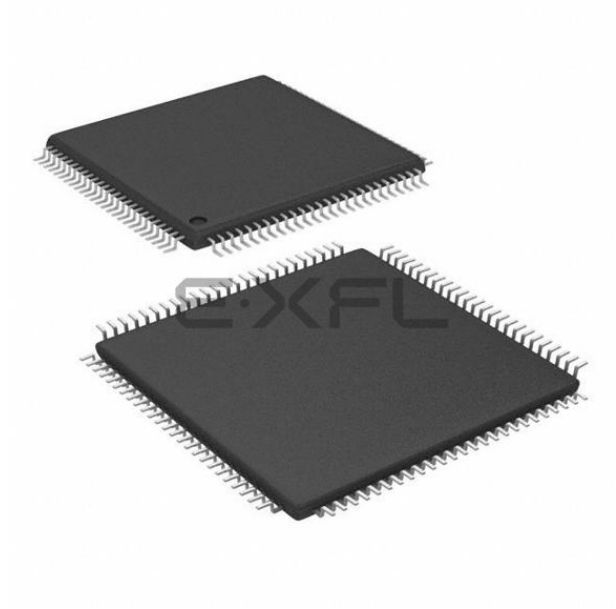


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	16-Bit
Speed	16MHz
Connectivity	I²C, PMP, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	84
Program Memory Size	128KB (43K x 24)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	8K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 16x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-TQFP
Supplier Device Package	100-TQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic24fj128ga010-i-pf

2.2 Program Memory Write/Erase Requirements

The Flash program memory on the PIC24FJXXXGA0XX devices has a specific write/erase requirement that must be adhered to for proper device operation. The rule is that any given word in memory must not be written more than twice before erasing the page in which it is located. Thus, the easiest way to conform to this rule is to write all the data in a programming block within one write cycle. The programming methods specified in this specification comply with this requirement.

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

2.3 Pin Diagrams

The pin diagrams for the PIC24FJXXXGA0XX family are shown in the following figures. The pins that are required for programming are listed in Table 2-1 and are shown in bold letters in the figures. Refer to the appropriate device data sheet for complete pin descriptions.

TABLE 2-1: PIN DESCRIPTIONS (DURING PROGRAMMING)

Pin Name	During Programming		
	Pin Name	Pin Type	Pin Description
$\overline{MCLR}$	$\overline{MCLR}$	P	Programming Enable
ENVREG	ENVREG	I	Enable for On-Chip Voltage Regulator
DISVREG(1)	DISVREG	I	Disable for On-Chip Voltage Regulator
VDD and AVDD(2)	VDD	P	Power Supply
VSS and AVSS(2)	VSS	P	Ground
VDDCORE	VDDCORE	P	Regulated Power Supply for Core
PGC1	PGC	I	Primary Programming Pin Pair: Serial Clock
PGD1	PGD	I/O	Primary Programming Pin Pair: Serial Data
PGC2	PGC	I	Secondary Programming Pin Pair: Serial Clock
PGD2	PGD	I/O	Secondary Programming Pin Pair: Serial Data

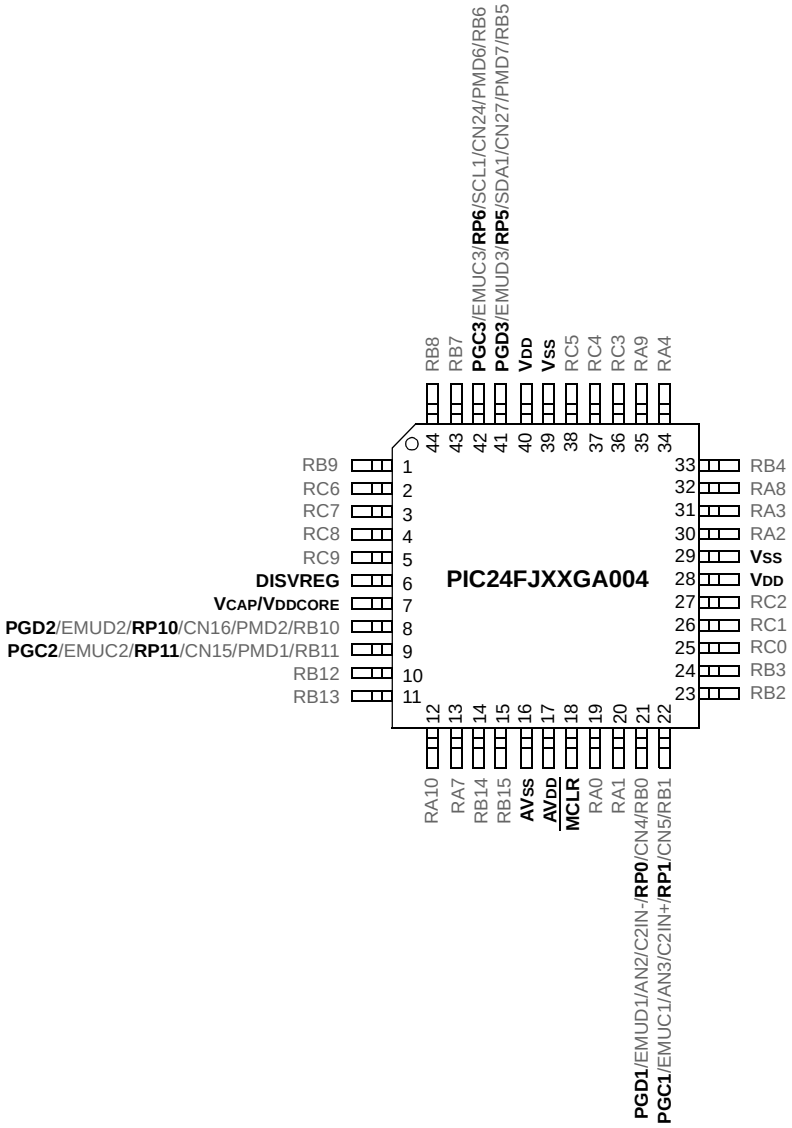
Legend: I = Input, O = Output, P = Power

