

## dsPIC33F/PIC24H Flash Programming Specification

### 1.0 DEVICE OVERVIEW

This document defines the programming specification for the dsPIC33F 16-bit Digital Signal Controller (DSC) and PIC24H 16-bit Microcontroller (MCU) families. This programming specification is required only for those developing programming support for the dsPIC33F/PIC24H family. Customers using only one of these devices should use development tools that already provide support for device programming.

Topics covered include:

1.0	Device Overview .....	1
2.0	Programming Overview of the dsPIC33F/PIC24H .....	1
3.0	Device Programming – Enhanced ICSP .....	8
4.0	The Programming Executive .....	19
5.0	Device Programming – ICSP .....	28
6.0	Programming the Programming Executive to Memory .....	45
7.0	Device ID .....	50
8.0	AC/DC Characteristics and Timing Requirements .....	54
	Appendix A: Hex File Format .....	57
	Appendix B: Device ID Register Silicon Errata Addendum .....	58
	Appendix C: Diagnostic and Calibration Registers .....	59
	Appendix D: Checksum Computation .....	61
	Appendix E: Revision History .....	74

### 2.0 PROGRAMMING OVERVIEW OF THE dsPIC33F/PIC24H

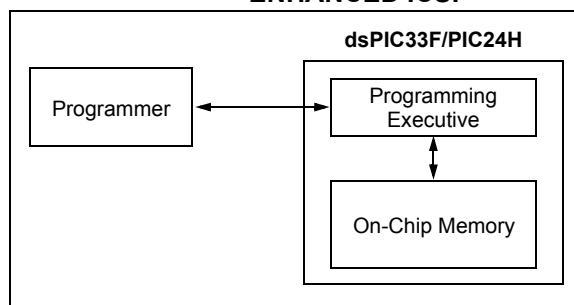
There are two methods of programming the dsPIC33F/PIC24H family of devices discussed in this programming specification. They are:

- In-Circuit Serial Programming™ (ICSP™) programming capability
- Enhanced In-Circuit Serial Programming

The ICSP programming method is the most direct method to program the device; however, it is also the slower of the two methods. It provides native, low-level programming capability to erase, program and verify the chip.

The Enhanced ICSP protocol uses a faster method that takes advantage of the programming executive, as illustrated in [Figure 2-1](#). 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 dsPIC33F/PIC24H Programming Specification devices without having to deal with the low-level programming protocols of the chip.

**FIGURE 2-1: PROGRAMMING SYSTEM OVERVIEW FOR ENHANCED ICSP™**



This specification is divided into major sections that describe the programming methods independently. [Section 3.0 “Device Programming – Enhanced ICSP”](#) describes the Enhanced ICSP method. [Section 5.0 “Device Programming – ICSP”](#) describes the ICSP method.

### 2.1 Power Requirements

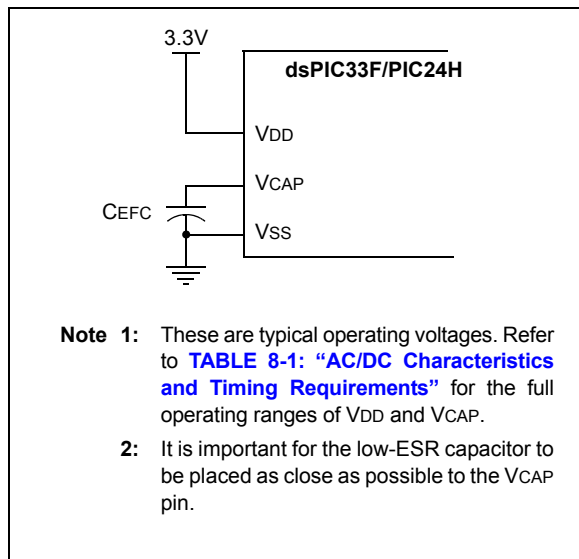
All devices in the dsPIC33F/PIC24H family are dual voltage supply designs: one supply for the core and another for the peripherals and I/O pins. A regulator is provided on-chip to alleviate the need for two external voltage supplies.

All of the dsPIC33F/PIC24H devices power their core digital logic at a nominal 2.5V. To simplify system design, all devices in the dsPIC33F/PIC24H Programming Specification family incorporate an on-chip regulator that allows the device to run its core logic from VDD.

The regulator provides power to the core from the other VDD pins. A low-ESR capacitor (such as tantalum) must be connected to the VCAP pin ([Figure 2-2](#)). This helps to maintain the stability of the regulator. The specifications for core voltage and capacitance are listed in [Section 8.0 “AC/DC Characteristics and Timing Requirements”](#).

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**FIGURE 2-2: CONNECTIONS FOR THE ON-CHIP REGULATOR**



## 2.2 Program Memory Write/Erase Requirements

The program Flash memory on the dsPIC33F/PIC24H has a specific write/erase requirement that must be adhered to, for proper device operation. The rule is that any given word in memory must not be written without first erasing the page in which it is located. Thus, the easiest way to conform to this rule is to write all the data in a programming block within one write cycle. The programming methods specified in this document comply with this requirement.

**Note:** A program memory word can be programmed twice before an erase, but only if (a) the same data is used in both program operations or (b) bits containing '1' are set to '0' but no '0' is set to '1'.

## 2.3 Pins Used During Programming

The pins that are used for programming are listed in [Table 2-1](#).

**Note:** Refer to the specific device data sheet for complete pin diagrams of the dsPIC33F/PIC24H devices.

**TABLE 2-1: PINS USED DURING PROGRAMMING**

Pin Name	During Programming		
	Pin Name	Pin Type	Pin Description
MCLR	MCLR	P	Programming Enable
VDD and AVDD <sup>(1)</sup>	VDD	P	Power Supply
VSS and AVSS <sup>(1)</sup>	VSS	P	Ground
VCAP	VCAP	P	CPU Logic Filter Capacitor Connection
PGEC1	PGEC1	I	Primary Programming Pin Pair: Serial Clock
PGED1	PGED1	I/O	Primary Programming Pin Pair: Serial Data
PGEC2	PGEC2	I	Secondary Programming Pin Pair: Serial Clock
PGED2	PGED2	I/O	Secondary Programming Pin Pair: Serial Data
PGEC3	PGEC3	I	Tertiary Programming Pin Pair: Serial Clock
PGED3	PGED3	I/O	Tertiary Programming Pin Pair: Serial Data

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

**Note 1:** All power supply and ground pins must be connected, including analog supplies (AVDD) and ground (AVSS).

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 2.4 Memory Map

The program memory map extends from 0x0 to 0xFFFFFE. Code storage is located at the base of the memory map and supports up to 88K instructions (about 256 Kbytes). Table 2-2 shows the program memory size and number of erase and program blocks present in each device variant. Each erase block or page contains 512 instructions and each program block or row, contains 64 instructions.

Locations 0x800000 through 0x800FFE are reserved for executive code memory. This region stores the programming executive and the debugging executive. The programming executive is used for

device programming and the debug executive is used for in-circuit debugging. This region of memory cannot be used to store user code.

Locations 0xF80000 through 0xF80017 are reserved for the device Configuration registers.

Locations 0xFF0000 and 0xFF0002 are reserved for the Device ID Word registers. These bits can be used by the programmer to identify which device type is being programmed. They are described in Section 7.0 “Device ID”. The Device ID registers read out normally, even after code protection is applied.

Figure 2-3 illustrates the memory map for the dsPIC33F/PIC24H family variants.

TABLE 2-2: CODE MEMORY SIZE

dsPIC33F/PIC24H Device	User Memory Address Limit (Instruction Words)	Write Blocks	Erase Blocks	Executive Memory Address Limit (Instruction Words)
dsPIC33FJ06GS101	0x000FFE (2K)	32	4	0x8007FE (1K)
dsPIC33FJ06GS102	0x000FFE (2K)	32	4	0x8007FE (1K)
dsPIC33FJ06GS202	0x000FFE (2K)	32	4	0x8007FE (1K)
dsPIC33FJ16GS402	0x002BFE (6K)	88	11	0x8007FE (1K)
dsPIC33FJ16GS404	0x002BFE (6K)	88	11	0x8007FE (1K)
dsPIC33FJ16GS502	0x002BFE (6K)	88	11	0x8007FE (1K)
dsPIC33FJ16GS504	0x002BFE (6K)	88	11	0x8007FE (1K)
dsPIC33FJ12GP201	0x001FFE (4K)	64	8	0x8007FE (1K)
dsPIC33FJ12GP202	0x001FFE (4K)	64	8	0x8007FE (1K)
dsPIC33FJ16GP304	0x002BFE (6K)	88	11	0x800FFE (2K)
dsPIC33FJ32GP202	0x0057FE (11K)	176	22	0x800FFE (2K)
dsPIC33FJ32GP204	0x0057FE (11K)	176	22	0x800FFE (2K)
dsPIC33FJ32GP302	0x0057FE (11K)	176	22	0x800FFE (2K)
dsPIC33FJ32GP304	0x0057FE (11K)	176	22	0x800FFE (2K)
dsPIC33FJ64GP202	0x00ABFE (22K)	344	43	0x800FFE (2K)
dsPIC33FJ64GP204	0x00ABFE (22K)	344	43	0x800FFE (2K)
dsPIC33FJ64GP206	0x00ABFE (22K)	344	43	0x800FFE (2K)
dsPIC33FJ64GP306	0x00ABFE (22K)	344	43	0x800FFE (2K)
dsPIC33FJ64GP310	0x00ABFE (22K)	344	43	0x800FFE (2K)
dsPIC33FJ64GP706	0x00ABFE (22K)	344	43	0x800FFE (2K)
dsPIC33FJ64GP708	0x00ABFE (22K)	344	43	0x800FFE (2K)
dsPIC33FJ64GP710	0x00ABFE (22K)	344	43	0x800FFE (2K)
dsPIC33FJ64GP802	0x00ABFE (22K)	344	43	0x800FFE (2K)
dsPIC33FJ64GP804	0x00ABFE (22K)	344	43	0x800FFE (2K)
dsPIC33FJ128GP202	0x0157FE (44K)	688	86	0x800FFE (2K)
dsPIC33FJ128GP204	0x0157FE (44K)	688	86	0x800FFE (2K)
dsPIC33FJ128GP206	0x0157FE (44K)	688	86	0x800FFE (2K)
dsPIC33FJ128GP306	0x0157FE (44K)	688	86	0x800FFE (2K)
dsPIC33FJ128GP310	0x0157FE (44K)	688	86	0x800FFE (2K)
dsPIC33FJ128GP706	0x0157FE (44K)	688	86	0x800FFE (2K)
dsPIC33FJ128GP708	0x0157FE (44K)	688	86	0x800FFE (2K)

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 2-2: CODE MEMORY SIZE (CONTINUED)**

dsPIC33F/PIC24H Device	User Memory Address Limit (Instruction Words)	Write Blocks	Erase Blocks	Executive Memory Address Limit (Instruction Words)
dsPIC33FJ128GP710	0x0157FE (44K)	688	86	0x800FFE (2K)
dsPIC33FJ128GP802	0x0157FE (44K)	688	86	0x800FFE (2K)
dsPIC33FJ128GP804	0x0157FE (44K)	688	86	0x800FFE (2K)
dsPIC33FJ256GP506	0x02ABFE (88K)	1368	171	0x800FFE (2K)
dsPIC33FJ256GP510	0x02ABFE (88K)	1368	171	0x800FFE (2K)
dsPIC33FJ256GP710	0x02ABFE (88K)	1368	171	0x800FFE (2K)
dsPIC33FJ12MC201	0x001FFE (4K)	64	8	0x8007FE (1K)
dsPIC33FJ12MC202	0x001FFE (4K)	64	8	0x8007FE (1K)
dsPIC33FJ16MC304	0x002BFE (6K)	88	11	0x800FFE (2K)
dsPIC33FJ32MC202	0x0057FE (11K)	176	22	0x800FFE (2K)
dsPIC33FJ32MC204	0x0057FE (11K)	176	22	0x800FFE (2K)
dsPIC33FJ32MC302	0x0057FE (11K)	176	22	0x800FFE (2K)
dsPIC33FJ32MC304	0x0057FE (11K)	176	22	0x800FFE (2K)
dsPIC33FJ64MC202	0x00ABFE (22K)	344	43	0x800FFE (2K)
dsPIC33FJ64MC204	0x00ABFE (22K)	344	43	0x800FFE (2K)
dsPIC33FJ64MC506	0x00ABFE (22K)	344	43	0x800FFE (2K)
dsPIC33FJ64MC508	0x00ABFE (22K)	344	43	0x800FFE (2K)
dsPIC33FJ64MC510	0x00ABFE (22K)	344	43	0x800FFE (2K)
dsPIC33FJ64MC706	0x00ABFE (22K)	344	43	0x800FFE (2K)
dsPIC33FJ64MC710	0x00ABFE (22K)	344	43	0x800FFE (2K)
dsPIC33FJ64MC802	0x00ABFE (22K)	344	43	0x800FFE (2K)
dsPIC33FJ64MC804	0x00ABFE (22K)	344	43	0x800FFE (2K)
dsPIC33FJ128MC202	0x0157FE (44K)	688	86	0x800FFE (2K)
dsPIC33FJ128MC204	0x0157FE (44K)	688	86	0x800FFE (2K)
dsPIC33FJ128MC506	0x0157FE (44K)	688	86	0x800FFE (2K)
dsPIC33FJ128MC510	0x0157FE (44K)	688	86	0x800FFE (2K)
dsPIC33FJ128MC706	0x0157FE (44K)	688	86	0x800FFE (2K)
dsPIC33FJ128MC708	0x0157FE (44K)	688	86	0x800FFE (2K)
dsPIC33FJ128MC710	0x0157FE (44K)	688	86	0x800FFE (2K)
dsPIC33FJ128MC802	0x0157FE (44K)	688	86	0x800FFE (2K)
dsPIC33FJ128MC804	0x0157FE (44K)	688	86	0x800FFE (2K)
dsPIC33FJ256MC510	0x02ABFE (88K)	1368	171	0x800FFE (2K)
dsPIC33FJ256MC710	0x02ABFE (88K)	1368	171	0x800FFE (2K)
PIC24HJ12GP201	0x001FFE (4K)	64	8	0x8007FE (1K)
PIC24HJ12GP202	0x001FFE (4K)	64	8	0x8007FE (1K)
PIC24HJ16GP304	0x002BFE (6K)	88	11	0x800FFE (2K)
PIC24HJ32GP202	0x0057FE (11K)	176	22	0x800FFE (2K)
PIC24HJ32GP204	0x0057FE (11K)	176	22	0x800FFE (2K)
PIC24HJ32GP302	0x0057FE (11K)	176	22	0x800FFE (2K)
PIC24HJ32GP304	0x0057FE (11K)	176	22	0x800FFE (2K)
PIC24HJ64GP202	0x00ABFE (22K)	344	43	0x800FFE (2K)
PIC24HJ64GP204	0x00ABFE (22K)	344	43	0x800FFE (2K)
PIC24HJ64GP206	0x00ABFE (22K)	344	43	0x800FFE (2K)

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 2-2: CODE MEMORY SIZE (CONTINUED)**

dsPIC33F/PIC24H Device	User Memory Address Limit (Instruction Words)	Write Blocks	Erase Blocks	Executive Memory Address Limit (Instruction Words)
PIC24HJ64GP210	0x00ABFE (22K)	344	43	0x800FFE (2K)
PIC24HJ64GP502	0x00ABFE (22K)	344	43	0x800FFE (2K)
PIC24HJ64GP504	0x00ABFE (22K)	344	43	0x800FFE (2K)
PIC24HJ64GP506	0x00ABFE (22K)	344	43	0x800FFE (2K)
PIC24HJ64GP510	0x00ABFE (22K)	344	43	0x800FFE (2K)
PIC24HJ128GP202	0x0157FE (44K)	688	86	0x800FFE (2K)
PIC24HJ128GP204	0x0157FE (44K)	688	86	0x800FFE (2K)
PIC24HJ128GP206	0x0157FE (44K)	688	86	0x800FFE (2K)
PIC24HJ128GP210	0x0157FE (44K)	688	86	0x800FFE (2K)
PIC24HJ128GP306	0x0157FE (44K)	688	86	0x800FFE (2K)
PIC24HJ128GP310	0x0157FE (44K)	688	86	0x800FFE (2K)
PIC24HJ128GP502	0x0157FE (44K)	688	86	0x800FFE (2K)
PIC24HJ128GP504	0x0157FE (44K)	688	86	0x800FFE (2K)
PIC24HJ128GP506	0x0157FE (44K)	688	86	0x800FFE (2K)
PIC24HJ128GP510	0x0157FE (44K)	688	86	0x800FFE (2K)
PIC24HJ256GP206	0x02ABFE (88K)	1368	171	0x800FFE (2K)
PIC24HJ256GP210	0x02ABFE (88K)	1368	171	0x800FFE (2K)
PIC24HJ256GP610	0x02ABFE (88K)	1368	171	0x800FFE (2K)
dsPIC33FJ64GP206A	0x00ABFE (22K)	344	43	0x800FFE (2k)
dsPIC33FJ64GP306A	0x00ABFE (22K)	344	43	0x800FFE (2k)
dsPIC33FJ64GP310A	0x00ABFE (22K)	344	43	0x800FFE (2k)
dsPIC33FJ64GP706A	0x00ABFE (22K)	344	43	0x800FFE (2k)
dsPIC33FJ64GP708A	0x00ABFE (22K)	344	43	0x800FFE (2k)
dsPIC33FJ64GP710A	0x00ABFE (22K)	344	43	0x800FFE (2k)
dsPIC33FJ64MC506A	0x00ABFE (22K)	344	43	0x800FFE (2k)
dsPIC33FJ64MC508A	0x00ABFE (22K)	344	43	0x800FFE (2k)
dsPIC33FJ64MC510A	0x00ABFE (22K)	344	43	0x800FFE (2k)
dsPIC33FJ64MC706A	0x00ABFE (22K)	344	43	0x800FFE (2k)
dsPIC33FJ64MC710A	0x00ABFE (22K)	344	43	0x800FFE (2k)
PIC24HJ64GP206A	0x00ABFE (22K)	344	43	0x800FFE (2k)
PIC24HJ64GP210A	0x00ABFE (22K)	344	43	0x800FFE (2k)
PIC24HJ64GP506A	0x00ABFE (22K)	344	43	0x800FFE (2k)
PIC24HJ64GP510A	0x00ABFE (22K)	344	43	0x800FFE (2k)
dsPIC33FJ128GP206A	0x0157FE (44K)	688	86	0x800FFE (2k)
dsPIC33FJ128GP306A	0x0157FE (44K)	688	86	0x800FFE (2k)
dsPIC33FJ128GP310A	0x0157FE (44K)	688	86	0x800FFE (2k)
dsPIC33FJ128GP706A	0x0157FE (44K)	688	86	0x800FFE (2k)
dsPIC33FJ128GP708A	0x0157FE (44K)	688	86	0x800FFE (2k)
dsPIC33FJ128GP710A	0x0157FE (44K)	688	86	0x800FFE (2k)
dsPIC33FJ128MC506A	0x0157FE (44K)	688	86	0x800FFE (2k)
dsPIC33FJ128MC510A	0x0157FE (44K)	688	86	0x800FFE (2k)
dsPIC33FJ128MC706A	0x0157FE (44K)	688	86	0x800FFE (2k)
dsPIC33FJ128MC708A	0x0157FE (44K)	688	86	0x800FFE (2k)

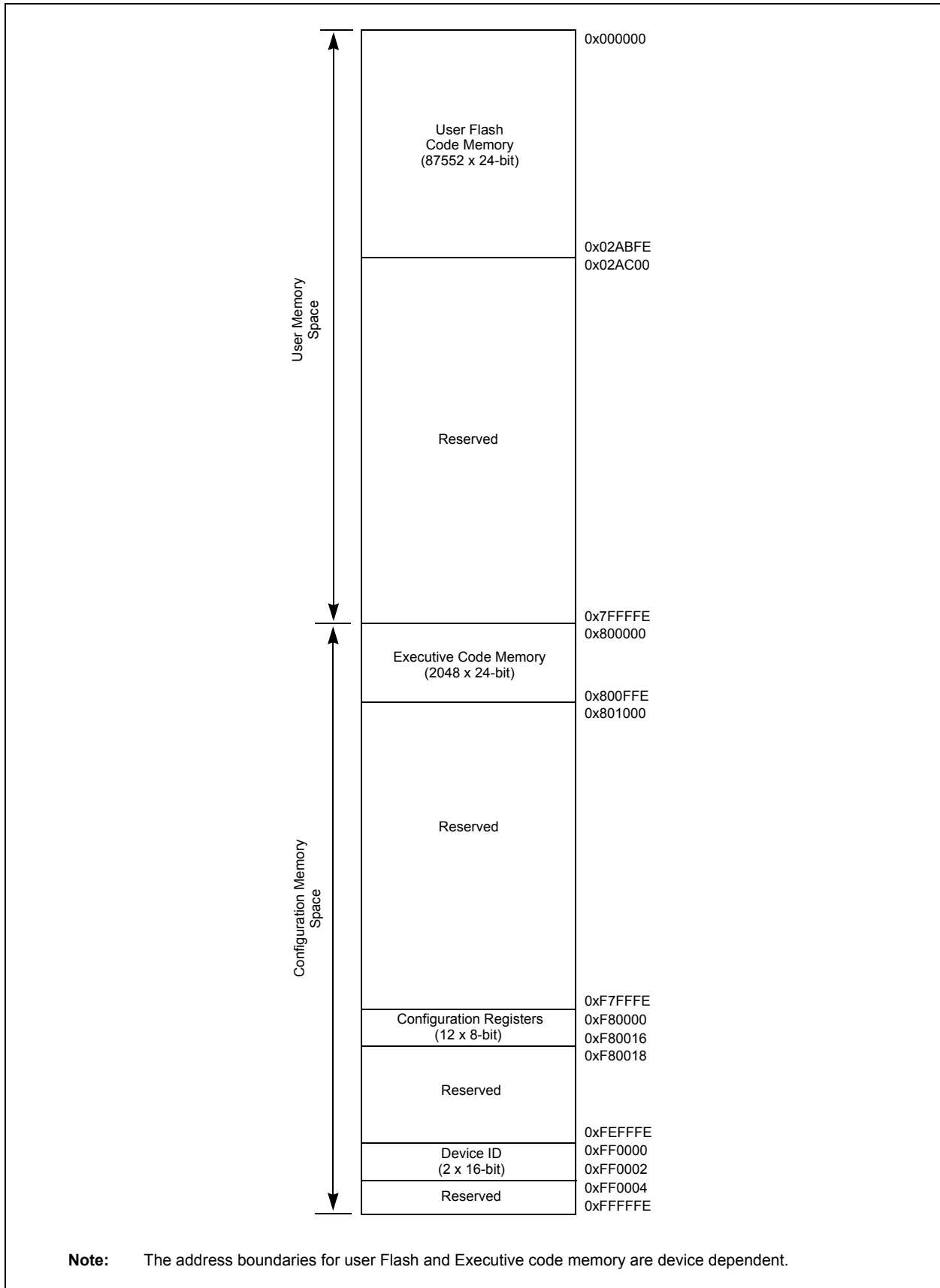
# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 2-2: CODE MEMORY SIZE (CONTINUED)**

dsPIC33F/PIC24H Device	User Memory Address Limit (Instruction Words)	Write Blocks	Erase Blocks	Executive Memory Address Limit (Instruction Words)
dsPIC33FJ128MC710A	0x0157FE (44K)	688	86	0x800FFE (2k)
PIC24HJ128GP206A	0x0157FE (44K)	688	86	0x800FFE (2k)
PIC24HJ128GP210A	0x0157FE (44K)	688	86	0x800FFE (2k)
PIC24HJ128GP306A	0x0157FE (44K)	688	86	0x800FFE (2k)
PIC24HJ128GP310A	0x0157FE (44K)	688	86	0x800FFE (2k)
PIC24HJ128GP506A	0x0157FE (44K)	688	86	0x800FFE (2k)
PIC24HJ128GP510A	0x0157FE (44K)	688	86	0x800FFE (2k)
dsPIC33FJ256GP506A	0x02ABFE (88K)	1368	171	0x800FFE (2k)
dsPIC33FJ256GP510A	0x02ABFE (88K)	1368	171	0x800FFE (2k)
dsPIC33FJ256GP710A	0x02ABFE (88K)	1368	171	0x800FFE (2k)
dsPIC33FJ256MC510A	0x02ABFE (88K)	1368	171	0x800FFE (2k)
dsPIC33FJ256MC710A	0x02ABFE (88K)	1368	171	0x800FFE (2k)
PIC24HJ256GP206A	0x02ABFE (88K)	1368	171	0x800FFE (2k)
PIC24HJ256GP210A	0x02ABFE (88K)	1368	171	0x800FFE (2k)
PIC24HJ256GP610A	0x02ABFE (88K)	1368	171	0x800FFE (2k)
dsPIC33FJ32GS406	0x0057FE (11K)	176	22	0x800FFE (2k)
dsPIC33FJ32GS606	0x0057FE (11K)	176	22	0x800FFE (2k)
dsPIC33FJ32GS608	0x0057FE (11K)	176	22	0x800FFE (2k)
dsPIC33FJ32GS610	0x0057FE (11K)	176	22	0x800FFE (2k)
dsPIC33FJ64GS406	0x00ABFE (22K)	344	43	0x800FFE (2k)
dsPIC33FJ64GS606	0x00ABFE (22K)	344	43	0x800FFE (2k)
dsPIC33FJ64GS608	0x00ABFE (22K)	344	43	0x800FFE (2k)
dsPIC33FJ64GS610	0x00ABFE (22K)	344	43	0x800FFE (2k)

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**FIGURE 2-3: PROGRAM MEMORY MAP**



# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 3.0 DEVICE PROGRAMMING – ENHANCED ICSP

This section discusses programming the device through Enhanced ICSP and the programming executive. The programming executive resides in executive memory (separate from code memory) and is executed when Enhanced ICSP Programming mode is entered. The programming executive provides the mechanism for the programmer (host device) to program and verify the dsPIC33F/PIC24H Programming Specification family devices using a simple command set and communication protocol. There are several basic functions provided by the programming executive:

- Read Memory
- Erase Memory
- Program Memory
- Blank Check
- Read Executive Firmware Revision

The programming executive performs the low-level tasks required for erasing, programming and verifying a device. This allows the programmer to program the device by issuing the appropriate commands and data. [Table 3-1](#) summarizes the commands. A detailed description for each command is provided in [Section 4.2 “Programming Executive Commands”](#).

**TABLE 3-1: COMMAND SET SUMMARY**

Command	Description
SCHECK	Sanity check.
READC	Read Configuration registers or Device ID registers.
READP	Read code memory.
PROGC	Program a Configuration register and verify.
PROGP	Program one row of code memory and verify.
ERASEP	Erase Page command.
CRCP SIGNATURE	Performs CRC on memory.
QBLANK	Query to check whether code memory is blank.
QVER	Query the software version.

