Welcome to E-XFL.COM

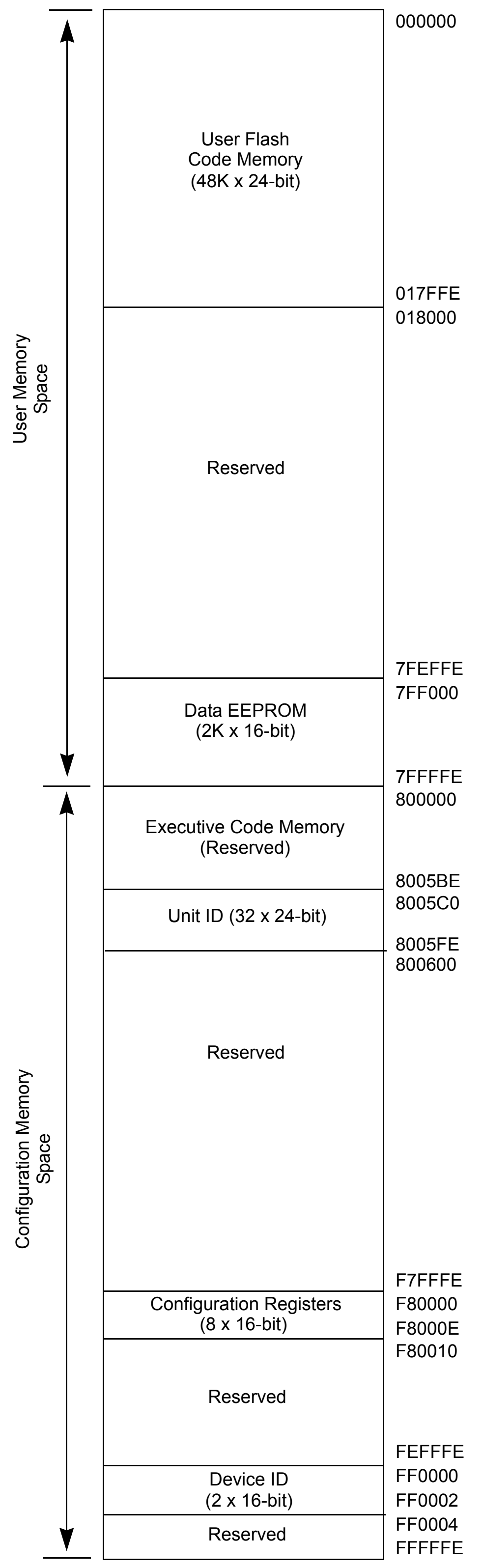
What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	dsPIC
Core Size	16-Bit
Speed	20 MIPS
Connectivity	CANbus, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, Motor Control PWM, QEI, POR, PWM, WDT
Number of I/O	68
Program Memory Size	144KB (48K x 24)
Program Memory Type	FLASH
EEPROM Size	4K x 8
RAM Size	8K x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	80-TQFP
Supplier Device Package	80-TQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/dspic30f6010t-20i-pf

FIGURE 2-2: PROGRAM MEMORY MAP



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

5.5.3 PROGRAMMING VERIFICATION

Once 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 once the entire device is programmed using a checksum computation, as described in **Section 6.8 "Checksum Computation"**.

5.6 Data EEPROM Programming

5.6.1 OVERVIEW

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

TABLE 5-3: DATA EEPROM SIZE

Device	Data EEPROM Size (Words)	Number of Rows
dsPIC30F2010	512	32
dsPIC30F2011	0	0
dsPIC30F2012	0	0
dsPIC30F3010	512	32
dsPIC30F3011	512	32
dsPIC30F3012	512	32
dsPIC30F3013	512	32
dsPIC30F3014	512	32
dsPIC30F4011	512	32
dsPIC30F4012	512	32
dsPIC30F4013	512	32
dsPIC30F5011	512	32
dsPIC30F5013	512	32
dsPIC30F5015	512	32
dsPIC30F5016	512	32
dsPIC30F6010	2048	128
dsPIC30F6010A	2048	128
dsPIC30F6011	1024	64
dsPIC30F6011A	1024	64
dsPIC30F6012	2048	128
dsPIC30F6012A	2048	128
dsPIC30F6013	1024	64
dsPIC30F6013A	1024	64
dsPIC30F6014	2048	128
dsPIC30F6014A	2048	128
dsPIC30F6015	2048	128

