

## Introduction

Arria™ GX II devices use SRAM cells to store configuration data. Because SRAM memory is volatile, configuration data must be downloaded to Arria GX devices each time the device powers up. Arria GX devices can be configured using one of five configuration schemes: the fast passive parallel (FPP), active serial (AS), passive serial (PS), passive parallel asynchronous (PPA), and Joint Test Action Group (JTAG) configuration schemes. All configuration schemes use either an external controller (for example, a MAX® II device or microprocessor) or a configuration device.

This chapter contains the following sections:

- “Configuration Features” on page 11–4
- “Fast Passive Parallel Configuration” on page 11–13
- “Active Serial Configuration (Serial Configuration Devices)” on page 11–32
- “Passive Serial Configuration” on page 11–44
- “Passive Parallel Asynchronous Configuration” on page 11–71
- “JTAG Configuration” on page 11–82
- “Device Configuration Pins” on page 11–90
- “Conclusion” on page 11–104

## Configuration Devices

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



For information on enhanced configuration devices, refer to the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* and the *Altera Enhanced Configuration Devices* chapters in volume 2 of the *Configuration Handbook*.

The Altera serial configuration devices (EPCS64, EPCS16, and EPCS4) support a single-device configuration solution for Arria GX devices and are used in the AS configuration scheme. Serial configuration devices offer a low cost, low pin count configuration solution.



For information on serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

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



For more information on EPC2 configuration devices, refer to the *Configuration Devices for SRAM-Based LUT Devices Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

## Configuration Schemes

The configuration scheme is selected by driving the Arria GX device MSEL pins either high or low, as shown in [Table 11–1](#). The MSEL pins are powered by the  $V_{CCINT}$  power supply of the bank they reside in. The MSEL [3 : 0] pins have 5-k $\Omega$  internal pull-down resistors that are always active. During power-on reset (POR) and during reconfiguration, the MSEL pins have to be at LVTTTL  $V_{IL}$  and  $V_{IH}$  levels to be considered a logic low and logic high.



To avoid any problems with detecting an incorrect configuration scheme, hard-wire the MSEL [ ] pins to  $V_{CCPD}$  and GND, without any pull-up or pull-down resistors. Do not drive the MSEL [ ] pins by a microprocessor or another device.

**Table 11–1. Arria GX Configuration Schemes (Part 1 of 2)**

Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0
Fast passive parallel (FPP)	0	0	0	0
Passive parallel asynchronous (PPA)	0	0	0	1
Passive serial (PS)	0	0	1	0
Remote system upgrade FPP (1)	0	1	0	0
Remote system upgrade PPA (1)	0	1	0	1
Remote system upgrade PS (1)	0	1	1	0
Fast AS (40 MHz) (2)	1	0	0	0
Remote system upgrade fast AS (40 MHz) (2)	1	0	0	1
FPP with decompression feature enabled (3)	1	0	1	1
Remote system upgrade FPP with decompression feature enabled (1), (3)	1	1	0	0
AS (20 MHz) (2)	1	1	0	1

**Table 11–1. Arria GX Configuration Schemes (Part 2 of 2)**

Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0
Remote system upgrade AS (20 MHz) (2)	1	1	1	0
JTAG-based configuration (5)	(4)	(4)	(4)	(4)

**Notes to Table 11–1:**

- (1) These schemes require that you drive the RUNLU pin to specify either remote update or local update. For more information about remote system upgrades in Arria GX devices, refer to the *Remote System Upgrades With Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.
- (2) Only the EPCS16 and EPCS64 devices support up to a 40 MHz DCLK. Other EPCS devices support up to a 20 MHz DCLK. Refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook* for more information.
- (3) These modes are only supported when using a MAX II device or a microprocessor with flash memory for configuration. In these modes, the host system must output a DCLK that is 4× the data rate.
- (4) Do not leave the MSEL pins floating. Connect them to V<sub>CCPD</sub> or ground. 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.
- (5) JTAG-based configuration takes precedence over other configuration schemes, which means MSEL pin settings are ignored.

Arria GX devices offer decompression and remote system upgrade features. Arria GX devices can receive a compressed configuration bitstream and decompress this data in real-time, reducing storage requirements and configuration time. You can make real-time system upgrades from remote locations of your Arria GX designs with the remote system upgrade feature.

Table 11–2 shows the uncompressed configuration file sizes for Arria GX devices.

**Table 11–2. Arria GX Uncompressed .rbf Sizes** *Note (1)*

Device	Data Size (Bits)	Data Size (MBytes)
EP1AGX20	9,640,672	1.205
EP1AGX35	9,640,672	1.205
EP1AGX50	16,951,824	2.119
EP1AGX60	16,951,824	2.119
EP1AGX90	25,699,104	3.212

**Note to Table 11–2:**

- (1) .rbf: Raw Binary File.

Use the data in [Table 11–2](#) to estimate the file size before design compilation. Different configuration file formats, such as a Hexidecimal (.hex) or Tabular Text File (.tff) format, will have different file sizes. However, for any specific version of the Quartus® II software, any design targeted for the same device will have the same uncompressed configuration file size. If you are using compression, the file size can vary after each compilation because the compression ratio is dependent on the design.

This chapter explains the Arria GX device configuration features and describes how to configure Arria GX devices using the supported configuration schemes. This chapter provides configuration pin descriptions and the Arria GX device configuration file formats. In this chapter, the generic term device(s) includes all Arria GX devices.



For more information on setting device configuration options or creating configuration files, refer to the [Software Settings](#) section in volume 2 of the *Configuration Handbook*.

## Configuration Features

Arria GX devices offer configuration data decompression to reduce configuration file storage and remote system upgrades to allow you to remotely update your Arria GX designs. [Table 11–3](#) summarizes which configuration features can be used in each configuration scheme.

<b>Table 11–3. Arria GX Configuration Features</b>			
<b>Configuration Scheme</b>	<b>Configuration Method</b>	<b>Decompression</b>	<b>Remote System Upgrade</b>
FPP	MAX II device or a Microprocessor with flash memory	✓ (1)	✓
	Enhanced Configuration Device	✓ (2)	✓
AS	Serial Configuration Device	✓	✓ (3)
PS	MAX II device or a Microprocessor with flash memory	✓	✓
	Enhanced Configuration Device	✓	✓
	Download cable	✓	
PPA	MAX II device or a Microprocessor with flash memory		✓
JTAG	MAX II device or a Microprocessor with flash memory		

### Notes to [Table 11–3](#):

- (1) In these modes, the host system must send a DCLK that is 4× the data rate.
- (2) The enhanced configuration device decompression feature is available, while the Arria GX decompression feature is not available.
- (3) Only remote update mode is supported when using the AS configuration scheme. Local update mode is not supported.

## Configuration Data Decompression

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



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

Arria GX devices support decompression in the FPP (when using a MAX II device/microprocessor + flash), AS, and PS configuration schemes. Decompression is not supported in the PPA configuration scheme nor in JTAG-based configuration.



When using FPP mode, the intelligent host must provide a DCLK that is 4× the data rate. Therefore, the configuration data must be valid for four DCLK cycles.

The decompression feature supported by Arria GX devices is different from the decompression feature in enhanced configuration devices (EPC16, EPC8, and EPC4 devices), although they both use the same compression algorithm. The data decompression feature in the enhanced configuration devices allows them to store compressed data and decompress the bitstream before transmitting it to the target devices. When using Arria GX devices in FPP mode with enhanced configuration devices, the decompression feature is available only in the enhanced configuration device, not the Arria GX device.

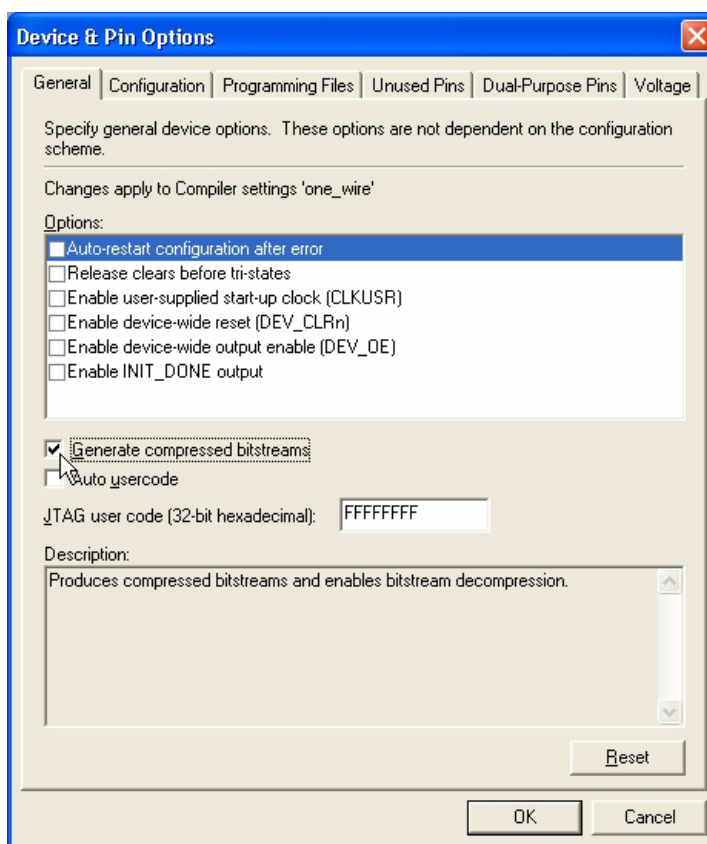
In PS mode, use the Arria GX decompression feature because sending compressed configuration data reduces configuration time. Do not use both the Arria GX device and the enhanced configuration device decompression features simultaneously. The compression algorithm is not intended to be recursive and could expand the configuration file instead of compressing it further.

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

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

To enable compression in the project's compiler settings, select **Device** under the **Assignments** menu to bring up the **Settings** window. After selecting your Arria GX device, open the **Device & Pin Options** window, and in the **General** settings tab, enable the check box for **Generate compressed bitstreams** (as shown in Figure 11–1).

**Figure 11–1. Enabling Compression for Arria GX Bitstreams in Compiler Settings**



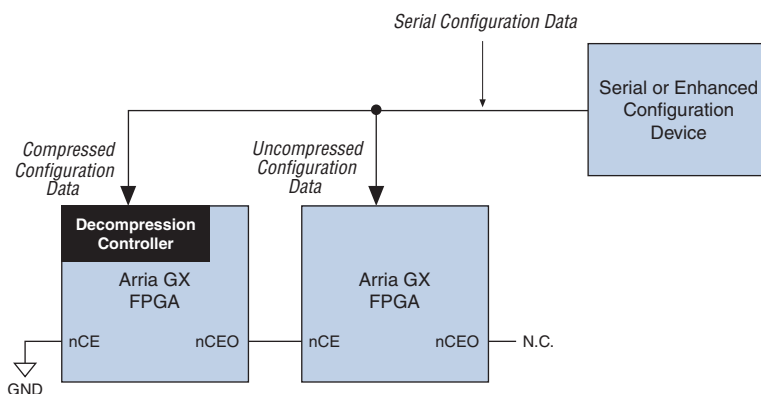
Compression can also be enabled when creating programming files from the **Convert Programming Files** window by following these steps:

1. Click **Convert Programming Files** (File menu).
2. Select the programming file type (POF, SRAM HEXOUT, RBF, or TTF).
3. For POF output files, select a configuration device.
4. In the **Input files to convert** box, select **SOF Data**.
5. Select **Add File** and add an Arria GX device SOF(s).
6. Select the name of the file you added to the **SOF Data** area and click **Properties**.
7. Check the **Compression** check box.

When multiple Arria GX devices are cascaded, you can selectively enable the compression feature for each device in the chain if you are using a serial configuration scheme. [Figure 11–2](#) depicts a chain of two Arria GX devices. The first Arria GX device has compression enabled and receives a compressed bitstream from the configuration device. The second Arria GX device has the compression feature disabled and receives uncompressed data.

In a multi-device FPP configuration chain, all Arria GX devices in the chain must either enable or disable the decompression feature. You can not selectively enable the compression feature for each device in the chain because of the DATA and DCLK relationship.

**Figure 11–2. Compressed & Uncompressed Configuration Data in the Same Configuration File**



You can generate programming files for this setup from the **Convert Programming Files** window (File menu) in the Quartus II software.

## Remote System Upgrade

Arria GX devices feature remote and local update.



For more information about this feature, refer to the *Remote System Upgrades with Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

## Power-On Reset Circuit

The POR circuit keeps the entire system in reset until the power supply voltage levels have stabilized on power-up. Upon power-up, the device does not release nSTATUS until  $V_{CCINT}$ ,  $V_{CCPD}$ , and  $V_{CCIO}$  of banks 3, 4, 7, and 8 are above the device's POR trip point. On power down,  $V_{CCINT}$  is monitored for brown-out conditions.

The passive serial mode ( $MSEL[3..0] = 0010$ ) and the Fast passive parallel mode ( $MSEL[3..0] = 0000$ ) always enable bank 3 to use the lower POR trip point consistent with 1.8- and 1.5-V signaling, regardless of the VCCSEL setting. For all other configuration modes, VCCSEL selects the POR trip point level. Refer to *"VCCSEL Pin"* on page 11–9 for more details.



In Arria GX devices, the pin-selectable option `PORSEL` allows you to select between a typical POR time setting of 12 ms or 100 ms. In both cases, you can extend the POR time by using an external component to assert the `nSTATUS` pin low.

## V<sub>CCPD</sub> Pins

Arria GX devices also offer a new power supply,  $V_{CCPD}$ , which must be connected to 3.3-V in order to power the 3.3-V/2.5-V buffer available on the configuration input pins and JTAG pins.  $V_{CCPD}$  applies to all the JTAG input pins (`TCK`, `TMS`, `TDI`, and `TRST`) and the configuration pins when `VCCSEL` is connected to ground. Refer to [Table 11–4](#) for information on the pins affected by `VCCSEL`.



$V_{CCPD}$  must ramp-up from 0-V to 3.3-V within 100 ms. If  $V_{CCPD}$  is not ramped up within this specified time, your Arria GX device will not configure successfully. If your system does not allow for a  $V_{CCPD}$  ramp-up time of 100 ms or less, you must hold `nCONFIG` low until all power supplies are stable.

## VCCSEL Pin

The `VCCSEL` pin selects the type of input buffer used on configuration input pins and it selects the POR trip point voltage level for  $V_{CCIO}$  bank 3 powered by `VCCIO3` pins.

The configuration input pins and the `PLL_ENA` pin ([Table 11–4](#)) have a dual buffer design. These pins have a 3.3-V/2.5-V input buffer and a 1.8-V/1.5-V input buffer. The `VCCSEL` input pin selects which input buffer is used during configuration. The 3.3-V/2.5-V input buffer is powered by  $V_{CCPD}$ , while the 1.8-V/1.5-V input buffer is powered by

$V_{CCIO}$ . After configuration, the dual-purpose configuration pins are powered by the  $V_{CCIO}$  pins. Table 11–4 shows the pins affected by  $VCCSEL$ .

