Chapter 3. JTAG & In-System Programmability

MII51003-1.1

IEEE Std. 1149.1 (JTAG) Boundary Scan Support

All MAX® II devices provide Joint Test Action Group (JTAG) boundary-scan test (BST) circuitry that complies with the IEEE Std. 1149.1-2001 specification. JTAG boundary-scan testing can only be performed at any time after V_{CCINT} and all V_{CCIO} banks have been fully powered and a t_{CONFIG} amount of time has passed. MAX II devices can also use the JTAG port for in-system programming together with either the Quartus® II software or hardware using Programming Object Files (**.pof**), Jam™ Standard Test and Programming Language (STAPL) Files (**.jam**) or Jam Byte-Code Files (**.jbc**).

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

MAX II devices support the JTAG instructions shown in Table 3–1.

Table 3–1. MAX II JTAG Instructions (Part 1 of 2)

JTAG Instruction	Instruction Code	Description
`SAMPLE/PRELOAD`	`00 0000 0101`	Allows a snapshot of signals at the device pins to be captured and examined during normal device operation, and permits an initial data pattern to be output at the device pins.
`EXTEST` *(1)*	`00 0000 1111`	Allows the external circuitry and board-level interconnects to be tested 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 data to pass synchronously through selected 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 the `USERCODE` to be serially shifted out of `TDO`. This register defaults to all 1's if not specified in the Quartus II software.
`IDCODE`	`00 0000 0110`	Selects the `IDCODE` register and places it between `TDI` and `TDO`, allowing the IDCODE to be serially shifted out of `TDO`.

Table 3–1. MAX II JTAG Instructions (Part 2 of 2)

JTAG Instruction	Instruction Code	Description
`HIGHZ` *(1)*	`00 0000 1011`	Places the 1-bit bypass register between the `TDI` and `TDO` pins, which allows the boundary scan test data to pass synchronously through selected devices to adjacent devices during normal device operation, while tri-stating all of the I/O pins.
`CLAMP` *(1)*	`00 0000 1010`	Places the 1-bit bypass register between the `TDI` and `TDO` pins, which allows the boundary scan test data to pass synchronously through selected devices to adjacent devices during normal device operation, while holding I/O pins to a state defined by the data in the boundary-scan register.
`USER0`	`00 0000 1100`	This instruction allows the user to define their own scan chain between `TDI` and `TDO` in the MAX II logic array. This instruction is also used for custom logic and JTAG interfaces.
`USER1`	`00 0000 1110`	This instruction allows the user to define their own scan chain between `TDI` and `TDO` in the MAX II logic array. This instruction is also used for custom logic and JTAG interfaces.
IEEE 1532 instructions	*(2)*	IEEE 1532 ISC instructions used when programming a MAX II device via the JTAG port.

Notes to Table 3–1:

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

(2) These instructions are shown in the 1532 BSDL files, which will be posted on the Altera® web site at **www.altera.com** when they are available.

The MAX II device instruction register length is 10 bits and the USERCODE register length is 32 bits. Tables 3–2 and 3–3 show the boundary-scan register length and device IDCODE information for MAX II devices.

Table 3–2. MAX II Boundary-Scan Register Length

Device	Boundary-Scan Register Length
EPM240	240
EPM570	480
EPM1270	636
EPM2210	816

Table 3–3. 32-Bit MAX II Device IDCODE

Device	Binary IDCODE (32 Bits) *(1)*				HEX IDCODE
	Version (4 Bits)	Part Number	Manufacturer Identity (11 Bits)	LSB (1 Bit) *(2)*	
EPM240	0000	0010 0000 1010 0001	000 0110 1110	1	0x020A10DD
EPM570	0000	0010 0000 1010 0010	000 0110 1110	1	0x020A20DD
EPM1270	0000	0010 0000 1010 0011	000 0110 1110	1	0x020A30DD
EPM2210	0000	0010 0000 1010 0100	000 0110 1110	1	0x020A40DD

Notes to Table 3–2:

(1) The most significant bit (MSB) is on the left.

(2) The IDCODE's least significant bit (LSB) is always 1.

For JTAG AC characteristics, refer to the chapter on *DC & Switching Characteristics*. For more information on JTAG BST, see the chapter on *IEEE 1149.1 (JTAG) Boundary-Scan Testing for MAX II Devices*.

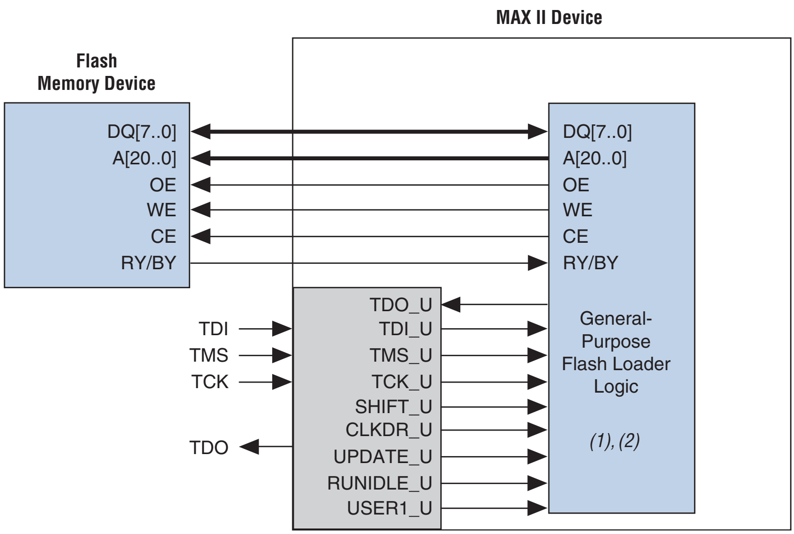
JTAG Translator

The JTAG translator feature allows you to access the JTAG TAP and state signals when either the USER0 or USER1 instruction is issued to the JTAG TAP. The USER0 and USER1 instructions bring the JTAG boundary scan chain (TDI) through the user logic instead of the MAX II device's boundary scan cells. Each USER instruction allows for one unique user-defined JTAG chain into the logic array.

General-Purpose Flash Loader

The JTAG translator ability to interface JTAG to non-JTAG devices is ideal for general-purpose flash memory devices (such as Intel or Fujitsu based devices) that require programming during in-circuit test. The flash memory devices can be used for FPGA configuration or be part of system memory. In many cases, the MAX II device is already connected to these devices as the configuration control logic between the FPGA and the flash device. Unlike ISP-capable CPLD devices, 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. This is slow and inefficient in most cases and impractical for large parallel flash devices. Using the MAX II device's JTAG translator as a general-purpose flash loader to program and verify flash contents provides a fast and cost-effective means of in-circuit programming during test. Figure 3–1 shows MAX II being used as a general-purpose flash loader.

Figure 3–1. MAX II JTAG Translator as General-Purpose Flash Loader



Notes to Figure 3–1:
(1) This block is implemented in LEs.
(2) This function will be supported in a future version of the Quartus II software.

In System Programmability

MAX II devices can be programmed in-system via the industry standard 4-pin IEEE Std. 1149.1 (JTAG) interface. In system programmability (ISP) offers quick, efficient iterations during design development and debugging cycles. The logic, circuitry, and interconnects in the MAX II architecture are configured with flash-based SRAM configuration elements. These SRAM elements require configuration data to be loaded each time the device is powered. The process of loading the SRAM data is called configuration. The on-chip configuration flash memory (CFM) block stores the SRAM element's configuration data. The CFM block stores the design's configuration pattern in a reprogrammable flash array. During ISP, the MAX II JTAG and ISP circuitry programs the design pattern into the CFM block's non-volatile flash array.

The MAX II 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 (i.e., 3.3 V/2.5 V or 1.8 V for the MAX II devices with a "G" ordering code). ISP can be performed anytime after V_{CCINT} and all V_{CCIO} banks have been fully powered 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} to eliminate board conflicts. The pull-up value ranges from 5 to 40 kZ. There are two other options in MAX II devices that allow user control of I/O state or behavior during ISP.

For more information, refer to "In-System Programming Clamp" on page 3–8 and "Real-Time ISP" on page 3–8.

These devices also offer an `ISP_DONE` bit that provides safe operation when 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 II devices is 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.

The MAX II 1532 BSDL files will be released on the Altera web site when available.

Jam Standard Test & Programming Language (STAPL)

The Jam STAPL JEDEC standard, JESD71, can be used to program MAX II devices with in-circuit testers, PCs, or embedded processors. The Jam byte code is also supported for MAX II devices. These software programming protocols provide a compact embedded solution for programming MAX II devices.

For more information, see the chapter on *Using Jam STAPL for ISP via an Embedded Processor*.

Programming Sequence

During in-system programming, 1532 instructions, addresses, and data are shifted into the MAX II device through the `TDI` input pin. Data is shifted out through the `TDO` output pin and compared against the expected data. Programming a pattern into the device requires the following six ISP steps. A stand-alone verification of a programmed pattern involves only stages 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.

1. *Enter ISP* – The enter ISP stage ensures that the I/O pins transition smoothly from user mode to ISP mode.

2. *Check ID* – Before any program or verify process, the silicon ID is checked. The time required to read this silicon ID is relatively small compared to the overall programming time.

3. *Bulk Erase* – Erasing the device in-system involves shifting in the instruction to erase the device and applying an erase pulse(s). The erase pulse is automatically generated internally by waiting in the run/test/idle state for the specified erase pulse time of 350 ms.

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 µs. This process is repeated for each address in the CFM block.

5. *Verify* – Verifying a MAX II 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 address.

6. *Exit ISP* – An exit ISP stage ensures that the I/O pins transition smoothly from ISP mode to user mode.

For `TCK` frequencies of 10 MHz, the erase and programming takes less than one second for EPM240 and EPM570 devices. Erase and programming times are less than two seconds for EPM1270 and less than three seconds for the EPM2210 devices. The `TCK` frequency can operate at up to 25 MHz in MAX II devices providing slight improvements in these ISP times.

UFM Programming

The Quartus II software, with the use of POF, Jam, or JBC files, supports programming of each user flash memory (UFM) block sector independent from 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 will program both 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 provides action commands for programming or reading the entire flash memory (UFM and CFM together) or each independently.

For more information, see the chapter on *Using Jam STAPL for ISP via an Embedded Processor*.

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 II devices to maintain a specific DC logic level during an in-field update. For these systems, an optional in-system programming clamp instruction exists in MAX II circuitry to control I/O behavior during the ISP sequence. The in-system programming clamp instruction enables the device to sample and sustain the value on an output pin (an input pin would remain tri-stated if sampled) or to explicitly set a logic high, logic low, or tri-state value on any pin. Setting these options is controlled on an individual pin basis using the Quartus II software.

Real-Time ISP

For systems that require more than 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 entire current design continues to operate in the SRAM logic array and I/O pins. A new programming file is updated into the MAX II device without halting the original design's operation, saving down-time costs for remote or field upgrades. The updated CFM block configures the new design into the SRAM upon the next power cycle. It is also possible to execute an immediate configuration of the SRAM without a power cycle by using a specific sequence of ISP commands. The configuration of SRAM 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}.

The Quartus II software, Jam STAPL player, and Jam Byte-Code Player provides the designer with the option to control real-time ISP (if used) and allows the designer to control the timeline of new design configuration into the SRAM (immediately or upon next power cycle).

Design Security

All MAX II devices contain a programmable security bit that controls access to the data programmed into the CFM block. When this bit is programmed, design programming information, stored in the CFM block, cannot be copied or retrieved. This feature provides a high level of design security because programmed data within flash memory cells is invisible. The security bit that controls this function, as well as all other programmed data, is reset only when the device is erased. The SRAM is also invisible and cannot be accessed regardless of the security bit setting. The UFM block data is not protected by the security bit and is accessible through JTAG or logic array connections.

Programming with External Hardware

MAX II devices can be programmed by downloading the information via in-circuit testers, embedded processors, the Altera® ByteblasterMV™, MasterBlaster™, ByteBlaste™r II, and USB-Blaster cables, and through the universal serial bus (USB)-based Altera Programming Unit (APU) with the appropriate adapter.

BP Microsystems, System General, and other programming hardware manufacturers provide programming support for Altera devices. Check their web sites for device support information.