PIC24FJ256GA705 Family Flash Programming Specification

1.0 DEVICE OVERVIEW

This document defines the programming specification for the PIC24FJ256GA705 family of 16-bit microcontrollers. This programming specification is required only for those developing programming support for the following devices:

- PIC24FJ256GA705
- PIC24FJ128GA705
- PIC24FJ64GA705
- PIC24FJ256GA704
- PIC24FJ128GA704
- PIC24FJ64GA704
- PIC24FJ256GA702
- PIC24FJ128GA702
- PIC24FJ64GA702

Topics covered include:

© 2015-2018 Microchip Technology Inc.

2.0 PROGRAMMING OVERVIEW

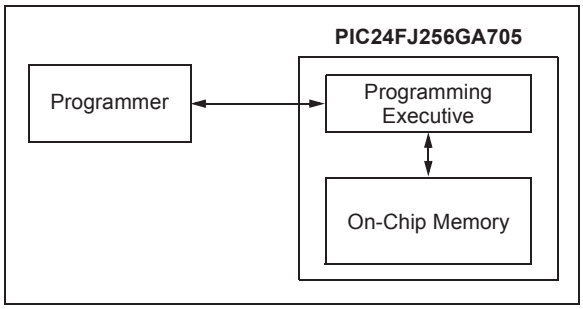
There are two methods of programming that are discussed in this programming specification:

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

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

The Enhanced ICSP protocol uses a faster method that takes advantage of the Programming Executive (PE), as illustrated in Figure 2-1. The PE 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 a PIC24FJ256GA705 family device without dealing with the low-level programming protocols.

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



This programming specification is divided into two major sections that describe the programming methods independently. **Section 3.0 "Device Programming – ICSP"** describes the ICSP method. **Section 4.0 "Device Programming – Enhanced ICSP"** describes the Enhanced ICSP method.

2.1 Required Connections

These devices require specific connections for programming to take place. These connections include power, VCAP, MCLR and one programming pair (PGEDx/PGECx). Table 2-1 describes these connections (refer to the specific device data sheet for pin descriptions and power connection requirements).

2.2 Power Requirements

All PIC24FJ256GA705 family devices power their core digital logic at a nominal 1.8V. To simplify system design, all devices in the PIC24FJ256GA705 family incorporate an on-chip regulator that allows the device to run its core logic from VDD.

The regulator provides power to the core from the other VDD pins. A low-ESR capacitor (such as ceramic or tantalum) must be connected to the VCAP pin (see 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 9.0 "AC/DC Characteristics and Timing Requirements"**.

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

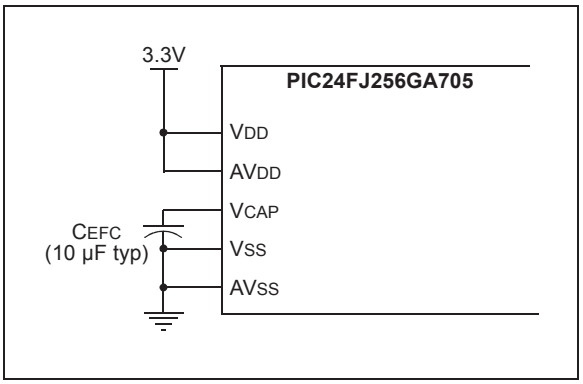


TABLE 2-1: PINS USED DURING PROGRAMMING

Pin Name	Pin Type	Pin Description
MCLR	I	Programming Enable
VDD and AVDD(1)	P	Power Supply(1)
VSS and AVSS(1)	P	Ground(1)
VCAP	P	On-Chip Voltage Regulator Filter Capacitor
PGECx	I	Programming Pin Pair: Serial Clock
PGEDx	I/O	Programming Pin Pair: Serial Data

Legend: I = Input O = Output P = Power

Note 1: All power supply and ground pins must be connected, including AVDD and AVSS.

© 2015-2018 Microchip Technology Inc.

2.3 Pin Diagrams

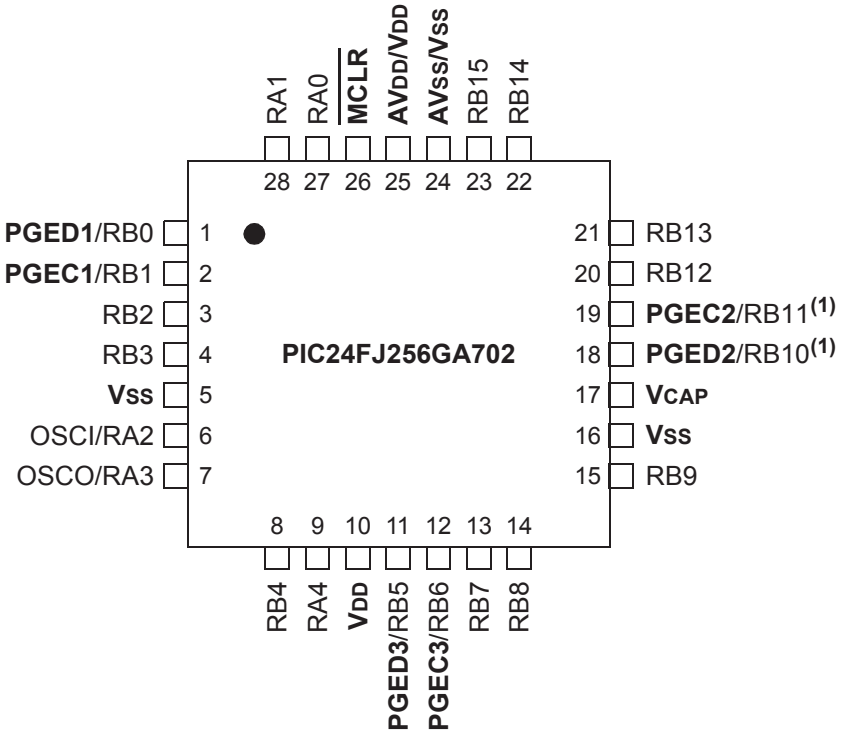
The following figures show the pin diagrams for the PIC24FJ256GA705 families. The pins that are required for programming are listed in Table 2-1 and are indicated in bold text in the figures. Refer to the appropriate device data sheet for complete pin descriptions.

2.3.1 PGECx AND PGEDx PIN PAIRS

All devices in the PIC24FJ256GA705 family have three separate pairs of programming pins, labeled 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.

Pin Diagrams (28-Pin QFN)

28-Pin QFN

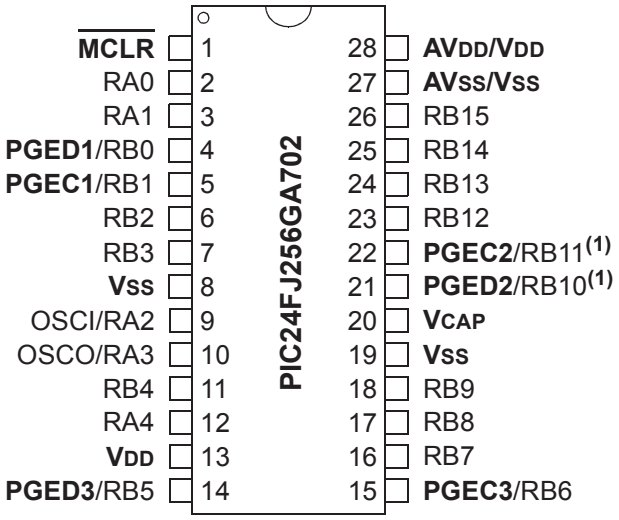


Legend: **Bold** indicates pins used in device programming; the complete list of functions associated with programming/emulation pins is shown in the accompanying table.

Note 1: The PGEC2/PGED2 pin pair on 28-pin packages does not support Enhanced ICSP™, but does allow normal ICSP operation.

Pin Diagrams (28-Pin PDIP)

28-Pin PDIP



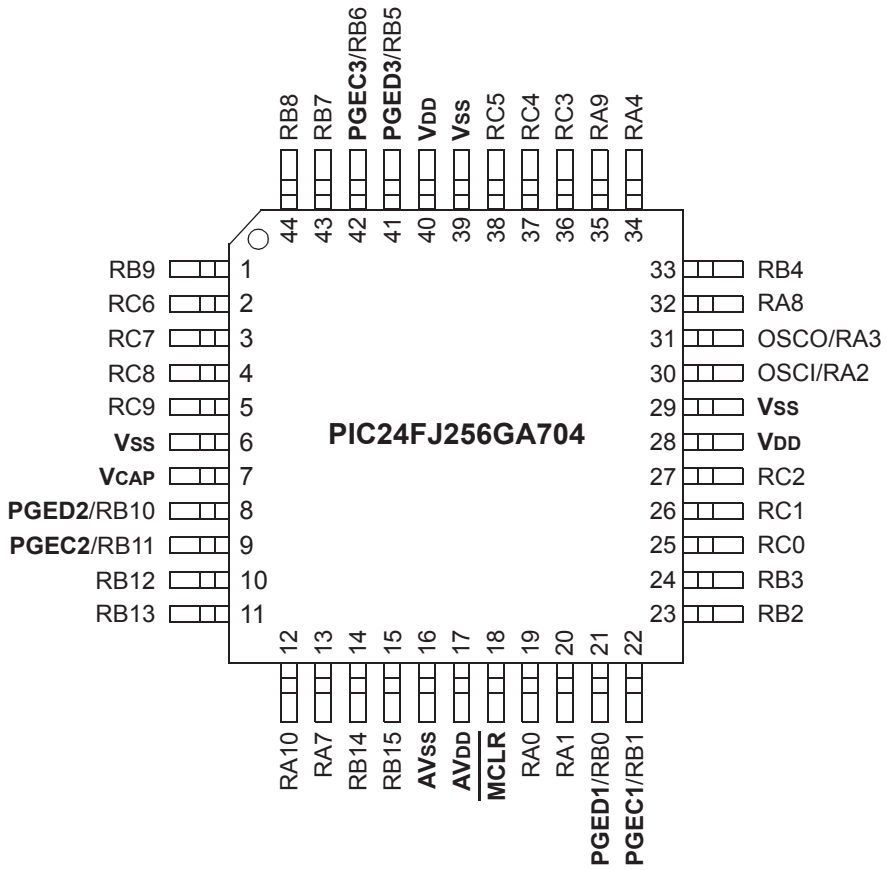
Legend: **Bold** indicates pins used in device programming; the complete list of functions associated with programming/emulation pins is shown in the accompanying table.

Note 1: The PGEC2/PGED2 pin pair on 28-pin packages does not support Enhanced ICSP™, but does allow normal ICSP operation.

© 2015-2018 Microchip Technology Inc.

Pin Diagrams (44-Pin TQFP/QFN)

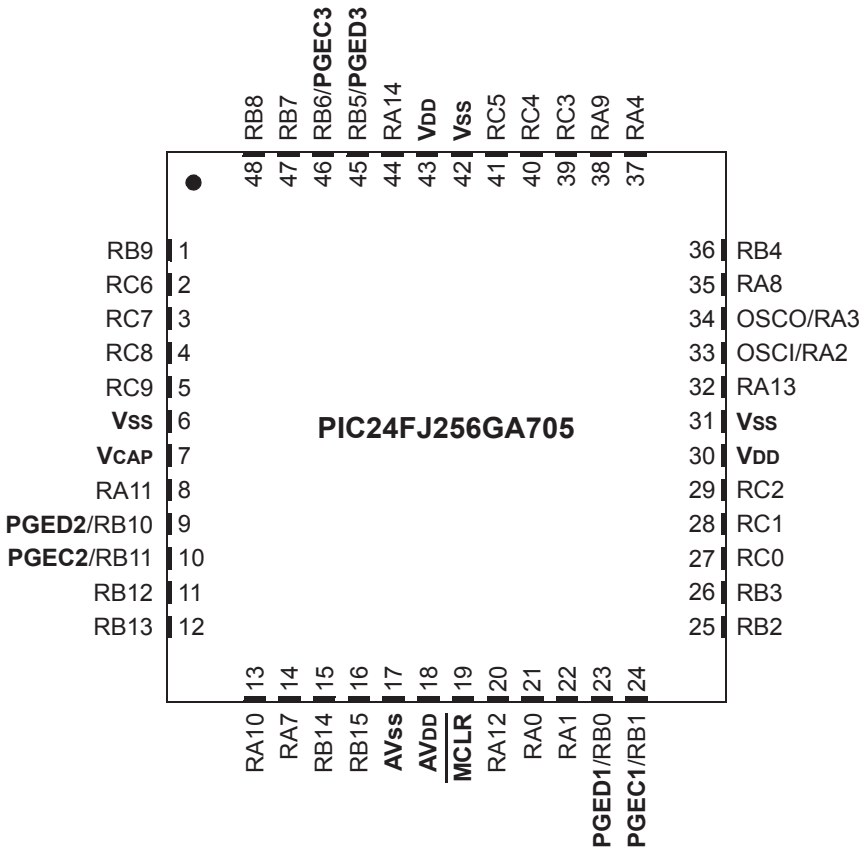
44-Pin TQFP/QFN



Legend: **Bold** indicates pins used in device programming; the complete list of functions associated with programming/emulation pins is shown in the accompanying table.

Pin Diagrams (48-Pin TQFP/QFN)

48-Pin TQFP/QFN



Legend: **Bold** indicates pins used in device programming; the complete list of functions associated with programming/emulation pins is shown in the accompanying table.

© 2015-2018 Microchip Technology Inc.

2.4 Program Memory Write/Erase Requirements

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

2.5 Memory Map

The program memory space is organized in word-addressable blocks. Although it is treated as 24 bits wide, it is more appropriate to think of each address of the program memory as a lower and upper word, with the upper byte of the upper word being unimplemented.

The lower word always has an even address, while the upper word has an odd address.

Program memory addresses are always word-aligned on the lower word and addresses are incremented or decremented by two during code execution. This arrangement also provides compatibility with data memory space addressing and makes it possible to access data in the program memory space.

Locations, 0x80 0100 through 0x80 0FFE, are reserved for executive code memory. This region stores the Programming Executive (PE) and the debugging executive, which is used for device programming. This region of memory cannot be used to store user code. See **Section 6.0 "The Programming Executive"** for more information.

Locations, 0x80 1700 through 0x80 17FE, are reserved for the customer data. This area can be used for storing product information, such as serial numbers, system manufacturing dates, manufacturing lot numbers and other application-specific information. It is described in **Section 2.6.3 "Customer OTP Memory"**.

Locations, 0xFF 0000 and 0xFF 0002, are reserved for the Device ID Word registers. These bits can be used by the programmer to identify which device type is being programmed. They are described in **Section 7.0 "Device ID"**. The Device ID registers read out normally, even after code protection is applied.

Figure 2-3 shows the generic memory map for the devices described in this specification. See the **"Memory Organization"** chapter in the specific device data sheet for exact memory addresses.

Table 2-2 lists the code memory size, the size of the erase blocks and the number of erase blocks present in each device variant.

