dsPIC30F Flash Programming Specification

1.0 OVERVIEW AND SCOPE

This document defines the programming specification for the dsPIC30F family of Digital Signal Controllers. The programming specification is required only for developers of third party tools that are used to program dsPIC30F devices. Customers using dsPIC30F devices should use development tools that already provide support for device programming.

This document includes programming specifications for the following devices:

- dsPIC30F2010, 2011, 2012
- dsPIC30F3010, 3011, 3012, 3013, 3014
- dsPIC30F4011, 4012, 4013
- dsPIC30F5011, 5013, 5015
- dsPIC30F6010, 6011, 6012, 6013, 6014
- dsPIC30F6011A, 6012A, 6013A, 6014A

2.0 PROGRAMMING OVERVIEW OF THE dsPIC30F

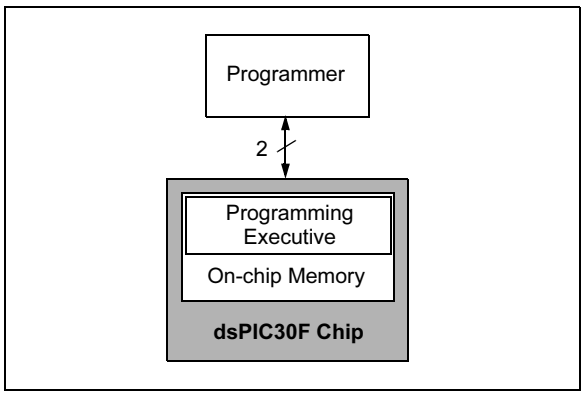
The dsPIC30F family of Digital Signal Controllers contains a region of on-chip memory used to simplify device programming. This region of memory can store a programming executive, which allows the dsPIC30F to be programmed faster than by traditional means. Once the programming executive is stored to memory by an external programmer (such as Microchip's MPLAB® ICD 2 or PRO MATE® II), it can then interact with the external programmer to efficiently program devices.

The programmer and programming executive have a master-slave relationship, where the programmer is the master programming device and the programming executive is the slave, as illustrated in Figure 2-1.

Two different methods are used to program the chip in the user's system. One method uses the In-Circuit Serial Programming™ (ICSP™) protocol and works with the programming executive. The other method uses Standard DUT Programming (STDP) protocol and does not use the programming executive.

The ICSP protocol uses the faster, high-voltage method that takes advantage of the programming executive. The programming executive provides all the necessary functionality to erase, program and verify the chip through a small command set. The command set allows the programmer to program the dsPIC30F without having to deal with the low-level programming protocols of the chip.

FIGURE 2-1: OVERVIEW OF dsPIC30F PROGRAMMING



The STDP programming method does not use the programming executive. It provides native, low-level programming capability to erase, program and verify the chip. This method is significantly slower because it uses control codes to serially execute instructions on the dsPIC30F device.

This specification describes both the ICSP and STDP programming methods. **Section 3.0 "Programming Executive Application"** describes the programming executive Application and **Section 5.0 "Device Programming"** describes its applications programmer's interface for the host programmer. **Section 11.0 "STDP Mode"** describes the STDP programming method.

2.1 Hardware Requirements

In ICSP mode, the dsPIC30F requires two programmable power supplies: one for VDD and one for MCLR. For bulk erase programming, which is required for erasing code protection bits, VDD must be greater than 4.5 volts. Refer to **Section 13.0 "AC/DC Characteristics and Timing Requirements"** for additional hardware parameters.

2.2 Pin Diagrams

The pin diagrams for the dsPIC30F family are shown in Figure 2-2 through Figure 2-26. Only the pins identified in Table are used for device programming. Refer to the appropriate device data sheet for complete pin descriptions.

TABLE 2-1: dsPIC30F PIN DESCRIPTIONS DURING PROGRAMMING

Pin Name	Pin Type	Pin Description
MCLR/VPP	P	Programming Enable
VDD	P	Power Supply
VSS	P	Ground
PGC	I	Serial Clock
PGD	I/O	Serial Data

Legend: I = Input, O = Output, P = Power

FIGURE 2-2: PIN DIAGRAMS (18-PIN PDIP, 18-PIN SOIC)

18-Pin PDIP and SOIC

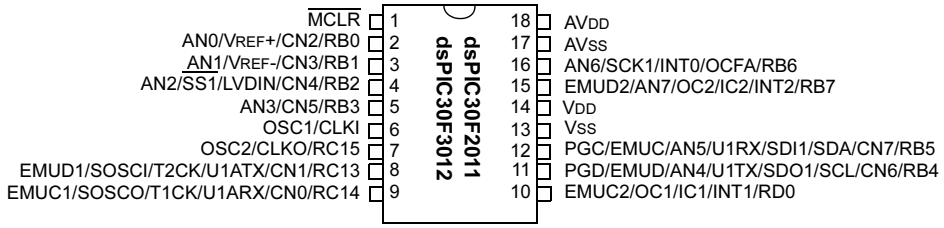
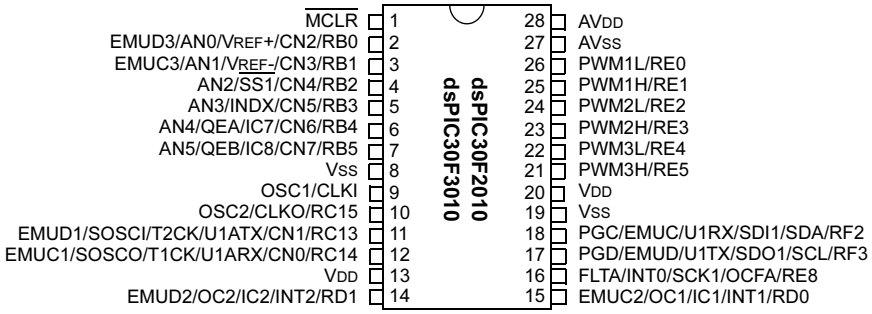
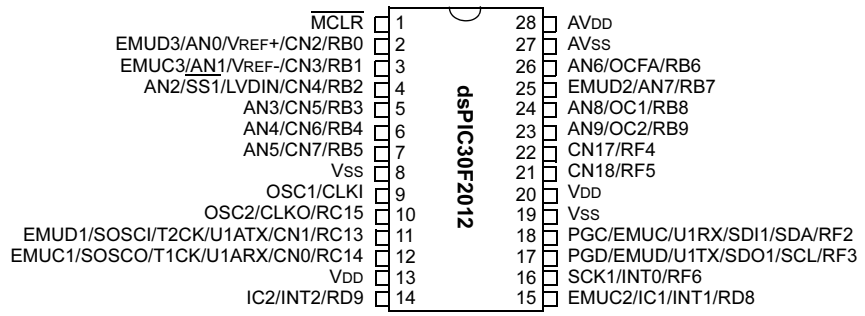


FIGURE 2-3: PIN DIAGRAMS (28-PIN PDIP, 28-PIN SOIC)

28-Pin PDIP and SOIC

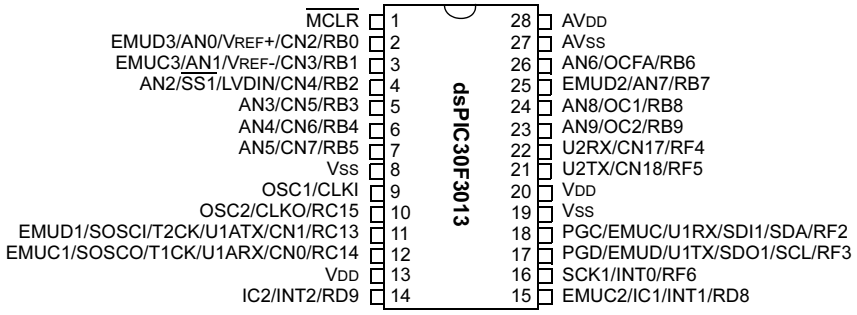




 © 2004 Microchip Technology Inc.

FIGURE 2-4: PIN DIAGRAMS (28-PIN PDIP, 28-PIN SOIC)

28-Pin PDIP and SOIC



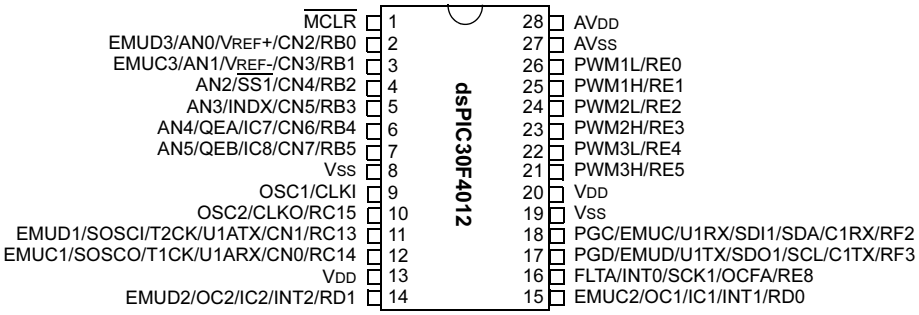
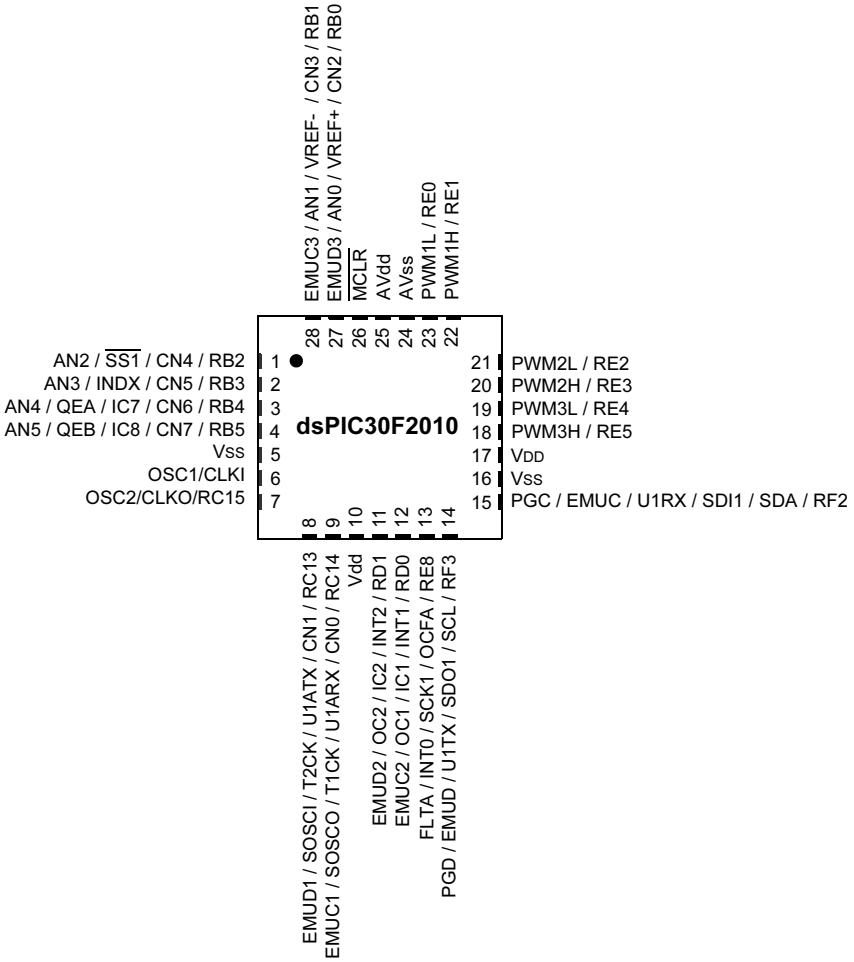


FIGURE 2-5: PIN DIAGRAM (28-PIN QFN)

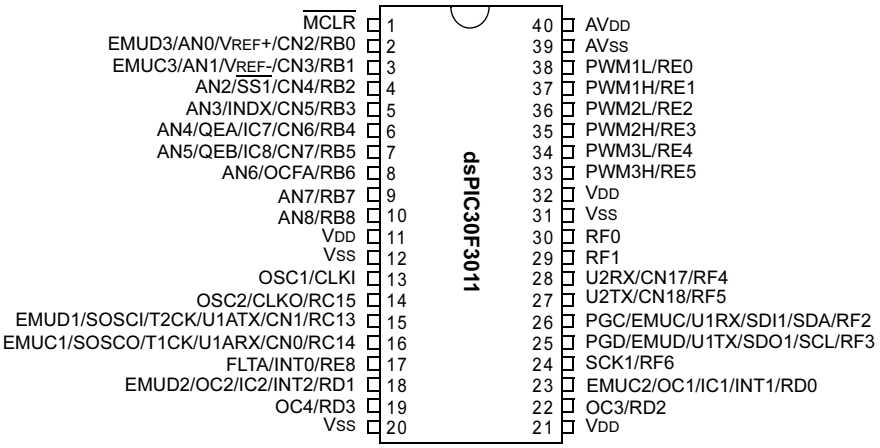
28-Pin QFN



 © 2004 Microchip Technology Inc.

FIGURE 2-6: PIN DIAGRAMS (40-PIN PDIP)

40-Pin PDIP



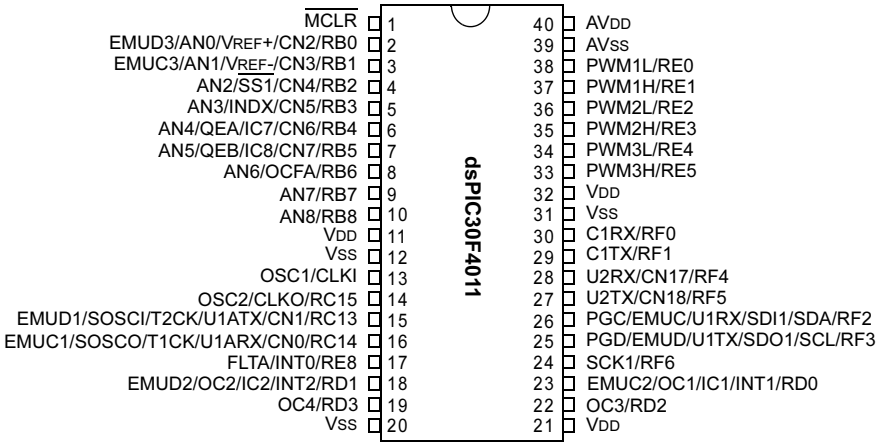
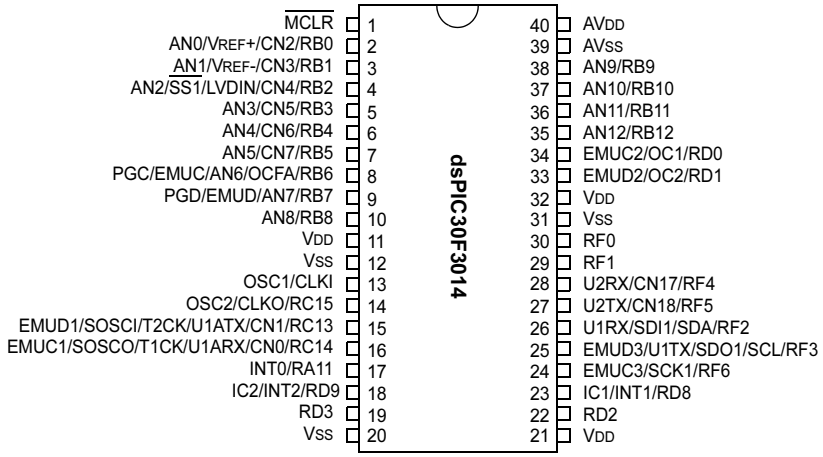
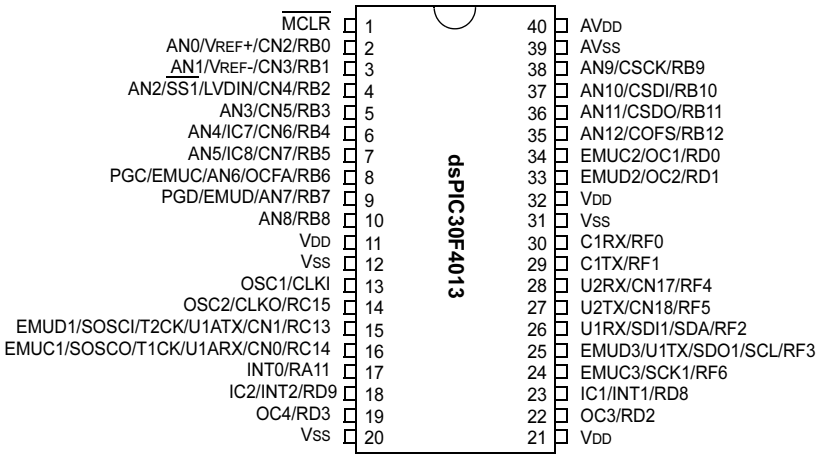


FIGURE 2-7: PIN DIAGRAMS (40-PIN PDIP)

40-Pin PDIP





 © 2004 Microchip Technology Inc.

FIGURE 2-8: PIN DIAGRAMS (44-PIN TQFP)

44-Pin TQFP

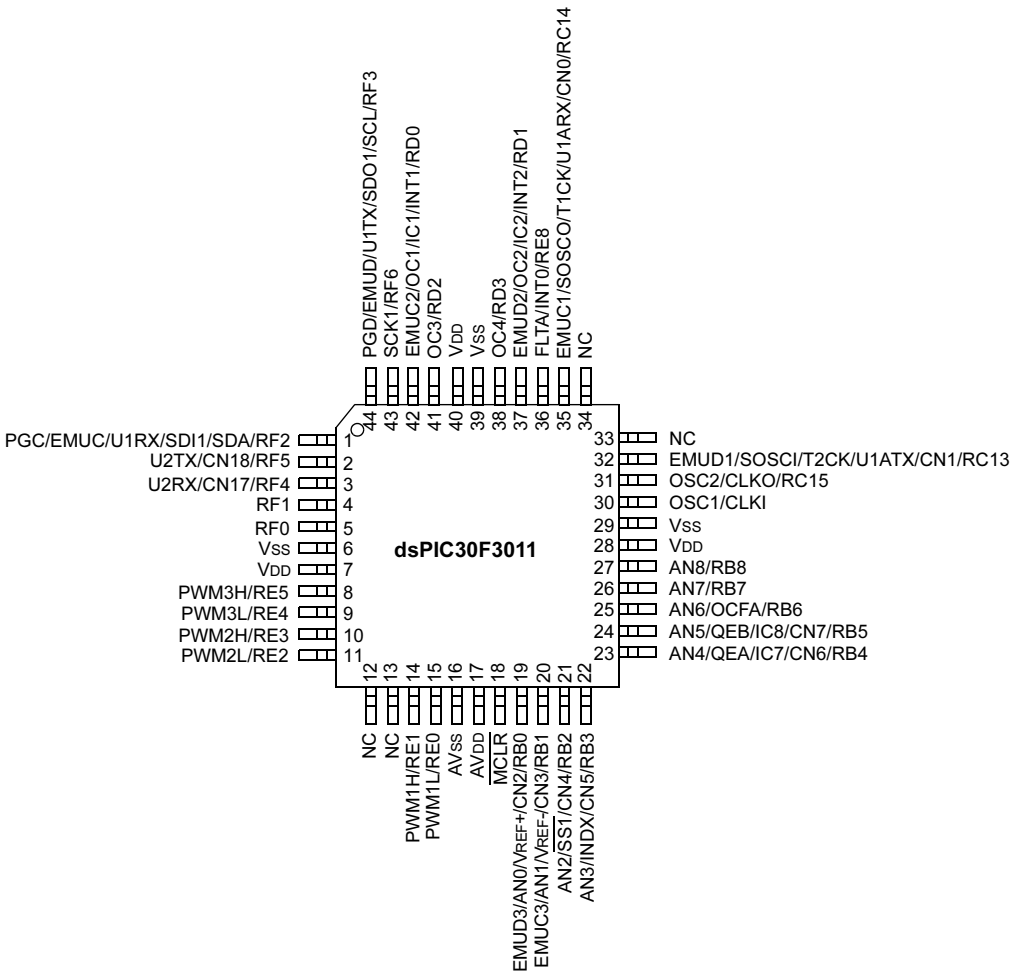
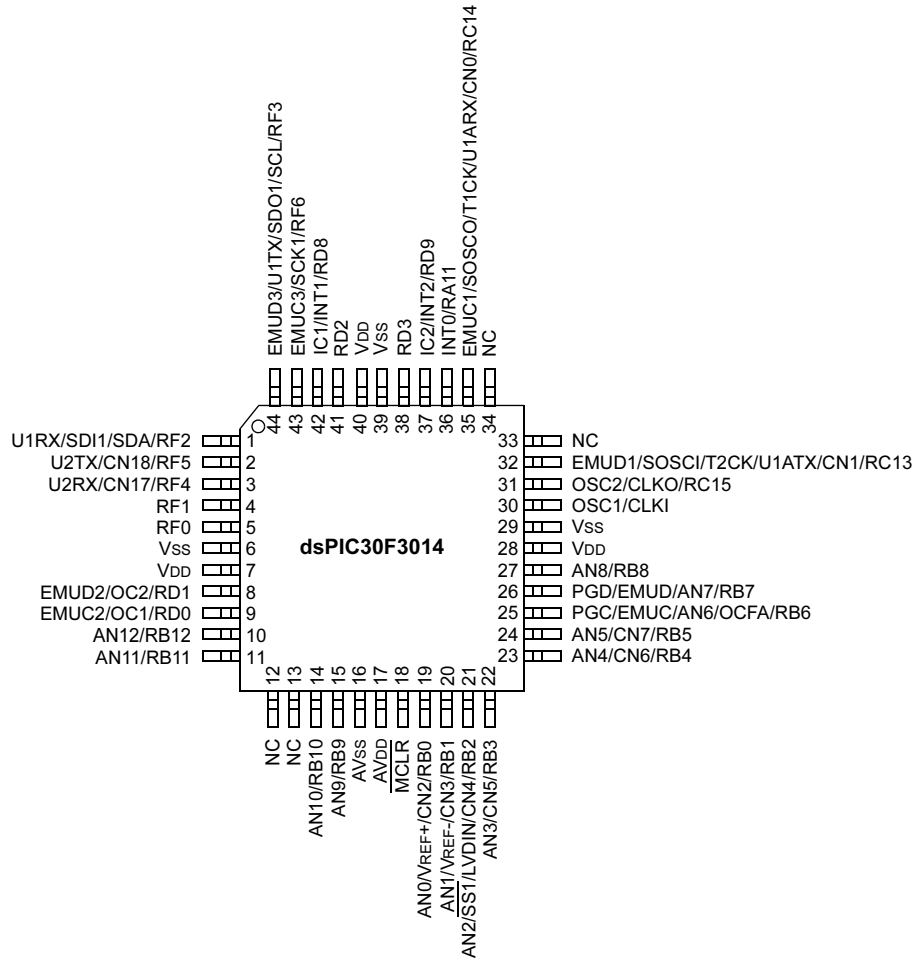


FIGURE 2-9: PIN DIAGRAMS (44-PIN TQFP)

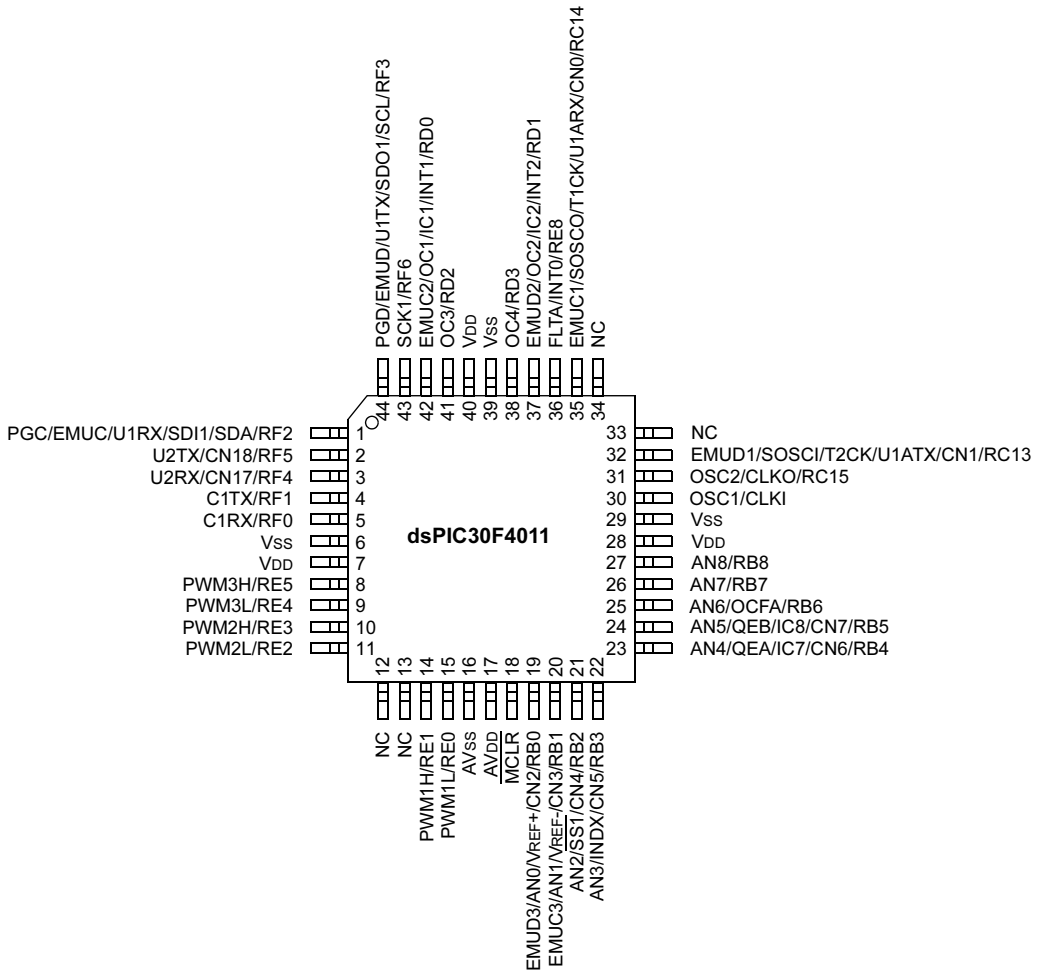
44-Pin TQFP



 © 2004 Microchip Technology Inc.

