dsPIC30F SMPS

dsPIC30F SMPS Flash Programming Specification

1.0 OVERVIEW AND SCOPE

This document defines the programming specification for the dsPIC30F Switched Mode Power Supply (SMPS) and Digital Power Conversion family of Digital Signal Controller (DSC) devices. The programming specification is required only for developers of third-party tools that are used to program dsPIC30F SMPS devices. Customers using dsPIC30F SMPS devices should use development tools that already provide support for device programming.

This document includes programming specifications for the following devices:

- dsPIC30F1010
- dsPIC30F2020
- dsPIC30F2023

The dsPIC30F SMPS family enters programming modes via the 32-bit serial key sequence clocked into the PGD line, similar to the dsPIC33F family. On the other hand, the programming operations themselves are similar to the other dsPIC30F devices.

The dsPIC30F SMPS family does not contain data EEPROM.

2.0 PROGRAMMING OVERVIEW OF THE dsPIC30F SMPS

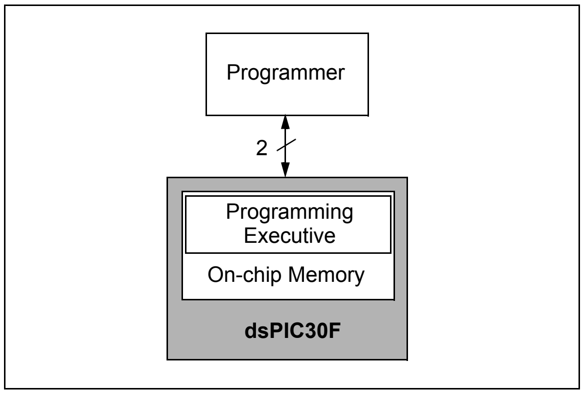
The dsPIC30F SMPS family of DSCs contains a region of on-chip memory used to simplify device programming. This memory region can store a programming executive, which allows the dsPIC30F SMPS to be programmed faster than the traditional method. Once the programming executive is stored to memory by an external programmer (such as Microchip's MPLAB® ICD 2 or PRO MATE® II), it can then interact with the external programmer to efficiently program devices.

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, as illustrated in Figure 2-1.

There are two different methods used to program the chip in the user's system. One method uses the Enhanced In-Circuit Serial Programming™ (Enhanced ICSP™) protocol and works with the programming executive. The other method uses In-Circuit Serial Programming (ICSP) protocol and does not use the programming executive.

The Enhanced ICSP protocol uses the faster, high-voltage method that takes advantage of the programming executive. The programming executive provides all the necessary functionality to erase, program and verify the chip through a small command set. The command set allows the programmer to program the dsPIC30F without having to deal with the low-level programming protocols of the chip.

FIGURE 2-1: OVERVIEW OF dsPIC30F SMPS FAMILY PROGRAMMING



The ICSP programming method does not use the programming executive. It provides native, low-level programming capability to erase, program and verify the chip. This method is significantly slower because it uses control codes to serially execute instructions on the dsPIC30F SMPS device.

This specification describes both the Enhanced ICSP and ICSP programming methods. **Section 3.0 "Programming Executive Application"** describes the programming executive application and **Section 5.0 "Device Programming"** describes its application programmer's interface for the host. programmer. **Section 11.0 "ICSP™ Mode"** describes the ICSP programming method.

2.1 Hardware Requirements

In Enhanced ICSP mode, the dsPIC30F SMPS requires a single programmable power supply for VDD. Refer to **Section 13.0 "AC/DC Characteristics and Timing Requirements"** for hardware parameters.

2.2 Pins Used During Programming

Table 2-2 lists the pins that are required for programming. Refer to the appropriate device data sheet for complete pin descriptions.

2.3 Program Memory Map

The program memory space extends from 0x0 to 0xFFFFFE. Code storage is located at the base of the memory map and supports up to 12 Kbytes (4K instruction words). Table 2-1 shows the location and program memory size of each device.

TABLE 2-1: CODE MEMORY MAP AND SIZE

dsPIC30F SMPS Device	Code Memory Map (Size in Instruction Words)
dsPIC30F1010	0x000000-0x000FFE (2K)
dsPIC30F2020	0x000000-0x001FFE (4K)
dsPIC30F2023	0x000000-0x001FFE (4K)

Locations 0x800000 through 0x8005BE are reserved for executive code memory. This region stores either the programming executive or the debugging executive. The programming executive is used for device programming, while the debug executive is used for in-circuit debugging. This region of memory cannot be used to store user code.

Locations 0xF80000 through 0xF8000E are reserved for the Configuration registers. The bits in these registers may be set to select various device options, and are described in **Section 5.7 "Configuration Bits Programming"**. The Configuration bits read out normally, even after code protection is applied.

Locations 0xFF0000 and 0xFF0002 are reserved for the Device ID registers. These bits can be used by the programmer to identify which device type is being programmed and are described in **Section 10.0 "Device ID"**. The device ID reads out normally, even after code protection is applied.

Figure 2-2 illustrates the memory map for the dsPIC30F SMPS devices.

2.4 Data EEPROM Memory

The dsPIC30F SMPS family has no data EEPROM.

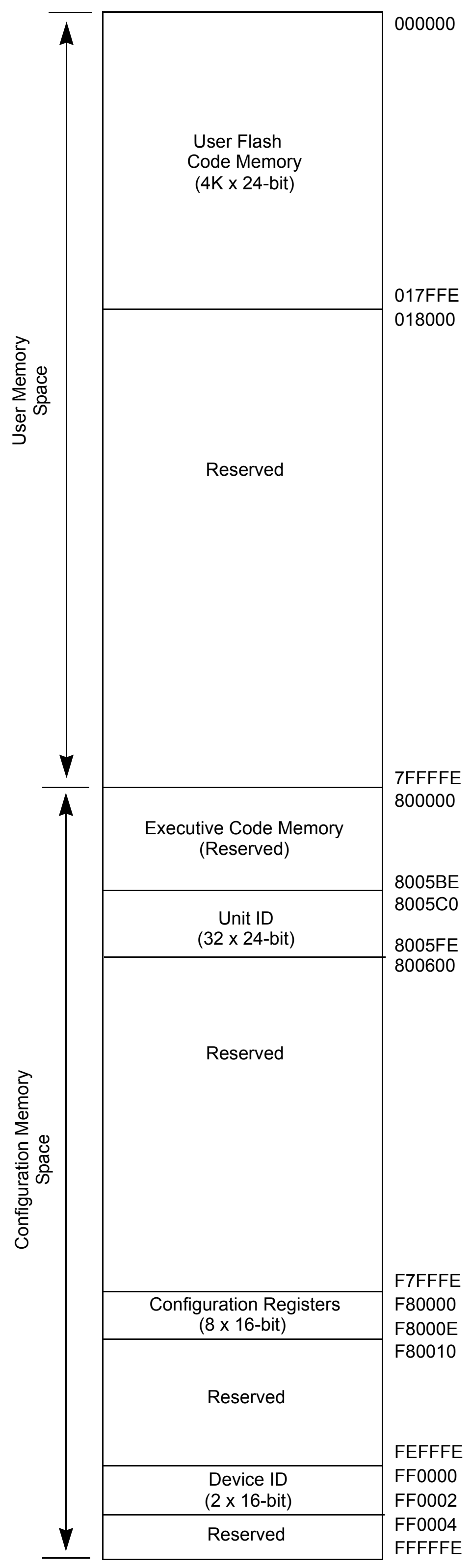
TABLE 2-2: PIN DESCRIPTIONS (PINS USED DURING PROGRAMMING)

Pin Name	During Programming		
	Pin Name	Pin Type	Pin Description
$\overline{\text{MCLR}}$	$\overline{\text{MCLR}}$	P	Programming Enable
VDD and AVDD[1]	VDD	P	Power Supply
VSS and AVSS[1]	VSS	P	Ground
PGC	PGC	I	Primary Programming Pin Pair: Serial Clock
PGD	PGD	I/O	Primary Programming Pin Pair: Serial Data
PGC1	PGC1	I	Secondary Programming Pin Pair: Serial Clock
PGD1	PGD1	I/O	Secondary Programming Pin Pair: Serial Data
PGC2	PGC2	I	Tertiary Programming Pin Pair: Serial Clock
PGD2	PGD2	I/O	Tertiary Programming Pin Pair: Serial Data

Legend: I = Input, O = Output, P = Power

Note 1: All power supply and ground pins must be connected, including analog supplies (AVDD) and ground (AVSS).

FIGURE 2-2: PROGRAM MEMORY MAP



Note: The address boundaries for user Flash code memory are device specific.

3.0 PROGRAMMING EXECUTIVE APPLICATION

3.1 Programming Executive Overview

The programming executive resides in executive 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 dsPIC30F SMPS, using a simple command set and communication protocol.

The following capabilities are provided by the programming executive:

- Read memory
  - Code memory
  - Configuration registers
  - Device ID
- Erase memory
  - Bulk Erase by segment
  - Code memory (by row)
- Program memory
  - Code memory
  - Configuration registers
- Query
  - Blank Device
  - Programming executive software version

The programming executive performs the low-level tasks required for erasing and programming. This allows the programmer to program the device by issuing the appropriate commands and data.

The programming procedure is outlined in **Section 5.0 "Device Programming"**.

3.2 Programming Executive Code Memory

The programming executive is stored in executive code memory and executes from this reserved region of memory. It requires no resources from user code memory.

3.3 Programming Executive Data RAM

The programming executive uses the device's data RAM for variable storage and program execution. Once the programming executive is run, no assumptions should be made about the contents of data RAM.

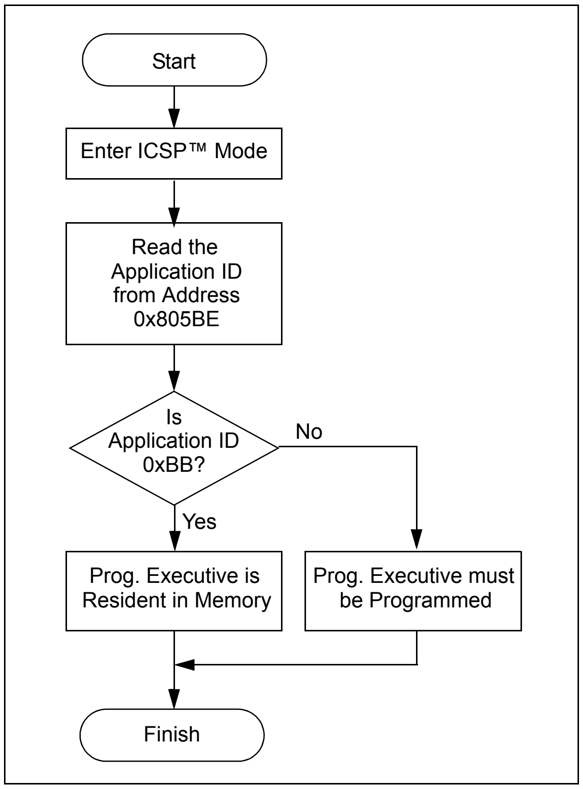
4.0 CONFIRMING THE CONTENTS OF EXECUTIVE MEMORY

The programmer must confirm that the programming executive is stored in executive memory, before the programming is begun. The procedure for this task is illustrated in Figure 4-1.

First, In-Circuit Serial Programming mode (ICSP) is entered. The unique application ID word stored in executive memory is then read. If the programming executive is resident, the application ID word is 0xBB, which means programming can resume as normal. However, if the application ID word is not 0xBB, the programming executive must be programmed to Executive Code memory using the method described in **Section 12.0 "Programming the Programming Executive to Memory"**.

Section 11.0 "ICSP™ Mode" describes the process for the ICSP programming method. **Section 11.11 "Reading the Application ID Word"** describes the procedure to read the application ID word in ICSP mode.

FIGURE 4-1: CONFIRMING PRESENCE OF PROGRAMMING EXECUTIVE


Start
Enter ICSP™ Mode
Read the Application ID from Address 0x805BE
Is Application ID 0xBB?
No
Yes
Prog. Executive is Resident in Memory
Prog. Executive must be Programmed
Finish

5.0 DEVICE PROGRAMMING

5.1 Overview of the Programming Process

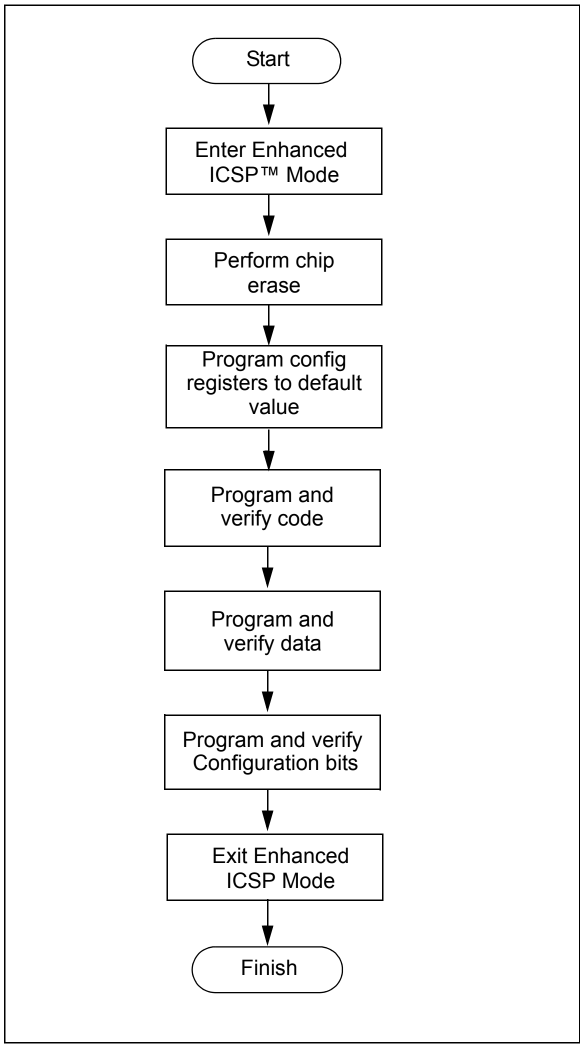
Once the programming executive has been verified in memory (or loaded if not present), the dsPIC30F SMPS can be programmed using the command set shown in Table 5-1. A detailed description for each command is provided in **Section 8.0 "Programming Executive Commands"**.