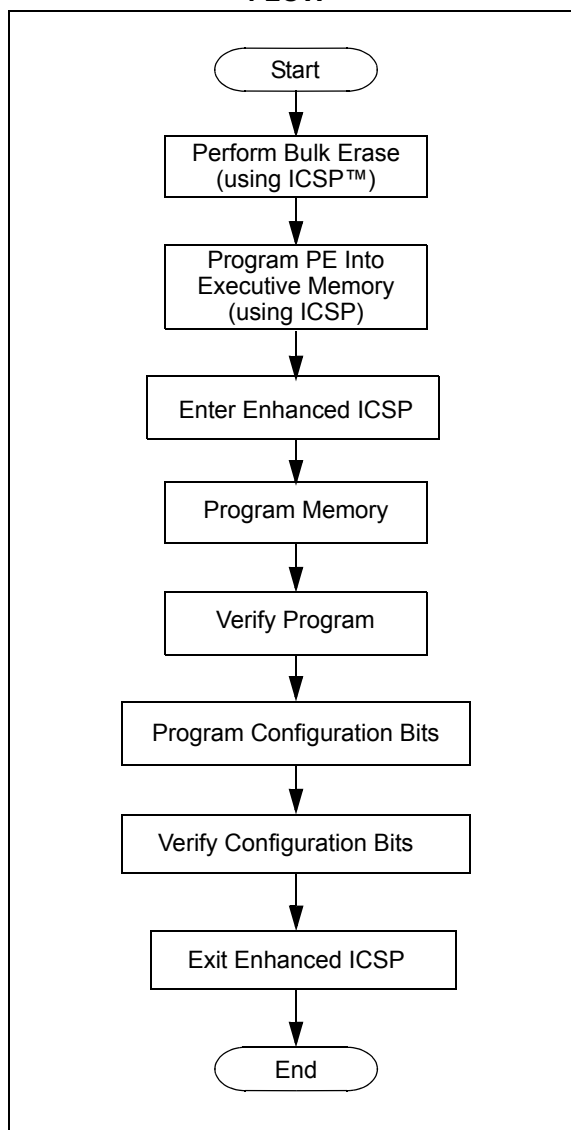
The programming executive uses the device’s data RAM for variable storage and program execution. After the programming executive is run, no assumptions should be made about the contents of data RAM.

## 3.1 Overview of the Programming Process

[Figure 3-1](#) illustrates the high-level overview of the programming process. After entering Enhanced ICSP mode, the programming executive is verified. Next, the device is erased. Then, the code memory is programmed, followed by the nonvolatile device Configuration registers. Code memory (including the Configuration registers) is then verified to ensure that programming was successful.

After the programming executive has been verified in memory (or loaded if not present), the dsPIC33F/PIC24H Programming Specification can be programmed using the command set shown in [Table 3-1](#).

**FIGURE 3-1: HIGH-LEVEL ENHANCED ICSP™ PROGRAMMING FLOW**



# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

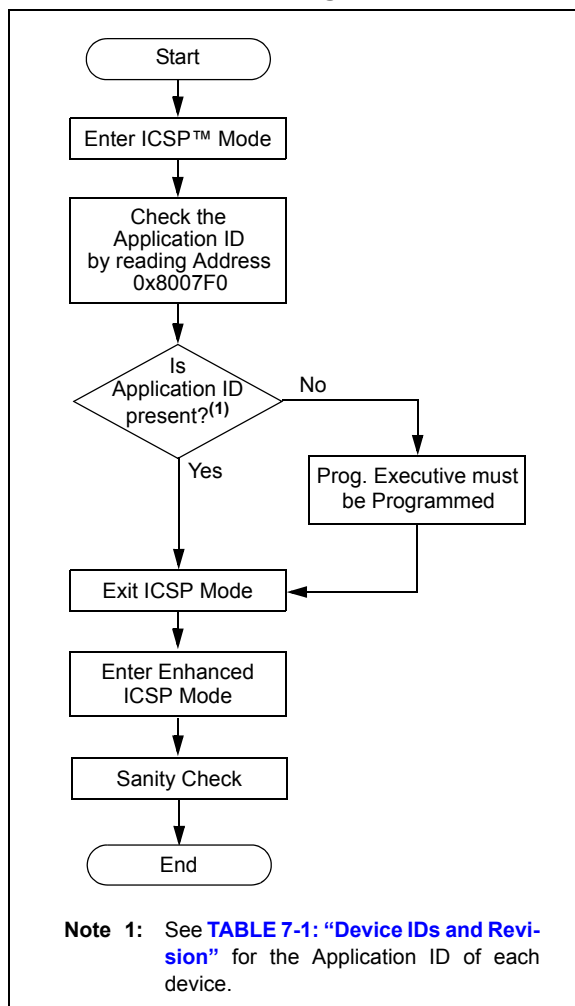
## 3.2 Confirming the Presence of the Programming Executive

Before programming, the programmer must confirm that the programming executive is stored in executive memory. The procedure for this task is illustrated in Figure 3-2.

First, ICSP mode is entered. Then, the unique Application ID Word stored in executive memory is read. If the programming executive is resident, the correct Application ID Word is read and programming can resume as normal. However, if the Application ID Word is not present, the programming executive must be programmed to executive code memory using the method described in Section 6.0 “Programming the Programming Executive to Memory”. See Table 7-1 for the Application ID of each device.

Section 5.0 “Device Programming – ICSP” describes the ICSP programming method. Section 5.11 “Reading the Application ID Word” describes the procedure for reading the Application ID Word in ICSP mode.

**FIGURE 3-2: CONFIRMING PRESENCE OF PROGRAMMING EXECUTIVE**



## 3.3 Entering Enhanced ICSP Mode

As illustrated in Figure 3-3, entering Enhanced ICSP Program/Verify mode requires three steps:

1. The  $\overline{\text{MCLR}}$  pin is briefly driven high then low.
2. A 32-bit key sequence is clocked into PGDx.
3.  $\overline{\text{MCLR}}$  is then driven high within a specified period of time and held.

The programming voltage applied to  $\overline{\text{MCLR}}$  is  $V_{IH}$ , which is essentially  $V_{DD}$  in case of dsPIC33F/PIC24H devices. There is no minimum time requirement for holding at  $V_{IH}$ . After  $V_{IH}$  is removed, an interval of at least P18 must elapse before presenting the key sequence on PGDx.

The key sequence is a specific 32-bit pattern, ‘0100 1101 0100 0011 0100 1000 0101 0000’ (more easily remembered as 0x4D434850 in hexadecimal format). The device will enter Program/Verify mode only if the key sequence is valid. The Most Significant bit (MSb) of the most significant nibble must be shifted in first.

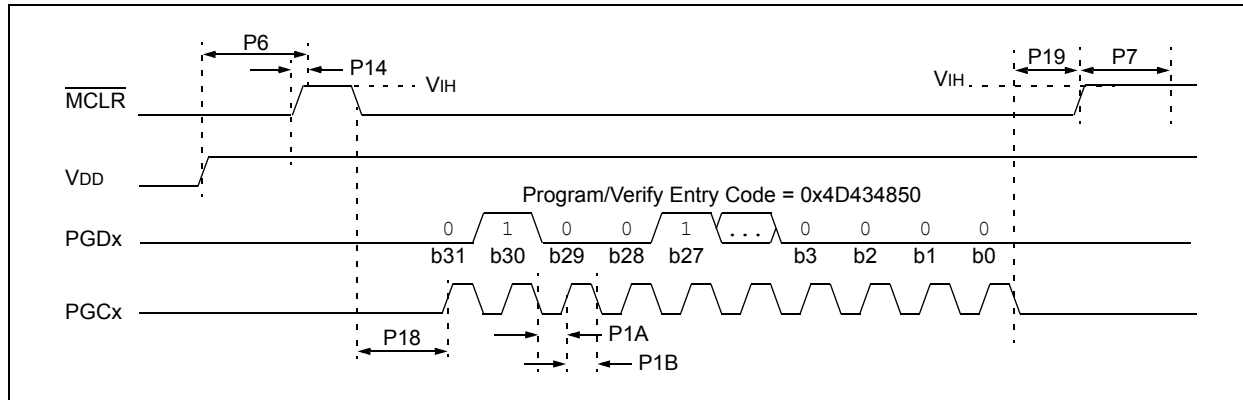
Once the key sequence is complete,  $V_{IH}$  must be applied to  $\overline{\text{MCLR}}$  and held at that level for as long as Program/Verify mode is to be maintained. An interval time of at least P19 and P7 must elapse before presenting data on PGDx. Signals appearing on PGDx before P7 has elapsed will not be interpreted as valid.

On successful entry, the program memory can be accessed and programmed in serial fashion. While in the Program/Verify mode, all unused I/Os are placed in the high-impedance state.

**Note:** When programming a device without Peripheral Pin Select (PPS) and in Enhanced ICSP mode, the SPI output pin (SDOx) may toggle while the device is being programmed.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

FIGURE 3-3: ENTERING ENHANCED ICSP™ MODE



## 3.4 Blank Check

The term “Blank Check” implies verifying that the device has been successfully erased and has no programmed memory locations. A blank or erased memory location is always read as ‘1’.

The Device ID registers (0xFF0000:0xFF0002) can be ignored by the Blank Check since this region stores device information that cannot be erased. The device Configuration registers are also ignored by the Blank Check. 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 is 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 before attempting to program the chip.

## 3.5 Code Memory Programming

### 3.5.1 PROGRAMMING METHODOLOGY

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

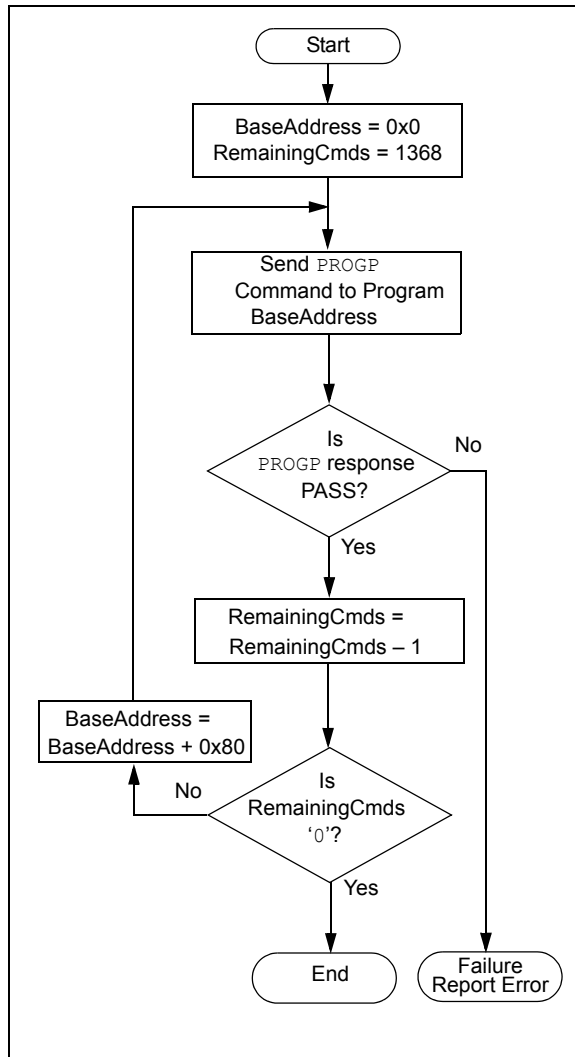
A flowchart for programming code memory is illustrated in Figure 3-4. In this example, all 88K instruction words of a dsPIC33F/PIC24H device are programmed. First, the number of commands to send (called ‘RemainingCmds’ in the flowchart) is set to 1368 and the destination address (called ‘BaseAddress’) is set to ‘0’. Next, one write block in the device is programmed with a `PROGP` command. Each `PROGP` command contains data for one row of code memory of the dsPIC33F/PIC24H. After the first command is processed successfully, ‘RemainingCmds’ is decremented by ‘1’ and compared with ‘0’. Since there are more `PROGP` commands to send, ‘BaseAddress’ is incremented by 0x80 to point to the next row of memory.

On the second `PROGP` command, the second row is programmed. This process is repeated until the entire device is programmed.

**Note:** If a bootloader needs to be programmed, the bootloader code must not be programmed into the first page of code memory. For example, if a bootloader located at address 0x200 attempts to erase the first page, it would inadvertently erase itself. Instead, program the bootloader into the second page (e.g., 0x400).

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**FIGURE 3-4: FLOWCHART FOR PROGRAMMING CODE MEMORY**



## 3.5.2 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.

## 3.5.3 CHECKSUM COMPUTATION

Only the Configuration registers are included in the checksum computation. The Device ID and Unit ID are not included in the checksum computation.

**TABLE D-1: "CHECKSUM COMPUTATION"** shows how this 16-bit computation can be made for each dsPIC33F and PIC24H device. Computations for read code protection are shown both enabled and disabled. The checksum values shown here assume that the Configuration registers are also erased. However, when code protection is enabled, the value of the FGS register is assumed to be 0x5.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 3.6 Configuration Bits Programming

### 3.6.1 OVERVIEW

The dsPIC33F/PIC24H devices have Configuration bits stored in twelve 8-bit Configuration registers, aligned on even configuration memory address boundaries. These bits can be set or cleared to select various device configurations. There are three types of Configuration bits: system operation bits, code-protect bits and unit ID bits. The system operation bits determine the power-on settings for system level

components, such as oscillator and Watchdog Timer. The code-protect bits prevent program memory from being read and written.

The register descriptions for the FBS, FSS, FGS, FOSCSEL, FOSC, FWDT, FPOR and FICD Configuration registers are shown in [Table 3-2](#).

**Note 1:** If any of the code-protect bits in FBS, FSS or FGS is clear, the entire device must be erased before it can be reprogrammed.

**TABLE 3-2: dsPIC33F/PIC24H CONFIGURATION BITS DESCRIPTION**

Bit Field	Register	Description
RBS<1:0>	FBS	<p><b>Boot Segment Data RAM Code Protection</b></p> <p>11 = No RAM is reserved for Boot Segment            10 = Small-Sized Boot RAM            [128 bytes of RAM are reserved for Boot Segment]            01 = Medium-Sized Boot RAM            [256 bytes of RAM are reserved for Boot Segment]            00 = Large-Sized Boot RAM            [1024 bytes of RAM are reserved for Boot Segment]</p>
BSS<2:0>	FBS	<p><b>Boot Segment Program Memory Code Protection</b></p> <p>111 = No Boot Segment            110 = Standard security, Small-sized Boot Program Flash            [Boot Segment ends at 0x0003FF in dsPIC33FJ06GS101/102/202, dsPIC33FJ16GS402/404/502/504, dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202 and PIC24HJ12GP201/202. Boot Segment ends at 0x0007FF in other all other devices.]            101 = Standard security, Medium-sized Boot Program Flash            [Boot Segment ends at 0x0007FF in dsPIC33FJ06GS101/102/202, dsPIC33FJ16GS402/404/502/504, dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202 and PIC24HJ12GP201/202. Boot Segment ends at 0x001FFF in all other devices.]            100 = Standard security, Large-sized Boot Program Flash            [Boot Segment ends at 0x000FFF in dsPIC33FJ06GS101/102/202, dsPIC33FJ16GS402/404/502/504, dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202 and PIC24HJ12GP201/202. Boot Segment ends at 0x003FFF in all other devices.]            011 = No Boot Segment            010 = High security, Small-sized Boot Program Flash            [Boot Segment ends at 0x0003FF in dsPIC33FJ06GS101/102/202, dsPIC33FJ16GS402/404/502/504, dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202 and PIC24HJ12GP201/202 devices. Boot Segment ends at 0x0007FF in all other devices.]            001 = High security, Medium-sized Boot Program Flash            [Boot Segment ends at 0x0007FF in dsPIC33FJ06GS101/102/202, dsPIC33FJ16GS402/404/502/504, dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202 and PIC24HJ12GP201/202 devices. Boot Segment ends at 0x001FFF in all other devices.]            000 = High security, Large-sized Boot Program Flash            [Boot Segment ends at 0x000FFF in dsPIC33FJ06GS101/102/202, dsPIC33FJ16GS402/404/502/504, dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202 and PIC24HJ12GP201/202 devices. Boot Segment ends at 0x003FFF in all other devices.]</p>

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 3-2: dsPIC33F/PIC24H CONFIGURATION BITS DESCRIPTION (CONTINUED)**

Bit Field	Register	Description
BWRP	FBS	<b>Boot Segment Program Memory Write Protection</b> 1 = Boot Segment program memory is not write-protected 0 = Boot Segment program memory is write-protected
RSS<1:0>	FSS	<b>Secure Segment Data RAM Code Protection</b> 11 = No Data RAM is reserved for Secure Segment 10 = Small-Sized Secure RAM [(256 – N) bytes of RAM are reserved for Secure Segment in all other devices.] 01 = Medium-Sized Secure RAM [(2048 – N) bytes of RAM are reserved for Secure Segment in all other devices.] 00 = Large-Sized Secure RAM [(4096 – N) bytes of RAM are reserved for Secure Segment in all other devices.] where N = Number of bytes of RAM reserved for Boot Sector.  <b>Note 1:</b> If the defined Boot Segment size is greater than or equal to the defined Secure Segment, then the Secure Segment size selection has no effect and the Secure Segment is disabled.
SSS<2:0>	FSS	<b>Secure Segment Program Memory Code Protection</b> 111 = No Secure Segment 110 = Standard security, Small-sized Secure Program Flash [Secure Segment ends at 0x001FFF for dsPIC33FJ64GPXXX/ dsPIC33FJ64MCXXX/PIC24HJ64GPXXX devices, and at 0x003FFF in other devices.] 101 = Standard security, Medium-sized Secure Program Flash [Secure Segment ends at 0x003FFF for dsPIC33FJ64GPXXX/ dsPIC33FJ64MCXXX/PIC24HJ64GPXXX devices, and at 0x007FFF in other devices.] 100 = Standard security, Large-sized Secure Program Flash [Secure Segment ends at 0x007FFF for dsPIC33FJ64GPXXX/ dsPIC33FJ64MCXXX/PIC24HJ64GPXXX devices, and at 0x00FFFF in other devices.] 011 = No Secure Segment 010 = High security, Small-sized Secure Program Flash [Secure Segment ends at 0x001FFF for dsPIC33FJ64GPXXX/ dsPIC33FJ64MCXXX/PIC24HJ64GPXXX devices, and at 0x003FFF in other devices.] 001 = High security, Medium-sized Secure Program Flash [Secure Segment ends at 0x003FFF for dsPIC33FJ64GPXXX/ dsPIC33FJ64MCXXX/PIC24HJ64GPXXX devices, and at 0x007FFF in other devices.] 000 = High security, Large-sized Secure Program Flash [Secure Segment ends at 0x007FFF for dsPIC33FJ64GPXXX/ dsPIC33FJ64MCXXX/PIC24HJ64GPXXX devices, and at 0x00FFFF in other devices.]
SWRP	FSS	<b>Secure Segment Program Memory Write Protection</b> 1 = Secure Segment program memory is not write-protected 0 = Secure Segment program memory is write-protected
GSS<1:0>	FGS	<b>General Segment Code-Protect bit</b> 11 = Code protection is disabled 10 = Standard security code protection is enabled 0x = High security code protection is enabled
GWRP	FGS	<b>General Segment Write-Protect bit</b> 1 = General Segment program memory is not write-protected 0 = General Segment program memory is write-protected

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 3-2: dsPIC33F/PIC24H CONFIGURATION BITS DESCRIPTION (CONTINUED)**

Bit Field	Register	Description
IESO	FOSCSEL	<b>Two-Speed Oscillator Start-Up Enable bit</b> 1 = Start-Up device with FRC, then automatically switch to the user-selected oscillator source when ready 0 = Start-Up device with user-selected oscillator source
FNOSC<2:0>	FOSCSEL	<b>Initial Oscillator Source Selection bits</b> 111 = Internal Fast RC (FRC) oscillator with postscaler 110 = Internal Fast RC (FRC) oscillator with divide-by-16 101 = LPRC oscillator 100 = Secondary (LP) oscillator 011 = Primary (XT, HS, EC) oscillator with PLL 010 = Primary (XT, HS, EC) oscillator 001 = Internal Fast RC (FRC) oscillator with PLL 000 = FRC oscillator
FCKSM<1:0>	FOSC	<b>Clock Switching Mode bits</b> 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
IOL1WAY	FOSC	<b>Peripheral Pin Select Configuration</b> 1 = Allow only one reconfiguration 0 = Allow multiple reconfigurations
OSCI0FNC	FOSC	<b>OSC2 Pin Function bit (except in XT and HS modes)</b> 1 = OSC2 is clock output 0 = OSC2 is general purpose digital I/O pin
POSCMD<1:0>	FOSC	<b>Primary Oscillator Mode Select bits</b> 11 = Primary oscillator disabled 10 = HS crystal oscillator mode 01 = XT crystal oscillator mode 00 = EC (external clock) mode
PLLKEN	FWDT	<b>PLL Lock Enable bit</b> 1 = Clock switch to PLL source waits for valid PLL lock signal 0 = Clock switch to PLL source ignores PLL lock signal
FWDTEN	FWDT	<b>Watchdog Enable bit</b> 1 = Watchdog always enabled (LPRC oscillator cannot be disabled. Clearing the SWDTEN bit in the RCON register will have no effect) 0 = Watchdog enabled/disabled by user software (LPRC can be disabled by clearing the SWDTEN bit in the RCON register)
WINDIS	FWDT	<b>Watchdog Timer Window Enable bit</b> 1 = Watchdog Timer in Non-Window mode 0 = Watchdog Timer in Window mode
WDTPRE	FWDT	<b>Watchdog Timer Prescaler bit</b> 1 = 1:128 0 = 1:32
WDTPOST	FWDT	<b>Watchdog Timer Postscaler bits</b> 1111 = 1:32,768 1110 = 1:16,384 • • • 0001 = 1:2 0000 = 1:1
PWMPIN	FPOR	<b>Motor Control PWM Module Pin mode</b> 1 = PWM module pins controlled by PORT register at device Reset (tri-stated) 0 = PWM module pins controlled by PWM module at device Reset (configured as output pins)

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 3-2: dsPIC33F/PIC24H CONFIGURATION BITS DESCRIPTION (CONTINUED)**

Bit Field	Register	Description
HPOL	FPOR	<b>Motor Control PWM High-Side Polarity bit</b> 1 = PWM module high-side output pins have active-high output polarity 0 = PWM module high-side output pins have active-low output polarity
LPOL	FPOR	<b>Motor Control PWM Low-Side Polarity bit</b> 1 = PWM module low-side output pins have active-high output polarity 0 = PWM module low-side output pins have active-low output polarity
ALTI2C	FPOR	<b>Alternate I<sup>2</sup>C™ pins</b> 1 = I <sup>2</sup> C mapped to SDA1/SCL1 pins 0 = I <sup>2</sup> C mapped to ASDA1/SACL1 pins
ALTQIO	FPOR	<b>Enable Alternate QEI pins</b> 1 = QEA1A, AEB1A and INDX1A are selected as inputs to QEI1 0 = QEA1, AEB1 and INDX1 are selected as inputs to QEI1
ALTSS1	FPOR	<b>Enable Alternate SS1 pins</b> 1 = SS1A is selected as I/O to SPI1 0 = SS1 is selected as I/O to SPI1
BOREN	FPOR	<b>Brown-out Reset Enable Bit</b> 1 = BOR is enabled in hardware 0 = BOR is disabled in hardware
FPWRT<2:0>	FPOR	<b>Power-on Reset Timer Value Select bits</b> 111 = PWRT = 128 ms 110 = PWRT = 64 ms 101 = PWRT = 32 ms 100 = PWRT = 16 ms 011 = PWRT = 8 ms 010 = PWRT = 4 ms 001 = PWRT = 2 ms 000 = PWRT Disabled
JTAGEN	FICD	<b>JTAG Enable bit</b> 1 = JTAG enabled 0 = JTAG disabled
ICS<1:0>	FICD	<b>ICD Communication Channel Select bits</b> 11 = Communicate on PGC1/EMUC1 and PGD1/EMUD1 10 = Communicate on PGC2/EMUC2 and PGD2/EMUD2 01 = Communicate on PGC3/EMUC3 and PGD3/EMUD3 00 = Reserved, do not use
CMPPOL0	FCMP	Comparator Hysteresis Polarity (for even numbered comparators) 1 = Hysteresis is applied to falling edge 0 = Hysteresis is applied to rising edge
HYST0<1:0>	FCMP	Comparator Hysteresis Select 11 = 45 mV Hysteresis 10 = 30 mV Hysteresis 01 = 15 mV Hysteresis 00 = No Hysteresis
CMPPOL1	FCMP	Comparator Hysteresis Polarity (for odd numbered comparators) 1 = Hysteresis is applied to falling edge 0 = Hysteresis is applied to rising edge
HYST1<1:0>	FCMP	Comparator Hysteresis Select 11 = 45 mV Hysteresis 10 = 30 mV Hysteresis 01 = 15 mV Hysteresis 00 = No Hysteresis
—	All	Unimplemented (read as '0', write as '0')

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 3-3: dsPIC33FJ06GS101/X02 AND dsPIC33FJ16GSX02/X04 DEVICE CONFIGURATION REGISTER MAP**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xF80000	FBS	—	—	—	—	BSS<2:0>			BWRP
0xF80002	Reserved	—	—	—	—	—	—	—	—
0xF80004	FGS	—	—	—	—	—	GSS<1:0>		GWRP
0xF80006	FOSCSEL	IESO	—	—	—	—	FNOSC<2:0>		
0xF80008	FOSC	FCKSM<1:0>		IOL1WAY	—	—	OSCIOFNC	POSCMD<1:0>	
0xF8000A	FWDT	FWDTEN	WINDIS	—	WDTPRE	WDTPOST<3:0>			
0xF8000C	FPOR	—	—	—	—	Reserved <sup>(2)</sup>	FPWRT<2:0>		
0xF8000E	FICD	Reserved <sup>(1)</sup>		JTAGEN <sup>(3)</sup>	—	—	—	ICS<1:0>	
0xF80010	FUID0	User Unit ID Byte 0							
0xF80012	FUID1	User Unit ID Byte 1							

**Legend:** — = unimplemented bit, read as '0'.

**Note 1:** These bits are reserved (read as '1') and must be programmed as '1'.

**2:** This bit reads the current programmed value.

**3:** The JTAGEN bit is set to '1' by factory default. Microchip programmers such as MPLAB<sup>®</sup> ICD 2 and REAL ICE™ in-circuit emulator clear this bit by default when connecting to a device.

**TABLE 3-4: dsPIC33FJ12GP201/202 AND PIC24HJ12GP201/201 DEVICE CONFIGURATION REGISTER MAP**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xF80000	FBS	—	—	—	—	BSS<2:0>			BWRP
0xF80002	Reserved	Reserved <sup>(1)</sup>							
0xF80004	FGS	—	—	—	—	—	GSS<1:0>		GWRP
0xF80006	FOSCSEL	IESO	—	—	—	—	FNOSC<2:0>		
0xF80008	FOSC	FCKSM<1:0>		IOL1WAY	—	—	OSCIOFNC	POSCMD<1:0>	
0xF8000A	FWDT	FWDTEN	WINDIS	—	WDTPRE	WDTPOST<3:0>			
0xF8000C	FPOR	—	—	—	ALT12C	—	FPWRT<2:0>		
0xF8000E	FICD	Reserved <sup>(1)</sup>		JTAGEN <sup>(2)</sup>	—	—	—	ICS<1:0>	
0xF80010	FUID0	User Unit ID Byte 0							
0xF80012	FUID1	User Unit ID Byte 1							
0xF80014	FUID2	User Unit ID Byte 2							
0xF80016	FUID3	User Unit ID Byte 3							

**Legend:** — = unimplemented bit, read as '0'.

**Note 1:** These reserved bits read as '1' and must be programmed as '1'.

**2:** The JTAGEN bit is set to '1' by factory default. Microchip programmers such as MPLAB ICD 2 and REAL ICE in-circuit emulator clear this bit by default when connecting to a device.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 3-5: dsPIC33FJ12MC201/202 DEVICE CONFIGURATION REGISTER MAP**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xF80000	FBS	—	—	—	—	BSS<2:0>			BWRP
0xF80002	Reserved	Reserved <sup>(1)</sup>							
0xF80004	FGS	—	—	—	—	—	GSS<1:0>		GWRP
0xF80006	FOSCSEL	IESO	—	—	—		FNOSC<2:0>		
0xF80008	FOSC	FCKSM<1:0>		IOL1WAY	—	—	OSCIOFNC	POSCMD<1:0>	
0xF8000A	FWDT	FWDTEN	WINDIS	—	WDTPRE	WDTPOST<3:0>			
0xF8000C	FPOR	PWMPIN	HPOL	LPOL	ALTI2C	—	FPWRT<2:0>		
0xF8000E	FICD	Reserved <sup>(1)</sup>		JTAGEN <sup>(2)</sup>	—	—	—	ICS<1:0>	
0xF80010	FUID0	User Unit ID Byte 0							
0xF80012	FUID1	User Unit ID Byte 1							
0xF80014	FUID2	User Unit ID Byte 2							
0xF80016	FUID3	User Unit ID Byte 3							

**Legend:** — = unimplemented bit, read as '0'.

**Note 1:** These reserved bits read as '1' and must be programmed as '1'.

**2:** The JTAGEN bit is set to '1' by factory default. Microchip programmers such as MPLAB ICD 2 and REAL ICE in-circuit emulator clear this bit by default when connecting to a device.