<b>Table 11–4. Pins Affected by the Voltage Level at <math>VCCSEL</math></b>		
<b>Pin</b>	<b><math>VCCSEL = \text{LOW}</math> (connected to GND)</b>	<b><math>VCCSEL = \text{HIGH}</math> (connected to <math>V_{CCPD}</math>)</b>
nSTATUS (when used as an input)	3.3/2.5-V input buffer is selected. Input buffer is powered by $V_{CCPD}$ .	1.8/1.5-V input buffer is selected. Input buffer is powered by $V_{CCIO}$ of the I/O bank. These input buffers are 3.3-V tolerant.
nCONFIG		
CONF_DONE (when used as an input)		
DATA[7..0]		
nCE		
DCLK (when used as an input)		
CS		
nWS		
nRS		
nCS		
CLKUSR		
DEV_OE		
DEV_CLRn		
RUnLU		
PLL_ENA		

$VCCSEL$  is sampled during power-up. Therefore, the  $VCCSEL$  setting cannot change on-the-fly or during a reconfiguration. The  $VCCSEL$  input buffer is powered by  $V_{CCINT}$  and has an internal 5-k $\Omega$  pull-down resistor that is always active.



$VCCSEL$  must be hardwired to  $V_{CCPD}$  or GND.

A logic high selects the 1.8-V/1.5-V input buffer, and a logic low selects the 3.3-V/2.5-V input buffer.  $VCCSEL$  should be set to comply with the logic levels driven out of the configuration device or MAX II device or a microprocessor with flash memory.

$VCCSEL$  also sets the POR trip point for I/O bank 3 to ensure that this I/O bank has powered up to the appropriate voltage levels before configuration begins. For passive serial (PS) mode ( $MSEL[3..0] = 0010$ ) and for Fast passive parallel (FPP) mode ( $MSEL[3..0] = 0000$ )

the POR circuitry selects the trip point associated with 1.5/1.8-V signaling. For all other configuration modes defined by MSEL[3..0] settings other than 00X0 (MSEL[1] = X, don't care), VCCSEL=GND selects the higher I/O Bank 3 POR trip point for 2.5V/3.3V signaling and VCCSEL=V<sub>CCPD</sub> selects the lower I/O Bank 3 POR trip point associated with 1.5V/1.8V signaling.

For all configuration modes with MSEL[3..0] not equal to 00X0 (MSEL[1] = X, don't care), if V<sub>CCIO</sub> of configuration bank 3 is powered by 1.8-V or 1.5-V and VCCSEL = GND, the voltage supplied to this I/O bank(s) may never reach the POR trip point, which prevents the device from beginning configuration.



The fast passive parallel (FPP) and passive serial (PS) modes always enable bank 3 to use the POR trip point to be consistent with 1.8- and 1.5-V signaling, regardless of the VCCSEL setting.

If the V<sub>CCIO</sub> of I/O bank 3 is powered by 1.5 or 1.8-V and the configuration signals used require 3.3- or 2.5-V signaling, you should set VCCSEL to V<sub>CCPD</sub> to enable the 1.8/1.5-V input buffers for configuration. The 1.8-V/1.5-V input buffers are 3.3-V tolerant.

Table 11–5 shows how you should set the VCCSEL, depending on the configuration mode, the voltage level on VCCIO3 pins that power bank 3, and the supported configuration input voltages.

**Table 11–5. Supported V<sub>CCSEL</sub> Setting based on Mode, VCCIO3, and Input Configuration Voltage**

Configuration Mode	V <sub>CCIO</sub> (Bank 3)	Configuration Input Signaling Voltage	V <sub>CCSEL</sub>
All modes	3.3-V/2.5-V	3.3-V/2.5-V	GND
All modes	1.8-V/1.5-V	3.3-V/2.5-V	V <sub>CCPD</sub> (1)
All modes	1.8-V/1.5-V	1.8-V/1.5-V	V <sub>CCPD</sub>
-	3.3-V/2.5-V	1.8-V/1.5-V	Not Supported

**Note to Table 11–5:**

- (1) The VCCSEL pin can also be connected to GND for PS (MSEL[3..0] = 0010) and FPP (MSEL[3..0] = 0000) modes.

The key is to ensure the V<sub>CCIO</sub> voltage of bank 3 is high enough to trip VCCIO3 POR trip point on power-up. Also, to make sure the configuration device meets the V<sub>IH</sub> for the configuration input pins based on the selected input buffer.

Table 11–6 shows the configuration mode support for banks 4, 7, and 8.

<b>Table 11–6. Arria GX Configuration Mode Support for Banks 4, 7, &amp; 8</b>			
<b>Configuration Mode</b>	<b>Configuration Voltage/<math>V_{CCIO}</math> Support for Banks 4, 7, &amp; 8</b>		
	<b>3.3/3.3</b>	<b>1.8/1.8</b>	<b>3.3/1.8</b>
	<b>VCCSEL = GND</b>	<b>VCCSEL = VCCPD</b>	<b>VCCSEL = GND</b>
Fast passive parallel	Y	Y	Y
Passive parallel asynchronous	Y	Y	Y
Passive serial	Y	Y	Y
Remote system upgrade FPP	Y	Y	Y
Remote system upgrade PPA	Y	Y	Y
Remote system upgrade PS	Y	Y	Y
Fast AS (40 MHz)	Y	Y	Y
Remote system upgrade fast AS (40 MHz)	Y	Y	Y
FPP with decompression	Y	Y	Y
Remote system upgrade FPP with decompression feature enabled	Y	Y	Y
AS (20 MHz)	Y	Y	Y
Remote system upgrade AS (20 MHz)	Y	Y	Y

You must verify the configuration output pins for your chosen configuration modes meet the  $V_{IH}$  of the configuration device. Refer to Table 11–22 for a consolidated list of configuration output pins.

The  $V_{IH}$  of 3.3 or 2.5 V configuration devices will not be met when the  $V_{CCIO}$  of the output configuration pins is 1.8 V or 1.5 V. Level shifters will be required to meet the input high level voltage threshold  $V_{IH}$ .

Note that AS mode is only applicable for 3.3-V configuration. If I/O bank 3 is less than 3.3V then level shifters are required on the output pins (DCLK, nCSO, ASDO) from the Arria GX device back to the EPCS device.

The VCCSEL signal does not control TDO or nCEO. During configuration, these pins drive out voltage levels corresponding to the  $V_{CCIO}$  supply voltage that powers the I/O bank containing the pin.



For more information on multi-volt support, including information on using TDO and nCEO in multi-volt systems, refer to the *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook*.

## Fast Passive Parallel Configuration

Fast passive parallel (FPP) configuration in Arria GX devices is designed to meet the continuously increasing demand for faster configuration times. Arria GX devices are designed with the capability of receiving byte-wide configuration data per clock cycle. [Table 11–7](#) shows the MSEL pin settings when using the FPP configuration scheme.

**Table 11–7. Arria GX MSEL Pin Settings for FPP Configuration Schemes**

Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0
FPP when not using remote system upgrade or decompression feature	0	0	0	0
FPP when using remote system upgrade (1)	0	1	0	0
FPP with decompression feature enabled (2)	1	0	1	1
FPP when using remote system upgrade and decompression feature (1), (2)	1	1	0	0

**Notes to Table 11–7:**

- (1) These schemes require that you drive the RUnLU pin to specify either remote update or local update. For more information about remote system upgrade in Arria GX devices, refer to the [Remote System Upgrades with Arria GX Devices](#) chapter in volume 2 of the *Arria GX Device Handbook*.
- (2) These modes are only supported when using a MAX II device or a microprocessor with flash memory for configuration. In these modes, the host system must output a DCLK that is 4× the data rate.

FPP configuration of Arria GX devices can be performed using an intelligent host, such as a MAX II device, a microprocessor, or an Altera enhanced configuration device.

### FPP Configuration Using a MAX II Device as an External Host

FPP configuration using compression and an external host provides the fastest method to configure Arria GX devices. In the FPP configuration scheme, a MAX II device can be used as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Arria GX device. Configuration data can be stored in RBF, HEX, or TTF format. When using the MAX II devices as an intelligent host, a design that controls the configuration process, such as fetching the data from flash memory and sending it to the device, must be stored in the MAX II device.

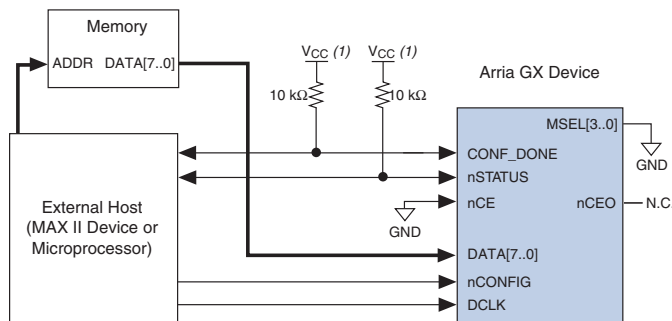


If you are using the Arria GX decompression feature, the external host must be able to send a DCLK frequency that is 4× the data rate.

The 4× DCLK signal does not require an additional pin and is sent on the DCLK pin. The maximum DCLK frequency is 100 MHz, which results in a maximum data rate of 200 Mbps. If you are not using the Arria GX decompression feature, the data rate is 8× the DCLK frequency.

Figure 11-3 shows the configuration interface connections between the Arria GX device and a MAX II device for single device configuration.

**Figure 11–3. Single Device FPP Configuration Using an External Host**



*Note to Figure 11–3:*

- (1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for the device.  $V_{CC}$  should be high enough to meet the  $V_{IH}$  specification of the I/O on the device and the external host.

Upon power-up, the Arria GX devices go through a power-on reset (POR). The POR delay is dependent on the PORSEL pin setting: when PORSEL is driven low, the POR time is approximately 100 ms; when PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device resets, holds nSTATUS low, and tri-states all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO\_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins have weak pull-up resistors, which are on (after POR) before and during configuration. If nIO\_pullup is driven high, the weak pull-up resistors are disabled.



The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*.

The configuration cycle consists of three stages: reset, configuration, and initialization. While `nCONFIG` or `nSTATUS` are low, the device is in the reset stage. To initiate configuration, the MAX II device must drive the `nCONFIG` pin from low to high.



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

When `nCONFIG` goes high, the device comes out of reset and releases the open-drain `nSTATUS` pin, which is then pulled high by an external 10-k $\Omega$  pull-up resistor. Once `nSTATUS` is released, the device is ready to receive configuration data and the configuration stage begins. When `nSTATUS` is pulled high, the MAX II device places the configuration data, one byte at a time, on the `DATA [7 . . 0]` pins.



Arria GX devices receive configuration data on the `DATA [7 . . 0]` pins and the clock is received on the `DCLK` pin. Data is latched into the device on the rising edge of `DCLK`. If you are using the Arria GX decompression feature, configuration data is latched on the rising edge of every fourth `DCLK` cycle. After the configuration data is latched in, it is processed during the following three `DCLK` cycles.

Data is continuously clocked into the target device until `CONF_DONE` goes high. The `CONF_DONE` pin goes high one byte early in parallel configuration (FPP and PPA) modes. The last byte is required for serial configuration (AS and PS) modes. After the device has received the next to last byte of the configuration data successfully, it releases the open-drain `CONF_DONE` pin, which is pulled high by an external 10-k $\Omega$  pull-up resistor. A low-to-high transition on `CONF_DONE` indicates configuration is complete and initialization of the device can begin. The `CONF_DONE` pin must have an external 10-k $\Omega$  pull-up resistor in order for the device to initialize.

In Arria GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional `CLKUSR` pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Arria GX device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving `DCLK` to the device after configuration is complete does not affect device operation.

You can also synchronize initialization of multiple devices or to delay initialization with the `CLKUSR` option. The **Enable user-supplied start-up clock (CLKUSR)** option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on `CLKUSR` does not affect the configuration process. The `CONF_DONE` pin goes high one byte early in parallel configuration (FPP and PPA) modes. The last byte is required for serial configuration (AS and PS) modes. After the `CONF_DONE` pin transitions high, `CLKUSR` is enabled after the time specified as  $t_{CD2CU}$ . After this time period elapses, Arria GX devices require 299 clock cycles to initialize properly and enter user mode. Arria GX devices support a `CLKUSR`  $f_{MAX}$  of 100 MHz.

An optional `INIT_DONE` pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. This **Enable INIT\_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the `INIT_DONE` pin is used, it is high because of an external 10-k $\Omega$  pull-up resistor 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 goes low. When initialization is complete, the `INIT_DONE` pin is released and pulled high. The MAX II device must be able to detect this low-to-high transition, which signals the device has entered user mode. When initialization is complete, the device enters user mode. In user-mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

To ensure `DCLK` and `DATA [7 . . 0]` are not left floating at the end of configuration, the MAX II device must drive them either high or low, whichever is convenient on your board. The `DATA [7 . . 0]` pins are available as user I/O pins after configuration. When you select the FPP scheme in the Quartus II software, as a default, these I/O pins are tri-stated in user mode. To change this default option in the Quartus II software, select the **Pins** tab of the **Device & Pin Options** dialog box.

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



If you are using the Arria GX decompression feature and need to stop `DCLK`, it can only be stopped three clock cycles after the last data byte was latched into the Arria GX device.

By stopping `DCLK`, the configuration circuit allows enough clock cycles to process the last byte of latched configuration data. When the clock restarts, the MAX II device must provide data on the `DATA [7 . . 0]` pins prior to sending the first `DCLK` rising edge.

If an error occurs during configuration, the device drives its `nSTATUS` pin low, resetting itself internally. The low signal on the `nSTATUS` pin also alerts the MAX II device that there is an error. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box) is turned on, the device releases `nSTATUS` after a reset time-out period (maximum of 100  $\mu$ s). After `nSTATUS` is released and pulled high by a pull-up resistor, the MAX II device can try to reconfigure the target device.



without needing to pulse `nCONFIG` low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2  $\mu$ s) on `nCONFIG` to restart the configuration process.

The MAX II device can also monitor the `CONF_DONE` and `INIT_DONE` pins to ensure successful configuration. The `CONF_DONE` pin must be monitored by the MAX II device to detect errors and determine when programming completes. If all configuration data is sent, but the `CONF_DONE` or `INIT_DONE` signals have not gone high, the MAX II device will reconfigure the target device.

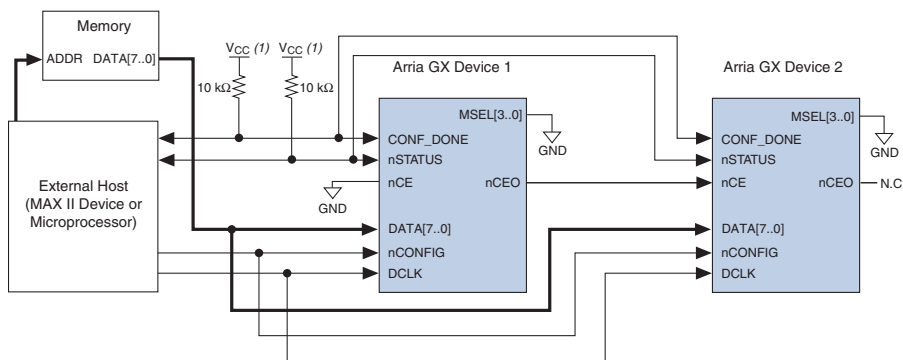


If the optional `CLKUSR` pin is used and `nCONFIG` is pulled low to restart configuration during device initialization, you need to ensure `CLKUSR` continues toggling during the time `nSTATUS` is low (maximum of 100  $\mu$ s).

