Section 24. Programming and Diagnostics

HIGHLIGHTS

This section of the manual contains the following topics:

24.1 INTRODUCTION

PIC24H 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 interfaces
- Enhanced debugging capabilities
- Boundary scan testing for device and board diagnostics

PIC24H devices incorporate three different programming and diagnostic modalities that provide a range of functions useful to the application developer. They are summarized in Table 24-1.

Table 24-1: Comparison of PIC24H Programming and Diagnostic Features

Feature	Interface	Device Integration	Functions
In-Circuit Serial Programming™ (ICSP™) programming method	PGCx and PGDx pins	Integrated with device core	Programming, debugging
Enhanced ICSP programming method	PGCx and PGDx pins	Hardware integrated with device core; firmware-based control	Programming
Joint Test Action Group (JTAG)	TDI, TDO, TMS and TCK pins	Peripheral to device core; partly integrated with I/O logic	Programming, Boundary Scan Testing (BST) diagnostics

24.2 IN-CIRCUIT SERIAL PROGRAMMING™

The In-Circuit Serial Programming (ICSP™) programming capability is Microchip's proprietary process for microcontroller programming in the target application. Originally introduced for 8-bit PIC16 devices, this method is used for virtually all Microchip microcontrollers. ICSP is the most direct method to program the device, whether the controller is embedded in a system or loaded into a device programmer.

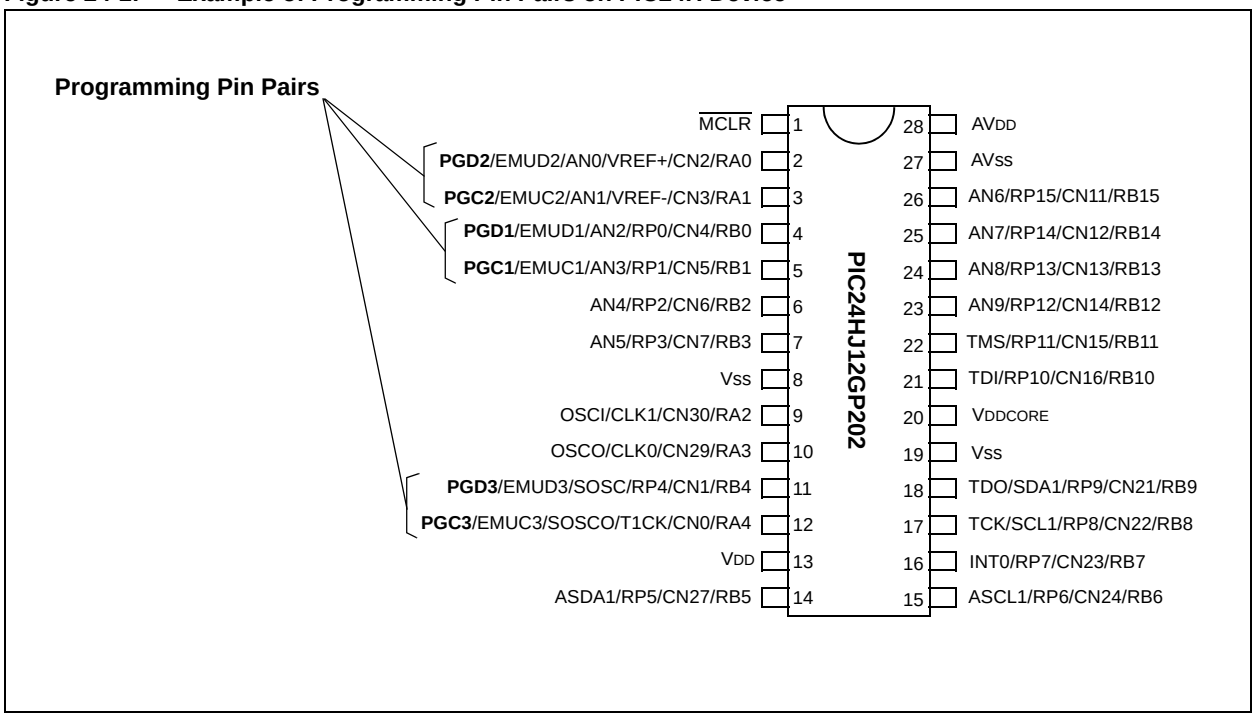
24.2.1 ICSP Interface

The ICSP interface uses two pins as its core. The Programming Data (PGD) pin 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 (PGC) pin clocks in data and controls the overall process.

Most PIC24H devices have more than one pair of PGC and PGD pins; these are multiplexed with other I/O or peripheral functions as shown in Figure 24-1. 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 you to incorporate ICSP on the pair of pins least constrained by the circuit design. All PGCx and PGDx pins are functionally tied together and behave identically. 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 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 programming device.

Figure 24-1: Example of Programming Pin Pairs on PIC24H Device



24.2.2 ICSP Operation

ICSP mode uses a combination of internal hardware and external control to program the target device. Programming data and instructions are provided on the PGD pin. A special set of 4-bit commands, combined with standard PIC24H instructions, controls the overall process of writing to the program memory. The PGD pin also returns data to the external programmer in response to queries.

The programming process is controlled by manipulating the PGC and $\overline{MCLR}$ pins. Entry into and exit from ICSP mode involves applying (or removing) voltage to $\overline{MCLR}$, while supplying a code sequence to PGD and a clock to PGC.

Any one of the PGCx/PGDx pairs can be used for programming. During programming, the clock train on PGC is also used to indicate the difference between 4-bit commands, programming control commands, and payload data to be programmed.

The internal process is regulated by a state machine built into the PIC24H core logic; however, overall control of the process must be provided by the external programming device. Microchip programming devices, such as MPLAB® PM 3 (used with MPLAB IDE development software), include the necessary hardware and algorithms to manage the programming process for PIC24H devices. Users who are interested in a more detailed description, or who are considering designing their own programming interface for a PIC24H device, should consult the *"dsPIC33F/PIC24H Programming Specification"* (DS70152).

24.2.3 ICSP and In-Circuit Debugging

The ICSP method also provides a hardware channel for in-circuit debugging, which allows external control of software debugging. Using the appropriate hardware interface and software environment, you can force the device to single-step through its code, track the actual content of multiple registers, and set software breakpoints.

To use in-circuit debugging, an external system must load a debugger executive program into the microcontroller. This task is handled automatically by many debugging tools, such as MPLAB ICD 2. For PIC24H devices, the program is loaded into the executive program memory in the configuration memory space. Although memory is implemented and code can be executed from these locations, the executive memory space is not available to the user application during normal operating modes. For details, refer to the *"dsPIC33F/PIC24H Programming Specification"* (DS70152).

Because of the memory location, use of the debugger executive has no impact on the size of the application being examined. The executive memory space allows use of the entire program memory for program code, without needing to leave reserve space for application debugging. In addition, its use means that the program memory content in normal and debug states is identical, which helps to simplify troubleshooting.

Depending on the particular PIC24H device, one or more ICSP ports can be used for programming. However, only one of these ICSP ports can be used for in-circuit debugging. Use the
following process to select which part to activate for debugging via your MPLAB IDE setup:

1. In the MPLAB IDE click *Configure > Configuration Bits* menu to display the Configurations Bits window.
2. In the Configuration Bits window select the appropriate debug pair setting under the Comm Channel Select Category.

Note:	For details on the configuration memory space, refer to the *"dsPIC33F/PIC24H Flash Programming Specifications" (DS-70152*).

24.3 ENHANCED IN-CIRCUIT SERIAL PROGRAMMING

The Enhanced ICSP protocol is an extension of the ICSP method. Enhanced ICSP uses the same physical interface as the original, but changes the location and execution of programming control.

ICSP mode uses a simple state machine to control each step of the programming process; however, the 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 overseen by an external programmer, the program executive manages most of the things 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 microcontroller's program memory. This avoids the need to repeatedly run ICSP command sequences to perform simple tasks. As a result, Enhanced ICSP mode can program or reprogram a device more quickly than ICSP mode.

Like the in-circuit debugger executive, the program executive does not reside in the user application program memory space. It is also loaded into the executive program memory. Since the debugger and Enhanced ICSP executives both use this memory space, in-circuit debugging is not available while Enhanced ICSP mode is being used for programming.

The program executive is not preprogrammed into PIC24H devices. If you need Enhanced ICSP, you must use standard ICSP to program the executive to the executive memory space. You can set this up directly in your software, or automatically using a compatible Microchip programming system.

For additional information on Enhanced ICSP and the program executive, refer to the *"dsPIC33F/PIC24H Programming Specification"* (DS70152).

24.4 JTAG BOUNDARY SCAN

As the complexity and density of board designs increase, testing electrical connections between the components on fully-assembled circuit boards poses many challenges. To address these challenges, the Joint Test Action Group (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.

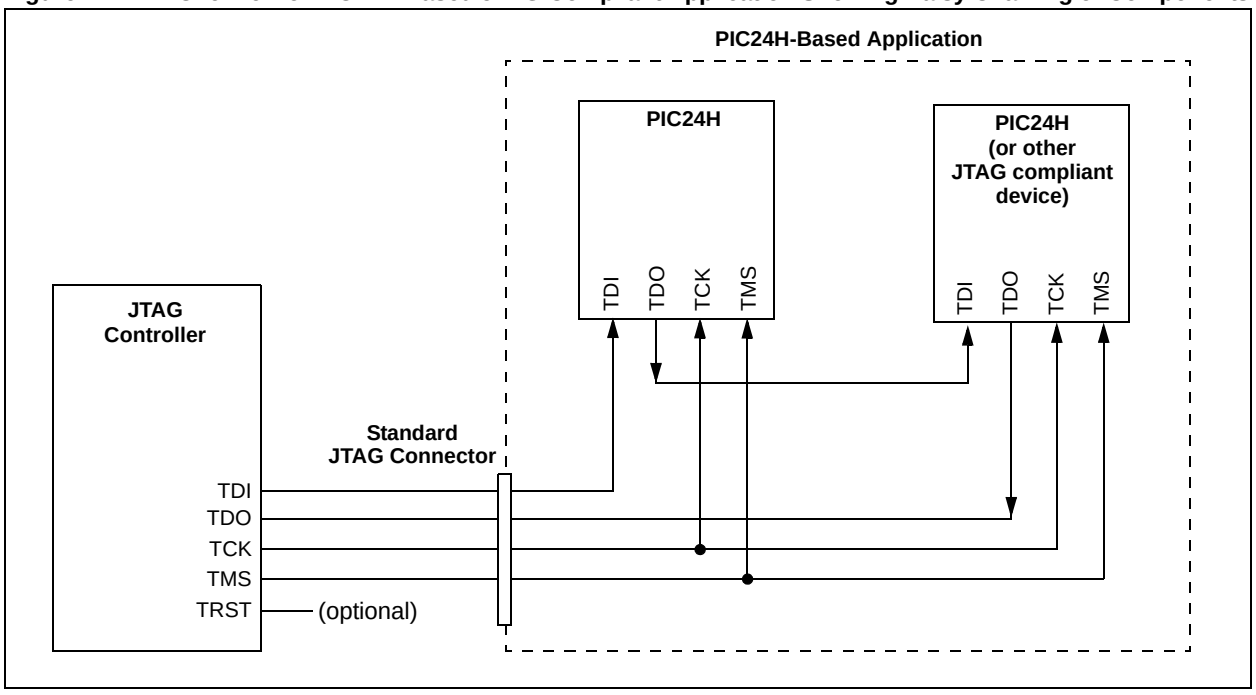
The JTAG boundary scan method adds a shift register stage adjacent to each of the component's I/O pins, which permits signals at the component boundaries to be controlled and observed using a defined set of scan test principles. An external tester or controller provides instructions and reads the results serially. 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 or 5-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, can be tested.

Figure 24-2 shows a typical application incorporating the JTAG boundary scan interface. In this example, a PIC24H Digital Signal Controller (DSC) is daisy-chained to a second JTAG compliant device. The Test Data Input (TDI) line from the external tester supplies data to the Test Data Input (TDI) pin of the first device in the chain (in this case, the DSC). The resulting test data for this two-device chain is provided from the Test Data Output (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 **Section 24.4.6 "JTAG Device Programming"**.

Figure 24-2: Overview of PIC24H-Based JTAG Compliant Application Showing Daisy Chaining of Components



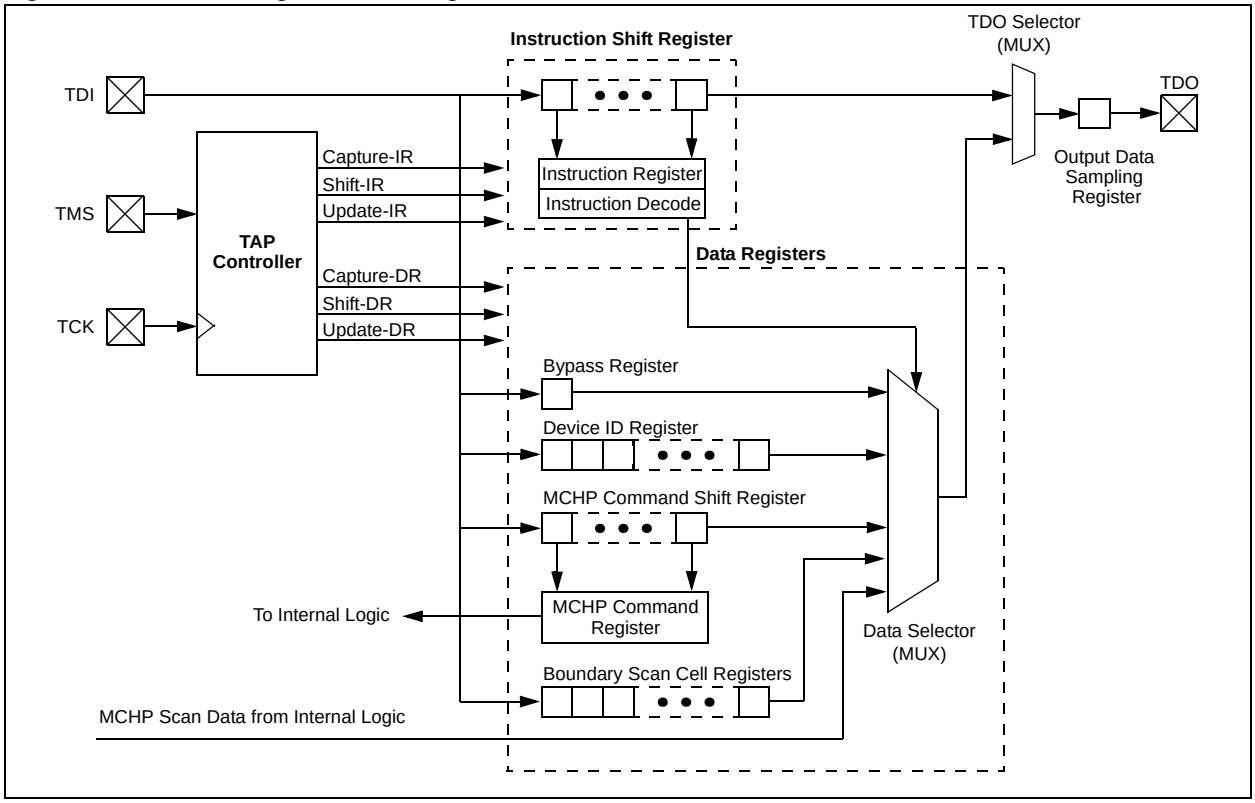
In the PIC24H device family, the hardware for the JTAG boundary scan is implemented as a peripheral module (i.e., outside of the CPU core) with additional integrated logic in all I/O ports. The PIC24H device family implements a 4-pin JTAG interface (refer to Table 24-2).

Interface Pin	Function
Test Clock Input (TCK)	Provides the clock for test logic
Test Mode Select Input (TMS)	Used by the Test Access Port (TAP) to control test operations
Test Data Input (TDI)	Serial input for test instructions and data
Test Data Output (TDO)	Serial output for test instructions and data

A logical block diagram of the JTAG module is shown in Figure 24-3. It consists of the following key elements:

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

Figure 24-3: JTAG Logical Block Diagram



24.4.1 Test Access Port (TAP) and TAP Controller

The Test Access Port (TAP) on the PIC24H device family 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 controller and the associated boundary scan pins are disabled by programming the JTAG Enable (JTAGEN) bit to '0' in Configuration register (FICD). The TAP controller, by default, is enabled in the bit's unprogrammed state. While enabled, the designated I/O pins become dedicated TAP pins. Use the following process to enable or disable the JTAG port via your MPLAB IDE setup:

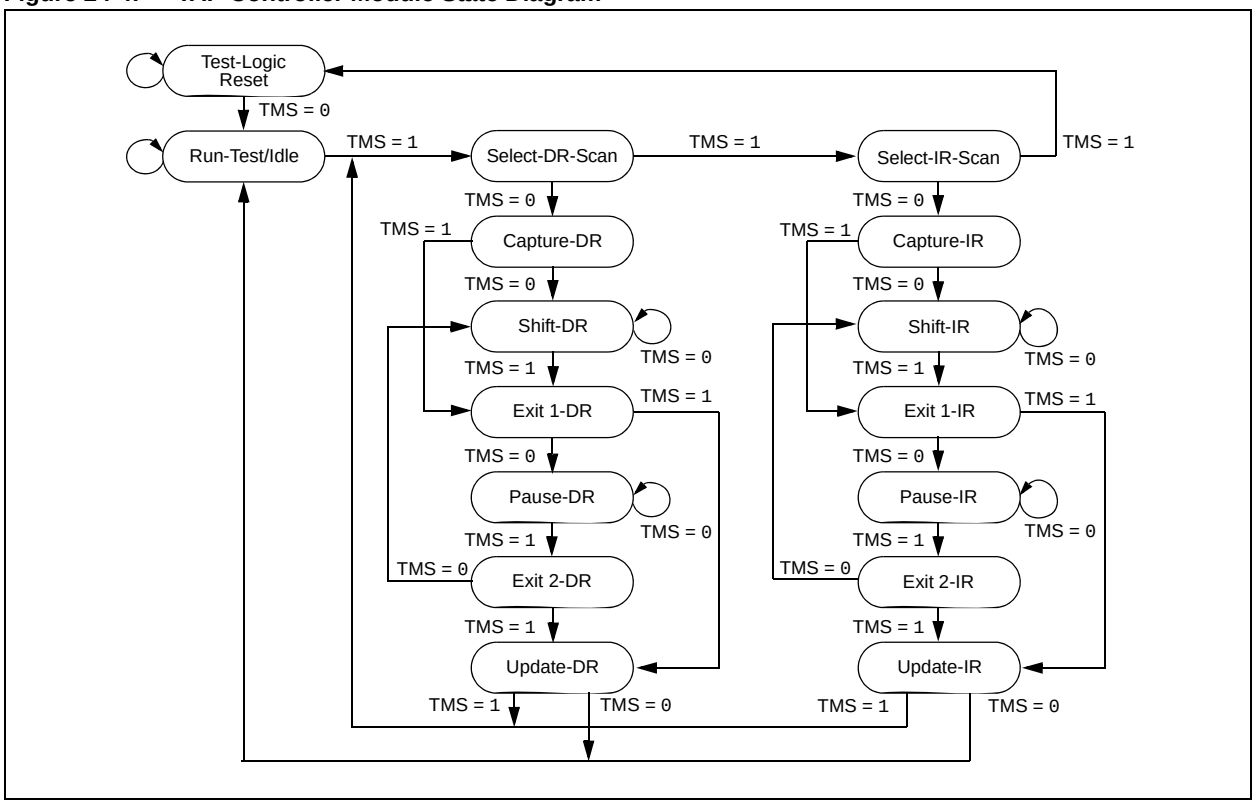
1. In the MPLAB IDE click *Configure > Configuration Bits* menu to display the Configurations Bits window.
2. In the Configuration Bits window select the Enable/Disable setting under the JTAG Port Enable Category.

Note:	For information on the FICD register, refer to the "*dsPIC33F/PIC24H Flash Programming Specifications*" (DS70152).

To minimize I/O loss due to JTAG scans, the optional TAP Reset ($\overline{TRST}$) input pin, specified in the standard, is not implemented on PIC24H devices. For convenience, a "soft" TAP Reset is 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. 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 PIC24H family devices is a synchronous finite state machine that implements the standard 16 states for JTAG scans. Figure 24-4 shows all the module states of the TAP controller. All Boundary Scan Test (BST) instructions and test results are communicated through the TAP via the TDI pin in a serial format, Least Significant bit first.

Figure 24-4: 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 24-4 shows the state changes on TMS as the controller cycles through its state machine. Figure 24-5 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 demonstrates how a TAP controller reads an instruction.

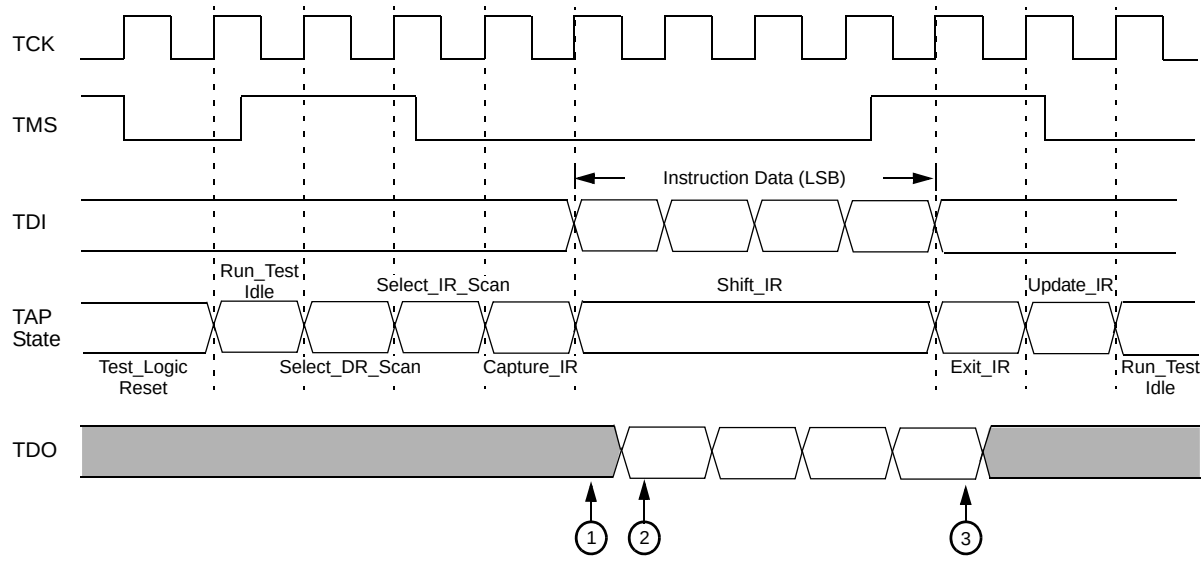
All TAP controller states are entered on the rising edge of the signal on the TCK pin. The TAP controller starts in the Test-Logic Reset state. Since the state of the TAP controller is dependent on the previous instruction, and therefore could be unknown, it is good programing practice to begin in the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

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

On the next two rising edges of TCK, TMS is held low, which 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 IR 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 IR and then moves back to the Run-Test/Idle state. Data, or another instruction, can now be shifted in to the appropriate data or IR.

Figure 24-5: 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.

24.4.2 JTAG Registers

The JTAG module uses a number of registers of various sizes as part of its operation. None of the JTAG registers are located within the device data memory space. They cannot be directly accessed by the user application in normal operating modes.

24.4.2.1 INSTRUCTION SHIFT REGISTER AND INSTRUCTION REGISTER

The 4-bit IR allows an instruction to be shifted into the device. The instruction selects the data register to access.

The parallel output from the Instruction register is latched to protect from the transient data patterns that occur in its shift register stages as new instruction data is entered. The latched parallel output is controlled, so that it can change state only in the Update-IR and Test-Logic-Reset controller states.

A list and description of implemented instructions is provided in **Section 24.4.4 "JTAG Instructions"**.

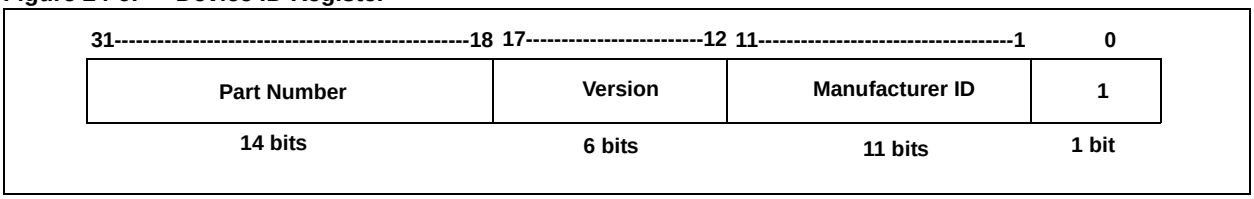
24.4.2.2 DATA REGISTERS

The PIC24H device family supports the JTAG data registers listed in Table 24-2.

Table 24-2: JTAG Data Registers

Register	Function
Bypass Register	Provides a minimum-length serial path for the movement of test data between TDI and TDO. This path can be selected when no other test data register needs to be accessed during a board-level test operation. Use of the Bypass register in a component speeds access to test data registers in other components on a board-level test data path.
Microchip Command Shift Register	This 8-bit shift register shifts in Microchip device-specific commands. The parallel output from the shift register is latched to protect from the transient data patterns that occur in its shift register stages as a new command is entered.
Device ID Register	This 32-bit device IR allows the manufacturer, part number, and variant of a component to be determined. The bit format of the PIC24H device is shown in Figure 24-6. It consists of an 11-bit manufacturer ID assigned by the IEEE (29h for Microchip Technology), device part number, and device revision number. For example, the JTAG ID for a PIC24HJ64GP206 device is: • Manufacturer ID = 0x29 • Part number = 0X41 • Silicon revision = A2 • JTAG ID = 0x01042053
Boundary Scan Register	Consists of a number of cells combined to form a single shift-register-based path that is connected between TDI and TDO when an appropriate instruction is selected.

Figure 24-6: Device ID Register



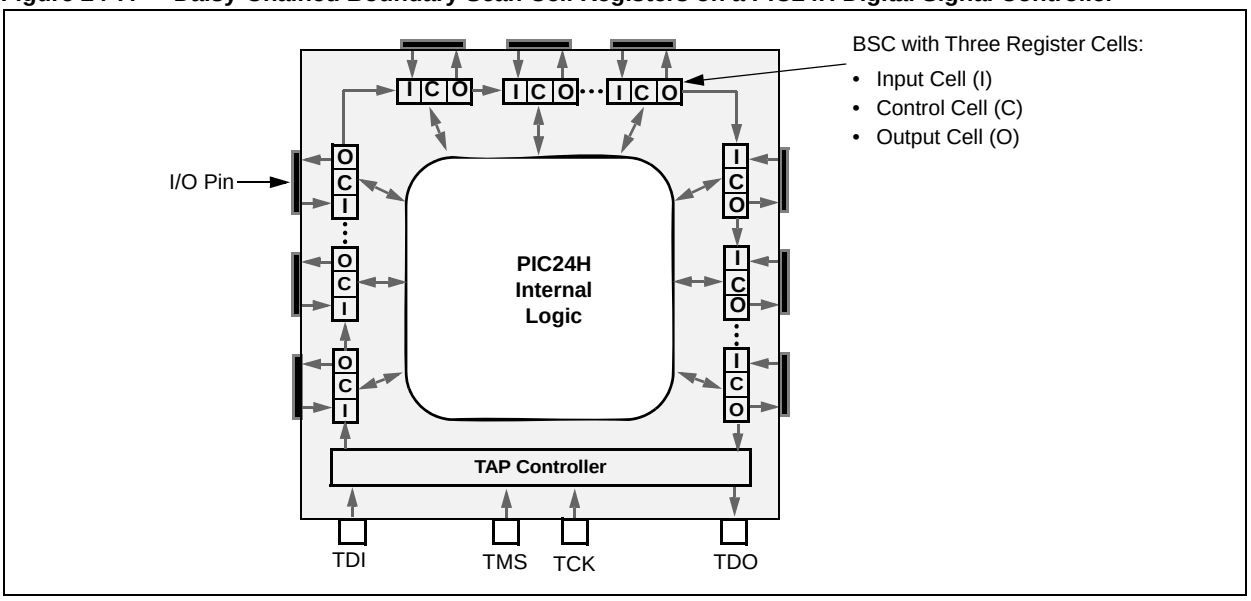
24.4.3 Boundary Scan Register

The Boundary Scan Register (BSR) is a large shift register that consists of all the I/O Boundary Scan Cells daisy-chained together, as shown in Figure 24-7. Each I/O pin has one Boundary Scan Cell (BSC). Each BSC contains three BSC registers: an input cell register, an output cell register and a control cell register. 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 PIC24HJ64GP206 has 50 I/O pins. Three BSC registers for each of the 50 I/Os yields a Boundary Scan Register length of 150 bits. Information on the I/O port pin count for a specific device is found in the specific BSDL files.

Note:	The Boundary Scan Cell is not used for power supply pins (VDD, VDDCORE, VSS, AVDD, AVSS). The pins that have the JTAG interconnect function and JTAG control are not part of the scan-chain and are not JTAG testable.

Figure 24-7: Daisy-Chained Boundary Scan Cell Registers on a PIC24H Digital Signal Controller



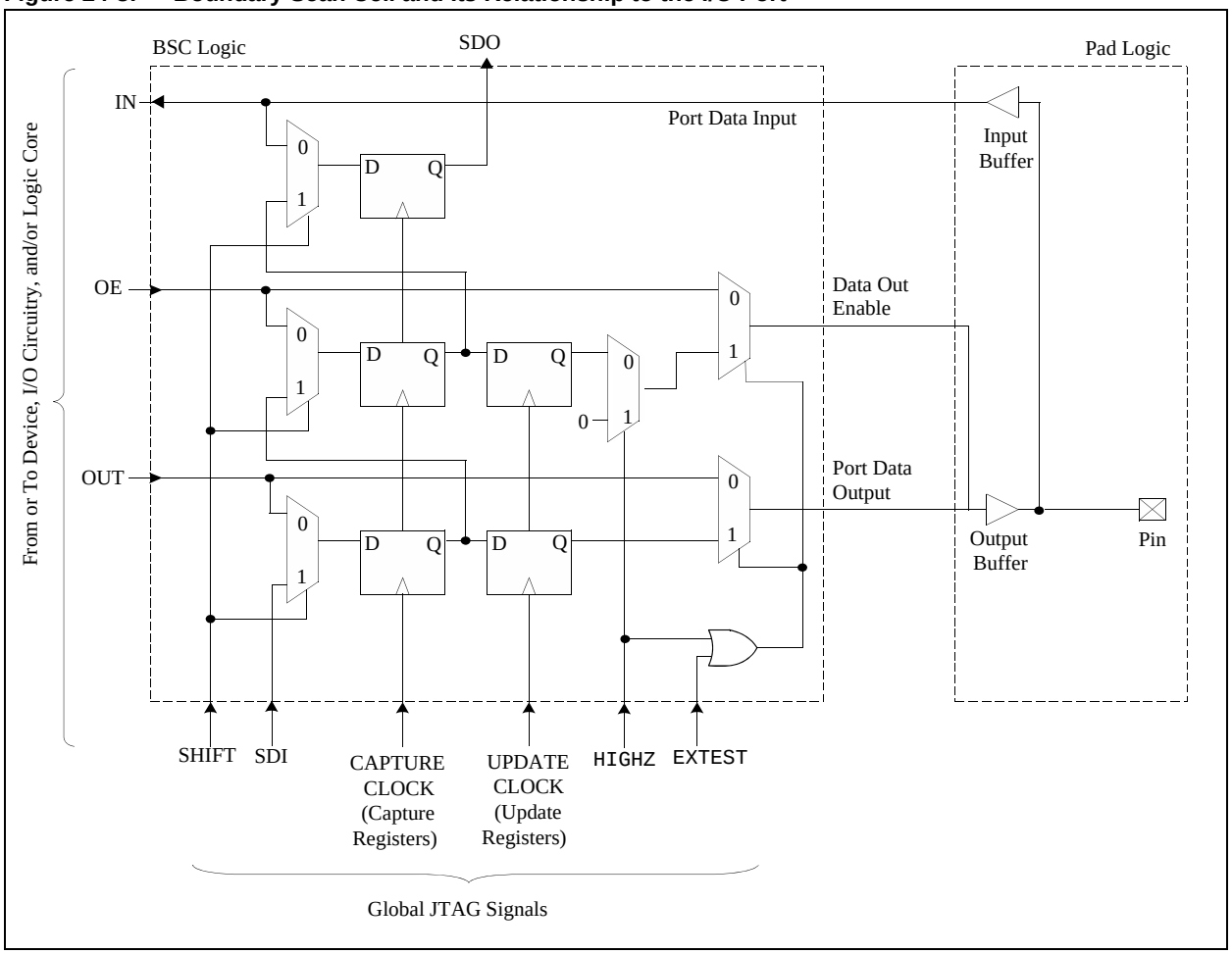
24.4.3.1 BOUNDARY SCAN CELL

The Boundary Scan Cell captures and overrides I/O input or output data values when JTAG is active. The Boundary Scan Cell 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. The capture cells also pass JTAG data along to the Boundary Scan register. Command signals from the TAP controller determine if the JTAG data is captured, and how and when it is clocked out of the Boundary Scan Cell.

The first register either captures internal data sent 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 output-enable values. The third register captures the input data from the I/O's input buffer.

Figure 24-8 shows a typical Boundary Scan Cell and its relationship to the I/O port.

Figure 24-8: Boundary Scan Cell and Its Relationship to the I/O Port



24.4.4 JTAG Instructions

PIC24H 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 Microchip-specific instructions. Table 24-3 describes these mandatory, optional, and Microchip-specific JTAG instructions.

Table 24-3: JTAG Instructions

JTAG Instruction	Description
Mandatory JTAG Instructions:	
BYPASS (0Fh)	Bypasses a device in a test chain. In Bypass mode, a single shift register stage provides a minimum-length serial path between the TDI and TDO pins.
SAMPLE/PRELOAD (01h)	Takes snapshots of the component's input and output signals without interfering with the normal operation of the assembled board. The snapshot is taken on the rising edge of TCK in the Capture-DR controller state. The data can be viewed by shifting through the component's TDO output. This instruction also allows the scanning of the BSR without interfering with normal operation of the on-chip system logic. For example, before the EXTEST instruction is selected, data can be loaded onto the latched parallel outputs using PRELOAD. As soon as the EXTEST instruction is transferred to the parallel output of the Instruction register, the preloaded data is driven through the system output pins. This ensures that known data, consistent at the board level, is driven immediately when the EXTEST instruction is entered. Without PRELOAD, indeterminate data would be driven until the first scan sequence had been completed.
EXTEST (03h)	Allows testing of off-chip circuitry and board level interconnections. Data typically is loaded onto the latched parallel outputs of the Boundary Scan shift register stages by using the PRELOAD instruction before the EXTEST instruction is selected. BSR cells at output pins are used to apply test stimuli. Those at input pins are used to capture test results.
Optional JTAG Instructions	
IDCODE (02h)	Selects a 32-bit identification register to be connected for serial access between TDI and TDO in the Shift-DR controller state. This instruction causes the 32-bit device identification word to be shifted out on the TDO pin.
HIGHZ (04h)	Places the component in a state in which all of its system logic outputs are placed in an inactive drive state (e.g., high impedance). In this state, an in-circuit test system drives the signals onto the connections normally driven by a component output without damaging the component. In the HIGHZ mode, the Bypass register is connected between TDI and TDO in the Shift-DR state.
Microchip-specific JTAG Instructions	
MCHP_SCAN (07h)	Selects the internal Microchip-specific scan register to be connected for serial access between the TDI and TDO in the Shift-DR controller state.
MCHP_CMD (08h)	Selects 8-bit Microchip Command shift register to be connected for serial access between the TDI and TDO in the Shift-DR controller state. This shift register supports up to 256 commands. The following two commands are available for the user; the rest are reserved: • JTAG_MCLR (01h): Performs a device Master Clear Reset while the JTAG interface is active; functionally equivalent to hardware $\overline{\text{MCLR}}$. The TAP interface itself is not reset. • JTAG_MUX (02h): Switches the JTAG interface to ICSP operation. After this command, TDI and TDO assume the PGD functions (split input and output, respectively), and TCK functions as PGC.

24.4.5 Boundary Scan Testing

Boundary Scan Testing is the method of controlling and observing the boundary pins of the JTAG-compliant device with software. Boundary Scan Testing 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 printed circuit board 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 Boundary Scan Testing is that it can be implemented without physical test probes. All that is needed is a 4-wire or 5-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 Boundary Scan Testing is that it can only evaluate digital signals and circuit continuity. It cannot measure input or output voltage levels or currents.

24.4.5.1 RELATED JTAG FILES

To implement Boundary Scan Testing, all JTAG test tools require a Boundary Scan Description Language (BSDL) file. BSDL is a subset of VHSIC Hardware Description Language (VHDL), and is described as part of IEEE 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:

- Pinout and package configuration for the particular device
- Physical location of the TAP pins
- Device ID register and the device ID
- Length of the IR
- Supported BST instructions and their binary codes
- Length and structure of the Boundary Scan register
- Boundary scan cell definition

Device-specific BSDL files are available at Microchip's web site, www.microchip.com. The name for each BSDL file is the device name and silicon revision. For example, `PIC24HJ64GP206.BSD` is the BSDL file for the PIC24HJ64GP206 device.

24.4.6 JTAG Device Programming

The JTAG interface can also be used to program PIC24H devices in their target applications. Using 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.

JTAG device programming actually uses the standard ICSP method over the four pins of the TAP interface. When triggered by the appropriate JTAG command sequence, the TDI, TDO, and TCK pins assume the functions of the PGD and PGC pins. Aside from this pin remapping, ICSP programming over the JTAG interface behaves exactly as it does over the standard ICSP interface.

Because of the added time overhead for switching the TAP interface, JTAG device programming takes slightly longer than standard ICSP programming over the PGC and PGD pins. Enhanced ICSP programming is not available with JTAG programming.

Following are the steps required for JTAG device programming:

1. Shift the `MCHP_CMD(08h)` instruction into the Instruction Shift Register. This Instruction selects the 8-bit Microchip Command register to be connected for serial access between the TDI and TDO pins.
2. Shift the `JTAG_MUX(02h)` instruction into the Microchip command register. This instruction switches the JTAG interface to ICSP operation. This command causes the TDI and TDO pins to assume the PGD functions and the TCK pin to assume the PGC function.

24.5 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 PIC24H device family, 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.	

Note: For additional application notes and code examples for the PIC24H device family, visit the Microchip web site (www.microchip.com).

24.6 REVISION HISTORY

Revision A (May 2007)

This is the initial released version of the document.