

This chapter describes the IEEE Standard 1149.1 JTAG BST circuitry that is supported in MAX[®] V devices and how you can enable concurrent in-system programming of multiple devices in a minimum time with the IEEE Standard 1532 in-system programmability (ISP). This chapter also describes the programming sequence, types of programming with the Quartus[®] II software or external hardware, and design security.

This chapter includes the following sections:

- “IEEE Std. 1149.1 Boundary-Scan Support” on page 6–1
- “In-System Programmability” on page 6–5

IEEE Std. 1149.1 Boundary-Scan Support

All MAX V devices provide JTAG BST circuitry that complies with the IEEE Std. 1149.1-2001 specification. You can only perform JTAG boundary-scan testing after you have fully powered the V_{CCINT} and all V_{CCIO} banks and a certain amount of configuration time (t_{CONFIG}) have passed. For in-system programming, MAX V devices can use the JTAG port with either the Quartus II software or hardware with Programmer Object File (.pof), Jam[™] Standard Test and Programming Language (STAPL) Format File (.jam), or Jam Byte Code Files (.jbc).

JTAG pins support 1.5-V, 1.8-V, 2.5-V, and 3.3-V I/O standards. The V_{CCIO} of the bank where it is located determines the supported voltage level and standard. The dedicated JTAG pins reside in Bank 1 of all MAX V devices.

Table 6–1 lists the JTAG instructions supported in MAX V devices.

Table 6–1. JTAG Instructions for MAX V Devices (Part 1 of 2)

JTAG Instruction	Instruction Code	Description
SAMPLE/PRELOAD	00 0000 0101	Allows you to capture and examine a snapshot of signals at the device pins if the device is operating in normal mode. Permits an initial data pattern to be an output at the device pins.
EXTEST (1)	00 0000 1111	Allows you to test the external circuitry and board-level interconnects by forcing a test pattern at the output pins and capturing test results at the input pins.
BYPASS	11 1111 1111	Places the 1-bit bypass register between the TDI and TDO pins, which allows the boundary-scan test (BST) data to pass synchronously through target devices to adjacent devices during normal device operation.
USERCODE	00 0000 0111	Selects the 32-bit USERCODE register and places it between the TDI and TDO pins, allowing you to shift the USERCODE register out of the TDO pin serially. If you do not specify the USERCODE in the Quartus II software, the 32-bit USERCODE register defaults to all 1's.

© 2011 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Table 6–1. JTAG Instructions for MAX V Devices (Part 2 of 2)

JTAG Instruction	Instruction Code	Description
IDCODE	00 0000 0110	Selects the IDCODE register and places it between the TDI and TDO pins, allowing you to shift the IDCODE register out of the TDO pin serially.
HIGHZ (1)	00 0000 1011	Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through target devices to adjacent devices if the device is operating in normal mode and tri-stating all the I/O pins.
CLAMP (1)	00 0000 1010	Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through target devices to adjacent devices during normal device operation and holding I/O pins to a state defined by the data in the boundary-scan register.
USER0	00 0000 1100	Allows you to define the scan chain between the TDI and TDO pins in the MAX V logic array. Use this instruction for custom logic and JTAG interfaces.
USER1	00 0000 1110	Allows you to define the scan chain between the TDI and TDO pins in the MAX V logic array. Use this instruction for custom logic and JTAG interfaces.
IEEE 1532 instructions	For the instruction codes of the IEEE 1532 instructions, refer to the IEEE 1532 BSDL Files page of the Altera website.	IEEE 1532 in-system concurrent (ISC) instructions used if programming a MAX V device through the JTAG port.

Note to Table 6–1:

(1) HIGHZ, CLAMP, and EXTTEST instructions do not disable weak pull-up resistors or bus hold features.



You must not issue unsupported JTAG instructions to the MAX V device because this may put the device into an unknown state, requiring a power cycle to recover device operation.

The MAX V device instruction register length is 10 bits and the USERCODE register length is 32 bits. Table 6–2 and Table 6–3 list the boundary-scan register length and device IDCODE information for MAX V devices.

Table 6–2. Boundary-Scan Register Length for MAX V Devices

Device	Boundary-Scan Register Length
5M40Z	240
5M80Z	240
5M160Z	240
5M240Z (1)	240
5M240Z (2)	480
5M570Z	480
5M1270Z (3)	636
5M1270Z (4)	816
5M2210Z	816

Notes to Table 6–2:


- (1) Not applicable to T144 package of the 5M240Z device.
- (2) Only applicable to T144 package of the 5M240Z device.
- (3) Not applicable to F324 package of the 5M1270Z device.
- (4) Only applicable to F324 package of the 5M1270Z device.


Table 6–3. 32-Bit IDCODE for MAX V Devices

Device	Binary IDCODE (32 Bits) (1)				HEX IDCODE
	Version (4 Bits)	Part Number	Manufacturer Identity (11 Bits)	LSB (1 Bit) (2)	
5M40Z	0000	0010 0000 1010 0101	000 0110 1110	1	0x020A50DD
5M80Z	0000	0010 0000 1010 0101	000 0110 1110	1	0x020A50DD
5M160Z	0000	0010 0000 1010 0101	000 0110 1110	1	0x020A50DD
5M240Z (3)	0000	0010 0000 1010 0101	000 0110 1110	1	0x020A50DD
5M240Z (4)	0000	0010 0000 1010 0110	000 0110 1110	1	0x020A60DD
5M570Z	0000	0010 0000 1010 0110	000 0110 1110	1	0x020A60DD
5M1270Z (5)	0000	0010 0000 1010 0011	000 0110 1110	1	0x020A30DD
5M1270Z (6)	0000	0010 0000 1010 0100	000 0110 1110	1	0x020A40DD
5M2210Z	0000	0010 0000 1010 0100	000 0110 1110	1	0x020A40DD

Notes to Table 6–2:

- (1) The MSB is on the left.
- (2) The LSB for IDCODE is always 1.
- (3) Not applicable to T144 package of the 5M240Z device.
- (4) Only applicable to T144 package of the 5M240Z device.
- (5) Not applicable to F324 package of the 5M1270Z device.
- (6) Only applicable to F324 package of the 5M1270Z device.

 For JTAG direct current (DC) characteristics, refer to the *DC and Switching Characteristics for MAX V Devices* chapter.

 For more information about JTAG BST, refer to the *JTAG Boundary-Scan Testing for MAX V Devices* chapter.

JTAG Block

If you issue either the USER0 or USER1 instruction to the JTAG test access port (TAP) controller, the MAX V JTAG block feature allows you to access the JTAG TAP controller and state signals. The USER0 and USER1 instructions bring the JTAG boundary-scan chain (TDI) through the user logic instead of the boundary-scan cells (BSCs) of MAX V devices. Each USER instruction allows for one unique user-defined JTAG chain into the logic array.

Parallel Flash Loader

MAX V devices have the ability to interface JTAG to non-JTAG devices and are suitable to use with the general flash memory devices that require programming during the in-circuit test. You can use the flash memory devices for FPGA configuration or be part of the system memory. In many cases, you can use the MAX V device as a bridge device that controls configuration between FPGA and flash devices. Unlike ISP-capable CPLDs, bulk flash devices do not have JTAG TAP pins or connections. For small flash devices, it is common to use the serial JTAG scan chain of a connected device to program the non-JTAG flash device but this is slow, inefficient, and impractical for large parallel flash devices. Using the MAX V JTAG block as a parallel flash loader (PFL) with the Quartus II software to program and verify flash contents provides a fast and cost-effective means of in-circuit programming during testing.


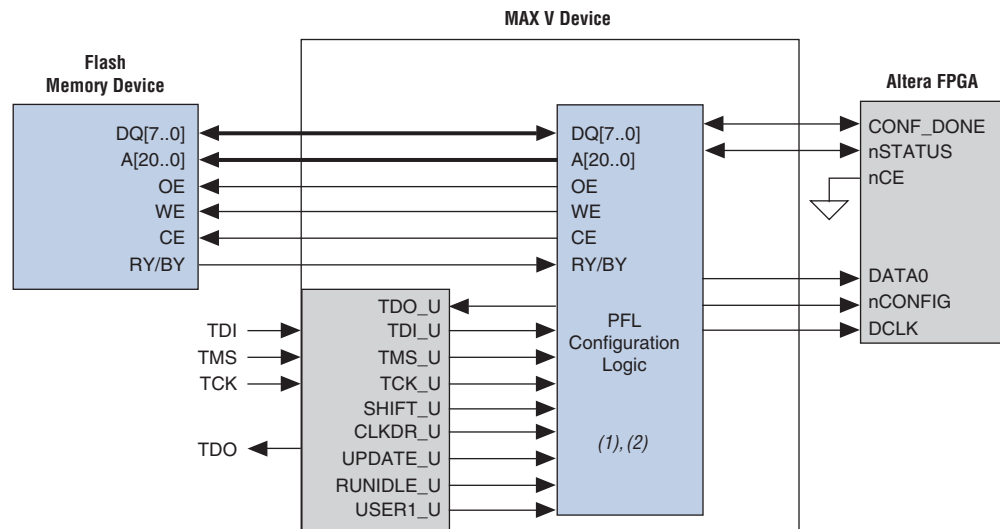
 For more information about PFL, refer to the *Parallel Flash Loader Megafunction User Guide*.

Figure 6-1 shows how you can use the MAX V JTAG block as a PFL.

Figure 6-1. PFL for MAX V Devices



Notes to Figure 6-1:

- (1) This block is implemented in logic elements (LEs).
- (2) This function is supported in the Quartus II software.

In-System Programmability

You can program MAX V devices in-system through the industry standard 4-pin IEEE Std. 1149.1 interface. ISP offers quick and efficient iterations during design development and debugging cycles. The flash-based SRAM configuration elements configure the logic, circuitry, and interconnects in the MAX V architecture. Each time the device is powered up, the configuration data is loaded into the SRAM elements. The process of loading the SRAM data is called configuration. The on-chip configuration flash memory (CFM) block stores the configuration data of the SRAM element. The CFM block stores the configuration pattern of your design in a reprogrammable flash array. During ISP, the MAX V JTAG and ISP circuitry programs the design pattern into the non-volatile flash array of the CFM block.

The MAX V JTAG and ISP controller internally generate the high programming voltages required to program the CFM cells, allowing in-system programming with any of the recommended operating external voltage supplies. You can configure the ISP anytime after you have fully powered V_{CCINT} and all V_{CCIO} banks, and the device has completed the configuration power-up time. By default, during in-system programming, the I/O pins are tri-stated and weakly pulled-up to V_{CCIO} banks to eliminate board conflicts. The in-system programming clamp and real-time ISP feature allow user control of the I/O state or behavior during ISP.

For more information, refer to “In-System Programming Clamp” on page 6-7 and “Real-Time ISP” on page 6-8.

These devices also offer an ISP_DONE bit that provides safe operation if in-system programming is interrupted. This ISP_DONE bit, which is the last bit programmed, prevents all I/O pins from driving until the bit is programmed.

IEEE 1532 Support

The JTAG circuitry and ISP instruction set in MAX V devices are compliant to the IEEE-1532-2002 programming specification. This provides industry-standard hardware and software for in-system programming among multiple vendor programmable logic devices (PLDs) in a JTAG chain.

- For more information about MAX V 1532 Boundary-Scan Description Language (.bsd) files, refer to the [IEEE 1532 BSDL Files](#) page of the Altera website.

Jam Standard Test and Programming Language

You can use the Jam STAPL to program MAX V devices with in-circuit testers, PCs, or embedded processors. The Jam byte code is also supported for MAX V devices. These software programming protocols provide a compact embedded solution for programming MAX V devices.

- For more information, refer to [AN 425: Using Command-Line Jam STAPL Solution for Device Programming](#).

Programming Sequence

During in-system programming, 1532 instructions, addresses, and data are shifted into the MAX V device through the TDI input pin. Data is shifted out through the TDO output pin and compared with the expected data.

To program a pattern into the device, follow these steps:

1. *Enter ISP*—The enter ISP stage ensures that the I/O pins transition smoothly from user mode to ISP mode.
2. *Check ID*—The silicon ID is checked before any Program or Verify process. The time required to read this silicon ID is relatively small compared to the overall programming time.
3. *Sector Erase*—Erasing the device in-system involves shifting in the instruction to erase the device and applying an erase pulse or pulses. The erase pulse is automatically generated internally by waiting in the run, test, or idle state for the specified erase pulse time of 500 ms for the CFM block and 500 ms for each sector of the user flash memory (UFM) block.
4. *Program*—Programming the device in-system involves shifting in the address, data, and program instruction and generating the program pulse to program the flash cells. The program pulse is automatically generated internally by waiting in the run/test/idle state for the specified program pulse time of 75 μ s. This process is repeated for each address in the CFM and UFM blocks.
5. *Verify*—Verifying a MAX V device in-system involves shifting in addresses, applying the verify instruction to generate the read pulse, and shifting out the data for comparison. This process is repeated for each CFM and UFM address.
6. *Exit ISP*—An exit ISP stage ensures that the I/O pins transition smoothly from ISP mode to user mode.

A stand-alone verification of a programmed pattern involves only steps 1, 2, 5, and 6. These steps are automatically executed by third-party programmers, the Quartus II software, or the Jam STAPL and Jam Byte-Code Players.

Table 6-4 lists the programming times for MAX V devices with in-circuit testers to execute the algorithm vectors in hardware. Because of data processing and data transfer limitations, software-based programming tools used with download cables are slightly slower.

Table 6-4. Family Programming Times for MAX V Devices


Description	5M40Z/ 5M80Z/ 5M160Z/ 5M240Z (1)	5M240Z (2)	5M570Z	5M1270Z (3)	5M1270Z (4)	5M2210Z	Unit
Erase + Program (1 MHz)	1.72	2.16	2.16	2.90	3.92	3.92	sec
Erase + Program (10 MHz)	1.65	1.99	1.99	2.58	3.40	3.40	sec
Verify (1 MHz)	0.09	0.17	0.17	0.30	0.49	0.49	sec
Verify (10 MHz)	0.01	0.02	0.02	0.03	0.05	0.05	sec
Complete Program Cycle (1 MHz)	1.81	2.33	2.33	3.20	4.41	4.41	sec
Complete Program Cycle (10 MHz)	1.66	2.01	2.01	2.61	3.45	3.45	sec

Notes to Table 6-4:

- (1) Not applicable to T144 package of the 5M240Z device.
- (2) Only applicable to T144 package of the 5M240Z device.
- (3) Not applicable to F324 package of the 5M1270Z device.
- (4) Only applicable to F324 package of the 5M1270Z device.

User Flash Memory Programming

The Quartus II software (with the use of .pof, .jam, or .jbc files) supports programming of the UFM block independent of the logic array design pattern stored in the CFM block. This allows updating or reading UFM contents through ISP without altering the current logic array design, or vice versa. By default, these programming files and methods program the entire flash memory contents, which includes the CFM block and UFM contents. The stand-alone embedded Jam STAPL Player and Jam Byte-Code Player provide action commands for programming or reading the entire flash memory (UFM and CFM together) or each independently.

 For more information, refer to *AN 425: Using the Command-Line Jam STAPL Solution for Device Programming*.

In-System Programming Clamp

By default, the IEEE 1532 instruction used for entering ISP automatically tri-states all I/O pins with weak pull-up resistors for the duration of the ISP sequence. However, some systems may require certain pins on MAX V devices to maintain a specific DC logic level during an in-field update. For these systems, you can use the optional in-system programming clamp instruction in the MAX V circuitry to control I/O

behavior during the ISP sequence. The in-system programming clamp instruction allows the device to sample and sustain the value on an output pin (an input pin remains tri-stated if sampled) or to set a logic high, logic low, or tri-state value explicitly on any pin. Setting these options is controlled on an individual pin basis with the Quartus II software.

 For more information, refer to [AN 630: Real-Time ISP and ISP Clamp for Altera CPLDs](#).

Real-Time ISP

For systems that require more than the DC logic level control of I/O pins, the real-time ISP feature allows you to update the CFM block with a new design image, while the current design continues to operate in the SRAM logic array and I/O pins. A new programming file is updated into the MAX V device without halting the original operation of your design, saving down-time costs for remote or field upgrades. The updated CFM block configures the new design into the SRAM after the next power cycle. You can execute an immediate SRAM configuration without a power cycle with a specific sequence of ISP commands. The SRAM configuration without a power cycle takes a specific amount of time (t_{CONFIG}). During this time, the I/O pins are tri-stated and weakly pulled-up to V_{CCIO} .

Design Security

All MAX V devices contain a programmable security bit that controls access to the data programmed into the CFM block. If this bit is programmed, you cannot copy or retrieve the design programming information stored in the CFM block. This feature provides a high-level design security because programmed data within flash memory cells is invisible. You can only reset the security bit that controls this function and other programmed data if the device is erased. The SRAM is also invisible and cannot be accessed regardless of the security bit setting. The security bit does not protect the UFM block data, and the UFM is accessible through JTAG or logic array connections.

Programming with External Hardware

You can program MAX V devices by downloading the information through in-circuit testers, embedded processors, the Altera® ByteBlaster™ II, EthernetBlaster II, EthernetBlaster, and USB-Blaster™ cables. You need to power up these cable's $V_{\text{CC(TRGT)}}$ with V_{CCIO} of Bank 1.

 For more information about the respective cables, refer to the [Cable & Adapter Drivers Information](#) page.

BP Microsystems, System General, and other programming hardware manufacturers provide programming support for Altera devices. For device support information, refer to their websites.

Document Revision History

Table 6-5 lists the revision history for this chapter.

Table 6-5. Document Revision History

Date	Version	Changes
May 2011	1.1	Updated "Programming with External Hardware" section.
December 2010	1.0	Initial release.