When the device is in user-mode, initiating a reconfiguration is done by transitioning the `nCONFIG` pin low-to-high. The `nCONFIG` pin should be low for at least 2  $\mu$ s. When `nCONFIG` is pulled low, the device also pulls `nSTATUS` and `CONF_DONE` low and all I/O pins are tri-stated. Once `nCONFIG` returns to a logic high level and `nSTATUS` is released by the device, reconfiguration begins.

Figure 11–4 shows how to configure multiple devices using a MAX II device. This circuit is similar to the FPP configuration circuit for a single device, except the Arria GX devices are cascaded for multi-device configuration.

**Figure 11–4. Multi-Device FPP Configuration Using an External Host**



**Note to Figure 11–4:**

- (1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.  $V_{CC}$  should be high enough to meet the  $V_{IH}$  specification of the I/O standard on the device and the external host.

In multi-device FPP configuration, 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. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the MAX II device. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA [7..0], and CONF\_DONE) are connected to every device in the chain. The configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF\_DONE pins are tied together, all devices initialize and enter user mode at the same time.

All nSTATUS and CONF\_DONE pins are tied together and if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices release their nSTATUS pins after a reset time-out period (maximum of 100  $\mu$ s). After all nSTATUS pins are released and pulled high, the MAX II device can try to reconfigure the chain without pulsing nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2  $\mu$ s) on nCONFIG to restart the configuration process.

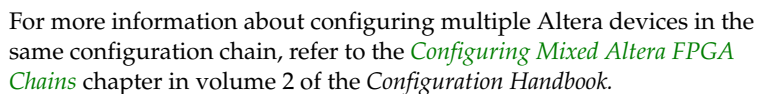
In a multi-device FPP configuration chain, all Arria GX devices in the chain must either enable or disable the decompression feature. You can not selectively enable the decompression feature for each device in the chain because of the DATA and DCLK relationship.

If a system has multiple devices that contain the same configuration data, tie all device nCE inputs to GND, and leave nCEO pins floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA [7..0], and CONF\_DONE) are connected to every device in the chain. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices start and complete configuration at the same time. [Figure 11–5](#) shows multi-device FPP configuration when both Arria GX devices are receiving the same configuration data.



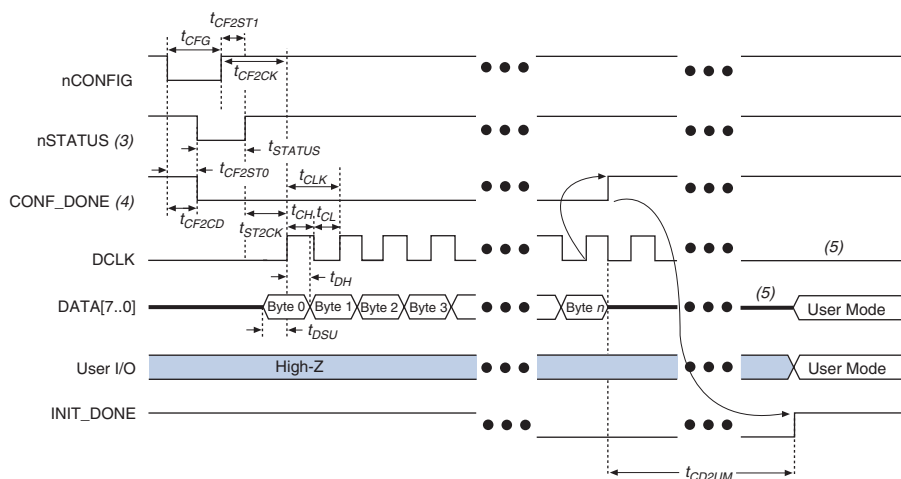
- (1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.  $V_{CC}$  should be high enough to meet the  $V_{IH}$  specification of the I/O on the device and the external host.
- (2) The  $nCE0$  pins of both Arria GX devices are left unconnected when configuring the same configuration data into multiple devices.

You can use a single configuration chain to configure Arria GX devices with other Altera devices that support FPP configuration, such as Stratix® devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, tie all of the device CONF\_DONE and nSTATUS pins together.



### FPP Configuration Timing

Figure 11–6 shows the timing waveform for FPP configuration when using a MAX II device as an external host. This waveform shows the timing when the decompression feature is not enabled.

**Figure 11–6. FPP Configuration Timing Waveform** Notes (1), (2)**Notes to Figure 11–6:**

- (1) This timing waveform should be used when the decompression feature is not used.
- (2) 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.
- (3) Upon power-up, the Arria GX device holds nSTATUS low for the time of the POR delay.
- (4) Upon power-up, before and during configuration, CONF\_DONE is low.
- (5) DCLK should not be left floating after configuration. It should be driven high or low, whichever is more convenient.
- (6) DATA[7..0] are available as user I/O pins after configuration and the state of these pins depends on the dual-purpose pin settings.

Table 11–8 defines the timing parameters for Arria GX devices for FPP configuration when the decompression feature is not enabled.

**Table 11–8. FPP Timing Parameters for Arria GX Devices (Part 1 of 2)** Notes (1), (2)

Symbol	Parameter	Min	Max	Units
$t_{CF2CD}$	nCONFIG low to CONF_DONE low		800	ns
$t_{CF2ST0}$	nCONFIG low to nSTATUS low		800	ns
$t_{CFG}$	nCONFIG low pulse width	2		$\mu$ s
$t_{STATUS}$	nSTATUS low pulse width	10	100 (3)	$\mu$ s
$t_{CF2ST1}$	nCONFIG high to nSTATUS high		100 (3)	$\mu$ s
$t_{CF2CK}$	nCONFIG high to first rising edge on DCLK	100		$\mu$ s
$t_{ST2CK}$	nSTATUS high to first rising edge of DCLK	2		$\mu$ s
$t_{DSU}$	Data setup time before rising edge on DCLK	5		ns

**Table 11–8. FPP Timing Parameters for Arria GX Devices (Part 2 of 2)** *Notes (1), (2)*

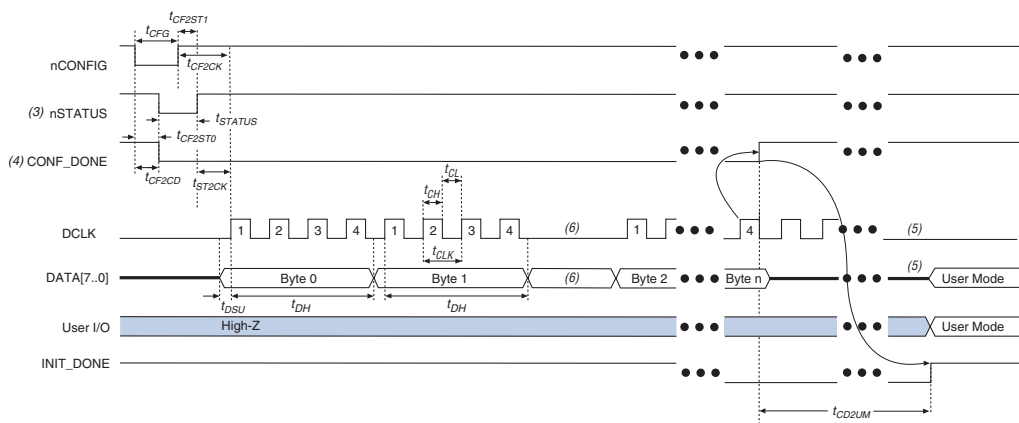
Symbol	Parameter	Min	Max	Units
$t_{DH}$	Data hold time after rising edge on DCLK	0		ns
$t_{CH}$	DCLK high time	4		ns
$t_{CL}$	DCLK low time	4		ns
$t_{CLK}$	DCLK period	10		ns
$f_{MAX}$	DCLK frequency		100	MHz
$t_R$	Input rise time		40	ns
$t_F$	Input fall time		40	ns
$t_{CD2UM}$	CONF_DONE high to user mode (4)	20	100	$\mu$ s
$t_{CD2CU}$	CONF_DONE high to CLKUSR enabled	$4 \times$ maximum DCLK period		
$t_{CD2UMC}$	CONF_DONE high to user mode with <b>CLKUSR</b> option on	$t_{CD2CU} + (299 \times \text{CLKUSR period})$		

**Notes to Table 11–8:**

- (1) This information is preliminary.
- (2) These timing parameters should be used when the decompression feature is not used.
- (3) This value is obtainable if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
- (4) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device.

Figure 11–7 shows the timing waveform for FPP configuration when using a MAX II device as an external host. This waveform shows the timing when the decompression feature is enabled.

**Figure 11–7. FPP Configuration Timing Waveform With Decompression Feature Enabled** Notes (1), (2)



**Notes to Figure 11–7:**

- (1) This timing waveform should be used when the decompression feature is used.
- (2) 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.
- (3) Upon power-up, the Arria GX device holds nSTATUS low for the time of the POR delay.
- (4) Upon power-up, before and during configuration, CONF\_DONE is low.
- (5) DCLK should not be left floating after configuration. It should be driven high or low, whichever is more convenient.
- (6) DATA [7 . . 0] are available as user I/O pins after configuration and the state of these pins depends on the dual-purpose pin settings. If needed, DCLK can be paused by holding it low. When DCLK restarts, the external host must provide data on the DATA [7 . . 0] pins prior to sending the first DCLK rising edge.

Table 11–9 defines the timing parameters for Arria GX devices for FPP configuration when the decompression feature is enabled.

<b>Table 11–9. FPP Timing Parameters for Arria GX Devices With Decompression Feature Enabled</b> <i>Notes (1), (2)</i>				
<b>Symbol</b>	<b>Parameter</b>	<b>Min</b>	<b>Max</b>	<b>Units</b>
$t_{CF2CD}$	nCONFIG low to CONF_DONE low		800	ns
$t_{CF2ST0}$	nCONFIG low to nSTATUS low		800	ns
$t_{CFG}$	nCONFIG low pulse width	2		μs
$t_{STATUS}$	nSTATUS low pulse width	10	100 (3)	μs
$t_{CF2ST1}$	nCONFIG high to nSTATUS high		100 (3)	μs
$t_{CF2CK}$	nCONFIG high to first rising edge on DCLK	100		μs
$t_{ST2CK}$	nSTATUS high to first rising edge of DCLK	2		μs
$t_{DSU}$	Data setup time before rising edge on DCLK	5		ns
$t_{DH}$	Data hold time after rising edge on DCLK	30		ns
$t_{CH}$	DCLK high time	4		ns
$t_{CL}$	DCLK low time	4		ns
$t_{CLK}$	DCLK period	10		ns
$f_{MAX}$	DCLK frequency		100	MHz
$t_{DATA}$	Data rate		200	Mbps
$t_R$	Input rise time		40	ns
$t_F$	Input fall time		40	ns
$t_{CD2UM}$	CONF_DONE high to user mode (4)	20	100	μs
$t_{CD2CU}$	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period		
$t_{CD2UMC}$	CONF_DONE high to user mode with <b>CLKUSR</b> option on	$t_{CD2CU} + (299 \times \text{CLKUSR period})$		

**Notes to Table 11–9:**

- (1) This information is preliminary.
- (2) These timing parameters should be used when the decompression feature is used.
- (3) This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
- (4) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device.



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

## FPP Configuration Using a Microprocessor

In the FPP configuration scheme, a microprocessor can control the transfer of configuration data from a storage device, such as flash memory, to the target Arria GX device.



All information in “FPP Configuration Using a MAX II Device as an External Host” on page 11–13 is also applicable when using a microprocessor as an external host. Refer to that section for all configuration and timing information.

## FPP Configuration Using an Enhanced Configuration Device

In the FPP configuration scheme, an enhanced configuration device sends a byte of configuration data every DCLK cycle to the Arria GX device. Configuration data is stored in the configuration device.



When configuring your Arria GX device using FPP mode and an enhanced configuration device, the enhanced configuration device decompression feature is available while the Arria GX decompression feature is not.

Figure 11–8 shows the configuration interface connections between a Arria GX device and the enhanced configuration device for single device configuration.



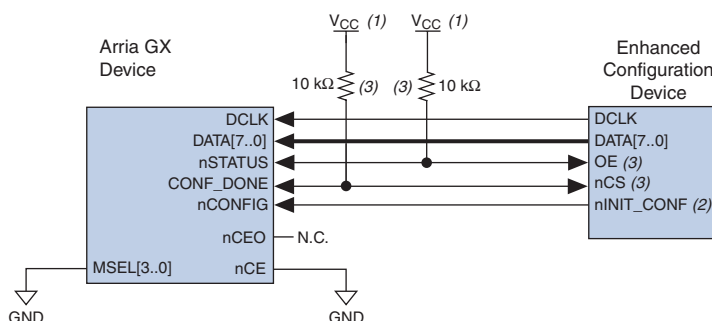
The figures in this chapter only show the configuration-related pins and the configuration pin connections between the configuration device and the device.



For more information on the enhanced configuration device and flash interface pins, such as PGM[2 . . 0], EXCLK, PORSEL, A[20 . . 0], and DQ[15 . . 0], refer to the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* in the *Configuration Handbook*.



**Figure 11–8. Single Device FPP Configuration Using an Enhanced Configuration Device**



**Notes to Figure 11–8:**

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



The value of the internal pull-up resistors on the enhanced configuration devices can be found in the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* in the *Configuration Handbook*.

When using enhanced configuration devices, you can connect the device's `nCONFIG` pin to `nINIT_CONF` pin of the enhanced configuration device, which allows the `INIT_CONF` JTAG instruction to initiate device configuration. The `nINIT_CONF` pin does not need to be connected if its functionality is not used. If `nINIT_CONF` is not used, `nCONFIG` must be pulled to  $V_{CC}$  either directly or through a resistor. An internal pull-up resistor on the `nINIT_CONF` pin is always active in the enhanced configuration devices, which means an external pull-up resistor should not be used if `nCONFIG` is tied to `nINIT_CONF`.

Upon power-up, the Arria GX device goes through a POR. The POR delay is dependent on the `PORSEL` pin setting: when `PORSEL` is driven low, the POR time is approximately 100 ms; when `PORSEL` is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold

nSTATUS low, and tri-state all user I/O pins. The configuration device also goes through a POR delay to allow the power supply to stabilize. The POR time for enhanced configuration devices can be set to either 100 ms or 2 ms, depending on its PORSEL pin setting. If the PORSEL pin is connected to GND, the POR delay is 100 ms. If the PORSEL pin is connected to  $V_{CC}$ , the POR delay is 2 ms. 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 selecting a POR time, you need to ensure that the device completes power-up before the enhanced configuration device exits POR. Altera recommends that you use a 12-ms POR time for the Arria GX device, and use a 100-ms POR time for the enhanced configuration device.

When both devices complete POR, they release their open-drain OE or nSTATUS pin, which is then pulled high by a pull-up resistor. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO\_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors, which are on (after POR) before and during configuration. If nIO\_pullup is driven high, the weak pull-up resistors are disabled.