TABLE 5-1: COMMAND SET SUMMARY

Command	Description
`SCHECK`	Sanity check
`READD`	Read Configuration registers and device ID
`READP`	Read code memory
`PROGP`	Program one row of code memory and verify
`PROGC`	Program Configuration bits and verify
`ERASEB`	Bulk Erase or Segment Erase
`ERASEP`	Erase code memory
`QBLANK`	Query if the code memory is blank
`QVER`	Query the software version

A high-level overview of the programming process is illustrated in Figure 5-1. The process begins by entering Enhanced ICSP mode. The chip is then bulk erased, which clears all memory to '1' and allows the device to be programmed. The Chip Erase is verified before programming begins. Next, the code memory, data Flash and Configuration bits are programmed. As these memories are programmed, they are each verified to ensure that programming was successful. If no errors are detected, the programming is complete and Enhanced ICSP mode is exited. If any of the verifications fail, the procedure should be repeated, starting from the Chip Erase.

FIGURE 5-1: PROGRAMMING FLOW


Start
Enter Enhanced ICSP™ Mode
Perform chip erase
Program config registers to default value
Program and verify code
Program and verify data
Program and verify Configuration bits
Exit Enhanced ICSP Mode
Finish

5.2 Entering Enhanced ICSP Mode

Figure 5-2 shows the three steps required to enter Enhanced ICSP Program/Verify mode:

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

The programming voltage applied to MCLR is VIH, which is essentially VDD in the case of dsPIC30F SMPS devices. There is no minimum time requirement for holding at VIH. After VIH is removed, an interval of at least P16 must elapse before presenting the key sequence on PGD.

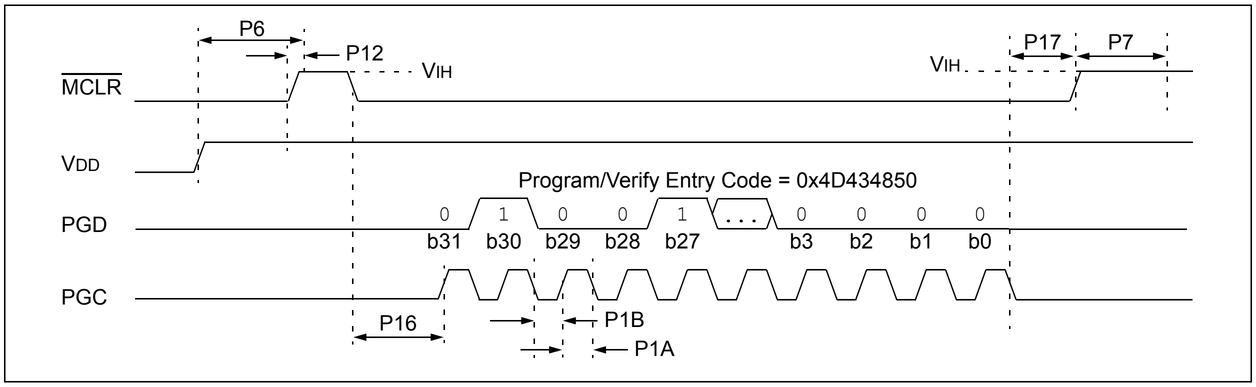
The key sequence is a specific 32-bit pattern, '`0100 1101 0100 0011 0100 1000 0101 0000`' (more easily remembered as 0x4D434850 in hexadecimal format). See **Appendix A: "Hex File Format"** for more information. The device enters Program/Verify mode only if the key sequence is valid. The Most Significant bit (MSb) of the most significant nibble must be shifted in first.

The key data is clocked on the rising edge of the clock PGC. Once the key sequence is complete, VIH must be applied to MCLR and held at that level for as long as Program/Verify mode has to be maintained. An interval of at least time P17 and P7 must elapse before presenting data on PGD. Signals appearing on PGD before P7 has elapsed will not be interpreted as valid.

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

Note: When in Enhanced ICSP mode, the SPI output pin, SDO1, will toggle while the device is being programmed.

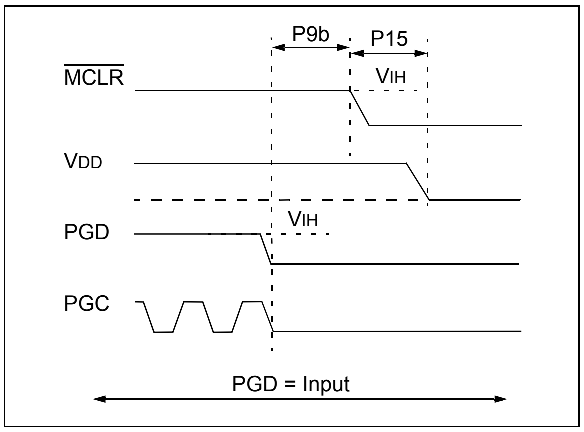
FIGURE 5-2: ENTERING ENHANCED ICSP™ MODE


P6
P12
VIH
MCLR
VIH
P17
P7
VDD
Program/Verify Entry Code = 0x4D434850
PGD
0 1 0 0 1 ... 0 0 0 0
b31 b30 b29 b28 b27 b3 b2 b1 b0
PGC
P16
P1B
P1A

5.3 Exiting Enhanced ICSP Mode

Exiting Program/Verify mode is done by removing VIH from MCLR, as illustrated in Figure 5-3. The only requirement for exit is that an interval P9b should elapse between the last clock and program signals on PGC and PGD before removing VIH.

FIGURE 5-3: EXITING ENHANCED ICSP™ MODE



5.4 Chip Erase

Before a chip can be programmed, it must be erased. The Bulk Erase command (`ERASEB`) is used to perform this task. Executing this command with the MS command field set to 0x3 erases all code memory and code-protect Configuration bits. The Chip Erase process sets all bits in these three memory regions to '`1`'.

Since code protection Configuration bits are not erasable, they must be manually set to '`1`' using multiple `PROGC` commands. One `PROGC` command must be sent for each Configuration register (see **Section 5.7 "Configuration Bits Programming"**).

Note: The Device ID registers cannot be erased. These registers remain intact after a Chip Erase is performed.

5.5 Blank Check

The term “Blank Check” means to verify whether the device has been successfully erased and has no programmed memory cells. A blank or erased memory cell reads as a ‘1’. The following memories must be blank checked:

- All implemented code memory
- All Configuration bits (for their default value)

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

The QBLANK command is used for the Blank Check. It determines if the code memory is erased by testing these memory regions. A ‘BLANK’ or ‘NOT BLANK’ response is returned. The READD command is used to read the Configuration registers. If it is determined that the device is not blank, it must be erased (see **Section 5.4 “Chip Erase”**) before attempting to program the chip.

5.6 Code Memory Programming

5.6.1 OVERVIEW

The panel architecture for the Flash code memory array consists of up to 128 rows of thirty-two, 24-bit instructions. Each panel stores up to 4K instruction words. Each dsPIC30F SMPS device has one memory panel (see Table 5-2).

TABLE 5-2: DEVICE CODE MEMORY SIZE

dsPIC30F SMPS Device	Code Size (24-bit Words)	Number of Rows	Number of Panels
dsPIC30F1010	2K	64	1
dsPIC30F2020	4K	128	1
dsPIC30F2023	4K	128	1

5.6.2 PROGRAMMING METHODOLOGY

Code memory is programmed with the PROGP command. PROGP programs one row of code memory to the memory address specified in the command. The number of PROGP commands required to program a device depends on the number of rows that must be programmed in the device.

A flowchart for programming of code memory is illustrated in Figure 5-4. In this example, all 4K instruction words of a dsPIC30F2020 device are programmed. First, the number of commands to send (called ‘RemainingCmds’ in the flowchart) is set to 128 and the destination address (called ‘BaseAddress’) is set to ‘0’.

Next, one row in the device is programmed with a PROGP command. Each PROGP command contains data for one row of code memory. After the first command is processed successfully, ‘RemainingCmds’ is decremented by ‘1’ and compared to ‘0’. Since there are more PROGP commands to send, ‘BaseAddress’ is incremented by 0x40 to point to the next row of memory.

On the second PROGP command, the second row of each memory panel is programmed. This process is repeated until the entire device is programmed. No special handling must be performed when a panel boundary is crossed.

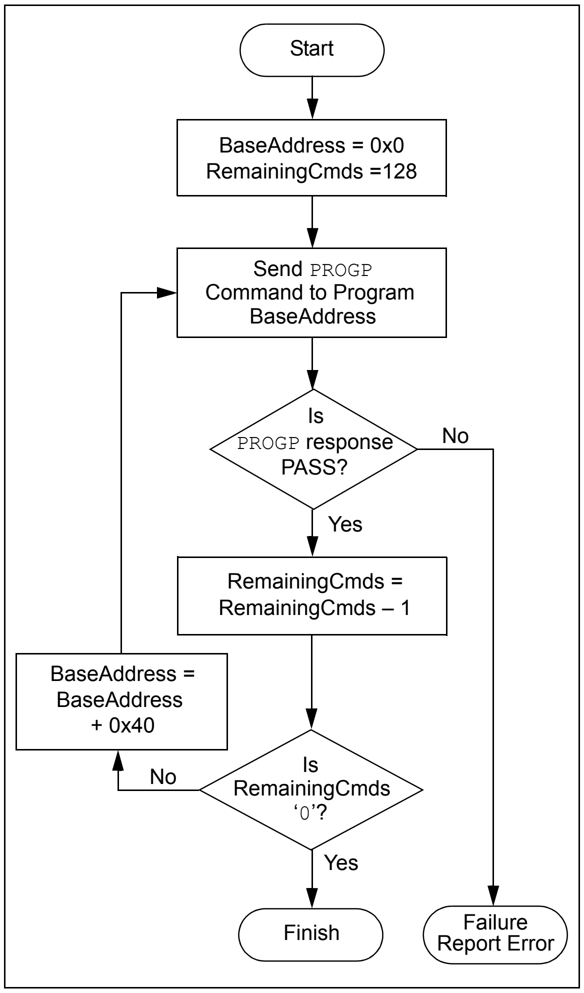
5.6.3 PROGRAMMING VERIFICATION

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

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

Alternatively, you can have the programmer perform the verification once the entire device is programmed using a checksum computation, as described in **Section 6.6 “Checksum Computation”**.

FIGURE 5-4: FLOWCHART FOR PROGRAMMING dsPIC30F SMPS CODE MEMORY



5.7 Configuration Bits Programming

5.7.1 OVERVIEW

The dsPIC30F SMPS has Configuration bits stored in seven 16-bit registers. 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 Watchdog Timer. The code-protect bits prevent program memory from being read and written.

Table 5-3 shows the Configuration registers for the SMPS devices, and Table 5-4 describes the individual bits.

Note: If user software performs an erase operation on the configuration fuse, it must be followed by a write operation to this fuse with the desired value, even if the desired value is the same as the state of the erased fuse.

TABLE 5-3: dsPIC30F SMPS FAMILY DEVICE CONFIGURATION REGISTER MAP

Name	Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FBS	F80000	—	—	—	—	BSS<2:0>			BWRP
Reserved	F80002	—	—	—	—	—	—	—	—
FGS	F80004	—	—	—	—	—	GSS<1:0>		GWRP
FOSCSEL	F80006	—	—	—	—	—	—	FNOSC<1:0>	
FOSC	F80008	FCKSM<1:0>		FRANGE	—	—	OSCIOFNC	POSCMD<1:0>	
FWDT	F8000A	FWDTEN	WINDIS	—	WDTPRE	WDTPOST<3:0>			
FPOR	F8000C	—	—	—	—	—	FPWRT<2:0>		
FICD	F8000E	BKBUG	—	—	—	—	—	ICS<1:0>	

Legend: — = unimplemented bit, read as '0'

Note 1: Refer to "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

TABLE 5-4: CONFIGURATION BITS DESCRIPTION FOR dsPIC30F1010/2020/2023 DEVICES

Bit Field	Register	Description
BSS<2:0>	FBS	**Boot Segment Program Memory Code Protection** `111` = No Boot Segment `110` = Standard security, Small-sized Boot Program Flash [Boot Segment ends at 0x0003FF] `101` = Standard security, Medium-sized Boot Program Flash [Boot Segment ends at 0x000FFF **Note:** This is for the dsPIC30F2020 and dsPIC30F2023 only.] `100` = No Boot Segment `011` = No Boot Segment `010` = High security, Small-sized Boot Program Flash [Boot Segment ends at 0x0003FF] `001` = High security, Medium-sized Boot Program Flash [Boot Segment ends at 0x000FFF **Note:** This is for the dsPIC30F2020 and dsPIC30F2023 only.] `000` = No Boot Segment
BWRP	FBS	**Boot Segment Program Memory Write Protection** `1` = Boot Segment program memory is not write-protected `0` = Boot program memory is write-protected
GSS<1:0>	FGS	**General Segment Code-Protect bit** `11` = Code protection is disabled `10` = Standard security code protection is enabled `0x` = Reserved
GWRP	FGS	**General Segment Write-Protect bit** `1` = General Segment program memory is not write-protected `0` = General Segment program memory is write-protected
FNOSC<1:0>	FOSCSEL	**Initial Oscillator Source Selection bits** `11` = Primary (HS, EC) oscillator with PLL `10` = Primary (HS, EC) oscillator `01` = Internal Fast RC (FRC) oscillator with PLL `00` = Internal Fast RC (FRC) oscillator
FCKSM<1:0>	FOSC	**Clock Switching Mode bits** `1x` = Clock switching is disabled, Fail-Safe Clock Monitor is disabled `01` = Clock switching is enabled, Fail-Safe Clock Monitor is disabled `00` = Clock switching is enabled, Fail-Safe Clock Monitor is enabled
FRANGE	FOSC	**Frequency Range Selection for FRC oscillator** `1` = High Range: nominal FRC frequency is 14.1 MHz `0` = Low Range: nominal FRC frequency is 9.7 MHz
OSCIOFNC	FOSC	**OSC2 Pin Function bit (except in HS mode)** `1` = OSC2 is clock output `0` = OSC2 is general purpose digital I/O pin
POSCMD<1:0>	FOSC	**Primary Oscillator Mode Select bits** `11` = Primary oscillator disabled `10` = HS crystal oscillator mode `01` = Reserved `00` = EC (external clock) mode
FWDTEN	FWDT	**Watchdog Enable bit** `1` = Watchdog always enabled (LPRC oscillator cannot be disabled. Clearing the SWDTEN bit in the RCON register will have no effect) `0` = Watchdog enabled/disabled by user software (LPRC can be disabled by clearing the SWDTEN bit in the RCON register)
WINDIS	FWDT	**Watchdog Timer Window Enable bit** `1` = Watchdog Timer in Non-Window mode `0` = Watchdog Timer in Window mode