FIGURE 2-10: PIN DIAGRAMS (44-PIN TQFP)

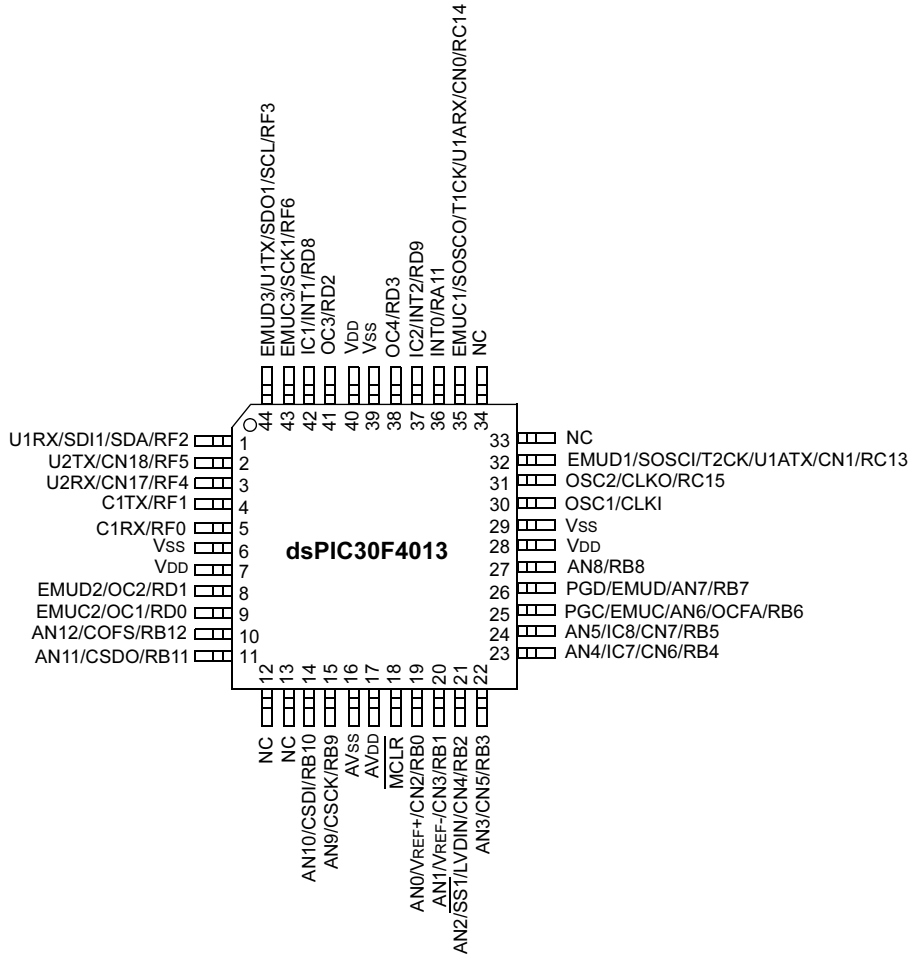
44-Pin TQFP



Note: Pinouts subject to change.

FIGURE 2-11: PIN DIAGRAMS (44-PIN TQFP)

44-Pin TQFP



 © 2004 Microchip Technology Inc.

FIGURE 2-12: PIN DIAGRAM (44-PIN QFN)

44-Pin QFN

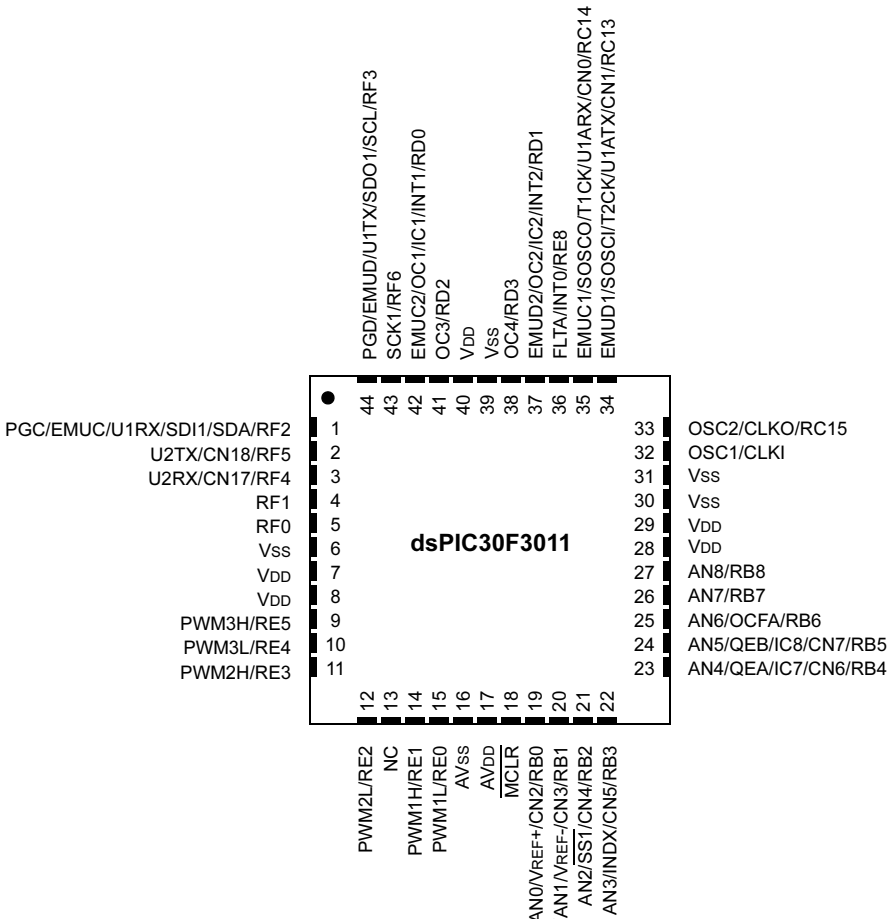
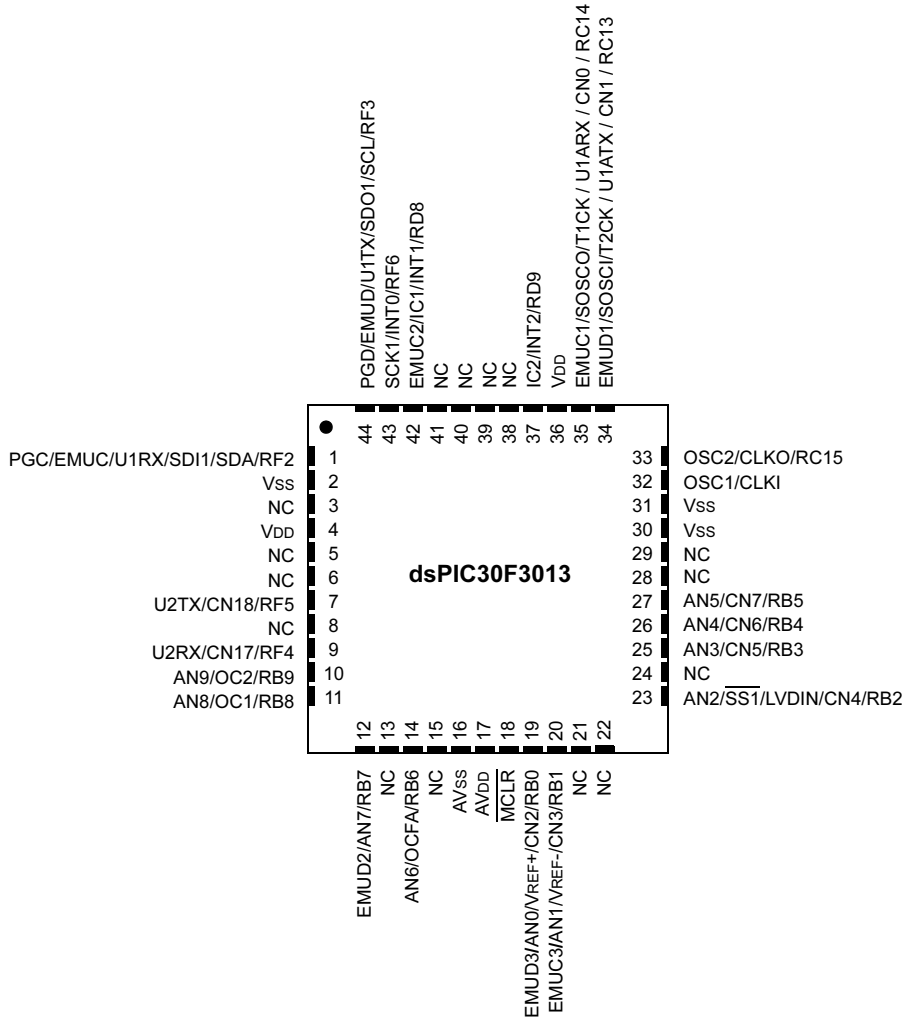


FIGURE 2-13: PIN DIAGRAM (44-PIN QFN)

44-Pin QFN



 © 2004 Microchip Technology Inc.

FIGURE 2-14: PIN DIAGRAM (44-PIN QFN)

44-Pin QFN

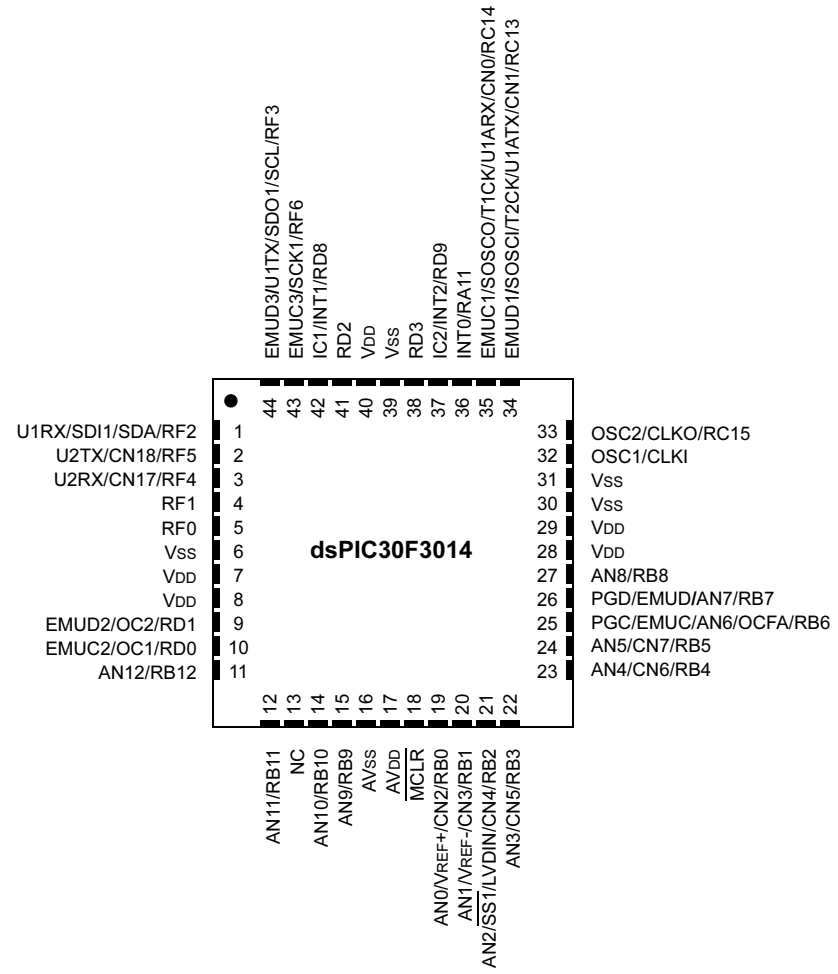
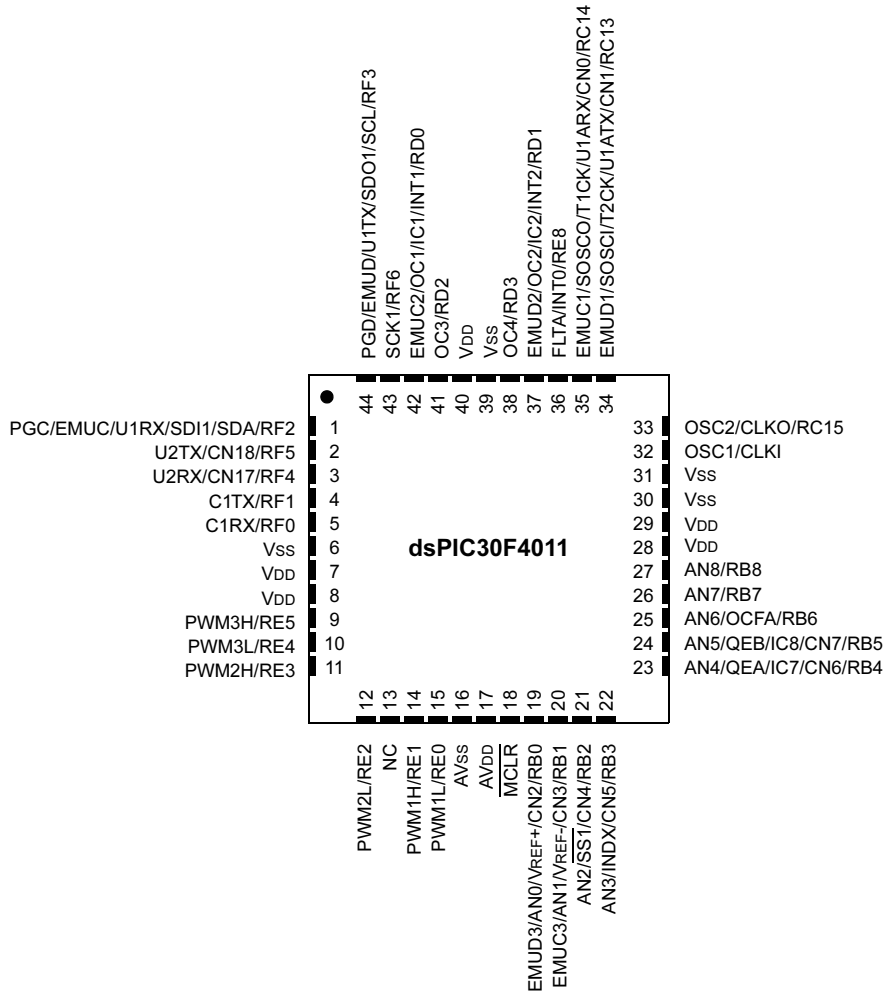


FIGURE 2-15: PIN DIAGRAM (44-PIN QFN)

44-Pin QFN



 © 2004 Microchip Technology Inc.

FIGURE 2-16: PIN DIAGRAM (44-PIN QFN)

44-Pin QFN

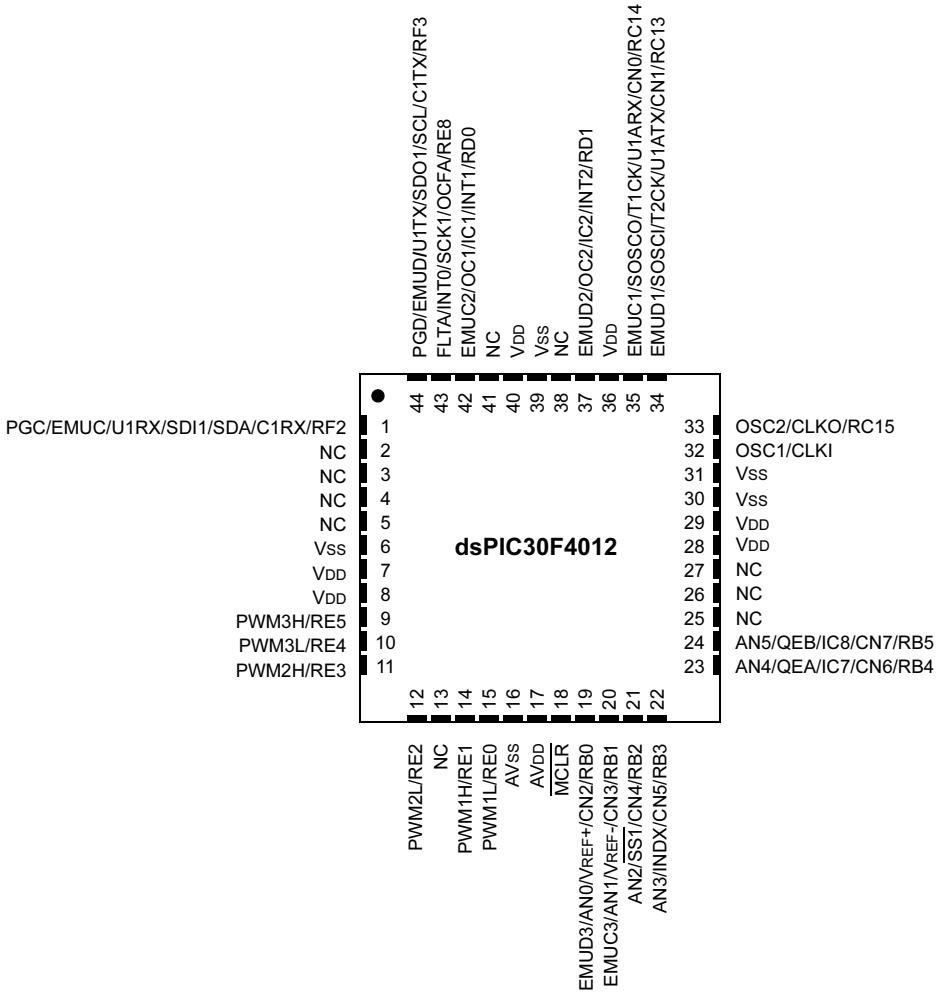
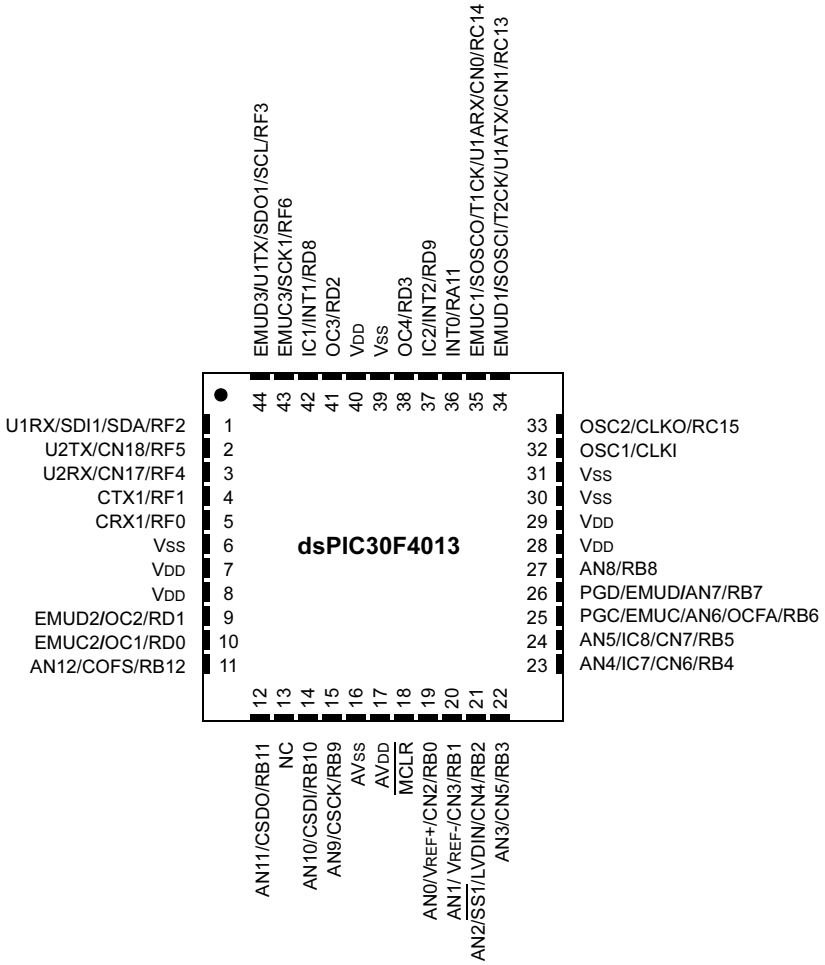


FIGURE 2-17: PIN DIAGRAM (44-PIN QFN)

44-Pin QFN



 © 2004 Microchip Technology Inc.

FIGURE 2-18: PIN DIAGRAMS (64-PIN TQFP)

64-Pin TQFP

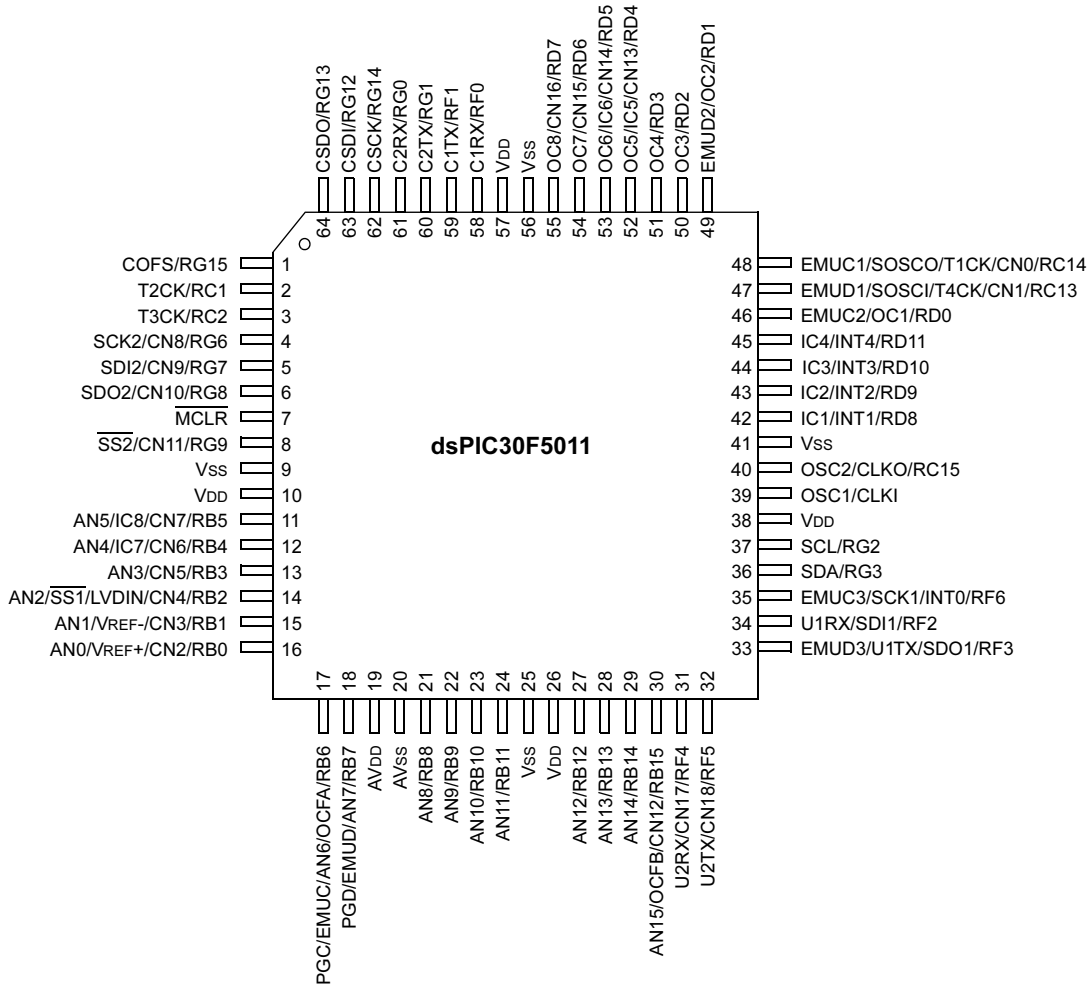
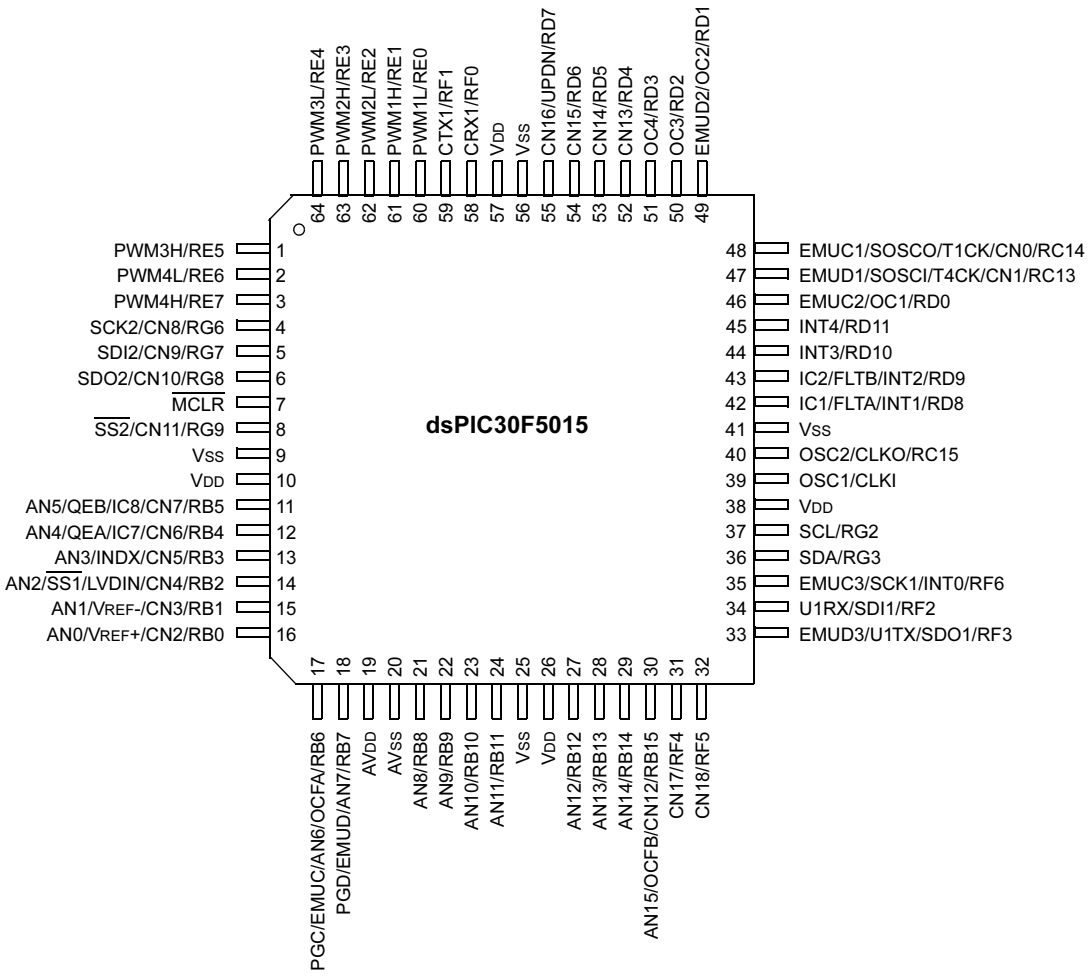


FIGURE 2-19: PIN DIAGRAMS (64-PIN TQFP)

64-Pin TQFP



 © 2004 Microchip Technology Inc.

FIGURE 2-20: PIN DIAGRAMS (64-PIN TQFP)

64-Pin TQFP

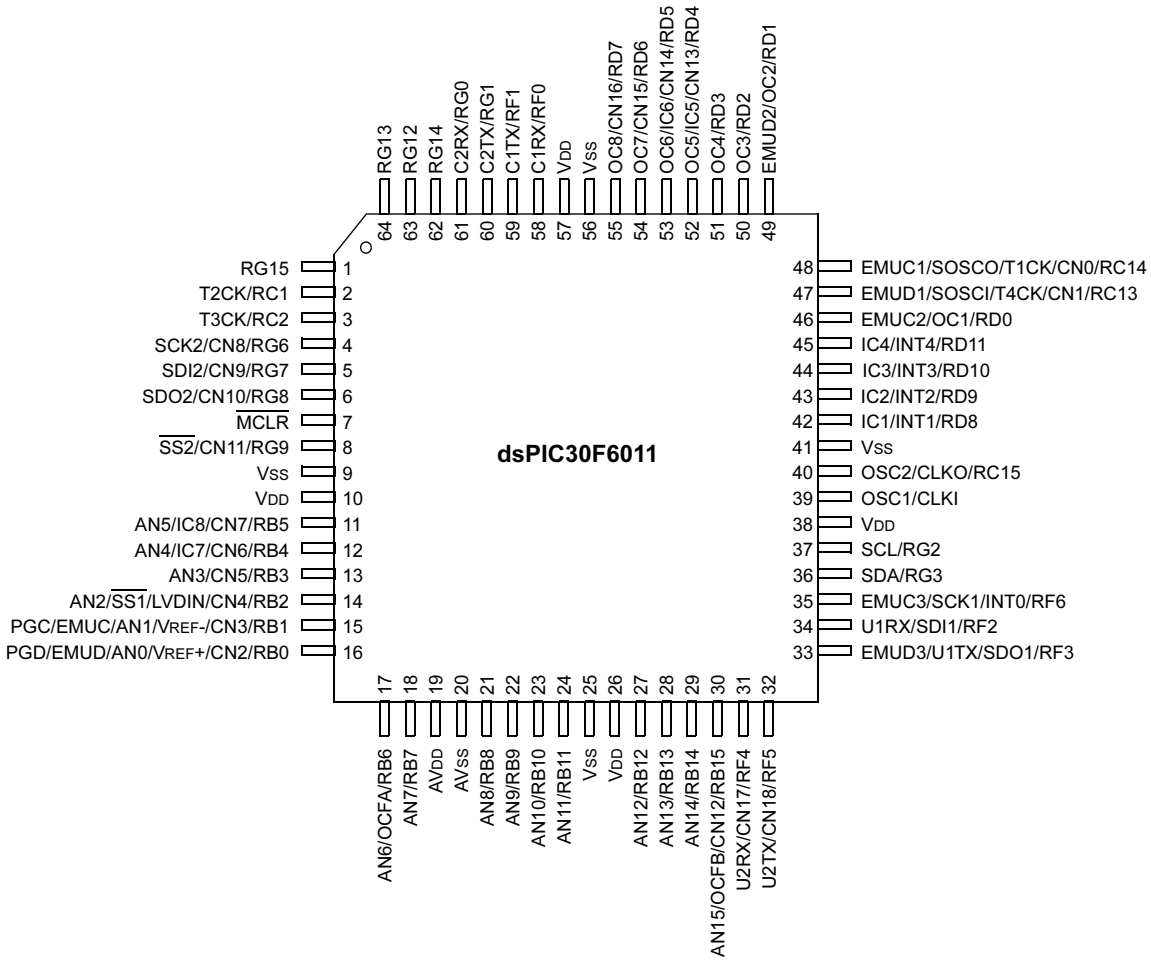
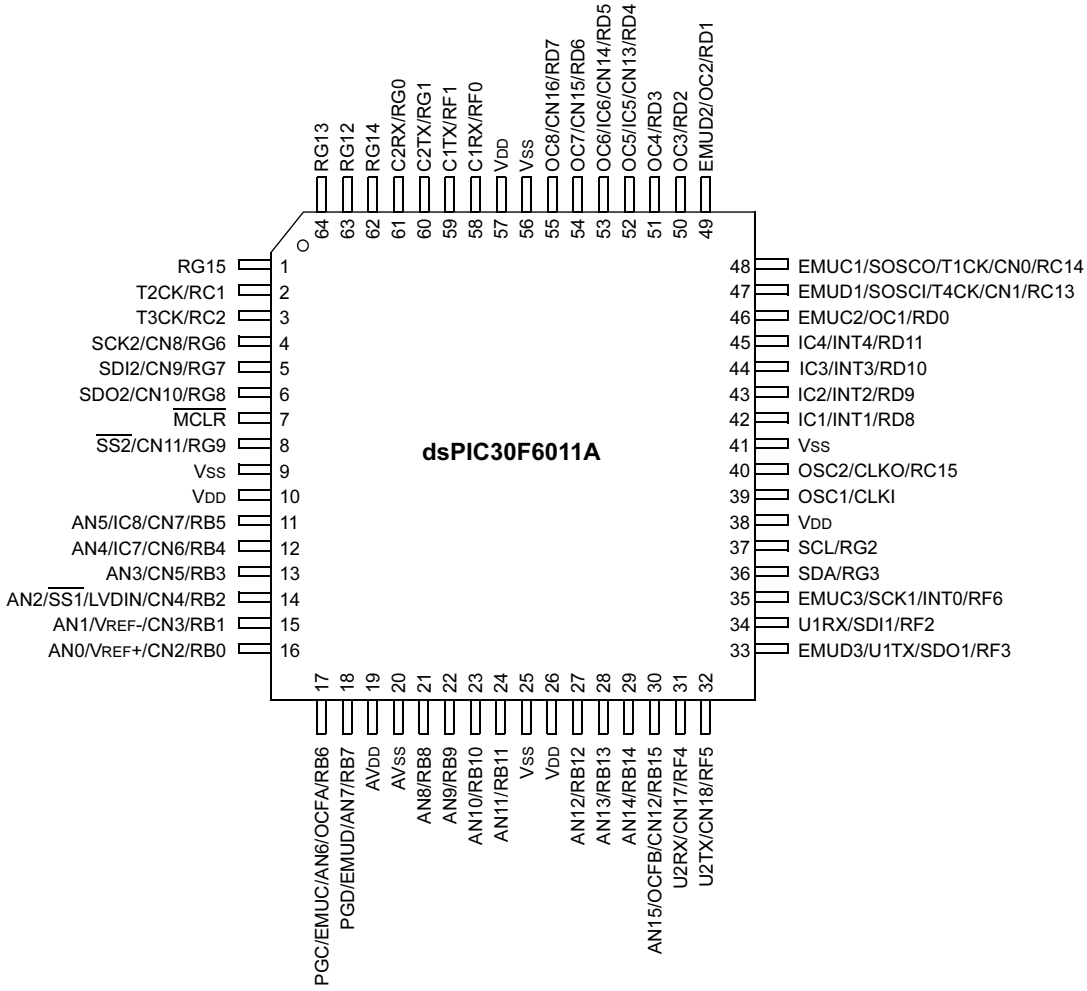


FIGURE 2-21: PIN DIAGRAMS (64-PIN TQFP)

64-Pin TQFP



Note: Pinouts subject to change.

 © 2004 Microchip Technology Inc.

FIGURE 2-22: PIN DIAGRAMS (64-PIN TQFP)

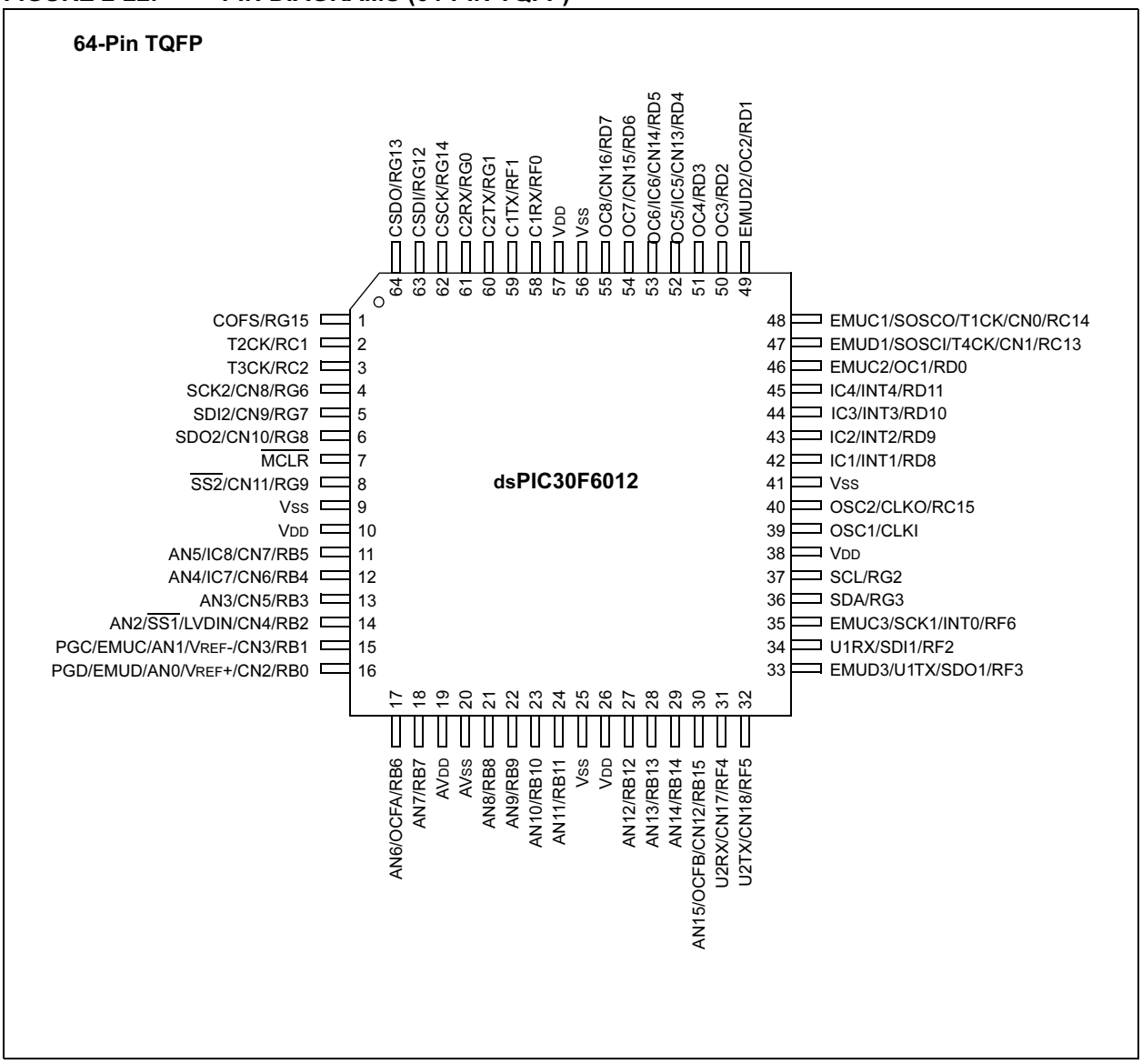
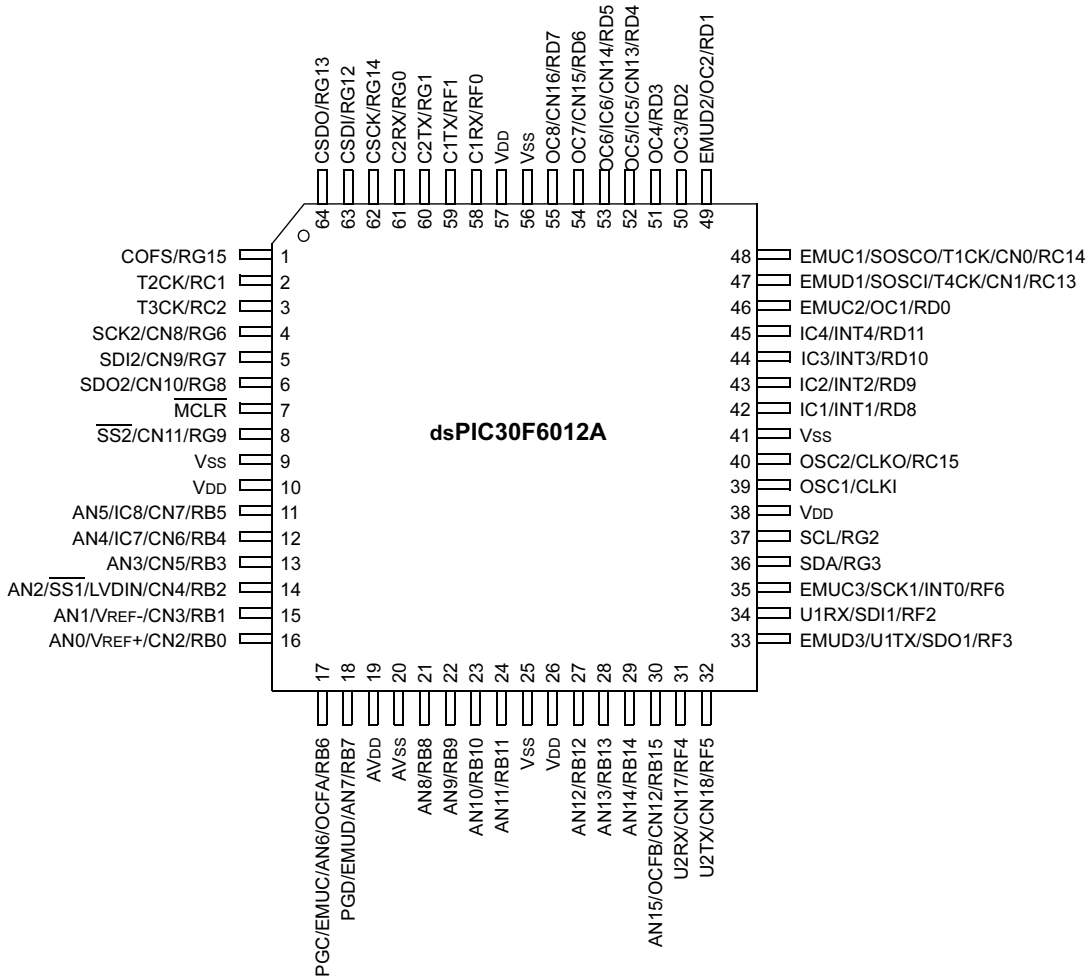


FIGURE 2-23: PIN DIAGRAMS (64-PIN TQFP)

64-Pin TQFP



Note: Pinouts subject to change.

 © 2004 Microchip Technology Inc.

FIGURE 2-24: PIN DIAGRAMS (80-PIN TQFP)

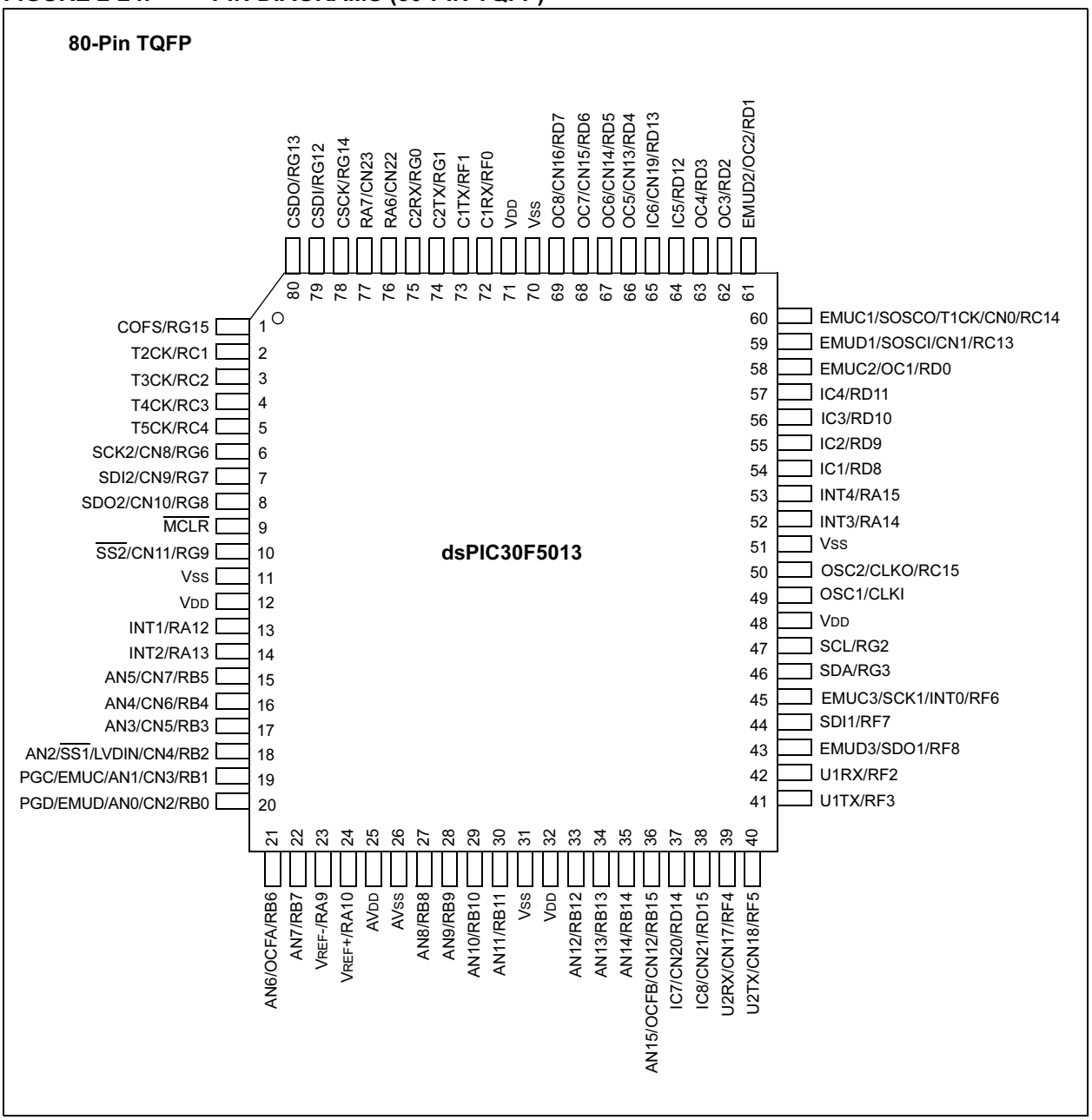
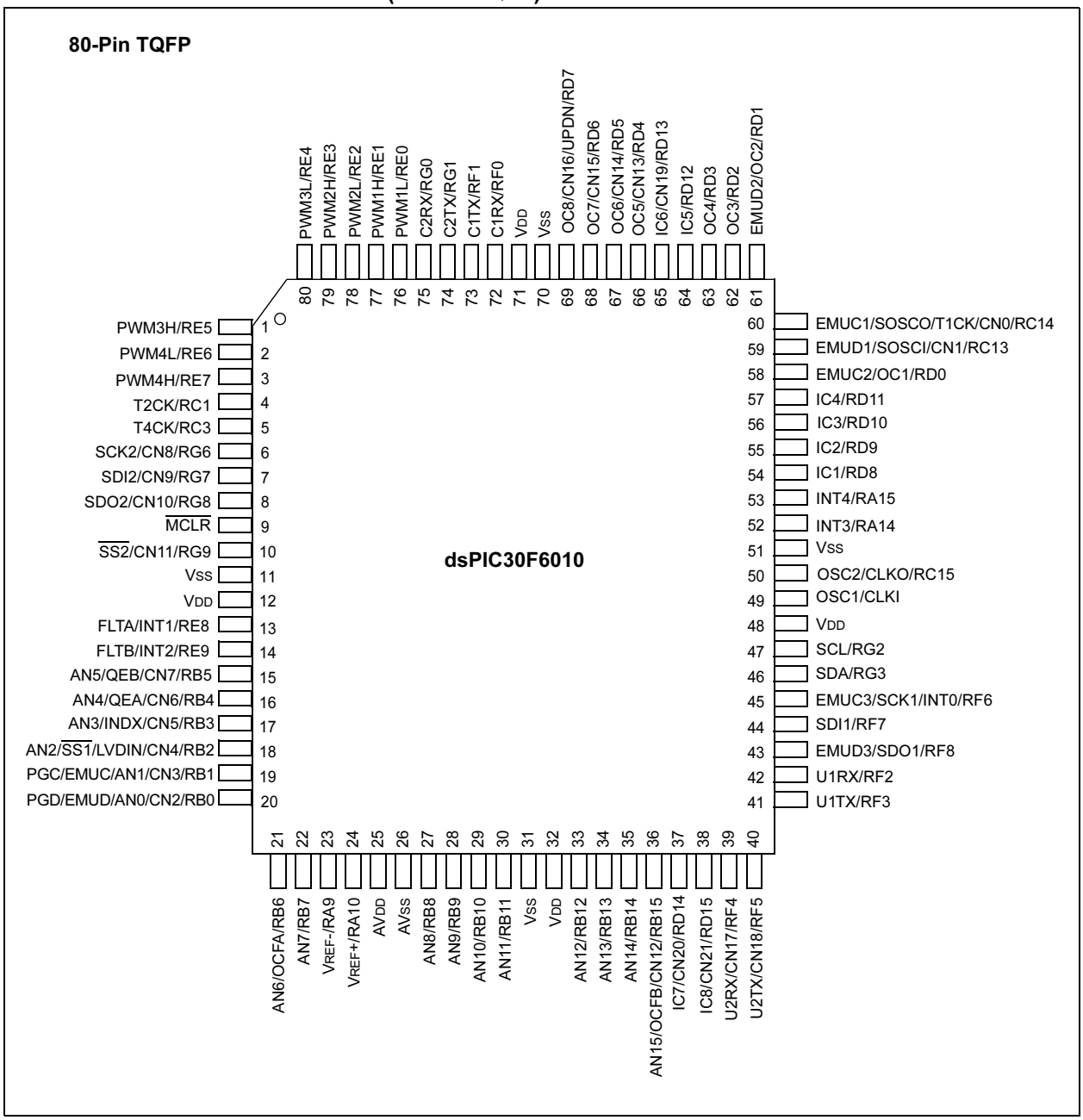


FIGURE 2-25: PIN DIAGRAMS (80-PIN TQFP)



 © 2004 Microchip Technology Inc.

FIGURE 2-26: PIN DIAGRAMS (80-PIN TQFP)

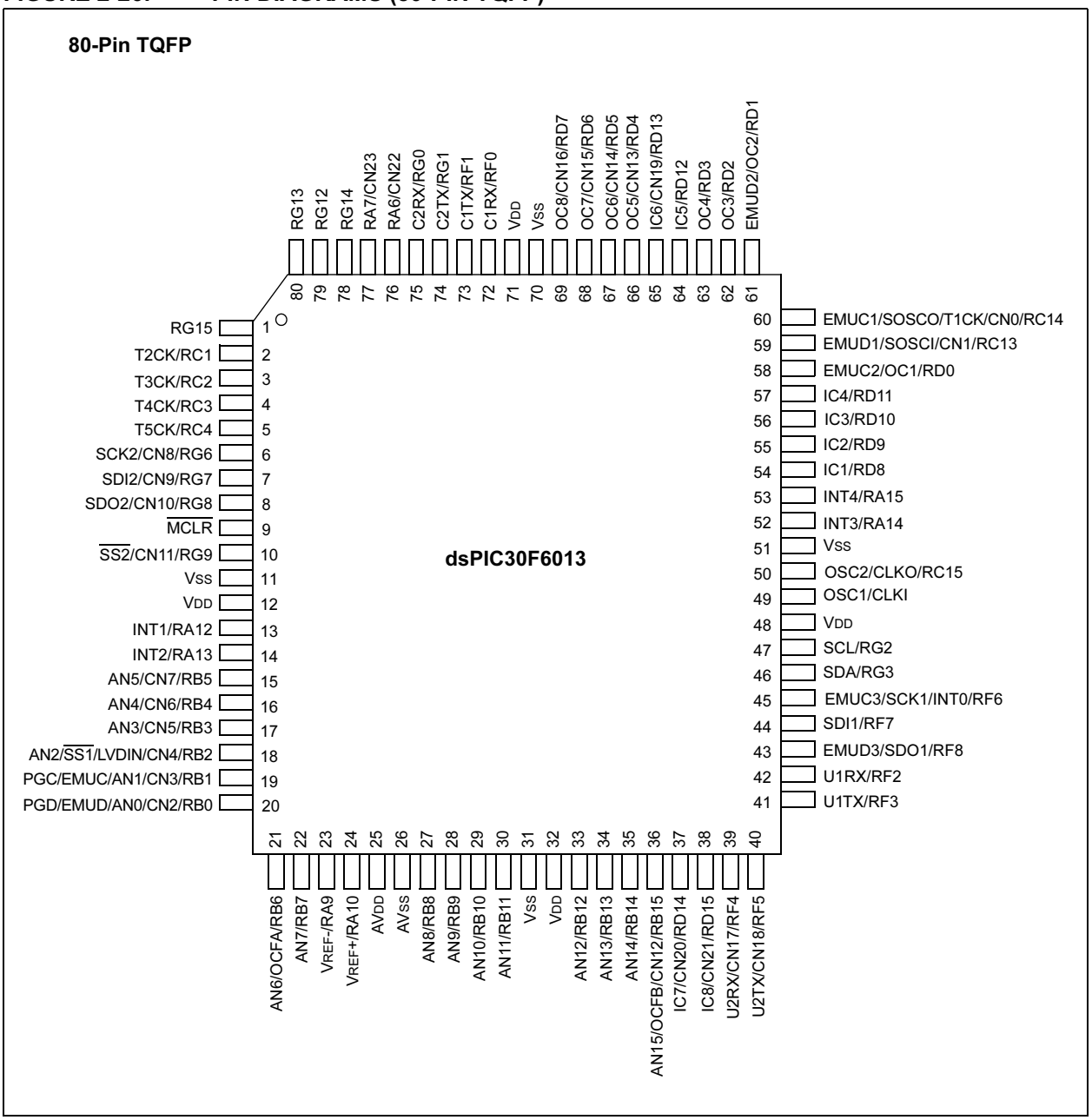
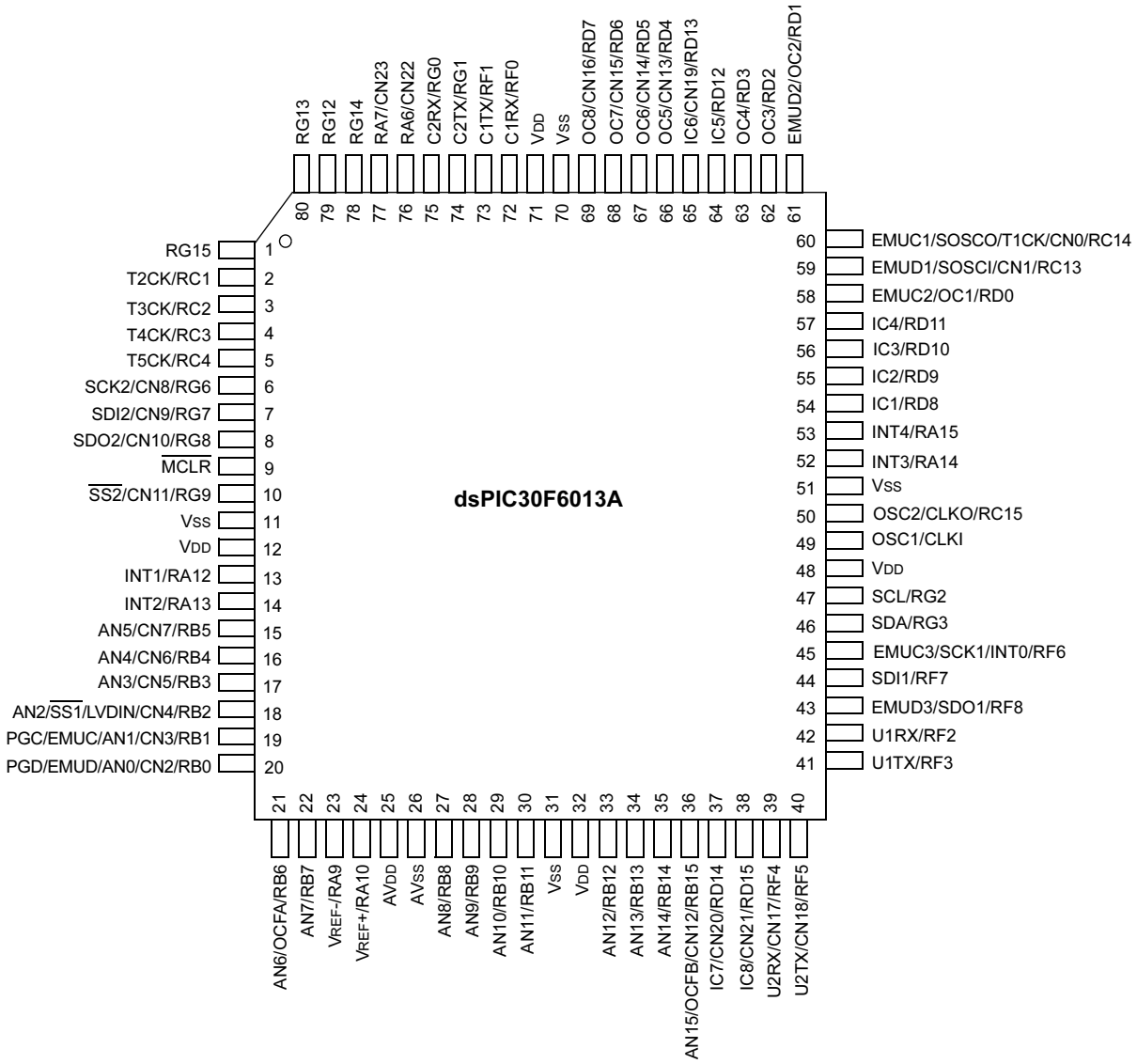


FIGURE 2-27: PIN DIAGRAMS (80-PIN TQFP)

80-Pin TQFP



Note: Pinouts subject to change.

 © 2004 Microchip Technology Inc.

FIGURE 2-28: PIN DIAGRAMS (80-PIN TQFP)

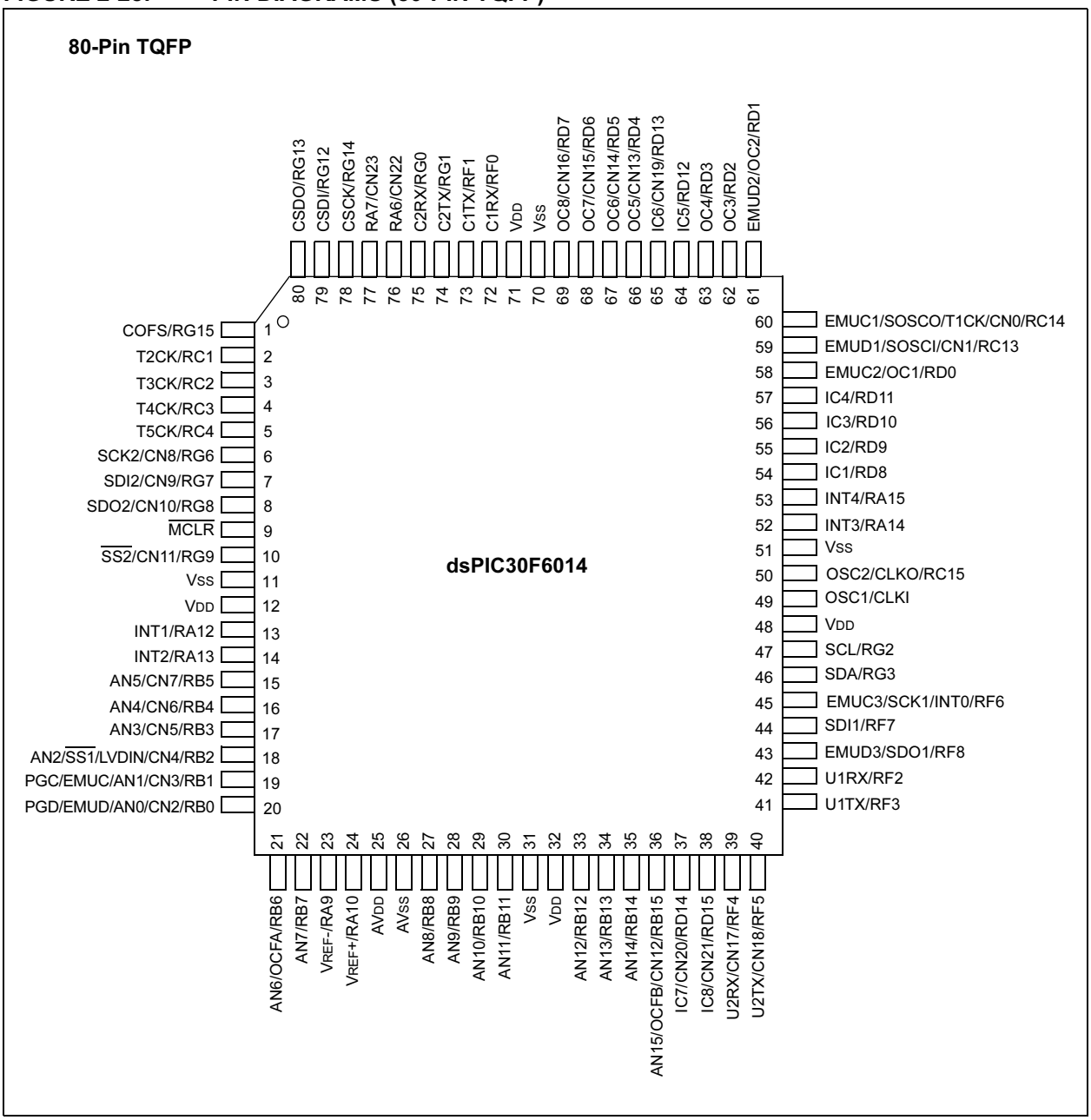
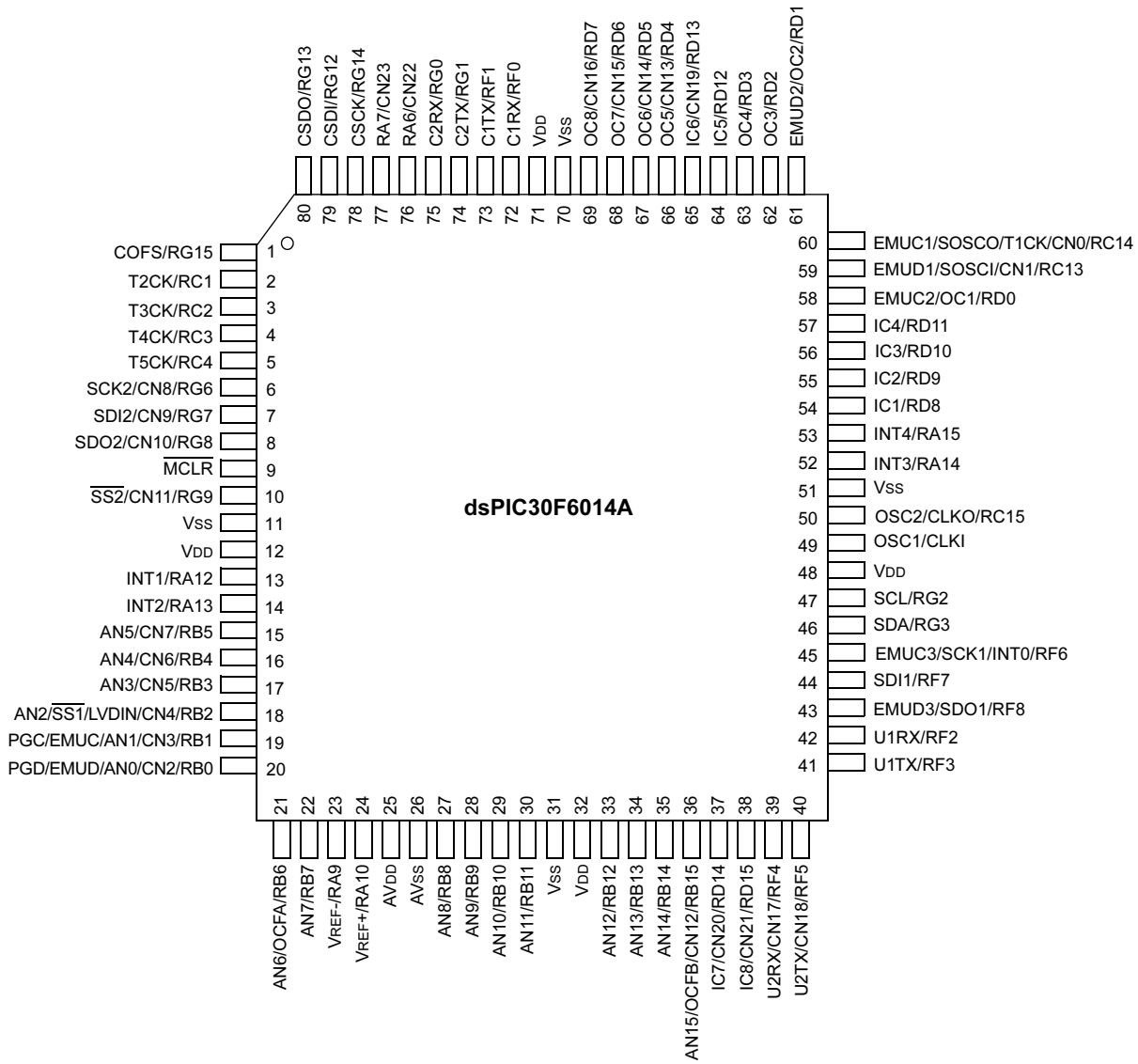


FIGURE 2-29: PIN DIAGRAMS (80-PIN TQFP)

80-Pin TQFP



Note: Pinouts subject to change.

 © 2004 Microchip Technology Inc.

2.3 Program Memory Map

The program memory space extends from 0x0 to 0xFFFFFE. Code storage is located at the base of the memory map and supports up to 144 Kbytes (48K instruction words). Code is stored in three, 48-Kbyte memory panels that reside on-chip. Table 2-2 shows the location and program memory size of each device variant.

Locations 0x800000 through 0x8005BE are reserved for executive code memory. This region either stores the programming executive or debugging executive. The programming executive is used for device programming, and the debug executive is used for in-circuit debugging. This region of memory can not be used to store user code.

Locations 0xF80000 through 0xF8000E are reserved for the Configuration registers. The bits in these registers may be set to select various device options, and are described in **Section 5.7 "Configuration Bits Programming"**. The configuration bits read out normally, even after code protection is applied.

Locations 0xFF0000 and 0xFF0002 are reserved for the Device ID registers. These bits can be used by the programmer to identify what device type is being programmed. They are described in **Section 10.0 "Device ID"**. The device ID reads out normally, even after code protection is applied.

Figure 2-30 shows the memory map for the dsPIC30F variants.

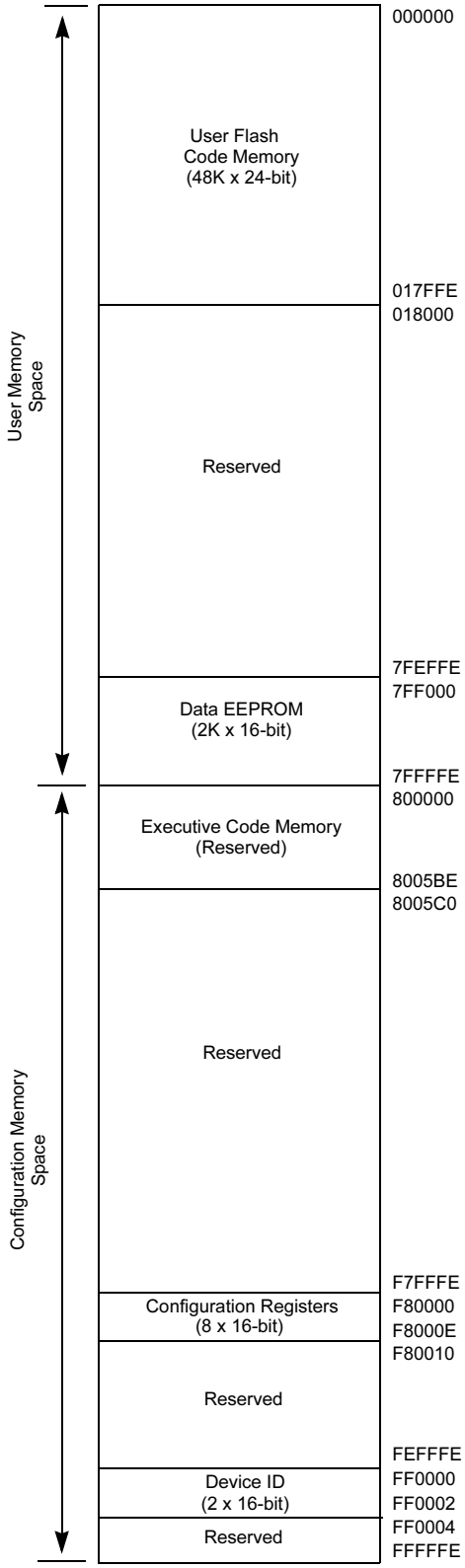
2.4 Data EEPROM Memory

The data EEPROM array supports up to 4 Kbytes of data and is located in one memory panel. It is mapped in program memory space, residing at the end of User Memory Space (see Figure 2-30). Table 2-2 shows the location and size of data EEPROM in each device variant.

TABLE 2-2: CODE MEMORY AND DATA EEPROM MAP AND SIZE

dsPIC30F Device	Code Memory Map (Size in Instruction Words)	Data EEPROM Memory Map (Size in Bytes)
2010	0x000000-0x001FFE (4K)	0x7FFC00-0x7FFFFE (1K)
2011	0x000000-0x001FFE (4K)	None (0K)
2012	0x000000-0x001FFE (4K)	None (0K)
3010	0x000000-0x003FFE (8K)	0x7FFC00-0x7FFFFE (1K)
3011	0x000000-0x003FFE (8K)	0x7FFC00-0x7FFFFE (1K)
3012	0x000000-0x003FFE (8K)	0x7FFC00-0x7FFFFE (1K)
3013	0x000000-0x003FFE (8K)	0x7FFC00-0x7FFFFE (1K)
3014	0x000000-0x003FFE (8K)	0x7FFC00-0x7FFFFE (1K)
4011	0x000000-0x007FFE (16K)	0x7FFC00-0x7FFFFE (1K)
4012	0x000000-0x007FFE (16K)	0x7FFC00-0x7FFFFE (1K)
4013	0x000000-0x007FFE (16K)	0x7FFC00-0x7FFFFE (1K)
5011	0x000000-0x00AFFE (22K)	0x7FFC00-0x7FFFFE (1K)
5013	0x000000-0x00AFFE (22K)	0x7FFC00-0x7FFFFE (1K)
5015	0x000000-0x00AFFE (22K)	0x7FFC00-0x7FFFFE (1K)
6010	0x000000-0x017FFE (48K)	0x7FF000-0x7FFFFE (4K)
6011	0x000000-0x015FFE (44K)	0x7FF800-0x7FFFFE (2K)
6011A	0x000000-0x015FFE (44K)	0x7FF800-0x7FFFFE (2K)
6012	0x000000-0x017FFE (48K)	0x7FF000-0x7FFFFE (4K)
6012A	0x000000-0x017FFE (48K)	0x7FF000-0x7FFFFE (4K)
6013	0x000000-0x015FFE (44K)	0x7FF800-0x7FFFFE (2K)
6013A	0x000000-0x015FFE (44K)	0x7FF800-0x7FFFFE (2K)
6014	0x000000-0x017FFE (48K)	0x7FF000-0x7FFFFE (4K)
6014A	0x000000-0x017FFE (48K)	0x7FF000-0x7FFFFE (4K)

FIGURE 2-30: PROGRAM MEMORY MAP



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

 © 2004 Microchip Technology Inc.

3.0 PROGRAMMING EXECUTIVE APPLICATION

3.1 Programming Executive Overview

The programming executive resides in executive memory and is executed when 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. After the programming executive has run, no assumptions should be made about the contents of data RAM.

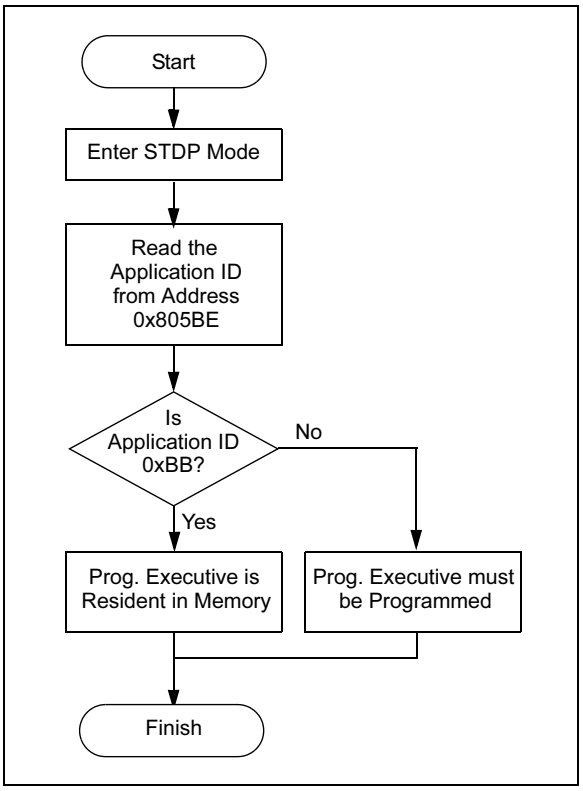
4.0 CONFIRMING THE CONTENTS OF EXECUTIVE MEMORY

Before programming can begin, the programmer must confirm that the programming executive is stored in executive memory. The procedure for this task is shown in Figure 4-1.

First, Standard DUT Programming mode (STDP) is entered. Then, the unique application ID word stored in executive memory is 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 "STDP Mode" describes the process for the STDP programming method. **Section 11.14 "Reading the Application ID Word"** describes the procedure for reading the application ID word in STDP mode.

FIGURE 4-1: CONFIRMING PRESENCE OF PROGRAMMING EXECUTIVE



 © 2004 Microchip Technology Inc.

5.0 DEVICE PROGRAMMING

5.1 Overview of the Programming Process

After 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"**.

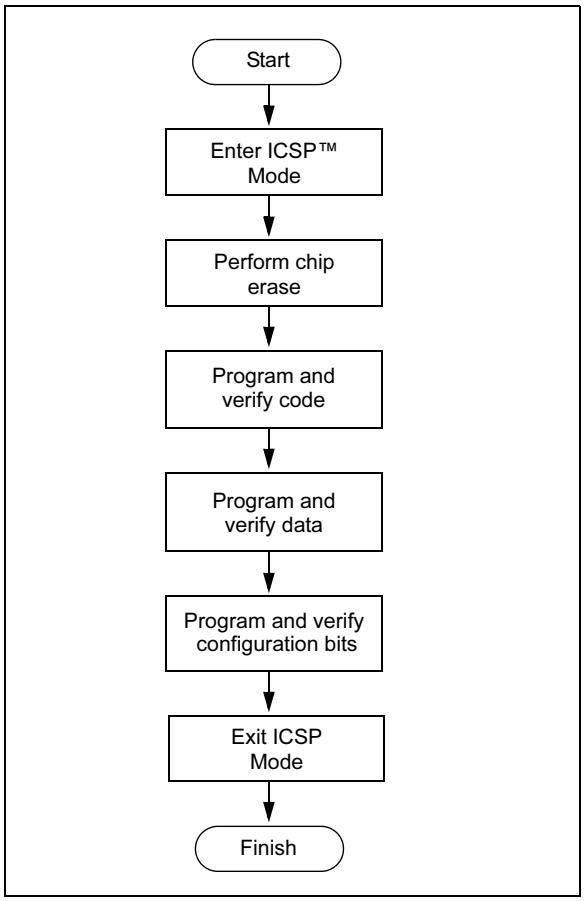
TABLE 5-1: COMMAND SET SUMMARY

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
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 shown in Figure 5-1. The process begins by entering 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 ICSP mode is exited. If any of the verifications failed, the procedure should be repeated, starting from the chip erase.

Section 5.2 "Entering ICSP Mode" through **Section 5.8 "Exiting ICSP Mode"** describe the programming process in detail.

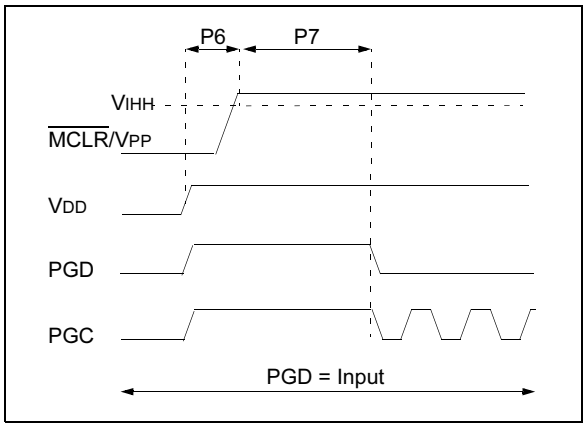
FIGURE 5-1: PROGRAMMING FLOW



5.2 Entering ICSP Mode

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

FIGURE 5-2: ENTERING ICSP MODE



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

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 are not erasable, 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"**).

Note:	The device ID registers can not 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 a '1'. The following memories must be blank checked:

- All implemented code memory
- All implemented data EEPROM
- All configuration bits

The Device ID registers (0xFF0000:0xFF0002) can be ignored by the blank check since this region stores device information that can not 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, data EEPROM and configuration bits are erased by testing these memory regions. A 'BLANK' or 'NOT BLANK' response is returned. If it is determined that the device is not blank, it must be erased (see **Section 5.3 "Chip Erase"**) before attempting to program the chip.

5.5 Code Memory Programming

5.5.1 OVERVIEW

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

TABLE 5-2: DEVICE CODE MEMORY SIZE

dsPIC30F Device	Code Size (24-bit Words)	Number of Rows	Number of Panels
2010	4K	128	1
2011	4K	128	1
2012	4K	128	1
3010	8K	256	1
3011	8K	256	1
3012	8K	256	1
3013	8K	256	1
3014	8K	256	1
4011	16K	512	1
4012	16K	512	1
4013	16K	512	1
5011	22K	704	2
5013	22K	704	2
5015	22K	704	2
6010	48K	1536	3
6011	44K	1408	3
6011A	44K	1408	3
6012	48K	1536	3
6012A	48K	1536	3
6013	44K	1408	3
6013A	44K	1408	3
6014	48K	1536	3
6014A	48K	1536	3

 © 2004 Microchip Technology Inc.

5.5.2 PROGRAMMING METHODOLOGY

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

A flowchart for programming of code memory is shown in Figure 5-3. In this example, all 48K instruction words of a dsPIC30F6014 device are programmed. First, the number of commands to send (called 'RemainingCmds' in the flowchart) is set to 1536 and the destination address (called 'BaseAddress') is set to '0'

Next, one row in the device is programmed with a PROGP command. Each PROGP command contains data for one row of code memory of the dsPIC30F6014. After the first command is processed successfully, 'RemainingCmds' is decremented by 1 and compared to 0. Since there are more PROGP commands to send, 'BaseAddress' is incremented by 0x40 to point to the next row of memory.

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

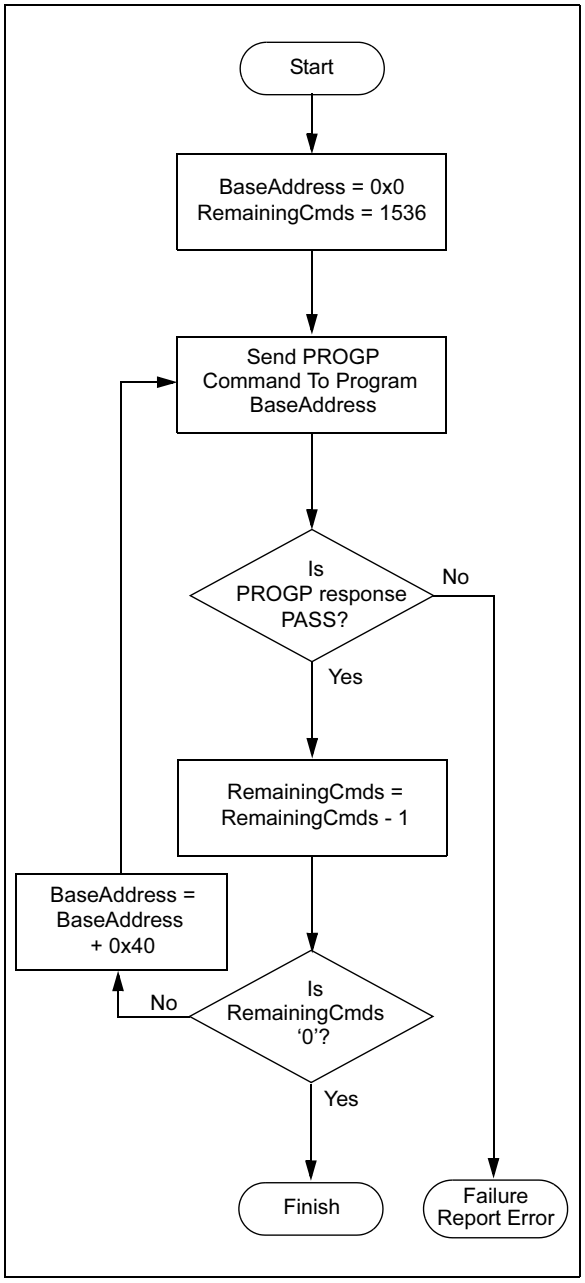
5.5.3 PROGRAMMING VERIFICATION

After code memory is programmed, the contents of memory can be verified to ensure that programming was successful. Verification requires code memory to be read back and compared against the copy held in the programmer's buffer.

The READP command can be used to read back all the programmed code memory.

Alternatively, you can have the programmer perform the verification after the entire device is programmed using a checksum computation as described in **Section 6.7 "Checksum Computation"**.

FIGURE 5-3: FLOWCHART FOR PROGRAMMING dsPIC30F6014 CODE MEMORY



5.6 Data EEPROM Programming

5.6.1 OVERVIEW

The panel architecture for the data EEPROM memory array consists of 128 rows of sixteen 16-bit data words. Each panel stores 2K words. All variants have either one or no memory panel. Devices with data EEPROM provide either 512 words, 1024 words or 2048 words of memory on the one panel (see Table 5-3).

TABLE 5-3: DATA EEPROM SIZE

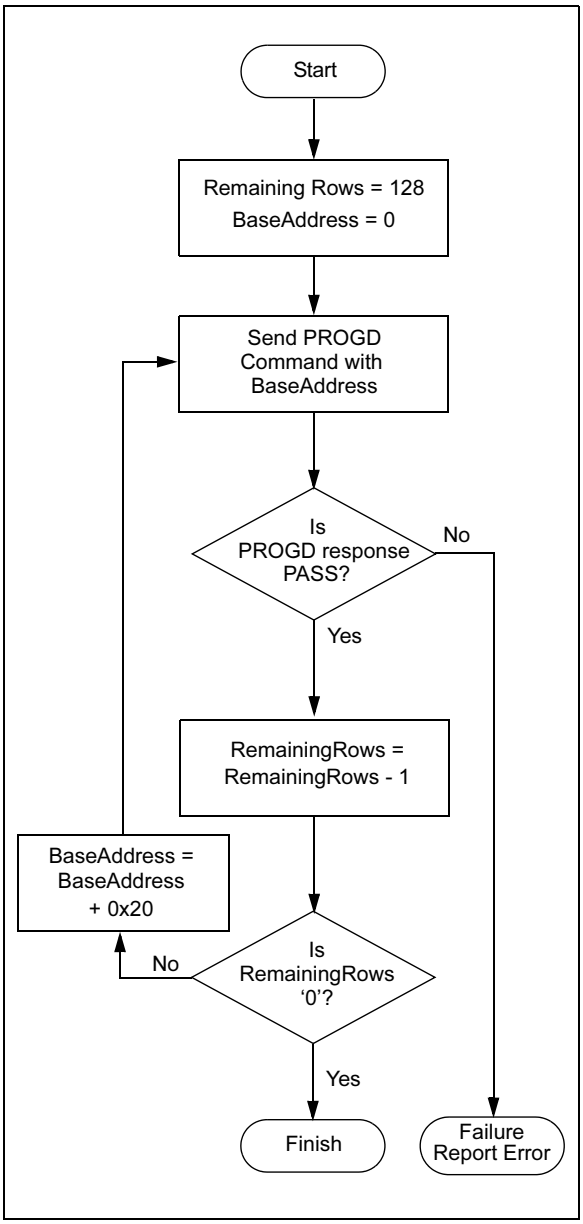
dsPIC30F Device	Data EEPROM Size (Words)	Number of Rows(1)
2010	512	32
2011	0	0
2012	0	0
3010	512	32
3011	512	32
3012	512	32
3013	512	32
3014	512	32
4011	512	32
4012	512	32
4013	512	32
5011	512	32
5013	512	32
5015	512	32
6010	2048	128
6011	1024	64
6011A	1024	64
6012	2048	128
6012A	2048	128
6013	1024	64
6013A	1024	64
6014	2048	128
6014A	2048	128

5.6.2 PROGRAMMING METHODOLOGY

The programming executive uses the PROGD command to program the data EEPROM. Figure 5-4 flowcharts the process. First, the number of rows to program (RemainingRows) is based on the device size, and the destination address (DestAddress) is set to 0. In this example, 128 rows (2048 words) of data EEPROM will be programmed.

The first PROGD command programs the first row of data EEPROM. After the command completes successfully, 'RemainingRows' is decremented by one and compared with zero. Since there are 127 more rows to program, 'BaseAddress' is incremented by 0x20 to point to the next row of data EEPROM. This process is then repeated until all 128 rows of data EEPROM are programmed.

FIGURE 5-4: FLOWCHART FOR PROGRAMMING dsPIC30F6014 DATA EEPROM



5.6.3 PROGRAMMING VERIFICATION

After the data EEPROM is programmed, the contents of memory can be verified to ensure that the programming was successful. Verification requires the data EEPROM to be read back and compared against the copy held in the programmer's buffer. The READD command reads back the programmed data EEPROM.

Alternatively, the programmer can perform the verification after the entire device is programmed using a checksum computation, as described in **Section 6.7 "Checksum Computation"**.

 © 2004 Microchip Technology Inc.

5.7 Configuration Bits Programming

5.7.1 OVERVIEW

The dsPIC30F has configuration bits stored in seven 16-bit registers. These bits can be set or cleared to select various device configurations. There are two types of configuration bits: system-operation bits and code-protect bits. The system-operation bits determine the power-on settings for system level components such as the oscillator and Watchdog Timer. The code-protect bits prevent program memory from being read and written.

The FOSC configuration register has three different register descriptions, based on device. The FOSC configuration register description for the dsPIC30F2010, dsPIC30F6010/6011/6012/6013/6014 devices is shown in Table 5-4.

The FOSC configuration register description for the dsPIC30F4011/4012 and dsPIC30F5011/5013 devices is shown in Table 5-5.

The FOSC configuration register description for all remaining devices (dsPIC30F2011/2012, dsPIC30F3010/ 3011/3012/3013, dsPIC30F3014/ 4013, dsPIC30F5015, dsPIC30F6011A/6012A/6013A/ 6014A) is shown in Table 5-6. Always use the correct register descriptions for your target processor.

The FWDT, FBORPOR, RESERVED1, RESERVED2, FGS and FICD configuration registers are not device dependent. The register descriptions for these configuration registers is shown in Table 5-7.

There are two configuration register maps, based on device. These are shown in Table 5-8 and Table 5-9.

TABLE 5-4: FOSC CONFIGURATION BITS DESCRIPTION FOR dsPIC30F2010, dsPIC30F6010/6011/6012/6013/6014

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 = Reserved (do not use) 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) 001x = HS – HS crystal oscillator mode. (10 MHz-25 MHz crystal) 000x = XTL – XTL crystal oscillator mode. (200 kHz-4 MHz crystal)

TABLE 5-5: FOSC CONFIGURATION BITS DESCRIPTION FOR dsPIC30F4011/4012 AND dsPIC30F501/5013

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)

 © 2004 Microchip Technology Inc.

TABLE 5-6: FOSC CONFIGURATION BITS DESCRIPTION FOR dsPIC30F2011/2012, dsPIC30F3010/3011/3012/3013/014, dsPIC30F4013, dsPIC30F5015, dsPIC30F6011A/6012A/6013A /6014A

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<2:0>	FOSC	Oscillator Source Selection on POR 111 =Primary Oscillator 110 =Reserved 10x =Reserved 011 =Alternate Oscillator 010 =Internal Low-Power RC Oscillator 001 =Internal Fast RC Oscillator (no PLL) 000 =Low-Power 32 KHz Oscillator (Timer1 Oscillator)
FPR<4:0>	FOSC	**Primary Oscillator Mode (when FOS<2:0> = 111b)** 11xxx =Reserved (do not use) 10111 =HS/3 w/ PLL 16X – HS/3 crystal oscillator with 16X PLL. (10 MHz - 25MHz crystal) 10110 =HS/3 w/ PLL 8X – HS/3 crystal oscillator with 8X PLL. (10 MHz - 25MHz crystal) 10101 =HS/3 w/ PLL 4X – HS/3 crystal oscillator with 4X PLL. (10 MHz - 25MHz crystal) 10100 =Reserved (do not use) 10011 =HS/2 w/ PLL 16X – HS/2 crystal oscillator with 16X PLL. (10 MHz - 25MHz crystal) 10010 =HS/2 w/ PLL 8X – HS/2 crystal oscillator with 8X PLL. (10 MHz - 25MHz crystal) 10001 =HS/2 w/ PLL 4X – HS/2 crystal oscillator with 4X PLL. (10 MHz - 25MHz crystal) 10000 =Reserved (do not use) 01111 =ECIO w/ PLL 16x – External clock with 16x PLL. OSC2 pin is I/O. 01110 =ECIO w/ PLL 8x – External clock with 8x PLL. OSC2 pin is I/O. 01101 =ECIO w/ PLL 4x – External clock with 4x PLL. OSC2 pin is I/O. 01100 =Reserved (do not use) 01011 = Reserved (do not use) 01010 =FRC w/ PLL 8x – Internal fast RC oscillator with 8x PLL. OSC2 pin is I/O. 01001 =Reserved (do not use) 01000 = Reserved (do not use) 00111 =XT w/ PLL 16X – XT crystal oscillator with 16X PLL. 00110 =XT w/ PLL 8X – XT crystal oscillator with 8X PLL. 00101 =XT w/ PLL 4X – XT crystal oscillator with 4X PLL. 00100 =Reserved (do not use) 00011 =FRC w/ PLL 16x – Internal fast RC oscillator with 8x PLL. OSC2 pin is I/O. 00010 =Reserved (do not use) 00001 =FRC w/ PLL 4x – Internal fast RC oscillator with 4x PLL. OSC2 pin is I/O. 00000 =Reserved (do not use)

TABLE 5-6: FOSC CONFIGURATION BITS DESCRIPTION FOR dsPIC30F2011/2012, dsPIC30F3010/3011/3012/3013/014, dsPIC30F4013, dsPIC30F5015, dsPIC30F6011A/6012A/6013A /6014A (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

Bit Field	Register	Description
FWPSA<1:0>	FWDT	Watchdog Timer Prescaler A 11 = 1:512 10 = 1:64 01 = 1:8 00 = 1:1
FWPSB<3:0>	FWDT	Watchdog Timer Prescaler B 1111 = 1:16 1110 = 1:15 • • • 0001 = 1:2 0000 = 1:1
FWDTEN	FWDT	Watchdog Enable 1 = Watchdog enabled (LPRC oscillator cannot be disabled. Clearing the SWDTEN bit in the RCON register will have no effect) 0 = Watchdog disabled (LPRC oscillator can be disabled by clearing the SWDTEN bit in the RCON register)
MCLREN	FBORPOR	Master Clear Enable 1 = Master Clear pin ($\overline{MCLR}$) is enabled. 0 = $\overline{MCLR}$ pin is disabled.
PWMPIN	FBORPOR	Motor Control PWM Module Pin Mode 1 = PWM module pins controlled by PORT register at device Reset (tristated) 0 = PWM module pins controlled by PWM module at device Reset (configured as output pins)
HPOL	FBORPOR	Motor Control PWM Module High Side Polarity 1 = PWM module high-side output pins have active-high output polarity. 0 = PWM module high-side output pins have active-low output polarity.
LPOL	FBORPOR	Motor Control PWM Module Low Side Polarity 1 = PWM module low-side output pins have active-high output polarity. 0 = PWM module low-side output pins have active-low output polarity.

 © 2004 Microchip Technology Inc.

TABLE 5-7: CONFIGURATION BITS DESCRIPTION (CONTINUED)

Bit Field	Register	Description
BOREN	FBORPOR	PBOR Enable 1 = PBOR enabled 0 = PBOR disabled
BORV<1:0>	FBORPOR	Brown-out Voltage Select 11 = 2.0V (not a valid operating selection) 10 = 2.7V 01 = 4.2V 00 = 4.5V
FPWRT<1:0>	FBORPOR	Power-on Reset Timer Value Select 11 = PWRT = 64 ms 10 = PWRT = 16 ms 01 = PWRT = 4 ms 00 = Power-up Timer disabled
GCP	FGS	General Code Segment Code Protect 1 = User memory is not code protected 0 = User memory is code protected
GWRP	FGS	General Code Segment Write Protect 1 = User program memory is not write protected 0 = User program memory is write protected
BKBUG	FICD	Debugger/Emulator Enable 1 = Device will reset into Operational mode 0 = Device will reset into Debug/Emulation mode
COE	FICD	Debugger/Emulator Enable 1 = Device will reset into Operational mode 0 = Device will reset into Clip-on Emulation mode
ICS<1:0>	FICD	ICD Communication Channel Select 11 =Communicate on PGC/EMUC and PGD/EMUD 10 =Communicate on EMUC1 and EMUD1 01 =Communicate on EMUC2 and EMUD2 00 =Communicate on EMUC3 and EMUD3
RESERVED	RESERVED1, RESERVED2, FGS	Reserved (read as '1', write as '1')
—	FOSC, FWDT, FBORPOR, FGS	Unimplemented (read as '0', write as '0')

TABLE 5-8: dsPIC30F CONFIGURATION REGISTERS (FOR dsPIC30F2010, dsPIC30F4011/4012, dsPIC30F5011/5013, 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	RESERVED1	—	—	Reserved(2)		—	—	—	Reserved(2)	—	—	—	—	Reserved(2)			
0xF80008	RESERVED2	—	—	Reserved(2)		—	—	Reserved(2)		—	—	—	—	Reserved(2)			
0xF8000A	FGS	—	—	—	—	—	—	—	—	—	—	—	—	—	Reserved(2)	GCP	GWRP
0xF8000C	FICD	BKBUG	COE	—	—	—	—	—	—	—	—	—	—	—	—	ICS<1:0>	

Note 1: On the 5011, 5013, 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 dsPIC30F2011/2012, dsPIC30F3010/3011/3012/3013/3014, dsPIC30F4013, dsPIC30F5015, dsPIC30F6011A/6012A/6013A/6014A)

Address	Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xF80000	FOSC	FCKSM<1:0>		—	—	—	FOS<2:0>			—	—	—	FPR<4:0>				
0xF80002	FWDT	FWDTEN	—	—	—	—	—	—	—	—	—	FWPSA<1:0>		FWPSB<3:0>			
0xF80004	FBORPOR	MCLREN	—	—	—	—	PWMPIN(1)	HPOL(1)	LPOL(1)	BOREN	—	BORV<1:0>		—	—	FPWRT<1:0>	
0xF80006	RESERVED1	—	—	Reserved(2)		—	—	—	Reserved(2)	—	—	—	—	Reserved(2)			
0xF80008	RESERVED2	—	—	Reserved(2)		—	—	Reserved(2)		—	—	—	—	Reserved(2)			
0xF8000A	FGS	—	—	—	—	—	—	—	—	—	—	—	—	—	Reserved(2)	GCP	GWRP
0xF8000C	FICD	BKBUG	COE	—	—	—	—	—	—	—	—	—	—	—	—	ICS<1:0>	

Note 1: On the 2011, 2012, 3012, 3013, 3014, 4013, 6011A, 6012A, 6013A and 6014A 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'.

Advance Information
© 2004 Microchip Technology Inc.

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 can not 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 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. A flowchart for configuration bit programming is shown in Figure 5-5.

Note:	If the General Code Segment Code Protect bit (GCP) is programmed to '0', code memory is code protected and can not 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

After 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 that the programming was successful.

Any unimplemented configuration bits are read-only and read '0'. The reserved bits in the RESERVED1, RESERVED2 and FGS registers are read-only and read '1'.

5.7.4 CODE-PROTECT CONFIGURATION BITS

The FGS configuration register is a special configuration register that controls code protection for the dsPIC30F. Two forms of code protection are provided. One form prevents code memory from being written (write protection), and the other prevents code memory from being read (read protection).

GWRP, FSG<0> controls write protection and GCP, FSG<1> controls read protection. Protection is enabled when the respective bit is '0'. The chip erase ERASEB command sets GWRP and GCP to '1', which allows the device to be programmed.

When write protection is enabled (GWRP = 0), any programming operation to code memory will fail. When read protection is enabled (GCP = 0), 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.

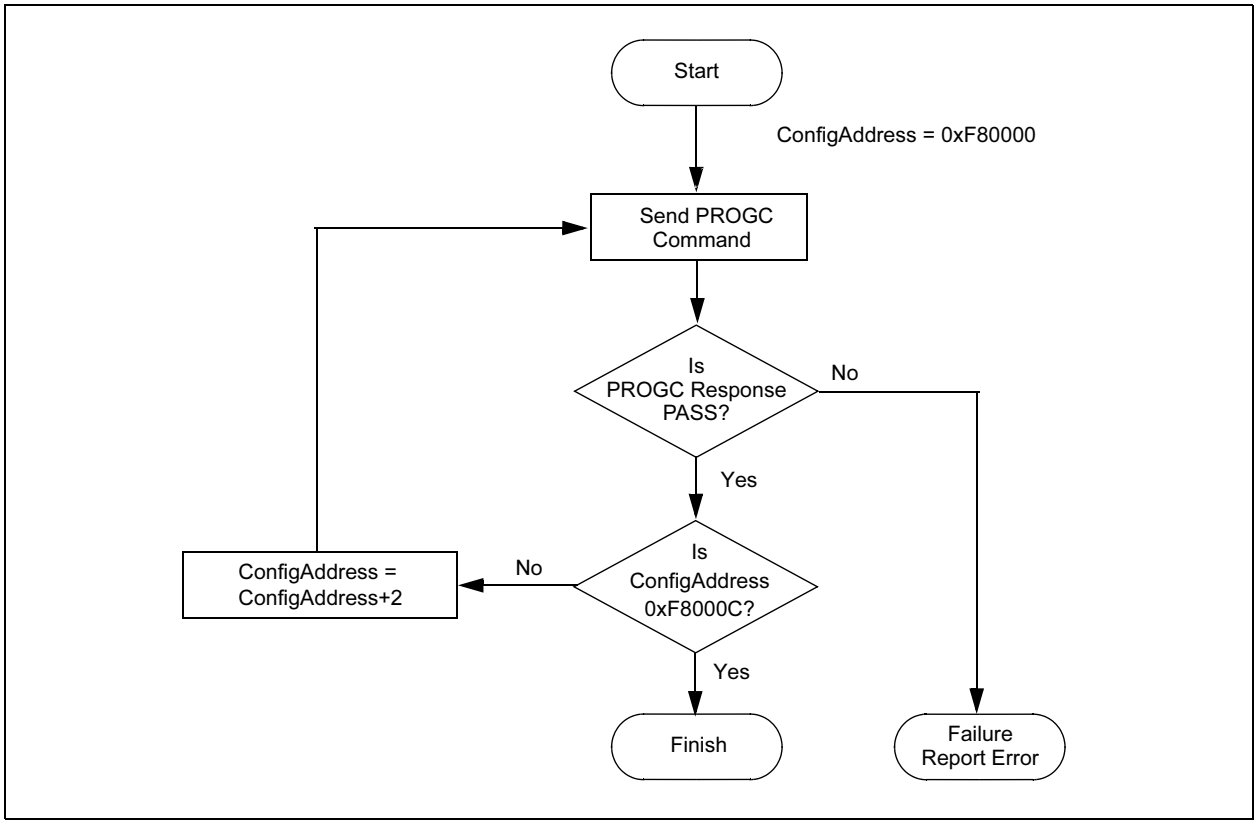
It is imperative that both GWRP and GCP are '1' while the device is being programmed and verified. Only after the device is programmed and verified should either GWRP or GCP be programmed to '0' (see **Section 5.7 "Configuration Bits Programming"**).

Note:	ERASEB is the only way to reprogram code protect bits from ON ('0') to OFF ('1').

5.8 Exiting ICSP Mode

The ICSP mode is exited by removing power from the device or bringing $\overline{MCLR}$ to V_{IL}. When normal user mode is next entered, the program that was stored using ICSP will execute.

FIGURE 5-5: CONFIGURATION BIT PROGRAMMING FLOW



 © 2004 Microchip Technology Inc.

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
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 can not 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 starting to program, 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 code protect bits, the entire chip must first be erased with the ERASEB command. Then, the code protect bits can be reprogrammed using the PROGC command.

Note:	If read or write code protection is enabled, no modifications can be made to any region of code memory until code protection is disabled. Code protection can only be disabled by performing a chip erase with the ERASEB command.

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 details on this command.

6.5 Data EEPROM Information in the Hex File

To allow portability of code, the programmer must read the data EEPROM information from the hex file. If data EEPROM information is not present, a simple warning message should be issued by the programmer. Similarly, when saving a hex 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.

6.6 Configuration Information in the Hex File

To allow portability of code, the programmer must read the Configuration register locations from the hex file. If configuration information is not present in the hex file, then a simple warning message should be issued by the programmer. Similarly, while saving a hex file, all configuration information must be included. An option to not include the configuration information can be provided.

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

6.7 Checksum Computation

Checksums for the dsPIC30F are 16 bits in size. The checksum is calculated by summing the following:

- Contents of code memory locations
- Contents of configuration registers

Table A-1 in Appendix A describes how to calculate the checksum for each device. All memory locations are summed one byte at a time, using only their native data size. Namely, configuration and device ID registers are summed by adding the lower two bytes of these locations (the upper byte is ignored), and code memory is summed by adding all three bytes of code memory.

Note:	The checksum calculation differs depending on the code protect setting. Table A-1 describes how to compute the checksum for an unprotected device and a read-protected device. Regardless of the code protect setting, the Configuration registers can always be read.

 © 2004 Microchip Technology Inc.

7.0 PROGRAMMER – PROGRAMMING EXECUTIVE COMMUNICATION

7.1 Communication Overview

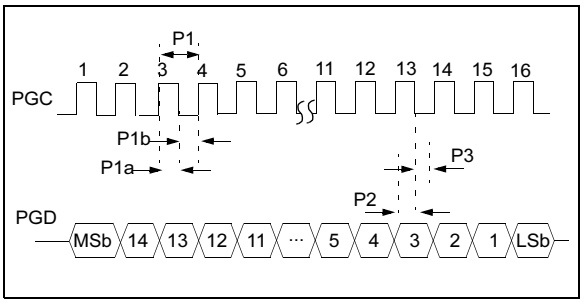
The programmer and programming executive have a master-slave relationship, where the programmer is the master programming device and the programming executive is the slave.

All communication is initiated by the programmer in the form of a command. Only one command at a time can be sent to the programming executive. In turn, the programming executive only sends one response to the programmer after receiving and processing a command. The Programming Executive command set is described in **Section 8.0 "Programming Executive Commands"**. The response set is described in **Section 9.0 "Programming Executive Responses"**.

7.2 Communication Interface and Protocol

The ICSP interface is a 2-wire SPI implemented using the PGC and PGD pins. The PGC pin is used as a clock input pin, and the clock source must be provided by the programmer. The PGD pin is used for sending command data to and receiving response data from the programming executive. All serial data is transmitted on the rising edge of PGC and latched on the falling edge of PGC. All data transmissions are sent Most Significant bit (MSb) first using 16-bit mode (see Figure 7-1).

FIGURE 7-1: PROGRAMMING EXECUTIVE SERIAL TIMING



Since a 2-wire SPI is used, and data transmissions are bidirectional, a simple protocol is used to control the direction of PGD. When the programmer completes a command transmission, it releases the PGD line and allows the programming executive to drive this line high. The programming executive keeps the PGD line high to indicate that it is processing the command.

After the programming executive has processed the command, it brings PGD low for 15 μsec to indicate to the programmer that the response is available to be clocked out. The programmer can begin to clock out the response 20 μsec after PGD is brought low, and it must provide the necessary amount of clock pulses to receive the entire response from the programming executive.

After the entire response is clocked out, the programmer should terminate the clock on PGC until it is time to send another command to the programming executive. This protocol is shown in Figure 7-2.

7.3 SPI Rate

In ICSP mode, the dsPIC30F operates from the Fast Internal RC oscillator, which has a nominal frequency of 8 MHz. This oscillator frequency yields an effective system clock frequency of 2 MHz. Since the SPI module in slave mode can operate up to one half of the system clock, the SPI can operate up to a 1 MHz data rate. To minimize programming time, it is recommended that a 1 MHz clock be provided by the programmer.

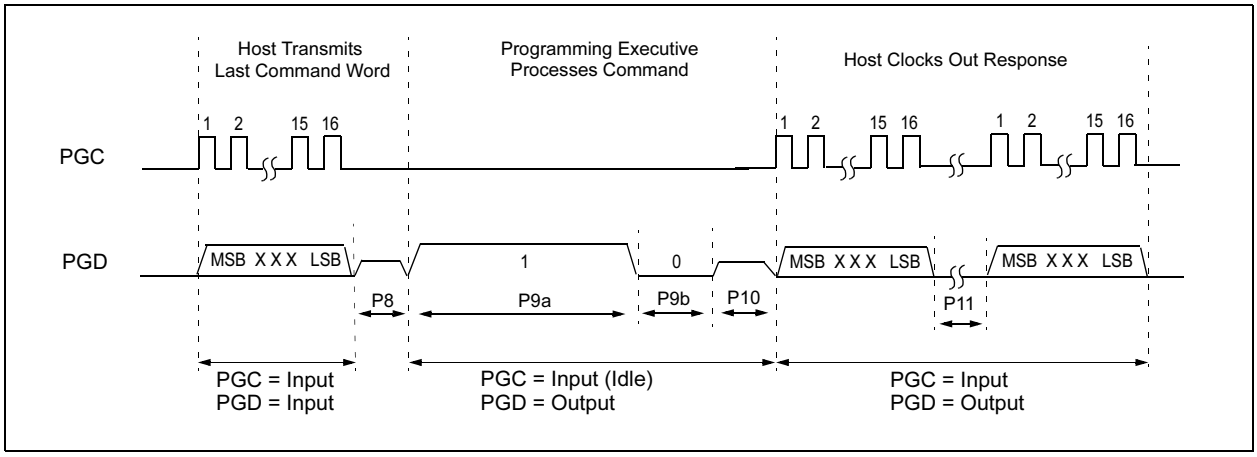
Note:	If the programmer provides the SPI with a clock faster than 1 MHz, the behavior of the programming executive will be unpredictable.

7.4 Time-outs

The programming executive uses no Watchdog or time-out for transmitting responses to the programmer. If the programmer does not follow the flow control mechanism using PGC as described in **Section 7.2 "Communication Interface and Protocol"**, it is possible that the programming executive will behave unexpectedly while trying to send a response to the programmer. Since the programming executive has no time-out, it is imperative that the programmer correctly follow the described communication protocol.

As a safety measure, the programmer should use the command time-outs identified in Table 8-1. If the command time-out expires, the programmer should reset the programming executive and start programming the device again.

FIGURE 7-2: PROGRAMMING EXECUTIVE – PROGRAMMER COMMUNICATION PROTOCOL



 © 2004 Microchip Technology Inc.

8.0 PROGRAMMING EXECUTIVE COMMANDS

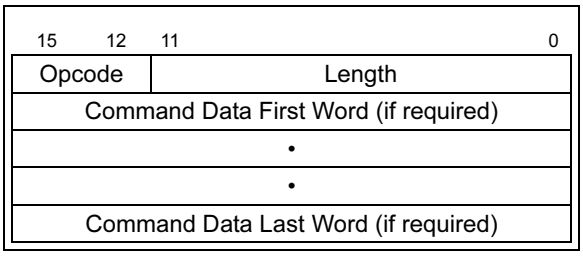
8.1 Command Set

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 (**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



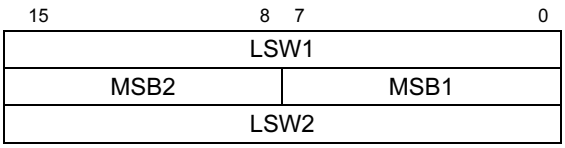
The command opcode must match one of those in the command set. Any command that is received which does not match the list in Table 8-1 will return a "NACK" response (see **Section 9.2.1 "`Opcode` Field"**).

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 can not 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"**.

TABLE 8-1: PROGRAMMING EXECUTIVE COMMAND SET

Opcode	Mnemonic	Length (16-bit words)	Time Out	Description
0x0	SCHECK	1	1 msec	Sanity check.
0x1	READD	4	1 msec/ row	Read N 16-bit words of data EEPROM, configuration registers or device ID starting from specified address.
0x2	READP	4	1 msec/ row	Read N 24-bit instruction words of code memory starting from specified address.
0x3	RESERVED	N/A	N/A	This command is reserved. It will return a NACK.
0x4	PROGD	19	5 msec	Program one row of data EEPROM at the specified address, then verify.
0x5	PROGP	51	5 msec	Program one row of code memory at the specified address, then verify.
0x6	PROGC	4	5 msec	Write byte or 16-bit word to specified configuration register.
0x7	ERASEB	2	5 msec	Bulk erase (entire chip by segment, code memory or data EEPROM).
0x8	ERASED	3	5 msec/ row	Erase rows of data EEPROM from specified address.
0x9	ERASEP	3	5 msec/ row	Erase rows of code memory from specified address.
0xA	QBLANK	3	1 msec	Query if the code memory and data EEPROM are blank.
0xB	QVER	1	1 msec	Query the programming executive software version.

Note 1: One row of code memory consists of (32) 24-bit words. Refer to Table 5-2 for device-specific information.
2: One row of data EEPROM consists of (16) 16-bit words. Refer to Table 5-3 for device-specific information.

© 2004 Microchip Technology Inc.

8.5 Command Descriptions

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

8.5.1 SCHECK COMMAND

15 12	11 0
Opcode	Length

Field	Description
Opcode	0x0
Length	0x1

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

Expected Response (2 words):
0x1000
0x0002

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

8.5.2 READD COMMAND

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

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

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

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

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

8.5.3 READP COMMAND

15 12	11 8	7 0
Opcode	Length	
N		
Reserved	Addr_MSB	
Addr_L:S		

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 the 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	Adddr_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 it is programmed to the destination address specified by Addr_MSB and Addr_LSB. The destination address should be a multiple of 0x20.

After 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.

© 2004 Microchip Technology Inc.

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		Assr_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	1	0
Opcode	Length		
Reserved			MS

Field	Description
Opcode	0x7
Length	0x2
Reserved	0x0
MS	Memory Select: 0x0 for Code Memory erase 0x1 for Data EEPROM erase 0x2 is Reserved 0x3 for Chip Erase

The ERASEB command performs a bulk erase. The MS field selects the memory to be bulk erased: Code Memory (0x0), Data EEPROM (0x1) or the entire chip (0x3).

When 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, device ID and configuration registers that are not code-protected remain intact after a chip erase.

Expected Response (2 words):
0x1700
0x0002

Note 1:	This command can not be used 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. There is no method to erase the code protect configuration bits in low voltage programming systems.

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 0x10. 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:	ERASED can not be used to erase the configuration registers or device ID. Code-protect configuration registers can only be erased with ERASEB, while the device ID is read-only.

© 2004 Microchip Technology Inc.

8.5.9 ERASEP COMMAND

15 12	11 8	7 0
Opcode	Length	
Num_Rows		Addr_MSB
Addr_LS		

Field	Description
Opcode	0x9
Length	0x3
Num_Rows	Number of rows to erase
Addr_MSB	MSB of 24-bit base address
Addr_LS	LS 16 bits of 24-bit base address

The ERASEP command erases the specified number of rows of code memory from the specified base address. The specified base address must be a multiple of 0x40.

After the erase is performed, all targeted words of code memory contain 0xFFFFFF.

Expected Response (2 words):
0x1900
0x0002

Note:	ERASEP can not be used to erase the configuration registers or device ID. Code-protect configuration registers can only be erased with ERASEB, while the device ID is read-only.

8.5.10 QBLANK COMMAND

15 12	11 0
Opcode	Length
PSize	
Reserved	DSize

Field	Description
Opcode	0xA
Length	0x3
PSize	Length of program memory to check (in 24-bit words), max of 49152
Reserved	0x0
DSize	Length of data memory to check (in 16-bit words), max of 2048

The QBLANK command queries the programming executive to determine if the contents of code memory, data EEPROM and code-protect configuration bits (GCP and GWRP) are blank (contains all '1's). The size of code memory and data EEPROM to check must be specified in the command.

The blank check for code memory begins at 0x0 and advances toward larger addresses for the specified number of instruction words. The blank check for data EEPROM begins at 0x7FFFFE and advances toward smaller addresses for the specified number of data words.

QBLANK returns a QE_Code of 0xF0 if the specified Code Memory, Data EEPROM and Code Protect bits are blank. Otherwise, QBLANK returns a QE_Code of 0x0F.

Expected Response (2 words for blank device):
0x1AF0
0x0002

Expected Response (2 words for non-blank device):
0x1A0F
0x0002

Note:	QBLANK does not check the system operation configuration bits since these bits are not set to '1' when a chip erase is performed.

8.5.11 QVER COMMAND

15 12	11 0
Opcode	Length

Field	Description
Opcode	0xB
Length	0x1

The QVER command queries the version of the programming executive software stored in test memory. The “version.revision” information is returned in the response’s QE_Code using a single byte with the following format: main version in upper nibble and revision in the lower nibble (i.e., 0x23 means version 2.3 of programming executive software).