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

When the power supplies have reached the appropriate operating voltages, the target device senses the low-to-high transition on nCONFIG and initiates the configuration cycle. The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. The beginning of configuration can be delayed by holding the nCONFIG or nSTATUS pin low.



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

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

When `nSTATUS` is pulled high, the configuration device's OE pin also goes high and the configuration device clocks data out to the device using the Arria GX device's internal oscillator. The Arria GX devices receive configuration data on the `DATA[7..0]` pins and the clock is received on the `DCLK` pin. A byte of data is latched into the device on each rising edge of `DCLK`.

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

In Arria GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional `CLKUSR` pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Arria GX device provides itself with enough clock cycles for proper initialization. You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the `CLKUSR` option. The **Enable user-supplied start-up clock (CLKUSR)** option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on `CLKUSR` will not affect the configuration process. After all configuration data has been accepted and `CONF_DONE` goes high, `CLKUSR` will be enabled after the time specified as  $t_{CD2CU}$ . After this time period elapses, Arria GX devices require 299 clock cycles to initialize properly and enter user mode. Arria GX devices support a `CLKUSR`  $f_{MAX}$  of 100 MHz.

An optional `INIT_DONE` pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT\_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the `INIT_DONE` pin is used, it will be high due to an external 10-k $\Omega$  pull-up resistor 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. When initialization is complete, the `INIT_DONE` pin will be released and pulled high. In user-mode, the user

I/O pins will no longer have weak pull-up resistors and will function as assigned in your design. The enhanced configuration device will drive DCLK low and DATA [7 . . 0] high at the end of configuration.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. Because the nSTATUS pin is tied to OE, the configuration device will also be reset. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box) is turned on, the device will automatically initiate reconfiguration if an error occurs. The Arria GX device releases its nSTATUS pin after a reset time-out period (maximum of 100  $\mu$ s). When the nSTATUS pin is released and pulled high by a pull-up resistor, the configuration device reconfigures the chain. If this option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2  $\mu$ s to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to V<sub>CC</sub>.

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 device has not configured successfully. Enhanced configuration devices wait for 64 DCLK cycles after the last configuration bit was sent for CONF\_DONE to reach a high state. In this case, the configuration device pulls its OE pin low, which in turn drives the target device's nSTATUS pin low. If the Auto-restart configuration after error option is set in the software, the target device resets and then releases its nSTATUS pin after a reset time-out period (maximum of 100  $\mu$ s). When nSTATUS returns to a logic high level, the configuration device will try to reconfigure the device.

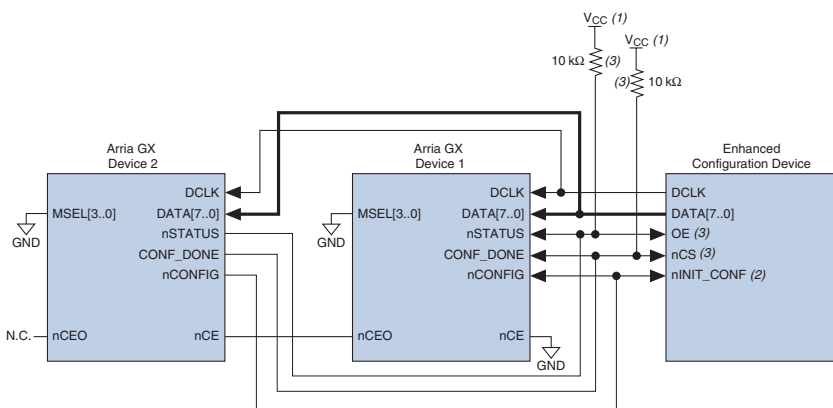
When CONF\_DONE is sensed low after configuration, the configuration device recognizes that the target device has not configured successfully. Therefore, your system should not pull CONF\_DONE low to delay initialization. Instead, you should use the CLKUSR option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain will initialize together if their CONF\_DONE pins are tied together.



If the optional CLKUSR pin is used and nCONFIG is pulled low to restart configuration during device initialization, ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 100  $\mu$ s).

When the device is in user-mode, a reconfiguration can be initiated by pulling the nCONFIG pin low. The nCONFIG pin should be low for at least 2  $\mu$ s. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF\_DONE low and all I/O pins are tri-stated. Because CONF\_DONE is

Figure 11-9 shows how to configure multiple Arria GX devices with an enhanced configuration device. This circuit is similar to the configuration device circuit for a single device, except the Arria GX devices are cascaded for multi-device configuration.



For more information on how to create configuration files for multi-device configuration chains, refer to the *Software Settings* section in volume 2 of the *Configuration Handbook*.

In multi-device FPP configuration, 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. All other configuration pins (`nCONFIG`, `nSTATUS`, `DCLK`, `DATA[7..0]`, and `CONF_DONE`) are connected to every device in the chain. Pay special attention to the configuration signals because they may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the `DCLK` and `DATA` lines are buffered for every fourth device.

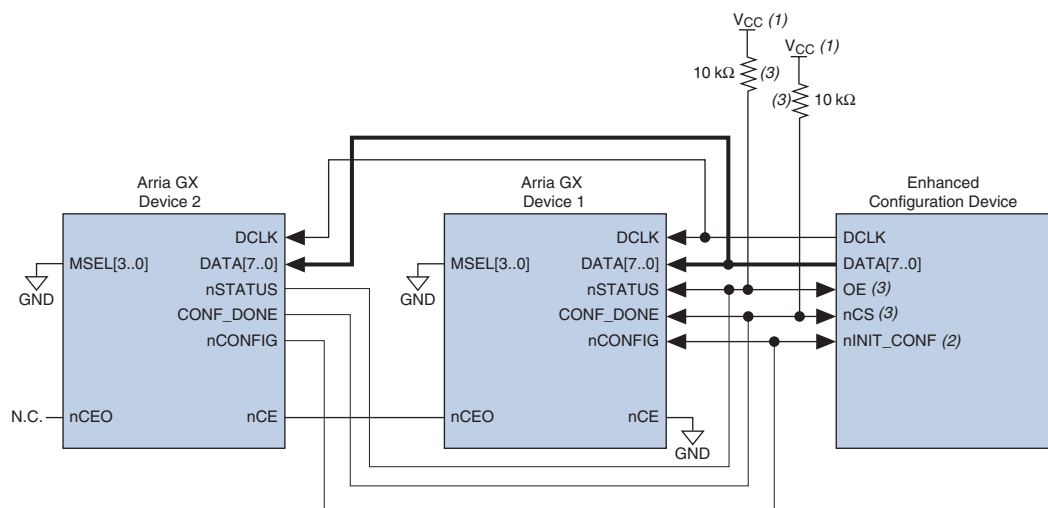
When configuring multiple devices, configuration does not begin until all devices release their `OE` or `nSTATUS` pins. Similarly, because all device `CONF_DONE` pins are tied together, all devices initialize and enter user mode at the same time.

Because all `nSTATUS` and `CONF_DONE` pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on `nSTATUS`, it resets the chain by pulling its `nSTATUS` pin low. This low signal drives the `OE` pin low on the enhanced configuration device and drives `nSTATUS` low on all devices, which causes them to enter a reset state. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices will automatically initiate reconfiguration if an error occurs. The devices will release their `nSTATUS` pins after a reset time-out period (maximum of 100  $\mu$ s). When all the `nSTATUS` pins are released and pulled high, the configuration device tries to reconfigure the chain. If the **Auto-restart configuration after error** option is turned off, the external system must monitor `nSTATUS` for errors and then pulse `nCONFIG` low for at least 2  $\mu$ s to restart configuration. The external system can pulse `nCONFIG` if `nCONFIG` is under system control rather than tied to `VCC`.

Your system may have multiple devices that contain the same configuration data. To support this configuration scheme, all device `nCE` inputs are tied to GND, while `nCEO` pins are left floating. All other configuration pins (`nCONFIG`, `nSTATUS`, `DCLK`, `DATA[7..0]`, and `CONF_DONE`) are connected to every device in the chain. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the `DCLK` and `DATA` lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. [Figure 11–10](#) shows multi-device FPP configuration when both Arria GX devices are receiving the same configuration data.

**Figure 11–10. Multiple-Device FPP Configuration Using an Enhanced Configuration Device When Both Devices Receive the Same Data**



**Notes to Figure 11–10:**

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

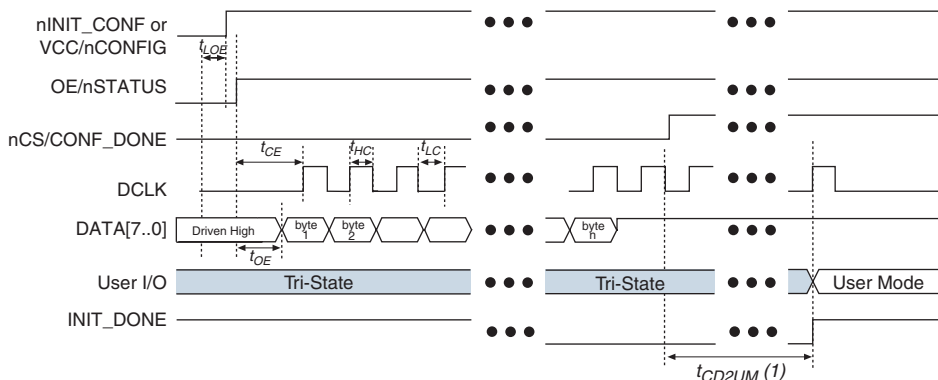
You can use a single enhanced configuration chain to configure multiple Arria GX devices with other Altera devices that support FPP configuration, such as Arria GX devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF\_DONE and nSTATUS pins must be tied together.



For more information about configuring multiple Altera devices in the same configuration chain, refer to the [Configuring Mixed Altera FPGA Chains](#) chapter in volume 2 of the *Configuration Handbook*.

Figure 11–11 shows the timing waveform for the FPP configuration scheme using an enhanced configuration device.

**Figure 11–11. Arria GX FPP Configuration Using an Enhanced Configuration Device Timing Waveform**



**Note to Figure 11–11:**

(1) The initialization clock can come from the Arria GX device's internal oscillator or the CLKUSR pin.



For timing information, refer to the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* in volume 2 of the *Configuration Handbook*.



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

## Active Serial Configuration (Serial Configuration Devices)

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



For more information on serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook*.

Serial configuration devices provide a serial interface to access configuration data. During device configuration, Arria GX devices read configuration data via the serial interface, decompresses data if necessary, and configures their SRAM cells. This scheme is referred to as the AS



configuration scheme because the device controls the configuration interface. This scheme contrasts with the PS configuration scheme, where the configuration device controls the interface.



The Arria GX decompression feature is fully available when configuring your Arria GX device using AS mode.

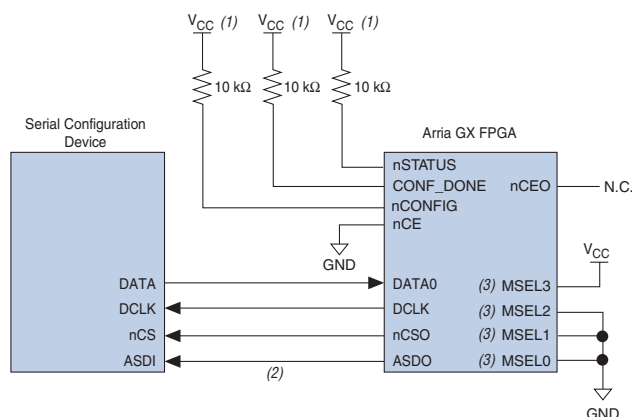
Table 11–10 shows the MSEL pin settings when using the AS configuration scheme.

<b>Table 11–10. Arria GX MSEL Pin Settings for AS Configuration Schemes</b>				
<b>Configuration Scheme</b>	<b>MSEL3</b>	<b>MSEL2</b>	<b>MSEL1</b>	<b>MSEL0</b>
Fast AS (40 MHz) (1)	1	0	0	0
Remote system upgrade fast AS (40 MHz) (1)	1	0	0	1
AS (20 MHz) (1)	1	1	0	1
Remote system upgrade AS (20 MHz) (1)	1	1	1	0

**Note to Table 11–10:**

- (1) Only the EPCS16 and EPCS64 devices support a DCLK up to 40 MHz clock; other EPCS devices support a DCLK up to 20 MHz. Refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook* for more information.

Serial configuration devices have a four-pin interface: serial clock input (DCLK), serial data output (DATA), AS data input (ASDI), and an active-low chip select (nCS). This four-pin interface connects to Arria GX device pins, as shown in Figure 11–12.

**Figure 11–12. Single Device AS Configuration****Notes to Figure 11–12:**

- (1) Connect the pull-up resistors to a 3.3-V supply.
- (2) Arria GX devices use the ASDO to ASDI path to control the configuration device.
- (3) If using an EPCS4 device, MSEL [3 : 0] should be set to **1101**. Refer to [Table 11–10](#) for more details.

Upon power-up, Arria GX devices go through a POR. The POR delay is dependent on the PORSEL pin setting. When PORSEL is driven low, the POR time is approximately 100 ms. If PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold nSTATUS and CONF\_DONE low, and tri-state all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO\_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO\_pullup is driven high, the weak pull-up resistors are disabled.



The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the [DC & Switching Characteristics](#) chapter in volume 1 of the *Arria GX Device Handbook*.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. After POR, the Arria GX devices release nSTATUS, which is pulled high by an external 10-kΩ pull-up resistor, and enters configuration mode.



To begin configuration, power the V<sub>CCINT</sub>, V<sub>CCIO</sub>, and V<sub>CCPD</sub> voltages (for the banks where the configuration and JTAG pins reside) to the appropriate voltage levels.

The serial clock (DCLK) generated by Arria GX devices controls the entire configuration cycle and provides the timing for the serial interface. Arria GX devices use an internal oscillator to generate DCLK. Using the MSEL [] pins, you can select to use either a 40- or 20-MHz oscillator.



Only the EPCS16 and EPCS64 devices support a DCLK up to 40-MHz clock; other EPCS devices support a DCLK up to 20-MHz.



Refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook* for more information.

The EPCS4 device only supports the smallest Arria GX (EP2S15) device, which is when the SOF compression is enabled. Because of its insufficient memory capacity, the EPCS1 device does not support any Arria GX devices.

Table 11–11 shows the active serial DCLK output frequencies.

<b>Table 11–11. Active Serial DCLK Output Frequency</b> <i>Note (1)</i>				
<b>Oscillator</b>	<b>Minimum</b>	<b>Typical</b>	<b>Maximum</b>	<b>Units</b>
40 MHz (2)	20	26	40	MHz
20 MHz	10	13	20	MHz

**Notes to Table 11–11:**

- (1) These values are preliminary.
- (2) Only the EPCS16 and EPCS64 devices support a DCLK up to 40-MHz clock; other EPCS devices support a DCLK up to 20-MHz. Refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook* for more information.

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

In configuration mode, Arria GX devices enable the serial configuration device by driving the nCS0 output pin low, which connects to the chip select (nCS) pin of the configuration device. Arria GX devices use the serial clock (DCLK) and serial data output (ASDO) pins to send operation commands and/or read address signals to the serial configuration device. The configuration device provides data on its serial data output (DATA) pin, which connects to the DATA0 input of the Arria GX devices.

