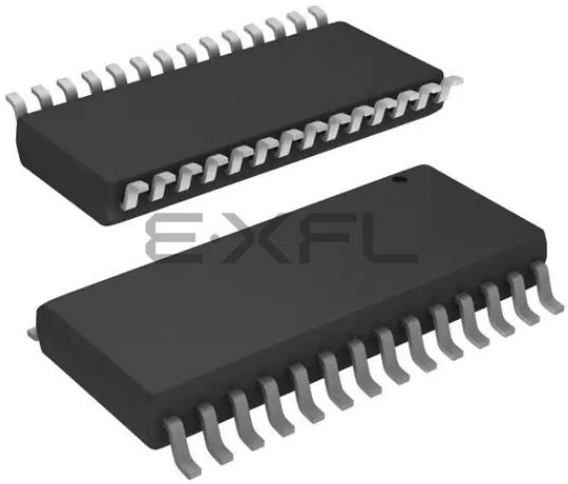
Microchip Technology - DSPIC30F4012T-20E/SO Datasheet



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.

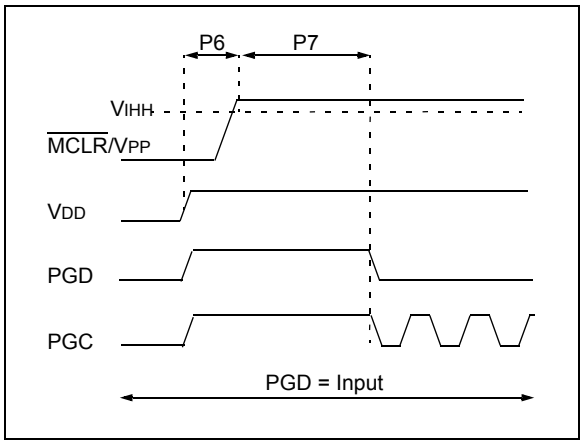
Applications of "Embedded - Microcontrollers"

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 ~ 125°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-20e-so

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 Code Memory Programming

5.5.1 OVERVIEW

The Flash code memory array consists of 512 rows of thirty-two, 24-bit instructions. Each panel stores 16K instruction words, and each dsPIC30F device has either 1, 2 or 3 memory panels (see Table 5-2).

TABLE 5-2: DEVICE CODE MEMORY SIZE

Device	Code Size (24-bit Words)	Number of Rows	Number of Panels
dsPIC30F2010	4K	128	1
dsPIC30F2011	4K	128	1
dsPIC30F2012	4K	128	1
dsPIC30F3010	8K	256	1
dsPIC30F3011	8K	256	1
dsPIC30F3012	8K	256	1
dsPIC30F3013	8K	256	1
dsPIC30F3014	8K	256	1
dsPIC30F4011	16K	512	1
dsPIC30F4012	16K	512	1
dsPIC30F4013	16K	512	1
dsPIC30F5011	22K	704	2
dsPIC30F5013	22K	704	2
dsPIC30F5015	22K	704	2
dsPIC30F5016	22K	704	2
dsPIC30F6010	48K	1536	3
dsPIC30F6010A	48K	1536	3
dsPIC30F6011	44K	1408	3
dsPIC30F6011A	44K	1408	3
dsPIC30F6012	48K	1536	3
dsPIC30F6012A	48K	1536	3
dsPIC30F6013	44K	1408	3
dsPIC30F6013A	44K	1408	3
dsPIC30F6014	48K	1536	3
dsPIC30F6014A	48K	1536	3
dsPIC30F6015	48K	1536	3

5.5.2 PROGRAMMING METHODOLOGY

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

A flowchart for programming of code memory is illustrated in Figure 5-3. In this example, all 48K instruction words of a dsPIC30F6014A device are programmed. First, the number of commands to send (called 'RemainingCmds' in the flowchart) is set to 1536 and the destination address (called 'BaseAddress') is set to '0'.

Next, one row in the device is programmed with a PROGP command. Each PROGP command contains data for one row of code memory of the dsPIC30F6014A. After the first command is processed successfully, 'RemainingCmds' is decremented by 1 and compared to 0. Since there are more PROGP commands to send, 'BaseAddress' is incremented by 0x40 to point to the next row of memory.

