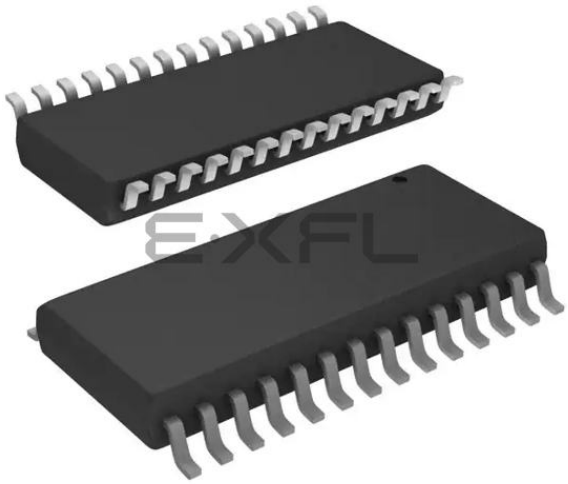
Microchip Technology - DSPIC30F4012T-20I/SO Datasheet



Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	dsPIC
Core Size	16-Bit
Speed	20 MIPS
Connectivity	CANbus, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, Motor Control PWM, QEI, POR, PWM, WDT
Number of I/O	20
Program Memory Size	48KB (16K x 24)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 5.5V
Data Converters	A/D 6x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/dspic30f4012t-20i-so

2.2 Pins Used During Programming

The pins identified in Table 2-1 are used for device programming. Refer to the appropriate device data sheet for complete pin descriptions.

TABLE 2-1: dsPIC30F PIN DESCRIPTIONS DURING PROGRAMMING

Pin Name	Pin Type	Pin Description
$\overline{\text{MCLR}}$/VPP	P	Programming Enable
VDD	P	Power Supply
VSS	P	Ground
PGC	I	Serial Clock
PGD	I/O	Serial Data

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

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 144 Kbytes (48K instruction words). Code is stored in three, 48 Kbyte memory panels that reside on-chip. Table 2-2 shows the location and program memory size of each device.

Locations 0x800000 through 0x8005BE are reserved for executive code memory. This region stores either the programming executive or 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"**.

Locations 0xFF0000 and 0xFF0002 are reserved for the Device ID registers. These bits can be used by the programmer to identify what 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 devices.

2.4 Data EEPROM Memory

The Data EEPROM array supports up to 4 Kbytes of data and is located in one memory panel. It is mapped in program memory space, residing at the end of User Memory Space (see Figure 2-2). Table 2-2 shows the location and size of data EEPROM in each device.

TABLE 2-2: CODE MEMORY AND DATA EEPROM MAP AND SIZE

Device	Code Memory map (Size in Instruction Words)	Data EEPROM Memory Map (Size in Bytes)
dsPIC30F2010	0x000000-0x001FFE (4K)	0x7FFC00-0x7FFFFE (1K)
dsPIC30F2011	0x000000-0x001FFE (4K)	None (0K)
dsPIC30F2012	0x000000-0x001FFE (4K)	None (0K)
dsPIC30F3010	0x000000-0x003FFE (8K)	0x7FFC00-0x7FFFFE (1K)
dsPIC30F3011	0x000000-0x003FFE (8K)	0x7FFC00-0x7FFFFE (1K)
dsPIC30F3012	0x000000-0x003FFE (8K)	0x7FFC00-0x7FFFFE (1K)
dsPIC30F3013	0x000000-0x003FFE (8K)	0x7FFC00-0x7FFFFE (1K)
dsPIC30F3014	0x000000-0x003FFE (8K)	0x7FFC00-0x7FFFFE (1K)
dsPIC30F4011	0x000000-0x007FFE (16K)	0x7FFC00-0x7FFFFE (1K)
dsPIC30F4012	0x000000-0x007FFE (16K)	0x7FFC00-0x7FFFFE (1K)
dsPIC30F4013	0x000000-0x007FFE (16K)	0x7FFC00-0x7FFFFE (1K)
dsPIC30F5011	0x000000-0x00AFFE (22K)	0x7FFC00-0x7FFFFE (1K)
dsPIC30F5013	0x000000-0x00AFFE (22K)	0x7FFC00-0x7FFFFE (1K)
dsPIC30F5015	0x000000-0x00AFFE (22K)	0x7FFC00-0x7FFFFE (1K)
dsPIC30F5016	0x000000-0x00AFFE (22K)	0x7FFC00-0x7FFFFE (1K)
dsPIC30F6010	0x000000-0x017FFE (48K)	0x7FF000-0x7FFFFE (4K)
dsPIC30F6010A	0x000000-0x017FFE (48K)	0x7FF000-0x7FFFFF (4K)
dsPIC30F6011	0x000000-0x015FFE (44K)	0x7FF800-0x7FFFFE (2K)
dsPIC30F6011A	0x000000-0x015FFE (44K)	0x7FF800-0x7FFFFE (2K)
dsPIC30F6012	0x000000-0x017FFE (48K)	0x7FF000-0x7FFFFE (4K)
dsPIC30F6012A	0x000000-0x017FFE (48K)	0x7FF000-0x7FFFFE (4K)
dsPIC30F6013	0x000000-0x015FFE (44K)	0x7FF800-0x7FFFFE (2K)
dsPIC30F6013A	0x000000-0x015FFE (44K)	0x7FF800-0x7FFFFE (2K)
dsPIC30F6014	0x000000-0x017FFE (48K)	0x7FF000-0x7FFFFE (4K)
dsPIC30F6014A	0x000000-0x017FFE (48K)	0x7FF000-0x7FFFFE (4K)
dsPIC30F6015	0x000000-0x017FFE (48K)	0x7FF000-0x7FFFFE (4K)

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, using a simple command set and communication protocol.

