dsPIC33CDVL64MC106 Family Flash Programming Specification

1.0 DEVICE OVERVIEW

This document defines the programming specification for the dsPIC33CDVL64MC106 Digital Signal Controller (DSC) device. This programming specification is required only for those developing programming support for the dsPIC33CDVL64MC106 device.

Customers only using this device for application development should use development tools that already provide support for device programming.

Topics covered include:

- **Section 1.0 “Device Overview”**
- **Section 2.0 “Programming Overview”**
- **Section 3.0 “Device Programming – ICSP”**
- **Section 4.0 “Device Programming – Enhanced ICSP”**
- **Section 5.0 “The Programming Executive”**
- **Section 6.0 “Device ID/Unique ID”**
- **Section 7.0 “CRC Checksum Computation”**
- **Section 8.0 “AC/DC Characteristics and Timing Requirements”**

2.0 PROGRAMMING OVERVIEW

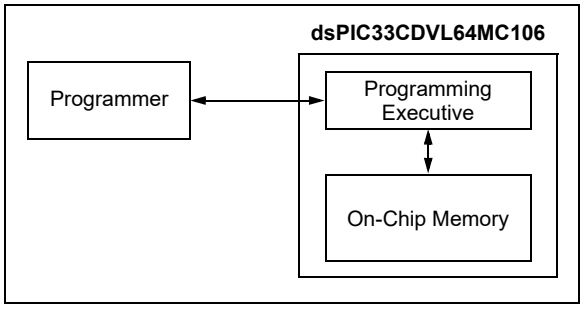
The following are the two methods of programming that are discussed in this programming specification:

- In-Circuit Serial Programming™ (ICSP™)
- 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 (PE), as illustrated in Figure 2-1. The PE 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 dsPIC33CDVL64MC106 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

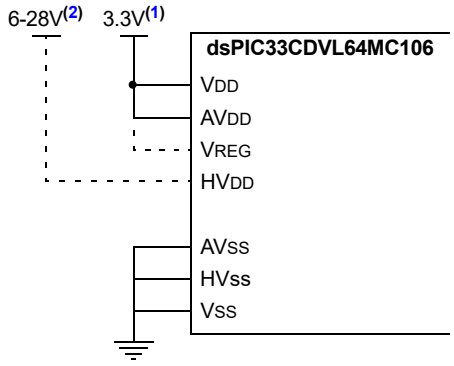
This device requires specific connections for programming to take place. These connections include power, MCLR and one programming pair (PGEDx/PGECx). Table 2-1 describes these connections (refer to the specific device data sheet for pin descriptions and power connection requirements).

2.2 Power Requirements

The dsPIC33CDVL64MC106 device powers its core digital logic at a nominal 1.2V and incorporates a capacitor-free, on-chip regulator that allows the device to run its core logic from VDD. The regulator provides power to the core from the other VDD pins and does not require an external CPU Logic Filter Capacitor (VCAP) connection. The dsPIC33CDVL64MC106 device also incorporates an on-chip 3.3V regulator. This regulator outputs 3.3V on the VREG pin when 6-28V are supplied to the HVDD pin.

VDD pins may be powered by either an external power supply or the VREG output pin. If an external power supply is used to power the VDD pins directly, the HVDD pin does not need to be powered to program the dsPIC33CDVL64MC106 device. The specifications for core voltage and capacitance are listed in **Section 8.0 "AC/DC Characteristics and Timing Requirements"**.

FIGURE 2-2: CONNECTIONS FOR THE ON-CHIP REGULATOR



Note 1: VDD/AVDD pins may be powered by either an external power supply or by the 3.3V VREG output. The 3.3V VREG output may be left disconnected while programming if the VDD/AVDD pins are being supplied by an external power supply.

2: HVDD is only required for programming if the 3.3V VREG output is being used to supply the VDD/AVDD pins. The HVDD pin may be left disconnected if the 3.3V VREG output is not being used.

TABLE 2-1: PINS USED DURING PROGRAMMING

Pin Name	Pin Type	Pin Description
MCLR	I	Programming Enable
HVDD	P	High-Voltage Power Supply(1)
VREG	P	3.3V Regulator Output
VDD and AVDD(1)	P	Power Supply(2)
VSS and AVSS(1)	P	Ground(2)
PGECx	I	Programming Pin Pair: Serial Clock
PGEDx	I/O	Programming Pin Pair: Serial Data

Legend: I = Input O = Output P = Power

Note 1: HVDD and VREG may be left disconnected during programming if the 3.3 VREG output is not being used to supply the VDD and AVDD pins.

2: All VDD, VSS, AVDD and AVSS power supply and ground pins must be connected during programming.

2.3 Pin Diagrams

Figure 2-3 provides the pin diagram for the dsPIC33CDVL64MC106 device. The pins that are required for programming are listed in Table 2-1 and are indicated in bold text in the figures. Refer to the specific device data sheet for complete pin descriptions.

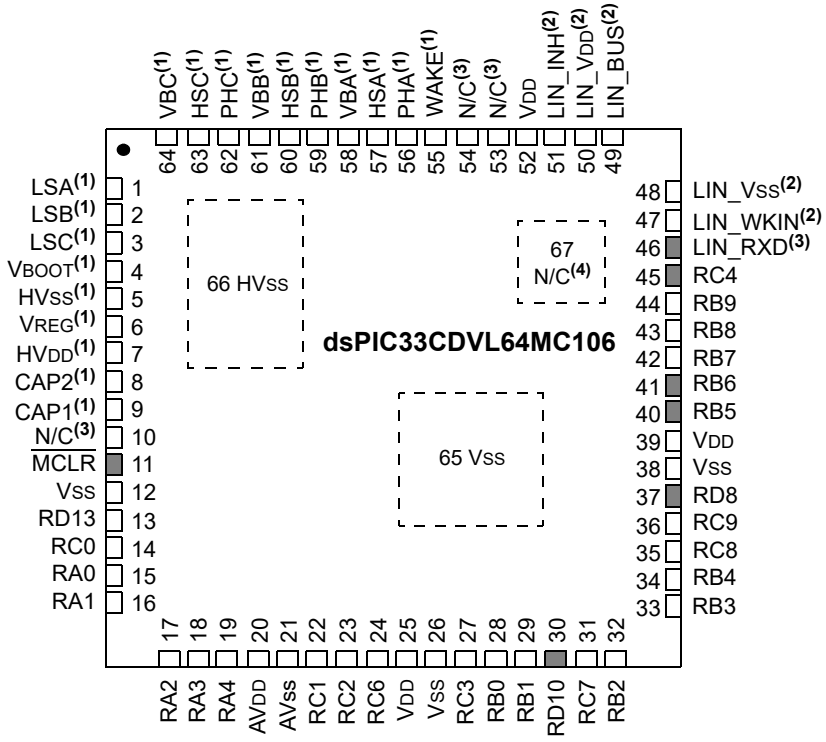
2.3.1 PGECx AND PGEDx PIN PAIRS

The dsPIC33CDVL64MC106 device has three separate pairs of programming pins, labeled as PGEC1/PGED1, PGEC2/PGED2 and PGEC3/PGED3. Any one of these pin pairs may be used for device programming by either ICSP or Enhanced ICSP. Unlike voltage supply and ground pins, it is not necessary to connect all three pin pairs to program the device. However, the programming method must use both pins of the same pair.

FIGURE 2-3: PIN DIAGRAMS

64-Pin QFN

= 5.5 VDC Tolerant



Note 1: These pins are specific to the MOSFET Gate Driver module.

2: These pins are specific to the LIN transceiver module.

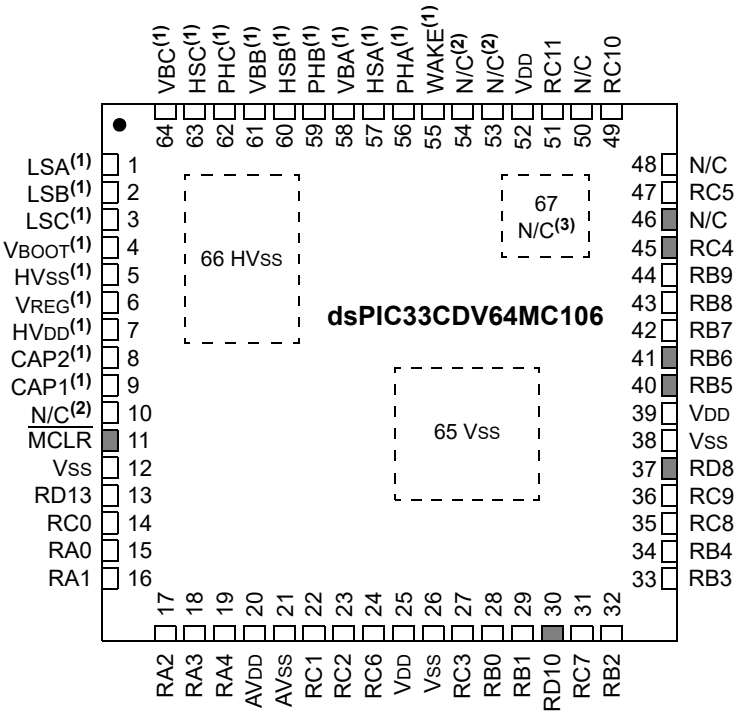
3: Pin is connected to a device interconnect.

4: Pin has no electrical connection to the device, but is recommended to be connected to VSS/LIN_VSS in the PCB design to improve thermal dissipation.

FIGURE 2-4: PIN DIAGRAMS (CONTINUED)

64-Pin QFN

■ = 5.5 VDC Tolerant



Note 1: These pins are specific to the MOSFET gate driver module.

2: Pin is connected to a device interconnect.

3: Pin has no electrical connection to the device, but is recommended to be connected to VSS in the PCB design to improve thermal dissipation.

2.4 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 are used in both program operations or (b) bits containing '1' are set to '0', but no '0' is set to '1'.

2.5 Memory Map

The program memory map extends from 0x000000 to 0xFFFFFE. User program code storage is located at the base of the memory map. The last row (128 instruction words) of the last page of implemented program memory is reserved for the device Configuration bits. Caution should be used when placing code in the last page of user memory. When programming the device configuration, the whole last page must first be erased, including any code located in the last page.

Table 2-3 lists the user memory address limit and available number of instruction words (including the device configuration area), the number of write blocks and the number of erase blocks present in each device variant.

Locations, 0x800000 through 0x800BFE, are reserved for executive code memory. This region stores the PE and the debugging executive, which is used for device programming. This region of memory cannot be used to store user code. See **Section 5.0 "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 6.0 "Device ID/Unique ID"**. The Device ID registers read out normally, even after code protection is applied.

The locations, 0x801700 to 0x8017FE, are a One-Time-Programmable (OTP) memory area. The user OTP 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 2.7 "User One-Time-Programmable (OTP) Memory"**.

Figure 2-5 through Figure 2-6 show generic memory maps for the device listed in Table 2-3. See the **"Memory Organization"** chapter in the specific device data sheet for exact memory addresses.