TABLE 5-4: CONFIGURATION BITS DESCRIPTION FOR dsPIC30F1010/2020/2023 DEVICES (CONTINUED)

Bit Field	Register	Description
WDTPRE	FWDT	**Watchdog Timer Prescaler bit** `1` = 1:128 `0` = 1:32
WDTPOST<3:0>	FWDT	**Watchdog Timer Postscaler bits** `1111` = 1:32,768 `1110` = 1:16,384 • • • `0001` = 1:2 `0000` = 1:1
FPWRT<2:0>	FPOR	**Power-on Reset Timer Value Select bits** `111` = PWRT = 128 ms `110` = PWRT = 64 ms `101` = PWRT = 32 ms `100` = PWRT = 16 ms `011` = PWRT = 8 ms `010` = PWRT = 4 ms `001` = PWRT = 2 ms `000` = PWRT Disabled
BKBUG	FICD	**Background Debug Enable bit** `1` = Device will reset in User mode `0` = Device will reset in Debug mode
ICS<1:0>	FICD	**ICD Communication Channel Select bits** `11` = Communicate on PGC/EMUC and PGD/EMUD `10` = Communicate on PGC1/EMUC1 and PGD1/EMUD1 `01` = Communicate on PGC2/EMUC2 and PGD2/EMUD2 `00` = Reserved, do not use
—	All	Unimplemented (read as ‘`0`’, write as ‘`0`’)

5.7.2 PROGRAMMING METHODOLOGY

System operation Configuration bits are inherently different than all other memory cells. Unlike code memory and code-protect Configuration bits, the system operation bits cannot be erased. If the chip is erased with the `ERASEB` command, the system operation bits retain their previous value. Consequently, you should make no assumption about the value of the system operation bits. They should always be programmed to their desired setting.

Configuration bits are programmed single word at a time using the `PROGC` command. The `PROGC` command specifies the configuration data and Configuration register address. When Configuration bits are programmed, any unimplemented bits must be programmed with a '`0`', and any reserved bits must be programmed with a '`1`'.

Four `PROGC` commands are required to program all the Configuration bits. A flowchart for Configuration bit programming is illustrated in Figure 5-5.

Note: If the General Code Segment Code Protect bits (GSS<1:0>) are programmed to '`10`', code memory is code-protected and cannot be read. Code memory must be verified before enabling read protection. See **Section 5.7.4 "CodeGuard™ Security Configuration Bits"** for more information about code-protect Configuration bits.

5.7.3 PROGRAMMING VERIFICATION

Once the Configuration bits are programmed, the contents of memory should be verified to ensure that the programming is successful. Verification requires the Configuration bits to be read back and compared against the copy held in the programmer's buffer. The `READD` command reads back the programmed Configuration bits and verifies that the programming was successful.

Any unimplemented Configuration bits are read-only and read as '`0`'.

5.7.4 CodeGuard™ SECURITY CONFIGURATION BITS

The FBS and FGS Configuration registers are special Configuration registers that control the size and level of code protection for the Boot Segment and General Segment, respectively. For each segment, two main forms of code protection are provided. One form prevents code memory from being written (write protection), while the other prevents code memory from being read (read protection). The dsPIC30F SMPS family devices do not contain a Secure Segment.

BWRP and GWRP bits control write protection and BSS<2:0> and GSS<1:0> bits control read protection. The Chip Erase `ERASEB` command sets all the code protection bits to '`1`', which allows the device to be programmed.

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

It is imperative that all code protection bits are '`1`' while the device is being programmed and verified. Only after the device is programmed and verified should any of the above bits be programmed to '`0`'.

Note: All bits in the FBS and FGS Configuration registers can only be programmed to a value of '`0`'. The `ERASEB` command (and also the General Segment Erase in the case of FGS) is the only way to reprogram code-protect bits from ON ('`0`') to OFF ('`1`').

Before performing any segment erase operation, the programmer must first determine whether the dsPIC30F device has defined a Boot Segment, and ensure that a segment does not get overwritten by operations on any other segment.

The BSS bit field in the FBS Configuration register can be read to determine whether a Boot Segment has been defined. If a Boot Segment has already been defined (and has probably already been programmed), the user must be warned about this fact.

A Bulk Erase operation is the recommended mechanism to allow a user to overwrite the Boot Segment (if one chooses to do so).

In general, the segments and CodeGuard Security related Configuration registers should be programmed in the following order:

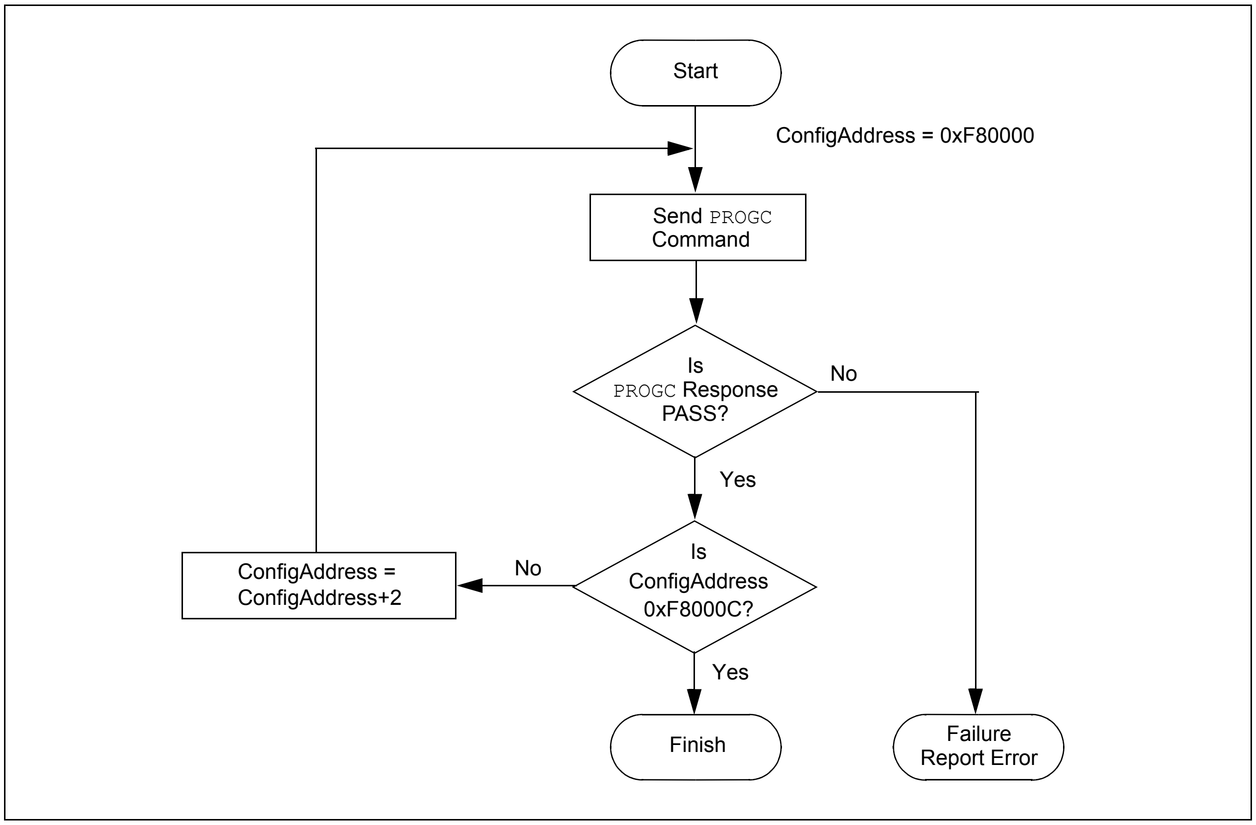
- FBS and Boot Segment
- FGS and General Segment

Note: If any of the code-protect bits in FBS or FGS is clear, then the entire device must be erased before it can be reprogrammed.

5.7.5 USER UNIT ID

The dsPIC30F SMPS devices contain 32 instructions of Unit ID. These are located at addresses 0x8005C0 through 0x8005FF. The Unit ID can be used for storing product information such as serial numbers, system manufacturing dates, manufacturing lot numbers and other such application-specific information. Programming the UNIT ID is similar to programming the Programming Executive (see **Section 12.0 "Programming the Programming Executive to Memory"** for details).

FIGURE 5-5: CONFIGURATION BIT PROGRAMMING FLOW



6.0 OTHER PROGRAMMING FEATURES

6.1 Erasing Memory

Memory is erased by using an `ERASEB` or `ERASEP` command, as detailed in **Section 8.5 "Command Descriptions"**. Code memory can be erased by row using the `ERASEP` command. When memory is erased, the affected memory locations are set to '`1`'s.

The `ERASEB` command provides several Bulk Erase options. Performing a Chip Erase with the `ERASEB` command clears all code memory and code protection registers. Alternatively, the `ERASEB` command can be used to selectively erase individual program memory segments. Table 6-1 summarizes the Erase options.

TABLE 6-1: ERASE OPTIONS

Command	Affected Region
`ERASEB`	Entire chip(1) or all code memory
`ERASEP`(2)	Specified rows of code memory

Note 1: The system operation Configuration registers and device ID registers are not erasable.

2: `ERASEP` cannot be used to erase code-protect Configuration bits. These bits must be erased using `ERASEB`.

6.2 Modifying Memory

Instead of bulk-erasing the device before starting to program, it is possible that you may want to modify only a section of an already programmed device. In this situation, Chip Erase is not a realistic option.

Instead, you can erase selective rows of code memory using the `ERASEP` command. You can then reprogram the modified rows with the `PROGP` command pairs.

In these cases, when code memory is programmed, single-panel programming must be specified in the `PROGP` command.

For modification of code-protect bits, the entire chip must first be erased with the `ERASEB` command. The code-protect bits can be reprogrammed using the `PROGC` command.

Note: If read or write code protection is enabled, no modifications can be made to any region of code memory until code protection is disabled. Code protection can only be disabled by performing a Chip Erase with the `ERASEB` command.

6.3 Reading Memory

The `READD` command reads the Configuration bits and device ID of the device. This command only returns 16-bit data and operates on 16-bit registers.

The `READP` command reads the code memory of the device. This command only returns 24-bit data packed as described in **Section 8.3 "Packed Data Format"**. The `READP` command can be used to read up to 32K instruction words of code memory (only 4K instruction words are present on the dsPIC30F SMPS devices).

Note: Reading an unimplemented memory location causes the programming executive to reset. All `READD` and `READP` commands **must** specify only valid memory locations.

6.4 Programming Executive Software Version

At times, it may be necessary to determine the version of programming executive stored in executive memory. The `QVER` command performs this function. See **Section 8.5.9 "QVER Command"** for details of `QVER` command.

6.5 Configuration Information in the Hex File

To allow portability of code, the programmer must read the Configuration register locations from the hex file (see **Appendix A: "Hex File Format"**). If configuration information is not present in the hex file, a simple warning message should be issued by the programmer. Similarly, while saving a hex file, all configuration information must be included. An option can be provided not to include the configuration information.

Microchip Technology Inc. feels strongly that this feature is important for the benefit of the end customer.

6.6 Checksum Computation

Checksums for the dsPIC30F SMPS are 16 bits in size. The checksum is calculated by summing the following:

- Contents of code memory locations
- Contents of Configuration registers

All memory locations are summed one byte at a time, using only their native data size. More specifically, configuration and device ID registers are summed by adding the lower two bytes of these locations (the upper byte is ignored), while code memory is summed by adding all three bytes of code memory.

Table 6-2 shows how this 16-bit computation can be made for each dsPIC30F SMPS device. Computations for read code protection are shown both enabled and disabled. The checksum values assume that the Configuration registers are also erased. However, when code protection is enabled, the value of the FGS register is assumed to be 0x0.

Note: The checksum calculation differs depending on the code-protect setting. Table 6-2 describes how to compute the checksum for an unprotected device and a read-protected device. Regardless of the code-protect setting, the Configuration registers can always be read.

