
Step 5: Set the NVMADRU/NVMADR register-pair to point to the correct row.		
0000	2xxxx3	MOV #DestinationAddress<15:0>, W3
0000	2xxxx4	MOV #DestinationAddress<23:16>, W4
0000	883953	MOV W3, NVMADR
0000	883964	MOV W4, NVMADRU
Step 6: Set the NVMCON to program two instruction words.		
0000	24001A	MOV #0x4001, W10
0000	000000	NOP
0000	88394A	MOV W10, NVMCON
0000	000000	NOP
0000	000000	NOP
Step 7: Initiate the write cycle.		
0000	200551	MOV #0x55, W1
0000	883971	MOV W1, NVMKEY
0000	200AA1	MOV #0xAA, W1
0000	883971	MOV W1, NVMKEY
0000	A8E729	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP

TABLE 5-2: PROGRAMMING THE PROGRAMMING EXECUTIVE (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 8: Wait for Program operation to complete and make sure WR bit is clear.		
—	—	Externally time 'P13' ms (see **Section 9.0 "AC/DC Characteristics and Timing Requirements"**) to allow sufficient time for the Program operation to complete.
0000	000000	NOP
0000	803940	MOV NVMCON, W0
0000	000000	NOP
0000	887C40	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
—	—	Repeat until the WR bit is clear.
Step 9: Repeat steps 3-8 until all code memory is programmed.		

5.4 Reading Executive Memory

Reading from executive memory is performed by executing a series of TBLRD instructions and clocking out the data using the REGOUT command.

Table 5-3 shows the ICSP programming details for reading executive memory.

To minimize reading time, the same packed data format that the programming executive uses is utilized. See **Section 6.2 "Programming Executive Commands"** for more details on the packed data format.

TABLE 5-3: SERIAL INSTRUCTION EXECUTION FOR READING CODE MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 2: Initialize TBLPAG and the read pointer (W6) for TBLRD instruction.		
0000	200xx0	MOV #<SourceAddress23:16>, W0
0000	8802A0	MOV W0, TBLPAG
0000	2xxxx6	MOV #<SourceAddress15:0>, W6

TABLE 5-3: SERIAL INSTRUCTION EXECUTION FOR READING CODE MEMORY (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 3: Initialize the write pointer (W7) and store the next four locations of code memory to W0:W5.		
0000	EB0380	CLR W7
0000	000000	NOP
0000	BA1B96	TBLRDL [W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BADBB6	TBLRDH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BADBD6	TBLRDH.B [++W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BA1BB6	TBLRDL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BA1B96	TBLRDL [W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BADBB6	TBLRDH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BADBD6	TBLRDH.B [++W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BA0BB6	TBLRDL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP

TABLE 5-3: SERIAL INSTRUCTION EXECUTION FOR READING CODE MEMORY (CONTINUED)

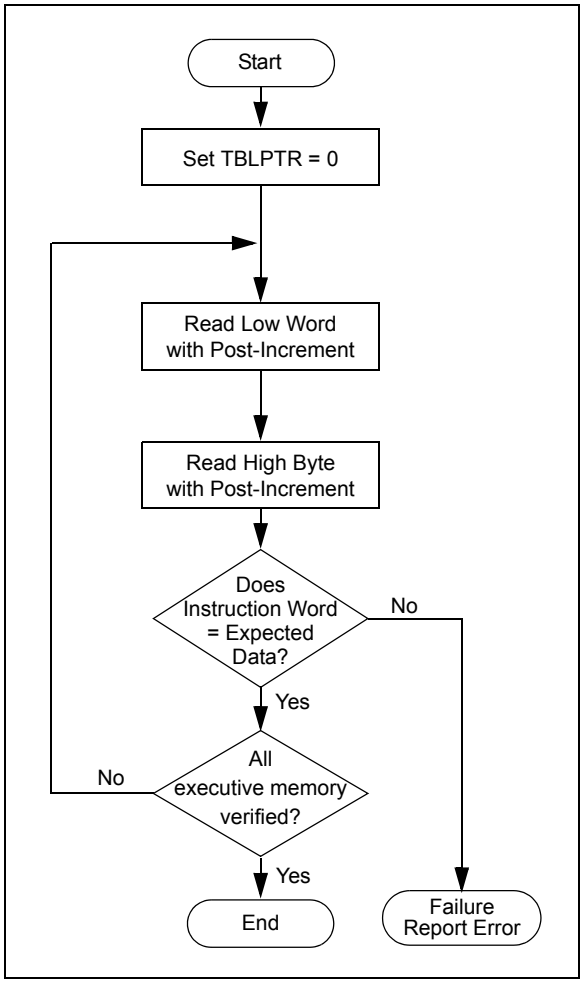
Command (Binary)	Data (Hex)	Description
Step 4: Output W0:W5 using the VISI register and REGOUT command.		
0000	887C40	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	000000	NOP
0000	887C41	MOV W1, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	000000	NOP
0000	887C42	MOV W2, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	000000	NOP
0000	887C43	MOV W3, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	000000	NOP
0000	887C44	MOV W4, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	000000	NOP
0000	887C45	MOV W5, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	000000	NOP
Step 5: Reset device internal PC.		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 6: Repeat steps 3-5 until all desired code memory is read.		

5.5 Verify Programming Executive

The verify step involves reading back the executive memory space and comparing it against the copy held in the programmer's buffer.

The verify process is illustrated in Figure 5-4. The lower word of the instruction is read, and then the lower byte of the upper word is read and compared against the instruction stored in the programmer's buffer. Refer to **Section 5.4 "Reading Executive Memory"** for implementation details of reading executive memory.