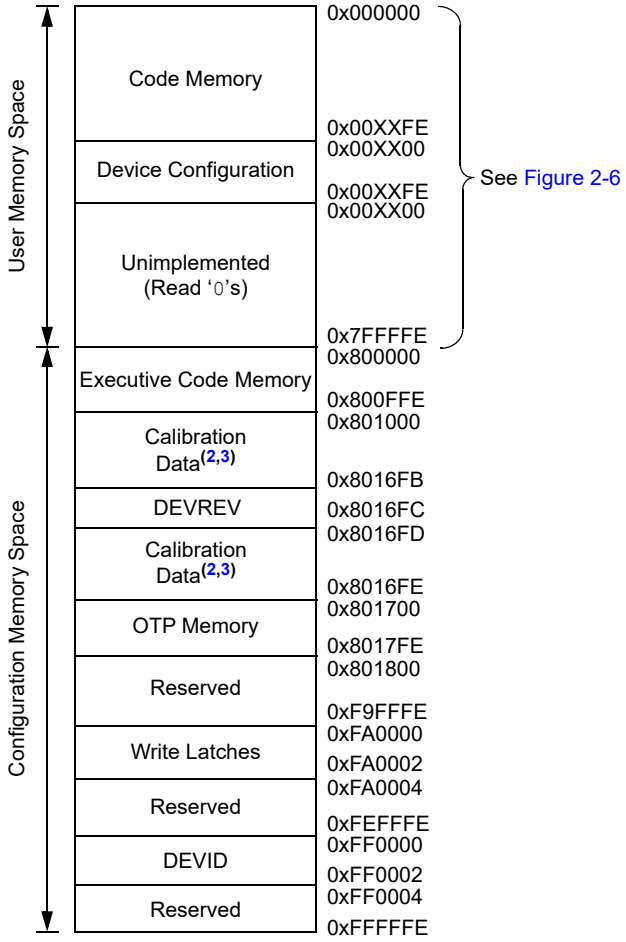
TABLE 2-2: DEVICE ROW AND PAGE SIZE

Instruction Words (24 bits)	
Row	128
Page	1024

TABLE 2-3: FLASH CODE MEMORY SIZE

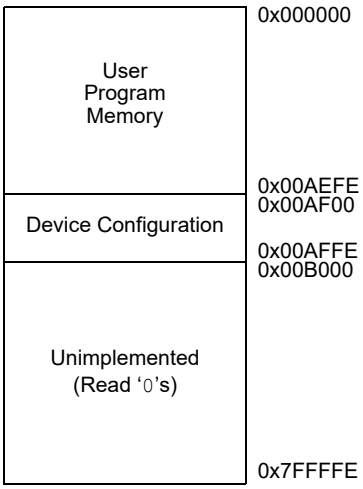
Device Family	User Memory Limit (Instruction Words)	Write Blocks/ No. of Rows	Erase Blocks/ No. of Pages
dsPIC33CDVL64MC106	0x00AFFE (22528)	176	22
dsPIC33CDV64MC106	0x00AFFE (22528)	176	22

FIGURE 2-5: PROGRAM MEMORY MAP FOR dsPIC33CDVL64MC106 DEVICE(1)



Note 1: Memory areas are not shown to scale.
2: Calibration data area must be maintained during programming.
3: Calibration data area includes UDID register location.

FIGURE 2-6: PROGRAM MEMORY MAP FOR dsPIC33CDVL64MC106 DEVICE(1)



Note 1: Memory areas are not shown to scale.

2.6 Configuration Bits

2.6.1 OVERVIEW

The Configuration bits are stored in the last row of the last page location 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 2-4 list the Configuration register address for each device memory size and partition configuration. Table 2-5 shows the Configuration register map. Refer to the **"Special Features"** chapter in the specific device data sheet for the full Configuration Word register descriptions for your device.

Special attention should be taken for the reserved Configuration bits. The FSIGN[15], FDEVOPT[8] and FDEVOPT[9] Configuration bits must always be programmed ('0'). The FPOR[5:4], FICD[7], FDEVOPT[7] and FDEVOPT[10] Configuration bits must always stay unprogrammed ('1').

TABLE 2-4: dsPIC33CDVL64MC106 CONFIGURATION ADDRESSES

Register Name	64k	32k
FSEC	0x00AF00	0x005F00
FBSLIM	0x00AF10	0x005F10
FSIGN	0x00AF14	0x005F14
FOSCSEL	0x00AF18	0x005F18
FOSC	0x00AF1C	0x005F1C
FWDT	0x00AF20	0x005F20
FPOR	0x00AF24	0x005F24
FICD	0x00AF28	0x005F28
FDMTIVTL	0x00AF2C	0x005F2C
FDMTIVTH	0x00AF30	0x005F30
FDMTCNTL	0x00AF34	0x005F34
FDMTCNTH	0x00AF38	0x005F38
FDMT	0x00AF3C	0x005F3C
FDEVOPT	0x00AF40	0x005F40
FALTREG	0x00AF44	0x005F44

TABLE 2-5: CONFIGURATION REGISTERS MAP

Register Name	Bits 23-16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FSEC	—	AIVTDIS	—	—	—	CSS[2:0]			CWRP	GSS[1:0]		GWRP	—	BSEN	BSS[1:0]		BWRP
FBSLIM	—	—	—	—	BSLIM[12:0]												
FSIGN	—	r(2)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
FOSCSEL	—	—	—	—	—	—	—	—	—	IESO	—	—	—	—	FNOSC[2:0]		
FOSC	—	—	—	—	XTBST	XTALCFG[1:0]		—	PLLKEN	FCKSM[1:0]		—	—	—	OSCIOFCN	POSCMD[1:0]	
FWDT	—	FWDTEN	SMWDT[4:0]					WDTWIN[1:0]		WINDIS	WDTCLKS[1:0]		RMWDT[4:0]				
FPOR	—	—	—	—	—	—	r(1)	—	—	—	BISTDIS	r(1)	r(1)	—	—	—	—
FICD	—	NOBTSWP	—	—	—	—	—	—	—	r(1)	—	JTAGEN	—	—	—	ICS[1:0]	
FDMTIVTL	—	DMTIVT[15:0]															
FDMTIVTH	—	DMTIVT[31:16]															
FDMTCNTL	—	DMTCNT[15:0]															
FDMTCNTH	—	DMTCNT[31:16]															
FDMT	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DMTDIS
FDEVOPT	—	—	—	SPI2PIN	—	—	SMB3EN	r(2)	r(2)	r(1)	—	—	—	ALTI2C1	r(1)	—	—
FALTREG	—	—	CTXT4[2:0]			—	CTXT3[2:0]			—	CTXT2[2:0]			—	CTXT1[2:0]		

Legend: — = unimplemented bit, read as '1'; r = reserved bit.

Note 1: Bit reserved, maintain as '1'.

2: Bit reserved, maintain as '0'.

2.6.2 CODE-PROTECT CONFIGURATION BITS

The device implements intermediate security features defined by the FSEC register. The Boot Segment (BS) is the highest privileged segment and the General Segment (GS) is the lowest privileged segment. The total user code memory can be split into BS or GS. The size of the segments is determined by the BSLIM[12:0] bits. The relative location of the segments within user space does not change, such that BS (if present) occupies the memory area just after the Vector Space (VS), Interrupt Vector Table (IVT), and the GS occupies the space just after BS (or if the Alternate Interrupt Vector Table (AIVT) is enabled, just after AIVT VS). The Configuration Segment (CS) is a small segment (less than a page, typically just one row) within user Flash address space that contains all user configuration data that are loaded by the NVM controller during the Reset sequence.

2.7 User One-Time-Programmable (OTP) Memory

The dsPIC33CDVL64MC106 device provides 128 words (64 pairs) of One-Time-Programmable (OTP) memory, located at addresses, 0x801700 through 0x8017FE. This memory can be used for persistent storage of application-specific information that will not be erased by reprogramming the device. This includes many types of information, such as:

- Application checksums
- Code revision information
- Product information
- Serial numbers
- System manufacturing dates
- Manufacturing lot numbers

Customer OTP memory may be programmed in any mode, including user RTSP mode, but it cannot be erased. Data are not cleared by a Chip Erase. See **Section 3.9 “Writing OTP Words”** for more information regarding OTP programming.

2.8 ICSP Write Inhibit

ICSP Write Inhibit is an access restriction feature that restricts all of Flash memory when activated. Once activated, ICSP Write Inhibit permanently prevents ICSP Flash programming and erase operations, and cannot be deactivated. This feature is intended to prevent alteration of Flash memory contents with behavior similar to One-Time-Programmable (OTP) devices.

RTSP, including erase and programming operations, is not restricted when ICSP Write Inhibit is activated; however, code to perform these actions must be programmed into the device before ICSP Write Inhibit is activated. This allows for a bootloader-type application to alter Flash contents when ICSP Write Inhibit is activated.

Entry into ICSP and Enhanced ICSP modes is not affected by ICSP Write Inhibit. In these modes, it will continue to be possible to read configuration memory space and any user memory space regions, which are not code-protected. With ICSP Write Inhibit, an attempt to set WR (NVMCON[15]) = 1 will maintain WR = 0, and instead, set WRERR (NVMCON[13]) = 1. All Enhanced ICSP erase and programming commands will have no effect, with self-checked programming commands returning a FAIL response opcode (or a PASS if the destination already exactly matched the requested programming data).

Once ICSP Write Inhibit is activated, it is not possible for a device executing in Debug mode to erase/write Flash, nor can a debug tool switch the device to Production mode. ICSP Write Inhibit should therefore only be activated on devices programmed for production.

The JTAG port, when enabled, can be used to map ICSP signals to JTAG I/O pins. All Flash erase/programming operations, initiated via the JTAG port, will therefore also be blocked after activating ICSP Write Inhibit

2.8.1 ACTIVATING ICSP WRITE INHIBIT

Note:	It is not possible to deactivate ICSP Write Inhibit.

ICSP Write Inhibit is activated by executing a pair of NVMCON Double-Word Programming commands to save two 16-bit activation values in the configuration memory space. The target NVM addresses and values required for activation are shown in Table 2-6.

TABLE 2-6: ICSP™ WRITE INHIBIT FUSE ADDRESSES AND CODES

ICSP Write Inhibit Fuse Address	Code Value
0x801028	0x006D63
0x80102C	0x006870

Once both addresses contain their activation values, ICSP Write Inhibit will take permanent effect on the next device Reset. Neither address can be reset, erased or otherwise modified, through any means, after being successfully programmed, even if one of the addresses has not been programmed.

Only the lower 16 data bits stored at the activation addresses are evaluated; the upper 8 bits and second 24-bit word, written by the Double-Word Programming NVMOPx bits, should be ‘0’s. The addresses can be programmed in any order and during separate ICSP/Enhanced ICSP/RTSP sessions, but any attempt to program an incorrect 16-bit value, or use a Row Programming operation to program the values, will be aborted without altering the existing data.

3.0 DEVICE PROGRAMMING – ICSP

ICSP mode is a special programming protocol that allows you to read and write to the device memory of the dsPIC33CDVL64MC106 device. 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 the contents of the executive memory to determine if the Programming Executive is present, and to write the Programming Executive to executive memory if it is missing, and then, Enhanced ICSP mode will be used.

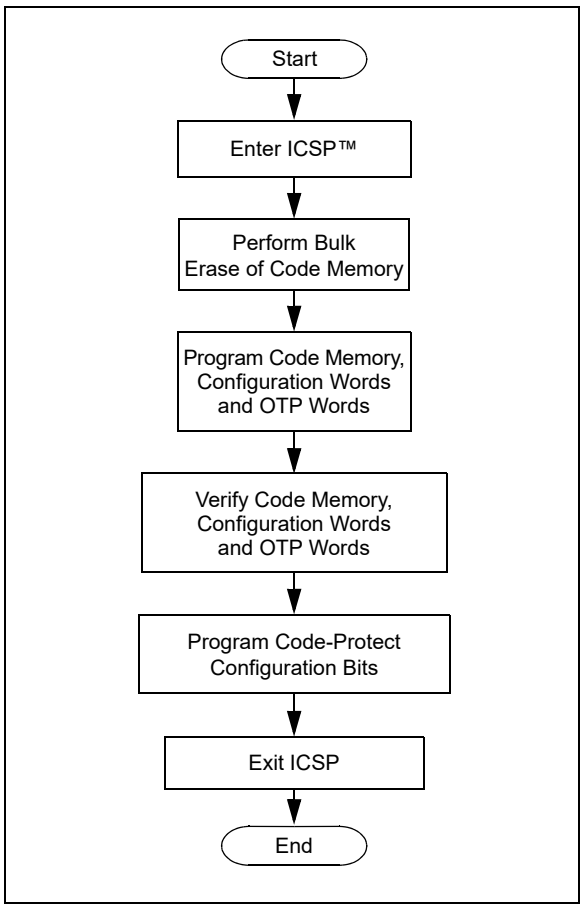
In ICSP mode, the system clock is taken from the PGECx pin, regardless of the device's Oscillator Configuration bits. All instructions are shifted serially into an internal buffer, then loaded into the Instruction Register (IR) and executed. No program fetching occurs from internal memory. Instructions 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 shows a high-level overview of the ICSP programming process. After entering ICSP mode, the first action is to Bulk Erase the code 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, programming the code-protect Configuration bits can be done if required.

