



## Section VI. Configuration & Test

This section provides configuration information for all of the supported configuration schemes for Cyclone® II devices. These configuration schemes use either a microprocessor, configuration device, or download cable. There is detailed information on how to design with Altera® configuration devices. The last chapter provides information on JTAG support in Cyclone II devices.

This section includes the following chapters:

- [Chapter 13, Configuring Cyclone II Devices](#)
- [Chapter 14, IEEE 1149.1 \(JTAG\) Boundary-Scan Testing for Cyclone II Devices](#)

### Revision History

Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the complete handbook.



## Introduction

Cyclone® II devices use SRAM cells to store configuration data. Since SRAM memory is volatile, configuration data must be downloaded to Cyclone II devices each time the device powers up. You can use the active serial (AS) configuration scheme, which can operate at a DCLK frequency up to 40 MHz, to configure Cyclone II devices. You can also use the passive serial (PS) and Joint Test Action Group (JTAG)-based configuration schemes to configure Cyclone II devices. Additionally, Cyclone II devices can receive a compressed configuration bitstream and decompress this data on-the-fly, reducing storage requirements and configuration time.

This chapter explains the Cyclone II configuration features and describes how to configure Cyclone II devices using the supported configuration schemes. This chapter also includes configuration pin descriptions and the Cyclone II configuration file format.



For more information on setting device configuration options or creating configuration files, see the *Software Settings* chapter in the *Configuration Handbook*.

## Cyclone II Configuration Overview

You can use the AS, PS, and JTAG configuration schemes to configure Cyclone II devices. You can select which configuration scheme to use by driving the Cyclone II device MSEL pins either high or low as shown in Table 13–1. The MSEL pins are powered by the V<sub>CCIO</sub> power supply of the bank they reside in. The MSEL [ 1 . . 0 ] pins have 9-k $\Omega$  internal pull-down resistors that are always active. During power-on reset (POR) and reconfiguration, the MSEL pins have to be at LVTTTL V<sub>IL</sub> or V<sub>IH</sub> levels to be considered a logic low or logic high, respectively. Therefore, to avoid any problems with detecting an incorrect configuration scheme, you should connect the MSEL [ ] pins to the V<sub>CCIO</sub> of the I/O bank they reside in and GND without any pull-up or pull-down resistors. The MSEL [ ] pins should not be driven by a microprocessor or another device.

**Table 13–1. Cyclone II Configuration Schemes**

Configuration Scheme	MSEL1	MSEL0
AS (20 MHz)	0	0
PS	0	1
Fast AS (40 MHz) (1)	1	0
JTAG-based Configuration (2)	(3)	(3)

**Notes to Table 13–1:**

- (1) Only the EPCS16 and EPCS64 devices support a DCLK up to 40 MHz clock; other EPCS devices support a DCLK up to 20 MHz. Refer to the *Serial Configuration Devices Data Sheet* for more information.
- (2) JTAG-based configuration takes precedence over other configuration schemes, which means MSEL pin settings are ignored.
- (3) Do not leave the MSEL pins floating; connect them to V<sub>CCIO</sub> or ground. These pins support the non-JTAG configuration scheme used in production. If you are only using JTAG configuration, you should connect the MSEL pins to ground.

You can download configuration data to Cyclone II FPGAs with the AS, PS, or JTAG interfaces using the options in Table 13–2.

**Table 13–2. Cyclone II Device Configuration Schemes**

Configuration Scheme	Description
AS configuration	Configuration using serial configuration devices (EPCS1, EPCS4, EPCS16 or EPCS64 devices)
PS configuration	Configuration using enhanced configuration devices (EPC4, EPC8, and EPC16 devices), EPC2 and EPC1 configuration devices, an intelligent host (microprocessor), or a download cable
JTAG-based configuration	Configuration via JTAG pins using a download cable, an intelligent host (microprocessor), or the Jam™ Standard Test and Programming Language (STAPL)

## Configuration File Format

Table 13–3 shows the approximate uncompressed configuration file sizes for Cyclone II devices. To calculate the amount of storage space required for multiple device configurations, add the file size of each device together.

Table 13–3. Cyclone II Raw Binary File (.rbf) Sizes		Note (1)
Device	Data Size (Bits)	Data Size (Bytes)
EP2C5	1,265,792	152,998
EP2C8	1,983,536	247,974
EP2C15	3,892,496	486,562
EP2C20	3,892,496	486,562
EP2C35	6,858,656	857,332
EP2C50	9,963,392	1,245,424
EP2C70	14,319,216	1,789,902

Note to Table 13–3:

(1) These values are preliminary.

Use the data in Table 13–3 only to estimate the file size before design compilation. Different configuration file formats, such as a Hexadecimal (.hex) or Tabular Text File (.tff) format, have different file sizes. However, for any specific version of the Quartus® II software, any design targeted for the same device has the same uncompressed configuration file size. If compression is used, the file size can vary after each compilation since the compression ratio is dependent on the design.

## Configuration Data Compression

Cyclone II devices support configuration data decompression, which saves configuration memory space and time. This feature allows you to store compressed configuration data in configuration devices or other memory and transmit this compressed bitstream to Cyclone II devices. During configuration, the Cyclone II device decompresses the bitstream in real time and programs its SRAM cells.



Preliminary data indicates that compression reduces configuration bitstream size by 35 to 55%.

Cyclone II devices support decompression in the AS and PS configuration schemes. Decompression is not supported in JTAG-based configuration.

Although they both use the same compression algorithm, the decompression feature supported by Cyclone II devices is different from the decompression feature in enhanced configuration devices (EPC16, EPC8, and EPC4 devices). The data decompression feature in the enhanced configuration devices allows them to store compressed data and decompress the bitstream before transmitting it to the target devices.

In PS mode, you should use the Cyclone II decompression feature since sending compressed configuration data reduces configuration time. You should not use both the Cyclone II device and the enhanced configuration device decompression features simultaneously. The compression algorithm is not intended to be recursive and could expand the configuration file instead of compressing it further.

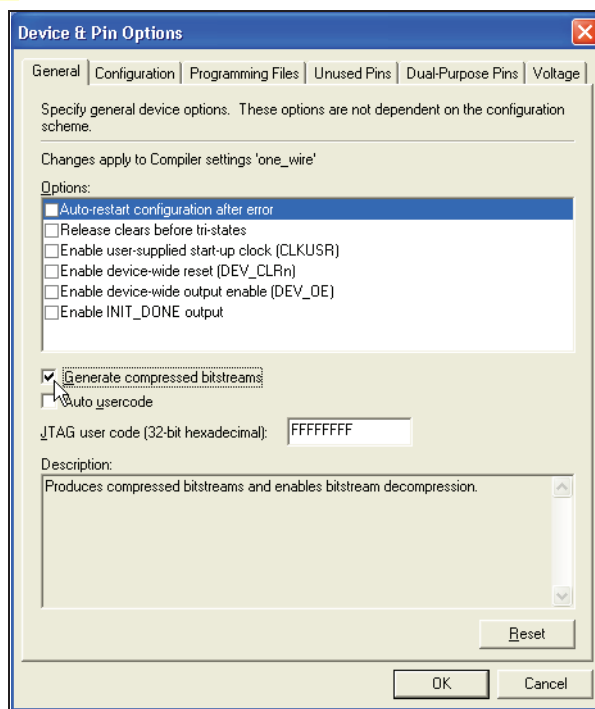
You should use the Cyclone II decompression feature during AS configuration if you need to save configuration memory space in the serial configuration device.

When you enable compression, the Quartus II software generates configuration files with compressed configuration data. This compressed file reduces the storage requirements in the configuration device or flash, and decreases the time needed to transmit the bitstream to the Cyclone II device. The time required by a Cyclone II device to decompress a configuration file is less than the time needed to transmit the configuration data to the FPGA.

There are two methods to enable compression for Cyclone II bitstreams: before design compilation (in the Compiler Settings menu) and after design compilation (in the **Convert Programming Files** window).

To enable compression in the project's compiler settings, select **Device** under the Assignments menu to bring up the settings window. After selecting your Cyclone II device open the **Device & Pin Options** window, and in the **General settings** tab enable the check box for **Generate compressed bitstreams** (see [Figure 13–1](#)).

**Figure 13–1. Enabling Compression for Cyclone II Bitstreams in Compiler Settings**

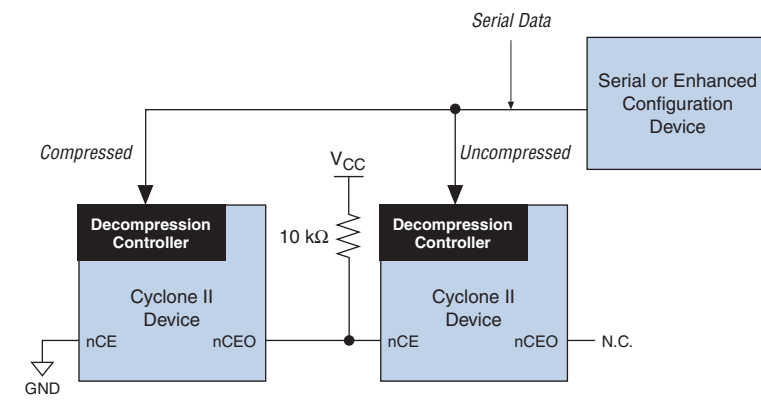


You can also use the following steps to enable compression when creating programming files from the Convert Programming Files window.

1. Click **Convert Programming Files** (File menu).
2. Select the Programming File type. Only Programmer Object Files (.pof), SRAM HEXOUT, RBF, or TTF files support compression.
3. For POFs, select a configuration device.
4. Select **Add File** and add a Cyclone II SRAM Object File(s) (.sof).
5. Select the name of the file you added to the SOF Data area and click on **Properties**.
6. Check the **Compression** check box.

When multiple Cyclone II devices are cascaded, the compression feature can be selectively enabled for each device in the chain. Figure 13–2 depicts a chain of two Cyclone II devices. The first Cyclone II device has compression enabled and therefore receives a compressed bitstream from the configuration device. The second Cyclone II device has the compression feature disabled and receives uncompressed data.

**Figure 13–2. Compressed & Uncompressed Configuration Data in a Programming File**



You can generate programming files (for example, POF files) for this setup in the Quartus II software.

### Active Serial Configuration (Serial Configuration Devices)

In the AS configuration scheme, Cyclone II devices are configured using a serial configuration device. These configuration devices are low-cost devices with non-volatile memory that feature a simple, four-pin interface and a small form factor. These features make serial configuration devices an ideal low-cost configuration solution.



For more information on serial configuration devices, see the *Serial Configuration Devices Data Sheet* in the Configuration Handbook.



Serial configuration devices provide a serial interface to access configuration data. During device configuration, Cyclone II devices read configuration data via the serial interface, decompress data if necessary, and configure their SRAM cells. The FPGA controls the configuration interface in the AS configuration scheme, while the external host (e.g., the configuration device or microprocessor) controls the interface in the PS configuration scheme.



The Cyclone II decompression feature is available when configuring your Cyclone II device using AS mode.

Table 13–4 shows the MSEL pin settings when using the AS configuration scheme.

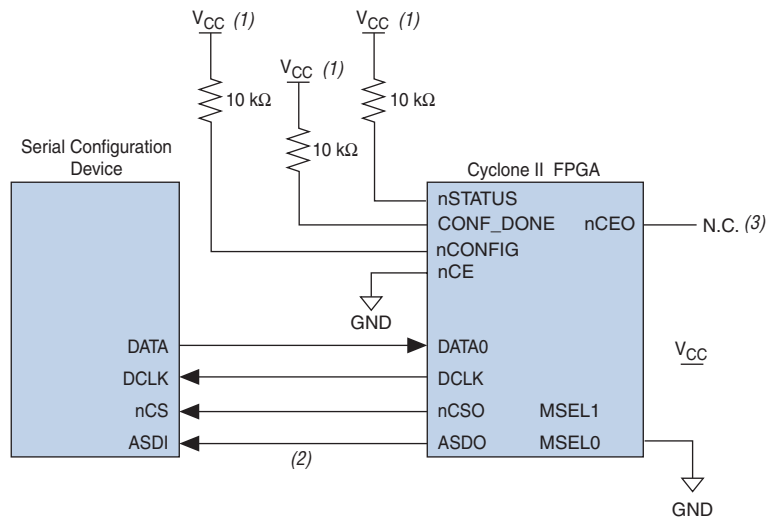
<b>Table 13–4. Cyclone II Configuration Schemes</b>		
<b>Configuration Scheme</b>	<b>MSEL1</b>	<b>MSEL0</b>
AS (20 MHz)	0	0
Fast AS (40 MHz) (1)	1	0

**Note to Table 13–4:**

- (1) Only the EPCS16 and EPCS64 devices support a DCLK up to 40 MHz clock; other EPCS devices support a DCLK up to 20 MHz. Refer to the *Serial Configuration Devices Data Sheet* for more information.

### Single Device AS Configuration

Serial configuration devices have a four-pin interface: serial clock input (DCLK), serial data output (DATA), AS data input (ASDI), and an active-low chip select ( $\overline{nCS}$ ). This four-pin interface connects to Cyclone II device pins, as shown in Figure 13–3.

**Figure 13–3. Single Device AS Configuration****Notes to Figure 13–3:**

- (1) Connect the pull-up resistors to a 3.3-V supply.
- (2) Cyclone II devices use the ASDO to ASDI path to control the configuration device.
- (3) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed another device's nCE pin.

Upon power-up, the Cyclone II device goes through a POR. During POR, the device resets, holds nSTATUS and CONF\_DONE low, and tri-states all user I/O pins. After POR, which typically lasts 100 ms, the Cyclone II device releases nSTATUS and enters configuration mode when the external 10-kΩ resistor pulls the nSTATUS pin high. Once the FPGA successfully exits POR, all user I/O pins continue to be tri-stated. Cyclone II devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.



The value of the weak pull-up resistors on the I/O pins that are on before and during configuration are available in the *DC Characteristics & Timing Specifications* chapter of the *Cyclone II Device Handbook*.

The configuration cycle consists of the reset, configuration, and initialization stages.

### Reset Stage

When  $\overline{nCONFIG}$  or  $\overline{nSTATUS}$  are low, the device is in reset. After POR, the Cyclone II device releases  $\overline{nSTATUS}$ . An external 10-k $\Omega$  pull-up resistor pulls the  $\overline{nSTATUS}$  signal high, and the Cyclone II device enters configuration mode.



$V_{CCINT}$  and  $V_{CCIO}$  of the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

### Configuration Stage

The serial clock (DCLK) generated by the Cyclone II device controls the entire configuration cycle and provides the timing for the serial interface. Cyclone II devices use an internal oscillator to generate DCLK. Using the MSEL[ ] pins, you can select either a 20- or 40-MHz oscillator. Although you can select either 20- or 40-MHz oscillator when designing with serial configuration devices, the 40-MHz oscillator provides faster configuration times. There is some variation in the internal oscillator frequency because of the process, temperature, and voltage conditions in Cyclone II devices. The internal oscillator is designed such that its maximum frequency is guaranteed to meet EPCS device specifications.

Table 13–5 shows the AS DCLK output frequencies.

<b>Table 13–5. AS DCLK Output Frequency</b> <i>Note (1)</i>				
Oscillator Selected	Minimum	Typical	Maximum	Units
40 MHz	20	26	40	MHz
20 MHz	10	13	20	MHz

*Note to Table 13–5:*

(1) These values are preliminary.

In both AS and Fast AS configuration schemes, the serial configuration device latches input and control signals on the rising edge of DCLK and drives out configuration data on the falling edge. Cyclone II devices drive out control signals on the falling edge of DCLK and latch configuration data on the falling edge of DCLK.

In configuration mode, the Cyclone II device enables the serial configuration device by driving its  $\overline{nCSO}$  output pin low, which connects to the chip select ( $\overline{nCS}$ ) pin of the configuration device. The Cyclone II device uses the serial clock (DCLK) and serial data output (ASDO) pins to send operation commands and/or read address signals to the serial

configuration device. The configuration device then provides data on its serial data output (DATA) pin, which connects to the DATA0 input of the Cyclone II device.

After the Cyclone II device receives all the configuration bits, it releases the open-drain CONF\_DONE pin, which is then pulled high by an external 10-k $\Omega$  resistor. Also, the Cyclone II device stops driving the DCLK signal. Initialization begins only after the CONF\_DONE signal reaches a logic high level. The CONF\_DONE pin must have an external 10-k $\Omega$  pull-up resistor in order for the device to initialize. All AS configuration pins (DATA0, DCLK, nCS0, and ASDO) have weak internal pull-up resistors which are always active. After configuration, these pins are set as input tri-stated and are pulled high by the internal weak pull-up resistors.

### *Initialization Stage*

In Cyclone II devices, the initialization clock source is either the Cyclone II 10-MHz (typical) internal oscillator (separate from the AS internal oscillator) or the optional CLKUSR pin. The internal oscillator is the default clock source for initialization. If the internal oscillator is used, the Cyclone II device provides itself with enough clock cycles for proper initialization. The advantage of using the internal oscillator is you do not need to send additional clock cycles from an external source to the CLKUSR pin during the initialization stage. Additionally, you can use the CLKUSR pin as a user I/O pin.

If you want to delay the initialization of the device, you can use the CLKUSR pin option. Using the CLKUSR pin allows you to control when your device enters user mode. The device can be delayed from entering user mode for an indefinite amount of time. When you enable the **User Supplied Start-Up Clock** option, the CLKUSR pin is the initialization clock source. Supplying a clock on CLKUSR does not affect the configuration process. After all configuration data has been accepted and CONF\_DONE goes high, Cyclone II devices require 299 clock cycles to initialize properly and support a CLKUSR  $f_{MAX}$  of 100 MHz.

Cyclone II devices offer an optional `INIT_DONE` pin which signals the end of initialization and the start of user mode with a low-to-high transition. The **Enable `INIT_DONE` output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** window. If you use the `INIT_DONE` pin, an external 10-k $\Omega$  pull-up resistor is required to pull the signal high when `nCONFIG` is low and during the beginning of configuration. Once the optional bit to enable `INIT_DONE` is programmed into the device (during the first frame of configuration data), the `INIT_DONE` pin goes low. When initialization is complete, the `INIT_DONE` pin is released and pulled high. This low-to-high transition signals that the FPGA has entered user mode. If you do not use the `INIT_DONE` pin, the initialization period is complete after `CONF_DONE` goes high and 299 clock cycles are sent to the `CLKUSR` pin or after the time  $t_{CF2UM}$  (see Table 13–8) if the Cyclone II device uses the internal oscillator.

### User Mode

When initialization is complete, the FPGA enters user mode. In user mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

When the Cyclone II device is in user mode, you can initiate reconfiguration by pulling the `nCONFIG` signal low. The `nCONFIG` signal should be low for at least 2  $\mu$ s. When `nCONFIG` is pulled low, the Cyclone II device is reset and enters the reset stage. The Cyclone II device also pulls `nSTATUS` and `CONF_DONE` low and all I/O pins are tri-stated. Once `nCONFIG` returns to a logic high level and `nSTATUS` is released by the Cyclone II device, reconfiguration begins.

### Error During Configuration

If an error occurs during configuration, the Cyclone II device drives the `nSTATUS` signal low to indicate a data frame error, and the `CONF_DONE` signal stays low. If you enable the **Auto-restart configuration after error** option in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box, the Cyclone II device resets the serial configuration device by pulsing `nCS0`, releases `nSTATUS` after a reset time-out period (about 40  $\mu$ s), and retries configuration. If the **Auto-restart configuration after error** option is turned off, the external system must monitor `nSTATUS` for errors and then pull `nCONFIG` low for at least 2  $\mu$ s to restart configuration.



If you use the optional `CLKUSR` pin and the `nCONFIG` pin is pulled low to restart configuration during device initialization, ensure `CLKUSR` continues to toggle during the time `nSTATUS` is low (a maximum of 40  $\mu$ s).



For more information on configuration issues, see the *Debugging Configuration Problems* chapter of the *Configuration Handbook* and the FPGA Configuration Troubleshooter on the Altera web site ([www.altera.com](http://www.altera.com)).

### Multiple Device AS Configuration

You can configure multiple Cyclone II devices using a single serial configuration device. You can cascade multiple Cyclone II devices using the chip-enable ( $nCE$ ) and chip-enable-out ( $nCEO$ ) pins. Connect the  $nCE$  pin of the first device in the chain to ground and connect the  $nCEO$  pin to the  $nCE$  pin of the next device in the chain. Use an external 10-k $\Omega$  pull-up resistor to pull the  $nCEO$  signal high to its  $V_{CCIO}$  level to help the internal weak pull-up resistor. When the first device captures all of its configuration data from the bitstream, it transitions its  $nCEO$  pin low, initiating the configuration of the next device in the chain. You can leave the  $nCEO$  pin of the last device unconnected or use it as a user I/O pin after configuration if the last device in chain is a Cyclone II device.



The Quartus II software sets the Cyclone II device  $nCEO$  pin as an output pin driving to ground by default. If the device is in a chain, and the  $nCEO$  pin is connected to the next device's  $nCE$  pin, you must make sure that the  $nCEO$  pin is not used as a user I/O pin after configuration. The software setting is in the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box in Quartus II software.

The first Cyclone II device in the chain is the configuration master and controls the configuration of the entire chain. Select the AS configuration scheme for the first Cyclone II device and the PS configuration scheme for the remaining Cyclone II devices (configuration slaves). Any other Altera® device that supports PS configuration can also be part of the chain as a configuration slave. In a multiple device chain, the  $nCONFIG$ ,  $nSTATUS$ ,  $CONF\_DONE$ ,  $DCLK$ , and  $DATA0$  pins of each device in the chain are connected (see Figure 13–4). Figure 13–4 shows the pin connections for this setup.



- As shown in [Figure 13-4](#), the `nSTATUS` and `CONF_DONE` pins on all target FPGAs are connected together with external pull-up resistors. These pins are open-drain bidirectional pins on the FPGAs. When the first device asserts `nCEO` (after receiving all of its configuration data), it releases its `CONF_DONE` pin. However, the subsequent devices in the chain keep the `CONF_DONE` signal low until they receive their configuration data. When all the target FPGAs in the chain have received their configuration data and have released `CONF_DONE`, the pull-up resistor pulls this signal high, and all devices simultaneously enter initialization mode.

During initialization, the initialization clock source is either the Cyclone II 10 MHz (typical) internal oscillator (separate from the AS internal oscillator) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Cyclone II device provides itself with enough clock cycles for proper initialization. The advantage of using the internal oscillator is you do not need to send additional clock cycles from an external source to the CLKUSR pin during the initialization stage. You can also make use of the CLKUSR pin as a user I/O pin, which means you have an additional user I/O pin.

If you want to delay the initialization of the devices in the chain, you can use the CLKUSR pin option. The CLKUSR pin allows you to control when your device enters user mode. This feature also allows you to control the order of when each device enters user mode by feeding a separate clock to each device's CLKUSR pin. By using the CLKUSR pins, you can choose any device in the multiple device chain to enter user mode first and have the other devices enter user mode at a later time.

Different device families may require a different number of initialization clock cycles. Therefore, if your multiple device chain consists of devices from different families, the devices may enter user mode at a slightly different time due to the different number of initialization clock cycles required. However, if the number of initialization clock cycles is similar across different device families or if the devices are from the same family, then the devices enter user mode at the same time. See the respective device family handbook for more information about the number of initialization clock cycles required.

If an error occurs at any point during configuration, the FPGA with the error drives the nSTATUS signal low. If you enable the **Auto-restart configuration after error** option, the entire chain begins reconfiguration after a reset time-out period (a maximum of 40  $\mu$ s). If the **Auto-restart configuration after error** option is turned off, a microprocessor or controller must monitor nSTATUS for errors and then pulse nCONFIG low to restart configuration. The microprocessor or controller can pulse nCONFIG if it is under system control rather than tied to V<sub>CC</sub>.



While you can cascade Cyclone II devices, serial configuration devices cannot be cascaded or chained together.



If you use the optional CLKUSR pin and the nCONFIG is pulled low to restart configuration during device initialization, make sure the CLKUSR pin continues to toggle while nSTATUS is low (a maximum of 40  $\mu$ s).



If the configuration bitstream size exceeds the capacity of a serial configuration device, you must select a larger configuration device and/or enable the compression feature. When configuring multiple devices, the size of the bitstream is the sum of the individual devices' configuration bitstreams.

### Configuring Multiple Cyclone II Devices with the Same Design

Certain designs require you to configure multiple Cyclone II devices with the same design through a configuration bitstream or SOF. You can do this through one of two methods, as described in this section. For both methods, the serial configuration devices cannot be cascaded or chained together.

#### *Multiple SOFs*

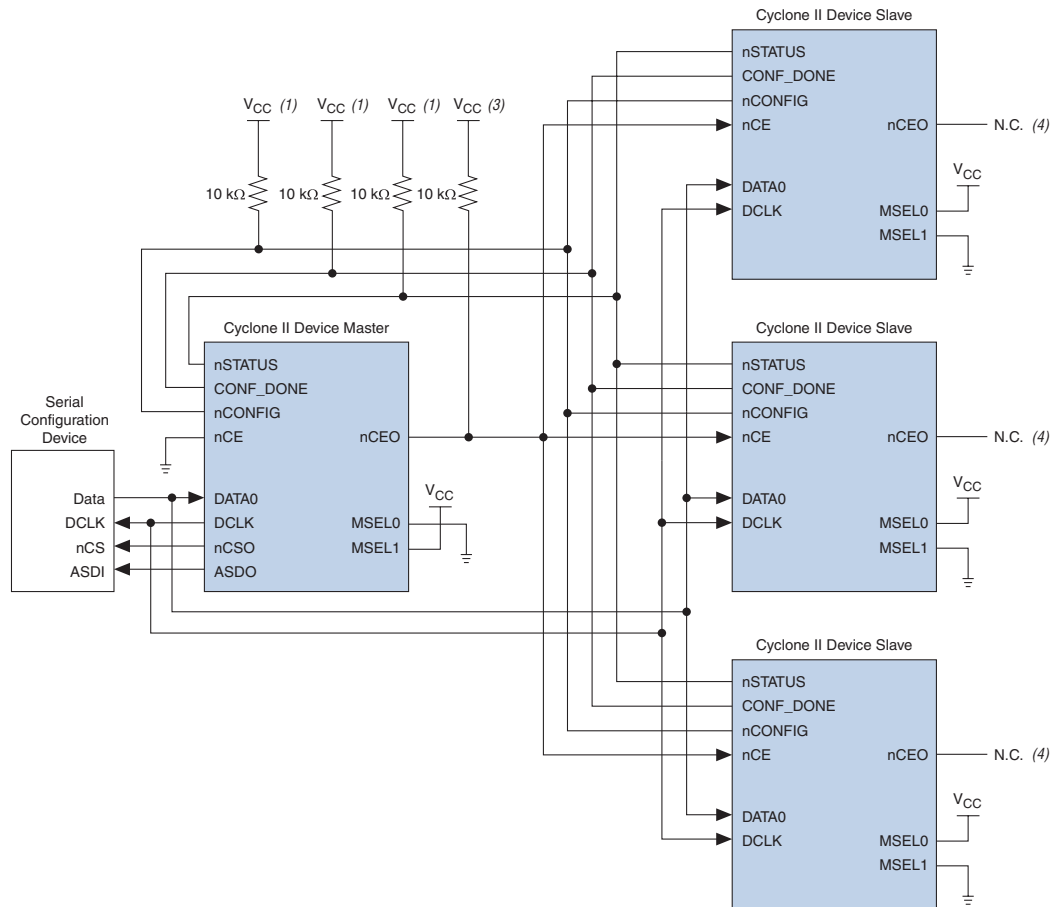
In the first method, two copies of the SOF file are stored in the serial configuration device. Use the first copy to configure the master Cyclone II device and the second copy to configure all remaining slave devices concurrently. In this setup, the master Cyclone II device is in AS mode, and the slave Cyclone II devices are in PS mode ( $MSEL=01$ ). See [Figure 13–5](#).

To configure four identical Cyclone II devices with the same SOF file, connect the three slave devices for concurrent configuration as shown in [Figure 13–5](#). The  $nCEO$  pin from the master device drives the  $nCE$  input pins on all three slave devices. Connect the configuration device's  $DATA$  and  $DCLK$  pins to the Cyclone II device's  $DATA$  and  $DCLK$  pins in parallel. During the first configuration cycle, the master device reads its configuration data from the serial configuration device while holding  $nCEO$  high. After completing its configuration cycle, the master drives  $nCE$  low and transmits the second copy of the configuration data to all three slave devices, configuring them simultaneously.

The advantage of using the setup in [Figure 13–5](#) is that you can have a different SOF file for the Cyclone II master device. However, all the Cyclone II slave devices must be configured with the same SOF file. The SOF files in this configuration method can be either compressed or uncompressed.



You can still use this method if the master and slave Cyclone II devices use the same SOF.



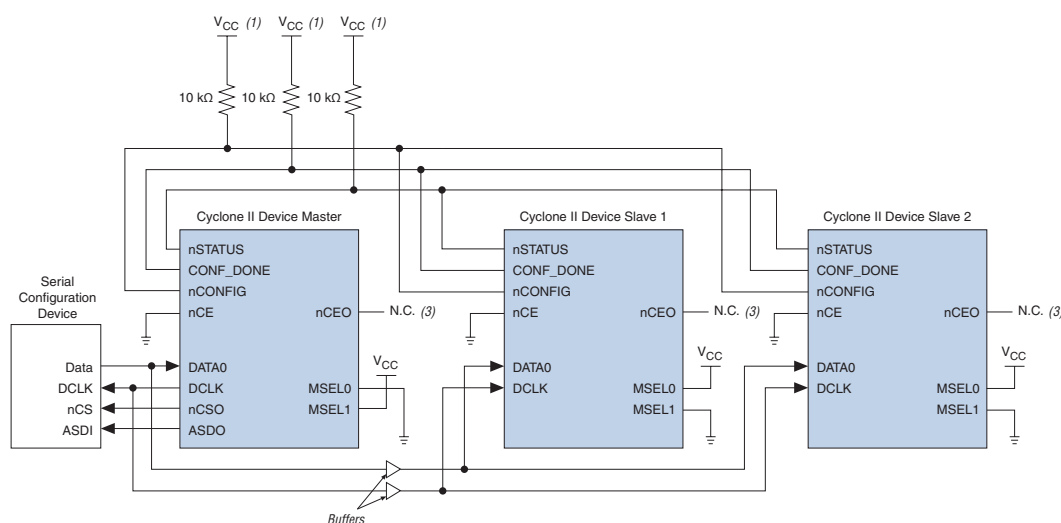
(1) Connect the pull up resistors to a 3.3 V supply

-

### Single SOF

The second method configures both the master and slave Cyclone II devices with the same SOF. The serial configuration device stores one copy of the SOF file. This setup is shown in Figure 13–6 where the master is setup in AS mode, and the slave devices are setup in PS mode (MSEL=01). You could setup one or more slave devices in the chain and all the slave devices are setup in the same way as shown in Figure 13–6.

**Figure 13–6. Multiple Device AS Configuration When FPGAs Receive the Same Data with a Single SOF**



**Notes to Figure 13–6:**

- (1) Connect the pull-up resistors to a 3.3-V supply.
- (2) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed another device's nCE pin.

In this setup, all the Cyclone II devices in the chain are connected for concurrent configuration. This can reduce the AS configuration time because all the Cyclone II devices are configured in one configuration cycle. Connect the nCE input pins of all the Cyclone II devices to ground. You can either leave the nCEO output pins on all the Cyclone II devices unconnected or use the nCEO output pins as normal user I/O pins. The DATA and DCLK pins are connected in parallel to all the Cyclone II devices.

You should put a buffer before the DATA and DCLK output from the master Cyclone II device to avoid signal strength and signal integrity issues. The buffer should not significantly change the DATA-to-DCLK relationships or delay them with respect to other AS signals (ASDI and nCS). Also, the buffer should only drive the slave Cyclone II devices, so that the timing between the master Cyclone II device and serial configuration device is unaffected.

This configuration method supports both compressed and uncompressed SOFs. Therefore, if the configuration bitstream size exceeds the capacity of a serial configuration device, you can enable the compression feature in the SOF file used or you can select a larger serial configuration device.

### Estimating AS Configuration Time

The AS configuration time is the time it takes to transfer data from the serial configuration device to the Cyclone II device. The Cyclone II DCLK output (generated from an internal oscillator) clocks this serial interface. As listed in Table 13–5, if you are using the 40-MHz oscillator, the DCLK minimum frequency is 20 MHz (50 ns). Therefore, the maximum configuration time estimate for an EP2C5 device (1,223,980 bits of uncompressed data) is:

$$\text{RBF size} \times (\text{maximum DCLK period} / 1 \text{ bit per DCLK cycle}) = \text{estimated maximum configuration time}$$

$$1,223,980 \text{ bits} \times (50 \text{ ns} / 1 \text{ bit}) = 61.2 \text{ ms}$$

To estimate the typical configuration time, use the typical DCLK period listed in Table 13–5. With a typical DCLK period of 38.46 ns, the typical configuration time is 47.1 ms. Enabling compression reduces the amount of configuration data that is transmitted to the Cyclone II device, which also reduces configuration time. On average, compression reduces configuration time by 50%.

## Programming Serial Configuration Devices

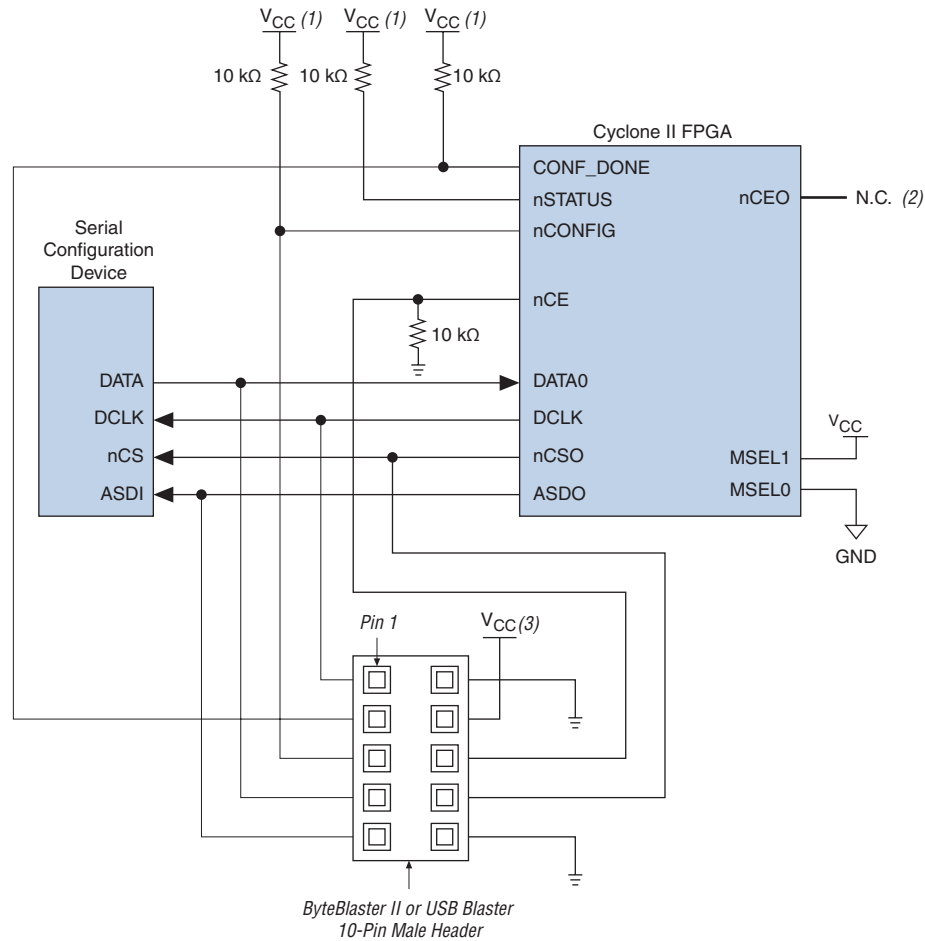
Serial configuration devices are non-volatile, flash-memory-based devices. You can program these devices in-system using the USB-Blaster™ or ByteBlaster™ II download cable. Alternatively, you can program them using the Altera Programming Unit (APU), supported third-party programmers, or a microprocessor with the SRunner software driver.

You can use the AS programming interface to program serial configuration devices in-system. During in-system programming, the download cable disables FPGA access to the AS interface by driving the nCE pin high. Cyclone II devices are also held in reset by pulling the nCONFIG signal low. After programming is complete, the download cable releases the nCE and nCONFIG signals, allowing the pull-down and pull-up resistor to drive GND and V<sub>CC</sub>, respectively. [Figure 13–7](#) shows the download cable connections to the serial configuration device.



For more information on the USB-Blaster download cable, see the *USB-Blaster USB Port Download Cable Data Sheet*. For more information on the ByteBlaster II cable, see the *ByteBlaster II Download Cable Data Sheet*.

**Figure 13–7. In-System Programming of Serial Configuration Devices**



**Notes to Figure 13–7:**

- (1) Connect these pull-up resistors to 3.3-V supply.
- (2) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed other device's nCE pin.
- (3) Power up the ByteBlaster II or USB Blaster cable's V<sub>CC</sub> with a 3.3-V supply.

You can use the Quartus II software with the APU and the appropriate configuration device programming adapter to program serial configuration devices. All serial configuration devices are offered in an 8-pin or 16-pin small outline integrated circuit (SOIC) package and can be programmed using the PLMSEPC-8 adapter.

Altera programming hardware (APU) or other third-party programming hardware can be used to program blank serial configuration devices before they are mounted onto PCBs. Alternatively, you can use an on-board microprocessor to program the serial configuration device on the PCB using C-based software drivers provided by Altera (i.e., the SRunner software driver).

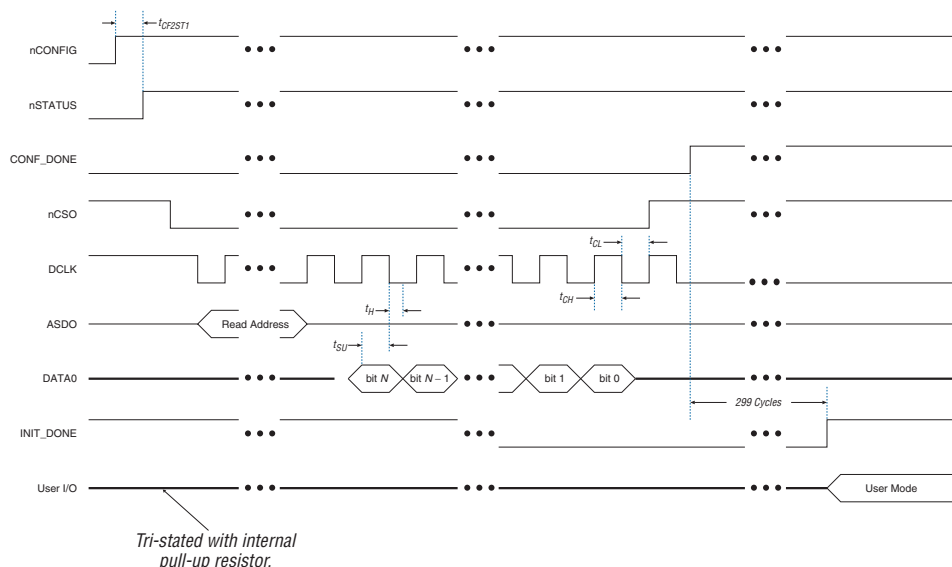
A serial configuration device can be programmed in-system by an external microprocessor using SRunner. SRunner is a software driver developed for embedded serial configuration device programming, which can be easily customized to fit in different embedded systems. SRunner can read a Raw Programming Data File (.rpd) and write to the serial configuration devices. The serial configuration device programming time using SRunner is comparable to the programming time when using the Quartus II Programmer.



For more information about SRunner, see the *SRunner: An Embedded Solution for EPCS Programming White Paper* and the source code on the Altera web site at [www.altera.com](http://www.altera.com). For more information on programming serial configuration devices, see the *Serial Configuration Devices Data Sheet* in the *Configuration Handbook*.

Figure 13–8 shows the timing waveform for the AS configuration scheme using a serial configuration device.

**Figure 13–8. AS Configuration Timing**



## PS Configuration

You can use an Altera configuration device, a download cable, or an intelligent host, such as a MAX<sup>®</sup> II device or microprocessor to configure a Cyclone II device with the PS scheme. In the PS scheme, an external host (configuration device, MAX II device, embedded processor, or host PC) controls configuration. Configuration data is input to the target Cyclone II devices via the `DATA0` pin at each rising edge of `DCLK`.



The Cyclone II decompression feature is fully available when configuring your Cyclone II device using PS mode.

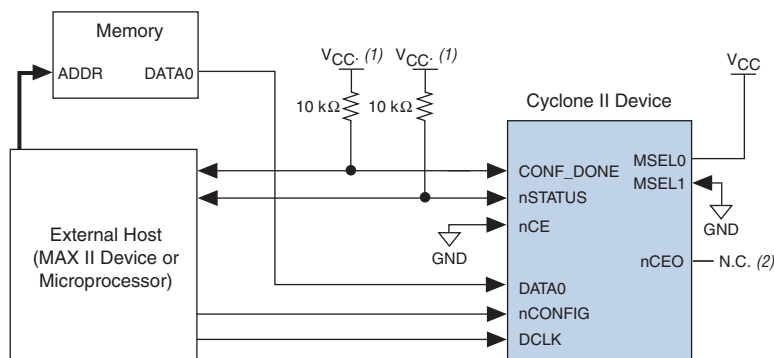
Table 13–6 shows the `MSEL` pin settings when using the PS configuration scheme.

<i>Table 13–6. Cyclone II MSEL Pin Settings for PS Configuration Schemes</i>		
Configuration Scheme	MSEL1	MSEL0
PS	0	1

### Single Device PS Configuration Using a MAX II Device as an External Host

In the PS configuration scheme, you can use a MAX II device as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Cyclone II device. Configuration data can be stored in RBF, HEX, or TTF format. Figure 13–9 shows the configuration interface connections between the Cyclone II device and a MAX II device for single device configuration.



**Figure 13–9. Single Device PS Configuration Using an External Host****Notes to Figure 13–9:**

- (1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the device.  $V_{CC}$  should be high enough to meet the VIH specification of the I/O on the device and the external host.
- (2) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed other device's nCE pin.

Upon power-up, the Cyclone II device goes through a POR, which lasts approximately 100 ms. During POR, the device resets, holds nSTATUS low, and tri-states all user I/O pins. Once the FPGA successfully exits POR, all user I/O pins continue to be tri-stated.



The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *Cyclone II Device Handbook*.

The configuration cycle consists of three stages: reset, configuration, and initialization.

**Reset Stage**

While the Cyclone II device's nCONFIG or nSTATUS pins are low, the device is in reset. To initiate configuration, the MAX II device must transition the Cyclone II nCONFIG pin from low to high.



$V_{CCINT}$  and  $V_{CCIO}$  of the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When the Cyclone II nCONFIG pin transitions high, the Cyclone II device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. Once nSTATUS is released, the FPGA is ready to receive configuration data and the MAX II device can start the configuration at any time.

### *Configuration Stage*

After the Cyclone II device's `nSTATUS` pin transitions high, the MAX II device should send the configuration data on the `DATA0` pin one bit at a time. If you are using configuration data in RBF, HEX, or TTF format, send the least significant bit (LSB) of each data byte first. For example, if the RBF contains the byte sequence 02 1B EE 01 FA, you should transmit the serial bitstream 0100-0000 1101-1000 0111-0111 1000-0000 0101-1111 to the device first.

The Cyclone II device receives configuration data on its `DATA0` pin and the clock on the `DCLK` pin. Data is latched into the FPGA on the rising edge of `DCLK`. Data is continuously clocked into the target device until the `CONF_DONE` pin transitions high. After the Cyclone II device receives all the configuration data successfully, it releases the open-drain `CONF_DONE` pin, which is pulled high by an external 10-k $\Omega$  pull-up resistor. A low-to-high transition on `CONF_DONE` indicates configuration is complete and initialization of the device can begin. The `CONF_DONE` pin must have an external 10-k $\Omega$  pull-up resistor in order for the device to initialize.

The configuration clock (`DCLK`) speed must be below the specified system frequency (see [Table 13-7](#)) to ensure correct configuration. No maximum `DCLK` period exists, which means you can pause configuration by halting `DCLK` for an indefinite amount of time.

### *Initialization Stage*

In Cyclone II devices, the initialization clock source is either the Cyclone II internal oscillator (typically 10 MHz) or the optional `CLKUSR` pin. The internal oscillator is the default clock source for initialization. If you use the internal oscillator, the Cyclone II device makes sure to provide enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. You do not need to provide additional clock cycles externally during the initialization stage. Driving `DCLK` back to the device after configuration is complete does not affect device operation. Additionally, if you use the internal oscillator as the clock source, you can use the `CLKUSR` pin as a user I/O pin.

If you want to delay the initialization of the device, you can use the `CLKUSR` pin. Using the `CLKUSR` pin allows you to control when your device enters user mode. You can delay the device from entering user mode for an indefinite amount of time.

The **Enable user-supplied start-up clock (CLKUSR)** option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR does not affect the configuration process. After all configuration data has been accepted and CONF\_DONE goes high, Cyclone II devices require 299 clock cycles to initialize properly and support a CLKUSR  $f_{MAX}$  of 100 MHz.



If the optional CLKUSR pin is being used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure that CLKUSR continues toggling during the time nSTATUS is low (maximum of 40  $\mu$ s).

An optional INIT\_DONE pin signals the end of initialization and the start of user mode with a low-to-high transition. By default, the INIT\_DONE output is disabled. You can enable the INIT\_DONE output by turning on the **Enable INIT\_DONE output** option in the Quartus II software. If you use the INIT\_DONE pin, an external 10-k $\Omega$  pull-up resistor pulls the pin high when nCONFIG is low and during the beginning of configuration. Once the optional bit to enable INIT\_DONE is programmed into the device (during the first frame of configuration data), the INIT\_DONE pin transitions low. When initialization is complete, the INIT\_DONE pin is released and pulled high. The MAX II device must be able to detect this low-to-high transition, which signals the FPGA has entered user mode.

If you want to use the INIT\_DONE pin as a user I/O pin, you should wait for the maximum value of  $t_{CD2UM}$  (see Table 13-7) after the CONF\_DONE signal transitions high so to ensure the Cyclone II device has been initialized properly and is in user mode.

Make sure the MAX II device does not drive the CONF\_DONE signal low during configuration, initialization, and before the device enters user mode.

### *User Mode*

When initialization is complete, the Cyclone II device enters user mode. In user mode, the user I/O pins no longer have pull-up resistors and function as assigned in your design.

To ensure DCLK and DATA0 are not left floating at the end of configuration, the MAX II device must drive them either high or low, whichever is convenient on your PCB. The Cyclone II device DATA0 pin is not available as a user I/O pin after configuration.

When the FPGA is in user mode, you can initiate a reconfiguration by transitioning the nCONFIG pin low-to-high. The nCONFIG pin must be low for at least 2  $\mu$ s. When the nCONFIG transitions low, the Cyclone II

device also pulls `nSTATUS` and `CONF_DONE` low and tri-states all I/O pins. Once the `nCONFIG` pin returns to a logic high level and the Cyclone II device releases the `nSTATUS` pin, the MAX II device can begin reconfiguration.

### *Error During Configuration*

If an error occurs during configuration, the Cyclone II device transitions its `nSTATUS` pin low, resetting itself internally. The low signal on the `nSTATUS` pin tells the MAX II device that there is an error. If you turn on the **Auto-restart configuration after error** option in the Quartus II software, the Cyclone II device releases `nSTATUS` after a reset time-out period (maximum of 40  $\mu$ s). After `nSTATUS` is released and pulled high by a pull-up resistor, the MAX II device can try to reconfigure the target device without needing to pulse `nCONFIG` low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2  $\mu$ s) on `nCONFIG` to restart the configuration process.

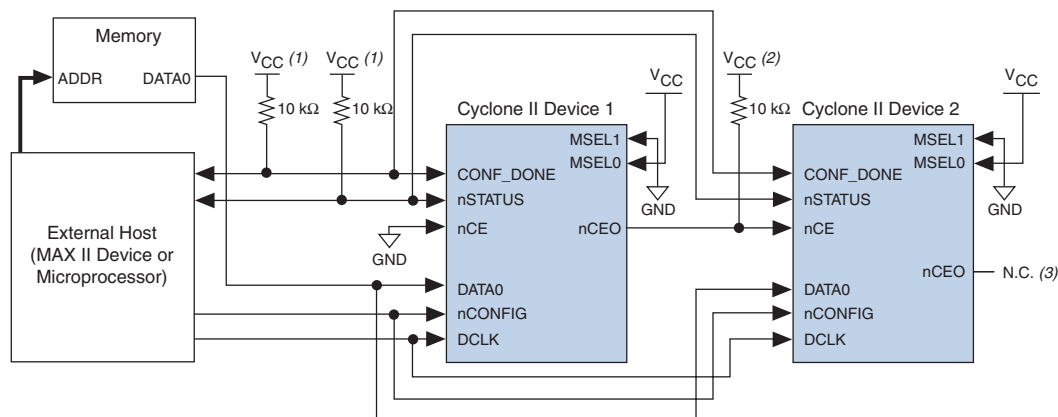
The MAX II device can also monitor the `CONF_DONE` and `INIT_DONE` pins to ensure successful configuration. The MAX II device must monitor the Cyclone II device's `CONF_DONE` pin to detect errors and determine when programming completes. If all configuration data is sent, but `CONF_DONE` or `INIT_DONE` do not transition high, the MAX II device must reconfigure the target device.



For more information on configuration issues, see the *Debugging Configuration Problems* chapter of the *Configuration Handbook* and the FPGA Configuration Troubleshooter on the Altera web site ([www.altera.com](http://www.altera.com)).

### **Multiple Device PS Configuration Using a MAX II Device as an External Host**

Figure 13–10 shows how to configure multiple devices using a MAX II device. This circuit is similar to the PS configuration circuit for a single device, except Cyclone II devices are cascaded for multiple device configuration.

**Figure 13–10. Multiple Device PS Configuration Using an External Host****Notes to Figure 13–10:**

- (1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.  $V_{CC}$  should be high enough to meet the  $V_{IH}$  specification of the I/O on the devices and the external host.
- (2) Connect the pull-up resistor to the  $V_{CCIO}$  supply voltage of I/O bank that the  $nCEO$  pin resides in.
- (3) The  $nCEO$  pin can be left unconnected or used as a user I/O pin when it does not feed another device's  $nCE$  pin.

In multiple device PS configuration, connect the first Cyclone II device's  $nCE$  pin to GND and connect the  $nCEO$  pin to the  $nCE$  pin of the next Cyclone II device in the chain. Use an external 10-k $\Omega$  pull-up resistor to pull the Cyclone II device's  $nCEO$  pin high to its  $V_{CCIO}$  level to help the internal weak pull-up resistor when the  $nCEO$  pin feeds next Cyclone II device's  $nCE$  pin. The input to the  $nCE$  pin of the last Cyclone II device in the chain comes from the previous Cyclone II device. After the first device completes configuration in a multiple device configuration chain, its  $nCEO$  pin transitions low to activate the second device's  $nCE$  pin, which prompts the second device to begin configuration within one clock cycle. Therefore, the MAX II device begins to transfer data to the next Cyclone II device without interruption. The  $nCEO$  pin is a dual-purpose pin in Cyclone II devices. You can leave the  $nCEO$  pin of the last device unconnected or use it as a user I/O pin after configuration if the last device in chain is a Cyclone II device.



The Quartus II software sets the Cyclone II device  $nCEO$  pin as a dedicated output by default. If the  $nCEO$  pin feeds the next device's  $nCE$  pin, you must make sure that the  $nCEO$  pin is not used as a user I/O after configuration. This software setting is in the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box in Quartus II software.

You must connect all other configuration pins (`nCONFIG`, `nSTATUS`, `DCLK`, `DATA0`, and `CONF_DONE`) to every Cyclone II device in the chain. The configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. You should buffer the `DCLK` and `DATA` lines for every fourth device. Because all device `CONF_DONE` pins are tied together, all devices initialize and enter user mode at the same time.

Since all `nSTATUS` and `CONF_DONE` pins are connected, if any Cyclone II device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first Cyclone II detects an error, it resets the chain by pulling its `nSTATUS` pin low. This behavior is similar to a single Cyclone II device detecting an error.

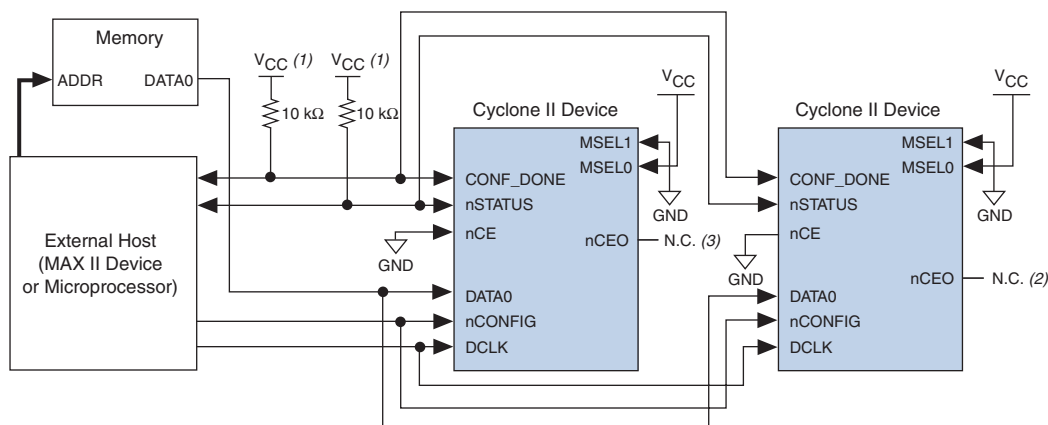
If the **Auto-restart configuration after error** option is turned on, the Cyclone II devices release their `nSTATUS` pins after a reset time-out period (maximum of 40  $\mu$ s). After all `nSTATUS` pins are released and pulled high, the MAX II device reconfigures the chain without pulsing `nCONFIG` low. If the **Auto-restart configuration after error** option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2  $\mu$ s) on `nCONFIG` to restart the configuration process.

If you want to delay the initialization of the devices in the chain, you can use the `CLKUSR` pin option. The `CLKUSR` pin allows you to control when your device enters user mode. This feature also allows you to control the order of when each device enters user mode by feeding a separate clock to each device's `CLKUSR` pin. By using the `CLKUSR` pins, you can choose any device in the multiple device chain to enter user mode first and have the other devices enter user mode at a later time.

Different device families may require a different number of initialization clock cycles. Therefore, if your multiple device chain consists of devices from different families, the devices may enter user mode at a slightly different time due to the different number of initialization clock cycles required. However, if the number of initialization clock cycles is similar across different device families or if the devices are from the same family, then the devices enter user mode at the same time. See the respective device family handbook for more information about the number of initialization clock cycles required.

If your system has multiple Cyclone II devices (in the same density and package) with the same configuration data, you can configure them in one configuration cycle by connecting all device's `nCE` pins to ground and connecting all the Cyclone II device's configuration pins (`nCONFIG`, `nSTATUS`, `DCLK`, `DATA0`, and `CONF_DONE`) together. You can also use the `nCEO` pin as a user I/O pin after configuration. The configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Make sure the `DCLK` and `DATA` lines are buffered for every fourth device. All devices start and complete configuration at the same time. Figure 13–11 shows multiple device PS configuration when both Cyclone II devices are receiving the same configuration data.

**Figure 13–11. Multiple Device PS Configuration When Both FPGAs Receive the Same Data**



**Notes to Figure 13–11:**

- (1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.  $V_{CC}$  should be high enough to meet the  $V_{IH}$  specification of the I/O on the devices and the external host.
- (2) The `nCEO` pins of both devices can be left unconnected or used as user I/O pins when configuring the same configuration data into multiple devices.

You can use a single configuration chain to configure Cyclone II devices with other Altera devices. Connect all the Cyclone II device's and all other Altera device's `CONF_DONE` and `nSTATUS` pins together so all devices in the chain complete configuration at the same time or that an error reported by one device initiates reconfiguration in all devices.



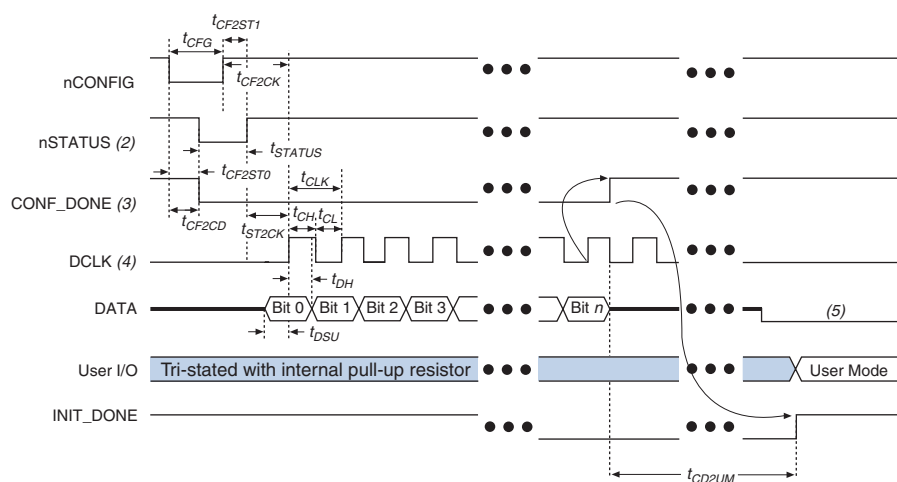
For more information on configuring multiple Altera devices in the same configuration chain, see *Configuring Mixed Altera FPGA Chains* in the *Configuration Handbook*.

### PS Configuration Timing

A PS configuration must meet the setup and hold timing parameters and the maximum clock frequency. When using a microprocessor or another intelligent host to control the PS interface, ensure that you meet these timing requirements.

Figure 13–12 shows the timing waveform for PS configuration for Cyclone II devices.

**Figure 13–12. PS Configuration Timing Waveform** *Note (1)*



#### Notes to Figure 13–12:

- (1) The beginning of this waveform shows the device in user mode. In user mode, nCONFIG, nSTATUS and CONF\_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
- (2) Upon power-up, the Cyclone II device holds nSTATUS low for the time of the POR delay.
- (3) Upon power-up, before and during configuration, CONF\_DONE is low.
- (4) In user mode, drive DCLK either high or low when using the PS configuration scheme, whichever is more convenient. When using the AS configuration scheme, DCLK is a Cyclone II output pin and should not be driven externally.
- (5) Do not leave the DATA pin floating after configuration. Drive it high or low, whichever is more convenient.



Table 13–7 defines the timing parameters for Cyclone II devices for PS configuration.

<b>Table 13–7. PS Timing Parameters for Cyclone II Devices</b>				
<b>Symbol</b>	<b>Parameter</b>	<b>Minimum</b>	<b>Maximum</b>	<b>Units</b>
$t_{POR}$	POR delay (1)	100		ms
$t_{CF2CD}$	nCONFIG low to CONF_DONE low		800	ns
$t_{CF2ST0}$	nCONFIG low to nSTATUS low		800	ns
$t_{CFG}$	nCONFIG low pulse width	2		$\mu$ s
$t_{STATUS}$	nSTATUS low pulse width	10	40 (2)	$\mu$ s
$t_{CF2ST1}$	nCONFIG high to nSTATUS high		40 (2)	$\mu$ s
$t_{CF2CK}$	nCONFIG high to first rising edge on DCLK	40		$\mu$ s
$t_{ST2CK}$	nSTATUS high to first rising edge on DCLK	1		$\mu$ s
$t_{DSU}$	Data setup time before rising edge on DCLK	7		ns
$t_{DH}$	Data hold time after rising edge on DCLK	0		ns
$t_{CH}$	DCLK high time	4		ns
$t_{CL}$	DCLK low time	4		ns
$t_{CLK}$	DCLK period	10		ns
$f_{MAX}$	DCLK frequency		100	MHz
$t_{CD2UM}$	CONF_DONE high to user mode (3)	18	40	$\mu$ s
$t_{CD2CU}$	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period		
$t_{CD2UMC}$	CONF_DONE high to user mode with CLKUSR option on	$t_{CD2CU} + (299 \times \text{CLKUSR period})$		

**Notes to Table 13–7:**

- (1) The POR delay minimum of 100 ms only applies for non “A” devices.
- (2) This value is applicable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
- (3) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting the device.



Device configuration options and how to create configuration files are discussed further in the *Software Settings* section in Volume 2 of the *Configuration Handbook*.

## PS Configuration Using a Microprocessor

In the PS configuration scheme, a microprocessor can control the transfer of configuration data from a storage device, such as flash memory, to the target Cyclone II device.



All information in the “Single Device PS Configuration Using a MAX II Device as an External Host” on page 13–22 section is also applicable when using a microprocessor as an external host. Refer to that section for all configuration information.

The MicroBlaster™ software driver allows you to configure Altera FPGAs, including Cyclone II devices, through the ByteBlaster II or ByteBlasterMV cable in PS mode. The MicroBlaster software driver supports a RBF programming input file and is targeted for embedded PS configuration. The source code is developed for the Windows NT operating system, although you can customize it to run on other operating systems.



Since the Cyclone II device can decompress the compressed configuration data on-the-fly during PS configuration, the MicroBlaster software can accept a compressed RBF file as its input file.



For more information on the MicroBlaster software driver, see the *Configuring the MicroBlaster Passive Serial Software Driver White Paper* and source files on the Altera web site at [www.altera.com](http://www.altera.com).

If you turn on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software, the Cyclone II devices does not enter user mode after the MicroBlaster has transmitted all the configuration data in the RBF file. You need to supply enough initialization clock cycles to CLKUSR pin to enter user mode.

### Single Device PS Configuration Using a Configuration Device

You can use an Altera configuration device (for example, an EPC2, EPC1, or enhanced configuration device) to configure Cyclone II devices using a serial configuration bitstream. Configuration data is stored in the configuration device. Figure 13–13 shows the configuration interface connections between the Cyclone II device and a configuration device.

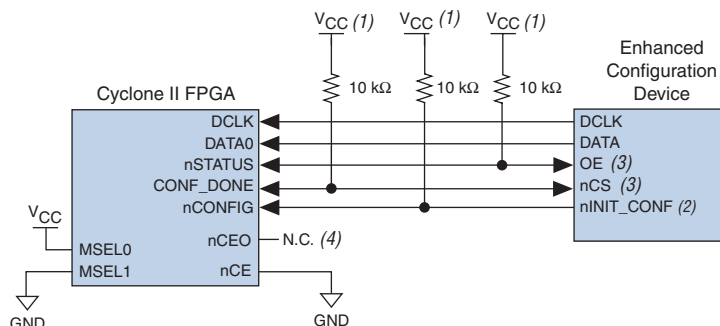


The figures in this chapter only show the configuration-related pins and the configuration pin connections between the configuration device and the FPGA.



For more information on enhanced configuration devices and flash interface pins (e.g., PGM[2 . . 0], EXCLK, PORSEL, A[20 . . 0], and DQ[15 . . 0]), see the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet*.

**Figure 13–13. Single Device PS Configuration Using an Enhanced Configuration Device**



**Notes to Figure 13–13:**

- (1) The pull-up resistor should be connected to the same supply voltage as the configuration device. This pull-up resistor is 10 kΩ
- (2) The `nINIT_CONF` pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the `nINIT_CONF` to `nCONFIG` line. The `nINIT_CONF` pin does not need to be connected if its functionality is not used. If `nINIT_CONF` is not used, `nCONFIG` must be pulled to  $V_{CC}$  either directly or through a resistor (if reconfiguration is required, a resistor is necessary).
- (3) The enhanced configuration devices' `OE` and `nCS` pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.
- (4) The `nCEO` pin can be left unconnected or used as a user I/O pin when it does not feed other device's `nCE` pin.



The value of the internal pull-up resistors on the enhanced configuration devices and EPC2 devices can be found in the *Enhanced Configuration Devices (EPC4, EPC8, & EPC16) Data Sheet* or the *Configuration Devices for SRAM-based LUT Devices Data Sheet*.

When using enhanced configuration devices or EPC2 devices, you can connect the Cyclone II `nCONFIG` pin to the configuration device `nINIT_CONF` pin, which allows the `INIT_CONF` JTAG instruction to initiate FPGA configuration. You do not need to connect the `nINIT_CONF` pin if you are not using it. If `nINIT_CONF` is not used or not available (e.g., on EPC1 devices), pull the `nCONFIG` signal to  $V_{CC}$  either directly or through a resistor (if reconfiguration is required, a resistor is necessary). An internal pull-up resistor on the `nINIT_CONF` pin is always active in enhanced configuration devices and EPC2 devices. Therefore, you do not need an external pull-up if `nCONFIG` is connected to `nINIT_CONF`.

Upon power-up, the Cyclone II device goes through a POR. During POR, the device reset, holds `nSTATUS` and `CONF_DONE` low, and tri-states all user I/O pins. After POR, which typically lasts 100 ms, the Cyclone II FPGA releases `nSTATUS` and enters configuration mode when this signal is pulled high by the external 10-k $\Omega$  resistor. Once the FPGA successfully exits POR, all user I/O pins continue to be tri-stated. Cyclone II devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The configuration device also goes through a POR delay to allow the power supply to stabilize. The maximum POR time for EPC2 or EPC1 devices is 200 ms. The POR time for enhanced configuration devices can be set to 100 ms or 2 ms, depending on the enhanced configuration device's `PORSEL` pin setting. If the `PORSEL` pin is connected to ground, the POR delay is 100 ms. If the `PORSEL` pin is connected to  $V_{CC}$ , the POR delay is 2 ms. You must power the Cyclone II device before or during the enhanced configuration device POR time. During POR, the configuration device transitions its `OE` pin low. This low signal delays configuration because the `OE` pin is connected to the target device's `nSTATUS` pin. When the target and configuration devices complete POR, they both release the `nSTATUS` to `OE` line, which is then pulled high by a pull-up resistor.

When the power supplies have reached the appropriate operating voltages, the target FPGA senses the low-to-high transition on `nCONFIG` and initiates the configuration cycle. The configuration cycle consists of three stages: reset, configuration, and initialization.



The Cyclone II device does not have a `PORSEL` pin.

### *Reset Stage*

While `nCONFIG` or `nSTATUS` is low, the device is in reset. You can delay configuration by holding the `nCONFIG` or `nSTATUS` pin low.



$V_{CCINT}$  and  $V_{CCIO}$  of the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When the `nCONFIG` signal goes high, the device comes out of reset and releases the `nSTATUS` pin, which is pulled high by a pull-up resistor. Enhanced configuration and EPC2 devices have an optional internal pull-up resistor on the `OE` pin. You can turn on this option in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up resistor is not used, you need to connect an external 10-k $\Omega$  pull-up resistor to the `OE` and `nSTATUS` line. Once `nSTATUS` is released, the FPGA is ready to receive configuration data and the configuration stage begins.

### *Configuration Stage*

When the `nSTATUS` pin transitions high, the configuration device's `OE` pin also transitions high and the configuration device clocks data out serially to the FPGA using its internal oscillator. The Cyclone II device receives configuration data on its `DATA0` pin and the clock is received on the `DCLK` pin. Data is latched into the FPGA on the rising edge of `DCLK`.

After the FPGA has received all configuration data successfully, it releases the open-drain `CONF_DONE` pin, which is pulled high by a pull-up resistor. Since the Cyclone II device's `CONF_DONE` pin is tied to the configuration device's `nCS` pin, the configuration device is disabled when `CONF_DONE` goes high. Enhanced configuration and EPC2 devices have an optional internal pull-up resistor on the `nCS` pin. You can turn this option on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If you do not use this internal pull-up resistor, you need to connect an external 10-k $\Omega$  pull-up resistor to the `nCS` and `CONF_DONE` line. A low-to-high transition on `CONF_DONE` indicates configuration is complete, and the device can begin initialization.

### *Initialization Stage*

In Cyclone II devices, the default initialization clock source is the Cyclone II internal oscillator (typically 10 MHz). Cyclone II devices can also use the optional `CLKUSR` pin. If your design uses the internal oscillator, the Cyclone II device supplies itself with enough clock cycles for proper initialization. The advantage of using the internal oscillator is you do not need to use another device or source to send additional clock cycles to the `CLKUSR` pin during the initialization stage. Additionally, you can use of the `CLKUSR` pin as a user I/O pin, which means you have an additional user I/O pin.

If you want to delay the initialization of the device, you can use the `CLKUSR` pin. Using the `CLKUSR` pin allows you to control when the Cyclone II device enters user mode. You can delay the Cyclone II devices from entering user mode for an indefinite amount of time. You can turn on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on `CLKUSR` does not affect the configuration process. After all configuration data is accepted and `CONF_DONE` goes high, Cyclone II devices require 299 clock cycles to properly initialize and support a `CLKUSR`  $f_{MAX}$  of 100 MHz.

An optional `INIT_DONE` pin is available, which signals the end of initialization and the start of user mode with a low-to-high transition. The **Enable INIT\_DONE output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If you use the `INIT_DONE` pin, an external 10-k $\Omega$  pull-up resistor pulls it high when

$\overline{nCONFIG}$  is low and during the beginning of configuration. Once the optional bit to enable  $INIT\_DONE$  is programmed into the device (during the first frame of configuration data), the  $INIT\_DONE$  pin goes low. When initialization is complete, the  $INIT\_DONE$  pin is released and pulled high. This low-to-high transition signals that the FPGA has entered user mode. If you do not use the  $INIT\_DONE$  pin, the initialization period is complete after the  $CONF\_DONE$  signal transitions high and 299 clock cycles are sent to the  $CLKUSR$  pin or after the time  $t_{CF2UM}$  (see Table 13–7) if the Cyclone II device uses the internal oscillator.

After successful configuration, if you intend to synchronize the initialization of multiple devices that are not in the same configuration chain, your system must not pull the  $CONF\_DONE$  signal low to delay initialization. Instead, use the optional  $CLKUSR$  pin to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain initialize together if their  $CONF\_DONE$  pins are tied together.



If the optional  $CLKUSR$  pin is being used and  $\overline{nCONFIG}$  is pulled low to restart configuration during device initialization, you need to ensure that  $CLKUSR$  continues toggling during the time  $\overline{nSTATUS}$  is low (maximum of 40  $\mu s$ ).

### User Mode

When initialization is complete, the FPGA enters user mode. In user mode, the user I/O pins do not have weak pull-up resistors and function as assigned in your design. Enhanced configuration devices and EPC2 devices drive  $\overline{DCLK}$  low and  $DATA0$  high (EPC1 devices drive the  $\overline{DCLK}$  pin low and tri-state the  $DATA$  pin) at the end of configuration.

When the FPGA is in user mode, pull the  $\overline{nCONFIG}$  pin low to begin reconfiguration. The  $\overline{nCONFIG}$  pin should be low for at least 2  $\mu s$ . When  $\overline{nCONFIG}$  transitions low, the Cyclone II device also pulls the  $\overline{nSTATUS}$  and  $CONF\_DONE$  pins low and all I/O pins are tri-stated. Because  $CONF\_DONE$  transitions low, this activates the configuration device since it will see its  $\overline{nCS}$  pin transition low. Once  $\overline{nCONFIG}$  returns to a logic high level and  $\overline{nSTATUS}$  is released by the FPGA, reconfiguration begins.

### Error During Configuration

If an error occurs during configuration, the Cyclone II drives its  $\overline{nSTATUS}$  pin low, resetting itself internally. Since the  $\overline{nSTATUS}$  pin is tied to OE, the configuration device is also reset. If you turn on the **Auto-restart configuration after error** option in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box, the FPGA automatically initiates reconfiguration if an error occurs. The Cyclone II

device releases its `nSTATUS` pin after a reset time-out period (maximum of 40  $\mu$ s). When the `nSTATUS` pin is released and pulled high by a pull-up resistor, the configuration device reconfigures the chain. If this option is turned off, the external system must monitor `nSTATUS` for errors and then pulse `nCONFIG` low for at least 2  $\mu$ s to restart configuration. The external system can pulse the `nCONFIG` pin if the pin is under system control rather than tied to  $V_{CC}$ .

Additionally, if the configuration device sends all of its data and then detects that the `CONF_DONE` pin has not transitioned high, it recognizes that the FPGA has not configured successfully. Enhanced configuration devices wait for 64 `DCLK` cycles after the last configuration bit was sent for the `CONF_DONE` pin to transition high. EPC2 devices wait for 16 `DCLK` cycles. After that, the configuration device pulls its OE pin low, which in turn drives the target device's `nSTATUS` pin low. If you turn on the **Auto-restart configuration after error** option in the Quartus II software, the target device resets and then releases its `nSTATUS` pin after a reset time-out period (maximum of 40  $\mu$ s). When `nSTATUS` transitions high again, the configuration device reconfigures the FPGA.

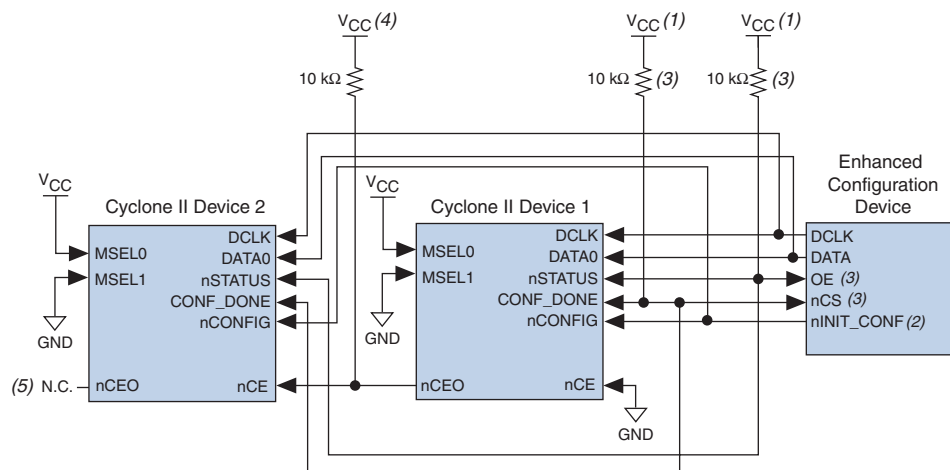


For more information on configuration issues, see the *Debugging Configuration Problems* chapter of the *Configuration Handbook* and the FPGA Configuration Troubleshooter on the Altera web site ([www.altera.com](http://www.altera.com)).

### Multiple Device PS Configuration Using a Configuration Device

You can use Altera enhanced configuration devices (EPC16, EPC8, and EPC4 devices) or EPC2 and EPC1 configuration devices to configure multiple Cyclone II devices in a PS configuration chain.

Figure 13–14 shows how to configure multiple devices with an enhanced configuration device. This circuit is similar to the configuration device circuit for a single device, except Cyclone II devices are cascaded for multiple device configuration.

**Figure 13–14. Multiple Device PS Configuration Using an Enhanced Configuration Device****Notes to Figure 13–14:**

- (1) The pull-up resistor should be connected to the same supply voltage as the configuration device.
- (2) The `nINIT_CONF` pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the `nINIT_CONF` to `nCONFIG` line. The `nINIT_CONF` pin does not need to be connected if its functionality is not used. If `nINIT_CONF` is not used, `nCONFIG` must be pulled to `VCC` either directly or through a resistor (if reconfiguration is required, a resistor is necessary).
- (3) The enhanced configuration devices' `OE` and `nCS` pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.
- (4) Connect the pull-up resistor to the `VCCIO` supply voltage of I/O bank that the `nCEO` pin resides in.
- (5) The `nCEO` pin can be left unconnected or used as a user I/O pin when it does not feed other device's `nCE` pin.



You cannot cascade enhanced configuration devices (EPC16, EPC8, and EPC4 devices).

When configuring multiple devices, you must generate the configuration device's POF from each project's SOF. You can combine multiple SOFs using the **Convert Programming Files** window in the Quartus II software.



For more information on how to create configuration files for multiple device configuration chains, see the *Software Settings* section in Volume 2 of the *Configuration Handbook*.

When configuring multiple devices with the PS scheme, connect the first Cyclone II device's `nCE` pin to GND and connect its `nCEO` pin to the `nCE` pin of the Cyclone II device in the chain. Use an external 10-kΩ pull-up resistor to pull the Cyclone II device's `nCEO` pin to the `VCCIO` level when



it feeds the next device's `nCE` pin. After the first device in the chain completes configuration, its `nCEO` pin transitions low to activate the second device's `nCE` pin, which prompts the second device to begin configuration. You can leave the `nCEO` pin of the last device unconnected or use it as a user I/O pin after configuration. The `nCEO` pin is a dual-purpose pin in Cyclone II devices.



The Quartus II software sets the Cyclone II device `nCEO` pin as an output pin driving to ground by default. If the device is in a chain, and the `nCEO` pin is connected to the next device's `nCE` pin, you must make sure that the `nCEO` pin is not used as a user I/O pin after configuration. This software setting is in the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box in Quartus II software.

Connect all other configuration pins (`nCONFIG`, `nSTATUS`, `DCLK`, `DATA0`, and `CONF_DONE`) to every Cyclone II device in the chain. The configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Buffer the `DCLK` and `DATA` lines for every fourth device.

When configuring multiple devices, configuration does not begin until all devices release their `OE` or `nSTATUS` pins. Similarly, since all device `CONF_DONE` pins are tied together, all devices initialize and enter user mode at the same time.

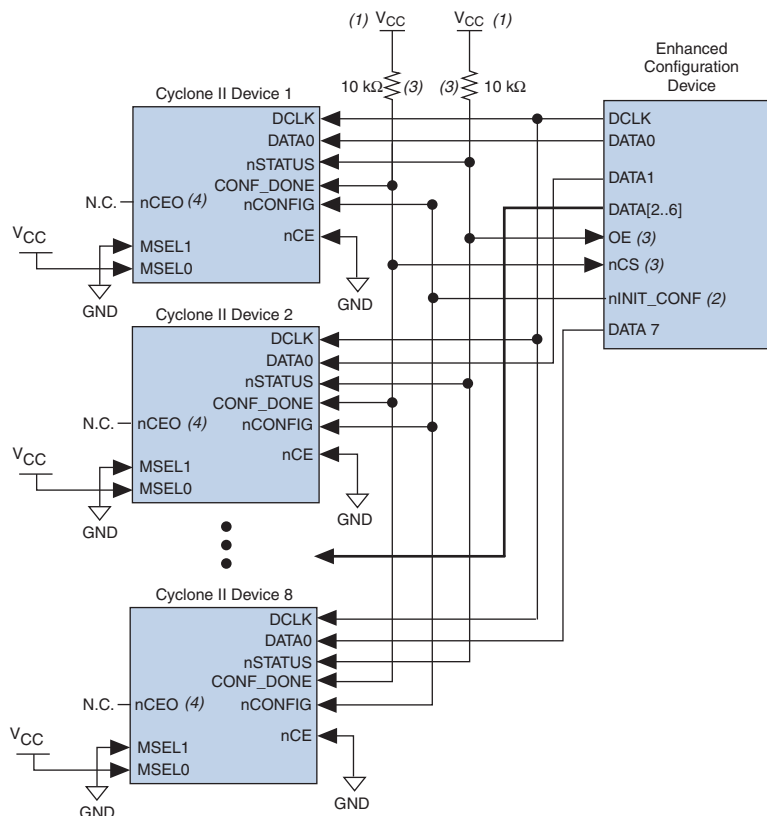
You should not pull `CONF_DONE` low to delay initialization. Instead, use the Quartus II software's **User-Supplied Start-Up Clock** option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain initialize together since their `CONF_DONE` pins are tied together.

Since all `nSTATUS` and `CONF_DONE` pins are connected, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if there is an error when configuring the first Cyclone II device, it resets the chain by pulling its `nSTATUS` pin low. This low signal drives the `OE` pin low on the enhanced configuration device and drives `nSTATUS` low on all FPGAs, which causes them to enter a reset state.

If the **Auto-restart configuration after error** option is turned on, the devices automatically initiate reconfiguration if an error occurs. The FPGAs release their `nSTATUS` pins after a reset time-out period (40  $\mu$ s maximum). When all the `nSTATUS` pins are released and pulled high, the configuration device reconfigures the chain. If the **Auto-restart configuration after error** option is turned off, a microprocessor or controller must monitor the `nSTATUS` pin for errors and then pulse

`nCONFIG` low for at least 2  $\mu$ s to restart configuration. The microprocessor or controller can only transition the `nCONFIG` pin low if the pin is under system control and not tied to  $V_{CC}$ .

The enhanced configuration devices support parallel configuration of up to eight devices. The  $n$ -bit ( $n = 1, 2, 4, \text{ or } 8$ ) PS configuration mode allows enhanced configuration devices to concurrently configure a chain of FPGAs. These devices do not have to be the same device family or density; they can be any combination of Altera FPGAs with different designs. An individual enhanced configuration device `DATA` pin is available for each targeted FPGA. Each `DATA` line can also feed a chain of FPGAs. [Figure 13–15](#) shows how to concurrently configure multiple devices using an enhanced configuration device.

**Figure 13–15. Concurrent PS Configuration of Multiple Devices Using an Enhanced Configuration Device****Notes to Table 13–15:**

- (1) The pull-up resistor should be connected to the same supply voltage as the configuration device.
- (2) The `nINIT_CONF` pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the `nINIT_CONF` to `nCONFIG` line. The `nINIT_CONF` pin does not need to be connected if its functionality is not used. If `nINIT_CONF` is not used, `nCONFIG` must be pulled to `VCC` either directly or through a resistor (if reconfiguration is required, a resistor is necessary).
- (3) The enhanced configuration devices' `OE` and `nCS` pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.
- (4) The `nCEO` pin can be left unconnected or used as a user I/O pin when it does not feed other device's `nCE` pin.

The Quartus II software only allows you to set *n* to 1, 2, 4, or 8. However, you can use these modes to configure any number of devices from 1 to 8. For example, if you configure three FPGAs, you would use the 4-bit PS mode. For the `DATA0`, `DATA1`, and `DATA2` lines, the corresponding SOF data is transmitted from the configuration device to the FPGA. For

DATA3, you can leave the corresponding bit 3 line blank in the Quartus II software. On the printed circuit board (PCB), leave the DATA3 line from the enhanced configuration device unconnected. Use the Quartus II **Convert Programming Files** window (Tools menu) setup for this scheme.

You can also connect two FPGAs to one of the configuration device's DATA pins while the other DATA pins drive one device each. For example, you could use the 2-bit PS mode to drive two FPGAs with DATA bit 0 (two EP2C5 devices) and the third device (an EP2C8 device) with DATA bit 1. In this example, the memory space required for DATA bit 0 is the sum of the SOF file size for the two EP2C5 devices.

$$1,223,980 \text{ bits} + 1,223,980 \text{ bits} = 2,447,960 \text{ bits}$$

The memory space required for DATA bit 1 is the SOF file size for on EP2C8 device (1,983,792 bits). Since the memory space required for DATA bit 0 is larger than the memory space required for DATA bit 1, the size of the POF file is  $2 \times 2,447,960 = 4,895,920$ .



For more information on using  $n$ -bit PS modes with enhanced configuration devices, see the *Using Altera Enhanced Configuration Devices* in the *Configuration Handbook*.

When configuring SRAM-based devices using  $n$ -bit PS modes, use [Table 13–8](#) to select the appropriate configuration mode for the fastest configuration times.

**Table 13–8. Recommended Configuration Using  $n$ -Bit PS Modes**

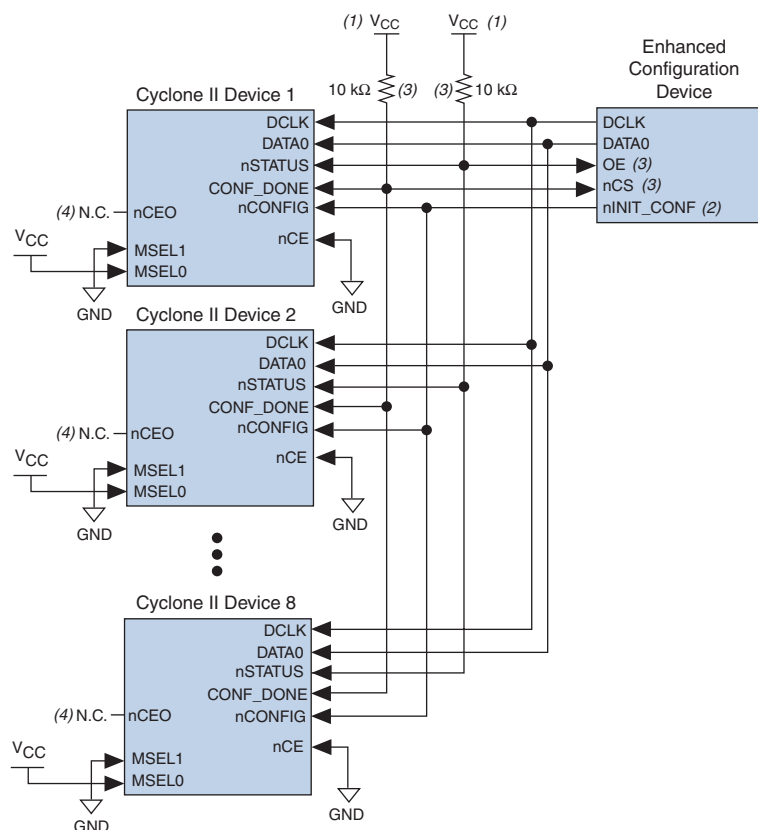
Number of Devices (1)	Recommended Configuration Mode
1	1-bit PS
2	2-bit PS
3	4-bit PS
4	4-bit PS
5	8-bit PS
6	8-bit PS
7	8-bit PS
8	8-bit PS

**Note to Table 13–8:**

- (1) Assume that each DATA line is only configuring one device, not a daisy chain of devices.

If your design has multiple Cyclone II devices of the same density and package that contain the same configuration data, connect the `nCE` inputs to GND and leave the `nCEO` pins floating. You can also use the `nCEO` pin as a user I/O pin. Connect the configuration device `nCONFIG`, `nSTATUS`, `DCLK`, `DATA0`, and `CONF_DONE` pins to each Cyclone II device in the chain. The configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Make sure that the `DCLK` and `DATA` lines are buffered for every fourth device. All devices start and complete configuration at the same time. [Figure 13–16](#) shows multiple device PS configuration when the Cyclone II devices are receiving the same configuration data.

**Figure 13–16. Multiple Device PS Configuration Using an Enhanced Configuration Device When FPGAs Receive the Same Data**



**Notes to Figure 13–16:**

- (1) The pull-up resistor should be connected to the same supply voltage as the configuration device.
- (2) The nINIT\_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT\_CONF to nCONFIG line. The nINIT\_CONF pin does not need to be connected if its functionality is not used. If nINIT\_CONF is not used, nCONFIG must be pulled to V<sub>CC</sub> either directly or through a resistor (if reconfiguration is required, a resistor is necessary).
- (3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.
- (4) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed other device's nCE pin.

You can cascade several EPC2 or EPC1 devices to configure multiple Cyclone II devices. The first configuration device in the chain is the master configuration device, and the subsequent devices are the slave devices. The master configuration device sends DCLK to the Cyclone II

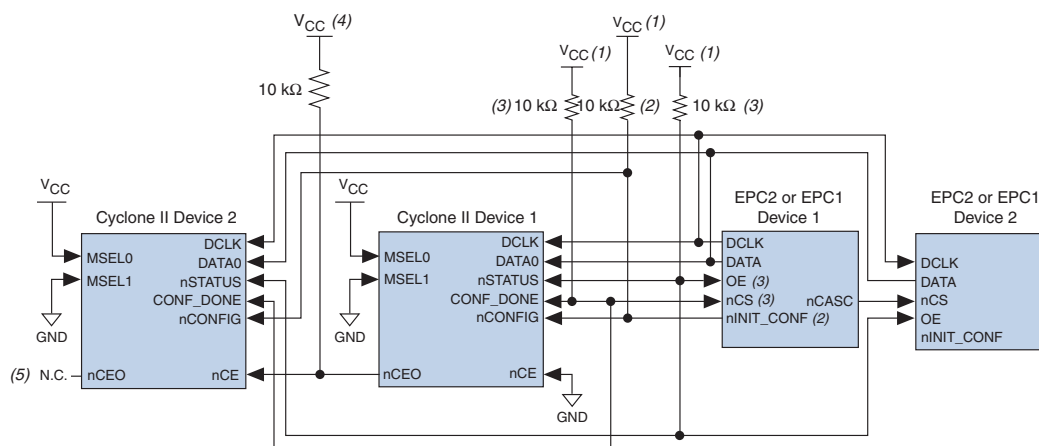
devices and to the slave configuration devices. Connect the first configuration device's `nCS` pin to all the Cyclone II device's `CONF_DONE` pins, and connect the `nCASC` pin to the `nCS` pin of the next configuration device in the chain. Leave the `nCASC` pin of the last configuration device floating. When the master configuration device sends all the data to the Cyclone II device, the configuration device transitions the `nCASC` pin low, which drives `nCS` on the next configuration device. Because a configuration device requires less than one clock cycle to activate a subsequent configuration device, the data stream is uninterrupted.



Enhanced configuration devices (EPC16, EPC8, and EPC4 devices) cannot be cascaded.

Since all `nSTATUS` and `CONF_DONE` pins are connected, if any device detects an error, the master configuration device stops configuration for the entire chain and the entire chain must be reconfigured. For example, if the master configuration device does not detect the Cyclone II device's `CONF_DONE` pin transitioning high at the end of configuration, it resets the entire chain by transitioning its `OE` pin low. This low signal drives the `OE` pin low on the slave configuration device(s) and drives `nSTATUS` low on all Cyclone II devices, causing them to enter a reset state. This behavior is similar to the FPGA detecting an error in the configuration data.

Figure 13–17 shows how to configure multiple devices using cascaded EPC2 or EPC1 devices.

**Figure 13–17. Multiple Device PS Configuration Using Cascaded EPC2 or EPC1 Devices****Notes to Figure 13–17:**

- (1) The pull-up resistor should be connected to the same supply voltage as the configuration device.
- (2) The `nINIT_CONF` pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the `nINIT_CONF` to `nCONFIG` line. The `nINIT_CONF` pin does not need to be connected if its functionality is not used. If `nINIT_CONF` is not used or not available (e.g., on EPC1 devices), `nCONFIG` must be pulled to `VCC` either directly or through a resistor (if reconfiguration is required, a resistor is necessary).
- (3) The enhanced configuration devices' and EPC2 devices' `OE` and `nCS` pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device option** when generating programming files.
- (4) Use an external 10-kΩ pull-up resistor to pull the `nCEO` pin high to the I/O bank `VCCIO` level to help the internal weak pull-up when it feeds next device's `nCE` pin.
- (5) The `nCEO` pin can be left unconnected or used as a user I/O pin when it does not feed other device's `nCE` pin.

When using enhanced configuration devices or EPC2 devices, you can connect the Cyclone II device's `nCONFIG` pin to the configuration device's `nINIT_CONF` pin, which allows the `INIT_CONF` JTAG instruction to initiate FPGA configuration. You do not need to connect the `nINIT_CONF` pin if it is not used. If the `nINIT_CONF` pin is not used or not available (for example, on EPC1 devices), pull the `nCONFIG` pin to `VCC` levels either directly or through a resistor (if reconfiguration is required, a resistor is necessary). An internal pull-up resistor on the `nINIT_CONF` pin is always active in the enhanced configuration devices and the EPC2 devices. Therefore, do not use an external pull-up resistor if you connect the `nCONFIG` pin to `nINIT_CONF`. If you use multiple EPC2 devices to configure a Cyclone II device(s), only connect the first EPC2 device's `nINIT_CONF` pin to the device's `nCONFIG` pin.



You can use a single configuration chain to configure Cyclone II devices with other Altera devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, connect all the Cyclone II device `CONF_DONE` pins and connect all Cyclone II device `nSTATUS` pins together.

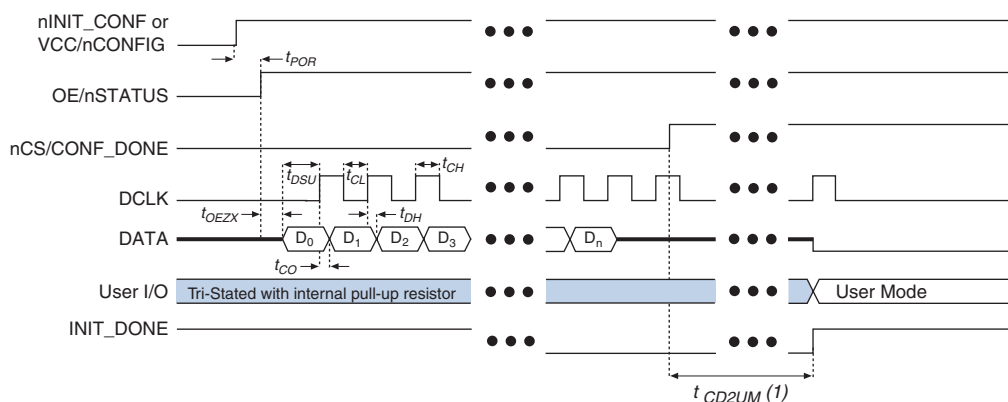


For more information on configuring multiple Altera devices in the same configuration chain, see the *Configuring Mixed Altera FPGA Chains* chapter in the *Configuration Handbook*.

During PS configuration, the design must meet the setup and hold timing parameters and maximum `DCLK` frequency. The enhanced configuration and EPC2 devices are designed to meet these interface timing specifications.

Figure 13–18 shows the timing waveform for the PS configuration scheme using a configuration device.

**Figure 13–18. Cyclone II PS Configuration Using a Configuration Device Timing Waveform**



**Note to Figure 13–18:**

- (1) Cyclone II devices enter user mode 299 clock cycles after `CONF_DONE` goes high. The initialization clock can come from the Cyclone II internal oscillator or the `CLKUSR` pin.



For timing information, refer to the *Enhanced Configuration Devices (EPC4, EPC8, and EPC16) Data Sheet* or the *Configuration Devices for SRAM-based LUT Devices Data Sheet* in the *Configuration Handbook*.



For more information on device configuration options and how to create configuration files, see the *Software Settings* section in Volume 2 of the *Configuration Handbook*.

## PS Configuration Using a Download Cable

In PS configuration, an intelligent host (e.g., a PC) can use a download cable to transfer data from a storage device to the Cyclone II device. You can use the Altera USB-Blaster universal serial bus (USB) port download cable, MasterBlaster™ serial/USB communications cable, ByteBlaster II parallel port download cable, or the ByteBlasterMV™ parallel port as a download cable.

Upon power up, the Cyclone II device goes through POR, which lasts approximately 100 ms for non “A” devices. During POR, the device resets, holds `nSTATUS` low, and tri-states all user I/O pins. Once the FPGA successfully exits POR, the `nSTATUS` pin is released and all user I/O pins continue to be tri-stated.



The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *Cyclone II Device Handbook*.

The configuration cycle consists of three stages: reset, configuration, and initialization. While the `nCONFIG` or `nSTATUS` pins are low, the device is in reset. To initiate configuration in this scheme, the download cable generates a low-to-high transition on the `nCONFIG` pin.



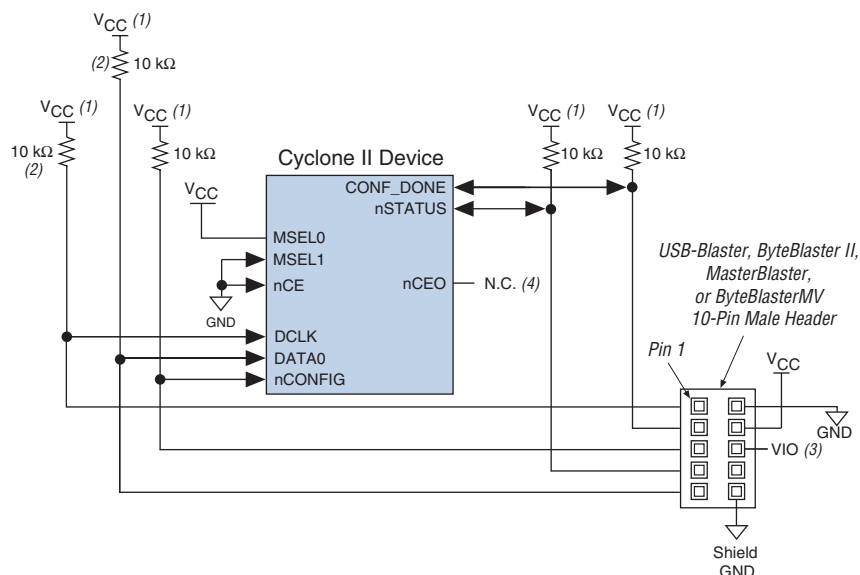
Make sure  $V_{CCINT}$  and  $V_{CCIO}$  for the banks where the configuration and JTAG pins reside are powered to the appropriate voltage levels in order to begin the configuration process.

When `nCONFIG` transitions high, the Cyclone II device comes out of reset and begins configuration. The Cyclone II device releases the open-drain `nSTATUS` pin, which is then pulled high by an external 10-k $\Omega$  pull-up resistor. Once `nSTATUS` transitions high, the Cyclone II device is ready to receive configuration data. The programming hardware or download cable then transmits the configuration data one bit at a time to the device's `DATA0` pin. The configuration data is clocked into the target device until `CONF_DONE` goes high. The `CONF_DONE` pin must have an external 10-k $\Omega$  pull-up resistor in order for the device to initialize.

When using a download cable, you cannot use the **Auto-restart configuration after error** option. You must manually restart configuration in the Quartus II software when an error occurs. Additionally, you cannot use the **Enable user-supplied start-up clock (CLKUSR)** option when programming the FPGA using the Quartus II programmer and download cable. This option is disabled in the SOF. Therefore, if you turn on the `CLKUSR` option, you do not need to provide a clock on `CLKUSR` when you are configuring the FPGA with the

Quartus II programmer and a download cable. Figure 13–19 shows the PS configuration for Cyclone II devices using a USB-Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV cable.

**Figure 13–19. PS Configuration Using a USB-Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV Cable**



**Notes to Figure 13–19:**

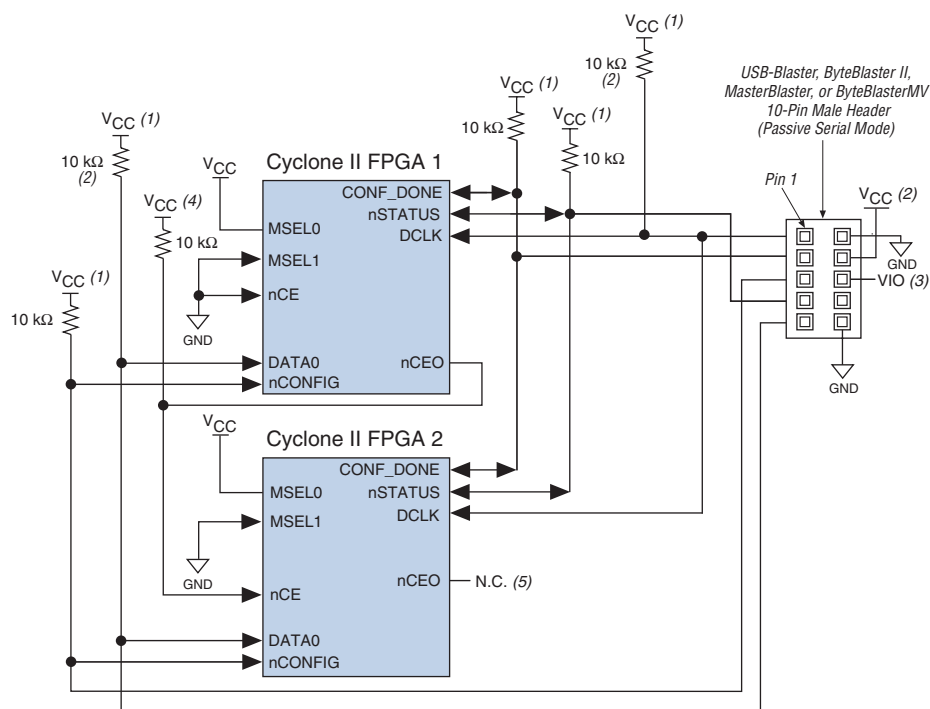
- (1) The pull-up resistor should be connected to the same supply voltage as the USB-Blaster, MasterBlaster (VIO pin), ByteBlaster II, or ByteBlasterMV cable.
- (2) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This is to ensure that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.
- (3) Pin 6 of the header is a VIO reference voltage for the MasterBlaster output driver. VIO should match the device's VCCIO. Refer to the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value. In the ByteBlasterMV, this pin is a no connect. In the USB-Blaster and ByteBlaster II, this pin is connected to nCE when it is used for AS programming, otherwise it is a no connect.
- (4) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed other device's nCE pin.

You can use a download cable to configure multiple Cyclone II devices by connecting each device's nCEO pin to the subsequent device's nCE pin. Connect the first Cyclone II device's nCE pin to GND and connect its nCEO pin to the nCE pin of the next device in the chain. Use an external 10-kΩ pull-up resistor to pull the nCEO pin high to VCCIO when it feeds next device's nCE pin. Connect all other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF\_DONE) on every device in the chain together. Because all CONF\_DONE pins are connected, all devices in the chain initialize and enter user mode at the same time.

In addition, because the `nSTATUS` pins are connected, all the Cyclone II devices in the chain stop configuration if any device detects an error. If this happens, you must manually restart configuration in the Quartus II software.

Figure 13–20 shows how to configure multiple Cyclone II devices with a download cable.

**Figure 13–20. Multiple Device PS Configuration Using a USB-Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV Cable**



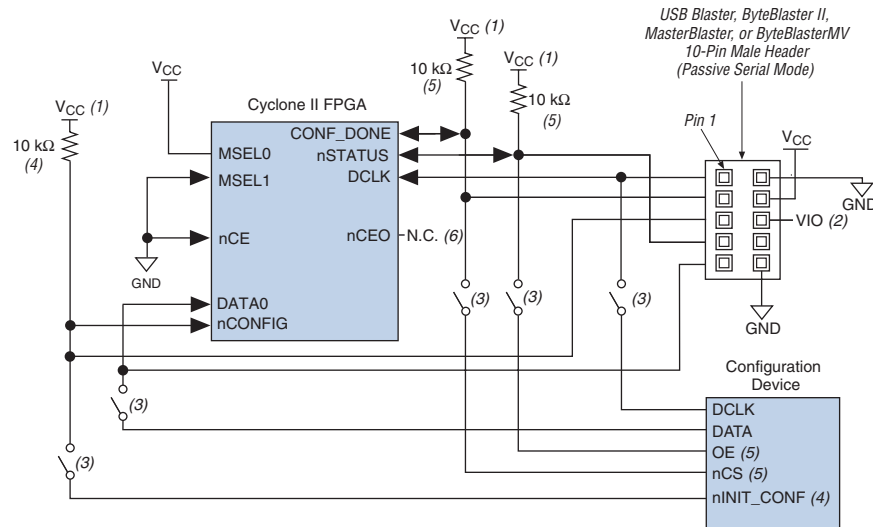
**Notes to Figure 13–20:**

- (1) The pull-up resistor should be connected to the same supply voltage as the USB-Blaster, MasterBlaster (V<sub>IO</sub> pin), ByteBlaster II, or ByteBlasterMV cable.
- (2) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This is to ensure that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.
- (3) Pin 6 of the header is a V<sub>IO</sub> reference voltage for the MasterBlaster output driver. V<sub>IO</sub> should match the device's V<sub>CCIO</sub>. Refer to the MasterBlaster Serial/USB Communications Cable Data Sheet for this value. In the ByteBlasterMV, this pin is a no connect. In the USB-Blaster and ByteBlaster II, this pin is connected to nCE when it is used for AS programming, otherwise it is a no connect.
- (4) Connect the pull-up resistor to the V<sub>CCIO</sub> supply voltage of I/O bank that the nCEO pin resides in.
- (5) The nCEO pin of the last device in chain can be left unconnected or used as a user I/O pin.

If you are using a download cable to configure Cyclone II devices on a PCB that also has configuration devices, you should electrically isolate the configuration devices from the target Cyclone II devices and cable. One way to isolate the configuration device is to add logic, such as a multiplexer, that can select between the configuration device and the cable. The multiplexer should allow bidirectional transfers on the nSTATUS and CONF\_DONE signals. Additionally, you can add switches to

the five common signals (nCONFIG, nSTATUS, DCLK, DATA0, and CONF\_DONE) between the cable and the configuration device. You can also remove the configuration device from the board when configuring the FPGA with the cable. Figure 13–21 shows a combination of a configuration device and a download cable to configure an FPGA.

**Figure 13–21. PS Configuration with a Download Cable & Configuration Device Circuit**



**Notes to Figure 13–21:**

- (1) The pull-up resistor should be connected to the same supply voltage as the configuration device.
- (2) Pin 6 of the header is a  $V_{IO}$  reference voltage for the MasterBlaster output driver.  $V_{IO}$  should match the device's  $V_{CCIO}$ . Refer to the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value. In the ByteBlasterMV, this pin is a no connect. In the USB-Blaster and ByteBlaster II, this pin is connected to nCE when it is used for AS programming, otherwise it is a no connect.
- (3) You should not attempt configuration with a download cable while a configuration device is connected to a Cyclone II device. Instead, you should either remove the configuration device from its socket when using the download cable or place a switch on the five common signals between the download cable and the configuration device.
- (4) The nINIT\_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active. This means an external pull-up resistor should not be used on the nINIT\_CONF to nCONFIG line. The nINIT\_CONF pin does not need to be connected if its functionality is not used. If nINIT\_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to  $V_{CC}$  either directly or through a resistor (if reconfiguration is required, a resistor is necessary).
- (5) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.
- (6) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed other device's nCE pin.



For more information on how to use the USB-Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV cables, refer to the following documents:

- *USB-Blaster USB Port Download Cable Data Sheet*
- *MasterBlaster Serial/USB Communications Cable Data Sheet*
- *ByteBlaster II Parallel Port Download Cable Data Sheet*
- *ByteBlasterMV Parallel Port Download Cable Data Sheet*

## JTAG Configuration

The Joint Test Action Group (JTAG) has developed a specification for boundary-scan testing. This boundary-scan test (BST) architecture allows you to test components on PCBs with tight lead spacing. The BST architecture can test pin connections without using physical test probes and capture functional data while a device is operating normally. The JTAG circuitry can also be used to shift configuration data into the device. The Quartus II software automatically generates SOF files that can be used for JTAG configuration with a download cable in the Quartus II programmer.



For more information on JTAG boundary-scan testing, see the following documents:

- *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Cyclone II Devices* chapter in Volume 2 of the *Cyclone II Device Handbook*
- *Jam Programming & Testing Language Specification*

Cyclone II devices are designed such that JTAG instructions have precedence over any device configuration modes. This means that JTAG configuration can take place without waiting for other configuration modes to complete. For example, if you attempt JTAG configuration of Cyclone II devices during PS configuration, PS configuration terminates and JTAG configuration begins. If the Cyclone II MSEL pins are set to AS or fast AS mode, the Cyclone II device does not output a DCLK signal when JTAG configuration takes place.



You cannot use the Cyclone II decompression feature if you are configuring your Cyclone II device when using JTAG-based configuration.

A device operating in JTAG mode uses the TDI, TDO, TMS, and TCK pins. The TCK pin has a weak internal pull-down resistor while the other JTAG input pins, TDI and TMS, have weak internal pull-up resistors. All user I/O pins are tri-stated during JTAG configuration. Table 13–9 explains each JTAG pin's function.

**Table 13–9. Dedicated JTAG Pins**

Pin Name	Pin Type	Description
TDI	Test data input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V <sub>CC</sub> .
TDO	Test data output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected.
TMS	Test mode select	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V <sub>CC</sub> .
TCK	Test clock input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND.



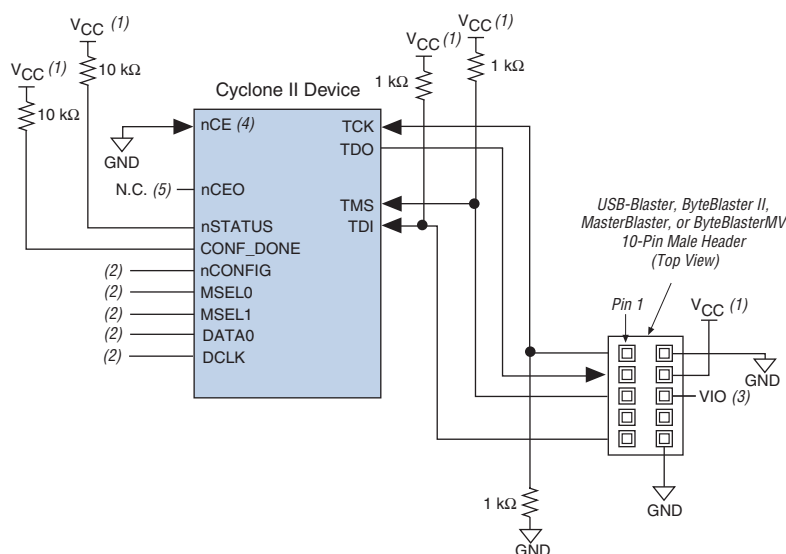
The TDO output is powered by the V<sub>CCIO</sub> power supply. If V<sub>CCIO</sub> is tied to 3.3-V, both the I/O pins and the JTAG TDO port drive at 3.3-V levels.



## Single Device JTAG Configuration

During JTAG configuration, you can use the USB-Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV download cable to download data to the device. Configuring Cyclone II devices through a cable is similar to programming devices in system. Figure 13–22 shows JTAG configuration of a single Cyclone II device using a download cable.

**Figure 13–22. JTAG Configuration of a Single Device Using a Download Cable**



**Notes to Figure 13–22:**

- (1) The pull-up resistor should be connected to the same supply voltage as the USB-Blaster, MasterBlaster (VIO pin), ByteBlaster II, or ByteBlasterMV cable.
- (2) Connect the nCONFIG and MSEL[1..0] pins to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect the nCONFIG pin to VCC, and the MSEL[1..0] pins to ground. In addition, pull DCLK and DATA0 to either high or low, whichever is convenient on your board.
- (3) Pin 6 of the header is a VIO reference voltage for the MasterBlaster output driver. VIO should match the device's VCCIO. Refer to the MasterBlaster Serial/USB Communications Cable Data Sheet for this value. In the ByteBlasterMV, this pin is a no connect. In the USB-Blaster and ByteBlaster II, this pin is connected to nCE when it is used for AS programming, otherwise it is a no connect.
- (4) nCE must be connected to GND or driven low for successful JTAG configuration.
- (5) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed other device's nCE pin.

To configure a single device in a JTAG chain, the programming software places all other devices in BYPASS mode. In BYPASS mode, Cyclone II devices pass programming data from the TDI pin to the TDO pin through a single bypass register without being affected internally. This scheme

enables the programming software to program or verify the target device. Configuration data driven into the target device appears on the TDO pin one clock cycle later.

The Quartus II software verifies successful JTAG configuration upon completion. At the end of configuration, the software checks the CONF\_DONE pin through the JTAG port. When the Quartus II software generates a JAM file for a multiple device chain, it contains instructions so that all the devices in the chain are initialized at the same time. If CONF\_DONE is not high, the Quartus II software indicates that configuration has failed. If the CONF\_DONE pin transitions high, the software indicates that configuration was successful. After the configuration bitstream is transmitted serially via the JTAG TDI port, the TCK port is clocked an additional 299 cycles to perform Cyclone II device initialization.

The **Enable user-supplied start-up clock (CLKUSR)** option has no affect on the device initialization since this option is disabled in the SOF when configuring the FPGA in JTAG using the Quartus II programmer and download cable. Therefore, if you turn on the CLKUSR option, you do not need to provide a clock on CLKUSR when you are configuring the FPGA with the Quartus II programmer and a download cable.

Cyclone II devices have dedicated JTAG pins that always function as JTAG pins. You can perform JTAG testing on Cyclone II devices before, after, and during configuration. Cyclone II devices support the BYPASS, IDCODE and SAMPLE instructions during configuration without interruption. All other JTAG instructions may only be issued by first interrupting configuration and reprogramming I/O pins using the CONFIG\_IO instruction.

The CONFIG\_IO instruction allows I/O buffers to be configured via the JTAG port. The CONFIG\_IO instruction interrupts configuration. This instruction allows you to perform board-level testing before configuring the Cyclone II device or waiting for a configuration device to complete configuration. If you interrupt configuration, the Cyclone II device must be reconfigured via JTAG (PULSE\_CONFIG instruction) or by pulsing nCONFIG low after JTAG testing is complete.



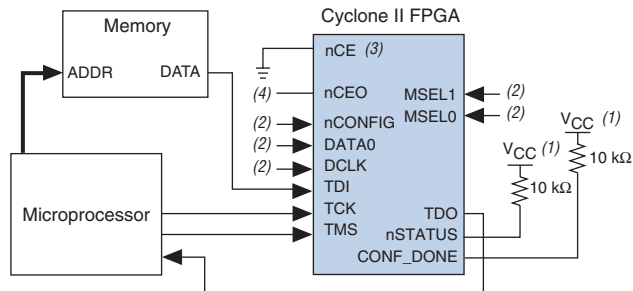
For more information, see the *MorphIO: An I/O Reconfiguration Solution for Altera White Paper*.

The chip-wide reset (DEV\_CLRn) and chip-wide output enable (DEV\_OE) pins on Cyclone II devices do not affect JTAG boundary-scan or programming operations. Toggling these pins does not affect JTAG operations (other than the usual boundary-scan operation).

When designing a Cyclone II board for JTAG configuration, use the guidelines in [Table 13–10](#) for the placement of the dedicated configuration pins.

<b>Table 13–10. Dedicated Configuration Pin Connections During JTAG Configuration</b>	
<b>Signal</b>	<b>Description</b>
nCE	On all Cyclone II devices in the chain, nCE should be driven low by connecting it to ground, pulling it low via a resistor, or driving it by some control circuitry. For devices that are also in multiple device AS, or PS configuration chains, the nCE pins should be connected to GND during JTAG configuration or JTAG configured in the same order as the configuration chain.
nCEO	On all Cyclone II devices in the chain, nCEO can be used as a user I/O or connected to the nCE of the next device. If nCEO is connected to the nCE of the next device, the nCEO pin must be pulled high to $V_{CCIO}$ by an external 10-k $\Omega$ pull-up resistor to help the internal weak pull-up resistor. If the nCEO pin is not connected to the nCE pin of the next device, you can use it as a user I/O pin after configuration.
MSEL	These pins must not be left floating. These pins support whichever non-JTAG configuration is used in production. If only JTAG configuration is used, you should tie these pins to ground.
nCONFIG	Driven high by connecting to $V_{CC}$ , pulling up via a resistor, or driven high by some control circuitry.
nSTATUS	Pull to $V_{CC}$ via a 10-k $\Omega$ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin should be pulled up to $V_{CC}$ individually. nSTATUS pulling low in the middle of JTAG configuration indicates that an error has occurred.
CONF_DONE	Pull to $V_{CC}$ via a 10-k $\Omega$ resistor. When configuring multiple devices in the same JTAG chain, each CONF_DONE pin should be pulled up to $V_{CC}$ individually. CONF_DONE going high at the end of JTAG configuration indicates successful configuration.
DCLK	Should not be left floating. Drive low or high, whichever is more convenient on your board.

[Figure 13–23](#) shows JTAG configuration of a Cyclone II device with a microprocessor.

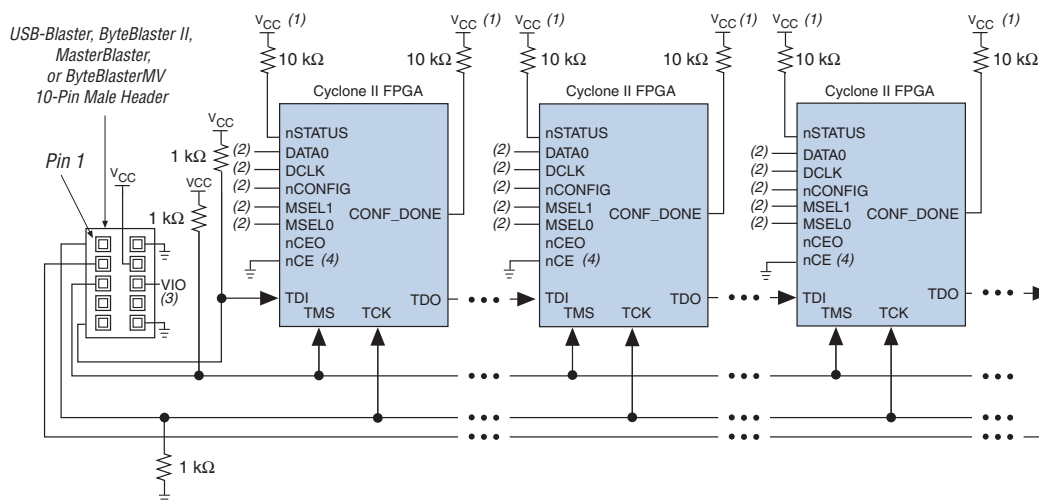
**Figure 13–23. JTAG Configuration of a Single Device Using a Microprocessor****Notes to Figure 13–23:**

- (1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.
- (2) Connect the nCONFIG and MSEL[1..0] pins to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect the nCONFIG pin to V<sub>CC</sub>, and the MSEL[1..0] pins to ground. In addition, pull DCLK and DATA0 to either high or low, whichever is convenient on your board.
- (3) nCE must be connected to GND or driven low for successful JTAG configuration.
- (4) If using an EPCS4 or EPCS1 device, set MSEL[1..0] to 00. See Table 13–4 for more details.

**JTAG Configuration of Multiple Devices**

When programming a JTAG device chain, one JTAG-compatible header is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capability of the download cable. When four or more devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and TMS pins with an on-board buffer.

JTAG-chain device programming is ideal when the system contains multiple devices, or when testing your system using JTAG BST circuitry. Figure 13–24 shows multiple device JTAG configuration.

**Figure 13–24. JTAG Configuration of Multiple Devices Using a Download Cable****Notes to Figure 13–24:**

- (1) The pull-up resistor should be connected to the same supply voltage as the USB-Blaster, MasterBlaster (VIO pin), ByteBlaster II or ByteBlasterMV cable.
- (2) Connect the nCONFIG and MSEL[1..0] pins to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect the nCONFIG pin to VCC, and the MSEL[1..0] pins to ground. In addition, pull DCLK and DATA0 to either high or low, whichever is convenient on your board.
- (3) Pin 6 of the header is a VIO reference voltage for the MasterBlaster output driver. VIO should match the device's VCCIO. Refer to the MasterBlaster Serial/USB Communications Cable Data Sheet for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB-Blaster and ByteBlaster II cable, this pin is connected to nCE when it is used for AS programming, otherwise it is a no connect.
- (4) nCE must be connected to ground or driven low for successful JTAG configuration.

Connect the nCE pin to GND or pull it low during JTAG configuration. In multiple device AS and PS configuration chains, connect the first device's nCE pin to GND and connect its nCEO pin to the nCE pin of the next device in the chain or you can use it as a user I/O pin after configuration.

After the first device completes configuration in a multiple device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. Therefore, if these devices are also in a JTAG chain, you should make sure the nCE pins are connected to GND during JTAG configuration or that the devices are JTAG configured in the same order as the configuration chain. As long as the devices are JTAG configured in the same order as the multiple device configuration chain, the nCEO pin of the previous device drives the nCE pin of the next device low when it has successfully been JTAG configured.



The Quartus II software sets the Cyclone II device `nCEO` pin as an output pin driving to ground by default. If the `nCEO` pin inputs to the next device's `nCE` pin, make sure that the `nCEO` pin is not used as a user I/O pin after configuration.

Other Altera devices that have JTAG support can be placed in the same JTAG chain for device programming and configuration.



For more information on configuring multiple Altera devices in the same configuration chain, see the *Configuring Mixed Altera FPGA Chains* chapter in the *Configuration Handbook*.

### Jam STAPL

Jam STAPL, JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP). Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard. The Jam player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine.



For more information on JTAG and Jam STAPL in embedded environments, see *AN 122: Using Jam STAPL for ISP & ICR via an Embedded Processor*. To download the Jam player, go to the Altera web site ([www.altera.com](http://www.altera.com)).

### Configuring Cyclone II FPGAs with JRunner

JRunner is a software driver that allows you to configure Cyclone II devices through the ByteBlaster II or ByteBlasterMV cables in JTAG mode. The programming input file supported is in `.rbf` format. JRunner also requires a Chain Description File (`.cdf`) generated by the Quartus II software. JRunner is targeted for embedded JTAG configuration. The source code has been developed for the Windows NT operating system (OS). You can customize the code to make it run on your embedded platform.



The RBF file used by the JRunner software driver can not be a compressed RBF file because JRunner uses JTAG-based configuration. During JTAG-based configuration, the real-time decompression feature is not available.



For more information on the JRunner software driver, see *JRunner Software Driver: An Embedded Solution for PLD JTAG Configuration* and the source files on the Altera web site.

## Combining JTAG & Active Serial Configuration Schemes

You can combine the AS configuration scheme with JTAG-based configuration. Set the `MSEL[1..0]` pins to 00 (AS mode) or 10 (Fast AS mode) in this setup, which uses two 10-pin download cable headers on the board. The first header programs the serial configuration device in the system via the AS programming interface, and the second header configures the Cyclone II directly via the JTAG interface.

If you try configuring the device using both schemes simultaneously, JTAG configuration takes precedence and AS configuration is terminated.

When a blank serial configuration device is attached to Cyclone II device, turn on the **Halt on-chip configuration controller** option under the Tools menu by clicking **Options**. The Options dialog box appears. In the **Category** list, select **Programmer** before starting the JTAG configuration with the Quartus II programmer. This option stops the AS reconfiguration loop from a blank serial configuration device before starting the JTAG configuration. This includes using the Serial Flash Loader IP because JTAG is used for configuring the Cyclone II device. Users do not need to recompile their Quartus II designs after turning on this Option.

## Programming Serial Configuration Devices In-System Using the JTAG Interface

Cyclone II devices in a single device chain or in a multiple device chain support in-system programming of a serial configuration device using the JTAG interface via the serial flash loader design. The board's intelligent host or download cable can use the four JTAG pins on the Cyclone II device to program the serial configuration device in system, even if the host or download cable cannot access the configuration device's configuration pins (`DCLK`, `DATA`, `ASDI`, and `nCS` pins).

The serial flash loader design is a JTAG-based in-system programming solution for Altera serial configuration devices. The serial flash loader is a bridge design for the FPGA that uses its JTAG interface to access the EPCS JIC (JTAG Indirect Configuration Device Programming) file and then uses the AS interface to program the EPCS device. Both the JTAG interface and AS interface are bridged together inside the serial flash loader design.

In a multiple device chain, you only need to configure the master Cyclone II device which is controlling the serial configuration device. The slave devices in the multiple device chain which are configured by the serial configuration device do not need to be configured when using this

feature. To use this feature successfully, set the `MSEL[1..0]` pins of the master Cyclone II device to select the AS configuration scheme or fast AS configuration scheme (see [Table 13-1](#)).



The Quartus II software version 4.1 and higher supports serial configuration device ISP through an FPGA JTAG interface using a JIC file.

The serial configuration device in-system programming through the Cyclone II JTAG interface has three stages, which are described in the following sections.

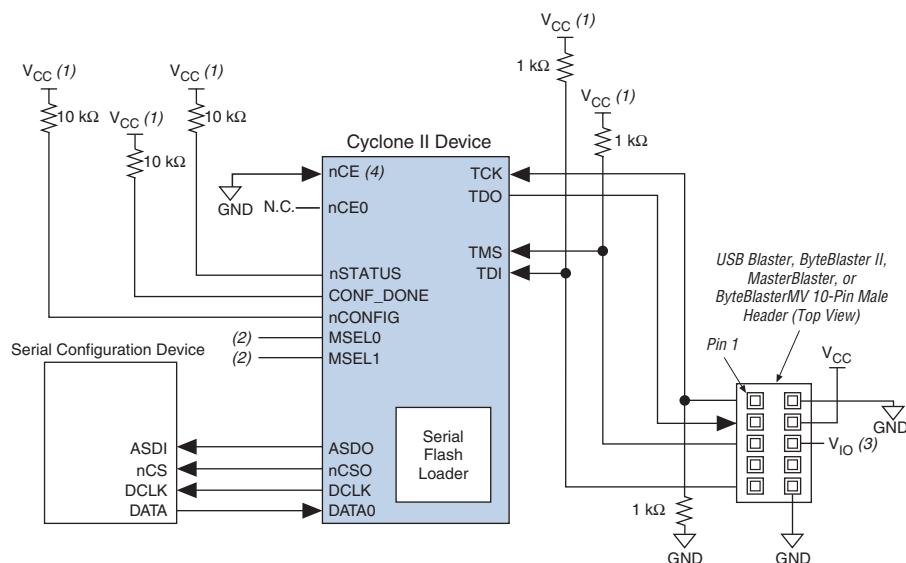
### *Loading the Serial Flash Loader Design*

The serial flash loader design is a design inside the Cyclone II device that bridges the JTAG interface and AS interface inside the Cyclone II device using glue logic.

The intelligent host uses the JTAG interface to configure the master Cyclone II device with a serial flash loader design. The serial flash loader design allows the master Cyclone II device to control the access of four serial configuration device pins, also known as the Active Serial Memory Interface (ASMI) pins, through the JTAG interface. The ASMI pins are the serial clock input (`DCLK`), serial data output (`DATA`), AS data input (`ASDI`), and an active-low chip select (`nCS`) pins.

If you configure a master Cyclone II device with a serial flash loader design, the master Cyclone II device can enter user mode even though the slave devices in the multiple device chain are not being configured. The master Cyclone II device can enter user mode with a serial flash loader design even though the `CONF_DONE` signal is externally held low by the other slave devices in chain. [Figure 13-25](#) shows the JTAG configuration of a single Cyclone II device with a serial flash loader design.



**Figure 13–25. JTAG Configuration of a Single Device Using a Download Cable****Notes to Figure 13–25:**

- (1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster ( $V_{IO}$  pin), ByteBlaster II, or ByteBlasterMV cable.
- (2) The  $nCONFIG$ ,  $MSEL[1..0]$  pins should be connected to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect  $nCONFIG$  to  $V_{CC}$ , and  $MSEL[1..0]$  to ground. Pull  $DCLK$  either high or low, whichever is convenient on your board.
- (3) Pin 6 of the header is a  $V_{IO}$  reference voltage for the MasterBlaster output driver.  $V_{IO}$  should match the device's  $V_{CCIO}$ . Refer to the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cables, this pin is connected to  $nCE$  when it is used for active serial programming, otherwise it is a no connect.
- (4)  $nCE$  must be connected to GND or driven low for successful JTAG configuration.

**ISP of Serial Configuration Device**

In the second stage, the serial flash loader design in the master Cyclone II device allows you to write the configuration data for the device chain into the serial configuration device by using the Cyclone II JTAG interface. The JTAG interface sends the programming data for the serial configuration device to the Cyclone II device first. The Cyclone II device then uses the ASMI pins to transmit the data to the serial configuration device.

*Reconfiguration*

After all the configuration data is written into the serial configuration device successfully, the Cyclone II device does not reconfigure by itself. The intelligent host issues the `PULSE_NCONFIG` JTAG instruction to initialize the reconfiguration process. During reconfiguration, the master Cyclone II device is reset and the serial flash loader design no longer exists in the Cyclone II device and the serial configuration device configures all the devices in the chain with your user design.

## Device Configuration Pins

This section describes the connections and functionality of all the configuration related pins on the Cyclone II device. [Table 13–11](#) describes the dedicated configuration pins, which are required to be connected properly on your board for successful configuration. Some of these pins may not be required for your configuration schemes.

**Table 13–11. Dedicated Configuration Pins on the Cyclone II Device (Part 1 of 5)**

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
MSEL[1..0]	N/A	All	Input	<p>This pin is a two-bit configuration input that sets the Cyclone II device configuration scheme. See <a href="#">Table 13–1</a> for the appropriate settings.</p> <p>You must connect these pins to <math>V_{CCIO}</math> or ground.</p> <p>The MSEL[1..0] pins have 9-k<math>\Omega</math> internal pull-down resistors that are always active.</p>
nCONFIG	N/A	All	Input	<p>This pin is a configuration control input. If this pin is pulled low during user mode, the FPGA loses its configuration data, enters a reset state, and tri-states all I/O pins. Transitioning this pin high initiates a reconfiguration.</p> <p>If your configuration scheme uses an enhanced configuration device or EPC2 device, you can connect the nCONFIG pin directly to <math>V_{CC}</math> or to the configuration device's nINIT_CONF pin.</p> <p>The input buffer on this pin supports hysteresis using Schmitt trigger circuitry.</p>

**Table 13–11. Dedicated Configuration Pins on the Cyclone II Device (Part 2 of 5)**

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nSTATUS	N/A	All	Bidirectional open-drain	<p>The Cyclone II device drives nSTATUS low immediately after power-up and releases it after the POR time.</p> <p>This pin provides a status output and input for the Cyclone II device. If the Cyclone II device detects an error during configuration, it drives the nSTATUS pin low to stop configuration. If an external source (for example, another Cyclone II device) drives the nSTATUS pin low during configuration or initialization, the target device enters an error state.</p> <p>Driving nSTATUS low after configuration and initialization does not affect the configured device. If your design uses a configuration device, driving nSTATUS low causes the configuration device to attempt to configure the FPGA, but since the FPGA ignores transitions on nSTATUS in user mode, the FPGA does not reconfigure. To initiate a reconfiguration, pull the nCONFIG pin low.</p> <p>The enhanced configuration devices' and EPC2 devices' OE and nCS pins are connected to the Cyclone II device's nSTATUS and CONF_DONE pins, respectively, and have optional internal programmable pull-up resistors. If you use these internal pull-up resistors on the enhanced configuration device, do not use external 10-k<math>\Omega</math> pull-up resistors on these pins. When using EPC2 devices, you should only use external 10-k<math>\Omega</math> pull-up resistors.</p> <p>The input buffer on this pin supports hysteresis using Schmitt trigger circuitry.</p>

**Table 13–11. Dedicated Configuration Pins on the Cyclone II Device (Part 3 of 5)**

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
CONF_DONE	N/A	All	Bidirectional open-drain	<p>This pin is a status output and input.</p> <p>The target Cyclone II device drives the CONF_DONE pin low before and during configuration. Once the Cyclone II device receives all the configuration data without error and the initialization cycle starts, it releases CONF_DONE. Driving CONF_DONE low during user mode does not affect the configured device. Do not drive CONF_DONE low before the device enters user mode.</p> <p>After the Cyclone II device receives all the data, the CONF_DONE pin transitions high, and the device initializes and enters user mode. The CONF_DONE pin must have an external 10-k<math>\Omega</math> pull-up resistor in order for the device to initialize.</p> <p>Driving CONF_DONE low after configuration and initialization does not affect the configured device.</p> <p>The enhanced configuration devices' and EPC2 devices' OE and nCS pins are connected to the Cyclone II device's nSTATUS and CONF_DONE pins, respectively, and have optional internal programmable pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external 10-k<math>\Omega</math> pull-up resistors should not be used on these pins. When using EPC2 devices, you should only use external 10-k<math>\Omega</math> pull-up resistors.</p> <p>The input buffer on this pin supports hysteresis using Schmitt trigger circuitry.</p>
nCE	N/A	All	Input	<p>This pin is an active-low chip enable. The nCE pin activates the device with a low signal to allow configuration. The nCE pin must be held low during configuration, initialization, and user mode. In single device configuration, it should be tied low. In multiple device configuration, nCE of the first device is tied low while its nCEO pin is connected to nCE of the next device in the chain.</p> <p>The nCE pin must also be held low for successful JTAG programming of the FPGA.</p> <p>The input buffer on this pin supports hysteresis using Schmitt trigger circuitry.</p>

**Table 13–11. Dedicated Configuration Pins on the Cyclone II Device (Part 4 of 5)**

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCEO	N/A if option is on. I/O if option is off.	All	Output	<p>This pin is an output that drives low when device configuration is complete. In single device configuration, you can leave this pin floating or use it as a user I/O pin after configuration. In multiple device configuration, this pin inputs the next device's nCE pin. The nCEO of the last device in the chain can be left floating or used as a user I/O pin after configuration.</p> <p>If you use the nCEO pin to feed next device's nCE pin, use an external 10-k<math>\Omega</math> pull-up resistor to pull the nCEO pin high to the V<sub>CCIO</sub> voltage of its I/O bank to help the internal weak pull-up resistor.</p> <p>Use the Quartus II software to make this pin a user I/O pin.</p>
ASDO	N/A in AS mode I/O in PS and JTAG mode	AS	Output	<p>This pin sends a control signal from the Cyclone II device to the serial configuration device in AS mode and is used to read out configuration data.</p> <p>In AS mode, ASDO has an internal pull-up that is always active.</p>
nCSO	N/A in AS mode I/O in PS and JTAG mode	AS	Output	<p>This pin sends an output control signal from the Cyclone II device to the serial configuration device in AS mode that enables the configuration device.</p> <p>In AS mode, nCSO has an internal pull-up resistor that is always active.</p>

<b>Table 13–11. Dedicated Configuration Pins on the Cyclone II Device (Part 5 of 5)</b>				
<b>Pin Name</b>	<b>User Mode</b>	<b>Configuration Scheme</b>	<b>Pin Type</b>	<b>Description</b>
DCLK	N/A	PS, AS	Input (PS) Output (AS)	<p>In PS configuration, DCLK is the clock input used to clock data from an external source into the target device. Data is latched into the Cyclone II device on the rising edge of DCLK.</p> <p>In AS mode, DCLK is an output from the Cyclone II device that provides timing for the configuration interface. In AS mode, DCLK has an internal pull-up that is always active.</p> <p>After configuration, this pin is tri-stated. If you are using a configuration device, it drives DCLK low after configuration is complete. If your design uses a control host, drive DCLK either high or low. Toggling this pin after configuration does not affect the configured device.</p> <p>The input buffer on this pin supports hysteresis using Schmitt trigger circuitry.</p>
DATA0	N/A	All	Input	<p>This is the data input pin. In serial configuration modes, bit-wide configuration data is presented to the target device on the DATA0 pin.</p> <p>In AS mode, DATA0 has an internal pull-up resistor that is always active.</p> <p>After configuration, EPC1 and EPC1441 devices tri-state this pin, while enhanced configuration and EPC2 devices drive this pin high.</p> <p>The input buffer on this pin supports hysteresis using Schmitt trigger circuitry.</p>

Table 13–12 describes the optional configuration pins. If these optional configuration pins are not enabled in the Quartus II software, they are available as general-purpose user I/O pins. Therefore during configuration, these pins function as user I/O pins and are tri-stated with weak pull-up resistors.

<b>Table 13–12. Optional Configuration Pins</b>			
<b>Pin Name</b>	<b>User Mode</b>	<b>Pin Type</b>	<b>Description</b>
CLKUSR	N/A if option is on. I/O if option is off.	Input	This is an optional user-supplied clock input that synchronizes the initialization of one or more devices. This pin is enabled by turning on the <b>Enable user-supplied start-up clock (CLKUSR)</b> option in the Quartus II software.
INIT_DONE	N/A if option is on. I/O if option is off.	Output open-drain	This is a status pin that can be used to indicate when the device has initialized and is in user mode. When <code>nCONFIG</code> is low and during the beginning of configuration, the <code>INIT_DONE</code> pin is tri-stated and pulled high due to an external 10-k $\Omega$ pull-up resistor. Once the option bit to enable <code>INIT_DONE</code> is programmed into the device (during the first frame of configuration data), the <code>INIT_DONE</code> pin goes low. When initialization is complete, the <code>INIT_DONE</code> pin is released and pulled high and the FPGA enters user mode. Thus, the monitoring circuitry must be able to detect a low-to-high transition. This pin is enabled by turning on the <b>Enable INIT_DONE output</b> option in the Quartus II software.
DEV_OE	N/A if option is on. I/O if option is off.	Input	Optional pin that allows the user to override all tri-states on the device. When this pin is driven low, all I/O pins are tri-stated. When this pin is driven high, all I/O pins behave as programmed. This pin is enabled by turning on the <b>Enable device-wide output enable (DEV_OE)</b> option in the Quartus II software.
DEV_CLRn	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all clears on all device registers. When this pin is driven low, all registers are cleared. When this pin is driven high, all registers behave as programmed. This pin is enabled by turning on the <b>Enable device-wide reset (DEV_CLRn)</b> option in the Quartus II software.

## Conclusion

Table 13–13 describes the dedicated JTAG pins. JTAG pins must be kept stable before and during configuration to prevent accidental loading of JTAG instructions. The TCK pin has a weak internal pull-down resistor and the TDI and TMS JTAG input pins have weak internal pull-up resistors.

<b>Table 13–13. Dedicated JTAG Pins</b>			
<b>Pin Name</b>	<b>User Mode</b>	<b>Pin Type</b>	<b>Description</b>
TDI	N/A	Input	<p>Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK.</p> <p>If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V<sub>CC</sub>.</p> <p>The input buffer on this pin supports hysteresis using Schmitt trigger circuitry.</p>
TDO	N/A	Output	<p>Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device.</p> <p>If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected.</p>
TMS	N/A	Input	<p>Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK.</p> <p>If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V<sub>CC</sub>.</p> <p>The input buffer on this pin supports hysteresis using Schmitt trigger circuitry.</p>
TCK	N/A	Input	<p>The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge.</p> <p>If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND.</p> <p>The input buffer on this pin supports hysteresis using Schmitt trigger circuitry.</p>

## Conclusion

Cyclone II devices can be configured in AS, PS or JTAG configuration schemes to fit your system's need. The AS configuration scheme supported by Cyclone II devices can now operate at a higher DCLK



frequency (up to 40 MHz), which reduces your configuration time. In addition, Cyclone II devices can receive a compressed configuration bitstream and decompress this data on-the-fly in the AS or PS configuration scheme, which further reduces storage requirements and configuration time.

## Document Revision History

Table 13–14 shows the revision history for this document.

<b>Table 13–14. Document Revision History</b>		
<b>Date &amp; Document Version</b>	<b>Changes Made</b>	<b>Summary of Changes</b>
February 2007 v3.1	<ul style="list-style-type: none"> <li>Added document revision history.</li> <li>Added <i>Note (1)</i> to Table 13–1.</li> <li>Added <i>Note (1)</i> to Table 13–4.</li> <li>Updated Figure 13–3.</li> <li>Updated Figures 13–6 and 13–7.</li> <li>Updated <i>Note (2)</i> to Figure 13–13.</li> <li>Updated “Single Device PS Configuration Using a Configuration Device” section.</li> <li>Updated <i>Note (2)</i> to Figure 13–14.</li> <li>Updated <i>Note (2)</i> to Figure 13–15.</li> <li>Updated <i>Note (2)</i> to Figure 13–16.</li> <li>Updated <i>Note (2)</i> to Figure 13–17.</li> <li>Updated <i>Note (4)</i> to Figure 13–21.</li> <li>Updated <i>Note (2)</i> to Figure 13–25.</li> </ul>	<ul style="list-style-type: none"> <li>Changed unit ‘kw’ to ‘kΩ’ in Figures 13–6 and 13–7.</li> <li>Added note about serial configuration devices supporting 20 MHz and 40 MHz DCLK.</li> <li>Added information about the need for a resistor on nCONFIG if reconfiguration is required.</li> <li>Added information about MSEL[1..0] internal pull-down resistor value.</li> </ul>
July 2005 v2.0	<ul style="list-style-type: none"> <li>Updated “Configuration Stage” section.</li> <li>Updated “PS Configuration Using a Download Cable” section.</li> <li>Updated Figures 13–8, 13–12, and 13–18.</li> </ul>	—
November 2004 v1.1	<ul style="list-style-type: none"> <li>Updated “Configuration Stage” section in “Single Device AS Configuration” section.</li> <li>Updated “Initialization Stage” section in “Single Device AS Configuration” section.</li> <li>Updated Figure 13–8.</li> <li>Updated “Initialization Stage” section in “Single Device PS Configuration Using a MAX II Device as an External Host” section.</li> <li>Updated Table 13–7.</li> <li>Updated “Single Device PS Configuration Using a Configuration Device” section.</li> <li>Updated “Initialization Stage” section in “Single Device PS Configuration Using a Configuration Device” section.</li> <li>Updated Figure 13–18.</li> <li>Updated “Single Device JTAG Configuration” section.</li> </ul>	—
June 2004 v1.0	Added document to the Cyclone II Device Handbook.	—

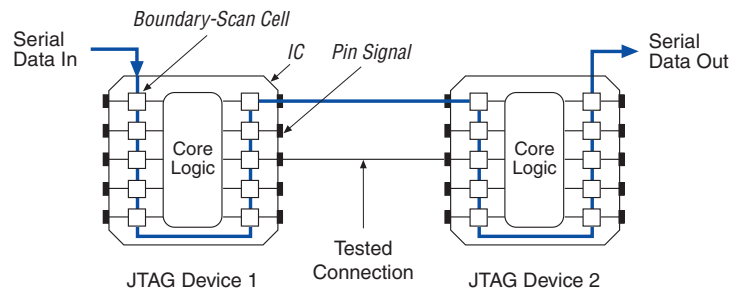
### Introduction

As printed circuit boards (PCBs) become more complex, the need for thorough testing becomes increasingly important. Advances in surface-mount packaging and PCB manufacturing have resulted in smaller boards, making traditional test methods (e.g., external test probes and “bed-of-nails” test fixtures) harder to implement. As a result, cost savings from PCB space reductions are sometimes offset by cost increases in traditional testing methods.

In the 1980s, the Joint Test Action Group (JTAG) developed a specification for boundary-scan testing that was later standardized as the IEEE Std. 1149.1 specification. This boundary-scan test (BST) architecture offers the capability to efficiently test components on PCBs with tight lead spacing.

This BST architecture tests pin connections without using physical test probes and captures functional data while a device is operating normally. Boundary-scan cells in a device force signals onto pins or capture data from pin or logic array signals. Forced test data is serially shifted into the boundary-scan cells. Captured data is serially shifted out and externally compared with expected results. Figure 14–1 shows the concept of boundary-scan testing.

**Figure 14–1. IEEE Std. 1149.1 Boundary-Scan Testing**



This chapter discusses how to use the IEEE Std. 1149.1 BST circuitry in Cyclone™ II devices, including:

- IEEE Std. 1149.1 BST architecture
- IEEE Std. 1149.1 boundary-scan register
- IEEE Std. 1149.1 BST operation control
- I/O voltage support in JTAG chain
- Using IEEE Std. 1149.1 BST circuitry
- Disabling IEEE Std. 1149.1 BST circuitry
- Guidelines for IEEE Std. 1149.1 boundary-scan testing
- Boundary-Scan Description Language (BSDL) support

In addition to BST, you can use the IEEE Std. 1149.1 controller for Cyclone II device in-circuit reconfiguration (ICR). However, this chapter only discusses the BST feature of the IEEE Std. 1149.1 circuitry.



For information on configuring Cyclone II devices via the IEEE Std. 1149.1 circuitry, see the *Configuring Cyclone II Devices* chapter in Volume 1 of the *Cyclone II Device Handbook*.

## IEEE Std. 1149.1 BST Architecture

A Cyclone II device operating in IEEE Std. 1149.1 BST mode uses four required pins, TDI, TDO, TMS and TCK. The optional TRST pin is not available in Cyclone II devices. TDI and TMS pins have weak internal pull-up resistors while TCK has weak internal pull-down resistors. All user I/O pins are tri-stated during JTAG configuration. [Table 14–1](#) summarizes the functions of each of these pins.

**Table 14–1. IEEE Std. 1149.1 Pin Descriptions**

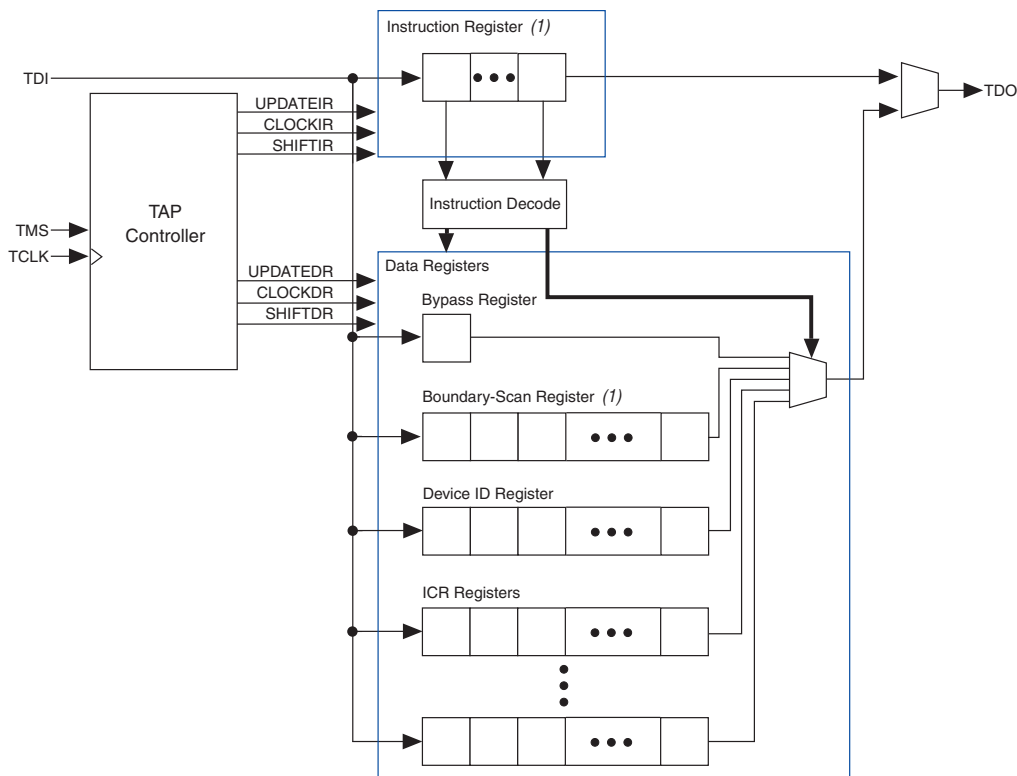
Pin	Description	Function
TDI	Test data input	Serial input pin for instructions as well as test and programming data. Signal applied to TDI is expected to change state at the falling edge of TCK. Data is shifted in on the rising edge of TCK.
TDO	Test data output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device.
TMS	Test mode select	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur at the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. During non-JTAG operation, TMS is recommended to be driven high.
TCK	Test clock input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. The clock input waveform should have a 50% duty cycle.

The IEEE Std. 1149.1 BST circuitry requires the following registers:

- The instruction register determines the action to be performed and the data register to be accessed.
- The bypass register is a 1-bit-long data register that provides a minimum-length serial path between TDI and TDO.
- The boundary-scan register is a shift register composed of all the boundary-scan cells of the device.

Figure 14–2 shows a functional model of the IEEE Std. 1149.1 circuitry.

**Figure 14–2. IEEE Std. 1149.1 Circuitry**



**Note to Figure 14–2:**

- (1) For register lengths, see the device data sheet in the *Configuration & Testing* chapter in Volume 1 of the *Cyclone II Device Handbook*.

IEEE Std. 1149.1 boundary-scan testing is controlled by a test access port (TAP) controller. For more information on the TAP controller, see “IEEE Std. 1149.1 BST Operation Control” on page 14–6. The TMS and TCK pins

operate the TAP controller, and the TDI and TDO pins provide the serial path for the data registers. The TDI pin also provides data to the instruction register, which then generates control logic for the data registers.

## IEEE Std. 1149.1 Boundary-Scan Register

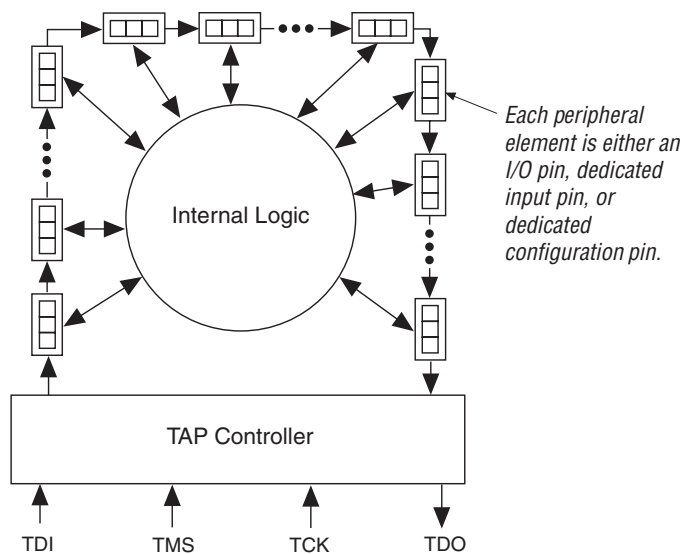


The boundary-scan register is a large serial shift register that uses the TDI pin as an input and the TDO pin as an output. The boundary-scan register consists of 3-bit peripheral elements that are associated with Cyclone II I/O pins. You can use the boundary-scan register to test external pin connections or to capture internal data.

See the *Configuration & Testing* chapter in Volume 1 of the *Cyclone II Device Handbook* for the Cyclone II device boundary-scan register lengths.

Figure 14–3 shows how test data is serially shifted around the periphery of the IEEE Std. 1149.1 device.

**Figure 14–3. Boundary-Scan Register**



### Boundary-Scan Cells of a Cyclone II Device I/O Pin

The Cyclone II device 3-bit boundary-scan cell (BSC) consists of a set of capture registers and a set of update registers. The capture registers can connect to internal device data via the  $OUT_J$  and  $OE_J$  signals, and connect

to external device data via the `PIN_IN` signal, while the update registers connect to external data through the `PIN_OUT` and `PIN_OE` signals. The global control signals for the IEEE Std. 1149.1 BST registers (for example, shift, clock, and update) are generated internally by the TAP controller. The `MODE` signal is generated by a decode of the instruction register. The data signal path for the boundary-scan register runs from the serial data in (`SDI`) signal to the serial data out (`SDO`) signal. The scan register begins at the `TDI` pin and ends at the `TDO` pin of the device.

Figure 14–4 shows the Cyclone II device's user I/O boundary-scan cell.

Figure 14–4. Cyclone II Device's User I/O BSC with IEEE Std. 1149.1 BST Circuitry

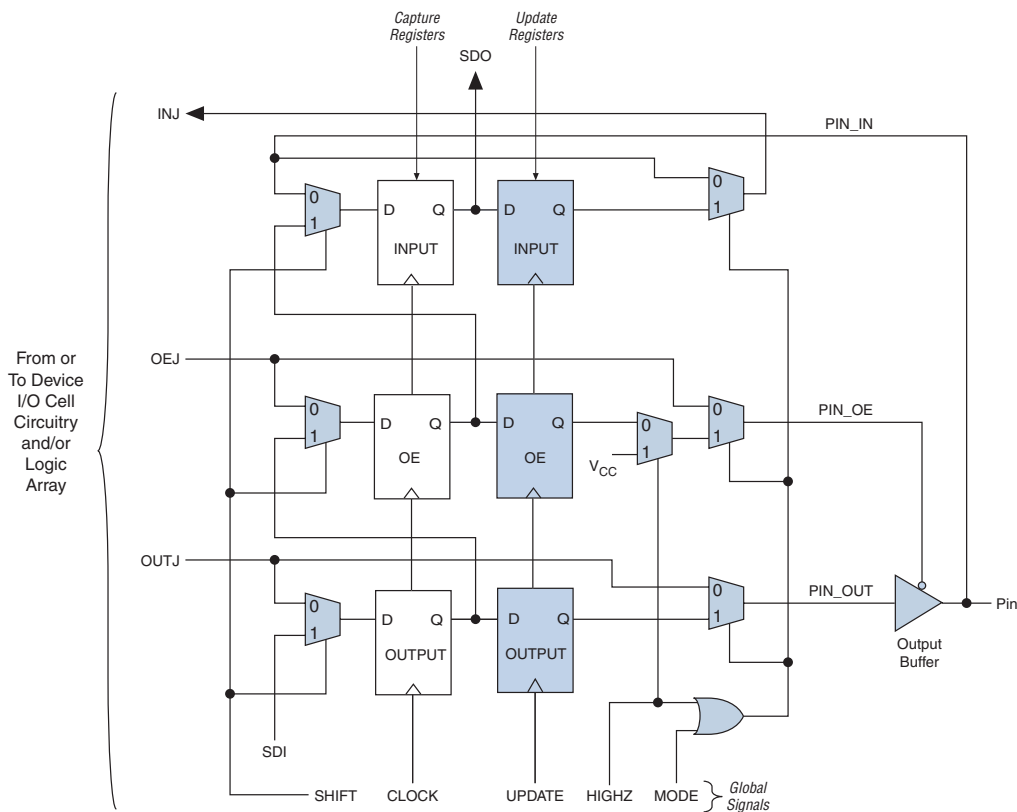


Table 14–2 describes the capture and update register capabilities of all types of boundary-scan cells within Cyclone II devices.

<b>Table 14–2. Cyclone II Device Boundary Scan Cell Descriptions</b> <i>Note (1)</i>							
Pin Type	Captures			Drives			Comments
	Output Capture Register	OE Capture Register	Input Capture Register	Output Update Register	OE Update Register	Input Update Register	
User I/O pins	OUTJ	OEJ	PIN_IN	PIN_OUT	PIN_OE	INJ	
Dedicated clock input	0	1	PIN_IN	N.C. (2)	N.C. (2)	N.C. (2)	PIN_IN drives to clock network or logic array
Dedicated input (3)	0	1	PIN_IN	N.C. (2)	N.C. (2)	N.C. (2)	PIN_IN drives to control logic
Dedicated bidirectional (open drain) (4)	0	OEJ	PIN_IN	N.C. (2)	N.C. (2)	N.C. (2)	PIN_IN drives to configuration control
Dedicated bidirectional (5)	OUTJ	OEJ	PIN_IN	N.C. (2)	N.C. (2)	N.C. (2)	OUTJ drives to output buffer

**Notes to Table 14–2:**

- (1) TDI, TDO, TMS, TCK, all V<sub>CC</sub> and GND pin types do not have BSCs.
- (2) N.C.: no connect.
- (3) This includes nCONFIG, MSEL0, MSEL1, DATA0, and nCE pins and DCLK (when not used in Active Serial mode).
- (4) This includes CONF\_DONE and nSTATUS pins.
- (5) This includes DCLK (when not used in Active Serial mode).

## IEEE Std. 1149.1 BST Operation Control

Cyclone II devices implement the following IEEE Std. 1149.1 BST instructions: SAMPLE/PRELOAD, EXTEST, BYPASS, IDCODE, USERCODE, CLAMP, and HIGHZ. The BST instruction length is 10 bits. These instructions are described later in this chapter.

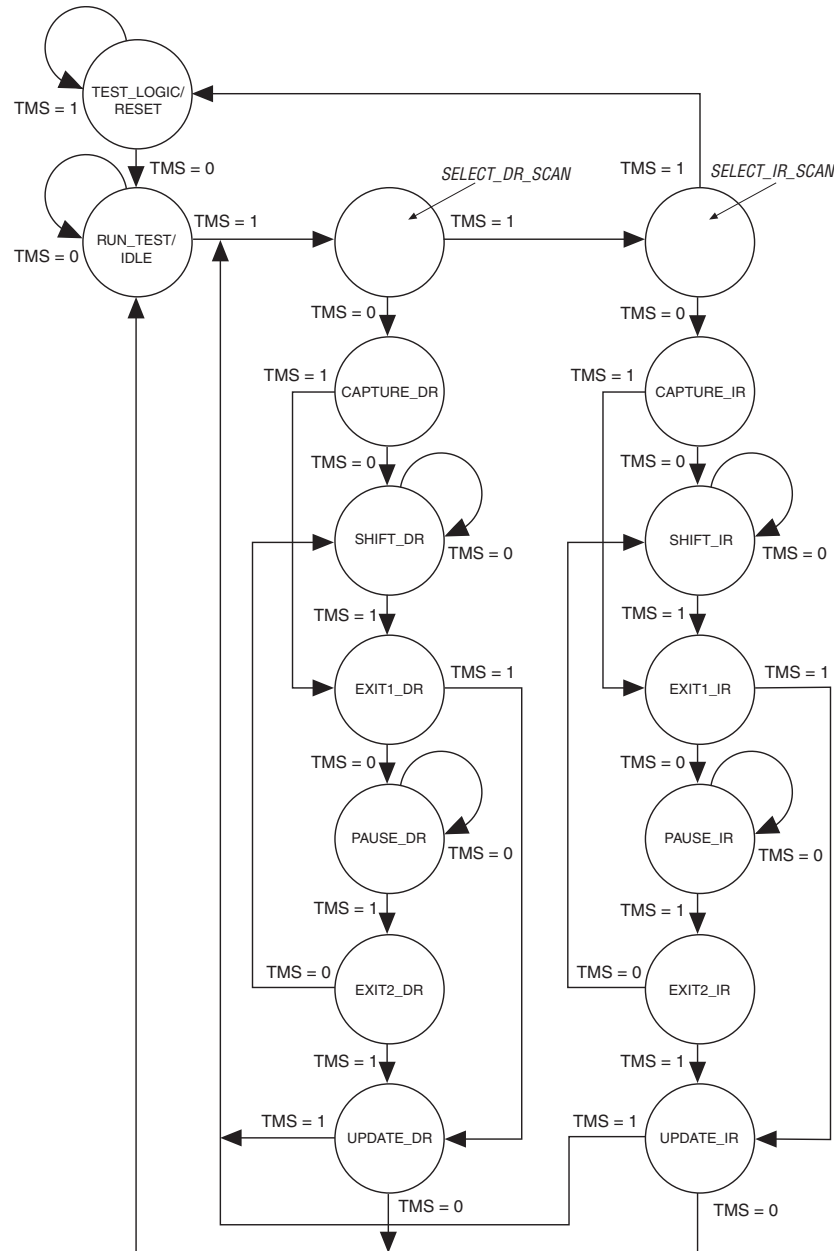


For summaries of the BST instructions and their instruction codes, see the *Configuration & Testing* chapter in Volume 1 of the *Cyclone II Device Handbook*.

The IEEE Std. 1149.1 test access port (TAP) controller, a 16-state state machine clocked on the rising edge of TCK, uses the TMS pin to control IEEE Std. 1149.1 operation in the device. Figure 14–5 shows the TAP controller state machine.

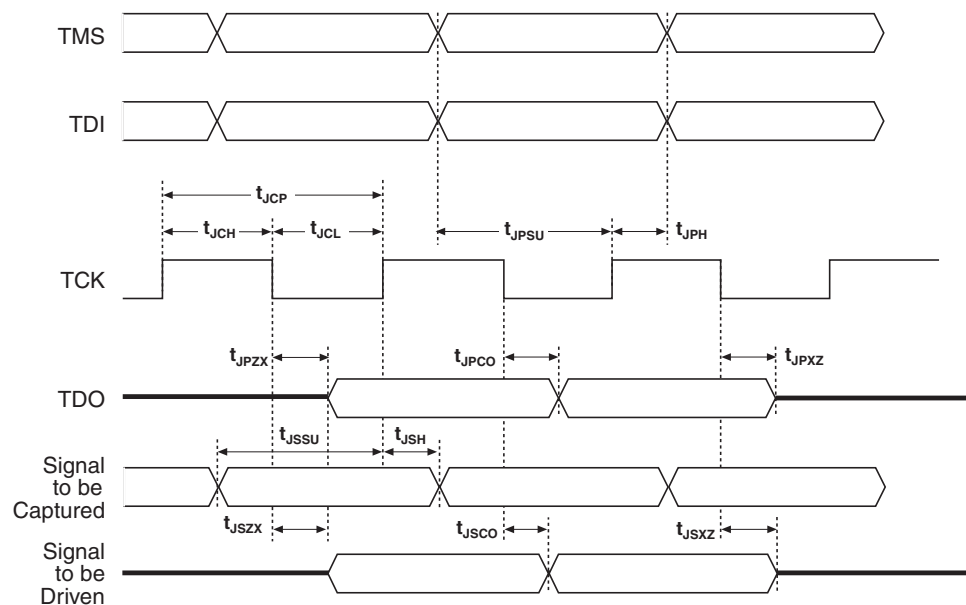


Figure 14–5. IEEE Std. 1149.1 TAP Controller State Machine

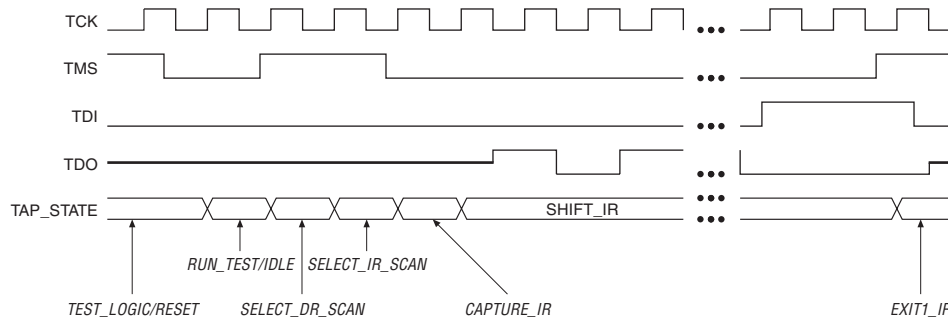


When the TAP controller is in the `TEST_LOGIC/RESET` state, the BST circuitry is disabled, the device is in normal operation, and the instruction register is initialized with `IDCODE` as the initial instruction. At device power-up, the TAP controller starts in this `TEST_LOGIC/RESET` state. In addition, forcing the TAP controller to the `TEST_LOGIC/RESET` state is done by holding `TMS` high for five `TCK` clock cycles. Once in the `TEST_LOGIC/RESET` state, the TAP controller remains in this state as long as `TMS` is held high (while `TCK` is clocked). Figure 14–6 shows the timing requirements for the IEEE Std. 1149.1 signals.

Figure 14–6. IEEE Std. 1149.1 Timing Waveforms



To start IEEE Std. 1149.1 operation, select an instruction mode by advancing the TAP controller to the shift instruction register (`SHIFT_IR`) state and shift in the appropriate instruction code on the `TDI` pin. The waveform diagram in Figure 14–7 represents the entry of the instruction code into the instruction register. It shows the values of `TCK`, `TMS`, `TDI`, `TDO`, and the states of the TAP controller. From the `RESET` state, `TMS` is clocked with the pattern 01100 to advance the TAP controller to `SHIFT_IR`.

**Figure 14–7. Selecting the Instruction Mode**

The TDO pin is tri-stated in all states except in the `SHIFT_IR` and `SHIFT_DR` states. The TDO pin is activated at the first falling edge of TCK after entering either of the shift states and is tri-stated at the first falling edge of TCK after leaving either of the shift states.

When the `SHIFT_IR` state is activated, TDO is no longer tri-stated, and the initial state of the instruction register is shifted out on the falling edge of TCK. TDO continues to shift out the contents of the instruction register as long as the `SHIFT_IR` state is active. The TAP controller remains in the `SHIFT_IR` state as long as TMS remains low.

During the `SHIFT_IR` state, an instruction code is entered by shifting data on the TDI pin on the rising edge of TCK. The last bit of the instruction code must be clocked at the same time that the next state, `EXIT1_IR`, is activated. Set TMS high to activate the `EXIT1_IR` state. Once in the `EXIT1_IR` state, TDO becomes tri-stated again. TDO is always tri-stated except in the `SHIFT_IR` and `SHIFT_DR` states. After an instruction code is entered correctly, the TAP controller advances to serially shift test data in one of seven modes (`SAMPLE/PRELOAD`, `EXTEST`, `BYPASS`, `IDCODE`, `USERCODE`, `CLAMP`, or `HIGHZ`) that are described below.

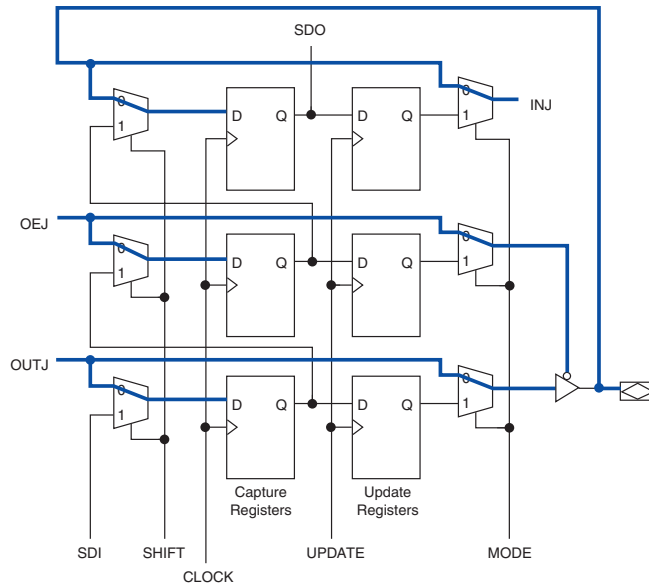
### **SAMPLE/PRELOAD Instruction Mode**

The `SAMPLE/PRELOAD` instruction mode allows you to take a snapshot of device data without interrupting normal device operation. You can also use this instruction to preload the test data into the update registers prior to loading the `EXTEST` instruction. [Figure 14–8](#) shows the capture, shift, and update phases of the `SAMPLE/PRELOAD` mode.

Figure 14–8. IEEE Std. 1149.1 BST SAMPLE/PRELOAD Mode

### Capture Phase

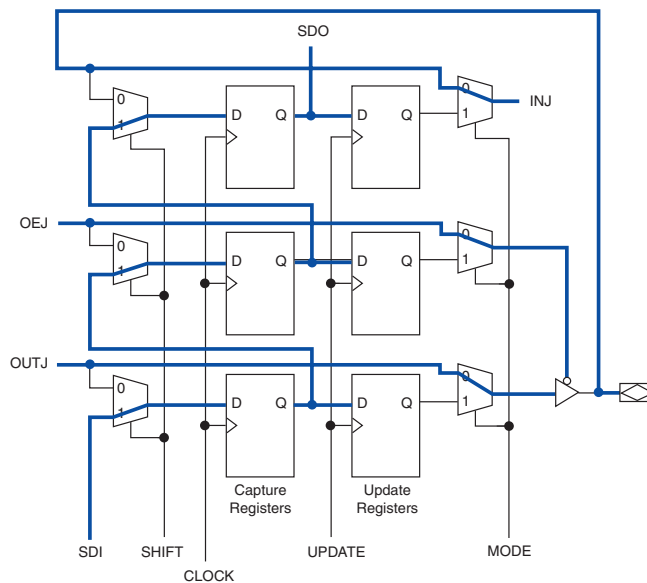
*In the capture phase, the signals at the pin, OEJ and OUTJ, are loaded into the capture registers. The CLOCK signals are supplied by the TAP controller's CLOCKDR output. The data retained in these registers consists of signals from normal device operation.*



### Shift & Update Phases

*In the shift phase, the previously captured signals at the pin, OEJ and OUTJ, are shifted out of the boundary-scan register via the TDO pin using CLOCK. As data is shifted out, the patterns for the next test can be shifted in via the TDI pin.*

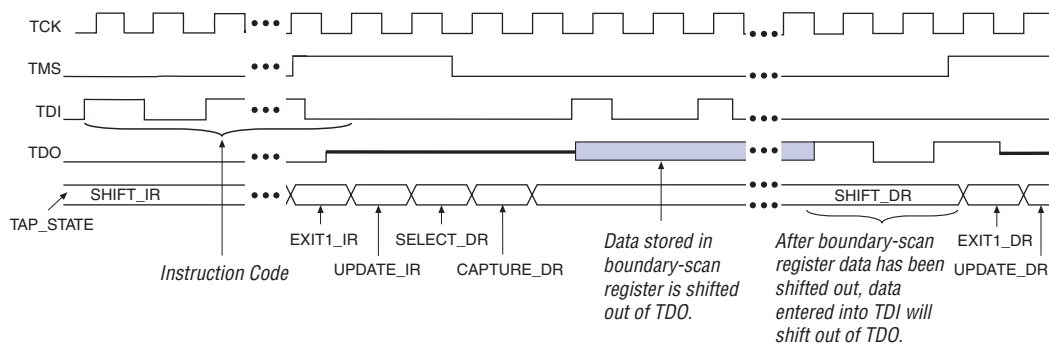
*In the update phase, data is transferred from the capture to the UPDATE registers using the UPDATE clock. The data stored in the UPDATE registers can be used for the EXTEST instruction.*



During the capture phase, multiplexers preceding the capture registers select the active device data signals. This data is then clocked into the capture registers. The multiplexers at the outputs of the update registers also select active device data to prevent functional interruptions to the device. During the shift phase, the boundary-scan shift register is formed by clocking data through capture registers around the device periphery, then out of the TDO pin. The device can simultaneously shift new test data into TDI and replace the contents of the capture registers. During the update phase, data in the capture registers is transferred to the update registers. This data can then be used in the EXTEST instruction mode. See “EXTEST Instruction Mode” on page 14–11 for more information.

Figure 14–9 shows the SAMPLE/PRELOAD waveforms. The SAMPLE/PRELOAD instruction code is shifted in through the TDI pin. The TAP controller advances to the CAPTURE\_DR state, then to the SHIFT\_DR state, where it remains if TMS is held low. The data that was present in the capture registers after the capture phase is shifted out of the TDO pin. New test data shifted into the TDI pin appears at the TDO pin after being clocked through the entire boundary-scan register. Figure 14–9 shows that the instruction code at TDI does not appear at the TDO pin until after the capture register data is shifted out. If TMS is held high on two consecutive TCK clock cycles, the TAP controller advances to the UPDATE\_DR state for the update phase.

**Figure 14–9. SAMPLE/PRELOAD Shift Data Register Waveforms**



## EXTEST Instruction Mode

The EXTEST instruction mode is used to check external pin connections between devices. Unlike the SAMPLE/PRELOAD mode, EXTEST allows test data to be forced onto the pin signals. By forcing known logic high and low levels on output pins, opens and shorts can be detected at pins of any device in the scan chain.

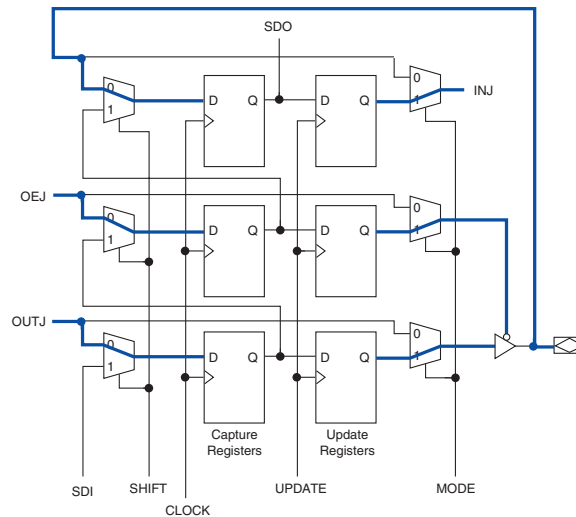
Figure 14–10 shows the capture, shift, and update phases of the EXTEST mode.

**Figure 14–10. IEEE Std. 1149.1 BST EXTEST Mode**

### Capture Phase

In the capture phase, the signals at the pin, OEJ and OUTJ, are loaded into the capture registers. The CLOCK signals are supplied by the TAP controller's CLOCKDR output. Previously retained data in the update registers drive the PIN\_IN, INJ, and allows the I/O pin to tri-state or drive a signal out.

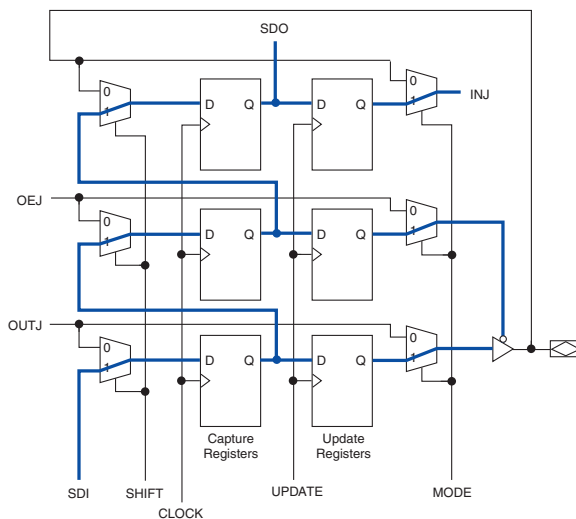
A "1" in the OEJ update register tri-states the output buffer.



### Shift & Update Phases

In the shift phase, the previously captured signals at the pin, OEJ and OUTJ, are shifted out of the boundary-scan register via the TDO pin using CLOCK. As data is shifted out, the patterns for the next test can be shifted in via the TDI pin.

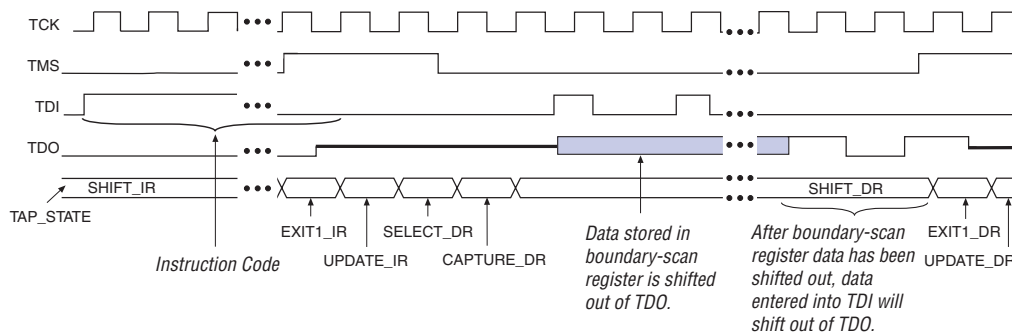
In the update phase, data is transferred from the capture registers to the update registers using the UPDATE clock. The update registers then drive the PIN\_IN, INJ, and allow the I/O pin to tri-state or drive a signal out.



EXTEST selects data differently than SAMPLE/PRELOAD. EXTEST chooses data from the update registers as the source of the output and output enable signals. Once the EXTEST instruction code is entered, the multiplexers select the update register data. Thus, data stored in these registers from a previous EXTEST or SAMPLE/PRELOAD test cycle can be forced onto the pin signals. In the capture phase, the results of this test data are stored in the capture registers, then shifted out of TDO during the shift phase. New test data can then be stored in the update registers during the update phase.

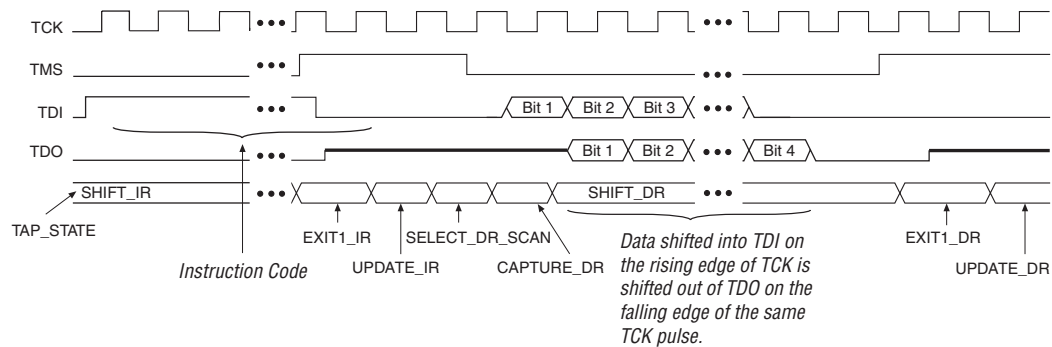
The EXTEST waveform diagram in Figure 14–11 resembles the SAMPLE/PRELOAD waveform diagram, except for the instruction code. The data shifted out of TDO consists of the data that was present in the capture registers after the capture phase. New test data shifted into the TDI pin appears at the TDO pin after being clocked through the entire boundary-scan register.

**Figure 14–11. EXTEST Shift Data Register Waveforms**



## BYPASS Instruction Mode

The BYPASS mode is activated when an instruction code of all 1's is loaded in the instruction register. The waveforms in Figure 14–12 show how scan data passes through a device once the TAP controller is in the SHIFT\_DR state. In this state, data signals are clocked into the bypass register from TDI on the rising edge of TCK and out of TDO on the falling edge of the same clock pulse.

**Figure 14–12. BYPASS Shift Data Register Waveforms**

### IDCODE Instruction Mode

The **IDCODE** instruction mode is used to identify the devices in an IEEE Std. 1149.1 chain. When **IDCODE** is selected, the device identification register is loaded with the 32-bit vendor-defined identification code. The device ID register is connected between the **TDI** and **TDO** ports, and the device **IDCODE** is shifted out. The **IDCODE** for Cyclone II devices are listed in the *Configuration & Testing* chapter in Volume 1 of the *Cyclone II Device Handbook*.

### USERCODE Instruction Mode

The **USERCODE** instruction mode is used to examine the user electronic signature (UES) within the devices along an IEEE Std. 1149.1 chain. When this instruction is selected, the device identification register is connected between the **TDI** and **TDO** ports. The user-defined UES is shifted into the device ID register in parallel from the 32-bit **USERCODE** register. The UES is then shifted out through the device ID register. The UES value is not user defined until after the device has been configured. Before configuration, the UES value is set to the default value.

### CLAMP Instruction Mode

The **CLAMP** instruction mode is used to allow the boundary-scan register to determine the state of the signals driven from the pins. In **CLAMP** instruction mode, the bypass register is selected as the serial path between the **TDI** and **TDO** ports.



If you are testing the device after configuring it, the programmable weak pull-up resistor or the bus hold feature overrides the `CLAMP` value (the value stored in the update register of the boundary-scan cell) at the pin.

### HIGHZ Instruction Mode

The `HIGHZ` instruction mode is used to set all of the user I/O pins to an inactive drive state. These pins are tri-stated until a new JTAG instruction is executed. When this instruction is loaded into the instruction register, the bypass register is connected between the `TDI` and `TDO` ports.

If you are testing the device after configuring it, the programmable weak pull-up resistor or the bus hold feature overrides the `HIGHZ` value at the pin.

### I/O Voltage Support in JTAG Chain

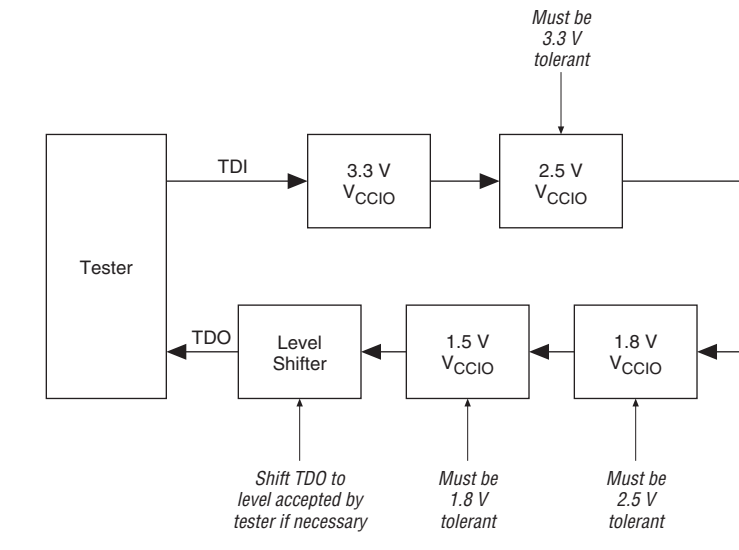
A JTAG chain can contain several different devices. However, you should be cautious if the chain contains devices that have different  $V_{CCIO}$  levels. The output voltage level of the `TDO` pin must meet the specifications of the `TDI` pin it drives. For Cyclone II devices, the `TDO` pin is powered by the  $V_{CCIO}$  power supply. Since the  $V_{CCIO}$  supply is 3.3 V, the `TDO` pin drives out 3.3 V.

Devices can interface with each other although they might have different  $V_{CCIO}$  levels. For example, a device with a 3.3-V `TDO` pin can drive to a device with a 5.0-V `TDI` pin because 3.3 V meets the minimum TTL-level  $V_{IH}$  for the 5.0-V `TDI` pin. JTAG pins on Cyclone II devices can support 2.5- or 3.3-V input levels.



For more information on MultiVolt I/O support, see the *Cyclone II Architecture* chapter in Volume 1 of the *Cyclone II Device Handbook*.

You can also interface the `TDI` and `TDO` lines of the devices that have different  $V_{CCIO}$  levels by inserting a level shifter between the devices. If possible, the JTAG chain should be built such that a device with a higher  $V_{CCIO}$  level drives to a device with an equal or lower  $V_{CCIO}$  level. This way, a level shifter may be required only to shift the `TDO` level to a level acceptable to the JTAG tester. [Figure 14–13](#) shows the JTAG chain of mixed voltages and how a level shifter is inserted in the chain.

**Figure 14–13. JTAG Chain of Mixed Voltages**

## Using IEEE Std. 1149.1 BST Circuitry

Cyclone II devices have dedicated JTAG pins, and the IEEE Std. 1149.1 BST circuitry is enabled upon device power-up. You can perform BST on Cyclone II FPGAs not only before and after configuration, but also during configuration. Cyclone II FPGAs support the **BYPASS**, **IDCODE**, and **SAMPLE** instructions during configuration without interrupting configuration. To send all other JTAG instructions, you must interrupt configuration using the **CONFIG\_IO** instruction.

The **CONFIG\_IO** instruction allows you to configure I/O buffers via the JTAG port, and when issued, interrupts configuration. This instruction allows you to perform board-level testing prior to configuring the Cyclone II FPGA or waiting for a configuration device to complete configuration. Once configuration has been interrupted and JTAG BST is complete, the part must be reconfigured via JTAG (**PULSE\_CONFIG** instruction) or by pulsing **nCONFIG** low.

When you perform JTAG boundary-scan testing before configuration, the **nCONFIG** pin must be held low.

The device-wide reset (**DEV\_CLRn**) and device-wide output enable (**DEV\_OE**) pins on Cyclone II devices do not affect JTAG boundary-scan or configuration operations. Toggling these pins does not disrupt BST operation any more than usual.

When designing a board for JTAG configuration of Cyclone II devices, the connections for the dedicated configuration pins need to be considered.



For more information on using the IEEE Std.1149.1 circuitry for device configuration, see the *Configuring Cyclone II Devices* chapter in Volume 1 of the *Cyclone II Device Handbook*.

## BST for Configured Devices

For a configured device, the input buffers are turned off by default for I/O pins that are set as output only in the design file. Nevertheless, executing the SAMPLE instruction will turn on the input buffers for the output pins. You can set the Quartus II software to always enable the input buffers on a configured device so it behaves the same as an unconfigured device for boundary-scan testing, allowing sample function on output pins in the design. This aspect can cause slight increase in standby current because the unused input buffer is always on. In the Quartus II software, do the following:

1. Choose **Settings** (Assignment menu).
2. Click **Assembler**.
3. Turn on **Always Enable Input Buffers**.
4. If you use the default setting with input disabled, you need to convert the default BSDL file to the design-specific BSDL file using the BSDLCustomizer script. For more information regarding BSDL file, refer to ["Boundary-Scan Description Language \(BSDL\) Support"](#).

## Disabling IEEE Std. 1149.1 BST Circuitry

The IEEE Std. 1149.1 BST circuitry for Cyclone II devices is enabled upon device power-up. Because this circuitry may be used for BST or in-circuit reconfiguration, this circuitry must be enabled only at specific times as mentioned in “Using IEEE Std. 1149.1 BST Circuitry” on page 14–16.

If the IEEE Std. 1149.1 circuitry will not be utilized at any time, the circuitry should be permanently disabled. Table 14–3 shows the pin connections necessary for disabling the IEEE Std. 1149.1 circuitry in Cyclone II devices to ensure that the circuitry is not inadvertently enabled when it is not needed.

**Table 14–3. Disabling IEEE Std. 1149.1 Circuitry**

JTAG Pins (1)	Connection for Disabling
TMS	V <sub>CC</sub>
TCK	GND
TDI	V <sub>CC</sub>
TDO	Leave open

**Note to Table 14–3:**

- (1) There is no software option to disable JTAG in Cyclone II devices. The JTAG pins are dedicated.

## Guidelines for IEEE Std. 1149.1 Boundary-Scan Testing

Use the following guidelines when performing boundary-scan testing with IEEE Std. 1149.1 devices:

- If the 10-bit checkerboard pattern “1010101010” does not shift out of the instruction register via the TDO pin during the first clock cycle of the SHIFT\_IR state, the TAP controller has not reached the proper state. To solve this problem, try one of the following procedures:
  - Verify that the TAP controller has reached the SHIFT\_IR state correctly. To advance the TAP controller to the SHIFT\_IR state, return to the RESET state and send the code 01100 to the TMS pin.
  - Check the connections to the V<sub>CC</sub>, GND, JTAG, and dedicated configuration pins on the device.

- Perform a `SAMPLE/PRELOAD` test cycle prior to the first `EXTEST` test cycle to ensure that known data is present at the device pins when the `EXTEST` mode is entered. If the `OEJ` update register contains a 0, the data in the `OUTJ` update register is driven out. The state must be known and correct to avoid contention with other devices in the system.
- Do not perform `EXTEST` testing during `ICR`. This instruction is supported before or after `ICR`, but not during `ICR`. Use the `CONFIG_IO` instruction to interrupt configuration, then perform testing, or wait for configuration to complete.
- If performing testing before configuration, hold the `nCONFIG` pin low.
- After configuration, any pins in a differential pin pair cannot be tested. Therefore, performing BST after configuration requires editing BSC group definitions that correspond to these differential pin pairs. The BSC group should be redefined as an internal cell. See the BSDL file for more information on editing.

For more information on boundary scan testing, contact Altera Applications.

## Boundary-Scan Description Language (BSDL) Support

The Boundary-Scan Description Language (BSDL), a subset of VHDL, provides a syntax that allows you to describe the features of an IEEE Std. 1149.1 BST-capable device that can be tested. Test software development systems then use the BSDL files for test generation, analysis, and failure diagnostics. For more information, or to receive BSDL files for IEEE Std. 1149.1-compliant Cyclone II devices, visit the Altera web site at [www.altera.com](http://www.altera.com).

## Conclusion

The IEEE Std. 1149.1 BST circuitry available in Cyclone II devices provides a cost-effective and efficient way to test systems that contain devices with tight lead spacing. Circuit boards with Altera and other IEEE Std. 1149.1-compliant devices can use the `EXTEST`, `SAMPLE/PRELOAD`, `BYPASS`, `IDCODE`, `USERCODE`, `CLAMP`, and `HIGHZ` modes to create serial patterns that internally test the pin connections between devices and check device operation.

## References

Bleeker, H., P. van den Eijnden, and F. de Jong. *Boundary-Scan Test: A Practical Approach*. Eindhoven, The Netherlands: Kluwer Academic Publishers, 1993.

Institute of Electrical and Electronics Engineers, Inc. *IEEE Standard Test Access Port and Boundary-Scan Architecture* (IEEE Std 1149.1-2001). New York: Institute of Electrical and Electronics Engineers, Inc., 2001.

## Document Revision History

---

Maunder, C. M., and R. E. Tulloss. *The Test Access Port and Boundary-Scan Architecture*. Los Alamitos: IEEE Computer Society Press, 1990.

## Document Revision History

Table 14–4 shows the revision history for this document.

Table 14–4. Document Revision History		
Date & Document Version	Changes Made	Summary of Changes
February 2007 v2.1	<ul style="list-style-type: none"><li>Added document revision history.</li><li>Added new section “BST for Configured Devices”.</li></ul>	<ul style="list-style-type: none"><li>Added information about ‘Always Enable Input Buffer’ option.</li></ul>
July 2005 v2.0	Moved the “JTAG Timing Specifications” section to the <i>DC Characteristics &amp; Timing Specifications</i> chapter.	
June 2004 v1.0	Added document to the Cyclone II Device Handbook.	