After all configuration bits are received by the Arria GX device, it releases the open-drain `CONF_DONE` pin, which is pulled high by an external 10-k $\Omega$  resistor. Initialization begins only after the `CONF_DONE` signal reaches a logic high level. All AS configuration pins (`DATA0`, `DCLK`, `nCSO`, and `ASDO`) have weak internal pull-up resistors that are always active. After configuration, these pins are set as input tri-stated and are driven high by the weak internal pull-up resistors. The `CONF_DONE` pin must have an external 10-k $\Omega$  pull-up resistor in order for the device to initialize.

In Arria GX devices, the initialization clock source is either the 10-MHz (typical) internal oscillator (separate from the active serial internal oscillator) or the optional `CLKUSR` pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Arria GX device provides itself with enough clock cycles for proper initialization. You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the `CLKUSR` option. The **Enable user-supplied start-up clock (CLKUSR)** option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. When you **Enable the user supplied start-up clock** option, the `CLKUSR` pin is the initialization clock source. Supplying a clock on `CLKUSR` will not affect the configuration process. After all configuration data has been accepted and `CONF_DONE` goes high, `CLKUSR` is enabled after 600 ns. After this time period elapses, Arria GX devices require 299 clock cycles to initialize properly and enter user mode. Arria GX devices support a `CLKUSR`  $f_{MAX}$  of 100 MHz.

An optional `INIT_DONE` pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT\_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the `INIT_DONE` pin is used, it will be high due to an external 10-k $\Omega$  pull-up resistor 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 goes low. When initialization is complete, the `INIT_DONE` pin is released and pulled high. This low-to-high transition signals that the device has entered user mode. When initialization is complete, the device enters user mode. In user mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

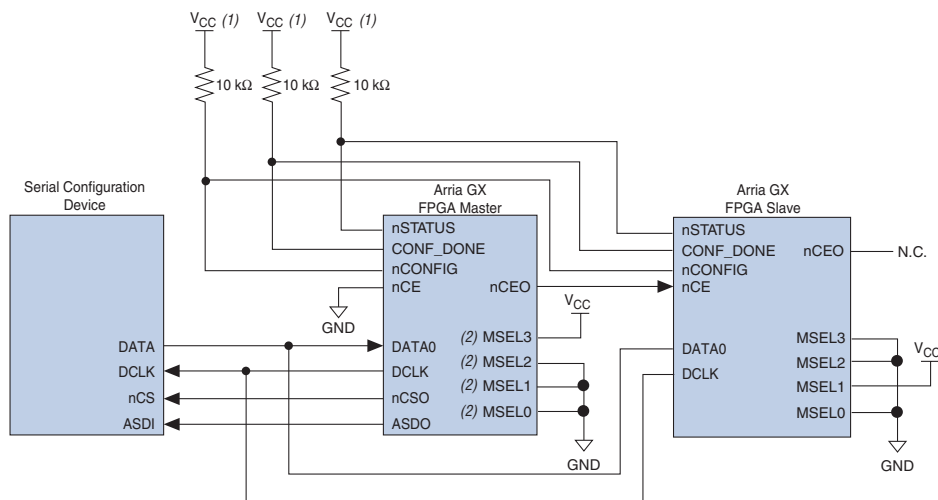
If an error occurs during configuration, Arria GX devices assert the `nSTATUS` signal low, indicating a data frame error, and the `CONF_DONE` signal stays low. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box) is turned on, the Arria GX device resets the configuration device by pulsing `nCSO`, releases `nSTATUS` after a reset

time-out period (maximum of 100  $\mu$ s), and retries configuration. If this option is turned off, the system must monitor `nSTATUS` for errors and then pulse `nCONFIG` low for at least 2  $\mu$ s to restart configuration.

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

You can configure multiple Arria GX devices using a single serial configuration device. You can cascade multiple Arria GX devices using the chip-enable (`nCE`) and chip-enable-out (`nCEO`) pins. The first device in the chain must have its `nCE` pin connected to ground. You must connect its `nCEO` pin to the `nCE` pin of the next device in the chain. When the first device captures all of its configuration data from the bitstream, it drives the `nCEO` pin low, enabling the next device in the chain. You must leave the `nCEO` pin of the last device unconnected. The `nCONFIG`, `nSTATUS`, `CONF_DONE`, `DCLK`, and `DATA0` pins of each device in the chain are connected (refer to [Figure 11–13](#)).

This first Arria GX device in the chain is the configuration master and controls configuration of the entire chain. You must connect its `MSEL` pins to select the AS configuration scheme. The remaining Arria GX devices are configuration slaves and you must connect their `MSEL` pins to select the PS configuration scheme. Any other Altera device that supports PS configuration can also be part of the chain as a configuration slave. [Figure 11–13](#) shows the pin connections for this setup.

**Figure 11–13. Multi-Device AS Configuration****Notes to Figure 11–13:**

- (1) Connect the pull-up resistors to a 3.3-V supply.
- (2) If using an EPCS4 device, MSEL[3..0] should be set to 1101. Refer to Table 11–10 on page 11–33 for more details.

As shown in Figure 11–13, the **nSTATUS** and **CONF\_DONE** pins on all target devices are connected together with external pull-up resistors. These pins are open-drain bidirectional pins on the devices. When the first device asserts **nCEO** (after receiving all of its configuration data), it releases its **CONF\_DONE** pin. But the subsequent devices in the chain keep this shared **CONF\_DONE** line low until they have received their configuration data. When all target devices in the chain have received their configuration data and have released **CONF\_DONE**, the pull-up resistor drives a high level on this line and all devices simultaneously enter initialization mode.

If an error occurs at any point during configuration, the **nSTATUS** line is driven low by the failing device. If you enable the Auto-restart configuration after error option, reconfiguration of the entire chain begins after a reset time-out period (a maximum of 100 μs). If the **Auto-restart configuration after error** 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<sub>CC</sub>**.

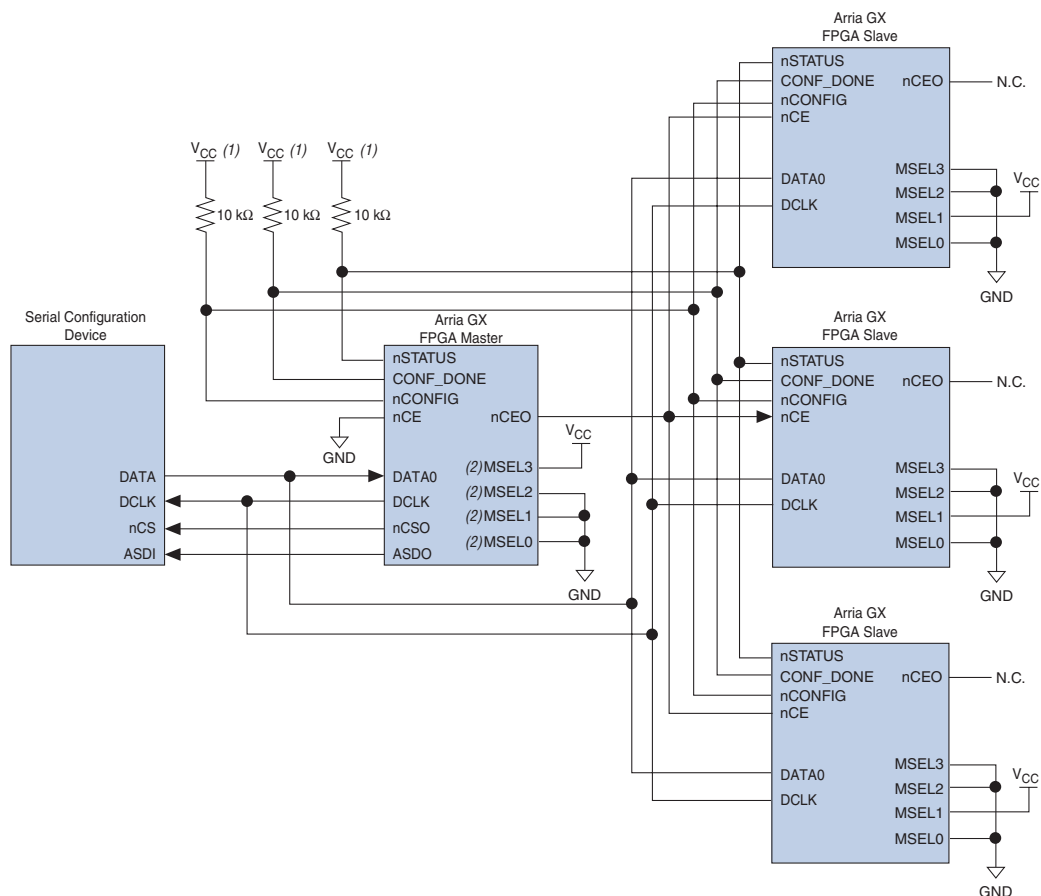


While you can cascade Arria GX devices, serial configuration devices cannot be cascaded or chained together.

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

A system may have multiple devices that contain the same configuration data. In active serial chains, this can be implemented by storing two copies of the SOF in the serial configuration device. The first copy would configure the master Arria GX device; the second copy would configure all remaining slave devices concurrently. All slave devices must be the same density and package. The setup is similar to [Figure 11-13](#), where the master is set up in active serial mode and the slave devices are set up in passive serial mode.

To configure four identical Arria GX devices with the same SOF, you could set up the chain similar to the example shown in [Figure 11-14](#). The first device is the master device and its MSEL pins should be set to select AS configuration. The other three slave devices are set up for concurrent configuration and its MSEL pins should be set to select PS configuration. The nCEO pin from the master device drives the nCE input pins on all three slave devices, and the DATA and DCLK pins connect in parallel to all four devices. During the first configuration cycle, the master device reads its configuration data from the serial configuration device while holding nCEO high. After completing its configuration cycle, the master drives nCE low and transmits the second copy of the configuration data to all three slave devices, configuring them simultaneously.

**Figure 11–14. Multi-Device AS Configuration When devices Receive the Same Data****Notes to Figure 11–14:**

- (1) Connect the pull-up resistors to a 3.3-V supply.
- (2) If using an EPCS4 device, MSEL[3..0] should be set to **1101**. Refer to [Table 11–10](#) on page 11–33 for more details.



## Estimating Active Serial Configuration Time

Active serial configuration time is dominated by the time it takes to transfer data from the serial configuration device to the Arria GX device. This serial interface is clocked by the Arria GX DCLK output (generated from an internal oscillator). As listed in [Table 11–11 on page 11–35](#), the DCLK minimum frequency when choosing to use the 40-MHz oscillator is 20 MHz (50 ns). Therefore, the maximum configuration time estimate for an EP2S15 device (5 MBits of uncompressed data) is:

RBF Size (minimum DCLK period / 1 bit per DCLK cycle) = estimated maximum configuration time

$$5 \text{ Mbits} \times (50 \text{ ns} / 1 \text{ bit}) = 250 \text{ ms}$$

To estimate the typical configuration time, use the typical DCLK period as listed in [Table 11–11](#). With a typical DCLK period of 38.46 ns, the typical configuration time is 192 ms. Enabling compression reduces the amount of configuration data that is transmitted to the Arria GX device, which also reduces configuration time. On average, compression reduces configuration time by 50%.

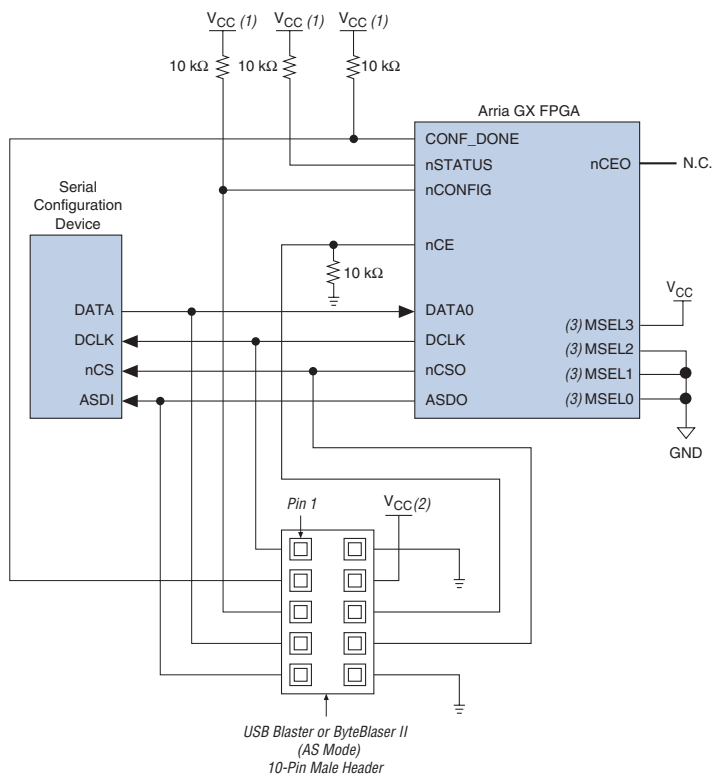
## Programming Serial Configuration Devices

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

You can perform in-system programming of serial configuration devices via the AS programming interface. During in-system programming, the download cable disables device access to the AS interface by driving the nCE pin high. Arria GX devices are also held in reset by a low level on nCONFIG. After programming is complete, the download cable releases nCE and nCONFIG, allowing the pull-down and pull-up resistors to drive GND and V<sub>CC</sub>, respectively. [Figure 11–15](#) shows the download cable connections to the serial configuration device.



For more information about the USB Blaster download cable, refer to the [USB-Blaster USB Port Download Cable User Guide](#). For more information about the ByteBlaster II cable, refer to the [ByteBlaster II Download Cable User Guide](#).

**Figure 11–15. In-System Programming of Serial Configuration Devices****Notes to Figure 11–15:**

- (1) Connect these pull-up resistors to 3.3-V supply.
- (2) Power up the ByteBlaster II cable's  $V_{CC}$  with a 3.3-V supply.
- (3) If using an EPCS4 device, MSEL [3 . . 0] should be set to **1101**. Refer to [Table 11–10 on page 11–33](#) for more details.

You can program serial configuration devices with the Quartus II software with the Altera programming hardware and the appropriate configuration device programming adapter. The EPCS1 and EPCS4 devices are offered in an eight-pin small outline integrated circuit (SOIC) package.

In production environments, serial configuration devices can be programmed using multiple methods. Altera programming hardware or other third-party programming hardware can be used to program blank serial configuration devices before they are mounted onto printed circuit

boards (PCBs). Alternatively, you can use an on-board microprocessor to program the serial configuration device in-system using C-based software drivers provided by Altera.

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



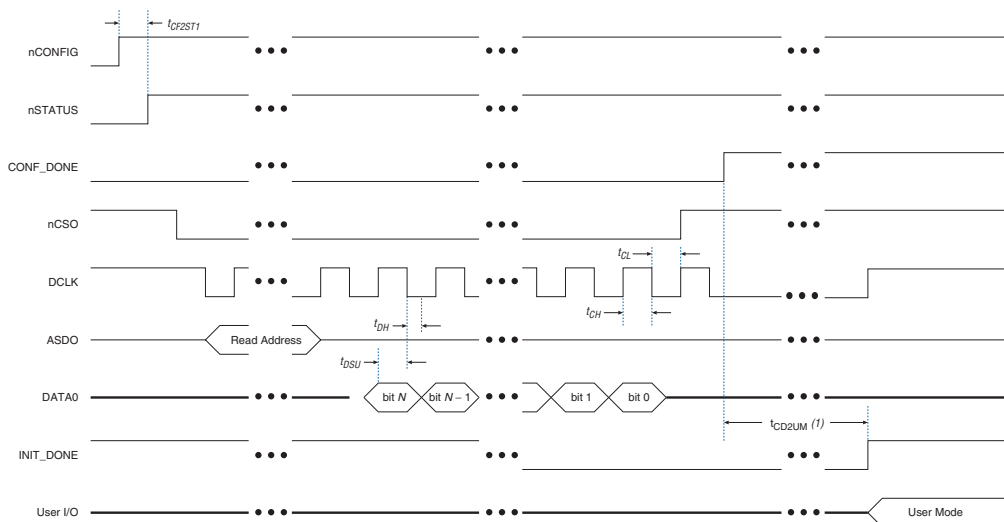
For more information about SRunner, refer to *AN 418: SRunner: An Embedded Solution for Serial Configuration* and the source code on the Altera web site at [www.altera.com](http://www.altera.com).



For more information on programming serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook*.

Figure 11–16 shows the timing waveform for the AS configuration scheme using a serial configuration device.

**Figure 11–16. AS Configuration Timing**



**Note to Figure 11–16:**

- (1) The initialization clock can come from the Arria GX device's internal oscillator or the CLKUSR pin.

Table 11–12 shows the AS timing parameters for Arria GX devices.

<b>Table 11–12. AS Timing Parameters for Arria GX Devices</b>					
<b>Symbol</b>	<b>Parameter</b>	<b>Condition</b>	<b>Minimum</b>	<b>Typical</b>	<b>Maximum</b>
$t_{CF2ST1}$	nCONFIG high to nSTATUS high				100
$t_{DSU}$	Data setup time before falling edge on DCLK		7		
$t_{DH}$	Data hold time after falling edge on DCLK		0		
$t_{CH}$	DCLK high time		10		
$t_{CL}$	DCLK low time		10		
$t_{CD2UM}$	CONF_DONE high to user mode		20		100

## Passive Serial Configuration

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



The Arria GX decompression feature is fully available when configuring your Arria GX device using PS mode.

Table 11–13 shows the MSEL pin settings when using the PS configuration scheme.

<b>Table 11–13. Arria GX MSEL Pin Settings for PS Configuration Schemes</b>				
<b>Configuration Scheme</b>	<b>MSEL3</b>	<b>MSEL2</b>	<b>MSEL1</b>	<b>MSEL0</b>
PS	0	0	1	0
PS when using Remote System Upgrade (1)	0	1	1	0

**Note to Table 11–13:**

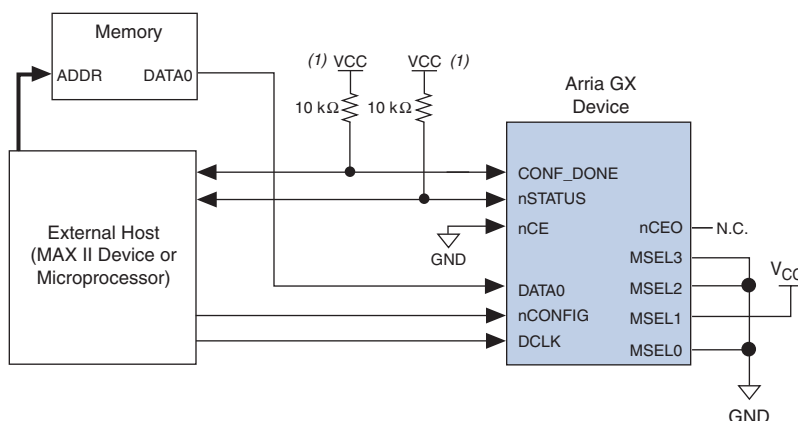
- (1) This scheme requires that you drive the RUNLÜ pin to specify either remote update or local update. For more information about remote system upgrade in Arria GX devices, refer to the *Remote System Upgrades with Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

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

In the PS configuration scheme, a MAX II device can be used as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Arria GX device. Configuration data can be stored in RBF, HEX, or TTF format.

Figure 11–17 shows the configuration interface connections between a Arria GX device and a MAX II device for single device configuration.

**Figure 11–17. Single Device PS Configuration Using an External Host**



**Note to Figure 11–17:**

- (1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the device.  $V_{CC}$  should be high enough to meet the  $V_{IH}$  specification of the I/O on the device and the external host.

Upon power-up, Arria GX devices go through a POR. The POR delay is dependent on the PORSEL pin setting; when PORSEL is driven low, the POR time is approximately 100 ms; when PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device resets, holds nSTATUS low, and tri-states all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO\_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO\_pullup is driven high, the weak pull-up resistors are disabled.



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

The configuration cycle consists of three stages: reset, configuration, and initialization. While `nCONFIG` or `nSTATUS` are low, the device is in reset. To initiate configuration, the MAX II device must generate a low-to-high transition on the `nCONFIG` pin.



`VCCINT`, `VCCIO`, and `VCCPD` of the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When `nCONFIG` goes high, the device comes out of reset and releases the open-drain `nSTATUS` pin, which is then pulled high by an external 10-k $\Omega$  pull-up resistor. Once `nSTATUS` is released, the device is ready to receive configuration data and the configuration stage begins. When `nSTATUS` is pulled high, the MAX II device should place the configuration data, one bit at a time, on the `DATA0` pin. If you are using configuration data in RBF, HEX, or TTF format, you must send the least significant bit (LSB) of each data byte first. For example, if the RBF contains the byte sequence 02 1B EE 01 FA, the serial bitstream you should transmit to the device is 0100-0000 1101-1000 0111-0111 1000-0000 0101-1111.

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

In Arria GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional `CLKUSR` pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Arria GX device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving `DCLK` to the device after configuration is complete does not affect device operation.

You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the `CLKUSR` option. The **Enable user-supplied start-up clock (CLKUSR)** option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on `CLKUSR` will not affect the configuration process. After all configuration data has been accepted and `CONF_DONE`

goes high, CLKUSR will be enabled after the time specified as  $t_{CD2CU}$ . After this time period elapses, Arria GX devices require 299 clock cycles to initialize properly and enter user mode. Arria GX devices support a CLKUSR  $f_{MAX}$  of 100 MHz.

An optional INIT\_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT\_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT\_DONE pin is used, it will be high due to an external 10-k $\Omega$  pull-up resistor 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. When initialization is complete, the INIT\_DONE pin will be released and pulled high. The MAX II device must be able to detect this low-to-high transition which signals the device has entered user mode. When initialization is complete, the device enters user mode. In user-mode, the user I/O pins will no longer have weak pull-up resistors and will function as assigned in your design.

To ensure DCLK and DATA0 are not left floating at the end of configuration, the MAX II device must drive them either high or low, whichever is convenient on your board. The DATA[0] pin is available as a user I/O pin after configuration. When the PS scheme is chosen in the Quartus II software, as a default, this I/O pin is tri-stated in user mode and should be driven by the MAX II device. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box.

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

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the MAX II device that there is an error. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box) is turned on, the Arria GX device releases nSTATUS after a reset time-out period (maximum of 100  $\mu$ s). After nSTATUS is released and pulled high by a pull-up resistor, the MAX II device can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2  $\mu$ s) on nCONFIG to restart the configuration process.

The MAX II device can also monitor the `CONF_DONE` and `INIT_DONE` pins to ensure successful configuration. The `CONF_DONE` pin must be monitored by the MAX II device to detect errors and determine when programming completes. If all configuration data is sent, but `CONF_DONE` or `INIT_DONE` have not gone high, the MAX II device must reconfigure the target device.

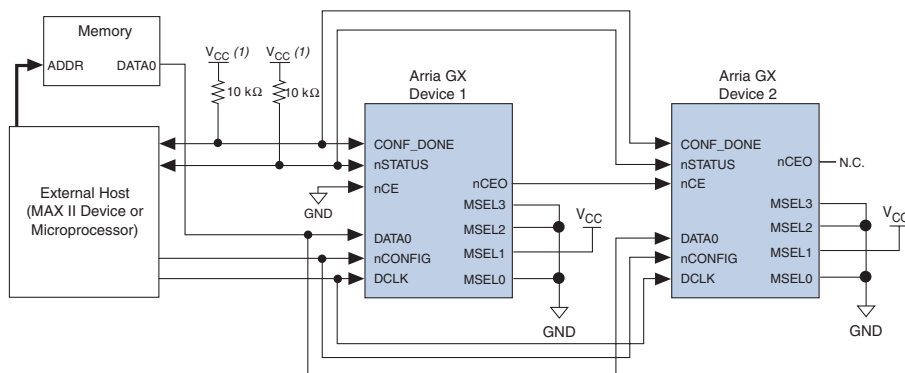


If the optional `CLKUSR` pin is being used and `nCONFIG` is pulled low to restart configuration during device initialization, you need to ensure that `CLKUSR` continues toggling during the time `nSTATUS` is low (maximum of 100  $\mu$ s).

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

Figure 11-18 shows how to configure multiple devices using a MAX II device. This circuit is similar to the PS configuration circuit for a single device, except Arria GX devices are cascaded for multi-device configuration.

**Figure 11-18. Multi-Device PS Configuration Using an External Host**



**Note to Figure 11-18:**

- (1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.  $V_{CC}$  should be high enough to meet the  $V_{IH}$  specification of the I/O on the device and the external host.

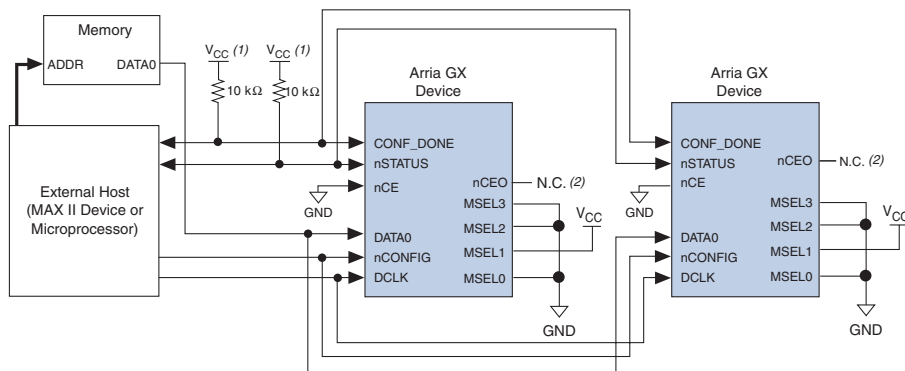


In multi-device PS configuration, the first device's `nCE` pin is connected to GND while its `nCEO` pin is connected to `nCE` of the next device in the chain. The last device's `nCE` input comes from the previous device, while its `nCEO` pin is left floating. After the first device completes configuration 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. The second device in the chain begins configuration within one clock cycle. Therefore, the transfer of data destinations is transparent to the MAX II device. All other configuration pins (`nCONFIG`, `nSTATUS`, `DCLK`, `DATA0`, and `CONF_DONE`) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the `DCLK` and `DATA` lines are buffered for every fourth device. Because all device `CONF_DONE` pins are tied together, all devices initialize and enter user mode at the same time.

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

If the **Auto-restart configuration after error** option is turned on, the devices release their `nSTATUS` pins after a reset time-out period (maximum of 100  $\mu$ s). After all `nSTATUS` pins are released and pulled high, the MAX II device can try to reconfigure the chain without needing to pulse `nCONFIG` low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2  $\mu$ s) on `nCONFIG` to restart the configuration process.

In your system, you can have multiple devices that contain the same configuration data. To support this configuration scheme, all device `nCE` inputs are tied to GND, while `nCEO` pins are left floating. All other configuration pins (`nCONFIG`, `nSTATUS`, `DCLK`, `DATA0`, and `CONF_DONE`) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the `DCLK` and `DATA` lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. [Figure 11-19](#) shows multi-device PS configuration when both Arria GX devices are receiving the same configuration data.

**Figure 11–19. Multiple-Device PS Configuration When Both Devices Receive the Same Data****Notes to Figure 11–19:**

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

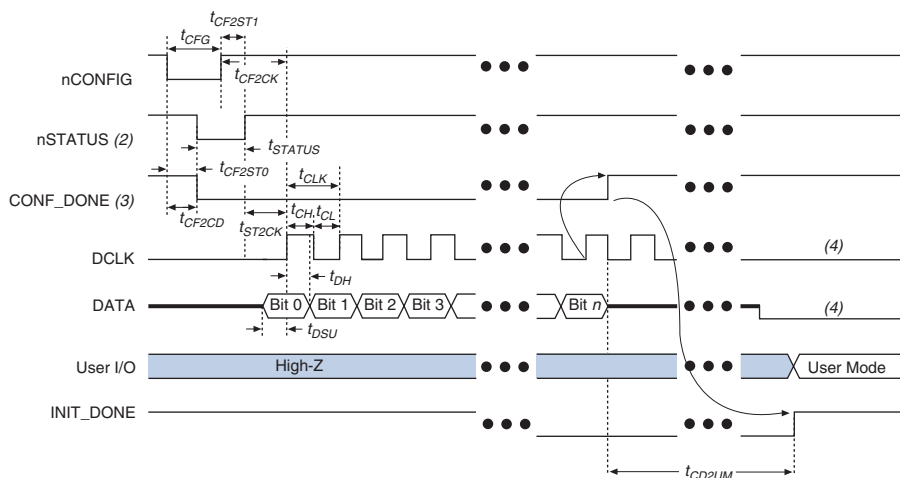
You can use a single configuration chain to configure Arria GX devices with other Altera devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device  $CONF\_DONE$  and  $nSTATUS$  pins must be tied together.



For more information about configuring multiple Altera devices in the same configuration chain, refer to the [Configuring Mixed Altera FPGA Chains](#) chapter in volume 2 of the *Configuration Handbook*.

**PS Configuration Timing**

Figure 11–20 shows the timing waveform for PS configuration when using a MAX II device as an external host.

**Figure 11–20. PS Configuration Timing Waveform** *Note (1)***Notes to Figure 11–20:**

- (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 Arria GX 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 more convenient. DATA [0] is available as a user I/O pin after configuration and the state of this pin depends on the dual-purpose pin settings.

Table 11–14 defines the timing parameters for Arria GX devices for PS configuration.

**Table 11–14. PS Timing Parameters for Arria GX Devices (Part 1 of 2)** *Note (1)*

Symbol	Parameter	Min	Max	Units
$t_{CF2CD}$	nCONFIG low to CONF_DONE low		800	ns
$t_{CF2ST0}$	nCONFIG low to nSTATUS low		800	ns
$t_{CFG}$	nCONFIG low pulse width	2		$\mu$ s
$t_{STATUS}$	nSTATUS low pulse width	10	100 (2)	$\mu$ s
$t_{CF2ST1}$	nCONFIG high to nSTATUS high		100 (2)	$\mu$ s
$t_{CF2CK}$	nCONFIG high to first rising edge on DCLK	100		$\mu$ s
$t_{ST2CK}$	nSTATUS high to first rising edge of DCLK	2		$\mu$ s
$t_{DSU}$	Data setup time before rising edge on DCLK	5		ns
$t_{DH}$	Data hold time after rising edge on DCLK	0		ns

**Table 11–14. PS Timing Parameters for Arria GX Devices (Part 2 of 2)** *Note (1)*

Symbol	Parameter	Min	Max	Units
$t_{CH}$	DCLK high time	4		ns
$t_{CL}$	DCLK low time	4		ns
$t_{CLK}$	DCLK period	10		ns
$f_{MAX}$	DCLK frequency		100	MHz
$t_R$	Input rise time		40	ns
$t_F$	Input fall time		40	ns
$t_{CD2UM}$	CONF_DONE high to user mode (3)	20	100	$\mu$ s
$t_{CD2CU}$	CONF_DONE high to CLKUSR enabled	$4 \times$ maximum DCLK period		
$t_{CD2UMC}$	CONF_DONE high to user mode with CLKUSR option on	$t_{CD2CU} + (299 \times \text{CLKUSR period})$		

**Notes to Table 11–14:**

- (1) This information is preliminary.
- (2) This value is applicable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
- (3) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting the device.



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

An example PS design that uses a MAX II device as the external host for configuration will be available when devices are available.

## PS Configuration Using a Microprocessor


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



All information in the “PS Configuration Using a MAX II Device as an External Host” on page 11–45 section is also applicable when using a microprocessor as an external host. Refer to that section for all configuration and timing information.

## PS Configuration Using a Configuration Device

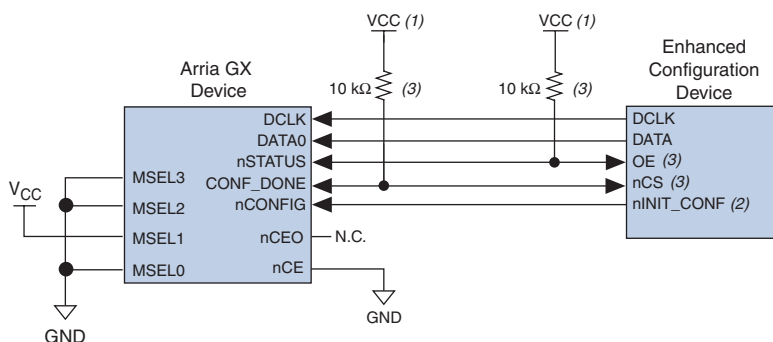
You can use an Altera configuration device, such as an enhanced configuration device or EPC2 device, to configure Arria GX devices using a serial configuration bitstream. Configuration data is stored in the configuration device. [Figure 11–21](#) shows the configuration interface connections between an Arria GX device and a configuration device.

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



For more information on the enhanced configuration device and flash interface pins (such as PGM[2..0], EXCLK, PORSEL, A[20..0], and DQ[15..0]), refer to the [Enhanced Configuration Devices \(EPC4, EPC8, & EPC16\) Data Sheet](#) chapter in volume 2 of the *Configuration Handbook*.

**Figure 11–21. Single Device PS Configuration Using an Enhanced Configuration Device**



### Notes to [Figure 11–21](#):

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



The value of the internal pull-up resistors on the enhanced configuration devices and EPC2 devices can be found in the Operating Conditions table of the [Enhanced Configuration Devices \(EPC4, EPC8, & EPC16\) Data Sheet](#) chapter or the [Configuration Devices for SRAM-based LUT Devices Data Sheet](#) chapter in volume 2 of the *Configuration Handbook*.

When using enhanced configuration devices or EPC2 devices, `nCONFIG` of the device can be connected to `nINIT_CONF` of the configuration device, which allows the `INIT_CONF` JTAG instruction to initiate device configuration. The `nINIT_CONF` pin does not need to be connected if its functionality is not used. An internal pull-up resistor on the `nINIT_CONF` pin is always active in enhanced configuration devices and EPC2 devices, which means an external pull-up resistor should not be used if `nCONFIG` is tied to `nINIT_CONF`.

Upon power-up, the Arria GX devices go through a POR. The POR delay is dependent on the `PORSEL` pin setting. When `PORSEL` is driven low, the POR time is approximately 100 ms. If `PORSEL` is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold `nSTATUS` low, and tri-state all user I/O pins. The configuration device also goes through a POR delay to allow the power supply to stabilize. The POR time for EPC2 devices is 200 ms (maximum). The POR time for enhanced configuration devices can be set to either 100 ms or 2 ms, depending on its `PORSEL` pin setting. If the `PORSEL` pin is connected to GND, the POR delay is 100 ms. If the `PORSEL` pin is connected to  $V_{CC}$ , the POR delay is 2 ms. 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 selecting a POR time, you need to ensure that the device completes power-up before the enhanced configuration device exits POR. Altera recommends that you choose a POR time for the Arria GX device of 12 ms, while selecting a POR time for the enhanced configuration device of 100 ms.

When both devices complete POR, they release their open-drain OE or `nSTATUS` pin, which is then pulled high by a pull-up resistor. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If `nIO_pullup` is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If `nIO_pullup` is driven high, the weak pull-up resistors are disabled.



The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*.

When the power supplies have reached the appropriate operating voltages, the target device senses the low-to-high transition on `nCONFIG` and initiates the configuration cycle. The configuration cycle consists of three stages: reset, configuration, and initialization. While `nCONFIG` or `nSTATUS` are low, the device is in reset. The beginning of configuration can be delayed by holding the `nCONFIG` or `nSTATUS` pin low.



To begin configuration, power the  $V_{CCINT}$ ,  $V_{CCIO}$ , and  $V_{CCPD}$  voltages (for the banks where the configuration and JTAG pins reside) to the appropriate voltage levels.

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

When  $nSTATUS$  is pulled high, OE of the configuration device also goes high and the configuration device clocks data out serially to the device using the Arria GX device's internal oscillator. Arria GX devices receive configuration data on the DATA0 pin and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK.

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

In Arria GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If you are using internal oscillator, the Arria GX device supplies itself with enough clock cycles for proper initialization. You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. You can turn on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF\_DONE goes high, CLKUSR will be enabled after the time specified as  $t_{CD2CU}$ . After this time period elapses, the Arria GX devices require 299 clock cycles to initialize properly and enter user mode. Arria GX devices support a CLKUSR  $f_{MAX}$  of 100 MHz.

An optional `INIT_DONE` pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT\_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If you are using the `INIT_DONE` pin, it will be high due to an external 10-k $\Omega$  pull-up resistor 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 goes low. When initialization is complete, the `INIT_DONE` pin is released and pulled high. This low-to-high transition signals that the device has entered user mode. In user-mode, the user I/O pins will no longer have weak pull-up resistors and will function as assigned in your design. Enhanced configuration devices and EPC2 devices drive `DCLK` low and `DATA0` high at the end of configuration.

If an error occurs during configuration, the device drives its `nSTATUS` pin low, resetting itself internally. Because the `nSTATUS` pin is tied to `OE`, the configuration device will also be reset. If the **Auto-restart configuration after error** option, available in the Quartus II software, from the **General** tab of the **Device & Pin Options** dialog box is turned on, the device automatically initiates reconfiguration if an error occurs. The Arria GX devices release the `nSTATUS` pin after a reset time-out period (maximum of 100  $\mu$ s). When the `nSTATUS` pin is released and pulled high by a pull-up resistor, the configuration device reconfigures the chain. If this option is turned off, the external system must monitor `nSTATUS` for errors and then pulse `nCONFIG` low for at least 2  $\mu$ s to restart configuration. The external system can pulse `nCONFIG` if `nCONFIG` is under system control rather than tied to `VCC`.

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 device has not configured successfully. Enhanced configuration devices wait for 64 `DCLK` cycles after the last configuration bit was sent for `CONF_DONE` to reach a high state. EPC2 devices wait for 16 `DCLK` cycles. In this case, the configuration device pulls its `OE` pin low, driving the target device's `nSTATUS` pin low. If the **Auto-restart configuration after error** option is set in the software, the target device resets and then releases its `nSTATUS` pin after a reset time-out period (maximum of 100  $\mu$ s). When `nSTATUS` returns to a logic high level, the configuration device tries to reconfigure the device.

When `CONF_DONE` is sensed low after configuration, the configuration device recognizes that the target device has not configured successfully. Therefore, your system should not pull `CONF_DONE` low to delay initialization. Instead, use the `CLKUSR` option to synchronize the



initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain will initialize together if their `CONF_DONE` pins are tied together.

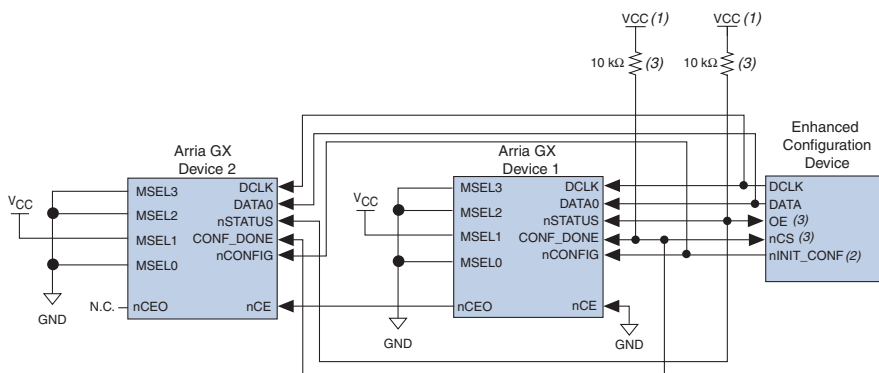


If you are using the optional `CLKUSR` pin and `nCONFIG` is pulled low to restart configuration during device initialization, you need to ensure that `CLKUSR` continues toggling during the time `nSTATUS` is low (maximum of 100  $\mu$ s).

When the device is in user-mode, pulling the `nCONFIG` pin low initiates a reconfiguration. The `nCONFIG` pin should be low for at least 2  $\mu$ s. When `nCONFIG` is pulled low, the device also pulls `nSTATUS` and `CONF_DONE` low and all I/O pins are tri-stated. Because `CONF_DONE` is pulled low, this activates the configuration device because it sees its `nCS` pin drive low. Once `nCONFIG` returns to a logic high level and `nSTATUS` is released by the device, reconfiguration begins.

Figure 11–22 shows how to configure multiple devices with an enhanced configuration device. This circuit is similar to the configuration device circuit for a single device, except Arria GX devices are cascaded for multi-device configuration.

**Figure 11–22. Multi-Device PS Configuration Using an Enhanced Configuration Device**



**Notes to Figure 11–22:**

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



Enhanced configuration devices cannot be cascaded.

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



For more information about creating configuration files for multi-device configuration chains, refer to the *Software Settings* section in volume 2 of the *Configuration Handbook*.

In multi-device PS configuration, 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, prompting the second device to begin configuration. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF\_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device.

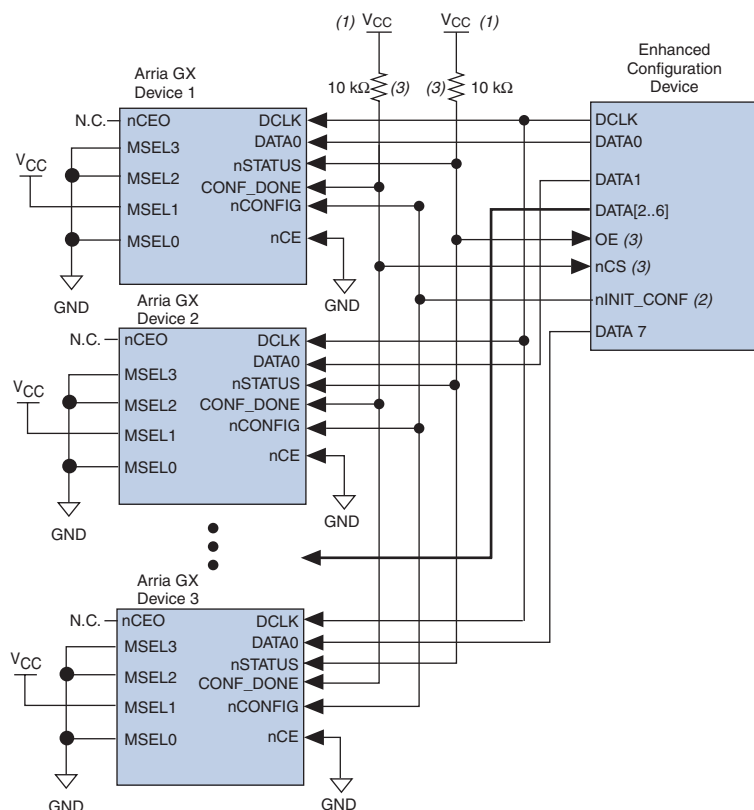
When configuring multiple devices, configuration does not begin until all devices release their OE or nSTATUS pins. Similarly, because all device CONF\_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Because all nSTATUS and CONF\_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This low signal drives the OE pin low on the enhanced configuration device and drives nSTATUS low on all devices, causing them to enter a reset state. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices will automatically initiate reconfiguration if an error occurs. The devices will release their nSTATUS pins after a reset time-out period (maximum of 100  $\mu$ s). When all the nSTATUS pins are released and pulled high, the configuration device tries to reconfigure the chain. If the **Auto-restart configuration after error** option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2  $\mu$ s to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to V<sub>CC</sub>.

The enhanced configuration devices also support parallel configuration of up to eight devices. The  $n$ -bit ( $n = 1, 2, 4, \text{ or } 8$ ) PS configuration mode allows enhanced configuration devices to concurrently configure devices or a chain of devices. In addition, these devices do not have to be the same device family or density as they can be any combination of Altera devices. An individual enhanced configuration device DATA line is available for each targeted device. Each DATA line can also feed a daisy chain of devices. [Figure 11–23](#) shows how to concurrently configure multiple devices using an enhanced configuration device.

**Figure 11–23. Concurrent PS Configuration of Multiple Devices Using an Enhanced Configuration Device**



**Notes to Figure 11–23:**

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

The Quartus II software only allows the selection of n-bit PS configuration modes, where n must be 1, 2, 4, or 8. However, you can use these modes to configure any number of devices from 1 to 8. When configuring SRAM-based devices using n-bit PS modes, use [Table 11–15](#) to select the appropriate configuration mode for the fastest configuration times.