**TABLE 3-6: dsPIC33FJ32GP202/204 AND dsPIC33FJ16GP304, AND PIC24HJ32GP202/204 AND PIC24HJ16GP304 DEVICE CONFIGURATION REGISTER MAP**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xF80000	FBS	—	—	—	—	BSS<2:0>			BWRP
0xF80002	Reserved	—	—	—	—	—	—	—	—
0xF80004	FGS	—	—	—	—	—	GSS<1:0>		GWRP
0xF80006	FOSCSEL	IESO	—	—	—		FNOSC<2:0>		
0xF80008	FOSC	FCKSM<1:0>		IOL1WAY	—	—	OSCIOFNC	POSCMD<1:0>	
0xF8000A	FWDT	FWDTEN	WINDIS	—	WDTPRE	WDTPOST<3:0>			
0xF8000C	FPOR	Reserved <sup>(2)</sup>			ALTI2C	—	FPWRT<2:0>		
0xF8000E	FICD	Reserved <sup>(1)</sup>		JTAGEN <sup>(3)</sup>	—	—	—	ICS<1:0>	
0xF80010	FUID0	User Unit ID Byte 0							
0xF80012	FUID1	User Unit ID Byte 1							
0xF80014	FUID2	User Unit ID Byte 2							
0xF80016	FUID3	User Unit ID Byte 3							

**Legend:** — = unimplemented bit, read as '0'.

**Note 1:** These bits are reserved (read as '1') and must be programmed as '1'.

**2:** These bits are reserved and always read as '1'.

**3:** The JTAGEN bit is set to '1' by factory default. Microchip programmers such as MPLAB ICD 2 and REAL ICE in-circuit emulator clear this bit by default when connecting to a device.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 3-7: dsPIC33FJ32MC202/204 AND dsPIC33FJ16MC304 DEVICE CONFIGURATION REGISTER MAP**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xF80000	FBS	—	—	—	—	BSS<2:0>			BWRP
0xF80002	RESERVED	—	—	—	—	—	—	—	—
0xF80004	FGS	—	—	—	—	—	GSS<1:0>		GWRP
0xF80006	FOSCSEL	IESO	—	—	—		FNOSC<2:0>		
0xF80008	FOSC	FCKSM<1:0>		IOL1WAY	—	—	OSCIOFNC	POSCMD<1:0>	
0xF8000A	FWDT	FWDTEN	WINDIS	—	WDTPRE	WDTPOST<3:0>			
0xF8000C	FPOR	PWMPIN	HPOL	LPOL	ALTI2C	—	FPWRT<2:0>		
0xF8000E	FICD	Reserved <sup>(1)</sup>		JTAGEN <sup>(2)</sup>	—	—	—	ICS<1:0>	
0xF80010	FUID0	User Unit ID Byte 0							
0xF80012	FUID1	User Unit ID Byte 1							
0xF80014	FUID2	User Unit ID Byte 2							
0xF80016	FUID3	User Unit ID Byte 3							

**Legend:** — = unimplemented bit, read as '0'.

**Note 1:** These bits are reserved (read as '1') and must be programmed as '1'.

**2:** The JTAGEN bit is set to '1' by factory default. Microchip programmers such as MPLAB ICD 2 and REAL ICE in-circuit emulator clear this bit by default when connecting to a device.

**TABLE 3-8: dsPIC33FJ32GP302/304, dsPIC33FJ64GPX02/X04 AND dsPIC33FJ128GPX02/X04, AND PIC24HJ32GP302/304, PIC24HJ64GPX02/X04 AND PIC24HJ128GPX02/X04 DEVICE CONFIGURATION REGISTER MAP**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xF80000	FBS	RBS<1:0>		—	—	BSS<2:0>			BWRP
0xF80002	FSS <sup>(1)</sup>	RSS<1:0>		—	—	SSS<2:0>			SWRP
0xF80004	FGS	—	—	—	—	—	GSS<1:0>		GWRP
0xF80006	FOSCSEL	IESO	—	—	—		FNOSC<2:0>		
0xF80008	FOSC	FCKSM<1:0>		IOL1WAY	—	—	OSCIOFNC	POSCMD<1:0>	
0xF8000A	FWDT	FWDTEN	WINDIS	—	WDTPRE	WDTPOST<3:0>			
0xF8000C	FPOR	Reserved <sup>(2)</sup>			ALTI2C	—	FPWRT<2:0>		
0xF8000E	FICD	Reserved <sup>(3)</sup>		JTAGEN <sup>(4)</sup>	—	—	—	ICS<1:0>	
0xF80010	FUID0	User Unit ID Byte 0							
0xF80012	FUID1	User Unit ID Byte 1							
0xF80014	FUID2	User Unit ID Byte 2							
0xF80016	FUID3	User Unit ID Byte 3							

**Legend:** — = unimplemented bit, read as '0'.

**Note 1:** This Configuration register is not available and reads as 0xFF on dsPIC33FJ32GP302/304 devices.

**2:** These bits are reserved and always read as '1'.

**3:** These bits are reserved (read as '1') and must be programmed as '1'.

**4:** The JTAGEN bit is set to '1' by factory default. Microchip programmers such as MPLAB ICD 2 and REAL ICE in-circuit emulator clear this bit by default when connecting to a device.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 3-9: dsPIC33FJ32MC302/304, dsPIC33FJ64MCX02/X04 AND dsPIC33FJ128MCX02/X04 DEVICE CONFIGURATION REGISTER MAP**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xF80000	FBS	RBS<1:0>		—	—	BSS<2:0>		BWRP	
0xF80002	FSS <sup>(1)</sup>	RSS<1:0>		—	—	SSS<2:0>		SWRP	
0xF80004	FGS	—	—	—	—	—	GSS<1:0>		GWRP
0xF80006	FOSCSEL	IESO	—	—	—	FNOSC<2:0>			
0xF80008	FOSC	FCKSM<1:0>		IOL1WAY	—	—	OSCIOFNC	POSCMD<1:0>	
0xF8000A	FWDT	FWDTEN	WINDIS	—	WDTPRE	WDTPOST<3:0>			
0xF8000C	FPOR	PWMPIN	HPOL	LPOL	ALTI2C	—	FPWRT<2:0>		
0xF8000E	FICD	Reserved <sup>(2)</sup>		JTAGEN <sup>(3)</sup>	—	—	—	ICS<1:0>	
0xF80010	FUID0	User Unit ID Byte 0							
0xF80012	FUID1	User Unit ID Byte 1							
0xF80014	FUID2	User Unit ID Byte 2							
0xF80016	FUID3	User Unit ID Byte 3							

**Legend:** — = unimplemented bit, read as '0'.

**Note 1:** This Configuration register is not available and reads as 0xFF on dsPIC33FJ32MC302/304 devices.

**2:** These bits are reserved (read as '1') and must be programmed as '1'.

**3:** The JTAGEN bit is set to '1' by factory default. Microchip programmers such as MPLAB ICD 2 and REAL ICE in-circuit emulator clear this bit by default when connecting to a device.

**TABLE 3-10: dsPIC33FJ32GS406/606/608/610 AND dsPIC33FJ64GS406/606/608/610 DEVICE CONFIGURATION REGISTER MAP**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xF80000	FBS	—	—	—	—	BSS<2:0>		BWRP	
0xF80004	FGS	—	—	—	—	—	GSS<1:0>		GWRP
0xF80006	FOSCSEL	IESO	—	—	—	—	FNOSC<2:0>		
0xF80008	FOSC	FCKSM<1:0>		—	—	—	OSCIOFNC	POSCMD<1:0>	
0xF8000A	FWDT	FWDTEN	WINDIS	—	WDTPRE	WDTPOST<3:0>			
0xF8000C	FPOR	—	ALTQIO	ALTSS1	—	—	FPWRT<2:0>		
0xF8000E	FICD	Reserved <sup>(1)</sup>		JTAGEN <sup>(2)</sup>	—	—	—	ICS<1:0>	
0xF80010	FCMP	—	—	CMPOL1 <sup>(3)</sup>	HYST1<1:0> <sup>(3)</sup>		CMPPOL0 <sup>(3)</sup>	HYST0<1:0> <sup>(3)</sup>	

**Legend:** — = unimplemented bit, read as '0'.

**Note 1:** These bits are reserved (read as '1') and must be programmed as '1'.

**2:** The JTAGEN bit is set to '1' by factory default. Microchip programmers such as MPLAB ICD 2 and REAL ICE in-circuit emulator clear this bit by default when connecting to a device.

**3:** These bits are reserved on dsPIC33FJXXXGS406 devices and always read as '1'.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 3-11: dsPIC33FJXXXGPX06A/X08A/X10A AND PIC24HJXXXGPX06A/X08A/X10A DEVICE CONFIGURATION REGISTER MAP**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xF80000	FBS	RBS<1:0>		—	—	BSS<2:0>		BWRP	
0xF80002	FSS	RSS<1:0>		—	—	SSS<2:0>		SWRP	
0xF80004	FGS	—	—	—	—	—	GSS1	GSS0	GWRP
0xF80006	FOSCSEL	IESO	Reserved <sup>(2)</sup>	—	—	—	FNOSC<2:0>		
0xF80008	FOSC	FCKSM<1:0>		—	—	—	OSCIOFNC	POSCMD<1:0>	
0xF8000A	FWDT	FWDTEN	WINDIS	PLLKEN <sup>(3)</sup>	WDTPRE	WDTPOST<3:0>			
0xF8000C	FPOR	Reserved <sup>(4)</sup>			—	—	FPWRT<2:0>		
0xF8000E	FICD	Reserved <sup>(1)</sup>		JTAGEN	—	—	—	ICS<1:0>	
0xF80010	FUID0	User Unit ID Byte 0							
0xF80012	FUID1	User Unit ID Byte 1							
0xF80014	FUID2	User Unit ID Byte 2							
0xF80016	FUID3	User Unit ID Byte 3							

**Legend:** — = unimplemented bit, read as '0'.

**Note 1:** These bits are reserved for use by development tools and must be programmed as '1'.

**2:** When read, this bit returns the current programmed value.

**3:** This bit is unimplemented on dsPIC33FJ64GPX06A/X08A/X10A and dsPIC33FJ128GPX06A/X08A/X10A devices and reads as '0'.

**4:** These bits are reserved and always read as '1'.

**TABLE 3-12: dsPIC33FJXXXMCX06A/X08A/X10A DEVICE CONFIGURATION REGISTER MAP**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xF80000	FBS	RBS<1:0>		—	—	BSS<2:0>		BWRP	
0xF80002	FSS	RSS<1:0>		—	—	SSS<2:0>		SWRP	
0xF80004	FGS	—	—	—	—	—	GSS1	GSS0	GWRP
0xF80006	FOSCSEL	IESO	Reserved <sup>(2)</sup>	—	—	—	FNOSC<2:0>		
0xF80008	FOSC	FCKSM<1:0>		—	—	—	OSCIOFNC	POSCMD<1:0>	
0xF8000A	FWDT	FWDTEN	WINDIS	PLLKEN <sup>(3)</sup>	WDTPRE	WDTPOST<3:0>			
0xF8000C	FPOR	PWMPIN	HPOL	LPOL	—	—	FPWRT<2:0>		
0xF8000E	FICD	Reserved <sup>(1)</sup>		JTAGEN	—	—	—	ICS<1:0>	
0xF80010	FUID0	User Unit ID Byte 0							
0xF80012	FUID1	User Unit ID Byte 1							
0xF80014	FUID2	User Unit ID Byte 2							
0xF80016	FUID3	User Unit ID Byte 3							

**Legend:** — = unimplemented bit, reads as '0'.

**Note 1:** These bits are reserved for use by development tools and must be programmed as '1'.

**2:** When read, this bit returns the current programmed value.

**3:** This bit is unimplemented on dsPIC33FJ64MCX06A/X08A/X10A and dsPIC33FJ128MCX06A/X08A/X10A devices and reads as '0'.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 3.6.2 PROGRAMMING METHODOLOGY

Configuration bits may be programmed a single byte at a time using the `PROGC` command. This command specifies the configuration data and Configuration register address.

Twelve `PROGC` commands are required to program all the Configuration bits. A flowchart for Configuration bit programming is illustrated in Figure 3-5.

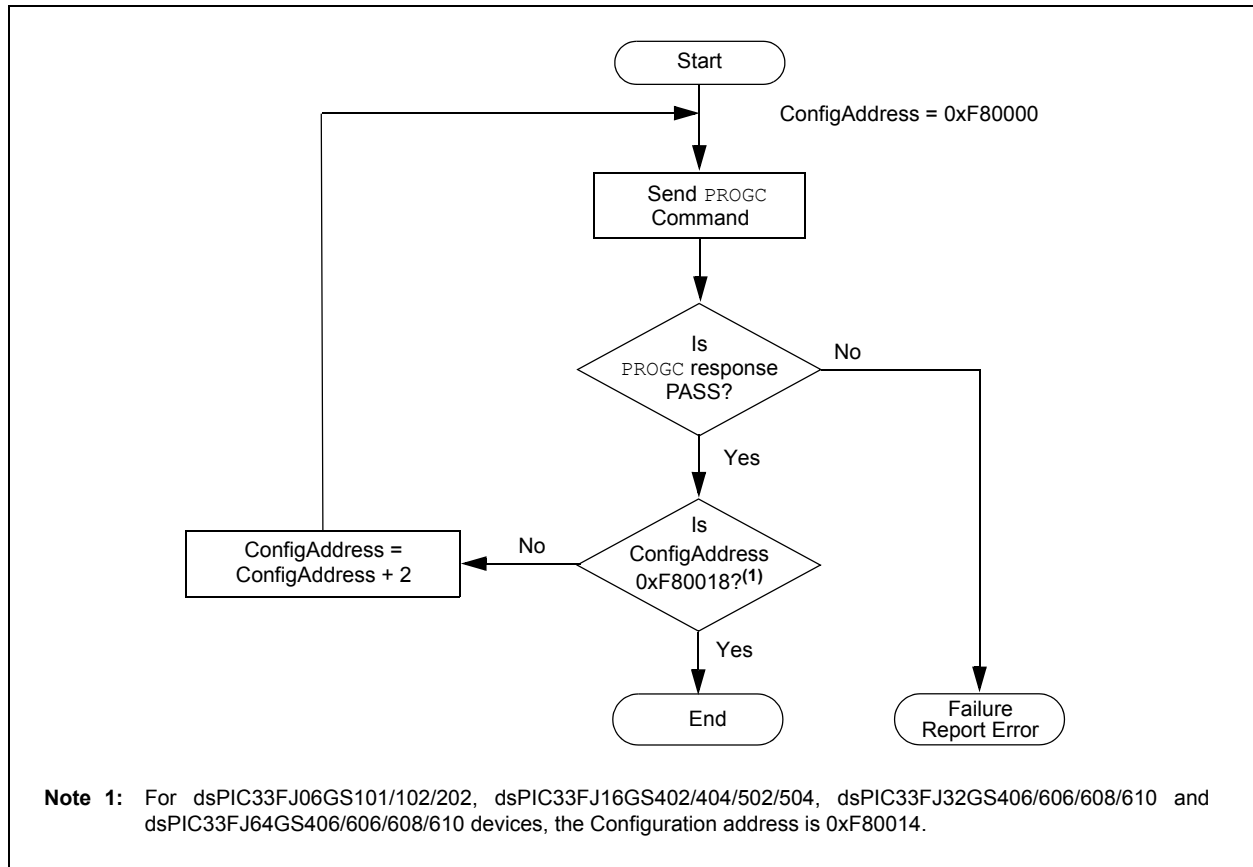
**Note:** If either of the General Code Segment Code-Protect bits ( $GSS<1:0>$ ) is programmed to '0', code memory is code-protected and cannot be read. Code memory must be verified before enabling read protection. See Section 3.6.4 "CodeGuard™ Security Configuration Bits" for detailed information about code-protect Configuration bits.

## 3.6.3 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 `READC` command reads back the programmed Configuration bits and verifies that the programming was successful.

Any unimplemented Configuration bits are read-only and read as '0'. The reserved bits are read-only and read as '1'.

FIGURE 3-5: CONFIGURATION BIT PROGRAMMING FLOW



**Note 1:** For dsPIC33FJ06GS101/102/202, dsPIC33FJ16GS402/404/502/504, dsPIC33FJ32GS406/606/608/610 and dsPIC33FJ64GS406/606/608/610 devices, the Configuration address is 0xF80014.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 3.6.4 CodeGuard™ SECURITY CONFIGURATION BITS

The FBS, FSS and FGS Configuration registers are special Configuration registers that control the size and level of code protection for the Boot Segment, Secure Segment and General Segment, respectively. For each segment, two main forms of code protection are provided. One form prevents code memory from being written (write protection), while the other prevents code memory from being read (read protection).

BWRP, SWRP and GWRP bits control write protection and BSS<2:0>, SSS<2:0> and GSS<1:0> bits control read protection. The Chip Erase `ERASEB` command sets all the code protection bits to '1', which allows the device to be programmed.

When write protection is enabled, any programming operation to code memory will fail. When read protection is enabled, any read from code memory will cause a '0x0' to be read, regardless of the actual contents of code memory. Since the programming executive always verifies what it programs, attempting to program code memory with read protection enabled will also result in failure.

It is imperative that all code protection bits are '1' while the device is being programmed and verified. Only after the device is programmed and verified should any of the above bits be programmed to '0'.

In addition to code memory protection, a part of Data RAM can be configured to be accessible only by code resident in the Boot Segment and/or Secure Segment. The sizes of these "reserved" sections are user-configurable, using the RBS<1:0> and RSS<1:0> bits.

**Note:** All bits in the FBS, FSS and FGS Configuration registers can only be programmed to a value of '0'. Bulk Erase and Segment Erase operations (depending on the segment) are the only way to reprogram code-protect bits from ON ('0') to OFF ('1').

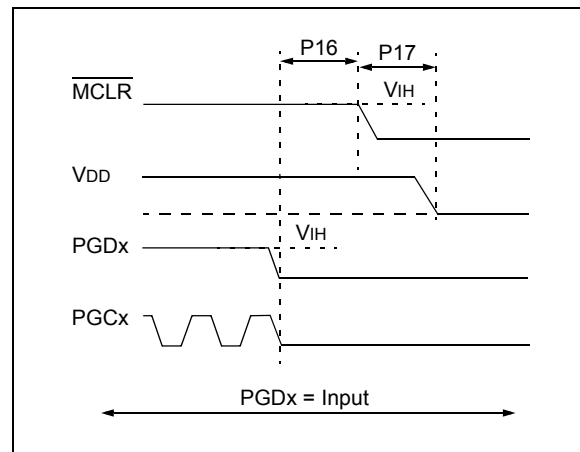
## 3.6.5 BENEFIT OF USER UNIT ID

The dsPIC33F/PIC24H devices provide up to four 8-bit Configuration registers (FUID0 through FUID3) for the user to store product-specific information, such as unit serial numbers and other product manufacturing data.

## 3.7 Exiting Enhanced ICSP Mode

Exiting Program/Verify mode is done by removing  $V_{IH}$  from MCLR, as illustrated in Figure 3-6. The only requirement for exit is that an interval P16 should elapse between the last clock and program signals on PGCx and PGDx before removing  $V_{IH}$ .

**FIGURE 3-6: EXITING ENHANCED ICSP™ MODE**



# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 4.0 THE PROGRAMMING EXECUTIVE

**Note:** The Programming Executive (PE) can be located within the following folder within your installation of MPLAB<sup>®</sup> IDE:

```
... \Microchip \MPLAB IDE \REAL ICE,  
and then selecting the Hex PE file,  
RIPE_01b_XXXXXX.hex.
```

### 4.1 Programming Executive Communication

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.

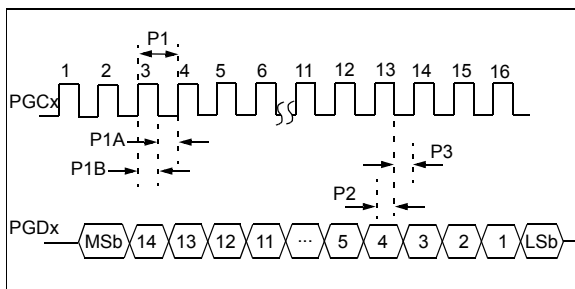
All communication is initiated by the programmer in the form of a command. Only one command at a time can be sent to the programming executive. In turn, the programming executive only sends one response to the programmer after receiving and processing a command. The programming executive command set is described in [Section 4.2 “Programming Executive Commands”](#). The response set is described in [Section 4.3 “Programming Executive Responses”](#).

#### 4.1.1 COMMUNICATION INTERFACE AND PROTOCOL

The ICSP/Enhanced ICSP interface is a 2-wire SPI implemented using the PGCx and PGDx pins. The PGCx pin is used as a clock input pin and the clock source must be provided by the programmer. The PGDx pin is used for sending command data to and receiving response data from the programming executive.

**Note:** For Enhanced ICSP, all serial data is transmitted on the falling edge of PGCx and latched on the rising edge of PGCx. All data transmissions are sent to the Most Significant bit first using 16-bit mode (see [Figure 4-1](#)).

**FIGURE 4-1: 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 PGDx. When the programmer completes a command transmission, it releases the PGDx line and allows the programming executive to drive this line high. The programming executive keeps the PGDx line high to indicate that it is processing the command.

After the programming executive has processed the command, it brings PGDx low (P9b) to indicate to the programmer that the response is available to be clocked out. The programmer can begin to clock out the response after maximum wait (P9b) 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 PGCx until it is time to send another command to the programming executive. This protocol is illustrated in [Figure 4-2](#).

#### 4.1.2 SPI RATE

In Enhanced ICSP mode, the dsPIC33F/PIC24H family devices operate from the Fast Internal RC oscillator, which has a nominal frequency of 7.3728 MHz. This oscillator frequency yields an effective system clock frequency of 1.8432 MHz. To ensure that the programmer does not clock too fast, it is recommended that a 1.85 MHz clock be provided by the programmer.

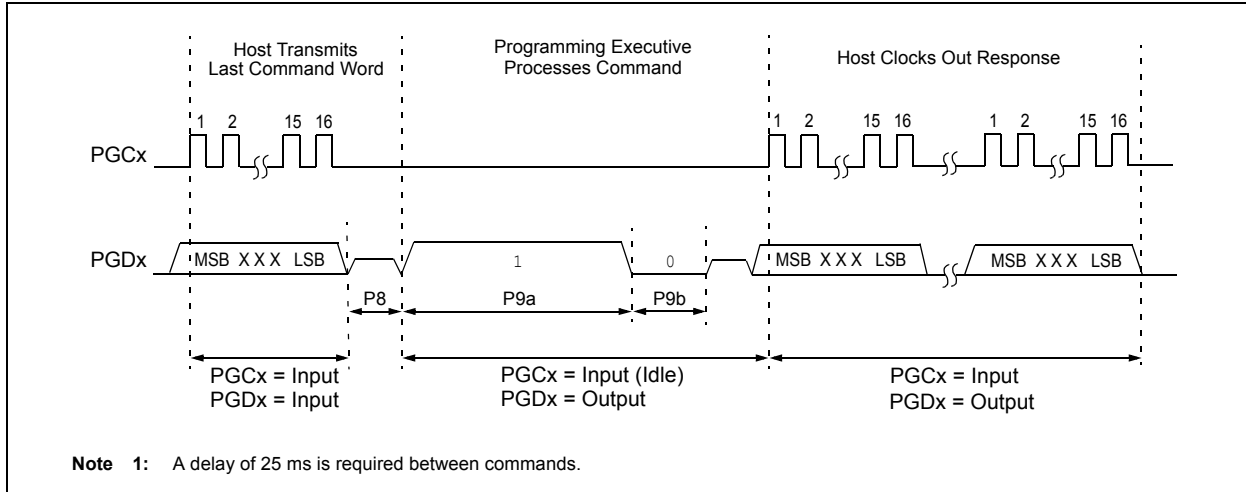
#### 4.1.3 TIME OUTS

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 PGCx as described in [Section 4.1.1 “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 4-1](#). If the command time out expires, the programmer should reset the programming executive and start programming the device again.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**FIGURE 4-2: PROGRAMMING EXECUTIVE – PROGRAMMER COMMUNICATION PROTOCOL**



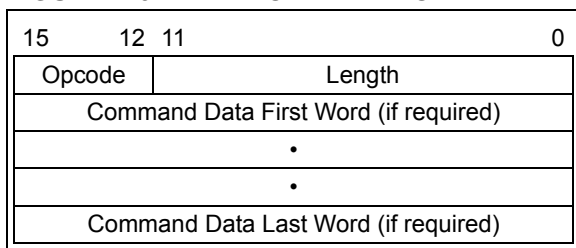
## 4.2 Programming Executive Commands

The programming executive command set is shown in [Table 4-1](#). 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 4.2.4 “Command Descriptions”](#)).

### 4.2.1 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 4-3](#)). 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 4-3: COMMAND FORMAT**



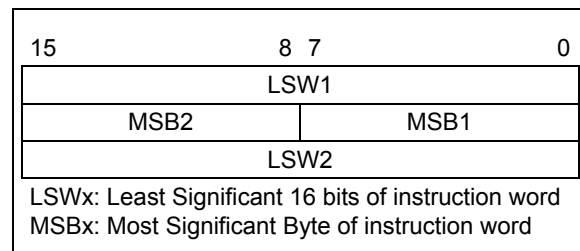
The command opcode must match one of those in the command set. Any command that is received which does not match the list in [Table 4-1](#) will return a “NACK” response (see [Section 4.3.1.1 “Opcode Field”](#)).

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.

### 4.2.2 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 illustrated in [Figure 4-4](#). This format minimizes traffic over the SPI and provides the programming executive with data that is properly aligned for performing table write operations.

**FIGURE 4-4: PACKED INSTRUCTION WORD FORMAT**



**Note:** When the number of instruction words transferred is odd, MSB2 is zero and LSW2 cannot be transmitted.

### 4.2.3 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 4.3.1.3 “QE\\_Code Field”](#).

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 4-1: PROGRAMMING EXECUTIVE COMMAND SET**

Opcode	Mnemonic	Length (16-bit words)	Time Out	Description
0x0	SCHECK	1	1 ms	Sanity check.
0x1	READC	3	1 ms	Read an 8-bit word from the specified Configuration register or Device ID register.
0x2	READP	4	1 ms/row	Read 'N' 24-bit instruction words of code memory starting from the specified address.
0x3	Reserved	N/A	N/A	This command is reserved. It will return a NACK.
0x4	PROGC	4	5 ms	Write an 8-bit word to the specified Configuration register.
0x5	PROGP	99	5 ms	Program one row of code memory at the specified address, then verify.
0x6	Reserved	N/A	N/A	This command is reserved. It will return a NACK.
0x7	Reserved	N/A	N/A	This command is reserved. It will return a NACK.
0x8	Reserved	N/A	N/A	This command is reserved. It will return a NACK.
0x9	ERASEP	3	20 ms	Command to erase a page.
0xA	Reserved	N/A	N/A	This command is reserved. It will return a NACK.
0xB	QVER	1	1 ms	Query the programming executive software version.
0xC	CRCP	5	1s	Performs a CRC-16 on the specified range of memory.
0xD	Reserved	N/A	N/A	This command is reserved. It will return a NACK.
0xE	QBLANK	5	700 ms	Query to check whether the code memory is blank.

