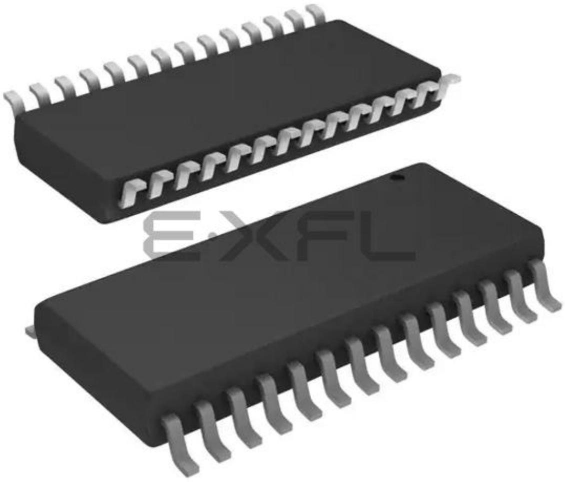
Microchip Technology - PIC24FJ48GA002-E/SO Datasheet



Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

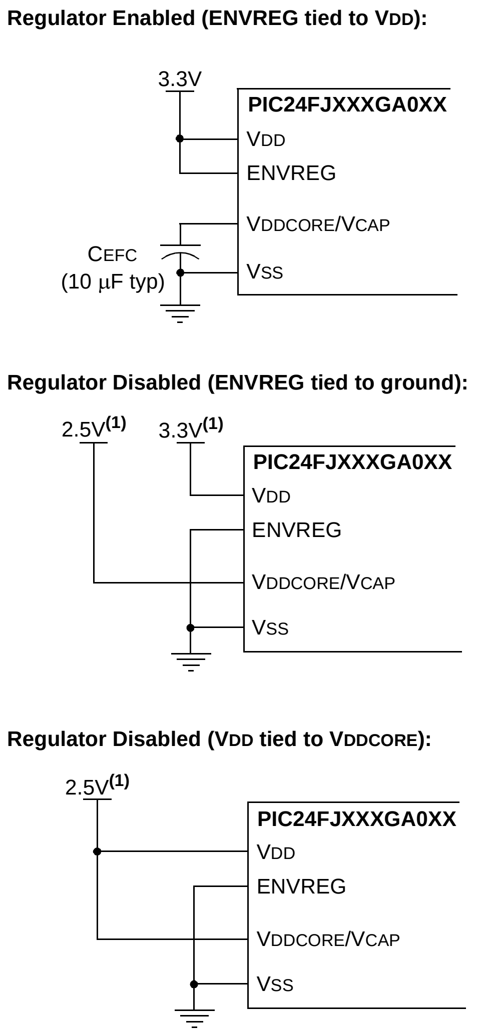
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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	32MHz
Connectivity	I²C, PMP, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	21
Program Memory Size	48KB (16K x 24)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	8K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 10x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic24fj48ga002-e-so

The regulator provides power to the core from the other VDD pins. A low-ESR capacitor (such as tantalum) must be connected to the VDDCORE pin (Figure 2-2 and Figure 2-3). This helps to maintain the stability of the regulator. The specifications for core voltage and capacitance are listed in **Section 7.0 "AC/DC Characteristics and Timing Requirements"**.

FIGURE 2-2: CONNECTIONS FOR THE ON-CHIP REGULATOR (64/80/100-PIN DEVICES)



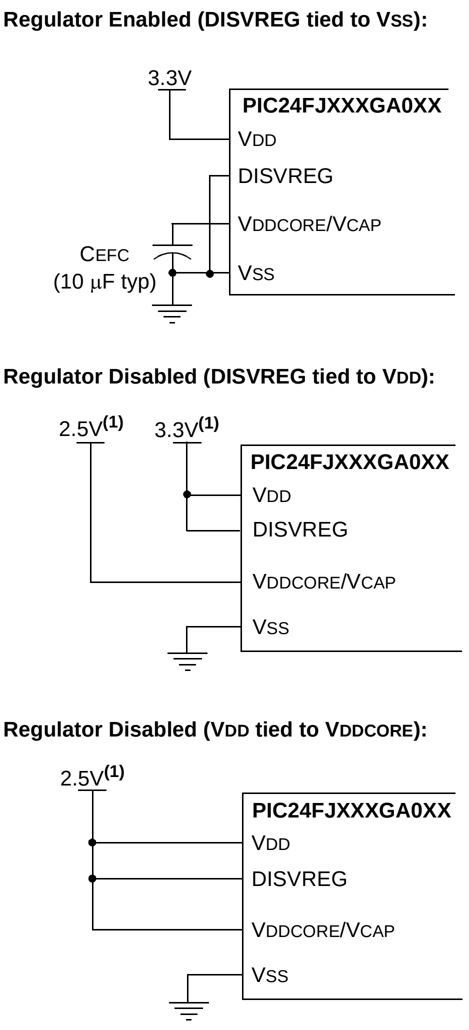
Regulator Enabled (ENVREG tied to VDD):

Regulator Disabled (ENVREG tied to ground):

Regulator Disabled (VDD tied to VDDCORE):

Note 1: These are typical operating voltages. Refer to **Section 7.0 "AC/DC Characteristics and Timing Requirements"** for the full operating ranges of VDD and VDDCORE.

FIGURE 2-3: CONNECTIONS FOR THE ON-CHIP REGULATOR (28/44-PIN DEVICES)



Regulator Enabled (DISVREG tied to VSS):

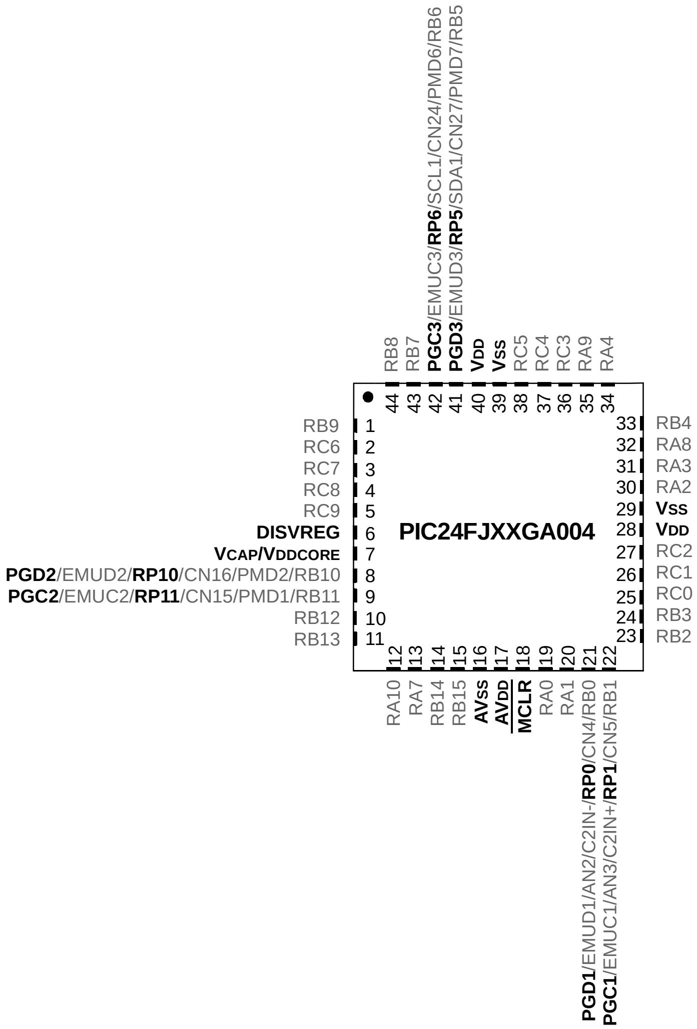
Regulator Disabled (DISVREG tied to VDD):

Regulator Disabled (VDD tied to VDDCORE):

Note 1: These are typical operating voltages. Refer to **Section 7.0 "AC/DC Characteristics and Timing Requirements"** for the full operating ranges of VDD and VDDCORE.

Pin Diagrams (Continued)

44-Pin QFN(1)

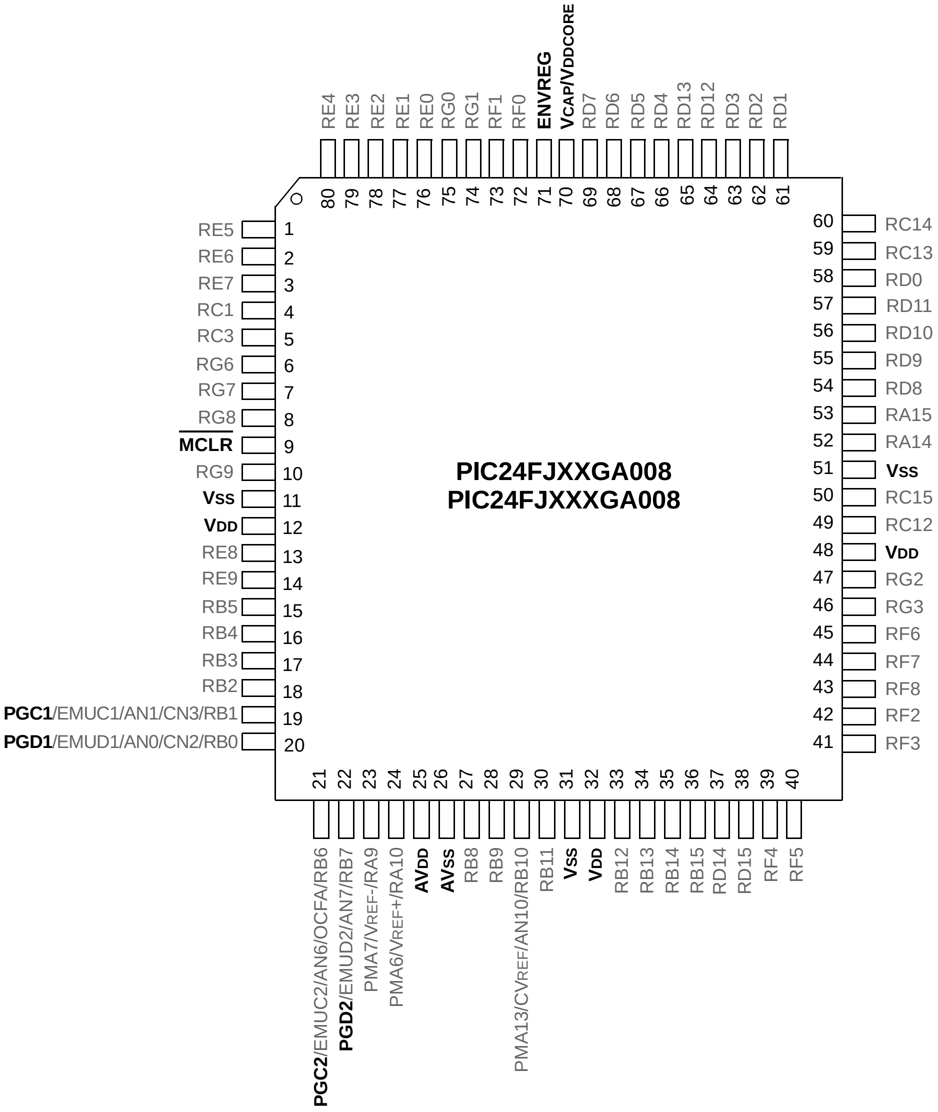


Legend: **RPx** represents remappable peripheral pins.

Note 1: The bottom pad of QFN packages should be connected to Vss.

Pin Diagrams (Continued)

80-Pin TQFP



3.2.1 SIX SERIAL INSTRUCTION EXECUTION

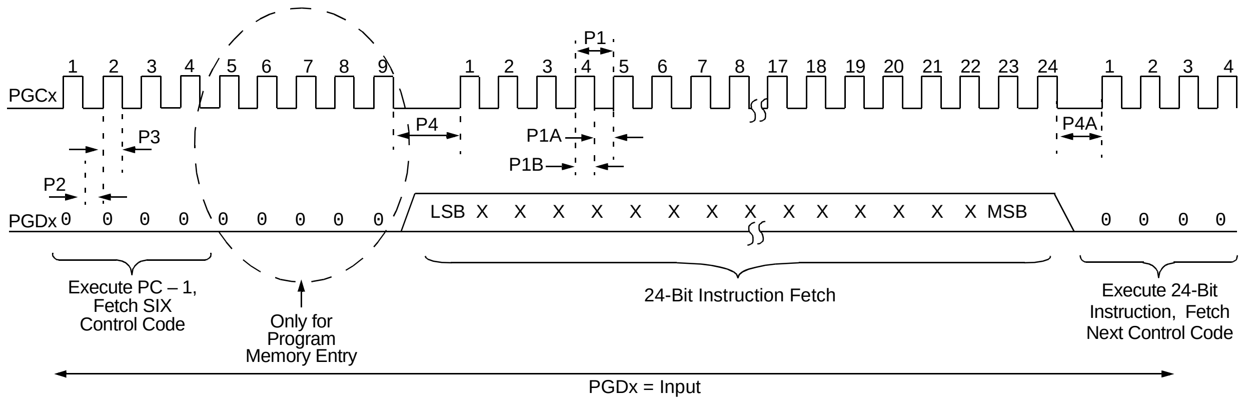
The SIX control code allows execution of the PIC24FJXXXGA0XX family assembly instructions. When the SIX code is received, the CPU is suspended for 24 clock cycles, as the instruction is then clocked into the internal buffer. Once the instruction is shifted in, the state machine allows it to be executed 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


P1
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 17 18 19 20 21 22 23 24
1 2 3 4
PGCx
P3
P2
P4
P1A
P1B
P4A
PGDx
0 0 0 0 0 0 0 0 0
LSB X X X X X X X X X X X X X X MSB
0 0 0 0
Execute PC – 1, Fetch SIX Control Code
Only for Program Memory Entry
24-Bit Instruction Fetch
Execute 24-Bit Instruction, Fetch Next Control Code
PGDx = Input

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 `NOP`s: one to execute the second half of the instruction and a second to stall the CPU to correct the pipeline.

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

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

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

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

3.3 Entering ICSP Mode

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

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

The programming voltage applied to $\overline{\text{MCLR}}$ is VIH, which is essentially VDD in the case of 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 $\overline{\text{MCLR}}$ and held at that level for as long as Program/Verify mode is to be maintained. An interval of at least time, P19 and P7, must elapse before presenting data on PGDx. Signals appearing on PGCx before P7 has elapsed will not be interpreted as valid.

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

FIGURE 3-4: ENTERING ICSP™ MODE

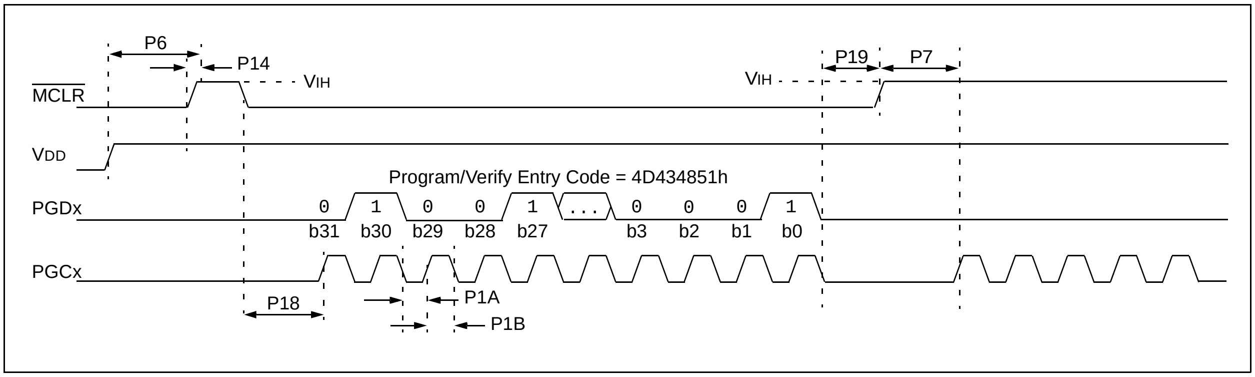


TABLE 3-5: SERIAL INSTRUCTION EXECUTION FOR WRITING CODE MEMORY (CONTINUED)

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.		

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

Command (Binary)	Data (Hex)	Description
Step 1: Exit the Reset vector.		
`0000`	`000000`	`NOP`
`0000`	`040200`	`GOTO 0x200`
`0000`	`000000`	`NOP`
Step 2: Initialize the Write Pointer (W7) for the `TBLWT` instruction.		
`0000`	`2xxxx7`	`MOV <CW2Address15:0>, W7`
Step 3: Set the NVMCON register to program CW2.		
`0000`	`24003A`	`MOV #0x4003, W10`
`0000`	`883B0A`	`MOV W10, NVMCON`
Step 4: Initialize the TBLPAG register.		
`0000`	`200xx0`	`MOV <CW2Address23:16>, W0`
`0000`	`880190`	`MOV W0, TBLPAG`
Step 5: Load the Configuration register data to W6.		
`0000`	`2xxxx6`	`MOV #<CW2_VALUE>, W6`
Step 6: Write the Configuration register data to the write latch and increment the Write Pointer.		
`0000`	`000000`	`NOP`
`0000`	`BB1B86`	`TBLWTL W6, [W7++]`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
Step 7: Initiate the write cycle.		
`0000`	`A8E761`	`BSET NVMCON, #WR`
`0000`	`000000`	`NOP`
`0000`	`000000`	`NOP`
Step 8: Repeat this step to poll the WR bit (bit 15 of NVMCON) until it is cleared by the hardware.		
`0000`	`040200`	`GOTO 0x200`
`0000`	`000000`	`NOP`
`0000`	`803B02`	`MOV NVMCON, W2`
`0000`	`883C22`	`MOV W2, VISI`
`0000`	`000000`	`NOP`
`0001`	`<VISI>`	`Clock out contents of the VISI register.`
`0000`	`000000`	`NOP`
Step 9: Reset device internal PC.		
`0000`	`040200`	`GOTO 0x200`
`0000`	`000000`	`NOP`
Step 10: Repeat Steps 5-9 to write CW1.		

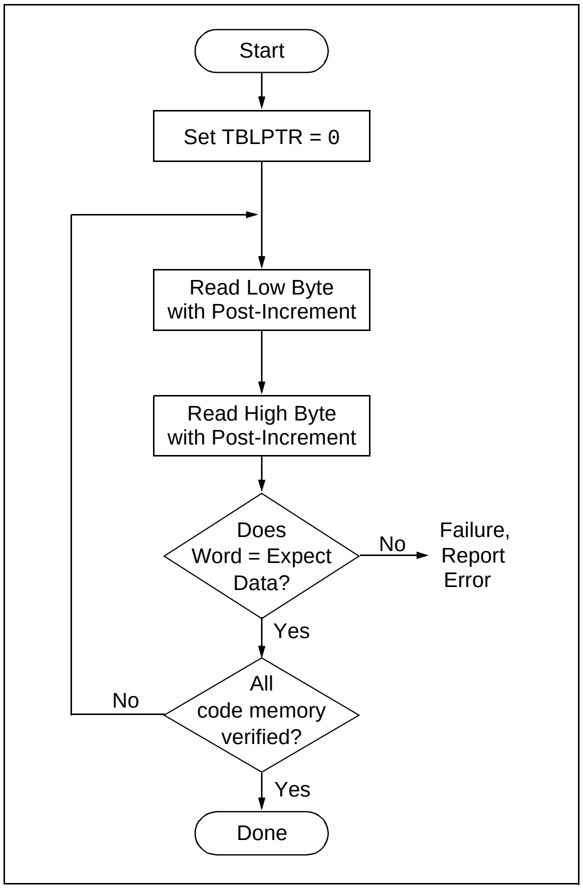
3.10 Verify Code Memory and Configuration Word

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

The verify process is shown in the flowchart in Figure 3-8. Memory reads occur a single byte at a time, so two bytes must be read to compare against the word in the programmer's buffer. Refer to **Section 3.8 "Reading Code Memory"** for implementation details of reading code memory.

> **Note:** Because the Configuration registers include the device code protection bit, code memory should be verified immediately after writing if code protection is enabled. This is because the device will not be readable or verifiable if a device Reset occurs after the code-protect bit in CW1 has been cleared.

FIGURE 3-8: VERIFY CODE MEMORY FLOW


Start
Set TBLPTR = 0
Read Low Byte with Post-Increment
Read High Byte with Post-Increment
Does Word = Expect Data?
No
Failure, Report Error
Yes
All code memory verified?
No
Yes
Done

3.11 Reading the Application ID Word

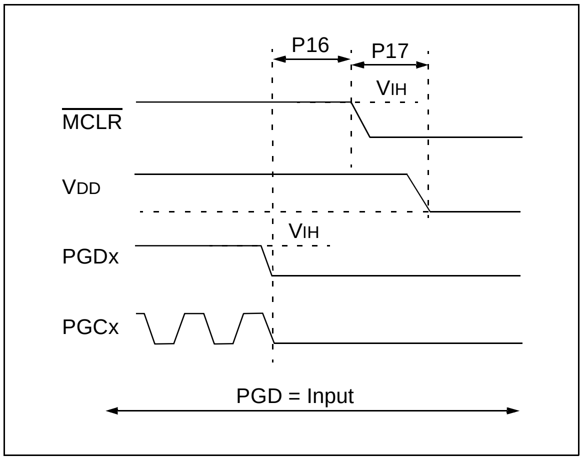
The Application ID Word is stored at address 8005BEh in executive code memory. To read this memory location, you must use the SIX control code to move this program memory location to the VISI register. Then, the REGOUT control code must be used to clock the contents of the VISI register out of the device. The corresponding control and instruction codes that must be serially transmitted to the device to perform this operation are shown in Table 3-10.

After the programmer has clocked out the Application ID Word, it must be inspected. If the Application ID has the value, BBh, the programming executive is resident in memory and the device can be programmed using the mechanism described in **Section 4.0 "Device Programming – Enhanced ICSP"**. However, if the Application ID has any other value, the programming executive is not resident in memory; it must be loaded to memory before the device can be programmed. The procedure for loading the programming executive to memory is described in **Section 5.4 "Programming the Programming Executive to Memory"**.

3.12 Exiting ICSP Mode

Exiting Program/Verify mode is done by removing VIH from $\overline{\text{MCLR}}$, as shown in Figure 3-9. The only requirement for exit is that an interval, P16, should elapse between the last clock and program signals on PGCx and PGDx before removing VIH.

FIGURE 3-9: EXITING ICSP™ MODE



4.4 Blank Check

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

The Device ID registers (FF0002h:FF0000h) can be ignored by the Blank Check since this region stores device information that cannot be erased. The device Configuration registers are also ignored by the Blank Check. Additionally, all unimplemented memory space should be ignored by the Blank Check.

The QBLANK command is used for the Blank Check. It determines if the code memory is erased by testing these memory regions. A 'BLANK' or 'NOT BLANK' response is returned. If it is determined that the device is not blank, it must be erased before attempting to program the chip.

4.5 Code Memory Programming

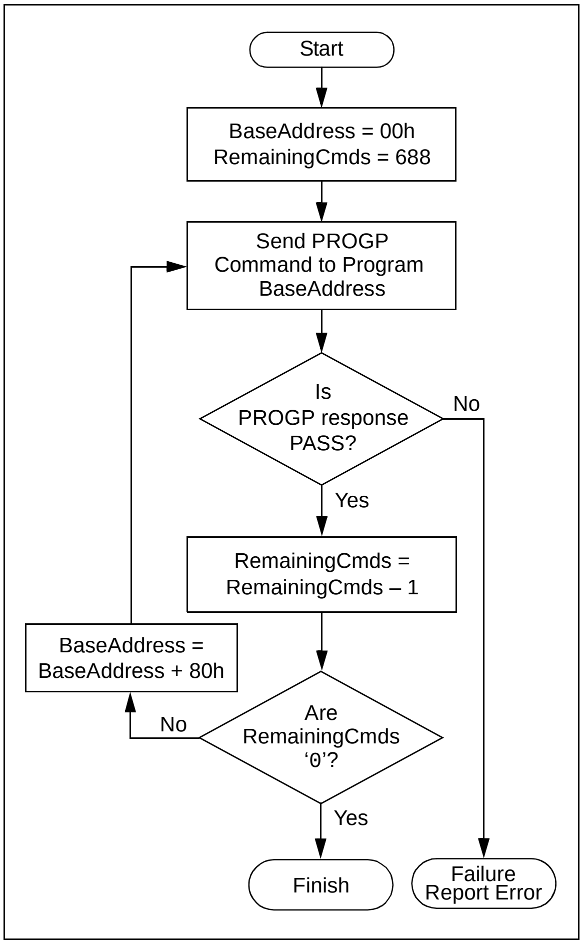
4.5.1 PROGRAMMING METHODOLOGY

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

A flowchart for programming code memory is shown in Figure 4-4. In this example, all 44K instruction words of a PIC24FJ128GA device are programmed. First, the number of commands to send (called 'RemainingCmds' in the flowchart) is set to 688 and the destination address (called 'BaseAddress') is set to '0'. Next, one write block in the device is programmed with a PROGP command. Each PROGP command contains data for one row of code memory of the PIC24FJXXXGA0XX device. After the first command is processed successfully, 'RemainingCmds' is decremented by 1 and compared with 0. Since there are more PROGP commands to send, 'BaseAddress' is incremented by 80h to point to the next row of memory.

On the second PROGP command, the second row is programmed. This process is repeated until the entire device is programmed. No special handling must be performed when a panel boundary is crossed.

FIGURE 4-4: FLOWCHART FOR PROGRAMMING CODE MEMORY



4.5.2 PROGRAMMING VERIFICATION

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

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

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

4.6 Configuration Bits Programming

4.6.1 OVERVIEW

The 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.

4.6.2 PROGRAMMING METHODOLOGY

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

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

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

4.6.3 PROGRAMMING VERIFICATION

After the Configuration bits are programmed, the contents of memory should be verified to ensure that the programming was successful. Verification requires the Configuration bits to be read back and compared against the copy held in the programmer's buffer. The READP command reads back the programmed Configuration bits and verifies that the programming was successful.

4.6.4 CODE-PROTECT CONFIGURATION BITS

CW1 Configuration register controls code protection for the PIC24FJXXXGA0XX family. Two forms of code protection are provided. One form prevents code memory from being written (write protection) and the other prevents code memory from being read (read protection).

GWRP (CW1<12>) controls write protection and GCP (CW1<13>) controls read protection. Protection is enabled when the respective bit is '0'.

Erasing sets GWRP and GCP to '1', which allows the device to be programmed.

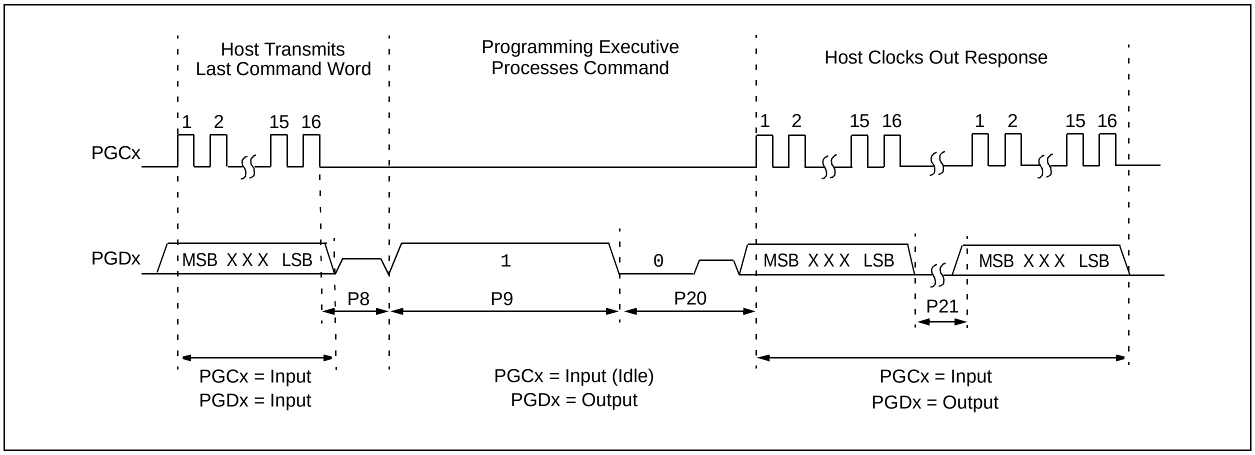
When write protection is enabled (GWRP = 0), any programming operation to code memory will fail.

When read protection is enabled (GCP = 0), any read from code memory will cause a 0h to be read, regardless of the actual contents of code memory. Since the programming executive always verifies what it programs, attempting to program code memory with read protection enabled also will result in failure.

It is imperative that both GWRP and GCP are '1' while the device is being programmed and verified. Only after the device is programmed and verified should either GWRP or GCP be programmed to '0' (see **Section 4.6 "Configuration Bits Programming"**).

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

FIGURE 5-3: PROGRAMMING EXECUTIVE – PROGRAMMER COMMUNICATION PROTOCOL



5.2 Programming Executive Commands

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

5.2.1 COMMAND FORMAT

All programming executive commands have a general format consisting of a 16-bit header and any required data for the command (see Figure 5-4). The 16-bit header consists of a 4-bit opcode field, which is used to identify the command, followed by a 12-bit command length field.

FIGURE 5-4: COMMAND FORMAT

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

The command opcode must match one of those in the command set. Any command that is received which does not match the list in Table 5-1 will return a "NACK" response (see **Section 5.3.1.1 "Opcode Field"**).

The command length is represented in 16-bit words since the SPI operates in 16-bit mode. The programming executive uses the command length field to determine the number of words to read from the SPI port. If the value of this field is incorrect, the command will not be properly received by the programming executive.

5.2.2 PACKED DATA FORMAT

When 24-bit instruction words are transferred across the 16-bit SPI interface, they are packed to conserve space using the format shown in Figure 5-5. This format minimizes traffic over the SPI and provides the programming executive with data that is properly aligned for performing table write operations.

FIGURE 5-5: PACKED INSTRUCTION WORD FORMAT

15 8	7 0
LSW1	
MSB2	MSB1
LSW2	

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

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

5.2.3 PROGRAMMING EXECUTIVE ERROR HANDLING

The programming executive will "NACK" all unsupported commands. Additionally, due to the memory constraints of the programming executive, no checking is performed on the data contained in the programmer command. It is the responsibility of the programmer to command the programming executive with valid command arguments or the programming operation may fail. Additional information on error handling is provided in **Section 5.3.1.3 "QE_Code Field"**.

5.2.6 READC COMMAND

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

Field	Description
Opcode	1h
Length	3h
N	Number of 8-bit Device ID registers to read (max. of 256)
Addr_MSB	MSB of 24-bit source address
Addr_LS	Least Significant 16 bits of 24-bit source address

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

When this command is used to read Device ID registers, the upper byte in every data word returned by the programming executive is 00h and the lower byte contains the Device ID register value.

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

1100h

2 + N

Device ID Register 1

...

Device ID Register N

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

5.2.7 READP COMMAND

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

Field	Description
Opcode	2h
Length	4h
N	Number of 24-bit instructions to read (max. of 32768)
Reserved	0h
Addr_MSB	MSB of 24-bit source address
Addr_LS	Least Significant 16 bits of 24-bit source address

The READP command instructs the programming executive to read N 24-bit words of code memory, including Configuration Words, starting from the 24-bit address specified by Addr_MSB and Addr_LS. This command can only be used to read 24-bit data. All data returned in response to this command uses the packed data format described in **Section 5.2.2 "Packed Data Format"**.

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

1200h

2 + 3 * N/2

Least significant program memory word 1

...

Least significant data word N

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

1200h

4 + 3 * (N – 1)/2

Least significant program memory word 1

...

MSB of program memory word N (zero padded)

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

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

Command (Binary)	Data (Hex)	Description
Step 7: Repeat Steps 5 and 6 to erase the second page of executive memory. The W1 Pointer should be incremented by 400h to point to the second page.		
Step 8: Initialize TBLPAG and NVMCON to write stored diagnostic and calibration as single words. Initialize W1 and W2 as Write and Read Pointers to rewrite stored Diagnostic and Calibration Words.		
0000	200800	MOV #0x80, W0
0000	880190	MOV W0, TBLPAG
0000	240031	MOV #0x4003, W1
0000	883B01	MOV W1, NVMCON
0000	207F00	MOV #0x07F0, W1
0000	2000C2	MOV #0xC, W2
0000	000000	NOP
Step 9: Perform write of a single word of calibration data and initiate single-word write cycle.		
0000	BB18B2	TBLWTL [W2++], [W1++]
0000	000000	NOP
0000	000000	NOP
0000	A8E761	BSET NVMCON, #15
0000	000000	NOP
0000	000000	NOP
Step 10: 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	803B00	MOV NVMCON, W0
0000	883C20	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	Clock out contents of VISI register.
0000	000000	NOP
Step 11: Repeat steps 9-10 seven more times to program the remainder of the Diagnostic and Calibration Words back into program memory.		
Step 12: Initialize the NVMCON to program 64 instruction words.		
0000	240010	MOV #0x4001, W0
0000	883B00	MOV W0, NVMCON
Step 13: Initialize TBLPAG and the Write Pointer (W7).		
0000	200800	MOV #0x80, W0
0000	880190	MOV W0, TBLPAG
0000	EB0380	CLR W7
0000	000000	NOP
Step 14: Load W0:W5 with the next four words of packed programming executive code and initialize W6 for programming. Programming starts from the base of executive memory (800000h) using W6 as a Read Pointer and W7 as a Write Pointer.		
0000	2<LSW0>0	MOV #<LSW0>, W0
0000	2<MSB1:MSB0>1	MOV #<MSB1:MSB0>, W1
0000	2<LSW1>2	MOV #<LSW1>, W2
0000	2<LSW2>3	MOV #<LSW2>, W3
0000	2<MSB3:MSB2>4	MOV #<MSB3:MSB2>, W4
0000	2<LSW3>5	MOV #<LSW3>, W5

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

Command (Binary)	Data (Hex)	Description
Step 15: Set the Read Pointer (W6) and load the (next four write) latches.		
0000	EB0300	`CLR W6`
0000	000000	`NOP`
0000	BB0BB6	`TBLWTL [W6++], [W7]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BBDBB6	`TBLWTH.B [W6++], [W7++]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BBEBB6	`TBLWTH.B [W6++], [++W7]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BB1BB6	`TBLWTL [W6++], [W7++]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BB0BB6	`TBLWTL [W6++], [W7]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BBDBB6	`TBLWTH.B [W6++], [W7++]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BBEBB6	`TBLWTH.B [W6++], [++W7]`
0000	000000	`NOP`
0000	000000	`NOP`
0000	BB1BB6	`TBLWTL [W6++], [W7++]`
0000	000000	`NOP`
0000	000000	`NOP`
Step 16: Repeat Steps 14-15, sixteen times, to load the write latches for the 64 instructions.		
Step 17: Initiate the programming cycle.		
0000	A8E761	`BSET NVMCON, #15`
0000	000000	`NOP`
0000	000000	`NOP`
Step 18: 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 19: Reset the device internal PC.		
0000	040200	`GOTO 0x200`
0000	000000	`NOP`
Step 20: Repeat Steps 14-19 until all 16 rows of executive memory have been programmed. On the final row, make sure to initialize the write latches at the Diagnostic and Calibration Words locations with 0xFFFFFF to ensure that the calibration is not overwritten.		

6.0 DEVICE DETAILS

6.1 Device ID

The Device ID region of memory can be used to determine mask, variant and manufacturing information about the chip. The Device ID region is 2 x 16 bits and it can be read using the READC command. This region of memory is read-only and can also be read when code protection is enabled.

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

TABLE 6-1: DEVICE IDs

Device	DEVID
PIC24FJ16GA002	0444h
PIC24FJ16GA004	044Ch
PIC24FJ32GA002	0445h
PIC24FJ32GA004	044Dh
PIC24FJ48GA002	0446h
PIC24FJ48GA004	044Eh
PIC24FJ64GA002	0447h
PIC24FJ64GA004	044Fh
PIC24FJ64GA006	0405h
PIC24FJ64GA008	0408h
PIC24FJ64GA010	040Bh
PIC24FJ96GA006	0406h
PIC24FJ96GA008	0409h
PIC24FJ96GA010	040Ch
PIC24FJ128GAGA006	0407h
PIC24FJ128GAGA008	040Ah
PIC24FJ128GAGA010	040Dh

TABLE 6-2: PIC24FJXXXGA0XX DEVICE ID REGISTERS

Address	Name	Bit															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FF0000h	DEVID	—		FAMID<7:0>								DEV<5:0>					
FF0002h	DEVREV	—							MAJRV<2:0>			—			DOT<2:0>		

TABLE 6-3: DEVICE ID BIT DESCRIPTIONS

Bit Field	Register	Description
FAMID<7:0>	DEVID	Encodes the family ID of the device
DEV<5:0>	DEVID	Encodes the individual ID of the device
MAJRV<2:0>	DEVREV	Encodes the major revision number of the device
DOT<2:0>	DEVREV	Encodes the minor revision number of the device

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).

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 $\overline{\text{MCLR}}$	—	5	µA	
D113	IDDP	Supply Current During Programming	—	2	mA	
D031	VIL	Input Low Voltage	VSS	0.2 VDD	V	
D041	VIH	Input High Voltage	0.8 VDD	VDD	V	
D080	VOL	Output Low Voltage	—	0.4	V	IOL = 8.5 mA @ 3.6V
D090	VOH	Output High Voltage	3.0	—	V	IOH = -3.0 mA @ 3.6V
D012	CIO	Capacitive Loading on I/O pin (PGDx)	—	50	pF	To meet AC specifications
D013	CF	Filter Capacitor Value on VCAP	4.7	10	µF	Required for controller core
P1	TPGC	Serial Clock (PGCx) Period	100	—	ns	
P1A	TPGCL	Serial Clock (PGCx) Low Time	40	—	ns	
P1B	TPGCH	Serial Clock (PGCx) High Time	40	—	ns	
P2	TSET1	Input Data Setup Time to Serial Clock ↑	15	—	ns	
P3	THLD1	Input Data Hold Time from PGCx ↑	15	—	ns	
P4	TDLY1	Delay Between 4-Bit Command and Command Operand	40	—	ns	
P4A	TDLY1A	Delay Between 4-Bit Command Operand and Next 4-Bit Command	40	—	ns	
P5	TDLY2	Delay Between Last PGCx ↓ of Command Byte to First PGCx ↑ of Read of Data Word	20	—	ns	
P6	TSET2	VDD ↑ Setup Time to $\overline{\text{MCLR}}$ ↑	100	—	ns	
P7	THLD2	Input Data Hold Time from $\overline{\text{MCLR}}$ ↑	25	—	ms	
P8	TDLY3	Delay Between Last PGCx ↓ of Command Byte to PGDx ↑ by Programming Executive	12	—	µs	
P9	TDLY4	Programming Executive Command Processing Time	40	—	µs	
P10	TDLY6	PGCx Low Time After Programming	400	—	ns	
P11	TDLY7	Chip Erase Time	400	—	ms	
P12	TDLY8	Page Erase Time	40	—	ms	
P13	TDLY9	Row Programming Time	2	—	ms	
P14	TR	$\overline{\text{MCLR}}$ Rise Time to Enter ICSP™ mode	—	1.0	µs	
P15	TVALID	Data Out Valid from PGCx ↑	10	—	ns	
P16	TDLY10	Delay Between Last PGCx ↓ and $\overline{\text{MCLR}}$ ↓	0	—	s	
P17	THLD3	$\overline{\text{MCLR}}$ ↓ to VDD ↓	100	—	ns	
P18	TKEY1	Delay from First $\overline{\text{MCLR}}$ ↓ to First PGCx ↑ for Key Sequence on PGDx	40	—	ns	
P19	TKEY2	Delay from Last PGCx ↓ for Key Sequence on PGDx to Second $\overline{\text{MCLR}}$ ↑	1	—	ms	
P20	TDLY11	Delay Between PGDx ↓ by Programming Executive to PGDx Driven by Host	23	—	µs	
P21	TDLY12	Delay Between Programming Executive Command Response Words	8	—	ns	

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

NOTES:

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

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

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

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

Trademarks

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, rfPIC, SmartShunt and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

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

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, PICkit, PICDEM, PICDEM.net, PICtail, PIC^{32} logo, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rfLAB, Select Mode, Total Endurance, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

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

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

Printed on recycled paper.

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

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