5.6.2 PROGRAMMING METHODOLOGY

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

The first `PROGD` command programs the first row of data EEPROM. Once the command completes successfully, 'RemainingRows' is decremented by 1 and compared with 0. Since there are 127 more rows to program, 'BaseAddress' is incremented by 0x20 to point to the next row of data EEPROM. This process is then repeated until all 128 rows of data EEPROM are programmed.

FIGURE 5-4: FLOWCHART FOR PROGRAMMING dsPIC30F6014A DATA EEPROM

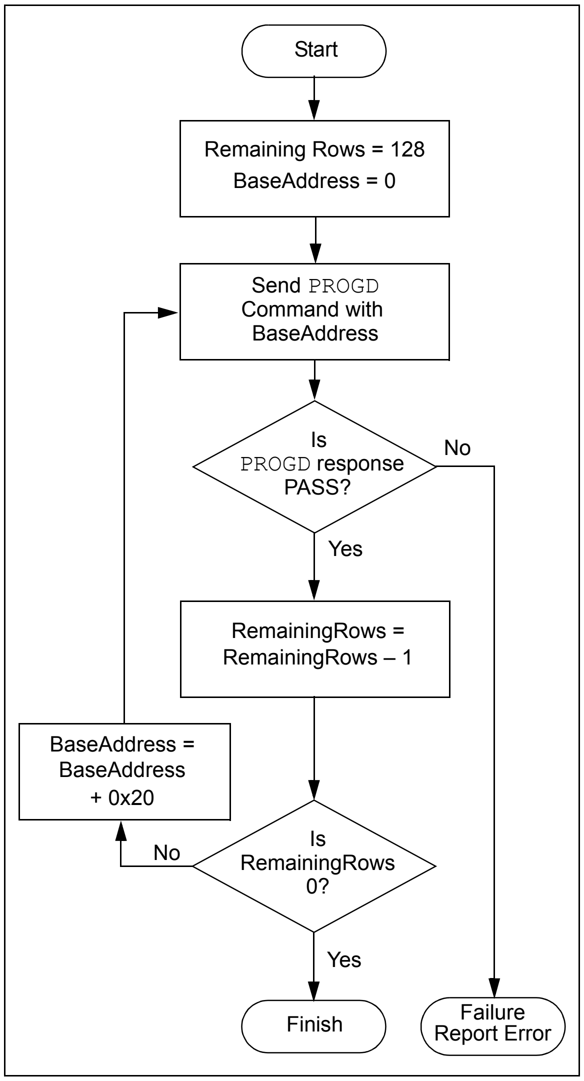


TABLE 5-6: FOSC CONFIGURATION BITS DESCRIPTION FOR dsPIC30F2011/2012, dsPIC30F3010/3011/3012/3013/3014, dsPIC30F4013, dsPIC30F5015/5016, dsPIC30F6010A/6011A/6012A/6013A/6014A AND dsPIC30F6015