FIGURE 5-4: VERIFY PROGRAMMING EXECUTIVE MEMORY FLOW



6.0 THE PROGRAMMING EXECUTIVE

Note: When executing code from within Executive Memory, a CALL or GOTO instruction cannot immediately follow a TBLRD instruction or the Program Counter will jump to an unknown location. To avoid this condition, add a NOP instruction between any TBLRD and CALL or GOTO instruction.

6.1 Programming Executive Communication

The programmer and programming executive have a master/slave relationship, where the programmer is the master programming device and the programming executive is the slave.

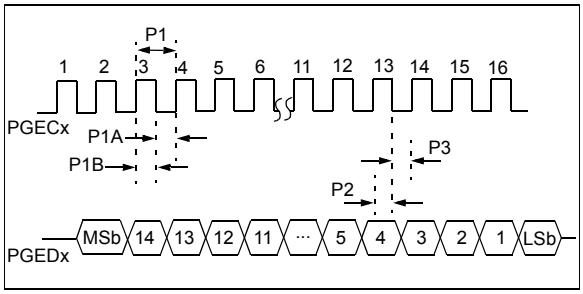
All communication is initiated by the programmer in the form of a command. Only one command at a time can be sent to the programming executive. In turn, the programming executive only sends one response to the programmer after receiving and processing a command. The programming executive command set is described in **Section 6.2 "Programming Executive Commands"**. The response set is described in **Section 6.3 "Programming Executive Responses"**.

6.1.1 COMMUNICATION INTERFACE AND PROTOCOL

The Enhanced ICSP interface is a 2-wire SPI implemented using the PGECx and PGEDx pins. The PGECx pin is used as a clock input pin and the clock source must be provided by the programmer. The PGEDx pin is used for sending command data to and receiving response data from the programming executive.

Note: For Enhanced ICSP, all serial data is transmitted on the falling edge of PGECx and latched on the rising edge of PGECx. All data transmissions are sent to the MSb first using 16-bit mode (see Figure 6-1).

FIGURE 6-1: PROGRAMMING EXECUTIVE SERIAL TIMING



Since a 2-wire SPI is used, and data transmissions are bidirectional, a simple protocol is used to control the direction of PGEDx. When the programmer completes a command transmission, it releases the PGEDx line and allows the programming executive to drive this line high. The programming executive keeps the PGEDx line high to indicate that it is processing the command.

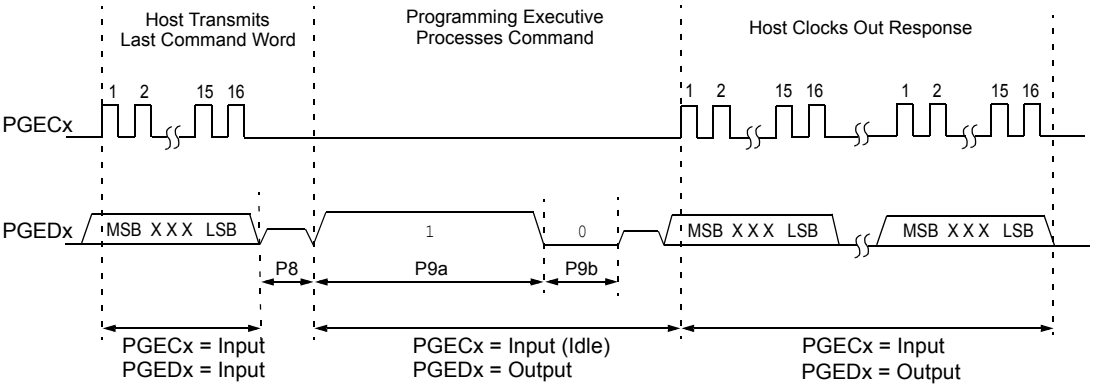
After the programming executive has processed the command, it brings PGEDx low (P9b) to indicate to the programmer that the response is available to be clocked out. The programmer can begin to clock out the response after maximum wait (P9b) and it must provide the necessary amount of clock pulses to receive the entire response from the programming executive.

After the entire response is clocked out, the programmer should terminate the clock on PGECx until it is time to send another command to the programming executive. This protocol is illustrated in Figure 6-2.

6.1.2 SPI RATE

In Enhanced ICSP mode, the dsPIC33E/PIC24E devices operate from the Fast Internal RC Oscillator, which has a nominal frequency of 7.3728 MHz. This oscillator frequency yields an effective system clock frequency of 1.8432 MHz. To ensure that the programmer does not clock too fast, it is recommended that a 1.85 MHz clock be provided by the programmer.

FIGURE 6-2: PROGRAMMING EXECUTIVE – PROGRAMMER COMMUNICATION PROTOCOL



Note 1: A delay of 25 ms is required between commands.

6.1.3 TIME OUTS

The programming executive uses no Watchdog or time out for transmitting responses to the programmer. If the programmer does not follow the flow control mechanism using PGECx as described in **Section 6.1.1 "Communication Interface and Protocol"**, it is possible that the programming executive will behave unexpectedly while trying to send a response to the programmer. Since the programming executive has no time out, it is imperative that the programmer correctly follow the described communication protocol.

As a safety measure, the programmer should use the command time outs identified in Table 6-1. If the command time out expires, the programmer should reset the programming executive and start programming the device again.

