Section 33. Programming and Diagnostics

HIGHLIGHTS

This section of the manual contains the following topics:

33.1 INTRODUCTION

PIC32MX family of devices provide a complete range of programming and diagnostic features that can increase the flexibility of any application using them. These features allow system designers to include:

- Simplified field programmability using two-wire In-Circuit Serial Programming™ (ICSP™) interfaces
- Debugging using ICSP
- Programming and debugging capabilities using the Enhanced Joint Test Action Group (EJTAG) extension of Joint Test Action Group (JTAG) interfaces
- JTAG Boundary scan testing for device and board diagnostics

PIC32MX family of devices incorporate two programming and diagnostic modules, and a Trace Controller, that provide a range of functions to the application developer. They are summarized in Table 33-1.

Figure 33-1: Block Diagram of Programming, Debugging and Trace Ports

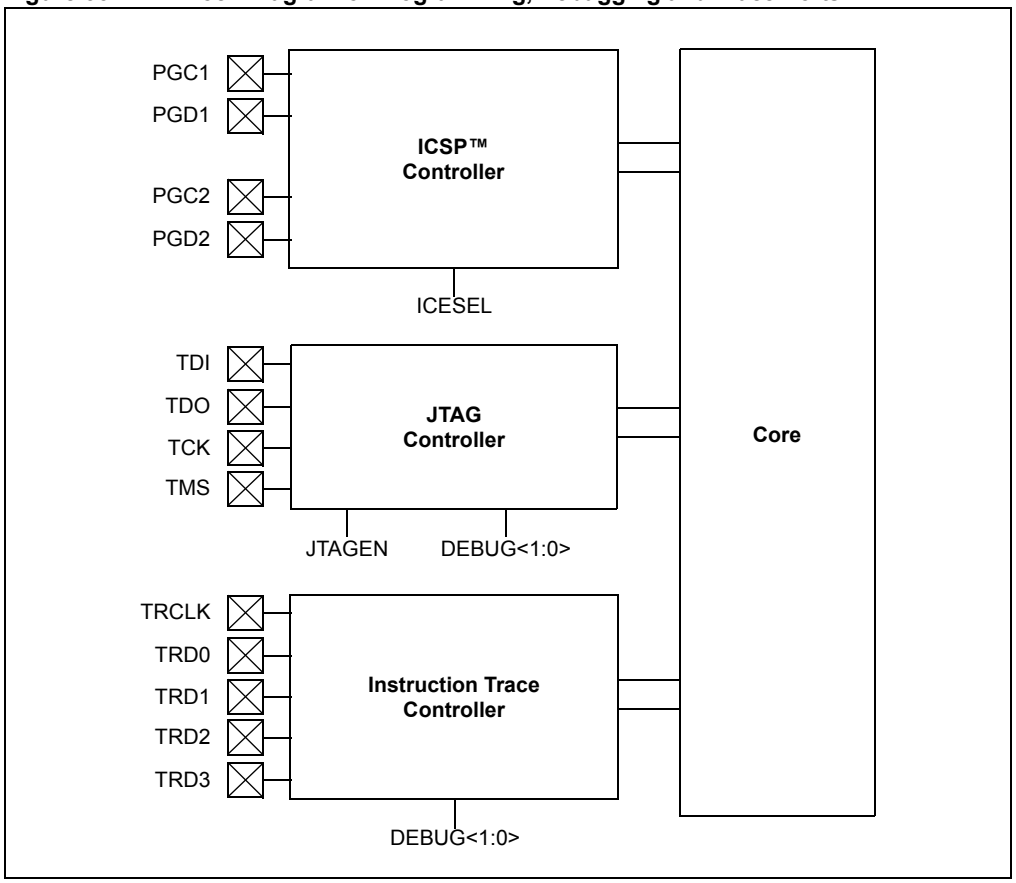


Table 33-1: Comparison of PIC32MX family Programming and Diagnostic Features

Functions	Pins Used	Interface
Boundary Scan	TDI, TDO, TMS and TCK pins	JTAG
Programming and Debugging	TDI, TDO, TMS and TCK pins	EJTAG
Programming and Debugging	PGCx and PGDx pins	ICSP™

33.2 CONTROL REGISTERS

The Programming and Diagnostics module consists of the following Special Function Registers (SFRs):

- DDPCON: Control Register for the Diagnostic Module

  DDPCONCLR, DDPCONSET, DDPCONINV: Atomic Bit Manipulation Write-only Registers for DDPCON

- DEVCFG0: Device Configuration Register 0

The following table summarizes all Programming and Diagnostics-related registers. Corresponding registers appear after the summary, followed by a detailed description of each register.

Table 33-2: Programming and Diagnostics SFR Summary

Name	Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
DDPCON	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	—	—	—	—	—	—	—
	7:0	—	—	—	—	JTAGEN	TROEN	—	—
DEVCFG0	31:24	—	—	—	CP	—	—	—	BWP
	23:16	—	—	—	—	PWP19	PWP18	PWP17	PWP16
	15:8	PWP15	PWP14	PWP13	PWP12	—	—	—	—
	7:0	—	—	—	—	ICESEL	—	DEBUG1	DEBUG0

Register 33-1: DDPCON: Debug Data Port Control Register

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 15							bit 8

r-x	r-x	r-x	r-x	R/W-1	R/W-0	r-x	r-x
—	—	—	—	JTAGEN	TROEN	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-4 **Reserved:** Write ‘0’; ignore read

bit 3 **JTAGEN:** JTAG Port Enable bit
- 1 = Enable JTAG Port
- 0 = Disable JTAG Port

bit 2 **TROEN:** Trace Output Enable bit
- 1 = Enable Trace Port
- 0 = Disable Trace Port

bit 1-0 **Reserved:** Write ‘1’; ignore read

Register 33-2: DEVCFG0: Device Configuration Register 0

r-1	r-1	r-1	R/P-1	r-1	r-1	r-1	R/P-1
—	—	—	CP	—	—	—	BWP
bit 31							bit 24

r-1	r-1	r-1	r-1	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	—	PWP19	PWP18	PWP17	PWP16
bit 23							bit 16

R/P-1	R/P-1	R/P-1	R/P-1	r-1	r-1	r-1	r-1
PWP15	PWP14	PWP13	PWP12	—	—	—	—
bit 15							bit 8

r-1	r-1	r-1	r-1	R/P-1	r-1	R/P-1	R/P-1
—	—	—	—	ICESEL	—	DEBUG1	DEBUG0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 3 **ICESEL:** ICE Debugger Port Select bit
1 = ICE Debugger uses PGC2/PGD2
0 = ICE Debugger uses PGC1/PGD1

bit 1-0 **DEBUG<1:0>:** Background Debugger Enable bits (forced to ‘11’ if Code-Protect is enabled)
11 = ICE Debugger Disabled
10 = ICE Debugger Enabled
01 = Reserved (same as ‘11’ setting)
00 = Reserved (same as ‘11’ setting)

33.3 OPERATION

The PIC32MX family of devices has multiple Programming and Debugging options including:

- In-Circuit Programming via ICSP
- In-Circuit Programming via EJTAG
- Debugging via ICSP
- Debugging via EJTAG
- Special Debug modes for select communication peripherals
- Boundary Scan

33.3.1 Device Programming Options

Note:	Following sections provide a brief overview of each programming option. For more detailed information, refer to the *"PIC32MX Flash Programming Specification"* (DS61145). For all device programming options, a minimum VDD requirement for Flash erase and programming operations is required. Refer to the specific device data sheet for further details.

