



Welcome to [E-XFL.COM](https://www.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, I <sup>2</sup> C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, Motor Control PWM, QEI, POR, PWM, WDT
Number of I/O	52
Program Memory Size	66KB (22K 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 16x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (10x10)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/microchip-technology/dspic30f5015t-20e-pt">https://www.e-xfl.com/product-detail/microchip-technology/dspic30f5015t-20e-pt</a>

# dsPIC30F Flash Programming Specification

## 3.0 PROGRAMMING EXECUTIVE APPLICATION

### 3.1 Programming Executive Overview

The programming executive resides in executive memory and is executed when Enhanced ICSP Programming mode is entered. The programming executive provides the mechanism for the programmer (host device) to program and verify the dsPIC30F, using a simple command set and communication protocol.

The following capabilities are provided by the programming executive:

- Read memory
  - Code memory and data EEPROM
  - Configuration registers
  - Device ID
- Erase memory
  - Bulk Erase by segment
  - Code memory (by row)
  - Data EEPROM (by row)
- Program memory
  - Code memory
  - Data EEPROM
  - Configuration registers
- Query
  - Blank Device
  - Programming executive software version

The programming executive performs the low-level tasks required for erasing and programming. This allows the programmer to program the device by issuing the appropriate commands and data.

The programming procedure is outlined in [Section 5.0 “Device Programming”](#).

### 3.2 Programming Executive Code Memory

The programming executive is stored in executive code memory and executes from this reserved region of memory. It requires no resources from user code memory or data EEPROM.

### 3.3 Programming Executive Data RAM

The programming executive uses the device's data RAM for variable storage and program execution. Once the programming executive has run, no assumptions should be made about the contents of data RAM.

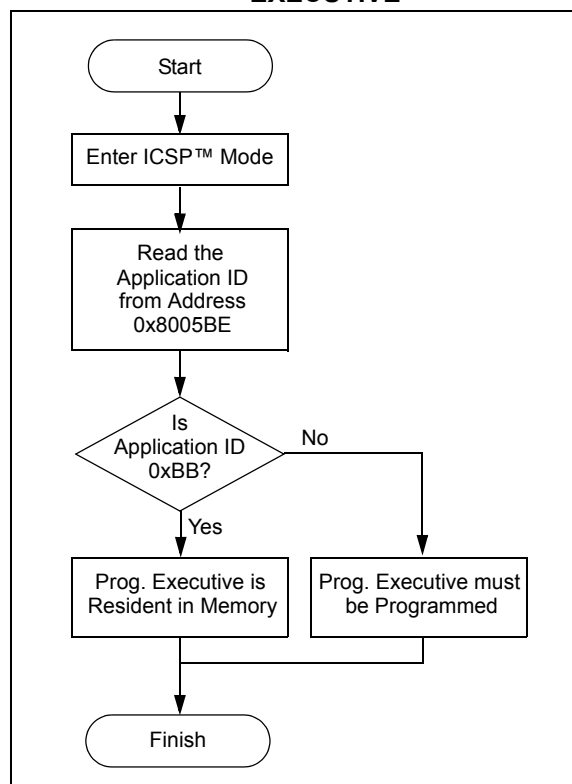
## 4.0 CONFIRMING THE CONTENTS OF EXECUTIVE MEMORY

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

First, ICSP mode is entered. The unique application ID word stored in executive memory is then read. If the programming executive is resident, the application ID word is 0xBB, which means programming can resume as normal. However, if the application ID word is not 0xBB, the programming executive must be programmed to Executive Code memory using the method described in [Section 12.0 “Programming the Programming Executive to Memory”](#).

[Section 11.0 “ICSP™ Mode”](#) describes the process for the ICSP programming method. [Section 11.13 “Reading the Application ID Word”](#) describes the procedure for reading the application ID word in ICSP mode.

**FIGURE 4-1: CONFIRMING PRESENCE OF THE PROGRAMMING EXECUTIVE**



# dsPIC30F Flash Programming Specification

## 5.0 DEVICE PROGRAMMING

### 5.1 Overview of the Programming Process

Once the programming executive has been verified in memory (or loaded if not present), the dsPIC30F can be programmed using the command set shown in [Table 5-1](#). A detailed description for each command is provided in [Section 8.0 “Programming Executive Commands”](#).