TABLE 6-1: PROGRAMMING EXECUTIVE COMMAND SET

Opcode	Mnemonic	Length (16-bit words)	Time-out	Description
0x0	SCHECK	1	1 ms	Sanity check.
0x1	READC	3	1 ms	Read an 8-bit word from the specified Device ID register.
0x2	READP	4	1 ms/word	Read 'N' 24-bit instruction words of code memory starting from the specified address.
0x3	PROG2W	6	5 ms	Program a double instruction word of code memory at the specified address and verify.
0x4	Reserved	N/A	N/A	This command is reserved. It will return a NACK.
0x5	PROGP	99	5 ms	Program 64 words of program memory at the specified starting address and verify.
0x6	Reserved	N/A	N/A	This command is reserved. It will return a NACK.
0x7	Reserved	N/A	N/A	This command is reserved. It will return a NACK.
0x8	Reserved	N/A	N/A	This command is reserved. It will return a NACK.
0x9	ERASEP	3	20 ms	Command to erase a page.
0xA	Reserved	N/A	N/A	This command is reserved. It will return a NACK.
0xB	QVER	1	1 ms	Query the programming executive software version.
0xC	CRCP	5	1s	Performs a CRC-16 on the specified range of memory.
0xD	Reserved	N/A	N/A	This command is reserved. It will return a NACK.
0xE	QBLANK	5	700 ms	Query to check whether the code memory is blank.

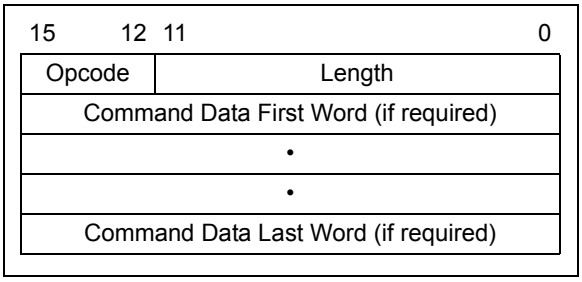
6.2 Programming Executive Commands

The programming executive command set is shown in Table 6-1. This table contains the opcode, mnemonic, length, time out and description for each command. Functional details on each command are provided in the command descriptions (**Section 6.2.4 "Command Descriptions"**).

6.2.1 COMMAND FORMAT

All programming executive commands have a general format consisting of a 16-bit header and any required data for the command (see Figure 6-3). The 16-bit header consists of a 4-bit opcode field, which is used to identify the command, followed by a 12-bit command length field.

FIGURE 6-3: COMMAND FORMAT



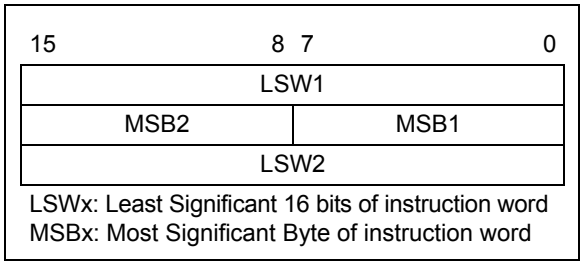
The command opcode must match one of those in the command set. Any command that is received which does not match the list in Table 6-1 will return a "NACK" response (see **Section 6.3.1.1 "Opcode Field"**).

The command length is represented in 16-bit words since the SPI operates in 16-bit mode. The programming executive uses the command length field to determine the number of words to read from the SPI port. If the value of this field is incorrect, the command will not be properly received by the programming executive.

6.2.2 PACKED DATA FORMAT

When 24-bit instruction words are transferred across the 16-bit SPI interface, they are packed to conserve space using the format illustrated in Figure 6-4. This format minimizes traffic over the SPI and provides the programming executive with data that is properly aligned for performing table write operations.

FIGURE 6-4: PACKED INSTRUCTION WORD FORMAT



Note: When the number of instruction words transferred is odd, MSB2 is zero and LSW2 cannot be transmitted.

6.2.3 PROGRAMMING EXECUTIVE ERROR HANDLING

The programming executive will "NACK" all unsupported commands. Additionally, due to the memory constraints of the programming executive, no checking is performed on the data contained in the programmer command. It is the responsibility of the programmer to command the programming executive with valid command arguments or the programming operation may fail. Additional information on error handling is provided in **Section 6.3.1.3 "QE_Code Field"**.

6.2.4 COMMAND DESCRIPTIONS

All commands supported by the programming executive are described in **Section 6.2.4.1 "SCHECK Command"** through **Section 6.2.4.7 "QVER Command"**.

6.2.4.1 SCHECK Command

15	12	11	0
Opcode		Length	

Field	Description
Opcode	0x0
Length	0x1

The SCHECK command instructs the programming executive to do nothing but generate a response. This command is used as a "Sanity Check" to verify that the programming executive is operational.

Expected Response (2 words):

0x1000
0x0002

Note:	This instruction is not required for programming, but is provided for development purposes only.

6.2.4.2 READC Command

15	12	11	8	7	0
Opcode		Length			
N			Addr_MSB		
Addr_LS					

Field	Description
Opcode	0x1
Length	0x3
N	Number of 16-bit Device ID registers to read (maximum of 256).
Addr_MSB	MSB of 24-bit source address.
Addr_LS	Least Significant 16 bits of 24-bit source address.

The READC command instructs the programming executive to read N Device ID registers, starting from the 24-bit address specified by Addr_MSB and Addr_LS. This command can only be used to read 8-bit or 16-bit data.

When this command is used to read Device ID registers, the upper byte in every data word returned by the programming executive is 0x00 and the lower byte contains the Device ID register value.

Expected Response (4 + 3 * (N – 1)/2 words for N odd):

0x1100
2 + N
Device ID Register 1
...
Device ID Register N

Note:	Reading unimplemented memory will cause the programming executive to reset. To prevent this from occurring, ensure that only memory locations present on a particular device are accessed.