Bit Field	Register	Description
FCKSM<1:0>	FOSC	**Clock Switching Mode** `1x` = Clock switching is disabled, Fail-Safe Clock Monitor is disabled `01` = Clock switching is enabled, Fail-Safe Clock Monitor is disabled `00` = Clock switching is enabled, Fail-Safe Clock Monitor is enabled
FOS<2:0>	FOSC	**Oscillator Source Selection on POR** `111` = Primary Oscillator `110` = Reserved `101` = Reserved `100` = Reserved `011` = Reserved `010` = Internal Low-Power RC Oscillator `001` = Internal Fast RC Oscillator (no PLL) `000` = Low-Power 32 kHz Oscillator (Timer1 Oscillator)
FPR<4:0>	FOSC	**Primary Oscillator Mode (when FOS<2:0> = `111b`)** `11xxx` = Reserved (do not use) `10111` = HS/3 w/PLL 16X – HS/3 crystal oscillator with 16X PLL (10 MHz-25 MHz crystal) `10110` = HS/3 w/PLL 8X – HS/3 crystal oscillator with 8X PLL (10 MHz-25 MHz crystal) `10101` = HS/3 w/PLL 4X – HS/3 crystal oscillator with 4X PLL (10 MHz-25 MHz crystal) `10100` = Reserved (do not use) `10011` = HS/2 w/PLL 16X – HS/2 crystal oscillator with 16X PLL (10 MHz-25 MHz crystal) `10010` = HS/2 w/PLL 8X – HS/2 crystal oscillator with 8X PLL (10 MHz-25 MHz crystal `10001` = HS/2 w/PLL 4X – HS/2 crystal oscillator with 4X PLL (10 MHz-25 MHz crystal) `10000` = Reserved (do not use) `01111` = ECIO w/PLL 16x – External clock with 16x PLL. OSC2 pin is I/O `01110` = ECIO w/PLL 8x – External clock with 8x PLL. OSC2 pin is I/O `01101` = ECIO w/PLL 4x – External clock with 4x PLL. OSC2 pin is I/O `01100` = Reserved (do not use) `01011` = Reserved (do not use) `01010` = FRC w/PLL 8x – Internal fast RC oscillator with 8x PLL. OSC2 pin is I/O `01001` = Reserved (do not use) `01000` = Reserved (do not use) `00111` = XT w/PLL 16X – XT crystal oscillator with 16X PLL `00110` = XT w/PLL 8X – XT crystal oscillator with 8X PLL `00101` = XT w/PLL 4X – XT crystal oscillator with 4X PLL `00100` = Reserved (do not use) `00011` = FRC w/PLL 16x – Internal fast RC oscillator with 8x PLL. OSC2 pin is I/O `00010` = Reserved (do not use) `00001` = FRC w/PLL 4x – Internal fast RC oscillator with 4x PLL. OSC2 pin is I/O `00000` = Reserved (do not use)

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

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

TABLE 5-8: dsPIC30F CONFIGURATION REGISTERS (FOR dsPIC30F2010, dsPIC30F4011/4012 AND dsPIC30F6010/ 6011/6012/6013/ 6014)

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

Note 1: On the 6011, 6012, 6013 and 6014, these bits are reserved (read as '1' and must be programmed as '1').
2: Reserved bits read as '1' and must be programmed as '1'.

TABLE 5-9: dsPIC30F CONFIGURATION REGISTERS (FOR dsPIC30F5011/5013)