TABLE 6-2: CHECKSUM COMPUTATION

SMPS Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAAAAAA at 0x0 and Last Code Address
dsPIC30F1010	Disabled	CFGB + SUM(0:000FFF)	0xEA69	0xE86B
	Enabled	CFGB	0x251	0x251
dsPIC30F2020	Disabled	CFGB + SUM(0:001FFF)	0xD269	0xD06B
	Enabled	CFGB	0x251	0x251
dsPIC30F2023	Disabled	CFGB + SUM(0:001FFF)	0xD269	0xD06B
	Enabled	CFGB	0x251	0x251

Item Description:

SUM(a:b) = Byte sum of locations a to b inclusive (all 3 bytes of code memory)

CFGB = **Configuration Block (masked)** = Byte sum of ((FBS&0x000F)+(FGS&0x0007)+(FOSCSEL&0x0003)+(FOSC&0x00E7)+(FWDT&0x00DF)+(FPOR(0x0007))+(FICD(0x0083))

7.0 PROGRAMMER – PROGRAMMING EXECUTIVE COMMUNICATION

7.1 Communication Overview

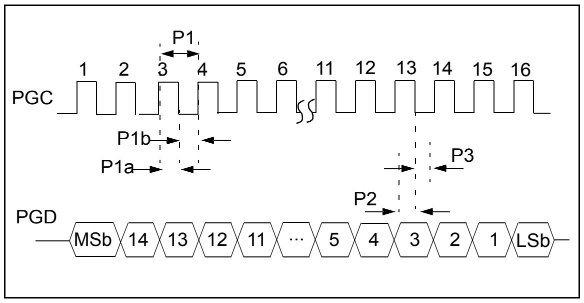
The programmer and programming executive have a master-slave relationship, where the programmer is the master programming device and the programming executive is the slave.

All communication is initiated by the programmer in the form of a command. Only one command at a time can be sent to the programming executive. In turn, the programming executive only sends one response to the programmer after receiving and processing a command. The programming executive command set is described in **Section 8.0 "Programming Executive Commands"**. The response set is described in **Section 9.0 "Programming Executive Responses"**.

7.2 Communication Interface and Protocol

The Enhanced ICSP interface is a 2-wire SPI interface implemented using the PGC and PGD pins. The PGC pin is used as a clock input pin, and the clock source must be provided by the programmer. The PGD pin is used for sending command data to, and receiving response data from the programming executive. All serial data is transmitted on the rising edge of PGC and is latched on the falling edge of PGC. All data transmissions are sent to the Most Significant bit first, using 16-bit mode (see Figure 7-1).

FIGURE 7-1: PROGRAMMING EXECUTIVE SERIAL TIMING



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

After the programming executive has processed the command, it brings PGD low for 15 µsec to indicate to the programmer that the response is available to be clocked out. The programmer can begin to clock out the response 20 µsec after PGD is brought low, and it must provide the necessary amount of clock pulses to receive the entire response from the programming executive.

Once the entire response is clocked out, the programmer should terminate the clock on PGC until it is time to send another command to the programming executive. This protocol is illustrated in Figure 7-2.

7.3 SPI Rate

In Enhanced ICSP mode, the dsPIC30F SMPS operates from the fast internal RC oscillator FRC, which has a nominal frequency of 10 or 15 MHz. This oscillator frequency yields an effective system clock frequency of 2.5 or 3.75 MHz. Since the SPI module operates in Slave mode, the programmer must limit the SPI clock rate to a frequency not greater than 1 MHz.

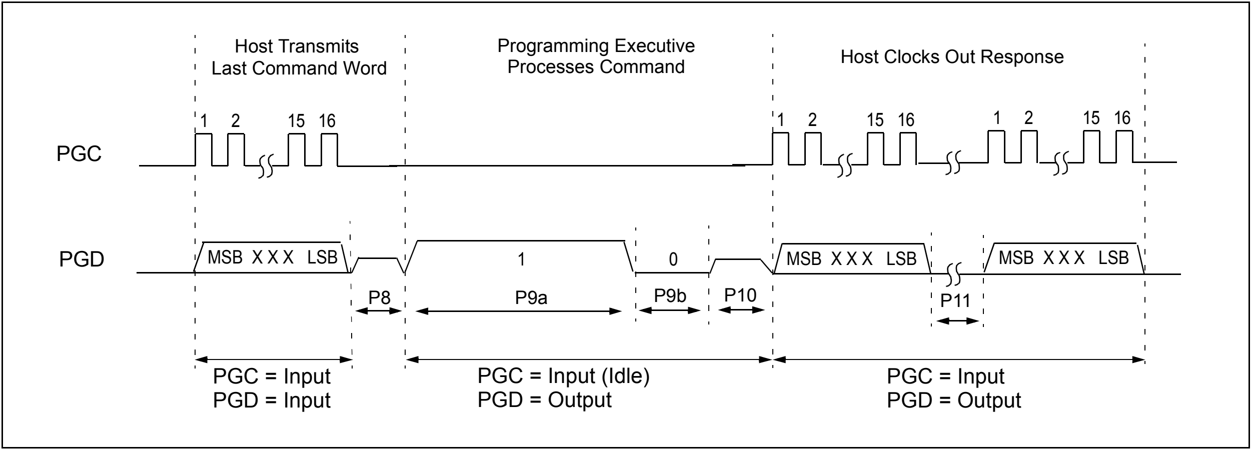
Note: If the programmer provides the SPI with a clock faster than 1 MHz, the behavior of the programming executive will be unpredictable.

7.4 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 PGC, as described in **Section 7.2 "Communication Interface and Protocol"**, it is possible that the programming executive will behave unexpectedly while trying to send a response to the programmer. Since the programming executive has no time out, it is imperative that the programmer correctly follow the described communication protocol.

As a safety measure, the programmer should use the command time outs identified in Table 8-1. If the command time out expires, the programmer should reset the programming executive and start programming the device again.

FIGURE 7-2: PROGRAMMING EXECUTIVE – PROGRAMMER COMMUNICATION PROTOCOL



8.0 PROGRAMMING EXECUTIVE COMMANDS

8.1 Command Set

The programming executive command set is shown in Table 8-1. This table contains the opcode, mnemonic, length, time out and description for each command. Functional details on each command are provided in the command descriptions (**Section 8.5 "Command Descriptions"**).

8.2 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 8-1). 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 8-1: COMMAND FORMAT

15 12	11 0
Opcode	Length
Command Data First Word (if required)	
•	
•	
Command Data Last Word (if required)	

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 8-1 will return a "NACK" response (see **Section 9.2.1 "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.

8.3 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 8-2. This format minimizes traffic over the SPI and provides the programming executive with data that is properly aligned for performing table write operations.

FIGURE 8-2: 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

Note:	When the number of instruction words transferred is odd, MSB2 is zero and lsw2 cannot be transmitted.

8.4 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 9.2.3 "QE_Code Field"**.

TABLE 8-1: PROGRAMMING EXECUTIVE COMMAND SET

Opcode	Mnemonic	Length (16-bit words)	Time Out	Description
0x0	SCHECK	1	1 msec	Sanity check.
0x1	READD	4	1 msec/row	Read N 16-bit words of Configuration registers or device ID starting from specified address.
0x2	READP	4	1 msec/row	Read N 24-bit instruction words of code memory starting from specified address.
0x3	Reserved	N/A	N/A	This command is reserved. It will return a NACK.
0x4	PROGD	19	5 msec	Not implemented.
0x5	PROGP(1)	51	5 msec	Program one row of code memory at the specified address, then verify.
0x6	PROGC	4	5 msec	Write byte or 16-bit word to specified Configuration register.
0x7	ERASEB	2	5 msec	Bulk Erase (entire program memory), or erase by segment.
0x8	ERASED	3	5 msec/row	Not Implemented.
0x9	ERASEP(1)	3	5 msec/row	Erase rows of code memory from specified address.
0xA	QBLANK	3	300 msec	Query if the code memory is blank.
0xB	QVER	1	1 msec	Query the programming executive software version.

Note 1: One row of code memory consists of thirty-two 24-bit words. Refer to Table 5-2 for device-specific information.

8.5 Command Descriptions

All commands supported by the programming executive are described in **Section 8.5.1 "SCHECK Command"** through **Section 8.5.9 "QVER Command"**.

8.5.1 SCHECK COMMAND

15 12	11 0
Opcode	Length

Field	Description
Opcode	0x0
Length	0x1

The `SCHECK` command instructs the programming executive to do nothing, but generate a response. This command is used as a "sanity check" to verify that the programming executive is operational.

Expected Response (2 words):
0x1000
0x0002

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

8.5.2 READD COMMAND

15 12	11 8	7 0
Opcode	Length	
Reserved0	N	
Reserved1		Addr_MSB
Addr_LS		

Field	Description
Opcode	0x1
Length	0x4
Reserved0	0x0
N	Number of 16-bit words to read (max of 2048)
Reserved1	0x0
Addr_MSB	MSB of 24-bit source address
Addr_LS	LS 16 bits of 24-bit source address

The `READD` command instructs the programming executive to read N 16-bit words of memory starting from the 24-bit address specified by Addr_MSB and Addr_LS. This command can only be used to read 16-bit data. It can be used to read Configuration registers and the device ID.

Expected Response (2 + N words):
0x1100
N + 2
Data word 1
...
Data word N

> **Note:** Reading unimplemented memory will cause the programming executive to reset.

8.5.3 READP COMMAND

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

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

The READP command instructs the programming executive to read N 24-bit words of code memory starting from the 24-bit address specified by Addr_MSB and Addr_LS. This command can only be used to read 24-bit data. All data returned in response to this command uses the packed data format described in **Section 8.3 "Packed Data Format"**.

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

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

Note: Reading unimplemented memory will cause the programming executive to reset.

8.5.4 PROGP COMMAND

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

Field	Description
Opcode	0x5
Length	0x33
Reserved	0x0
Addr_MSB	MSB of 24-bit destination address
Addr_LS	LS 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 47
D_48	16-bit data word 48

The PROGP command instructs the programming executive to program one row of code memory (32 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 0x40.

The data to program memory, located in command words D_1 through D_48, must be arranged using the packed instruction word format shown in Figure 8-2.

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

Expected Response (2 words):
0x1500
0x0002

Note: Refer to Table 5-2 for code memory size information.

8.5.5 PROGC COMMAND

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

Field	Description
Opcode	0x6
Length	0x4
Reserved	0x0
Addr_MSB	MSB of 24-bit destination address
Addr_LS	LS 16 bits of 24-bit destination address
Data	Data to program

The PROGC command programs data to the specified Configuration register and verifies whether the programming was successful. Configuration registers are 16 bits wide, and this command allows one Configuration register to be programmed.

Expected Response (2 words):
0x1600
0x0002

Note: This command can only be used for programming Configuration registers.

8.5.6 ERASEB COMMAND

15 12	11	2 0
Opcode	Length	
Reserved		MS

Field	Description
Opcode	0x7
Length	0x2
Reserved	0x0
MS	Select memory to erase: 0x0 = All Code in General Segment 0x1 = Reserved 0x2 = All Code in General Segment, interrupt vectors, and FGS Configuration register 0x3 = Full Chip Erase 0x4 = All Code in Boot and General Segments, and the FBS and FGS Configuration registers 0x5 = All Code in General Segment, and the FGS Configuration register 0x6 = Reserved 0x7 = Reserved

The ERASEB command performs a Bulk Erase. The MS field selects the memory to be bulk erased, with options for erasing individual memory segments.

When Full Chip Erase is selected, the following memory regions are erased:

- All code memory (even if code-protected)
- All code-protect Configuration registers, including Unit ID

Only the executive code memory, device ID and Configuration registers that are not code-protected remain intact after a Full Chip Erase.

Expected Response (2 words):
0x1700
0x0002

8.5.7 ERASEP COMMAND

15 12	11 8	7 0
Opcode	Length	
Num_Rows		Addr_MSB
Addr_LS		

Field	Description
Opcode	0x9
Length	0x3
Num_Rows	Number of rows to erase
Addr_MSB	MSB of 24-bit base address
Addr_LS	LS 16 bits of 24-bit base address

The ERASEP command erases the specified number of rows of code memory from the specified base address. The specified base address must be a multiple of 0x40.

Once the erase is performed, all targeted words of code memory contain 0xFFFFFF.

Expected Response (2 words):
0x1900
0x0002

Note: The ERASEP command cannot be used to erase the Configuration registers or device ID. CodeGuard Security code-protect Configuration registers can only be erased with ERASEB, while the device ID is read-only.

8.5.8 QBLANK COMMAND

15 12	11 0
Opcode	Length
PSize	
Reserved	DSize

Field	Description
Opcode	0xA
Length	0x3
PSize	Length of program memory to check (in 24-bit words), max of 49152
Reserved	0x0
DSize	Length of data memory to check (in 16-bit words), max of 2048

The QBLANK command queries the programming executive to determine whether the contents of code memory is 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 0x0 and advances toward larger addresses for the specified number of instruction words.

The QBLANK command 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):
0x1AF0
0x0002

Expected Response (2 words for non-blank device):
0x1A0F
0x0002

Note: The QBLANK command does not check the system Configuration registers. The READD command must be used to determine the state of the Configuration registers.

8.5.9 QVER COMMAND

15 … 12	11 … 0
Opcode	Length

Field	Description
Opcode	0xB
Length	0x1

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

Expected Response (2 words):
0x1BMN (where "MN" stands for version M.N)
0x0002

9.0 PROGRAMMING EXECUTIVE RESPONSES

9.1 Overview

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

The programming executive response set is shown in Table 9-1. This table contains the opcode, mnemonic and description for each response. The response format is described in **Section 9.2 "Response Format"**.

TABLE 9-1: PROGRAMMING EXECUTIVE RESPONSE SET

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

9.2 Response Format

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

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

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

9.2.1 Opcode FIELD

The Opcode is a 4-bit field in the first word of the response. The Opcode indicates how the command is processed (see Table 9-1). If the command is processed successfully, the response opcode is PASS. If there is 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.

9.2.2 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.

9.2.3 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 9-2.

TABLE 9-2: 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 9-3. 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 verification of the programming for the `PROGD`, `PROGP` or `PROGC` command fails, the QE_Code is set to 0x1. For all other programming executive errors, the QE_Code is 0x2.

TABLE 9-3: QE_Code FOR NON-QUERY COMMANDS

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

9.2.4 RESPONSE LENGTH

The response length indicates the length of the programming executive's response in 16-bit words. This field includes the 2 words of the response header. With the exception of the response for the `READD` and `READP` commands, the length of each response is only 2 words.