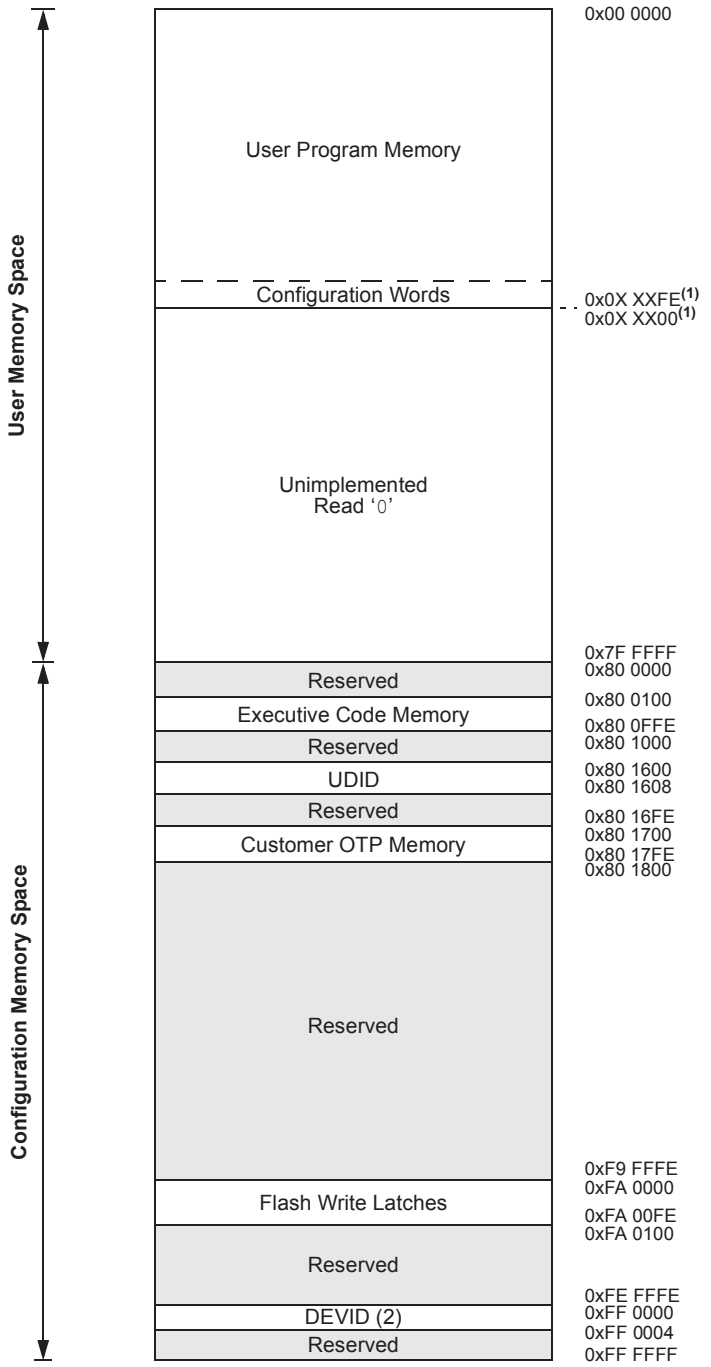
TABLE 2-2: PROGRAM MEMORY SIZES AND BOUNDARIES(2)

Device	Program Memory Upper Boundary (Instruction Words)	Write Blocks(1)	Erase Blocks(1)
PIC24FJ256GA70X	0x02 AFFE (86K)	1376	172
PIC24FJ128GA70X	0x01 5FFE (44K)	704	88
PIC24FJ64GA70X	0x00 AFFE (22K)	352	44

Note 1: 1 Write Block (Row) = 128 Instruction Words; 1 Erase Block (Page) = 1024 Instruction Words.

2: To maintain integer page sizes, the memory sizes are not exactly half of each other.

FIGURE 2-3: PROGRAM MEMORY MAP



Legend: Memory areas are not shown to scale.

Note 1: Exact boundary addresses are determined by the size of the implemented program memory. See Table 2-2 for details.

© 2015-2018 Microchip Technology Inc.

2.6 Configuration Bits

2.6.1 OVERVIEW

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

Table 2-3 lists the Configuration register address range for each device. Table 2-4 lists all of the Configuration bits found in the PIC24FJ256GA705 family devices, as well as their Configuration register locations. Refer to the **"Special Features"** chapter in the specific device data sheet for the full Configuration register description for a specific device.

2.6.2 CODE-PROTECT CONFIGURATION BITS

The devices implement an intermediate security feature defined by the FSEC register. The Boot Segment (BS) is the higher privileged segment and the General Segment (GS) is the lower privileged segment. The total user code memory can be split into BS or GS. The size of the segments is determined by the BSLIM<12:0> bits. The relative location of the segments within user space does not change, such that BS (if present) occupies the memory area just after the Interrupt Vector Table (IVT), and the GS occupies the space just after the BS (or if the Alternate IVT is enabled, just after it).

The Configuration Segment (or CS) is a small segment (less than a page, typically just one row) within user Flash address space. It contains all user configuration data that is loaded by the NVM controller during the Reset sequence.

TABLE 2-3: CONFIGURATION WORD ADDRESSES

Configuration Register	PIC24FJ256GA70X	PIC24FJ128GA70X	PIC24FJ64GA70X
FSEC	0x02 AF00	0x01 5F00	0x00 AF00
FBSLIM	0x02 AF10	0x01 5F10	0x00 AF10
FSIGN	0x02 AF14	0x01 5F14	0x00 AF14
FOSCSEL	0x02 AF18	0x01 5F18	0x00 AF18
FOSC	0x02 AF1C	0x01 5F1C	0x00 AF1C
FWDT	0x02 AF20	0x01 5F20	0x00 AF20
FPOR	0x02 AF24	0x01 5F24	0x00 AF24
FICD	0x02 AF28	0x01 5F28	0x00 AF28
FDEVOPT1	0x02 AF2C	0x01 5F2C	0x00 AF2C

TABLE 2-4: CONFIGURATION REGISTER MAP

Register Name	Bits 23-16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FSEC	—	AIVTDIS	—	—	—	CSS<2:0>			CWRP	GSS<1:0>		GWRP	—	BSS<2:0>			BWRP
FBSLIM	—	—	—	—	BSLIM<12:0>												
FSIGN	—	SIGN	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
FOSCSEL	—	—	—	—	—	—	—	PLL96DIV<1:0>		IESO	PLLMODE<3:0>				FNOSC<2:0>		
FOSC	—	—	—	—	—	—	—	—	—	FCKSM<1:0>		IOL1WAY	PLLSS	SOSCSEL	OSCIOFNC	POSCMD<1:0>	
FWDT	—	—	WDTCLKS<1:0>		—	WDTCMX	—	WDTWIN<1:0>		WINDIS	FWDTEN<1:0>		WDTPRE	WDTPS<3:0>			
FPOR	—	—	—	—	—	—	—	—	—	—	—	—	—	DNVPEN	RETVRDIS	BOREN<1:0>	
FICD	—	NOBTSWP	—	—	—	—	—	—	—	BKBUG	—	JTAGEN	—	—	—	ICS<1:0>	
FDEVOPT1	—	—	—	—	—	—	—	—	—	—	—	—	DOPT<4:1>				—

Legend: — = unimplemented, read as '1'

© 2015-2018 Microchip Technology Inc.

2.6.3 CUSTOMER OTP MEMORY

PIC24FJ256GA705 family devices provide 384 bytes of One-Time-Programmable (OTP) memory, located at addresses, 0x80 1700 through 0x80 17FE. This memory can be used for persistent storage of applicationspecific information that will not be erased by reprogramming the device. This includes many types of information, such as:

- Application checksums
- Code revision information
- Product information
- Serial numbers
- System manufacturing dates
- Manufacturing lot numbers

Customer OTP memory may be programmed in any mode, including user RTSP mode, but it cannot be erased. Data is not cleared by a Chip Erase.

Note:	The OTP memory resides within the ECC controlled Flash memory region, but is exempt from erase operations. Therefore, the OTP must strictly be written to once. Writing multiple times to the OTP region may cause a double-bit ECC error, which will force a processor Reset when the OTP region is read. There is no mechanism to remove the ECC error if the OTP area has been written more than once.

3.0 DEVICE PROGRAMMING – ICSP

ICSP mode is a special programming protocol that allows you to read and write to device memory. The ICSP mode is the most direct method used to program the device, which is accomplished by applying control codes and instructions serially to the device, using the PGECx and PGEDx pins. ICSP mode also has the ability to read executive memory to determine if the Programming Executive (PE) is present and to write the PE to executive memory if Enhanced ICSP mode will be used.

In ICSP mode, the system clock is taken from the PGECx 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. PGEDx is used to shift data in, and PGECx is used as both the serial shift clock and the CPU execution clock.

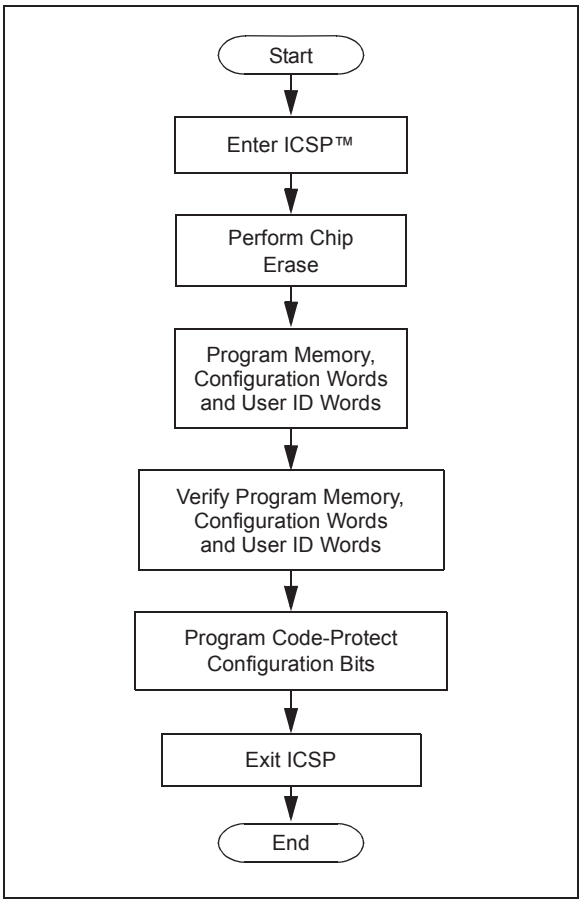
Note 1: During ICSP operation, the operating frequency of PGECx must not exceed 5 MHz.

2: ICSP mode is slower than Enhanced ICSP mode for programming.

3.1 Overview of the Programming Process

Figure 3-1 illustrates the high-level overview of the programming process. After entering ICSP mode, the first action is to Chip Erase program memory. Next, the code memory is programmed, followed by the device Configuration bits. Code memory (including the Configuration bits) is then verified to ensure that programming was successful. Then, the code-protect Configuration bits are programmed, if required.

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



© 2015-2018 Microchip Technology Inc.

3.2 Entering ICSP Mode

As shown in Figure 3-2, entering ICSP Program/Verify mode requires three steps:

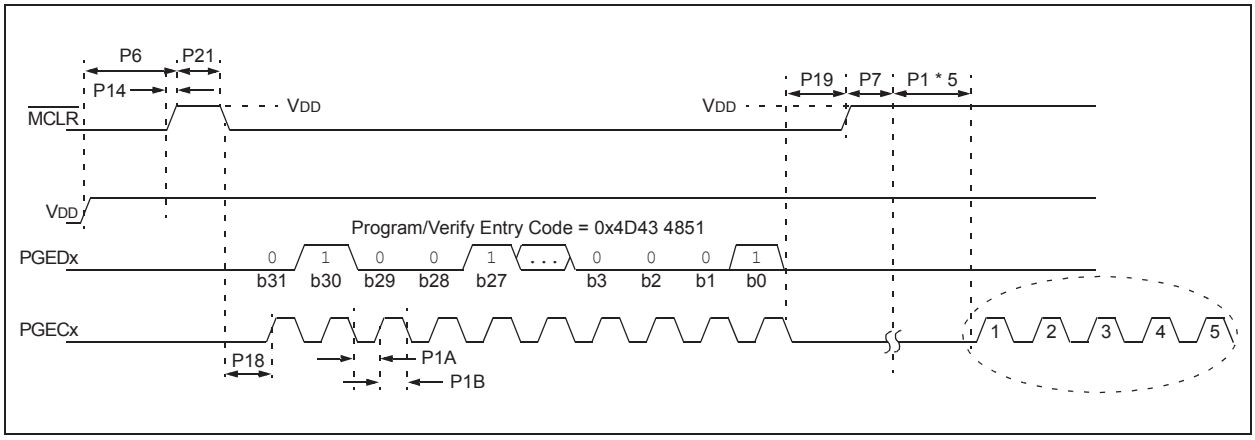
1. $\overline{\text{MCLR}}$ is briefly driven high, then low (P21).
2. A 32-bit key sequence is clocked into PGEDx. The interval of at least P18 must elapse before presenting the key sequence on PGEDx.
3. $\overline{\text{MCLR}}$ is held low during a specified period, P19, and then driven high.
4. After a P7 + 5 * P1 delay, five clock pulses must be generated on the PGECx pin.

Note: If a capacitor is present on the $\overline{\text{MCLR}}$ pin, the high time for entering ICSP mode can vary.

The key sequence is a specific 32-bit pattern, '0100 1101 0100 0011 0100 1000 0101 0001' (more easily remembered as 0x4D43 4851 in hexadecimal). The device will enter ICSP mode only if the sequence is valid. The Most Significant bit (MSb) of the most significant nibble must be shifted in first.

On successful entry, the program memory can be accessed and programmed in serial fashion.

FIGURE 3-2: ENTERING ICSP™ MODE



3.3 ICSP Operation

After entering into ICSP mode, the CPU is Idle. Execution of the CPU is governed by an internal state machine. A 4-bit control code is clocked in using PGECx and PGEDx, 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 through 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 register.
0010-1111	N/A	Reserved.

3.3.1 SIX SERIAL INSTRUCTION EXECUTION

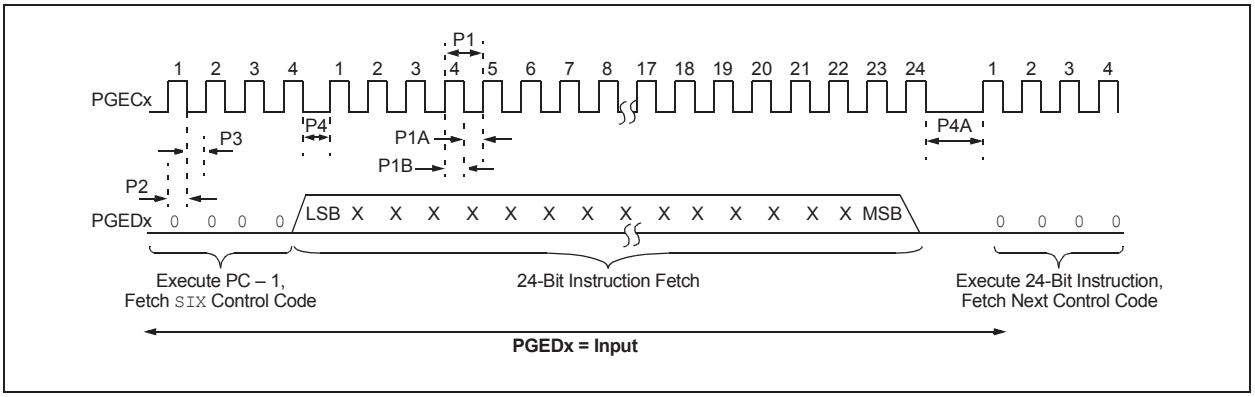
The SIX control code allows execution of assembly instructions. When the SIX code is received, the CPU is suspended for 24 clock cycles as the instruction is then clocked into the internal buffer. Once the instruction is shifted in, the state machine allows it to be executed over the next four clock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

Note 1: The device will latch input PGEDx data on the rising edge of PGECx. For all data transmissions, the Least Significant bit (LSb) is transmitted first.

2: TBLRDH, TBLRDL, TBLWTH and TBLWTL instructions must be followed by two NOP instructions.

3: During ISCP programming, the CLKO pin will toggle. If external logic is connected to CLKO, be sure that this toggling will not effect the circuitry during the programming sequence.

FIGURE 3-3: SIX SERIAL EXECUTION



© 2015-2018 Microchip Technology Inc.

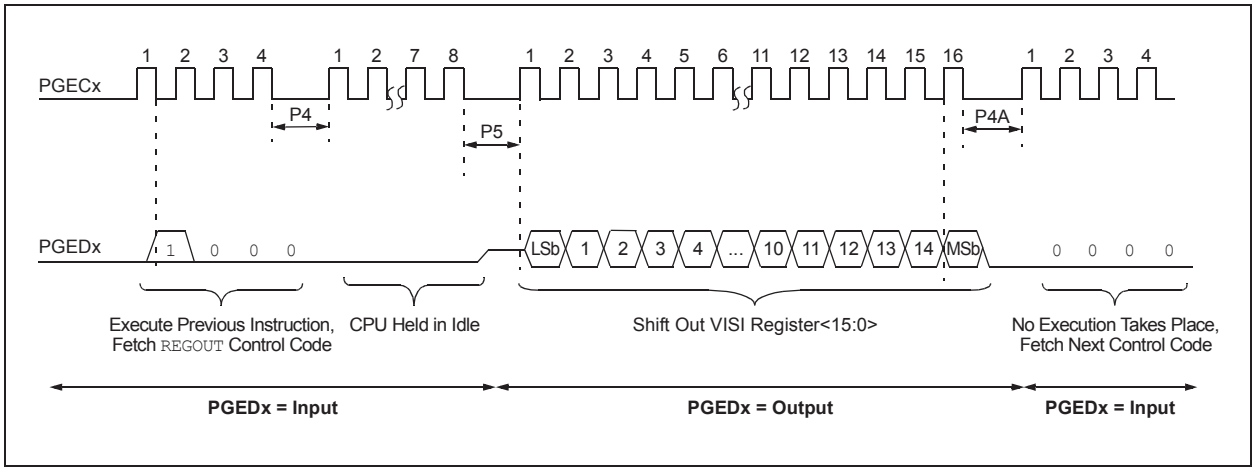
3.3.2 REGOUT SERIAL INSTRUCTION EXECUTION

The REGOUT control code allows for data to be extracted from the device in ICSP mode. It is used to clock the contents of the VISI register out of the device over the PGEDx pin. After the REGOUT control code is received, the CPU is held Idle for eight cycles. After these eight cycles, an additional 16 cycles are required to clock the data out (see Figure 3-4).

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

Note:	The device will output data on the PGEDx line on the rising edge of PGECx. For all data transmissions, the Least Significant bit (LSb) is transmitted first.

FIGURE 3-4: REGOUT SERIAL EXECUTION



3.4 Flash Memory Programming in ICSP Mode

3.4.1 PROGRAMMING OPERATIONS

Flash memory write/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. The WR control bit is cleared by hardware when the programming operation is complete. The ICSP programmer must supply enough clock pulses on the PGECx pin to complete the erase or program operation. Refer to **Section 9.0 "AC/DC Characteristics and Timing Requirements"** for detailed information about the maximum number of clock pulses required for erase or write operations.

TABLE 3-2: NVMCON ERASE OPERATIONS

NVMCON Value	Erase Operation
0x400E	Chip Erase (erases user memory; does not erase executive memory, Device ID or customer OTP).
0x4003	Erase a page of program or executive memory.

TABLE 3-3: NVMCON WRITE OPERATIONS

NVMCON Value	Write Operation
0x4001	Double-word program operation.
0x4002	Row programming operation.

3.4.2 STARTING AND STOPPING A PROGRAMMING CYCLE

For protection against accidental operations, the erase/write initiate sequence must be written to the NVMKEY register to allow any erase or program operation to proceed. The three instructions following the start of the programming sequence should be NOPs. To start an erase or write sequence, the following steps must be completed:

1. Write 0x55 to the NVMKEY register.
2. Write 0xAA to the NVMKEY register.
3. Set the WR bit in the NVMCON register.
4. Execute three NOP instructions.

All erase and write cycles are self-timed. The WR bit can be polled to supply enough PGECx clock pulses for the operation and determine if the erase or write cycle has been completed.

© 2015-2018 Microchip Technology Inc.

3.5 Erasing Program Memory

The general procedure for erasing user memory is shown in Figure 3-5. The process for Chip Erase and Page Erase are all similar, and are described in Table 3-4 through Table 3-5.

The last row of the last page of program memory contains the Configuration Words. Before programming these Words, they must be erased. If they are erased with a Page Erase operation, all other rows in the page will also be erased. Users may want to either avoid using the rest of this page for application code or ensure that the non-configuration data in the CS page is copied before the erase and reprogrammed afterwards.

Note 1: Program memory must be erased before writing any data to program memory.

2: For Page Erase operations, the NVMADR/NVMADRU registers must also be loaded with the address of the page to be erased.

FIGURE 3-5: ERASE FLOW

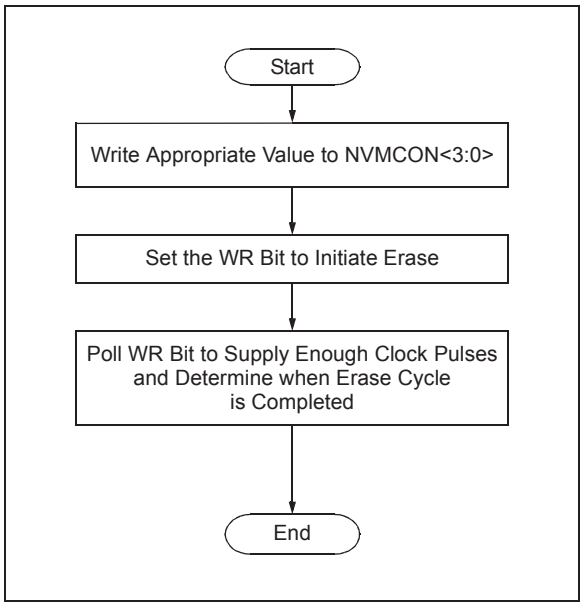


TABLE 3-4: SERIAL EXECUTION FOR CHIP ERASE

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
Step 2: Configure the NVMCON register to perform a Chip Erase.		
0000	2400E0	MOV #0x400E, W0
0000	883B00	MOV W0, NVMCON
Step 3: Set the WR bit.		
0000	200550	MOV #0x55, W0
0000	883B30	MOV W0, NVMKEY
0000	200AA0	MOV #0xAA, W0
0000	883B30	MOV W0, NVMKEY
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 4: Repeat this step to poll the WR bit until it is cleared by hardware.		
0000	040200	GOTO 0x200
0000	000000	NOP
0000	803B02	MOV NVMCON, W2
0000	000000	NOP
0000	883C22	MOV W2, VISI
0000	000000	NOP
0001	<VISI>	; Clock out the contents of the VISI register.
0000	000000	NOP
Step 5: Clear the WREN bit.		
0000	200000	MOV #0000, W0
0000	883B00	MOV W0, NVMCON

TABLE 3-5: SERIAL EXECUTION FOR PAGE 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 register to erase a page.		
0000	240030	MOV #0x4003, W0
0000	883B00	MOV W0, NVMCON
Step 3: Load the address of the page to be erased into the NVMADR register pair.		
0000	200000	MOV #PAGE_ADDR_LO, W0
0000	883B10	MOV W0, NVMADR
0000	200000	MOV #PAGE_ADDR_HI, W0
0000	883B20	MOV W0, NVMADRU
Step 4: Set the WR bit.		
0000	200550	MOV #0x55, W0
0000	883B30	MOV W0, NVMKEY
0000	200AA0	MOV #0xAA, W0
0000	883B30	MOV W0, NVMKEY
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 5: Repeat this step to poll the WR bit until it is cleared by hardware.		
0000	040200	GOTO 0x200
0000	000000	NOP
0000	803B02	MOV NVMCON, W2
0000	000000	NOP
0000	883C22	MOV W2, VISI
0000	000000	NOP
0001	<VISI>	; Clock out the contents of the VISI register.
0000	000000	NOP
Step 6: Clear the WREN bit.		
0000	200000	MOV #0000, W0
0000	883B00	MOV W0, NVMCON

© 2015-2018 Microchip Technology Inc.

3.6 Writing Code Memory

For PIC24FJ256GA705 devices, code memory is written in three steps:

- Writing the data to memory-mapped write latches (located in the configuration memory space at addresses, 0xFA 0000 through 0xFA 00FE);
- Setting a destination address; and
- Initiating the memory write sequence

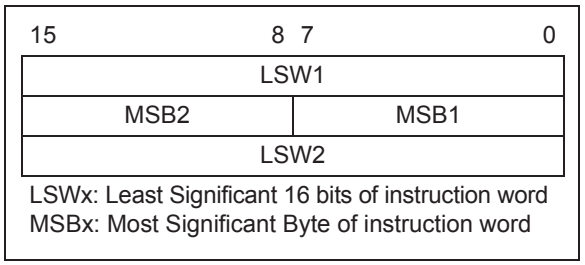
There are two methods available for writing to code memory: double-word writes using the write latches or 128-word row writes. Figure 3-7 provides a high-level description of the two methods.

Double-word writes program code memory with two instruction words at a time. Two words are loaded into the write latches. Next, the write sequence is initiated, and finally, the WR bit is checked for the sequence to be complete. This process continues for all the data to be programmed.

Table 3-6 provides an example of ICSP programming for a double-word write operation.

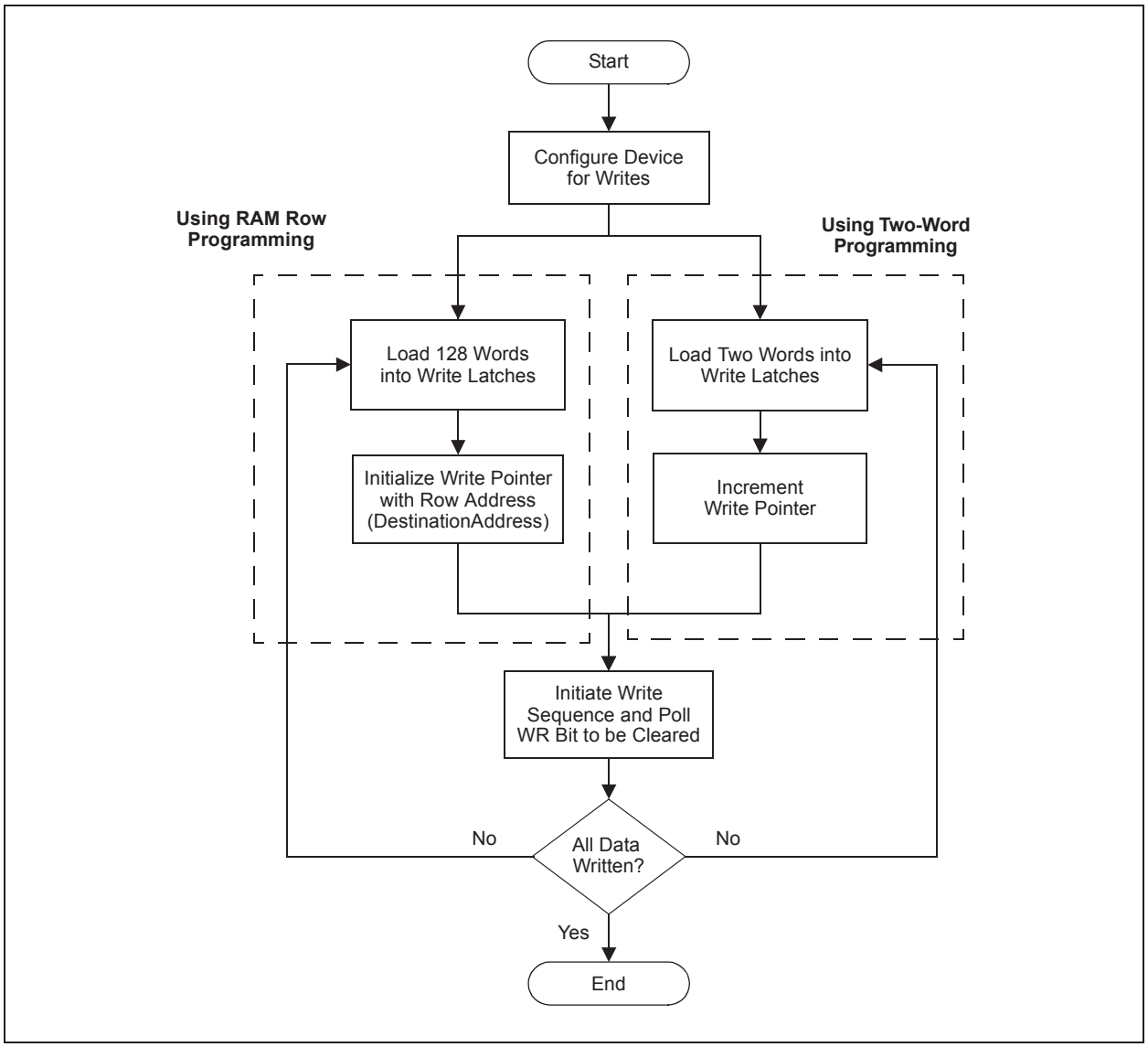
The data loaded into the programming latches must be in the packed format, as shown in Figure 3-6.

FIGURE 3-6: PACKED INSTRUCTION WORD FORMAT



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

FIGURE 3-7: PROGRAM CODE MEMORY FLOW



© 2015-2018 Microchip Technology Inc.

TABLE 3-6: SERIAL INSTRUCTION EXECUTION FOR PROGRAMMING CODE MEMORY: TWO-WORD LATCH WRITES

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 TBLPAG register for writing to the latches.		
0000	200FAC	MOV #0xFA, W12
0000	8802AC	MOV W12, TBLPAG
Step 3: Load W0:W2 with the next two packed instruction words to program.		
0000	2xxxx0	MOV #<LSW0>, W0
0000	2xxxx1	MOV #<MSB1:MSB0>, W1
0000	2xxxx2	MOV #<LSW1>, W2
Step 4: Set the Read Pointer (W6) and Write Pointer (W7), and load the (next set of) write latches.		
0000	EB0300	CLR W6
0000	000000	NOP
0000	EB0380	CLR W7
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.W [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
Step 5: Set the NVMADRU/NVMADR register pair to point to the correct address.		
0000	2xxxx3	MOV #DestinationAddress<15:0>, W3
0000	2xxxx4	MOV #DestinationAddress<23:16>, W4
0000	883B13	MOV W3, NVMADR
0000	883B24	MOV W4, NVMADRU
Step 6: Set the NVMCON register to program two instruction words.		
0000	24001A	MOV #0x4001, W10
0000	883B0A	MOV W10, NVMCON
0000	000000	NOP
Step 7: Initiate the write cycle.		
0000	200551	MOV #0x55, W1
0000	883B31	MOV W1, NVMKEY
0000	200AA1	MOV #0xAA, W1
0000	883B31	MOV W1, NVMKEY
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP

TABLE 3-6: SERIAL INSTRUCTION EXECUTION FOR PROGRAMMING CODE MEMORY: TWO-WORD LATCH WRITES (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 8: Wait for program operation to complete and make sure the WR bit is clear.		
0000	803B00	MOV NVMCON, W0
0000	883C20	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	; Clock out the contents of the VISI register.
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
—	—	; Repeat until the WR bit is clear.
Step 9: Repeat Steps 3-8 until all code memory is programmed.		
Step 10: Clear the WREN bit.		
0000	200000	MOV #0000, W0
0000	883B00	MOV W0, NVMCON

© 2015-2018 Microchip Technology Inc.

Row writes program one row (128 instruction words) at a time. First, the Table Pointer is initialized to point to the program latches and data is written into them with Table Writes. Next, the Write Pointer is initialized (NVMADRU and NVMADR register pair) with the row address (DestinationAddress). Finally, the write sequence is initiated and the WR bit is checked for the row programming to be complete. This process is repeated for all data to be programmed. Table 3-7 shows the ICSP programming details for row writes.