33.3.1.1 IN-CIRCUIT SERIAL PROGRAMMING™ (ICSP™)

ICSP is Microchip's proprietary solution to providing microcontroller programming in the target application. ICSP is also the most direct method to program the device, whether the controller is embedded in a system or loaded into a device programmer.

33.3.1.1.1 ICSP Interface

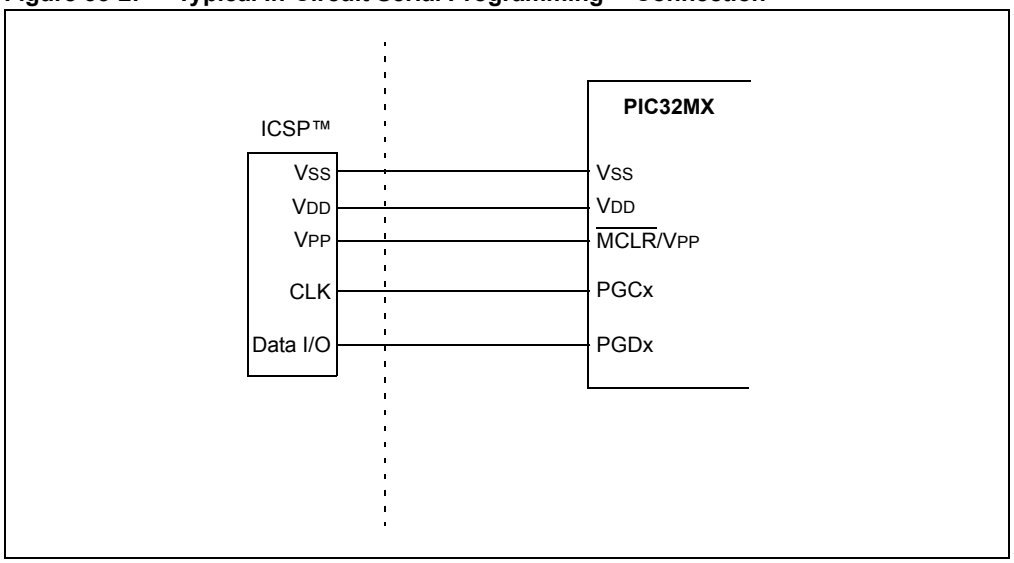
ICSP uses two pins as the core of its interface. The programming data line (PGD) functions as both an input and an output, allowing programming data to be read in and device information to be read out on command. The programming clock line (PGC) is used to clock in data and control the overall process.

Most PIC32MX family of devices have more than one pair of PGC and PGD pins; these are multiplexed with other I/O or peripheral functions. Individual ICSP pin pairs are indicated by number (e.g., PGC1/PGD1, etc.), and are generically referred to as 'PGCx' and 'PGDx'. The multiple PGCx/PGDx pairs provide additional flexibility in system design by allowing users to incorporate ICSP on the pair of pins that is least constrained by the circuit design. All PGCx and PGDx pins are functionally tied together and behave identically, and any one pair can be used for successful device programming. The only limitation is that both pins from the same pair must be used.

In addition to the PGCx and PGDx pins, ICSP requires that all voltage supply (including voltage regulator pin ENVREG) and ground pins on the device must be connected. The MCLR pin, which is used with PGCx to enter and control the programming process, must also be connected to the programmer.

A typical In-Circuit Serial Programming connection is shown in Figure 33-2.

Figure 33-2: Typical In-Circuit Serial Programming™ Connection



33.3.1.1.2 ICSP Operation

ICSP uses a combination of internal hardware and external control to program the target device. Programming data and instructions are provided on PGD. ICSP uses a special set of commands to control the overall process, combined with standard PIC32MX family of instructions to execute the actual writing of the program memory. PGD also returns data to the external programmer when responding to queries.

Control of the programming process is achieved by manipulating PGC and $\overline{\text{MCLR}}$. Entry into and exit from Programming mode involves applying (or removing) voltage to $\overline{\text{MCLR}}$ while supplying a code sequence to PGD and a clock to PGC. Any one of the PGCx/PGDx pairs can be used to enter programming.

The internal process is regulated by a state machine built into the PIC32MX family of core logic; however, overall control of the process must be provided by the external programming device. Microchip programming devices, such as the MPLAB® PM 3 (used with MPLAB IDE software), include the necessary hardware and algorithms to manage the programming process for PIC32MX family. Users who are interested in a more detailed description, or who are considering designing their own programming interface for PIC32MX family devices, should refer the appropriate PIC32MX family of device programming specification.

33.3.1.2 ENHANCED IN-CIRCUIT SERIAL PROGRAMMING (EICSP)

The Enhanced In-Circuit Serial Programming (EICSP) protocol is an extension of the original ICSP. It uses the same physical interface as the original, but changes the location and execution of programming control to a software application written to the PIC32MX family of devices. Use of Enhanced ICSP results in a significant decrease in overall programming time.

ICSP uses a simple state machine to control each step of the programming process; however, that state machine is controlled by an external programmer. In contrast, Enhanced ICSP uses an on-board bootloader, known as the program executive, to manage the programming process. While overall device programming is still controlled by an external programmer, the program executive manages most of the tasks that must be directly controlled by the programmer in standard ICSP.

The program executive implements its own command set, wider in range than the original ICSP, that can directly erase, program and verify the device program memory. This avoids the need to repeatedly run ICSP command sequences to perform simple tasks. As a result, Enhanced ICSP is capable of programming or reprogramming a device faster than the original ICSP.

The program executive is not preprogrammed into PIC32MX family of devices. If Enhanced ICSP is needed, the user must use standard ICSP to program the executive to the executive memory space in RAM. This can be done directly by the user, or automatically, using a compatible Microchip programming system. After the Programming Executive is written the device can be programmed using Enhanced ICSP.