The response to the `READD` command is N + 2 words, where N is the number of words specified in the `READD` command.

The response to the `READP` command uses the packed instruction word format described in **Section 8.3 "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.

10.0 DEVICE ID

The device ID region is 2 x 16 bits and can be read using the `READD` command. This region of memory is read-only and can also be read when code protection is enabled.

Table 10-1 shows the device ID for each device, Table 10-2 shows the device ID registers and Table 10-3 describes the bit field of each register.

TABLE 10-1: DEVICE IDs

SMPS Device	DEVID	Silicon Revision				
		A0	A1	A2	A3	A4
dsPIC30F1010	0x0404	—	0x1000	0x1002	0x1003	—
dsPIC30F2020	0x0400	0x1000	0x1001	0x1002	0x1003	0x1004
dsPIC30F2023	0x0403	—	0x1000	0x1002	0x1003	—

TABLE 10-2: 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<15:0>															
0xFF0002	DEVREV	PROC<3:0>				▨	▨	▨	REV<2:0>			▨	▨	▨	DOT<2:0>		

TABLE 10-3: DEVICE ID BITS DESCRIPTION

Bit Field	Register	Description
DEVID<15:0>	DEVID	Encodes the device ID.
PROC<3:0>	DEVREV	Encodes the process of the device (always read as 0x001).
REV<2:0>	DEVREV	Encodes the major revision number of the device.
DOT<2:0>	DEVREV	Encodes the minor revision number of the device.

11.0 ICSP™ MODE

11.1 ICSP Mode

ICSP mode is a special programming protocol that allows you to read and write to dsPIC30F SMPS programming executive. The ICSP mode is the second (and slower) method used to program the device. This mode also has the ability to read the contents of executive memory to determine if the programming executive is present. This capability is accomplished by applying control codes and instructions serially to the device using pins PGC and PGD.

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

Data is transmitted on the rising edge and latched on the falling edge of PGC. For all data transmissions, the Least Significant bit (LSb) is transmitted first.

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

2: On the first serial word (in ICSP mode) to be shifted into the device following a Reset, an additional 5 clocks must be provided to the device on the PGC pin.

3: Because ICSP is slower, it is recommended that only Enhanced ICSP mode be used for device programming, as described in **Section 5.1 "Overview of the Programming Process"**.

11.2 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 is clocked in using PGC and PGD, and this control code is used to command the CPU (see Table 11-1).

The SIX control code is used to send instructions to the CPU for execution, while the REGOUT control code is used to read data out of the device via the VISI register. The operation details of ICSP mode are provided in **Section 11.2.1 "Six Serial Instruction Execution"** and **Section 11.2.2 "REGOUT Serial Instruction Execution"**.

TABLE 11-1: CPU CONTROL CODES IN ICSP™ MODE

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

11.2.1 SIX SERIAL INSTRUCTION EXECUTION

The SIX control code allows execution of dsPIC30F SMPS assembly instructions. When the SIX code is received, the CPU is suspended for 24 clock cycles as the instruction is then clocked into the internal buffer. Once the instruction is shifted in, the state machine allows it to be executed over the next four clock cycles. While the received instruction is executed, the state machine simultaneously shifts in the next 4-bit command (see Figure 11-2).

Note 1: Coming out of Reset, the first 4-bit control code is always forced to SIX, and a forced `NOP` instruction is executed by the CPU. Once the forced SIX is clocked in, ICSP operation resumes as normal (the next 24 clock cycles load the first instruction word to the CPU).

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

11.2.2 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 PGD pin. Once the REGOUT control code is received, eight clock cycles are required to process the command. During this time, the CPU is held idle. After these eight cycles, an additional 16 cycles are required to clock the data out (see Figure 11-3).

The REGOUT instruction is unique because the PGD pin is an input when the control code is transmitted to the device. However, once the control code is processed, the PGD pin becomes an output as the VISI register is shifted out. After the contents of the VISI are shifted out, PGD becomes an input again as the state machine holds the CPU idle until the next 4-bit control code is shifted in.

Note: Once the contents of VISI are shifted out, the dsPIC® DSC device maintains PGD as an output until the first rising edge of the next clock is received.

FIGURE 11-1: PROGRAM ENTRY AFTER RESET

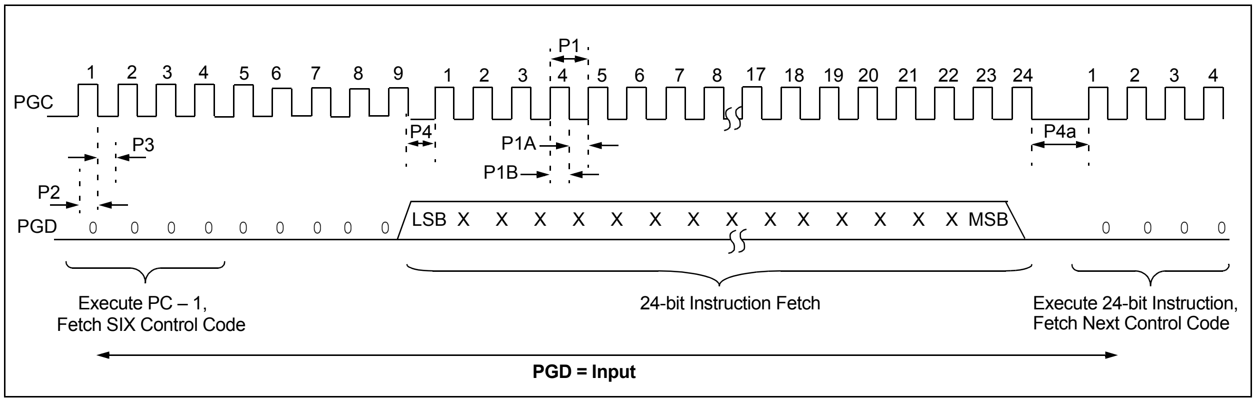


FIGURE 11-2: SIX SERIAL EXECUTION

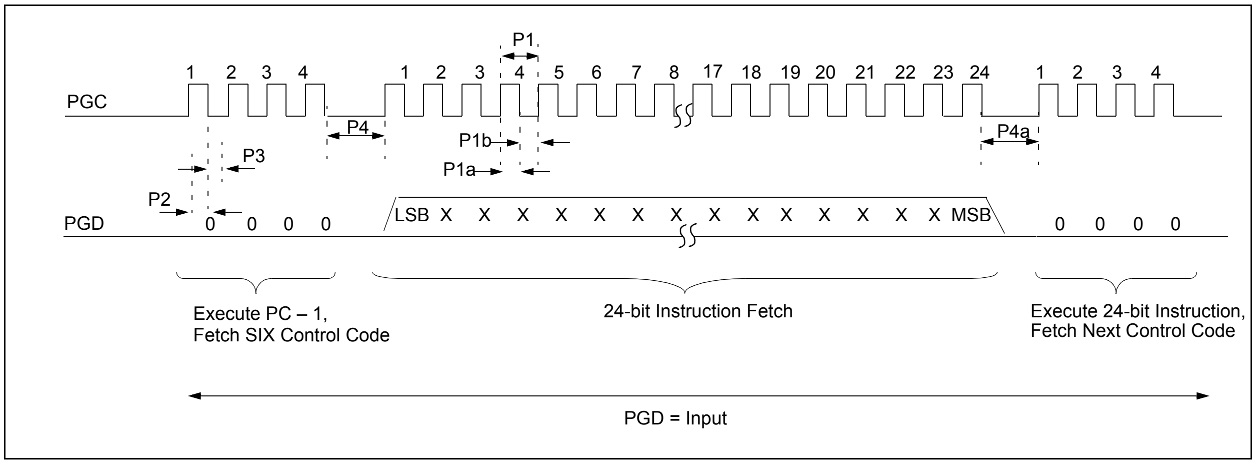
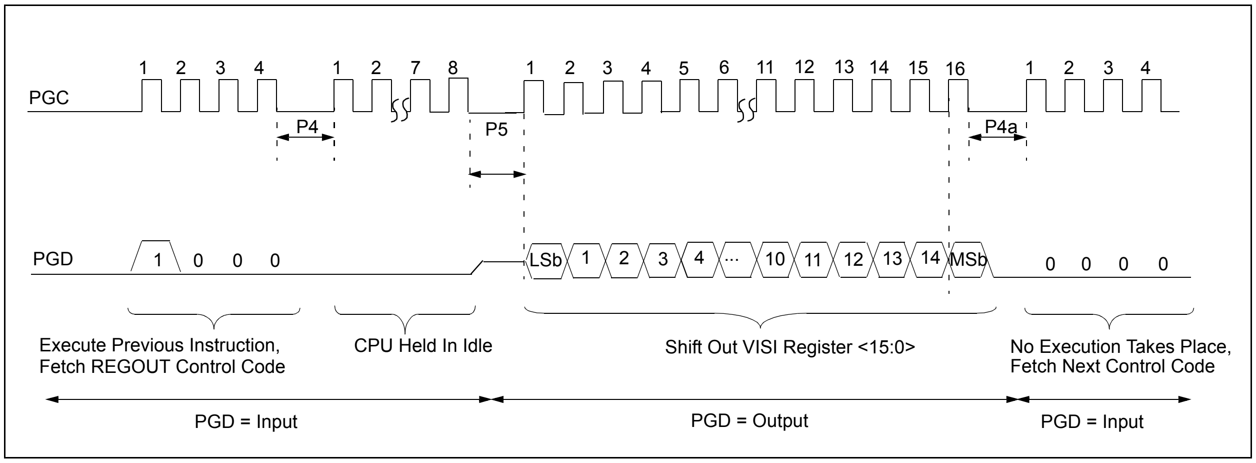


FIGURE 11-3: REGOUT SERIAL EXECUTION



11.3 Entering ICSP Mode

Figure 11-4 shows the three required steps to enter the ICSP Program/Verify mode:

1. $\overline{\text{MCLR}}$ is briefly driven high then low.
2. A 32-bit key sequence is clocked into PGD.
3. $\overline{\text{MCLR}}$ is then driven high within a specified period of time and held.

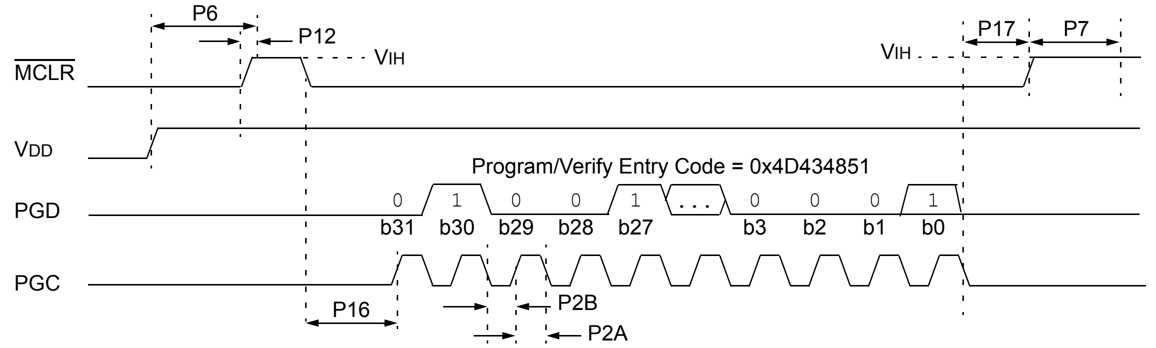
The programming voltage applied to $\overline{\text{MCLR}}$ is VIH, which is essentially VDD in the case of dsPIC30F SMPS devices. There is no minimum time requirement for holding at VIH. After VIH is removed, an interval of at least P16 must elapse before presenting the key sequence on PGD.

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

Once the key sequence is complete, VIH must be applied to $\overline{\text{MCLR}}$ and held at that level for as long as Program/Verify mode has to be maintained. An interval of at least time P17 and P7 must elapse before presenting data on PGD. Signals appearing on PGD before P7 has elapsed will not be interpreted as valid.

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

FIGURE 11-4: ENTERING ICSP™ MODE


P6
P12
VIH
VIH
P17
P7
MCLR
VDD
Program/Verify Entry Code = 0x4D434851
PGD
0 1 0 0 1 ... 0 0 0 1
b31 b30 b29 b28 b27 b3 b2 b1 b0
PGC
P16
P2B
P2A

11.4 Flash Memory Programming in ICSP Mode

Programming in ICSP mode is described in **Section 11.4.1 "Programming Operations"** through **Section 11.4.3 "Starting and Stopping a Programming Cycle"**.

Step-by-step procedures are described in **Section 11.5 "Bulk Erasing Program Memory"** through **Section 11.11 "Reading the Application ID Word"**.

All programming operations must use serial execution, as described in **Section 11.2 "ICSP Operation"**.

11.4.1 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 11-2) or write operation (Table 11-3), writing a key sequence to enable the programming and initiating the programming by setting the WR control bit, NVMCON<15>.

In ICSP mode, all programming operations are externally timed. An external 2 msec delay must be used between setting the WR control bit and clearing the WR control bit to complete the programming operation.

TABLE 11-2: NVMCON ERASE OPERATIONS

NVMCON Value	Erase Operation
0x407F	Erases all code memory, data memory, executive memory and code-protect bits (does not erase UNIT ID).
0x406E	Erases Boot Segment, General Segment and Interrupt Vector Table, then erase FBS and FGS Configuration registers.
0x404E	Erases General Segment, then erase FGS Configuration register.
0x4042	Erases General Segment without erasing Interrupt Vector Table (when Boot Segment is not defined).
0x4072	Erases all executive memory.
0x4071	Erases 1 row (32 instruction words) from 1 panel of code memory.

TABLE 11-3: NVMCON WRITE OPERATIONS

NVMCON Value	Write Operation
0x4008	Writes 1 word to configuration memory.
0x4001	Writes 1 row (32 instruction words) into 1 panel of program memory.

