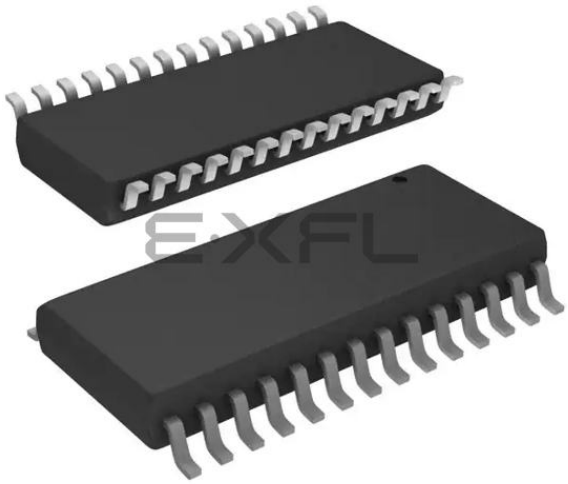

Welcome to E-XFL.COM

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	30 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	12KB (4K x 24)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 5.5V
Data Converters	A/D 6x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/dspic30f2010-30i-sog

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

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

TABLE 5-6: 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 5-7: 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 5-8: 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.

6.0 OTHER PROGRAMMING FEATURES

6.1 Erasing Memory

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

ERASEB provides several Bulk Erase options. Performing a Chip Erase with the ERASEB command clears all code memory, data EEPROM and code protection registers. Alternatively, ERASEB can be used to selectively erase either all code memory or data EEPROM. Erase options are summarized in Table 6-1.

TABLE 6-1: ERASE OPTIONS

Command	Affected Region
ERASEB	Entire chip(1) or all code memory or all data EEPROM, or erase by segment
ERASED	Specified rows of data EEPROM
ERASEP(2)	Specified rows of code memory

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

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

6.2 Modifying Memory

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

Instead, you can erase selective rows of code memory and data EEPROM using ERASEP and ERASED, respectively. You can then reprogram the modified rows with the PROGP and PROGD command pairs. In these cases, when code memory is programmed, single-panel programming must be specified in the PROGP command.

For modification of Advanced Code Protection bits for a particular segment, the entire chip must first be erased with the ERASEB command. Alternatively, on devices that support Advanced Security, individual segments (code and/or data EEPROM) may be erased, by suitably changing the MS (Memory Select) field in the ERASEB command. The code-protect Configuration bits can then be reprogrammed using the PROGC command.

Note: If read or write code protection is enabled for a segment, no modifications can be made to that segment until code protection is disabled. Code protection can only be disabled by performing a Chip Erase or by performing a Segment Erase operation for the required segment.

6.3 Reading Memory

The READD command reads the data EEPROM, Configuration bits and device ID of the device. This command only returns 16-bit data and operates on 16-bit registers. READD can be used to return the entire contents of data EEPROM.

The READP command reads the code memory of the device. This command only returns 24-bit data packed as described in **Section 8.3 "Packed Data Format"**. READP can be used to read up to 32K instruction words of code memory.

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

6.4 Programming Executive Software Version

At times, it may be necessary to determine the version of programming executive stored in executive memory. The QVER command performs this function. See **Section 8.5.11 "QVER Command"** for more details about this command.

6.5 Data EEPROM Information in the Hexadecimal File

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

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

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.

8.5.5 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.

8.5.7 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.

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 18: Unlock the NVMCON to erase 1 row of data memory.		
0000	200558	MOV #0x55, W8
0000	883B38	MOV W8, NVMKEY
0000	200AA9	MOV #0xAA, W9
0000	883B39	MOV W9, NVMKEY
Step 19: 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 20: Update the row address stored in NVMADR.		
0000	430307	ADD W6, W7, W6
0000	883B16	MOV W6, NVMADR
Step 21: Reset device internal PC.		
0000	040100	GOTO 0x100
0000	000000	NOP
Step 22: Repeat Steps 17-21 until all rows of data memory are erased.		

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 11-8: 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.		

11.11 Reading Configuration Memory

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

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

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

Command (Binary)	Data (Hexadecimal)	Description
Step 1: Exit the Reset vector.		
0000	040100	GOTO 0x100
0000	040100	GOTO 0x100
0000	000000	NOP
Step 2: Initialize TBLPAG, and the read pointer (W6) and the write pointer (W7) for TBLRD instruction.		
0000	200F80	MOV #0xF8, W0
0000	880190	MOV W0, TBLPAG
0000	EB0300	CLR W6
0000	EB0380	CLR W7
0000	000000	NOP
Step 3: Read the Configuration register and write it to the VISI register (located at 0x784).		
0000	BA0BB6	TBLRDL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP
0000	883C20	MOV W0, VISI
0000	000000	NOP
Step 4: Output the VISI register using the REGOUT command.		
0001	<VISI>	Clock out contents of VISI register
0000	000000	NOP
Step 5: Reset device internal PC.		
0000	040100	GOTO 0x100
0000	000000	NOP
Step 6: Repeat steps 3-5 six times to read all of configuration memory.		