**Note:** One row of code memory consists of (64) 24-bit words. Refer to [Table 2-2](#) for device-specific information.

## 4.2.4 COMMAND DESCRIPTIONS

All commands supported by the programming executive are described in [Section 4.2.5 "SCHECK Command"](#) through [Section 4.2.13 "QVER Command"](#).

## 4.2.5 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.

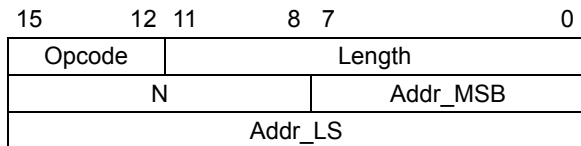
### Expected Response (2 words):

0x1000  
0x0002

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

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 4.2.6 READC COMMAND



Field	Description
Opcode	0x1
Length	0x3
N	Number of 8-bit Configuration registers or Device ID registers to read (maximum of 256).
Addr_MSB	MSB of 24-bit source address.
Addr_LS	Least Significant 16 bits of 24-bit source address.

The READC command instructs the programming executive to read N Configuration registers or Device ID registers, starting from the 24-bit address specified by Addr\_MSB and Addr\_LS. This command can only be used to read 8-bit or 16-bit data.

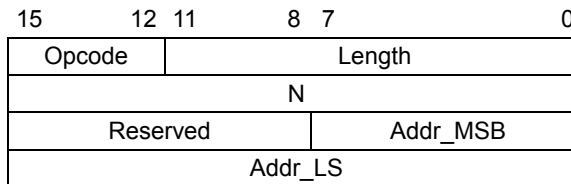
When this command is used to read Configuration registers, the upper byte in every data word returned by the programming executive is 0x00 and the lower byte contains the Configuration register value.

### Expected Response ( $4 + 3 * (N - 1)/2$ words for N odd):

0x1100  
 $2 + N$   
 Configuration register or Device ID Register 1  
 ...  
 Configuration register or Device ID Register N

**Note:** Reading unimplemented memory will cause the programming executive to reset. Please ensure that only memory locations present on a particular device are accessed.

## 4.2.7 READP COMMAND



Field	Description
Opcode	0x2
Length	0x4
N	Number of 24-bit instructions to read (maximum of 32768).
Reserved	0x0
Addr_MSB	MSB of 24-bit source address.
Addr_LS	Least Significant 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 4.2.2 "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. Please ensure that only memory locations present on a particular device are accessed.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 4.2.8 PROGC COMMAND

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

Field	Description
Opcode	0x4
Length	0x4
Reserved	0x0
Addr_MSB	MSB of 24-bit destination address.
Addr_LS	Least Significant 16 bits of 24-bit destination address.
Data	8-bit data word.

The PROGC command instructs the programming executive to program a single Configuration register, located at the specified memory address.

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

### Expected Response (2 words):

0x1400  
0x0002

## 4.2.9 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	0x63
Reserved	0x0
Addr_MSB	MSB of 24-bit destination address.
Addr_LS	Least Significant 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 95.
D_96	16-bit data word 96.

The PROGP command instructs the programming executive to program one row of code memory (64 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 0x80.

The data to program the memory, located in command words D\_1 through D\_96, must be arranged using the packed instruction word format illustrated in [Figure 4-4](#).

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 2-2](#) for code memory size information.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 4.2.10 ERASEP COMMAND

15            12 11            8 7                            0

Opcode	Length		
NUM_PAGES		Addr_MSB	
Addr_LS			

Field	Description
Opcode	0x9
Length	0x3
NUM_PAGES	Up to 255
Addr_MSB	Most Significant Byte of the 24-bit address
Addr_LS	Least Significant 16 bits of the 24-bit address

The **ERASEP** command instructs the programming executive to page erase [NUM\_PAGES] of code memory. The code memory must be erased at an “even” 512 instruction word address boundary

**Expected Response (2 words):**

0x1900  
0x0002

## 4.2.11 CRCP COMMAND

15            12 11            8 7                            0

Opcode	Length		
Reserved		Addr_MSB	
Addr_LSW			
Reserved		Size_MSB	
Size_LSW			

Field	Description
Opcode	0xC
Length	0x5
Reserved	0x0
Addr_MSB	Most Significant Byte of 24-bit address
Addr_LSW	Least Significant 16 bits of 24-bit address
Size	Number of 24-bit locations (address range divided by 2)

The **CRCP** command performs a CRC-16 on the range of memory specified. This command can substitute for a full chip verify. Data is shifted in a packed method as demonstrated in [Figure 4-4](#), byte-wise Least Significant Byte first.

**Example:**

CRC-CITT-16 with test data of “123456789” becomes 0x29B1

**Expected Response (3 words):**

QE\_Code: 0x1C00  
Length: 0x0003  
CRC Value: 0XXXXX

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 4.2.12 QBLANK COMMAND

15	12	11	0
Opcode		Length	
Reserved		Size_MSB	
Size_LSW			
Reserved		Addr_MSB	
Addr_LSW			

Field	Description
Opcode	0xE
Length	0x5
Size	Length of program memory to check (in 24-bit words) + Addr_MS
Addr_MSB	Most Significant Byte of the 24-bit address
Addr_LSW	Least Significant 16 bits of the 24-bit address

The QBLANK command queries the programming executive to determine if the contents of code memory are blank (contains all '1's). The size of code memory to check must be specified in the command.

The Blank Check for code memory begins at [Addr] and advances toward larger addresses for the specified number of instruction words.

QBLANK returns a QE\_Code of 0xF0 if the specified code memory is blank; otherwise, QBLANK returns a QE\_Code of 0x0F.

### Expected Response (2 words for blank device):

0x1EF0  
0x0002

### Expected Response (2 words for non-blank device):

0x1E0F  
0x0002

**Note:** The QBLANK command does not check the system operation Configuration bits since these bits are not set to '1' when a Chip Erase is performed.

## 4.2.13 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

## 4.3 Programming Executive Responses

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 4-2. This table contains the opcode, mnemonic and description for each response. The response format is described in Section 4.3.1 "Response Format".

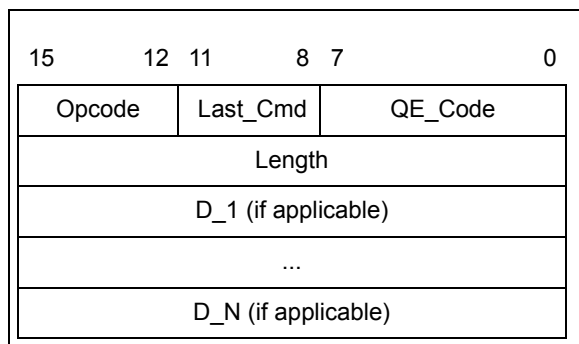
**TABLE 4-2: PROGRAMMING EXECUTIVE RESPONSE OPCODES**

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

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 4.3.1 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).

### 4.3.1.1 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 4-2](#)). 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.

### 4.3.1.2 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.

### 4.3.1.3 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 4-3](#).

**TABLE 4-3: QE\_Code FOR QUERIES**

Query	QE_Code
<code>QBLANK</code>	0x0F = Code memory is NOT blank 0xF0 = Code memory is blank
<code>QVER</code>	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 4-4](#). If a command is successfully processed, the returned QE\_Code is set to 0x0, which indicates that there is no error in the command processing. If the verify of the programming for the `PROGP` or `PROGC` command fails, the QE\_Code is set to 0x1. For all other programming executive errors, the QE\_Code is 0x2.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

---

---

**TABLE 4-4: QE\_Code FOR NON-QUERY COMMANDS**

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

#### 4.3.1.4 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 `READP` command, the length of each response is only 2 words.

The response to the `READP` command uses the packed instruction word format described in [Section 4.2.2 "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.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 5.0 DEVICE PROGRAMMING – ICSP

**Note:** Any development tool that modifies the configuration memory on dsPIC33FJ06GS101/102/202, dsPIC33FJ16GS402/404/502/504, dsPIC33FJ32GS406/606/608/610 and dsPIC33FJ64GS406/606/608/610 devices must take care to preserve the data contained in the last six words of program memory. Refer to [Appendix C: “Diagnostic and Calibration Registers”](#) for more information.

ICSP mode is a special programming protocol that allows you to read and write to dsPIC33F/PIC24H device family memory. The ICSP mode is the most direct method used to program the device; however, note that Enhanced ICSP is faster. ICSP 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 PGCx and PGDx.

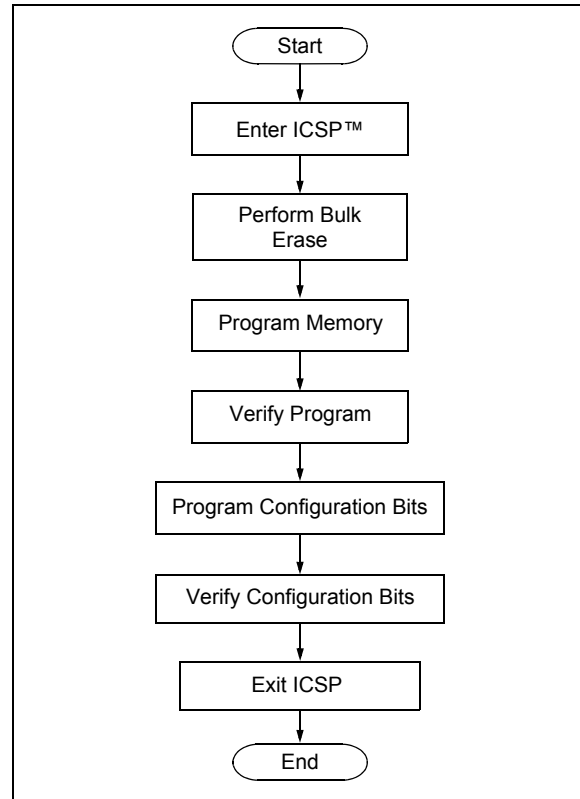
In ICSP mode, the system clock is taken from the PGCx pin, regardless of the device’s oscillator Configuration bits. All instructions are shifted serially into an internal buffer, then loaded into the instruction register and executed. No program fetching occurs from internal memory. Instructions are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

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

## 5.1 Overview of the Programming Process

Figure 5-1 illustrates the high-level overview of the programming process. After entering ICSP mode, the first action is to Bulk Erase the device. Next, the code memory is programmed, followed by the device Configuration registers. Code memory (including the Configuration registers) is then verified to ensure that programming was successful. Then, program the code-protect Configuration bits, if required.

**FIGURE 5-1: HIGH-LEVEL ICSP™ PROGRAMMING FLOW**



# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 5.2 Entering ICSP Mode

As illustrated in [Figure 5-5](#), entering ICSP Program/Verify mode requires three steps:

1.  $\overline{\text{MCLR}}$  is briefly driven high then low (P21)<sup>(1)</sup>.
2. A 32-bit key sequence is clocked into PGDx.
3.  $\overline{\text{MCLR}}$  is then driven high within a specified period of time and held.

**Note 1:** The  $\overline{\text{MCLR}}$  capacitor value can vary the high time required for entering ICSP mode.

The programming voltage applied to  $\overline{\text{MCLR}}$  is  $V_{IH}$ , which is essentially  $V_{DD}$  in the case of dsPIC33F/PIC24H devices. There is no minimum time requirement for holding at  $V_{IH}$ . After  $V_{IH}$  is removed, an interval of at least P18 must elapse before presenting the key sequence on PGDx.

The key sequence is a specific 32-bit pattern, '0100 1101 0100 0011 0100 1000 0101 0001' (more easily remembered as 0x4D434851 in hexadecimal). The device will enter Program/Verify mode only if the sequence is valid. The Most Significant bit of the most significant nibble must be shifted in first.

Once the key sequence is complete,  $V_{IH}$  must be applied to  $\overline{\text{MCLR}}$  and held at that level for as long as Program/Verify mode is to be maintained. An interval of at least time P19 and P7 must elapse before presenting data on PGDx. Signals appearing on PGDx before P7 has elapsed will not be interpreted as valid.

On successful entry, the program memory can be accessed and programmed in serial fashion. While in ICSP mode, all unused I/Os are placed in the high-impedance state.

## 5.3 ICSP Operation

After entering into ICSP mode, the CPU is Idle. Execution of the CPU is governed by an internal state machine. A 4-bit control code is clocked in using PGCx and PGDx and this control code is used to command the CPU (see [Table 5-1](#)).

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.

**TABLE 5-1: CPU CONTROL CODES IN ICSP™ MODE**

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

### 5.3.1 SIX SERIAL INSTRUCTION EXECUTION

The SIX control code allows execution of dsPIC33F/PIC24H Programming Specification 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 5-3](#)).

**Note 1:** Coming out of the ICSP entry sequence, the first 4-bit control code is always forced to SIX and a forced NOP instruction is executed by the CPU. Five additional PGCx clocks are needed on start-up, thereby resulting in a 9-bit SIX command instead of the normal 4-bit SIX command. After the forced SIX is clocked in, ICSP operation resumes as normal (the next 24 clock cycles load the first instruction word to the CPU). See [Figure 5-2](#) for details.

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

### 5.3.2 REGOUT SERIAL INSTRUCTION EXECUTION

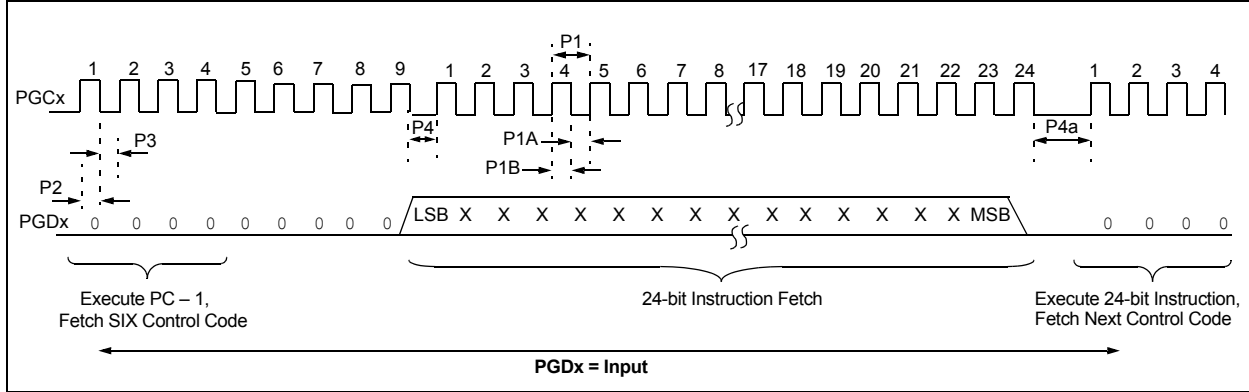
The REGOUT control code allows for data to be extracted from the device in ICSP mode. It is used to clock the contents of the VISI register out of the device over the PGDx pin. After the REGOUT control code is received, the CPU is held Idle for eight cycles. After these eight cycles, an additional 16 cycles are required to clock the data out (see [Figure 5-4](#)).

The REGOUT code is unique because the PGDx pin is an input when the control code is transmitted to the device. However, after the control code is processed, the PGDx pin becomes an output as the VISI register is shifted out.

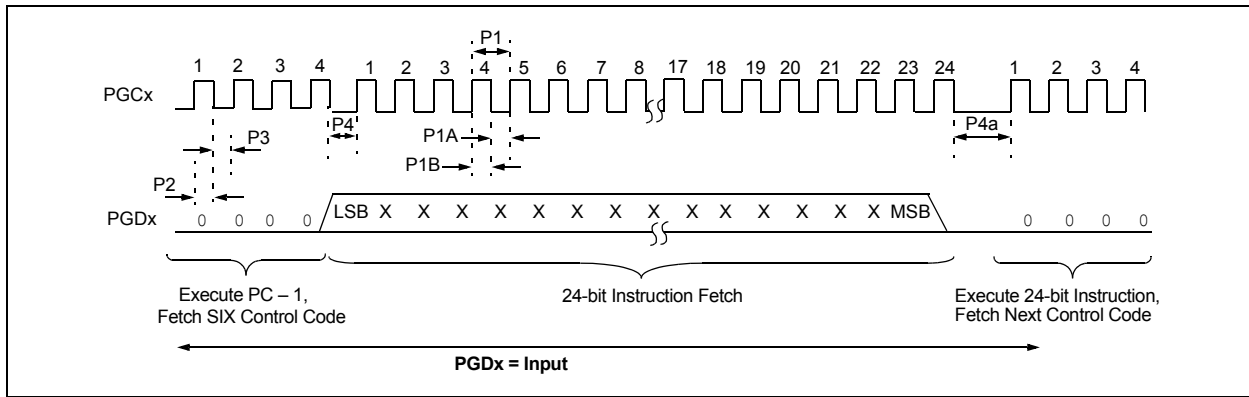
**Note:** The device will latch input PGDx data on the rising edge of PGCx and will output data on the PGDx line on the rising edge of PGCx. For all data transmissions, the Least Significant bit (LSb) is transmitted first.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

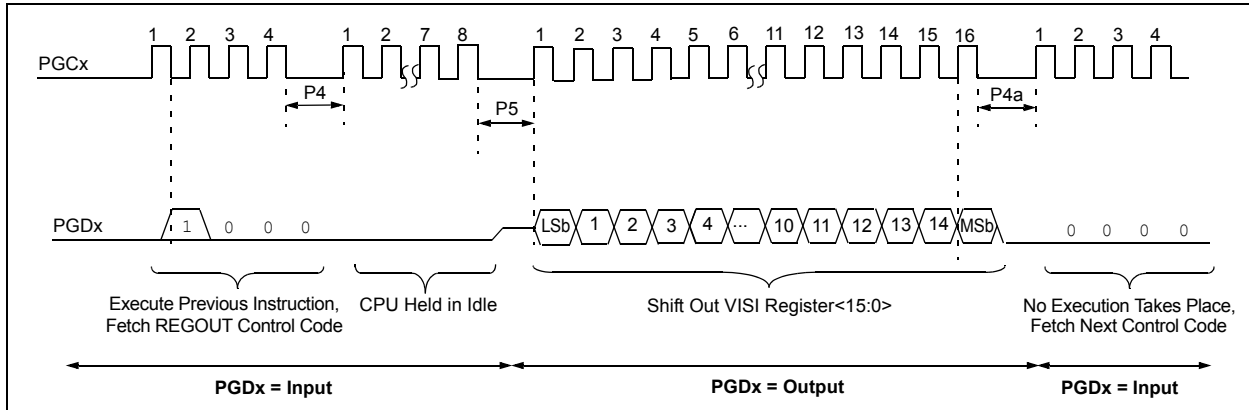
**FIGURE 5-2: PROGRAM ENTRY AFTER RESET**



**FIGURE 5-3: SIX SERIAL EXECUTION**



**FIGURE 5-4: REGOUT SERIAL EXECUTION**



# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 5.4 Flash Memory Programming in ICSP Mode

### 5.4.1 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 5-2) or write operation (Table 5-3) and initiating the programming by setting the WR control bit (NVMCON<15>).

In ICSP mode, all programming operations are self-timed. There is an internal delay between the user setting the WR control bit and the automatic clearing of the WR control bit when the programming operation is complete. Please refer to Section 8.0 “AC/DC Characteristics and Timing Requirements” for detailed information about the delays associated with various programming operations.

TABLE 5-3: NVMCON WRITE OPERATIONS

NVMCON Value	Write Operation
0x4001	Program 1 row (64 instruction words) of code memory or executive memory.
0x4000	Write a Configuration register byte.
0x4003	Program a code memory word.

### 5.4.2 STARTING AND STOPPING A PROGRAMMING CYCLE

The WR bit (NVMCON<15>) is used to start an erase or write cycle. Setting the WR bit initiates the programming cycle.

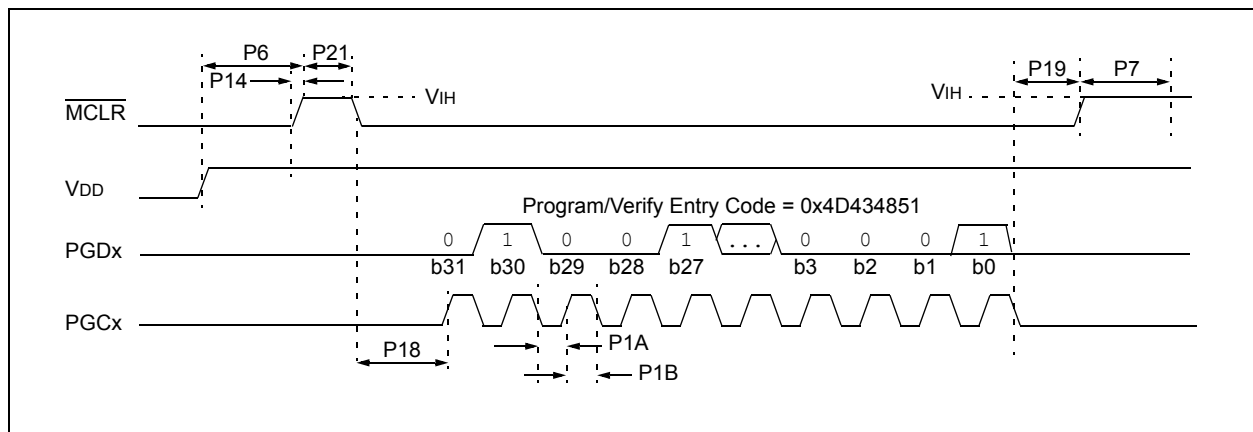
All erase and write cycles are self-timed. The WR bit should be polled to determine if the erase or write cycle has been completed. Starting a programming cycle is performed as follows:

```
BSET NVMCON, #WR
```

TABLE 5-2: NVMCON ERASE OPERATIONS

NVMCON Value	Erase Operation
0x404F	Erase all code memory, executive memory and CodeGuard™ Configuration registers (does not erase Unit ID or Device ID registers).
0x404D	Erase General Segment and FGS Configuration register.
0x404C	Erase Secure Segment and FSS Configuration register. This operation will also erase the General Segment and FGS Configuration register.
0x4042	Erase a page of code memory or executive memory.

FIGURE 5-5: ENTERING ICSP™ MODE



# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## REGISTER 5-1: NVMCON: FLASH MEMORY CONTROL REGISTER

R/SO-0 <sup>(1)</sup>	R/W-0 <sup>(1)</sup>	R/W-0 <sup>(1)</sup>	U-0	U-0	U-0	U-0	U-0
WR	WREN	WRERR	—	—	—	—	—
bit 15							bit 8

U-0	R/W-0 <sup>(1)</sup>	U-0	U-0	R/W-0 <sup>(1)</sup>	R/W-0 <sup>(1)</sup>	R/W-0 <sup>(1)</sup>	R/W-0 <sup>(1)</sup>
—	ERASE	—	—	NVMOP<3:0> <sup>(2)</sup>			
bit 7							bit 0

<b>Legend:</b>	SO = Satiabie only bit		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

- bit 15      **WR:** Write Control bit  
 1 = Initiates a Flash memory program or erase operation. The operation is self-timed and the bit is cleared by hardware once operation is complete  
 0 = Program or erase operation is complete and inactive
- bit 14      **WREN:** Write Enable bit  
 1 = Enable Flash program/erase operations  
 0 = Inhibit Flash program/erase operations
- bit 13      **WRERR:** Write Sequence Error Flag bit  
 1 = An improper program or erase sequence attempt or termination has occurred (bit is set automatically on any set attempt of the WR bit)  
 0 = The program or erase operation completed normally
- bit 12-7    **Unimplemented:** Read as '0'
- bit 6        **ERASE:** Erase/Program Enable bit  
 1 = Perform the erase operation specified by NVMOP<3:0> on the next WR command  
 0 = Perform the program operation specified by NVMOP<3:0> on the next WR command
- bit 5-4     **Unimplemented:** Read as '0'
- bit 3-0     **NVMOP<3:0>:** NVM Operation Select bits<sup>(2)</sup>  
If ERASE = 1:  
 1111 = Memory Bulk Erase operation  
 1110 = Reserved  
 1101 = Erase General Segment  
 1100 = Erase Secure Segment  
 1011 = Reserved  
 0011 = No operation  
 0010 = Memory page erase operation  
 0001 = No operation  
 0000 = Erase a single Configuration register byte  
If ERASE = 0:  
 1111 = No operation  
 1110 = Reserved  
 1101 = No operation  
 1100 = No operation  
 1011 = Reserved  
 0011 = Memory word program operation  
 0010 = No operation  
 0001 = Memory row program operation  
 0000 = Program a single Configuration register byte

- Note 1:** These bits can only be reset on a Power-on Reset (POR).  
**2:** All other combinations of NVMOP<3:0> are unimplemented.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

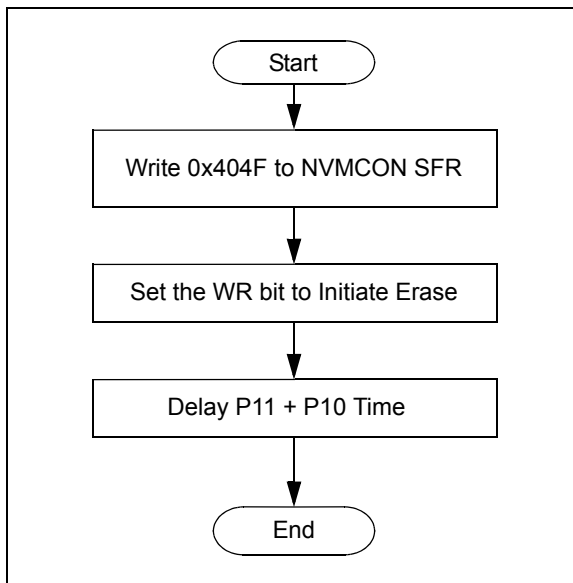
## 5.5 Erasing Program Memory

The procedure for erasing program memory (all of code memory, data memory, executive memory and code-protect bits) consists of setting NVMCON to 0x404F and then executing the programming cycle. For segment erase operations, the NVMCON value should be modified suitably, according to [Table 5-2](#).

[Figure 5-6](#) illustrates the ICSP programming process for Bulk Erasing program memory. This process includes the ICSP command code, which must be transmitted (for each instruction) Least Significant bit first, using the PGCx and PGDx pins (see [Figure 5-2](#)).

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

**FIGURE 5-6: BULK ERASE FLOW**



If a Segment Erase operation is required, Step 3 must be modified with the appropriate NVMCON value as per [Table 5-2](#).

The ability to individually erase various segments is a critical component of the CodeGuard™ Security features on dsPIC33F/PIC24H devices. An individual code segment may be erased without affecting other segments. In addition, the Configuration register corresponding to the erased code segment also gets erased. For example, the user may want to erase the code in the General Segment without erasing a bootloader located in the Boot Segment.

The Secure Segment Erase command is used to erase the Secure Segment and the FSS Configuration register. The General Segment Erase command is used to erase the General Segment and the FGS Configuration register. This command is only effective if a Boot Segment or Secure Segment has been enabled.

**Note 1:** The Boot Segment and FBS Configuration register can only be erased using a Bulk Erase.

**2:** A Secure Segment Erase operation also erases the General Segment and FGS Configuration register. This is true even if Secure Segment is present on a device but not enabled.