11.4.2 UNLOCKING NVMCON FOR PROGRAMMING

Writes to the WR bit (NVMCON<15>) are locked to prevent accidental programming from taking place. Writing a key sequence to the NVMKEY register unlocks the WR bit and allows it to be written to. The unlock sequence is performed as follows:

```
MOV     #0x55, W8
MOV     W8, NVMKEY
MOV     #0xAA, W9
MOV     W9, NVMKEY
```

Note: Any working register or working register pair can be used to write the unlock sequence.

11.4.3 STARTING AND STOPPING A PROGRAMMING CYCLE

Once the unlock key sequence has been written to the NVMKEY register, the WR bit (NVMCON<15>) is used to start and stop an erase or write cycle. Setting the WR bit initiates the programming cycle. Clearing the WR bit terminates the programming cycle.

All erase and write cycles must be externally timed. An external delay must be used between setting and clearing the WR bit. Starting and stopping a programming cycle is performed as follows:

```
BSET    NVMCON, #WR
<Wait 2 msec>
BCLR    NVMCON, #WR
```

11.5 Bulk Erasing Program Memory

The procedure for bulk erasing program memory (all code memory, data memory, executive memory and code-protect bits) consists of setting NVMCON to 0x407F, unlocking NVMCON for erasing and then executing the programming cycle.

Table 11-4 shows the ICSP programming process for bulk erasing program memory. This process includes the ICSP command code, which must be transmitted (for each instruction) to the Least Significant bit first using the PGC and PGD pins (see Figure 11-2).

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

TABLE 11-4: SERIAL INSTRUCTION EXECUTION FOR BULK ERASING PROGRAM MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		

`0000`
`0000`
`0000`
	`040100`
	`040100`
	`000000`
		`GOTO 0x100`
		`GOTO 0x100`
		`NOP`

Step 2: Set the NVMCON to erase all Program Memory.		

`00000`
`0000`
	`2407FA`
	`883B0A`
		`MOV #0x407F, W10`
		`MOV W10, NVMCON`

Step 3: Unlock the NVMCON for programming.		

`0000`
`0000`
`0000`
`0000`
	`200558`
	`883B38`
	`200AA9`
	`883B39`
		`MOV #0x55, W8`
		`MOV W8, NVMKEY`
		`MOV #0xAA, W9`
		`MOV W9, NVMKEY`

Step 4: Initiate the erase cycle.		

`0000`
`0000`
`0000`
`0000`
`0000`
—
 	
`0000`
`0000`
`0000`
`0000`
`0000`
	`A8E761`
	`000000`
	`000000`
	`000000`
	`000000`
	—
	
	`A9E761`
	`000000`
	`000000`
	`000000`
	`000000`
		`BSET NVMCON, #WR`
		`NOP`
		`NOP`
		`NOP`
		`NOP`
		`Externally time 'P19a' msec` (see **Section 13.0 "AC/DC Characteristics and Timing Requirements"**)
		`BCLR NVMCON, #WR`
		`NOP`
		`NOP`
		`NOP`
		`NOP`

11.6 Row Erasing Program Memory

Instead of using a Bulk Erase operation, each region of memory can be individually erased by row. In this case, all of the code memory, executive memory and Configuration registers must be erased one row at a time. The procedure for erasing individual rows of program memory is quite different from the procedure for erasing the entire chip. This procedure is detailed in Table 11-5. If a Segment Erase operation is required, Step 3 must be modified with the appropriate NVMCON value as per Table 11-2.

However, since this method is more time consuming and does not clear the code-protect bits, it is not recommended in most cases.

Note 1: Program memory must be erased before writing any data to program memory.

TABLE 11-5: SERIAL INSTRUCTION EXECUTION FOR ROW ERASING PROGRAM MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		

`0000`
`0000`
`0000`
	`040100`
	`040100`
	`000000`
		`GOTO 0x100`
		`GOTO 0x100`
		`NOP`

Step 2: Initialize NVMADR and NVMADRU to erase code memory and initialize W7 for row address updates.		

`0000`
`0000`
`0000`
`0000`
	`EB0300`
	`883B16`
	`883B26`
	`200407`
		`CLR W6`
		`MOV W6, NVMADR`
		`MOV W6, NVMADRU`
		`MOV #0x40, W7`

Step 3: Set NVMCON to erase 1 row of code memory.		

`0000`
`0000`
	`24071A`
	`883B0A`
		`MOV #0x4071, W10`
		`MOV W10, NVMCON`

Step 4: Unlock the NVMCON to erase 1 row of code memory.		

`0000`
`0000`
`0000`
`0000`
	`200558`
	`883B38`
	`200AA9`
	`883B39`
		`MOV #0x55, W8`
		`MOV W8, NVMKEY`
		`MOV #0xAA, W9`
		`MOV W9, NVMKEY`

Step 5: Initiate the erase cycle.		

`0000`
`0000`
`0000`
`0000`
`0000`
—
 	
`0000`
`0000`
`0000`
`0000`
`0000`
	`A8E761`
	`000000`
	`000000`
	`000000`
	`000000`
	—
	
	`A9E761`
	`000000`
	`000000`
	`000000`
	`000000`
		`BSET NVMCON, #WR`
		`NOP`
		`NOP`
		`NOP`
		`NOP`
		`Externally time 'P19a' msec` (see **Section 13.0 “AC/DC Characteristics and Timing Requirements”**)
		`BCLR NVMCON, #WR`
		`NOP`
		`NOP`
		`NOP`
		`NOP`

TABLE 11-5: SERIAL INSTRUCTION EXECUTION FOR ROW ERASING PROGRAM MEMORY (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 6: Update the row address stored in NVMADRU:NVMADR. When W6 rolls over to 0x0, NVMADRU must be incremented.		

0000
0000
0000
0000
	430307
	AF0042
	EC2764
	883B16
		`ADD W6, W7, W6`
		`BTSC SR, #C`
		`INC NVMADRU`
		`MOV W6, NVMADR`

Step 7: Reset device internal PC.		

0000
0000
	040100
	000000
		`GOTO 0x100`
		`NOP`

Step 8: Repeat Steps 3-7 until all rows of code memory are erased.		
Step 9: Initialize NVMADR and NVMADRU to erase executive memory and initialize W7 for row address updates.		

0000
0000
0000
0000
0000
	EB0300
	883B16
	200807
	883B27
	200407
		`CLR W6`
		`MOV W6, NVMADR`
		`MOV #0x80, W7`
		`MOV W7, NVMADRU`
		`MOV #0x40, W7`

Step 10: Set NVMCON to erase 1 row of executive memory.		

0000
0000
	24071A
	883B0A
		`MOV #0x4071, W10`
		`MOV W10, NVMCON`

Step 11: Unlock the NVMCON to erase 1 row of executive memory.		

0000
0000
0000
0000
	200558
	883B38
	200AA9
	883B39
		`MOV #0x55, W8`
		`MOV W8, NVMKEY`
		`MOV #0xAA, W9`
		`MOV W9, NVMKEY`

Step 12: Initiate the erase cycle.		

0000
0000
0000
0000
0000
—
 	
0000
0000
0000
0000
0000
	A8E761
	000000
	000000
	000000
	000000
	—
	
	A9E761
	000000
	000000
	000000
	000000
		`BSET NVMCON, #WR`
		`NOP`
		`NOP`
		`NOP`
		`NOP`
		`Externally time 'P19a' msec` (see **Section 13.0 "AC/DC Characteristics and Timing Requirements"**)
		`BCLR NVMCON, #WR`
		`NOP`
		`NOP`
		`NOP`
		`NOP`

Step 13: Update the row address stored in NVMADR.		

0000
0000
	430307
	883B16
		`ADD W6, W7, W6`
		`MOV W6, NVMADR`

Step 14: Reset device internal PC.		

0000
0000
	040100
	000000
		`GOTO 0x100`
		`NOP`

Step 15: Repeat Steps 10-14 until all 24 rows of executive memory are erased.		

11.7 Writing Configuration Memory

The FOSC, FWDT, FPOR and FICD registers are not erasable. It is recommended that all Configuration registers can be set to a default value after erasing program memory. The FWDT, FPOR and FICD registers can be set to a default all '1's value by programming 0xFFFF to each register. Since these registers contain unimplemented bits that read as '0', the default values shown in Table 11-6 will be read instead of 0xFFFF. The recommended default FOSC value is 0xC100, which selects the FRC clock oscillator setting.

The FBS and FGS Configuration registers are special since they enable code protection for the device. For security purposes, once the bits in these registers are programmed to '0' (to enable code protection), they can only be set back to '1' by performing a Bulk Erase as described in **Section 11.5 "Bulk Erasing Program Memory"**, or by using a Segment Erase operation. Programming the FBS or FGS Configuration register bits from a '0' to '1' is not possible, but they can be programmed from a '1' to a '0' to enable code protection.

Table 11-7 shows the ICSP programming details for clearing the Configuration registers. In Step 1, the Reset vector is exited. In Step 2, the write pointer (W7) is loaded with 0x0000, which is the original destination address (in TBLPAG 0xF8 of program memory). In Step 3, the NVMCON is set to program one Configuration register. In Step 4, the TBLPAG register is initialized to 0xF8 for writing to the Configuration registers. In Step 5, the value to write to each Configuration register (0xFFFF) is loaded to W6. In Step 6, the Configuration register data is written to the write latch using the `TBLWTL` instruction. In Steps 7 and 8, the NVMCON is unlocked for programming and the programming cycle is initiated, as described in **Section 11.4 "Flash Memory Programming in ICSP Mode"**. In Step 9, the internal PC is set to 0x100 as a safety measure to prevent the PC from incrementing into unimplemented memory. Finally, Steps 3-9 are repeated six times until all seven Configuration registers are programmed.

TABLE 11-6: DEFAULT CONFIGURATION REGISTER VALUES

Address	Register	Default Value
0xF80000	FBS	0x000F
0xF80002	RESERVED	0x0000
0xF80004	FGS	0x0007
0xF80006	FOSCSEL	0x0003
0xF80008	FOSC	0x00E7
0xF8000A	FWDT	0x00DF
0xF8000C	FPOR	0x0007
0xF8000E	FICD	0x0083

TABLE 11-7: SERIAL INSTRUCTION EXECUTION FOR WRITING CONFIGURATION REGISTERS

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		

0000
0000
0000
	040100
	040100
	000000
		`GOTO 0x100`
		`GOTO 0x100`
		`NOP`

Step 2: Initialize the write pointer (W7) for the `TBLWT` instruction.		
0000	200007	`MOV #0x0000, W7`
Step 3: Set the NVMCON to program 1 Configuration register.		

0000
0000
	24008A
	883B0A
		`MOV #0x4008, W10`
		`MOV W10, NVMCON`

Step 4: Initialize the TBLPAG register.		

0000
0000
	200F80
	880190
		`MOV #0xF8, W0`
		`MOV W0, TBLPAG`

Step 5: Load the Configuration register data to W6.		
0000	2xxxx0	`MOV #<CONFIG_VALUE>, W6`
Step 6: Write the Configuration register data to the write latch and increment the write pointer.		

0000
0000
0000
	BB1B86
	000000
	000000
		`TBLWTL W6, [W7++]`
		`NOP`
		`NOP`

Step 7: Unlock the NVMCON for programming.		

0000
0000
0000
0000
	200558
	883B38
	200AA9
	883B39
		`MOV #0x55, W8`
		`MOV W8, NVMKEY`
		`MOV #0xAA, W9`
		`MOV W9, NVMKEY`

TABLE 11-7: SERIAL INSTRUCTION EXECUTION FOR WRITING CONFIGURATION REGISTERS (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 8: Initiate the write cycle.		
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
—	—	Externally time 'P18a' msec (see **Section 13.0 "AC/DC Characteristics and Timing Requirements"**)
0000	A9E761	BCLR NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 9: Reset device internal PC.		
0000	040100	GOTO 0x100
0000	000000	NOP
Step 10: Repeat steps 3-9 until all seven Configuration registers are programmed.		

11.8 Writing Code Memory

The procedure for writing code memory is similar to the procedure for clearing the Configuration registers, except that 32 instruction words are programmed at a time. To facilitate this operation, working registers W0:W5 are used as temporary holding registers for the data to be programmed.

Table 11-8 shows the ICSP programming details, including the serial pattern with the ICSP command code, which must be transmitted Least Significant bit first using the PGC and PGD pins (see Figure 11-2). In Step 1, the Reset vector is exited. In Step 2, the NVMCON register is initialized for single panel programming of code memory. In Step 3, the 24-bit starting destination address for programming is loaded into the TBLPAG register and W7 register. The upper byte of the starting destination address is stored to TBLPAG, while the lower 16 bits of the destination address are stored to W7.

To minimize the programming time, the same packed instruction format that the programming executive uses is utilized (Figure 8-2). In Step 4, four packed instruction words are stored to working registers W0:W5 using the `MOV` instruction and the read pointer W6 is initialized. The contents of W0:W5 holding the packed instruction word data is shown in Figure 11-5. In Step 5, eight `TBLWT` instructions are used to copy the data from W0:W5 to the write latches of code memory.

Since code memory is programmed 32 instruction words at a time, Steps 4 and 5 are repeated eight times to load all the write latches (Step 6).

After the write latches are loaded, programming is initiated by writing to the NVMKEY and NVMCON registers in Steps 7 and 8. In Step 9, the internal PC is reset to 0x100. This is a precautionary measure to prevent the PC from incrementing into unimplemented memory when large devices are being programmed. Lastly, in Step 10, Steps 2-9 are repeated until all of code memory is programmed.

FIGURE 11-5: PACKED INSTRUCTION WORDS IN W0:W5

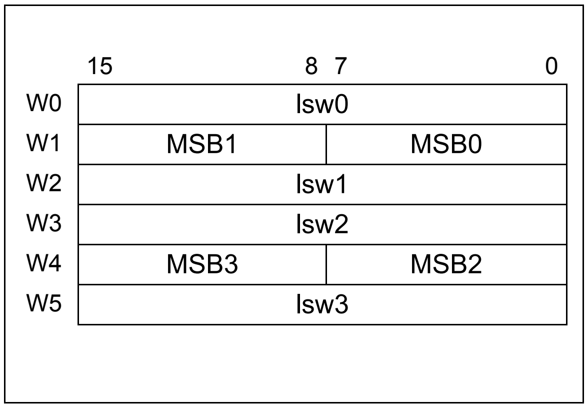


TABLE 11-8: SERIAL INSTRUCTION EXECUTION FOR WRITING CODE MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		

`0000`
`0000`
`0000`
	`040100`
	`040100`
	`000000`
		`GOTO 0x100`
		`GOTO 0x100`
		`NOP`

Step 2: Set the NVMCON to program 32 instruction words.		

`0000`
`0000`
	`24001A`
	`883B0A`
		`MOV #0x4001, W10`
		`MOV W10, NVMCON`

Step 3: Initialize the write pointer (W7) for `TBLWT` instruction.		

`0000`
`0000`
`0000`
	`200xx0`
	`880190`
	`2xxxx7`
		`MOV #<DestinationAddress23:16>, W0`
		`MOV W0, TBLPAG`
		`MOV #<DestinationAddress15:0>, W7`

Step 4: Initialize the read pointer (W6) and load W0:W5 with the next 4 instruction words to program.		

`0000`
`0000`
`0000`
`0000`
`0000`
`0000`
	`2xxxx0`
	`2xxxx1`
	`2xxxx2`
	`2xxxx3`
	`2xxxx4`
	`2xxxx5`
		`MOV #<LSW0>, W0`
		`MOV #<MSB1:MSB0>, W1`
		`MOV #<LSW1>, W2`
		`MOV #<LSW2>, W3`
		`MOV #<MSB3:MSB2>, W4`
		`MOV #<LSW3>, W5`

TABLE 11-8: SERIAL INSTRUCTION EXECUTION FOR WRITING CODE MEMORY (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 5: Set the read pointer (W6) and load the (next set of) write latches.		
`0000`	`EB0300`	`CLR W6`
`0000`	`000000`	`NOP`
`0000`	`BB0BB6`	`TBLWTL [W6++], [W7]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`BBDBB6`	`TBLWTH.B [W6++], [W7++]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`BBEBB6`	`TBLWTH.B [W6++], [++W7]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`BB1BB6`	`TBLWTL [W6++], [W7++]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`BB0BB6`	`TBLWTL [W6++], [W7]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`BBDBB6`	`TBLWTH.B [W6++], [W7++]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`BBEBB6`	`TBLWTH.B [W6++], [++W7]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`BB1BB6`	`TBLWTL [W6++], [W7++]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
Step 6: Repeat steps 4-5 eight times to load the write latches for 32 instructions.		
Step 7: Unlock the NVMCON for writing.		
`0000`	`200558`	`MOV #0x55, W8`
`0000`	`883B38`	`MOV W8, NVMKEY`
`0000`	`200AA9`	`MOV #0xAA, W9`
`0000`	`883B39`	`MOV W9, NVMKEY`
Step 8: Initiate the write cycle.		
`0000`	`A8E761`	`BSET NVMCON, #WR`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
—	—	`Externally time 'P18a' msec` (see **Section 13.0 "AC/DC Characteristics and Timing Requirements"**)
`0000`	`A9E761`	`BCLR NVMCON, #WR`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
Step 9: Reset device internal PC.		
`0000`	`040100`	`GOTO 0x100`
`0000`	`000000`	`NOP`
Step 10: Repeat steps 2-9 until all code memory is programmed.		

11.9 Reading Code Memory

Reading from code memory is performed by executing a series of `TBLRD` instructions and clocking-out the data using the REGOUT command. To ensure efficient execution and facilitate verification on the programmer, four instruction words are read from the device at a time.

Table 11-9 shows the ICSP programming details for reading code memory. In Step 1, the Reset vector is exited. In Step 2, the 24-bit starting source address for reading is loaded into the TBLPAG and W6 registers. The upper byte of the starting source address is stored to TBLPAG, while the lower 16 bits of the source address are stored to W6.

To minimize the reading time, the packed instruction word format that was utilized for writing is also used for reading (see Figure 11-5). In Step 3, the write pointer W7 is initialized, and four instruction words are read from code memory and stored to working registers W0:W5. In Step 4, the four instruction words are clocked out of the device from the VISI register using the REGOUT command. In Step 5, the internal PC is reset to 0x100, as a precautionary measure, to prevent the PC from incrementing into unimplemented memory when large devices are being read. Lastly, in Step 6, Steps 4-5 are repeated until the desired amount of code memory is read.

TABLE 11-9: SERIAL INSTRUCTION EXECUTION FOR READING CODE MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	040100	`GOTO 0x100`
0000	040100	`GOTO 0x100`
0000	000000	`NOP`
Step 2: Initialize TBLPAG and the read pointer (W6) for `TBLRD` instruction.		
0000	200xx0	`MOV #<SourceAddress23:16>, W0`
0000	880190	`MOV W0, TBLPAG`
0000	2xxxx6	`MOV #<SourceAddress15:0>, W6`
Step 3: Initialize the write pointer (W7) and store the next four locations of code memory to W0:W5.		
0000	EB0380	`CLR W7`
0000	BA1B96	`TBLRDL [W6], [W7++]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BADBB6	`TBLRDH.B [W6++], [W7++]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BADBD6	`TBLRDH.B [++W6], [W7++]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BA1BB6	`TBLRDL [W6++], [W7++]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BA1B96	`TBLRDL [W6], [W7++]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BADBB6	`TBLRDH.B [W6++], [W7++]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BADBD6	`TBLRDH.B [++W6], [W7++]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BA0BB6	`TBLRDL [W6++], [W7]`
0000	000000	`NOP`
0000	000000	`NOP`

TABLE 11-9: 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`	`883C20`	`MOV  W0, VISI`
`0000`	`000000`	`NOP`
`0001`	`<VISI>`	`Clock out contents of VISI register`
`0000`	`000000`	`NOP`
`0000`	`883C21`	`MOV  W1, VISI`
`0000`	`000000`	`NOP`
`0001`	`<VISI>`	`Clock out contents of VISI register`
`0000`	`000000`	`NOP`
`0000`	`883C22`	`MOV  W2, VISI`
`0000`	`000000`	`NOP`
`0001`	`<VISI>`	`Clock out contents of VISI register`
`0000`	`000000`	`NOP`
`0000`	`883C23`	`MOV  W3, VISI`
`0000`	`000000`	`NOP`
`0001`	`<VISI>`	`Clock out contents of VISI register`
`0000`	`000000`	`NOP`
`0000`	`883C24`	`MOV  W4, VISI`
`0000`	`000000`	`NOP`
`0001`	`<VISI>`	`Clock out contents of VISI register`
`0000`	`000000`	`NOP`
`0000`	`883C25`	`MOV  W5, VISI`
`0000`	`000000`	`NOP`
`0001`	`<VISI>`	`Clock out contents of VISI register`
`0000`	`000000`	`NOP`
Step 5: Reset the device internal PC.		
`0000`	`040100`	`GOTO 0x100`
`0000`	`000000`	`NOP`
Step 6: Repeat steps 4-5 until all desired code memory is read.		

11.10 Reading Configuration Memory

The procedure for reading configuration memory is similar to the procedure for reading code memory, except that 16-bit data words are read instead of 24-bit words. Since there are seven Configuration registers, they are read one register at a time.

Table 11-10 shows the ICSP programming details for reading all of the configuration memory. Note that the TBLPAG register is hard-coded to 0xF8 (the upper byte address of configuration memory), and the read pointer W6 is initialized to 0x0000.

TABLE 11-10: SERIAL INSTRUCTION EXECUTION FOR READING ALL CONFIGURATION MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		

0000
0000
0000
	040100
	040100
	000000
		`GOTO 0x100`
		`GOTO 0x100`
		`NOP`

Step 2: Initialize TBLPAG, and the read pointer (W6) and the write pointer (W7) for `TBLRD` instruction.		

0000
0000
0000
0000
	200F80
	880190
	EB0300
	EB0380
		`MOV #0xF8, W0`
		`MOV W0, TBLPAG`
		`CLR W6`
		`CLR W7`

Step 3: Read the Configuration register and write it to the VISI register (located at 0x784).		

0000
0000
0000
0000
0000
	BA0BB6
	000000
	000000
	883C20
	000000
		`TBLRDL [W6++], [W7]`
		`NOP`
		`NOP`
		`MOV W0, VISI`
		`NOP`

Step 4: Output the VISI register using the REGOUT command.		

0001
0000
	<VISI>
	000000
		`Clock out contents of VISI register`
		`NOP`

Step 5: Reset device internal PC.		

0000
0000
	040100
	000000
		`GOTO 0x100`
		`NOP`

Step 6: Repeat steps 3-5 until all desired code memory is read.		

11.11 Reading the Application ID Word

The application ID word is stored at address 0x8005BE 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. The REGOUT control code must then be used to clock the contents of the VISI register out of the device. The corresponding control and instruction codes that must be serially transmitted to the device to perform this operation are shown in Table 11-11.

Once the programmer has clocked-out the application ID word, it must be inspected. If the application ID has the value 0xBB, the programming executive is resident in memory and the device can be programmed using the mechanism described in **Section 5.0 "Device Programming"**. However, if the application ID has any other value, the programming executive is not resident in memory. It must be loaded to memory before the device can be programmed. The procedure for loading the programming executive to memory is described in **Section 12.0 "Programming the Programming Executive to Memory"**.

11.12 Exiting ICSP Mode

To exit from Program/Verify mode, remove VIH from $\overline{\text{MCLR}}$, as illustrated in Figure 11-6. The only requirement for exit is that an interval P9b should elapse between the last clock and program signals on PGC and PGD before removing VIH.

FIGURE 11-6: EXITING ICSP™ MODE

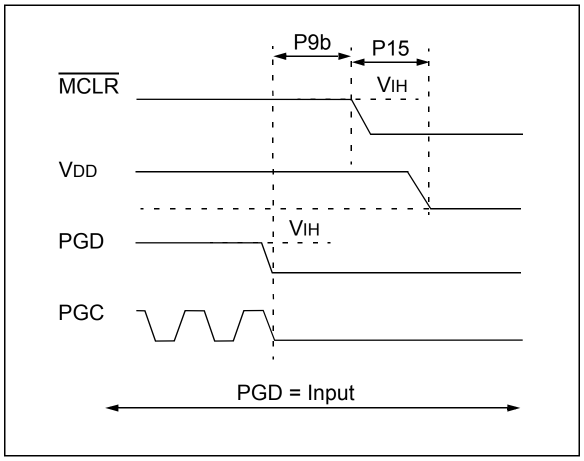


TABLE 11-11: SERIAL INSTRUCTION EXECUTION FOR READING THE APPLICATION ID WORD

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
`0000`	`040100`	`GOTO 0x100`
`0000`	`040100`	`GOTO 0x100`
`0000`	`000000`	`NOP`
Step 2: Initialize TBLPAG and the read pointer (W0) for `TBLRD` instruction.		
`0000`	`200800`	`MOV #0x80, W0`
`0000`	`880190`	`MOV W0, TBLPAG`
`0000`	`205BE0`	`MOV #0x5BE, W0`
`0000`	`207841`	`MOV VISI, W1`
`0000`	`BA0890`	`TBLRDL [W0], [W1]`
`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`
`0000`	`000000`	`NOP`

12.0 PROGRAMMING THE PROGRAMMING EXECUTIVE TO MEMORY

12.1 Overview

If it is determined that the programming executive does not reside in executive memory (as described in **Section 4.0 "Confirming The Contents of Executive Memory"**), it must be programmed into executive memory using ICSP and the techniques described in **Section 11.0 "ICSP™ Mode"**.

Storing the programming executive to executive memory is similar to normal programming of code memory. Namely, the executive memory must first be erased, and then the programming executive must be programmed 32 words at a time. This control flow is summarized in Table 12-1.