11.12 Reading Data Memory

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

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

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

Command (Binary)	Data (Hexadecimal)	Description
Step 1: Exit the Reset vector.		
0000	040100	GOTO 0x100
0000	040100	GOTO 0x100
0000	000000	NOP
Step 2: Initialize TBLPAG and the read pointer (W6) for TBLRD instruction.		
0000	2007F0	MOV #0x7F, W0
0000	880190	MOV W0, TBLPAG
0000	2xxxx6	MOV #<SourceAddress15:0>, W6
Step 3: Initialize the write pointer (W7) and store the next four locations of code memory to W0:W5.		
0000	EB0380	CLR W7
0000	000000	NOP
0000	BA1BB6	TBLRDL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BA1BB6	TBLRDL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BA1BB6	TBLRDL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BA1BB6	TBLRDL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
Step 4: Output W0:W5 using the VISI register and REGOUT command.		
0000	883C20	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register
0000	000000	NOP
0000	883C21	MOV W1, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register
0000	000000	NOP
0000	883C22	MOV W2, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register
0000	000000	NOP
0000	883C23	MOV W3, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register
0000	000000	NOP
Step 5: Reset device internal PC.		
0000	040100	GOTO 0x100
0000	000000	NOP
Step 6: Repeat steps 3-5 until all desired data memory is read.		

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

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))

TABLE A-1: CHECKSUM COMPUTATION (CONTINUED)

Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAAAAAA at 0x0 and Last Code Address
dsPIC30F5016	Disabled	CFGB+SUM(0:00AFFF)	0xFC06	0xFA08
	Enabled	CFGB	0x0404	0x0404
dsPIC30F6010	Disabled	CFGB+SUM(0:017FFF)	0xC406	0xC208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F6010A	Disabled	CFGB+SUM(0:017FFF)	0xC406	0xC208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F6011	Disabled	CFGB+SUM(0:015FFF)	0xF406	0xF208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F6011A	Disabled	CFGB+SUM(0:015FFF)	0xF406	0xF208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F6012	Disabled	CFGB+SUM(0:017FFF)	0xC406	0xC208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F6012A	Disabled	CFGB+SUM(0:017FFF)	0xC406	0xC208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F6013	Disabled	CFGB+SUM(0:015FFF)	0xF406	0xF208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F6013A	Disabled	CFGB+SUM(0:015FFF)	0xF406	0xF208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F6014	Disabled	CFGB+SUM(0:017FFF)	0xC406	0xC208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F6014A	Disabled	CFGB+SUM(0:017FFF)	0xC406	0xC208
	Enabled	CFGB	0x0404	0x0404
dsPIC30F6015	Disabled	CFGB+SUM(0:017FFF)	0xC406	0xC208
	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))

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.

APPENDIX C: REVISION HISTORY

Note: Revision histories were not recorded for revisions A through H. The previous revision (J), was published in August 2007.

Revision K (November 2010)

This version of the document includes the following updates:

- Added Note three to **Section 5.2 "Entering Enhanced ICSP Mode"**
- Updated the first paragraph of **Section 10.0 "Device ID"**
- Updated Table 10-1: Device IDs
- Removed the VARIANT bit and updated the bit definition for the DEVID register in Table 10-2: dsPIC30F Device ID Registers
- Removed the VARIANT bit and updated the bit field definition and description for the DEVID register in Table 10-3: Device ID Bits Description
- Updated Note 3 in **Section 11.3 "Entering ICSP Mode"**
- Updated Step 11 in Table 11-4: Serial Instruction Execution for BUlk Erasing Program Memory (Only in Normal-voltage Systems)
- Updated Steps 5, 12 and 19 in Table 11-5: Serial Instruction Execution for Erasing Program Memory (Either in Low-voltage or Normal-voltage Systems)
- Updated Steps 5, 6 and 8 in Table 11-7: Serial Instruction Execution for Writing Configuration Registers
- Updated Steps 6 and 8 in Table 11-8: Serial Instruction Execution for Writing Code Memory
- Updated Steps 6 and 8 in Table 11-9: Serial Instruction Execution for Writing Data EEPROM
- Updated Entering ICSP™ Mode (see Figure 11-4)
- Updated Steps 4 and 11 in Table 12-1: Programming the Programming Executive
- Renamed parameters: P12 to P12a and P13 to P13a, and added parameters P12b and P13b in Table 13-1: AC/DC Characteristics