For additional information on Enhanced ICSP and the program executive, refer to the *"PIC32MX Flash Programming Specification"* (DS61145).

33.3.1.3 EJTAG DEVICE PROGRAMMING USING THE JTAG INTERFACE

The JTAG interface can also be used to program PIC32MX family of devices in their target applications. Using EJTAG with the JTAG interface allows application designers to include a dedicated test and programming port into their applications, with a single 4-pin interface, without imposing the circuit constraints that the ICSP interface may require.

33.3.1.4 ENHANCED JTAG PROGRAMMING USING THE JTAG INTERFACE

Enhanced JTAG programming uses the standard JTAG interface but uses a Programming Executive written to RAM. Use of the Programming Executive with the JTAG interface provides a significant improvement in programming speed.

33.3.2 Debugging

33.3.2.1 ICSP AND IN-CIRCUIT DEBUGGING

ICSP also provides a hardware channel for the In-Circuit Debugger (ICD) which allows externally controlled debugging of software. Using the appropriate hardware interface and software environment, users can force the device to single-step through its code, track the actual content of multiple registers and set software breakpoints.

To use ICD, an external system that supports ICD must load a debugger executive program into the microcontroller. This is automatically handled by many debugger tools, such as the MPLAB IDE. For PIC32MX family of devices, the program is loaded into the last page of the Boot Flash memory space. When not debugging, the application is free to use the last page of Boot Flash memory.

PIC32MX family ICSP supports standard debugging functions including memory and register viewing and modification. Breakpoints can be set and the program execution may be stopped or started. In addition to these functions registers or memory contents can be viewed and modified while the CPU is running.

In contrast with programming, only one of the ICSP ports may be used for ICD. If more than one ICSP port is implemented a Configuration bit determines which port is available. Depending on the particular PIC32MX family of device, there may be two or more ICSP ports that can be selected for this function. The active ICSP debugger port is selected by the ICESEL Configuration bit(s). For information on specific devices, refer to the appropriate device data sheet.

33.3.2.2 EJTAG DEBUGGING

The industry standard EJTAG interface allows Third Party EJTAG tools to be used for debugging. Using the EJTAG interface, memory and registers can be viewed and modified. Breakpoints can be set and the program execution may be stopped, started or single-stepped.

33.3.3 Special Debug Modes for Select Communication Peripherals

To aid in debugging applications certain I/O peripherals have a user controllable bit to override the Freeze function in the peripheral. This allows the module to continue to send any data buffered within the peripheral even when a debugger attempts to halt the peripheral. The Debug mode control bits for these peripherals are contained in the DDPCON register.

33.3.4 JTAG Boundary Scan

As the complexity and density of board designs increases, testing electrical connections between the components on fully assembled circuit boards poses many challenges. To address these challenges, the JTAG developed a method for boundary scan testing that was later standardized as IEEE 1149.1-2001, *"IEEE Standard Test Access Port and Boundary Scan Architecture"*. Since its adoption, many microcontroller manufacturers have added device programming to the capabilities of the test port.

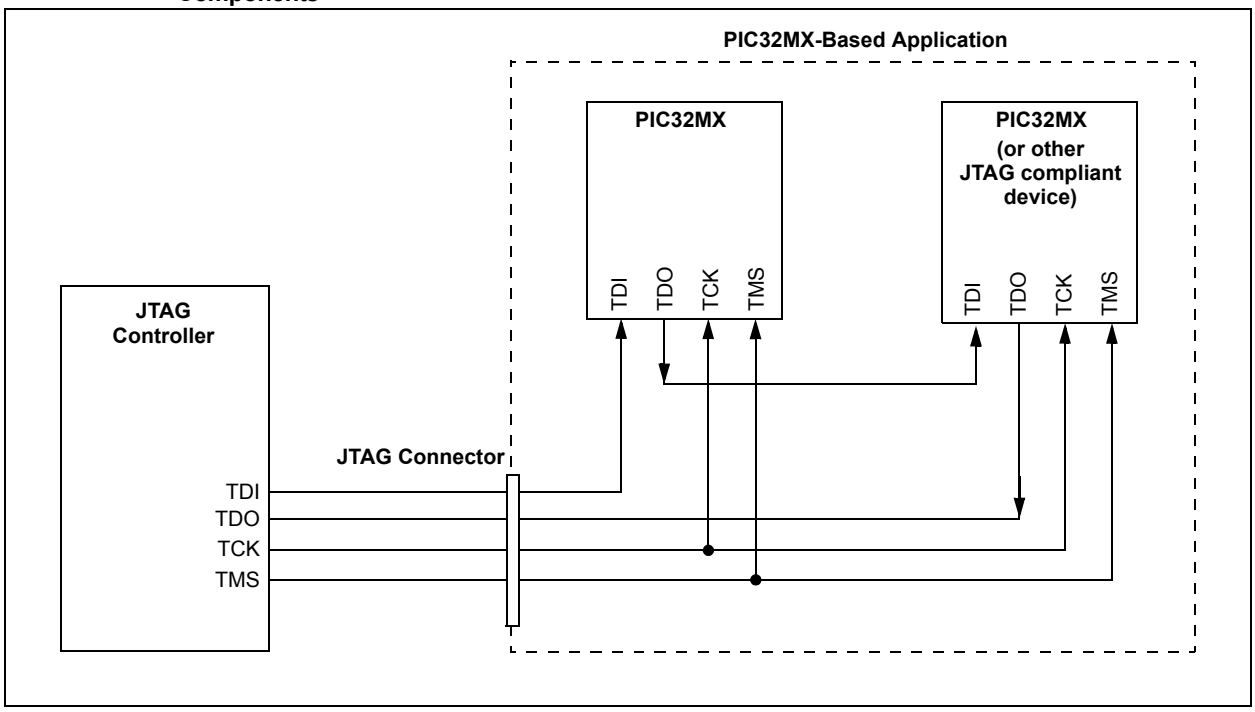
The JTAG boundary scan method is the process of adding a Shift register stage adjacent to each of the component's I/O pins. This permits signals at the component boundaries to be controlled and observed, using a defined set of scan test principles. An external tester or controller provides instructions and reads the results in a serial fashion. The external device also provides common clock and control signals. Depending on the implementation, access to all test signals is provided through a standardized 4-pin interface.

In system-level applications, individual JTAG enabled components are connected through their individual testing interfaces (in addition to their more standard application-specific connections). Devices are connected in a series or daisy-chained fashion, with the test output of one device connected exclusively to the test input of the next device in the chain. Instructions in the JTAG boundary scan protocol allow the testing of any one device in the chain, or any combination of devices, without testing the entire chain. In this method, connections between components, as well as connections at the boundary of the application, may be tested.