TABLE 12-1: PROGRAMMING THE PROGRAMMING EXECUTIVE

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector and erase executive memory.		
`0000`	`040100`	`GOTO 0x100`
`0000`	`040100`	`GOTO 0x100`
`0000`	`000000`	`NOP`
Step 2: Initialize the NVMCON to erase executive memory.		
`0000`	`24072A`	`MOV #0x4072, W10`
`0000`	`883B0A`	`MOV W10, NVMCON`
Step 3: Unlock the NVMCON for programming.		
`0000`	`200558`	`MOV #0x55, W8`
`0000`	`883B38`	`MOV W8, NVMKEY`
`0000`	`200AA9`	`MOV #0xAA, W9`
`0000`	`883B39`	`MOV W9, NVMKEY`
Step 4: Initiate the erase cycle.		
`0000`	`A8E761`	`BSET NVMCON, #15`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
—	—	`Externally time 'P19a' msec` (see **Section 13.0 "AC/DC Characteristics and Timing Requirements"**)
`0000`	`A9E761`	`BCLR NVMCON, #15`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
Step 5: Initialize the NVMCON to program 32 instruction words.		
`0000`	`24001A`	`MOV #0x4001, W10`
`0000`	`883B0A`	`MOV W10, NVMCON`
Step 6: Initialize TBLPAG and the write pointer (W7).		
`0000`	`200800`	`MOV #0x80, W0`
`0000`	`880190`	`MOV W0, TBLPAG`
`0000`	`EB0380`	`CLR W7`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`

TABLE 12-1: PROGRAMMING THE PROGRAMMING EXECUTIVE (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 7: Load W0:W5 with the next 4 words of packed programming executive code and initialize W6 for programming. Programming starts from the base of executive memory (0x800000) using W6 as a read pointer and W7 as a write pointer.		
`0000`	`2<LSW0>0`	`MOV #<LSW0>, W0`
`0000`	`2<MSB1:MSB0>1`	`MOV #<MSB1:MSB0>, W1`
`0000`	`2<LSW1>2`	`MOV #<LSW1>, W2`
`0000`	`2<LSW2>3`	`MOV #<LSW2>, W3`
`0000`	`2<MSB3:MSB2>4`	`MOV #<MSB3:MSB2>, W4`
`0000`	`2<LSW3>5`	`MOV #<LSW3>, W5`
Step 8: Set the read pointer (W6) and load the (next four write) latches.		
`0000`	`EB0300`	`CLR W6`
`0000`	`BB0BB6`	`TBLWTL [W6++], [W7]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`BBDBB6`	`TBLWTH.B [W6++], [W7++]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`BBEBB6`	`TBLWTH.B [W6++], [++W7]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`BB1BB6`	`TBLWTL [W6++], [W7++]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`BB0BB6`	`TBLWTL [W6++], [W7]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`BBDBB6`	`TBLWTH.B [W6++], [W7++]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`BBEBB6`	`TBLWTH.B [W6++], [++W7]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`BB1BB6`	`TBLWTL [W6++], [W7++]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
Step 9: Repeat Steps 7-8 eight times to load the write latches for the 32 instructions.		
Step 10: Unlock the NVMCON for programming.		
`0000`	`200558`	`MOV #0x55, W8`
`0000`	`883B38`	`MOV W8, NVMKEY`
`0000`	`200AA9`	`MOV #0xAA, W9`
`0000`	`883B39`	`MOV W9, NVMKEY`
Step 11: Initiate the programming cycle.		
`0000`	`A8E761`	`BSET NVMCON, #15`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
—	—	`Externally time 'P18a' msec` (see **Section 13.0 “AC/DC Characteristics and Timing Requirements”**)
`0000`	`A9E761`	`BCLR NVMCON, #15`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
Step 12: Reset the device internal PC.		
`0000`	`040100`	`GOTO 0x100`
`0000`	`000000`	`NOP`
Step 13: Repeat Steps 7-12 until all 23 rows of executive memory are programmed.		

12.2 Programming Verification

After the programming executive has been programmed to executive memory using ICSP, it must be verified. Verification is performed by reading out the contents of executive memory and comparing it with the image of the programming executive stored in the programmer.

Reading the contents of executive memory can be performed using the similar technique described in **Section 11.9 "Reading Code Memory"**. A procedure for reading executive memory is shown in Table 12-2. Note that, in Step 2, the TBLPAG register is set to 0x80 such that executive memory may be read.

TABLE 12-2: READING EXECUTIVE MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	040100	`GOTO 0x100`
0000	040100	`GOTO 0x100`
0000	000000	`NOP`
Step 2: Initialize `TBLPAG` and the read pointer (W6) for `TBLRD` instruction.		
0000	200800	`MOV #0x80, W0`
0000	880190	`MOV W0, TBLPAG`
0000	EB0300	`CLR W6`
Step 3: Initialize the write pointer (W7), and store the next four locations of executive memory to W0:W5.		
0000	EB0380	`CLR W7`
0000	BA1B96	`TBLRDL [W6], [W7++]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BADBB6	`TBLRDH.B [W6++], [W7++]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BADBD6	`TBLRDH.B [++W6], [W7++]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BA1BB6	`TBLRDL [W6++], [W7++]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BA1B96	`TBLRDL [W6], [W7++]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BADBB6	`TBLRDH.B [W6++], [W7++]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BADBD6	`TBLRDH.B [++W6], [W7++]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BA1BB6	`TBLRDL [W6++], [W7]`
0000	000000	`NOP`
0000	000000	`NOP`

TABLE 12-2: READING EXECUTIVE MEMORY (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 4: Output W0:W5 using the VISI register and REGOUT command.		
`0000`	`883C20`	`MOV W0, VISI`
`0000`	`000000`	`NOP`
`0001`	—	`Clock out contents of VISI register`
`0000`	`883C21`	`MOV W1, VISI`
`0000`	`000000`	`NOP`
`0001`	—	`Clock out contents of VISI register`
`0000`	`883C22`	`MOV W2, VISI`
`0000`	`000000`	`NOP`
`0001`	—	`Clock out contents of VISI register`
`0000`	`883C23`	`MOV W3, VISI`
`0000`	`000000`	`NOP`
`0001`	—	`Clock out contents of VISI register`
`0000`	`883C24`	`MOV W4, VISI`
`0000`	`000000`	`NOP`
`0001`	—	`Clock out contents of VISI register`
`0000`	`883C25`	`MOV W5, VISI`
`0000`	`000000`	`NOP`
`0001`	—	`Clock out contents of VISI register`
Step 5: Reset the device internal PC.		
`0000`	`040100`	`GOTO 0x100`
`0000`	`000000`	`NOP`
Step 6: Repeat Steps 3-5 until all 736 instruction words of executive memory are read.		

13.0 AC/DC CHARACTERISTICS AND TIMING REQUIREMENTS

Table 13-1 lists AC/DC characteristics and timing requirements.

TABLE 13-1: AC/DC CHARACTERISTICS AND TIMING REQUIREMENTS

AC/DC Characteristics			Standard Operating Conditions (unless otherwise stated) Operating Temperature: 25° C is recommended			
Param. No.	**Sym**	**Characteristic**	**Min**	**Max**	**Units**	**Conditions**
D110	VIHH	Programming Voltage on $\overline{\text{MCLR}}$	0.8 VDD	VDD	V	—
D112	IPP	Programming Current on $\overline{\text{MCLR}}$	—	5	mA	—
D113	IDDP	Supply Current during programming	—	30	mA	Row Erase Program memory
						—
			—	30	mA	Bulk Erase
D001	VDD	Supply voltage	3.0	5.5	V	—
D002	VDDBULK	Supply voltage for Bulk Erase programming	3.0	5.5	V	—
D031	VIL	Input Low Voltage	VSS	0.2 VSS	V	—
D041	VIH	Input High Voltage	0.8 VDD	VDD	V	—
D080	VOL	Output Low Voltage	—	0.6	V	IOL = 8.5 mA
D090	VOH	Output High Voltage	VDD – 0.7	—	V	IOH = -3.0 mA
D012	CIO	Capacitive Loading on I/O Pin (PGD)	—	50	pF	To meet AC specifications
P1	TPGC	Serial Clock (PGC) Period	100	—	ns	—
P1A	TPGCL	Serial Clock (PGC) Low Time	40	—	ns	—
P1B	TPGCH	Serial Clock (PGC) High Time	40	—	ns	—
P2	TSET1	Input Data Setup Time to Serial Clock ↓	15	—	ns	—
P3	THLD1	Input Data Hold Time from PGC ↓	15	—	ns	—
P4	TDLY1	Delay between 4-bit Command and Command Operand	40	—	ns	—
P4A	TDLY1A	Delay between 4-bit Command Operand and Next 4-bit Command	40	—	ns	—
P5	TDLY2	Delay between Last PGC ↓of Command Byte to First PGC ↑ 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 $\overline{\text{MCLR}}$ ↑	500	—	ns	—
P8	TDLY3	Delay between Last PGC ↓of Command Byte to PGD ↑ by Programming Executive	20	—	ms	—
P9a	TDLY4	Programming Executive Command Processing Time	10	—	ms	—
P9b	TDLY5	Delay between PGD ↓by Programming Executive to PGD Released by Programming Executive	15	—	ms	—
P10	TDLY6	PGC Low Time After Programming	400	—	ns	—

TABLE 13-1: AC/DC CHARACTERISTICS AND TIMING REQUIREMENTS (CONTINUED)

AC/DC Characteristics			**Standard Operating Conditions (unless otherwise stated)** Operating Temperature: 25° C is recommended			
Param. No.	**Sym**	**Characteristic**	**Min**	**Max**	**Units**	**Conditions**
P11	TDLY7	Delay to allow Bulk Erase to Occur	200	—	ms	—
P12	TR	$\overline{\text{MCLR}}$ Rise Time to Enter ICSP™ mode	—	1.0	ms	—
P13	TVALID	Data Out Valid from PGC ↑	10	—	ns	—
P14	TDLY8	Delay between Last PGC ↓and $\overline{\text{MCLR}}$ ↓	0	—	s	—
P15	THLD3	$\overline{\text{MCLR}}$ ↓to VDD ↓	—	100	ns	—
P16	TKEY1	Delay from First $\overline{\text{MCLR}}$ ↓to First PGC ↑ for Key Sequence on PGD	40	—	ns	—
P17	TKEY2	Delay from Last PGC ↓for Key Sequence on PGD to Second $\overline{\text{MCLR}}$ ↑	40	—	ns	—
P18a	TPROG	Row Programming cycle time	1	4	ms	ICSP mode
P18b	Tprog	Row programming cycle time	0.8	2.6	ms	Enhanced ICSP mode
P19a	TERA	Bulk/Row Erase cycle time	1	4	ms	ICSP mode
P19b	TERA	Bulk/Row Erase cycle time	0.8	2.6	ms	Enhanced ICSP mode

APPENDIX A: HEX FILE FORMAT

Flash programmers process the standard HEX format used by the Microchip development tools. The format supported is the Intel® HEX 32 Format (INHX32). Please refer to Appendix A in the "*MPASM™ Assembler, MPLINK™ Linker, MPLIB™ Object Librarian User's Guide"* (DS33014) for more information about hex file formats.

The basic format of the hex file is:

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

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

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

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

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

The hex file will be produced with the following contents:

```
:020000040000fa
:040200003322110096
:00000001FF
```

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

APPENDIX B: REVISION HISTORY

Revision A (February 2007)

This is the initial released version of this document.

Revision B (September 2007)

- Updated paragraph in **Section 2.2 "Pins Used During Programming"** to include reference to location of complete pin diagrams
- Removed all pin diagrams
- Modified the first sentence in **Section 5.7.1 "Overview"** to read as follows: The dsPIC30F SMPS has Configuration bits stored in **seven** 16-bit registers. Bold text denotes the change.
- Added the following note after the second paragraph in **Section 5.7.1 "Overview"**:

> **Note:** If user software performs an erase operation on the configuration fuse, it must be followed by a write operation to this fuse with the desired value, even if the desired value is the same as the state of the erased fuse.

- Added the following sentence to the end of the paragraph in **Section 5.7.5 "User Unit ID"**: Programming the UNIT ID is similar to programming the Programming Executive (see **Section 12.0 "Programming the Programming Executive to Memory"** for details).
- Updated instructions in the following tables:
  - Table 11-4 (removed two `NOP`s before the `GOTO` instruction in Step 1 and added two `NOP`s after `BSET` and `BCLR` in Step 12)
  - Table 11-5 (removed two `NOP`s before the `GOTO` instruction in Step 1 and added two `NOP`s after `BSET` and `BCLR` in Step 5 and Step 12)
  - Table 11-7 (removed two `NOP`s before the `GOTO` instruction in Step 1, changed BB1B96 to BB1B86 in Step 6, added two `NOP`s after `BSET` and `BCLR` in Step 8, and changed 9 to seven in Step 10)
  - Table 11-8 (removed two `NOP`s before the `GOTO` instruction in Step 1)
  - Table 11-9 (removed two `NOP`s before the `GOTO` instruction in Step 1)
  - Table 11-10 (removed two `NOP`s before the `GOTO` instruction in Step 1)
  - Table 11-11 (removed two `NOP`s before the `GOTO` instruction in Step 1)
  - Table 12-1 (removed two `NOP`s before the `GOTO` instruction in Step 1 and added two `NOP`s after `BSET` and `BCLR` in Step 4 and Step 11)
  - Table 12-2 (removed two `NOP`s before the `GOTO` instruction in Step 1)
- Replaced the number 9 with the word seven in the last sentence of the last paragraph of **Section 11.7 "Writing Configuration Memory"**.
- Changed the max values for parameters TPROG and TERA in Table 13-1
- Added **Appendix A: "Hex File Format"**.

Revision C (October 2010)

This version of the document includes the following updates:

- Updates to text and formatting have been incorporated throughout the document
- Added Note 3 to **Section 5.2 "Entering Enhanced ICSP Mode"**
- Updated the Device Configuration Register Map (see Table 5-3)
- Updated the assumed FGS register value from 0x5 to 0x0 in **Section 6.6 "Checksum Computation"**
- Updated Table 6-2: Checksum Computation
- Updated the first paragraph of **Section 10.0 "Device ID"**
- Updated Table 10-1: Device IDs
- Removed the VARIANT bit and updated the bit definition for the DEVID register in Table 10-2: Device ID Registers
- Removed the VARIANT bit and updated the bit field definition and description for the DEVID register in Table 10-3: Device ID Bits Description
- Added Figure 11-1: Program Entry After Reset
- Updated Note 3 in **Section 11.3 "Entering ICSP Mode"**
- Removed Note 2 in **Section 11.6 "Row Erasing Program Memory"**
- Updated Step 5 and 12 in Table 11-5: Serial Instruction Execution for Row Erasing Program Memory
- Updated Step 8 in Table 11-7: Serial Instruction Execution for Writing Configuration registers
- Updated Step 8 in Table 11-8: Serial Instruction Execution for Writing Code Memory
- Updated Step 4 and 11 in Table 12-1: Programming the Programming Executive
- Added parameters P18b and P19b in Table 13-1: AC/DC Characteristics and Timing Requirements

ISBN: 978-1-60932-610-4

MICROCHIP

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://support.microchip.com
Web Address:
www.microchip.com

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

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

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

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

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

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

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

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

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

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

ASIA/PACIFIC

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

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

China - Beijing
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

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

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

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

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

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

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

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

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

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

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

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

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

ASIA/PACIFIC

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

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

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

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

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

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

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

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

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

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

Taiwan - Hsin Chu
Tel: 886-3-6578-300
Fax: 886-3-6578-370

Taiwan - Kaohsiung
Tel: 886-7-213-7830
Fax: 886-7-330-9305

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

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

EUROPE

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

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

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

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

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

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

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

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

08/04/10