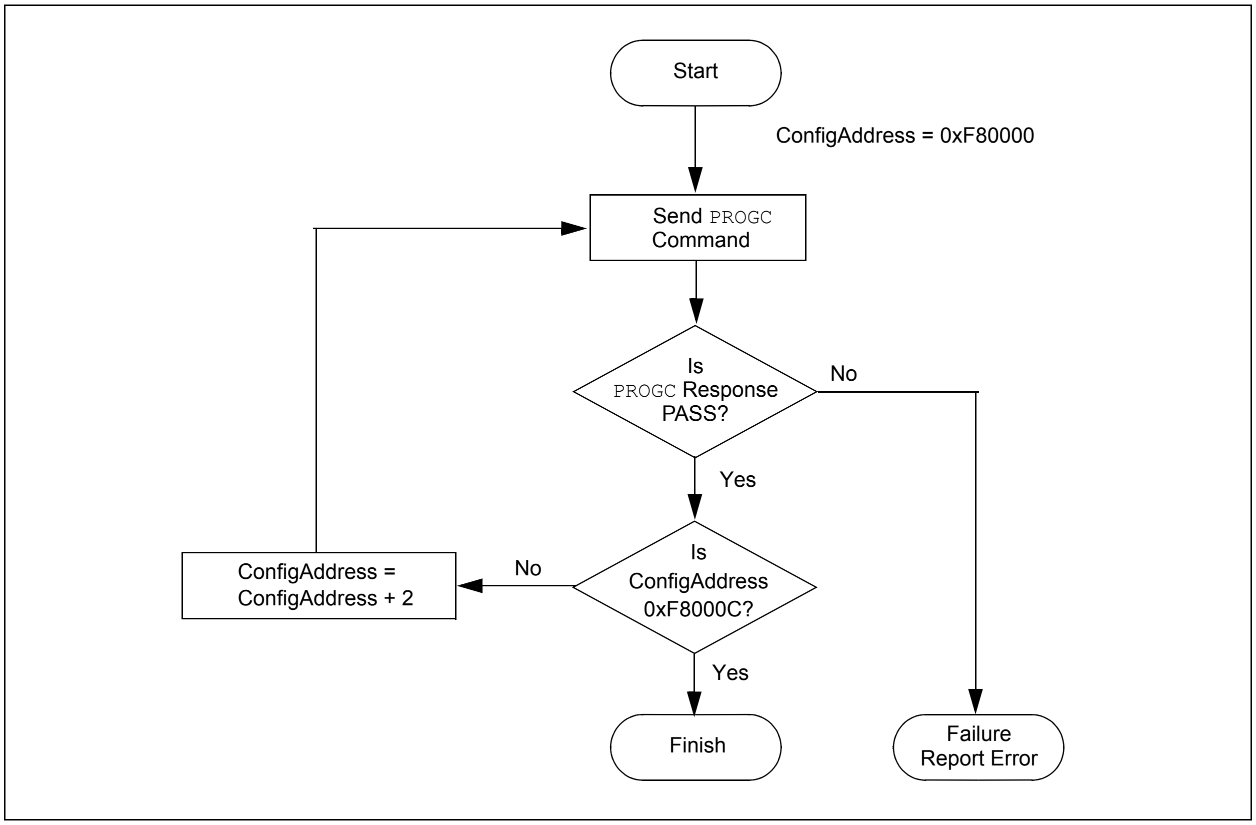
Address	Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xF80000	FOSC	FCKSM<1:0>		—	—	—	—	FOS<1:0>		—	—	—	—	FPR<3:0>			
0xF80002	FWDT	FWDTEN	—	—	—	—	—	—	—	—	—	FWPSA<1:0>		FWPSB<3:0>			
0xF80004	FBORPOR	MCLREN	—	—	—	—	Reserved(1)			BOREN	—	BORV<1:0>		—	—	FPWRT<1:0>	
0xF80006	FBS	—	—	RBS<1:0>		—	—	—	EBS	—	—	—	—	BSS<2:0>			BWRP
0xF80008	FSS	—	—	RSS<1:0>		—	—	ESS<1:0>		—	—	—	—	SSS<2:0>			SWRP
0xF8000A	FGS	—	—	—	—	—	—	—	—	—	—	—	—	—	GSS<1:0>		GWRP
0xF8000C	FICD	BKBUG	COE	—	—	—	—	—	—	—	—	—	—	—	—	ICS<1:0>	

Note 1: Reserved bits read as '1' and must be programmed as '1'.