**TABLE 5-1: COMMAND SET SUMMARY**

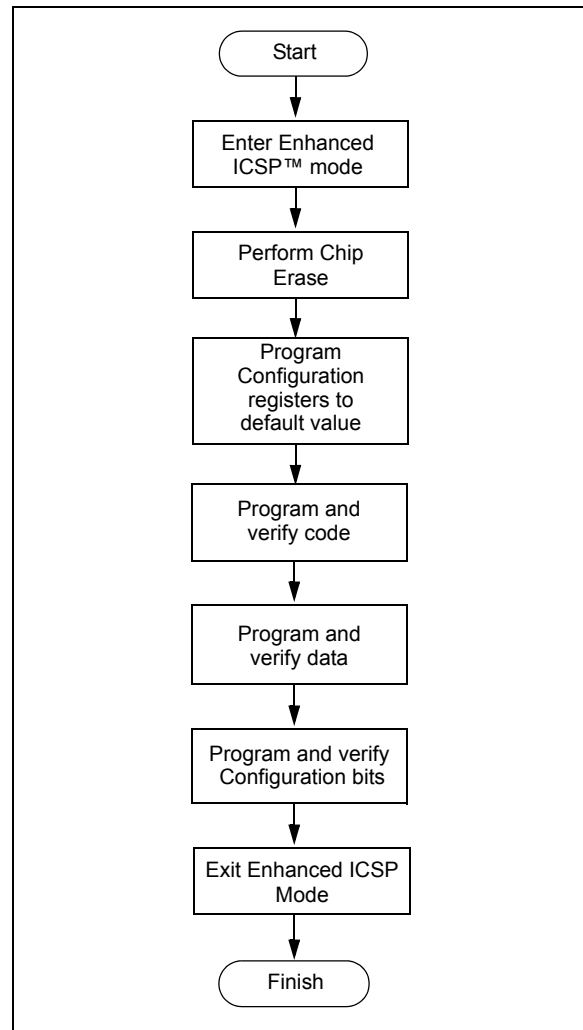
Command	Description
SCHECK	Sanity check
READD	Read data EEPROM, Configuration registers and device ID
READP	Read code memory
PROGD	Program one row of data EEPROM and verify
PROGP	Program one row of code memory and verify
PROGC	Program Configuration bits and verify
ERASEB	Bulk Erase, or erase by segment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

A high-level overview of the programming process is illustrated in [Figure 5-1](#). The process begins by entering Enhanced ICSP mode. The chip is then bulk erased, which clears all memory to ‘1’ and allows the device to be programmed. The Chip Erase is verified before programming begins. Next, the code memory, data Flash and Configuration bits are programmed. As these memories are programmed, they are each verified to ensure that programming was successful. If no errors are detected, the programming is complete and Enhanced ICSP mode is exited. If any of the verifications fail, the procedure should be repeated, starting from the Chip Erase.

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

[Section 5.2 “Entering Enhanced ICSP Mode”](#) through [Section 5.8 “Exiting Enhanced ICSP Mode”](#) describe the programming process in detail.

**FIGURE 5-1: PROGRAMMING FLOW**

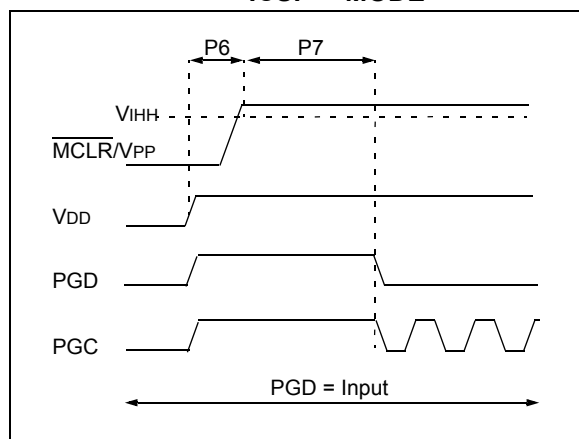


# dsPIC30F Flash Programming Specification

## 5.2 Entering Enhanced ICSP Mode

The Enhanced ICSP mode is entered by holding PGC and PGD high, and then raising MCLR/VPP to VIH (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.

# dsPIC30F Flash Programming Specification

**TABLE 5-7: CONFIGURATION BITS DESCRIPTION**

Bit Field	Register	Description
FWPSA<1:0>	FWDT	<b>Watchdog Timer Prescaler A</b> 11 = 1:512 10 = 1:64 01 = 1:8 00 = 1:1
FWPSB<3:0>	FWDT	<b>Watchdog Timer Prescaler B</b> 1111 = 1:16 1110 = 1:15 . . . 0001 = 1:2 0000 = 1:1
FWDTEN	FWDT	<b>Watchdog Enable</b> 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	<b>Master Clear Enable</b> 1 = Master Clear pin (MCLR) is enabled 0 = MCLR pin is disabled
PWMPIN	FBORPOR	<b>Motor Control PWM Module Pin Mode</b> 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	<b>Motor Control PWM Module High-Side Polarity</b> 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	<b>Motor Control PWM Module Low-Side Polarity</b> 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	<b>PBOR Enable</b> 1 = PBOR enabled 0 = PBOR disabled
BORV<1:0>	FBORPOR	<b>Brown-out Voltage Select</b> 11 = 2.0V (not a valid operating selection) 10 = 2.7V 01 = 4.2V 00 = 4.5V
FPWRT<1:0>	FBORPOR	<b>Power-on Reset Timer Value Select</b> 11 = PWRT = 64 ms 10 = PWRT = 16 ms 01 = PWRT = 4 ms 00 = Power-up Timer disabled
RBS<1:0>	FBS	<b>Boot Segment Data RAM Code Protection (only present in dsPIC30F5011/5013/6010A/6011A/6012A/6013A/6014A/6015)</b> 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 5-10: 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 <sup>(1)</sup>	HPOL <sup>(1)</sup>	LPOL <sup>(1)</sup>	BOREN	—	BORV<1:0>		—	—	FPWRT<1:0>	
0xF80006	FBS	—	—	Reserved <sup>(2)</sup>		—	—	—	Reserved <sup>(2)</sup>	—	—	—	—	Reserved <sup>(2)</sup>			
0xF80008	FSS	—	—	Reserved <sup>(2)</sup>		—	—	Reserved <sup>(2)</sup>		—	—	—	—	Reserved <sup>(2)</sup>			
0xF8000A	FGS	—	—	—	—	—	—	—	—	—	—	—	—	—	Reserved <sup>(3)</sup>	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 <sup>(1)</sup>	HPOL <sup>(1)</sup>	LPOL <sup>(1)</sup>	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').

# dsPIC30F Flash Programming Specification

## 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.

# dsPIC30F Flash Programming Specification

## 8.0 PROGRAMMING EXECUTIVE COMMANDS

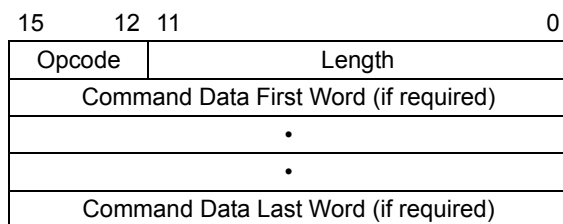
### 8.1 Command Set

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**



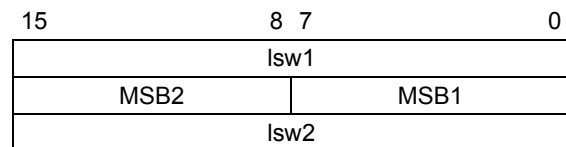
The command opcode must match one of those in the command set. Any command that is received which does not match the list in [Table 8-1](#) will return a “NACK” response (see [Section 9.2.1 “Opcode Field”](#)).

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**



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”](#).



# dsPIC30F Flash Programming Specification

## 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.

# dsPIC30F Flash Programming Specification

## 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.

# dsPIC30F Flash Programming Specification

## 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.

# dsPIC30F Flash Programming Specification

## 11.0 ICSP™ MODE

### 11.1 ICSP Mode

ICSP mode is a special programming protocol that allows you to read and write to the dsPIC30F programming executive. The ICSP mode is the second (and slower) method used to program the device. This mode also has the ability to read the contents of executive memory to determine whether the programming executive is present. This capability is accomplished by applying control codes and instructions serially to the device using pins PGC and PGD.

In ICSP mode, the system clock is taken from the PGC pin, regardless of the device's oscillator Configuration bits. All instructions are first shifted serially into an internal buffer, then loaded into the Instruction register and executed. No program fetching occurs from internal memory. Instructions are fed in 24 bits at a time. PGD is used to shift data in and PGC is used as both the serial shift clock and the CPU execution clock.