Before performing any segment erase operation, the programmer must first determine if the dsPIC33F/PIC24H device has defined a Boot Segment or Secure Segment, and ensure that a segment does not get overwritten by operations on any other segment.

The BSS bit field in the FBS Configuration register can be read to determine whether a Boot Segment has been defined. If a Boot Segment has already been defined (and probably already been programmed), the user must be warned about this fact. Similarly, the SSS bit field in the FSS Configuration register can be read to determine whether a Secure Segment has been defined. If a Secure Segment has already been defined (and probably already been programmed), the user must be warned about this fact.

A Bulk Erase operation is the recommended mechanism to allow a user to overwrite the Boot Segment (if one chooses to do so).

In general, the segments and CodeGuard Security-related Configuration registers should be programmed in the following order:

- FBS and Boot Segment
- FSS and Secure Segment
- FGS and General Segment

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 5-4: SERIAL INSTRUCTION EXECUTION FOR BULK ERASING CODE MEMORY**

Command (Binary)	Data (Hex)	Description
<b>Step 1: Exit the Reset vector.</b>		
0000	040200	GOTO 0x200
0000	040200	GOTO 0x200
0000	000000	NOP
<b>Step 2: Set the NVMCON to erase all program memory.</b>		
0000	2404FA	MOV #0x404F, W10
0000	883B0A	MOV W10, NVMCON
<b>Step 3: Initiate the erase cycle.</b>		
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
<b>Step 4: Wait for Bulk Erase operation to complete and make sure WR bit is clear.</b>		
—	—	Externally time 'P11' msec (see <a href="#">Section 8.0 “AC/DC Characteristics and Timing Requirements”</a> ) to allow sufficient time for the Bulk Erase operation to complete.

## 5.6 Writing Code Memory

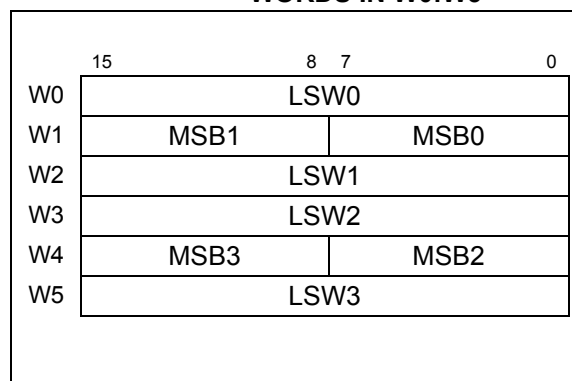
The procedure for writing code memory is similar to the procedure for writing the Configuration registers, except that 64 instruction words are programmed at a time. To facilitate this operation, working registers, W0:W5, are used as temporary holding registers for the data to be programmed.

[Table 5-5](#) shows the ICSP programming details, including the serial pattern with the ICSP command code, which must be transmitted Least Significant bit first using the PGCx and PGDx pins (see [Figure 5-2](#)). In Step 1, the Reset vector is exited. In Step 2, the NVMCON register is initialized for 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 in TBLPAG and the lower 16 bits of the destination address are stored in W7.

To minimize the programming time, the same packed instruction format that the programming executive uses is utilized (see [Figure 4-4](#)). In Step 4, four packed instruction words are stored in 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 are illustrated in [Figure 5-7](#). 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 64 instruction words at a time, Steps 4 and 5 are repeated 16 times to load all the write latches (Step 6).

After the write latches are loaded, programming is initiated by writing to the NVMCON register in Steps 7 and 8. In Step 9, the internal PC is reset to 0x200. 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 5-7: PACKED INSTRUCTION WORDS IN W0:W5**



# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 5-5: SERIAL INSTRUCTION EXECUTION FOR WRITING CODE MEMORY**

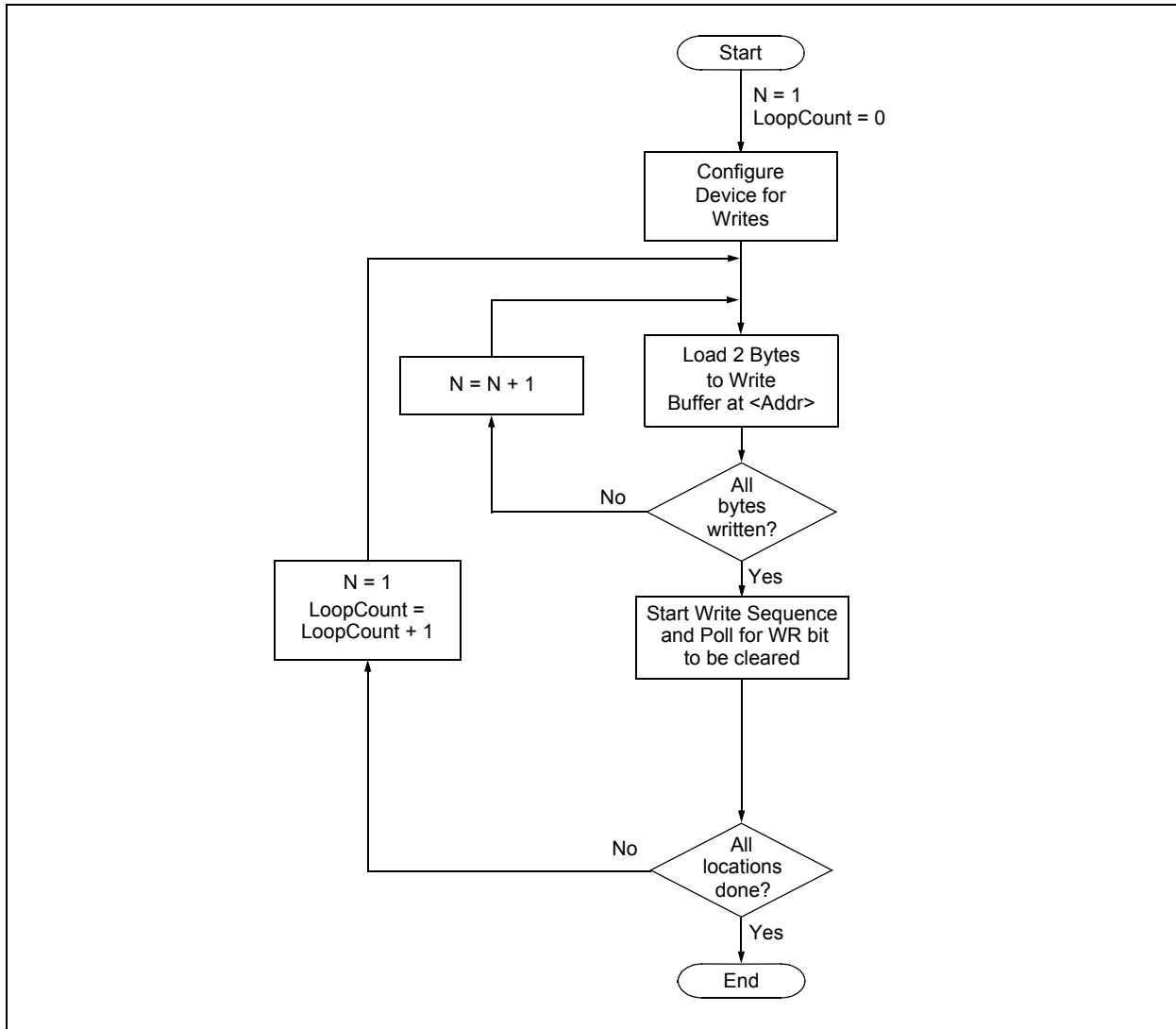
Command (Binary)	Data (Hex)	Description
<b>Step 1: Exit the Reset vector.</b>		
0000	040200	GOTO 0x200
0000	040200	GOTO 0x200
0000	000000	NOP
<b>Step 2: Set the NVMCON to program 64 instruction words.</b>		
0000	24001A	MOV #0x4001, W10
0000	883B0A	MOV W10, NVMCON
<b>Step 3: Initialize the write pointer (W7) for TBLWT instruction.</b>		
0000	200xx0	MOV #<DestinationAddress23:16>, W0
0000	880190	MOV W0, TBLPAG
0000	2xxxx7	MOV #<DestinationAddress15:0>, W7
<b>Step 4: Initialize the read pointer (W6) and load W0:W5 with the next 4 instruction words to program.</b>		
0000	2xxxx0	MOV #<LSW0>, W0
0000	2xxxx1	MOV #<MSB1:MSB0>, W1
0000	2xxxx2	MOV #<LSW1>, W2
0000	2xxxx3	MOV #<LSW2>, W3
0000	2xxxx4	MOV #<MSB3:MSB2>, W4
0000	2xxxx5	MOV #<LSW3>, W5
<b>Step 5: Set the read pointer (W6) and load the (next set of) write latches.</b>		
0000	EB0300	CLR W6
0000	000000	NOP
0000	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
<b>Step 6: Repeat steps 4-5 sixteen times to load the write latches for 64 instructions.</b>		
<b>Step 7: Initiate the write cycle.</b>		
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 5-5: SERIAL INSTRUCTION EXECUTION FOR WRITING CODE MEMORY (CONTINUED)**

Command (Binary)	Data (Hex)	Description
<b>Step 8:</b> Wait for Row Program operation to complete and make sure WR bit is clear.		
—	—	Externally time 'P13' msec (see <a href="#">Section 8.0 “AC/DC Characteristics and Timing Requirements”</a> ) to allow sufficient time for the Row Program operation to complete.
0000	803B00	MOV NVMCON, W0
0000	883C20	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	040200	GOTO 0x200
0000	000000	NOP
—	—	Repeat until the WR bit is clear.
<b>Step 9:</b> Repeat steps 3-8 until all code memory is programmed.		

**FIGURE 5-8: PROGRAM CODE MEMORY FLOW**



# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 5.7 Writing Configuration Memory

The 8-bit Configuration registers are programmable, one register at a time. The default programming values recommended for the Configuration registers are shown in [Table 5-6](#). The recommended default FOSCSEL value is 0x00, which selects the FRC clock oscillator setting.

The FBS, FGS and FGS Configuration registers are special since they enable code protection for the device. For security purposes, once any bit in these registers is programmed to '0' (to enable code protection), it can only be set back to '1' by performing a Bulk Erase as described in [Section 5.5 "Erasing Program Memory"](#). Programming any of these bits from a '0' to '1' is not possible, but they may be programmed from a '1' to '0' to enable code protection.

**TABLE 5-6: DEFAULT CONFIGURATION REGISTER VALUES**

dsPIC33FJ06GS101/102/202 AND dsPIC33FJ16GS402/404/502/504		
Address	Name	Default Value
0xF80000	FBS	0x0F
0xF80004	FGS	0x07
0xF80006	FOSCSEL	0x87
0xF80008	FOSC	0xE7
0xF8000A	FWDT	0xDF
0xF8000C	FPOR	0x0F
0xF8000E	FICD	0xE3
0xF80010	FUID0	0xFF
0xF80012	FUID1	0xFF
dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202 AND PIC24HJ12GP201/202		
Address	Name	Default Value
0xF80000	FBS	0x0F
0xF80004	FGS	0x07
0xF80006	FOSCSEL	0x87
0xF80008	FOSC	0xE7
0xF8000A	FWDT	0xDF
0xF8000C	FPOR	0xF7
0xF8000E	FICD	0xE3
0xF80010	FUID0	0xFF
0xF80012	FUID1	0xFF
0xF80014	FUID2	0xFF
0xF80016	FUID3	0xFF
dsPIC33FJ32GP202/204, dsPIC33FJ32MC202/204 AND PIC24HJ32GP202/204 dsPIC33FJ16GP304, dsPIC33FJ16MC304 AND PIC24HJ16GP304		
Address	Name	Default Value
0xF80000	FBS	0x0F
0xF80004	FGS	0x07
0xF80006	FOSCSEL	0x87
0xF80008	FOSC	0xE7
0xF8000A	FWDT	0xDF
0xF8000C	FPOR	0xF7
0xF8000E	FICD	0xE3
0xF80010	FUID0	0xFF
0xF80012	FUID1	0xFF
0xF80014	FUID2	0xFF
0xF80016	FUID3	0xFF

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 5-6: DEFAULT CONFIGURATION REGISTER VALUES (CONTINUED)**

dsPIC33FJ32GP302/304, dsPIC33FJ32MC302/304 AND PIC24HJ32GP302/304 dsPIC33FJ64GPX02/X04, dsPIC33FJ64MCX02/X04 AND PIC24HJ128GPX02/X04 dsPIC33FJ128GPX02/X04, dsPIC33FJ128MCX02/X04 AND PIC24HJ128GPX02/X04		
Address	Name	Default Value
0xF80000	FBS	0x0F
0xF80002	FSS	0xCF/0xFF <sup>(1)</sup>
0xF80004	FGS	0x07
0xF80006	FOSCSEL	0x87
0xF80008	FOSC	0xE7
0xF8000A	FWDT	0xDF
0xF8000C	FPOR	0xF7
0xF8000E	FICD	0xE3
0xF80010	FUID0	0xFF
0xF80012	FUID1	0xFF
0xF80014	FUID2	0xFF
0xF80016	FUID3	0xFF

**Note 1:** For dsPIC33FJ32GP302/304 devices, the default value for the FSS Configuration register is 0xFF.

dsPIC33FJ32GS406/606/608/610 AND dsPIC33FJ64GS406/606/608/610		
Address	Name	Default Value
0xF80000	FBS	0x0F
0xF80004	FGS	0x07
0xF80006	FOSCSEL	0x87
0xF80008	FOSC	0xC7
0xF8000A	FWDT	0xDF
0xF8000C	FPOR	0x67
0xF8000E	FICD	0xE3
0xF80010	FCMP	0x3F
dsPIC33FJXXXGPX06A/X08A/X10A, dsPIC33FJXXXMCX06A/X08A/X10A AND PIC24HJXXXGPX06A/X08A/X10A		
Address	Name	Default Value
0xF80000	FBS	0xCF
0xF80002	FSS	0xCF
0xF80004	FGS	0x07
0xF80006	FOSCSEL	0x00
0xF80008	FOSC	0xC7
0xF8000A	FWDT	0xFF
0xF8000C	FPOR	0xE7
0xF8000E	FICD	0xE3
0xF80010	FUID0	0xFF
0xF80012	FUID1	0xFF
0xF80014	FUID2	0xFF
0xF80016	FUID3	0xFF

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

---

---

Table 5-7 shows the ICSP programming details for clearing the Configuration registers. In Step 1, the Reset vector is exited. In Step 2, the write pointer (W7) is loaded with 0x0000, which is the original destination address (in TBLPAG, 0xF8 of program memory). In Step 3, the NVMCON is set to program one Configuration register. In Step 4, the TBLPAG register is initialized to 0xF8 for writing to the Configuration registers. In Step 5, the value to write to each Configuration register is loaded to W0. In Step 6, the Configuration register data is written to the write latch using the `TBLWTL` instruction. In Steps 7 and 8, the programming cycle is initiated. In Step 9, the internal PC is set to 0x200 as a safety measure to prevent the PC from incrementing into unimplemented memory. Lastly, Steps 4-9 are repeated until all twelve Configuration registers are written.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 5-7: SERIAL INSTRUCTION EXECUTION FOR WRITING CONFIGURATION REGISTERS**

Command (Binary)	Data (Hex)	Description
<b>Step 1:</b> Exit the Reset vector.		
0000	040200	GOTO 0x200
0000	040200	GOTO 0x200
0000	000000	NOP
<b>Step 2:</b> Initialize the write pointer (W7) for the TBLWT instruction.		
0000	200007	MOV #0x0000, W7
<b>Step 3:</b> Set the NVMCON register to program one Configuration register.		
0000	24000A	MOV #0x4000, W10
0000	883B0A	MOV W10, NVMCON
<b>Step 4:</b> Initialize the TBLPAG register.		
0000	200F80	MOV #0xF8, W0
0000	880190	MOV W0, TBLPAG
<b>Step 5:</b> Load the Configuration register data to W6.		
0000	2xxxx0	MOV #<CONFIG_VALUE>, W0
<b>Step 6:</b> Write the Configuration register data to the write latch and increment the write pointer.		
0000	BB1B80	TBLWTL W0, [W7++]
0000	000000	NOP
0000	000000	NOP
<b>Step 7:</b> Initiate the write cycle.		
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
<b>Step 8:</b> Wait for the Configuration Register Write operation to complete and make sure WR bit is clear.		
—	—	Externally time 'P20' msec (see <a href="#">Section 8.0 "AC/DC Characteristics and Timing Requirements"</a> ) to allow sufficient time for the Configuration Register Write operation to complete.
0000	803B00	MOV NVMCON, W0
0000	883C20	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	040200	GOTO 0x200
0000	000000	NOP
—	—	Repeat until the WR bit is clear.
<b>Step 9:</b> Repeat steps 5-8 until all twelve Configuration registers are written.		

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 5.8 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.

Table 5-8 shows the ICSP programming details for reading code memory. In Step 1, the Reset vector is exited. In Step 2, the 24-bit starting source address for reading is loaded into the `TBLPAG` register and `W6` register. The upper byte of the starting source address is stored in `TBLPAG` and the lower 16 bits of the source address are stored in `W6`.

To minimize the reading time, the packed instruction word format that was utilized for writing is also used for reading (see Figure 5-7). In Step 3, the write pointer, `W7`, is initialized. In Step 4, two instruction words are read from code memory and clocked out of the device, through the `VISI` register, using the `REGOUT` command. Step 4 is repeated until the desired amount of code memory is read.

**TABLE 5-8: SERIAL INSTRUCTION EXECUTION FOR READING CODE MEMORY**

Command (Binary)	Data (Hex)	Description
<b>Step 1: Exit the Reset vector.</b>		
0000	040200	GOTO 0x200
0000	040200	GOTO 0x200
0000	000000	NOP
<b>Step 2: Initialize TBLPAG and the read pointer (W6) for TBLRD instruction.</b>		
0000	200xx0	MOV #<SourceAddress23:16>, W0
0000	880190	MOV W0, TBLPAG
0000	2xxxx6	MOV #<SourceAddress15:0>, W6
<b>Step 3: Initialize the write pointer (W7) and store the next four locations of code memory to W0:W5.</b>		
0000	EB0380	CLR W7
0000	000000	NOP
0000	BA1B96	TBLRDL [W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BADBB6	TBLRDH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BADBD6	TBLRDH.B [++W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BA1BB6	TBLRDL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BA1B96	TBLRDL [W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BADBB6	TBLRDH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BADBD6	TBLRDH.B [++W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BA0BB6	TBLRDL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 5-8: SERIAL INSTRUCTION EXECUTION FOR READING CODE MEMORY (CONTINUED)**

Command (Binary)	Data (Hex)	Description
<b>Step 4:</b> 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
<b>Step 5:</b> Repeat step 4 until all desired code memory is read.		
<b>Step 6:</b> Reset device internal PC.		
0000	040200	GOTO 0x200
0000	000000	NOP

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 5.9 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 (with the upper byte read being all '0's) instead of 24-bit words. Since there are twelve Configuration registers, they are read one register at a time.

Table 5-9 shows the ICSP programming details for reading all of the configuration memory. Note that the TBLPAG register is hard coded to 0xF8 (the upper byte address of configuration memory) and the read pointer, W6, is initialized to 0x0000.

**TABLE 5-9: SERIAL INSTRUCTION EXECUTION FOR READING ALL CONFIGURATION MEMORY**

Command (Binary)	Data (Hex)	Description
<b>Step 1: Exit the Reset vector.</b>		
0000	040200	GOTO 0x200
0000	040200	GOTO 0x200
0000	000000	NOP
<b>Step 2: Initialize TBLPAG, the read pointer (W6) and the write pointer (W7) for TBLRD instruction.</b>		
0000	200F80	MOV #0xF8, W0
0000	880190	MOV W0, TBLPAG
0000	EB0300	CLR W6
0000	207847	MOV #VISI, W7
0000	000000	NOP
<b>Step 3: Read the Configuration register and write it to the VISI register (located at 0x784) and clock out the VISI register using the REGOUT command.</b>		
0000	BA0BB6	TBLRDL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
<b>Step 4: Repeat step 3 twelve times to read all the Configuration registers.</b>		
<b>Step 5: Reset device internal PC.</b>		
0000	040200	GOTO 0x200
0000	000000	NOP

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

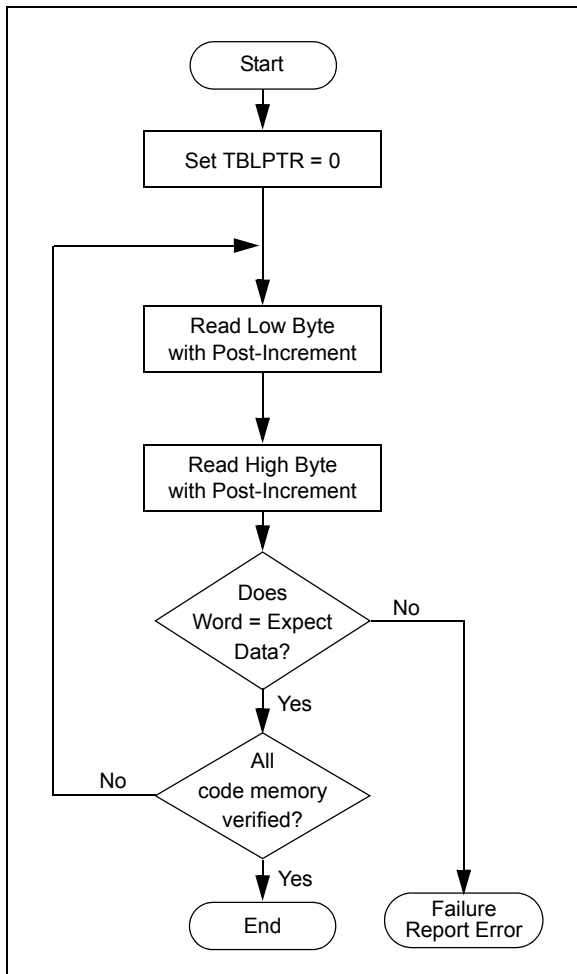
## 5.10 Verify Code Memory and Configuration Word

The verify step involves reading back the code memory space and comparing it against the copy held in the programmer's buffer. The Configuration registers are verified with the rest of the code.

The verify process is illustrated in Figure 5-9. Memory reads occur a single byte at a time, so two bytes must be read to compare against the word in the programmer's buffer. Refer to Section 5.8 "Reading Code Memory" for implementation details of reading code memory.

**Note:** Because the Configuration registers include the device code protection bit, code memory should be verified immediately after writing, if the code protection is enabled. This is because the device will not be readable or verifiable if a device Reset occurs after the code-protect bit in the FGS Configuration register has been cleared.

FIGURE 5-9: VERIFY CODE MEMORY FLOW



## 5.11 Reading the Application ID Word

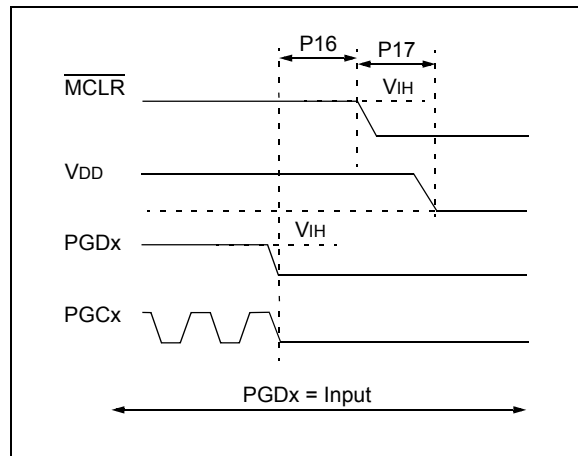
The Application ID Word is stored at address 0x8007F0 in executive code memory. To read this memory location, you must use the SIX control code to move this program memory location to the VISI register. Then, the REGOUT control code must be used to clock the contents of the VISI register out of the device. The corresponding control and instruction codes that must be serially transmitted to the device to perform this operation are shown in Table 5-10.

After the programmer has clocked out the Application ID Word, it must be inspected. If the application ID has the value 0xCB, the programming executive is resident in memory and the device can be programmed using the mechanism described in Section 3.0 "Device Programming – Enhanced ICSP". 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 6.0 "Programming the Programming Executive to Memory".

## 5.12 Exiting ICSP Mode

Exiting Program/Verify mode is done by removing  $V_{IH}$  from MCLR, as illustrated in Figure 5-10. The only requirement for exit is that an interval P16 should elapse between the last clock and program signals on PGCx and PGDx before removing  $V_{IH}$ .

FIGURE 5-10: EXITING ICSP™ MODE



# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 5-10: SERIAL INSTRUCTION EXECUTION FOR READING THE APPLICATION ID WORD**

Command (Binary)	Data (Hex)	Description
<b>Step 1: Exit the Reset vector.</b>		
0000	040200	GOTO 0x200
0000	040200	GOTO 0x200
0000	000000	NOP
<b>Step 2: Initialize TBLPAG and the read pointer (W0) for TBLRD instruction.</b>		
0000	200800	MOV #0x80, W0
0000	880190	MOV W0, TBLPAG
0000	207F00	MOV #0x7FO, W0
0000	207841	MOV #VISI, W1
0000	000000	NOP
0000	BA0890	TBLRDL [W0], [W1]
0000	000000	NOP
0000	000000	NOP
<b>Step 3: Output the VISI register using the REGOUT command.</b>		
0001	<VISI>	Clock out contents of the VISI register.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 6.0 PROGRAMMING THE PROGRAMMING EXECUTIVE TO MEMORY

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 64 words at a time. This control flow is summarized in [Table 6-1](#).

### 6.1 Overview

If it is determined that the programming executive is not present in executive memory (as described in [Section 3.2 “Confirming the Presence of the Programming Executive”](#)), it must be programmed into executive memory using ICSP, as described in [Section 5.0 “Device Programming – ICSP”](#).

**Note:** The programming executive must always be erased before it is programmed, as described in [Table 6-1](#).

**TABLE 6-1: PROGRAMMING THE PROGRAMMING EXECUTIVE**

Command (Binary)	Data (Hex)	Description
<b>Step 1: Exit the Reset vector and erase executive memory.</b>		
0000	040200	GOTO 0x200
0000	040200	GOTO 0x200
0000	000000	NOP
<b>Step 2: Initialize the NVMCON to erase a page of executive memory.</b>		
0000	24042A	MOV #0x4042, W10
0000	883B0A	MOV W10, NVMCON
<b>Step 3: Initiate the erase cycle, wait for erase to complete and make sure WR bit is clear.</b>		
0000	200080	MOV #0x80, W0
0000	880190	MOV W0, TBLPAG
0000	200001	MOV #0x00, W1
0000	000000	NOP
0000	BB0881	TBLWTL W1, [W1]
0000	000000	NOP
0000	000000	NOP
0000	A8E761	BSET NVMCON, #15
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
—	—	Externally time 'P12' msec (see <a href="#">Section 8.0 “AC/DC Characteristics and Timing Requirements”</a> ) to allow sufficient time for the Page Erase operation to complete.
0000	803B00	MOV NVMCON, W0
0000	883C20	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register. Repeat instruction until the WR bit is clear.
<b>Step 4: Repeat Step 3 to erase all pages of executive memory (incrementing the page erase pointer, i.e., W1 pointer should be incremented by 0x400 to point to the second page).</b>		
<b>Step 5: Initialize the NVMCON to program 64 instruction words.</b>		
0000	24001A	MOV #0x4001, W10
0000	883B0A	MOV W10, NVMCON
<b>Step 6: Initialize TBLPAG and the write pointer (W7).</b>		
0000	200800	MOV #0x80, W0
0000	880190	MOV W0, TBLPAG
0000	EB0380	CLR W7
0000	000000	NOP

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 6-1: PROGRAMMING THE PROGRAMMING EXECUTIVE (CONTINUED)**

Command (Binary)	Data (Hex)	Description
<b>Step 7:</b> 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
<b>Step 8:</b> Set the read pointer (W6) and load the (next four 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	BEBBB6	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	BEBBB6	TBLWTH.B[W6++], [++W7]
0000	000000	NOP
0000	000000	NOP
0000	BB1BB6	TBLWTL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
<b>Step 9:</b> Repeat Steps 7-8 sixteen times to load the write latches for the 64 instructions.		
<b>Step 10:</b> Initiate the programming cycle.		
0000	A8E761	BSET NVMCON, #15
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 6-1: PROGRAMMING THE PROGRAMMING EXECUTIVE (CONTINUED)**

Command (Binary)	Data (Hex)	Description
<b>Step 11:</b> Wait for the Row Program operation to complete.		
—	—	Externally time 'P13' msec (see <a href="#">Section 8.0 “AC/DC Characteristics and Timing Requirements”</a> ) to allow sufficient time for the Row Program operation to complete.
0000	803B00	MOV NVMCON, W0
0000	883C20	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	040200	GOTO 0x200
0000	000000	NOP
—	—	Repeat until the WR bit is clear.
<b>Step 12:</b> Repeat Steps 7-11 until all 32 rows of executive memory are programmed.		