A typical application incorporating the JTAG boundary scan interface is shown in Figure 33-3. In this example, a PIC32MX family of microcontroller is daisy-chained to a second JTAG compliant device. Note that the TDI line from the external tester supplies data to the TDI pin of the first device in the chain (in this case, the microcontroller). The resulting test data for this two-device chain is provided from the TDO pin of the second device to the TDO line of the tester.

This section describes the JTAG module and its general use. Users interested in using the JTAG interface for device programming should refer to the *"PIC32MX Flash Programming Specification"* (DS61145) for more information.

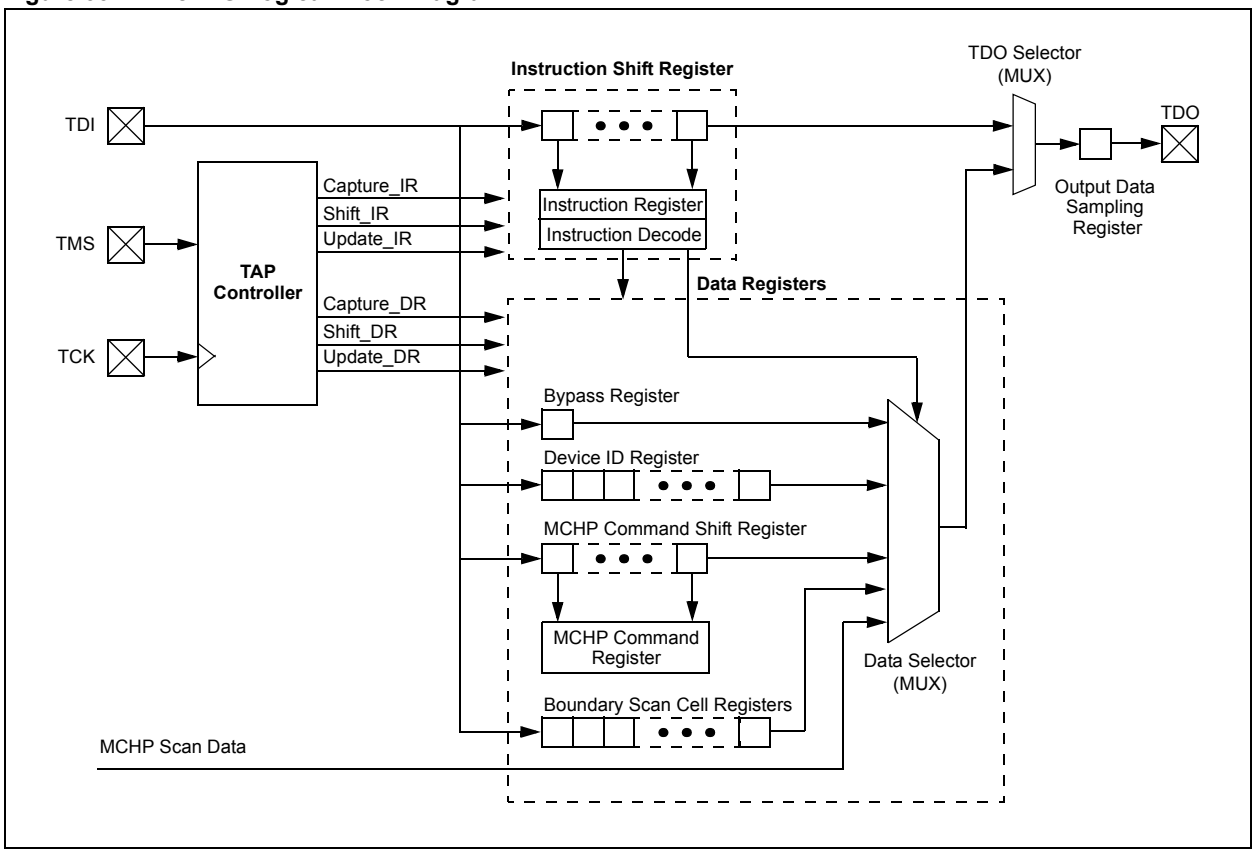
Figure 33-3: Overview of PIC32MX Family-Based JTAG Compliant Application Showing Daisy-Chaining of Components



In PIC32MX family of devices, the hardware for the JTAG boundary scan is implemented as a peripheral module (i.e., outside of the CPU core) with additional integrated logic in all I/O ports. A logical block diagram of the JTAG module is shown in Figure 33-4. It consists of the following key elements:

- Test Access Port (TAP) Interface Pins (TDI, TMS, TCK and TDO)
- TAP Controller
- Instruction Shift Register and Instruction Register (IR)
- Data Registers (DR)

Figure 33-4: JTAG Logical Block Diagram



33.3.4.1 TEST ACCESS PORT (TAP) AND TAP CONTROLLER

The Test Access Port (TAP) on the PIC32MX family of devices is a general purpose port that provides test access to many built-in support functions and test logic defined in IEEE Standard 1149.1. The TAP is enabled by the JTAGEN bit in the DDPCON register. The TAP is enabled, JTAGEN = 1, by default when the device exits Power-on-Reset (POR) or any device Reset. Once enabled, the designated I/O pins become dedicated TAP pins. See the specific PIC32MX device data sheet for details about enabling the JTAG module and identifying JTAG control pins.