The following capabilities are provided by the programming executive:

- Read memory
  - Code memory and data EEPROM
  - Configuration registers
  - Device ID
- Erase memory
  - Bulk Erase by segment
  - Code memory (by row)
  - Data EEPROM (by row)
- Program memory
  - Code memory
  - Data EEPROM
  - 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 or data EEPROM.

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 has run, no assumptions should be made about the contents of data RAM.

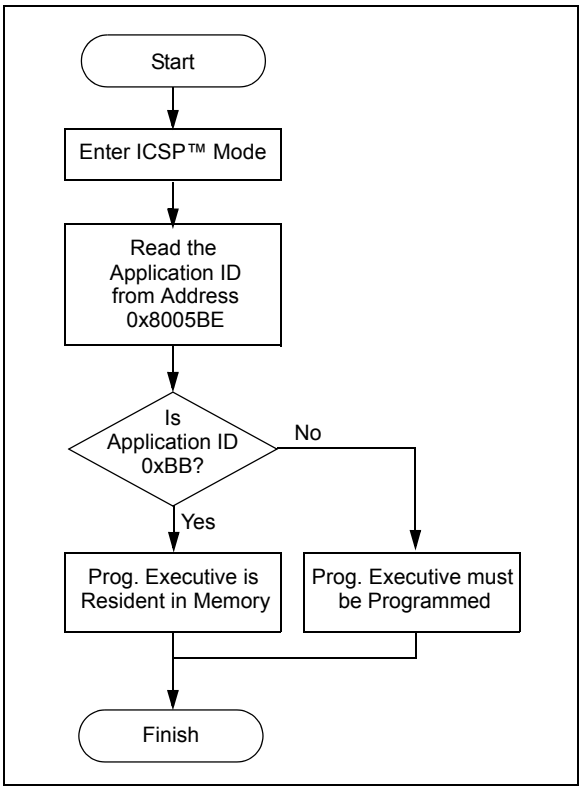
4.0 CONFIRMING THE CONTENTS OF EXECUTIVE MEMORY

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

First, ICSP mode 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 11.13 "Reading the Application ID Word"** describes the procedure for reading the application ID word in ICSP mode.

FIGURE 4-1: CONFIRMING PRESENCE OF THE PROGRAMMING EXECUTIVE



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 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

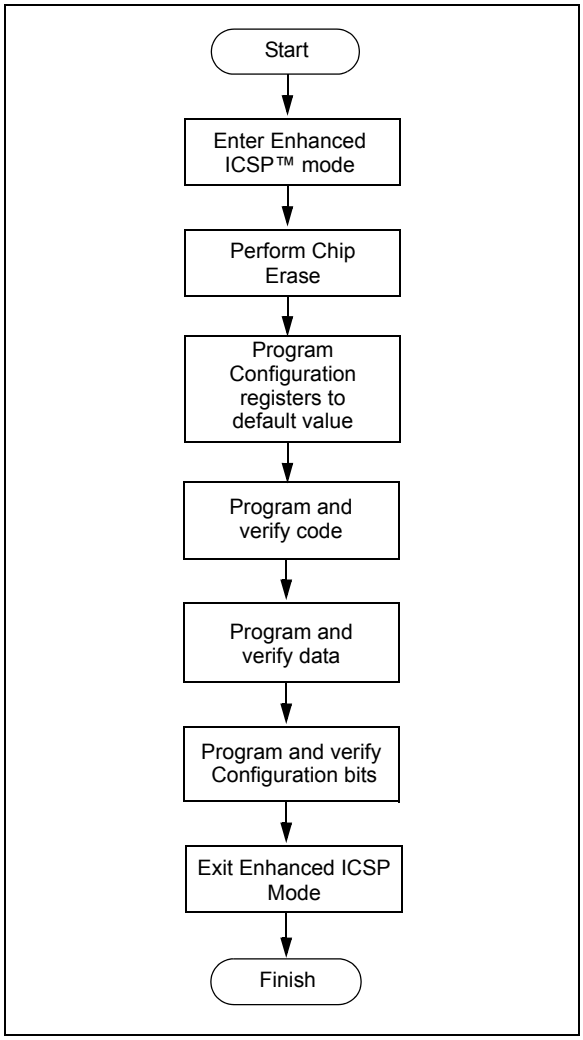
Command	Description
SCHECK	Sanity check
READD	Read data EEPROM, Configuration registers and device ID
READP	Read code memory
PROGD	Program one row of data EEPROM and verify
PROGP	Program one row of code memory and verify
PROGC	Program Configuration bits and verify
ERASEB	Bulk Erase, or erase by segment
ERASED	Erase data EEPROM
ERASEP	Erase code memory
QBLANK	Query if the code memory and data EEPROM are 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.