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



3.2 Entering ICSP Mode

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

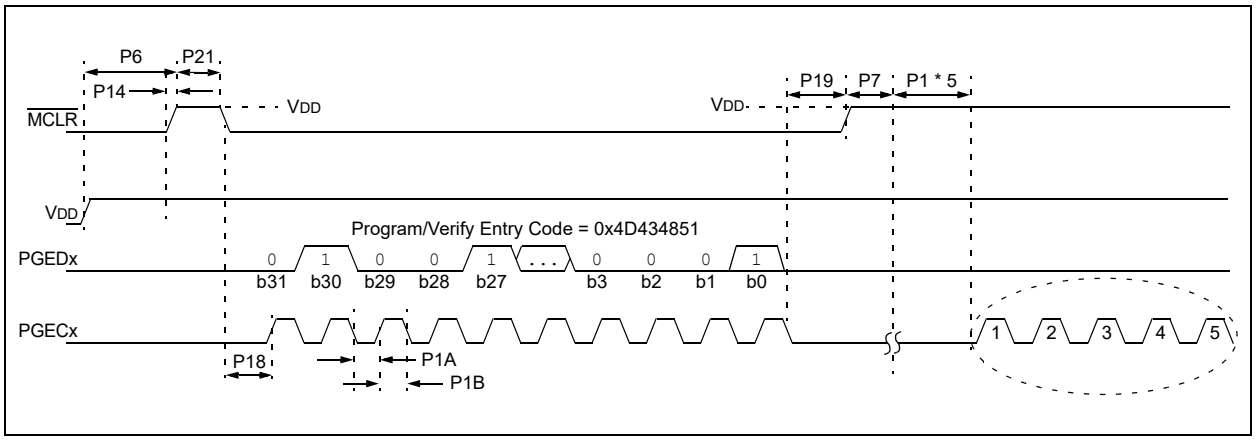
1. $\overline{\text{MCLR}}$ is briefly driven high, then low (P21).
2. A 32-bit key sequence is clocked into PGEDx. An interval of at least P18 must elapse before presenting the key sequence on PGEDx.
3. $\overline{\text{MCLR}}$ is held low during a specified period, P19, and then driven high.
4. After a P7 + 5 * P1 delay, five clock pulses must be generated on the PGECx pin.

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

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 ICSP mode only if the sequence is valid. The Most Significant bit (MSb) of the most significant nibble must be shifted in first.

On successful entry, the program memory can be accessed and programmed in serial fashion.

FIGURE 3-2: ENTERING ICSP™ MODE



3.3 ICSP Operation

Upon entry into ICSP mode, the CPU is Idle. Execution of the CPU is governed by an internal state machine. A 4-bit control code must be 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 through the VISI register.

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

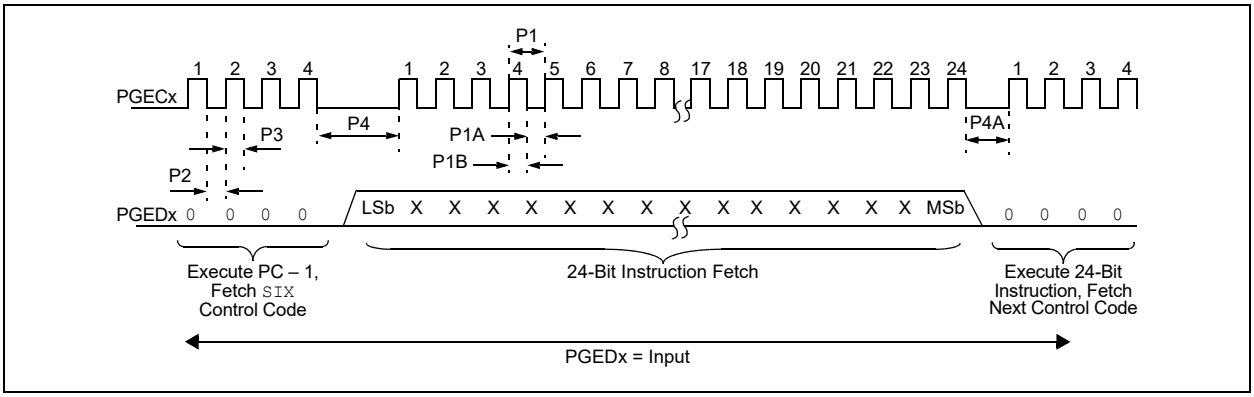
4-Bit Control Code	Mnemonic	Description
0000	SIX	Shift in 24-bit instruction and execute.
0001	REGOUT	Shift out the VISI register.
0010-1111	N/A	Reserved.

3.3.1 SIX SERIAL INSTRUCTION EXECUTION

The SIX control code allows execution of the dsPIC33 family 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 PGECx clock cycles. While the received instruction is executed, the state machine simultaneously shifts in the next 4-bit command (see Figure 3-3).

Note: Data bits on PGEDx are latched on the rising edge of the PGECx clock.

FIGURE 3-3: SIX SERIAL EXECUTION



3.3.1.1 Differences Between Execution of SIX and a Normal Instruction

There are some differences between executing instructions normally and using the ICSP SIX command. As a result, the code examples in this specification may not match those for performing the same functions during normal device operation.

The important differences are:

- Two-word instructions require two SIX operations to clock in all of the necessary data.

  Examples of two-word instructions are GOTO and CALL.
- Two-cycle instructions require two SIX operations to complete.

  The first SIX operation shifts in the instruction and begins to execute it. A second SIX operation, which should shift in a NOP to avoid losing data, provides the CPU clocks required to finish executing the instruction.

  Examples of two-cycle instructions are Table Read (TBLRD) and Table Write (TBLWT) instructions.
- Must provide NOP instruction during Stall to account for pipeline changes.

  A CPU Stall occurs when an instruction modifies a register that is used for Indirect Addressing by the instruction immediately following the CPU Stall. During normal operation, the CPU will automatically force a NOP while the new data are read. While using ICSP, the CPU stalls under the same conditions, but an instruction needs to be provided to generate the clocks to get through the Stall cycle. Therefore, any indirect references to a recently modified register should be preceded with a NOP.

  For example, the instructions, MOV #0x0, W0, followed by, MOV[W0], W1, must have a NOP inserted in between.

  If a two-cycle instruction modifies a register which is used indirectly, it will require two following NOPs: one to execute the second half of the instruction and the other NOP stalls the CPU to correct the pipeline.

  For example, instructions such as, TBLWTL [W0++], [W1], should be followed by two NOPs.
- The device Program Counter (PC) continues to automatically increment during ICSP instruction execution, even though the Flash memory is not being used. As a result, the PC may be incremented so that it points to invalid memory locations.

  Examples of invalid memory spaces are unimplemented Flash addresses or the vector space (location: 0x0 to 0x1FF).

  If the PC points to these locations, the device will reset, possibly interrupting the ICSP operation. To prevent this, instructions should be periodically executed to reset the PC to a safe space. The optimal method of achieving this is to perform a "GOTO 0x200" instruction.

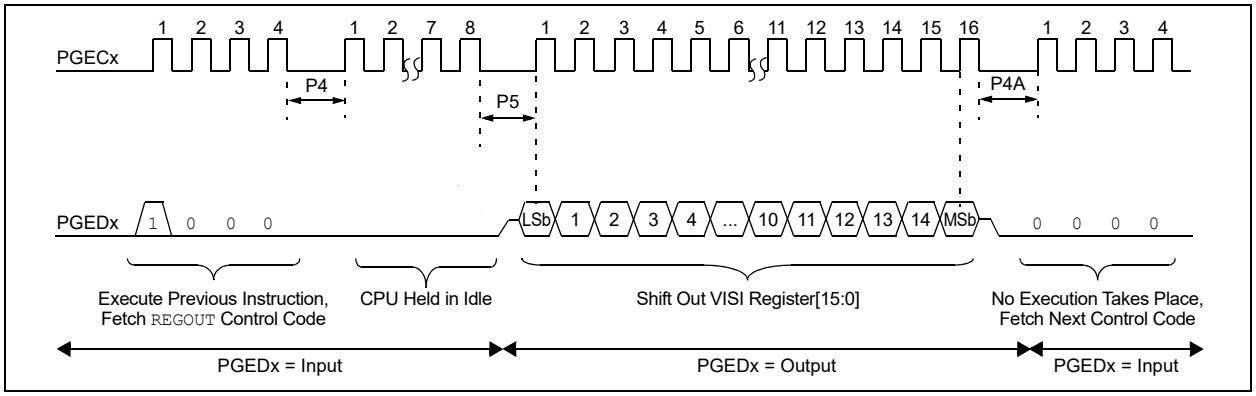
3.3.2 REGOUT SERIAL INSTRUCTION EXECUTION

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 3-4).

The REGOUT code is unique as 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 1:	After the contents of VISI are shifted out, the dsPIC33CDVL64MC106 device maintains PGEDx as an output until the first rising edge of the next clock is received.
2:	Data change on the falling edge and latch on the rising edge of PGECx. For all data transmissions, the Least Significant bit (LSb) is transmitted first.

FIGURE 3-4: 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]).

The PGECx clock is required to complete the programming operation. The WR control bit is cleared by hardware when the operation is finished. Refer to **Section 8.0 "AC/DC Characteristics and Timing Requirements"** for detailed information about the maximum time required for various programming operations.

TABLE 3-2: NVMCON ERASE OPERATIONS

NVMCON Value	Erase Operation
0x400E	Bulk Erase of user memory only (does not erase Device ID, Programming Executive memory and OTP Words).
0x4003	Page Erase of program or Programming Executive memory.

TABLE 3-3: NVMCON WRITE OPERATIONS

NVMCON Value	Write Operation
0x4001	Double-Word Programming operation.
0x4003	Row Programming operation.

3.4.2 STARTING AND STOPPING A PROGRAMMING CYCLE

For protection against accidental operations, the erase/write initiation 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 `NOP`s. To start an 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.

The WR bit can be polled to generate enough clock cycles for the programming operation and to determine if the erase or write cycle has been completed.

REGISTER 3-1: NVMCON: NONVOLATILE MEMORY (NVM) CONTROL REGISTER

R/SO-0[1]	R/W-0[1]	R/W-0[1]	R/W-0	U-0	U-0	R/W-0	R/C-0
WR	WREN	WRERR	NVMSIDL[2]	—	—	RPDF[6]	URERR[6]
bit 15							bit 8

U-0	U-0	U-0	U-0	R/W-0[1]	R/W-0[1]	R/W-0[1]	R/W-0[1]
—	—	—	—	NVMOP[3:0][3,4]			
bit 7							bit 0

Legend:	C = Clearable bit	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 = Enables Flash program/erase operations
- 0 = Inhibits Flash program/erase operations

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

bit 12 **NVMSIDL:** NVM Stop in Idle Control bit[2]
- 1 = Flash voltage regulator goes into Standby mode during Idle mode
- 0 = Flash voltage regulator is active during Idle mode

bit 11-10 **Unimplemented:** Read as '0'

bit 9 **RPDF:** Row Programming Data Format Control bit[6]
- 1 = Row data to be stored in RAM are in compressed format
- 0 = Row data to be stored in RAM are in uncompressed format

bit 8 **URERR:** Row Programming Data Underrun Error Flag bit[6]
- 1 = Row Programming operation has been terminated due to data underrun error
- 0 = No data underrun error has occurred

bit 7-4 **Unimplemented:** Read as '0'

Note 1: These bits can only be reset on a POR.

2: If this bit is set, there will be minimal power savings (IIDLE), and upon exiting Idle mode, there is a delay (TVREG) 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 on a 4-word boundary are programmed during execution of this operation.

6: Not used in ICSP™ mode.

REGISTER 3-1: NVMCON: NONVOLATILE MEMORY (NVM) CONTROL REGISTER (CONTINUED)

bit 3-0 **NVMOP[3:0]:** NVM Operation Select bits(1,3,4)

1111 = Reserved
1110 = User memory Bulk Erase operation
1010 = Reserved
1001 = Reserved
1000 = Reserved
0101 = Reserved
0100 = Inactive Partition memory erase operation
0011 = Memory Page Erase operation
0010 = Memory Row Program operation(6)
0001 = Memory Double-Word Program operation(5)
0000 = Reserved

Note 1: These bits can only be reset on a POR.

2: If this bit is set, there will be minimal power savings (IIDLE), and upon exiting Idle mode, there is a delay (TVREG) 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 on a 4-word boundary are programmed during execution of this operation.

6: Not used in ICSP™ mode.

3.5 Erasing Program Memory

Figure 3-5 shows a high-level overview for the Bulk Erase of code memory.

Table 3-4 provides the ICSP programming process for erasing the program memory.

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

FIGURE 3-5: BULK ERASE FLOW

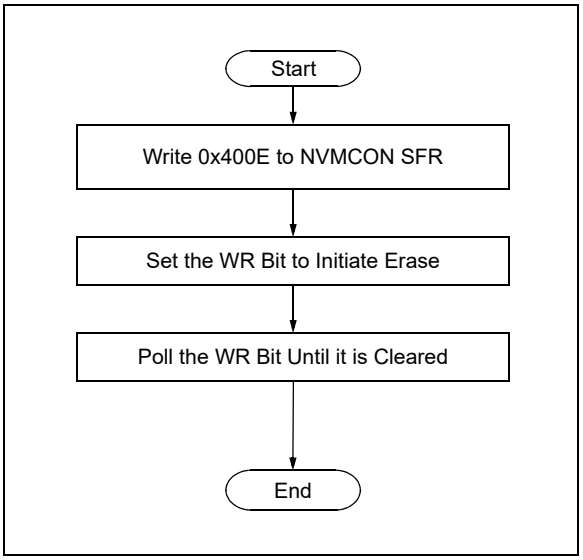


TABLE 3-4: SERIAL INSTRUCTION EXECUTION FOR BULK ERASE OF CODE MEMORY

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 register to erase all user program memory.		
0000	2400EA	MOV #0x400E, W10
0000	88468A	MOV W10, NVMCON
0000	000000	NOP
0000	000000	NOP
Step 3: Initiate the erase cycle.		
0000	200551	MOV #0x55, W1
0000	8846B1	MOV W1, NVMKEY
0000	200AA1	MOV #0xAA, W1
0000	8846B1	MOV W1, NVMKEY
0000	A8E8D1	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP

TABLE 3-4: SERIAL INSTRUCTION EXECUTION FOR BULK ERASE OF CODE MEMORY (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 4: Generate clock pulses for the code memory Bulk Erase operation to complete until the WR bit is clear.		
0000	000000	NOP
0000	804680	MOV NVMCON, W0
0000	000000	NOP
0000	887E60	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the 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.

3.6 Page Erase

Figure 3-6 shows a high-level overview for erasing a page of code memory.

Table 3-5 provides the ICSP programming details for erasing a page of code memory.

> **Note:** For Page Erase operations, the NVMCON value must be modified as per Table 3-2. The NVMADR/U registers must point to any of the locations of the page to be erased.

FIGURE 3-6: PAGE ERASE FLOW

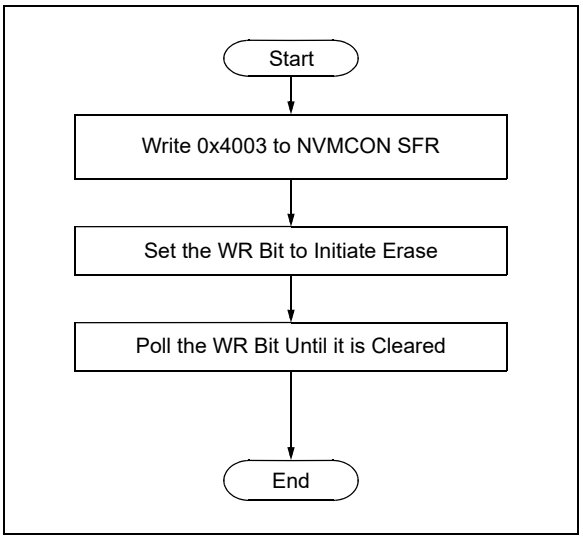


TABLE 3-5: SERIAL INSTRUCTION EXECUTION FOR ERASING A PAGE OF CODE MEMORY

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 NVMADRU/NVMADR register pair to point to the correct page to be erased.		
0000	2xxxx3	MOV #DestinationAddress<15:0>, W3
0000	2xxxx4	MOV #DestinationAddress<23:16>, W4
0000	884693	MOV W3, NVMADR
0000	8846A4	MOV W4, NVMADRU
Step 3: Set the NVMCON register to erase the first page of executive memory.		
0000	24003A	MOV #0x4003, W10
0000	88468A	MOV W10, NVMCON
0000	000000	NOP
0000	000000	NOP

TABLE 3-5: SERIAL INSTRUCTION EXECUTION FOR ERASING A PAGE OF CODE MEMORY (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 4: Initiate the erase cycle.		
0000	200551	MOV #0x55, W1
0000	8846B1	MOV W1, NVMKEY
0000	200AA1	MOV #0xAA, W1
0000	8846B1	MOV W1, NVMKEY
0000	A8E8D1	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 5: Generate clock pulses for the Page Erase operation to complete until the WR bit is clear.		
0000	000000	NOP
0000	804680	MOV NVMCON, W0
0000	000000	NOP
0000	887E60	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the 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.

3.7 Writing Code Memory

Figure 3-8 shows a high-level overview for writing the code memory.

Table 3-6 provides the ICSP programming details for writing the code memory.

Code memory is written two instruction words at a time. Two words are loaded into the write latches, located at 0xFA0000 and 0xFA0002, using the packed data format shown in Figure 3-7. The destination address is loaded into the NVMADR and NVMADRU registers. Next, the write cycle is initiated by setting the WREN bit in the NVMCON register. The WR bit in NVMCON will be cleared in hardware once the double-word write is complete. This process is repeated for all memory locations to be programmed.

FIGURE 3-7: PACKED INSTRUCTION WORD FORMAT

15	8	7	0
LSW1			
MSB2		MSB1	
LSW2			

LSWx: Least Significant 16 bits of instruction word
MSBx: Most Significant Byte of instruction word

FIGURE 3-8: PROGRAM CODE MEMORY FLOW

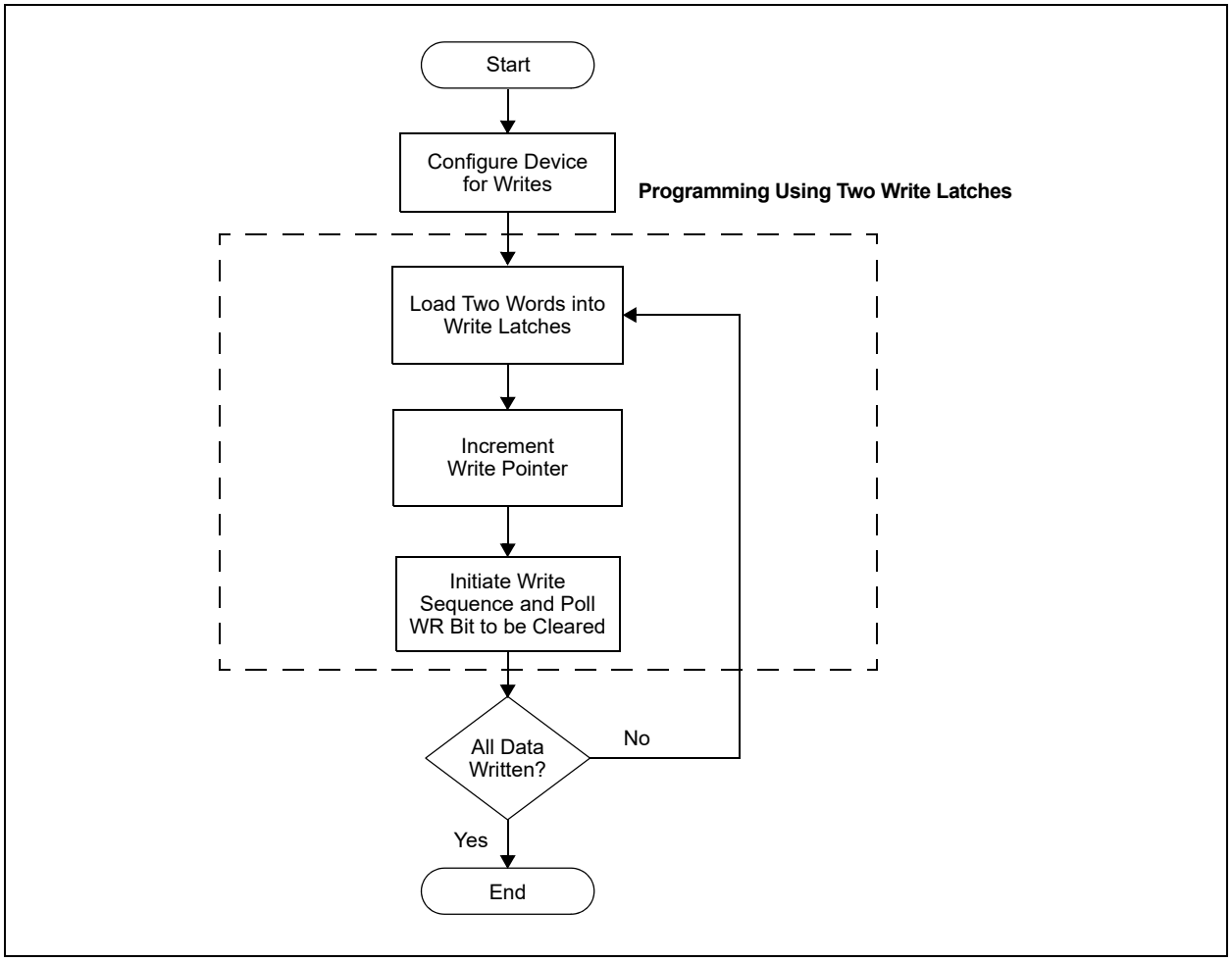


TABLE 3-6: SERIAL INSTRUCTION EXECUTION FOR PROGRAMMING CODE MEMORY: TWO-WORD LATCH WRITES

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 the latches.		
0000	200FAC	MOV #0xFA, W12
0000	8802AC	MOV W12, TBLPAG
Step 3: Load W0:W2 with the next two packed instruction words to program.		
0000	2xxxx0	MOV #<LSW0>, W0
0000	2xxxx1	MOV #<MSB1:MSB0>, W1
0000	2xxxx2	MOV #<LSW1>, W2
Step 4: Set the Read Pointer (W6) and Write 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	BB0B96	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	884693	MOV W3, NVMADR
0000	8846A4	MOV W4, NVMADRU
Step 6: Set the NVMCON register to program two instruction words.		
0000	24001A	MOV #0x4001, W10
0000	000000	NOP
0000	88468A	MOV W10, NVMCON
0000	000000	NOP
0000	000000	NOP

TABLE 3-6: SERIAL INSTRUCTION EXECUTION FOR PROGRAMMING CODE MEMORY: TWO-WORD LATCH WRITES (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 7: Initiate the write cycle.		
0000	200551	MOV #0x55, W1
0000	8846B1	MOV W1, NVMKEY
0000	200AA1	MOV #0xAA, W1
0000	8846B1	MOV W1, NVMKEY
0000	A8E8D1	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 8: Generate clock pulses for the program operation to complete until the WR bit is clear.		
0000	000000	NOP
0000	804680	MOV NVMCON, W0
0000	00000	NOP
0000	887E60	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the 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 Configuration Bits

The procedure for writing Configuration bits is similar to the procedure for writing code memory.

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.

Table 3-7 provides the ICSP programming details for writing the Configuration bits.

The code protection can be enabled by programming '0' in the Code Protection 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 '1' to ensure that the verification can be performed properly. After verification is finished, the code protection bits can be programmed to '0' by using a word write to the appropriate Configuration register.

TABLE 3-7: SERIAL INSTRUCTION EXECUTION FOR WRITING CONFIGURATION 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 the latches.		
0000	200FAC	MOV #0xFA, W12
0000	8802AC	MOV W12, TBLPAG
Step 3: Load W0:W1 with the next two Configuration Words to program.		
0000	2xxxx0	MOV #<Config1 lower word data>, W0
0000	2xxxx1	MOV #<Config1 upper word data>, W1
0000	2xxxx2	MOV #<Config2 lower word data>, W2
0000	2xxxx3	MOV #<Config2 upper word data>, W3
Step 4: Set the Write Pointer (W3) and load the write latches.		
0000	EB0300	CLR W6
0000	000000	NOP
0000	BB0B00	TBLWTL W0, [W6]
0000	000000	NOP
0000	000000	NOP
0000	BB9B01	TBLWTH W1, [W6++]
0000	000000	NOP
0000	000000	NOP
0000	BB0B02	TBLWTL W2, [W6]
0000	000000	NOP
0000	000000	NOP
0000	BB9B03	TBLWTH W3, [W6++]
0000	000000	NOP
0000	000000	NOP
Step 5: Set the NVMADRU/NVMADR register pair to point to the correct Configuration Word address.		
0000	2xxxx4	MOV #DestinationAddress<15:0>, W4
0000	2xxxx5	MOV #DestinationAddress<23:16>, W5
0000	884694	MOV W4, NVMADR
0000	8846A5	MOV W5, NVMADRU

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

Command (Binary)	Data (Hex)	Description
Step 6: Set the NVMCON register to program two instruction words.		
0000	24001A	MOV #0x4001, W10
0000	000000	NOP
0000	88468A	MOV W10, NVMCON
0000	000000	NOP
0000	000000	NOP
Step 7: Initiate the write cycle.		
0000	200551	MOV #0x55, W1
0000	8846B1	MOV W1, NVMKEY
0000	200AA1	MOV #0xAA, W1
0000	8846B1	MOV W1, NVMKEY
0000	A8E8D1	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 8: Generate clock pulses for the program operation to complete until the WR bit is clear.		
0000	000000	NOP
0000	804680	MOV NVMCON, W0
0000	000000	NOP
0000	887E60	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the 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 Configuration registers are programmed.		