Note 1: Applies to 28 and 44-pin devices only.

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

Pin Diagrams (Continued)

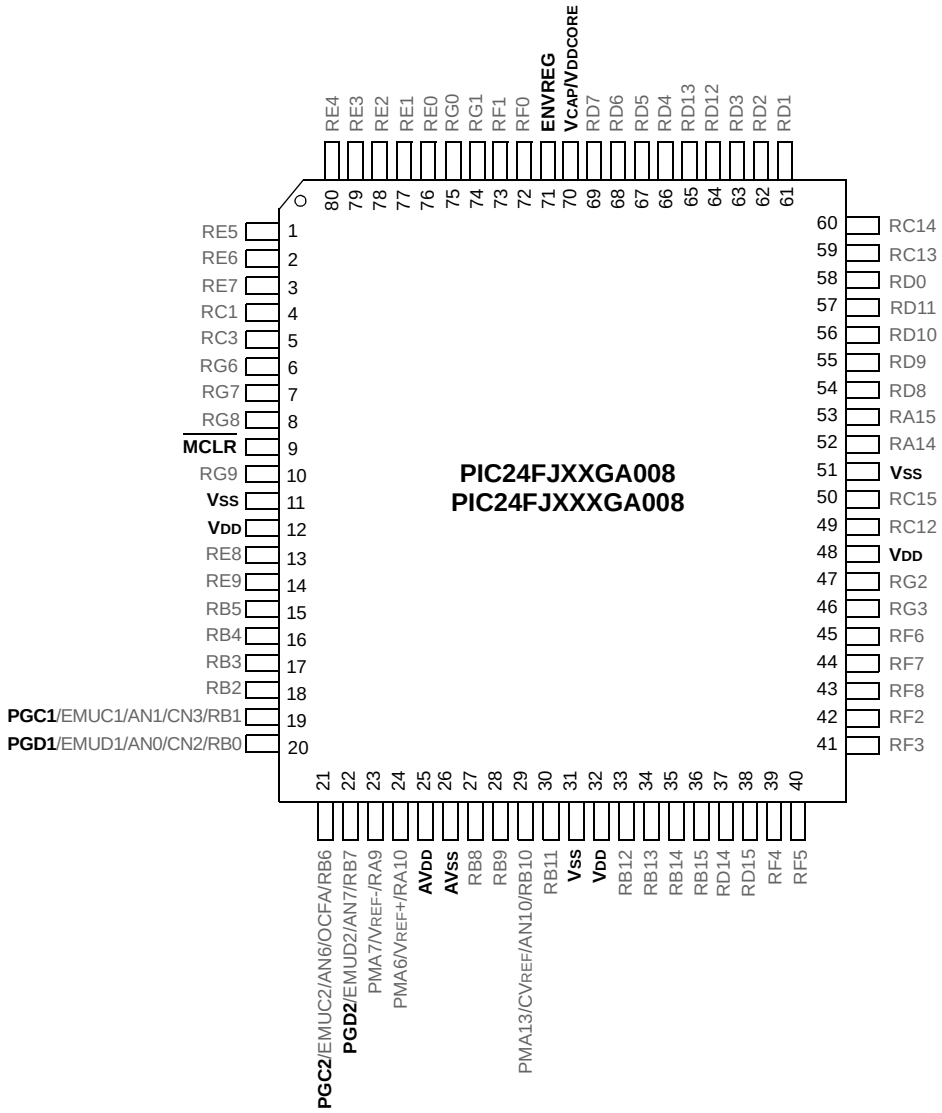
44-Pin TQFP



Legend: **RPx** represents remappable peripheral pins.