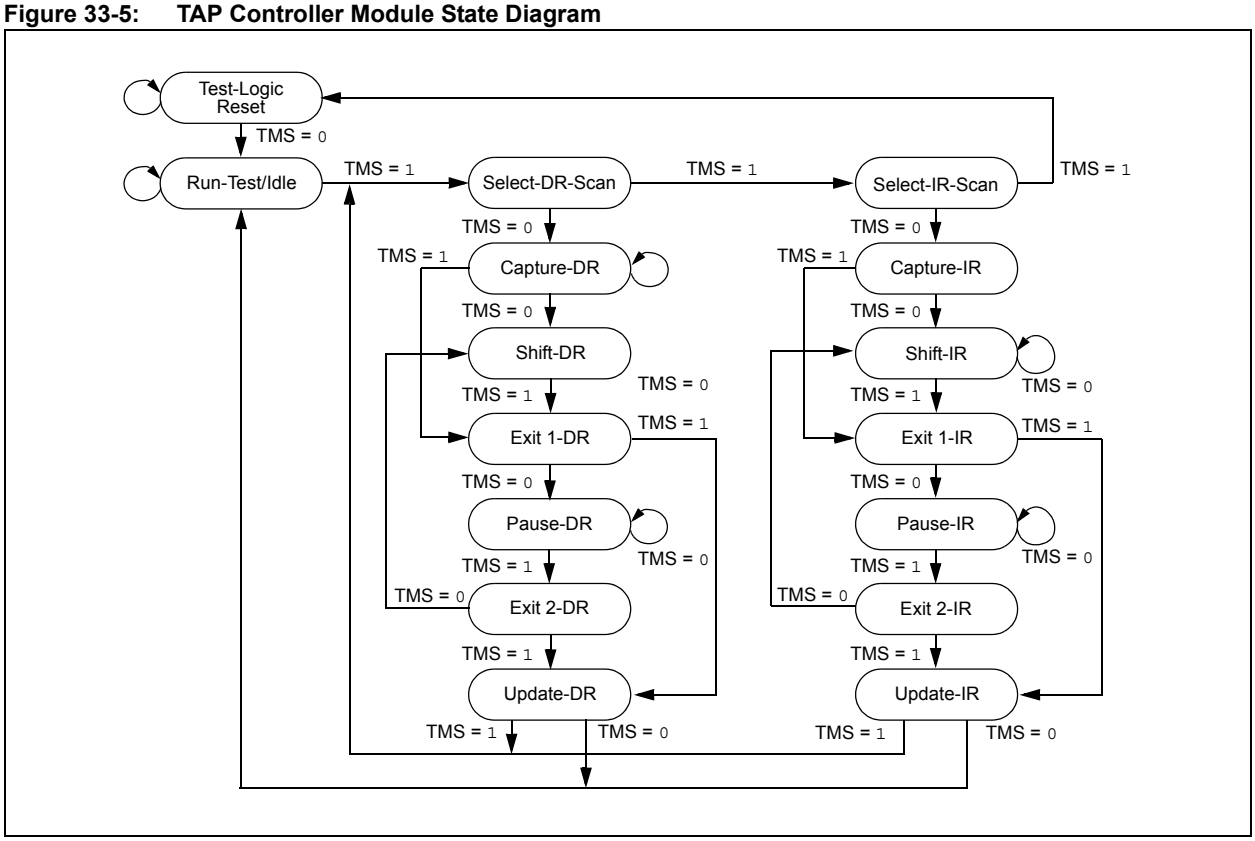
The PIC32MX family implements a 4-pin JTAG interface with these pins:

- TCK (Test Clock Input): Provides the clock for test logic.
- TMS (Test Mode Select Input): Used by the TAP to control test operations.
- TDI (Test Data Input): Serial input for test instructions and data.
- TDO (Test Data Output): Serial output for test instructions and data.

To minimize I/O loss due to JTAG, the optional TAP Reset input pin, specified in the standard, is not implemented on PIC32MX family of devices. For convenience, a “soft” TAP Reset has been included in the TAP controller, using the TMS and TCK pins. To force a port Reset, apply a logic high to the TMS pin for at least 5 rising edges of TCK. Note that device Resets (including POR) do not automatically result in a TAP Reset; this must be done by the external JTAG controller using the soft TAP Reset.

The TAP controller on the PIC32MX family of devices is a synchronous finite state machine that implements the standard 16 states for JTAG. Figure 33-5 shows all the module states of the TAP controller. All Boundary Scan Testing (BST) instructions and test results are communicated through the TAP via the TDI pin in a serial format, Least Significant bit (LSb) first.

Figure 33-5: TAP Controller Module State Diagram



By manipulating the state of TMS and the clock pulses on TCK, the TAP controller can be moved through all of the defined module states to capture, shift and update various instruction and/or data registers. Figure 33-5 shows the state changes on TMS as the controller cycles through its state machine. Figure 33-6 shows the timing of TMS and TCK while transitioning the controller through the appropriate module states for shifting in an instruction. In this example, the sequence shown demonstrates how an instruction is read by the TAP controller.

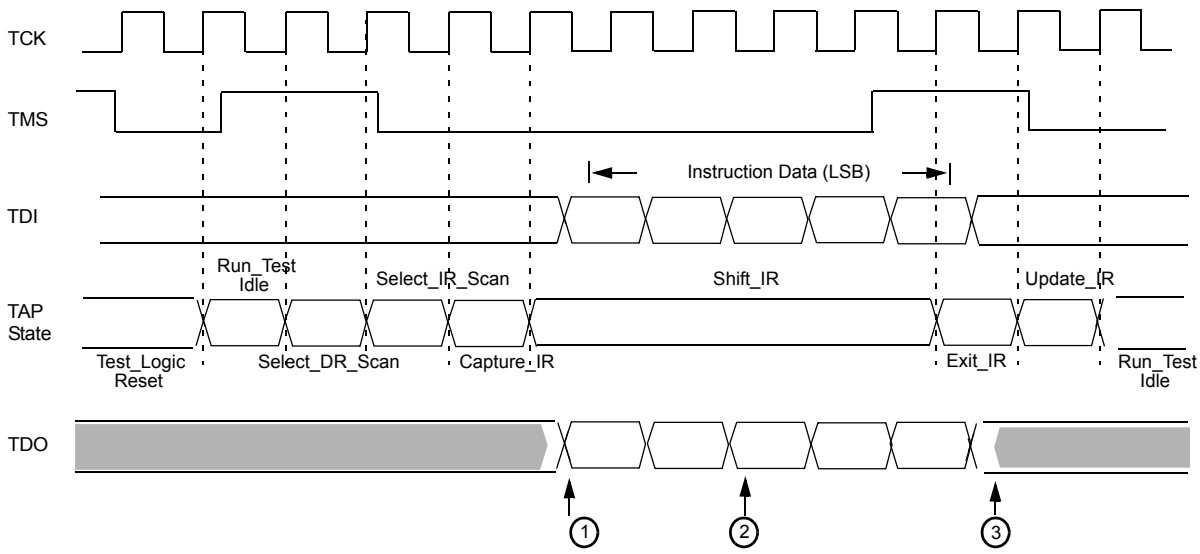
All TAP controller states are entered on the rising edge of the TCK pin. In this example, the TAP controller starts in the Test-Logic Reset state. Since the state of the TAP controller is dependent on the previous instruction, and therefore could be unknown, it is good programing practice to begin in the Test-Logic Reset state.

When TMS is asserted low on the next rising edge of TCK, the TAP controller will move into the Run-Test/Idle state. On the next two rising edges of TCK, TMS is high; this moves the TAP controller to the Select-IR-Scan state.

On the next two rising edges of TCK, TMS is held low; this moves the TAP controller into the Shift-IR state. An instruction is shifted in to the Instruction Shift register via the TDI on the next four rising edges of TCK. After the TAP controller enters this state, the TDO pin goes from a high-impedance state to active. The controller shifts out the initial state of the Instruction Register (IR) on the TDO pin, on the falling edges of TCK, and continues to shift out the contents of the Instruction Register while in the Shift-IR state. The TDO returns to the high-impedance state on the first falling edge of TCK upon exiting the shift state.