Expected Response (2 words):
0x1BMN (where “MN” stands for version M.N)
0x0002

 © 2004 Microchip Technology Inc.

9.0 PROGRAMMING EXECUTIVE RESPONSES

9.1 Overview

The programming executive sends a response to the programmer for each command that it receives. The response indicates if the command was processed correctly. It includes any required response data or error data.

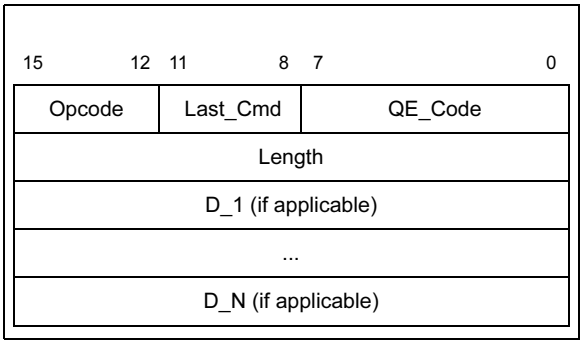
The programming executive response set is shown in Table 9-1. This table contains the opcode, mnemonic and description for each response. The response format is described in **Section 9.2 "Response Format"**.

TABLE 9-1: PROGRAMMING EXECUTIVE RESPONSE SET

Opcode	Mnemonic	Description
0x1	PASS	Command successfully processed.
0x2	FAIL	Command unsuccessfully processed.
0x3	NACK	Command not known.

9.2 Response Format

All programming executive responses have a general format consisting of a two-word header and any required data for the command.



Field	Description
Opcode	Response opcode.
Last_Cmd	Programmer command that generated the response.
QE_Code	Query code or Error code.
Length	Response length in 16-bit words (includes 2 header words.)
D_1	First 16-bit data word (if applicable).
D_N	Last 16-bit data word (if applicable).

9.2.1 Opcode FIELD

The Opcode is a 4-bit field in the first word of the response. The Opcode indicates how the command was processed (see Table 9-1). If the command was processed successfully, the response opcode is PASS. If there was an error in processing the command, the response opcode is FAIL, and the QE_Code indicates the reason for the failure. If the command sent to the programming executive is not identified, the programming executive returns a NACK response.

9.2.2 Last_Cmd FIELD

The Last_Cmd is a 4-bit field in the first word of the response and indicates the command that the programming executive processed. Since the programming executive can only process one command at a time, this field is technically not required. However, it can be used to verify that the programming executive correctly received the command that the programmer transmitted.

9.2.3 QE_Code FIELD

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

When the programming executive processes one of the two query commands (QBLANK or QVER), the returned opcode is always PASS, and the QE_Code holds the query response data. The format of the QE_Code for both queries is shown in Table 9-2.

TABLE 9-2: QE_Code FOR QUERIES

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

When the programming executive processes any command other than a Query, the QE_Code represents an error code. Supported error codes are shown in Table 9-3. If a command is successfully processed, the returned QE_Code is set to 0x0, which indicates that there was no error in the command processing. If the verify of the programming for the PROGD, PROGP or PROGC command fails, the QE_Code is set to 0x1. For all other programming executive errors, the QE_Code is 0x2.

TABLE 9-3: QE_Code FOR NON-QUERY COMMANDS

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

9.2.4 RESPONSE LENGTH

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

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

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

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

 © 2004 Microchip Technology Inc.

10.0 DEVICE ID

The device ID region of memory can be used to determine mask, variant and manufacturing information about the chip. The device ID region is 2 x 16-bits and it can be read using the READD command. This region of memory is read-only and can also be read when code protection is enabled.

Table 10-1 shows the device ID for each device, Table 10-2 shows the device ID registers and Table 10-3 describes the bit field of each register.

TABLE 10-1: DEVICE IDS

dsPIC30F Device	Mask	Variant	DEVID
2010	0x1	0x0	0x0040
2011	0x3	0x0	0x00C0
	0x9	0x0	0x0240
2012	0x3	0x2	0x00C2
	0x9	0x1	0x0241
3010	0x7	0x0	0x01C0
3011	0x7	0x0	0x01C1
3012	0x3	0x1	0x00C1
3013	0x3	0x3	0x00C3
3014	0x5	0x0	0x0140
4011	0x4	0x1	0x0101
4012	0x4	0x0	0x0100
4013	0x5	0x1	0x0141
5011	0x2	0x0	0x0080
5013	0x2	0x1	0x0081
5015	0x8	0x0	0x0200
6010	0x6	0x8	0x0188
6011	0x6	0x12	0x0192
6011A	0xB	0x0	0x02C0
6012	0x6	0x13	0x0193
6012A	0xB	0x1	0x02C1
6013	0x6	0x17	0x0197
6013A	0xB	0x2	0x02C2
6014	0x6	0x18	0x0198
6014A	0xB	0x3	0x02C3

TABLE 10-2: dsPIC30F DEVICE ID REGISTERS

Address	Name	Bit															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF0000	DEVID	MASK<9:0>										VARIANT<5:0>					
0xFF0002	DEVREV	PROC<3:0>				REV<5:0>						DOT<5:0>					

TABLE 10-3: DEVICE ID BITS DESCRIPTION

Bit Field	Register	Description
MASK<9:0>	DEVID	Encodes the MASKSET ID of the device.
VARIANT<5:0>	DEVID	Encodes the VARIANT derived from MASKSET of the device.
PROC<3:0>	DEVREV	Encodes the process of the device.
REV<5:0>	DEVREV	Encodes the major revision number of the device.
DOR<5:0>	DEVREV	Encodes the minor revision number of the device.

11.0 STDP MODE

11.1 STDP Mode

STDP mode is a special programming protocol that allows you to read and write to dsPIC30F memory. The STDP 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 if 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 STDP mode, the system clock is taken from the PGC pin, regardless of the device's oscillator configuration bits. All instructions are 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:	During STDP operation, the operating frequency of PGC must not exceed 5 MHz.

11.2 STDP Operation

Upon entry into STDP 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, and the REGOUT control code is used to read data out of the device via the VISI register. The operation details of STDP 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 STDP 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 11-1).

Note 1:	Coming out of Reset, the first 4-bit control code is always forced to SIX and a forced NOP instruction is executed by the CPU. After the forced SIX is clocked in, STDP operation resumes as normal (the next 24 clock cycles load the first instruction word to the CPU).
2:	TBLRDH, TBLRDL, TBLWTH and TBLWTL instructions must be followed by a NOP instruction.

11.2.2 REGOUT SERIAL INSTRUCTION EXECUTION

The REGOUT control code allows for data to be extracted from the device in STDP mode. It is used to clock the contents of the VISI register out of the device over the PGD pin. After the REGOUT control code is received, eight clock cycles are required to process the command. During this time, the CPU is held idle. After these eight cycles, an additional 16 cycles are required to clock the data out (see Figure 11-2).

The REGOUT instruction is unique because the PGD pin is an input when the control code is transmitted to the device. However, after the control code is processed, the PGD pin becomes an output as the VISI register is shifted out. After the contents of the VISI are shifted out, PGD becomes an input again as the state machine holds the CPU idle until the next 4-bit control code is shifted in.

Note:	After the contents of VISI are shifted out, the dsPIC device maintains PGD as an output until the first rising edge of the next clock is received.

 © 2004 Microchip Technology Inc.

FIGURE 11-1: SIX SERIAL EXECUTION

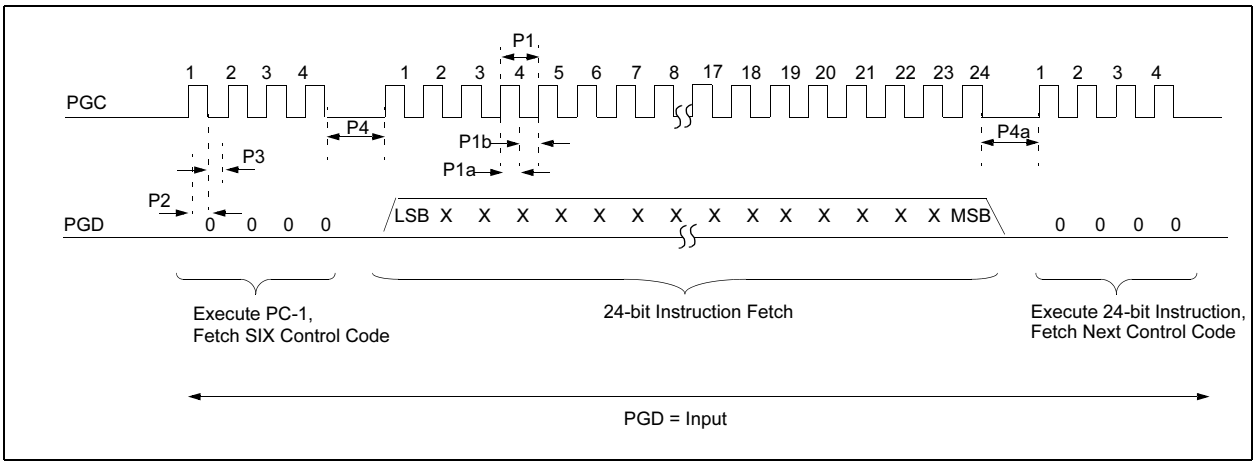
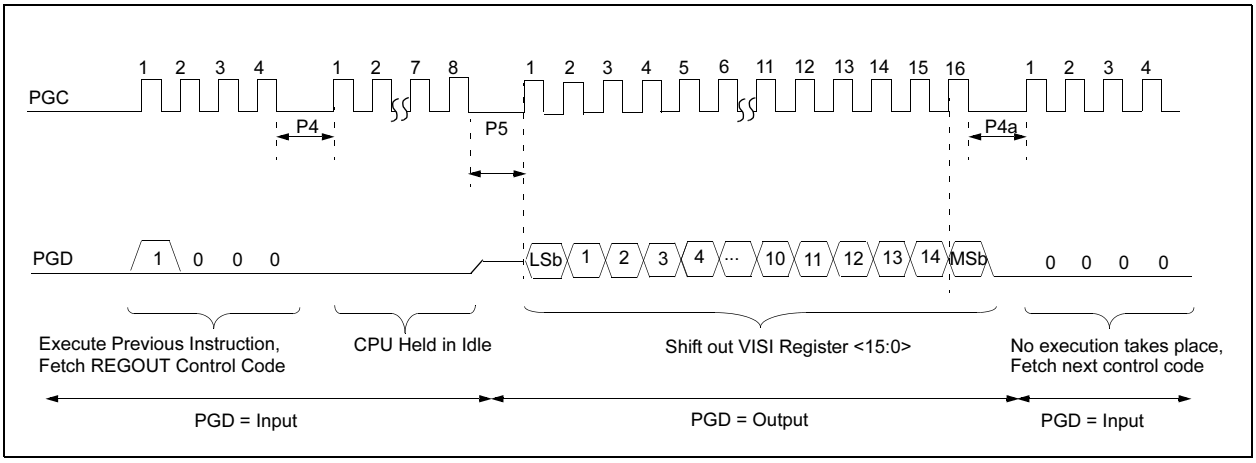


FIGURE 11-2: REGOUT SERIAL EXECUTION



11.3 Entering STDP Mode

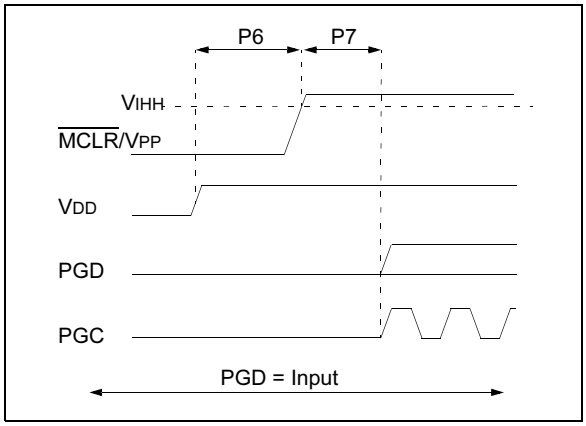
The STDP mode is entered by holding PGC and PGD low, and then raising MCLR/VPP to VIHH (high voltage), as shown in Figure 11-3.

Note 1: The sequence that places the device into STDP mode places all unused I/O pins to the high impedance state.

2: Once STDP mode is entered, the PC is set to 0x0 (the Reset vector).

3: Before leaving the Reset vector with a GOTO instruction, two NOP instructions must be executed.

FIGURE 11-3: ENTERING STDP MODE



11.4 Flash Memory Programming in STDP Mode

Programming in STDP Mode is described in **Section 11.4.1 "Programming Operations"** through **Section 11.4.3 "Starting and Stopping a Programming Cycle"**. Step-by-step procedures are described in **Section 11.5 "Erasing Program Memory in Normal Voltage Systems"** through **Section 11.14 "Reading the Application ID Word"**. All programming operations must use serial execution, as described in **Section 11.2 "STDP Operation"**.

11.4.1 PROGRAMMING OPERATIONS

Flash memory write and erase operations are controlled by the NVMCON register. Programming is performed by setting NVMCON to select the type of erase operation (Table 11-2) or write operation (Table 11-3), writing a key sequence to enable the programming, and initiating the programming by setting the WR control bit, NVMCON<15>.

In STDP mode, all programming operations are externally timed. An external 2-msec delay must be used between setting the WR control bit and clearing the WR control bit to complete the programming operation.

TABLE 11-2: NVMCON ERASE OPERATIONS

NVMCON Value	Erase Operation
0x407F	Erase all code memory, data memory, executive memory and code protect bits (does not erase UNIT ID).
0x4075	Erase 1 row (16 words) of data memory.
0x4074	Erase 1 word of data memory.
0x4072	Erase all executive memory.
0x4071	Erase 1 row (32 instruction words) from 1 panel of code memory.

TABLE 11-3: NVMCON WRITE OPERATIONS

NVMCON Value	Write Operation
0x4008	Write 1 word to configuration memory.
0x4005	Write 1 row (16 words) to data memory.
0x4004	Write 1 word to data memory.
0x4001	Write 1 row (32 instruction words) into 1 panel of program memory.

 © 2004 Microchip Technology Inc.

11.4.2 UNLOCKING NVMCON FOR PROGRAMMING

Writes to the WR bit (NVMCON<15>) are locked to prevent accidental programming from taking place. Writing a key sequence to the NVMKEY register unlocks the WR bit and allows it to be written to. The unlock sequence is performed as follows:

```
MOV    #0x55, W8
MOV    W8, NVMKEY
MOV    #0xAA, W9
MOV    W9, NVMKEY
```

Note:	Any working register or working register pair can be used to write the unlock sequence.

11.4.3 STARTING AND STOPPING A PROGRAMMING CYCLE

After the unlock key sequence has been written to the NVMKEY register, the WR bit (NVMCON<15>) is used to start and stop an erase or write cycle. Setting the WR bit initiates the programming cycle. Clearing the WR bit terminates the programming cycle.

All erase and write cycles must be externally timed. An external delay must be used between setting and clearing the WR bit. Starting and stopping a programming cycle is performed as follows:

```
BSET   NVMCON, #WR
<Wait 2 msec>
BCLR   NVMCON, #WR
```

11.5 Erasing Program Memory in Normal Voltage Systems

The procedure for erasing program memory (all of code memory, data memory, executive memory and code-protect bits) consists of setting NVMCON to 0x407F, unlocking NVMCON for erasing, and then executing the programming cycle. This method of bulk erasing program memory only works for systems where VDD is between 4.5 volts and 5.5 volts. The method for erasing program memory for systems with a lower VDD (3.0 volts-4.5 volts) is described in **Section 11.6 "Erasing Program Memory in Low Voltage Systems"**.

Table 11-4 shows the STDP programming process for bulk erasing program memory. This process includes the STDP command code, which must be transmitted (for each instruction) Least Significant bit first using the PGC and PGD pins (see Figure 11-1).

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

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

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	000000	NOP
0000	040100	GOTO 0x100
0000	000000	NOP
Step 2: Set the NVMCON to erase all of Program Memory.		
00000	2407FA	MOV #0x407F, W10
0000	883B0A	MOV W10, NVMCON
Step 3: Unlock the NVMCON for programming.		
0000	200558	MOV #0x55, W8
0000	883B38	MOV W8, NVMKEY
0000	200AA9	MOV #0xAA, W9
0000	883B39	MOV W9, NVMKEY
Step 4: Initiate the erase cycle.		
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
—	—	Externally time 2 msec
0000	A9E761	BCLR NVMCON, #WR
0000	000000	NOP
0000	000000	NOP

11.6 Erasing Program Memory in Low Voltage Systems

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

Due to security restrictions, the code protection bits GCP and GWRP in the FGS register can not be erased in low voltage systems. Once any bits in the FGS register are programmed to '0', they can only be set back to '1' by performing a bulk erase in a normal voltage system.

Normal voltage systems may also erase Program Memory by following the procedure in Table 11-5. However, since this method is more time consuming and does not clear the code protect bits, it is not recommended.

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

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

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	000000	NOP
0000	040100	GOTO 0x100
0000	000000	NOP
Step 2: Initialize NVMADR and NVMADRU to erase code memory and initialize W7 for row address updates.		
0000	EB0300	CLR W6
0000	883B16	MOV W6, NVMADR
0000	883B26	MOV W6, NVMADRU
0000	200407	MOV #0x40, W7
Step 3: Set NVMCON to erase 1 row of code memory.		
0000	24071A	MOV #0x4071, W10
0000	883B0A	MOV W10, NVMCON
Step 4: Unlock the NVMCON to erase 1 row of code memory.		
0000	200558	MOV #0x55, W8
0000	883B38	MOV W8, NVMKEY
0000	200AA9	MOV #0xAA, W9
0000	883B39	MOV W9, NVMKEY
Step 5: Initiate the erase cycle.		
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
—	—	Externally time 2 msec
0000	A9E761	BCLR NVMCON, #WR
0000	000000	NOP
0000	000000	NOP

© 2004 Microchip Technology Inc.

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

Command (Binary)	Data (Hex)	Description
Step 6: Update the row address stored in NVMADRU:NVMADR. When W6 rolls over to 0x0, NVMADRU must be incremented.		
0000	430307	ADD W6, W7, W6
0000	AF0042	BTSC SR, #C
0000	EC2764	INC NVMADRU
0000	883B16	MOV W6, NVMADR
Step 7: Reset device internal PC.		
0000	040100	GOTO 0x100
0000	000000	NOP
Step 8: Repeat Steps 3-7 until all rows of code memory are erased.		
Step 9: Initialize NVMADR and NVMADRU to erase executive memory and initialize W7 for row address updates.		
0000	EB0300	CLR W6
0000	883B16	MOV W6, NVMADR
0000	200806	MOV #0x80, W6
0000	883B26	MOV W6, NVMADRU
0000	200407	MOV #0x40, W7
Step 10: Set NVMCON to erase 1 row of executive memory.		
0000	24071A	MOV #0x4071, W10
0000	883B0A	MOV W10, NVMCON
Step 11: Unlock the NVMCON to erase 1 row of executive memory.		
0000	200558	MOV #0x55, W8
0000	883B38	MOV W8, NVMKEY
0000	200AA9	MOV #0xAA, W9
0000	883B39	MOV W9, NVMKEY
Step 12: Initiate the erase cycle.		
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
—	—	Externally time 2 msec
0000	A9E761	BCLR NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
Step 13: Update the row address stored in NVMADR.		
0000		ADD W6, W7, W6
0000		MOV W6, NVMADR
Step 14: Reset device internal PC.		
0000	040100	GOTO 0x100
0000	000000	NOP
Step 15: Repeat Steps 10-14 until all 24 rows of executive memory are erased.		
Step 16: Initialize NVMADR and NVMADRU to erase data memory and initialize W7 for row address updates.		
0000	2XXXX6	MOV #<lower 16-bits of starting Data EEPROM address>, W6
0000	883B16	MOV W6, NVMADR
0000	2007F6	MOV #0x7F, W6
0000	883B16	MOV W6, NVMADRU
0000	200207	MOV #0x20, W7
Step 17: Set NVMCON to erase 1 row of data memory.		
0000	24075A	MOV #0x4075, W10
0000	883B0A	MOV W10, NVMCON

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

Command (Binary)	Data (Hex)	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 2 msec
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.		

 © 2004 Microchip Technology Inc.

11.7 Clearing the Configuration Registers

Configuration registers are not erasable. It is recommended that all configuration register bits be cleared by programming them to 0x1 *after* erasing Program Memory. With the exception of the FGS register, Configuration registers can be cleared by programming 0xFFFF to each register. However, each Configuration register contains unimplemented bits that will read as '0' (see Table and Table 5-9). This means that when a verify is performed after programming the configuration registers with 0xFFFF, the cleared values shown in Table 11-6 and Table 11-7 will be read.

The FGS configuration register is special since it enables code protection for the device. For security purposes, once the GCP or GWRP bits in this register are programmed to '0' (to enable code protection), they can only be set back to '1' by performing a bulk erase as described in **Section 11.5 "Erasing Program Memory in Normal Voltage Systems"**. Programming the FGS configuration register 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 11-8 shows the STDP programming details for clearing the configuration registers. In Step 1 the Reset vector is exited. In Step 2 the NVMCON register is set to program 1 configuration register. In Step 3 the write pointer (W7) is loaded with 0x0000, which is the original destination address (in TBLPAG 0xF8 of Program Memory). In Step 4 the TBLPAG register is initialized to 0xF8, for writing to the configuration registers. In Step 5 the value to write to each configuration register (0xFFFF) is loaded to W0. In Step 6 the read pointer (W6) is cleared (to point to working register W0) and the write latch is written using the TBLWTL instruction. In Steps 7 and 8 NVMCON is unlocked for programming and the programming cycle is initiated as described in **Section 11.4 "Flash Memory Programming in STDP 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 4-9 are repeated six times until all configuration register are cleared.

TABLE 11-6: CLEARED CONFIGURATION REGISTER VALUES (FOR dsPIC30F2010, dsPIC30F4011/4012, dsPIC30F5011/5013, dsPIC30F6010/6011/6012/6013/6014)

Address	Register	Cleared Value
0xF80000	FOSC	0xC30F
0xF80002	FWDT	0x803F
0xF80004	FBORPOR	0x87B3
0xF80006	RESERVED1	0x310F
0xF80008	RESERVED2	0x330F
0xF8000A	FGS	0x0007
0xF8000C	FICD	0xC003

TABLE 11-7: CLEARED CONFIGURATION REGISTER VALUES (FOR dsPIC30F2011/2012, dsPIC30F3010/3011/3012/ 3013/3014, dsPIC30F4013, dsPIC30F5015, dsPIC30F6011A/6012A/6013A/6014A)

Address	Register	Cleared Value
0xF80000	FOSC	0xC71F
0xF80002	FWDT	0x803F
0xF80004	FBORPOR	0x87B3
0xF80006	RESERVED1	0x310F
0xF80008	RESERVED2	0x330F
0xF8000A	FGS	0x0007
0xF8000C	FICD	0xC003

TABLE 11-8: SERIAL INSTRUCTION EXECUTION FOR CLEARING CONFIGURATION REGISTERS

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	000000	NOP
0000	040100	GOTO 0x100
0000	000000	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	24008A	MOV #0x4008, W10
0000	883B0A	MOV W10, NVMCON
Step 4: Initialize the TBLPAG register.		
0000	200F80	MOV #0xF8, W0
0000	880190	MOV W0, TBLPAG
Step 5: Load the configuration register data to W0.		
0000	2FFFF0	MOV #0xFFFF, W0
Step 6: Set the read pointer (W6) and load the write latch.		
0000	EB0300	CLR W6
0000	BB1B96	TBLWTL [W6], [W7++]
0000	000000	NOP
0000	000000	NOP
Step 7: Unlock the NVMCON for programming.		
0000	200558	MOV #0x55, W8
0000	883B38	MOV W8, NVMKEY
0000	200AA9	MOV #0xAA, W9
0000	883B39	MOV W9, NVMKEY
Step 8: Initiate the write cycle.		
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
—	—	Externally time 2 msec
0000	A9E761	BCLR NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
Step 9: Reset device internal PC.		
0000	040100	GOTO 0x100
0000	000000	NOP
Step 10: Repeat steps 3-9 until all 7 configuration registers are cleared.		

