

Introduction

As time-to-market pressure increases, design engineers require advanced system-level products to ensure problem-free development and manufacturing. Programmable logic devices (PLDs) with in-system programmability (ISP) can help accelerate development time, facilitate in-field upgrades, simplify the manufacturing flow, lower inventory costs, and improve printed circuit board (PCB) testing capabilities. Altera® ISP-capable MAX® II devices can be programmed and reprogrammed in-system via the IEEE Std. 1149.1 Joint Test Action Group (JTAG) interface. This interface allows MAX II devices to be programmed and the PCB to be functionally tested in a single manufacturing step, saving testing time and assembly costs. This chapter describes guidelines you should follow to design successfully with ISP, including:

- “General ISP Guidelines” on page 11-1
- “IEEE Std. 1149.1 Signals” on page 11-4
- “Sequential versus Concurrent Programming” on page 11-6
- “ISP Troubleshooting Guidelines” on page 11-7
- “ISP via Embedded Processors” on page 11-9
- “ISP via In-Circuit Testers” on page 11-10

General ISP Guidelines

This section provides guidelines that help you design successfully for ISP-capable MAX II devices. These guidelines should be used regardless of your specific design implementation.

Operating Conditions

Each MAX II device has several parametric ratings, or operating conditions, that are required for proper operation. Although MAX II devices can exceed these conditions when in user mode and still operate correctly, these conditions should not be exceeded during in-system programming. Violating any of the operating conditions during in-system programming can result in programming failures or incorrectly programmed devices. V_{CCIO} of all I/O banks and V_{CCINT} of the device must be fully powered up for ISP to function.

ISP Voltage

The V_{CCINT} and V_{CCIO} level specified in the device operating conditions table must be maintained on the V_{CCINT} and V_{CCIO} pins during in-system programming to ensure that the device's flash cells are programmed correctly. The V_{CCINT} and V_{CCIO} specification applies for both commercial- and industrial-temperature-grade devices.

Input Voltages

The *MAX II Device Family Data Sheet* lists the MAX II device input voltage specification in the absolute maximum ratings and the recommended operating conditions tables. The input voltages in the absolute maximum rating table refers to the maximum voltage which the device can tolerate before risking permanent damage.

The recommended operating conditions table specify the voltage range for safe device operation. Make sure all pins that transition during in-system programming do not have a ground or V_{CC} overshoot. Overshoot problems typically occur on free-running clocks or data buses that can toggle during in-system programming. All pins that have an overshoot greater than 1.0 V must have series termination.



For more information about the recommended operating conditions and the absolute maximum ratings for MAX II devices and termination, refer to the *DC and Switching Characteristics* chapter in the *MAX II Device Handbook* and *AN 75: High-Speed Board Designs*, respectively.

UFM Operations During In-System Programming

If your design allows you to access the MAX II UFM (write or erase), you must ensure that all the erase or write operations of the UFM are completed before starting any ISP session (including stand-alone verify, examine, setting security bit, and reading the contents of the UFM). You should never start an ISP session when any erase or write operation of the UFM is on going, as this may put the device in an unrecoverable state. However, this restriction does not apply to the read operation of the UFM.

If you cannot ensure that any erase or write operation of the UFM is complete before attempting an ISP operation to the MAX II device, then you should enable the real-time ISP feature. When used properly, this feature can help guard against any UFM/ISP operation contention. When real-time ISP is enabled, the programming algorithm used by the Quartus® II software or the Jam™ (.jam)/Jam Byte-Code (.jbc) files will wait 500 ms before it begins any operation. This is the same amount of time it takes for one UFM sector to be erased (that is, the real-time ISP programming algorithm waits for what may have been a previously started UFM erase sequence to complete).

However, if you are using a real-time ISP feature, no other UFM operations are allowed after that time (no address shifting, no data shifting, and no read, write, or erase operations). This can be controlled by monitoring the RTP_BUSY signal on the altufm_none megafunction. When real-time ISP is under way, the RTP_BUSY output signal on the UFM block goes high. You can monitor this signal and ensure that all UFM operations from the logic array cease until real-time ISP is complete. This user-generated control logic is only necessary for the altufm_none megafunction, which provides no auto-generated logic. The other interfaces for the altufm megafunction (altufm_parallel, altufm_spi, altufm_i2c) contain control logic that automatically monitors the RTP_BUSY signal and ceases operations to the UFM when a real-time ISP operation is under way.

Interrupting In-System Programming

Altera does not recommend interrupting the programming process. However, the MAX II device has an `ISP_DONE` bit that will only be set at the very end of a successful program sequence. The I/O pins will only drive out if this bit is set. This prevents a partially programmed device from driving out and operating unpredictably.

MultiVolt Devices and Power-Up Sequences

For the JTAG circuitry to operate correctly during in-system programming or boundary-scan testing, all devices in a JTAG chain must be in the same state. Therefore, in systems with multiple power supply voltages, the JTAG pins must be held in the test-logic-reset state until all devices in the chain are completely powered up. This procedure is particularly important because systems with multiple power supplies cannot power all voltage levels simultaneously.

MAX II devices have the MultiVolt feature and can use more than one power supply voltage: V_{CCINT} and V_{CCIO} for each I/O bank. V_{CCINT} provides power to the JTAG circuitry; V_{CCIO} provides power to input pins and output drivers for output pins, including TDO. Therefore, when using two power supply voltages, the JTAG circuitry must be held in the test-logic-reset state until both power supplies are turned on. If the JTAG pins are not held in the test-logic-reset state, in-system programming errors can occur.

V_{CCIO} Powered before V_{CCINT}

If V_{CCIO} is powered up before V_{CCINT} , the JTAG circuitry is not active but TDO is tri-stated. Even though the JTAG circuitry is not active, if the next device in the JTAG chain is powered up with the same trace as V_{CCIO} , its JTAG circuitry must stay in the test-logic-reset state. Because all TMS and TCK signals are common, they must be disabled for all devices in the chain. Therefore, the JTAG pins must be disabled by pulling TCK low and TMS high.

I/O Pins Tri-Stated during In-System Programming

By default, all device I/O pins are tri-stated during in-system programming. In addition, the MAX II device provides a weak pull-up resistor during ISP. The purpose of this weak pull-up resistor is to eliminate the need for external pull-up resistors on tri-stated I/O pins.

For pins that are used to drive signals and require a particular value during in-system programming (for example, output enable or chip enable signals), you can use the in-system programming clamp feature or the real-time ISP feature available for MAX II devices. These two features ensure that each I/O pin is clamped to a specific state during in-system programming.



For more information, refer to the *In-System Programming Clamp* and *Real-Time ISP* sections in the *JTAG and In-System Programmability* chapter in the *MAX II Device Handbook*.

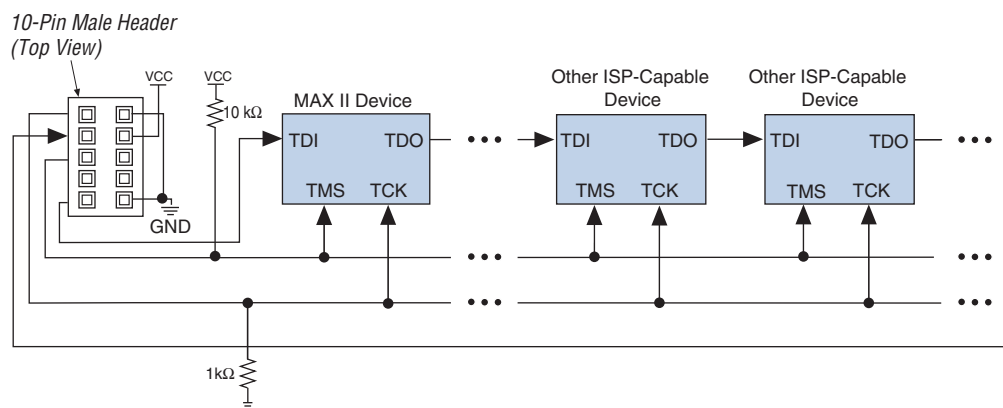
Pull-Up and Pull-Down of JTAG Pins During In-System Programming

A MAX II device operating in in-system programming mode requires four pins: TDI, TDO, TMS, and TCK. The detailed description and function of each pin can be found in the *IEEE 1149.1 (JTAG) Boundary-Scan Testing for MAX II Devices* chapter in the *MAX II Device Handbook*.

Three of the four JTAG pins have internal weak pull-up or pull-down resistors. The TDI and TMS pins have internal weak pull-up resistors while the TCK pin has an internal weak pull-down resistor. However, for device programming in a JTAG chain, there might be devices that do not have internal pull-up or pull-down resistors. Altera recommends to externally pull TMS high through 10-k Ω and TCK low through 1-k Ω resistors. Pulling-up the TDI signal externally for the MAX II device is optional.

Figure 11-1 shows the external pull-up and pull-down for TMS and TCK of the JTAG chain. The TDO pin does not have internal pull-up or pull-down resistors, and does not require external pull-up or pull-down resistors.

Figure 11-1. External Pull-Up and Pull-Down Resistors for TMS and TCK of a JTAG Chain



The TMS pin is pulled high so that the TAP controller will remain in the TEST_LOGIC/RESET state even if there is input from TCK. To prevent TCK from pulsing high, the TCK pin is pulled low during power-up. Pulling TCK high is not recommended because an increase in the power supply to the pull-up resistor causes the TCK to pulse high; thus, it is possible for the TAP controller to reach an unintended state.

IEEE Std. 1149.1 Signals

This section provides guidelines for programming with the IEEE Std. 1149.1 (JTAG) interface.

TCK Signal

Most in-system programming failures are caused by a noisy TCK signal. Noisy transitions on rising or falling edges can cause incorrect clocking of the IEEE Std. 1149.1 Test Access Port (TAP) controller. Incorrect clocking can cause the state machine to transition to an unknown state, leading to in-system programming failures.

Further, because the TCK signal must drive all IEEE Std. 1149.1 devices in the chain in parallel, the signal may have a high fan-out. Like any other high fan-out user-mode clock, you must manage a clock tree to maintain signal integrity. Typical errors that result from clock integrity problems are invalid ID messages, blank-check errors, or verification errors.

Altera recommends pulling the TCK signal low through the internal weak pull-down resistor or an external 1-k Ω resistor.

Fast TCK edges combined with board inductance can cause overshoot problems. When this combination occurs, you must either reduce inductance on the trace or reduce the switching rate by selecting a transistor-to-transistor logic (TTL) driver chip with a slower slew rate. Altera does not recommend using resistor and capacitor (RC) networks to slow down edge rates, because they can violate the device's input specifications. In most cases, using a driver chip prevents the edge rate from being too slow. Altera recommends using driver chips that do not glitch upon power-up.

Programming via a Download Cable

You can program MAX II devices using a MasterBlaster™, ByteBlasterMV™, ByteBlaster™ II, or USB Blaster download cable. Using a PC or UNIX workstation with the Quartus II software programmer, Programmer Object File (.pof), Jam™ Files (.jam), or Jam Byte-Code Files (.jbc) can be downloaded to the MAX II devices via the download cable.

If you are using the download cables and your JTAG chain contains three or more devices, Altera recommends adding a buffer to the chain. You should select a buffer with slow transitions to minimize noise, but be sure that the transition rates can still meet TCK performance requirements of your chain.

If you must extend the download cable, you can attach a standard PC parallel or USB port cable to the download cable. Do not extend the 10-pin header portion of the download cable; extending this portion of the cable can cause noise and in-system programming problems.




Different download cables will have different programming times. For more information about the MasterBlaster, ByteBlasterMV, ByteBlaster II, or USB Blaster download cable, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*, *ByteBlasterMV Download Cable User Guide*, *ByteBlaster II Download Cable User Guide*, or *USB-Blaster Download Cable User Guide*.

Disabling IEEE Std. 1149.1 Circuitry

By default, the JTAG circuitry in MAX II devices is always enabled because they have dedicated JTAG pins and circuitry. The JTAG circuitry must be enabled during ISP and boundary-scan testing, but disabled at all other times. If your design does not use ISP or boundary-scan test (BST) circuitry, Altera recommends disabling the IEEE Std. 1149.1 circuitry.

To disable the JTAG circuitry, Altera recommends pulling TMS high and TCK low. Pulling TCK low ensures that a rising edge does not occur on TCK during the power-up sequence. You can pull TCK high, but you must first pull TMS high. Pulling TMS high first ensures that the rising edge or edges on TCK do not cause the JTAG state machine to leave the test-logic-reset state.

 For more information about disabling the IEEE 1149.1 circuitry, refer to the *Disabling IEEE Std. 1149.1 BST Circuitry* section of the *IEEE 1149.1 (JTAG) Boundary-Scan Testing* chapter in the *MAX II Device Handbook*.

Working with Different Voltage Levels

When devices in a JTAG chain operate at different voltage levels, a device's output voltage specification must meet the subsequent device's input voltage specification. If the devices do not meet this criteria, you must add additional circuitry, such as a level-shifter, to adjust the voltage levels. For example, when a 5.0-V device drives a 2.5-V device, you must adjust the 5.0-V device's output voltage to meet the 2.5-V device's input voltage specification.

Because all devices in a JTAG chain are tied together, you must also ensure that the first device's TDO output meets the subsequent device's TDI input voltage specification to program a chain of devices successfully.

All MAX II devices include a MultiVolt I/O feature, which allows these devices to interface with systems that have different supply voltages. All MAX II devices can be set for 3.3-V, 2.5-V, 1.8-V, or 1.5-V I/O operation. The JTAG pins of MAX II devices support these voltage levels. Refer to the *MAX II Architecture* chapter in the *MAX II Device Handbook* for I/O standard compatibility for each V_{CCIO} voltage. For example, V_{CCIO1} at 3.3 V does not allow JTAG input pins to accept 1.8- or 1.5-V signals.

Sequential versus Concurrent Programming

This section describes how to program multiple devices using sequential and concurrent programming. The JTAG chain setup for sequential and concurrent programming is similar, only the programming algorithms are different.

Sequential Programming

Sequential programming is the process of programming multiple devices in a chain, one device at a time. After the first device in the chain is finished being programmed, the next device is programmed. This sequence continues until all specified devices in the JTAG chain are programmed. After a device is programmed, it will be in bypass mode to allow data to be passed to the subsequent devices in the chain. The devices in the chain do not go into user mode until all the devices are programmed.

Concurrent Programming

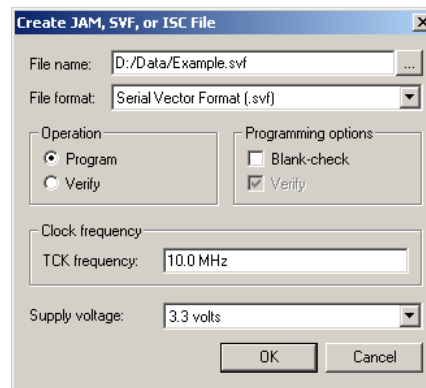
Concurrent programming is used to program devices from the same family (for example, the MAX II family) in parallel. The programming time is slightly longer than the time needed to program the largest device in the chain, resulting in considerably faster programming times than sequential programming (where programming time is equal to the sum of individual programming times for all devices). Higher clock rates for shifting data result in even greater time savings.

Concurrent programming of devices can be done using Serial Vector Format files (.svf), Jam files, or JBC files created from the Quartus II software. See [Figure 11-2](#).

1. On the Tools menu, click **Programmer**.
2. Click **Add File** and select programming files for the respective devices.

3. On the File menu, point to **Create/Update** and click **Create JAM, SVF, or ISC File**.
4. Specify a file in the File format list.
5. Click **OK**.

Figure 11-2. Create JAM, SVF, or ISC File



ISP Troubleshooting Guidelines

This section provides tips for troubleshooting ISP-related problems.

Invalid ID and Unrecognized Device Messages

The first step during in-system programming is to check the device's silicon ID. If the silicon ID does not match, an Invalid ID or Unrecognized Device error is generated. Typical causes for this error are shown below:

- Download cable connected incorrectly
- TDO is not connected
- Incomplete JTAG chain
- Noisy TCK signal
- Jam Player ported incorrectly

Download Cable Connected Incorrectly

You will receive an error if the download cable is connected incorrectly to the parallel or USB port or if it is not receiving power from your board.

 For more information about installing the MasterBlaster, ByteBlasterMV, ByteBlaster, or USB Blaster download cable, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*, *ByteBlasterMV Download Cable User Guide*, *ByteBlaster II Download Cable User Guide*, or *USB-Blaster Download Cable User Guide*.

TDO Is Not Connected

You will receive an error if the TDO port of one device in the chain is not connected. During in-system programming, data must be shifted in and out of each device in the JTAG chain through the JTAG pins. Therefore, each device's TDO port must be connected to the subsequent device's TDI port, and the last device's TDO port must be connected to the download cable's TDO port.

Incomplete JTAG Chain

You will receive an error if the JTAG chain is not complete. To check if an incomplete JTAG chain is causing the error, use an oscilloscope to monitor vectors coming out of each device in the chain. If each device's TDO port does not toggle during in-system programming, your JTAG chain is not complete.

Noisy TCK Signal

Noise on the TCK signal is the most common reason for in-system programming errors. Noisy transitions on rising or falling edges can cause incorrect clocking of the IEEE Std. 1149.1 TAP controller, causing the state machine to be lost and in-system programming to fail. For more information about dealing with noisy TCK signals, refer to ["TCK Signal" on page 11-4](#).

Jam Player Ported Incorrectly

You will receive an error if the Jam Player was not ported correctly for your platform. To check if the Jam Player is causing the error, apply the IDCODE instruction to the target device using a Jam file. You can use a Jam file to load an IDCODE instruction and then shift out the IDCODE value. This test determines if the JTAG chain is set up correctly and if you can read and write to the JTAG chain properly.

You can download the `idcode.zip` file from the Altera website to obtain the `idcode.jam` file.

Troubleshooting Tips

This section discusses some additional suggestions for troubleshooting ISP issues.

Verify the JTAG Chain Continuity

For in-system programming to occur successfully, the number of devices physically in the JTAG chain must match the number reported in the Quartus II software. The following steps show one simple way to verify that the JTAG chain is connected properly.

1. Open the Programmer in the Quartus II software.
2. Click **Auto Detect** in the Programmer. The Quartus II software reports the number of devices found on the JTAG chain. If this fails, check the JTAG chain to make sure it is not broken.

Check the V_{CC} Level of the Board During In-System Programming

Using an oscilloscope, monitor the V_{CCINT} signal on your JTAG chain and set the trigger to the minimum V_{CC} level listed in the recommended operating conditions table of the appropriate device family data sheet. If a trigger occurs during in-system programming, the devices may need more current than is being supplied by the existing power supply. Try replacing the existing power supply with one that provides more current.

Power-Up Problems

Excessive voltage or current on I/O pins during power-up can cause one of the devices in the JTAG chain to experience latch-up. Check if any of the devices are hot to the touch; hot devices have probably experienced latch-up and may have been damaged. In this situation, check all voltage sources to make sure that excessive voltage or current is not being fed into the device. Then, replace the affected device and try programming again.

Random Signals on JTAG Pins

During normal operation, each device's TAP controller must be in the test-logic-reset state. To force the device back into this state, try pulling the TMS signal high and pulsing the TCK clock signal six times. If the device then powers-up successfully, you must add a higher pull-down resistor on the TCK signal.

Software Issues

Failures during in-system programming may occasionally be related to the Quartus II software. Software-related issues are documented in the Find Answers section under the Support Center on the Altera website at www.altera.com. Search the database for information relating to software issues that interfere with in-system programming.

ISP via Embedded Processors

This section provides guidelines for programming ISP-capable devices using the Jam Standard Test and Programming Language (STAPL) and an embedded processor.

Processor and Memory Requirements

The Jam Byte-Code Player supports 8-bit and higher processors; the ASCII Jam Player supports 16-bit and higher processors. The Jam Player uses memory in a predictable manner, which simplifies in-field upgrades by confining updates to the Jam File. The Jam Player memory uses both ROM and dynamic memory (RAM). ROM is used to store the Jam Player binary and the Jam File; dynamic memory is used when the Jam Player is called.



For information about how to estimate the maximum amount of RAM and ROM required by the Jam Player, refer to the *Using Jam STAPL for ISP via an Embedded Processor* chapter in the *MAX II Device Handbook*.

Porting the Jam Player

The Altera Jam Player (both Byte-Code and ASCII versions) works with a PC parallel port. To port the Jam Player to your processor, you only need to modify the `jamstub.c` or `jbistub.c` file (for the ASCII Jam Player or Jam Byte-Code Player, respectively). All other files should remain the same. If the Jam Player is ported incorrectly, an Unrecognized Device error is generated. The most common causes for this error are listed below:

- After porting the Jam Player, the TDO value may be read in reversed polarity. This problem may occur because the default I/O code in the Jam Player assumes the use of the PC parallel port.
- Although the TMS and TDI signals are clocked in on the rising edge of TCK, outputs do not change until the falling edge of TCK. This situation causes a half TCK clock cycle lag in reading out the values. If the TDO transition is expected on the rising edge, the data appears to be offset by one clock.
- Altera recommends using registers to synchronize the output transitions. In addition, some processor data ports use a register to synchronize the output signals. For example, reading and writing to the PC's parallel port is accomplished by reading and writing to registers. The use of these registers must be taken into consideration when reading and writing to the JTAG chain. Incorrect accounting of these registers can cause the values to either lead or lag the expected value.

ISP via In-Circuit Testers

MAX II devices can also be in-system programmed via in-circuit testers. For more information about using Agilent's 3070 in-circuit tester to in-system program MAX II devices, refer to the *Using Jam STAPL for ISP via an Embedded Processor* chapter in the *MAX II Device Handbook*.

Conclusion

The information provided in this document is based on development experiences and customer issues resolved by Altera. For more information about resolving in-system programming problems, contact Altera Applications.

Referenced Documents

This chapter references the following documents:

- *AN 75: High-Speed Board Designs*
- *ByteBlasterMV Download Cable User Guide*
- *ByteBlaster II Download Cable User Guide*
- *DC and Switching Characteristics* chapter in the *MAX II Device Handbook*
- *IEEE 1149.1 (JTAG) Boundary-Scan Testing for MAX II Devices* chapter in the *MAX II Device Handbook*
- *JTAG and In-System Programmability* chapter in the *MAX II Device Handbook*
- *MasterBlaster Serial/USB Communications Cable User Guide*

- *USB-Blaster Download Cable User Guide*
- *Using Jam STAPL for ISP via an Embedded Processor* chapter in the *MAX II Device Handbook*

Document Revision History

Table 11-1 shows the revision history for this chapter.

Table 11-1. Document Revision History

| Date and Revision | Changes Made | Summary of Changes |
|----------------------------|---|---------------------------------------|
| October 2008, version 1.7 | <ul style="list-style-type: none"> ■ Updated New Document Format. | — |
| December 2007, version 1.6 | <ul style="list-style-type: none"> ■ Updated “Pull-Up and Pull-Down of JTAG Pins During In-System Programming” section. ■ Added “Referenced Documents” section. | External pull-up for TDI is optional. |
| December 2006, version 1.5 | <ul style="list-style-type: none"> ■ Added document revision history. | — |
| August 2006, version 1.4 | <ul style="list-style-type: none"> ■ Corrected Figure 11-1. | — |
| January 2005, version 1.3 | <ul style="list-style-type: none"> ■ Previously published as Chapter 12. No changes to content. | — |
| December 2004, version 1.2 | <ul style="list-style-type: none"> ■ Added section User Flash Memory Operations During In-System Programming. | — |
| June 2004, version 1.1 | <ul style="list-style-type: none"> ■ Pull-up resistor values. Textual updates. | — |