If Advanced Security features are enabled, then individual Segment Erase operations need to be performed, based on user selections (i.e., based on the specific needs of the user application). The specific operations that are used typically depend on the order in which various segments need to be programmed for a given application or system.

Section 5.2 "Entering Enhanced ICSP Mode" through **Section 5.8 "Exiting Enhanced ICSP Mode"** describe the programming process in detail.

FIGURE 5-1: PROGRAMMING FLOW



5.5.3 PROGRAMMING VERIFICATION

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

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

Alternatively, you can have the programmer perform the verification once the entire device is programmed using a checksum computation, as described in **Section 6.8 "Checksum Computation"**.

5.6 Data EEPROM Programming

5.6.1 OVERVIEW

The panel architecture for the data EEPROM memory array consists of 128 rows of sixteen 16-bit data words. Each panel stores 2K words. All devices have either one or no memory panels. Devices with data EEPROM provide either 512 words, 1024 words or 2048 words of memory on the one panel (see Table 5-3).

TABLE 5-3: DATA EEPROM SIZE

Device	Data EEPROM Size (Words)	Number of Rows
dsPIC30F2010	512	32
dsPIC30F2011	0	0
dsPIC30F2012	0	0
dsPIC30F3010	512	32
dsPIC30F3011	512	32
dsPIC30F3012	512	32
dsPIC30F3013	512	32
dsPIC30F3014	512	32
dsPIC30F4011	512	32
dsPIC30F4012	512	32
dsPIC30F4013	512	32
dsPIC30F5011	512	32
dsPIC30F5013	512	32
dsPIC30F5015	512	32
dsPIC30F5016	512	32
dsPIC30F6010	2048	128
dsPIC30F6010A	2048	128
dsPIC30F6011	1024	64
dsPIC30F6011A	1024	64
dsPIC30F6012	2048	128
dsPIC30F6012A	2048	128
dsPIC30F6013	1024	64
dsPIC30F6013A	1024	64
dsPIC30F6014	2048	128
dsPIC30F6014A	2048	128
dsPIC30F6015	2048	128

5.6.2 PROGRAMMING METHODOLOGY

The programming executive uses the `PROGD` command to program the data EEPROM. Figure 5-4 illustrates the flowchart of the process. Firstly, the number of rows to program (RemainingRows) is based on the device size, and the destination address (DestAddress) is set to '0'. In this example, 128 rows (2048 words) of data EEPROM will be programmed.

The first `PROGD` command programs the first row of data EEPROM. Once the command completes successfully, 'RemainingRows' is decremented by 1 and compared with 0. Since there are 127 more rows to program, 'BaseAddress' is incremented by 0x20 to point to the next row of data EEPROM. This process is then repeated until all 128 rows of data EEPROM are programmed.