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

FIGURE 5-3: FLOWCHART FOR PROGRAMMING dsPIC30F6014A CODE MEMORY

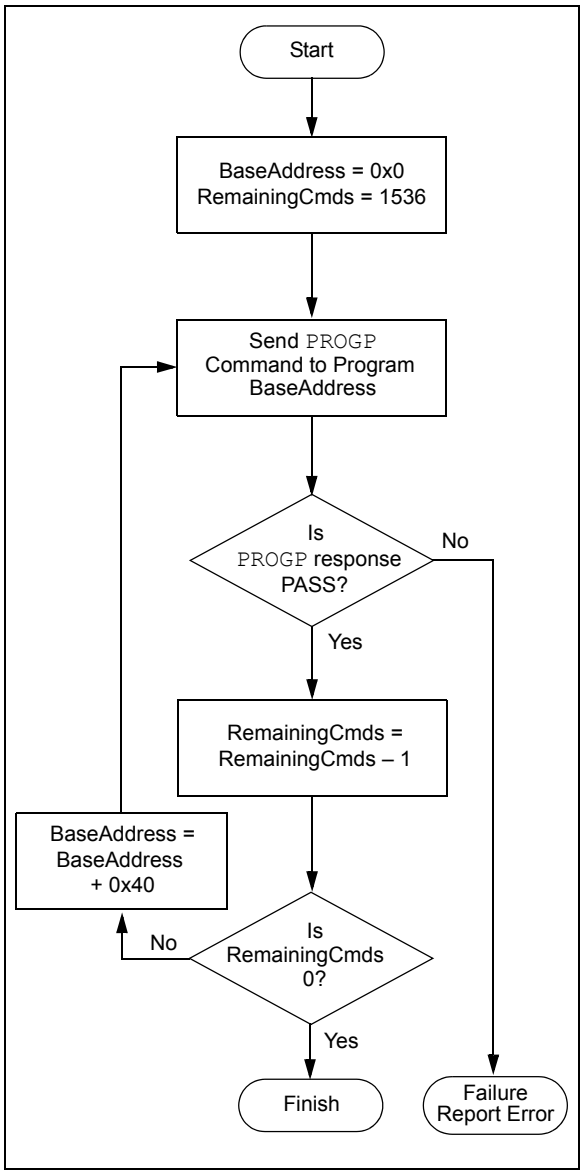


TABLE 5-6: FOSC CONFIGURATION BITS DESCRIPTION FOR dsPIC30F2011/2012, dsPIC30F3010/3011/3012/3013/3014, dsPIC30F4013, dsPIC30F5015/5016, dsPIC30F6010A/6011A/6012A/6013A/6014A AND dsPIC30F6015 (CONTINUED)

Bit Field	Register	Description
FPR<4:0>	FOSC	**Alternate Oscillator Mode (when FOS<2:0> = 011b)** 1xxxx = Reserved (do not use) 0111x = Reserved (do not use) 01101 = Reserved (do not use) 01100 = ECIO – External clock. OSC2 pin is I/O 01011 = EC – External clock. OSC2 pin is system clock output (FOSC/4) 01010 = Reserved (do not use) 01001 = ERC – External RC oscillator. OSC2 pin is system clock output (FOSC/4) 01000 = ERCIO – External RC oscillator. OSC2 pin is I/O 00111 = Reserved (do not use) 00110 = Reserved (do not use) 00101 = Reserved (do not use) 00100 = XT – XT crystal oscillator (4 MHz-10 MHz crystal) 00010 = HS – HS crystal oscillator (10 MHz-25 MHz crystal) 00001 = Reserved (do not use) 00000 = XTL – XTL crystal oscillator (200 kHz-4 MHz crystal)

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'.

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.

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 Command Descriptions

All commands that are supported by the programming executive are described in **Section 8.5.1 "SCHECK Command"** through **Section 8.5.11 "QVER Command"**.

8.5.1 SCHECK COMMAND

15 12	11 0
Opcode	Length

Field	Description
Opcode	0x0
Length	0x1

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

Expected Response (2 words):
0x1000
0x0002

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

8.5.2 READD COMMAND

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

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

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

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

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

8.5.3 READP COMMAND

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

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

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

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

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

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

8.5.4 PROGD COMMAND

15 12	11 8	7 0
Opcode	Length	
Reserved	Addr_MSB	
Addr_LS		
D_1		
D_2		
...		
D_16		

Field	Description
Opcode	0x4
Length	0x13
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 words 3 through 15
D_16	16-bit data word 16

The PROGD command instructs the programming executive to program one row of data EEPROM. The data to be programmed is specified by the 16 data words (D_1, D_2,..., D_16) and is programmed to the destination address specified by Addr_MSB and Addr_LSB. The destination address should be a multiple of 0x20.

Once the row of data EEPROM has been programmed, the programming executive verifies the programmed data against the data in the command.

Expected Response (2 words):
0x1400
0x0002

Note:	Refer to Table 5-3 for data EEPROM size information.

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.

8.5.9 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.

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.

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.		

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.		

11.13 Reading the Application ID Word

The application ID word is stored at address 0x8005BE in executive code memory. To read this memory location, you must use the SIX control code to move this program memory location to the VISI register. The REGOUT control code must then be used to clock the contents of the VISI register out of the device. The corresponding control and instruction codes that must be serially transmitted to the device to perform this operation are shown in Table 11-13.

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

11.14 Exiting ICSP Mode

After confirming that the programming executive is resident in memory, or loading the programming executive, ICSP mode is exited by removing power to the device or bringing $\overline{MCLR}$ to V_{IL}. Programming can then take place by following the procedure outlined in **Section 5.0 "Device Programming"**.

TABLE 11-13: SERIAL INSTRUCTION EXECUTION FOR READING THE APPLICATION ID WORD

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 (W0) for TBLRD instruction.		
0000	200800	MOV #0x80, W0
0000	880190	MOV W0, TBLPAG
0000	205BE0	MOV #0x5BE, W0
0000	207841	MOV VISI, W1
0000	000000	NOP
0000	BA0890	TBLRDL [W0], [W1]
0000	000000	NOP
0000	000000	NOP
Step 3: Output the VISI register using the REGOUT command.		
0001	<VISI>	Clock out contents of the VISI register
0000	000000	NOP

12.2 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

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.