5.8 Exiting Enhanced ICSP Mode

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

FIGURE 5-5: CONFIGURATION BIT PROGRAMMING FLOW



6.0 OTHER PROGRAMMING FEATURES

6.1 Erasing Memory

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

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

TABLE 6-1: ERASE OPTIONS

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

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

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

6.2 Modifying Memory

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

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

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

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

6.3 Reading Memory

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

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

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

6.4 Programming Executive Software Version

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

6.5 Data EEPROM Information in the Hexadecimal File

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

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

TABLE 8-1: PROGRAMMING EXECUTIVE COMMAND SET

Opcode	Mnemonic	Length (16-bit words)	Time Out	Description
0x0	`SCHECK`	1	1 ms	Sanity check.
0x1	`READD`	4	1 ms/row	Read N 16-bit words of data EEPROM, Configuration registers or device ID starting from specified address.
0x2	`READP`	4	1 ms/row	Read N 24-bit instruction words of code memory starting from specified address.
0x3	Reserved	N/A	N/A	This command is reserved. It will return a NACK.
0x4	`PROGD`(2)	19	5 ms	Program one row of data EEPROM at the specified address, then verify.
0x5	`PROGP`(1)	51	5 ms	Program one row of code memory at the specified address, then verify.
0x6	`PROGC`	4	5 ms	Write byte or 16-bit word to specified Configuration register.
0x7	`ERASEB`	2	5 ms	Bulk Erase (entire code memory or data EEPROM), or erase by segment.
0x8	`ERASED`(2)	3	5 ms/row	Erase rows of data EEPROM from specified address.
0x9	`ERASEP`(1)	3	5 ms/row	Erase rows of code memory from specified address.
0xA	`QBLANK`	3	300 ms	Query if the code memory and data EEPROM are blank.
0xB	`QVER`	1	1 ms	Query the programming executive software version.

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

8.5 Command Descriptions

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

8.5.1 `SCHECK` COMMAND

15–12	11–0
Opcode	Length

Field	Description
Opcode	0x0
Length	0x1

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

Expected Response (2 words):
0x1000
0x0002

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

8.5.2 `READD` COMMAND

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

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

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

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

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

9.2.3 QE_Code FIELD

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

When the programming executive processes one of the two query commands (`QBLANK` or `QVER`), the returned opcode is always PASS and the QE_Code holds the query response data. The format of the QE_Code for both queries is shown in Table 9-3.

TABLE 9-3: QE_Code FOR QUERIES

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

When the programming executive processes any command other than a Query, the QE_Code represents an error code. Supported error codes are shown in Table 9-4. If a command is successfully processed, the returned QE_Code is set to 0x0, which indicates that there was no error in the command processing. If the verify of the programming for the `PROGD`, `PROGP` or `PROGC` command fails, the QE_Code is set to 0x1. For all other programming executive errors, the QE_Code is 0x2.

TABLE 9-4: QE_Code FOR NON-QUERY COMMANDS

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

9.2.4 RESPONSE LENGTH

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

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

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

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

11.0 ICSP™ MODE

11.1 ICSP Mode

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

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

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

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

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

11.2 ICSP Operation

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

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

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

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

11.2.1 SIX SERIAL INSTRUCTION EXECUTION

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

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

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

11.3 Entering ICSP Mode

The ICSP mode is entered by holding PGC and PGD low, raising $\overline{\text{MCLR}}$/VPP to VIHH (high voltage), and then performing additional steps as illustrated in Figure 11-4.

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

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

3: Before leaving the Reset vector, execute two `GOTO` instructions, followed by a single `NOP` instruction must be executed.

FIGURE 11-4: ENTERING ICSP™ MODE

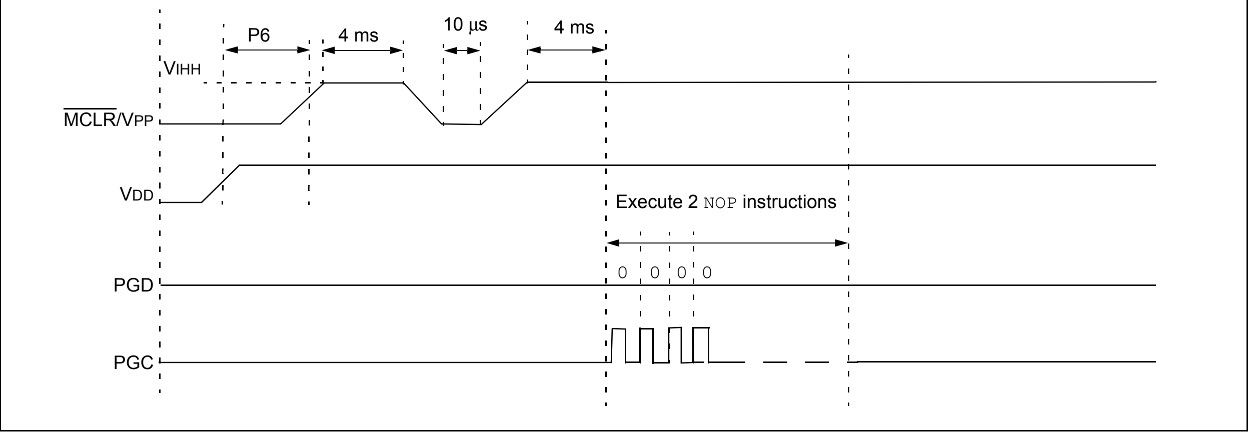


Table 11-4 shows the ICSP programming process for bulk-erasing program memory. This process includes the ICSP command code, which must be transmitted (for each instruction) to the Least Significant bit first using the PGC and PGD pins (see Figure 11-2).

If an individual Segment Erase operation is required, the NVMCON value must be replaced by the value for the corresponding Segment Erase operation.

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

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

Command (Binary)	Data (Hexadecimal)	Description
Step 1: Exit the Reset vector.		

0000
0000
0000
	040100
	040100
	000000
		`GOTO 0x100`
		`GOTO 0x100`
		`NOP`

Step 2: Set NVMCON to program the FBS Configuration register.(1)		

0000
0000
	24008A
	883B0A
		`MOV #0x4008, W10`
		`MOV W10, NVMCON`

Step 3: Initialize the TBLPAG and write pointer (W7) for TBLWT instruction for Configuration register.(1)		

0000
0000
0000
	200F80
	880190
	200067
		`MOV #0xF8, W0`
		`MOV W0, TBLPAG`
		`MOV #0x6, W7`

Step 4: Load the Configuration Register data to W6.(1)		

0000
0000
	EB0300
	000000
		`CLR W6`
		`NOP`

Step 5: Load the Configuration Register write latch. Advance W7 to point to next Configuration register.(1)		
0000	BB1B86	`TBLWTL W6, [W7++]`
Step 6: Unlock the NVMCON for programming the Configuration register.(1)		

0000
0000
0000
0000
	200558
	200AA9
	883B38
	883B39
		`MOV #0x55, W8`
		`MOV #0xAA, W9`
		`MOV W8, NVMKEY`
		`MOV W9, NVMKEY`

Step 7: Initiate the programming cycle.(1)		

0000
0000
0000
—
0000
0000
0000
0000
0000
	A8E761
	000000
	000000
	—
	000000
	000000
	A9E761
	000000
	000000
		`BSET NVMCON, #WR`
		`NOP`
		`NOP`
		`Externally time 2 ms`
		`NOP`
		`NOP`
		`BCLR NVMCON, #WR`
		`NOP`
		`NOP`

Step 8: Repeat steps 5-7 one time to program 0x0000 to RESERVED2 Configuration register.(1)		
Step 9: Set the NVMCON to erase all Program Memory.		

00000
0000
	2407FA
	883B0A
		`MOV #0x407F, W10`
		`MOV W10, NVMCON`

Step 10: Unlock the NVMCON for programming.		

Note 1: Steps 2-8 are only required for the dsPIC30F5011/5013 devices. These steps may be skipped for all other devices in the dsPIC30F family.