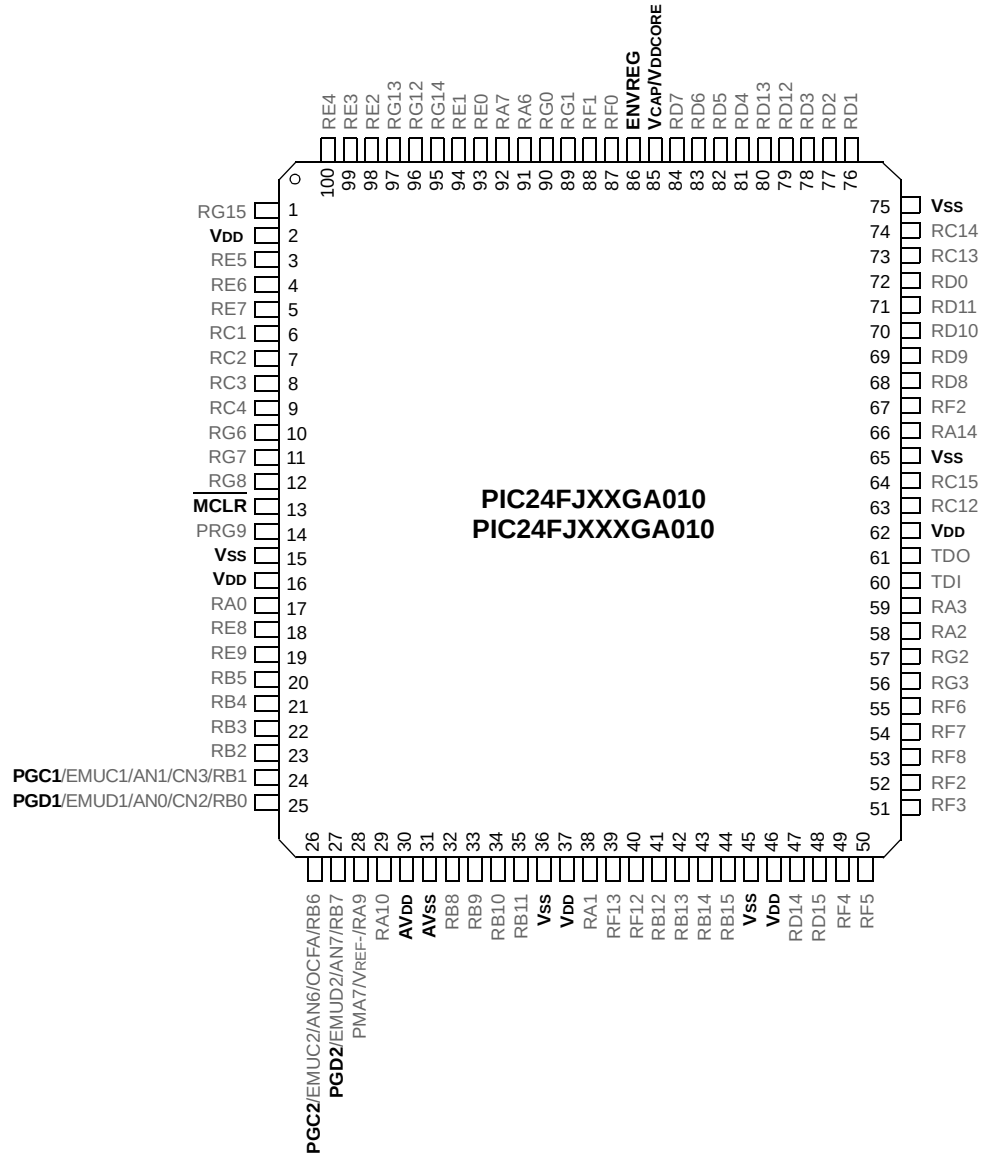
Pin Diagrams (Continued)

80-Pin TQFP



Pin Diagrams (Continued)

100-Pin TQFP



3.2.1 SIX SERIAL INSTRUCTION EXECUTION

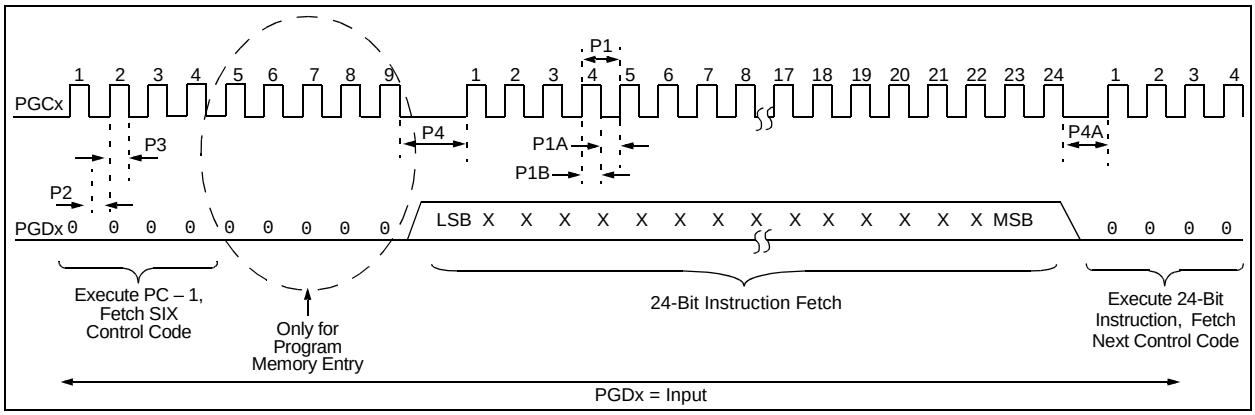
The SIX control code allows execution of the PIC24FJXXXGA0XX family assembly instructions. When the SIX code is received, the CPU is suspended for 24 clock cycles, as the instruction is then clocked into the internal buffer. Once the instruction is shifted in, the state machine allows it to be executed over the next four PGC clock cycles. While the received instruction is executed, the state machine simultaneously shifts in the next 4-bit command (see Figure 3-2).

Coming out of Reset, the first 4-bit control code is always forced to SIX and a forced NOP instruction is executed by the CPU. Five additional PGCx clocks are needed on start-up, resulting in a 9-bit SIX command instead of the normal 4-bit SIX command.

After the forced SIX is clocked in, ICSP operation resumes as normal. That is, the next 24 clock cycles load the first instruction word to the CPU.

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

FIGURE 3-2: SIX SERIAL EXECUTION



3.2.1.1 Differences Between Execution of SIX and Normal Instructions

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

The important differences are:

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

  Examples of two-word instructions are GOTO and CALL.

- Two-cycle instructions require two SIX operations.

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

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

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

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

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

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

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

  Instructions such as `tblwtl [W0++],[W1]` should be followed by two NOPs.

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

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

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

3.3 Entering ICSP Mode

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

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

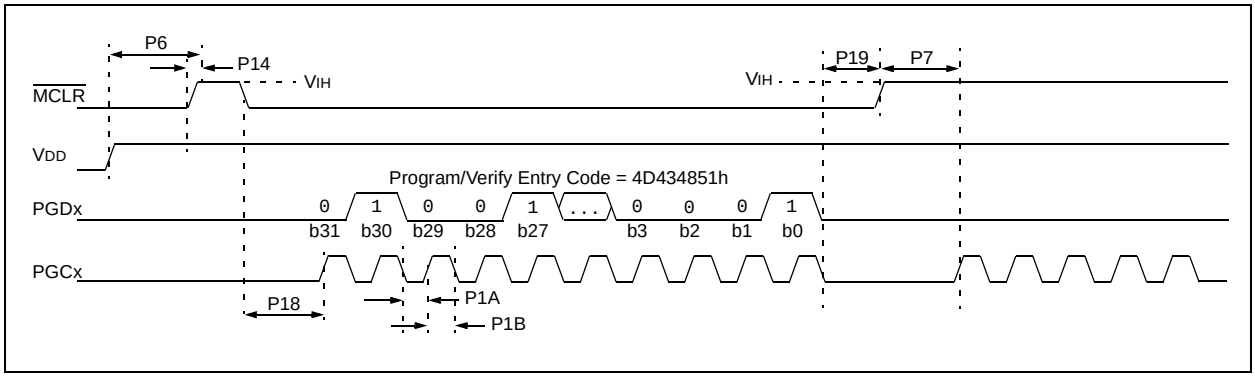
The programming voltage applied to MCLR is VIH, which is essentially VDD in the case of PIC24FJXXXGA0XX devices. There is no minimum time requirement for holding at VIH. After VIH is removed, an interval of at least P18 must elapse before presenting the key sequence on PGDx.

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

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

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

FIGURE 3-4: ENTERING ICSP™ MODE



3.6 Writing Code Memory

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

Table 3-5 shows the ICSP programming details, including the serial pattern with the ICSP command code which must be transmitted, Least Significant bit first, using the PGCx and PGDx pins (see Figure 3-2).

In Step 1, the Reset vector is exited. In Step 2, the NVMCON register is initialized for programming a full row of code memory. In Step 3, the 24-bit starting destination address for programming is loaded into the TBLPAG register and W7 register. (The upper byte of the starting destination address is stored in TBLPAG and the lower 16 bits of the destination address are stored in W7.)

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

In Step 4, four packed instruction words are stored in working registers, W0:W5, using the MOV instruction, and the Read Pointer, W6, is initialized. The contents of W0:W5 (holding the packed instruction word data) are shown in Figure 3-6.

In Step 5, eight TBLWT instructions are used to copy the data from W0:W5 to the write latches of code memory. Since code memory is programmed 64 instruction words at a time, Steps 4 and 5 are repeated 16 times to load all the write latches (Step 6).

After the write latches are loaded, programming is initiated by writing to the NVMCON register in Steps 7 and 8. In Step 9, the internal PC is reset to 200h. This is a precautionary measure to prevent the PC from incrementing into unimplemented memory when large devices are being programmed. Lastly, in Step 10, Steps 3-9 are repeated until all of code memory is programmed.

FIGURE 3-6: PACKED INSTRUCTION WORDS IN W<0:5>

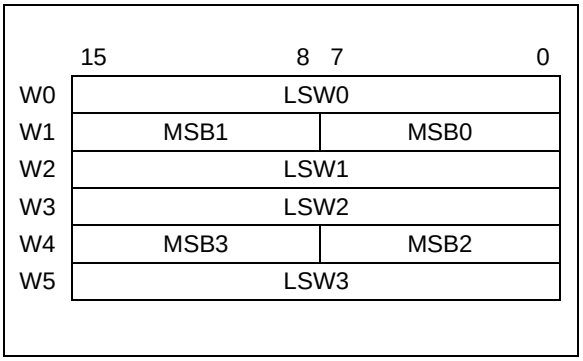


TABLE 3-5: SERIAL INSTRUCTION EXECUTION FOR WRITING CODE MEMORY

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

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

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

3.7 Writing Configuration Words

The PIC24FJXXXGA0XX family configuration is stored in Flash Configuration Words at the end of the user space program memory and in multiple register Configuration Words located in the test space.

These registers reflect values read at any Reset from program memory locations. The values can be changed only by programming the content of the corresponding Flash Configuration Word and resetting the device. The Reset forces an automatic reload of the Flash stored configuration values by sequencing through the dedicated Flash Configuration Words and transferring the data into the Configuration registers. To change the values of the Flash Configuration Word once it has been programmed, the device must be Chip Erased, as described in **Section 3.5 “Erasing Program Memory”**, and reprogrammed to the desired value. It is not possible to program a ‘0’ to ‘1’, but they may be programmed from a ‘1’ to ‘0’ to enable code protection.

Table 3-7 shows the ICSP programming details for programming the Configuration Word locations, including the serial pattern with the ICSP command code which must be transmitted, Least Significant bit first, using the PGCx and PGDx pins (see Figure 3-2).

In Step 1, the Reset vector is exited. In Step 2, the NVMCON register is initialized for programming of code memory. In Step 3, the 24-bit starting destination address for programming is loaded into the TBLPAG register and W7 register.

The TBLPAG register must be loaded with the following:

- 96 and 64 Kbyte devices – 00h
- 128 Kbyte devices – 01h

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

TABLE 3-6: DEFAULT CONFIGURATION REGISTER VALUES

Address	Name	Default Value
Last Word	CW1	7FFFh(1)
Last Word – 2	CW2	FFFFh

Note 1: CW1<15> is reserved and must be programmed to ‘0’.

3.8 Reading Code Memory

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

Table 3-8 shows the ICSP programming details for reading code memory. In Step 1, the Reset vector is exited. In Step 2, the 24-bit starting source address for reading is loaded into the TBLPAG register and W6 register. The upper byte of the starting source address is stored in TBLPAG and the lower 16 bits of the source address are stored in W6.

To minimize the reading time, the packed instruction word format that was utilized for writing is also used for reading (see Figure 3-6). In Step 3, the Write Pointer, W7, is initialized. In Step 4, two instruction words are read from code memory and clocked out of the device, through the VISI register, using the REGOUT command. Step 4 is repeated until the desired amount of code memory is read.

TABLE 3-8: SERIAL INSTRUCTION EXECUTION FOR READING CODE MEMORY

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

3.9 Reading Configuration Words

The procedure for reading configuration memory is similar to the procedure for reading code memory, except that 16-bit data words are read (with the upper byte read being all '0's) instead of 24-bit words. Since there are two Configuration registers, they are read one register at a time.

Table 3-9 shows the ICSP programming details for reading the Configuration Words. Note that the TBLPAG register must be loaded with 00h for 96 Kbyte and below devices and 01h for 128 Kbyte devices (the upper byte address of configuration memory), and the Read Pointer, W6, is initialized to the lower 16 bits of the Configuration Word location.

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

Command (Binary)	Data (Hex)	Description
Step 1: Exit Reset vector.		
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
Step 2: Initialize TBLPAG, the Read Pointer (W6) and the Write Pointer (W7) for TBLRD instruction.		
0000	200xx0	MOV <CW2Address23:16>, W0
0000	880190	MOV W0, TBLPAG
0000	2xxxx7	MOV <CW2Address15:0>, W6
0000	207847	MOV #VISI, W7
0000	000000	NOP
Step 3: Read the Configuration register and write it to the VISI register (located at 784h), and clock out the VISI register using the REGOUT command.		
0000	BA0BB6	TBLRDL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register
0000	000000	NOP
Step 4: Repeat Step 3 again to read Configuration Word 1.		
Step 5: Reset device internal PC.		
0000	040200	GOTO 0x200
0000	000000	NOP

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

Command (Binary)	Data (Hex)	Description
Step 1: Exit Reset vector.		
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
Step 2: Initialize TBLPAG and the Read Pointer (W0) for TBLRD instruction.		
0000	200800	MOV #0x80, W0
0000	880190	MOV W0, TBLPAG
0000	205BE0	MOV #0x5BE, W0
0000	207841	MOV #VISI, W1
0000	000000	NOP
0000	BA0890	TBLRDL [W0], [W1]
0000	000000	NOP
0000	000000	NOP
Step 3: Output the VISI register using the REGOUT command.		
0001	<VISI>	Clock out contents of the VISI register
0000	000000	NOP

4.0 DEVICE PROGRAMMING – ENHANCED ICSP

This section discusses programming the device through Enhanced ICSP and the programming executive. The programming executive resides in executive memory (separate from code memory) and is executed when Enhanced ICSP Programming mode is entered. The programming executive provides the mechanism for the programmer (host device) to program and verify the PIC24FJXXXGA0XX devices using a simple command set and communication protocol. There are several basic functions provided by the programming executive:

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

The programming executive performs the low-level tasks required for erasing, programming and verifying a device. This allows the programmer to program the device by issuing the appropriate commands and data. Table 4-1 summarizes the commands. A detailed description for each command is provided in **Section 5.2 "Programming Executive Commands"**.

TABLE 4-1: COMMAND SET SUMMARY

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

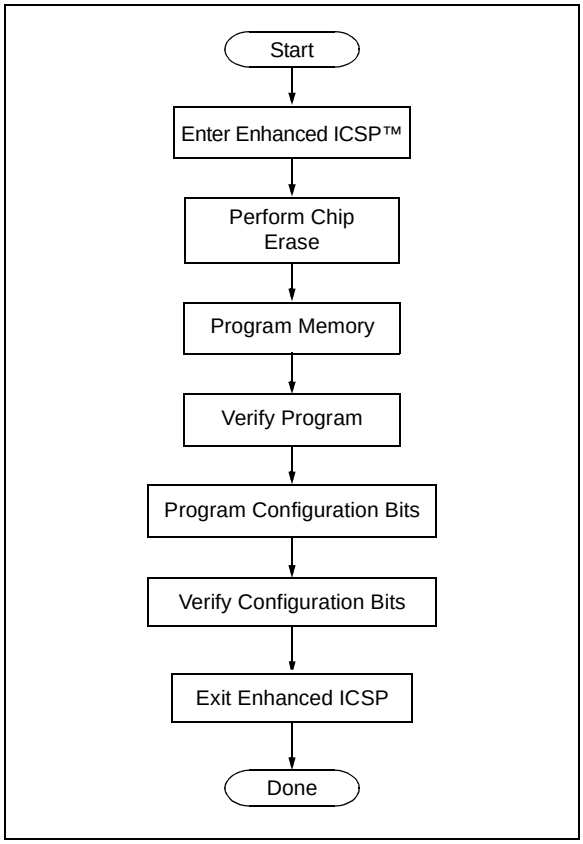
The programming executive uses the device's data RAM for variable storage and program execution. After the programming executive has run, no assumptions should be made about the contents of data RAM.

4.1 Overview of the Programming Process

Figure 4-1 shows the high-level overview of the programming process. After entering Enhanced ICSP mode, the programming executive is verified. Next, the device is erased. Then, the code memory is programmed, followed by the configuration locations. Code memory (including the Configuration registers) is then verified to ensure that programming was successful.

After the programming executive has been verified in memory (or loaded if not present), the PIC24FJXXXGA0XX family can be programmed using the command set shown in Table 4-1.

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



4.2 Confirming the Presence of the Programming Executive

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

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

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

4.5.2 PROGRAMMING VERIFICATION

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

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

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

4.6 Configuration Bits Programming

4.6.1 OVERVIEW

The PIC24FJXXXGA0XX family has Configuration bits stored in the last two locations of implemented program memory (see Table 2-2 for locations). These bits can be set or cleared to select various device configurations. There are three types of Configuration bits: system operation bits, code-protect bits and unit ID bits. The system operation bits determine the power-on settings for system level components, such as oscillator and Watchdog Timer. The code-protect bits prevent program memory from being read and written.

The register descriptions for the CW1 and CW2 Configuration registers are shown in Table 4-2.

TABLE 4-2: PIC24FJXXXGA0XX FAMILY CONFIGURATION BITS DESCRIPTION

Bit Field	Register	Description
I2C1SEL(1)	CW2<2>	I2C1 Pin Mapping bit 1 = Default location for SCL1/SDA1 pins 0 = Alternate location for SCL1/SDA1 pins
$\overline{\text{DEBUG}}$	CW1<11>	Background Debug Enable bit 1 = Device will reset in User mode 0 = Device will reset in Debug mode
FCKSM1:FCKSM0	CW2<7:6>	Clock Switching Mode bits 1x = Clock switching is disabled, Fail-Safe Clock Monitor is disabled 01 = Clock switching is enabled, Fail-Safe Clock Monitor is disabled 00 = Clock switching is enabled, Fail-Safe Clock Monitor is enabled
FNOSC2:FNOSC0	CW2<10:8>	Initial Oscillator Source Selection bits 111 = Internal Fast RC (FRCDIV) oscillator with postscaler 110 = Reserved 101 = Low-Power RC (LPRC) oscillator 100 = Secondary (SOSC) oscillator 011 = Primary (XTPLL, HSPLL, ECPLL) oscillator with PLL 010 = Primary (XT, HS, EC) oscillator 001 = Internal Fast RC (FRCPLL) oscillator with postscaler and PLL 000 = Fast RC (FRC) oscillator
FWDTEN	CW1<7>	Watchdog Timer Enable bit 1 = Watchdog Timer always enabled (LPRC oscillator cannot be disabled; clearing the SWDTEN bit in the RCON register will have no effect) 0 = Watchdog Timer enabled/disabled by user software (LPRC can be disabled by clearing the SWDTEN bit in the RCON register)
GCP	CW1<13>	General Segment Code-Protect bit 1 = User program memory is not code-protected 0 = User program memory is code-protected
GWRP	CW1<12>	General Segment Write-Protect bit 1 = User program memory is not write-protected 0 = User program memory is write-protected
ICS	CW1<8>	ICD Communication Channel Select bit 1 = Communicate on PGC2/EMUC2 and PGD2/EMUD2 0 = Communicate on PGC1/EMUC1 and PGD1/EMUD1

Note 1: Available on 28 and 44-pin packages only.

2: Available only on 28 and 44-pin devices with a silicon revision of 3042h or higher.

5.2.8 PROGC COMMAND

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

Field	Description
Opcode	4h
Length	4h
Reserved	0h
Addr_MSB	MSB of 24-bit destination address
Addr_LS	Least Significant 16 bits of 24-bit destination address
Data	8-bit data word

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

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

Expected Response (2 words):

1400h
0002h

5.2.9 PROGP COMMAND

15 12	11 8	7 0
Opcode	Length	
Reserved	Addr_MSB	
Addr_LS		
D_1		
D_2		
...		
D_96		

Field	Description
Opcode	5h
Length	63h
Reserved	0h
Addr_MSB	MSB of 24-bit destination address
Addr_LS	Least Significant 16 bits of 24-bit destination address
D_1	16-bit data word 1
D_2	16-bit data word 2
...	16-bit data word 3 through 95
D_96	16-bit data word 96

The PROGP command instructs the programming executive to program one row of code memory, including Configuration Words (64 instruction words), to the specified memory address. Programming begins with the row address specified in the command. The destination address should be a multiple of 80h.

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

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

Expected Response (2 words):

1500h
0002h

Note: Refer to Table 2-3 for code memory size information.

5.3.1.3 QE_Code Field

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

When the programming executive processes one of the two query commands (QBLANK or QVER), the returned opcode is always PASS and the QE_Code holds the query response data. The format of the QE_Code for both queries is shown in Table 5-3.

TABLE 5-3: QE_Code FOR QUERIES

Query	QE_Code
QBLANK	0Fh = Code memory is NOT blank F0h = Code memory is blank
QVER	0xMN, where programming executive software version = M.N (i.e., 32h means software version 3.2)

When the programming executive processes any command other than a query, the QE_Code represents an error code. Supported error codes are shown in Table 5-4. If a command is successfully processed, the returned QE_Code is set to 0h, which indicates that there was no error in the command processing. If the verify of the programming for the PROGP or PROGC command fails, the QE_Code is set to 1h. For all other programming executive errors, the QE_Code is 2h.

TABLE 5-4: QE_Code FOR NON-QUERY COMMANDS

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

5.3.1.4 Response Length

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

With the exception of the response for the READP command, the length of each response is only 2 words.

The response to the READP command uses the packed instruction word format described in **Section 5.2.2 "Packed Data Format"**. When reading an odd number of program memory words (N odd), the response to the READP command is (3 * (N + 1)/2 + 2) words. When reading an even number of program memory words (N even), the response to the READP command is (3 * N/2 + 2) words.

6.2 Checksum Computation

Checksums for the PIC24FJXXXGA0XX family are 16 bits in size. The checksum is calculated by summing the following:

- Contents of code memory locations
- Contents of Configuration registers

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

TABLE 6-4: CHECKSUM COMPUTATION

Device	Read Code Protection	Checksum Computation	Erased Checksum Value	Checksum with 0xAAAAAA at 0x0 and Last Code Address
PIC24FJ16GA002	Disabled	CFGB + SUM(0:02BFB)	0xBB5A	0xB95C
	Enabled	0	0x0000	0x0000
PIC24FJ16GA004	Disabled	CFGB + SUM(0:02BFB)	0xBB5A	0xB95C
	Enabled	0	0x0000	0x0000
PIC24FJ32GA002	Disabled	CFGB + SUM(0:057FB)	0x795A	0x775C
	Enabled	0	0x0000	0x0000
PIC24FJ32GA004	Disabled	CFGB + SUM(0:057FB)	0x795A	0x775C
	Enabled	0	0x0000	0x0000
PIC24FJ48GA002	Disabled	CFGB + SUM(0:083FB)	0x375A	0x355C
	Enabled	0	0x0000	0x0000
PIC24FJ48GA004	Disabled	CFGB + SUM(0:083FB)	0x375A	0x355C
	Enabled	0	0x0000	0x0000
PIC24FJ64GA002	Disabled	CFGB + SUM(0:0ABFB)	0xFB5A	0xF95C
	Enabled	0	0x0000	0x0000
PIC24FJ64GA004	Disabled	CFGB + SUM(0:0ABFB)	0xFB5A	0xF95C
	Enabled	0	0x0000	0x0000
PIC24FJ64GA006	Disabled	CFGB + SUM(0:0ABFB)	0xFACC	0xF8CE
	Enabled	0	0x0000	0x0000
PIC24FJ64GA008	Disabled	CFGB + SUM(0:0ABFB)	0xFACC	0xF8CE
	Enabled	0	0x0000	0x0000
PIC24FJ64GA010	Disabled	CFGB + SUM(0:0ABFB)	0xFACC	0xF8CE
	Enabled	0	0x0000	0x0000
PIC24FJ96GA006	Disabled	CFGB + SUM(0:0FFFB)	0x7CCC	0x7ACE
	Enabled	0	0x0000	0x0000
PIC24FJ96GA008	Disabled	CFGB + SUM(0:0FFFB)	0x7CCC	0x7ACE
	Enabled	0	0x0000	0x0000
PIC24FJ96GA010	Disabled	CFGB + SUM(0:0FFFB)	0x7CCC	0x7ACE
	Enabled	0	0x0000	0x0000

Legend:

Item		Description
SUM[a:b]	=	Byte sum of locations, a to b inclusive (all 3 bytes of code memory)
CFGB	=	Configuration Block (masked), 64/80/100-Pin Devices = Byte sum of (CW1 & 0x7DDF + CW2 & 0x87E3) 28/44-Pin Devices = Byte sum of (CW1 & 0x7FDF + CW2 & 0xFFF7)

Note: CW1 address is last location of implemented program memory; CW2 is (last location – 2).

TABLE 6-4: CHECKSUM COMPUTATION (CONTINUED)

Device	Read Code Protection	Checksum Computation	Erased Checksum Value	Checksum with 0xAAAAAA at 0x0 and Last Code Address
PIC24FJ128GAGA006	Disabled	CFGB + SUM(0:0157FB)	0xF8CC	0xF6CE
	Enabled	0	0x0000	0x0000
PIC24FJ128GAGA008	Disabled	CFGB + SUM(0:0157FB)	0xF8CC	0xF6CE
	Enabled	0	0x0000	0x0000
PIC24FJ128GAGA010	Disabled	CFGB + SUM(0:0157FB)	0xF8CC	0xF6CE
	Enabled	0	0x0000	0x0000

Legend: Item Description
SUM[a:b] = Byte sum of locations, a to b inclusive (all 3 bytes of code memory)
CFGB = Configuration Block (masked),
64/80/100-Pin Devices = Byte sum of (CW1 & 0x7DDF + CW2 & 0x87E3)
28/44-Pin Devices = Byte sum of (CW1 & 0x7FDF + CW2 & 0xFFF7)

Note: CW1 address is last location of implemented program memory; CW2 is (last location – 2).

7.0 AC/DC CHARACTERISTICS AND TIMING REQUIREMENTS

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

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
D111	VDD	Supply Voltage During Programming	VDDCORE + 0.1	3.60	V	Normal programming(1,2)
D112	IPP	Programming Current on $\overline{MCLR}$	—	5	μA	
D113	IDDP	Supply Current During Programming	—	2	mA	
D031	VIL	Input Low Voltage	VSS	0.2 VDD	V	
D041	VIH	Input High Voltage	0.8 VDD	VDD	V	
D080	VOL	Output Low Voltage	—	0.4	V	IOL = 8.5 mA @ 3.6V
D090	VOH	Output High Voltage	3.0	—	V	IOH = -3.0 mA @ 3.6V
D012	CIO	Capacitive Loading on I/O pin (PGDx)	—	50	pF	To meet AC specifications
D013	CF	Filter Capacitor Value on VCAP	4.7	10	μF	Required for controller core
P1	TPGC	Serial Clock (PGCx) Period	100	—	ns	
P1A	TPGCL	Serial Clock (PGCx) Low Time	40	—	ns	
P1B	TPGCH	Serial Clock (PGCx) High Time	40	—	ns	
P2	TSET1	Input Data Setup Time to Serial Clock ↑	15	—	ns	
P3	THLD1	Input Data Hold Time from PGCx ↑	15	—	ns	
P4	TDLY1	Delay Between 4-Bit Command and Command Operand	40	—	ns	
P4A	TDLY1A	Delay Between 4-Bit Command Operand and Next 4-Bit Command	40	—	ns	
P5	TDLY2	Delay Between Last PGCx ↓ of Command Byte to First PGCx ↑ of Read of Data Word	20	—	ns	
P6	TSET2	VDD ↑ Setup Time to $\overline{MCLR}$ ↑	100	—	ns	
P7	THLD2	Input Data Hold Time from $\overline{MCLR}$ ↑	25	—	ms	
P8	TDLY3	Delay Between Last PGCx ↓ of Command Byte to PGDx ↑ by Programming Executive	12	—	μs	
P9	TDLY4	Programming Executive Command Processing Time	40	—	μs	
P10	TDLY6	PGCx Low Time After Programming	400	—	ns	
P11	TDLY7	Chip Erase Time	400	—	ms	
P12	TDLY8	Page Erase Time	40	—	ms	
P13	TDLY9	Row Programming Time	2	—	ms	
P14	TR	$\overline{MCLR}$ Rise Time to Enter ICSP™ mode	—	1.0	μs	
P15	TVALID	Data Out Valid from PGCx ↑	10	—	ns	
P16	TDLY10	Delay Between Last PGCx ↓ and $\overline{MCLR}$ ↓	0	—	s	
P17	THLD3	$\overline{MCLR}$ ↓ to VDD ↓	100	—	ns	
P18	TKEY1	Delay from First $\overline{MCLR}$ ↓ to First PGCx ↑ for Key Sequence on PGDx	40	—	ns	
P19	TKEY2	Delay from Last PGCx ↓ for Key Sequence on PGDx to Second $\overline{MCLR}$ ↑	1	—	ms	
P20	TDLY11	Delay Between PGDx ↓ by Programming Executive to PGDx Driven by Host	23	—	μs	
P21	TDLY12	Delay Between Programming Executive Command Response Words	8	—	ns	

Note 1: VDDCORE must be supplied to the VDDCORE/VCAP pin if the on-chip voltage regulator is disabled. See **Section 2.1 "Power Requirements"** for more information. (Minimum VDDCORE allowing Flash programming is 2.25V.)

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

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

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

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

Trademarks

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

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