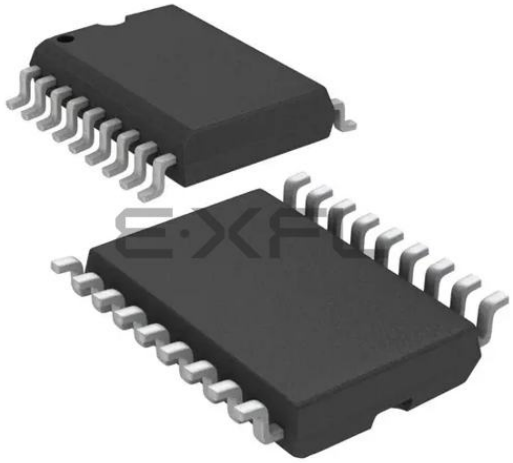


Welcome to E-XFL.COM

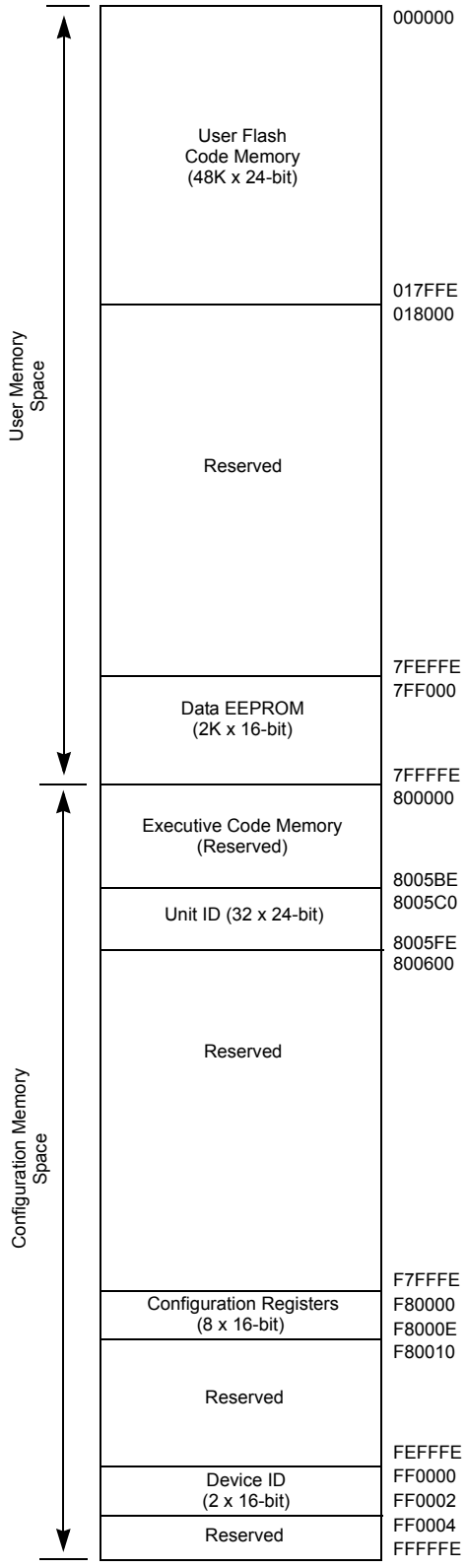
What is "Embedded - Microcontrollers"?

"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, POR, PWM, WDT
Number of I/O	12
Program Memory Size	24KB (8K 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 8x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	18-SOIC (0.295", 7.50mm Width)
Supplier Device Package	18-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/dspic30f3012t-20i-so

FIGURE 2-2: PROGRAM MEMORY MAP

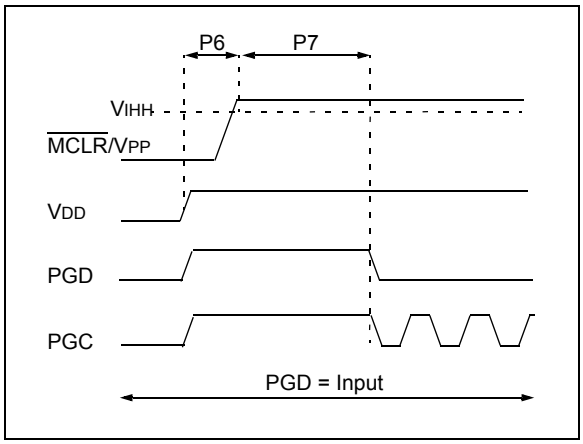


Note: The address boundaries for user Flash code memory and data EEPROM are device-dependent.

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.