FIGURE 5-4: FLOWCHART FOR PROGRAMMING dsPIC30F6014A DATA EEPROM

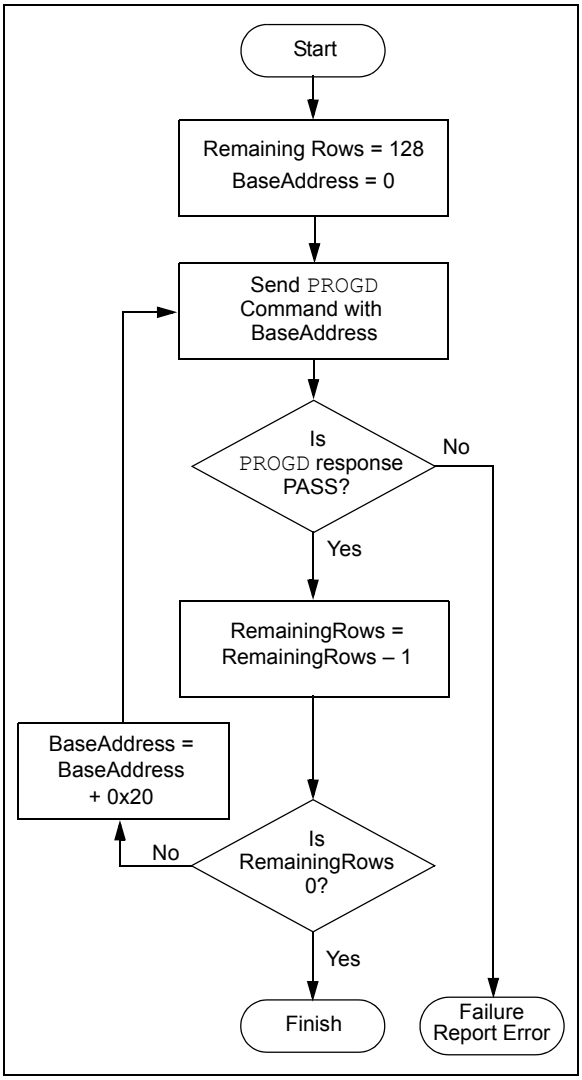


TABLE 5-5: FOSC CONFIGURATION BITS DESCRIPTION FOR dsPIC30F4011/4012 AND dsPIC30F5011/5013

Bit Field	Register	Description
FCKSM<1:0>	FOSC	**Clock Switching Mode** 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
FOS<1:0>	FOSC	**Oscillator Source Selection on POR** 11 = Primary Oscillator 10 = Internal Low-Power RC Oscillator 01 = Internal Fast RC Oscillator 00 = Low-Power 32 kHz Oscillator (Timer1 Oscillator)
FPR<3:0>	FOSC	**Primary Oscillator Mode** 1111 = ECIO w/PLL 16X – External Clock mode with 16X PLL. OSC2 pin is I/O 1110 = ECIO w/PLL 8X – External Clock mode with 8X PLL. OSC2 pin is I/O 1101 = ECIO w/PLL 4X – External Clock mode with 4X PLL. OSC2 pin is I/O 1100 = ECIO – External Clock mode. OSC2 pin is I/O 1011 = EC – External Clock mode. OSC2 pin is system clock output (FOSC/4) 1010 = FRC w/PLL 8x – Internal fast RC oscillator with 8x PLL. OSC2 pin is I/O 1001 = ERC – External RC Oscillator mode. OSC2 pin is system clock output (FOSC/4) 1000 = ERCIO – External RC Oscillator mode. OSC2 pin is I/O 0111 = XT w/PLL 16X – XT Crystal Oscillator mode with 16X PLL 0110 = XT w/PLL 8X – XT Crystal Oscillator mode with 8X PLL 0101 = XT w/PLL 4X – XT Crystal Oscillator mode with 4X PLL 0100 = XT – XT Crystal Oscillator mode (4 MHz-10 MHz crystal) 0011 = FRC w/PLL 16x – Internal fast RC oscillator with 16x PLL. OSC2 pin is I/O 0010 = HS – HS Crystal Oscillator mode (10 MHz-25 MHz crystal) 0001 = FRC w/PLL 4x – Internal fast RC oscillator with 4x PLL. OSC2 pin is I/O 0000 = XTL – XTL Crystal Oscillator mode (200 kHz-4 MHz crystal)

TABLE 5-7: CONFIGURATION BITS DESCRIPTION (CONTINUED)

Bit Field	Register	Description
EBS	FBS	**Boot Segment Data EEPROM Code Protection (only present in dsPIC30F5011/5013/6010A/6011A/6012A/6013A/6014A/6015)** 1 = No Data EEPROM is reserved for Boot Segment 0 = 128 bytes of Data EEPROM are reserved for Boot Segment in dsPIC30F5011/5013, and 256 bytes in dsPIC30F6010A/6011A/6012A/6013A/6014A/6015
BSS<2:0>	FBS	**Boot Segment Program Memory Code Protection (only present in dsPIC30F5011/5013/6010A/6011A/6012A/6013A/6014A/6015)** 111 = No Boot Segment 110 = Standard security; Small-sized Boot Program Flash [Boot Segment starts after BS and ends at 0x0003FF] 101 = Standard security; Medium-sized Boot Program Flash [Boot Segment starts after BS and ends at 0x000FFF] 100 = Standard security; Large-sized Boot Program Flash [Boot Segment starts after BS and ends at 0x001FF] 011 = No Boot Segment 010 = High security; Small-sized Boot Program Flash [Boot Segment starts after BS and ends at 0x0003FF] 001 = High security; Medium-sized Boot Program Flash [Boot Segment starts after BS and ends at 0x000FFF] 000 = High security; Large-sized Boot Program Flash [Boot Segment starts after BS and ends at 0x001FFF]
BWRP	FBS	**Boot Segment Program Memory Write Protection (only present in dsPIC30F5011/5013/6010A/6011A/6012A/6013A/6014A/6015)** 1 = Boot Segment program memory is not write-protected 0 = Boot Segment program memory is write-protected
RSS<1:0>	FSS	**Secure Segment Data RAM Code Protection (only present in dsPIC30F5011/5013/6010A/6011A/6012A/6013A/6014A/6015)** 11 = No Data RAM is reserved for Secure Segment 10 = Small-sized Secure RAM [(256 – N) bytes of RAM are reserved for Secure Segment] 01 = Medium-sized Secure RAM [(768 – N) bytes of RAM are reserved for Secure Segment in dsPIC30F5011/5013, and (2048 – N) bytes in dsPIC30F6010A/6011A/6012A/6013A/6014A/6015] 00 = Large-sized Secure RAM [(1024 – N) bytes of RAM are reserved for Secure Segment in dsPIC30F5011/5013, and (4096 – N) bytes in dsPIC30F6010A/6011A/6012A/6013A/6014A/6015] where N = Number of bytes of RAM reserved for Boot Sector.
ESS<1:0>	FSS	**Secure Segment Data EEPROM Code Protection (only present in dsPIC30F5011/5013/6010A/6011A/6012A/6013A/6014A/6015)** 11 = No Data EEPROM is reserved for Secure Segment 10 = Small-sized Secure Data EEPROM [(128 – N) bytes of Data EEPROM are reserved for Secure Segment in dsPIC30F5011/5013, and (256 – N) bytes in dsPIC30F6010A/6011A/6012A/6013A/6014A/6015] 01 = Medium-sized Secure Data EEPROM [(256 – N) bytes of Data EEPROM are reserved for Secure Segment in dsPIC30F5011/5013, and (512 – N) bytes in dsPIC30F6010A/6011A/6012A/6013A/6014A/6015] 00 = Large-sized Secure Data EEPROM [(512 – N) bytes of Data EEPROM are reserved for Secure Segment in dsPIC30F5011/5013, (1024 – N) bytes in dsPIC30F6011A/6013A, and (2048 – N) bytes in dsPIC30F6010A/6012A/6014A/6015] where N = Number of bytes of Data EEPROM reserved for Boot Sector.

TABLE 5-10: dsPIC30F CONFIGURATION REGISTERS (FOR dsPIC30F2011/2012, dsPIC30F3010/3011/3012/3013/3014, dsPIC30F4013 AND dsPIC30F5015/5016)

Address	Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xF80000	FOSC	FCKSM<1:0>		—	—	—	FOS<2:0>			—	—	—	FPR<4:0>				
0xF80002	FWDT	FWDTEN	—	—	—	—	—	—	—	—	—	FWPSA<1:0>		FWPSB<3:0>			
0xF80004	FBORPOR	MCLREN	—	—	—	—	PWMPIN(1)	HPOL(1)	LPOL(1)	BOREN	—	BORV<1:0>		—	—	FPWRT<1:0>	
0xF80006	FBS	—	—	Reserved(2)		—	—	—	Reserved(2)	—	—	—	—	Reserved(2)			
0xF80008	FSS	—	—	Reserved(2)		—	—	Reserved(2)		—	—	—	—	Reserved(2)			
0xF8000A	FGS	—	—	—	—	—	—	—	—	—	—	—	—	—	Reserved(3)	GCP	GWRP
0xF8000C	FICD	BKBUG	COE	—	—	—	—	—	—	—	—	—	—	—	—	ICS<1:0>	

Note 1: On the 2011, 2012, 3012, 3013, 3014 and 4013, these bits are reserved (read as '1' and must be programmed as '1').
2: Reserved bits read as '1' and must be programmed as '1'.
3: The FGS<2> bit is a read-only copy of the GCP bit (FGS<1>).

TABLE 5-11: dsPIC30F CONFIGURATION REGISTERS (FOR dsPIC30F6010A/6011A/6012A/6013A/6014A AND dsPIC30F6015)

Address	Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xF80000	FOSC	FCKSM<1:0>		—	—	—	FOS<2:0>			—	—	—	FPR<4:0>				
0xF80002	FWDT	FWDTEN	—	—	—	—	—	—	—	—	—	FWPSA<1:0>		FWPSB<3:0>			
0xF80004	FBORPOR	MCLREN	—	—	—	—	PWMPIN(1)	HPOL(1)	LPOL(1)	BOREN	—	BORV<1:0>		—	—	FPWRT<1:0>	
0xF80006	FBS	—	—	RBS<1:0>		—	—	—	EBS	—	—	—	—	BSS<2:0>			BWRP
0xF80008	FSS	—	—	RSS<1:0>		—	—	ESS<1:0>		—	—	—	—	SSS<2:0>			SWRP
0xF8000A	FGS	—	—	—	—	—	—	—	—	—	—	—	—	—	GSS<1:0>		GWRP
0xF8000C	FICD	BKBUG	COE	—	—	—	—	—	—	—	—	—	—	—	—	ICS<1:0>	

Note 1: On the 6011A, 6012A, 6013A and 6014A, these bits are reserved (read as '1' and must be programmed as '1').

5.7.2 PROGRAMMING METHODOLOGY

System operation Configuration bits are inherently different than all other memory cells. Unlike code memory, data EEPROM 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 as a 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. Figure 5-5 illustrates the flowchart of Configuration bit programming.

Note: If the General Code Segment Code Protect (GCP) bit is programmed to '0', code memory is code-protected and cannot be read. Code memory must be verified before enabling read protection. See **Section 5.7.4 "Code-Protect 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 was 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 whether the programming was successful.

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

5.7.4 CODE-PROTECT CONFIGURATION BITS

The FBS, FSS and FGS Configuration registers are special Configuration registers that control the size and level of code protection for the Boot Segment, Secure 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 BWRP, SWRP and GWRP bits control write protection; and BSS<2:0>, SSS<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' (see **Section 5.7 "Configuration Bits Programming"**).

In addition to code memory protection, parts of data EEPROM and/or data RAM can be configured to be accessible only by code resident in the Boot Segment and/or Secure Segment. The sizes of these "reserved" sections are user-configurable, using the EBS, RBS<1:0>, ESS<1:0> and RSS<1:0> bits.

Note 1: All bits in the FBS, FSS and FGS Configuration registers can only be programmed to a value of '0'. ERASEB is the only way to reprogram code-protect bits from ON ('0') to OFF ('1').

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

6.6 Configuration Information in the Hexadecimal File

To allow portability of code, the programmer must read the Configuration register locations from the hexadecimal file. If configuration information is not present in the hexadecimal file, a simple warning message should be issued by the programmer. Similarly, while saving a hexadecimal file, all configuration information must be included. An option to not include the configuration information can be provided.

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

6.7 Unit ID

The dsPIC30F 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.

A Bulk Erase does not erase the Unit ID locations. Instead, erase all executive memory using steps 1-4 as shown in Table 12-1, and program the Unit ID along with the programming executive. Alternately, use a Row Erase to erase the row containing the Unit ID locations.

6.8 Checksum Computation

Checksums for the dsPIC30F are 16 bits in size. The checksum is to total sum of the following:

- Contents of code memory locations
- Contents of Configuration registers

Table A-1 describes how to calculate the checksum for each device. 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.

Note:	The checksum calculation differs depending on the code-protect setting. Table A-1 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.

7.0 PROGRAMMER – PROGRAMMING EXECUTIVE COMMUNICATION

7.1 Communication Overview

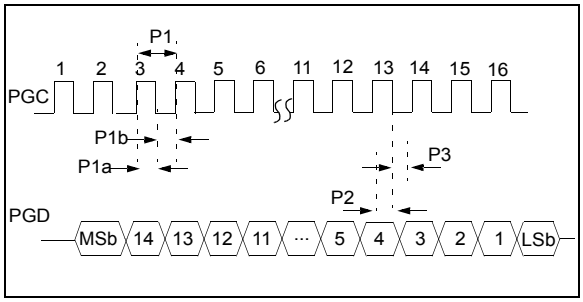
The programmer and programming executive have a master-slave relationship, where the programmer is the master programming device and the programming executive is the slave.

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 falling edge of PGC and latched on the rising edge of PGC. All data transmissions are sent Most Significant bit (MSb) 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

TABLE 8-1: PROGRAMMING EXECUTIVE COMMAND SET

Opcode	Mnemonic	Length (16-bit words)	Time Out	Description
0x0	SCHECK	1	1 ms	Sanity check.
0x1	READD	4	1 ms/row	Read N 16-bit words of data EEPROM, Configuration registers or device ID starting from specified address.
0x2	READP	4	1 ms/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(2)	19	5 ms	Program one row of data EEPROM at the specified address, then verify.
0x5	PROGP(1)	51	5 ms	Program one row of code memory at the specified address, then verify.
0x6	PROGC	4	5 ms	Write byte or 16-bit word to specified Configuration register.
0x7	ERASEB	2	5 ms	Bulk Erase (entire code memory or data EEPROM), or erase by segment.
0x8	ERASED(2)	3	5 ms/row	Erase rows of data EEPROM from specified address.
0x9	ERASEP(1)	3	5 ms/row	Erase rows of code memory from specified address.
0xA	QBLANK	3	300 ms	Query if the code memory and data EEPROM are blank.
0xB	QVER	1	1 ms	Query the programming executive software version.

Note 1: One row of code memory consists of (32) 24-bit words. Refer to Table 5-2 for device-specific information.
2: One row of data EEPROM consists of (16) 16-bit words. Refer to Table 5-3 for device-specific information.

8.5.9 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. Code-protect Configuration registers can only be erased with the ERASEB command, while the device ID is read-only.

8.5.10 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 if the contents of code memory and data EEPROM are blank (contains all '1's). The size of code memory and data EEPROM 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 Blank Check for data EEPROM begins at 0x7FFFFE and advances toward smaller addresses for the specified number of data words.

QBLANK returns a QE_Code of 0xF0 if the specified code memory and data EEPROM are 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.

11.0 ICSP™ MODE

11.1 ICSP Mode

ICSP mode is a special programming protocol that allows you to read and write to the dsPIC30F 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 whether 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: Because ICSP is slower, it is recommended that only Enhanced ICSP (E-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 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 the ICSP entry sequence, the first 4-bit control code is always forced to SIX and a forced NOP instruction is executed by the CPU. Five additional PGC clocks are needed on start-up, thereby resulting in a 9-bit SIX command instead of the normal 4-bit SIX command. After the forced SIX is clocked in, ICSP operation resumes as normal (the next 24 clock cycles load the first instruction word to the CPU). See Figure 11-1 for details.

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

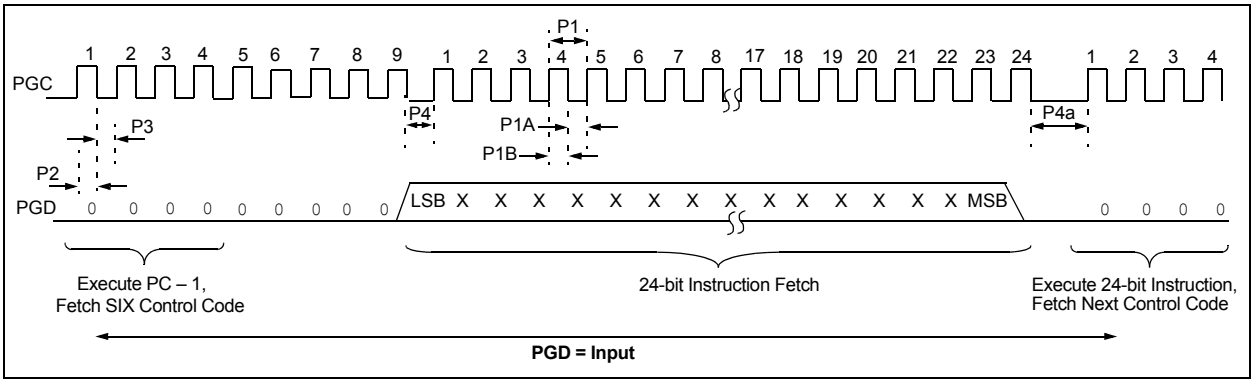


FIGURE 11-2: SIX SERIAL EXECUTION

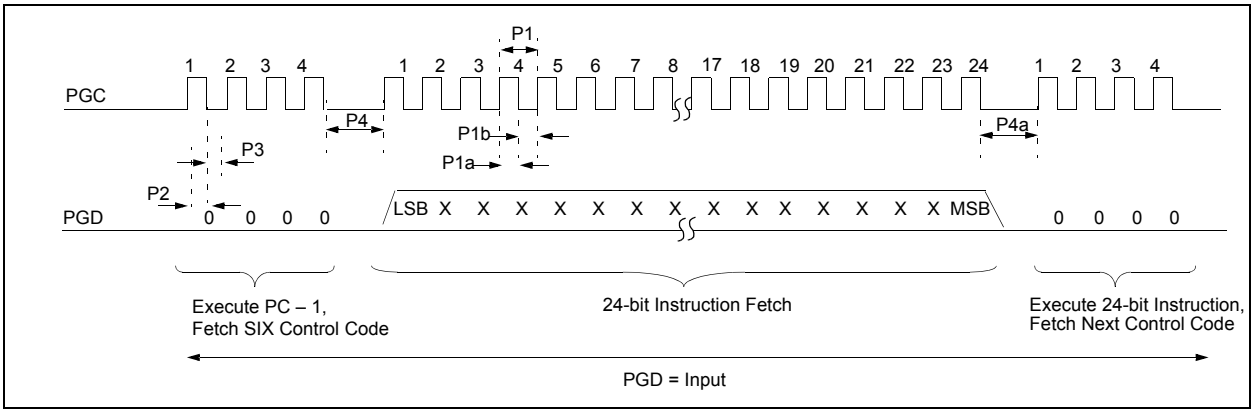
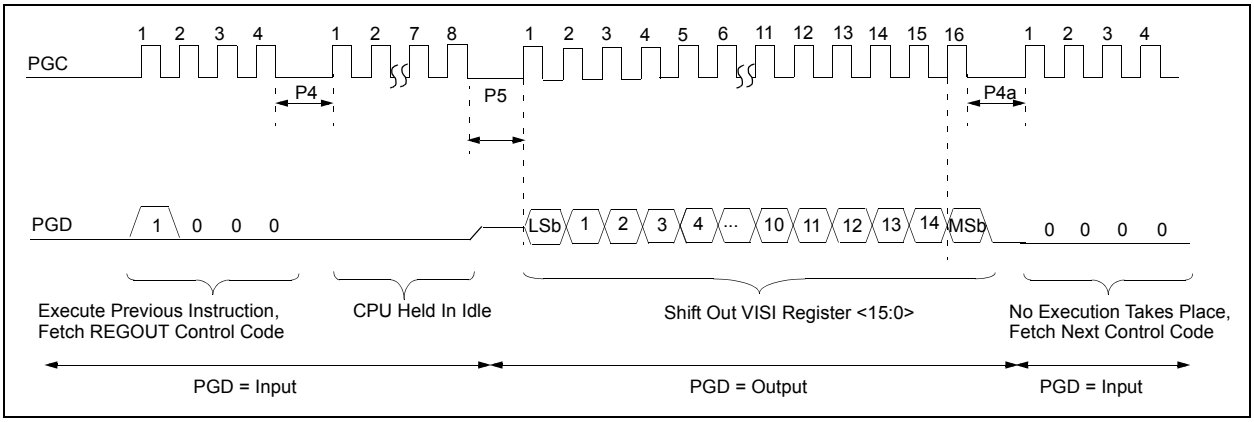


FIGURE 11-3: REGOUT SERIAL EXECUTION



11.4 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 "Erasing Program Memory in Normal-Voltage Systems"** through **Section 11.13 "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 ms 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	Erase all code memory, data memory (does not erase UNIT ID).
0x4075	Erase 1 row (16 words) of data EEPROM.
0x4074	Erase 1 word of data EEPROM.
0x4072	Erase all executive memory.
0x4071	Erase 1 row (32 instruction words) from 1 panel of code memory.
0x406E	Erase Boot Secure and General Segments, then erase FBS, FSS and FGS configuration registers.
0x4066	Erase all Data EEPROM allocated to Boot Segment.
0x405E	Erase Secure and General Segments, then erase FSS and FGS configuration registers.
0x4056	Erase all Data EEPROM allocated to Secure Segment.
0x404E	Erase General Segment, then erase FGS configuration register.
0x4046	Erase all Data EEPROM allocated to General Segment.

TABLE 11-3: NVMCON WRITE OPERATIONS

NVMCON Value	Write Operation
0x4008	Write 1 word to configuration memory.
0x4005	Write 1 row (16 words) to data memory.
0x4004	Write 1 word to data memory.
0x4001	Write 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 ms>
BCLR    NVMCON, #WR
```

11.5 Erasing Program Memory in Normal-Voltage Systems

The procedure for 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. This method of bulk erasing program memory only works for systems where VDD is between 4.5 volts and 5.5 volts. The method for erasing program memory for systems with a lower VDD (3.0 volts-4.5 volts) is described in **Section 6.1 "Erasing Memory"**.

TABLE 11-4: SERIAL INSTRUCTION EXECUTION FOR BULK ERASING PROGRAM MEMORY (ONLY IN NORMAL-VOLTAGE SYSTEMS) (CONTINUED)

Command (Binary)	Data (Hexadecimal)	Description
0000	200558	MOV #0x55, W8
0000	883B38	MOV W8, NVMKEY
0000	200AA9	MOV #0xAA, W9
0000	883B39	MOV W9, NVMKEY
Step 11: Initiate the erase cycle.		
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
—	—	Externally time 'P13a' ms (see **Section 13.0 "AC/DC Characteristics and Timing Requirements"**)
0000	000000	NOP
0000	000000	NOP
0000	A9E761	BCLR NVMCON, #WR
0000	000000	NOP
0000	000000	NOP

Note 1: Steps 2-8 are only required for the dsPIC30F5011/5013 devices. These steps may be skipped for all other devices in the dsPIC30F family.

11.7 Writing Configuration Memory

The FOSC, FWDT, FBORPOR and FICD registers are not erasable. It is recommended that all Configuration registers be set to a default value after erasing program memory. The FWDT, FBORPOR 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 FGS, FBS and FSS Configuration registers are special since they enable code protection for the device. For security purposes, once any bit in these registers is programmed to '0' (to enable some code protection feature), it can only be set back to '1' by performing a Bulk Erase or Segment Erase as described in **Section 11.5 "Erasing Program Memory in Normal-Voltage Systems"**. Programming these bits from a '0' to '1' is not possible, but they may 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 the 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. Lastly, Steps 3-9 are repeated six times until all seven Configuration registers are cleared.

TABLE 11-6: DEFAULT CONFIGURATION REGISTER VALUES

Address	Register	Default Value
0xF80000	FOSC	0xC100
0xF80002	FWDT	0x803F
0xF80004	FBORPOR	0x87B3
0xF80006	FBS	0x310F
0xF80008	FSS	0x330F
0xF8000A	FGS	0x0007
0xF8000C	FICD	0xC003

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

Command (Binary)	Data (Hexadecimal)	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 0000	2xxxx0 000000	MOV #<CONFIG_VALUE>, W0 NOP

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

Command (Binary)	Data (Hexadecimal)	Description
Step 6: Write the Configuration register data to the write latch and increment the write pointer.		
0000	BB1B96	TBLWTL W6, [W7++]
0000	000000	NOP
0000	000000	NOP
Step 7: 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 8: Initiate the write cycle.		
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
—	—	Externally time 'P12a' ms (see **Section 13.0 "AC/DC Characteristics and Timing Requirements"**)
0000	000000	NOP
0000	000000	NOP
0000	A9E761	BCLR NVMCON, #WR
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 7 Configuration registers are cleared.		

11.11 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 11-11 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 11-11: SERIAL INSTRUCTION EXECUTION FOR READING ALL CONFIGURATION MEMORY

Command (Binary)	Data (Hexadecimal)	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) and the write pointer (W7) for TBLRD instruction.		
0000	200F80	MOV #0xF8, W0
0000	880190	MOV W0, TBLPAG
0000	EB0300	CLR W6
0000	EB0380	CLR W7
0000	000000	NOP
Step 3: Read the Configuration register and write it to the VISI register (located at 0x784).		
0000	BA0BB6	TBLRDL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP
0000	883C20	MOV W0, VISI
0000	000000	NOP
Step 4: Output the VISI register using the REGOUT command.		
0001	<VISI>	Clock out contents of VISI register
0000	000000	NOP
Step 5: Reset device internal PC.		
0000	040100	GOTO 0x100
0000	000000	NOP
Step 6: Repeat steps 3-5 six times to read all of configuration memory.		

11.12 Reading Data Memory

The procedure for reading data memory is similar to that of reading code memory, except that 16-bit data words are read instead of 24-bit words. Since less data is read in each operation, only working registers W0:W3 are used as temporary holding registers for the data to be read.

Table 11-12 shows the ICSP programming details for reading data memory. Note that the TBLPAG register is hard-coded to 0x7F (the upper byte address of all locations of data memory).

TABLE 11-12: SERIAL INSTRUCTION EXECUTION FOR READING DATA MEMORY

Command (Binary)	Data (Hexadecimal)	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	2007F0	MOV #0x7F, 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	000000	NOP
0000	BA1BB6	TBLRDL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BA1BB6	TBLRDL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BA1BB6	TBLRDL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BA1BB6	TBLRDL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
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
Step 5: Reset device internal PC.		
0000	040100	GOTO 0x100
0000	000000	NOP
Step 6: Repeat steps 3-5 until all desired data memory is read.		

APPENDIX B: 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 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.