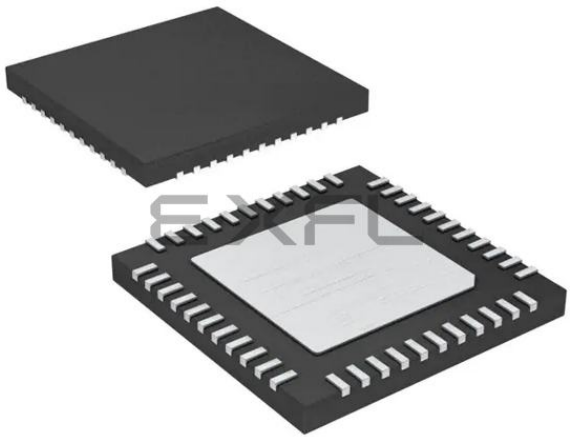
Microchip Technology - DSPIC30F3010T-20I/ML Datasheet



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	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, Motor Control PWM, QEI, POR, PWM, WDT
Number of I/O	20
Program Memory Size	24KB (8K x 24)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	1K 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	44-VQFN Exposed Pad
Supplier Device Package	44-QFN (8x8)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/dspic30f3010t-20i-ml

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

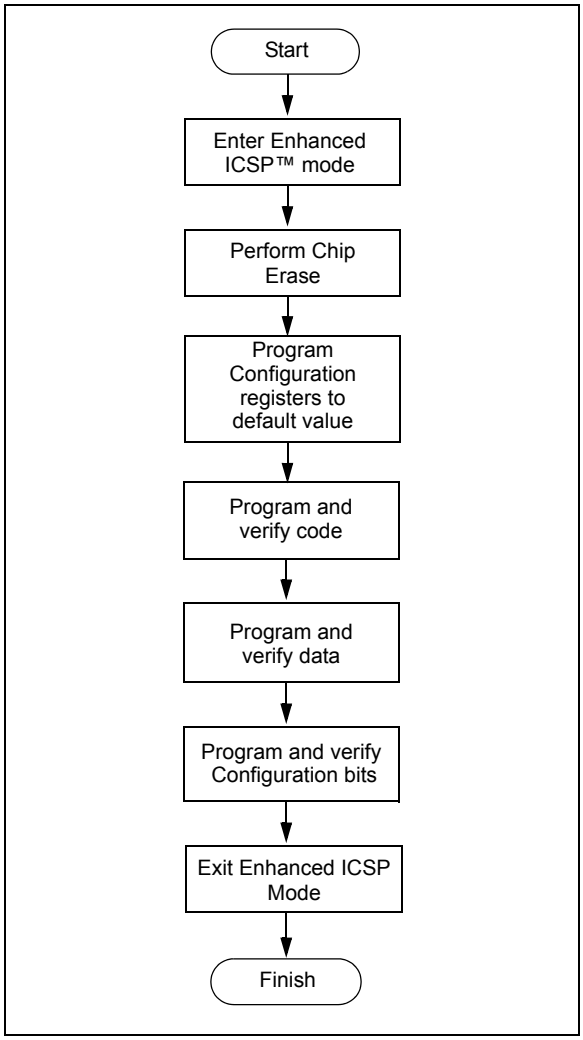
Command	Description
SCHECK	Sanity check
READD	Read data EEPROM, Configuration registers and device ID
READP	Read code memory
PROGD	Program one row of data EEPROM and verify
PROGP	Program one row of code memory and verify
PROGC	Program Configuration bits and verify
ERASEB	Bulk Erase, or erase by segment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

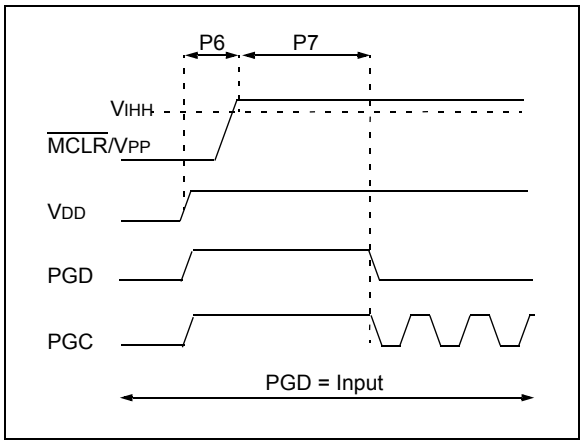
FIGURE 5-1: PROGRAMMING FLOW



5.2 Entering Enhanced ICSP Mode

The Enhanced ICSP mode is entered by holding PGC and PGD high, and then raising MCLR/VPP to VIHH (high voltage), as illustrated in Figure 5-2. In this mode, the code memory, data EEPROM and Configuration bits can be efficiently programmed using the programming executive commands that are serially transferred using PGC and PGD.

FIGURE 5-2: ENTERING ENHANCED ICSP™ MODE



Note 1: The sequence that places the device into Enhanced ICSP mode places all unused I/Os in the high-impedance state.

2: Before entering Enhanced ICSP mode, clock switching must be disabled using ICSP, by programming the FCKSM<1:0> bits in the FOSC Configuration register to '11' or '10'.

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

5.3 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, data EEPROM and code-protect Configuration bits. The Chip Erase process sets all bits in these three memory regions to '1'.

Since non-code-protect Configuration bits cannot be erased, 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"**).

If Advanced Security features are enabled, then individual Segment Erase operations would need to be performed, depending on which segment needs to be programmed at a given stage of system programming. The user should have the flexibility to select specific segments for programming.

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

5.4 Blank Check

The term "Blank Check" means to verify that the device has been successfully erased and has no programmed memory cells. A blank or erased memory cell reads as '1'. The following memories must be blank checked:

- All implemented code memory
- All implemented data EEPROM
- 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 and data EEPROM are 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 5.3 "Chip Erase"**) before attempting to program the chip.

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 5-7: CONFIGURATION BITS DESCRIPTION (CONTINUED)

Bit Field	Register	Description
SSS<2:0>	FSS	**Secure Segment Program Memory Code Protection (only present in dsPIC30F5011/5013/6010A/6011A/6012A/6013A/6014A/6015)** 111 = No Secure Segment 110 = Standard security; Small-sized Secure Program Flash [Secure Segment starts after BS and ends at 0x001FFF] 101 = Standard security; Medium-sized Secure Program Flash [Secure Segment starts after BS and ends at 0x003FFF] 100 = Standard security; Large-sized Secure Program Flash [Secure Segment starts after BS and ends at 0x007FFF] 011 = No Secure Segment 010 = High security; Small-sized Secure Program Flash [Secure Segment starts after BS and ends at 0x001FFF] 001 = High security; Medium-sized Secure Program Flash [Secure Segment starts after BS and ends at 0x003FFF] 000 = High security; Large-sized Secure Program Flash [Secure Segment starts after BS and ends at 0x007FFF]
SWRP	FSS	**Secure Segment Program Memory Write Protection (only present in dsPIC30F5011/5013/6010A/6011A/6012A/6013A/6014A/6015)** 1 = Secure Segment program memory is not write-protected 0 = Secure program memory is write-protected
GSS<1:0>	FGS	**General Segment Program Memory Code Protection (only present in dsPIC30F5011/5013/6010A/6011A/6012A/6013A/6014A/6015)** 11 = Code protection is disabled 10 = Standard security code protection is enabled 0x = High security code protection is enabled
GCP	FGS	**General Segment Program Memory Code Protection (present in all devices except dsPIC30F5011/5013/6010A/6011A/6012A/6013A/6014A/6015)** 1 = General Segment program memory is not code-protected 0 = General Segment program memory is code-protected
GWRP	FGS	**General Segment Program Memory Write Protection** 1 = General Segment program memory is not write-protected 0 = General Segment program memory is write-protected
BKBUG	FICD	**Debugger/Emulator Enable** 1 = Device will reset into Operational mode 0 = Device will reset into Debug/Emulation mode
COE	FICD	**Debugger/Emulator Enable** 1 = Device will reset into Operational mode 0 = Device will reset into Clip-on Emulation mode
ICS<1:0>	FICD	**ICD Communication Channel Select** 11 = Communicate on PGC/EMUC and PGD/EMUD 10 = Communicate on EMUC1 and EMUD1 01 = Communicate on EMUC2 and EMUD2 00 = Communicate on EMUC3 and EMUD3
RESERVED	FBS, FSS, FGS	**Reserved (read as '1', write as '1')**
—	All	**Unimplemented (read as '0', write as '0')**

TABLE 5-8: dsPIC30F CONFIGURATION REGISTERS (FOR dsPIC30F2010, dsPIC30F4011/4012 AND dsPIC30F6010/ 6011/6012/6013/ 6014)

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<1:0>		—	—	—	—	FPR<3:0>			
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(2)	GCP	GWRP
0xF8000C	FICD	BKBUG	COE	—	—	—	—	—	—	—	—	—	—	—	—	ICS<1:0>	

Note 1: On the 6011, 6012, 6013 and 6014, 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'.

TABLE 5-9: dsPIC30F CONFIGURATION REGISTERS (FOR dsPIC30F5011/5013)

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<1:0>		—	—	—	—	FPR<3:0>			
0xF80002	FWDT	FWDTEN	—	—	—	—	—	—	—	—	—	FWPSA<1:0>		FWPSB<3:0>			
0xF80004	FBORPOR	MCLREN	—	—	—	—	Reserved(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: Reserved bits 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.

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 (see **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"**.

8.5.7 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 = All Data EEPROM in General Segment 0x2 = All Code and Data EEPROM in General Segment, interrupt vectors and FGS Configuration register 0x3 = Full Chip Erase 0x4 = All Code and Data EEPROM in Boot, Secure and General Segments, and FBS, FSS and FGS Configuration registers 0x5 = All Code and Data EEPROM in Secure and General Segments, and FSS and FGS Configuration registers 0x6 = All Data EEPROM in Boot Segment 0x7 = All Data EEPROM in Secure Segment

The ERASEB command performs a Bulk Erase. The MS field selects the memory to be bulk erased, with options for erasing Code and/or Data EEPROM in individual memory segments.

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

- All code memory (even if code-protected)
- All data EEPROM
- All code-protect Configuration registers

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

Expected Response (2 words):
0x1700
0x0002

Note:	A Full Chip Erase cannot be performed in low-voltage programming systems (V_{DD} less than 4.5 volts). ERASED and ERASEP must be used to erase code memory, executive memory and data memory. Alternatively, individual Segment Erase operations may be performed.

8.5.8 ERASED COMMAND

15 12	11 8	7 0
Opcode	Length	
Num_Rows		Addr_MSB
Addr_LS		

Field	Description
Opcode	0x8
Length	0x3
Num_Rows	Number of rows to erase (max of 128)
Addr_MSB	MSB of 24-bit base address
Addr_LS	LS 16 bits of 24-bit base address

The ERASED command erases the specified number of rows of data EEPROM from the specified base address. The specified base address must be a multiple of 0x20. Since the data EEPROM is mapped to program space, a 24-bit base address must be specified.

After the erase is performed, all targeted bytes of data EEPROM will contain 0xFF.

Expected Response (2 words):
0x1800
0x0002

Note:	The ERASED 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.

8.5.11 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 is 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

As shown in Example 9-1, all programming executive responses have a general format consisting of a two word header and any required data for the command. Table 9-2 lists the fields and their descriptions.

EXAMPLE 9-1: FORMAT

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

TABLE 9-2: FIELDS AND DESCRIPTIONS

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 was 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 whether the programming executive correctly received the command that the programmer transmitted.

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

Device	DEVID	Silicon Revision							
		A0	A1	A2	A3	A4	B0	B1	B2
dsPIC30F2010	0x0040	0x1000	0x1001	0x1002	0x1003	0x1004	—	—	—
dsPIC30F2011	0x0240	—	0x1001	—	—	—	—	—	—
dsPIC30F2012	0x0241	—	0x1001	—	—	—	—	—	—
dsPIC30F3010	0x01C0	0x1000	0x1001	0x1002	—	—	—	—	—
dsPIC30F3011	0x01C1	0x1000	0x1001	0x1002	—	—	—	—	—
dsPIC30F3012	0x00C1	—	—	—	—	—	0x1040	0x1041	—
dsPIC30F3013	0x00C3	—	—	—	—	—	0x1040	0x1041	—
dsPIC30F3014	0x0160	—	0x1001	0x1002	—	—	—	—	—
dsPIC30F4011	0x0101	—	0x1001	0x1002	0x1003	0x1003	—	—	—
dsPIC30F4012	0x0100	—	0x1001	0x1002	0x1003	0x1003	—	—	—
dsPIC30F4013	0x0141	—	0x1001	0x1002	—	—	—	—	—
dsPIC30F5011	0x0080	—	0x1001	0x1002	0x1003	0x1003	—	—	—
dsPIC30F5013	0x0081	—	0x1001	0x1002	0x1003	0x1003	—	—	—
dsPIC30F5015	0x0200	0x1000	—	—	—	—	—	—	—
dsPIC30F5016	0x0201	0x1000	—	—	—	—	—	—	—
dsPIC30F6010	0x0188	—	—	—	—	—	—	0x1040	0x1042
dsPIC30F6010A	0x0281	—	—	0x1002	0x1003	0x1004	—	—	—
dsPIC30F6011	0x0192	—	—	—	0x1003	—	—	0x1040	0x1042
dsPIC30F6011A	0x02C0	—	—	0x1002	—	—	0x1040	0x1041	—
dsPIC30F6012	0x0193	—	—	—	0x1003	—	—	0x1040	0x1042
dsPIC30F6012A	0x02C2	—	—	0x1002	—	—	0x1040	0x1041	—
dsPIC30F6013	0x0197	—	—	—	0x1003	—	—	0x1040	0x1042
dsPIC30F6013A	0x02C1	—	—	0x1002	—	—	0x1040	0x1041	—
dsPIC30F6014	0x0198	—	—	—	0x1003	—	—	0x1040	0x1042
dsPIC30F6014A	0x02C3	—	—	0x1002	—	—	0x1040	0x1041	—
dsPIC30F6015	0x0280	—	—	0x1002	0x1003	0x1004	—	—	—

TABLE 10-2: dsPIC30F 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<5:0>						DOT<5:0>					

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

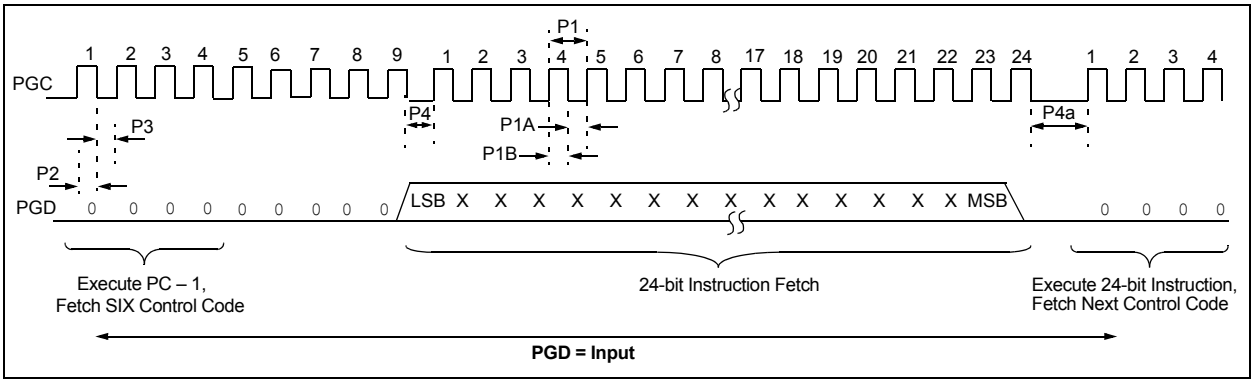


FIGURE 11-2: SIX SERIAL EXECUTION

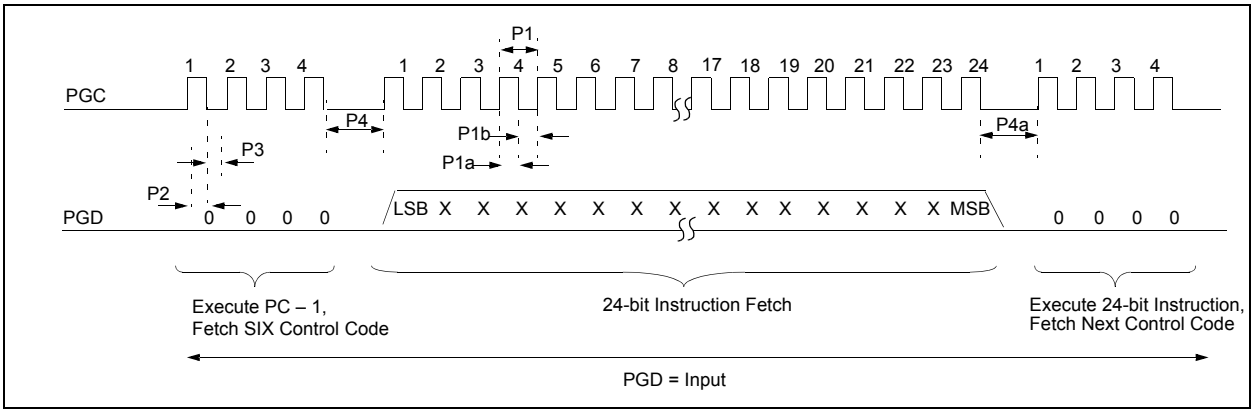
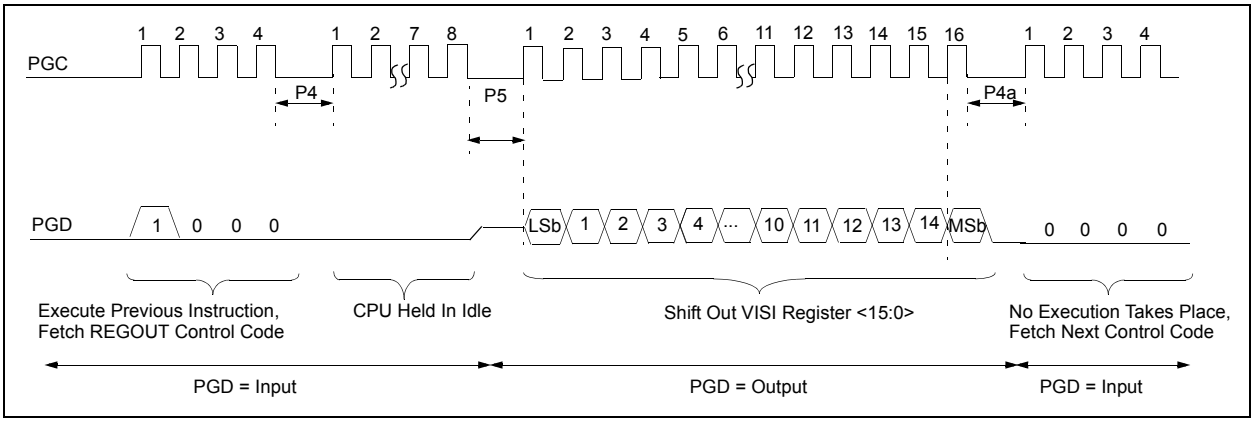


FIGURE 11-3: REGOUT SERIAL EXECUTION



11.6 Erasing Program Memory in Low-Voltage Systems

The procedure for erasing program memory (all code memory and data memory) in low-voltage systems (with VDD between 2.5 volts and 4.5 volts) is quite different than the procedure for erasing program memory in normal-voltage systems. Instead of using a Bulk Erase operation, each region of memory must be individually erased by row. Namely, all of the code memory, executive memory and data memory must be erased one row at a time. This procedure is detailed in Table 11-5.

Due to security restrictions, the FBS, FSS and FGS register cannot be erased in low-voltage systems. Once any bits in the FGS register are programmed to '0', they can only be set back to '1' by performing a Bulk Erase in a normal-voltage system. Alternatively, a Segment Erase operation can be performed instead of a Bulk Erase.

Normal-voltage systems can also be used to erase program memory as shown in Table 11-5. However, since this method is more time-consuming and does not clear the code-protect bits, it is not recommended.

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

TABLE 11-5: SERIAL INSTRUCTION EXECUTION FOR ERASING PROGRAM MEMORY (EITHER IN LOW-VOLTAGE OR NORMAL-VOLTAGE SYSTEMS)

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 NVMADR and NVMADRU to erase code memory and initialize W7 for row address updates.		
0000	EB0300	CLR W6
0000	883B16	MOV W6, NVMADR
0000	883B26	MOV W6, NVMADRU
0000	200407	MOV #0x40, W7
Step 3: Set NVMCON to erase 1 row of code memory.		
0000	24071A	MOV #0x4071, W10
0000	883B0A	MOV W10, NVMCON
Step 4: Unlock the NVMCON to erase 1 row of code memory.		
0000	200558	MOV #0x55, W8
0000	883B38	MOV W8, NVMKEY
0000	200AA9	MOV #0xAA, W9
0000	883B39	MOV W9, NVMKEY
Step 5: 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

TABLE 11-5: SERIAL INSTRUCTION EXECUTION FOR ERASING PROGRAM MEMORY (EITHER IN LOW-VOLTAGE OR NORMAL-VOLTAGE SYSTEMS) (CONTINUED)

Command (Binary)	Data (Hexadecimal)	Description
Step 6: Update the row address stored in NVMADRU:NVMADR. When W6 rolls over to 0x0, NVMADRU must be incremented.		
0000	430307	ADD W6, W7, W6
0000	AF0042	BTSC SR, #C
0000	EC2764	INC NVMADRU
0000	883B16	MOV W6, NVMADR
Step 7: Reset device internal PC.		
0000	040100	GOTO 0x100
0000	000000	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	EB0300	CLR W6
0000	883B16	MOV W6, NVMADR
0000	200807	MOV #0x80, W7
0000	883B27	MOV W7, NVMADRU
0000	200407	MOV #0x40, W7
Step 10: Set NVMCON to erase 1 row of executive memory.		
0000	24071A	MOV #0x4071, W10
0000	883B0A	MOV W10, NVMCON
Step 11: Unlock the NVMCON to erase 1 row of executive memory.		
0000	200558	MOV #0x55, W8
0000	883B38	MOV W8, NVMKEY
0000	200AA9	MOV #0xAA, W9
0000	883B39	MOV W9, NVMKEY
Step 12: 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
Step 13: Update the row address stored in NVMADR.		
0000	430307	ADD W6, W7, W6
0000	883B16	MOV W6, NVMADR
Step 14: Reset device internal PC.		
0000	040100	GOTO 0x100
0000	000000	NOP
Step 15: Repeat Steps 10-14 until all 24 rows of executive memory are erased.		
Step 16: Initialize NVMADR and NVMADRU to erase data memory and initialize W7 for row address updates.		
0000	2XXXX6	MOV #<lower 16-bits of starting Data EEPROM address>, W6
0000	883B16	MOV W6, NVMADR
0000	2007F6	MOV #0x7F, W6
0000	883B16	MOV W6, NVMADRU
0000	200207	MOV #0x20, W7
Step 17: Set NVMCON to erase 1 row of data memory.		
0000	24075A	MOV #0x4075, W10
0000	883B0A	MOV W10, NVMCON

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 11-10: SERIAL INSTRUCTION EXECUTION FOR READING CODE MEMORY (CONTINUED)

Command (Binary)	Data (Hexadecimal)	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 3-5 until all desired code memory is read.		

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.		

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. 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 (Hexadecimal)	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
—	—	Externally time 'P13a' ms (see **Section 13.0 "AC/DC Characteristics and Timing Requirements"**)
0000	000000	NOP
0000	000000	NOP
0000	A9E761	BCLR NVMCON, #15
0000	000000	NOP
0000	000000	NOP
Step 5: Initialize the 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
Step 6: Initialize the NVMCON to program 32 instruction words.		
0000	24001A	MOV #0x4001, W10
0000	883B0A	MOV W10, NVMCON
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

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 same technique described in **Section 11.10 "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 (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	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	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
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 13-1: AC/DC CHARACTERISTICS (CONTINUED)

AC/DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating Temperature: 25° C is recommended			
Param. No.	**Sym**	**Characteristic**	**Min**	**Max**	**Units**	**Conditions**
P9b	TDLY5	Delay between PGD ↓by programming executive to PGD released by programming executive	15	—	µs	—
P10	TDLY6	Delay between PGD released by programming executive to first PGC ↑ of response	5	—	µs	—
P11	TDLY7	Delay between clocking out response words	10	—	µs	—
P12a	TPROG	Row Programming cycle time	1	4	ms	ICSP mode
P12b	TPROG	Row Programming cycle time	0.8	2.6	ms	Enhanced ICSP mode
P13a	TERA	Bulk/Row Erase cycle time	1	4	ms	ICSP mode
P13b	TERA	Bulk/Row Erase cycle time	0.8	2.6	ms	Enhanced ICSP mode

APPENDIX A: DEVICE-SPECIFIC INFORMATION

A.1 Checksum Computation

The checksum computation is described in **Section 6.8 "Checksum Computation"**. Table A-1 shows how this 16-bit computation can be made for each dsPIC30F 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 0x5.

A.2 dsPIC30F5011 and dsPIC30F5013

A.2.1 ICSP PROGRAMMING

The dsPIC30F5011 and dsPIC30F5013 processors require that the FBS and FSS registers be programmed with 0x0000 before the device is chip erased. The steps to perform this action are shown in Table 11-4.

A.2.2 ENHANCED ICSP PROGRAMMING

The dsPIC30F5011 and dsPIC30F5013 processors require that the FBS and FSS registers be programmed with 0x0000 using the `PROGC` command before the `ERASEB` command is used to erase the chip.

TABLE A-1: CHECKSUM COMPUTATION

Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAAAAAA at 0x0 and Last Code Address
dsPIC30F2010	Disabled	CFGB+SUM(0:001FFF)	0xD406	0xD208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F2011	Disabled	CFGB+SUM(0:001FFF)	0xD406	0xD208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F2012	Disabled	CFGB+SUM(0:001FFF)	0xD406	0xD208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F3010	Disabled	CFGB+SUM(0:003FFF)	0xA406	0xA208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F3011	Disabled	CFGB+SUM(0:003FFF)	0xA406	0xA208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F3012	Disabled	CFGB+SUM(0:003FFF)	0xA406	0xA208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F3013	Disabled	CFGB+SUM(0:003FFF)	0xA406	0xA208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F3014	Disabled	CFGB+SUM(0:003FFF)	0xA406	0xA208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F4011	Disabled	CFGB+SUM(0:007FFF)	0x4406	0x4208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F4012	Disabled	CFGB+SUM(0:007FFF)	0x4406	0x4208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F4013	Disabled	CFGB+SUM(0:007FFF)	0x4406	0x4208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F5011	Disabled	CFGB+SUM(0:00AFFF)	0xFC06	0xFA08
	Enabled	CFGB	0x0404	0x0404
dsPIC30F5013	Disabled	CFGB+SUM(0:00AFFF)	0xFC06	0xFA08
	Enabled	CFGB	0x0404	0x0404
dsPIC30F5015	Disabled	CFGB+SUM(0:00AFFF)	0xFC06	0xFA08
	Enabled	CFGB	0x0404	0x0404

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 ((FOSC&0xC10F) + (FWDT&0x803F) + (FBORPOR&0x87B3) + (FBS&0x310F) + (FSS&0x330F) + (FGS&0x0007) + (FICD&0xC003))