6.2.4.3 READP Command

15 12	11 8	7 0
Opcode	Length	
N		
Reserved	Addr_MSB	
Addr_LS		

Field	Description
Opcode	0x2
Length	0x4
N	Number of 24-bit instructions to read (maximum of 32768).
Reserved	0x0
Addr_MSB	MSB of 24-bit source address.
Addr_LS	Least Significant 16 bits of 24-bit source address.

The READP command instructs the programming executive to read N 24-bit words of code memory, starting from the 24-bit address specified by Addr_MSB and Addr_LS. This command can only be used to read 24-bit data. All data returned in the response to this command uses the packed data format described in **Section 6.2.2 "Packed Data Format"**.

Expected Response (2 + 3 * N/2 words for N even):
0x1200
2 + 3 * N/2
Least significant program memory word 1
...
Least significant data word N

Expected Response (4 + 3 * (N – 1)/2 words for N odd):
0x1200
4 + 3 * (N – 1)/2
Least significant program memory word 1
...
MSB of program memory word N (zero padded)

Note:	Reading unimplemented memory will cause the programming executive to reset. To prevent this from occurring, ensure that only memory locations present on a particular device are accessed.

6.2.4.4 PROG2W Command

15 12	11 8	7 0
Opcode	Length	
Reserved	Addr_MSB	
Addr_LS		
DataL_LS		
DataH_MSB	DataL_MSB	
DataH_LS		

Field	Description
Opcode	0x3
Length	0x6
DataL_MSB	MSB of 24-bit data for low instruction word.
DataH_MSB	MSB of 24-bit data for high instruction word.
Addr_MSB	MSB of 24-bit destination address.
Addr_LS	Least Significant 16 bits of 24-bit destination address.
DataL_LS	Least Significant 16 bits of 24-bit data for low instruction word.
DataH_LS	Least Significant 16 bits of 24-bit data for high instruction word.

The PROG2W command instructs the programming executive to program two instruction words of code memory (6 bytes) to the specified memory address.

After the words have been programmed to code memory, the programming executive verifies the programmed data against the data in the command.

Expected Response (2 words):
0x1300
0x0002

6.2.4.5 PROGP Command

15 12	11 8	7 0
Opcode	Length	
Reserved	Addr_MSB	
Addr_LS		
D_1		
D_2		
...		
D_N		

Field	Description
Opcode	0x5
Length	0x63
Reserved	0x0
Addr_MSB	MSB of 24-bit destination address.
Addr_LS	Least Significant 16 bits of 24-bit destination address.
D_1	16-bit data word 1.
D_2	16-bit data word 2.
...	16-bit data word 3 through 95.
D_96	16-bit data word 96.

The PROGP command instructs the programming executive to program 64 instruction words starting at the specified memory address.

The data to program the memory, located in command words D_1 through D_96, must be arranged using the packed instruction word format illustrated in Figure 6-4.

After all data has been programmed to code memory, the programming executive verifies the programmed data against the data in the command.

Expected Response (2 words):

0x1500
0x0002

Note: Refer to Table 2-2 for code memory size information.

6.2.4.6 ERASEP Command

15 12	11 8	7 0
Opcode	Length	
NUM_PAGES	Addr_MSB	
Addr_LS		

Field	Description
Opcode	0x9
Length	0x3
NUM_PAGES	Up to 255
Addr_MSB	Most Significant Byte of the 24-bit address
Addr_LS	Least Significant 16 bits of the 24-bit address

The ERASEP command instructs the programming executive to page erase [NUM_PAGES] of code memory. The code memory must be erased at an “even” 512 instruction word address boundary

Expected Response (2 words):

0x1900
0x0002

6.2.4.7 QVER Command

15 12	11 0
Opcode	Length

Field	Description
Opcode	0xB
Length	0x1

The QVER command queries the version of the programming executive software stored in test memory. The “version.revision” information is returned in the response’s QE_Code using a single byte with the following format: main version in upper nibble and revision in the lower nibble (i.e., 0x23 means version 2.3 of programming executive software).

Expected Response (2 words):

0x1BMN (where “MN” stands for version M.N)

0x0002

6.2.4.8 CRCP Command

15 12	11 8	7 0
Opcode	Length	
Reserved		Addr_MSB
Addr_LSW		
Reserved		Size_MSB
Size_LSW		

Field	Description
Opcode	0xC
Length	0x5
Addr_MSB	Most Significant Byte of 24-bit address
Addr_LSW	Least Significant 16-bits of 24-bit address
Size	Number of 24-bit locations (address range divided by 2)

The CRCP command performs a CRC-16 on the range of memory specified. This command can substitute for a full chip verify. Data is shifted in a packed method, byte-wise Least Significant Byte (LSB) first.

Example:

CRC-CCITT-16 with test data of “123456789” becomes 0x29B1

Expected Response (3 words):

QE_Code: 0x1C00

Length: 0x0003

CRC Value: 0xXXXX

6.2.4.9 QBLANK Command

15 12	11 0
Opcode	Length
Reserved	Size_MSB
Size_LSW	
Reserved	Addr_MSB
Addr_LSW	

Field	Description
Opcode	0xE
Length	0x5
Size	Length of program memory to check (in 24-bit words) + Addr_MS
Addr_MSB	Most Significant Byte of the 24-bit address
Addr_LSW	Least Significant 16 bits of the 24-bit address

The QBLANK command queries the programming executive to determine if the contents of code memory are blank (contains all '1's). The size of code memory to check must be specified in the command.

The Blank Check for code memory begins at [Addr] and advances toward larger addresses for the specified number of instruction words.

QBLANK returns a QE_Code of 0xF0 if the specified code memory is blank; otherwise, QBLANK returns a QE_Code of 0x0F.

Expected Response (2 words for blank device):

0x1EF0
0x0002

Expected Response (2 words for non-blank device):

0x1E0F
0x0002

Note:	Ensure that the address range selected excludes the Configuration bytes at the top of memory when using the QBLANK command; otherwise, a non-blank response will be generated.

6.3 Programming Executive Responses

The programming executive sends a response to the programmer for each command that it receives. The response indicates if the command was processed correctly. It includes any required response data or error data.

The programming executive response set is shown in Table 6-2. This table contains the opcode, mnemonic and description for each response. The response format is described in **Section 6.3.1 "Response Format"**.

TABLE 6-2: PROGRAMMING EXECUTIVE RESPONSE OPCODES

Opcode	Mnemonic	Description
0x1	PASS	Command successfully processed.
0x2	FAIL	Command unsuccessfully processed.
0x3	NACK	Command not known.

6.3.1 RESPONSE FORMAT

All programming executive responses have a general format consisting of a two-word header and any required data for the command.

15 12	11 8	7 0
Opcode	Last_Cmd	QE_Code
Length		
D_1 (if applicable)		
...		
D_N (if applicable)		

Field	Description
Opcode	Response opcode.
Last_Cmd	Programmer command that generated the response.
QE_Code	Query code or error code.
Length	Response length in 16-bit words (includes 2 header words).
D_1	First 16-bit data word (if applicable).
D_N	Last 16-bit data word (if applicable).

6.3.1.1 Opcode Field

The opcode is a 4-bit field in the first word of the response. The opcode indicates how the command was processed (see Table 6-2). If the command was processed successfully, the response opcode is PASS. If there was an error in processing the command, the response opcode is FAIL and the QE_Code indicates the reason for the failure. If the command sent to the programming executive is not identified, the programming executive returns a NACK response.

6.3.1.2 Last_Cmd Field

The Last_Cmd is a 4-bit field in the first word of the response and indicates the command that the programming executive processed. Since the programming executive can only process one command at a time, this field is technically not required. However, it can be used to verify that the programming executive correctly received the command that the programmer transmitted.

6.3.1.3 QE_Code Field

The QE_Code is a byte in the first word of the response. This byte is used to return data for query commands and error codes for all other commands.

When the programming executive processes one of the two query commands (`QBLANK` or `QVER`), the returned opcode is always PASS and the QE_Code holds the query response data. The format of the QE_Code for both queries is shown in Table 6-3.

TABLE 6-3: QE_Code FOR QUERIES

Query	QE_Code
`QBLANK`	0x0F = Code memory is NOT blank 0xF0 = Code memory is blank
`QVER`	0xMN, where programming executive software version = M.N (i.e., 0x32 means software version 3.2).

When the programming executive processes any command other than a Query, the QE_Code represents an error code. Supported error codes are shown in Table 6-4. If a command is successfully processed, the returned QE_Code is set to 0x0, which indicates that there is no error in the command processing. If the verify of the programming for the `PROGW` command fails, the QE_Code is set to 0x1. For all other programming executive errors, the QE_Code is 0x2.

TABLE 6-4: QE_Code FOR NON-QUERY COMMANDS

QE_Code	Description
0x0	No error.
0x1	Verify failed.
0x2	Other error.

6.3.1.4 Response Length

The response length indicates the length of the programming executive's response in 16-bit words. This field includes the 2 words of the response header.

With the exception of the response for the read commands, the length of each response is only 2 words.

The response to the read commands uses the packed instruction word format described in **Section 6.2.2 "Packed Data Format"**. When reading an odd number of program memory words (N odd), the response to the `READP` command is (3 * (N + 1)/2 + 2) words. When reading an even number of program memory words (N even), the response to the `READP` command is (3 * N/2 + 2) words.

7.0 DEVICE ID

The Device ID region of memory can be used to determine variant and manufacturing information about the chip. This region of memory is read-only and can be read when code protection is enabled.

Table 7-1 lists the identification information for each device. Table 7-2 shows the Device ID registers.

TABLE 7-1: DEVICE IDs AND REVISION

Device	DEVID Register Value	Application ID	DEVREV Register Value and Silicon Revision	JTAG ID
PIC24EP32GP202	0x1C19	0xDE	0x4000 – A0	Register 7-1
PIC24EP32GP203	0x1C1A	0xDE		
PIC24EP32GP204	0x1C18	0xDE		
dsPIC33EP32GP502	0x1C0D	0xDE		
dsPIC33EP32GP503	0x1C0E	0xDE		
dsPIC33EP32GP504	0x1C0C	0xDE		
PIC24EP32MC202	0x1C11	0xDE		
PIC24EP32MC203	0x1C12	0xDE		
PIC24EP32MC204	0x1C10	0xDE		
dsPIC33EP32MC202	0x1C01	0xDE		
dsPIC33EP32MC203	0x1C02	0xDE		
dsPIC33EP32MC204	0x1C00	0xDE		
dsPIC33EP32MC502	0x1C05	0xDE		
dsPIC33EP32MC503	0x1C06	0xDE		
dsPIC33EP32MC504	0x1C04	0xDE		
PIC24EP64GP202	0x1D39	0xDE	0x4002 – A2	Register 7-1
PIC24EP64GP203	0x1D3A	0xDE		
PIC24EP64GP204	0x1D38	0xDE		
PIC24EP64GP206	0x1D3B	0xDE		
dsPIC33EP64GP502	0x1D2D	0xDE		
dsPIC33EP64GP503	0x1D2E	0xDE		
dsPIC33EP64GP504	0x1D2C	0xDE		
dsPIC33EP64GP506	0x1D2F	0xDE		
PIC24EP64MC202	0x1D31	0xDE		
PIC24EP64MC203	0x1D32	0xDE		
PIC24EP64MC204	0x1D30	0xDE		
PIC24EP64MC206	0x1D33	0xDE		
dsPIC33EP64MC202	0x1D21	0xDE		
dsPIC33EP64MC203	0x1D22	0xDE		
dsPIC33EP64MC204	0x1D20	0xDE		
dsPIC33EP64MC206	0x1D23	0xDE		
dsPIC33EP64MC502	0x1D25	0xDE		
dsPIC33EP64MC503	0x1D26	0xDE		
dsPIC33EP64MC504	0x1D24	0xDE		
dsPIC33EP64MC506	0x1D27	0xDE		

TABLE 7-1: DEVICE IDs AND REVISION (CONTINUED)

Device	DEVID Register Value	Application ID	DEVREV Register Value and Silicon Revision	JTAG ID
PIC24EP128GP202	0x1E59	0xDE	0x4000 – A0	Register 7-1
PIC24EP128GP204	0x1E58	0xDE		
PIC24EP128GP206	0x1E5B	0xDE		
dsPIC33EP128GP502	0x1E4D	0xDE		
dsPIC33EP128GP504	0x1E4C	0xDE		
dsPIC33EP128GP506	0x1E4F	0xDE		
PIC24EP128MC202	0x1E51	0xDE		
PIC24EP128MC204	0x1E50	0xDE		
PIC24EP128MC206	0x1E53	0xDE		
dsPIC33EP128MC202	0x1E41	0xDE		
dsPIC33EP128MC204	0x1E40	0xDE		
dsPIC33EP128MC206	0x1E43	0xDE		
dsPIC33EP128MC502	0x1E45	0xDE		
dsPIC33EP128MC504	0x1E44	0xDE		
dsPIC33EP128MC506	0x1E47	0xDE		
PIC24EP256GP202	0x1F79	0xDE		
PIC24EP256GP204	0x1F78	0xDE		
PIC24EP256GP206	0x1F7B	0xDE		
dsPIC33EP256GP502	0x1F6D	0xDE		
dsPIC33EP256GP504	0x1F6C	0xDE		
dsPIC33EP256GP506	0x1F6F	0xDE		
PIC24EP256MC202	0x1F71	0xDE		
PIC24EP256MC204	0x1F70	0xDE		
PIC24EP256MC206	0x1F73	0xDE		
dsPIC33EP256MC202	0x1F61	0xDE		
dsPIC33EP256MC204	0x1F60	0xDE		
dsPIC33EP256MC206	0x1F63	0xDE		
dsPIC33EP256MC502	0x1F65	0xDE		
dsPIC33EP256MC504	0x1F64	0xDE		
dsPIC33EP256MC506	0x1F67	0xDE		
PIC24EP512GP202	0x1799	0xDE	0x4003 – A3	Register 7-1
PIC24EP512GP204	0x1798	0xDE		
PIC24EP512GP206	0x179B	0xDE		
dsPIC33EP512GP502	0x178D	0xDE		
dsPIC33EP512GP504	0x178C	0xDE		
dsPIC33EP512GP506	0x178F	0xDE		
PIC24EP512MC202	0x1791	0xDE		
PIC24EP512MC204	0x1790	0xDE		
PIC24EP512MC206	0x1793	0xDE		
dsPIC33EP512MC202	0x1781	0xDE		
dsPIC33EP512MC204	0x1780	0xDE		
dsPIC33EP512MC206	0x1783	0xDE		
dsPIC33EP512MC502	0x1785	0xDE		
sdPIC33EP512MC504	0x1784	0xDE		
dsPIC33EP512MC506	0x1787	0xDE		

TABLE 7-2: DEVICE ID REGISTERS

Address	Name	Bit															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF0000	DEVID	DEVID Value															
0xFF0002	DEVREV	DEVREV Value															

REGISTER 7-1: JTAG ID REGISTER

31 28	27 12	11 0
DEVREV<3:0>	DEVID<15:0>	Manufacturer ID (0x053)
4 bits	16 bits	12 bits

8.0 CHECKSUM COMPUTATION

Checksums for devices are 16 bits in size. The checksum is calculated by summing the following:

- Contents of code memory locations
- Contents of Configuration bytes

All memory locations, including configuration bytes, are summed by adding all three bytes of each memory address. For certain configuration bytes a read mask is used to ignore the bits which are reserved.

Table 8-1 is an example of the checksum calculation for the dsPIC33EP64MC506 device.

Table 8-2 describes the Configuration bit masks for each device.

TABLE 8-1: CHECKSUM COMPUTATION EXAMPLE

Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAAAAAA at 0x0 and Last Code Address
dsPIC33EP64MC506	Disabled	FLASH SUM(0:0x00AFEA) + CFGB SUM(0x00AFEC:0x00AFFE)	0xF748(1)	0xF54A(1)
	Enabled	Reads of program memory return 0x00	0x0000	0x0000

Item Description:
FLASH SUM(a:b) = Byte sum of locations a to b inclusive (all 3 bytes of code memory)
CFGB SUM(a:b) = Byte sum of locations a to b inclusive (all 3 bytes of configuration memory) using any read masks shown in Table 8-2.
CFGB SUM(0x00AFEC:0x00AFFE) = (29* 0xFF + (FICD & 0x67))

Note 1: For calculating this checksum, the default Reset value for all Configuration bits is used, except for JTAGEN, which is configured as '0' (disabled) to match Microchip's Development Tools default configuration. CFGB SUM = ((29 * 0xFF) + (0x47))

TABLE 8-2: CONFIGURATION BIT MASKS

Device	Configuration Bit Masks
	FICD
dsPIC33EPXXXGP50X	0x67
dsPIC33EPXXXMC20X	0x67
dsPIC33EPXXXMC50X	0x67
PIC24EPXXXGP20X	0x67
PIC24EPXXXMC20X	0x67

9.0 AC/DC CHARACTERISTICS AND TIMING REQUIREMENTS

Table 9-1 lists the AC/DC characteristics and timing requirements.