On the next three rising edges of TCK, the TAP controller exits the Shift_IR state, updates the Instruction Register and then moves back to the Run-Test/Idle state. Data, or another instruction, can now be shifted in to the appropriate Data or Instruction Register.

Figure 33-6: TAP State Transitions for Shifting in an Instruction



Note 1: TDO pin is always in a high-impedance state, until the first falling edge of TCK, in either the Shift_IR or Shift_DR states.

2: TDO is no longer high-impedance; the initial state of the Instruction Register (IR) is shifted out on the falling edge of TCK.

3: TDO returns to high-impedance again on the first falling edge of TCK in the Exit_IR state.

33.3.4.2 JTAG REGISTERS

The JTAG module uses a number of registers of various sizes as part of its operation. In terms of bit count, most of the JTAG registers are single-bit register cells, integrated into the I/O ports. Regardless of their location within the module, none of the JTAG registers are located within the device data memory space, and cannot be directly accessed by the user in normal operating modes.

33.3.4.2.1 Instruction Shift Register and Instruction Register

The Instruction Shift register is a 5-bit shift register used for selecting the actions to be performed and/or what data registers to be accessed. Instructions are shifted in, Least Significant bit first, and then decoded.

A list and description of implemented instructions is given in **Section 33.3.4.4 "JTAG Instructions"**.

33.3.4.2.2 Data Registers

Once an instruction is shifted in and updated into the Instruction Register, the TAP controller places certain data registers between the TDI and TDO pins. Additional data values can then be shifted into these data registers as needed.

The PIC32MX device family supports three data registers:

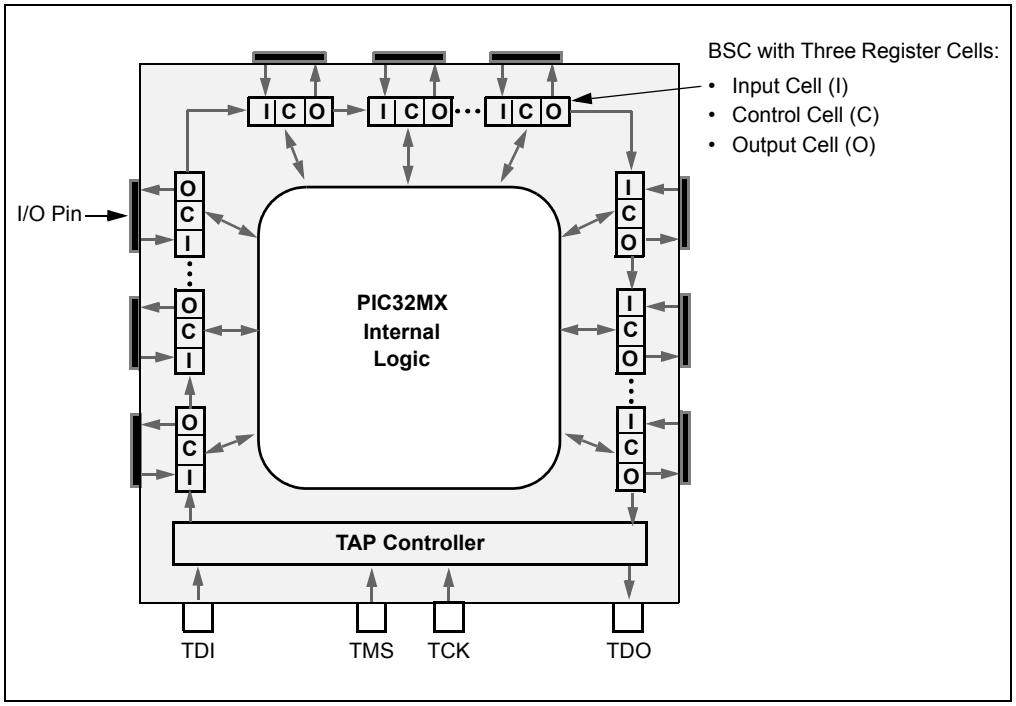
- BYPASS Register: A single-bit register which allows the boundary scan test data to pass through the selected device to adjacent devices. The BYPASS register is placed between the TDI and TDO pins when the `BYPASS` instruction is active.
- Device ID Register: A 32-bit part identifier. It consists of an 11-bit manufacturer ID assigned by the IEEE (29h for Microchip Technology), device part number and device revision identifier. When the `IDCODE` instruction is active, the device ID register is placed between the TDI and TDO pins. The device data ID is then shifted out on to the TDO pin, on the next 32 falling edges of TCK, after the TAP controller is in the Shift_DR.
- MCHP Command Shift Register: An 8-bit shift register that is placed between the TDI and TDO pins when the `MCHP_CMD` instruction is active. This shift register is used to shift in Microchip commands.

33.3.4.3 BOUNDARY SCAN REGISTER (BSR)

The BSR is a large shift register that is comprised of all the I/O Boundary Scan Cells (BSCs), daisy-chained together (Figure 33-7). Each I/O pin has one BSC, each containing 3 BSC registers: an input cell, an output cell and a control cell. When the `SAMPLE/PRELOAD` or `EXTEST` instructions are active, the BSR is placed between the TDI and TDO pins, with the TDI pin as the input and the TDO pin as the output.

The size of the BSR depends on the number of I/O pins on the device. For example, the 100-pin PIC32MX general purpose parts have 82 I/O pins. With 3 BSC registers for each of the 82 I/Os, this yields a Boundary Scan register length of 244 bits. This is due to the $\overline{\text{MCLR}}$ pin being an input-only BSR cell. Information on the I/O port pin count of other PIC32MX family of devices can be found in their specific device data sheets.

Figure 33-7: Daisy-Chained Boundary Scan Cell Registers on a PIC32MX Family of Microcontroller



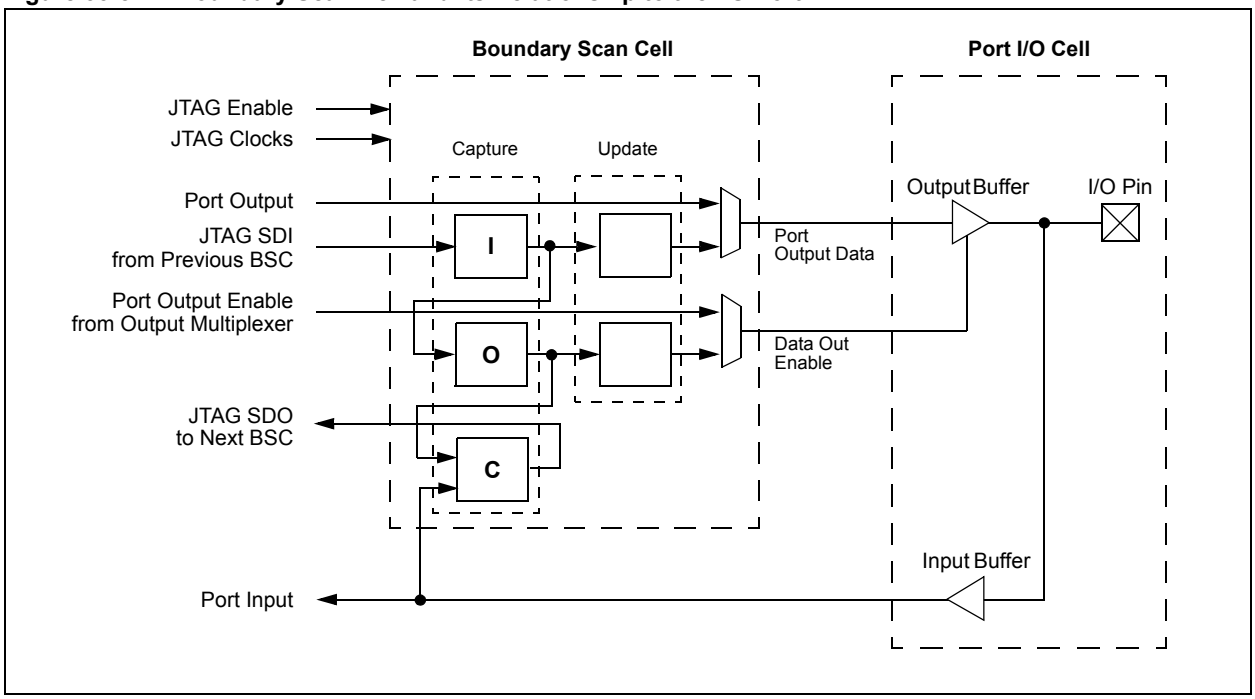
33.3.4.3.1 Boundary Scan Cell (BSC)

The function of the BSC is to capture and override I/O data values when JTAG is active. The BSC consists of three Single-Bit Capture register cells and two Single-Bit Holding register cells. The capture cells are daisy-chained to capture the port's input, output and control (output-enable) data, as well as pass JTAG data along the Boundary Scan register. Command signals from the TAP controller determine if the port of JTAG data is captured, and how and when it is clocked out of the BSC.

The first register either captures internal data destined to the output driver, or provides serially scanned in data for the output driver. The second register captures internal output-enable control from the output driver and also provides serially scanned in output-enable values. The third register captures the input data from the I/O's input buffer.

Figure 33-8 shows a typical BSC and its relationship to the I/O port's structure.

Figure 33-8: Boundary Scan Cell and Its Relationship to the I/O Port



33.3.4.4 JTAG INSTRUCTIONS

PIC32MX family of devices support the mandatory instruction set specified by IEEE 1149.1, as well as several optional public instructions defined in the specification. These devices also implement instructions that are specific to Microchip devices.

The mandatory JTAG instructions are:

- `BYPASS` (0x1F): Used for bypassing a device in a test chain; this allows the testing of off-chip circuitry and board-level interconnections.
- `SAMPLE/PRELOAD` (0x02): Captures the I/O states of the component, providing a snapshot of its operation.
- `EXTEST` (0x06): Allows the external circuitry and interconnections to be tested, by either forcing various test patterns on the output pins, or capturing test results from the input pins.

Microchip has implemented optional JTAG instructions and manufacturer-specific JTAG commands in PIC32MX family of devices. Please refer to Tables 33-3, 33-4, 33-5 and 33-6.

Table 33-3: JTAG Commands

OPCODE	Name	Device Integration
0x1F	Bypass	Bypasses device in test chain
0x00	HIGHZ	Places device in a high-impedance state, all pins are forced to inputs
0x01	ID Code	Shifts out the device's ID code
0x02	Sample/Preload	Samples all pins or loads a specific value into output latch
0x06	EXTEST	Boundary Scan

Table 33-4: Microchip TAP IR Commands

OPCODE	Name	Device Integration
0x01	MTAP_IDCODE	Shifts out the devices ID code
0x07	MTAP_COMMAND	Configure Microchip TAP controller for DR commands
0x04	MTAP_SW_MTAP	Select Microchip TAP controller
0x05	MTAP_SW_ETAP	Select EJTAG TAP controller

Table 33-5: Microchip TAP 8-bit DR Commands

OPCODE	Name	Device Integration
0x00	MCHP_STATUS	Perform NOP and return Status
0xD1	MCHP_ASERT_RST	Request Assert Device Reset
0xD0	MCHP_DE_ASSERT_RST	Request De-Assert Device Reset
0xFC	MCHP_ERASE	Perform a Chip Erase
0xFE	MCHP_FLASH_ENABLE	Enables fetches and loads to the Flash from the CPU
0xFD	MCHP_FLASH_DISABLE	Disables fetches and loads to the Flash from the CPU
0xFF	MCHP_READ_CONFIG	Forces device to reread the configuration settings and initialize accordingly

Table 33-6: EJTAG Commands

OPCODE	Name	Device Integration	Data Length for the Following DR
0x00	—	Not Used	—
0x01	IDCODE	Selects the Devices ID Code register	32 bits
0x02	—	Not Used	—
0x03	IMPCODE	Selects Implementation Register	—
0x04(2)	MTAP_SW_MTAP	Select Microchip TAP controller	—
0x05(2)	MTAP_SW_ETAP	Select EJTAG TAP controller	—
0x06-0x07	—	Not Used	—
0x08	ADDRESS	Selects the Address Register	32 bits
0x09	DATA	Selects the Data Register	32 bits
0x0A	CONTROL	Selects the EJTAG control register	32 bits
0x0B	ALL	Selects the Address, Data, EJTAG control register	96 bits
0x0C	EJTAGBOOT	Forces the CPU to take a Debug Exception after boot	1-bit
0x0D	NORMALBOOT	Makes the CPU execute the reset handler after a boot	1-bit
0x0E	FASTDATA	Selects the Data and Fast Data Registers	1-bit
0x0F-0x1B	—	Reserved	—
0x1C-0xFE	—	Not Used	—
0xFF	—	Select the Bypass Register	—

Note 1: For complete information about EJTAG commands and protocol, refer to the EJTAG Specification, which is available from the MIPS Technologies web site (www.mips.com).

2: This opcode is not an EJTAG command, but is recognized by the Microchip implementation.

33.3.5 Boundary Scan Testing (BST)

Boundary Scan Testing (BST) is the method of controlling and observing the boundary pins of the JTAG compliant device, like those of the PIC32MX family, utilizing software control. BST can be used to test connectivity between devices by daisy-chaining JTAG compliant devices to form a single scan chain. Several scan chains can exist on a PCB to form multiple scan chains. These multiple scan chains can then be driven simultaneously to test many components in parallel. Scan chains can contain both JTAG compliant devices and non-JTAG compliant devices.

A key advantage of BST is that it can be implemented without physical test probes; all that is needed is a 4-wire interface and an appropriate test platform. Since JTAG boundary scan has been available for many years, many software tools exist for testing scan chains without the need for extensive physical probing. The main drawback to BST is that it can only evaluate digital signals and circuit continuity; it cannot measure input or output voltage levels or currents.

33.3.5.1 RELATED JTAG FILES

To implement BST, all JTAG test tools will require a Boundary Scan Description Language (BSDL) file. BSDL is a subset of VHDL (VHSIC Hardware Description Language), and is described as part of IEEE Std. 1149.1. The device-specific BSDL file describes how the standard is implemented on a particular device and how it operates.

The BSDL file for a particular device includes the following:

- The pinout and package configuration for the particular device
- The physical location of the TAP pins
- The Device ID register and the device ID
- The length of the Instruction Register
- The supported BST instructions and their binary codes
- The length and structure of the Boundary Scan register
- The boundary scan cell definition

The name for each BSDL file is the device name and silicon revision. For example, `PIC32MX320F128L_A2.BSD` is the BSDL file for the PIC32MX320F128L, silicon revision A2.

33.4 INTERRUPTS

Programming and Debugging operations are not performed during code execution and are therefore not affected by interrupts. Trace Operations will report the change in code execution when an interrupt occurs but the Trace Controller is not affected by interrupts.

33.5 I/O PINS

In order to interface to the numerous Programming and Debugging options available and still provide peripherals access to the pins, the Programming and Debugging I/O pins are multiplexed with peripheral pins. Table 33-7 describes the function of the Programming and Debug related pins.

Table 33-7: Programming and Debugging Pin Functions

Pin Name	Function				Description
	Program Mode	Debug Mode	Trace Mode	Boundary Scan Mode	
$\overline{\text{MCLR}}$	$\overline{\text{MCLR}}$	$\overline{\text{MCLR}}$	$\overline{\text{MCLR}}$	$\overline{\text{MCLR}}$	Master Clear, used to enter ICSP™ mode and to override JTAGEN (DDPCON<3>)
PGC1	PGC1/ Alternate	PGC1/ Alternate	PGC1/ Alternate	Alternate	ICSP Clock, determined by ICESEL Configuration bit (DEVCFG0<3>)
PGD1	PGD1/ Alternate	PGD1/ Alternate	PGD1/ Alternate	Alternate	ICSP Data, determined by ICESEL (DEVCFG0<3>) and DEBUG Configuration bits (DEVCFG0<1:0>)
PGC2	PGC2/ Alternate	PGC2/ Alternate	PGC2/ Alternate	Alternate	Alternate ICSP Clock, determined by ICESEL (DEVCFG0<3>) and DEBUG Configuration bits (DEVCFG0<1:0>)
PGD2	PGD2/ Alternate	PGD2/ Alternate	PGD2/ Alternate	Alternate	Alternate ICSP Data, determined by ICESEL (DEVCFG0<3>) and DEBUG Configuration bits (DEVCFG0<1:0>)
TCK	TCK	TCK	TCK	TCK	JTAG Clock, determined by JTAGEN control bit (DDPCON<3>)
TDO	TDO	TDO	TDO	TDO	JTAG Data Out, determined by JTAGEN control bit (DDPCON<3>)
TDI	TDI	TDI	TDI	TDI	JTAG Data in, determined by JTAGEN control bit (DDPCON<3>)
TMS	TMS	TMS	TMS	TMS	JTAG Test Mode Select, determined by JTAGEN control bit (DDPCON<3>)
TRCLK	Alternate	Alternate	TRCLK	Alternate	Trace Clock, determined by TROEN control bit (DDPCON<2>)
TRD0	Alternate	Alternate	TRD0	Alternate	Trace Data, determined by TROEN control bit (DDPCON<2>)
TRD1	Alternate	Alternate	TRD1	Alternate	Trace Data, determined by TROEN control bit (DDPCON<2>)
TRD2	Alternate	Alternate	TRD2	Alternate	Trace Data, determined by TROEN control bit (DDPCON<2>)
TRD3	Alternate	Alternate	TRD3	Alternate	Trace Data, determined by TROEN control bit (DDPCON<2>)

33.6 OPERATION IN POWER-SAVING MODES

The PIC32MX family must be awake for all programming and debugging operations.

33.7 EFFECTS OF RESETS

33.7.1 Device Reset

A device Reset by asserting $\overline{\text{MCLR}}$ while in ICSP mode will force the ICSP to exit. Asserting $\overline{\text{MCLR}}$ will force an exit from EJTAG mode.

33.7.2 Watchdog Timer Reset

A Watchdog Timer (WDT) Reset during Erase will not abort the Erase cycle. The WDT event flag will be set to show that a WDT Reset has occurred.

A WDT Reset during an EJTAG session will reset the TAP controller to the Microchip TAP controller.

A WDT Reset during Programming will abort the programming sequence.

33.8 APPLICATION IDEAS

For implementation of ICSP programming, refer to the *"PIC32MX Flash Programming Specification"* (DS61145).

33.9 RELATED APPLICATION NOTES

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the PIC32MX family of devices, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the programming and diagnostics are:

Title	Application Note #
No related application notes at this time.	N/A

Note: Please visit the Microchip web site (www.microchip.com) for additional application notes and code examples for the PIC32MX family of devices.

33.10 REVISION HISTORY

Revision A (September 2007)

This is the initial released version of this document.

Revision B (October 2007)

Updated document to remove Confidential status.

Revision C (April 2008)

Revised status to Preliminary; Revised U-0 to r-x.

Revision D (June 2008)

Add Note to Section 33.3.1; Revised Section 33.3.2.1; Revised Table 33-7; Change Reserved bits from “Maintain as” to “Write”.

Revision E (August 2009)

This revision includes the following updates:

- Minor updates to text and formatting have been incorporated throughout the document.
- Added the FJTAGEN bit and removed bits DDPU1, DDPU2 and DDPSPI1 from Table 33-2: Programming and Diagnostics SFR Summary.
- Added FJTAGEN bit and removed bits DDPUSB, DDPU1, DDPU2 and DDPSPI1 from Register 33-1: DDPCON: Debug Data Port Control Register.

NOTES: