

Introduction

You can configure Stratix® and Stratix GX devices using one of several configuration schemes. All configuration schemes use either a microprocessor, configuration device, or a download cable. See [Table 11-1](#).

Table 11-1. Stratix & Stratix GX Device Configuration Schemes

Configuration Scheme	Typical Use
Fast passive parallel (FPP)	Configuration with a parallel synchronous configuration device or microprocessor interface where eight bits of configuration data are loaded on every clock cycle.
Passive serial (PS)	Configuration with a serial synchronous microprocessor interface or the MasterBlaster™ communications cable, USB Blaster, ByteBlaster™ II, or ByteBlasterMV parallel port download cable.
Passive parallel asynchronous (PPA)	Configuration with a parallel asynchronous microprocessor interface. In this scheme, the microprocessor treats the target device as memory.
Remote/local update FPP	Configuration using a Nios™ (16-bit ISA) and Nios® II (32-bit ISA) or other embedded processor. Allows you to update the Stratix or Stratix GX device configuration remotely using the FPP scheme to load data.
Remote/local update PS	Passive serial synchronous configuration using a Nios or other embedded processor. Allows you to update the Stratix or Stratix GX device configuration remotely using the PS scheme to load data.
Remote/local update PPA	Passive parallel asynchronous configuration using a Nios or other embedded processor. In this scheme, the Nios microprocessor treats the target device as memory. Allows you to update the Stratix or Stratix GX device configuration remotely using the PPA scheme to load data.
Joint Test Action Group (JTAG)	Configuration through the IEEE Std. 1149.1 JTAG pins. You can perform JTAG configuration with either a download cable or an embedded device. Ability to use SignalTap® II Embedded Logic Analyzer.

This chapter discusses how to configure one or more Stratix or Stratix GX devices. It should be used together with the following documents:

- *MasterBlaster Serial/USB Communications Cable Data Sheet*
- *USB Blaster USB Port Download Cable Development Tools Data Sheet*
- *ByteBlaster II Parallel Port Download Cable Data Sheet*
- *ByteBlasterMV Parallel Port Download Cable Data Sheets*
- *Configuration Devices for SRAM-Based LUT Devices Data Sheet*
- *Enhanced Configuration Devices (EPC4, EPC8, & EPC16) Data Sheet*

- The *Remote System Configuration with Stratix & Stratix GX Devices* chapter

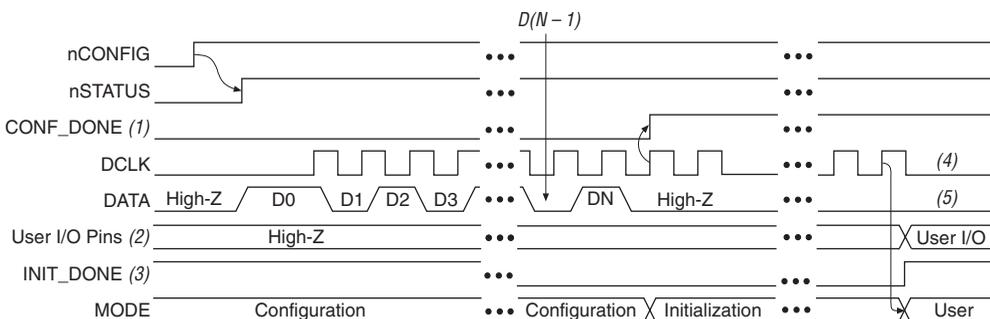


For more information on setting device configuration options or generating configuration files, see the *Software Setting* chapter in *Volume 2 of the Configuration Handbook*.

Device Configuration Overview

During device operation, the FPGA stores configuration data in SRAM cells. Because SRAM memory is volatile, you must load the SRAM cells with the configuration data each time the device powers up. After configuration, the device must initialize its registers and I/O pins. After initialization, the device enters user mode. [Figure 11–1](#) shows the state of the device during the configuration, initialization, and user mode.

Figure 11–1. Stratix & Stratix GX Configuration Cycle



Notes to [Figure 11–1](#):

- (1) During initial power up and configuration, CONF_DONE is low. After configuration, CONF_DONE goes high. If the device is reconfigured, CONF_DONE goes low after nCONFIG is driven low.
- (2) User I/O pins are tri-stated during configuration. Stratix and Stratix GX devices also have a weak pull-up resistor on I/O pins during configuration that are enabled by nIO_PULLUP. After initialization, the user I/O pins perform the function assigned in the user's design.
- (3) If the INIT_DONE pin is used, it will be high because of an external 10 kΩ resistor pull-up when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low.
- (4) DCLK should not be left floating. It should be driven high or low.
- (5) DATA0 should not be left floating. It should be driven high or low.

You can load the configuration data for the Stratix or Stratix GX device using a passive configuration scheme. When using any passive configuration scheme, the Stratix or Stratix GX device is incorporated into a system with an intelligent host, such as a microprocessor, that controls the configuration process. The host supplies configuration data from a storage device (e.g., a hard disk, RAM, or other system memory). When using passive configuration, you can change the target device's

functionality while the system is in operation by reconfiguring the device. You can also perform in-field upgrades by distributing a new programming file to system users.

The following sections describe the MSEL[2..0], VCCSEL, PORSEL, and nIO_PULLUP pins used in Stratix and Stratix GX device configuration.

MSEL[2..0] Pins

You can select a Stratix or Stratix GX device configuration scheme by driving its MSEL2, MSEL1, and MSEL0 pins either high or low, as shown in Table 11–2.

Description	MSEL2	MSEL1	MSEL0
FPP configuration	0	0	0
PPA configuration	0	0	1
PS configuration	0	1	0
Remote/local update FPP (1)	1	0	0
Remote/local update PPA (1)	1	0	1
Remote/local update PS (1)	1	1	0
JTAG-based configuration (3)	(2)	(2)	(2)

Notes to Table 11–2:

- (1) These schemes require that you drive a secondary pin `RUNLU` to specify whether to perform a remote update or local update.
- (2) Do not leave MSEL pins floating. Connect them to `VCCIO` or `GND`. These pins support the non-JTAG configuration scheme used in production. If only JTAG configuration is used you should connect the MSEL pins to ground.
- (3) JTAG-based configuration takes precedence over other configuration schemes, which means the MSEL pins are ignored.

The MSEL[] pins can be tied to `VCCIO` of the I/O bank they reside in or ground.

VCCSEL Pins

You can configure Stratix and Stratix GX devices using the 3.3-, 2.5-, 1.8-, or 1.5-V LVTTTL I/O standard on configuration and JTAG input pins. VCCSEL is a dedicated input on Stratix and Stratix GX devices that selects between 3.3-V/2.5-V input buffers and 1.8-V/1.5-V input buffers for dedicated configuration input pins. A logic low supports 3.3-V/2.5-V signaling, and a logic high supports 1.8-V/1.5-V signaling. A logic high can also support 3.3-V/2.5-V signaling. VCCSEL affects the configuration

related I/O banks (3, 4, 7, and 8) where the following pins reside: TDI, TMS, TCK, TRST, MSEL0, MSEL1, MSEL2, nCONFIG, nCE, DCLK, PLL_ENA, CONF_DONE, nSTATUS. The VCCSEL pin can be pulled to 1.5, 1.8, 2.5, or 3.3-V for a logic high level. There is an internal 2.5-k Ω pull-down resistor on VCCSEL. Therefore, if you are using a pull-up resistor to pull up this signal, you need to use a 1-k Ω resistor.

VCCSEL also sets the power-on-reset (POR) trip point for all the configuration related I/O banks (3, 4, 7, and 8), ensuring that these I/O banks have powered up to the appropriate voltage levels before configuration begins. Upon power-up, the FPGA does not release nSTATUS until V_{CCINT} and all of the V_{CCIO}s of the configuration I/O banks are above their POR trip points. If you set VCCSEL to ground (logic low), this sets the POR trip point for all configuration I/O banks to a voltage consistent with 3.3-V/2.5-V signaling. When VCCSEL = 0, the POR trip point for these I/O banks may be as high as 1.8 V. If V_{CCIO} of any of the configuration banks is set to 1.8 or 1.5 V, the voltage supplied to this I/O bank(s) may never reach the POR trip point, which will not allow the FPGA to begin configuration.

 If the V_{CCIO} of I/O banks 3, 4, 7, or 8 is set to 1.5 or 1.8 V and the configuration signals used require 3.3-V or 2.5-V signaling you should set VCCSEL to V_{CC} (logic high) in order to lower the POR trip point to enable successful configuration.

Table 11–3 shows how you should set the VCCSEL depending on the V_{CCIO} setting of the configuration I/O banks and your configuration input signaling voltages.

V _{CCIO} (banks 3,4,7,8)	Configuration Input Signaling Voltage	V _{CCSEL}
3.3-V/2.5-V	3.3-V/2.5-V	GND
1.8-V/1.5-V	3.3-V/2.5-V/1.8-V/1.5-V	VCC
3.3-V/2.5-V	1.8-V/1.5-V	Not Supported

The VCCSEL signal does not control any of the dual-purpose pins, including the dual-purpose configuration pins, such as the DATA [7 . . 0] and PPA pins (nWS, nRS, CS, nCS, and RDYnBSY). During configuration, these dual-purpose pins drive out voltage levels corresponding to the V_{CCIO} supply voltage that powers the I/O bank containing the pin. After configuration, the dual-purpose pins inherit the I/O standards specified in the design.

PORSEL Pins

PORSEL is a dedicated input pin used to select POR delay times of 2 ms or 100 ms during power-up. When the PORSEL pin is connected to ground, the POR time is 100 ms; when the PORSEL pin is connected to VCC, the POR time is 2 ms. There is an internal 2.5-k Ω pull-down resistor on PORSEL. Therefore if you are using a pull-up resistor to pull up this signal, you need to use a 1-k Ω resistor.

When using enhanced configuration devices to configure Stratix devices, make sure that the PORSEL setting of the Stratix device is the same or faster than the PORSEL setting of the enhanced configuration device. If the FPGA is not powered up after the enhanced configuration device exits POR, the CONF_DONE signal will be high since the pull-up resistor is pulling this signal high. When the enhanced configuration device exits POR, OE of the enhanced configuration device is released and pulled high by a pull-up resistor. Since the enhanced configuration device sees its nCS/CONF_DONE signal also high, it enters a test mode. Therefore, you must ensure the FPGA powers up before the enhanced configuration device exits POR.

For more margin, the 100-ms setting can be selected when using an enhanced configuration device to allow the Stratix FPGA to power-up before configuration is attempted (see [Table 11-4](#)).

PORSEL Settings	POR Time (ms)
GND	100
V _{CC}	2

nIO_PULLUP Pins

The nIO_PULLUP pin enables a built-in weak pull-up resistor to pull all user I/O pins to VCCIO before and during device configuration. If nIO_PULLUP is connected to VCC during configuration, the weak pull-ups on all user I/O pins and all dual-purpose pins are disabled. If connected to ground, the pull-ups are enabled during configuration. The nIO_PULLUP pin can be pulled to 1.5, 1.8, 2.5, or 3.3-V for a logic level high. There is an internal 2.5-k Ω pull-down resistor on nIO_PULLUP. Therefore, if you are using a pull-up resistor to pull up this signal, you need to use a 1-k Ω resistor.

TDO & nCEO Pins

TDO and nCEO pins drive out the same voltage levels as the V_{CCIO} that powers the I/O bank where the pin resides. You must select the V_{CCIO} supply for the bank containing TDO accordingly. For example, when using the ByteBlasterMV cable, the V_{CCIO} for the bank containing TDO must be powered up at 3.3-V. The current strength for TDO is 12 mA.

Configuration File Size

Tables 11–5 and 11–6 summarize the approximate configuration file size required for each Stratix and Stratix GX device. To calculate the amount of storage space required for multi-device configurations, add the file size of each device together.

Table 11–5. Stratix Configuration File Sizes

Device	Raw Binary File (.rbf) Size (Bits)
EP1S10	3,534,640
EP1S20	5,904,832
EP1S25	7,894,144
EP1S30	10,379,368
EP1S40	12,389,632
EP1S60	17,543,968
EP1S80	23,834,032

Table 11–6. Stratix GX Configuration File Sizes

Device	Raw Binary File Size (Bits)
EP1SGX10C	3,579,928
EP1SGX10D	3,579,928
EP1SGX25C	7,951,248
EP1SGX25D	7,951,248
EP1SGX25F	7,951,248
EP1SGX40D	12,531,440
EP1SGX40G	12,531,440

You should only use the numbers in Tables 11–5 and 11–6 to estimate the file size before design compilation. The exact file size may vary because different Altera® Quartus® II software versions may add a slightly

different number of padding bits during programming. However, for any specific version of the Quartus II software, any design targeted for the same device has the same configuration file size.

Altera Configuration Devices

The Altera enhanced configuration devices (EPC16, EPC8, and EPC4 devices) support a single-device configuration solution for high-density FPGAs and can be used in the FPP and PS configuration schemes. They are ISP-capable through its JTAG interface. The enhanced configuration devices are divided into two major blocks, the controller and the flash memory.



For information on enhanced configuration devices, see the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* and the *Using Altera Enhanced Configuration Devices* chapter in the *Configuration Handbook*.

The EPC2 and EPC1 configuration devices provide configuration support for the PS configuration scheme. The EPC2 device is ISP-capable through its JTAG interface. The EPC2 and EPC1 can be cascaded to hold large configuration files.



For more information on EPC2, EPC1, and EPC1441 configuration devices, see the *Configuration Devices for SRAM-Based LUT Devices Data Sheet*.

Configuration Schemes

This section describes how to configure Stratix and Stratix GX devices with the following configuration schemes:

- PS Configuration with Configuration Devices
- PS Configuration with a Download Cable
- PS Configuration with a Microprocessor
- FPP Configuration
- PPA Configuration
- JTAG Programming & Configuration
- JTAG Programming & Configuration of Multiple Devices

PS Configuration

PS configuration of Stratix and Stratix GX devices can be performed using an intelligent host, such as a MAX[®] device, microprocessor with flash memory, an Altera configuration device, or a download cable. In the PS scheme, an external host (MAX device, embedded processor, configuration device, or host PC) controls configuration. Configuration data is clocked into the target Stratix devices via the DATA0 pin at each rising edge of DCLK.

PS Configuration with Configuration Devices

The configuration device scheme uses an Altera configuration device to supply data to the Stratix or Stratix GX device in a serial bitstream (see [Figure 11-3](#)).

In the configuration device scheme, `nCONFIG` is usually tied to `VCC` (when using EPC16, EPC8, EPC4, or EPC2 devices, `nCONFIG` may be connected to `nINIT_CONF`). Upon device power-up, the target Stratix or Stratix GX device senses the low-to-high transition on `nCONFIG` and initiates configuration. The target device then drives the open-drain `CONF_DONE` pin low, which in-turn drives the configuration device's `nCS` pin low. When exiting power-on reset (POR), both the target and configuration device release the open-drain `nSTATUS` pin.

Before configuration begins, the configuration device goes through a POR delay of up to 200 ms to allow the power supply to stabilize (power the Stratix or Stratix GX device before or during the POR time of the configuration device). This POR delay has a maximum of 200 ms for EPC2 devices. For enhanced configuration devices, you can select between 2 ms and 100 ms by connecting `PORSEL` pin to `VCC` or `GND`, accordingly. During this time, the configuration device drives 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 release `nSTATUS`, which is then pulled high by a pull-up resistor.

When configuring multiple devices, configuration does not begin until all devices release their `OE` or `nSTATUS` pins. When all devices are ready, the configuration device clocks data out serially to the target devices using an internal oscillator.

After successful configuration, the Stratix FPGA starts initialization using the 10-MHz internal oscillator as the reference clock. After initialization, this internal oscillator is turned off. The `CONF_DONE` pin is released by the target device and then pulled high by a pull-up resistor. When initialization is complete, the FPGA enters user mode. The `CONF_DONE` pin must have an external 10-k Ω pull-up resistor in order for the device to initialize.

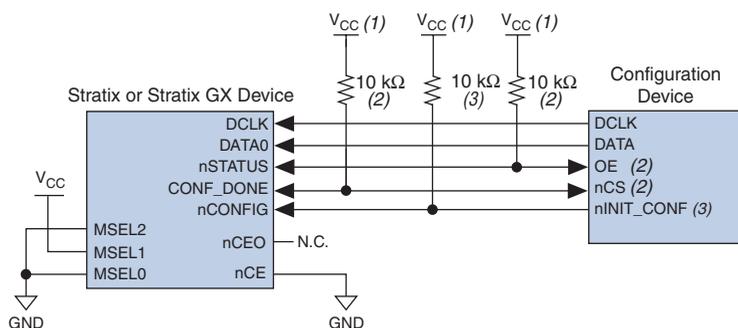
If an error occurs during configuration, the target device drives its `nSTATUS` pin low, resetting itself internally and resetting the configuration device. If the **Auto-Restart Configuration on Frame Error** option—available in the Quartus II **Global Device Options** dialog box (Assign menu)—is turned on, the device reconfigures automatically if an error occurs. To find this option, choose **Compiler Settings** (Processing menu), then click on the **Chips & Devices** tab.

If this option is turned off, the external system must monitor `nSTATUS` for errors and then pulse `nCONFIG` low to restart configuration. The external system can pulse `nCONFIG` if it is under system control rather than tied to V_{CC} . When configuration is complete, the target device releases `CONF_DONE`, which disables the configuration device by driving `nCS` high. The configuration device drives `DCLK` low before and after configuration.

In addition, if the configuration device sends all of its data and then detects that `CONF_DONE` has not gone high, it recognizes that the target device has not configured successfully. In this case, the configuration device pulses its `OE` pin low for a few microseconds, driving the target device's `nSTATUS` pin low. If the **Auto-Restart Configuration on Frame Error** option is set in the software, the target device resets and then pulses its `nSTATUS` pin low. When `nSTATUS` returns high, the configuration device reconfigures the target device. When configuration is complete, the configuration device drives `DCLK` low.

Do not pull `CONF_DONE` low to delay initialization. Instead, use the Quartus II software's **Enable User-Supplied Start-Up Clock (CLKUSR)** option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain initialize together. When `CONF_DONE` is driven low after device configuration, the configuration device recognizes that the target device has not configured successfully.

Figure 11–2 shows how to configure one Stratix or Stratix GX device with one configuration device.

Figure 11–2. Single Device Configuration Circuit

Notes to Figure 11–2:

- (1) The pull-up resistor should be connected to the same supply voltage as the configuration device.
- (2) The enhanced configuration devices and EPC2 devices have internal programmable pull-ups on OE and nCS. You should only use the internal pull-ups of the configuration device if the nSTATUS and CONF_DONE signals are pulled up to 3.3 V or 2.5 V (not 1.8 V or 1.5 V). If external pull-ups are used, they should be 10 kΩ.
- (3) The nINIT_CONF pin is available on EPC16, EPC8, EPC4, and EPC2 devices. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} through a resistor. The nINIT_CONF pin has an internal pull-up resistor that is always active in EPC16, EPC8, EPC4, and EPC2 devices. These devices do not need an external pull-up resistor on the nINIT_CONF pin.

Figure 11–3 shows how to configure multiple Stratix and Stratix GX devices with multiple EPC2 or EPC1 configuration devices.

Restart Configuration on Frame Error option is not turned on, the Stratix or Stratix GX devices drive `nSTATUS` low until they are reset with a low pulse on `nCONFIG`.

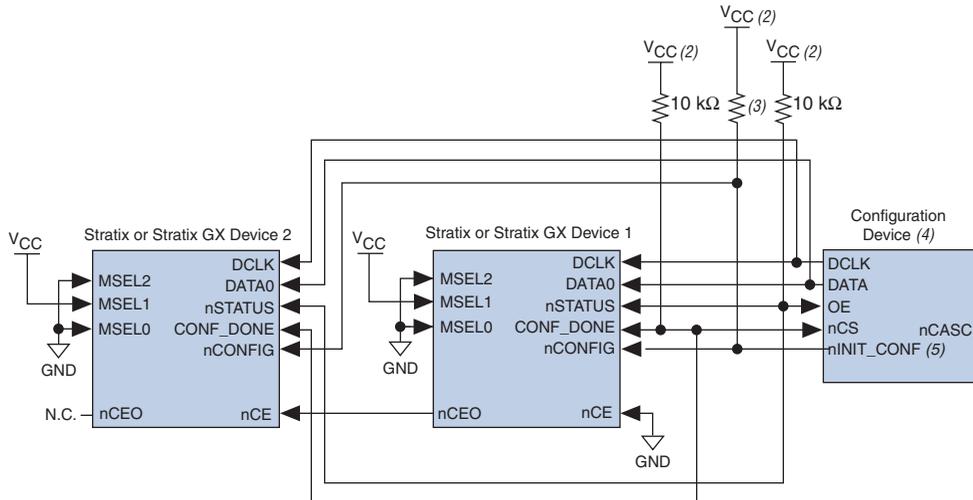
You can also cascade several EPC2/EPC1 configuration devices to configure multiple Stratix and Stratix GX devices. When all data from the first configuration device is sent, it drives `nCASC` low, which in turn drives `nCS` on the subsequent configuration device. Because a configuration device requires less than one clock cycle to activate a subsequent configuration device, the data stream is uninterrupted.



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

You can use a single configuration chain to configure multiple Stratix and Stratix GX devices. In this scheme, the `nCEO` pin of the first device is connected to the `nCE` pin of the second device in the chain. If there are additional devices, connect the `nCE` pin of the next device to the `nCEO` pin of the previous device. To configure properly, all of the device `CONF_DONE` and `nSTATUS` pins must be tied together.

Figure 11–4 shows an example of configuring multiple Stratix and Stratix GX devices using a configuration device.

Figure 11–4. Configuring Multiple Stratix & Stratix GX Devices with A Single Configuration Device *Note (1)***Notes to Figure 11–4:**

- (1) When performing multi-device active serial configuration, you must generate the configuration device programmer object file (**.pof**) from each project's SOF. You can combine multiple SOFs using the Quartus II software through the **Device & Pin Option** dialog box. For more information on how to create configuration and programming files, see the *Software Settings* section in the *Configuration Handbook, Volume 2*.
- (2) The pull-up resistor should be connected to the same supply voltage as the configuration device.
- (3) The enhanced configuration devices and EPC2 devices have internal programmable pull-ups on OE and nCS. You should only use the internal pull-ups of the configuration device if the nSTATUS and CONF_DONE signals are pulled up to 3.3 V or 2.5 V (not 1.8 V or 1.5 V). If external pull-ups are used, they should be 10 k Ω .
- (4) EPC16, EPC8, and EPC4 configuration devices cannot be cascaded.
- (5) The nINIT_CONF pin is available on EPC16, EPC8, EPC4, and EPC2 devices. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} through a resistor. The nINIT_CONF pin has an internal pull-up resistor that is always active in EPC16, EPC8, EPC4, and EPC2 devices. These devices do not need an external pull-up resistor on the nINIT_CONF pin.

Table 11–7 shows the status of the device DATA pins during and after configuration.

Table 11–7. DATA Pin Status Before & After Configuration		
Pins	Stratix or Stratix GX Device	
	During	After
DATA0 (1)	Used for configuration	User defined
DATA [7 . . 1] (2)	Used in some configuration modes	User defined
I/O Pins	Tri-state	User defined

Notes to Table 11–7:

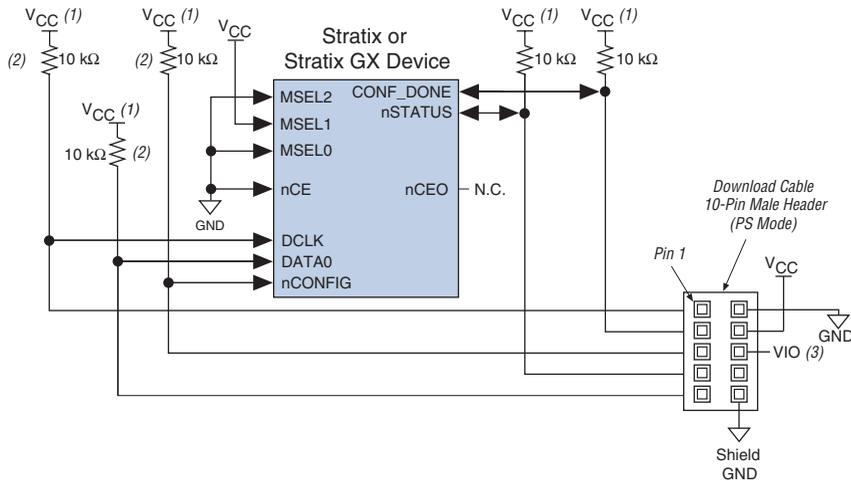
- (1) The status shown is for configuration with a configuration device.
- (2) The function of these pins depends upon the settings specified in the Quartus II software using the **Device & Pin Option** dialog box (see the *Software Settings* section in the *Configuration Handbook, Volume 2*, and the Quartus II Help software for more information).

PS Configuration with a Download Cable

In PS configuration with a download cable, an intelligent host transfers data from a storage device to the Stratix or Stratix GX device through the MasterBlaster, USB-Blaster, ByteBlaster II or ByteBlasterMV cable. To initiate configuration in this scheme, the download cable generates a low-to-high transition on the nCONFIG pin. The programming hardware then places the configuration data one bit at a time on the device's DATA0 pin. The data is clocked into the target device until CONF_DONE goes high. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

When using programming hardware for the Stratix or Stratix GX device, turning on the **Auto-Restart Configuration on Frame Error** option does not affect the configuration cycle because the Quartus II software must restart configuration when an error occurs. Additionally, the **Enable User-Supplied Start-Up Clock (CLKUSR)** option has no effect on the device initialization since this option is disabled in the SOF when programming the FPGA using the Quartus II software programmer and a 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.

Figure 11–5 shows PS configuration for the Stratix or Stratix GX device using a MasterBlaster, USB-Blaster, ByteBlaster II or ByteBlasterMV cable.

Figure 11–5. PS Configuration Circuit with a Download Cable**Notes to Figure 11–5:**

- (1) You should connect the pull-up resistor to the same supply voltage as the MasterBlaster (V_{IO} pin) or ByteBlasterMV cable.
- (2) The pull-up resistors on the DATA0 and DCLK pins are only needed if the download cable is the only configuration scheme used on the board. This is to ensure that the DATA0 and DCLK pins are not left floating after configuration. For example, if the design also uses a configuration device, the pull-up resistors on the DATA0 and DCLK pins are not necessary.
- (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}. This pin is a no-connect pin for the ByteBlasterMV header.

You can use programming hardware to configure multiple Stratix and Stratix GX devices by connecting each device's nCEO pin to the subsequent device's nCE pin. All other configuration pins are connected to each device in the chain.

Because all CONF_DONE pins are tied together, all devices in the chain initialize and enter user mode at the same time. In addition, because the nSTATUS pins are tied together, the entire chain halts configuration if any device detects an error. In this situation, the Quartus II software must restart configuration; the **Auto-Restart Configuration on Frame Error** option does not affect the configuration cycle.

Figure 11–6 shows how to configure multiple Stratix and Stratix GX devices with a MasterBlaster or ByteBlasterMV cable.

PS Configuration with a Microprocessor

In PS configuration with a microprocessor, a microprocessor transfers data from a storage device to the target Stratix or Stratix GX device. To initiate configuration in this scheme, the microprocessor must generate a low-to-high transition on the `nCONFIG` pin and the target device must release `nSTATUS`. The microprocessor or programming hardware then places the configuration data one bit at a time on the `DATA0` pin of the Stratix or Stratix GX device. The least significant bit (LSB) of each data byte must be presented first. Data is clocked continuously into the target device until `CONF_DONE` goes high.

After all configuration data is sent to the Stratix or Stratix GX device, the `CONF_DONE` pin goes high to show successful configuration and the start of initialization. The `CONF_DONE` pin must have an external 10-k Ω pull-up resistor in order for the device to initialize. Initialization, by default, uses an internal oscillator, which runs at 10 MHz. After initialization, this internal oscillator is turned off. If you are using the `clkusr` option, after all data is transferred `clkusr` must be clocked an additional 136 times for the Stratix or Stratix GX device to initialize properly. Driving `DCLK` to the device after configuration is complete does not affect device operation.

Handshaking signals are not used in PS configuration modes. Therefore, the configuration clock speed must be below the specified frequency to ensure correct configuration. No maximum `DCLK` period exists. You can pause configuration by halting `DCLK` for an indefinite amount of time.

If the target device detects an error during configuration, it drives its `nSTATUS` pin low to alert the microprocessor. The microprocessor can then pulse `nCONFIG` low to restart the configuration process. Alternatively, if the **Auto-Restart Configuration on Frame Error** option is turned on in the Quartus II software, the target device releases `nSTATUS` after a reset time-out period. After `nSTATUS` is released, the microprocessor can reconfigure the target device without needing to pulse `nCONFIG` low.

The microprocessor can also monitor the `CONF_DONE` and `INIT_DONE` pins to ensure successful configuration. If the microprocessor sends all data and the initialization clock starts but `CONF_DONE` and `INIT_DONE` have not gone high, it must reconfigure the target device. By default the `INIT_DONE` output is disabled. You can enable the `INIT_DONE` output by turning on **Enable INIT_DONE output** option in the Quartus II software.

If you do not turn on the **Enable INIT_DONE output** option in the Quartus II software, you are advised to wait for the maximum value of `tCD2UM` (see [Table 11-8](#)) after the `CONF_DONE` signal goes high to ensure the device has been initialized properly and that it has entered user mode.

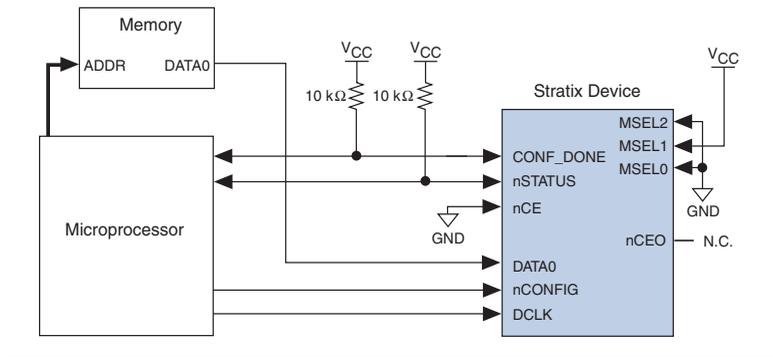
During configuration and initialization, and before the device enters user mode, the microprocessor must not drive the CONF_DONE signal low.



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

Figure 11–8 shows the circuit for PS configuration with a microprocessor.

Figure 11–8. PS Configuration Circuit with Microprocessor



PS Configuration Timing

Figure 11–9 shows the PS configuration timing waveform for Stratix and Stratix GX devices. Table 11–8 shows the PS timing parameters for Stratix and Stratix GX devices.

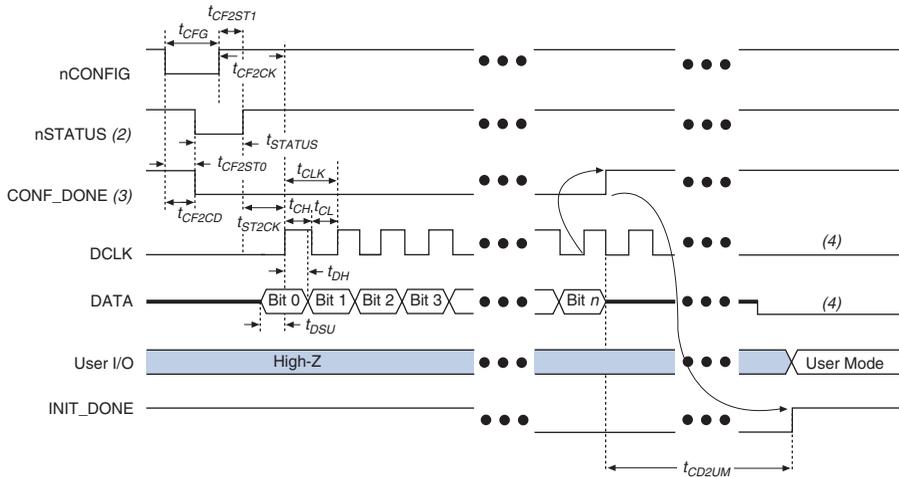
Table 11–8. PS Timing Parameters for Stratix & Stratix GX Devices

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns
t_{CF2ST1}	nCONFIG high to nSTATUS high		40 (2)	μ s
t_{CFG}	nCONFIG low pulse width	40		μ s
t_{STATUS}	nSTATUS low pulse width	10	40 (2)	μ s
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	40		μ s
t_{ST2CK}	nSTATUS high to first rising edge on DCLK	1		μ 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 maximum frequency		100	MHz
t_{CD2UM}	CONF_DONE high to user mode (1)	6	20	μ s

Notes to Table 11–8:

- (1) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSER, multiply the clock period by 136 to obtain this value.
- (2) This value is obtainable if users do not delay configuration by extending the nSTATUS low pulse width.

Figure 11–9. PS Timing Waveform for Stratix & Stratix GX Devices Note (1)

**Notes to Figure 11–9:**

- (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 Stratix 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) DCLK should not be left floating after configuration. It should be driven high or low, whichever is convenient. DATA [] is available as user I/Os after configuration and the state of these pins depends on the dual-purpose pin settings.

FPP Configuration

Parallel configuration of Stratix and Stratix GX devices meets the continuously increasing demand for faster configuration times. Stratix and Stratix GX devices can receive byte-wide configuration data per clock cycle, and guarantee a configuration time of less than 100 ms with a 100-MHz configuration clock. Stratix and Stratix GX devices support programming data bandwidth up to 800 megabits per second (Mbps) in this mode. You can use parallel configuration with an EPC16, EPC8, or EPC4 device, or a microprocessor.

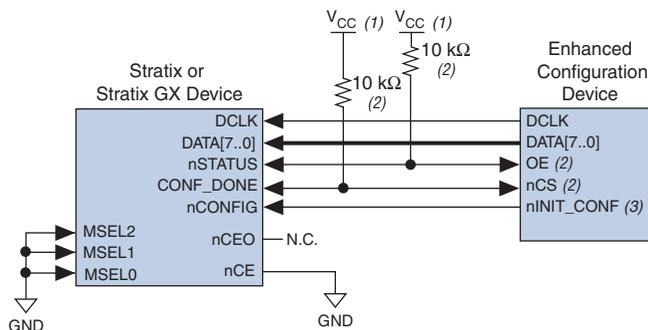
This section discusses the following schemes for FPP configuration in Stratix and Stratix GX devices:

- FPP Configuration Using an Enhanced Configuration Device
- FPP Configuration Using a Microprocessor

FPP Configuration Using an Enhanced Configuration Device

When using FPP with an enhanced configuration device, it supplies data in a byte-wide fashion to the Stratix or Stratix GX device every DCLK cycle. See [Figure 11–10](#).

Figure 11–10. FPP Configuration Using Enhanced Configuration Devices



Notes to [Figure 11–10](#):

- (1) The pull-up resistors should be connected to the same supply voltage as the configuration device.
- (2) The enhanced configuration devices and EPC2 devices have internal programmable pull-ups on OE and nCS. You should only use the internal pull-ups of the configuration device if the nSTATUS and CONF_DONE signals are pulled up to 3.3 V or 2.5 V (not 1.8 V or 1.5 V). If external pull-ups are used, they should be 10 kΩ.
- (3) The nINIT_CONF pin is available on EPC16, EPC8, EPC4, and EPC2 devices. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} through a resistor. The nINIT_CONF pin has an internal pull-up resistor that is always active in EPC16, EPC8, EPC4, and EPC2 devices. These devices do not need an external pull-up resistor on the nINIT_CONF pin.

In the enhanced configuration device scheme, nCONFIG is tied to nINIT_CONF. On power up, the target Stratix or Stratix GX device senses the low-to-high transition on nCONFIG and initiates configuration. The target Stratix or Stratix GX device then drives the open-drain CONF_DONE pin low, which in-turn drives the enhanced configuration device's nCS pin low.

Before configuration starts, there is a 2-ms POR delay if the PORSEL pin is connected to V_{CC} in the enhanced configuration device. If the PORSEL pin is connected to ground, the POR delay is 100 ms. When each device determines that its power is stable, it releases its nSTATUS or OE pin. Because the enhanced configuration device's OE pin is connected to the target Stratix or Stratix GX device's nSTATUS pin, configuration is delayed until both the nSTATUS and OE pins are released by each device. The nSTATUS and OE pins are pulled up by a resistor on their respective

devices once they are released. When configuring multiple devices, connect the `nSTATUS` pins together to ensure configuration only happens when all devices release their `OE` or `nSTATUS` pins. The enhanced configuration device then clocks data out in parallel to the Stratix or Stratix GX device using a 66-MHz internal oscillator, or drives it to the Stratix or Stratix GX device through the `EXTCLK` pin.

If there is an error during configuration, the Stratix or Stratix GX device drives the `nSTATUS` pin low, resetting itself internally and resetting the enhanced configuration device. The Quartus II software provides an **Auto-restart configuration after error** option that automatically initiates the reconfiguration whenever an error occurs. See the *Software Settings* chapter in Volume 2 of the *Configuration Handbook* for information on how to turn this option on or off.

If this option is turned off, you must set monitor `nSTATUS` to check for errors. To initiate reconfiguration, pulse `nCONFIG` low. The external system can pulse `nCONFIG` if it is under system control rather than tied to V_{CC} . Therefore, `nCONFIG` must be connected to `nINIT_CONF` if you want to reprogram the Stratix or Stratix GX device on the fly.

When configuration is complete, the Stratix or Stratix GX device releases the `CONF_DONE` pin, which is then pulled up by a resistor. This action disables the EPC16, EPC8, or EPC4 enhanced configuration device as `nCS` is driven high. Initialization, by default, uses an internal oscillator, which runs at 10 MHz. After initialization, this internal oscillator is turned off. When initialization is complete, the Stratix or Stratix GX device enters user mode. The enhanced configuration device drives `DCLK` low before and after configuration.



`CONF_DONE` goes high one byte early in parallel synchronous (FPP) and asynchronous (PPA) modes using a microprocessor with `.rbf`, `.hex`, and `.ttf` file formats. This does not apply to FPP mode for enhanced configuration devices using `.pof` file format. This also does not apply to serial modes.

If, after sending out all of its data, the enhanced configuration device does not detect `CONF_DONE` going high, it recognizes that the Stratix or Stratix GX device has not configured successfully. The enhanced configuration device pulses its `OE` pin low for a few microseconds, driving the `nSTATUS` pin on the Stratix or Stratix GX device low. If the **Auto-restart configuration after error** option is on, the Stratix or Stratix GX device resets and then pulses its `nSTATUS` low. When `nSTATUS` returns high, reconfiguration is restarted (see [Figure 11–11 on page 11–25](#)).

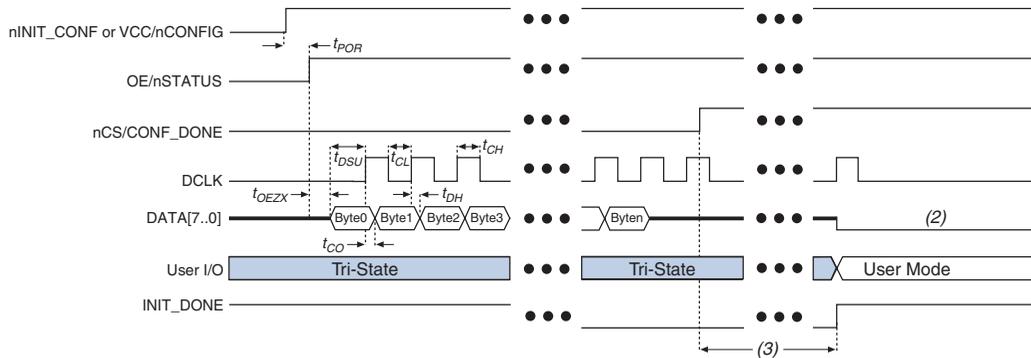
Do not drive CONF_DONE low after device configuration to delay initialization. Instead, use the **Enable User-Supplied Start-Up Clock (CLKUSR)** option in the **Device & Pin Options** dialog box. You can use this option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain initialize together.

After the first Stratix or Stratix GX device completes configuration during multi-device configuration, its nCEO pin activates the second Stratix or Stratix GX device's nCE pin, prompting the second device to begin configuration. Because CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time. Because nSTATUS pins are tied together, configuration stops for the whole chain if any device (including enhanced configuration devices) detects an error. Also, if the enhanced configuration device does not detect a high on CONF_DONE at the end of configuration, it pulses its OE low for a few microseconds to reset the chain. The low OE pulse drives nSTATUS low on all Stratix and Stratix GX devices, causing them to enter an error state. This state is similar to a Stratix or Stratix GX device detecting an error.

If the **Auto-restart configuration after error** option is on, the Stratix and Stratix GX devices release their nSTATUS pins after a reset time-out period. When the nSTATUS pins are released and pulled high, the configuration device reconfigures the chain. If the **Auto-restart configuration after error** option is off, nSTATUS stays low until the Stratix and Stratix GX devices are reset with a low pulse on nCONFIG.

Figure 11–11 shows the FPP configuration with a configuration device timing waveform for Stratix and Stratix GX devices.

Figure 11–11. FPP Configuration with a Configuration Device Timing Waveform Note (1)



Notes to Figure 11–11:

- (1) For timing information, see the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet*.
- (2) The configuration device drives DATA high after configuration.
- (3) Stratix and Stratix GX devices enter user mode 136 clock cycles after CONF_DONE goes high.

FPP Configuration Using a Microprocessor

When using a microprocessor for parallel configuration, the microprocessor transfers data from a storage device to the Stratix or Stratix GX device through configuration hardware. To initiate configuration, the microprocessor needs to generate a low-to-high transition on the nCONFIG pin and the Stratix or Stratix GX device must release nSTATUS. The microprocessor then places the configuration data to the DATA [7..0] pins of the Stratix or Stratix GX device. Data is clocked continuously into the Stratix or Stratix GX device until CONF_DONE goes high.

The configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists. You can pause configuration by halting DCLK for an indefinite amount of time.

After all configuration data is sent to the Stratix or Stratix GX device, the CONF_DONE pin goes high to show successful configuration and the start of initialization. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. Initialization, by default, uses an internal oscillator, which runs at 10 MHz. After initialization, this internal oscillator is turned off. If you are using the `clkusr` option, after all data is transferred `clkusr` must be clocked an additional 136 times for the Stratix or Stratix GX device to initialize properly. Driving DCLK to the device after configuration is complete does not affect device operation. 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 do not turn on the **Enable `INIT_DONE` output** option in the Quartus II software, you are advised to wait for maximum value of t_{CD2UM} (see [Table 11-9](#)) after the `CONF_DONE` signal goes high to ensure the device has been initialized properly and that it has entered user mode.

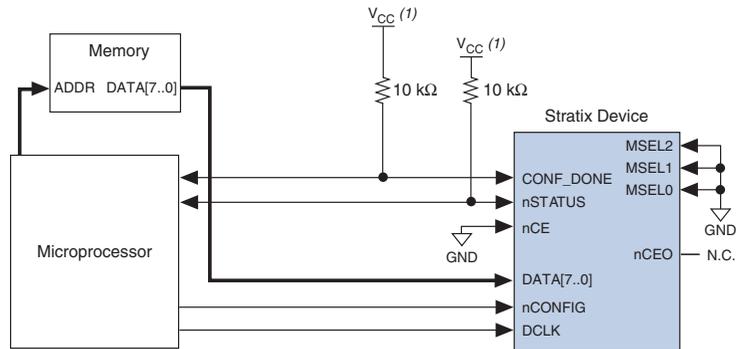
During configuration and initialization and before the device enters user mode, the microprocessor must not drive the `CONF_DONE` signal low.

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

If the Stratix or Stratix GX device detects an error during configuration, it drives `nSTATUS` low to alert the microprocessor. The pin on the microprocessor connected to `nSTATUS` must be an input. The microprocessor can then pulse `nCONFIG` low to restart the configuration error. With the **Auto-restart configuration after error** option on, the Stratix or Stratix GX device releases `nSTATUS` after a reset time-out period. After `nSTATUS` is released, the microprocessor can reconfigure the Stratix or Stratix GX device without pulsing `nCONFIG` low.

The microprocessor can also monitor the `CONF_DONE` and `INIT_DONE` pins to ensure successful configuration. If the microprocessor sends all the data and the initialization clock starts but `CONF_DONE` and `INIT_DONE` have not gone high, it must reconfigure the Stratix or Stratix GX device. After waiting the specified 136 `DCLK` cycles, the microprocessor should restart configuration by pulsing `nCONFIG` low.

[Figure 11-12](#) shows the circuit for Stratix and Stratix GX parallel configuration using a microprocessor.

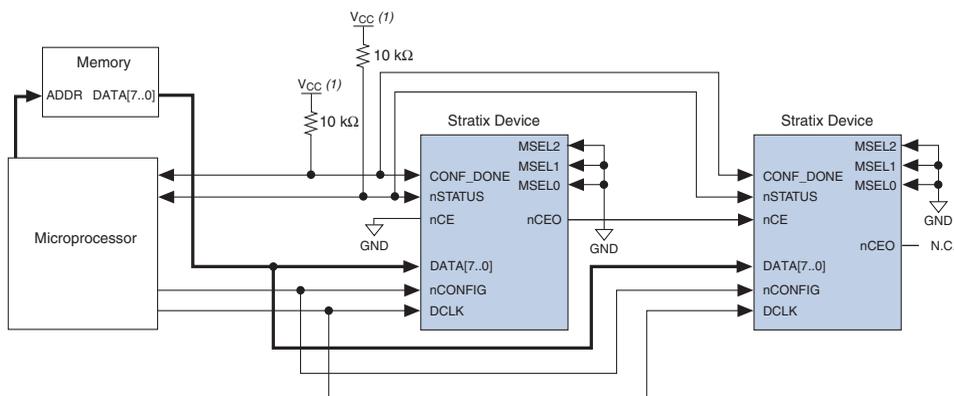
Figure 11–12. Parallel Configuration Using a Microprocessor**Note to Figure 11–12:**

- (1) The pull-up resistors should be connected to any V_{CC} that meets the Stratix high-level input voltage (V_{IH}) specification.

For multi-device parallel configuration with a microprocessor, the $nCEO$ pin of the first Stratix or Stratix GX device is cascaded to the second device's nCE pin. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the microprocessor. Because the $CONF_DONE$ pins of the devices are connected together, all devices initialize and enter user mode at the same time.

Because the $nSTATUS$ pins are also tied together, if any of the devices detects an error, the entire chain halts configuration and drives $nSTATUS$ low. The microprocessor can then pulse $nCONFIG$ low to restart configuration. If the **Auto-restart configuration after error** option is on, the Stratix and Stratix GX devices release $nSTATUS$ after a reset time-out period. The microprocessor can then reconfigure the devices once $nSTATUS$ is released. [Figure 11–13](#) shows multi-device configuration using a microprocessor. [Figure 11–14](#) shows multi-device configuration when both Stratix and Stratix GX devices are receiving the same data. In this case, the microprocessor sends the data to both devices simultaneously, and the devices configure simultaneously.

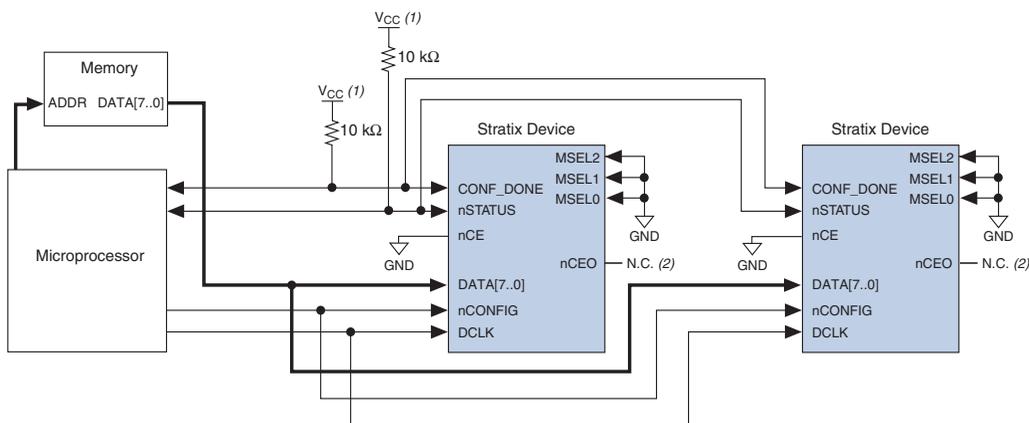
Figure 11–13. Parallel Data Transfer in Serial Configuration with a Microprocessor



Note to Figure 11–13:

- (1) You should connect the pull-up resistors to any V_{CC} that meets the Stratix high-level input voltage (V_{IH}) specification.

Figure 11–14. Multiple Device Parallel Configuration with the Same Data Using a Microprocessor



Notes to Figure 11–14:

- (1) You should connect the pull-up resistors to any V_{CC} that meets the Stratix high-level input voltage (V_{IH}) specification.
- (2) The nCEO pins are left unconnected when configuring the same data into multiple Stratix or Stratix GX devices.

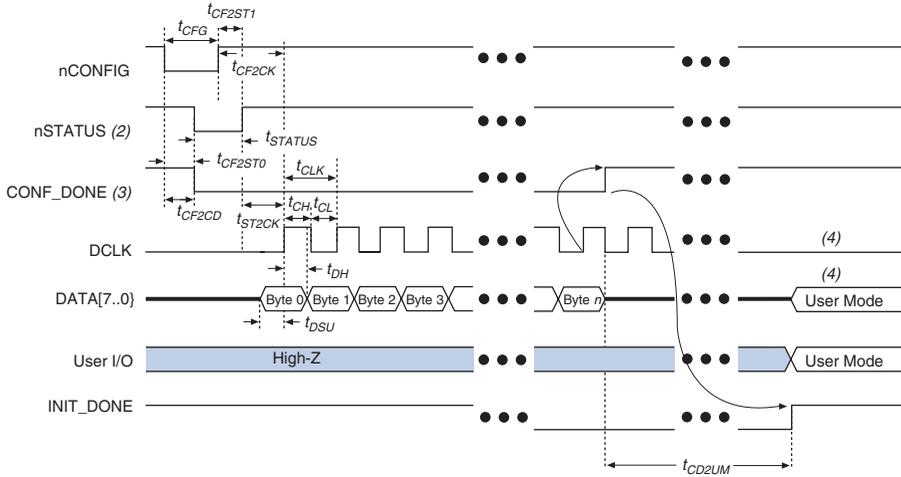


For more information on configuring multiple Altera devices in the same configuration chain, see the *Configuring Mixed Altera FPGA Chains* chapter in the *Configuration Handbook, Volume 2*.

FPP Configuration Timing

Figure 11-15 shows FPP timing waveforms for configuring a Stratix or Stratix GX device in FPP mode. Table 11-9 shows the FPP timing parameters for Stratix or Stratix GX devices.

Figure 11-15. Timing Waveform for Configuring Devices in FPP Mode Note (1)



Notes to Figure 11-15:

- (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 Stratix 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) **DCLK** should not be left floating after configuration. It should be driven high or low, whichever is convenient. **DATA []** is available as user I/Os after configuration and the state of these pins depends on the dual-purpose pin settings.

Symbol	Parameter	Min	Max	Units
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	40		μ 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_{CFG}	nCONFIG low pulse width	40		μ s
t_{CH}	DCLK high time	4		ns
t_{CL}	DCLK low time	4		ns
t_{CLK}	DCLK period	10		ns

Table 11–9. FPP Timing Parameters for Stratix & Stratix GX Devices (Part 2 of 2)

Symbol	Parameter	Min	Max	Units
f_{MAX}	DCLK frequency		100	MHz
t_{CD2UM}	CONF_DONE high to user mode (1)	6	20	μs
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns
t_{CF2ST1}	nCONFIG high to nSTATUS high		40 (2)	μs
t_{STATUS}	nSTATUS low pulse width	10	40 (2)	μs
t_{ST2CK}	nSTATUS high to firstrising edge of DCLK	1		μs

Notes to Table 11–9:

- (1) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 136 to obtain this value.
- (2) This value is obtainable if users do not delay configuration by extending the nSTATUS low pulse width.

PPA Configuration

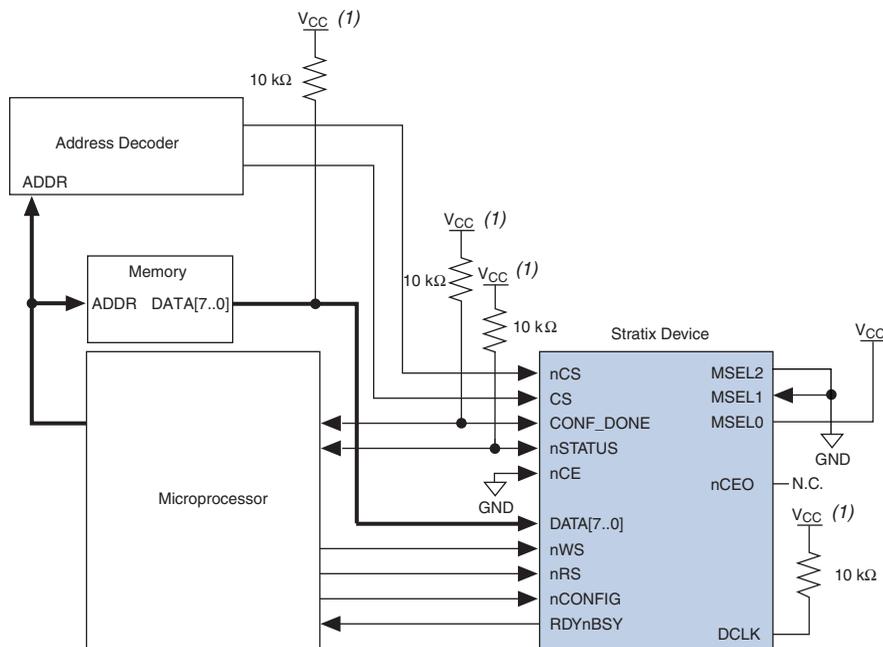
In PPA schemes, a microprocessor drives data to the Stratix or Stratix GX device through a download cable. When using a PPA scheme, use a 1-k Ω pull-up resistor to pull the DCLK pin high to prevent unused configuration pins from floating.

To begin configuration, the microprocessor drives nCONFIG high and then asserts the target device's nCS pin low and CS pin high. Next, the microprocessor places an 8-bit configuration word on the target device's data inputs and pulses nWS low. On the rising edge of nWS, the target device latches a byte of configuration data and then drives its RDYnBSY signal low, indicating that it is processing the byte of configuration data. The microprocessor then performs other system functions while the Stratix or Stratix GX device is processing the byte of configuration data.

Next, the microprocessor checks nSTATUS and CONF_DONE. If nSTATUS is high and CONF_DONE is low, the microprocessor sends the next data byte. If nSTATUS is low, the device is signaling an error and the microprocessor should restart configuration. However, if nSTATUS is high and all the configuration data is received, the device is ready for initialization. At the beginning of initialization, CONF_DONE goes high to indicate that configuration is complete. The CONF_DONE pin must have an external 10-k Ω pull-up resistor in order for the device to initialize. Initialization, by default, uses an internal oscillator, which runs at 10 MHz. After initialization, this internal oscillator is turned off. When initialization is complete, the Stratix or Stratix GX device enters user mode.

Figure 11-16 shows the PPA configuration circuit. An optional address decoder controls the device's nCS and CS pins. This decoder allows the microprocessor to select the Stratix or Stratix GX device by accessing a particular address, simplifying the configuration process.

Figure 11-16. PPA Configuration Circuit



Note to Figure 11-16:

(1) The pull-up resistor should be connected to the same supply voltage as the Stratix or Stratix GX device.

The device's nCS or CS pins can be toggled during PPA configuration if the design meets the specifications for t_{CSSU} , t_{WSP} and t_{CSH} given in Table 11-10 on page 11-36. The microprocessor can also directly control the nCS and CS signals. You can tie one of the nCS or CS signals to its active state (i.e., nCS may be tied low) and toggle the other signal to control configuration.

Stratix and Stratix GX devices can serialize data internally without the microprocessor. When the Stratix or Stratix GX device is ready for the next byte of configuration data, it drives RDYnBSY high. If the microprocessor senses a high signal when it polls RDYnBSY, the microprocessor strobes the next byte of configuration data into the device. Alternatively, the nRS signal can be strobed, causing the RDYnBSY signal to appear on DATA7. Because RDYnBSY does not need to

be monitored, reading the state of the configuration data by strobing nRS low saves a system I/O port. Do not drive data onto the data bus while nRS is low because it causes contention on $DATA7$. If the nRS pin is not used to monitor configuration, you should tie it high. To simplify configuration, the microprocessor can wait for the total time of $t_{BUSY}(\text{max}) + t_{RDY2WS} + t_{W2SB}$ before sending the next data bit.

After configuration, the nCS , CS , nRS , nWS , and $RDYnBSY$ pins act as user I/O pins. However, if the PPA scheme is chosen in the Quartus II software, these I/O pins are tri-stated by default in user mode and should be driven by the microprocessor. To change the default settings in the Quartus II software, select **Device & Pin Option** (Compiler Setting menu).

If the Stratix or Stratix GX device detects an error during configuration, it drives $nSTATUS$ low to alert the microprocessor. The microprocessor can then pulse $nCONFIG$ low to restart the configuration process.

Alternatively, if the **Auto-Restart Configuration on Frame Error** option is turned on, the Stratix or Stratix GX device releases $nSTATUS$ after a reset time-out period. After $nSTATUS$ is released, the microprocessor can reconfigure the Stratix or Stratix GX device. At this point, the microprocessor does not need to pulse $nCONFIG$ low.

The microprocessor can also monitor the $CONF_DONE$ and $INIT_DONE$ pins to ensure successful configuration. The microprocessor must monitor the $nSTATUS$ pin to detect errors and the $CONF_DONE$ pin to determine when programming completes ($CONF_DONE$ goes high one byte early in parallel mode). If the microprocessor sends all configuration data and starts initialization but $CONF_DONE$ is not asserted, the microprocessor must reconfigure the Stratix or Stratix GX device.

By default, the $INIT_DONE$ 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 do not turn on the **Enable $INIT_DONE$ output** option in the Quartus II software, you are advised to wait for the maximum value of t_{CD2UM} (see [Table 11-10](#)) after the $CONF_DONE$ signal goes high to ensure the device has been initialized properly and that it has entered user mode.

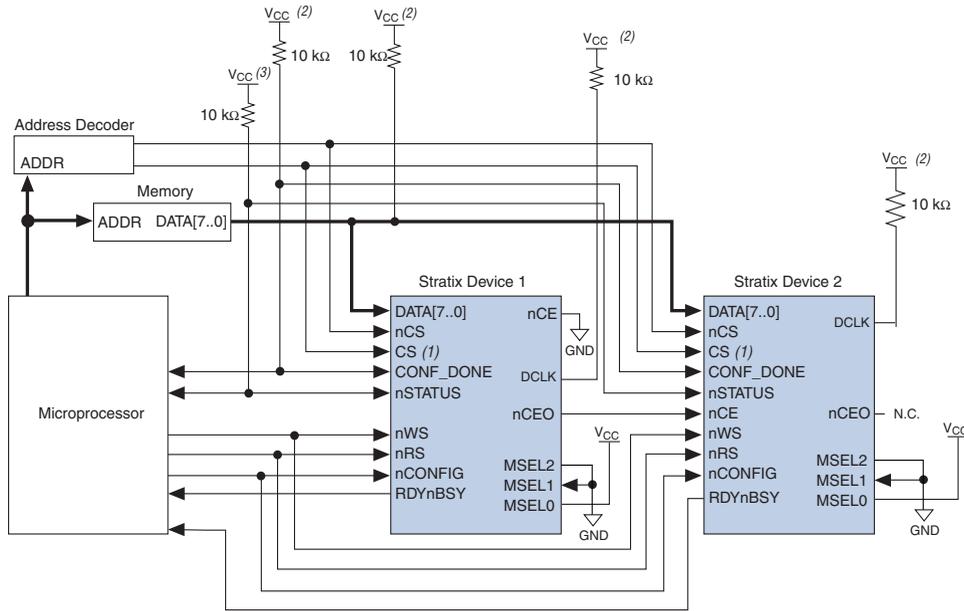
During configuration and initialization, and before the device enters user mode, the microprocessor must not drive the $CONF_DONE$ signal low.



If the optional $CLKUSR$ pin is 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 μs).

You can also use PPA mode to configure multiple Stratix and Stratix GX devices. Multi-device PPA configuration is similar to single-device PPA configuration, except that the Stratix and Stratix GX devices are cascaded. After you configure the first Stratix or Stratix GX device, $nCE0$ is asserted, which asserts the nCE pin on the second device, initiating configuration. Because the second Stratix or Stratix GX device begins configuration within one write cycle of the first device, the transfer of data destinations is transparent to the microprocessor. All Stratix and Stratix GX device $CONF_DONE$ pins are tied together; therefore, all devices initialize and enter user mode at the same time. See Figure 11-17.

Figure 11-17. PPA Multi-Device Configuration Circuit



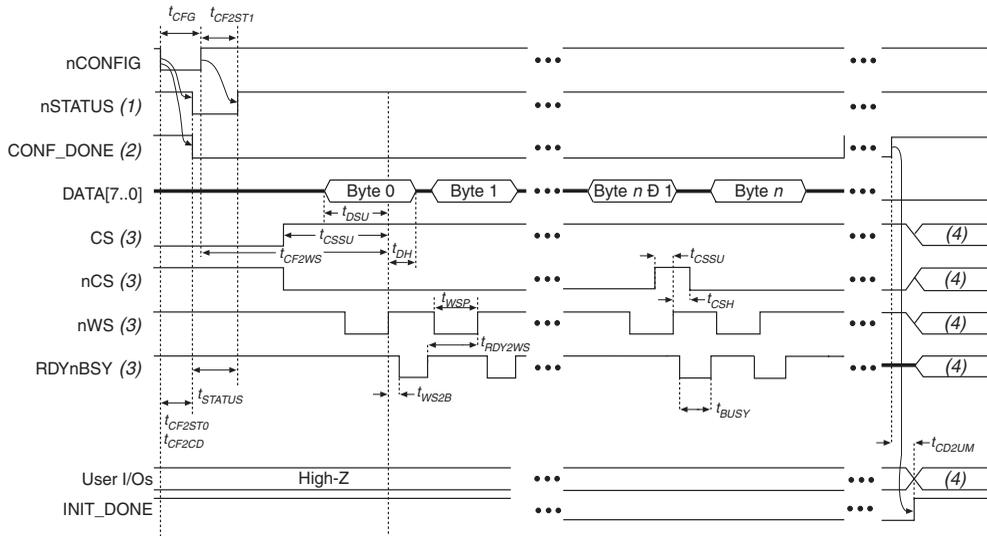
Notes to Figure 11-17:

- (1) If not used, you can connect the CS pin to V_{CC} directly. If not used, the nCS pin can be connected to GND directly.
- (2) Connect the pull-up resistor to the same supply voltage as the Stratix or Stratix GX device.

PPA Configuration Timing

Figure 11–18 shows the Stratix and Stratix GX device timing waveforms for PPA configuration.

Figure 11–18. PPA Timing Waveforms for Stratix & Stratix GX Devices

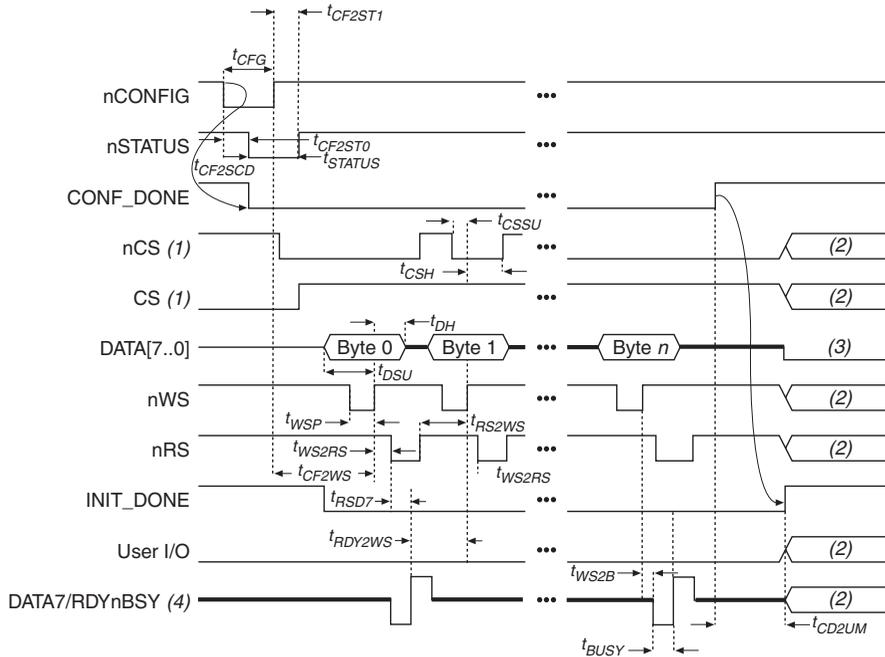


Notes to Figure 11–18:

- (1) Upon power-up, nSTATUS is held low for the time of the POR delay.
- (2) Upon power-up, before and during configuration, CONF_DONE is low.
- (3) After configuration, the state of CS, nCS, nWS, and RDYnBSY depends on the design programmed into the Stratix or Stratix GX device.
- (4) Device I/O pins are in user mode.

Figure 11–19 shows the Stratix and Stratix GX timing waveforms when using strobed nRS and nWS signals.

Figure 11–19. PPA Timing Waveforms Using Strobed nRS & nWS Signals



Notes to Figure 11–19:

- (1) The user can toggle nCS or CS during configuration if the design meets the specification for t_{CSSU} , t_{WSP} and t_{CSH} .
- (2) Device I/O pins are in user mode.
- (3) The DATA [7 . . 0] pins are available as user I/Os after configuration and the state of these pins depends on the dual-purpose pin settings. Do not leave DATA [7 . . 0] floating. If these pins are not used in user-mode, you should drive them high or low, whichever is more convenient.
- (4) DATA7 is a bidirectional pin. It represents an input for data input, but represents an output to show the status of RDYnBSY.

Table 11–10 defines the Stratix and Stratix GX timing parameters for PPA configuration

Symbol	Parameter	Min	Max	Units
t _{CF2WS}	nCONFIG high to first rising edge on nWS	40		μs
t _{DSU}	Data setup time before rising edge on nWS	10		ns
t _{DH}	Data hold time after rising edge on nWS	0		ns
t _{CSSU}	Chip select setup time before rising edge on nWS	10		ns
t _{CSH}	Chip select hold time after rising edge on nWS	0		ns
t _{WSP}	nWS low pulse width	15		ns
t _{CFG}	nCONFIG low pulse width	40		μs
t _{WS2B}	nWS rising edge to RDYnBSY low		20	ns
t _{BUSY}	RDYnBSY low pulse width	7	45	ns
t _{RDY2WS}	RDYnBSY rising edge to nWS rising edge	15		ns
t _{WS2RS}	nWS rising edge to nRS falling edge	15		ns
t _{RS2WS}	nRS rising edge to nWS rising edge	15		ns
t _{RSd7}	nRS falling edge to DATA7 valid with RDYnBSY signal		20	ns
t _{CD2UM}	CONF_DONE high to user mode (1)	6	20	μs
t _{STATUS}	nSTATUS low pulse width	10	40 (2)	μs
t _{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t _{CF2ST0}	nCONFIG low to nSTATUS low		800	ns
t _{CF2ST1}	nCONFIG high to nSTATUS high		40 (2)	μs

Notes to Table 11–10:

- (1) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 136 to obtain this value.
- (2) This value is obtained if you do not delay configuration by extending the nstatus to low pulse width.



For information on how to create configuration and programming files for this configuration scheme, see the *Software Settings* section in the *Configuration Handbook, Volume 2*.

JTAG Programming & Configuration

The JTAG has developed a specification for boundary-scan testing. This boundary-scan test (BST) architecture offers the capability to efficiently test components on printed circuit boards (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. You can also use the JTAG circuitry to shift configuration data into the device.



For more information on JTAG boundary-scan testing, see *AN 39: IEEE 1149.1 (JTAG) Boundary-Scan Testing in Altera Devices*.

To use the SignalTap® II embedded logic analyzer, you need to connect the JTAG pins of your Stratix device to a download cable header on your PCB.



For more information on SignalTap II, see the *Design Debugging Using SignalTap II Embedded Logic Analyzer* chapter in the *Quartus II Handbook, Volume 2*.

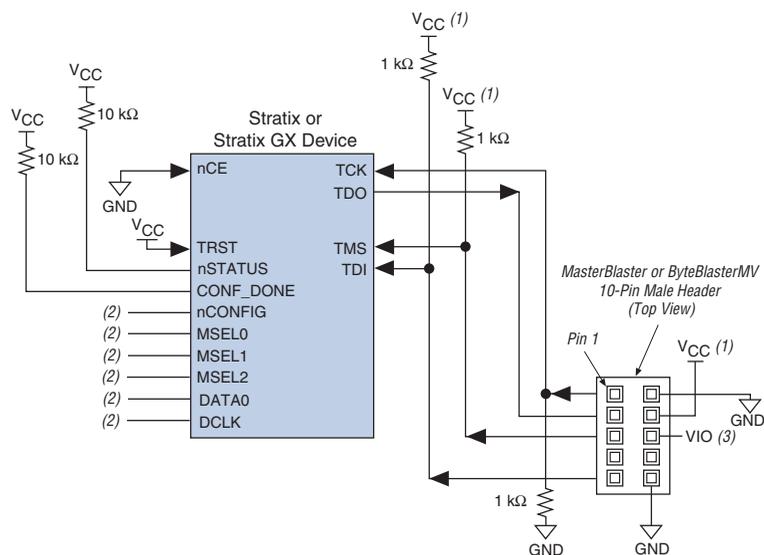
A device operating in JTAG mode uses four required pins, TDI, TDO, TMS, and TCK, and one optional pin, TRST. The four JTAG input pins (TDI, TMS, TCK and TRST) have weak, internal pull-up resistors, whose values range from 20 to 40 k Ω . All other pins are tri-stated during JTAG configuration. Do not begin JTAG configuration until all other configuration is complete. [Table 11–11](#) shows each JTAG pin's function.

Table 11–11. JTAG Pin Descriptions

Pin	Description	Function
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. The VCCSEL pin controls the input buffer selection.
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. The high level output voltage is determined by VCCIO.
TMS	Test mode select	Input pin that provides the control signal to determine the transitions of the Test Access Port (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. The VCCSEL pin controls the input buffer selection.
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 VCCSEL pin controls the input buffer selection.
TRST	Test reset input (optional)	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. The VCCSEL pin controls the input buffer selection.

During JTAG configuration, data is downloaded to the device on the PCB through the MasterBlaster or ByteBlasterMV header. Configuring devices through a cable is similar to programming devices in-system. One difference is to connect the TRST pin to V_{CC} to ensure that the TAP controller is not reset. See Figure 11–20.

Figure 11–20. JTAG Configuration of a Single Device



Notes to Figure 11–20:

- (1) You should connect the pull-up resistor to the same supply voltage as the download cable.
- (2) You should connect the nCONFIG, MSEL0, and MSEL1 pins to support a non-JTAG configuration scheme. If you only use JTAG configuration, connect nCONFIG to V_{CC} , and MSEL0, MSEL1, and MSEL2 to ground. Pull DATA0 and DCLK to high or low.
- (3) V_{IO} is a reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO} . See the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value.

To configure a single device in a JTAG chain, the programming software places all other devices in BYPASS mode. In BYPASS mode, 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 device appears on the TDO pin one clock cycle later.

Stratix and Stratix GX devices have dedicated JTAG pins. You can perform JTAG testing on Stratix and Stratix GX devices before and after, but not during configuration. The chip-wide reset and output enable pins on Stratix and Stratix GX 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 board for JTAG configuration of Stratix and Stratix GX devices, you should consider the regular configuration pins. [Table 11–12](#) shows how you should connect these pins during JTAG configuration.

Table 11–12. Dedicated Configuration Pin Connections During JTAG Configuration

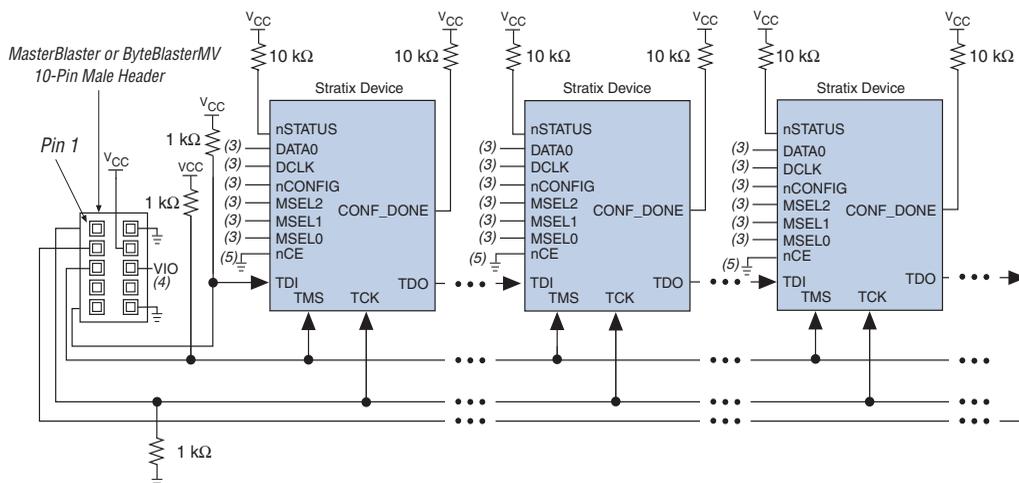
Signal	Description
nCE	On all Stratix and Stratix GX devices in the chain, nCE should be driven low by connecting it to ground, pulling it low via a resistor, or driving it by some control circuitry. For devices that are also in multi-device PS, FPP or PPA configuration chains, the nCE pins should be connected to GND during JTAG configuration or JTAG configured in the same order as the configuration chain.
nCEO	On all Stratix and Stratix GX devices in the chain, nCEO can be left floating or connected to the nCE of the next device. See nCE pin description above.
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 both pins to ground.
nCONFIG	nCONFIG must be driven high through the JTAG programming process. Driven high by connecting to V _{CC} , pulling high via a resistor, or driven by some control circuitry.
nSTATUS	Pull to V _{CC} via a 10-k Ω 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 Ω 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.
DATA0	Should not be left floating. Drive low or high, whichever is more convenient on your board.

JTAG Programming & Configuration of Multiple Devices

When programming a JTAG device chain, one JTAG-compatible header, such as the ByteBlasterMV header, is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capacity of the download cable. However, when more than five 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 PCB contains multiple devices, or when testing the PCB using JTAG BST circuitry. Figure 11–21 shows multi-device JTAG configuration.

Figure 11–21. Multi-Device JTAG Configuration Notes (1), (2)



Notes to Figure 11–21:

- (1) Stratix, Stratix GX, APEX™ II, APEX 20K, Mercury™, ACEX® 1K, and FLEX® 10K devices can be placed within the same JTAG chain for device programming and configuration.
- (2) For more information on all configuration pins connected in this mode, see Table 11–11 on page 11–37.
- (3) Connect the nCONFIG, MSEL0, MSEL1, and MSEL2 pins to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect nCONFIG to V_{CC}, and MSEL0, MSEL1, and MSEL2 to ground. Pull DATA0 and DCLK to either high or low.
- (4) V_{I/O} is a reference voltage for the MasterBlaster output driver. V_{I/O} should match the device's V_{CCIO}. See the MasterBlaster Serial/USB Communications Cable Data Sheet for this value.
- (5) nCE must be connected to GND or driven low for successful JTAG configuration.

The nCE pin must be connected to GND or driven low during JTAG configuration. In multi-device PS, FPP and PPA configuration chains, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-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 multi-device configuration chain, the nCEO of the previous device drives nCE of the next device low when it has successfully been JTAG configured.

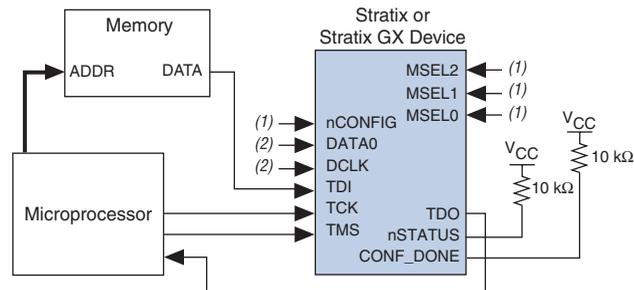
The Quartus II software verifies successful JTAG configuration upon completion. The software checks the state of `CONF_DONE` through the JTAG port. If `CONF_DONE` is not in the correct state, the Quartus II software indicates that configuration has failed. If `CONF_DONE` is in the correct state, the software indicates that configuration was successful.

 If `VCCIO` is tied to 3.3 V, both the I/O pins and JTAG TDO port drive at 3.3-V levels.

Do not attempt JTAG and non-JTAG configuration simultaneously. When configuring through JTAG, allow any non-JTAG configuration to complete first.

Figure 11–22 shows the JTAG configuration of a Stratix or Stratix GX device with a microprocessor.

Figure 11–22. JTAG Configuration of Stratix & Stratix GX Devices with a Microprocessor



Notes to Figure 11–22:

- (1) Connect the `nCONFIG`, `MSEL2`, `MSEL1`, and `MSEL0` pins to support a non-JTAG configuration scheme. If your design only uses JTAG configuration, connect the `nCONFIG` pin to `VCC` and the `MSEL2`, `MSEL1`, and `MSEL0` pins to ground.
- (2) Pull `DATA0` and `DCLK` to either high or low.

Configuration with JRunner Software Driver

JRunner is a software driver that allows you to configure Altera FPGAs through the ByteBlasterMV download cable in JTAG mode. The programming input file supported is in Raw Binary File (`.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. You can customize the code to make it run on other platforms.



For more information on the JRunner software driver, see the *JRunner Software Driver: An Embedded Solution to the JTAG Configuration White Paper* and zip file.

Jam STAPL Programming & Test Language

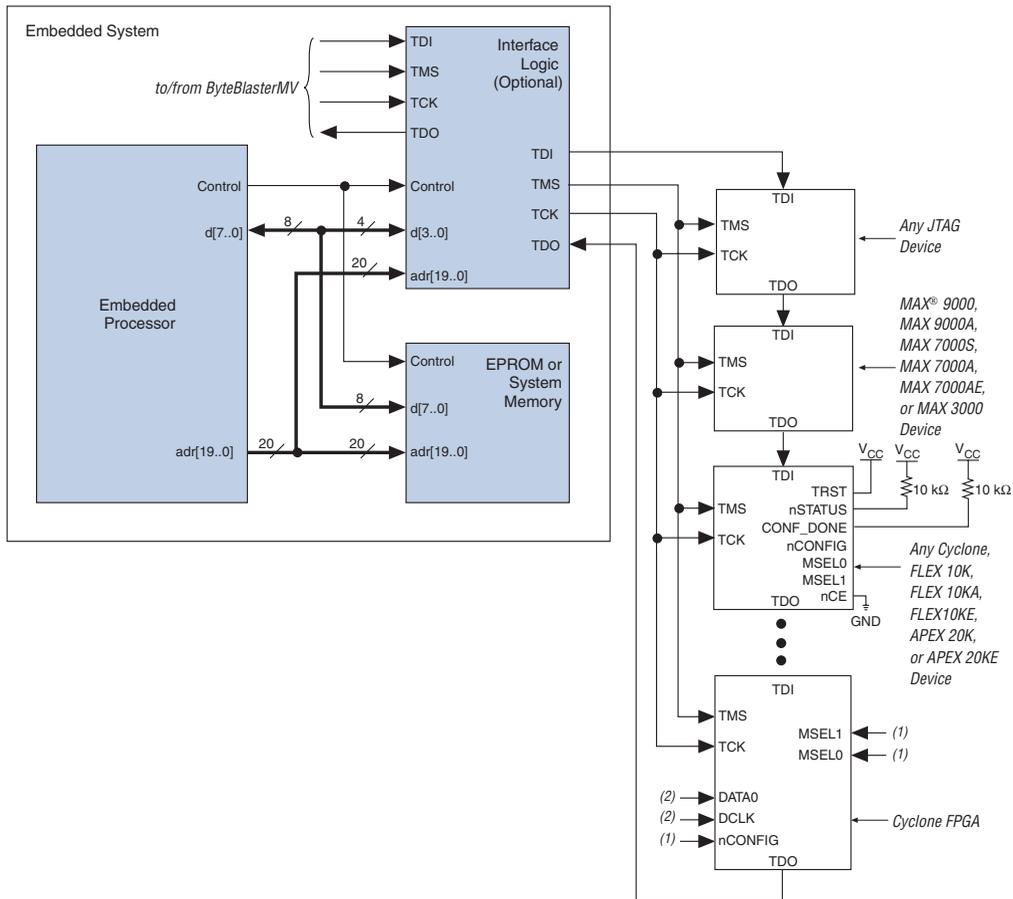
The Jam™ Standard Test and Programming Language (STAPL), JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP) purposes. Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems, using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard.

Connecting the JTAG Chain to the Embedded Processor

There are two ways to connect the JTAG chain to the embedded processor. The most straightforward method is to connect the embedded processor directly to the JTAG chain. In this method, four of the processor pins are dedicated to the JTAG interface, saving board space but reducing the number of available embedded processor pins.

[Figure 11–23](#) illustrates the second method, which is to connect the JTAG chain to an existing bus through an interface PLD. In this method, the JTAG chain becomes an address on the existing bus. The processor then reads from or writes to the address representing the JTAG chain.

Figure 11–23. Embedded System Block Diagram



Notes to Figure 11–23:

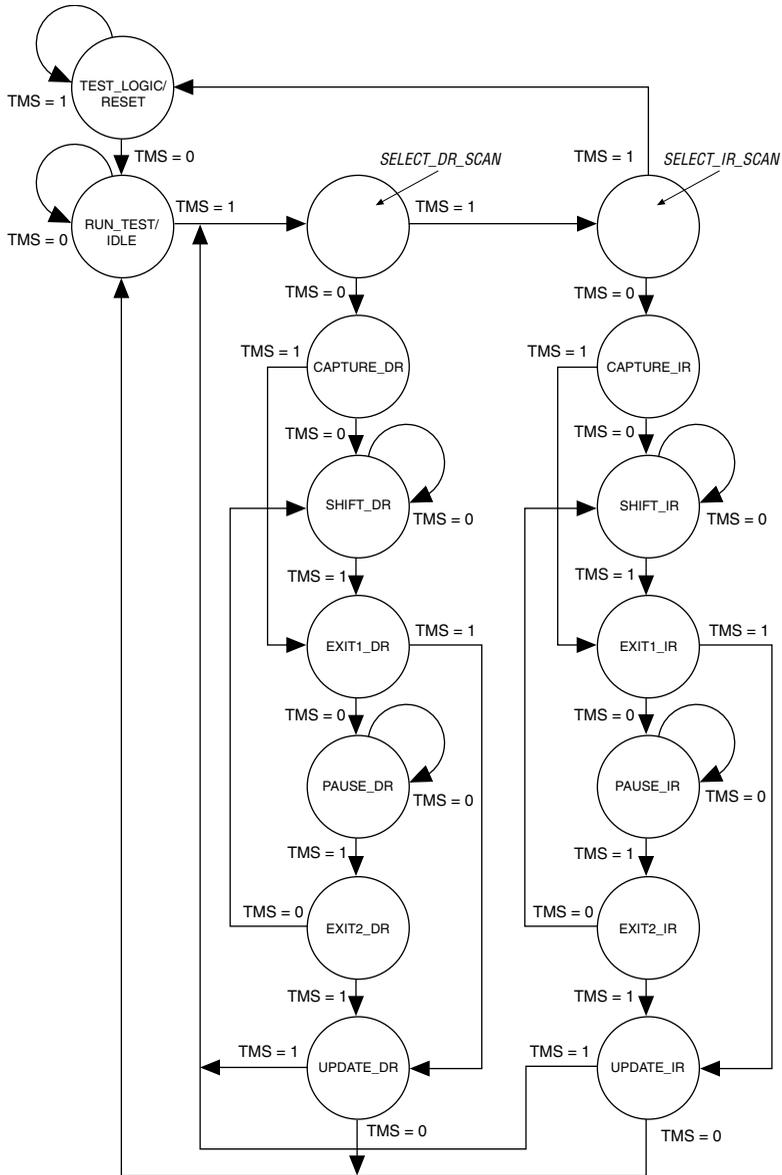
- (1) Connect the nCONFIG, MSEL2, MSEL1, and MSEL0 pins to support a non-JTAG configuration scheme. If your design only uses JTAG configuration, connect the nCONFIG pin to V_{CC} and the MSEL2, MSEL1, and MSEL0 pins to ground.
- (2) Pull DATA0 and DCLK to either high or low.

Both JTAG connection methods should include space for the MasterBlaster or ByteBlasterMV header connection. The header is useful during prototyping because it allows you to verify or modify the Stratix or Stratix GX device’s contents. During production, you can remove the header to save cost.

Program Flow

The Jam Player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine. The TAP controller is a 16-state state machine that is clocked on the rising edge of TCK, and uses the TMS pin to control JTAG operation in a device. [Figure 11–24](#) shows the flow of an IEEE Std. 1149.1 TAP controller state machine.

Figure 11–24. JTAG TAP Controller State Machine



While the Jam Player provides a driver that manipulates the TAP controller, the Jam Byte-Code File (**.jbc**) provides the high-level intelligence needed to program a given device. All Jam instructions that

send JTAG data to the device involve moving the TAP controller through either the data register leg or the instruction register leg of the state machine. For example, loading a JTAG instruction involves moving the TAP controller to the `SHIFT_IR` state and shifting the instruction into the instruction register through the TDI pin. Next, the TAP controller is moved to the `RUN_TEST/IDLE` state where a delay is implemented to allow the instruction time to be latched. This process is identical for data register scans, except that the data register leg of the state machine is traversed.

The high-level Jam instructions are the `DRSCAN` instruction for scanning the JTAG data register, the `IRSCAN` instruction for scanning the instruction register, and the `WAIT` command that causes the state machine to sit idle for a specified period of time. Each leg of the TAP controller is scanned repeatedly, according to instructions in the JBC file, until all of the target devices are programmed.

Figure 11–25 illustrates the functional behavior of the Jam Player when it parses the JBC file. When the Jam Player encounters a `DRSCAN`, `IRSCAN`, or `WAIT` instruction, it generates the proper data on TCK, TMS, and TDI to complete the instruction. The flow diagram shows branches for the `DRSCAN`, `IRSCAN`, and `WAIT` instructions. Although the Jam Player supports other instructions, they are omitted from the flow diagram for simplicity.

Figure 11–25. Jam Player Flow Diagram (Part 1 of 2)

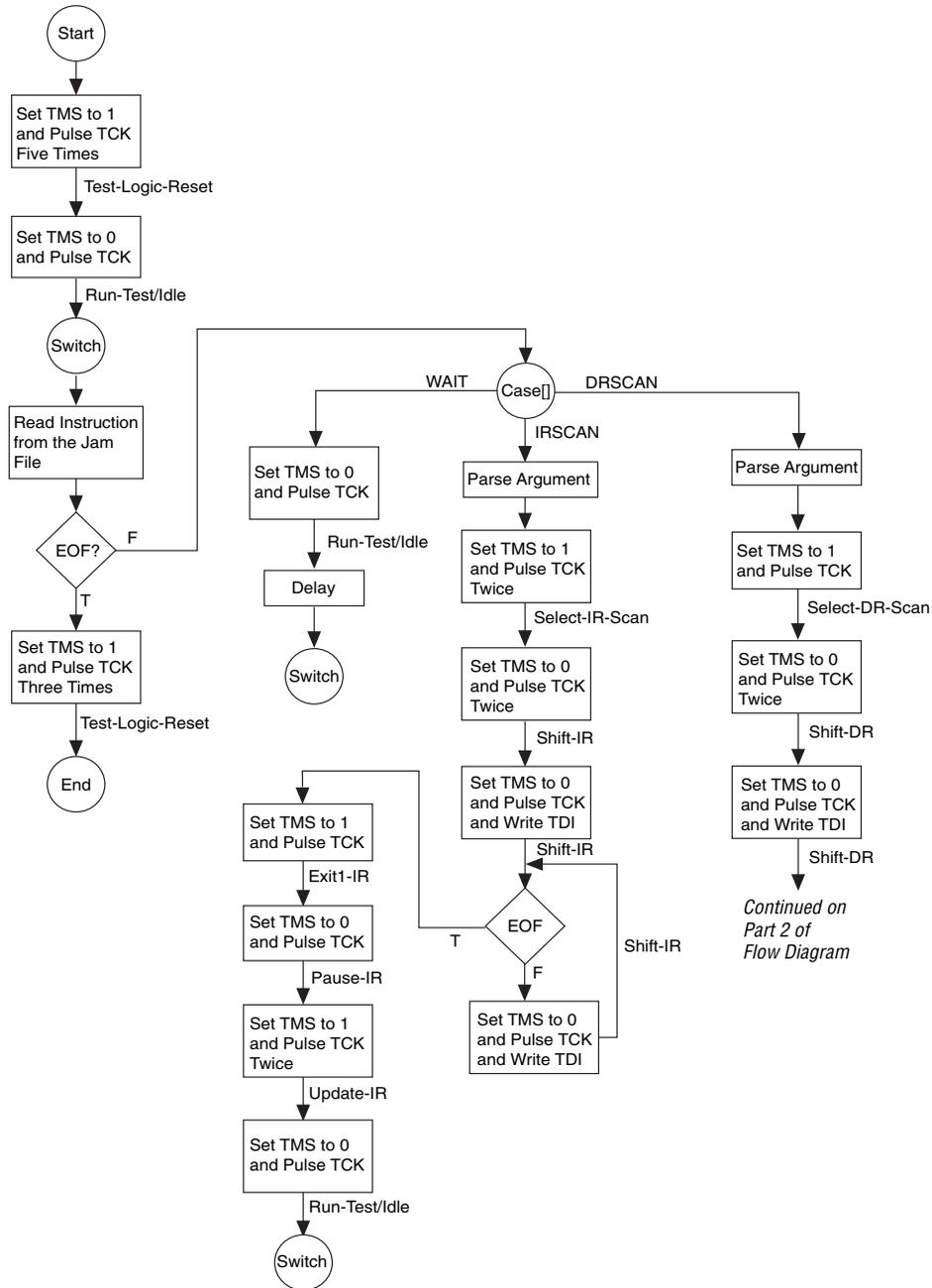
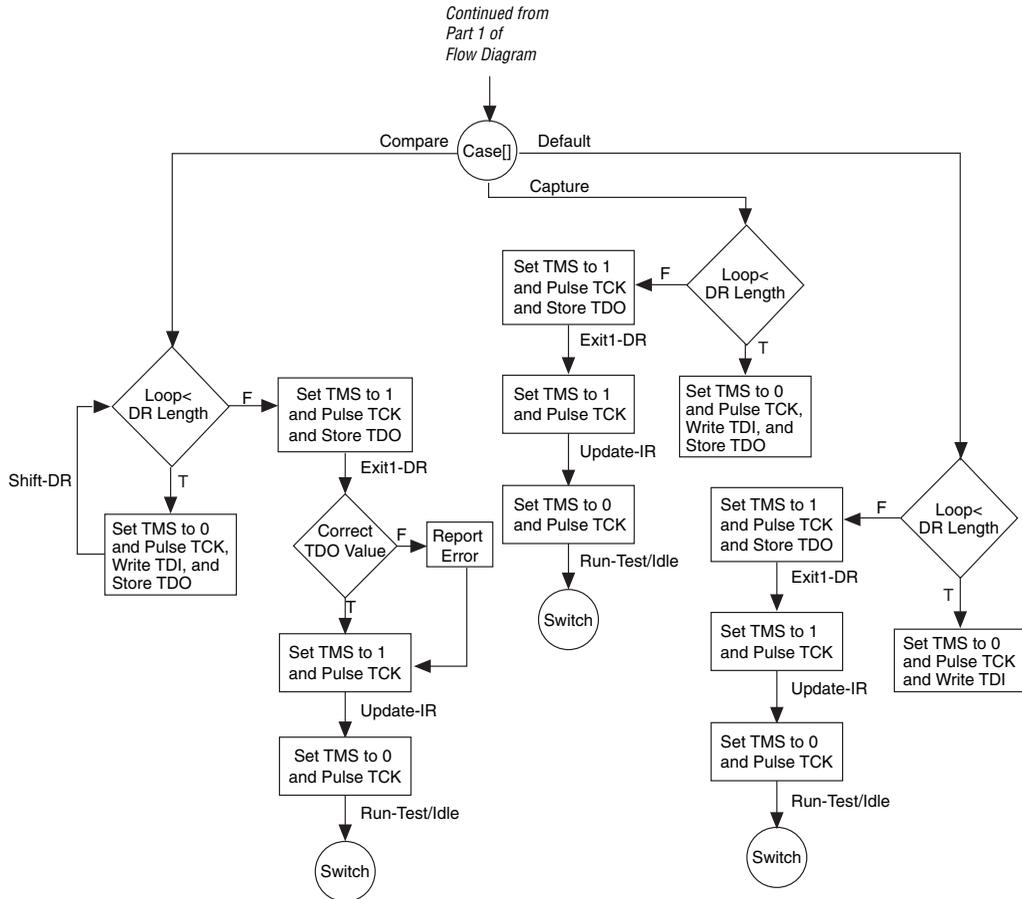


Figure 11–26. Jam Player Flow Diagram (Part 2 of 2)



Execution of a Jam program starts at the beginning of the program. The program flow is controlled using GOTO, CALL/RETURN, and FOR/NEXT structures. The GOTO and CALL statements see labels that are symbolic names for program statements located elsewhere in the Jam program. The language itself enforces almost no constraints on the organizational structure or control flow of a program.



The Jam language does not support linking multiple Jam programs together or including the contents of another file into a Jam program.

Jam Instructions

Each Jam statement begins with one of the instruction names listed in [Table 11–13](#). The instruction names, including the names of the optional instructions, are reserved keywords that you cannot use as variable or label identifiers in a Jam program.

Table 11–13. Instruction Names		
BOOLEAN	INTEGER	PREIR
CALL	IRSCAN	PRINT
CRC	IRSTOP	PUSH
DRSCAN	LET	RETURN
DRSTOP	NEXT	STATE
EXIT	NOTE	WAIT
EXPORT	POP	VECTOR (1)
FOR	POSTDR	VMAP (1)
GOTO	POSTIR	–
IF	PREDR	–

Note to Table 11–13:

(1) This instruction name is an optional language extension.

[Table 11–14](#) shows the state names that are reserved keywords in the Jam language. These keywords correspond to the state names specified in the IEEE Std. 1149.1 JTAG specification.

Table 11–14. Reserved Keywords (Part 1 of 2)	
IEEE Std. 1149.1 JTAG State Names	Jam Reserved State Names
Test-Logic-Reset	RESET
Run-Test-Idle	IDLE
Select-DR-Scan	DRSELECT
Capture-DR	DRCAPTURE
Shift-DR	DRSHIFT
Exit1-DR	DREXIT1
Pause-DR	DRPAUSE
Exit2-DR	DREXIT2
Update-DR	DRUPDATE
Select-IR-Scan	IRSELECT

Table 11–14. Reserved Keywords (Part 2 of 2)

IEEE Std. 1149.1 JTAG State Names	Jam Reserved State Names
Capture-IR	IRCAPTURE
Shift-IR	IRSHIFT
Exit1-IR	IREXIT1
Pause-IR	IRPAUSE
Exit2-IR	IREXIT2
Update-IR	IRUPDATE

Example Jam File that Reads the IDCODE

Figure 11–27 illustrates the flexibility and utility of the Jam STAPL. The example reads the IDCODE out of a single device in a JTAG chain.



The array variable, `I_IDCODE`, is initialized with the IDCODE instruction bits ordered the LSB first (on the left) to most significant bit (MSB) (on the right). This order is important because the array field in the IRSCAN instruction is always interpreted, and sent, MSB to LSB.

Figure 11–27. Example Jam File Reading IDCODE

```

BOOLEAN read_data[32];
BOOLEAN I_IDCODE[10] = BIN 1001101000; `assumed
BOOLEAN ONES_DATA[32] = HEX FFFFFFFF;
INTEGER i;
`Set up stop state for IRSCAN
IRSTOP IRPAUSE;
`Initialize device
STATE RESET;
IRSCAN 10, I_IDCODE[0..9]; `LOAD IDCODE INSTRUCTION
STATE IDLE;
WAIT 5 USEC, 3 CYCLES;
DRSCAN 32, ONES_DATA[0..31], CAPTURE
read_data[0..31];
`CAPTURE IDCODE
PRINT "IDCODE:";
FOR i=0 to 31;
PRINT read_data[i];
NEXT i;
EXIT 0;

```

Configuring Using the MicroBlaster Driver

The MicroBlaster™ software driver allows you to configure Altera devices in an embedded environment using PS or FPP mode. The MicroBlaster software driver supports a Raw Binary File (.rbf) programming input file. The source code is developed for the Windows NT operating system, although you can customize it to run on other operating systems. For more information on the MicroBlaster software driver, go to the Altera web site (www.altera.com).

Device Configuration Pins

The following tables describe the connections and functionality of all the configuration related pins on the Stratix or Stratix GX device. [Table 11–15](#) 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 11–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 1 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
VCCSEL	N/A	All	Input	<p>Dedicated input that selects which input buffer is used on the configuration input pins; nCONFIG, DCLK, RUNLU, nCE, nWS, nRS, CS, nCS and CLKUSR.</p> <p>The VCCSEL input buffer is powered by V_{CCINT} and has an internal 2.5 kΩ pull-down resistor that is always active.</p> <p>A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) selects the 1.8-V/1.5-V input buffer, and a logic low selects the 3.3-V/2.5-V input buffer. See the “V_{CCSEL} Pins” section for more details.</p>
PORSEL	N/A	All	Input	<p>Dedicated input which selects between a POR time of 2 ms or 100 ms. A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) selects a POR time of about 2 ms and a logic low selects POR time of about 100 ms.</p> <p>The PORSEL input buffer is powered by V_{CCINT} and has an internal 2.5 kΩ pull-down resistor that is always active.</p>

Table 11–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 2 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nIO_PULLUP	N/A	All	Input	<p>Dedicated input that chooses whether the internal pull-ups on the user I/Os and dual-purpose I/Os (DATA [7 . . 0], nWS, nRS, RDYnBSY, nCS, CS, RU_nLU, PGM [], CLKUSR, INIT_DONE, DEV_OE, DEV_CLR) are on or off before and during configuration. A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) turns off the weak internal pull-ups, while a logic low turns them on.</p> <p>The nIO_PULLUP input buffer is powered by V_{CCINT} and has an internal 2.5 kΩ pull-down resistor that is always active.</p>
MSEL [2 . . 0]	N/A	All	Input	<p>3-bit configuration input that sets the Stratix or Stratix GX device configuration scheme. See Table 11–2 for the appropriate connections.</p> <p>These pins can be connected to V_{CCIO} of the I/O bank they reside in or ground. This pin uses Schmitt trigger input buffers.</p>
nCONFIG	N/A	All	Input	<p>Configuration control input. Pulling this pin low during user-mode causes the FPGA to lose its configuration data, enter a reset state, tri-state all I/O pins. Returning this pin to a logic high level initiates a reconfiguration.</p> <p>If your configuration scheme uses an enhanced configuration device or EPC2 device, nCONFIG can be tied directly to V_{CC} or to the configuration device's nINIT_CONF pin. This pin uses Schmitt trigger input buffers.</p>

Table 11–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 3 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nSTATUS	N/A	All	Bidirectional open-drain	<p>The device drives nSTATUS low immediately after power-up and releases it after the POR time.</p> <p>Status output. If an error occurs during configuration, nSTATUS is pulled low by the target device. Status input. If an external source 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 a configuration device is used, 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, nCONFIG must be pulled low.</p> <p>The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external 10-kΩ pull-up resistors should not be used on these pins. When using EPC2 devices, only external 10-kΩ pull-up resistors should be used.</p> <p>This pin uses Schmitt trigger input buffers.</p>

Table 11–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 4 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
CONF_DONE	N/A	All	Bidirectional open-drain	<p>Status output. The target FPGA drives the CONF_DONE pin low before and during configuration. Once all configuration data is received without error and the initialization cycle starts, the target device releases CONF_DONE.</p> <p>Status input. After all data is received and CONF_DONE goes high, the target device initializes and enters user mode. The CONF_DONE pin must have an external 10-kΩ 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 have optional internal programmable pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external 10-kΩ pull-up resistors should not be used on these pins. When using EPC2 devices, only external 10-kΩ pull-up resistors should be used.</p> <p>This pin uses Schmitt trigger input buffers.</p>
nCE	N/A	All	Input	<p>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 multi-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. This pin uses Schmitt trigger input buffers.</p>

Table 11–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 5 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCEO	N/A	All Multi-Device Schemes	Output	<p>Output that drives low when device configuration is complete. In single device configuration, this pin is left floating. In multi-device configuration, this pin feeds the next device's nCE pin. The nCEO of the last device in the chain is left floating.</p> <p>The voltage levels driven out by this pin are dependent on the V_{CCIO} of the I/O bank it resides in.</p>
DCLK	N/A	Synchronous configuration schemes (PS, FPP)	Input (PS, FPP)	<p>In PS and FPP configuration, DCLK is the clock input used to clock data from an external source into the target device. Data is latched into the FPGA on the rising edge of DCLK.</p> <p>In PPA mode, DCLK should be tied high to V_{CC} to prevent this pin from floating.</p> <p>After configuration, this pin is tri-stated. In schemes that use a configuration device, DCLK is driven low after configuration is done. In schemes that use a control host, DCLK should be driven either high or low, whichever is more convenient. Toggling this pin after configuration does not affect the configured device. This pin uses Schmitt trigger input buffers.</p>
DATA0	I/O	PS, FPP, PPA	Input	<p>Data input. In serial configuration modes, bit-wide configuration data is presented to the target device on the DATA0 pin. The V_{IH} and V_{IL} levels for this pin are dependent on the V_{CCIO} of the I/O bank that it resides in.</p> <p>After configuration, DATA0 is available as a user I/O and the state of this pin depends on the Dual-Purpose Pin settings.</p> <p>After configuration, EPC1 and EPC1441 devices tri-state this pin, while enhanced configuration and EPC2 devices drive this pin high.</p>

Table 11–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 6 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DATA [7 . . 1]	I/O	Parallel configuration schemes (FPP and PPA)	Inputs	<p>Data inputs. Byte-wide configuration data is presented to the target device on DATA [7 . . 0]. The V_{IH} and V_{IL} levels for these pins are dependent on the V_{CCIO} of the I/O banks that they reside in.</p> <p>In serial configuration schemes, they function as user I/Os during configuration, which means they are tri-stated.</p> <p>After PPA or FPP configuration, DATA [7 . . 1] are available as a user I/Os and the state of these pin depends on the Dual-Purpose Pin settings.</p>
DATA7	I/O	PPA	Bidirectional	<p>In the PPA configuration scheme, the DATA7 pin presents the RDY_{nBSY} signal after the nRS signal has been strobed low. The V_{IL} and V_{IL} levels for this pin are dependent on the V_{CCIO} of the I/O bank that it resides in.</p> <p>In serial configuration schemes, it functions as a user I/O during configuration, which means it is tri-stated.</p> <p>After PPA configuration, DATA7 is available as a user I/O and the state of this pin depends on the Dual-Purpose Pin settings.</p>
nWS	I/O	PPA	Input	<p>Write strobe input. A low-to-high transition causes the device to latch a byte of data on the DATA [7 . . 0] pins.</p> <p>In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated.</p> <p>After PPA configuration, nWS is available as a user I/O and the state of this pin depends on the Dual-Purpose Pin settings.</p>

Table 11–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 7 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nRS	I/O	PPA	Input	<p>Read strobe input. A low input directs the device to drive the RDYnBSY signal on the DATA7 pin.</p> <p>If the nRS pin is not used in PPA mode, it should be tied high. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated.</p> <p>After PPA configuration, nRS is available as a user I/O and the state of this pin depends on the Dual-Purpose Pin settings.</p>
RDYnBSY	I/O	PPA	Output	<p>Ready output. A high output indicates that the target device is ready to accept another data byte. A low output indicates that the target device is busy and not ready to receive another data byte.</p> <p>In PPA configuration schemes, this pin drives out high after power-up, before configuration and after configuration before entering user-mode. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated.</p> <p>After PPA configuration, RDYnBSY is available as a user I/O and the state of this pin depends on the Dual-Purpose Pin settings.</p>

Table 11–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 8 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCS/CS	I/O	PPA	Input	<p>Chip-select inputs. A low on nCS and a high on CS select the target device for configuration. The nCS and CS pins must be held active during configuration and initialization.</p> <p>During the PPA configuration mode, it is only required to use either the nCS or CS pin. Therefore, if only one chip-select input is used, the other must be tied to the active state. For example, nCS can be tied to GND while CS is toggled to control configuration. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated.</p> <p>After PPA configuration, nCS and CS are available as a user I/Os and the state of these pins depends on the Dual-Purpose Pin settings.</p>
RU _n LU	N/A if using Remote Configuration; I/O if not	Remote Configuration in FPP, PS or PPA	Input	<p>Input that selects between remote update and local update. A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) selects remote update and a logic low selects local update.</p> <p>When not using remote update or local update configuration modes, this pins is available as general-purpose user I/O pin.</p>
PGM [2 . . 0]	N/A if using Remote Configuration; I/O if not using	Remote Configuration in FPP, PS or PPA	Input	<p>These output pins select one of eight pages in the memory (either flash or enhanced configuration device) when using a remote configuration mode.</p> <p>When not using remote update or local update configuration modes, these pins are available as general-purpose user I/O pins.</p>

Table 11–16 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-ups.

Pin Name	User Mode	Pin Type	Description
CLKUSR	N/A if option is on. I/O if option is off.	Input	Optional user-supplied clock input. Synchronizes the initialization of one or more devices. This pin is enabled by turning on the Enable user-supplied start-up clock (CLKUSR) option in the Quartus II software.
INIT_DONE	N/A if option is on. I/O if option is off.	Output open-drain	Status pin. Can be used to indicate when the device has initialized and is in user mode. When $\overline{n}CONFIG$ is low and during the beginning of configuration, the INIT_DONE pin is tri-stated and pulled high due to an external 10-k Ω pull-up. Once the option 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 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 Enable INIT_DONE output 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/Os are tri-stated. When this pin is driven high, all I/Os behave as programmed. This pin is enabled by turning on the Enable device-wide output enable (DEV_OE) 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 Enable device-wide reset (DEV_CLRn) option in the Quartus II software.

Table 11–17 describes the dedicated JTAG pins. JTAG pins must be kept stable before and during configuration to prevent accidental loading of JTAG instructions. If you plan to use the SignalTap II Embedded Logic Analyzer, you will need to connect the JTAG pins of your device to a JTAG header on your board.

Pin Name	User Mode	Pin Type	Description
TDI	N/A	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 _{CC} . This pin uses Schmitt trigger input buffers.
TDO	N/A	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	N/A	Input	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 _{CC} . This pin uses Schmitt trigger input buffers.
TCK	N/A	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. This pin uses Schmitt trigger input buffers.
TRST	N/A	Input	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND. This pin uses Schmitt trigger input buffers.