 © 2004 Microchip Technology Inc.

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 11-9 shows the STDP programming details, including the serial pattern with the STDP command code, which must be transmitted Least Significant bit first using the PGC and PGD pins (see Figure 11-1). 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, and 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 3-9 are repeated until all of code memory is programmed.

FIGURE 11-4: PACKED INSTRUCTION WORDS IN W0:W5

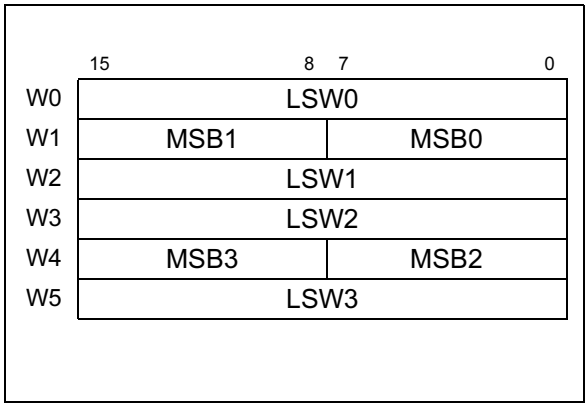


TABLE 11-9: SERIAL INSTRUCTION EXECUTION FOR WRITING CODE MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	000000	NOP
0000	040100	GOTO 0x100
0000	000000	NOP
Step 2: Set the NVMCON to program 32 instruction words.		
0000	24001A	MOV #0x4001, W10
0000	883B0A	MOV W10, NVMCON
Step 3: Initialize the write pointer (W7) for TBLWT instruction.		
0000	200xx0	MOV #<DestinationAddress23:16>, W0
0000	880190	MOV W0, TBLPAG
0000	2xxxx7	MOV #<DestinationAddress15:0>, W7
Step 4: Initialize the read pointer (W6) and load W0:W5 with the next 4 instruction words to program.		
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

TABLE 11-9: SERIAL INSTRUCTION EXECUTION FOR WRITING CODE MEMORY (CONTINUED)

Command (Binary)	Data (Hex)	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 3-5 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
0000	000000	NOP
0000	000000	NOP
—	—	Externally time 2 msec
0000	A9E761	BCLR NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
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.		

 © 2004 Microchip Technology Inc.

11.9 Writing Configuration Memory

The method for writing the configuration registers is identical to the method of clearing the configuration registers (Table 11-8). However, instead of writing 0xFFFF to the configuration register in Step 5, you write the desired value to the configuration register.

11.10 Writing Data Memory

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

Table 11-10 shows the STDP programming details for writing data memory. Note that a different NVMCON value is required to write to data memory, and that the TBLPAG register is hard coded to 0x7F (the upper byte address of all locations of data memory).

TABLE 11-10: SERIAL INSTRUCTION EXECUTION FOR WRITING DATA MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	000000	NOP
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 3-4 four times to load the write latches for 16 data words.		

TABLE 11-10: SERIAL INSTRUCTION EXECUTION FOR WRITING DATA MEMORY (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 7: Unlock the NVMCON for writing.		
0000	200558	MOV #0x55, W8
0000	883B38	MOV W8, NVMKEY
0000	200AA9	MOV #0xAA, W9
0000	883B39	MOV W9, NVMKEY
Step 8: Initiate the write cycle.		
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
—	—	Externally time 2 msec
0000	A9E761	BCLR NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
Step 9: Reset device internal PC.		
0000	040100	GOTO 0x100
0000	000000	NOP
Step 10: Repeat steps 2-9 until all data memory is programmed.		

 © 2004 Microchip Technology Inc.

11.11 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 a time.

Table 11-11 shows the STDP 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 register and W6 register. The upper byte of the starting source address is stored to TBLPAG, and 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 11-4). 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 11-11: SERIAL INSTRUCTION EXECUTION FOR READING CODE MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit Reset vector.		
0000	000000	NOP
0000	000000	NOP
0000	040100	GOTO 0x100
0000	000000	NOP
Step 2: Initialize TBLPAG and the read pointer (W6) for TBLRD instruction.		
0000	200xx0	MOV #<SourceAddress23:16>, W0
0000	880190	MOV W0, TBLPAG
0000	2xxxx6	MOV #<SourceAddress15:0>, W6
Step 3: Initialize the write pointer (W7) and store the next four locations of code memory to W0:W5.		
0000	EB0380	CLR W7
0000	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

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

Command (Binary)	Data (Hex)	Description
Step 4: Output W0:W5 using the VISI register and REGOUT command.		
0000	883C20	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register
0000	000000	NOP
0000	883C21	MOV W1, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register
0000	000000	NOP
0000	883C22	MOV W2, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register
0000	000000	NOP
0000	883C23	MOV W3, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register
0000	000000	NOP
0000	883C24	MOV W4, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register
0000	000000	NOP
0000	883C25	MOV W5, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register
0000	000000	NOP
Step 5: Reset the device internal PC.		
0000	040100	GOTO 0x100
0000	000000	NOP
Step 6: Repeat steps 3-5 until all desired code memory is read.		

 © 2004 Microchip Technology Inc.

11.12 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 11-12 shows the STDP programming details for reading all of 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 11-12: SERIAL INSTRUCTION EXECUTION FOR READING ALL CONFIGURATION MEMORY

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

11.13 Reading Data Memory

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

Table 11-13 shows the STDP 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 11-13: SERIAL INSTRUCTION EXECUTION FOR READING DATA MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit Reset vector.		
0000	000000	NOP
0000	000000	NOP
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	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.		

 © 2004 Microchip Technology Inc.

11.14 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. Then, the REGOUT control code must 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 11-14.

After 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 memory is described in **Section 12.0 "Programming the Programming Executive to Memory"**.

11.15 Exiting STDP Mode

After confirming that the programming executive is resident in memory or loading the programming executive, STDP 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 11-14: SERIAL INSTRUCTION EXECUTION FOR READING THE APPLICATION ID WORD

Command (Binary)	Data (Hex)	Description
Step 1: Exit Reset vector.		
0000	000000	NOP
0000	000000	NOP
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	205FE0	MOV #0x5BE, W0
0000	207841	MOV VISI, W1
0000	BA0890	TBLRDL [W0], [W1]
0000	000000	NOP
0000	000000	NOP
Step 3: Output the VISI register using the REGOUT command.		
0001	<VISI>	Clock out contents of the VISI register
0000	000000	NOP

12.0 PROGRAMMING THE PROGRAMMING EXECUTIVE TO MEMORY

12.1 Overview

If it is determined that the programming executive does not reside in executive memory (as described in **Section 4.0 "Confirming The Contents of Executive Memory"**), it must be programmed into executive memory using STDP and the techniques described in **Section 11.0 "STDP Mode"**.

Storing the programming executive to executive memory is similar to normal programming of code memory. Namely, the executive memory must first be erased, and then the programming executive must be programmed 32 words at a time. This control flow is summarized in Table 12-1.

TABLE 12-1: PROGRAMMING THE PROGRAMMING EXECUTIVE

Command (Binary)	Data (Hex)	Description
Step 1: Exit Reset vector and erase executive memory.		
0000	000000	NOP
0000	000000	NOP
0000	040100	GOTO 0x100
0000	000000	NOP
Step 2: Initialize the NVMCON to erase executive memory.		
0000	24072A	MOV #0x4072, W10
0000	883B0A	MOV W10, NVMCON
Step 3: Unlock the NVMCON for programming.		
0000	200558	MOV #0x55, W8
0000	883B38	MOV W8, NVMKEY
0000	200AA9	MOV #0xAA, W9
0000	883B39	MOV W9, NVMKEY
Step 4: Initiate the erase cycle.		
0000	A8E761	BSET NVMCON, #15
0000	000000	NOP
0000	000000	NOP
—	—	Externally time 2 msec
0000	A9E761	BCLR NVMCON, #15
0000	000000	NOP
0000	000000	NOP
Step 5: Initialize the NVMCON to program 32 instruction words.		
0000	24001A	MOV #0x4001, W10
0000	883B0A	MOV W10, NVMCON
Step 6: Initialize TBLPAG and the write pointer (W7)		
0000	200800	MOV #0x80, W0
0000	880190	MOV W0, TBLPAG
0000	EB0380	CLR W7
0000	000000	NOP
0000	000000	NOP
Step 7: Load W0:W5 with the next 4 words of packed programming executive code and initialize W6 for programming. Programming starts from the base of executive memory (0x800000) using W6 as a read pointer and W7 as a write pointer.		
0000	2<LSW0>0	MOV #<LSW0>, W0
0000	2<MSB1:MSB0>1	MOV #<MSB1:MSB0>, W1
0000	2<LSW1>2	MOV #<LSW1>, W2
0000	2<LSW2>3	MOV #<LSW2>, W3
0000	2<MSB3:MSB2>4	MOV #<MSB3:MSB2>, W4
0000	2<LSW3>5	MOV #<LSW3>, W5

 © 2004 Microchip Technology Inc.

TABLE 12-1: PROGRAMMING THE PROGRAMMING EXECUTIVE (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 8: Set the read pointer (W6) and load the (next four write) latches.		
0000	EB0300	CLR W6
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 9: Repeat Steps 7-8 eight times to load the write latches for the 32 instructions.		
Step 10: Unlock the NVMCON for programming.		
0000	200558	MOV #0x55, W8
0000	883B38	MOV W8, NVMKEY
0000	200AA9	MOV #0xAA, W9
0000	883B39	MOV W9, NVMKEY
Step 11: Initiate the programming cycle.		
0000	A8E761	BSET NVMCON, #15
0000	000000	NOP
0000	000000	NOP
—	—	Externally time 2 msec
0000	A9E761	BCLR NVMCON, #15
0000	000000	NOP
0000	000000	NOP
Step 12: Reset the device internal PC.		
0000	040100	GOTO 0x100
0000	000000	NOP
Step 13: Repeat Steps 7-12 until all 23 rows of executive memory are programmed.		

12.2 Programming Verification

After the programming executive has been programmed to executive memory using STDP, it must be verified. Verification is performed by reading out the contents of executive memory and comparing it with the image of the programming executive stored in the programmer.

Reading the contents of executive memory can be performed using the same technique described in **Section 11.11 "Reading Code Memory"**. A procedure for reading executive memory is shown in Table 12-2. Note that in Step 2 the TBLPAG register is set to 0x80 such that executive memory may be read.

TABLE 12-2: READING EXECUTIVE MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	000000	NOP
0000	040100	GOTO 0x100
0000	000000	NOP
Step 2: Initialize TBLPAG and the read pointer (W6) for TBLRD instruction.		
0000	200800	MOV #0x80, W0
0000	880190	MOV W0, TBLPAG
0000	EB0380	CLR W6
Step 3: Initialize the write pointer (W7), and store the next four locations of executive memory to W0:W5.		
0000	EB0380	CLR W7
0000	BA1B96	TBLRDL [W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BADBB6	TBLRDH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BADBD6	TBLRDH.B [++W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BA1BB6	TBLRDL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BA1B96	TBLRDL [W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BADBB6	TBLRDH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BADBD6	TBLRDH.B [++W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BA1BB6	TBLRDL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP

 © 2004 Microchip Technology Inc.

TABLE 12-2: READING EXECUTIVE MEMORY (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 4: Output W0:W5 using the VISI register and REGOUT command.		
0000	883C20	MOV W0, VISI
0000	000000	NOP
0001	—	Clock out contents of VISI register
0000	883C21	MOV W1, VISI
0000	000000	NOP
0001	—	Clock out contents of VISI register
0000	883C22	MOV W2, VISI
0000	000000	NOP
0001	—	Clock out contents of VISI register
0000	883C23	MOV W3, VISI
0000	000000	NOP
0001	—	Clock out contents of VISI register
0000	883C24	MOV W4, VISI
0000	000000	NOP
0001	—	Clock out contents of VISI register
0000	883C25	MOV W5, VISI
0000	000000	NOP
0001	—	Clock out contents of VISI register
Step 5: Reset the device internal PC.		
0000	040100	GOTO 0x100
0000	000000	NOP
Step 6: Repeat Steps 3-5 until all 736 instruction words of executive memory are read.		

13.0 AC/DC CHARACTERISTICS AND TIMING REQUIREMENTS

TABLE 13-1: AC/DC CHARACTERISTICS

AC/DC Characteristics			Standard Operating Conditions (unless otherwise stated) Operating Temperature: 25°C is recommended			
Param. No.	**Sym**	**Characteristic**	**Min**	**Max**	**Units**	**Conditions**
D110	VIHH	High Programming Voltage on $\overline{MCLR}$/ VPP	9.00	13.25	V	
D112	IPP	Programming Current on $\overline{MCLR}$/VPP	—	300	μA	
D113	IDDP	Supply Current during programming	—	30	mA	Row erase Program memory
			—	30	mA	Row erase Data EEPROM
			—	30	mA	Bulk erase
D001	VDD	Supply voltage	2.5	5.5	V	
D002	VDDBULK	Supply voltage for bulk erase programming	4.5	5.5	V	
D031	VIL	Input Low Voltage	VSS	0.2 VSS	V	
D041	VIH	Input High Voltage	0.8 VDD	VDD	V	
D080	VOL	Output Low Voltage	—	0.6	V	IOL = 8.5 mA
D090	VOH	Output High Voltage	VDD - 0.7	—	V	IOH = -3.0 mA
D012	CIO	Capacitive Loading on I/O Pin (PGD)	—	50	pF	To meet AC specifications
P1	TSCLK	Serial Clock (PGC) period	50	—	ns	STDP mode
			1	—	μs	ICSP mode
P1a	TSCLKL	Serial Clock (PGC) low time	20	—	ns	STDP mode
			400	—	ns	ICSP mode
P1b	TSCLKH	Serial Clock (PGC) high time	20	—	ns	STDP mode
			400	—	ns	ICSP mode
P2	TSET1	Input Data Setup Timer to PGC ↓	15	—	ns	
P3	THLD1	Input Data Hold Time from PGC ↓	15	—	ns	
P4	T$\overline{dly}$1	Delay between 4-bit command and command operand	20	—	ns	
P4a	TDLY1a	Delay between 4-bit command operand and next 4-bit command	20	—	ns	
P5	TDLY2	Delay between last PGC ↓ of command to first PGC ↑ of VISI output	20	—	ns	
P6	TSET2	VDD ↑ setup time to $\overline{MCLR}$/VPP	100	—	ns	
P7	THLD2	Input data hold time from $\overline{MCLR}$/VPP ↑	2	—	μs	STDP mode
			5	—	ms	ICSP mode
P8	TDLY3	Delay between last PGC ↓ of command word to PGD driven ↑ by programming executive	20	—	μs	
P9a	TDLY4	Programming Executive Command processing time	10	—	μs	

© 2004 Microchip Technology Inc.

TABLE 13-1: AC/DC CHARACTERISTICS (CONTINUED)

AC/DC Characteristics			Standard Operating Conditions (unless otherwise stated) Operating Temperature: 25°C is recommended			
Param. No.	**Sym**	**Characteristic**	**Min**	**Max**	**Units**	**Conditions**
P9b	TDLY5	Delay between PGD ↓ by programming executive to PGD released by programming executive	15	—	μs	
P10	TDLY6	Delay between PGD released by programming executive to first PGC ↑ of response	5	—	μs	
P11	TDLY7	Delay between clocking out response words	10	—	μs	
P12	TPROG	Programming cycle time	1	2	ms	STDP mode
P13	TERA	Erase cycle time	1	2	ms	STDP mode

APPENDIX A: DEVICE SPECIFIC INFORMATION

A.1 Checksum Computation

The checksum computation is described in **Section 6.7 "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 also the configuration registers are erased. However, when code protection is enabled, the value of the FGS register is assumed to be 0x5.

TABLE A-1: CHECKSUM COMPUTATION

Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAAAAAA at 0x0 and Last Code Address
dsPIC30F2010	Disabled	CFGB1+SUM(0:001FFF)	0xD417	0xD219
	Enabled	CFGB1	0x0415	0x0415
dsPIC30F2011	Disabled	CFGB2+SUM(0:001FFF)	0xD42B	0xD22D
	Enabled	CFGB2	0x0429	0x0429
dsPIC30F2012	Disabled	CFGB2+SUM(0:001FFF)	0xD42B	0xD22D
	Enabled	CFGB2	0x0429	0x0429
dsPIC30F3010	Disabled	CFGB2+SUM(0:003FFF)	0xA42B	0xA22D
	Enabled	CFGB2	0x0429	0x0429
dsPIC30F3011	Disabled	CFGB2+SUM(0:003FFF)	0xA42B	0xA22D
	Enabled	CFGB2	0x0429	0x0429
dsPIC30F3012	Disabled	CFGB2+SUM(0:003FFF)	0xA42B	0xA22D
	Enabled	CFGB2	0x0429	0x0429
dsPIC30F3013	Disabled	CFGB2+SUM(0:003FFF)	0xA42B	0xA22D
	Enabled	CFGB2	0x0429	0x0429
dsPIC30F3014	Disabled	CFGB2+SUM(0:003FFF)	0xA42B	0xA22D
	Enabled	CFGB2	0x0429	0x0429

Item Description:

SUM(a:b) = Byte sum of locations a to b inclusive (all 3 bytes of code memory)

CFGB1 **Configuration Block 1 (masked)** = Byte sum of ((FOSC&0xC30F) + (FWDT&0x803F) + (FBORPOR&0x87B3) + (RESERVED1&0x310F) + (RESERVED2&0x330F) + (FGS&0x0007) + (FICD&0xC003))

CFGB2 **Configuration Block 2 (masked)** = Byte sum of ((FOSC&0xC71F) + (FWDT&0x803F) + (FBORPOR&0x87B3) + (RESERVED1&0x310F) + (RESERVED2&0x330F) + (FGS&0x0007) + (FICD&0xC003))

© 2004 Microchip Technology Inc.

TABLE A-1: CHECKSUM COMPUTATION (CONTINUED)

Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAAAAAA at 0x0 and Last Code Address
dsPIC30F4011	Disabled	CFGB1+SUM(0:007FFF)	0x4417	0x4219
	Enabled	CFGB1	0x0415	0x0415
dsPIC30F4012	Disabled	CFGB1+SUM(0:007FFF)	0x4417	0x4219
	Enabled	CFGB1	0x0415	0x0415
dsPIC30F4013	Disabled	CFGB2+SUM(0:007FFF)	0x442B	0x422D
	Enabled	CFGB2	0x0429	0x0429
dsPIC30F5011	Disabled	CFGB1+SUM(0:00AFFF)	0xFC17	0xFA19
	Enabled	CFGB1	0x0415	0x0415
dsPIC30F5013	Disabled	CFGB1+SUM(0:00AFFF)	0xFC17	0xFA19
	Enabled	CFGB1	0x0415	0x0415
dsPIC30F5015	Disabled	CFGB2+SUM(0:00AFFF)	0xFC2B	0xFA2D
	Enabled	CFGB2	0x0429	0x0429
dsPIC30F6010	Disabled	CFGB1+SUM(0:017FFF)	0xC417	0xC219
	Enabled	CFGB1	0x0415	0x0415
dsPIC30F6011	Disabled	CFGB1+SUM(0:015FFF)	0xF417	0xF219
	Enabled	CFGB1	0x0415	0x0415
dsPIC30F6011A	Disabled	CFGB2+SUM(0:015FFF)	0xF42B	0xF22D
	Enabled	CFGB2	0x0429	0x0429
dsPIC30F6012	Disabled	CFGB1+SUM(0:017FFF)	0xC417	0xC219
	Enabled	CFGB1	0x0415	0x0415
dsPIC30F6012A	Disabled	CFGB2+SUM(0:017FFF)	0xC42B	0xC220
	Enabled	CFGB2	0x0429	0x0429
dsPIC30F6013	Disabled	CFGB1+SUM(0:015FFF)	0xF417	0xF219
	Enabled	CFGB1	0x0415	0x0415
dsPIC30F6013A	Disabled	CFGB2+SUM(0:015FFF)	0xF42B	0xF22D
	Enabled	CFGB2	0x0429	0x0429
dsPIC30F6014	Disabled	CFGB1+SUM(0:017FFF)	0xC417	0xC219
	Enabled	CFGB1	0x0415	0x0415
dsPIC30F6014A	Disabled	CFGB2+SUM(0:017FFF)	0xC42B	0xC22D
	Enabled	CFGB2	0x0429	0x0429

Item Description:

SUM(a:b) = Byte sum of locations a to b inclusive (all 3 bytes of code memory)

CFGB1 **Configuration Block 1 (masked)** = Byte sum of ((FOSC&0xC30F) + (FWDT&0x803F) + (FBORPOR&0x87B3) + (RESERVED1&0x310F) + (RESERVED2&0x330F) + (FGS&0x0007) + (FICD&0xC003))

CFGB2 **Configuration Block 2 (masked)** = Byte sum of ((FOSC&0xC71F) + (FWDT&0x803F) + (FBORPOR&0x87B3) + (RESERVED1&0x310F) + (RESERVED2&0x330F) + (FGS&0x0007) + (FICD&0xC003))

NOTES:

 © 2004 Microchip Technology Inc.

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's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KEELOQ, microID, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, PowerSmart, rfPIC, and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AmpLab, FilterLab, MXDEV, MXLAB, PICMASTER, SEEVAL, SmartSensor 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, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, PICkit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, rfLAB, rfPICDEM, Select Mode, Smart Serial, SmartTel and Total Endurance 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.

© 2004, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.

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

Microchip received ISO/TS-16949:2002 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona and Mountain View, California in October 2003. The Company's quality system processes and procedures are for its PICmicro® 8-bit MCUs, 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.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://support.microchip.com
Web Address:
www.microchip.com

Atlanta
Alpharetta, GA
Tel: 770-640-0034
Fax: 770-640-0307

Boston
Westford, MA
Tel: 978-692-3848
Fax: 978-692-3821

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

San Jose
Mountain View, CA
Tel: 650-215-1444
Fax: 650-961-0286

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8676-6200
Fax: 86-28-8676-6599

China - Fuzhou
Tel: 86-591-8750-3506
Fax: 86-591-8750-3521

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Shunde
Tel: 86-757-2839-5507
Fax: 86-757-2839-5571

China - Qingdao
Tel: 86-532-502-7355
Fax: 86-532-502-7205

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-2229-0061
Fax: 91-80-2229-0062

India - New Delhi
Tel: 91-11-5160-8631
Fax: 91-11-5160-8632

Japan - Kanagawa
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Taiwan - Hsinchu
Tel: 886-3-572-9526
Fax: 886-3-572-6459

EUROPE

Austria - Weis
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

Denmark - Ballerup
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Massy
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Ismaning
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

England - Berkshire
Tel: 44-118-921-5869
Fax: 44-118-921-5820

10/20/04

© 2004 Microchip Technology Inc.