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

Command (Binary)	Data (Hexadecimal)	Description
Step 18: Unlock the NVMCON to erase 1 row of data memory.		
`0000`	`200558`	`MOV #0x55, W8`
`0000`	`883B38`	`MOV W8, NVMKEY`
`0000`	`200AA9`	`MOV #0xAA, W9`
`0000`	`883B39`	`MOV W9, NVMKEY`
Step 19: Initiate the erase cycle.		
`0000`	`A8E761`	`BSET NVMCON, #WR`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
—	—	`Externally time 'P13a' ms` (see **Section 13.0 “AC/DC Characteristics and Timing Requirements”**)
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`A9E761`	`BCLR NVMCON, #WR`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
Step 20: Update the row address stored in NVMADR.		
`0000`	`430307`	`ADD W6, W7, W6`
`0000`	`883B16`	`MOV W6, NVMADR`
Step 21: Reset device internal PC.		
`0000`	`040100`	`GOTO 0x100`
`0000`	`000000`	`NOP`
Step 22: Repeat Steps 17-21 until all rows of data memory are erased.		

11.8 Writing Code Memory

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

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

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

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

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

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

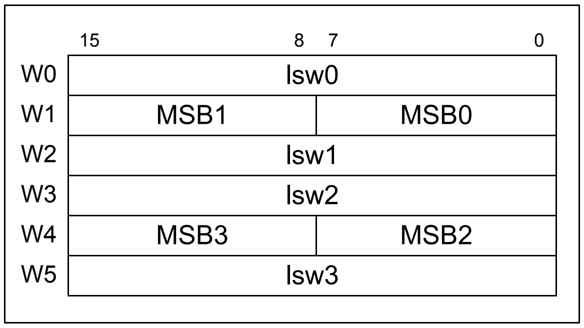


TABLE 11-8: SERIAL INSTRUCTION EXECUTION FOR WRITING CODE MEMORY

Command (Binary)	Data (Hexadecimal)	Description
Step 1: Exit the Reset vector.		

`0000`
`0000`
`0000`
	`040100`
	`040100`
	`000000`
		`GOTO 0x100`
		`GOTO 0x100`
		`NOP`

Step 2: Set the NVMCON to program 32 instruction words.		

`0000`
`0000`
	`24001A`
	`883B0A`
		`MOV #0x4001, W10`
		`MOV W10, NVMCON`

Step 3: Initialize the write pointer (W7) for `TBLWT` instruction.		

`0000`
`0000`
`0000`
	`200xx0`
	`880190`
	`2xxxx7`
		`MOV #<DestinationAddress23:16>, W0`
		`MOV W0, TBLPAG`
		`MOV #<DestinationAddress15:0>, W7`

Step 4: Initialize the read pointer (W6) and load W0:W5 with the next 4 instruction words to program.		

`0000`
`0000`
`0000`
`0000`
`0000`
`0000`
	`2xxxx0`
	`2xxxx1`
	`2xxxx2`
	`2xxxx3`
	`2xxxx4`
	`2xxxx5`
		`MOV #<LSW0>, W0`
		`MOV #<MSB1:MSB0>, W1`
		`MOV #<LSW1>, W2`
		`MOV #<LSW2>, W3`
		`MOV #<MSB3:MSB2>, W4`
		`MOV #<LSW3>, W5`

11.12 Reading Data Memory

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

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

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

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

11.13 Reading the Application ID Word

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

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

11.14 Exiting ICSP Mode

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

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

Command (Binary)	Data (Hexadecimal)	Description
Step 1: Exit the Reset vector.		
`0000`	`040100`	`GOTO 0x100`
`0000`	`040100`	`GOTO 0x100`
`0000`	`000000`	`NOP`
Step 2: Initialize TBLPAG and the read pointer (W0) for `TBLRD` instruction.		
`0000`	`200800`	`MOV #0x80, W0`
`0000`	`880190`	`MOV W0, TBLPAG`
`0000`	`205BE0`	`MOV #0x5BE, W0`
`0000`	`207841`	`MOV VISI, W1`
`0000`	`000000`	`NOP`
`0000`	`BA0890`	`TBLRDL [W0], [W1]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
Step 3: Output the VISI register using the REGOUT command.		
`0001`	`<VISI>`	`Clock out contents of the VISI register`
`0000`	`000000`	`NOP`

13.0 AC/DC CHARACTERISTICS AND TIMING REQUIREMENTS

TABLE 13-1: AC/DC CHARACTERISTICS

AC/DC CHARACTERISTICS			**Standard Operating Conditions (unless otherwise stated)** Operating Temperature: 25° C is recommended			
Param. No.	**Sym**	**Characteristic**	**Min**	**Max**	**Units**	**Conditions**
D110	V_{IHH}	High Programming Voltage on $\overline{MCLR}/V_{PP}$	9.00	13.25	V	—
D112	I_{PP}	Programming Current on $\overline{MCLR}/V_{PP}$	—	300	μA	—
D113	I_{DDP}	Supply Current during programming	—	30	mA	Row Erase Program memory
			—	30	mA	Row Erase Data EEPROM
			—	30	mA	Bulk Erase
D001	V_{DD}	Supply voltage	2.5	5.5	V	—
D002	V_{DDBULK}	Supply voltage for Bulk Erase programming	4.5	5.5	V	—
D031	V_{IL}	Input Low Voltage	V_{SS}	0.2 V_{SS}	V	—
D041	V_{IH}	Input High Voltage	0.8 V_{DD}	V_{DD}	V	—
D080	V_{OL}	Output Low Voltage	—	0.6	V	I_{OL} = 8.5 mA
D090	V_{OH}	Output High Voltage	V_{DD} - 0.7	—	V	I_{OH} = -3.0 mA
D012	C_{IO}	Capacitive Loading on I/O Pin (PGD)	—	50	pF	To meet AC specifications
P1	T_{SCLK}	Serial Clock (PGC) period	50	—	ns	ICSP™ mode
			1	—	μs	Enhanced ICSP mode
P1a	T_{SCLKL}	Serial Clock (PGC) low time	20	—	ns	ICSP mode
			400	—	ns	Enhanced ICSP mode
P1b	T_{SCLKH}	Serial Clock (PGC) high time	20	—	ns	ICSP mode
			400	—	ns	Enhanced ICSP mode
P2	T_{SET1}	Input Data Setup Timer to PGC ↓	15	—	ns	—
P3	T_{HLD1}	Input Data Hold Time from PGC ↓	15	—	ns	—
P4	T_{DLY1}	Delay between 4-bit command and command operand	20	—	ns	—
P4a	T_{DLY1a}	Delay between 4-bit command operand and next 4-bit command	20	—	ns	—
P5	T_{DLY2}	Delay between last PGC ↓of command to first PGC ↑ of VISI output	20	—	ns	—
P6	T_{SET2}	V_{DD} ↑ setup time to $\overline{MCLR}/V_{PP}$	100	—	ns	—
P7	T_{HLD2}	Input data hold time from $\overline{MCLR}/V_{PP}$ ↑	2	—	μs	ICSP mode
			5	—	ms	Enhanced ICSP mode
P8	T_{DLY3}	Delay between last PGC ↓of command word to PGD driven ↑ by programming executive	20	—	μs	—
P9a	T_{DLY4}	Programming Executive Command processing time	10	—	μs	—

TABLE A-1: CHECKSUM COMPUTATION (CONTINUED)

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

Item Description:

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

CFGB = **Configuration Block (masked)** = Byte sum of ((FOSC&0xC10F) + (FWDT&0x803F) + (FBORPOR&0x87B3) + (FBS&0x310F) + (FSS&0x330F) + (FGS&0x0007) + (FICD&0xC003))

NOTES: