PIC24FJ64GA1/GB0 Families Flash Programming Specification

1.0 DEVICE OVERVIEW

This document defines the programming specification for the PIC24FJ64GA1/GB0 families of 16-bit microcontroller devices. This programming specification is required only for those developing programming support for the PIC24FJ64GA1/GB0 families. Customers using only one of these devices should use development tools that already provide support for device programming.

This specification includes programming specifications for the following devices:

- PIC24FJ32GA102
- PIC24FJ32GA104
- PIC24FJ32GB002
- PIC24FJ32GB004
- PIC24FJ64GA102
- PIC24FJ64GA104
- PIC24FJ64GB002
- PIC24FJ64GB004

2.0 PROGRAMMING OVERVIEW OF THE PIC24FJ64GA1/GB0 FAMILIES

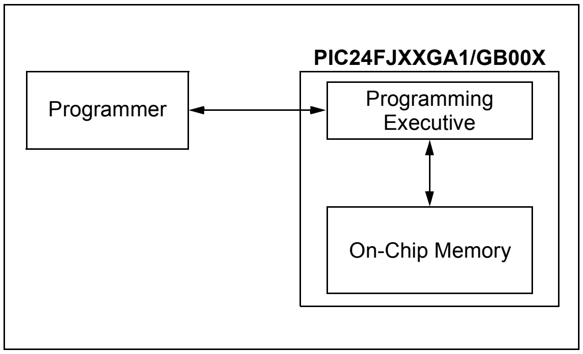
There are two methods of programming the PIC24FJ64GA1/GB0 families of devices discussed in this programming specification. They are:

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

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 In-Circuit Serial Programming (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 PIC24FJ64GA1/GB0 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 4.0 "Device Programming – Enhanced ICSP"** describes the Run-Time Self-Programming (RTSP) method. **Section 3.0 "Device Programming – ICSP"** describes the In-Circuit Serial Programming method.

2.1 Power Requirements

All devices in the PIC24FJ64GA1/GB0 families are dual voltage supply designs: one supply for the core and peripherals, and another for the I/O pins. A regulator is provided on-chip to alleviate the need for two external voltage supplies.

All PIC24FJ64GA1/GB0 devices power their core digital logic at a nominal 2.5V. To simplify system design, all devices in the PIC24FJ64GA1/GB0 families incorporate an on-chip regulator that allows the device to run its core logic from V_{DD}.

The regulator provides power to the core from the other V_{DD} pins. A low-ESR capacitor (such as tantalum) must be connected to the V_{DDCORE} pin (Table 2-1 and Figure 2-2). This helps to maintain the stability of the regulator. The specifications for core voltage and capacitance are listed in **Section 7.0 "AC/DC Characteristics and Timing Requirements"**.

2.2 Program Memory Write/Erase Requirements

The Flash program memory on PIC24FJ64GA1/GB0 devices 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 more than twice before 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 specification comply with this requirement.

Note:	Writing to a location multiple times without erasing is *not* recommended.

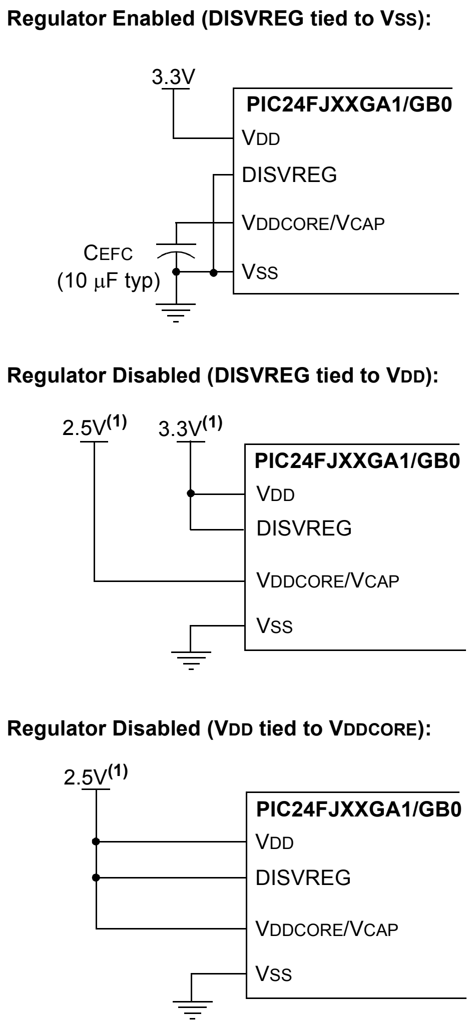
2.3 Pin Diagrams

The pin diagrams for the PIC24FJ64GA1/GB0 families are shown in Figure 2-3 through Figure 2-6. The pins that are required for programming are listed in Table 2-1 and are shown in bold letters in the figures. Refer to the appropriate device data sheet for complete pin descriptions.

2.3.1 PGCx AND PGDx PIN PAIRS

All of the devices in the PIC24FJ64GA1/GB0 families have three separate pairs of programming pins, labelled as PGEC1/PGED1, PGEC2/PGED2 and PGEC3/PGED3. Any one of these pin pairs may be used for device programming by either ICSP or Enhanced ICSP. Unlike voltage supply and ground pins, it is not necessary to connect all three pin pairs to program the device. However, the programming method must use both pins of the same pair.

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



Regulator Enabled (DISVREG tied to V_{SS}):

Regulator Disabled (DISVREG tied to V_{DD}):

Regulator Disabled (V_{DD} tied to V_{DDCORE}):

Note 1: These are typical operating voltages. Refer to **Section 7.0 "AC/DC Characteristics and Timing Requirements"** for the full operating ranges of V_{DD} and V_{DDCORE}.

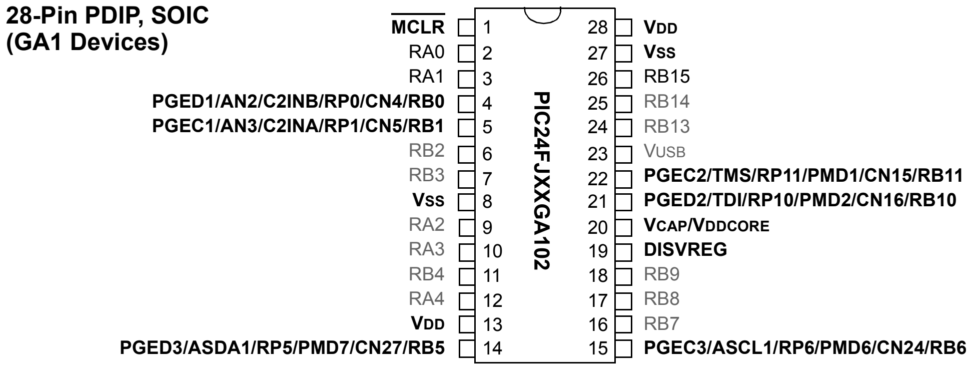
TABLE 2-1: PIN DESCRIPTIONS (DURING PROGRAMMING)

Pin Name	During Programming		
	Pin Name	Pin Type	Pin Description
$\overline{\text{MCLR}}$	$\overline{\text{MCLR}}$	P	Programming Enable
DISVREG	DISVREG	I	Disable for On-Chip Voltage Regulator
VDD and AVDD(1)	VDD	P	Power Supply
VSS and AVSS(1)	VSS	P	Ground
VDDCORE	VDDCORE	P	Regulated Power Supply for Core
PGECx	PGCx	I	Programming Pin Pairs 1, 2 and 3: Serial Clock
PGEDx	PGDx	I/O	Programming Pin Pairs 1, 2 and 3: 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).

FIGURE 2-3: PIN DIAGRAMS


28-Pin PDIP, SOIC (GA1 Devices)

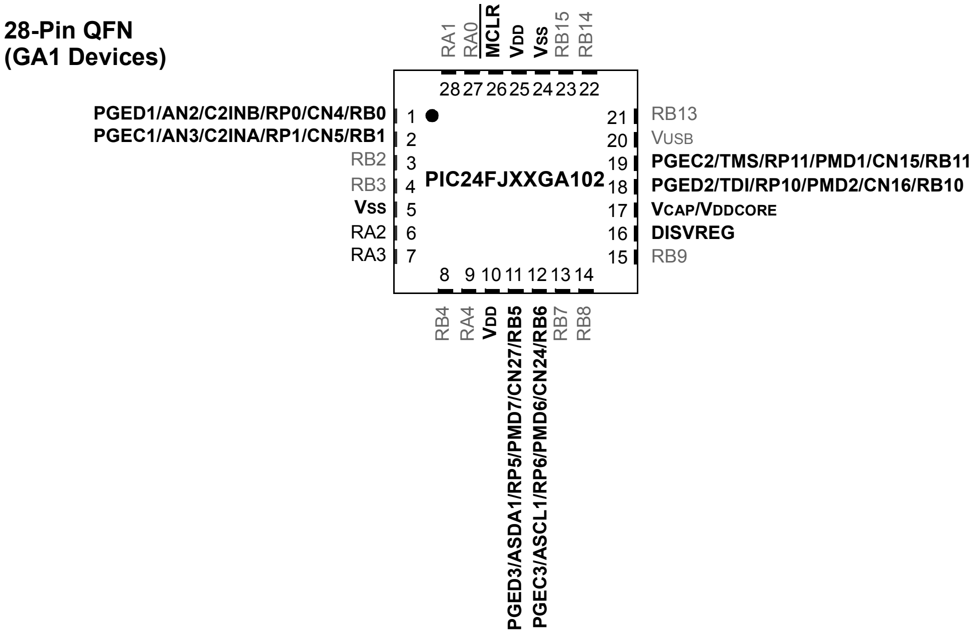
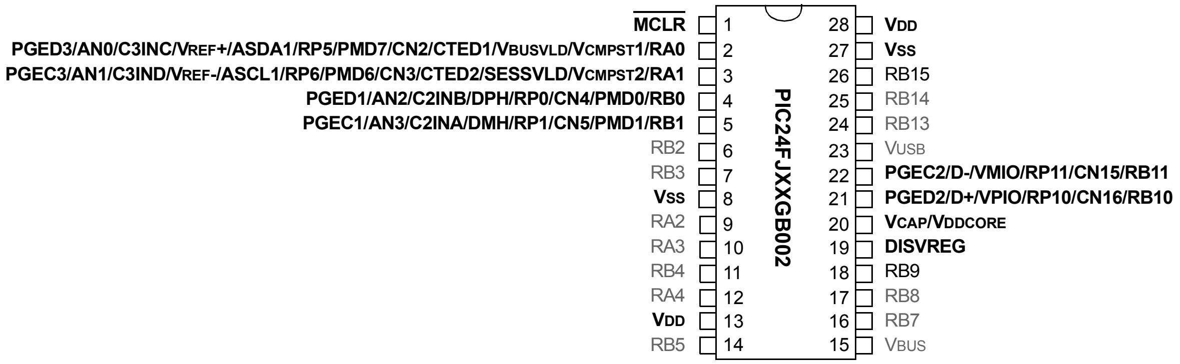

28-Pin QFN (GA1 Devices)

FIGURE 2-4: PIN DIAGRAMS (CONTINUED)

28-Pin PDIP, SOIC
(GB0 Devices)



28-Pin QFN
(GB0 Devices)

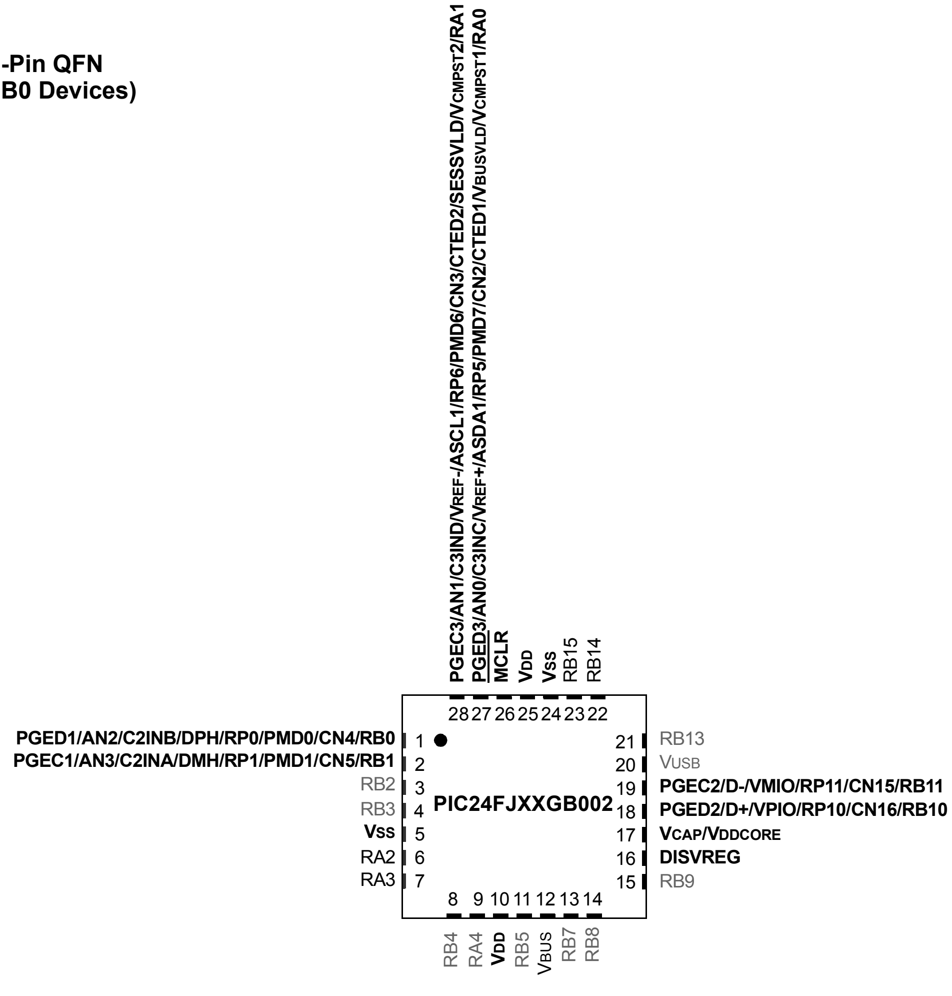


FIGURE 2-5: PIN DIAGRAMS (CONTINUED)

44-Pin TQFP, QFN
(GA1 Devices)

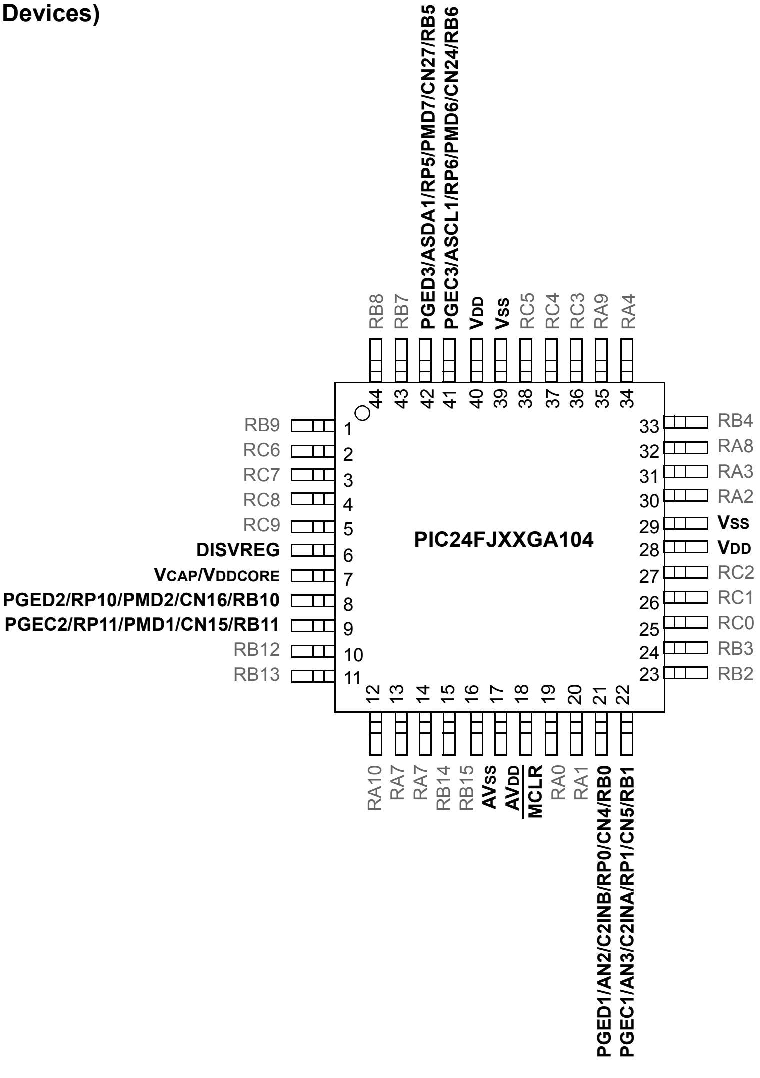
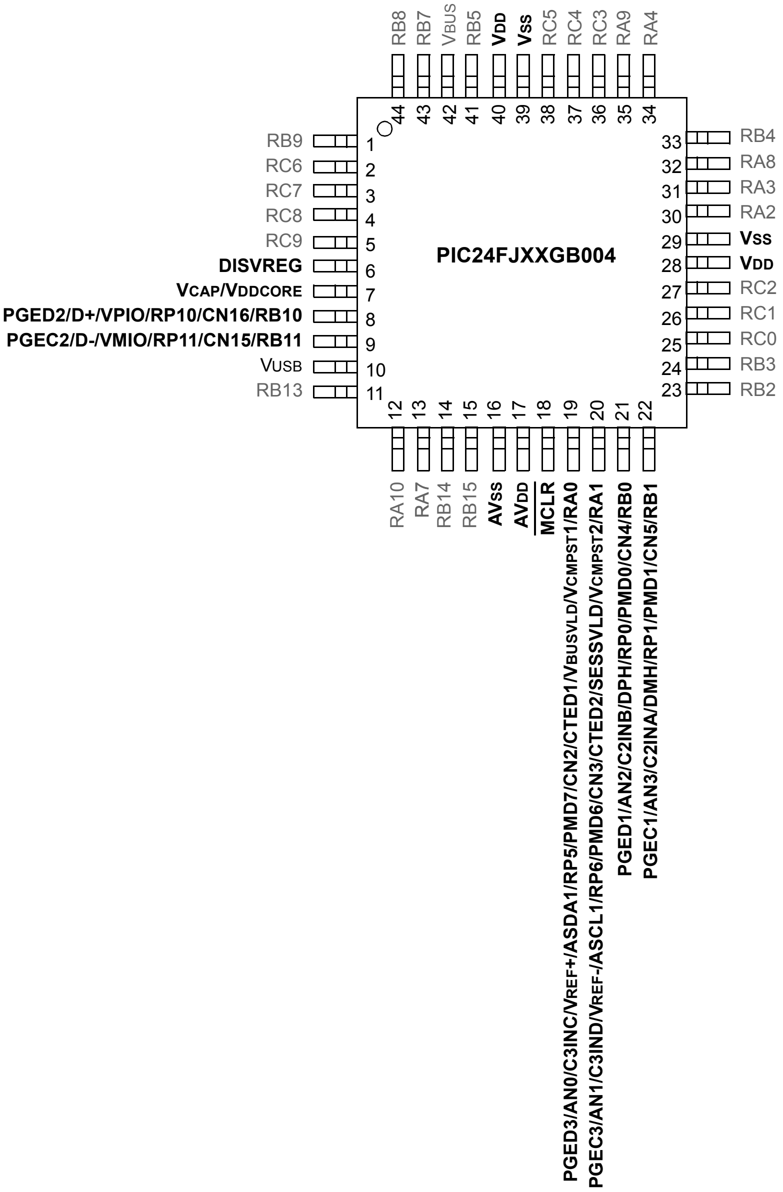


FIGURE 2-6: PIN DIAGRAMS (CONTINUED)

44-Pin TQFP, QFN
(GB0 Devices)



2.4 Memory Map

The program memory map extends from 000000h to FFFFFEh. Code storage is located at the base of the memory map and supports up to 22K instruction words (about 64 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.

The last four implemented program memory locations are reserved for the Flash Configuration Words. The reserved addresses are also shown in Table 2-2.

Locations, 800000h through 8007FEh, 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 debugging executive is used for in-circuit debugging. This region of memory can not be used to store user code.

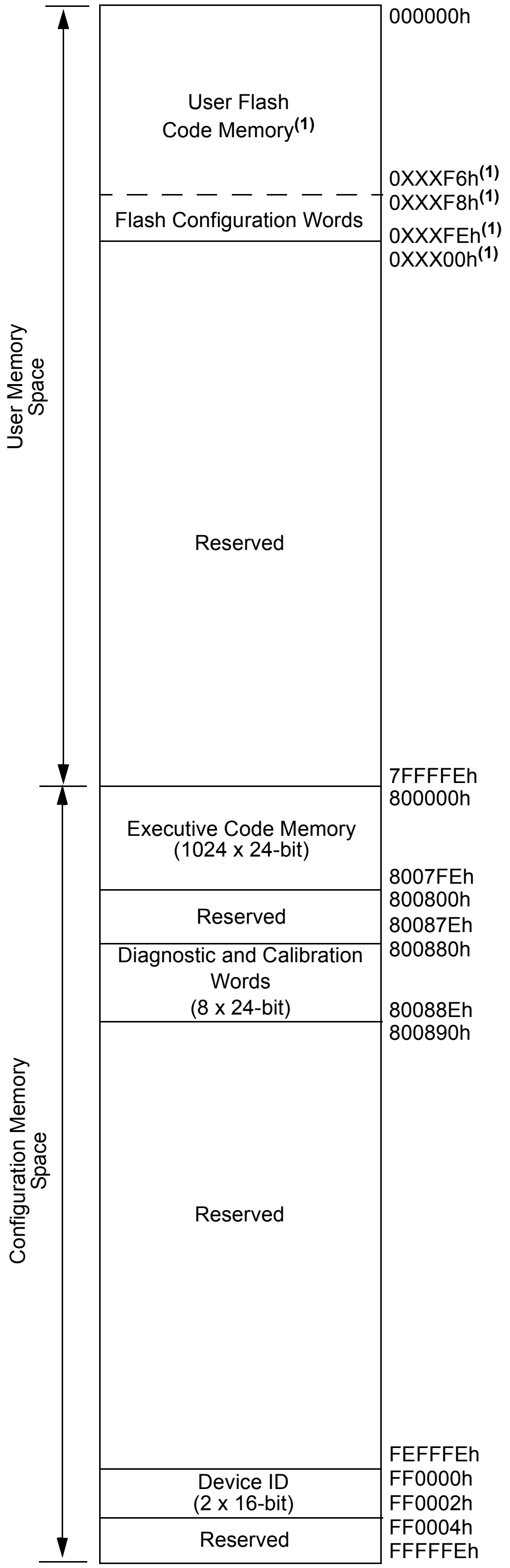
Locations, FF0000h and FF0002h, are reserved for the Device ID registers. These bits can be used by the programmer to identify what device type is being programmed. They are described in **Section 6.1 "Device ID"**. The Device ID registers read out normally, even after code protection is applied.

Figure 2-7 shows the memory map for the PIC24FJ64GA1/GB0 family variants.

TABLE 2-2: CODE MEMORY SIZE AND FLASH CONFIGURATION WORD LOCATIONS FOR PIC24FJ64GA1/GB0 DEVICES

Device	User Memory Address Limit (Instruction Words)	Write Blocks	Erase Blocks	Configuration Word Addresses			
				1	2	3	4
PIC24FJ32GA10X	0057FEh (11K)	176	22	0057FEh	0057FCh	0057FAh	0057F8h
PIC24FJ32GB00X							
PIC24FJ64GA10X	00ABFEh (22K)	344	43	00ABFEh	00ABFCh	00ABFAh	00ABF8h
PIC24FJ64GB00X							

FIGURE 2-7: PROGRAM MEMORY MAP



Note 1: The size and address boundaries for user Flash code memory are device dependent. See Table 2-2 for details.

3.0 DEVICE PROGRAMMING – ICSP

ICSP mode is a special programming protocol that allows you to read and write to the memory of PIC24FJ64GA1/GB0 devices. The ICSP mode is the most direct method used to program the device; note, however, 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.

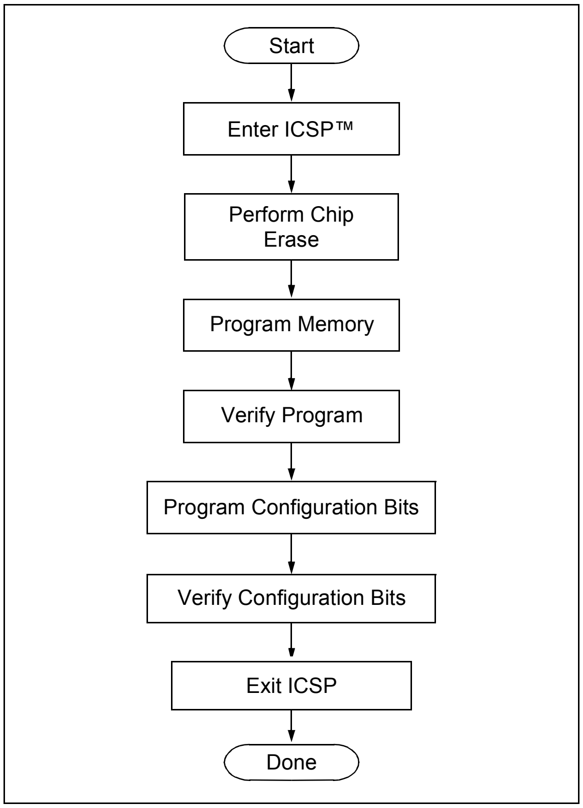
In ICSP mode, the system clock is taken from the PGCx pin, regardless of the device's oscillator Configuration bits. All instructions are shifted serially into an internal buffer, then loaded into the Instruction Register (IR) and executed. No program fetching occurs from internal memory. Instructions 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 10 MHz.

3.1 Overview of the Programming Process

Figure 3-1 shows the high-level overview of the programming process. After entering ICSP mode, the first action is to Chip 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 3-1: HIGH-LEVEL ICSP™ PROGRAMMING FLOW



3.2 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 PGCx and PGDx, and this control code is used to command the CPU (see Table 3-1).

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 3-1: CPU CONTROL CODES IN ICSP™ MODE

4-Bit Control Code	Mnemonic	Description
`0000`	SIX	Shift in 24-bit instruction and execute.
`0001`	REGOUT	Shift out the VISI (0784h) register.
`0010-1111`	N/A	Reserved.

3.2.1 SIX SERIAL INSTRUCTION EXECUTION

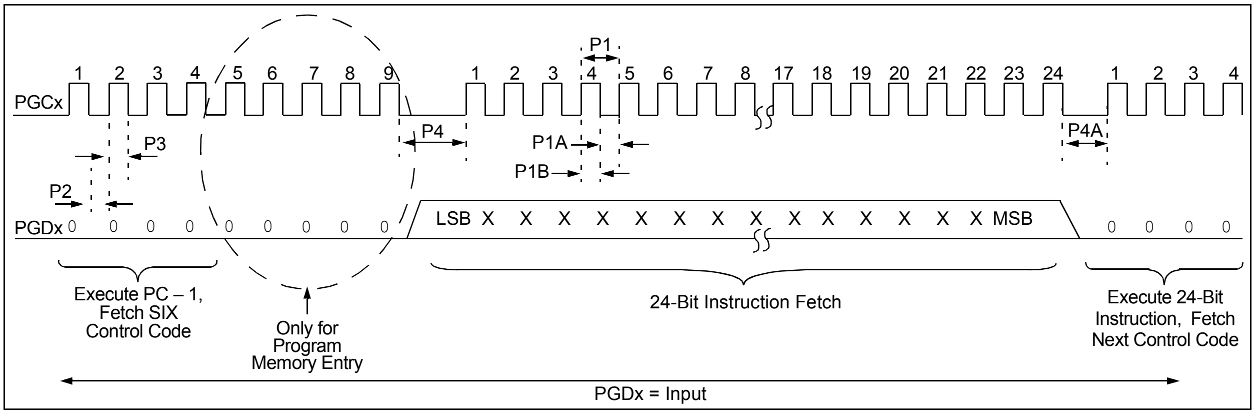
The SIX control code allows execution of PIC24F family assembly instructions. When the SIX code is received, the CPU is suspended for 24 clock cycles, as the instruction is then clocked into the internal buffer. Once the instruction is shifted in, the state machine allows it to be executed over the next four PGC clock cycles. While the received instruction is executed, the state machine simultaneously shifts in the next 4-bit command (see Figure 3-2).

Coming out of Reset, the first 4-bit control code is always forced to SIX and a forced `NOP` instruction is executed by the CPU. Five additional PGCx clocks are needed on start-up, 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. That is, the next 24 clock cycles load the first instruction word to the CPU.

Note:	To account for this forced `NOP`, all example code in this specification begins with a `NOP` to ensure that no data is lost.

FIGURE 3-2: SIX SERIAL EXECUTION



3.2.1.1 Differences Between Execution of SIX and Normal Instructions

There are some differences between executing instructions normally and using the SIX ICSP command. As a result, the code examples in this specification may not match those for performing the same functions during normal device operation.

The important differences are:

- Two-word instructions require two SIX operations to clock in all the necessary data.

  Examples of two-word instructions are `GOTO` and `CALL`.

- Two-cycle instructions require two SIX operations.

  The first SIX operation shifts in the instruction and begins to execute it. A second SIX operation – which should shift in a `NOP` to avoid losing data – provides the CPU clocks required to finish executing the instruction.

  Examples of two-cycle instructions are table read and table write instructions.

- The CPU does not automatically stall to account for pipeline changes.

  A CPU stall occurs when an instruction modifies a register that is used for Indirect Addressing by the following instruction.

  During normal operation, the CPU will automatically force a `NOP` while the new data is read. When using ICSP, there is no automatic stall, so any indirect references to a recently modified register should be preceded by a `NOP`.

  For example, the instructions, `MOV #0x0,W0` and `MOV [W0],W1`, must have a `NOP` inserted between them.

  If a two-cycle instruction modifies a register that is used indirectly, it will require two following `NOP`s: one to execute the second half of the instruction and a second to stall the CPU to correct the pipeline.

  Instructions, such as `TBLWTL [W0++],[W1]`, should be followed by two `NOP`s.

- The device Program Counter (PC) continues to automatically increment during ICSP instruction execution, even though the Flash memory is not being used.

  As a result, the PC may be incremented to point to invalid memory locations. Invalid memory spaces include unimplemented Flash addresses and the vector space (locations 0x0 to 0x1FF).

  If the PC points to these locations, the device will reset, possibly interrupting the ICSP operation. To prevent this, instructions should be periodically executed to reset the PC to a safe space. The optimal method to accomplish this is to perform a `GOTO 0x200`.

3.2.2 REGOUT SERIAL INSTRUCTION EXECUTION

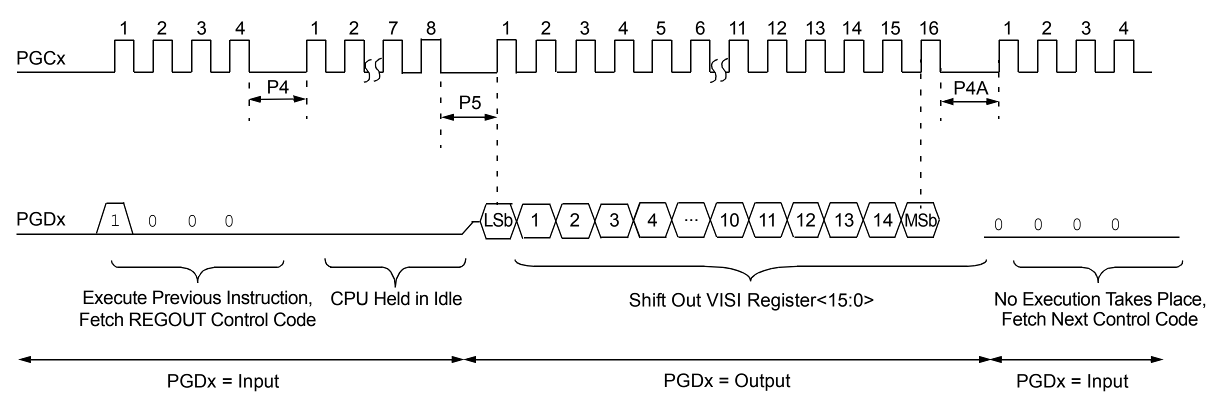
The REGOUT control code allows for data to be extracted from the device in ICSP mode. It is used to clock the contents of the VISI register, out of the device, over the PGDx pin. After the REGOUT control code is received, the CPU is held Idle for 8 cycles. After these 8 cycles, an additional 16 cycles are required to clock the data out (see Figure 3-3).

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 1: After the contents of VISI are shifted out, PIC24FJ64GA1/GB0 devices maintain PGDx as an output until the first rising edge of the next clock is received.

2: Data changes on the falling edge and latches on the rising edge of PGCx. For all data transmissions, the Least Significant bit (LSb) is transmitted first.

FIGURE 3-3: REGOUT SERIAL EXECUTION



3.3 Entering ICSP Mode

As shown in Figure 3-4, entering ICSP Program/Verify mode requires three steps:

1. $\overline{\text{MCLR}}$ 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.

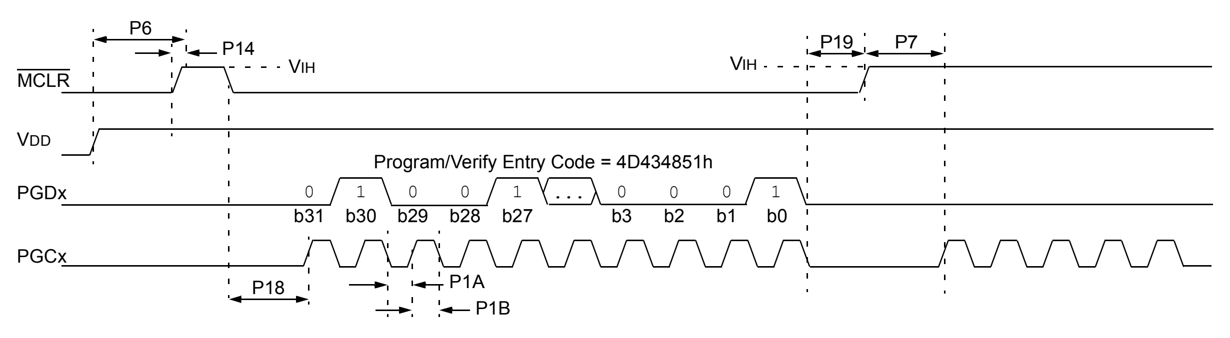
The programming voltage applied to $\overline{\text{MCLR}}$ is VIH, which is essentially VDD in the case of PIC24FJ64GA1/GB0 devices. There is no minimum time requirement for holding at VIH. After VIH is removed, an interval of at least P18 must elapse before presenting the key sequence on PGDx.

The key sequence is a specific 32-bit pattern: ‘`0100 1101 0100 0011 0100 1000 0101 0001`’ (more easily remembered as 4D434851h in hexadecimal). The device will enter Program/Verify mode only if the sequence is valid. The Most Significant bit (MSb) of the most significant nibble must be shifted in first.

Once the key sequence is complete, VIH 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 PGCx 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.

FIGURE 3-4: ENTERING ICSP™ MODE



3.4 Flash Memory Programming in ICSP Mode

3.4.1 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 3-2) or write operation (Table 3-3) 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 7.0 "AC/DC Characteristics and Timing Requirements"** for information about the delays associated with various programming operations.

TABLE 3-2: NVMCON ERASE OPERATIONS

NVMCON Value	Erase Operation
404Fh	Erase all code memory, executive memory and Configuration registers (does not erase Unit ID or Device ID registers).
4042h	Erase a page of code memory or executive memory.

TABLE 3-3: NVMCON WRITE OPERATIONS

NVMCON Value	Write Operation
4003h	Write a single instruction word.
4001h	Program 1 row (64 instruction words) of code memory or executive memory.

3.4.2 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
```

3.5 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 404Fh and executing the programming cycle.

A Chip Erase can erase all of user memory. A Table Write instruction should be executed prior to performing the Chip Erase to ensure the Chip Erase occurs correctly.

The Table Write instruction is executed:

- If the TBLPAG register points to user space (is less than 0x80), the Chip Erase will erase only user memory and Flash Configuration Words.
- If the TBLPAG register points to configuration space (is greater than or equal to 0x80), the Chip Erase is not allowed. The configuration space can be erased one page at a time.

Note: The Chip Erase is not allowed when the TBLPAG points to the configuration space to avoid the Diagnostic and Calibration Words from getting erased.

Figure 3-5 displays the ICSP programming process for performing a Chip Erase. This process includes the ICSP command code, which must be transmitted (for each instruction), LSb first, using the PGCx and PGDx pins (see Figure 3-2).

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

FIGURE 3-5: CHIP ERASE FLOW

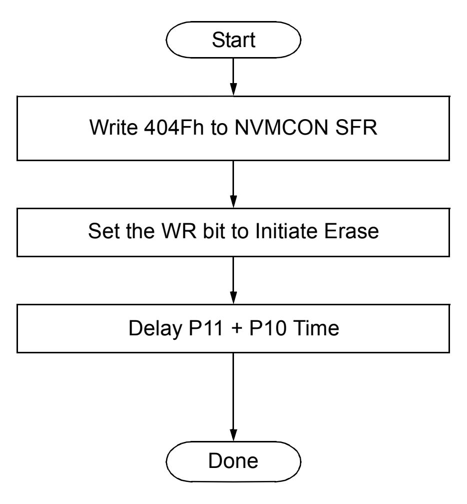


TABLE 3-4: SERIAL INSTRUCTION EXECUTION FOR CHIP ERASE

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	`NOP`
0000	040200	`GOTO    0x200`
0000	000000	`NOP`
Step 2: Set the NVMCON to erase all program memory.		
0000	2404FA	`MOV     #0x404F, W10`
0000	883B0A	`MOV     W10, NVMCON`
Step 3: Set TBLPAG and perform dummy table write to select what portions of memory are erased.		
0000	2xxxx0	`MOV     #<PAGEVAL>, W0`
0000	880190	`MOV     W0, TBLPAG`
0000	200000	`MOV     #0x0000, W0`
0000	BB0800	`TBLWTL  W0,[W0]`
0000	000000	`NOP`
0000	000000	`NOP`
Step 4: Initiate the erase cycle.		
0000	A8E761	`BSET    NVMCON, #WR`
0000	000000	`NOP`
0000	000000	`NOP`
Step 5: Repeat this step to poll the WR bit (bit 15 of NVMCON) until it is cleared by the hardware.		
0000	040200	`GOTO    0x200`
0000	000000	`NOP`
0000	803B02	`MOV     NVMCON, W2`
0000	883C22	`MOV     W2, VISI`
0000	000000	`NOP`
0001	<VISI>	`Clock out contents of the VISI register.`
0000	000000	`NOP`

3.6 Writing Code Memory

The procedure for writing code memory is the same as 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 3-5 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 3-2).

In Step 1, the Reset vector is exited. In Step 2, the NVMCON register is initialized for programming a full row 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.)

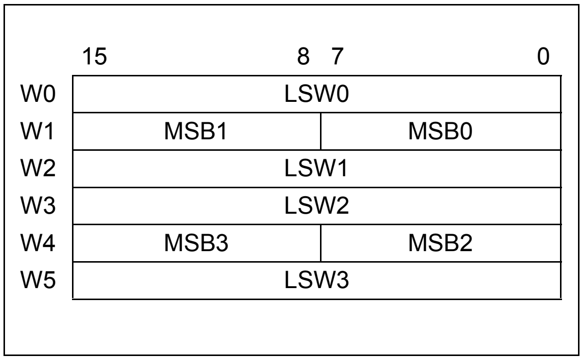
To minimize the programming time, a packed instruction format is used (Figure 3-6).

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 shown in Figure 3-6.

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 200h. 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 3-6: PACKED INSTRUCTION WORDS IN W0:W5



	15 … 8	7 … 0
W0	LSW0	
W1	MSB1	MSB0
W2	LSW1	
W3	LSW2	
W4	MSB3	MSB2
W5	LSW3	

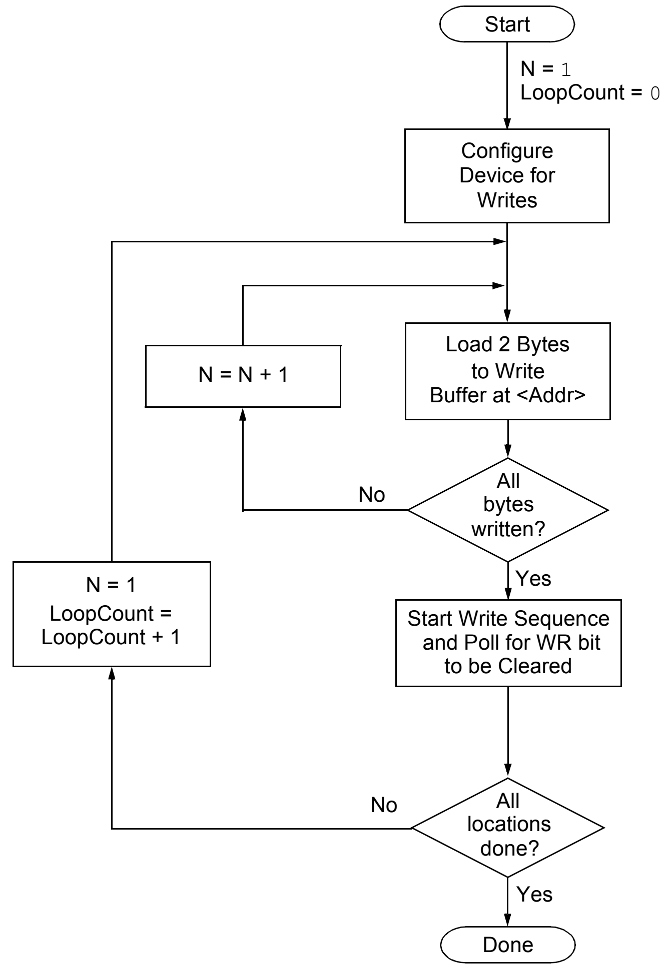
TABLE 3-5: SERIAL INSTRUCTION EXECUTION FOR WRITING CODE MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
Step 2: Set the NVMCON to program 64 instruction words.		
0000	24001A	MOV #0x4001, W10
0000	883B0A	MOV W10, NVMCON
Step 3: Initialize the Write Pointer (W7) for `TBLWT` instruction.		
0000	200xx0	MOV #<DestinationAddress23:16>, W0
0000	880190	MOV W0, TBLPAG
0000	2xxxx7	MOV #<DestinationAddress15:0>, W7
Step 4: Load W0:W5 with the next 4 instruction words to program.		
0000	2xxxx0	MOV #<LSW0>, W0
0000	2xxxx1	MOV #<MSB1:MSB0>, W1
0000	2xxxx2	MOV #<LSW1>, W2
0000	2xxxx3	MOV #<LSW2>, W3
0000	2xxxx4	MOV #<MSB3:MSB2>, W4
0000	2xxxx5	MOV #<LSW3>, W5

TABLE 3-5: SERIAL INSTRUCTION EXECUTION FOR WRITING CODE MEMORY (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 5: Set the Read Pointer (W6) and load the (next set of) write latches.		
0000	EB0300	CLR W6
0000	000000	NOP
0000	BB0BB6	TBLWTL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP
0000	BBDBB6	TBLWTH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BBEBB6	TBLWTH.B [W6++], [++W7]
0000	000000	NOP
0000	000000	NOP
0000	BB1BB6	TBLWTL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BB0BB6	TBLWTL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP
0000	BBDBB6	TBLWTH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BBEBB6	TBLWTH.B [W6++], [++W7]
0000	000000	NOP
0000	000000	NOP
0000	BB1BB6	TBLWTL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
Step 6: Repeat Steps 4 and 5, sixteen times, to load the write latches for 64 instructions.		
Step 7: Initiate the write cycle.		
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
Step 8: Repeat this step to poll the WR bit (bit 15 of NVMCON) until it is cleared by the hardware.		
0000	040200	GOTO 0x200
0000	000000	NOP
0000	803B02	MOV NVMCON, W2
0000	883C22	MOV W2, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the VISI register.
0000	000000	NOP
Step 9: Reset device internal PC.		
0000	040200	GOTO 0x200
0000	000000	NOP
Step 10: Repeat Steps 3-9 until all code memory is programmed.		

FIGURE 3-7: PROGRAM CODE MEMORY FLOW



3.7 Writing Configuration Words

Device configuration for PIC24FJ64GA1/GB0 devices is stored in Flash Configuration Words at the end of the user space program memory, and in multiple Configuration Word registers located in the test space. These registers reflect values read at any Reset from program memory locations. The values for the Configuration Words for the default device configurations are listed in Table 3-6.

The values can be changed only by programming the content of the corresponding Flash Configuration Word and resetting the device. The Reset forces an automatic reload of the Flash stored configuration values by sequencing through the dedicated Flash Configuration Words and transferring the data into the Configuration registers.

For the PIC24FJ64GA1/GB0 families, the bit at CW1<15> has a default state of ‘0’. This bit must always be maintained as ‘0’ to ensure device functionality, regardless of the settings of other Configuration bits.

To change the values of the Flash Configuration Word once it has been programmed, the device must be Chip Erased, as described in **Section 3.5 “Erasing Program Memory”**, and reprogrammed to the desired value. It is not possible to program a ‘0’ to ‘1’, but they may be programmed from a ‘1’ to ‘0’ to enable code protection.

TABLE 3-6: DEFAULT CONFIGURATION REGISTER VALUES

Address	Name	Default Value
Last Word	CW1	7FFFh
Last Word – 2	CW2	FFFFh
Last Word – 4	CW3	FFFFh
Last Word – 6	CW4	FFFFh

Table 3-7 shows the ICSP programming details for programming the Configuration Word locations, 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 3-2).

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 TBLPAG register must be loaded with 00h for all (32 and 64-Kbyte) devices.

To verify the data by reading the Configuration Words after performing the write in order, the code protection bits initially should be programmed to a ‘1’ to ensure that the verification can be performed properly. After verification is finished, the code protection bit can be programmed to a ‘0’ by using a word write to the appropriate Configuration Word.

TABLE 3-7: SERIAL INSTRUCTION EXECUTION FOR WRITING CONFIGURATION REGISTERS

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
`0000`	`000000`	`NOP`
`0000`	`040200`	`GOTO    0x200`
`0000`	`000000`	`NOP`
Step 2: Initialize the Write Pointer (W7) for the `TBLWT` instruction.		
`0000`	`2xxxx7`	`MOV     #<CW4Address15:0>, W7`
Step 3: Set the NVMCON register to program CW4.		
`0000`	`24003A`	`MOV     #0x4003, W10`
`0000`	`883B0A`	`MOV     W10, NVMCON`
Step 4: Initialize the TBLPAG register.		
`0000`	`200xx0`	`MOV     #<CW4Address23:16>, W0`
`0000`	`880190`	`MOV     W0, TBLPAG`
Step 5: Load the Configuration register data to W6.		
`0000`	`2xxxx6`	`MOV     #<CW4_VALUE>, W6`
Step 6: Write the Configuration register data to the write latch and increment the Write Pointer.		
`0000`	`000000`	`NOP`
`0000`	`BB1B86`	`TBLWTL  W6, [W7++]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
Step 7: Initiate the write cycle.		
`0000`	`A8E761`	`BSET    NVMCON, #WR`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
Step 8: Repeat this step to poll the WR bit (bit 15 of NVMCON) until it is cleared by the hardware.		
`0000`	`040200`	`GOTO    0x200`
`0000`	`000000`	`NOP`
`0000`	`803B02`	`MOV     NVMCON, W2`
`0000`	`883C22`	`MOV     W2, VISI`
`0000`	`000000`	`NOP`
`0001`	`<VISI>`	`Clock out contents of the VISI register.`
`0000`	`000000`	`NOP`
Step 9: Reset device internal PC.		
`0000`	`040200`	`GOTO    0x200`
`0000`	`000000`	`NOP`
Step 10: Repeat Steps 5-9 to write CW3 to CW1.		

3.8 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 3-8 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 3-6). 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 3-8: SERIAL INSTRUCTION EXECUTION FOR READING CODE MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit Reset vector.		
`0000`	`000000`	`NOP`
`0000`	`040200`	`GOTO 0x200`
`0000`	`000000`	`NOP`
Step 2: Initialize TBLPAG and the Read Pointer (W6) for `TBLRD` instruction.		
`0000`	`200xx0`	`MOV #<SourceAddress23:16>, W0`
`0000`	`880190`	`MOV W0, TBLPAG`
`0000`	`2xxxx6`	`MOV #<SourceAddress15:0>, W6`
Step 3: Initialize the Write Pointer (W7) to point to the VISI register.		
`0000`	`207847`	`MOV #VISI, W7`
`0000`	`000000`	`NOP`
Step 4: Read and clock out the contents of the next two locations of code memory, through the VISI register, using the REGOUT command.		
`0000`	`BA0B96`	`TBLRDL [W6], [W7]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0001`	`<VISI>`	`Clock out contents of VISI register`
`0000`	`000000`	`NOP`
`0000`	`BADBB6`	`TBLRDH.B [W6++], [W7++]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`BAD3D6`	`TBLRDH.B [++W6], [W7--]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0001`	`<VISI>`	`Clock out contents of VISI register`
`0000`	`000000`	`NOP`
`0000`	`BA0BB6`	`TBLRDL [W6++], [W7]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0001`	`<VISI>`	`Clock out contents of VISI register`
`0000`	`000000`	`NOP`
Step 5: Reset device internal PC.		
`0000`	`040200`	`GOTO 0x200`
`0000`	`000000`	`NOP`
Step 6: Repeat Steps 4 and 5 until all desired code memory is read.		

3.9 Reading Configuration Words

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. Configuration Words are read one register at a time.

Table 3-9 shows the ICSP programming details for reading the Configuration Words. Note that the TBLPAG register must be loaded with 00h, and the Read Pointer, W6, is initialized to the lower 16 bits of the Configuration Word location.

TABLE 3-9: SERIAL INSTRUCTION EXECUTION FOR READING ALL CONFIGURATION MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit Reset vector.		
`0000`	`000000`	`NOP`
`0000`	`040200`	`GOTO 0x200`
`0000`	`000000`	`NOP`
Step 2: Initialize TBLPAG, the Read Pointer (W6) and the Write Pointer (W7) for `TBLRD` instruction.		
`0000`	`200xx0`	`MOV #<CW4Address23:16>, W0`
`0000`	`880190`	`MOV W0, TBLPAG`
`0000`	`2xxxx6`	`MOV #<CW4Address15:0>, W6`
`0000`	`207847`	`MOV #VISI, W7`
`0000`	`000000`	`NOP`
Step 3: Read the Configuration register and write it to the VISI register (located at 784h), and clock out the VISI register using the REGOUT command.		
`0000`	`BA0BB6`	`TBLRDL [W6++], [W7]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0001`	`<VISI>`	`Clock out contents of VISI register`
`0000`	`000000`	`NOP`
Step 4: Repeat Step 3 three times to read CW3 to CW1.		
Step 5: Reset device internal PC.		
`0000`	`040200`	`GOTO 0x200`
`0000`	`000000`	`NOP`

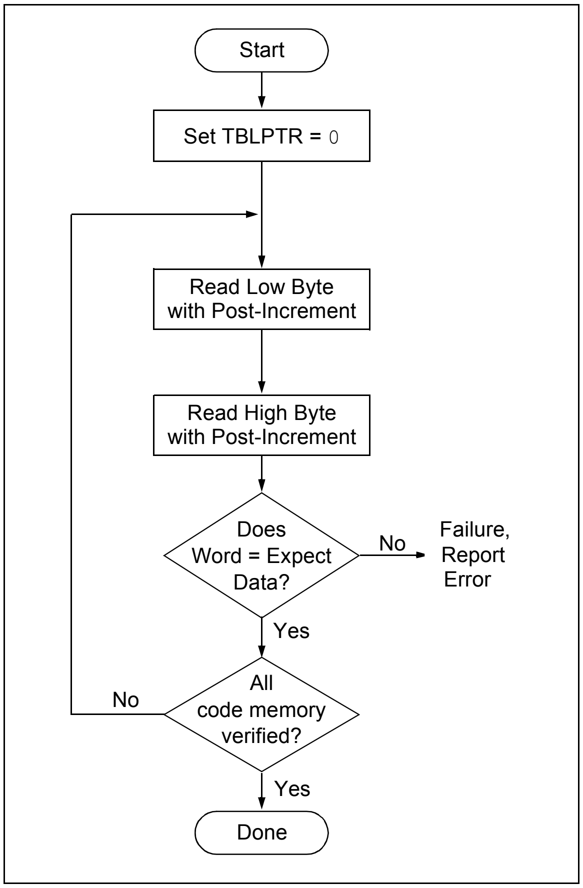
3.10 Verify Code Memory and Configuration Word

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 shown in the flowchart in Figure 3-8. 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 3.8 "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 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 CW1 has been cleared.

FIGURE 3-8: VERIFY CODE MEMORY FLOW



3.11 Reading the Application ID Word

The Application ID Word is stored at address, 8007F0h, 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 3-10.

After the programmer has clocked out the Application ID Word, it must be inspected. If the Application ID has the value, CBh, the programming executive is resident in memory and the device can be programmed using the mechanism described in **Section 4.0 "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 5.4 "Programming the Programming Executive to Memory"**.

3.12 Exiting ICSP Mode

Exiting Program/Verify mode is done by removing VIH from $\overline{\text{MCLR}}$, as shown in Figure 3-9. 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 VIH.

FIGURE 3-9: EXITING ICSP™ MODE

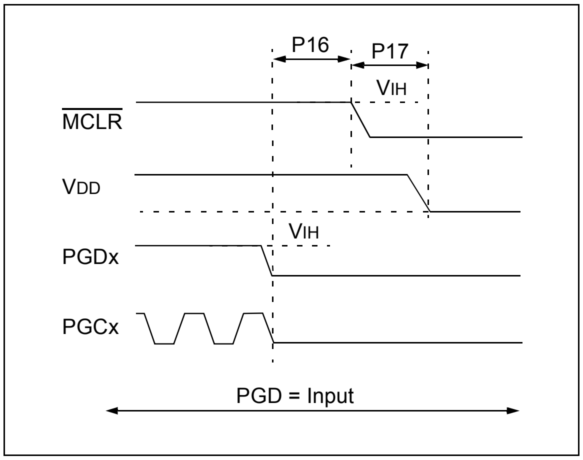


TABLE 3-10: SERIAL INSTRUCTION EXECUTION FOR READING THE APPLICATION ID WORD

Command (Binary)	Data (Hex)	Description
Step 1: Exit Reset vector.		
`0000`	`000000`	`NOP`
`0000`	`040200`	`GOTO    0x200`
`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`	`207F00`	`MOV     #0x7F0, 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`

4.0 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 PIC24FJ64GA1/GB0 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 4-1 summarizes the commands. A detailed description for each command is provided in **Section 5.2 “Programming Executive Commands”**.

TABLE 4-1: COMMAND SET SUMMARY

Command	Description
SCHECK	Sanity Check
READC	Read Device ID Registers
READP	Read Code Memory
PROGP	Program One Row of Code Memory and Verify
PROGW	Program One Word of Code Memory and Verify
QBLANK	Query if the Code Memory is Blank
QVER	Query the Software Version

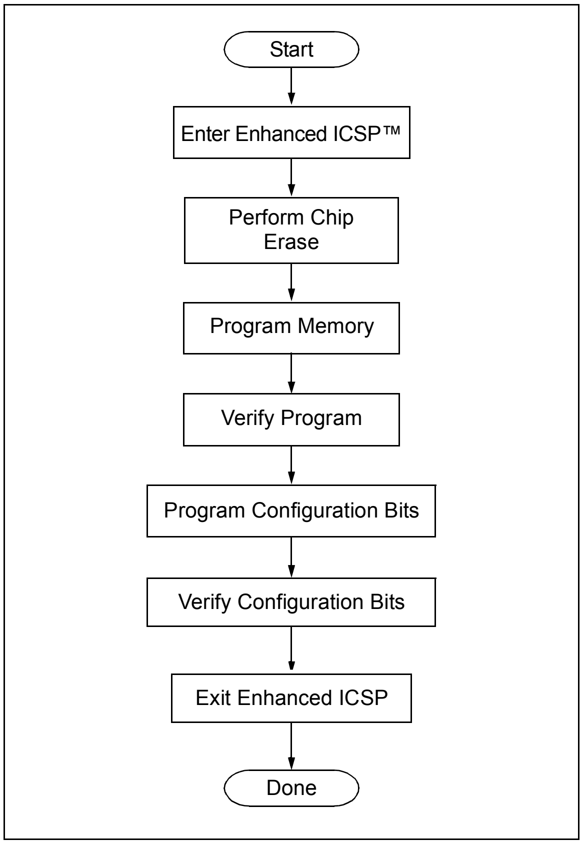
The programming executive uses the device’s data RAM for variable storage and program execution. After the programming executive has run, no assumptions should be made about the contents of data RAM.

4.1 Overview of the Programming Process

Figure 4-1 shows 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 configuration locations. 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 PIC24FJ64GA1/GB0 families can be programmed using the command set shown in Table 4-1.

FIGURE 4-1: HIGH-LEVEL ENHANCED ICSP™ PROGRAMMING FLOW



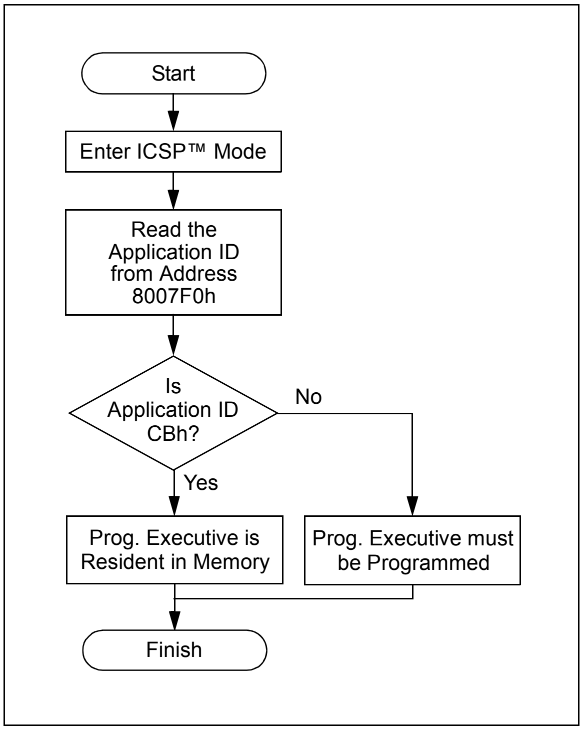
4.2 Confirming the Presence of the Programming Executive

Before programming can begin, the programmer must confirm that the programming executive is stored in executive memory. The procedure for this task is shown in Figure 4-2.

First, In-Circuit Serial Programming mode (ICSP) is entered. Then, the unique Application ID Word stored in executive memory is read. If the programming executive is resident, the Application ID Word is CBh, which means programming can resume as normal. However, if the Application ID Word is not CBh, the programming executive must be programmed to executive code memory using the method described in **Section 5.4 “Programming the Programming Executive to Memory”**.

Section 3.0 “Device Programming – ICSP” describes the ICSP programming method. **Section 3.11 “Reading the Application ID Word”** describes the procedure for reading the Application ID Word in ICSP mode.

FIGURE 4-2: CONFIRMING PRESENCE OF PROGRAMMING EXECUTIVE



4.3 Entering Enhanced ICSP Mode

As shown in Figure 4-3, 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.

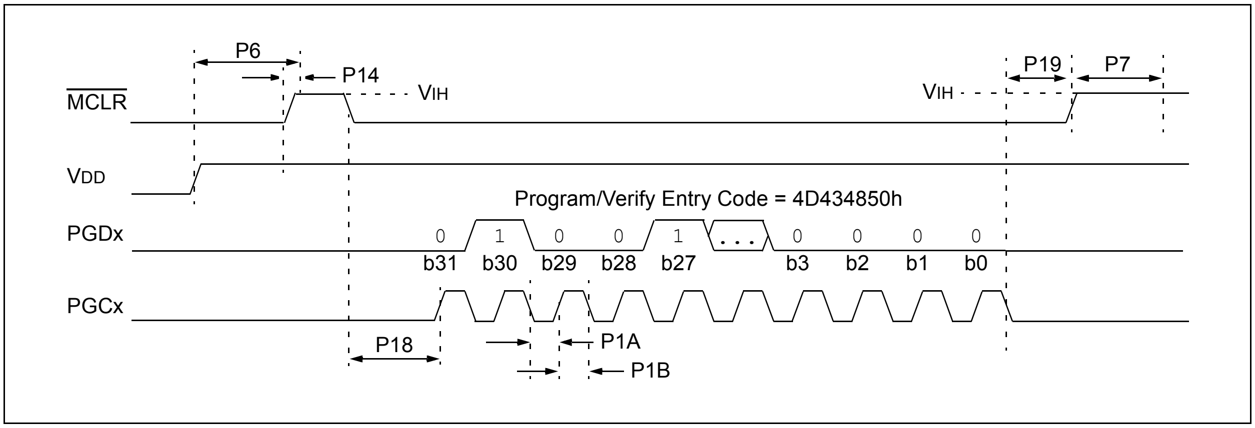
The programming voltage applied to $\overline{\text{MCLR}}$ is V_{IH}, which is essentially V_{DD} in the case of PIC24FJ64GA1/GB0 devices. There is no minimum time requirement for holding at V_{IH}. After V_{IH} is removed, an interval of at least P18 must elapse before presenting the key sequence on PGDx.

The key sequence is a specific 32-bit pattern: '`0100 1101 0100 0011 0100 1000 0101 0000`' (more easily remembered as 4D434850h 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 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 the Program/Verify mode, all unused I/Os are placed in the high-impedance state.

FIGURE 4-3: ENTERING ENHANCED ICSP™ MODE



4.4 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 (FF0002h:FF0000h) 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 by 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.

4.5 Code Memory Programming

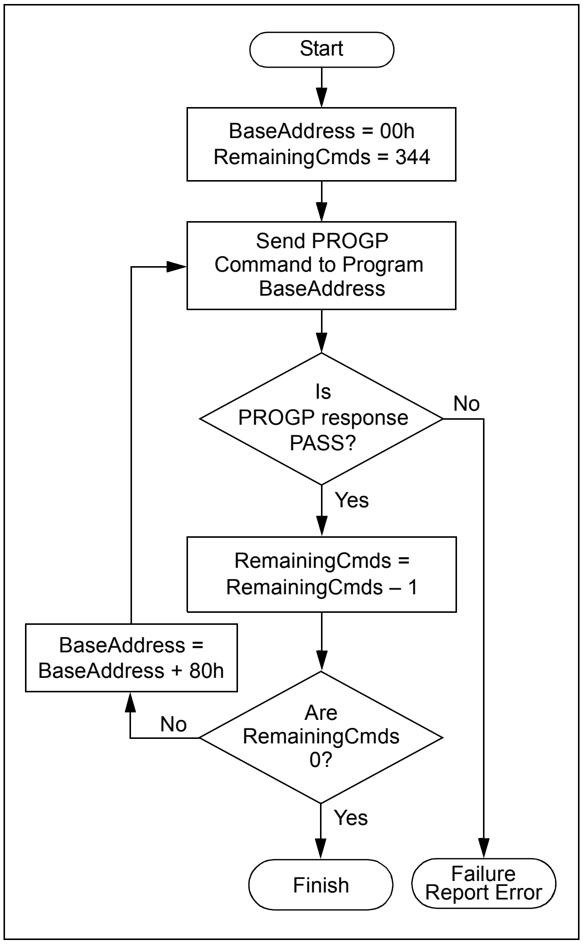
4.5.1 PROGRAMMING METHODOLOGY

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 the code memory of the PIC24FJ64GA1/GB0 families is shown in Figure 4-4. In this example, all 22K instruction words of a 64-Kbyte device are programmed. First, the number of commands to send (called ‘RemainingCmds’ in the flowchart) is set to 344 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 device. 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 80h 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. No special handling must be performed when a panel boundary is crossed.

FIGURE 4-4: FLOWCHART FOR PROGRAMMING CODE MEMORY



4.5.2 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 of the programmed code memory.

Alternatively, you can have the programmer perform the verification after the entire device is programmed using a checksum computation.

4.6 Configuration Bits Programming

4.6.1 OVERVIEW

The PIC24FJ64GA1/GB0 families have Configuration bits stored in the last three locations of implemented program memory (see Table 2-2 for locations). 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 descriptions for the Configuration bits in the Flash Configuration Words are shown in Table 4-2.

Note:	Although not implemented with a specific function, the bit at CW1<15> must always be maintained as '`0`' to ensure device functionality, regardless of the settings of other Configuration bits.

TABLE 4-2: PIC24FJ64GA1/GB0 CONFIGURATION BITS DESCRIPTION

Bit Field	Register	Description
$\overline{\text{DEBUG}}$	CW1<11>	Background Debug Enable bit `1` = Device will reset in User mode `0` = Device will reset in Debug mode
DSWDTEN	CW4<7>	Deep Sleep Watchdog Timer Enable bit `1` = DSWDT enabled `0` = DSWDT disabled
DSWDTOSC	CW4<4>	DSWDT Reference Clock Select bit `1` = DSWDT uses LPRC as reference clock `0` = DSWDT uses SOSC as reference clock
DSWDTPS<3:0>	CW4<3:0>	DSWDT Postscaler Select bits (assumes a DSWDT 1:32 prescaler) `1111` = 1:2,147,483,648 (25.7 days) `1110` = 1:536,870,912 (6.4 days) `1101` = 1:134,217,728 (38.5 hours) `1100` = 1:33,554,432 (9.6 hours) `1011` = 1:8,388,608 (2.4 hours) `1010` = 1:2,097,152 (36 minutes) `1001` = 1:524,288 (9 minutes) `1000` = 1:131,072 (135 seconds) `0111` = 1:32,768 (34 seconds) `0110` = 1:8,192 (8.5 seconds) `0101` = 1:2,048 (2.1 seconds) `0100` = 1:512 (528 ms) `0011` = 1:128 (132 ms) `0010` = 1:32 (33 ms) `0001` = 1:8 (8.3 ms) `0000` = 1:2 (2.1 ms)
DSBOR	CW4<6>	Deep Sleep BOR Enable bit `1` = BOR enabled in Deep Sleep `0` = BOR disabled in Deep Sleep (does not affect Sleep mode)
FCKSM<1:0>	CW2<7:6>	Clock Switching Mode bits `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

Note 1: Available on PIC24FJXXXGB0XX devices only.

TABLE 4-2: PIC24FJ64GA1/GB0 CONFIGURATION BITS DESCRIPTION (CONTINUED)

Bit Field	Register	Description
FNOSC<2:0>	CW2<10:8>	Initial Oscillator Source Selection bits 111 = Internal Fast RC (FRCDIV) oscillator with postscaler 110 = Reserved 101 = Low-Power RC (LPRC) oscillator 100 = Secondary (SOSC) oscillator 011 = Primary (XTPLL, HSPLL, ECPLL) oscillator with PLL 010 = Primary (XT, HS, EC) oscillator 001 = Internal Fast RC (FRCPLL) oscillator with postscaler and PLL 000 = Fast RC (FRC) oscillator
FWDTEN	CW1<7>	Watchdog Timer Enable bit 1 = Watchdog Timer always enabled (LPRC oscillator cannot be disabled; clearing the SWDTEN bit in the RCON register will have no effect) 0 = Watchdog Timer enabled/disabled by user software (LPRC can be disabled by clearing the SWDTEN bit in the RCON register)
FWPSA	CW1<4>	Watchdog Timer Postscaler bit 1 = 1:128 0 = 1:32
GCP	CW1<13>	General Segment Code-Protect bit 1 = User program memory is not code-protected 0 = User program memory is code-protected
GWRP	CW1<12>	General Segment Write-Protect bit 1 = User program memory is not write-protected 0 = User program memory is write-protected
ICS<1:0>	CW1<9,8>	ICD Emulator Pin Placement Select bits 11 = Emulator functions are shared with PGEC1/PGED1 10 = Emulator functions are shared with PGEC2/PGED2 01 = Emulator functions are shared with PGEC3/PGED3 00 = Reserved; do not use
IESO	CW2<15>	Internal External Switchover bit 1 = Two-Speed Start-up enabled 0 = Two-Speed Start-up disabled
IOL1WAY	CW2<4>	IOLOCK Bit One-Way Set Enable bit 0 = The IOLOCK bit can be set and cleared as needed (provided an unlocking sequence is executed) 1 = The IOLOCK bit can only be set once (provided an unlocking sequence is executed). Once IOLOCK is set, this prevents any possible future RP register changes.
JTAGEN	CW1<14>	JTAG Enable bit 1 = JTAG enabled 0 = JTAG disabled
OSCIOFNC	CW2<5>	OSC2 Pin Function bit (except in XT and HS modes) 1 = OSC2 is clock output 0 = OSC2 is general purpose digital I/O pin

Note 1: Available on PIC24FJXXXGB0XX devices only.

TABLE 4-2: PIC24FJ64GA1/GB0 CONFIGURATION BITS DESCRIPTION (CONTINUED)

Bit Field	Register	Description
PLLDIV<2:0>[(1)]	CW2<14:12>	USB 96 MHz PLL Prescaler Select bits `111` = Oscillator input divided by 12 (48 MHz input) `110` = Oscillator input divided by 8 (32 MHz input) `101` = Oscillator input divided by 6 (24 MHz input) `100` = Oscillator input divided by 5 (20 MHz input) `011` = Oscillator input divided by 4 (16 MHz input) `010` = Oscillator input divided by 3 (12 MHz input) `001` = Oscillator input divided by 2 (8 MHz input) `000` = Oscillator input used directly (4 MHz input)
PLL96MHZ[(1)]	CW2<11>	USB 96 MHz PLL Start-up Enable bit `1` = 96 MHz PLL is enabled by user in software (controlled with PLLEN bit in CLKDIV<5>) `0` = 96 MHz PLL is enabled automatically on start-up
POSCMD<1:0>	CW2<1:0>	Primary Oscillator Mode Select bits `11` = Primary oscillator disabled `10` = HS Crystal Oscillator mode `01` = XT Crystal Oscillator mode `00` = EC (External Clock) mode
RTCOSC	CW4<5>	RTCC Reference Clock Select bit `1` = RTCC uses SOSC as reference clock `0` = RTCC uses LPRC as reference clock
SOSCSEL<1:0>	CW3<9:8>	Secondary Oscillator Mode Select bits `11` = Default (high drive strength) SOSC mode `01` = Low-Power (low drive strength) SOSC mode `x0` = External Clock Input (SCLKI) mode
WDTPOST<3:0>	CW1<3:0>	Watchdog Timer Prescaler bits `1111` = 1:32,768 `1110` = 1:16,384 . . . `0001` = 1:2 `0000` = 1:1
WINDIS	CW1<6>	Windowed WDT bit `1` = Watchdog Timer in Non-Window mode `0` = Watchdog Timer in Window mode; FWDTEN must be '`1`'
WPCFG	CW3<14>	Configuration Word Code Page Protection Select bit `1` = Last page (at the top of program memory) and Flash Configuration Words are not protected `0` = Last page and Flash Configuration Words are code-protected
WPDIS	CW3<13>	Segment Write Protection Disable bit `1` = Segmented code protection disabled `0` = Segmented code protection enabled; protected segment defined by WPEND, WPCFG and WPFPx Configuration bits
WPEND	CW3<15>	Segment Write Protection End Page Select bit `1` = Protected code segment lower boundary is at the bottom of program memory (000000h); upper boundary is the code page specified by WPFP<5:0> `0` = Protected code segment upper boundary is at the last page of program memory; lower boundary is the code page specified by WPFP<5:0>

Note 1: Available on PIC24FJXXXGB0XX devices only.

TABLE 4-2: PIC24FJ64GA1/GB0 CONFIGURATION BITS DESCRIPTION (CONTINUED)

Bit Field	Register	Description
WPFP<5:0>	CW3<5:0>	Protected Code Segment Boundary Page bits Designates the 512 instruction page that is the boundary of the protected code segment, starting with Page 0 at the bottom of program memory. If WPEND = `1`: Last address of designated code page is the upper boundary of the segment. If WPEND = `0`: First address of designated code page is the lower boundary of the segment.
WUTSEL<1:0>	CW2<14:13>	Voltage Regulator Standby Mode Wake-up Time Select bits `11` = Default regulator wake time used `01` = Fast regulator wake time used `x0` = Reserved; do not use

Note 1: Available on PIC24FJXXXGB0XX devices only.

4.6.2 PROGRAMMING METHODOLOGY

Configuration bits may be programmed, a single byte at a time, using the PROGW command. This command specifies the configuration data and Configuration register address. When Configuration bits are programmed, any unimplemented or reserved bits must be programmed with a '`1`'.

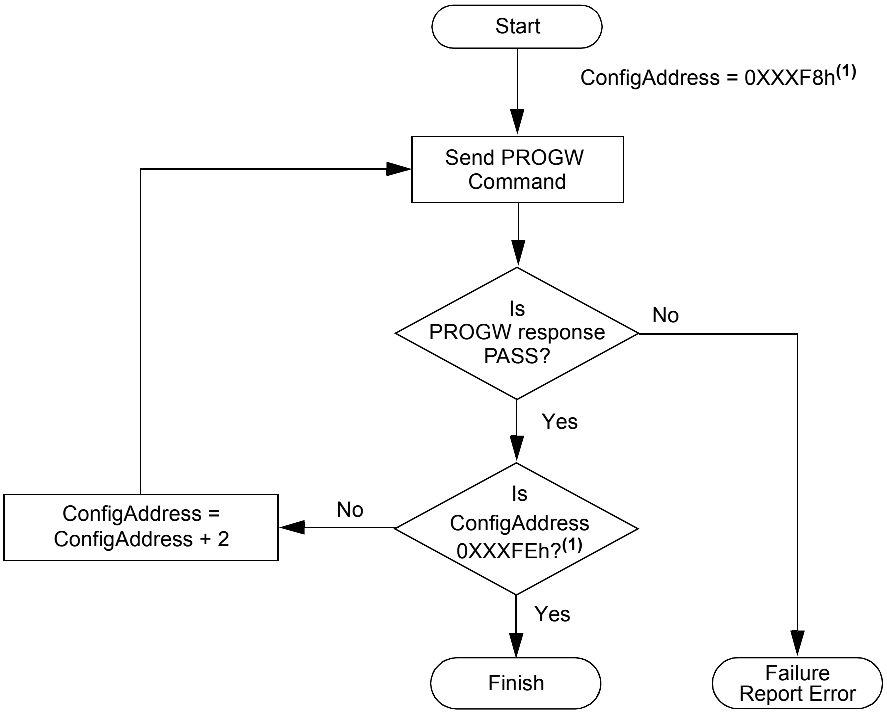
Four PROGW commands are required to program the Configuration bits. A flowchart for Configuration bit programming is shown in Figure 4-5.

Note:	If the General Segment Code-Protect bit (GCP) is programmed to '`0`', code memory is code-protected and can not be read. Code memory must be verified before enabling read protection. See **Section 4.6.4 "Code-Protect Configuration Bits"** for more information about code-protect Configuration bits.

4.6.3 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 READP command reads back the programmed Configuration bits and verifies that the programming was successful.

FIGURE 4-5: CONFIGURATION BIT PROGRAMMING FLOW



Note 1: Refer to Table 2-2 for Flash Configuration Word addresses.

4.6.4 CODE-PROTECT CONFIGURATION BITS

PIC24FJ64GA1/GB0 family devices provide two complimentary methods to protect application code from overwrites and erasures. These also help to protect the device from inadvertent configuration changes during run time. Additional information is available in the product data sheet.

4.6.4.1 GENERAL SEGMENT PROTECTION

For all devices in the PIC24FJ64GA1/GB0 families, the on-chip program memory space is treated as a single block, known as the General Segment (GS). Code protection for this block is controlled by one Configuration bit, GCP. This bit inhibits external reads and writes to the program memory space. It has no direct effect in normal execution mode.

Write protection is controlled by the GWRP bit in the Configuration Word. When GWRP is programmed to '`0`', internal write and erase operations to program memory are blocked.

4.6.4.2 CODE SEGMENT PROTECTION

In addition to global General Segment protection, a separate subrange of the program memory space can be individually protected against writes and erases. This area can be used for many purposes where a separate block of write and erase-protected code is needed, such as bootloader applications. Unlike common boot block implementations, the specially protected segment in PIC24FJ64GA1/GB0 devices can be located by the user anywhere in the program space, and configured in a wide range of sizes.

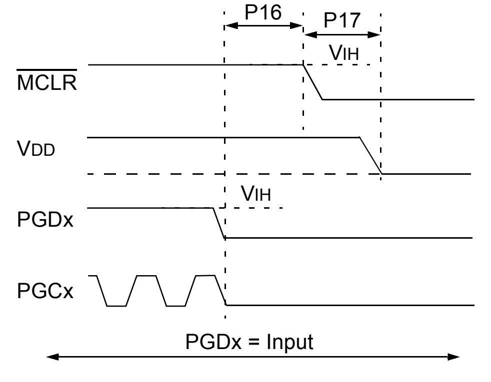
Code segment protection provides an added level of protection to a designated area of program memory by disabling the NVM safety interlock whenever a write or erase address falls within a specified range. It does not override General Segment protection controlled by the GCP or GWRP bits. For example, if GCP and GWRP are enabled, enabling segmented code protection for the bottom half of program memory does not undo General Segment protection for the top half.

Note:	Bulk Erasing in ICSP mode is the only way to reprogram code-protect bits from an ON state ('`0`') to an OFF state ('`1`').

4.7 Exiting Enhanced ICSP Mode

Exiting Program/Verify mode is done by removing V_{IH} from $\overline{MCLR}$, as shown in Figure 4-6. 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 4-6: EXITING ENHANCED ICSP™ MODE



5.0 THE PROGRAMMING EXECUTIVE

5.1 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 5.2 "Programming Executive Commands"**. The response set is described in **Section 5.3 "Programming Executive Responses"**.

5.1.1 COMMUNICATION INTERFACE AND PROTOCOL

The 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.

Data transmits to the device must change on the rising edge and hold on the falling edge. Data receives from the device must change on the falling edge and hold on the rising edge.

All data transmissions are sent to the Most Significant bit (MSb) first, using 16-bit mode (see Figure 5-1).

FIGURE 5-1: PROGRAMMING EXECUTIVE SERIAL TIMING FOR DATA RECEIVED FROM DEVICE

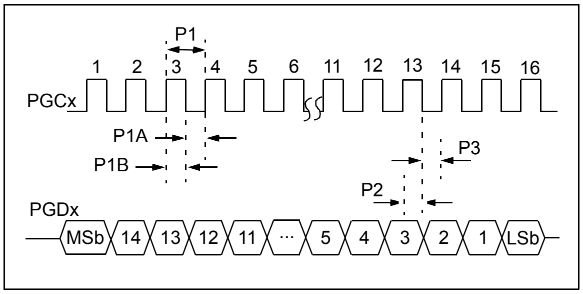
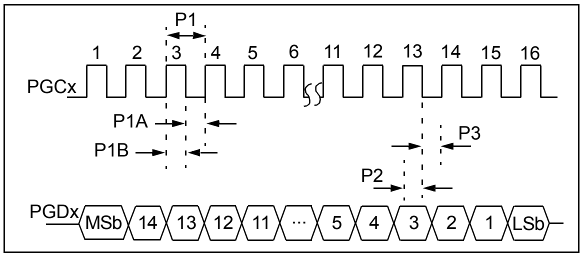


FIGURE 5-2: PROGRAMMING EXECUTIVE SERIAL TIMING FOR DATA TRANSMITTED TO DEVICE



Since a 2-wire SPI is used, and data transmissions are half duplex, 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 for 15 µs to indicate to the programmer that the response is available to be clocked out. The programmer can begin to clock out the response 23 µs after PGDx is brought low, and it must provide the necessary amount of clock pulses to receive the entire response from the programming executive.

After the entire response is clocked out, the programmer should terminate the clock on PGCx until it is time to send another command to the programming executive. This protocol is shown in Figure 5-3.

5.1.2 SPI RATE

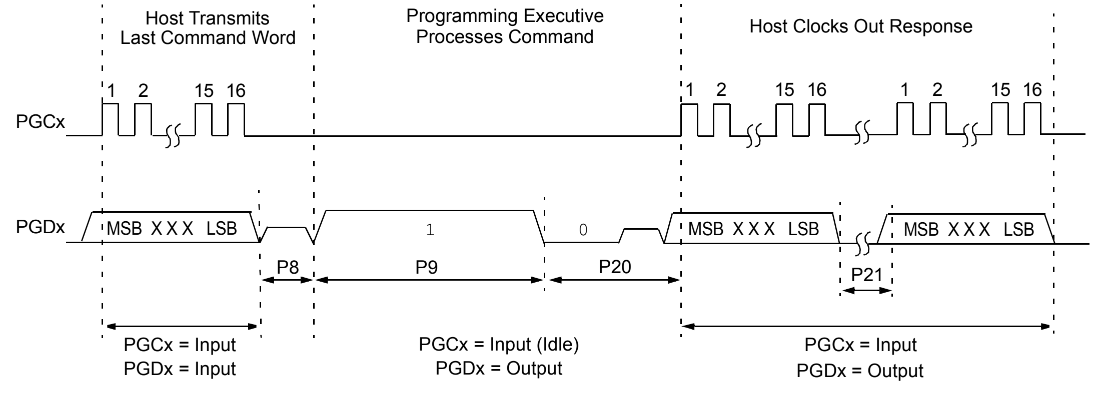
In Enhanced ICSP mode, the PIC24FJ64GA1/GB0 devices operate from the Internal Fast RC oscillator (FRCDIV), which has a nominal frequency of 8 MHz. This oscillator frequency yields an effective system clock frequency of 4 MHz. To ensure that the programmer does not clock too fast, it is recommended that a 4 MHz clock be provided by the programmer.

5.1.3 TIME-OUTS

The programming executive uses no Watchdog Timer 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 5.1.1 "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 5-1. If the command time-out expires, the programmer should reset the programming executive and start programming the device again.

FIGURE 5-3: PROGRAMMING EXECUTIVE – PROGRAMMER COMMUNICATION PROTOCOL



5.2 Programming Executive Commands

The programming executive command set is shown in Table 5-1. This table contains the opcode, mnemonic, length, time-out and description for each command. Functional details on each command are provided in **Section 5.2.4 “Command Descriptions”**.

5.2.1 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 5-4). 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 5-4: COMMAND FORMAT

15 12	11 0
Opcode	Length
Command Data First Word (if required)	
•	
•	
Command Data Last Word (if required)	

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 5-1, will return a “NACK” response (see **Section 5.3.1.1 “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.

5.2.2 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 shown in Figure 5-5. This format minimizes traffic over the SPI and provides the programming executive with data that is properly aligned for performing table write operations.

FIGURE 5-5: PACKED INSTRUCTION WORD FORMAT

15 8	7 0
LSW1	
MSB2	MSB1
LSW2	

LSWx: Least Significant 16 bits of instruction word
MSBx: Most Significant Bytes of instruction word

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

5.2.3 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 5.3.1.3 “QE_Code Field”**.

TABLE 5-1: PROGRAMMING EXECUTIVE COMMAND SET

Opcode	Mnemonic	Length (16-bit words)	Time-out	Description
0h	SCHECK	1	1 ms	Sanity check.
1h	READC	3	1 ms	Read an 8-bit word from the specified Device ID register.
2h	READP	4	1 ms/row	Read N 24-bit instruction words of code memory starting from the specified address.
3h	RESERVED	N/A	N/A	This command is reserved. It will return a NACK.
4h	PROGC	4	5 ms	Write an 8-bit word to the specified Device ID registers.
5h	PROGP	99	5 ms	Program one row of code memory at the specified address, then verify.(1)
6h	RESERVED	N/A	N/A	This command is reserved. It will return a NACK.
7h	RESERVED	N/A	N/A	This command is reserved. It will return a NACK.
8h	RESERVED	N/A	N/A	This command is reserved. It will return a NACK.
9h	RESERVED	N/A	N/A	This command is reserved. It will return a NACK.
Ah	QBLANK	3	TBD	Query if the code memory is blank.
Bh	QVER	1	1 ms	Query the programming executive software version.
Ch	RESERVED	N/A	N/A	This command is reserved. It will return a NACK.
Dh	PROGW	4	5 ms	Program one instruction word of code memory at the specified address, then verify.

Legend: TBD = To Be Determined

Note 1: One row of code memory consists of (64) 24-bit words. Refer to Table 2-2 for device-specific information.

5.2.4 COMMAND DESCRIPTIONS

All commands supported by the programming executive are described in **Section 5.2.5 "SCHECK Command"** through **Section 5.2.12 "QVER Command"**.

5.2.5 SCHECK COMMAND

15 12	11 0
Opcode	Length

Field	Description
Opcode	0h
Length	1h

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):

1000h
0002h

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

5.2.6 READC COMMAND

15 12	11 8	7 0
Opcode	Length	
N		Addr_MSB
Addr_LS		

Field	Description
Opcode	1h
Length	3h
N	Number of 8-bit Device ID registers to read (max. 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 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 Device ID registers, the upper byte in every data word returned by the programming executive is 00h and the lower byte contains the Device ID register value.

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

1100h

2 + N

Device ID Register 1

...

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.

5.2.7 READP COMMAND

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

Field	Description
Opcode	2h
Length	4h
N	Number of 24-bit instructions to read (max. of 32768)
Reserved	0h
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, including Configuration Words, starting from the 24-bit address specified by Addr_MSB and Addr_LS. This command can only be used to read 24-bit data. All data returned in response to this command uses the packed data format described in **Section 5.2.2 "Packed Data Format"**.

Expected Response (2 + 3 * N/2 words for N even):

1200h

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):

1200h

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.

5.2.8 PROGC COMMAND

<table>
<tr><th>15 12</th><th>11 8</th><th>7 0</th></tr>
<tr><td>Opcode</td><td colspan="2">Length</td></tr>
<tr><td colspan="2">Reserved</td><td>Addr_MSB</td></tr>
<tr><td colspan="3">Addr_LS</td></tr>
<tr><td colspan="3">Data</td></tr>
</table>

Field	Description
Opcode	4h
Length	4h
Reserved	0h
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 Device ID 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):

1400h
0002h

5.2.9 PROGP COMMAND

<table>
<tr><th>15 12</th><th>11 8</th><th>7 0</th></tr>
<tr><td>Opcode</td><td colspan="2">Length</td></tr>
<tr><td colspan="2">Reserved</td><td>Addr_MSB</td></tr>
<tr><td colspan="3">Addr_LS</td></tr>
<tr><td colspan="3">D_1</td></tr>
<tr><td colspan="3">D_2</td></tr>
<tr><td colspan="3">...</td></tr>
<tr><td colspan="3">D_96</td></tr>
</table>

Field	Description
Opcode	5h
Length	63h
Reserved	0h
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, including Configuration Words (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 80h.

The data to program to memory, located in command words, D_1 through D_96, must be arranged using the packed instruction word format shown in Figure 5-5.

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):

1500h
0002h

Note:	Refer to Table 2-2 for code memory size information.

5.2.10 PROGW COMMAND

15 12	11 8	7 0
Opcode	Length	
Data_MSB		Addr_MSB
Addr_LS		
Data_LS		

Field	Description
Opcode	Dh
Length	4h
Reserved	0h
Addr_MSB	MSB of 24-bit destination address
Addr_LS	Least Significant 16 bits of 24-bit destination address
Data_MSB	MSB of 24-bit data
Data_LS	Least Significant 16 bits of 24-bit data

The PROGW command instructs the programming executive to program one word of code memory (3 bytes) to the specific memory address.

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

Expected Response (2 words):

1600h
0002h

5.2.11 QBLANK COMMAND

15 12	11 0
Opcode	Length
PSize_MSW	
PSize_LSW	

Field	Description
Opcode	Ah
Length	3h
PSize	Length of program memory to check in 24-bit words plus one (max. of 49152)

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

The Blank Check for code memory begins at 0h and advances toward larger addresses for the specified number of instruction words.

QBLANK returns a QE_Code of F0h if the specified code memory and code-protect bits are blank; otherwise, QBLANK returns a QE_Code of 0Fh.

Expected Response (2 words for blank device):

1AF0h

0002h

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

1A0Fh

0002h

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

5.2.12 QVER COMMAND

15 12	11 0
Opcode	Length

Field	Description
Opcode	Bh
Length	1h

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., 23h means version 2.3 of programming executive software).

Expected Response (2 words):

1BMNh (where “MN” stands for version M.N)
0002h

5.3 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 5-2. This table contains the opcode, mnemonic and description for each response. The response format is described in **Section 5.3.1 “Response Format”**.

TABLE 5-2: PROGRAMMING EXECUTIVE RESPONSE OP CODES

Opcode	Mnemonic	Description
1h	PASS	Command successfully processed
2h	FAIL	Command unsuccessfully processed
3h	NACK	Command not known

5.3.1 RESPONSE FORMAT

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

15 12	11 8	7 0
Opcode	Last_Cmd	QE_Code
Length		
D_1 (if applicable)		
...		
D_N (if applicable)		

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)

5.3.1.1 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 5-2). 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.

5.3.1.2 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.

5.3.1.3 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 5-3.

TABLE 5-3: QE_Code FOR QUERIES

Query	QE_Code
QBLANK	0Fh = Code memory is NOT blank F0h = Code memory is blank
QVER	0xMN, where programming executive software version = M.N (i.e., 32h 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 5-4. If a command is successfully processed, the returned QE_Code is set to 0h, which indicates that there was 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 1h. For all other programming executive errors, the QE_Code is 2h.

TABLE 5-4: QE_Code FOR NON-QUERY COMMANDS

QE_Code	Description
0h	No error
1h	Verify failed
2h	Other error

5.3.1.4 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 5.2.2 "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.

5.4 Programming the Programming Executive to Memory

5.4.1 OVERVIEW

If it is determined that the programming executive is not present in executive memory (as described in **Section 4.2 "Confirming the Presence of the Programming Executive"**), it must be programmed into executive memory using ICSP, as described in **Section 3.0 "Device Programming – ICSP"**.

Storing the programming executive to executive memory is similar to normal programming of code memory. Namely, the executive memory must be erased, and then the programming executive must be programmed, 64 words at a time. This control flow is summarized in Table 5-5.

TABLE 5-5: PROGRAMMING THE PROGRAMMING EXECUTIVE

Command (Binary)	Data (Hex)	Description
Step 1: Exit Reset vector and erase executive memory.		
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
Step 2: Initialize the NVMCON to erase executive memory.		
0000	240420	MOV #0x4042, W0
0000	883B00	MOV W0, NVMCON
Step 3: Initialize Erase Pointers to first page of executive and then initiate the erase cycle.		
0000	200800	MOV #0x80, W0
0000	880190	MOV W0, TBLPAG
0000	200001	MOV #0x0, 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
Step 4: Repeat this step to poll the WR bit (bit 15 of NVMCON) until it is cleared by the hardware.		
0000	040200	GOTO 0x200
0000	000000	NOP
0000	803B02	MOV NVMCON, W2
0000	883C22	MOV W2, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the VISI register.
0000	000000	NOP
Step 5: Repeat Steps 3 and 4 to erase the second page of executive memory. The W1 Pointer should be incremented by 400h to point to the second page.		
Step 6: Initialize the NVMCON to program 64 instruction words.		
0000	240010	MOV #0x4001, W0
0000	883B00	MOV W0, NVMCON
Step 7: Initialize TBLPAG and the Write Pointer (W7).		
0000	200800	MOV #0x80, W0
0000	880190	MOV W0, TBLPAG
0000	EB0380	CLR W7
0000	000000	NOP

TABLE 5-5: PROGRAMMING THE PROGRAMMING EXECUTIVE (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 8: Load W0:W5 with the next four words of packed programming executive code and initialize W6 for programming. Programming starts from the base of executive memory (800000h) 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
Step 9: 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	BBEBB6	TBLWTH.B [W6++], [++W7]
0000	000000	NOP
0000	000000	NOP
0000	BB1BB6	TBLWTL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BB0BB6	TBLWTL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP
0000	BBDBB6	TBLWTH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BBEBB6	TBLWTH.B [W6++], [++W7]
0000	000000	NOP
0000	000000	NOP
0000	BB1BB6	TBLWTL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
Step 10: Repeat Steps 8 and 9, sixteen times, to load the write latches for the 64 instructions.		
Step 11: Initiate the programming cycle.		
0000	A8E761	BSET NVMCON, #15
0000	000000	NOP
0000	000000	NOP
Step 12: Repeat this step to poll the WR bit (bit 15 of NVMCON) until it is cleared by the hardware.		
0000	040200	GOTO 0x200
0000	000000	NOP
0000	803B02	MOV NVMCON, W2
0000	883C22	MOV W2, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of the VISI register.
0000	000000	NOP
Step 13: Reset the device internal PC.		
0000	040200	GOTO 0x200
0000	000000	NOP
Step 14: Repeat Steps 8 through 13 until all 16 rows of executive memory have been programmed.		

5.4.2 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 3.8 “Reading Code Memory”**. A procedure for reading executive memory is shown in Table 5-6. Note that in Step 2, the TBLPAG register is set to 80h, such that executive memory may be read. The last eight words of executive memory should be verified with stored values of the Diagnostic and Calibration Words to ensure accuracy.

TABLE 5-6: READING EXECUTIVE MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
`0000`	`000000`	`NOP`
`0000`	`040200`	`GOTO 0x200`
`0000`	`000000`	`NOP`
Step 2: Initialize TBLPAG and the Read Pointer (W6) for `TBLRD` instruction.		
`0000`	`200800`	`MOV #0x80, W0`
`0000`	`880190`	`MOV W0, TBLPAG`
`0000`	`EB0300`	`CLR W6`
Step 3: Initialize the Write Pointer (W7) to point to the VISI register.		
`0000`	`207847`	`MOV #VISI, W7`
`0000`	`000000`	`NOP`
Step 4: Read and clock out the contents of the next two locations of executive memory through the VISI register using the REGOUT command.		
`0000`	`BA0B96`	`TBLRDL [W6], [W7]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0001`	`<VISI>`	`Clock out contents of VISI register`
`0000`	`000000`	`NOP`
`0000`	`BADBB6`	`TBLRDH.B [W6++], [W7++]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0000`	`BAD3D6`	`TBLRDH.B [++W6], [W7--]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0001`	`<VISI>`	`Clock out contents of VISI register`
`0000`	`000000`	`NOP`
`0000`	`BA0BB6`	`TBLRDL [W6++], [W7]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
`0001`	`<VISI>`	`Clock out contents of VISI register`
`0000`	`000000`	`NOP`
Step 5: Reset the device internal PC.		
`0000`	`040200`	`GOTO 0x200`
`0000`	`000000`	`NOP`
Step 6: Repeat Steps 4 and 5 until all desired executive memory is read.		

6.0 DEVICE DETAILS

6.1 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 6-1 shows the Device ID for each device, Table 6-2 shows the Device ID registers and Table 6-3 describes the bit field of each register.

TABLE 6-1: DEVICE IDs

Device	DEVID
PIC24FJ32GA102	4202h
PIC24FJ64GA102	4206h
PIC24FJ32GA104	420Ah
PIC24FJ64GA104	420Eh
PIC24FJ32GB002	4203h
PIC24FJ64GB002	4207h
PIC24FJ32GB004	420Bh
PIC24FJ64GB004	420Fh

TABLE 6-2: PIC24FJ64GA1/GB0 DEVICE ID REGISTERS

<table>
<tr><th rowspan="2">Address</th><th rowspan="2">Name</th><th colspan="16">Bit</th></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>FF0000h</td><td>DEVID</td><td colspan="8">FAMID<7:0></td><td colspan="8">DEV<7:0></td></tr>
<tr><td>FF0002h</td><td>DEVREV</td><td colspan="12">—</td><td colspan="4">REV<3:0></td></tr>
</table>

TABLE 6-3: DEVICE ID BIT DESCRIPTIONS

Bit Field	Register	Description
FAMID<7:0>	DEVID	Encodes the family ID of the device
DEV<7:0>	DEVID	Encodes the individual ID of the device
REV<3:0>	DEVREV	Encodes the sequential (numerical) revision identifier of the device

6.2 Checksum Computation

Checksums for the PIC24FJ64GA1/GB0 families are 16 bits in size. The checksum is calculated by summing the following:

- Contents of code memory locations
- Contents of Configuration registers

Table 6-4 describes how to calculate the checksum for each device. All memory locations are summed, one byte at a time, using only their native data size. More specifically, Configuration registers are summed by adding the lower two bytes of these locations (the upper byte is ignored), while code memory is summed by adding all three bytes of code memory.

TABLE 6-4: CHECKSUM COMPUTATION

Device	Read Code Protection	Checksum Computation	Erased Checksum Value	Checksum with 0xAAAAAA at 0x0 and Last Code Address
PIC24FJ32GA102	Disabled	CFGB + SUM(0:057F7)	7883h	7685h
	Enabled	0	0000h	0000h
PIC24FJ32GA104	Disabled	CFGB + SUM(0:057F7)	7883h	7685h
	Enabled	0	0000h	0000h
PIC24FJ64GA102	Disabled	CFGB + SUM(0:0ABF7)	FA83h	F885h
	Enabled	0	0000h	0000h
PIC24FJ64GA104	Disabled	CFGB + SUM(0:0ABF7)	FA83h	F885h
	Enabled	0	0000h	0000h
PIC24FJ32GB002	Disabled	CFGB + SUM(0:057F7)	7883h	7685h
	Enabled	0	0000h	0000h
PIC24FJ32GB004	Disabled	CFGB + SUM(0:057F7)	7883h	7685h
	Enabled	0	0000h	0000h
PIC24FJ64GB002	Disabled	CFGB + SUM(0:0ABF7)	FA83h	F885h
	Enabled	0	0000h	0000h
PIC24FJ64GB004	Disabled	CFGB + SUM(0:0ABF7)	FA83h	F885h
	Enabled	0	0000h	0000h

Legend:

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 (CW1 & 0x7FFF + CW2 & 0xFFFF + CW3 & 0xFFFF + CW4 & 0xFFFF)

Note: CW1 address is last location of implemented program memory; CW2 is the (last location – 2); CW3 is the (last location – 4); CW4 is the (last location – 6).

7.0 AC/DC CHARACTERISTICS AND TIMING REQUIREMENTS

Standard Operating Conditions
Operating Temperature: 0°C to +70°C. Programming at +25°C is recommended.

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
D111A	VDD	Supply Voltage During Programming	VDDCORE + 0.1	3.60	V	Normal programming[1,2]
D111B	VDDCORE	Voltage on VDDCORE Pin During Programming	2.25	2.75	V	Normal programming[1,2]
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	—	0.4	V	IOL = 8.5 mA @ 3.6V
D090	VOH	Output High Voltage	3.0	—	V	IOH = -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	4.7	10	μF	Required for controller core
P1	TPGC	Serial Clock (PGCx) Period	100	—	ns	
P1A	TPGCL	Serial Clock (PGCx) Low Time	40	—	ns	
P1B	TPGCH	Serial Clock (PGCx) High Time	40	—	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 4-Bit Command Operand and Next 4-Bit Command	40	—	ns	
P5	TDLY2	Delay Between Last PGCx ↓ of Command Byte 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	
P9	TDLY4	Programming Executive Command Processing Time	40	—	μs	
P10	TDLY6	PGCx Low Time After Programming	400	—	ns	
P11	TDLY7	Chip Erase Time	400	—	ms	
P12	TDLY8	Page Erase Time	40	—	ms	
P13	TDLY9	Row Programming Time	2	—	ms	
P14	TR	$\overline{\text{MCLR}}$ 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 $\overline{\text{MCLR}}$ ↓	0	—	s	
P17	THLD3	$\overline{\text{MCLR}}$ ↓ to VDD ↓	100	—	ns	
P18	TKEY1	Delay from First $\overline{\text{MCLR}}$ ↓ to First PGCx ↑ for Key Sequence on PGDx	40	—	ns	
P19	TKEY2	Delay from Last PGCx ↓ for Key Sequence on PGDx to Second $\overline{\text{MCLR}}$ ↑	1	—	ms	
P20	TDLY11	Delay Between PGDx ↓ by Programming Executive to PGDx Driven by Host	23	—	µs	
P21	TDLY12	Delay Between Programming Executive Command Response Words	8	—	ns	

Note 1: VDDCORE must be supplied to the VDDCORE/VCAP pin if the on-chip voltage regulator is disabled. See **Section 2.1 "Power Requirements"** for more information.

2: 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.

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, rfPIC and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

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

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Octopus, Omniscient Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICtail, PIC^{32} logo, REAL ICE, rfLAB, 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.

© 2009, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

♻ Printed on recycled paper.

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

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

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

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

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

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

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4080

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: 886-7-536-4818
Fax: 886-7-536-4803

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

03/26/09