To minimize programming time, the data to be programmed is stored in the W0:W5 registers in a packed data format (Figure 3-8). This is the same packed format used by the PE. See **Section 6.2.2 "Packed Data Format"** for additional information.

FIGURE 3-8: PACKED INSTRUCTION WORD STORAGE IN W0:W5

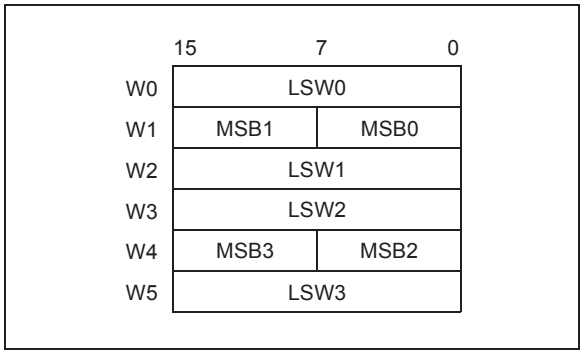


TABLE 3-7: SERIAL INSTRUCTION EXECUTION FOR PROGRAMMING CODE MEMORY: ROW WRITES

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 register to program 128 instruction words.		
0000	240020	MOV #0X4002, W0
0000	883B00	MOV W0, NVMCON
Step 3: Initialize the TBLPAG register for writing to the latches.		
0000	200FAC	MOV #0xFA, W12
0000	8802AC	MOV W12, TBLPAG
Step 4: Load W0:W5 with the next four 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
Step 5: Set the Read Pointer (W6) and load the (next set of) write latches.		
0000	EB0300	CLR W6
0000	000000	NOP
0000	EB0380	CLR W7
0000	000000	NOP
0000	BB0BB6	TBLWTL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP

TABLE 3-7: SERIAL INSTRUCTION EXECUTION FOR PROGRAMMING CODE MEMORY: ROW WRITES (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 5: Set the Read Pointer (W6) and load the (next set of) write latches. (Continued)		
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, for a total of 32 times, to load the write latches with 128 instructions.		
Step 7: Set the NVMADRU/NVMADR register pair to point to the correct address.		
0000	2xxxx3	MOV #DestinationAddress<15:0>, W3
0000	2xxxx4	MOV #DestinationAddress<23:16>, W4
0000	883B13	MOV W3, NVMADR
0000	883B24	MOV W4, NVMADRU
Step 8: Execute the WR bit unlock sequence and initiate the write cycle.		
0000	200550	MOV #0x55, W0
0000	883B30	MOV W0, NVMKEY
0000	200AA0	MOV #0xAA, W0
0000	883B30	MOV W0, NVMKEY
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 9: Repeat this step to poll the WR bit until it is cleared by 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 the contents of the VISI register.
0000	000000	NOP
Step 10: Reset the device's internal Program Counter (PC).		
0000	040200	GOTO 0x200
0000	000000	NOP
Step 11: Repeat Steps 3 through 9 until all code memory is programmed.		
Step 12: Clear the WREN bit.		
0000	200000	MOV #0000, W0
0000	883B00	MOV W0, NVMCON

© 2015-2018 Microchip Technology Inc.

3.7 Writing Configuration Bits

The procedure for writing Configuration bits is similar to the procedure for writing code memory. Table 3-8 shows the ICSP programming details for writing the Configuration bits.

To change the values of the Configuration bits once they have been programmed, the device must be erased, as described in **Section 3.5 "Erasing Program Memory"**, and reprogrammed to the desired value. Note that it is only possible to program a Configuration bit from '1' to '0' to enable code protection; it is not possible to program it from '0' to '1'.

TABLE 3-8: SERIAL INSTRUCTION EXECUTION FOR PROGRAMMING CONFIGURATION WORDS: TWO-WORD LATCH WRITES

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 TBLPAG register for writing to the latches.		
0000	200FAC	MOV #0xFA, W12
0000	8802AC	MOV W12, TBLPAG
Step 3: Load W0:W1 with the next two Configuration Words to program.		
0000	2xxxx0	MOV #<Config lower word data>, W0
0000	2FFxx1	MOV #<Config upper word data>, W1
—	—	; Upper word is 0xFFFF for all Configuration Words except FBTSEQ
0000	2FFFF2	MOV #0xFFFF, W2
Step 4: Set the Read Pointer (W6) and Write Pointer (W7), and load the (next set of) write latches.		
0000	EB0300	CLR W6
0000	000000	NOP
0000	EB0380	CLR W7
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.W [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
Step 5: Set the NVMADRU/NVMADR register pair to point to the correct address.		
0000	2xxxx3	MOV #DestinationAddress<15:0>, W3
0000	2xxxx4	MOV #DestinationAddress<23:16>, W4
0000	883B13	MOV W3, NVMADR
0000	883B24	MOV W4, NVMADRU

TABLE 3-8: SERIAL INSTRUCTION EXECUTION FOR PROGRAMMING CONFIGURATION WORDS: TWO-WORD LATCH WRITES (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 6: Set the NVMCON register to program two instruction words.		
0000	24001A	MOV #0x4001, W10
0000	883B0A	MOV W10, NVMCON
0000	000000	NOP
Step 7: Initiate the write cycle.		
0000	200551	MOV #0x55, W1
0000	883B31	MOV W1, NVMKEY
0000	200AA1	MOV #0xAA, W1
0000	883B31	MOV W1, NVMKEY
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 8: Wait for program operation to complete and make sure the WR bit is clear.		
0000	803B00	MOV NVMCON, W0
0000	883C20	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	; Clock out the contents of the VISI register.
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
—	—	; Repeat until the WR bit is clear.
Step 9: Repeat Steps 3-8 until all code memory is programmed.		
Step 10: Clear the WREN bit.		
0000	200000	MOV #0000, W0
0000	883B00	MOV W0, NVMCON

© 2015-2018 Microchip Technology Inc.

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 3-9 shows the ICSP programming details for reading code memory.

To minimize reading time, the same packed data format that the PE uses is utilized. See **Section 6.2 "Programming Executive Commands"** for more details on the packed data format.

3.9 Reading Configuration Words

The procedure for reading Configuration Words is identical to the procedure for reading code memory, shown in Table 3-9. Since there are multiple Configuration Words, they are read one at a time.

TABLE 3-9: SERIAL INSTRUCTION EXECUTION FOR READING 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: Initialize the Write Pointer (W7) to point to the VISI register.		
0000	207847	MOV #VISI, W7
0000	000000	NOP
Step 3: Initialize the TBLPAG register and the Read Pointer (W6) for a TBLRD instruction.		
0000	200xx0	MOV #<SourceAddress23:16>, W0
0000	8802A0	MOV W0, TBLPAG
0000	2xxxx6	MOV #<SourceAddress15:0>, W6
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 the contents of the 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 the contents of the VISI register.
0000	000000	NOP
0000	BA0BB6	TBLRDL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP
0001	<VISI>	; Clock out the contents of the VISI register.
0000	000000	NOP
Step 5: Reset the device's internal Program Counter.		
0000	040200	GOTO 0x200
0000	000000	NOP
Step 6: Repeat Steps 3 through 5 until all desired code memory is read (note that "Reset the device's internal Program Counter" will be Step 5).		

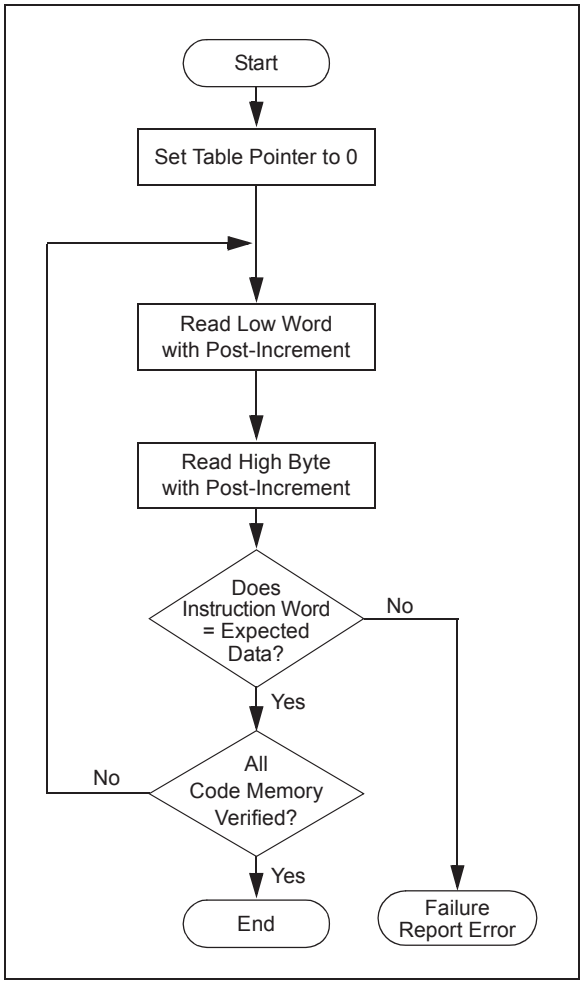
3.10 Verify Code Memory and Configuration Bits

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

The verify process is illustrated in Figure 3-9. The lower word of the instruction is read, and then the lower byte of the upper word is read and compared against the instruction stored in the programmer's buffer. Refer to **Section 3.8 "Reading Code Memory"** for implementation details of reading code memory.

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

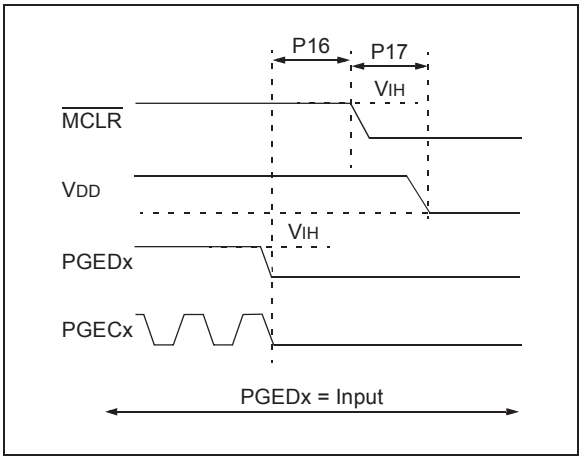
FIGURE 3-9: VERIFY CODE MEMORY FLOW



3.11 Exiting ICSP Mode

Exiting Program/Verify mode is done by removing VIH from MCLR, as illustrated in Figure 3-10. The only requirement for exit is that an interval, P16, should elapse between the last clock, and program signals on PGECx and PGEDx, before removing VIH.

FIGURE 3-10: EXITING ICSP™ MODE



© 2015-2018 Microchip Technology Inc.

4.0 DEVICE PROGRAMMING – ENHANCED ICSP

This section discusses programming the device through Enhanced ICSP and the Programming Executive (PE). The PE resides in executive memory (separate from code memory) and is executed when Enhanced ICSP Programming mode is entered. The PE provides the mechanism for the programmer (host device) to program and verify the PIC24FJ256GA705 family devices, using a simple command set and communication protocol. There are several basic functions provided by the PE:

- Read Memory
- Erase Memory
- Program Memory
- Blank Check

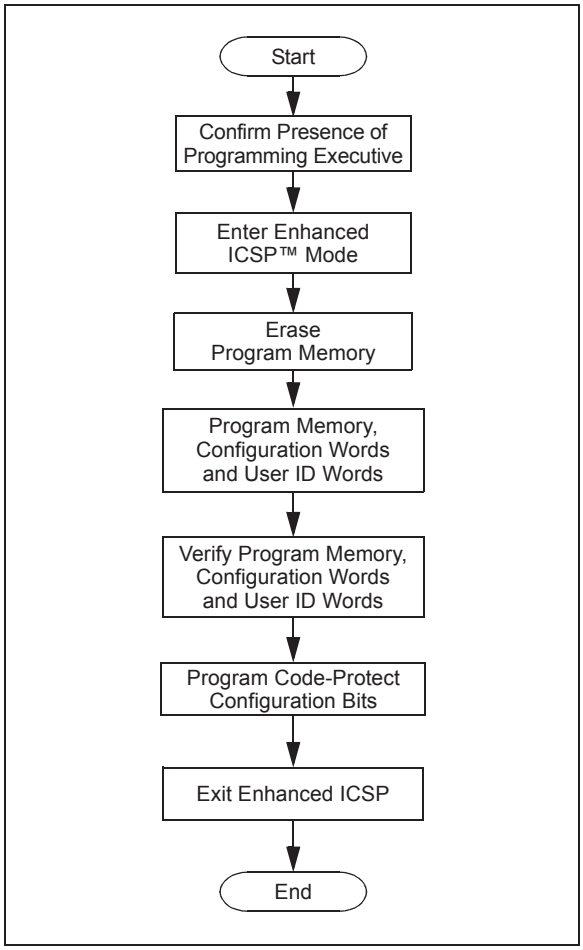
The PE performs the low-level tasks required for erasing, programming and verifying a device. This allows the programmer to program the device by issuing the appropriate commands and data. A detailed description for each command is provided in **Section 6.2 "Programming Executive Commands"**.

Note:	The PE uses the device's data RAM for variable storage and program execution. After running the PE, 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. First, it must be determined if the PE is present in executive memory, and then, Enhanced ICSP mode is entered. The program memory is then erased, and the program memory and Configuration Words are programmed and verified. Last, the code-protect Configuration bits are programmed (if required) and Enhanced ICSP mode is exited.

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



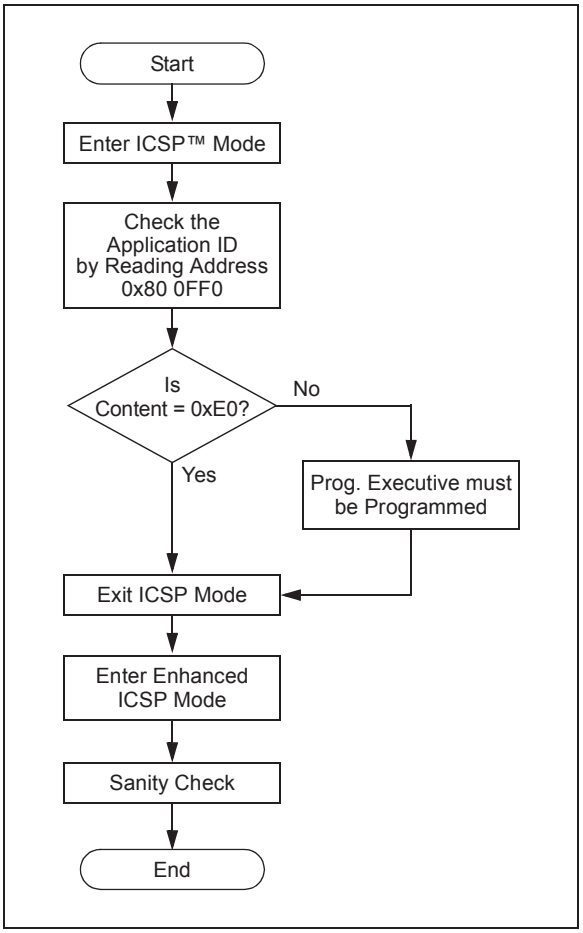
4.2 Confirming the Presence of the Programming Executive

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

First, ICSP mode is entered. Then, the unique Application ID Word, stored in executive memory, is read. If the Application ID has the value, 0xE0, the Programming Executive is resident in memory and the device can be programmed. However, if the Application ID Word is not present, the PE must be programmed to executive code memory using the method described in **Section 5.0 "Programming the Programming Executive to Memory"**.

Section 3.0 "Device Programming – ICSP" describes the ICSP programming method. **Section 4.3 "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



© 2015-2018 Microchip Technology Inc.

4.3 Reading the Application ID Word

The Application ID Word is stored at address, 0x80 0FF0, 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 4-1.

If the Application ID has the value, 0xE0, the Programming Executive is resident in memory and the device can be programmed using the mechanism described in this section. However, if the Application ID has any other value, the PE is not resident in memory; it must be loaded to memory before the device can be programmed. The procedure for loading the PE to memory is described in **Section 5.0 "Programming the Programming Executive to Memory"**.

TABLE 4-1: SERIAL INSTRUCTION EXECUTION FOR READING THE APPLICATION ID WORD

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 (W0) for the TBLRD instruction.		
0000	200800	MOV #0x80, W0
0000	8802A0	MOV W0, TBLPAG
0000	20FF00	MOV #0xFF0, W0
0000	207841	MOV #VISI, W1
0000	000000	NOP
0000	BA0890	TBLRDL [W0], [W1]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 3: Output the VISI register using the REGOUT command.		
0001	<VISI>	; Clock out the contents of the VISI register.

4.4 Entering Enhanced ICSP Mode

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

1. The MCLR pin is briefly driven high and then low.
2. A 32-bit key sequence is clocked into PGEDx.
3. MCLR is then driven high within a specified period of time and held.

The programming voltage applied to MCLR is VIH, which is essentially VDD in PIC24FJ256GA705 family 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 PGEDx.

The key sequence is a specific 32-bit pattern, '0100 1101 0100 0011 0100 1000 0101 0000' (more easily remembered as 0x4D43 4850 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, VIH must be applied to MCLR and held at that level for as long as Program/Verify mode is to be maintained. An interval time of at least P19, P7 and P1 * 5 must elapse before presenting data on PGEDx. During the P7 interval, the programmer's PGEDx and PGECx lines must be tri-stated.

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

Note:	The PGEC2/PGED2 pair on 28-pin packages does not support Enhanced ICSP, but does allow normal ICSP operation.

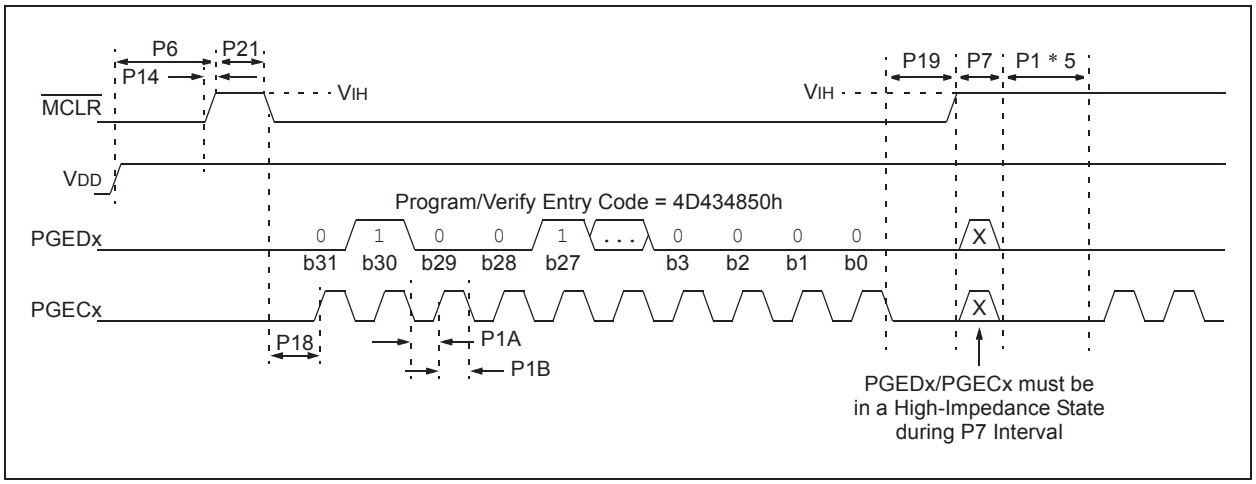
4.5 Blank Check

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 (0xFF 0000:0xFF 0002) can be ignored by the Blank Check, since this region stores device information that cannot be erased. Additionally, all unimplemented memory space should be ignored 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.

FIGURE 4-3: ENTERING ENHANCED ICSP™ MODE



© 2015-2018 Microchip Technology Inc.

4.6 Code Memory Programming

4.6.1 PROGRAMMING METHODOLOGY

There are two commands that can be used for programming code memory when utilizing the PE. The PROG2W command programs and verifies two 24-bit instruction words into the program memory, starting at the address specified. The second and faster command, PROGP, allows up to 128 instruction words (each 24 bits) to be programmed and verified into program memory, starting at the address specified. See **Section 6.0 "The Programming Executive"** for a full description for each of these commands.

Figure 4-4 and Figure 4-5 show the programming methodology for the PROG2W and PROGP commands. In both instances, 87552 instruction words of the device are programmed.

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

FIGURE 4-4: FLOWCHART FOR DOUBLE-WORD PROGRAMMING

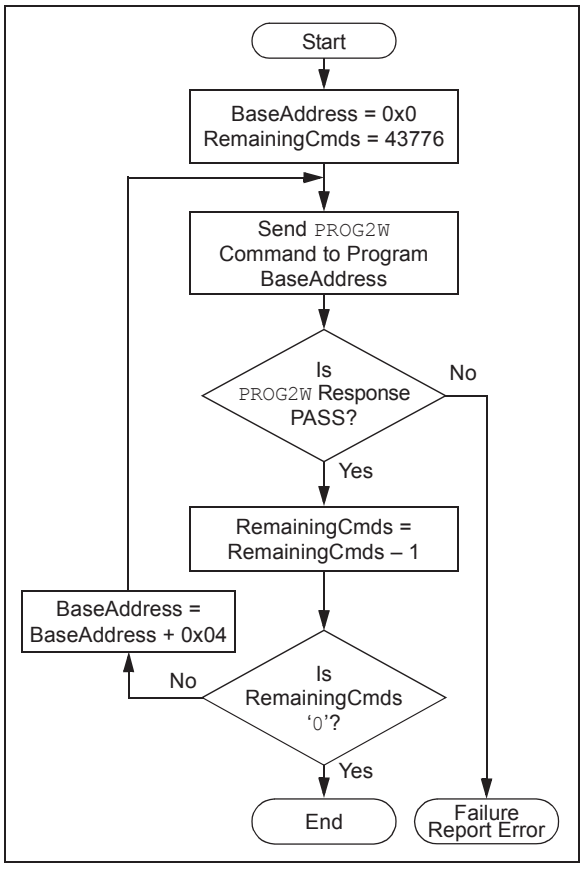
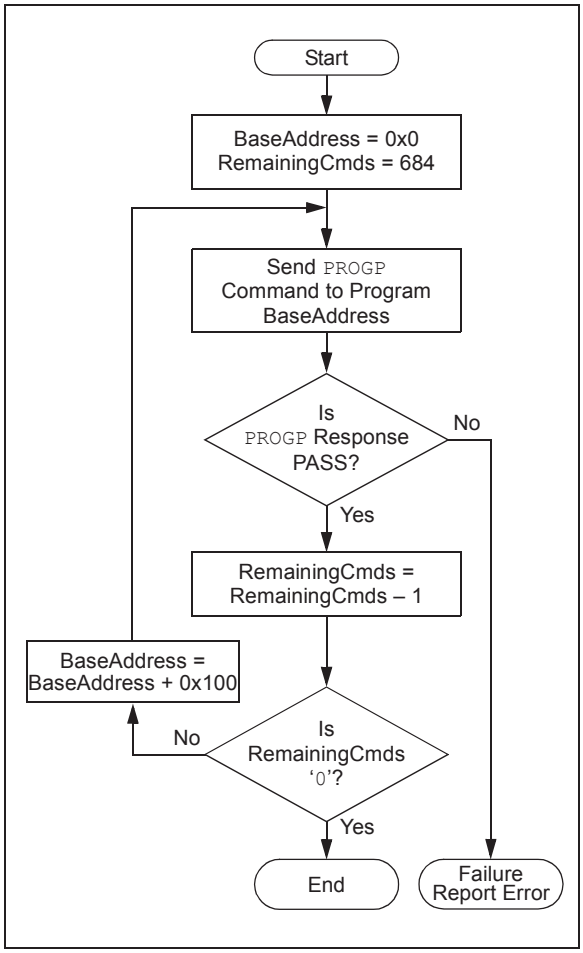


FIGURE 4-5: FLOWCHART FOR MULTIPLE WORD PROGRAMMING

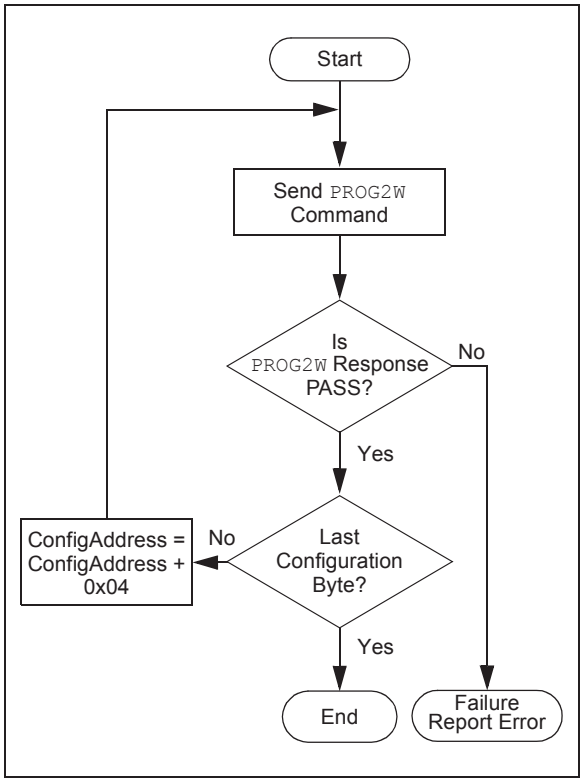


4.7 Configuration Bit Programming

Configuration bits are programmed one at a time using the PROG2W command. This command specifies the configuration data and address. When Configuration bits are programmed, any unimplemented bits must be programmed with a '1'.

Multiple PROG2W commands are required to program all Configuration bits. A flowchart for Configuration bit programming is shown in Figure 4-6.

FIGURE 4-6: CONFIGURATION BIT PROGRAMMING FLOW



4.8 Programming Verification

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

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

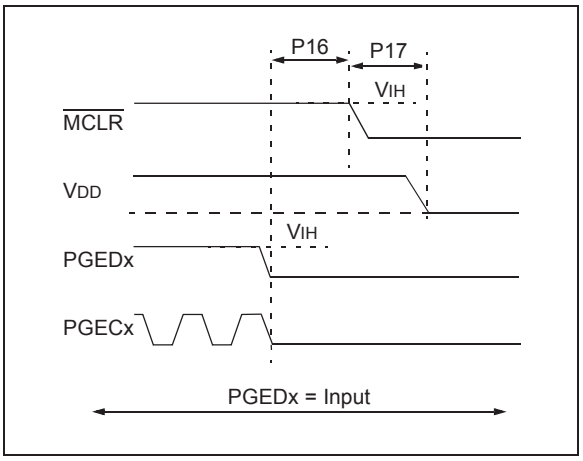
Alternatively, you can have the programmer perform the verification after the entire device is programmed using a checksum computation.

See **Section 8.0 "Checksum Computation"** for more information on calculating the checksum.

4.9 Exiting Enhanced ICSP Mode

Exiting Program/Verify mode is done by removing VIH from MCLR, as illustrated in Figure 4-7. The only requirement for exit is that an interval, P16, should elapse between the last clock, and program signals on PGECx and PGEDx, before removing VIH.

FIGURE 4-7: EXITING ENHANCED ICSP™ MODE



© 2015-2018 Microchip Technology Inc.

5.0 PROGRAMMING THE PROGRAMMING EXECUTIVE TO MEMORY

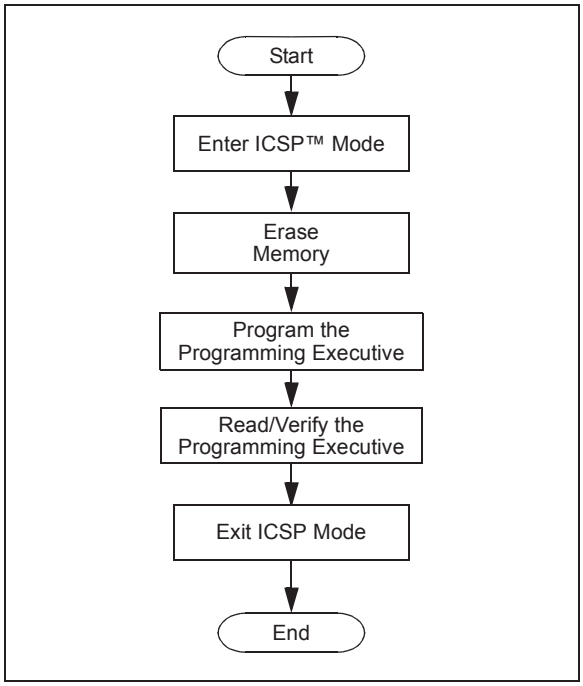
Note: The Programming Executive (PE) can be obtained from each device page on the Microchip website: www.microchip.com.

5.1 Overview

If it is determined that the PE is not present in executive memory (as described in **Section 4.2 "Confirming the Presence of the Programming Executive"**), the PE must be programmed to executive memory.

Figure 5-1 shows the high-level process of programming the PE into executive memory. First, ICSP mode must be entered, and executive memory and user memory are erased; then, the PE is programmed and verified. Finally, ICSP mode is exited.

FIGURE 5-1: HIGH-LEVEL PROGRAMMING EXECUTIVE PROGRAM FLOW



5.2 Erasing Executive Memory

Executive memory can be erased through a series of Page Erase operations, as shown in Figure 5-2. This consists of setting NVMCON to 0x4003, executing the programming cycle and repeating for the rest of the pages of executive memory.

Table 5-1 illustrates the ICSP programming process for Bulk Erasing memory.

Note: The PE must always be erased before it is programmed, as described in Figure 5-1.

FIGURE 5-2: BULK ERASE FLOW

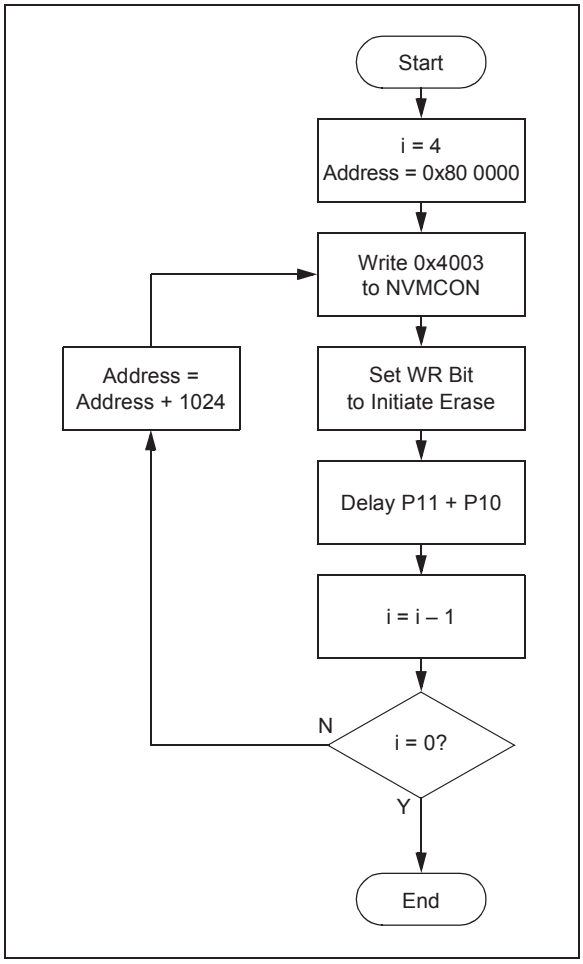


TABLE 5-1: SERIAL INSTRUCTION EXECUTION FOR ERASING EXECUTIVE MEMORY

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
Step 2: Set the NVMCON register to erase a page.		
0000	240030	MOV #0x4003, W0
0000	883B00	MOV W0, NVMCON
Step 3: Load the address of the page to be erased into the NVMADR register pair.		
0000	200004	MOV #0000, W4
0000	883B14	MOV W4, NVMADR
0000	200800	MOV #0080, W0
0000	883B20	MOV W0, NVMADRU
Step 4: Set the WR bit.		
0000	200550	MOV #0x55, W0
0000	883B30	MOV W0, NVMKEY
0000	200AA0	MOV #0xAA, W0
0000	883B30	MOV W0, NVMKEY
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 5: Repeat this step to poll the WR bit until it is cleared by 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 the contents of the VISI register.
0000	000000	NOP
Step 6: Increment W4 by 1024 (0x400).		
0000	204003	MOV #400, W3
0000	418204	ADD W3, W4, W4
0000	883B14	MOV W4, NVMADR
Step 7: Repeat Steps 4-6 until the entire test memory has been erased.		
Step 8: Clear the WREN bit.		
0000	200000	MOV #0000, W0
0000	883B00	MOV W0, NVMCON

© 2015-2018 Microchip Technology Inc.

5.3 Program the Programming Executive

Storing the PE to executive memory is similar to normal programming of code memory. The executive memory must first be erased and then programmed, using either two-word writes (two instruction words) or row writes (128 instruction words). The control flow for both methods is identical to that for programming code memory, as shown in Figure 3-7.

Table 5-2 and Table 5-3 illustrate the ICSP programming processes for PE memory. To minimize programming time, the same packed data format that the PE uses is utilized. See **Section 6.2 "Programming Executive Commands"** for more details on the packed data format.

TABLE 5-2: PROGRAMMING THE PROGRAMMING EXECUTIVE (TWO-WORD LATCH WRITES)

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 TBLPAG register for writing to the latches.		
0000	200FAC	MOV #0xFA, W12
0000	8802AC	MOV W12, TBLPAG
Step 3: Load W0:W2 with the next two packed instruction words to program.		
0000	2xxxx0	MOV #<LSW0>, W0
0000	2xxxx1	MOV #<MSB1:MSB0>, W1
0000	2xxxx2	MOV #<LSW1>, W2
Step 4: Set the Read Pointer (W6) and the Write Pointer (W7), and load the write latches.		
0000	EB0300	CLR W6
0000	000000	NOP
0000	EB0380	CLR W7
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.W [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
Step 5: Set the NVMADRU/NVMADR register pair to point to the correct row.		
0000	2xxxx3	MOV #DestinationAddress<15:0>, W3
0000	2xxxx4	MOV #DestinationAddress<23:16>, W4
0000	883B13	MOV W3, NVMADR
0000	883B24	MOV W4, NVMADRU
Step 6: Set the NVMCON register to program two instruction words.		
0000	24001A	MOV #0x4001, W10
0000	000000	NOP
0000	883B0A	MOV W10, NVMCON
0000	000000	NOP
0000	000000	NOP

TABLE 5-2: PROGRAMMING THE PROGRAMMING EXECUTIVE (TWO-WORD LATCH WRITES) (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 7: Initiate the write cycle.		
0000	200551	MOV #0x55, W1
0000	883B31	MOV W1, NVMKEY
0000	200AA1	MOV #0xAA, W1
0000	883B31	MOV W1, NVMKEY
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 8: Wait for program operation to complete and make sure the WR bit is clear.		
0000	000000	NOP
0000	803B00	MOV NVMCON, W0
0000	000000	NOP
0000	883C20	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	; Clock out the contents of the VISI register.
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
—	—	; Repeat until the WR bit is clear.
Step 9: Repeat Steps 3-8 until all code memory is programmed.		
Step 10: Clear the WREN bit.		
0000	200000	MOV #0000, W0
0000	883B00	MOV W0, NVMCON

© 2015-2018 Microchip Technology Inc.

TABLE 5-3: PROGRAMMING THE PROGRAMMING EXECUTIVE (ROW WRITES)

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 register to program 128 instruction words.		
0000	240020	MOV #0X4002, W0
0000	883B00	MOV W0, NVMCON
Step 3: Initialize the TBLPAG register for writing to the latches.		
0000	200FAC	MOV #0xFA, W12
0000	8802AC	MOV W12, TBLPAG
Step 4: Load W0:W5 with the next four 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
Step 5: Set the Read Pointer (W6) and load the (next set of) write latches.		
0000	EB0300	CLR W6
0000	000000	NOP
0000	EB0380	CLR W7
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, for a total of 32 times, to load the write latches with 128 instructions.		

TABLE 5-3: PROGRAMMING THE PROGRAMMING EXECUTIVE (ROW WRITES) (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 7: Set the NVMADRU/NVMADR register pair to point to the correct address.		
0000	2xxxx3	MOV #DestinationAddress<15:0>, W3
0000	2xxxx4	MOV #DestinationAddress<23:16>, W4
0000	883B13	MOV W3, NVMADR
0000	883B24	MOV W4, NVMADRU
Step 8: Execute the WR bit unlock sequence and initiate the write cycle.		
0000	200550	MOV #0x55, W0
0000	883B30	MOV W0, NVMKEY
0000	200AA0	MOV #0xAA, W0
0000	883B30	MOV W0, NVMKEY
0000	A8E761	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
Step 9: Repeat this step to poll the WR bit until it is cleared by hardware.		
0000	040200	GOTO 0x200
0000	000000	NOP
0000	803B00	MOV NVMCON, W2
0000	A8E761	MOV W2, VISI
0000	000000	NOP
0001	<VISI>	; Clock out the contents of the VISI register.
0000	000000	NOP
Step 10: Reset the device's internal Program Counter.		
0000	040200	GOTO 0x200
0000	000000	NOP
Step 11: Repeat Steps 3 through 9 until all code memory is programmed.		
Step 12: Clear the WREN bit.		
0000	200000	MOV #0000, W0
0000	883B00	MOV W0, NVMCON

© 2015-2018 Microchip Technology Inc.

5.4 Reading Executive Memory

Reading from executive memory is performed by executing a series of TBLRD instructions and clocking out the data using the REGOUT command.

Table 5-4 shows the ICSP programming details for reading executive memory.

To minimize reading time, the same packed data format that the PE uses is utilized. See **Section 6.2 "Programming Executive Commands"** for more details on the packed data format.

TABLE 5-4: SERIAL EXECUTION FOR 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 the TBLPAG register and the Read Pointer (W6) for the TBLRD instruction.		
0000	200xx0	MOV #<SourceAddress23:16>, W0
0000	8802A0	MOV W0, TBLPAG
0000	2xxxx6	MOV #<SourceAddress15:0>, W6
Step 3: Initialize the Write Pointer (W7) and store the next four locations of code memory in W0:W5.		
0000	EB0380	CLR W7
0000	000000	NOP
0000	BA1B96	TBLRDL [W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BADBB6	TBLRDH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BADBD6	TBLRDH.B [++W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BA1BB6	TBLRDL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BA1B96	TBLRDL [W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BADBB6	TBLRDH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BADBD6	TBLRDH.B [++W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BA0BB6	TBLRDL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP

TABLE 5-4: SERIAL EXECUTION FOR READING EXECUTIVE MEMORY (CONTINUED)

Command (Binary)	Data (Hex)	Description
Step 4: Output W0:W5 using the VISI register and the REGOUT command.		
0000	883C20	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	; Clock out the contents of the VISI register.
0000	000000	NOP
0000	883C21	MOV W1, VISI
0000	000000	NOP
0001	<VISI>	; Clock out the contents of the VISI register.
0000	000000	NOP
0000	883C22	MOV W2, VISI
0000	000000	NOP
0001	<VISI>	; Clock out the contents of the VISI register.
0000	000000	NOP
0000	883C23	MOV W3, VISI
0000	000000	NOP
0001	<VISI>	; Clock out the contents of the VISI register.
0000	000000	NOP
0000	883C24	MOV W4, VISI
0000	000000	NOP
0001	<VISI>	; Clock out the contents of the VISI register.
0000	000000	NOP
0000	883C25	MOV W5, VISI
0000	000000	NOP
0001	<VISI>	; Clock out the contents of the VISI register.
0000	000000	NOP
Step 5: Reset the device's internal Program Counter.		
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
Step 6: Repeat Steps 3 through 5 until all desired code memory is read (note that "Reset the device's internal Program Counter" will be Step 5).		

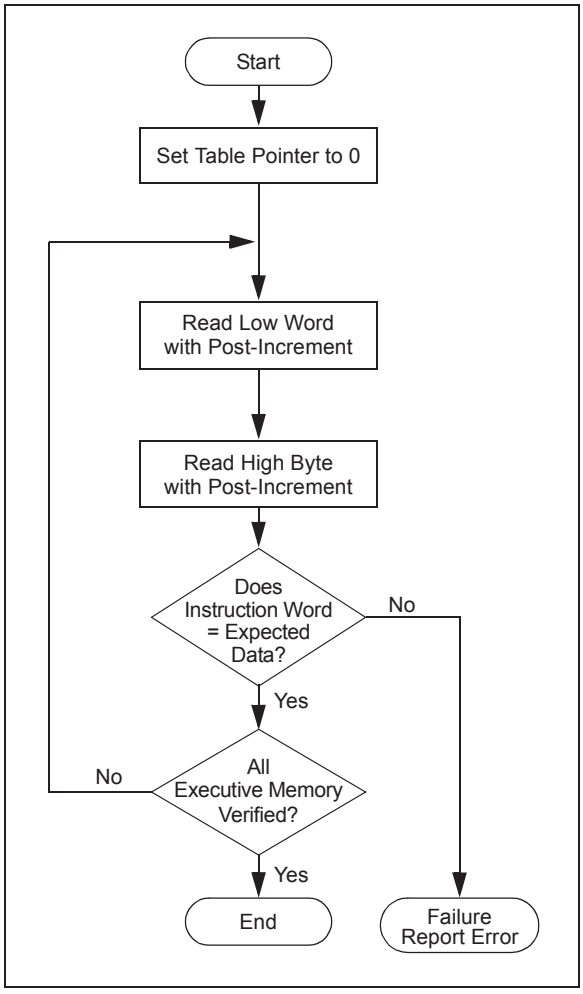
© 2015-2018 Microchip Technology Inc.

5.5 Verify Programming Executive

The verify step involves reading back the executive memory space and comparing it against the copy held in the programmer's buffer.

The verify process is illustrated in Figure 5-3. The lower word of the instruction is read, and then the lower byte of the upper word is read and compared against the instruction stored in the programmer's buffer. Refer to **Section 5.4 "Reading Executive Memory"** for implementation details of reading executive memory.

FIGURE 5-3: VERIFY PROGRAMMING EXECUTIVE MEMORY FLOW



6.0 THE PROGRAMMING EXECUTIVE

6.1 Programming Executive Communication

The programmer and PE have a master-slave relationship, where the programmer is the master programming device and the PE 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 PE. In turn, the PE only sends one response to the programmer after receiving and processing a command. The PE command set is described in **Section 6.2 "Programming Executive Commands"**. The response set is described in **Section 6.3 "Programming Executive Responses"**.

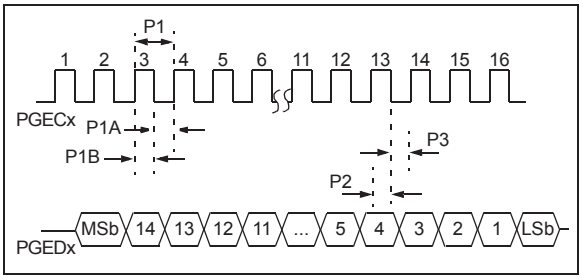
6.1.1 COMMUNICATION INTERFACE AND PROTOCOL

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

Note:	For Enhanced ICSP, all serial data is transmitted on the falling edge of PGECx and latched on the rising edge of PGECx. All data transmissions are sent to the MSb first, using 16-bit mode (see Figure 6-1).

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

FIGURE 6-1: PROGRAMMING EXECUTIVE SERIAL TIMING



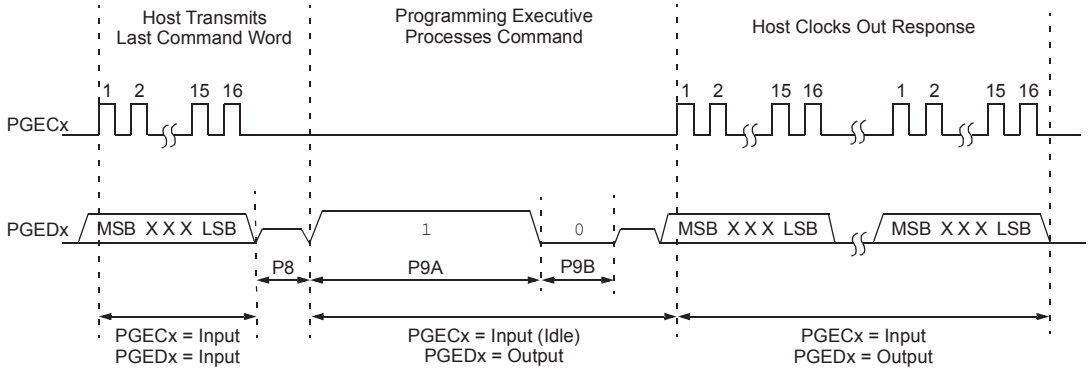
After the PE has processed the command, it brings PGEDx low (P9B) to indicate to the programmer that the response is available to be clocked out. The programmer can begin to clock out the response after a maximum wait (P9B) and it must provide the necessary amount of clock pulses to receive the entire response from the PE.

After the entire response is clocked out, the programmer should terminate the clock on PGECx until it is time to send another command to the PE. This protocol is illustrated in Figure 6-2.

6.1.2 SPI RATE

In Enhanced ICSP mode, the PIC24FJ256GA705 family devices operate from the Fast Internal RC (FRC) Oscillator, 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 2 MHz clock be provided by the programmer.

FIGURE 6-2: PROGRAMMING EXECUTIVE – PROGRAMMER COMMUNICATION PROTOCOL



Note 1: See Table 9-1 for timing information.

© 2015-2018 Microchip Technology Inc.

6.1.3 TIME-OUTS

The PE uses no Watchdog Timer or time-out for transmitting responses to the programmer. If the programmer does not follow the flow control mechanism using PGECx, as described in **Section 6.1.1 "Communication Interface and Protocol"**, it is possible that the PE will behave unexpectedly while trying to send a response to the programmer. Since the PE 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 6-1. If the command time-out expires, the programmer should reset the PE and start programming the device again.

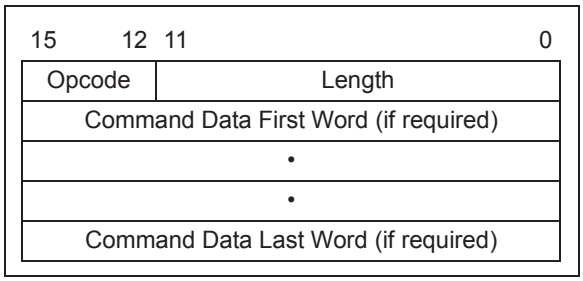
6.2 Programming Executive Commands

The PE command set is shown in Table 6-1. This table contains the opcode, mnemonic, length, time-out and description for each command. Functional details on each command are provided in the command descriptions (**Section 6.2.4 "Command Descriptions"**).

6.2.1 COMMAND FORMAT

All PE commands have a general format, consisting of a 16-bit header and any required data for the command (see Figure 6-3). The 16-bit header consists of a 4-bit opcode field, which is used to identify the command, followed by a 12-bit command length field.

FIGURE 6-3: COMMAND FORMAT



The command opcode must match one of those in the command set. Any command that is received, which does not match the list in Table 6-1, will return a "NACK" response (see **Section 6.3.1.1 "Opcode Field"**).

The command length is represented in 16-bit words since the SPI operates in 16-bit mode. The PE 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 PE.

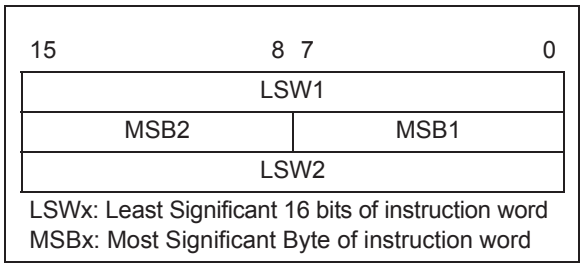
TABLE 6-1: PROGRAMMING EXECUTIVE COMMAND SET

Opcode	Mnemonic	Length (16-bit words)	Time-out	Description
0x0	SCHECK	1	1 ms	Sanity check.
0x1	READC	3	1 ms	Read an 8-bit word from the specified Configuration register or Device ID register.
0x2	READP	4	1 ms/row	Read 'N' 24-bit instruction words of primary Flash memory, starting from the specified address.
0x3	PROG2W	6	5 ms	Program a double instruction word of code memory at the specified address and verify.
0x4	Reserved	N/A	N/A	This command is reserved; it will return a NACK.
0x5	PROGP	99	5 ms	Program 128 words of program memory at the specified starting address, then verify.
0x6	Reserved	N/A	N/A	This command is reserved; it will return a NACK.
0x7	ERASEB	1	125 ms	Chip Erase the device.
0x8	Reserved	N/A	N/A	This command is reserved; it will return a NACK.
0x9	ERASEP	3	25 ms	Command to erase a page.
0xA	Reserved	N/A	N/A	This command is reserved; it will return a NACK.
0xB	QVER	1	1 ms	Query the PE software version.
0xC	CRCP	5	1s	Perform a CRC-16 on the specified range of memory.
0xD	Reserved	N/A	N/A	This command is reserved; it will return a NACK.
0xE	QBLANK	5	700 ms	Query to check whether the code memory is blank.

6.2.2 PACKED DATA FORMAT

When 24-bit instruction words are transferred across the 16-bit SPI interface, they are packed to conserve space using the format illustrated in Figure 6-4. This format minimizes traffic over the SPI and provides the PE with data that is properly aligned for performing Table Write operations.

FIGURE 6-4: PACKED INSTRUCTION WORD FORMAT



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

6.2.3 PROGRAMMING EXECUTIVE ERROR HANDLING

The PE will “NACK” all unsupported commands. Additionally, due to the memory constraints of the PE, no checking is performed on the data contained in the programmer command. It is the responsibility of the programmer to command the PE with valid command arguments or the programming operation may fail. Additional information on error handling is provided in **Section 6.3.1.3 “QE_Code Field”**.

6.2.4 COMMAND DESCRIPTIONS

All commands supported by the PE are described in **Section 6.2.4.1 “SCHECK Command”** through **Section 6.2.4.10 “QBLANK Command”**.

6.2.4.1 SCHECK Command

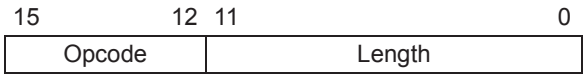


Table 6-2 shows the description for the SCHECK command.

TABLE 6-2: COMMAND DESCRIPTION

Field	Description
Opcode	0x0
Length	0x1

The SCHECK command instructs the PE to do nothing but generate a response. This command is used as a “Sanity Check” to verify that the PE is operational.

Expected Response (2 words):

0x1000 0x0002

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

© 2015-2018 Microchip Technology Inc.

6.2.4.2 READC Command

15 12	11 8	7 0
Opcode	Length	
N		Addr_MSB
Addr_LS		

Table 6-3 shows the description for the READC command.

TABLE 6-3: COMMAND DESCRIPTION

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

The READC command instructs the PE to read N Configuration registers or Device ID registers, starting from the 24-bit address specified by Addr_MSB and Addr_LS. This command can only be used to read 8-bit or 16-bit data.

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

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

0x1100

2 + N

Configuration Register or Device ID Register 1

...

Configuration Register or Device ID Register N

Note: Reading unimplemented memory will cause the PE to reset. To prevent this from occurring, ensure that only memory locations present on a particular device are accessed.

6.2.4.3 READP Command

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

Table 6-4 shows the description for the READP command.

TABLE 6-4: COMMAND DESCRIPTION

Field	Description
Opcode	0x2
Length	0x4
N	Number of 24-bit instructions to read (maximum of 32768)
Reserved	0x0
Addr_MSB	MSB of 24-bit source address
Addr_LS	Least Significant 16 bits of 24-bit source address

The READP command instructs the PE to read N 24-bit words of code memory, starting from the 24-bit address specified by Addr_MSB and Addr_LS. This command can only be used to read 24-bit data. All data returned in response to this command uses the packed data format described in **Section 6.2.2 "Packed Data Format"**.

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

0x1200

2 + 3 * N/2

Least Significant Program Memory Word 1

...

Least Significant Data Word N

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

0x1200

4 + 3 * (N – 1)/2

Least Significant Program Memory Word 1

...

MSB of Program Memory Word N (zero-padded)

Note: Reading unimplemented memory will cause the PE to reset. To prevent this from occurring, ensure that only memory locations present on a particular device are accessed.

6.2.4.4 PROG2W Command

15 12	11 8	7 0
Opcode	Length	
Reserved		Addr_MSB
Addr_LS		
DataL_LS		
DataH_MSB		DataL_MSB
DataH_LS		

Table 6-5 shows the description for the PROG2W command.

TABLE 6-5: COMMAND DESCRIPTION

Field	Description
Opcode	0x3
Length	0x6
DataL_MSB	MSB of 24-bit data for low instruction word
DataH_MSB	MSB of 24-bit data for high instruction word
Addr_MSB	MSB of 24-bit destination address
Addr_LS	Least Significant 16 bits of 24-bit destination address
DataL_LS	Least Significant 16 bits of 24-bit data for low instruction word
DataH_LS	Least Significant 16 bits of 24-bit data for high instruction word

The PROG2W command instructs the PE to program two instruction words of code memory (6 bytes) to the specified memory address.

After the words have been programmed to code memory, the PE verifies the programmed data against the data in the command.

Expected Response (2 words):

0x1300

0x0002

6.2.4.5 PROGP Command

15 12	11 8	7 0
Opcode	Length	
Reserved		Addr_MSB
Addr_LS		
D_1		
D_2		
...		
D_N		

Table 6-6 shows the description for the PROGP command.

TABLE 6-6: COMMAND DESCRIPTION

Field	Description
Opcode	0x5
Length	0x63
Reserved	0x0
Addr_MSB	MSB of 24-bit destination address
Addr_LS	Least Significant 16 bits of 24-bit destination address
D_1	16-bit Data Word 1
D_2	16-bit Data Word 2
...	16-bit Data Word 3 through 95
D_96	16-bit Data Word 96

The PROGP command instructs the PE to program one row of code memory (128 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 0x100.

The data to program the memory, located in command words, D_1 through D_96, must be arranged using the packed instruction word format illustrated in Figure 6-4.

After all data has been programmed to code memory, the PE verifies the programmed data against the data in the command.

Expected Response (2 words):

0x1500

0x0002

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

© 2015-2018 Microchip Technology Inc.

6.2.4.6 ERASEB Command

15 12	11 8	7 0
Opcode		Length

Table 6-7 shows the description for the ERASEB command.

TABLE 6-7: COMMAND DESCRIPTION

Field	Description
Opcode	0x7
Length	0x1

The ERASEB command instructs the PE to perform a Chip Erase (i.e., erase all of the primary Flash memory and code-protect bits).

Expected Response (2 words):

0x1700

0x0002

6.2.4.7 ERASEP Command

15 12	11 8	7 0
Opcode	Length	
NUM_PAGES	Addr_MSB	
Addr_LS		

Table 6-8 shows the description for the ERASEP command.

TABLE 6-8: COMMAND DESCRIPTION

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

The ERASEP command instructs the PE to Page Erase [NUM_PAGES] of code memory. The code memory must be erased at an “even” 1024 instruction words address boundary.

Expected Response (2 words):

0x1900

0x0002

6.2.4.8 QVER Command

15 12	11 0
Opcode	Length

Table 6-9 shows the description for the QVER command.

TABLE 6-9: COMMAND DESCRIPTION

Field	Description
Opcode	0xB
Length	0x1

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

Expected Response (2 words):

0x1BMN (where “MN” stands for version M.N)

0x0002

6.2.4.9 CRCP Command

15 12	11 8	7 0
Opcode	Length	
Reserved	Addr_MSB	
Addr_LSW		
Reserved	Size_MSB	
Size_LSW		

Table 6-10 shows the description for the CRCP command.

TABLE 6-10: COMMAND DESCRIPTION

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

The CRCP command performs a CRC-16 on the range of memory specified. This command can substitute for a full chip verify. Data is shifted in a packed method, as demonstrated in Figure 6-4, byte-wise, Least Significant Byte (LSB) first.

Example:

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

Expected Response (3 words):

QE_Code: 0x1C00

Length: 0x0003

CRC Value: 0xXXXX

6.2.4.10 QBLANK Command

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

Table 6-11 shows the description for the QBLANK command.

TABLE 6-11: COMMAND DESCRIPTION

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

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

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

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

Expected Response (2 words for blank device):

0x1DF0

0x0002

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

0x1D0F

0x0002

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

© 2015-2018 Microchip Technology Inc.

6.3 Programming Executive Responses

The PE 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 PE response set is shown in Table 6-12. This table contains the opcode, mnemonic and description for each response. The response format is described in **Section 6.3.1 "Response Format"**.

TABLE 6-12: PROGRAMMING EXECUTIVE RESPONSE OPCODES

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

6.3.1 RESPONSE FORMAT

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

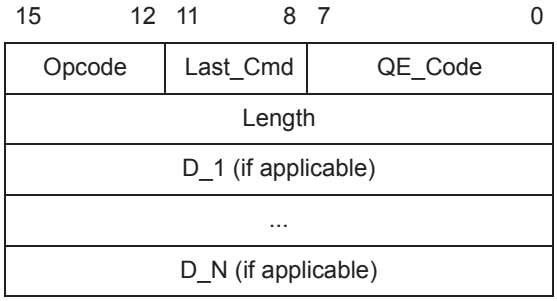


Table 6-13 shows the description of the response format.

TABLE 6-13: RESPONSE FORMAT DESCRIPTION

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).

6.3.1.1 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 6-12). 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 PE is not identified, the PE returns a NACK response.

6.3.1.2 Last_Cmd Field

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

6.3.1.3 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 PE 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 6-14.

TABLE 6-14: QE_Code FOR QUERIES

Query	QE_Code
QBLANK	0x0F = Code memory is NOT blank 0xF0 = Code memory is blank
QVER	0xMN, where PE Software Version = M.N (i.e., 0x32 means Software Version 3.2)

When the PE processes any command other than a query, the QE_Code represents an error code. Supported error codes are shown in Table 6-15. If a command is successfully processed, the returned QE_Code is set to 0x0, which indicates that there is no error in the command processing. If the verify of the programming for the PROGW command fails, the QE_Code is set to 0x1. For all other PE errors, the QE_Code is set to 0x2.

TABLE 6-15: QE_Code FOR NON-QUERY COMMANDS

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

6.3.1.4 Response Length

The response length indicates the length of the PE's response in 16-bit words. This field includes the two words of the response header.

With the exception of the response for the read commands, the length of each response is only two words.

The response to the READP commands uses the packed instruction word format described in **Section 6.2.2 "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.

© 2015-2018 Microchip Technology Inc.

7.0 DEVICE ID

The Device ID region of memory can be used to determine variant and manufacturing information about the chip. This region of memory is read-only and can be read when code protection is enabled. The DEVID register (0xFF 0000) identifies the specific part number of the device, while DEVREV (0xFF 0002) shows the silicon revision level.

Table 7-1 lists the identification information for each device. Table 7-2 shows the Device ID registers and Table 7-3 describes the bit field of each register.

TABLE 7-1: DEVICE IDs

Device	DEVID
PIC24FJ64GA702	0x7506
PIC24FJ128GA702	0x750A
PIC24FJ256GA702	0x750E
PIC24FJ64GA704	0x7505
PIC24FJ128GA704	0x7509
PIC24FJ256GA704	0x750D
PIC24FJ64GA705	0x7507
PIC24FJ128GA705	0x750B
PIC24FJ256GA705	0x750F

TABLE 7-2: PIC24FJ256GA705 FAMILY DEVICE ID REGISTERS

Address	Name	Bit															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF 0000	DEVID	FAMID<7:0>								DEV<7:0>							
0xFF 0002	DEVREV	—												REV<3:0>			

TABLE 7-3: DEVICE ID BIT FIELD 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.

7.1 Unique Device Identifier (UDID)

All PIC24FJ256GA705 family devices are individually encoded during final manufacturing with a Unique Device Identifier or UDID. The UDID cannot be erased by a Bulk Erase command or any other user-accessible means. This feature allows for manufacturing traceability of Microchip Technology devices in applications where this is a requirement. It may also be used by the application manufacturer for any number of things that may require unique identification, such as:

- Tracking the device
- Unique serial number
- Unique security key

The UDID comprises five 24-bit program words. When taken together, these fields form a unique 120-bit identifier.

The UDID is stored in five read-only locations, located between 0x80 1600 and 0x80 1608 in the device configuration space. Table 7-4 lists the addresses of the identifier words and shows their contents.

TABLE 7-4: UDID ADDRESSES

UDID	Address	Description
UDID1	0x80 1600	UDID Word 1
UDID2	0x80 1602	UDID Word 2
UDID3	0x80 1604	UDID Word 3
UDID4	0x80 1606	UDID Word 4
UDID5	0x80 1608	UDID Word 5

© 2015-2018 Microchip Technology Inc.

8.0 CHECKSUM COMPUTATION

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

- Contents of code memory locations
- Contents of Configuration Words

All memory locations, including Configuration Words, are summed by adding all three bytes of each memory address. The checksum is computed “bytewise”, with the final result truncated to 16 bits. In the dsPIC33 architecture, each Flash memory address contains two bytes (if an even address) or one byte (if an odd address, since the upper byte is implemented and is always 0x00). When computing the checksum, both the upper and lower bytes of the word at a given address should be added to the running sum, separately as bytes, instead of as a single 16-bit word.

For example, in a program that contains two words with contents such as:

Contents at address 0x0000 = 0xABCD

Contents at address 0x0001 = 0x00EF

The checksum over the 0x0000:0x0001 region is: 0xCD + 0xAB + 0xEF + 0x00 = 0x0267.

The CFGB block checksum is also a “bytewise” sum of all contents of the configuration block address region, and is computed identically to the PROG region, with the exception that some Configuration registers may require certain bits to be AND masked out and excluded during the checksum computation.

TABLE 8-1: CONFIGURATION BIT MASKS

Devices	Configuration Bit Mask	
	FSIGN	FICD
PIC24FJ256GA705	0xFF 7FFF	0xFF FFDF

TABLE 8-2: CHECKSUM COMPUTATION

Device	Read Code Protection	Checksum Computation	Erased Value	Value with 0xAA AAAA at 0x0 and Last Code Address
PIC24FJ256GA705	Disabled	PROG[0x00:0x02 AEFF] + CFGB[0x02 AF00:0x02 AFFF]	0xF760(1)	0xF562(1)
	Enabled	0	0x0000	0x0000
PIC24FJ128GA705	Disabled	PROG[0x00:0x01 5EFF] + CFGB[0x01 5F00:0x01 5FFF]	0xEF60(1)	0xED62(1)
	Enabled	0	0x0000	0x0000
PIC24FJ64GA705	Disabled	PROG[0x00:0xAEFF] + CFGB[0xAF00:0xAFFF]	0xF760(1)	0xF562(1)
	Enabled	0	0x0000	0x0000

Legend: PROG[a:b] = Program memory byte sum of locations, a to b inclusive (all 3 bytes of code memory)
CFGB[c:d] = Configuration memory byte sum of locations, c to d inclusive (all 3 bytes of code memory)

Note 1: For the checksum computation example, the Configuration bits are set to the default configuration values after erasing the part.

© 2015-2018 Microchip Technology Inc.

9.0 AC/DC CHARACTERISTICS AND TIMING REQUIREMENTS

Table 9-1 lists the AC/DC characteristics and timing requirements.

TABLE 9-1: AC/DC CHARACTERISTICS AND TIMING REQUIREMENTS

Standard Operating Conditions Operating Temperature: -40ºC to +85ºC. Programming at +25ºC is recommended.						
Param. No.	**Symbol**	**Characteristic**	**Min.**	**Max.**	**Units**	**Conditions**
D111	VDD	Supply Voltage During Programming	2.0	3.6	V	See **Notes 1** and **2**
D113	IDDP	Supply Current During Programming	—	8	mA	See **Note 2**
D114	IPEAK	Instantaneous Peak Current During Start-up	—	—	mA	See **Note 2**
D031	VIL	Input Low Voltage	—	0.2 VDD	V	See **Note 2**
D041	VIH	Input High Voltage	0.8 VDD	VDD	V	See **Note 2**
D080	VOL	Output Low Voltage	—	0.8	V	See **Note 2**
D090	VOH	Output High Voltage	0.8 VDD	VDD	V	See **Note 2**
D012	CIO	Capacitive Loading on I/O Pin (PGEDx)	—	50	pF	See **Note 2**
P1	TPGC	Serial Clock (PGECx) Period (ICSP™)	200	—	ns	
P1	TPGC	Serial Clock (PGECx) Period (Enhanced ICSP)	500	—	ns	
P1A	TPGCL	Serial Clock (PGECx) Low Time (ICSP)	80	—	ns	
P1A	TPGCL	Serial Clock (PGECx) Low Time (Enhanced ICSP)	200	—	ns	
P1B	TPGCH	Serial Clock (PGECx) High Time (ICSP)	80	—	ns	
P1B	TPGCH	Serial Clock (PGECx) High Time (Enhanced ICSP)	200	—	ns	
P2	TSET1	Input Data Setup Time to Serial Clock ↓	15	—	ns	
P3	THLD1	Input Data Hold Time from PGECx ↓	15	—	ns	
P4	TDLY1	Delay Between 4-Bit Command and Command Operand	40	—	ns	
P4A	TDLY1A	Delay Between Command Operand and Next 4-Bit Command	40	—	ns	
P5	TDLY2	Delay Between Last PGECx ↓ of Command to First PGECx ↑ 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 $\overline{\text{MCLR}}$ ↑	50	—	ms	
P8	TDLY3	Delay Between Last PGECx ↓ of Command Byte to PGEDx ↑ by PE	12	—	µs	

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

2: Time depends on the FRC accuracy and the value of the FRC Oscillator Tuning register. Refer to the **"Electrical Characteristics"** chapter in the specific device data sheet.

3: This time applies to Program Memory Words, Configuration Words and User ID Words.

TABLE 9-1: AC/DC CHARACTERISTICS AND TIMING REQUIREMENTS (CONTINUED)

Standard Operating Conditions Operating Temperature: -40ºC to +85ºC. Programming at +25ºC is recommended.						
Param. No.	**Symbol**	**Characteristic**	**Min.**	**Max.**	**Units**	**Conditions**
P9A	TDLY4	PE Command Processing Time	10	—	µs	
P9B	TDLY5	Delay Between PGEDx ↓ by PE to PGEDx Released by PE	15	23	µs	
P10	TDLY6	PGECx Low Time After Programming	400	—	ns	
P11	TDLY7	Chip Erase Time	16	20	ms	
P12	TDLY8	Page Erase Time	16	20	ms	See **Note 2**
P13	TDLY9	Double-Word Programming Time	16	20	µs	See **Notes 2** and **3**
P14	TR	$\overline{\text{MCLR}}$ Rise Time to Enter ICSP mode	—	1.0	µs	
P15	TVALID	Data Out Valid from PGECx ↑	10	—	ns	
P16	TDLY10	Delay Between Last PGECx ↓ 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 PGECx ↑ for Key Sequence on PGEDx	1	—	ms	
P19	TKEY2	Delay from Last PGECx ↓ for Key Sequence on PGEDx to Second $\overline{\text{MCLR}}$ ↑	25	—	ns	
P21	TMCLRH	$\overline{\text{MCLR}}$ High Time	—	500	µs	

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

2: Time depends on the FRC accuracy and the value of the FRC Oscillator Tuning register. Refer to the **"Electrical Characteristics"** chapter in the specific device data sheet.

3: This time applies to Program Memory Words, Configuration Words and User ID Words.

© 2015-2018 Microchip Technology Inc.

Software License Agreement

The software supplied herewith by Microchip Technology Incorporated (the “Company”) is intended and supplied to you, the Company’s customer, for use solely and exclusively with products manufactured by the Company.

The software is owned by the Company and/or its supplier, and is protected under applicable copyright laws. All rights are reserved. Any use in violation of the foregoing restrictions may subject the user to criminal sanctions under applicable laws, as well as to civil liability for the breach of the terms and conditions of this license.

THIS SOFTWARE IS PROVIDED IN AN “AS IS” CONDITION. NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.

APPENDIX A: SPI PORT SCANNING SOFTWARE ROUTINE

EXAMPLE 9-1: SPI PORT SCANNING SOFTWARE CODE EXAMPLE

```
/*******************************************************************************
 * © 2016 Microchip Technology Inc.
 *
 * FileName: spi_search.s
 * Dependencies: none
 * Processor: PIC24
 * Compiler: XC16
 * IDE: MPLAB® X
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * Description: This file performs a search of the correct
 * PGEC/PGED port and connect SPI2 slave to it.
 *
 *******************************************************************************/
/*******************************************************************************
 * MICROCHIP SOFTWARE NOTICE AND DISCLAIMER: You may use this software, and
 * any derivatives created by any person or entity by or on your behalf,
 * exclusively with Microchip's products in accordance with applicable
 * software license terms and conditions, a copy of which is provided for
 * your reference in accompanying documentation. Microchip and its licensors
 * retain all ownership and intellectual property rights in the
 * accompanying software and in all derivatives hereto.
 *
 * This software and any accompanying information is for suggestion only.
 * It does not modify Microchip's standard warranty for its products. You
 * agree that you are solely responsible for testing the software and
 * determining its suitability. Microchip has no obligation to modify,
 * test, certify, or support the software.
 *
 * THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
 * EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED
 * WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
 * PARTICULAR PURPOSE APPLY TO THIS SOFTWARE, ITS INTERACTION WITH
 * MICROCHIP'S PRODUCTS, COMBINATION WITH ANY OTHER PRODUCTS, OR USE IN ANY
 * APPLICATION.
 *
 * IN NO EVENT, WILL MICROCHIP BE LIABLE, WHETHER IN CONTRACT, WARRANTY,
 * TORT (INCLUDING NEGLIGENCE OR BREACH OF STATUTORY DUTY), STRICT
 * LIABILITY, INDEMNITY, CONTRIBUTION, OR OTHERWISE, FOR ANY INDIRECT,
 * SPECIAL, PUNITIVE, EXEMPLARY, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE,
 * FOR COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE SOFTWARE,
 * HOWSOEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY
 * OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWABLE BY LAW,
 * MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS
 * SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID
 * DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.
 *
 * MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
 * THESE TERMS.
 *******************************************************************************/
```

EXAMPLE 9-1: SPI PORT SCANNING SOFTWARE CODE EXAMPLE (CONTINUED)

```
.include "xc.inc"

.global  _sd_Port_Search

.pushsection .text, code

;--------------------------------------------------------------------------------
; Constants
;--------------------------------------------------------------------------------

.equ  PPS_SPI2_DATA_SCK_SDI_INPUT, #RPINR22  ; PPS register for data and sck input

.equ  PPS_PGEC1_PGED1_INPUT, #0x0100
.equ  PPS_PGEC2_PGED2_INPUT, #0x0B0A
.equ  PPS_PGEC3_PGED3_INPUT, #0x0605

.equ  ANS_PGEC1_PGED1_REG, ANSB
.equ  ANS_PGEC2_PGED2_REG, ANSB
.equ  ANS_PGEC3_PGED3_REG, ANSB

.equ  ANS_PGEC1_PGED1_MASK, #0xFFFC
.equ  ANS_PGEC2_PGED2_MASK, #0xFFFF
.equ  ANS_PGEC3_PGED3_MASK, #0xFFFF

.equ  PORT_FOUND, #1
.equ  PORT_NOT_FOUND, #0

;--------------------------------------------------------------------------------
; Code
;--------------------------------------------------------------------------------

;--------------------------------------------------------------------------------
; Returns port number found on w0, 0 if no port found
; It will also leave SPI2 connected to the identified port
;--------------------------------------------------------------------------------
_sd_Port_Search:
    rcall PPS_Unlock                    ; Must unlock PPS to scan PGED/PGEC ports
    rcall SPI_Init                      ; SPI1 as master and SPI2 as slave for loopback test

    rcall ICSP_Scan1
    cp    w0, #PORT_FOUND
    bra   z, port_search_found

    rcall ICSP_Scan2
    cp    w0, #PORT_FOUND
    bra   z, port_search_found

    rcall ICSP_Scan3
    cp    w0, #PORT_FOUND
    bra   z, port_search_found

port_search_not_found:
    bclr  SPI1CON1L, #15                ; disable SPI1
    clr   w0                            ; return 0
    return

port_search_found:
    bclr  SPI1CON1L, #15                ; disable SPI1
    mov   w1, w0                        ; return port number
    return
```


© 2015-2018 Microchip Technology Inc.

EXAMPLE 9-1: SPI PORT SCANNING SOFTWARE CODE EXAMPLE (CONTINUED)

```
;--------------------------------------------------------------------------------
PPS_Unlock:
    mov   #0, w0                        ; place zero in w0 to unlock PPS
    mov   #OSCCONL, w1                  ; OSCCONL (low byte) unlock sequence
    mov   #0x46, w2                     ; Preparing unlock sequence
    mov   #0x57, w3                     ; Preparing unlock sequence
    mov.b w2, [w1]                      ; Write 0x46
    mov.b w3, [w1]                      ; Write 0x9A
    mov.b w0, [w1]                      ; unlock PPS (IOLOCK bit = 0)
    return

;--------------------------------------------------------------------------------
; Initialize SPI1 as master and SPI2 as slave
; for loopback test on each programming port
;--------------------------------------------------------------------------------
SPI_Init:
    clr   SPI1BRGL                      ; clear SPI1BRGL
    clr   SPI1CON1H                     ; clear SPI1CON1H
    clr   SPI1CON1L                     ; clear SPI1CON1L
    mov   #0x8120, w0                   ; set SPIEN, CKE and MSTEN (master)
    mov   w0, SPI1CON1L

    clr   SPI2CON1H                     ; clear SPI2CON1H
    clr   SPI2CON1L                     ; clear SPI2CON1L
    mov   #0x0100, w0                   ; set CKE (slave)
    mov   w0, SPI2CON1L
    return

;--------------------------------------------------------------------------------
; Returns PORT_FOUND, or PORT_NOT_FOUND on w0
; Returns port number on w1
;--------------------------------------------------------------------------------
ICSP_Scan1:
    mov   #ANS_PGEC1_PGED1_MASK, w0     ; RB0 and RB1 are digital inputs for SCK and SDI
    mov   w0, ANS_PGEC1_PGED1_REG
    mov   #PPS_PGEC1_PGED1_INPUT, w0
    mov   w0, PPS_SPI2_DATA_SCK_SDI_INPUT

    rcall _sd_Loopback_Test

    mov   #1, w1
    return

;--------------------------------------------------------------------------------
ICSP_Scan2:
    mov   #ANS_PGEC2_PGED2_MASK, w0     ; RB0 and RB1 are digital inputs for SCK and SDI
    mov   w0, ANS_PGEC2_PGED2_REG
    mov   #PPS_PGEC2_PGED2_INPUT, w0
    mov   w0, PPS_SPI2_DATA_SCK_SDI_INPUT

    rcall _sd_Loopback_Test

    mov   #2, w1
    return
```

EXAMPLE 9-1: SPI PORT SCANNING SOFTWARE CODE EXAMPLE (CONTINUED)

```
;--------------------------------------------------------------------------------
ICSP_Scan3:
    mov   #ANS_PGEC3_PGED3_MASK, w0       ; RB0 and RB1 are digital inputs for SCK and SDI
    mov   w0, ANS_PGEC3_PGED3_REG
    mov   #PPS_PGEC3_PGED3_INPUT, w0
    mov   w0, PPS_SPI2_DATA_SCK_SDI_INPUT

    rcall _sd_Loopback_Test

    mov   #3, w1
    return

;--------------------------------------------------------------------------------
; Returns PORT_FOUND, or PORT_NOT_FOUND on w0
;--------------------------------------------------------------------------------
_sd_Loopback_Test:

    bset  SPI2CON1L, #15                  ; enable SPI2

    mov   #0x00a5, w0                     ; 0xa5 pattern
    mov   w0, SPI1BUFL                    ; send

loopback_wait_data:
    btss  SPI1STATL, #0                   ; wait for the data
    bra   loopback_wait_data

    mov   SPI1BUFL, w1
    mov   SPI2BUFL, w1                    ; read data
    bclr  SPI2CON1L, #15                  ; disable SPI2

    cp    w0, w1
    bra   z, loopback_found

    mov   #PORT_NOT_FOUND, w0
    return

loopback_found:
    mov   #PORT_FOUND, w0
    return

.popsection
```


© 2015-2018 Microchip Technology Inc.

APPENDIX B: REVISION HISTORY

Revision A (July 2015)

Original version of the programming specification.

Revision B (February 2017)

Replaced all diagrams in **"Pin Diagrams (28-Pin QFN)"**, **"Pin Diagrams (28-Pin PDIP)"**, **"Pin Diagrams (44-Pin TQFP/QFN)"** and **"Pin Diagrams (48-Pin TQFP/QFN)"**.

Edits made to Table 2-2 and Table 2-3. Replaced Table 2-4 with new table.

Edits made to Figure 2-3.

Replaced **Section 8.0 "Checksum Computation"** with new text.

Added **Appendix A: "SPI Port Scanning Software Routine"**.

Changed Hex data formatting throughout the document.

Revision C (November 2018)

Updated **Section 4.4 "Entering Enhanced ICSP Mode"** and Figure 4-3.

Updated the **"Pin Diagrams (28-Pin QFN)"** and **"Pin Diagrams (28-Pin PDIP)"** sections.

NOTES:

© 2015-2018 Microchip Technology Inc.

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

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

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

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE**.** Microchip disclaims all liability arising from this information and its use. Use of Microchip 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 unless otherwise stated.

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KeeLoq, Kleer, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, 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.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

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

© 2018, Microchip Technology Incorporated, All Rights Reserved.

ISBN: 978-1-5224-3884-7

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://www.microchip.com/support
Web Address:
www.microchip.com

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

Austin, TX
Tel: 512-257-3370

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

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

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

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

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

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

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

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

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

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

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

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

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

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

Finland - Espoo
Tel: 358-9-4520-820

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

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-67-3636

Germany - Karlsruhe
Tel: 49-721-625370

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

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

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

Italy - Padova
Tel: 39-049-7625286

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

Norway - Trondheim
Tel: 47-7288-4388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

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

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

© 2015-2018 Microchip Technology Inc.