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

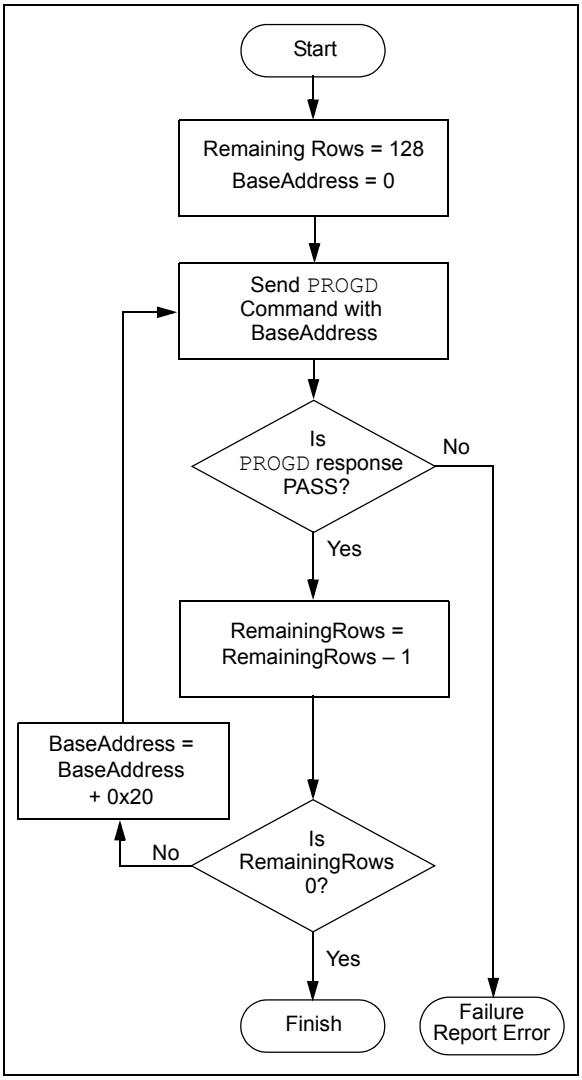


TABLE 5-7: CONFIGURATION BITS DESCRIPTION

Bit Field	Register	Description
FWPSA<1:0>	FWDT	**Watchdog Timer Prescaler A** 11 = 1:512 10 = 1:64 01 = 1:8 00 = 1:1
FWPSB<3:0>	FWDT	**Watchdog Timer Prescaler B** 1111 = 1:16 1110 = 1:15 • • • 0001 = 1:2 0000 = 1:1
FWDTEN	FWDT	**Watchdog Enable** 1 = Watchdog enabled (LPRC oscillator cannot be disabled. Clearing the SWDTEN bit in the RCON register will have no effect) 0 = Watchdog disabled (LPRC oscillator can be disabled by clearing the SWDTEN bit in the RCON register)
MCLREN	FBORPOR	**Master Clear Enable** $\overline{\text{MCLR}}$ 1 = Master Clear pin ($\overline{\text{MCLR}}$) is enabled 0 = $\overline{\text{MCLR}}$ pin is disabled
PWMPIN	FBORPOR	**Motor Control PWM Module Pin Mode** 1 = PWM module pins controlled by PORT register at device Reset (tri-stated) 0 = PWM module pins controlled by PWM module at device Reset (configured as output pins)
HPOL	FBORPOR	**Motor Control PWM Module High-Side Polarity** 1 = PWM module high-side output pins have active-high output polarity 0 = PWM module high-side output pins have active-low output polarity
LPOL	FBORPOR	**Motor Control PWM Module Low-Side Polarity** 1 = PWM module low-side output pins have active-high output polarity 0 = PWM module low-side output pins have active-low output polarity
BOREN	FBORPOR	**PBOR Enable** 1 = PBOR enabled 0 = PBOR disabled
BORV<1:0>	FBORPOR	**Brown-out Voltage Select** 11 = 2.0V (not a valid operating selection) 10 = 2.7V 01 = 4.2V 00 = 4.5V
FPWRT<1:0>	FBORPOR	**Power-on Reset Timer Value Select** 11 = PWRT = 64 ms 10 = PWRT = 16 ms 01 = PWRT = 4 ms 00 = Power-up Timer disabled
RBS<1:0>	FBS	**Boot Segment Data RAM Code Protection (only present in dsPIC30F5011/5013/6010A/6011A/6012A/6013A/6014A/6015)** 11 = No Data RAM is reserved for Boot Segment 10 = Small-sized Boot RAM [128 bytes of RAM are reserved for Boot Segment] 01 = Medium-sized Boot RAM [256 bytes of RAM are reserved for Boot Segment] 00 = Large-sized Boot RAM [512 bytes of RAM are reserved for Boot Segment in dsPIC30F5011/5013, and 1024 bytes in dsPIC30F6010A/6011A/6012A/6013A/6014A/6015]

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'.

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

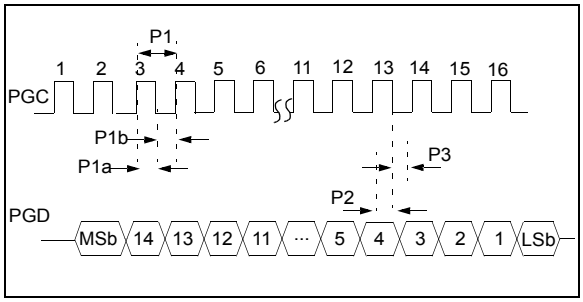
The programmer and programming executive have a master-slave relationship, where the programmer is the master programming device and the programming executive is the slave.

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

8.5.5 PROGP COMMAND

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

Field	Description
Opcode	0x5
Length	0x33
Reserved	0x0
Addr_MSB	MSB of 24-bit destination address
Addr_LS	LS 16 bits of 24-bit destination address
D_1	16-bit data word 1
D_2	16-bit data word 2
...	16-bit data word 3 through 47
D_48	16-bit data word 48

The PROGP command instructs the programming executive to program one row of code memory (32 instruction words) to the specified memory address. Programming begins with the row address specified in the command. The destination address should be a multiple of 0x40.

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

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

Expected Response (2 words):
0x1500
0x0002

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

8.5.6 PROGC COMMAND

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

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

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

Expected Response (2 words):
0x1600
0x0002

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

8.5.9 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.

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.

9.2.3 QE_Code FIELD

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

When the programming executive processes one of the two query commands (QBLANK or QVER), the returned opcode is always PASS and the QE_Code holds the query response data. The format of the QE_Code for both queries is shown in Table 9-3.

TABLE 9-3: QE_Code FOR QUERIES

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

When the programming executive processes any command other than a Query, the QE_Code represents an error code. Supported error codes are shown in Table 9-4. If a command is successfully processed, the returned QE_Code is set to 0x0, which indicates that there was no error in the command processing. If the verify of the programming for the PROGD, PROGP or PROGC command fails, the QE_Code is set to 0x1. For all other programming executive errors, the QE_Code is 0x2.

TABLE 9-4: QE_Code FOR NON-QUERY COMMANDS

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

9.2.4 RESPONSE LENGTH

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

With the exception of the response for the READD and READP commands, the length of each response is only 2 words.

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

The response to the READP command uses the packed instruction word format described in **Section 8.3 "Packed Data Format"**. When reading an odd number of program memory words (N odd), the response to the READP command is (3 • (N + 1)/2 + 2) words. When reading an even number of program memory words (N even), the response to the READP command is (3 • N/2 + 2) words.

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

If an individual Segment Erase operation is required, the NVMCON value must be replaced by the value for the corresponding Segment Erase operation.

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

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

Command (Binary)	Data (Hexadecimal)	Description
Step 1: Exit the Reset vector.		
0000 0000 0000	040100 040100 000000	GOTO 0x100 GOTO 0x100 NOP
Step 2: Set NVMCON to program the FBS Configuration register.(1)		
0000 0000	24008A 883B0A	MOV #0x4008, W10 MOV W10, NVMCON
Step 3: Initialize the TBLPAG and write pointer (W7) for TBLWT instruction for Configuration register.(1)		

0000
0000
0000
	200F80
	880190
	200067
		MOV #0xF8, W0
		MOV W0, TBLPAG
		MOV #0x6, W7

Step 4: Load the Configuration Register data to W6.(1)		
0000 0000	EB0300 000000	CLR W6 NOP
Step 5: Load the Configuration Register write latch. Advance W7 to point to next Configuration register.(1)		
0000	BB1B86	TBLWTL W6, [W7++]
Step 6: Unlock the NVMCON for programming the Configuration register.(1)		

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

Step 7: Initiate the programming cycle.(1)		
0000 0000 0000 – 0000 0000 0000 0000 0000	A8E761 000000 000000 – 000000 000000 A9E761 000000 000000	BSET NVMCON, #WR NOP NOP Externally time 2 ms NOP NOP BCLR NVMCON, #WR NOP NOP
Step 8: Repeat steps 5-7 one time to program 0x0000 to RESERVED2 Configuration register.(1)		
Step 9: Set the NVMCON to erase all Program Memory.		
00000 0000	2407FA 883B0A	MOV #0x407F, W10 MOV W10, NVMCON
Step 10: Unlock the NVMCON for programming.		

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.

11.6 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 11-7: 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.		

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

Command (Binary)	Data (Hexadecimal)	Description
Step 5: Set the read pointer (W6) and load the (next set of) write latches.		
0000	EB0300	CLR W6
0000	000000	NOP
0000	BB0BB6	TBLWTL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP
0000	BBDBB6	TBLWTH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BBEBB6	TBLWTH.B [W6++], [++W7]
0000	000000	NOP
0000	000000	NOP
0000	BB1BB6	TBLWTL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BB0BB6	TBLWTL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP
0000	BBDBB6	TBLWTH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BBEBB6	TBLWTH.B [W6++], [++W7]
0000	000000	NOP
0000	000000	NOP
0000	BB1BB6	TBLWTL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
Step 6: Repeat steps 4-5 eight times to load the write latches for 32 instructions.		
Step 7: Unlock the NVMCON for writing.		
0000	200558	MOV #0x55, W8
0000	883B38	MOV W8, NVMKEY
0000	200AA9	MOV #0xAA, W9
0000	883B39	MOV W9, NVMKEY
Step 8: Initiate the write cycle.		
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
–	–	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 2-9 until all code memory is programmed.		

11.9 Writing Data EEPROM

The procedure for writing data EEPROM is very similar to the procedure for writing code memory, except that fewer words are programmed in each operation. When writing data EEPROM, one row of data EEPROM is programmed at a time. Each row consists of sixteen 16-bit data words. Since fewer words are programmed during each operation, only working registers W0:W3 are used as temporary holding registers for the data to be programmed.

Table 11-9 shows the ICSP programming details for writing data EEPROM. Note that a different NVMCON value is required to write to data EEPROM, and that the TBLPAG register is hard-coded to 0x7F (the upper byte address of all locations of data EEPROM).

TABLE 11-9: SERIAL INSTRUCTION EXECUTION FOR WRITING DATA EEPROM

Command (Binary)	Data (Hexadecimal)	Description
Step 1: Exit the Reset vector.		
0000	040100	GOTO 0x100
0000	040100	GOTO 0x100
0000	000000	NOP
Step 2: Set the NVMCON to write 16 data words.		
0000	24005A	MOV #0x4005, W10
0000	883B0A	MOV W10, NVMCON
Step 3: Initialize the write pointer (W7) for TBLWT instruction.		
0000	2007F0	MOV #0x7F, W0
0000	880190	MOV W0, TBLPAG
0000	2xxxx7	MOV #<DestinationAddress15:0>, W7
Step 4: Load W0:W3 with the next 4 data words to program.		
0000	2xxxx0	MOV #<WORD0>, W0
0000	2xxxx1	MOV #<WORD1>, W1
0000	2xxxx2	MOV #<WORD2>, W2
0000	2xxxx3	MOV #<WORD3>, W3
Step 5: Set the read pointer (W6) and load the (next set of) write latches.		
0000	EB0300	CLR W6
0000	000000	NOP
0000	BB1BB6	TBLWTL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BB1BB6	TBLWTL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BB1BB6	TBLWTL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BB1BB6	TBLWTL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
Step 6: Repeat steps 4-5 four times to load the write latches for 16 data words.		

TABLE 11-9: SERIAL INSTRUCTION EXECUTION FOR WRITING DATA EEPROM (CONTINUED)

Command (Binary)	Data (Hexadecimal)	Description
Step 7: Unlock the NVMCON for writing.		
0000	200558	MOV #0x55, W8
0000	883B38	MOV W8, NVMKEY
0000	200AA9	MOV #0xAA, W9
0000	883B39	MOV W9, NVMKEY
Step 8: Initiate the write cycle.		
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
—	—	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 2-9 until all data memory is programmed.		

11.10 Reading Code Memory

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

Table 11-10 shows the ICSP programming details for reading code memory. In Step 1, the Reset vector is exited. In Step 2, the 24-bit starting source address for reading is loaded into the TBLPAG and W6 registers. The upper byte of the starting source address is stored to TBLPAG, while the lower 16 bits of the source address are stored to W6.

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

TABLE 11-10: SERIAL INSTRUCTION EXECUTION FOR READING CODE 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	200xx0	MOV #<SourceAddress23:16>, W0
0000	880190	MOV W0, TBLPAG
0000	2xxxx6	MOV #<SourceAddress15:0>, W6
Step 3: Initialize the write pointer (W7) and store the next four locations of code memory to W0:W5.		
0000	EB0380	CLR W7
0000	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	BA0BB6	TBLRDL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP

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

TABLE 12-2: READING EXECUTIVE 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	—	Clock out contents of VISI register
0000	883C21	MOV W1, VISI
0000	000000	NOP
0001	—	Clock out contents of VISI register
0000	883C22	MOV W2, VISI
0000	000000	NOP
0001	—	Clock out contents of VISI register
0000	883C23	MOV W3, VISI
0000	000000	NOP
0001	—	Clock out contents of VISI register
0000	883C24	MOV W4, VISI
0000	000000	NOP
0001	—	Clock out contents of VISI register
0000	883C25	MOV W5, VISI
0000	000000	NOP
0001	—	Clock out contents of VISI register
Step 5: Reset the device internal PC.		
0000	040100	GOTO 0x100
0000	000000	NOP
Step 6: Repeat Steps 3-5 until all 736 instruction words of executive memory are read.		