TABLE 9-1: AC/DC CHARACTERISTICS AND TIMING REQUIREMENTS

Standard Operating Conditions Operating Temperature: -40ºC to +85ºC. Programming at +25ºC is recommended.						
Param. No.	**Symbol**	**Characteristic**	**Min.**	**Max.**	**Units**	**Conditions**
D111	VDD	Supply Voltage During Programming	—	—	V	See **Note 1** and **Note 2**
D113	IDDP	Supply Current During Programming	—	—	mA	See **Note 2**
D114	IPEAK	Instantaneous peak current during startup	—	—	mA	See **Note 2**
D031	VIL	Input Low Voltage	—	—	V	See **Note 2**
D041	VIH	Input High Voltage	—	—	V	See **Note 2**
D080	VOL	Output Low Voltage	—	—	V	See **Note 2**
D090	VOH	Output High Voltage	—	—	V	See **Note 2**
D012	CIO	Capacitive Loading on I/O pin (PGEDx)	—	—	pF	See **Note 2**
P1	TPGC	Serial Clock (PGECx) Period (ICSP™)	200	—	ns	—
P1	TPGC	Serial Clock (PGECx) Period (Enhanced ICSP)	500	—	ns	—
P1A	TPGCL	Serial Clock (PGECx) Low Time (ICSP)	80	—	ns	—
P1A	TPGCL	Serial Clock (PGECx) Low Time (Enhanced ICSP)	200	—	ns	—
P1B	TPGCH	Serial Clock (PGECx) High Time (ICSP)	80	—	ns	—
P1B	TPGCH	Serial Clock (PGECx) High Time (Enhanced ICSP)	200	—	ns	—
P2	TSET1	Input Data Setup Time to Serial Clock ↓	15	—	ns	—
P3	THLD1	Input Data Hold Time from PGECx ↓	15	—	ns	—
P4	TDLY1	Delay between 4-bit Command and Command Operand	40	—	ns	—
P4A	TDLY1A	Delay between Command Operand and Next 4-bit Command	40	—	ns	—
P5	TDLY2	Delay between Last PGECx ↓of Command to First PGECx ↑ of Read of Data Word	20	—	ns	—
P6	TSET2	VDD ↑ Setup Time to $\overline{\text{MCLR}}$ ↑	100	—	ns	—
P7	THLD2	Input Data Hold Time from $\overline{\text{MCLR}}$ ↑	50	—	ms	—
P8	TDLY3	Delay between Last PGECx ↓of Command Byte to PGEDx ↑ by Programming Executive	12	—	µs	—
P9a	TDLY4	Programming Executive Command Processing Time	10	—	µs	—

Note 1: VDD must also be supplied to the AVDD pins during programming. AVDD and AVSS should always be within ±0.3V of VDD and VSS, respectively.

2: Refer to the **"Electrical Characteristics"** section in the specific device data sheet for the Minimum and Maximum values.

3: This time applies to Program Memory words, Configuration words, and User ID words.

TABLE 9-1: AC/DC CHARACTERISTICS AND TIMING REQUIREMENTS (CONTINUED)

Standard Operating Conditions Operating Temperature: -40ºC to +85ºC. Programming at +25ºC is recommended.						
Param. No.	**Symbol**	**Characteristic**	**Min.**	**Max.**	**Units**	**Conditions**
P9b	TDLY5	Delay between PGEDx ↓by Programming Executive to PGEDx Released by Programming Executive	15	23	µs	—
P10	TDLY6	PGECx Low Time After Programming	400	—	ns	—
P11	TDLY7	Bulk Erase Time	—	21	ms	—
P12	TDLY8	Page Erase Time	—	—	ms	See **Note 2**
P13	TDLY9	Double Word Programming Time	—	—	µs	See **Note 2** and **Note 3**
P14	TR	$\overline{\text{MCLR}}$ Rise Time to Enter ICSP mode	—	1.0	µs	—
P15	TVALID	Data Out Valid from PGECx ↑	10	—	ns	—
P16	TDLY10	Delay between Last PGECx ↓and $\overline{\text{MCLR}}$ ↓	0	—	s	—
P17	THLD3	$\overline{\text{MCLR}}$ ↓to VDD ↓	100	—	ns	—
P18	TKEY1	Delay from First $\overline{\text{MCLR}}$ ↓to First PGECx ↑ for Key Sequence on PGEDx	1	—	ms	—
P19	TKEY2	Delay from Last PGECx ↓for Key Sequence on PGEDx to Second $\overline{\text{MCLR}}$ ↑	25	—	ns	—
P21	TMCLRH	$\overline{\text{MCLR}}$ High Time	—	500	µs	—

Note 1: VDD must also be supplied to the AVDD pins during programming. AVDD and AVSS should always be within ±0.3V of VDD and VSS, respectively.

2: Refer to the **"Electrical Characteristics"** section in the specific device data sheet for the Minimum and Maximum values.

3: This time applies to Program Memory words, Configuration words, and User ID words.

NOTES:

APPENDIX A: HEX FILE FORMAT

Flash programmers process the standard HEX format used by the Microchip development tools. The format supported is the Intel® HEX32 Format (INHX32). For more information about hex file formats, please refer to **Section 1.7.5 "Hex File Formats (.hex, .hxl, .hxh)"** in the *"MPASM™ Assembler, MPLINK™ Object Linker, and MPLIB™ Object Librarian User's Guide"* (DS33014).

The basic format of the hex file is:

```
:BBAAAATTHHHH...HHHHCC
```

Each data record begins with a 9-character prefix and always ends with a 2-character checksum. All records begin with a colon ':', regardless of the format. The individual elements are described below.

- BB – is a two-digit hexadecimal byte count representing the number of data bytes that appear on the line. Divide this number by two to get the number of words per line.
- AAAA – is a four-digit hexadecimal address representing the starting address of the data record. Format is high byte first followed by low byte. The address is doubled because this format only supports 8 bits. Divide the value by two to find the real device address.
- TT – is a two-digit record type that will be '00' for data records, '01' for end-of-file records and '04' for extended-address record.
- HHHH – is a four-digit hexadecimal data word. Format is low byte followed by high byte. There will be BB/2 data words following TT.
- CC – is a two-digit hexadecimal checksum that is the two's complement of the sum of all the preceding bytes in the line record.

Because the Intel hex file format is byte-oriented, and the 16-bit program counter is not, program memory sections require special treatment. Each 24-bit program word is extended to 32 bits by inserting a so-called "phantom byte". Each program memory address is multiplied by 2 to yield a byte address.

As an example, a section that is located at 0x100 in program memory will be represented in the hex file as 0x200.

The hex file will be produced with the following contents:

```
:020000040000fa
:04020000003322110096
:00000001FF
```

Note that the data record (line 2) has a load address of 0200, while the source code specified address is 0x100. In addition, the data is represented in "little-endian" format, meaning the Least Significant Byte (LSB) appears first. The phantom byte appears last, before the checksum.

APPENDIX B: REVISION HISTORY

Revision A (May 2011)

This is the initial released version of this document.

Revision B (August 2011)

This revision includes the following updates:

- Updated the User Memory Address Limit values for Code Memory Size (see Table 2-2)
- Added a Reserved row for addresses 0057EC, 00AFEC, 0157EC, and 02AFFC to the Configuration Byte Register Map (see Table 2-3)
- Updated Step 4 in Serial Instruction Execution for Writing Code Memory (see Table 3-5)
- Updated the Hex PE file name (see the first note box in **Section 5.0 "Programming the Programming Executive to Memory"**)
- Updated Step 4 in Programming the Programming Executive (see Table 5-2)
- Added a cautionary note at the beginning of **Section 6.0 "The Programming Executive"**
- Updated **Section 6.2.4.4 "PROG2W Command"**
- Added Silicon Revision A2 (0x4002) for select devices in Device IDs and Revision (see Table 7-1)
- The following devices were removed from Device IDs and Revision (see Table 7-1):
  - PIC24EP128GP203
  - dsPIC33EP128GP503
  - PIC24EP128MC203
  - dsPIC33EP128MC203
  - dsPIC33EP128MC503
- Updated the Checksum Computation Example (see Table 8-1)
- Minor updates to text and formatting have been incorporated throughout the document

Revision C (December 2011)

This revision includes the following updates:

- New information on User ID Words was added as follows:
  - **FIGURE 2-2: "Program Memory Map"**
  - **Section 2.5 "User ID Words"**
  - **FIGURE 3-1: "High-Level ICSP™ Programming Flow"**
  - **TABLE 3-2: "NVMCON Erase Operations"**
  - **Register 3-1: "NVMCON: Non-Volatile Memory (NVM) Control Register"**
  - **3.8 "Writing User ID Words"**
  - **3.11 "Reading User ID Words"**
  - **FIGURE 4-1: "High-Level Enhanced ICSP™ Programming Flow"**

Revision C (December 2011) (Continued)

- Added the Configuration Bit Address Range (Bytes) column to the Code Memory Size table (see Table 2-2)
- Replaced the Configuration Byte Register Map table (see Table 2-3)
- Updated the Checksum Computation Example (see Table 8-1)
- Replaced the Configuration Bit Masks table (see Table 8-2)
- Minor updates to text and formatting have been incorporated throughout the document

Revision D (March 2012)

This revision includes the following updates:

- Added the following devices to the Code Memory Size table (see Table 2-2):
  - dsPIC33EP512GP50X
  - dsPIC33EP512MC50X
  - dsPIC33EP512MC20X
  - PIC24EP512MC20X
  - PIC24EP512GP20X
- Added the following devices to the Device IDs and Revision table (see Table 7-1):
  - PIC24EP512GP202
  - PIC24EP512GP204
  - PIC24EP512GP206
  - dsPIC33EP512GP502
  - dsPIC33EP512GP504
  - dsPIC33EP512GP506
  - PIC24EP512MC202
  - PIC24EP512MC204
  - PIC24EP512MC206
  - dsPIC33EP512MC202
  - dsPIC33EP512MC204
  - dsPIC33EP512MC206
  - dsPIC33EP512MC502
  - sdPIC33EP512MC504
  - dsPIC33EP512MC506

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.

Trademarks

ISBN: 978-1-62076-142-7

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.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Hangzhou
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Osaka
Tel: 81-66-152-7160
Fax: 81-66-152-9310

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-330-9305

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

11/29/11