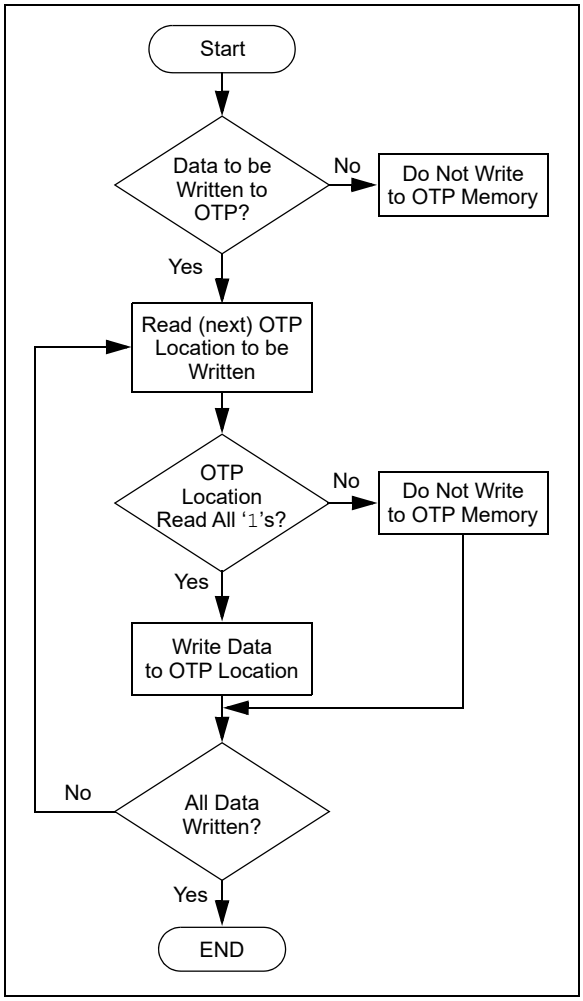
3.9 Writing OTP Words

The procedure for writing to user OTP memory is similar to the procedure for writing to user program memory, except that each of the 64 OTP double-word pairs can only be written once. Both words in each OTP location must be written together using the same two-word latch write process used to write code memory.

Writing anything, with the exception of all '1's, to an OTP location generates an ECC checksum and renders that location used. Attempting to write to an OTP location that has already been programmed will cause an ECC checksum error the next time that location is read. Care should be taken to avoid writing to OTP locations that have already been programmed or may need to be programmed at a later time. See Figure 2-5 for the location of user OTP memory.

Figure 3-9 shows a high-level overview of the OTP programming process.

FIGURE 3-9: OTP PROGRAMMING PROCESS



3.10 Reading OTP Words

The procedure for reading OTP Words is similar to the procedure for reading code memory. Since there are multiple OTP Words, they are read one at a time.

3.11 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 provides the ICSP programming details for reading code memory.

To minimize reading time, the same packed data format that the write procedure uses is utilized. See **Section 3.7 "Writing Code Memory"** 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 the TBLPAG register and the Read Pointer (W6) for the 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	887E60	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the VISI register.
0000	000000	NOP
0000	887E61	MOV W1, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the VISI register.
0000	000000	NOP
0000	887E62	MOV W2, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the VISI register.
0000	000000	NOP
0000	887E63	MOV W3, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the VISI register.
0000	000000	NOP
0000	887E64	MOV W4, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the VISI register.
0000	000000	NOP
0000	887E65	MOV W5, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the VISI register.
0000	000000	NOP
Step 5: Reset the device's 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.		

3.12 Reading Configuration Registers

The procedure for reading the Configuration Words is similar to the procedure for reading code memory. Since there are multiple Configuration Words, they are read one at a time.

Table 3-9 provides the ICSP programming details for reading the Configuration Words.

TABLE 3-9: SERIAL INSTRUCTION EXECUTION FOR READING CONFIGURATION 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, the Write Pointer (W7) and the Read Pointer (W6) for the TBLRD instruction.		
0000	200xx0	MOV #<Address23:16>, W0
0000	20FCC7	MOV #VISI, W7
0000	8802A0	MOV W0, TBLPAG
0000	2xxxx6	MOV #<Address15:0>, W6
Step 3: Store the Configuration register and send the contents of the VISI register.		
0000	000000	NOP
0000	BA8B96	TBLRDH [W6], [W7]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0001	<VISI>	Clock out contents of the VISI register.
0000	BA0B96	TBLRDL [W6], [W7]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0001	<VISI>	Clock out contents of the VISI register.
Step 4: Repeat Steps 1-3 until all Configuration registers are read.		

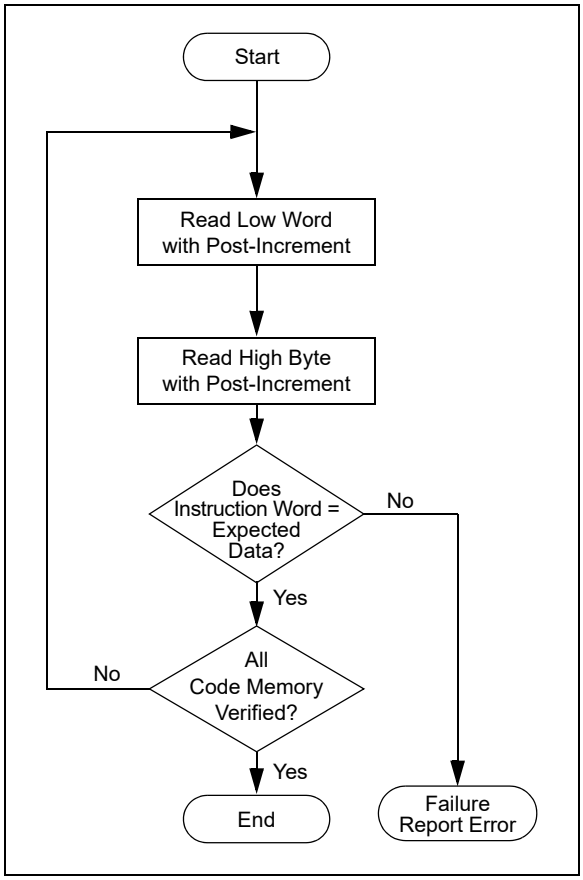
3.13 Verify Code Memory and Configuration Bits

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 shown in Figure 3-10. 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 3.11 "Reading Code Memory"** for implementation details of reading code memory.

Note:	Because the Configuration Words include the device code protection bit, code memory should be verified immediately after writing if 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.

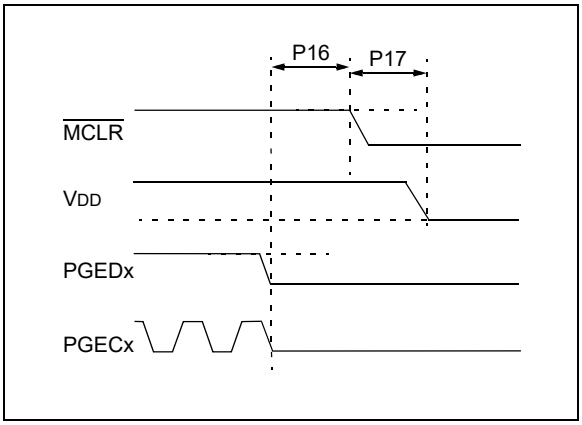
FIGURE 3-10: VERIFY CODE MEMORY FLOW



3.14 Exiting ICSP Mode

Exiting Program/Verify mode is done by removing VDD from MCLR, as shown in Figure 3-11. The only requirement for exit is that an interval, P16, should elapse between the last clock, and the program signals on PGECx and PGEDx before removing VDD.

FIGURE 3-11: 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 dsPIC33CDVL64MC106 device 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

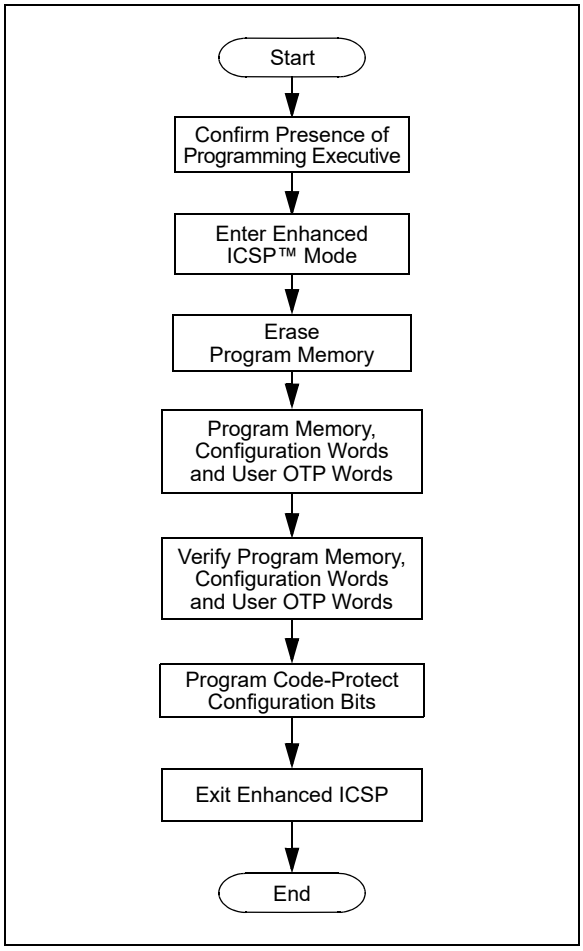
The Programming Executive performs the low-level tasks required for erasing, programming and verifying a device. This allows the programmer to program the device by issuing the appropriate commands and data. A detailed description for each command is provided in **Section 5.2 "Programming Executive Commands"**.

Note:	The PE uses the device's data RAM for variable storage and program execution. After running the PE, 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, then the 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



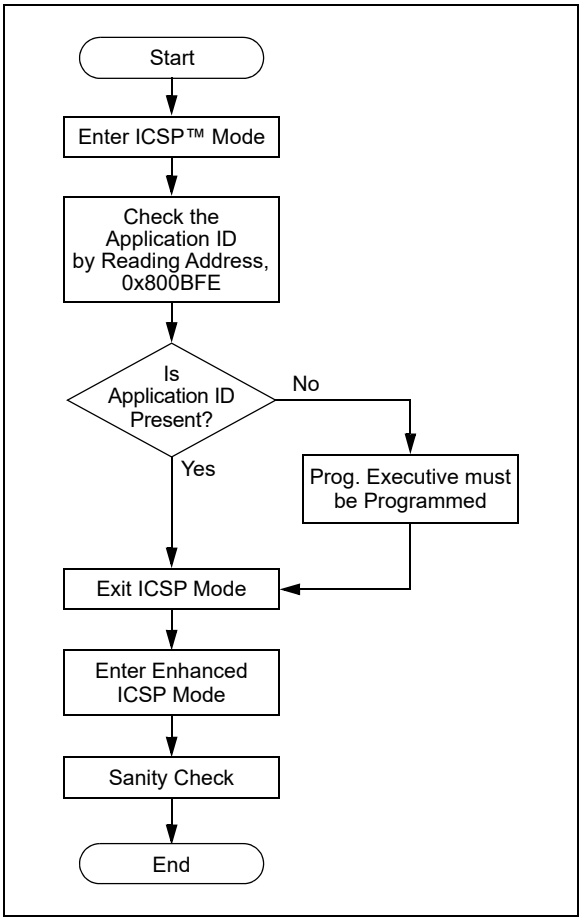
4.2 Confirming the Presence of the Programming Executive

Before programming can begin, the programmer must confirm that the Programming Executive is stored in executive memory. The procedure for this task is shown in Figure 4-2.

First, In-Circuit Serial Programming (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, 0xDF, is read and programming can resume as normal. However, if the Application ID Word is not present, the PE must be programmed to executive code memory using the method described in **Section 5.0 "The Programming Executive"**.

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



4.3 Reading the Application ID Word

The Application ID Word is stored at the address, 0x800BFE, 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 provided 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, 0xDF, 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 into memory before the device can be programmed. The procedure for loading the Programming Executive to memory is described in **Section 5.0 "The Programming Executive"**.

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 the TBLPAG register and the Read Pointer (W0) for the TBLRD instruction.		
0000	200800	MOV #0x80, W0
0000	8802A0	MOV W0, TBLPAG
0000	20BFE0	MOV #0xBFE, W0
0000	20FCC1	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 shown in Figure 4-3, entering Enhanced ICSP Program/Verify mode requires three steps:

1. The MCLR pin is briefly driven high, then low.
2. A 32-bit key sequence is clocked into PGEDx. An interval of at least P18 must elapse before presenting the key sequence on PGEDx.
3. MCLR is held within a specified period of time, then driven high.

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, VDD 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 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.

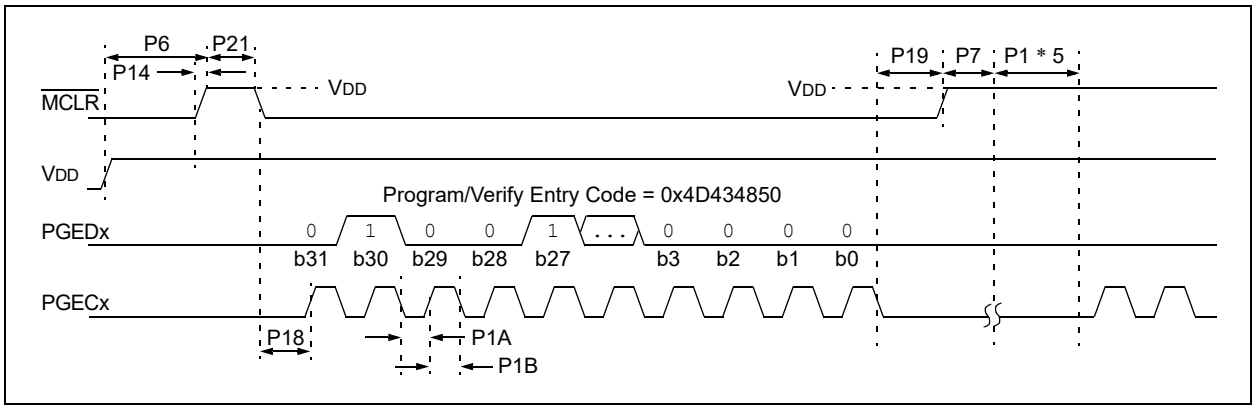
4.5 Blank Check

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 and Calibration registers should be ignored by 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 specified address. The second and faster command, PROGP, programs and verifies an entire row of 128 24-bit instruction words to program memory, starting at the specified address. Please ensure that the starting address is on a row boundary when using Row Programming. See **Section 5.0 "The Programming Executive"** for a full description of each of these commands.

Figure 4-4 and Figure 4-5 show a high-level overview of the code memory programming process using the PROG2W and PROGP commands.

FIGURE 4-4: FLOWCHART FOR DOUBLE-WORD PROGRAMMING

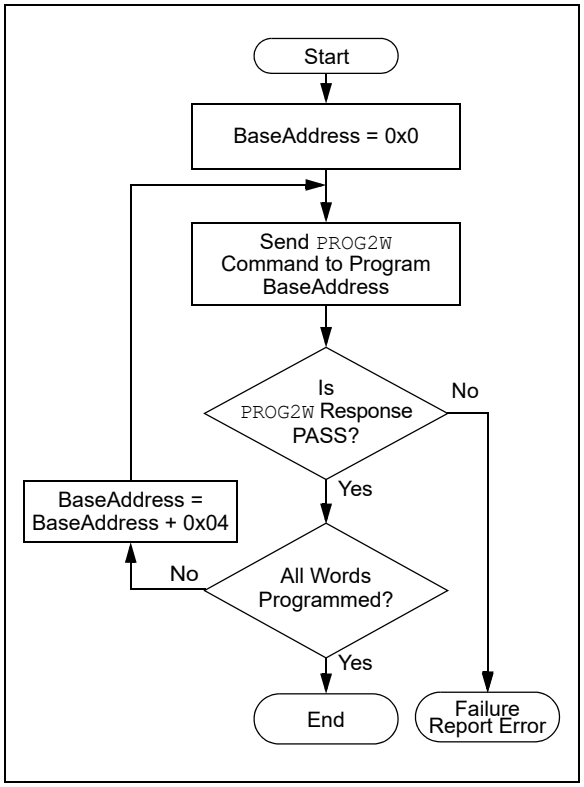
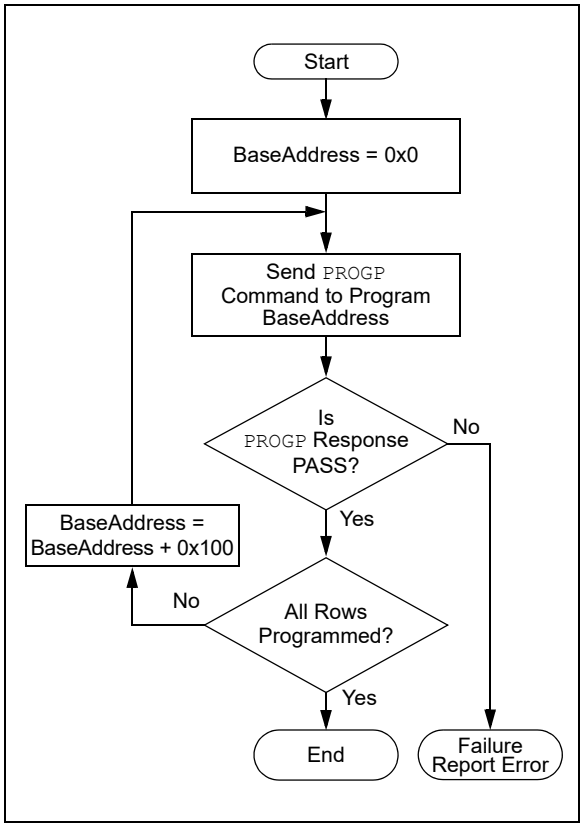


FIGURE 4-5: FLOWCHART FOR ROW PROGRAMMING

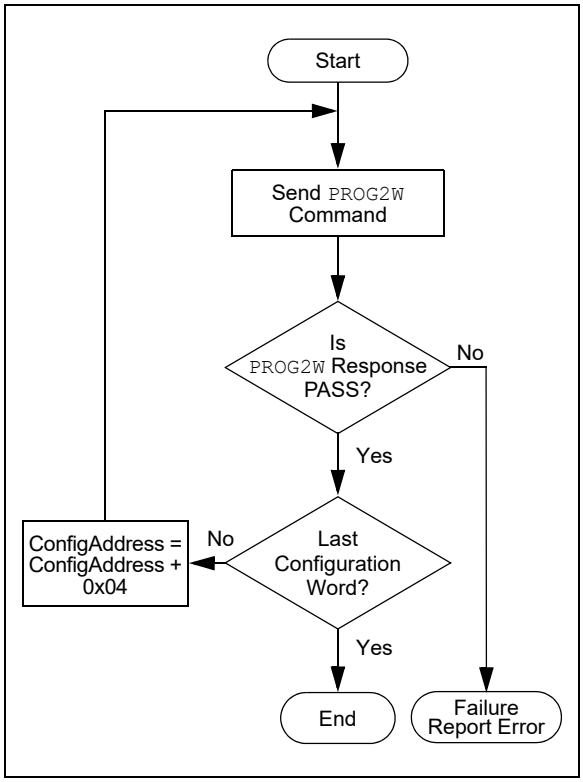


4.7 Configuration Bits Programming

The Configuration bits are programmed, one 16-bit word 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 the code memory is programmed, the contents of the memory should be verified to ensure that the 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.

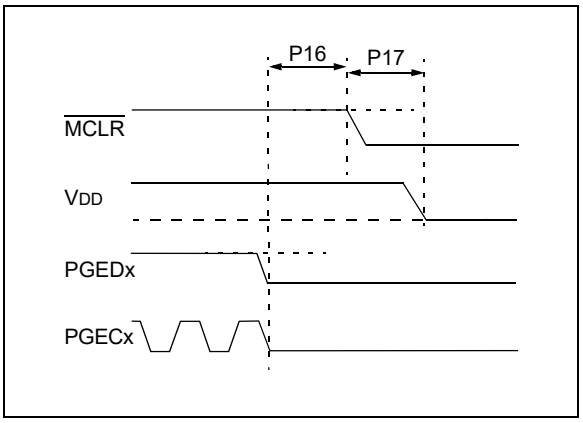
Alternatively, the programmer can perform the verification after the entire device is programmed using a checksum computation.

See **Section 7.0 "CRC Checksum Computation"** for more information on calculating the checksum.

4.9 Exiting Enhanced ICSP Mode

Exiting Program/Verify mode is done by removing VDD from MCLR, as shown 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 VDD.

FIGURE 4-7: EXITING ENHANCED ICSP™ MODE



5.0 THE PROGRAMMING EXECUTIVE

Note: The Programming Executive can be obtained from each device page on the Microchip website: www.microchip.com.

5.1 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.

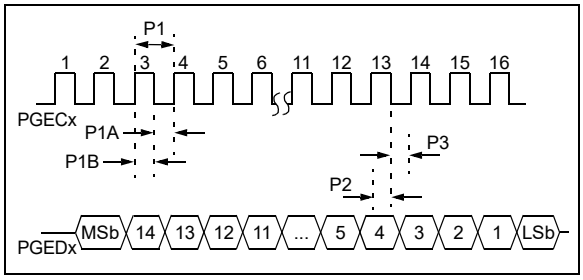
All communication is initiated by the programmer in the form of a command. Only one command at a time can be sent to the Programming Executive. In turn, the PE only sends one response to the programmer after receiving and processing a command. The PE command set is described in **Section 5.2 "Programming Executive Commands"**. The response set is described in **Section 5.3 "Programming Executive Responses"**.

5.1.1 COMMUNICATION INTERFACE AND PROTOCOL

The ICSP/Enhanced ICSP interface is a two-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 PE.

Note: For Enhanced ICSP, all serial data are 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 5-1).

FIGURE 5-1: PROGRAMMING EXECUTIVE SERIAL TIMING



Since a two-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 PE to drive this line high. The PE keeps the PGEDx line high to indicate that it is processing the command.

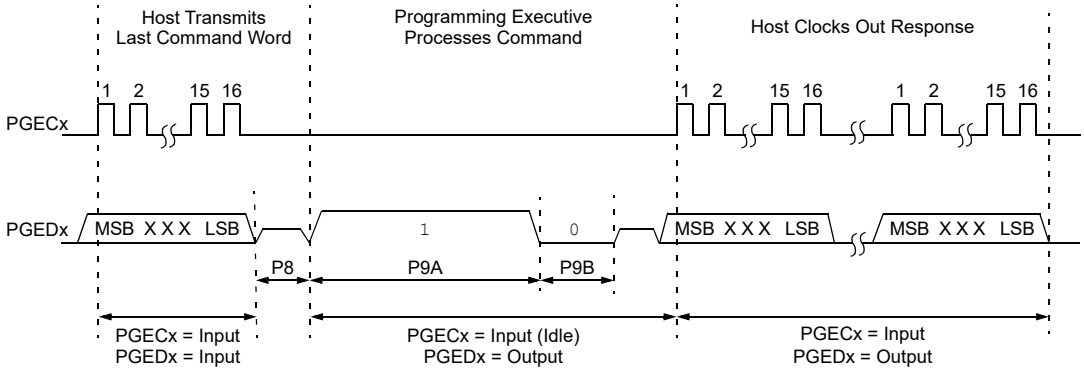
After the PE has processed the command, it brings PGEDx low (P9B) to indicate to the programmer that the response is available to be clocked out. The programmer can begin to clock out the response after a maximum wait (P9B) and the programmer must provide the necessary amount of clock pulses to receive the entire response from the PE.

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 shown in Figure 5-2.

5.1.2 SPI RATE

In Enhanced ICSP mode, the dsPIC33CDVL64MC106 device operates from the internal Fast RC (FRC) oscillator, which has a nominal frequency of 8 MHz. This oscillator frequency yields an effective system clock frequency of 4 MHz. To ensure that the programmer does not clock too fast, it is recommended that a 2 MHz clock be provided by the programmer.

FIGURE 5-2: PROGRAMMING EXECUTIVE – PROGRAMMER COMMUNICATION PROTOCOL



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

5.1.3 TIME-OUTS

The Programming Executive uses no Watchdog Timer 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 5.1.1 "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 PE has no time-out, it is imperative that the programmer correctly follows the described communication protocol.

As a safety measure, the programmer should use the command time-outs identified in Table 5-1. If the command time-out expires, the programmer should reset the PE and start programming the device again.

TABLE 5-1: PROGRAMMING EXECUTIVE COMMAND SET

Opcode	Mnemonic	Length (16-bit words)	Time-out	Description
0x0	SCHECK	1	1 ms	Sanity check.
0x1	Reserved	N/A	N/A	—
0x2	READP	4	1 ms/row	Read 'N' 24-bit instruction words of the user Flash memory, Configuration Word or Device ID register, 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	195	5 ms	Program 128 words of program memory at the specified starting address, then verify.
0x6	Reserved	N/A	N/A	This command is reserved; it will return a NACK.
0x7	ERASEB	1	125 ms	Bulk Erase user memory.
0x8	Reserved	N/A	N/A	This command is reserved; it will return a NACK.
0x9	ERASEP	3	25 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	Perform 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.

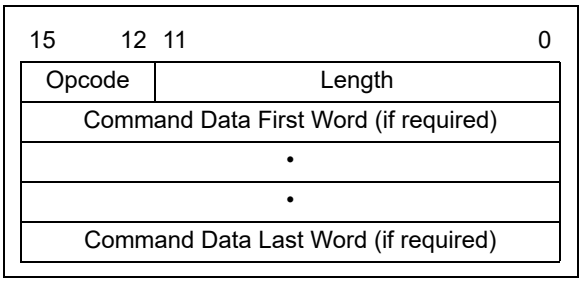
5.2 Programming Executive Commands

The Programming Executive command set is shown in Table 5-1. 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 (see **Section 5.2.4 "Command Descriptions"**).

5.2.1 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 5-3). 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 5-3: COMMAND FORMAT



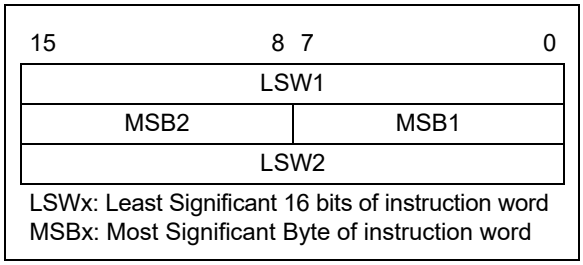
The command opcode must match one of those in the command set. Any command that is received which does not match the list in Table 5-1 will return a "NACK" response (see **Section 5.3.1.1 "Opcode Field"**).

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.

5.2.2 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 shown in Figure 5-4. This format minimizes traffic over the SPI and provides the Programming Executive with data that are properly aligned for performing Table Write operations.

FIGURE 5-4: PACKED INSTRUCTION WORD FORMAT



LSWx: Least Significant 16 bits of instruction word
MSBx: Most Significant Byte of instruction word

5.2.3 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 5.3.1.3 "QE_Code Field"**.

5.2.4 COMMAND DESCRIPTIONS

All commands supported by the Programming Executive are described in **Section 5.2.4.1 "SCHECK Command"** through **Section 5.2.4.9 "QBLANK Command"**.

5.2.4.1 SCHECK Command

15 12	11 0
Opcode	Length

Table 5-2 provides the description for the SCHECK command.

TABLE 5-2: COMMAND DESCRIPTION

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.

5.2.4.2 READP Command

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

Table 5-3 provides the description for the READP command.

TABLE 5-3: COMMAND DESCRIPTION

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, Flash Configuration Words or Device ID registers, 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 response to this command use the packed data format described in **Section 5.2.2 "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 1:	Reading unimplemented memory will cause the Programming Executive to reset. Please ensure that only memory locations present on a particular device are accessed.
2:	When the READP command is used to read Device ID registers, the upper byte (bits[23:16]) of each word returned by the Programming Executive should be ignored.

5.2.4.3 PROG2W Command

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

Table 5-4 provides the description for the PROG2W command.

TABLE 5-4: COMMAND DESCRIPTION

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

5.2.4.4 PROGP Command

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

Table 5-5 provides the description for the PROGP command.

TABLE 5-5: COMMAND DESCRIPTION

Field	Description
Opcode	0x5
Length	0xC3
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 191
D_192	16-bit Data Word 192

The PROGP command instructs the Programming Executive to program one row of code memory (128 instruction words) to the specified memory address. Programming begins with the row address specified in the command. The destination address should be a multiple of 0x100.

The data to program the memory, located in command words, D_1 through D_192, must be arranged using the packed instruction word format illustrated in Figure 5-4.

After all data have 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 2-3 for code memory size information.

5.2.4.5 ERASEB Command

15 12	11 8	7 0
Opcode		Length

Table 5-6 provides the description for the ERASEB command.

TABLE 5-6: COMMAND DESCRIPTION

Field	Description
Opcode	0x7
Length	0x1

The ERASEB command instructs the Programming Executive to perform a Bulk Erase of the user Flash memory.

Expected Response (2 words):

0x1700
0x0002

5.2.4.6 ERASEP Command

15 12	11 8	7 0
Opcode	Length	
NUM_PAGES		Addr_MSB
Addr_LS		

Table 5-7 provides the description for the ERASEP command.

TABLE 5-7: COMMAND DESCRIPTION

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" 1024 instruction word address boundary.

Expected Response (2 words):

0x1900
0x0002

5.2.4.7 QVER Command

15 12	11 0
Opcode	Length

Table 5-8 provides the description for the QVER command.

TABLE 5-8: COMMAND DESCRIPTION

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 a revision in the lower nibble (i.e., 0x23 means Version 2.3 of the Programming Executive software).

Expected Response (2 words):

0x1BMN (where "MN" stands for Version M.N)
0x0002

5.2.4.8 CRCP Command

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

Table 5-9 provides the description for the CRCP command.

TABLE 5-9: COMMAND DESCRIPTION

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 two)

The CRCP command performs a CRC-16 on the range of memory specified. This command can substitute for a full chip verify. Data are shifted in a packed method as shown in Figure 5-4, bytewise, 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

5.2.4.9 QBLANK Command

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

Table 5-10 provides the description for the QBLANK command.

TABLE 5-10: COMMAND DESCRIPTION

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: The QBLANK command does not check the system operation Configuration bits, since these bits are not set to '1' when a Chip Erase is performed.

5.3 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 5-11. This table contains the opcode, mnemonic and description for each response. The response format is described in **Section 5.3.1 "Response Format"**.

TABLE 5-11: PROGRAMMING EXECUTIVE RESPONSE OPCODES

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

5.3.1 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)		

Table 5-12 provides the description of the response format.

TABLE 5-12: RESPONSE FORMAT DESCRIPTION

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 two header words)
D_1	First 16-bit data word (if applicable)
D_N	Last 16-bit data word (if applicable)

5.3.1.1 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 5-11). 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.

5.3.1.2 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.

5.3.1.3 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 5-13.

TABLE 5-13: 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 5-14. 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 5-14: QE_Code FOR NON-QUERY COMMANDS

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

5.3.1.4 Response Length

The response length indicates the length of the Programming Executive's response in 16-bit words. This field includes the two words of the response header.

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

The response to the `READP` commands uses the packed instruction word format described in **Section 5.2.2 "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.

5.4 Programming the Programming Executive to Memory

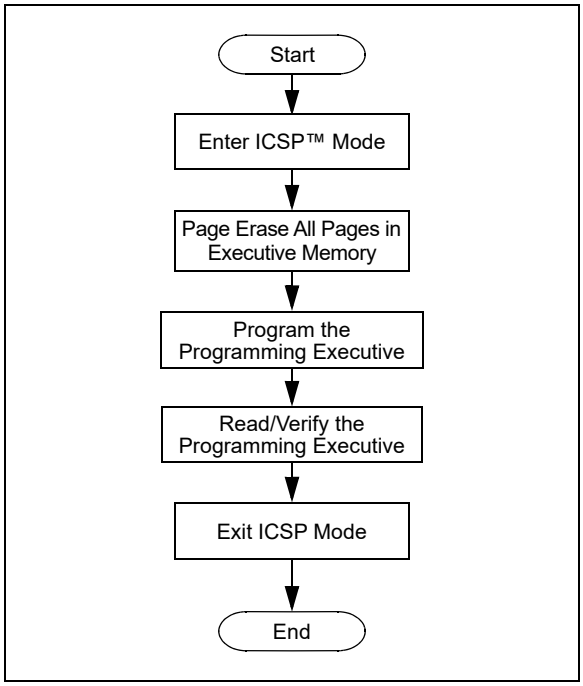
Note:	The Programming Executive can be obtained from each device page on the Microchip website: www.microchip.com.

5.4.1 OVERVIEW

If it is determined that the Programming Executive is not present in the 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 5-5 shows the high-level process of programming the Programming Executive into executive memory. First, the ICSP mode must be entered and the executive memory must be erased. Then, the Programming Executive is programmed and verified. Finally, ICSP mode is exited.

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



5.4.2 ERASING EXECUTIVE MEMORY

The procedure for erasing each page of executive memory is similar to that of erasing program memory and is shown in Figure 3-6. It consists of setting NVMCON to 0x4003 and then executing the programming cycle.

Table 5-15 shows the ICSP programming process for erasing the executive code memory.

Note:	The Programming Executive memory must always be erased before it is programmed, as described in Figure 5-5.

TABLE 5-15: SERIAL INSTRUCTION EXECUTION FOR ERASING ALL PAGES OF 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 NVMADRU/NVMADR register pair to point to the correct page of executive memory to be erased.		
0000	2xxxx3	MOV #DestinationAddress<15:0>, W3
0000	2xxxx4	MOV #DestinationAddress<23:16>, W4
0000	884693	MOV W3, NVMADR
0000	8846A4	MOV W4, NVMADRU
Step 3: Set the NVMCON register to erase the first page of executive memory.		
0000	24003A	MOV #0x4003, W10
0000	88468A	MOV W10, NVMCON
0000	000000	NOP
0000	000000	NOP
Step 4: Initiate the erase cycle.		
0000	200551	MOV #0x55, W1
0000	8846B1	MOV W1, NVMKEY
0000	200AA1	MOV #0xAA, W1
0000	8846B1	MOV W1, NVMKEY
0000	A8F1A1	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 5: Generate clock pulses for the Page Erase operation to complete until the WR bit is clear.		
0000	000000	NOP
0000	804680	MOV NVMCON, W0
0000	000000	NOP
0000	887E60	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the 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 6: Repeat Steps 2-5 for all pages of executive memory.		

5.4.3 PROGRAM THE PROGRAMMING EXECUTIVE

Storing the PE to executive memory is similar to normal programming of code memory. The executive memory must first be erased and then programmed using two-word writes (two instruction words). The control flow for this method is summarized in Figure 5-6.

Table 5-16 provides the ICSP programming processes for PE memory. To minimize programming time, the same packed data format that the PE uses is utilized. See **Section 5.2 "Programming Executive Commands"** for more details on the packed data format.

FIGURE 5-6: PROGRAMMING EXECUTIVE PROGRAM FLOW

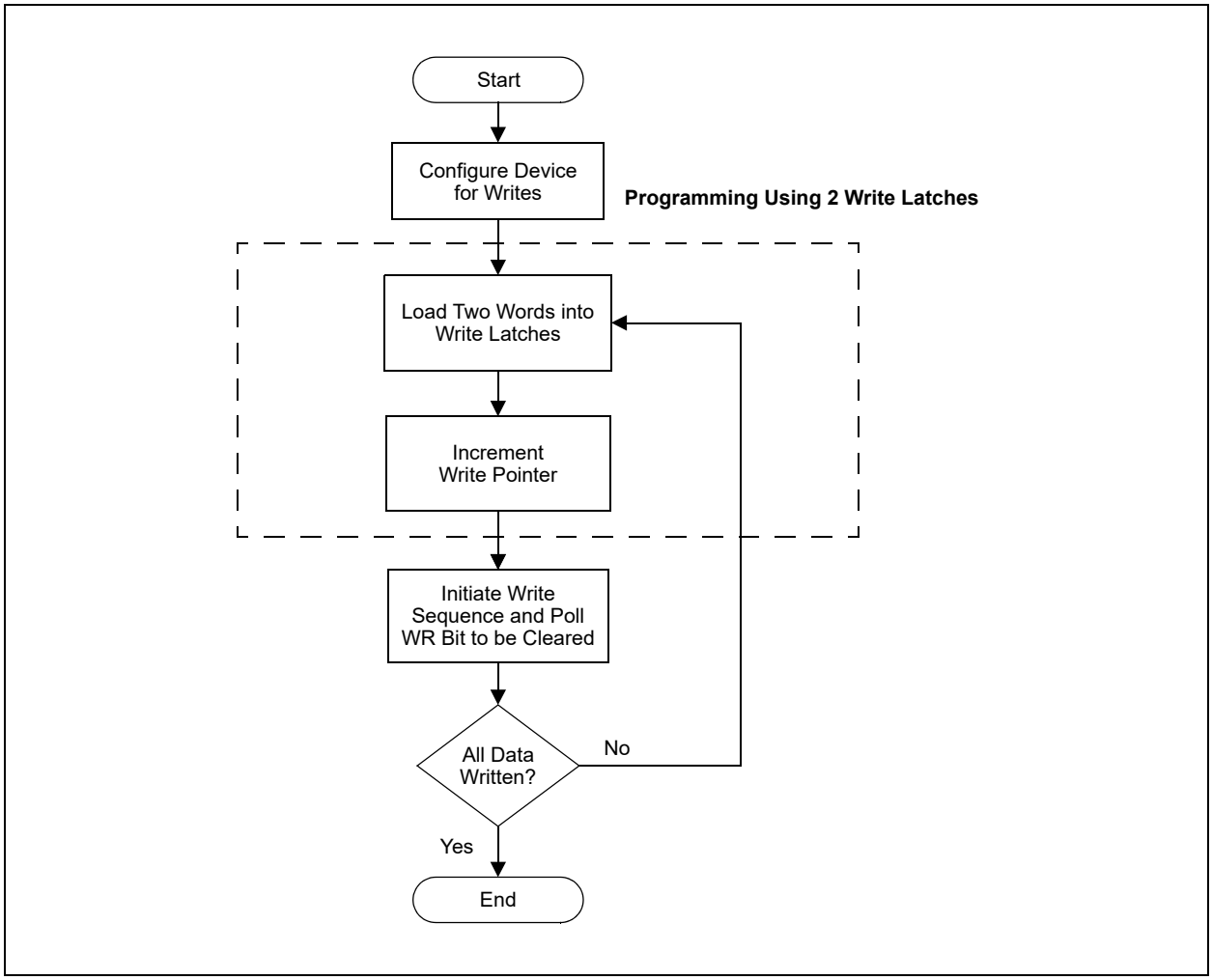


TABLE 5-16: PROGRAMMING THE PROGRAMMING EXECUTIVE (TWO-WORD LATCH WRITES)

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 the latches.		
0000	200FAC	MOV #0xFA, W12
0000	8802AC	MOV W12, TBLPAG
Step 3: Load W0:W2 with the next two packed instruction words to program.		
0000	2xxxx0	MOV #<LSW0>, W0
0000	2xxxx1	MOV #<MSB1:MSB0>, W1
0000	2xxxx2	MOV #<LSW1>, W2
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	BB0B96	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	884693	MOV W3, NVMADR
0000	8846A4	MOV W4, NVMADRU
Step 6: Set the NVMCON register to program two instruction words.		
0000	24001A	MOV #0x4001, W10
0000	000000	NOP
0000	88468A	MOV W10, NVMCON
0000	000000	NOP
0000	000000	NOP

TABLE 5-16: PROGRAMMING THE PROGRAMMING EXECUTIVE (TWO-WORD LATCH WRITES) (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 7: Initiate the write cycle.		
0000	200551	MOV #0x55, W1
0000	8846B1	MOV W1, NVMKEY
0000	200AA1	MOV #0xAA, W1
0000	8846B1	MOV W1, NVMKEY
0000	A8F1A1	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.		
0000	000000	NOP
0000	804680	MOV NVMCON, W0
0000	000000	NOP
0000	887E60	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the 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.		

5.4.4 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 5-17 provides the ICSP programming details for reading executive memory.

To minimize reading time, the same packed data format that the PE uses is utilized. See **Section 5.2 "Programming Executive Commands"** for more details on the packed data format.

TABLE 5-17: 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 the TBLPAG register and the Read Pointer (W6) for the TBLRD instruction.		
0000	200xx0	MOV #<SourceAddress23:16>, W0
0000	8802A0	MOV W0, TBLPAG
0000	2xxxx6	MOV #<SourceAddress15:0>, W6

TABLE 5-17: 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 5-17: SERIAL INSTRUCTION EXECUTION FOR READING CODE MEMORY (CONTINUED)

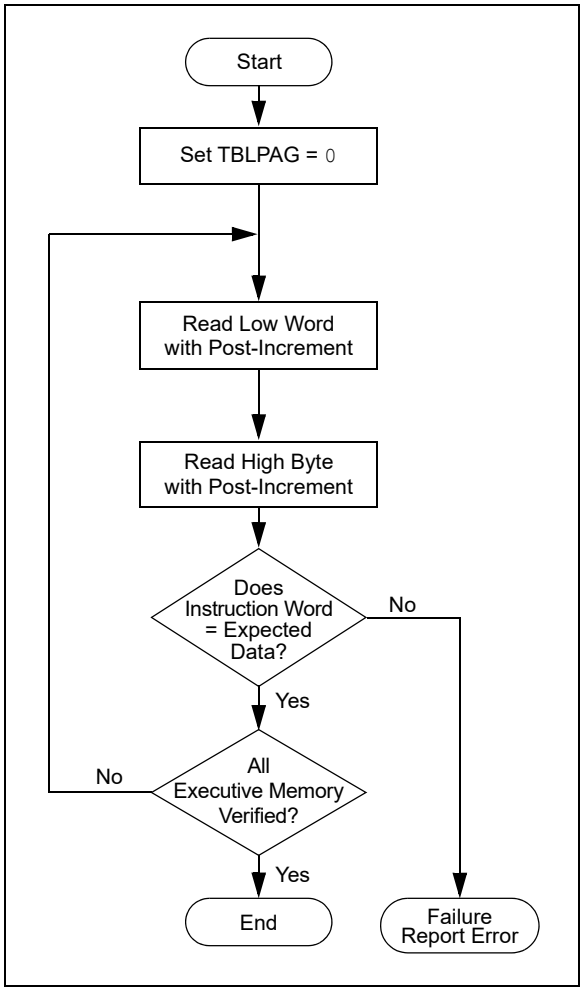
Command (Binary)	Data (Hex)	Description
Step 4: Output W0:W5 using the VISI register and REGOUT command.		
0000	887E60	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the VISI register.
0000	000000	NOP
0000	887E61	MOV W1, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the VISI register.
0000	000000	NOP
0000	887E62	MOV W2, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the VISI register.
0000	000000	NOP
0000	887E63	MOV W3, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the VISI register.
0000	000000	NOP
0000	887E64	MOV W4, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the VISI register.
0000	000000	NOP
0000	887E65	MOV W5, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the VISI register.
0000	000000	NOP
Step 5: Reset the device's 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.		

5.4.5 PROGRAMMING VERIFICATION

The verification is performed by reading back the executive memory space and comparing it against the copy held in the programmer's buffer.

The verification process is shown in Figure 5-7. 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 5.4.4 "Reading Executive Memory"** for implementation details of reading executive memory.

FIGURE 5-7: VERIFY PROGRAMMING EXECUTIVE MEMORY FLOW



6.0 DEVICE ID/UNIQUE 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 6-1 lists the identification information for the device. Table 6-2 shows the Device ID registers. Table 6-3 shows that the JTAG ID is generated given a device DEVREV and DEVID.

TABLE 6-1: DEVICE ID

Device	DEVID
dsPIC33CDVL64MC106	0x991A
dsPIC33CDV64MC106	0x991B

TABLE 6-2: 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															
0x8016FC	DEVREV	DEVREV Value															

TABLE 6-3: JTAG ID

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

6.1 Unique Device ID (UDID)

The dsPIC33CDVL64MC106 device is individually encoded during final manufacturing with a Unique Device Identifier or UDID. The UDID cannot be erased by a Bulk Erase command or any other user-accessible means. This feature allows for manufacturing traceability of Microchip Technology devices in applications where this is a requirement. It may also be used by the application manufacturer for any number of things that may require unique identification, such as:

- Tracking the device
- Unique serial number
- Unique security key

The UDID comprises five 24-bit program words. When taken together, these fields form a unique 120-bit identifier. The UDID is stored in five read-only locations in the device configuration space. Table 6-4 lists the addresses of the Identifier Words and shows their contents.

TABLE 6-4: UDID ADDRESSES

UDID	Address	Content
UDID1	0x801200	UDID Word 1
UDID2	0x801202	UDID Word 2
UDID3	0x801204	UDID Word 3
UDID4	0x801206	UDID Word 4
UDID5	0x801208	UDID Word 5

7.0 CRC CHECKSUM COMPUTATION

Unlike older PIC® devices, the Microchip toolchain runs a 32-bit CRC calculation on the entire Hex file to calculate its checksum. The checksum uses the standard CRC-32 algorithm with the polynomial, 0x4C11DB7.

EXAMPLE 7-1: CRC CHECKSUM COMPUTATION

$$(x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1)$$

8.0 AC/DC CHARACTERISTICS AND TIMING REQUIREMENTS

Table 8-1 lists the AC/DC characteristics and timing requirements.

TABLE 8-1: AC/DC CHARACTERISTICS AND TIMING REQUIREMENTS

Standard Operating Conditions Operating Temperature: -40°C to +125°C. Programming at +25°C is recommended.						
Param. No.	**Symbol**	**Characteristic**	**Min.**	**Max.**	**Units**	**Conditions**
D111	VDD	Supply Voltage During Programming	3.0	3.60	V	Normal programming(1)
D113	IDDP	Supply Current During Programming	—	10	mA	
D114	IPEAK	Instantaneous Peak Current During Start-up	—	200	mA	
D031	VIL	Input Low Voltage	VSS	0.2 VDD	V	
D041	VIH	Input High Voltage	0.7 VDD	VDD	V	
D080	VOL	Output Low Voltage	—	0.4	V	
D090	VOH	Output High Voltage	2.4	—	V	
D012	CIO	Capacitive Loading on I/O Pin (PGEDx)	—	50	pF	
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 PE	12	—	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: Time depends on the FRC accuracy and the value of the FRC Oscillator Tuning register. Refer to the **"Electrical Characteristics"** chapter in the specific device data sheet.

TABLE 8-1: AC/DC CHARACTERISTICS AND TIMING REQUIREMENTS (CONTINUED)

Standard Operating Conditions Operating Temperature: -40°C to +125°C. Programming at +25°C is recommended.						
Param. No.	**Symbol**	**Characteristic**	**Min.**	**Max.**	**Units**	**Conditions**
P9A	TDLY4	Programming Executive Command Processing Time	10	—	µs	
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	—	16.0	ms	See **Note 2**
P12	TDLY8	Page Erase Time	—	4.2	ms	See **Note 2**
P13	TDLY9	Double-Word Write Time	—	34.5	µs	See **Note 2**
		Row Write Time		1.1	ms	See **Note 2**
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: Time depends on the FRC accuracy and the value of the FRC Oscillator Tuning register. Refer to the **"Electrical Characteristics"** chapter in the specific device data sheet.

APPENDIX A: REVISION HISTORY

Revision A (September 2020)

Original version of the programming specification created for the dsPIC33CDVL64MC106 device.

Revision B (August 2021)

Added the dsPIC33CDV64MC106 device.

Updated Figure 2-3 and added Figure 2-4.

Updated Table 2-3 and Table 6-1.

Revision C (January 2023)

Updated Figure 2-5.

Updated Table 6-2.

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable" Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Trademarks

ISBN: 978-1-6683-1884-3

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

Austin, TX
Tel: 512-257-3370

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

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

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

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

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

China - Beijing
Tel: 86-10-8569-7000

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

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

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

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

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

China - Shanghai
Tel: 86-21-3326-8000

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

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

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

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

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

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

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

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

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

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

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

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

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

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

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

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

Denmark - Copenhagen
Tel: 45-4485-5910
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

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

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-72400

Germany - Karlsruhe
Tel: 49-721-625370

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

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

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

Italy - Padova
Tel: 39-049-7625286

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

Norway - Trondheim
Tel: 47-7288-4388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

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

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820