Data is transmitted on the rising edge and latched on the falling edge of PGC. For all data transmissions, the Least Significant bit (LSb) is transmitted first.

**Note 1:** During ICSP operation, the operating frequency of PGC must not exceed 5 MHz.

**2:** Because ICSP is slower, it is recommended that only Enhanced ICSP (E-ICSP) mode be used for device programming, as described in [Section 5.1 "Overview of the Programming Process"](#).

### 11.2 ICSP Operation

Upon entry into ICSP mode, the CPU is idle. Execution of the CPU is governed by an internal state machine. A 4-bit control code is clocked in using PGC and PGD, and this control code is used to command the CPU (see [Table 11-1](#)).

The SIX control code is used to send instructions to the CPU for execution, while the REGOUT control code is used to read data out of the device via the VISI register. The operation details of ICSP mode are provided in [Section 11.2.1 "SIX Serial Instruction Execution"](#) and [Section 11.2.2 "REGOUT Serial Instruction Execution"](#).

**TABLE 11-1: CPU CONTROL CODES IN ICSP™ MODE**

4-bit Control Code	Mnemonic	Description
0000b	SIX	Shift in 24-bit instruction and execute.
0001b	REGOUT	Shift out the VISI register.
0010b-1111b	N/A	Reserved.

#### 11.2.1 SIX SERIAL INSTRUCTION EXECUTION

The SIX control code allows execution of dsPIC30F assembly instructions. When the SIX code is received, the CPU is suspended for 24 clock cycles as the instruction is then clocked into the internal buffer. Once the instruction is shifted in, the state machine allows it to be executed over the next four clock cycles. While the received instruction is executed, the state machine simultaneously shifts in the next 4-bit command (see [Figure 11-2](#)).

**Note 1:** Coming out of the ICSP entry sequence, the first 4-bit control code is always forced to SIX and a forced NOP instruction is executed by the CPU. Five additional PGC clocks are needed on start-up, thereby resulting in a 9-bit SIX command instead of the normal 4-bit SIX command. After the forced SIX is clocked in, ICSP operation resumes as normal (the next 24 clock cycles load the first instruction word to the CPU). See [Figure 11-1](#) for details.

**2:** TBLRDH, TBLRDL, TBLWTH and TBLWTL instructions must be followed by a NOP instruction.

# dsPIC30F Flash Programming Specification

## 11.8 Writing Code Memory

The procedure for writing code memory is similar to the procedure for clearing the Configuration registers, except that 32 instruction words are programmed at a time. To facilitate this operation, working registers W0:W5 are used as temporary holding registers for the data to be programmed.

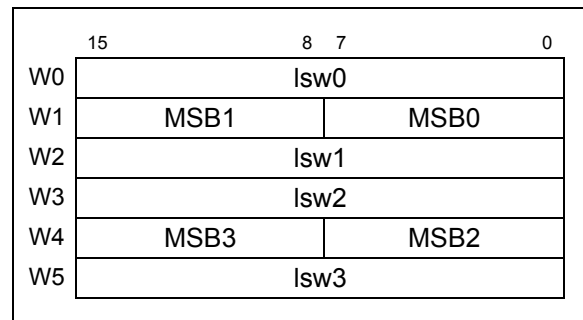
Table 11-8 shows the ICSP programming details, including the serial pattern with the ICSP command code, which must be transmitted Least Significant bit first using the PGC and PGD pins (see Figure 11-2). In Step 1, the Reset vector is exited. In Step 2, the NVMCON register is initialized for single-panel programming of code memory. In Step 3, the 24-bit starting destination address for programming is loaded into the TBLPAG register and W7 register. The upper byte of the starting destination address is stored to TBLPAG, while the lower 16 bits of the destination address are stored to W7.

To minimize the programming time, the same packed instruction format that the programming executive uses is utilized (Figure 8-2). In Step 4, four packed instruction words are stored to working registers W0:W5 using the MOV instruction and the read pointer W6 is initialized. The contents of W0:W5 holding the packed instruction word data is shown in Figure 11-4.

In Step 5, eight TBLWT instructions are used to copy the data from W0:W5 to the write latches of code memory. Since code memory is programmed 32 instruction words at a time, Steps 4 and 5 are repeated eight times to load all the write latches (Step 6).

After the write latches are loaded, programming is initiated by writing to the NVMKEY and NVMCON registers in Steps 7 and 8. In Step 9, the internal PC is reset to 0x100. This is a precautionary measure to prevent the PC from incrementing into unimplemented memory when large devices are being programmed. Lastly, in Step 10, Steps 2-9 are repeated until all of code memory is programmed.

**FIGURE 11-5: PACKED INSTRUCTION WORDS IN W0:W5**



**TABLE 11-8: SERIAL INSTRUCTION EXECUTION FOR WRITING CODE MEMORY**

Command (Binary)	Data (Hexadecimal)	Description
<b>Step 1: Exit the Reset vector.</b>		
0000	040100	GOTO 0x100
0000	040100	GOTO 0x100
0000	000000	NOP
<b>Step 2: Set the NVMCON to program 32 instruction words.</b>		
0000	24001A	MOV #0x4001, W10
0000	883B0A	MOV W10, NVMCON
<b>Step 3: Initialize the write pointer (W7) for TBLWT instruction.</b>		
0000	200xx0	MOV #<DestinationAddress23:16>, W0
0000	880190	MOV W0, TBLPAG
0000	2xxxx7	MOV #<DestinationAddress15:0>, W7
<b>Step 4: Initialize the read pointer (W6) and load W0:W5 with the next 4 instruction words to program.</b>		
0000	2xxxx0	MOV #<LSW0>, W0
0000	2xxxx1	MOV #<MSB1:MSB0>, W1
0000	2xxxx2	MOV #<LSW1>, W2
0000	2xxxx3	MOV #<LSW2>, W3
0000	2xxxx4	MOV #<MSB3:MSB2>, W4
0000	2xxxx5	MOV #<LSW3>, W5

# dsPIC30F Flash Programming Specification

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

# dsPIC30F Flash Programming Specification

## 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
<b>Step 1: Exit the Reset vector.</b>		
0000	040100	GOTO 0x100
0000	040100	GOTO 0x100
0000	000000	NOP
<b>Step 2: Initialize TBLPAG and the read pointer (W6) for TBLRD instruction.</b>		
0000	200xx0	MOV #<SourceAddress23:16>, W0
0000	880190	MOV W0, TBLPAG
0000	2xxxx6	MOV #<SourceAddress15:0>, W6
<b>Step 3: Initialize the write pointer (W7) and store the next four locations of code memory to W0:W5.</b>		
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

# dsPIC30F Flash Programming Specification

## 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
<b>Step 1: Exit the Reset vector.</b>		
0000	040100	GOTO 0x100
0000	040100	GOTO 0x100
0000	000000	NOP
<b>Step 2: Initialize TBLPAG and the read pointer (W6) for TBLRD instruction.</b>		
0000	2007F0	MOV #0x7F, W0
0000	880190	MOV W0, TBLPAG
0000	2xxxx6	MOV #<SourceAddress15:0>, W6
<b>Step 3: Initialize the write pointer (W7) and store the next four locations of code memory to W0:W5.</b>		
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
<b>Step 4: Output W0:W5 using the VISI register and REGOUT command.</b>		
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
<b>Step 5: Reset device internal PC.</b>		
0000	040100	GOTO 0x100
0000	000000	NOP
<b>Step 6: Repeat steps 3-5 until all desired data memory is read.</b>		



# dsPIC30F Flash Programming Specification

## 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) + (FWDTE&0x803F) + (FBORPOR&0x87B3) + (FBS&0x310F) + (FSS&0x330F) + (FGS&0x0007) + (FICD&0xC003))

# dsPIC30F Flash Programming Specification

**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))

# dsPIC30F Flash Programming Specification

---

## 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:

```
:BBAAAATTTHHHH...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

<b>Note:</b> 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](#)

---

**Note the following details of the code protection feature on Microchip devices:**

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

---

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

#### **Trademarks**

The Microchip name and logo, the Microchip logo, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, PIC<sup>32</sup> logo, rfPIC and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Omniscent Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICTail, REAL ICE, rLAB, Select Mode, Total Endurance, TSHARC, UniWinDriver, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2010, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

ISBN: 978-1-60932-636-4

*Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*

**QUALITY MANAGEMENT SYSTEM**  
**CERTIFIED BY DNV**  
**== ISO/TS 16949:2002 ==**