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 6.2 Programming Verification

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

Reading the contents of executive memory can be performed using the same technique described in [Section 5.8 “Reading Code Memory”](#). A procedure for reading executive memory is shown in [Table 6-2](#). Note that in Step 2, the TBLPAG register is set to 0x80, such that executive memory may be read.

**TABLE 6-2: READING EXECUTIVE MEMORY**

Command (Binary)	Data (Hex)	Description
<b>Step 1: Exit the Reset vector.</b>		
0000	040200	GOTO 0x200
0000	040200	GOTO 0x200
0000	000000	NOP
<b>Step 2: Initialize TBLPAG and the read pointer (W6) for TBLRD instruction.</b>		
0000	200800	MOV #0x80, W0
0000	880190	MOV W0, TBLPAG
0000	EB0300	CLR W6
<b>Step 3: Initialize the write pointer (W7) and store the next four locations of code memory to W0:W5.</b>		
0000	EB0380	CLR W7
0000	000000	NOP
0000	BA1B96	TBLRDL [W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BADBB6	TBLRDH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BADBD6	TBLRDH.B [++W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BA1BB6	TBLRDL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BA1B96	TBLRDL [W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BADBB6	TBLRDH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BADBD6	TBLRDH.B [++W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BA0BB6	TBLRDL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 6-2: READING EXECUTIVE MEMORY (CONTINUED)**

Command (Binary)	Data (Hex)	Description
<b>Step 4:</b> 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
<b>Step 5:</b> Reset the device internal PC.		
0000	040200	GOTO 0x200
0000	000000	NOP
<b>Step 6:</b> Repeat Steps 4-5 until all 2048 instruction words of executive memory are read.		

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 7.0 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 READC command. This region of memory is read-only and can also be read when code protection is enabled.

Table 7-1 lists the identification information for each device.

Table 7-2 shows the Device ID registers.

Register 7-1, Register 7-2 and Register 7-3 provide the JTAG ID Type registers.

**TABLE 7-1: DEVICE IDs AND REVISION**

Device	DEVID Register Value	Application ID	DEVREV Register Value and Silicon Revision	JTAG ID	
dsPIC33FJ06GS101	0x0C00	0xCB	0x3000 – A0 Revision	<a href="#">Register 7-1</a> <a href="#">Register 7-1</a> <a href="#">Register 7-1</a> <a href="#">Register 7-1</a> <a href="#">Register 7-1</a>	
dsPIC33FJ06GS102	0x0C01	0xCB	0x3001 – A1 Revision		
dsPIC33FJ06GS202	0x0C02	0xCB	0x3002 – A2 Revision		
dsPIC33FJ16GS402	0x0C04	0xCB	0x3003 – A3 Revision		
dsPIC33FJ16GS404	0x0C06	0xCB	0x3004 – A4 Revision		
dsPIC33FJ16GS502	0x0C03	0xCB			
dsPIC33FJ16GS504	0x0C05	0xCB			
dsPIC33FJ12GP201	0x0802	0xCB	0x3001 – A2 Revision		<a href="#">Register 7-1</a> <a href="#">Register 7-1</a> <a href="#">Register 7-1</a> <a href="#">Register 7-1</a> <a href="#">Register 7-1</a>
dsPIC33FJ12GP202	0x0803	0xCB	0x3002 – A3 Revision		
dsPIC33FJ12MC201	0x0800	0xCB	0x3003 – A4 Revision		
dsPIC33FJ12MC202	0x0801	0xCB	0x3005 – A5 Revision		
PIC24HJ12GP201	0x080A	0xCB			
PIC24HJ12GP202	0x080B	0xCB			
dsPIC33FJ16GP304	0x0F07	0xCB	0x3001 – A2 Revision	<a href="#">Register 7-1</a> <a href="#">Register 7-1</a> <a href="#">Register 7-1</a> <a href="#">Register 7-1</a> <a href="#">Register 7-1</a>	
dsPIC33FJ16MC304	0x0F03	0xCB	0x3002 – A3 Revision		
PIC24HJ16GP304	0x0F17	0xCB	0x3004 – A4 Revision		
dsPIC33FJ32GP202	0x0F0D	0xCB	0x3005 – A5 Revision		
dsPIC33FJ32GP204	0x0F0F	0xCB			
dsPIC33FJ32MC202	0x0F09	0xCB			
dsPIC33FJ32MC204	0x0F0B	0xCB			
PIC24HJ32GP202	0x0F1D	0xCB			
PIC24HJ32GP204	0x0F1F	0xCB			
dsPIC33FJ64GP206	0x00C1	0xCB	0x3002 – A2 Revision		<a href="#">Register 7-2</a> <a href="#">Register 7-2</a> <a href="#">Register 7-3</a>
dsPIC33FJ64GP306	0x00CD	0xCB	0x3004 – A3 Revision		
dsPIC33FJ64GP310	0x00CF	0xCB	0x3040 – A4 Revision		
dsPIC33FJ64GP706	0x00D5	0xCB			
dsPIC33FJ64GP708	0x00D6	0xCB			
dsPIC33FJ64GP710	0x00D7	0xCB			
dsPIC33FJ64MC506	0x0089	0xCB			
dsPIC33FJ64MC508	0x008A	0xCB			
dsPIC33FJ64MC510	0x008B	0xCB			
dsPIC33FJ64MC706	0x0091	0xCB			
dsPIC33FJ64MC710	0x0097	0xCB			
PIC24HJ64GP206	0x0041	0xCB			
PIC24HJ64GP210	0x0047	0xCB			
PIC24HJ64GP506	0x0049	0xCB			
PIC24HJ64GP510	0x004B	0xCB			

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 7-1: DEVICE IDs AND REVISION (CONTINUED)**

Device	DEVID Register Value	Application ID	DEVREV Register Value and Silicon Revision	JTAG ID
dsPIC33FJ128GP206	0x00D9	0xCB	0x3002 – A2 Revision	Register 7-2 Register 7-2 Register 7-3
dsPIC33FJ128GP306	0x00E5	0xCB	0x3004 – A3 Revision	
dsPIC33FJ128GP310	0x00E7	0xCB	0x3040 – A4 Revision	
dsPIC33FJ128GP706	0x00ED	0xCB		
dsPIC33FJ128GP708	0x00EE	0xCB		
dsPIC33FJ128GP710	0x00EF	0xCB		
dsPIC33FJ128MC506	0x00A1	0xCB		
dsPIC33FJ128MC510	0x00A3	0xCB		
dsPIC33FJ128MC706	0x00A9	0xCB		
dsPIC33FJ128MC708	0x00AE	0xCB		
dsPIC33FJ128MC710	0x00AF	0xCB		
PIC24HJ128GP206	0x005D	0xCB		
PIC24HJ128GP210	0x005F	0xCB		
PIC24HJ128GP306	0x0065	0xCB		
PIC24HJ128GP310	0x0067	0xCB		
PIC24HJ128GP506	0x0061	0xCB		
PIC24HJ128GP510	0x0063	0xCB		
dsPIC33FJ256GP506	0x00F5	0xCB	0x3002 – A2 Revision	Register 7-2 Register 7-2 Register 7-3
dsPIC33FJ256GP510	0x00F7	0xCB	0x3004 – A3 Revision	
dsPIC33FJ256GP710	0x00FF	0xCB	0x3040 – A4 Revision	
dsPIC33FJ256MC510	0x00B7	0xCB		
dsPIC33FJ256MC710	0x00BF	0xCB		
PIC24HJ256GP206	0x0071	0xCB		
PIC24HJ256GP210	0x0073	0xCB		
PIC24HJ256GP610	0x007B	0xCB		
dsPIC33FJ32GP302	0x0605	0xCB	0x3001 – A1 Revision	Register 7-1 Register 7-1 Register 7-1 Register 7-1
dsPIC33FJ32GP304	0x0607	0xCB	0x3002 – A2 Revision	
dsPIC33FJ32MC302	0x0601	0xCB	0x3002 – A3 Revision	
dsPIC33FJ32MC304	0x0603	0xCB	0x3003 – A4 Revision	
dsPIC33FJ64GP202	0x0615	0xCB		
dsPIC33FJ64GP204	0x0617	0xCB		
dsPIC33FJ64GP802	0x061D	0xCB		
dsPIC33FJ64GP804	0x061F	0xCB		
dsPIC33FJ64MC202	0x0611	0xCB		
dsPIC33FJ64MC204	0x0613	0xCB		
dsPIC33FJ64MC802	0x0619	0xCB		
dsPIC33FJ64MC804	0x061B	0xCB		
dsPIC33FJ128GP202	0x0625	0xCB		
dsPIC33FJ128GP204	0x0627	0xCB		
dsPIC33FJ128GP802	0x062D	0xCB		
dsPIC33FJ128GP804	0x062F	0xCB		
dsPIC33FJ128MC202	0x0621	0xCB		
dsPIC33FJ128MC204	0x0623	0xCB		
dsPIC33FJ128MC802	0x0629	0xCB		



# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

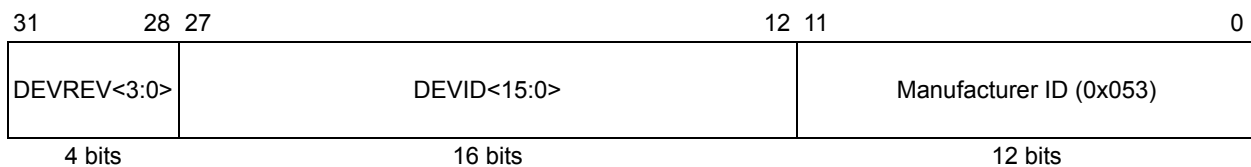
**TABLE 7-1: DEVICE IDs AND REVISION (CONTINUED)**

Device	DEVID Register Value	Application ID	DEVREV Register Value and Silicon Revision	JTAG ID
dsPIC33FJ256GP506A	0x07F5	0xCB	0x3002 – A2 Revision 0x3003 – A3 Revision	Register 7-1 Register 7-1
dsPIC33FJ256GP510A	0x07F7	0xCB		
dsPIC33FJ256GP710A	0x07FF	0xCB		
dsPIC33FJ256MC510A	0x07B7	0xCB		
dsPIC33FJ256MC710A	0x07BF	0xCB		
PIC24HJ256GP206A	0x0771	0xCB		
PIC24HJ256GP210A	0x0773	0xCB		
PIC24HJ256GP610A	0x077B	0xCB		
dsPIC33FJ32GS406	0x4000	0xCB	0x3000 – A0 Revision	Register 7-1
dsPIC33FJ64GS406	0x4001	0xCB		
dsPIC33FJ32GS606	0x4002	0xCB		
dsPIC33FJ64GS606	0x4003	0xCB		
dsPIC33FJ32GS608	0x4004	0xCB		
dsPIC33FJ64GS608	0x4005	0xCB		
dsPIC33FJ32GS610	0x4006	0xCB		
dsPIC33FJ64GS610	0x4007	0xCB		

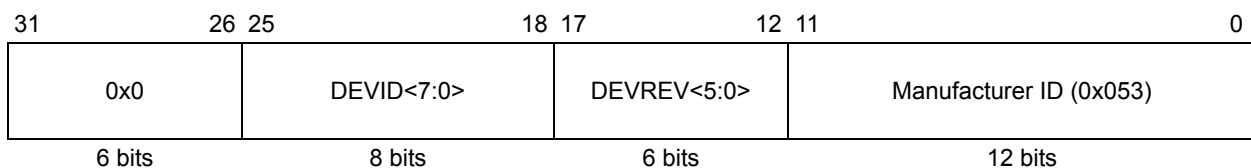
**TABLE 7-2: dsPIC33F/PIC24H PROGRAMMING SPECIFICATION DEVICE ID REGISTERS**

Address	Name	Bit															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF0000	DEVID	DEVID Value															
0xFF0002	DEVREV	DEVREV Value															

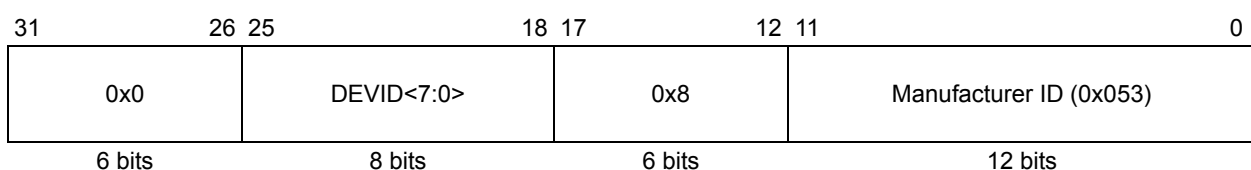
**REGISTER 7-1: JTAG ID REGISTER TYPE 1**



**REGISTER 7-2: JTAG ID REGISTER TYPE 2**



**REGISTER 7-3: JTAG ID REGISTER TYPE 3**



# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## 8.0 AC/DC CHARACTERISTICS AND TIMING REQUIREMENTS

Table 8-1 lists the AC/DC characteristics and timing requirements.

**TABLE 8-1: AC/DC CHARACTERISTICS AND TIMING REQUIREMENTS**

Standard Operating Conditions						
Operating Temperature: -40°C-85°C. Programming at 25°C is recommended.						
Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
D111	VDD	Supply Voltage During Programming	3.0	3.60	V	Normal programming <sup>(1)</sup>
D112	IPP	Programming Current on $\overline{\text{MCLR}}$	—	5	μA	—
D113	IDDP	Supply Current During Programming	—	2	mA	—
D031	VIL	Input Low Voltage	VSS	0.2 VDD	V	—
D041	VIH	Input High Voltage	0.8 VDD	VDD	V	—
D080	VOL	Output Low Voltage	—	0.6	V	IO <sub>L</sub> = 8.5 mA @ 3.6V
D090	VOH	Output High Voltage	VDD - 0.7	—	V	IO <sub>H</sub> = -3.0 mA @ 3.6V
D012	CIO	Capacitive Loading on I/O pin (PGDx)	—	50	pF	To meet AC specifications
D013	CF	Filter Capacitor Value on VCAP	1	10	μF	Required for controller core
P1	TPGC	Serial Clock (PGCx) Period (ICSP™)	200	—	ns	—
P1	TPGCL	Serial Clock (PGCx) Period (Enhanced ICSP)	500	—	ns	—
P1A	TPGCL	Serial Clock (PGCx) Low Time (ICSP)	80	—	ns	—
P1A	TPGCL	Serial Clock (PGCx) Low Time (Enhanced ICSP)	200	—	ns	—
P1B	TPGCH	Serial Clock (PGCx) High Time (ICSP)	80	—	ns	—
P1B	TPGCH	Serial Clock (PGCx) High Time (Enhanced ICSP)	200	—	ns	—
P2	TSET1	Input Data Setup Time to Serial Clock ↓	15	—	ns	—
P3	THLD1	Input Data Hold Time from PGCx ↓	15	—	ns	—
P4	TDLY1	Delay between 4-bit Command and Command Operand	40	—	ns	—
P4A	TDLY1A	Delay between Command Operand and Next 4-bit Command	40	—	ns	—
P5	TDLY2	Delay between Last PGCx ↓ of Command to First PGCx ↑ of Read of Data Word	20	—	ns	—
P6	TSET2	VDD ↑ Setup Time to $\overline{\text{MCLR}}$ ↑	100	—	ns	—
P7	THLD2	Input Data Hold Time from $\overline{\text{MCLR}}$ ↑	25	—	ms	—
P8	TDLY3	Delay between Last PGCx ↓ of Command Byte to PGDx ↑ by Programming Executive	12	—	μs	—
P9a	TDLY4	Programming Executive Command Processing Time	10	—	μs	—

**Note 1:** VDD must also be supplied to the AVDD pins during programming. AVDD and AVSS should always be within ±0.3V of VDD and VSS, respectively.

**2:** Time depends on the FRC accuracy and the value of the FRC Oscillator tuning register. Refer to “**Electrical Characteristics**” section in the specific device data sheet.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE 8-1: AC/DC CHARACTERISTICS AND TIMING REQUIREMENTS (CONTINUED)**

<b>Standard Operating Conditions</b>						
Operating Temperature: -40°C-85°C. Programming at 25°C is recommended.						
Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
P9b	TDLY5	Delay between PGDx ↓ by Programming Executive to PGDx Released by Programming Executive	15	23	μs	—
P10	TDLY6	PGCx Low Time After Programming	400	—	ns	—
P11	TDLY7	Bulk Erase Time	330	—	ms	See <b>Note 2</b>
P12	TDLY8	Page Erase Time	19.5	—	ms	See <b>Note 2</b>
P13	TDLY9	Row Programming Time	1.28	—	ms	See <b>Note 2</b>
P14	TR	MCLR Rise Time to Enter ICSP mode	—	1.0	μs	—
P15	TVALID	Data Out Valid from PGCx ↑	10	—	ns	—
P16	TDLY10	Delay between Last PGCx ↓ and MCLR ↓	0	—	s	—
P17	THLD3	MCLR ↓ to VDD ↓	—	100	ns	—
P18	TKEY1	Delay from First MCLR ↓ to First PGCx ↑ for Key Sequence on PGDx	1	—	μs	—
P19	TKEY2	Delay from Last PGCx ↓ for Key Sequence on PGDx to Second MCLR ↑	25	—	ns	—
P20	TDLY11	Maximum Wait Time for Configuration Register Programming	—	25	ms	—
P21	TMCLRH	MCLR High Time	—	500	μs	—

**Note 1:** VDD must also be supplied to the AVDD pins during programming. AVDD and AVSS should always be within ±0.3V of VDD and VSS, respectively.

**2:** Time depends on the FRC accuracy and the value of the FRC Oscillator tuning register. Refer to “**Electrical Characteristics**” section in the specific device data sheet.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

---

## APPENDIX A: HEX FILE FORMAT

Flash programmers process the standard HEX format used by the Microchip development tools. The format supported is the Intel® HEX32 Format (INHX32). Please refer to Appendix A in the “MPASM™ User’s Guide” (DS33014) for more information about hex file formats.

The basic format of the hex file is:

```
:BBAAAATTHHHH...HHHHCC
```

Each data record begins with a 9-character prefix and always ends with a 2-character checksum. All records begin with ‘:’ regardless of the format. The individual elements are described below.

- **BB** – is a two-digit hexadecimal byte count representing the number of data bytes that appear on the line. Divide this number by two to get the number of words per line.
- **AAAA** – is a four-digit hexadecimal address representing the starting address of the data record. Format is high byte first followed by low byte. The address is doubled because this format only supports 8 bits. Divide the value by two to find the real device address.
- **TT** – is a two-digit record type that will be ‘00’ for data records, ‘01’ for end-of-file records and ‘04’ for extended-address record.
- **HHHH** – is a four-digit hexadecimal data word. Format is low byte followed by high byte. There will be  $BB/2$  data words following **TT**.
- **CC** – is a two-digit hexadecimal checksum that is the two’s complement of the sum of all the preceding bytes in the line record.

Because the Intel hex file format is byte-oriented, and the 16-bit program counter is not, program memory sections require special treatment. Each 24-bit program word is extended to 32 bits by inserting a so-called “phantom byte”. Each program memory address is multiplied by 2 to yield a byte address.

As an example, a section that is located at 0x100 in program memory will be represented in the hex file as 0x200.

The hex file will be produced with the following contents:

```
:020000040000fa
:040200003322110096
:00000001FF
```

Notice that the data record (line 2) has a load address of 0200, while the source code specified address 0x100. Note also that the data is represented in “little-endian” format, meaning the Least Significant Byte (LSB) appears first. The phantom byte appears last, just before the checksum.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

---

## APPENDIX B: DEVICE ID REGISTER SILICON ERRATA ADDENDUM

This silicon errata issue was previously published in the following documents, which are available from the Microchip web site:

- dsPIC33FJXXXGPX06/X08/X10 Rev. A2 Silicon Errata (DS80306)
- dsPIC33FJXXXMCX06/X08/X10 Rev. A2 Silicon Errata (DS80307)
- PIC24HJXXXGPX06/X08/X10 Rev. A2 Silicon Errata (DS80280)

This issue is included in this document to further assist customers.

### 1. Module: Device ID Register

On a few devices, the content of the Device ID register can change from the factory programmed default value immediately after RTSP or ICSP™ Flash programming.

As a result, development tools will not recognize these devices and will generate an error message indicating that the device ID and the device part number do not match. Additionally, some peripherals will be reconfigured and will not function as described in the device data sheet.

Refer to **Section 5. “Flash Programming”** (DS70191), of the “*dsPIC33F Family Reference Manual*” for an explanation of RTSP and ICSP Flash programming.

#### **Work around**

All RTSP and ICSP Flash programming routines must be modified as follows:

1. No word programming is allowed. Any word programming must be replaced with row programming.
2. During row programming, load write latches as described in **5.4.2.3 “Loading Write Latches”** of **Section 5. “Flash Programming”** (DS70191).
3. After latches are loaded, reload any latch location (in a given row) that has 5 LSB set to 0x18, with the original data. For example, reload one of the following latch locations with the desired data:  
0XXXXX18, 0XXXXX38, 0XXXXX58,  
0XXXXX78, 0XXXXX98, 0XXXXXB8,  
0XXXXXD8, 0XXXXXF8
4. Start row programming by setting NVMOP<3:0> = ‘0001’ (Memory row program operation) in the NVMCON register.
5. After row programming is complete, verify the contents of Flash memory.
6. If Flash verification errors are found, repeat steps 2 through 5. If Flash verification errors are found after a second iteration, report this problem to Microchip.

Steps 1 through 5 in the work around are implemented in MPLAB® IDE version 7.61 for the MPLAB ICD 2, MPLAB REAL ICE™ in-circuit emulator and MPLAB PM3 tools.

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## APPENDIX C: DIAGNOSTIC AND CALIBRATION REGISTERS

For dsPIC33FJ06GS101/102/202, dsPIC33FJ16GS402/404/502/504, dsPIC33FJ32GS406/606/608/610 and dsPIC33FJ64GS406/606/608/610 devices, the last six instruction words of program memory store diagnostic and calibration information. Any development tool that modifies this memory must take care to preserve the data contained in these registers.

Table C-1 provides an example of how the diagnostic and calibration registers are read. This table can be added to the programming executive between steps 1 and 2.

Table C-2 shows an example of how the diagnostic and calibration registers are restored once the program memory has been erased. This table can be added to the programming executive between steps 4 and 5.

**TABLE C-1: READ DIAGNOSTIC AND CALIBRATION REGISTERS**

Command (Binary)	Data (Hex)	Description
<b>Step 1: Initialize pointer to RAM.</b>		
0000	208006	MOV #0x800, W6
0000	000000	NOP
<b>Step 2: Fill section of RAM.</b>		
0000	EB9B00	SETM [W6++]
0000	000000	NOP
<b>Step 3: Repeat step two: 0x67 times.</b>		
<b>Step 4: Initialize pointers to read diagnostic and Calibration Words for storage.</b>		
0000	200800	MOV #0x80, W0
0000	880190	MOV W0, TBLPAG
0000	207F47	MOV #0x7F4, W7
0000	208AE6	MOV #0x8AE, W6
0000	000000	NOP
<b>Step 5: Read in diagnostic and Calibration Words.</b>		
0000	BA1B17	TBLRDLD[W7], [W6++]
0000	000000	NOP
0000	000000	NOP
0000	BADB37	TBLRDH.B[W7++], [W6++]
0000	000000	NOP
0000	000000	NOP
0000	BADB57	TBLRDH.B[++W7], [W6++]
0000	000000	NOP
0000	000000	NOP
0000	BA1B37	TBLRDLD[W7++], [W6++]
0000	000000	NOP
0000	000000	NOP
<b>Step 6: Repeat step 5 three times to read all 6 instruction words.</b>		

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE C-2: RESTORE DIAGNOSTIC AND CALIBRATION REGISTERS**

Command (Binary)	Data (Hex)	Description
<b>Step 1: Initialize TBLPAG and NVMCON to write stored diagnostic and Calibration Words.</b>		
0000	24001A	MOV #0x4001, W10
0000	883B0A	MOV W10, NVMCON
0000	200800	MOV #0x80, W0
0000	880190	MOV W0, TBLPAG
0000	207807	MOV #0x780, W7
0000	208006	MOV #0x800, W6
0000	000000	NOP
<b>Step 2: Write stored diagnostic and Calibration Words.</b>		
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
<b>Step 3: Repeat step 2: thirty-two times.</b>		
<b>Step 4: Initiate the write cycle.</b>		
0000	ABE761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
<b>Step 5: Wait for Row Program operation to complete and make sure WR bit is clear.</b>		
—	—	Externally time 'P13' msec (see <a href="#">Section 8.0 “AC/DC Characteristics and Timing Requirements”</a> ) to allow sufficient time for the Row Program operation to complete.
0000	803B00	MOV NVMCON, W0
0000	883C20	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register
0000	040200	GOTO 0x200
0000	000000	NOP
—	—	Repeat until the WR bit is clear

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## APPENDIX D: CHECKSUM COMPUTATION

TABLE D-1: CHECKSUM COMPUTATION

Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAAAAAA at 0x0 and Last Code Address
dsPIC33FJ06GS101	Disabled	CFGB + SUM(0:000FFF)	0xEB55	0xE957
	Enabled	CFGB	0x0353	0x0353
dsPIC33FJ06GS102	Disabled	CFGB + SUM(0:000FFF)	0xEB55	0xE957
	Enabled	CFGB	0x0353	0x0353
dsPIC33FJ06GS202	Disabled	CFGB + SUM(0:000FFF)	0xEB55	0xE957
	Enabled	CFGB	0x0353	0x0353
dsPIC33FJ16GS402	Disabled	CFGB + SUM(0:002BFF)	0xC155	0xBF57
	Enabled	CFGB	0x0353	0x0353
dsPIC33FJ16GS404	Disabled	CFGB + SUM(0:002BFF)	0xC155	0xBF57
	Enabled	CFGB	0x0353	0x0353
dsPIC33FJ16GS502	Disabled	CFGB + SUM(0:002BFF)	0xC155	0xBF57
	Enabled	CFGB	0x0353	0x0353
dsPIC33FJ16GS504	Disabled	CFGB + SUM(0:002BFF)	0xC155	0xBF57
	Enabled	CFGB	0x0353	0x0353
dsPIC33FJ12GP201	Disabled	CFGB + SUM(0:001FFF)	0xD43D	0xD23F
	Enabled	CFGB	0x043B	0x043B
dsPIC33FJ12GP202	Disabled	CFGB + SUM(0:001FFF)	0xD43D	0xD23F
	Enabled	CFGB	0x043B	0x043B
dsPIC33FJ16GP304	Disabled	CFGB + SUM(0:002BFF)	0xC23D	0xC03F
	Enabled	CFGB	0x043B	0x043B

**Item Description:**

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

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0x0F) + (FICD & 0xE3))

(for dsPIC33FJ06GS101/102/202 and dsPIC33FJ16GS402/404/502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202, PIC24HJ12GP201/202, dsPIC33FJ32GP202/204, dsPIC33FJ32MC202/204, PIC24HJ32GP202/204, dsPIC33FJ16GP304, dsPIC33FJ16MC304, PIC24HJ16GP304, dsPIC33FJ32GP302/304, dsPIC33FJ32MC302/304 and PIC24HJ32GP302/304 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ64GP202/204, dsPIC33F64MC202/204, PIC24HJ64GP202/204, dsPIC33FJ64GP802/804, dsPIC33FJ64MC802/804, PIC24HJ64GP502/504, dsPIC33FJ128GP202/204, dsPIC33FJ128MC202/204, PIC24HJ128GP202/204, dsPIC33FJ128GP802/804, dsPIC33FJ128MC802/804 and PIC24HJ128GP502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOSCSEL & 0xA7) + (FOSC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJXXXGPX06/X08/X10, dsPIC33FJ64GPX06A/X08A/X10A, dsPIC33FJ128GPX06A/X08A/X10A, dsPIC33FJXXXMCX06/X08/X10, dsPIC33FJ64MCX06A/X08A/X10A, dsPIC33FJ128MCX06A/X08A/X10A, PIC24HJXXXGPX06/X08/X10, PIC24HJ64GPX06A/X08A/X10A and PIC24HJ128GPX06A/X08A/X10A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOSCSEL & 0xA7) + (FOSC & 0xC7) + (FWDT & 0xFF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJ256GP506A/510A/710A, dsPIC33FJ256MC510A/710A and PIC24HJ256GP206A/210A/610A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0x67) + (FICD & 0xE3))

(for dsPIC33FJ32GS406/606/608/610 and dsPIC33FJ64GS406/606/608/610 devices)

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE D-1: CHECKSUM COMPUTATION (CONTINUED)**

Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAAAAAA at 0x0 and Last Code Address
dsPIC33FJ32GP202	Disabled	CFGB + SUM(0:0057FF)	0x803D	0x7E3F
	Enabled	CFGB	0x043B	0x043B
dsPIC33FJ32GP204	Disabled	CFGB + SUM(0:0057FF)	0x803D	0x7E3F
	Enabled	CFGB	0x043B	0x043B
dsPIC33FJ32GP302	Disabled	CFGB + SUM(0:0057FF)	0x803D	0x7E3F
	Enabled	CFGB	0x043B	0x043B
dsPIC33FJ32GP304	Disabled	CFGB + SUM(0:0057FF)	0x803D	0x7E3F
	Enabled	CFGB	0x043B	0x043B
dsPIC33FJ64GP202	Disabled	CFGB + SUM(0:00ABFF)	0x03CC	0x01CE
	Enabled	CFGB	0x05CA	0x05CA
dsPIC33FJ64GP204	Disabled	CFGB + SUM(0:00ABFF)	0x03CC	0x01CE
	Enabled	CFGB	0x05CA	0x05CA
dsPIC33FJ64GP206	Disabled	CFGB + SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64GP306	Disabled	CFGB + SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64GP310	Disabled	CFGB + SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64GP706	Disabled	CFGB + SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64GP708	Disabled	CFGB + SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA

**Item Description:**

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

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0x0F) + (FICD & 0xE3))

(for dsPIC33FJ06GS101/102/202 and dsPIC33FJ16GS402/404/502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202, PIC24HJ12GP201/202, dsPIC33FJ32GP202/204, dsPIC33FJ32MC202/204, PIC24HJ32GP202/204, dsPIC33FJ16GP304, dsPIC33FJ16MC304, PIC24HJ16GP304, dsPIC33FJ32GP302/304, dsPIC33FJ32MC302/304 and PIC24HJ32GP302/304 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ64GP202/204, dsPIC33F64MC202/204, PIC24HJ64GP202/204, dsPIC33FJ64GP802/804, dsPIC33FJ64MC802/804, PIC24HJ64GP502/504, dsPIC33FJ128GP202/204, dsPIC33FJ128MC202/204, PIC24HJ128GP202/204, dsPIC33FJ128GP802/804, dsPIC33FJ128MC802/804 and PIC24HJ128GP502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0xA7) + (FOC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJXXXGPX06/X08/X10, dsPIC33FJ64GPX06A/X08A/X10A, dsPIC33FJ128GPX06A/X08A/X10A, dsPIC33FJXXXMCX06/X08/X10, dsPIC33FJ64MCX06A/X08A/X10A, dsPIC33FJ128MCX06A/X08A/X10A, PIC24HJXXXGPX06/X08/X10, PIC24HJ64GPX06A/X08A/X10A and PIC24HJ128GPX06A/X08A/X10A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0xA7) + (FOC & 0xC7) + (FWDT & 0xFF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJ256GP506A/510A/710A, dsPIC33FJ256MC510A/710A and PIC24HJ256GP206A/210A/610A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0x67) + (FICD & 0xE3))

(for dsPIC33FJ32GS406/606/608/610 and dsPIC33FJ64GS406/606/608/610 devices)

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE D-1: CHECKSUM COMPUTATION (CONTINUED)**

Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAAAAAA at 0x0 and Last Code Address
dsPIC33FJ64GP710	Disabled	CFGB + SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64GP802	Disabled	CFGB + SUM(0:00ABFF)	0x03CC	0x01CE
	Enabled	CFGB	0x05CA	0x05CA
dsPIC33FJ64GP804	Disabled	CFGB + SUM(0:00ABFF)	0x03CC	0x01CE
	Enabled	CFGB	0x05CA	0x05CA
dsPIC33FJ128GP202	Disabled	CFGB + SUM(0:0157FF)	0x01CC	0xFFCE
	Enabled	CFGB	0x05CA	0x05CA
dsPIC33FJ128GP204	Disabled	CFGB + SUM(0:0157FF)	0x01CC	0xFFCE
	Enabled	CFGB	0x05CA	0x05CA
dsPIC33FJ128GP206	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128GP306	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128GP310	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128GP706	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128GP708	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128GP710	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA

**Item Description:**

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

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0x0F) + (FICD & 0xE3))

(for dsPIC33FJ06GS101/102/202 and dsPIC33FJ16GS402/404/502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202, PIC24HJ12GP201/202, dsPIC33FJ32GP202/204, dsPIC33FJ32MC202/204, PIC24HJ32GP202/204, dsPIC33FJ16GP304, dsPIC33FJ16MC304, PIC24HJ16GP304, dsPIC33FJ32GP302/304, dsPIC33FJ32MC302/304 and PIC24HJ32GP302/304 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ64GP202/204, dsPIC33F64MC202/204, PIC24HJ64GP202/204, dsPIC33FJ64GP802/804, dsPIC33FJ64MC802/804, PIC24HJ64GP502/504, dsPIC33FJ128GP202/204, dsPIC33FJ128MC202/204, PIC24HJ128GP202/204, dsPIC33FJ128GP802/804, dsPIC33FJ128MC802/804 and PIC24HJ128GP502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOSCSEL & 0xA7) + (FOSC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJXXXGPX06/X08/X10, dsPIC33FJ64GPX06A/X08A/X10A, dsPIC33FJ128GPX06A/X08A/X10A, dsPIC33FJXXXMCX06/X08/X10, dsPIC33FJ64MCX06A/X08A/X10A, dsPIC33FJ128MCX06A/X08A/X10A, PIC24HJXXXGPX06/X08/X10, PIC24HJ64GPX06A/X08A/X10A and PIC24HJ128GPX06A/X08A/X10A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOSCSEL & 0xA7) + (FOSC & 0xC7) + (FWDT & 0xFF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJ256GP506A/510A/710A, dsPIC33FJ256MC510A/710A and PIC24HJ256GP206A/210A/610A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0x67) + (FICD & 0xE3))

(for dsPIC33FJ32GS406/606/608/610 and dsPIC33FJ64GS406/606/608/610 devices)

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE D-1: CHECKSUM COMPUTATION (CONTINUED)**

Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAAAAAA at 0x0 and Last Code Address
dsPIC33FJ128GP802	Disabled	CFGB + SUM(0:0157FF)	0x01CC	0xFFCE
	Enabled	CFGB	0x05CA	0x05CA
dsPIC33FJ128GP804	Disabled	CFGB + SUM(0:0157FF)	0x01CC	0xFFCE
	Enabled	CFGB	0x05CA	0x05CA
dsPIC33FJ256GP506	Disabled	CFGB + SUM(0:02ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ256GP510	Disabled	CFGB + SUM(0:02ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ256GP710	Disabled	CFGB + SUM(0:02ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ12MC201	Disabled	CFGB + SUM(0:001FFF)	0xD43D	0xD23F
	Enabled	CFGB	0x043B	0x043B
dsPIC33FJ12MC202	Disabled	CFGB + SUM(0:001FFF)	0xD43D	0xD23F
	Enabled	CFGB	0x043B	0x043B
dsPIC33FJ16MC304	Disabled	CFGB + SUM(0:002BFF)	0xC23D	0xC03F
	Enabled	CFGB	0x043B	0x043B
dsPIC33FJ32MC202	Disabled	CFGB + SUM(0:0057FF)	0x803D	0x7E3F
	Enabled	CFGB	0x043B	0x043B
dsPIC33FJ32MC204	Disabled	CFGB + SUM(0:0057FF)	0x803D	0x7E3F
	Enabled	CFGB	0x043B	0x043B
dsPIC33FJ32MC302	Disabled	CFGB + SUM(0:0057FF)	0x803D	0x7E3F
	Enabled	CFGB	0x043B	0x043B

**Item Description:**

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

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0x0F) + (FICD & 0xE3))

(for dsPIC33FJ06GS101/102/202 and dsPIC33FJ16GS402/404/502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202, PIC24HJ12GP201/202, dsPIC33FJ32GP202/204, dsPIC33FJ32MC202/204, PIC24HJ32GP202/204, dsPIC33FJ16GP304, dsPIC33FJ16MC304, PIC24HJ16GP304, dsPIC33FJ32GP302/304, dsPIC33FJ32MC302/304 and PIC24HJ32GP302/304 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ64GP202/204, dsPIC33F64MC202/204, PIC24HJ64GP202/204, dsPIC33FJ64GP802/804, dsPIC33FJ64MC802/804, PIC24HJ64GP502/504, dsPIC33FJ128GP202/204, dsPIC33FJ128MC202/204, PIC24HJ128GP202/204, dsPIC33FJ128GP802/804, dsPIC33FJ128MC802/804 and PIC24HJ128GP502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0xA7) + (FOC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJXXXGPX06/X08/X10, dsPIC33FJ64GPX06A/X08A/X10A, dsPIC33FJ128GPX06A/X08A/X10A, dsPIC33FJXXXMCX06/X08/X10, dsPIC33FJ64MCX06A/X08A/X10A, dsPIC33FJ128MCX06A/X08A/X10A, PIC24HJXXXGPX06/X08/X10, PIC24HJ64GPX06A/X08A/X10A and PIC24HJ128GPX06A/X08A/X10A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0xA7) + (FOC & 0xC7) + (FWDT & 0xFF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJ256GP506A/510A/710A, dsPIC33FJ256MC510A/710A and PIC24HJ256GP206A/210A/610A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0x67) + (FICD & 0xE3))

(for dsPIC33FJ32GS406/606/608/610 and dsPIC33FJ64GS406/606/608/610 devices)

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE D-1: CHECKSUM COMPUTATION (CONTINUED)**

Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAAAAAA at 0x0 and Last Code Address
dsPIC33FJ32MC304	Disabled	CFGB + SUM(0:0057FF)	0x803D	0x7E3F
	Enabled	CFGB	0x043B	0x043B
dsPIC33FJ64MC202	Disabled	CFGB + SUM(0:00ABFF)	0x03CC	0x01CE
	Enabled	CFGB	0x05CA	0x05CA
dsPIC33FJ64MC204	Disabled	CFGB + SUM(0:00ABFF)	0x03CC	0x01CE
	Enabled	CFGB	0x05CA	0x05CA
dsPIC33FJ64MC506	Disabled	CFGB + SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64MC508	Disabled	CFGB + SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64MC510	Disabled	CFGB + SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64MC706	Disabled	CFGB + SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64MC710	Disabled	CFGB + SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64MC802	Disabled	CFGB + SUM(0:00ABFF)	0x03CC	0x01CE
	Enabled	CFGB	0x05CA	0x05CA
dsPIC33FJ64MC804	Disabled	CFGB + SUM(0:00ABFF)	0x03CC	0x01CE
	Enabled	CFGB	0x05CA	0x05CA
dsPIC33FJ128MC202	Disabled	CFGB + SUM(0:0157FF)	0x01CC	0xFFCE
	Enabled	CFGB	0x05CA	0x05CA

**Item Description:**

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

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0x0F) + (FICD & 0xE3))

(for dsPIC33FJ06GS101/102/202 and dsPIC33FJ16GS402/404/502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202, PIC24HJ12GP201/202, dsPIC33FJ32GP202/204, dsPIC33FJ32MC202/204, PIC24HJ32GP202/204, dsPIC33FJ16GP304, dsPIC33FJ16MC304, PIC24HJ16GP304, dsPIC33FJ32GP302/304, dsPIC33FJ32MC302/304 and PIC24HJ32GP302/304 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ64GP202/204, dsPIC33FJ64MC202/204, PIC24HJ64GP202/204, dsPIC33FJ64GP802/804, dsPIC33FJ64MC802/804, PIC24HJ64GP502/504, dsPIC33FJ128GP202/204, dsPIC33FJ128MC202/204, PIC24HJ128GP202/204, dsPIC33FJ128GP802/804, dsPIC33FJ128MC802/804 and PIC24HJ128GP502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOSCSEL & 0xA7) + (FOSC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJXXXGPX06/X08/X10, dsPIC33FJ64GPX06A/X08A/X10A, dsPIC33FJ128GPX06A/X08A/X10A, dsPIC33FJXXXMCX06/X08/X10, dsPIC33FJ64MCX06A/X08A/X10A, dsPIC33FJ128MCX06A/X08A/X10A, PIC24HJXXXGPX06/X08/X10, PIC24HJ64GPX06A/X08A/X10A and PIC24HJ128GPX06A/X08A/X10A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOSCSEL & 0xA7) + (FOSC & 0xC7) + (FWDT & 0xFF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJ256GP506A/510A/710A, dsPIC33FJ256MC510A/710A and PIC24HJ256GP206A/210A/610A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0x67) + (FICD & 0xE3))

(for dsPIC33FJ32GS406/606/608/610 and dsPIC33FJ64GS406/606/608/610 devices)

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE D-1: CHECKSUM COMPUTATION (CONTINUED)**

Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAAAAAA at 0x0 and Last Code Address
dsPIC33FJ128MC204	Disabled	CFGB + SUM(0:0157FF)	0x01CC	0xFFCE
	Enabled	CFGB	0x05CA	0x05CA
dsPIC33FJ128MC506	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128MC510	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128MC706	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128MC708	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128MC710	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ256MC510	Disabled	CFGB + SUM(0:02ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ256MC710	Disabled	CFGB + SUM(0:02ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128MC802	Disabled	CFGB + SUM(0:0157FF)	0x01CC	0xFFCE
	Enabled	CFGB	0x05CA	0x05CA
dsPIC33FJ128MC804	Disabled	CFGB + SUM(0:0157FF)	0x01CC	0xFFCE
	Enabled	CFGB	0x05CA	0x05CA
PIC24HJ12GP201	Disabled	CFGB + SUM(0:001FFF)	0xD43D	0xD23F
	Enabled	CFGB	0x043B	0x043B

**Item Description:**

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

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0x0F) + (FICD & 0xE3))

(for dsPIC33FJ06GS101/102/202 and dsPIC33FJ16GS402/404/502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202, PIC24HJ12GP201/202, dsPIC33FJ32GP202/204, dsPIC33FJ32MC202/204, PIC24HJ32GP202/204, dsPIC33FJ16GP304, dsPIC33FJ16MC304, PIC24HJ16GP304, dsPIC33FJ32GP302/304, dsPIC33FJ32MC302/304 and PIC24HJ32GP302/304 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ64GP202/204, dsPIC33F64MC202/204, PIC24HJ64GP202/204, dsPIC33FJ64GP802/804, dsPIC33FJ64MC802/804, PIC24HJ64GP502/504, dsPIC33FJ128GP202/204, dsPIC33FJ128MC202/204, PIC24HJ128GP202/204, dsPIC33FJ128GP802/804, dsPIC33FJ128MC802/804 and PIC24HJ128GP502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0xA7) + (FOC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJXXXGPX06/X08/X10, dsPIC33FJ64GPX06A/X08A/X10A, dsPIC33FJ128GPX06A/X08A/X10A, dsPIC33FJXXXMCX06/X08/X10, dsPIC33FJ64MCX06A/X08A/X10A, dsPIC33FJ128MCX06A/X08A/X10A, PIC24HJXXXGPX06/X08/X10, PIC24HJ64GPX06A/X08A/X10A and PIC24HJ128GPX06A/X08A/X10A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0xA7) + (FOC & 0xC7) + (FWDT & 0xFF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJ256GP506A/510A/710A, dsPIC33FJ256MC510A/710A and PIC24HJ256GP206A/210A/610A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0x67) + (FICD & 0xE3))

(for dsPIC33FJ32GS406/606/608/610 and dsPIC33FJ64GS406/606/608/610 devices)

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE D-1: CHECKSUM COMPUTATION (CONTINUED)**

Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAAAAAA at 0x0 and Last Code Address
PIC24HJ12GP202	Disabled	CFGB + SUM(0:001FFF)	0xD43D	0xD23F
	Enabled	CFGB	0x043B	0x043B
PIC24HJ16GP304	Disabled	CFGB + SUM(0:002BFF)	0xC23D	0xC03F
	Enabled	CFGB	0x043B	0x043B
PIC24HJ32GP202	Disabled	CFGB + SUM(0:0057FF)	0x803D	0x7E3F
	Enabled	CFGB	0x043B	0x043B
PIC24HJ32GP204	Disabled	CFGB + SUM(0:0057FF)	0x803D	0x7E3F
	Enabled	CFGB	0x043B	0x043B
PIC24HJ32GP302	Disabled	CFGB + SUM(0:0057FF)	0x803D	0x7E3F
	Enabled	CFGB	0x043B	0x043B
PIC24HJ32GP304	Disabled	CFGB + SUM(0:0057FF)	0x803D	0x7E3F
	Enabled	CFGB	0x043B	0x043B
PIC24HJ64GP202	Disabled	CFGB + SUM(0:00ABFF)	0x03CC	0x01CE
	Enabled	CFGB	0x05CA	0x05CA
PIC24HJ64GP204	Disabled	CFGB + SUM(0:00ABFF)	0x03CC	0x01CE
	Enabled	CFGB	0x05CA	0x05CA
PIC24HJ64GP206	Disabled	CFGB + SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ64GP210	Disabled	CFGB + SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ64GP502	Disabled	CFGB + SUM(0:00ABFF)	0x03CC	0x01CE
	Enabled	CFGB	0x05CA	0x05CA

**Item Description:**

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

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0x0F) + (FICD & 0xE3))

(for dsPIC33FJ06GS101/102/202 and dsPIC33FJ16GS402/404/502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202, PIC24HJ12GP201/202, dsPIC33FJ32GP202/204, dsPIC33FJ32MC202/204, PIC24HJ32GP202/204, dsPIC33FJ16GP304, dsPIC33FJ16MC304, PIC24HJ16GP304, dsPIC33FJ32GP302/304, dsPIC33FJ32MC302/304 and PIC24HJ32GP302/304 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ64GP202/204, dsPIC33F64MC202/204, PIC24HJ64GP202/204, dsPIC33FJ64GP802/804, dsPIC33FJ64MC802/804, PIC24HJ64GP502/504, dsPIC33FJ128GP202/204, dsPIC33FJ128MC202/204, PIC24HJ128GP202/204, dsPIC33FJ128GP802/804, dsPIC33FJ128MC802/804 and PIC24HJ128GP502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOSCSEL & 0xA7) + (FOSC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJXXXGPX06/X08/X10, dsPIC33FJ64GPX06A/X08A/X10A, dsPIC33FJ128GPX06A/X08A/X10A, dsPIC33FJXXXMCX06/X08/X10, dsPIC33FJ64MCX06A/X08A/X10A, dsPIC33FJ128MCX06A/X08A/X10A, PIC24HJXXXGPX06/X08/X10, PIC24HJ64GPX06A/X08A/X10A and PIC24HJ128GPX06A/X08A/X10A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOSCSEL & 0xA7) + (FOSC & 0xC7) + (FWDT & 0xFF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJ256GP506A/510A/710A, dsPIC33FJ256MC510A/710A and PIC24HJ256GP206A/210A/610A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0x67) + (FICD & 0xE3))

(for dsPIC33FJ32GS406/606/608/610 and dsPIC33FJ64GS406/606/608/610 devices)

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE D-1: CHECKSUM COMPUTATION (CONTINUED)**

Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAAAAAA at 0x0 and Last Code Address
PIC24HJ64GP504	Disabled	CFGB + SUM(0:00ABFF)	0x03CC	0x01CE
	Enabled	CFGB	0x05CA	0x05CA
PIC24HJ64GP506	Disabled	CFGB + SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ64GP510	Disabled	CFGB + SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ128GP202	Disabled	CFGB + SUM(0:0157FF)	0x01CC	0xFFCE
	Enabled	CFGB	0x05CA	0x05CA
PIC24HJ128GP204	Disabled	CFGB + SUM(0:0157FF)	0x01CC	0xFFCE
	Enabled	CFGB	0x05CA	0x05CA
PIC24HJ128GP206	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ128GP210	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ128GP306	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ128GP310	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ128GP502	Disabled	CFGB + SUM(0:0157FF)	0x01CC	0xFFCE
	Enabled	CFGB	0x05CA	0x05CA

**Item Description:**

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

**CFGB = Configuration Block (masked) =** Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0x0F) + (FICD & 0xE3))

(for dsPIC33FJ06GS101/102/202 and dsPIC33FJ16GS402/404/502/504 devices)

**CFGB = Configuration Block (masked) =** Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202, PIC24HJ12GP201/202, dsPIC33FJ32GP202/204, dsPIC33FJ32MC202/204, PIC24HJ32GP202/204, dsPIC33FJ16GP304, dsPIC33FJ16MC304, PIC24HJ16GP304, dsPIC33FJ32GP302/304, dsPIC33FJ32MC302/304 and PIC24HJ32GP302/304 devices)

**CFGB = Configuration Block (masked) =** Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ64GP202/204, dsPIC33F64MC202/204, PIC24HJ64GP202/204, dsPIC33FJ64GP802/804, dsPIC33FJ64MC802/804, PIC24HJ64GP502/504, dsPIC33FJ128GP202/204, dsPIC33FJ128MC202/204, PIC24HJ128GP202/204, dsPIC33FJ128GP802/804, dsPIC33FJ128MC802/804 and PIC24HJ128GP502/504 devices)

**CFGB = Configuration Block (masked) =** Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0xA7) + (FOC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJXXXGPX06/X08/X10, dsPIC33FJ64GPX06A/X08A/X10A, dsPIC33FJ128GPX06A/X08A/X10A, dsPIC33FJXXXMCX06/X08/X10, dsPIC33FJ64MCX06A/X08A/X10A, dsPIC33FJ128MCX06A/X08A/X10A, PIC24HJXXXGPX06/X08/X10, PIC24HJ64GPX06A/X08A/X10A and PIC24HJ128GPX06A/X08A/X10A devices)

**CFGB = Configuration Block (masked) =** Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0xA7) + (FOC & 0xC7) + (FWDT & 0xFF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJ256GP506A/510A/710A, dsPIC33FJ256MC510A/710A and PIC24HJ256GP206A/210A/610A devices)

**CFGB = Configuration Block (masked) =** Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0x67) + (FICD & 0xE3))

(for dsPIC33FJ32GS406/606/608/610 and dsPIC33FJ64GS406/606/608/610 devices)

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE D-1: CHECKSUM COMPUTATION (CONTINUED)**

Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAAAAAA at 0x0 and Last Code Address
PIC24HJ128GP504	Disabled	CFGB + SUM(0:0157FF)	0x01CC	0xFFCE
	Enabled	CFGB	0x05CA	0x05CA
PIC24HJ128GP506	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ128GP510	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ256GP206	Disabled	CFGB + SUM(0:02ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ256GP210	Disabled	CFGB + SUM(0:02ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ256GP610	Disabled	CFGB + SUM(0:02ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64GP206A	Disabled	CFGB +SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64GP306A	Disabled	CFGB +SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64GP310A	Disabled	CFGB +SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64GP706A	Disabled	CFGB +SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64GP708A	Disabled	CFGB +SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA

**Item Description:**

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

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0x0F) + (FICD & 0xE3))

(for dsPIC33FJ06GS101/102/202 and dsPIC33FJ16GS402/404/502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202, PIC24HJ12GP201/202, dsPIC33FJ32GP202/204, dsPIC33FJ32MC202/204, PIC24HJ32GP202/204, dsPIC33FJ16GP304, dsPIC33FJ16MC304, PIC24HJ16GP304, dsPIC33FJ32GP302/304, dsPIC33FJ32MC302/304 and PIC24HJ32GP302/304 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ64GP202/204, dsPIC33F64MC202/204, PIC24HJ64GP202/204, dsPIC33FJ64GP802/804, dsPIC33FJ64MC802/804, PIC24HJ64GP502/504, dsPIC33FJ128GP202/204, dsPIC33FJ128MC202/204, PIC24HJ128GP202/204, dsPIC33FJ128GP802/804, dsPIC33FJ128MC802/804 and PIC24HJ128GP502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOSCSEL & 0xA7) + (FOSC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJXXXGPX06/X08/X10, dsPIC33FJ64GPX06A/X08A/X10A, dsPIC33FJ128GPX06A/X08A/X10A, dsPIC33FJXXXMCX06/X08/X10, dsPIC33FJ64MCX06A/X08A/X10A, dsPIC33FJ128MCX06A/X08A/X10A, PIC24HJXXXGPX06/X08/X10, PIC24HJ64GPX06A/X08A/X10A and PIC24HJ128GPX06A/X08A/X10A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOSCSEL & 0xA7) + (FOSC & 0xC7) + (FWDT & 0xFF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJ256GP506A/510A/710A, dsPIC33FJ256MC510A/710A and PIC24HJ256GP206A/210A/610A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0x67) + (FICD & 0xE3))

(for dsPIC33FJ32GS406/606/608/610 and dsPIC33FJ64GS406/606/608/610 devices)

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE D-1: CHECKSUM COMPUTATION (CONTINUED)**

Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAAAAAA at 0x0 and Last Code Address
dsPIC33FJ64GP710A	Disabled	CFGB +SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64MC506A	Disabled	CFGB +SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64MC508A	Disabled	CFGB +SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64MC510A	Disabled	CFGB +SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64MC706A	Disabled	CFGB +SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ64MC710A	Disabled	CFGB +SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ64GP206A	Disabled	CFGB +SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ64GP210A	Disabled	CFGB +SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ64GP506A	Disabled	CFGB +SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ64GP510A	Disabled	CFGB +SUM(0:00ABFF)	0x03BC	0x01BE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128GP206A	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA

**Item Description:**

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

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0x0F) + (FICD & 0xE3))

(for dsPIC33FJ06GS101/102/202 and dsPIC33FJ16GS402/404/502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202, PIC24HJ12GP201/202, dsPIC33FJ32GP202/204, dsPIC33FJ32MC202/204, PIC24HJ32GP202/204, dsPIC33FJ16GP304, dsPIC33FJ16MC304, PIC24HJ16GP304, dsPIC33FJ32GP302/304, dsPIC33FJ32MC302/304 and PIC24HJ32GP302/304 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ64GP202/204, dsPIC33F64MC202/204, PIC24HJ64GP202/204, dsPIC33FJ64GP802/804, dsPIC33FJ64MC802/804, PIC24HJ64GP502/504, dsPIC33FJ128GP202/204, dsPIC33FJ128MC202/204, PIC24HJ128GP202/204, dsPIC33FJ128GP802/804, dsPIC33FJ128MC802/804 and PIC24HJ128GP502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0xA7) + (FOC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJXXXGPX06/X08/X10, dsPIC33FJ64GPX06A/X08A/X10A, dsPIC33FJ128GPX06A/X08A/X10A, dsPIC33FJXXXMCX06/X08/X10, dsPIC33FJ64MCX06A/X08A/X10A, dsPIC33FJ128MCX06A/X08A/X10A, PIC24HJXXXGPX06/X08/X10, PIC24HJ64GPX06A/X08A/X10A and PIC24HJ128GPX06A/X08A/X10A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0xA7) + (FOC & 0xC7) + (FWDT & 0xFF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJ256GP506A/510A/710A, dsPIC33FJ256MC510A/710A and PIC24HJ256GP206A/210A/610A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0x67) + (FICD & 0xE3))

(for dsPIC33FJ32GS406/606/608/610 and dsPIC33FJ64GS406/606/608/610 devices)

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE D-1: CHECKSUM COMPUTATION (CONTINUED)**

Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAAAAAA at 0x0 and Last Code Address
dsPIC33FJ128GP306A	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128GP310A	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128GP706A	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128GP708A	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128GP710A	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128MC506A	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128MC510A	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128MC706A	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128MC708A	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ128MC710A	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ128GP206A	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA

**Item Description:**

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

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0x0F) + (FICD & 0xE3))

(for dsPIC33FJ06GS101/102/202 and dsPIC33FJ16GS402/404/502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202, PIC24HJ12GP201/202, dsPIC33FJ32GP202/204, dsPIC33FJ32MC202/204, PIC24HJ32GP202/204, dsPIC33FJ16GP304, dsPIC33FJ16MC304, PIC24HJ16GP304, dsPIC33FJ32GP302/304, dsPIC33FJ32MC302/304 and PIC24HJ32GP302/304 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ64GP202/204, dsPIC33F64MC202/204, PIC24HJ64GP202/204, dsPIC33FJ64GP802/804, dsPIC33FJ64MC802/804, PIC24HJ64GP502/504, dsPIC33FJ128GP202/204, dsPIC33FJ128MC202/204, PIC24HJ128GP202/204, dsPIC33FJ128GP802/804, dsPIC33FJ128MC802/804 and PIC24HJ128GP502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOSCSEL & 0xA7) + (FOSC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJXXXGPX06/X08/X10, dsPIC33FJ64GPX06A/X08A/X10A, dsPIC33FJ128GPX06A/X08A/X10A, dsPIC33FJXXXMCX06/X08/X10, dsPIC33FJ64MCX06A/X08A/X10A, dsPIC33FJ128MCX06A/X08A/X10A, PIC24HJXXXGPX06/X08/X10, PIC24HJ64GPX06A/X08A/X10A and PIC24HJ128GPX06A/X08A/X10A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOSCSEL & 0xA7) + (FOSC & 0xC7) + (FWDT & 0xFF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJ256GP506A/510A/710A, dsPIC33FJ256MC510A/710A and PIC24HJ256GP206A/210A/610A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOSCSEL & 0x87) + (FOSC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0x67) + (FICD & 0xE3))

(for dsPIC33FJ32GS406/606/608/610 and dsPIC33FJ64GS406/606/608/610 devices)

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE D-1: CHECKSUM COMPUTATION (CONTINUED)**

Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAAAAAA at 0x0 and Last Code Address
PIC24HJ128GP210A	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ128GP306A	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ128GP310A	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ128GP506A	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFDE
	Enabled	CFGB	0x05BA	0x05BA
PIC24HJ128GP510A	Disabled	CFGB + SUM(0:0157FF)	0x01BC	0xFFBE
	Enabled	CFGB	0x05BA	0x05BA
dsPIC33FJ256GP506A	Disabled	CFGB +SUM(0:02ABFF)	0x03DC	0x01DE
	Enabled	CFGB	0x05DA	0x05DA
dsPIC33FJ256GP510A	Disabled	CFGB +SUM(0:02ABFF)	0x03DC	0x01DE
	Enabled	CFGB	0x05DA	0x05DA
dsPIC33FJ256GP710A	Disabled	CFGB +SUM(0:02ABFF)	0x03DC	0x01DE
	Enabled	CFGB	0x05DA	0x05DA
dsPIC33FJ256MC510A	Disabled	CFGB +SUM(0:02ABFF)	0x03DC	0x01DE
	Enabled	CFGB	0x05DA	0x05DA
dsPIC33FJ256MC710A	Disabled	CFGB +SUM(0:02ABFF)	0x03DC	0x01DE
	Enabled	CFGB	0x05DA	0x05DA
PIC24HJ256GP206A	Disabled	CFGB +SUM(0:02ABFF)	0x03DC	0x01DE
	Enabled	CFGB	0x05DA	0x05DA

**Item Description:**

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

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0x0F) + (FICD & 0xE3))

(for dsPIC33FJ06GS101/102/202 and dsPIC33FJ16GS402/404/502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202, PIC24HJ12GP201/202, dsPIC33FJ32GP202/204, dsPIC33FJ32MC202/204, PIC24HJ32GP202/204, dsPIC33FJ16GP304, dsPIC33FJ16MC304, PIC24HJ16GP304, dsPIC33FJ32GP302/304, dsPIC33FJ32MC302/304 and PIC24HJ32GP302/304 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ64GP202/204, dsPIC33F64MC202/204, PIC24HJ64GP202/204, dsPIC33FJ64GP802/804, dsPIC33FJ64MC802/804, PIC24HJ64GP502/504, dsPIC33FJ128GP202/204, dsPIC33FJ128MC202/204, PIC24HJ128GP202/204, dsPIC33FJ128GP802/804, dsPIC33FJ128MC802/804 and PIC24HJ128GP502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0xA7) + (FOC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJXXXGPX06/X08/X10, dsPIC33FJ64GPX06A/X08A/X10A, dsPIC33FJ128GPX06A/X08A/X10A, dsPIC33FJXXXMCX06/X08/X10, dsPIC33FJ64MCX06A/X08A/X10A, dsPIC33FJ128MCX06A/X08A/X10A, PIC24HJXXXGPX06/X08/X10, PIC24HJ64GPX06A/X08A/X10A and PIC24HJ128GPX06A/X08A/X10A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0xA7) + (FOC & 0xC7) + (FWDT & 0xFF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJ256GP506A/510A/710A, dsPIC33FJ256MC510A/710A and PIC24HJ256GP206A/210A/610A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0x67) + (FICD & 0xE3))

(for dsPIC33FJ32GS406/606/608/610 and dsPIC33FJ64GS406/606/608/610 devices)

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

**TABLE D-1: CHECKSUM COMPUTATION (CONTINUED)**

Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAFFFFFFF at 0x0 and Last Code Address
PIC24HJ256GP210A	Disabled	CFGB +SUM(0:02ABFF)	0x03DC	0x01DE
	Enabled	CFGB	0x05DA	0x05DA
PIC24HJ256GP610A	Disabled	CFGB +SUM(0:02ABFF)	0x03DC	0x01DE
	Enabled	CFGB	0x05DA	0x05DA
dsPIC33FJ32GS406	Disabled	CFGB +SUM(0:0057FF)	0x7F8D	0x7D8F
	Enabled	CFGB	0x038B	0x038B
dsPIC33FJ64GS406	Disabled	CFGB +SUM(0:00ABFF)	0x018D	0xFF8F
	Enabled	CFGB	0x038B	0x038B
dsPIC33FJ32GS606	Disabled	CFGB +SUM(0:0057FF)	0x7F8D	0x7D8F
	Enabled	CFGB	0x038B	0x038B
dsPIC33FJ64GS606	Disabled	CFGB +SUM(0:00ABFF)	0x018D	0xFF8F
	Enabled	CFGB	0x038B	0x038B
dsPIC33FJ32GS608	Disabled	CFGB +SUM(0:0057FF)	0x7F8D	0x7D8F
	Enabled	CFGB	0x038B	0x038B
dsPIC33FJ64GS608	Disabled	CFGB +SUM(0:00ABFF)	0x018D	0xFF8F
	Enabled	CFGB	0x038B	0x038B
dsPIC33FJ32GS610	Disabled	CFGB +SUM(0:0057FF)	0x7F8D	0x7D8F
	Enabled	CFGB	0x038B	0x038B
dsPIC33FJ64GS610	Disabled	CFGB +SUM(0:00ABFF)	0x018D	0xFF8F
	Enabled	CFGB	0x038B	0x038B

**Item Description:**

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

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0x0F) + (FICD & 0xE3))

(for dsPIC33FJ06GS101/102/202 and dsPIC33FJ16GS402/404/502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202, PIC24HJ12GP201/202, dsPIC33FJ32GP202/204, dsPIC33FJ32MC202/204, PIC24HJ32GP202/204, dsPIC33FJ16GP304, dsPIC33FJ16MC304, dsPIC33FJ32GP302/304, dsPIC33FJ32MC302/304 and PIC24HJ32GP302/304 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xE7) + (FWDT & 0xDF) + (FPOR & 0xF7) + (FICD & 0xE3))

(for dsPIC33FJ64GP202/204, dsPIC33F64MC202/204, PIC24HJ64GP202/204, dsPIC33FJ64GP802/804, dsPIC33FJ64MC802/804, PIC24HJ64GP502/504, dsPIC33FJ128GP202/204, dsPIC33FJ128MC202/204, PIC24HJ128GP202/204, dsPIC33FJ128GP802/804, dsPIC33FJ128MC802/804 and PIC24HJ128GP502/504 devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0xA7) + (FOC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJXXXGPX06/X08/X10, dsPIC33FJ64GPX06A/X08A/X10A, dsPIC33FJ128GPX06A/X08A/X10A, dsPIC33FJXXXMCX06/X08/X10, dsPIC33FJ64MCX06A/X08A/X10A, dsPIC33FJ128MCX06A/X08A/X10A, PIC24HJXXXGPX06/X08/X10, PIC24HJ64GPX06A/X08A/X10A and PIC24HJ128GPX06A/X08A/X10A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0xCF) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0xA7) + (FOC & 0xC7) + (FWDT & 0xFF) + (FPOR & 0xE7) + (FICD & 0xE3))

(for dsPIC33FJ256GP506A/510A/710A, dsPIC33FJ256MC510A/710A and PIC24HJ256GP206A/210A/610A devices)

**CFGB = Configuration Block (masked)** = Byte sum of ((FBS & 0x0F) + (FGS & 0x07) + (FOCSEL & 0x87) + (FOC & 0xC7) + (FWDT & 0xDF) + (FPOR & 0x67) + (FICD & 0xE3))

(for dsPIC33FJ32GS406/606/608/610 and dsPIC33FJ64GS406/606/608/610 devices)

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

---

---

Table D-2 provides an example of how to calculate the checksum when using CodeGuard™ Security with Boot Segment Program Flash code protection with Standard Security and small-sized boot Flash.

**TABLE D-2: CHECKSUM COMPUTATION EXAMPLE WHEN USING CodeGuard™ SECURITY**

Device	Read Code Protection	Checksum Computation	Erased Value
dsPIC33FJ256GP506A	Disabled	CFGB1 <sup>(1)</sup> + SUM(0:2ABFF) – SUM(0200:07FE)	0xFD01

**Note 1:** CFGB1 = Byte sum of ((FBS & 0xCD) + (FSS & 0xCF) + (FGS & 0x07) + (FOCSEL & 0xA7) + (FOSC & 0xC7) + (FWDT & 0xFF) + (FPOR & 0xE7) + (FICD & 0xE3))

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

---

## APPENDIX E: REVISION HISTORY

### Revision C (June 2006)

- Added code protection Configuration register descriptions
- Added information about Unit ID
- Added `ERASES`, `ERASEG` and `ERASEC` programming executive commands
- Added checksum computation equation

### Revision D (March 2007)

- Added information specific to the dsPIC33FJ12GP201/202, dsPIC33FJ12MC201/202 and PIC24HJ12GP201/202 devices in several sections, including pinout diagrams, program memory sizes and Device ID values
- Added specific checksum computations for all dsPIC33F and PIC24H devices
- Updated ICSP bulk/page erase and row/byte program code examples to show externally timed operation (waiting for specific delay periods)
- Added the P20 timing characteristic
- Updated timing characteristics and references to the timing characteristics
- Updated the ICSP code examples

### Revision E (June 2007)

- Removed list of devices from first page. This document covers all released dsPIC33F/PIC24H devices, therefore, references to specific devices are included when applicable
- Added information specific to the following new devices in several sections, including program memory sizes, Configuration bit descriptions, checksum computations, Configuration registers, and Device ID values:
  - dsPIC33FJ32GP202
  - dsPIC33FJ32GP204
  - dsPIC33FJ16GP304
  - dsPIC33FJ32MC202
  - dsPIC33FJ32MC204
  - dsPIC33FJ16MC304
  - PIC24HJ32GP202
  - PIC24HJ32GP204
  - PIC24HJ16GP304
- Updated paragraph in [Section 2.3 “Pins Used During Programming”](#) to include reference to location of complete pin diagrams
- Renamed Table 2-1 to “Pins Used During Programming”
- Reordered list of devices in Table 2-2, Table 3-2 and Table 7-1 by device family and memory size
- Removed all pin diagrams
- Updated checksum information in Table 3-2 for the following devices:
  - dsPIC33FJ12GP201
  - dsPIC33FJ12GP202
  - dsPIC33FJ12MC201
  - dsPIC33FJ12MC201
  - PIC24HJ12GP201
  - PIC24HJ12GP202
- Added a Note to [Section 3.6.1 “Overview”](#)
- Changed bit 5 of the FOSCSEL register to unimplemented (—) in Table 3-4
- Modified the first paragraph of [Section 3.6.2 “Programming Methodology”](#)
- Added [Appendix A: “Hex File Format”](#)

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## Revision F (March 2008)

- Removed Preliminary status from document footer
- Added information specific to the following new devices in several sections, including program memory sizes, Configuration bit descriptions, checksum computations, Configuration registers and Device ID values:
  - dsPIC33FJ06GS101
  - dsPIC33FJ06GS102
  - dsPIC33FJ06GS202
  - dsPIC33FJ16GS402
  - dsPIC33FJ16GS404
  - dsPIC33FJ16GS502
  - dsPIC33FJ16GS504
  - dsPIC33FJ32GP302
  - dsPIC33FJ32GP304
  - dsPIC33FJ32MC302
  - dsPIC33FJ32MC304
  - dsPIC33FJ64GP202
  - dsPIC33FJ64GP204
  - dsPIC33FJ64GP802
  - dsPIC33FJ64GP804
  - dsPIC33FJ64MC202
  - dsPIC33FJ64MC204
  - dsPIC33FJ64MC802
  - dsPIC33FJ64MC804
- Added a Note to **Section 2.2 “Program Memory Write/Erase Requirements”** (preserving last eight instruction words of test memory for dsPIC33FJX6GSXXX devices)
- Added BOREN bit (FPOR<3>) to Configuration Register Map and added Note 5 (see Table 3-4)
- Updated Note 1 in **Section 5.3.1 “SIX Serial Instruction Execution”**
- Updated Note in **Section 5.3.2 “REGOUT Serial Instruction Execution”**
- Updated Figure 5-2
- Added Register 5-1
- Updated Table 5-7 and Table 5-8
- Added **Appendix B: “Device ID Register Silicon Errata Addendum”**

- dsPIC33FJ128GP202	- dsPIC33FJ128GP802
- dsPIC33FJ128GP204	- dsPIC33FJ128GP804
- dsPIC33FJ128MC202	- dsPIC33FJ128MC204
- dsPIC33FJ128MC802	- dsPIC33FJ128MC804
- PIC24HJ32GP302	- PIC24HJ32GP304
- PIC24HJ64GP202	- PIC24HJ64GP204
- PIC24HJ64GP502	- PIC24HJ64GP504
- PIC24HJ128GP202	- PIC24HJ128GP204
- PIC24HJ128GP502	- PIC24HJ128GP504

## Revision G (March 2009)

This revision includes minor updates to the text such as typographical errors, formatting and grammar.

The following detail the major updates:

- Added **Appendix C: “Diagnostic and Calibration Registers”**
- Reordered the appendixes so that the Revision History appears last in the document
- Updated all occurrences of VDDCORE to VCAP/ VDDCORE
- Updated all occurrences of PGC<sub>n</sub> and PGD<sub>n</sub> (where *n* = 1, 2 or 3) to PGEC<sub>n</sub> and PGD<sub>n</sub>, respectively
- Added information specific to the following new devices in several sections, including Configuration bit descriptions, checksum computations, Configuration registers and Device ID values:
  - dsPIC33FJ64GP206A
  - dsPIC33FJ64GP306A
  - dsPIC33FJ64GP310A
  - dsPIC33FJ64GP706A
  - dsPIC33FJ64GP708A
  - dsPIC33FJ64GP710A
  - dsPIC33FJ64MC506A
  - dsPIC33FJ64MC508A
  - dsPIC33FJ64MC510A
  - dsPIC33FJ64MC706A
  - dsPIC33FJ64MC710A
  - PIC24HJ64GP206A
  - PIC24HJ64GP210A
  - PIC24HJ64GP506A
  - PIC24HJ64GP510A
  - dsPIC33FJ128GP206A
  - dsPIC33FJ128GP306A
  - dsPIC33FJ128GP310A
  - dsPIC33FJ128GP706A
  - dsPIC33FJ128GP708A
  - dsPIC33FJ128GP710A
  - dsPIC33FJ128MC506A
  - dsPIC33FJ128MC510A
  - dsPIC33FJ128MC706A
  - dsPIC33FJ128MC708A
  - dsPIC33FJ128MC710A
  - PIC24HJ128GP206A
  - PIC24HJ128GP210A
  - PIC24HJ128GP306A
  - PIC24HJ128GP310A
  - PIC24HJ128GP506A
  - PIC24HJ128GP510A
  - PIC24HJ256GP206A
  - PIC24HJ256GP210A
  - PIC24HJ256GP610A
  - dsPIC33FJ32GS406
  - dsPIC33FJ32GS606
  - dsPIC33FJ32GS608
  - dsPIC33FJ32GS610
  - dsPIC33FJ64GS406
  - dsPIC33FJ64GS606
  - dsPIC33FJ64GS608
  - dsPIC33FJ64GS610
- Moved Note from **Section 2.2 “Program Memory Write/Erase Requirements”** to **Section 5.0 “Device Programming – ICSP”**
- Added Note 2 to **Figure 2-2**
- Changed the user memory limit for dsPIC33FJ16GS402/404/502/504 devices to 0x002BFE in **Table 2-2**
- Changed the Configuration Register ending address to 0xF80016 in **Figure 2-3**

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

## Revision G (March 2009) (Continued)

- Removed `PROGW` and renamed `QBLANK` to `QBLANKEX` in the Command Set Summary, and added the following commands (see [Table 3-1](#)):
  - `ERASEB`
  - `ERASEP`
  - `CRCP`
- Updated the flowchart in [Figure 3-2](#)
- Updated the second paragraph in [Section 3.2 “Confirming the Presence of the Programming Executive”](#)
- Changed the CFGB calculation for dsPIC33FJ06GS101/102/202 and dsPIC33FJ16GS402/404/502/504 devices to:  
+ (FPOR & 0x0F)
- Removed Note 2 in [Section 3.6.1 “Overview”](#)
- Updated the Erased Value and Value with 0xAAAAAA at 0x0 and Last Code Address columns in [Table 3-2](#) for the following devices:
  - dsPIC33FJ64GP202
  - dsPIC33FJ64GP204
  - dsPIC33FJ64GP802
  - dsPIC33FJ64GP804
  - dsPIC33FJ64MC202
  - dsPIC33FJ64MC204
  - dsPIC33FJ64MC802
  - dsPIC33FJ64MC804
  - PIC24HJ64GP202
  - PIC24HJ64GP204
  - PIC24HJ64GP502
  - PIC24HJ64GP504
- Removed COE and BKBUG Configuration Bit Descriptions in [Table 3-2](#) and [Table 3-3](#)
- Added PLLKEN to FWDT in [Table 3-2](#) and [Table 3-3](#)
- Added ALTQIO and ALTSS1 to FPOR in [Table 3-2](#) and [Table 3-3](#)
- Removed Notes for the following Configuration Bit Descriptions in [Table 3-2](#):
  - RBS
  - RSS (removed Note 1)
  - SSS
  - SWRP
  - IOL1WAY
  - ALTI2C
  - BOREN
- Added Note 6 and Note 7 to [Table 3-3](#)
- Added Note 1 to [Figure 3-5](#)
- Added Note to the beginning of [Section 4.0 “The Programming Executive”](#)
- Updated the third paragraph in [Section 4.1.1 “Communication Interface and Protocol”](#) to clarify the clock response time
- Changed the clock speed to be provided by the programmer from 7.35 MHz to 1.85 MHz in [Section 4.1.2 “SPI Rate”](#)
- Added [Figure 5-2](#) and updated [Figure 5-3](#) (formerly [Figure 5-2](#))
- Removed FUID2 and FUID3 from [Table 5-6](#)
- Added new Default Configuration Register Value table in support of dsPIC33FJ32GSX06/60X/610 and dsPIC33FJ64GSX06/60X/610 devices (see [Table 5-9](#))
- Updated description for FWDT and added Note 1 in [Table 5-10](#)
- Added seven new op code instructions to Step 3 in [Table 6-1](#)
- Updated Step 4 in [Table 6-1](#) to reflect the TBLPG pointer increment
- Added new columns (Application ID and JTAG ID) to [Table 7-1](#)
- Added JTAG Type registers (see [Register 7-1](#), [Register 7-2](#) and [Register 7-3](#))
- The following updates were made to [Table 8-1](#)
  - Updated parameters P1A and P1B and added additional characteristics and values
  - Updated characteristic for parameter P16
  - Updated minimum value and units for parameter P18
  - Added new parameter P21

# dsPIC33F/PIC24H PROGRAMMING SPECIFICATION

---

## Revision H (October 2010)

This revision includes the following updates:

- Text and formatting updates have been incorporated throughout the document
- All references to VCAP/VDDCORE have been changed to: VCAP
- All occurrences of PGC and PGD have been changed to PGCx and PGDx, respectively
- Added topics covered in [Section 1.0 “Device Overview”](#)
- Moved the Checksum Computation table to the appendix (see [Table D-1](#))
- Updated all occurrences of TBLPG to TBLPAG throughout the document
- Removed the ERASEB command from the Command Set Summary (see [Table 3-1](#))
- Updated the High-Level Enhanced ICSP™ Programming Flow (see [Figure 3-1](#))
- Replaced the Device Configuration Register Map (previously [Table 4-3](#)) with individual tables for each dsPIC33F/PIC24H device family (see [Table 3-3](#) through [Table 3-12](#))
- Updated the Note in [Section 3.6.2 “Programming Methodology”](#)
- Changed Opcode 0x7 to Reserved in the Programming Executive Command Set (see [Table 4-1](#))
- Removed 4.2.10 “ERASEB Command”
- Updated the table cross-references in [Section 5.7 “Writing Configuration Memory”](#)
- Combined all Default Configuration Register Values tables into one table (see [Table 5-6](#))
- Relocated the paragraph on ICSP programming details, which now appears just before [Table 5-7](#)
- Added a Note to [Section 6.1 “Overview”](#)
- Updated Step 4 in Programming the Programming Executive (see [Table 6-1](#))
- Updated Device IDs and Revision IDs (see [Table 7-1](#))
- Updated parameters D111, P1, P1A and P1B in the AC/DC Characteristics and Timing Requirements (see [Table 8-1](#))
- Added Checksum Computation Example When Using CodeGuard™ Security (see [Table D-2](#)).

---

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

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

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

---

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

**Trademarks**

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


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

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

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

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

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

 Printed on recycled paper.

ISBN: 978-1-60932-596-1

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

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



# MICROCHIP

## 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](http://www.microchip.com)

**Atlanta**  
Duluth, GA  
Tel: 678-957-9614  
Fax: 678-957-1455

**Boston**  
Westborough, MA  
Tel: 774-760-0087  
Fax: 774-760-0088

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

**Cleveland**  
Independence, OH  
Tel: 216-447-0464  
Fax: 216-447-0643

**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

**Santa Clara**  
Santa Clara, CA  
Tel: 408-961-6444  
Fax: 408-961-6445

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

### ASIA/PACIFIC

**Asia Pacific Office**  
Suites 3707-14, 37th Floor  
Tower 6, The Gateway  
Harbour City, Kowloon  
Hong Kong  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**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: 86-28-8665-5511  
Fax: 86-28-8665-7889

**China - Chongqing**  
Tel: 86-23-8980-9588  
Fax: 86-23-8980-9500

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

**China - Nanjing**  
Tel: 86-25-8473-2460  
Fax: 86-25-8473-2470

**China - Qingdao**  
Tel: 86-532-8502-7355  
Fax: 86-532-8502-7205

**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 - Wuhan**  
Tel: 86-27-5980-5300  
Fax: 86-27-5980-5118

**China - Xian**  
Tel: 86-29-8833-7252  
Fax: 86-29-8833-7256

**China - Xiamen**  
Tel: 86-592-2388138  
Fax: 86-592-2388130

**China - Zhuhai**  
Tel: 86-756-3210040  
Fax: 86-756-3210049

### ASIA/PACIFIC

**India - Bangalore**  
Tel: 91-80-3090-4444  
Fax: 91-80-3090-4123

**India - New Delhi**  
Tel: 91-11-4160-8631  
Fax: 91-11-4160-8632

**India - Pune**  
Tel: 91-20-2566-1512  
Fax: 91-20-2566-1513

**Japan - Yokohama**  
Tel: 81-45-471- 6166  
Fax: 81-45-471-6122

**Korea - Daegu**  
Tel: 82-53-744-4301  
Fax: 82-53-744-4302

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

**Malaysia - Kuala Lumpur**  
Tel: 60-3-6201-9857  
Fax: 60-3-6201-9859

**Malaysia - Penang**  
Tel: 60-4-227-8870  
Fax: 60-4-227-4068

**Philippines - Manila**  
Tel: 63-2-634-9065  
Fax: 63-2-634-9069

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

**Taiwan - Hsin Chu**  
Tel: 886-3-6578-300  
Fax: 886-3-6578-370

**Taiwan - Kaohsiung**  
Tel: 886-7-213-7830  
Fax: 886-7-330-9305

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

**Thailand - Bangkok**  
Tel: 66-2-694-1351  
Fax: 66-2-694-1350

### EUROPE

**Austria - Wels**  
Tel: 43-7242-2244-39  
Fax: 43-7242-2244-393

**Denmark - Copenhagen**  
Tel: 45-4450-2828  
Fax: 45-4485-2829

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

**Germany - Munich**  
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



**Spain - Madrid**  
Tel: 34-91-708-08-90  
Fax: 34-91-708-08-91

**UK - Wokingham**  
Tel: 44-118-921-5869  
Fax: 44-118-921-5820

08/04/10

## Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

-  [View PIC24HJ64GP206A-I/MR on WIN SOURCE](#)
-  [Microchip Technology](#) Information

## Optimize Your Supply Chain with WIN SOURCE Solutions

-  Global Sourcing Solution
-  Obsolete Management
-  Cost Control Management
-  Shortage Management
-  Alternative Solution
-  Excess Inventory Management