<b>Table 11–15. Recommended Configuration Using n-Bit PS Modes</b>	
<b>Number of Devices (1)</b>	<b>Recommended Configuration Mode</b>
1	1-bit PS
2	2-bit PS
3	4-bit PS
4	4-bit PS
5	8-bit PS
6	8-bit PS
7	8-bit PS
8	8-bit PS

**Note to [Table 11–15](#):**

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

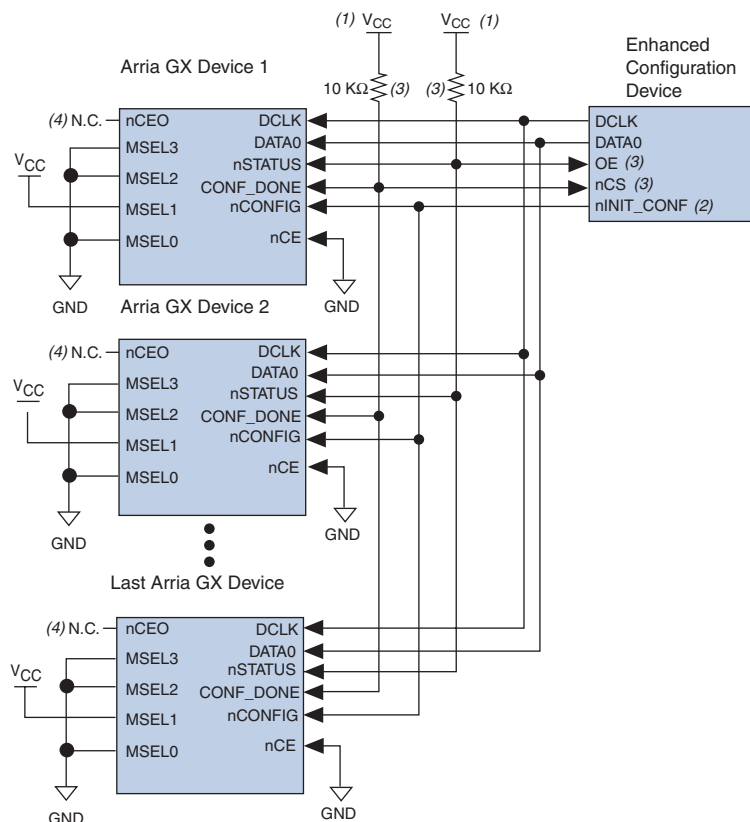
For example, if you configure three devices, you would use the 4-bit PS mode. For the DATA0, DATA1, and DATA2 lines, the corresponding SOF data is transmitted from the configuration device to the device. For DATA3, you can leave the corresponding Bit3 line blank in the Quartus II software. On the PCB, leave the DATA3 line from the enhanced configuration device unconnected.

Alternatively, you can daisy chain two devices to one DATA line while the other DATA lines drive one device each. For example, you could use the 2-bit PS mode to drive two devices with DATA Bit0 (two EP2S15 devices) and the third device (EP2S30 device) with DATA Bit1. This 2-bit PS configuration scheme requires less space in the configuration flash memory, but can increase the total system configuration time.

A system may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF\_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete

configuration at the same time. Figure 11–24 shows multi-device PS configuration when the Arria GX devices are receiving the same configuration data.

**Figure 11–24. Multiple-Device PS Configuration Using an Enhanced Configuration Device When Devices Receive the Same Data**



**Notes to Figure 11–24:**

- (1) The pull-up resistor should be connected to the same supply voltage as the configuration device.
- (2) The nINIT\_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT\_CONF-nCONFIG line. The nINIT\_CONF pin does not need to be connected if its functionality is not used. If nINIT\_CONF is not used, nCONFIG must be pulled to VCC either directly or through a resistor.
- (3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.
- (4) The nCEO pins of all devices are left unconnected when configuring the same configuration data into multiple devices.

You can cascade several EPC2 devices to configure multiple Arria GX devices. The first configuration device in the chain is the master configuration device, while the subsequent devices are the slave devices. The master configuration device sends DCLK to the Arria GX devices and to the slave configuration devices. The first EPC device's nCS pin is connected to the CONF\_DONE pins of the devices, while its nCASC pin is connected to nCS of the next configuration device in the chain. The last device's nCS input comes from the previous device, while its nCASC pin is left floating. When all data from the first configuration device is sent, it drives nCASC low, which in turn drives nCS on the next configuration device. A configuration device requires less than one clock cycle to activate a subsequent configuration device, so the data stream is uninterrupted.

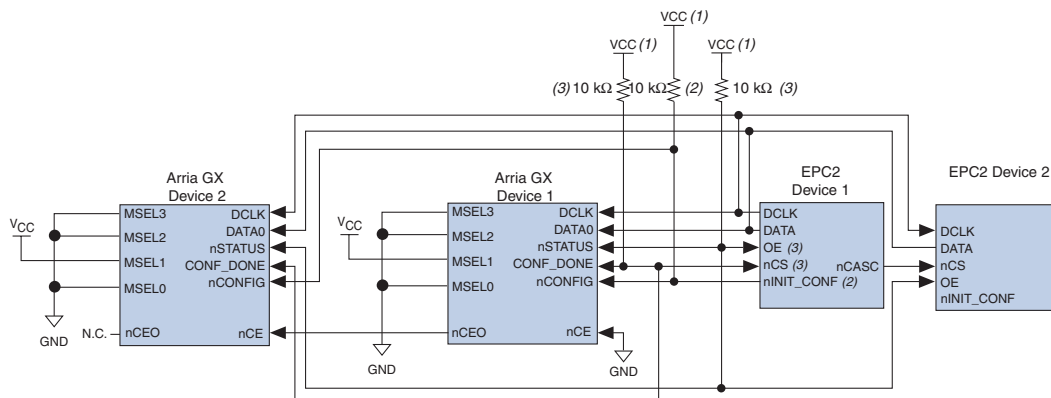


Enhanced configuration devices cannot be cascaded.

Because all nSTATUS and CONF\_DONE pins are tied together, if any device detects an error, the master configuration device stops configuration for the entire chain and the entire chain must be reconfigured. For example, if the master configuration device does not detect CONF\_DONE going high at the end of configuration, it resets the entire chain by pulling its OE pin low. This low signal drives the OE pin low on the slave configuration device(s) and drives nSTATUS low on all devices, causing them to enter a reset state. This behavior is similar to the device detecting an error in the configuration data.

Figure 11–25 shows how to configure multiple devices using cascaded EPC2 devices.

**Figure 11–25. Multi-Device PS Configuration Using Cascaded EPC2 Devices**



**Notes to Figure 11–25:**

- (1) The pull-up resistor should be connected to the same supply voltage as the configuration device.
- (2) The `nINIT_CONF` pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the `nINIT_CONF`-`nCONFIG` line. The `nINIT_CONF` pin does not need to be connected if its functionality is not used.
- (3) The enhanced configuration devices' and EPC2 devices' `OE` and `nCS` pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external 10-k $\Omega$  pull-up resistors should not be used. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

When using enhanced configuration devices or EPC2 devices, `nCONFIG` of the device can be connected to `nINIT_CONF` of the configuration device, allowing the `INIT_CONF` JTAG instruction to initiate device configuration. The `nINIT_CONF` pin does not need to be connected if its functionality is not used. An internal pull-up resistor on the `nINIT_CONF` pin is always active in the enhanced configuration devices and the EPC2 devices, which means that you shouldn't be using an external pull-up resistor if `nCONFIG` is tied to `nINIT_CONF`. If you are using multiple EPC2 devices to configure a Arria GX device(s), only the first EPC2 has its `nINIT_CONF` pin tied to the device's `nCONFIG` pin.

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

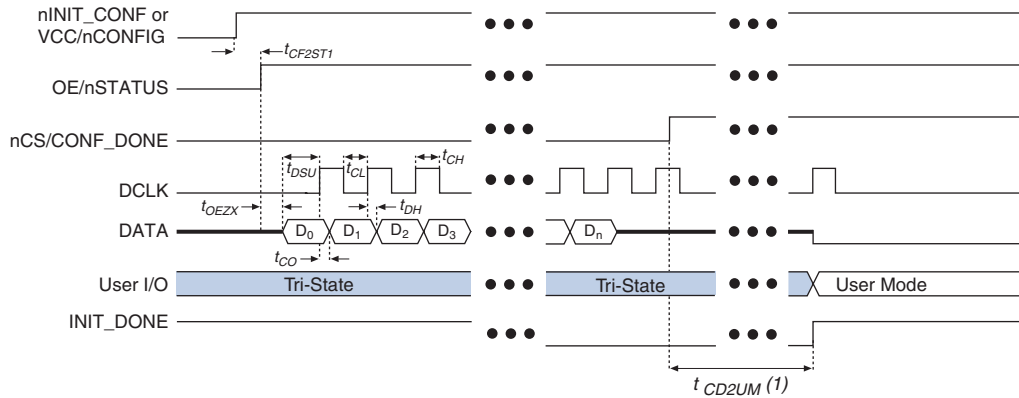




For more information on configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

Figure 11–26 shows the timing waveform for the PS configuration scheme using a configuration device.

**Figure 11–26. Arria GX PS Configuration Using a Configuration Device Timing Waveform**



**Note to Figure 11–26:**

- (1) The initialization clock can come from the Arria GX device's internal oscillator or the CLKUSR pin.



For timing information, refer to the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* chapter or the *Configuration Devices for SRAM-Based LUT Devices Data Sheet* chapter in volume 2 of the *Configuration Handbook*.



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

## PS Configuration Using a Download Cable

In this section, the generic term “download cable” includes the Altera USB-Blaster™ universal serial bus (USB) port download cable, MasterBlaster™ serial/USB communications cable, ByteBlaster™ II parallel port download cable, and the ByteBlaster MV parallel port download cable.

In PS configuration with a download cable, an intelligent host (such as a PC) transfers data from a storage device to the device via the USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV cable.

Upon power-up, the Arria GX devices go through a POR. The POR delay is dependent on the PORSEL pin setting. When PORSEL is driven low, the POR time is approximately 100 ms. If PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold nSTATUS low, and tri-state all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO\_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO\_pullup is driven high, the weak pull-up resistors are disabled.



The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*.

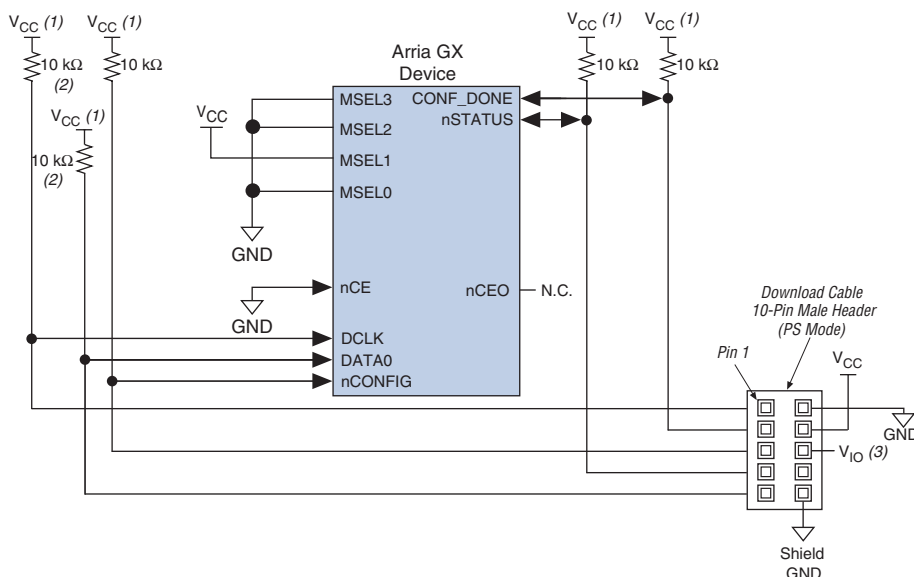
The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration in this scheme, the download cable generates a low-to-high transition on the nCONFIG pin.



To begin configuration, power the  $V_{CCINT}$ ,  $V_{CCIO}$ , and  $V_{CCPD}$  voltages (for the banks where the configuration and JTAG pins reside) to the appropriate voltage levels.

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

When using a download cable, setting the **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs. Additionally, the **Enable user-supplied start-up clock (CLKUSR)** option has no effect on the device initialization because this option is disabled in the SOF when programming the device using the Quartus II programmer and download cable. Therefore, if you turn on the CLKUSR option, you do not need to provide a clock on CLKUSR when you are configuring the device with the Quartus II programmer and a download cable. [Figure 11–27](#) shows PS configuration for Arria GX devices using a USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV cable.

**Figure 11–27. PS Configuration Using a USB Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV Cable****Notes to Figure 11–27:**

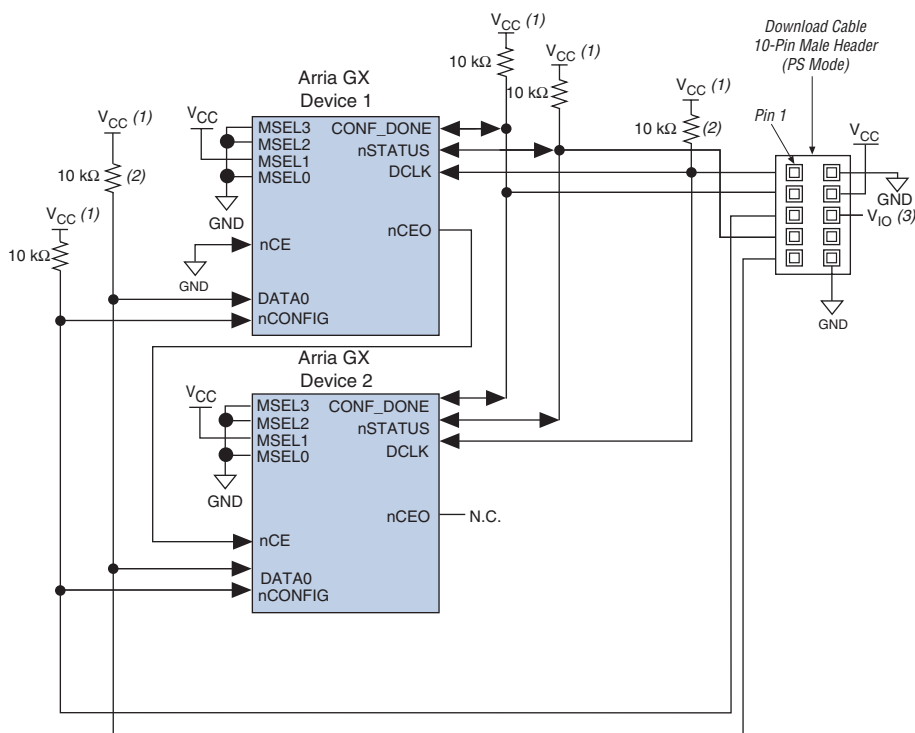
- (1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster ( $V_{IO}$  pin), ByteBlaster II, or ByteBlasterMV cable.
- (2) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This ensures that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.
- (3) Pin 6 of the header is a  $V_{IO}$  reference voltage for the MasterBlaster output driver.  $V_{IO}$  should match the device's  $V_{CCIO}$ . Refer to the [MasterBlaster Serial/USB Communications Cable User Guide](#) for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for active serial programming, otherwise it is a no connect.

You can use a download cable to configure multiple Arria GX devices by connecting each device's nCEO pin to the subsequent device's nCE pin. The first device's nCE pin is connected to GND while its nCEO pin is connected to the 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. All other configuration pins, nCONFIG, nSTATUS, DCLK, DATA0, and CONF\_DONE are connected to every 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. The **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs.

Figure 11–28 shows how to configure multiple Arria GX devices with a download cable.

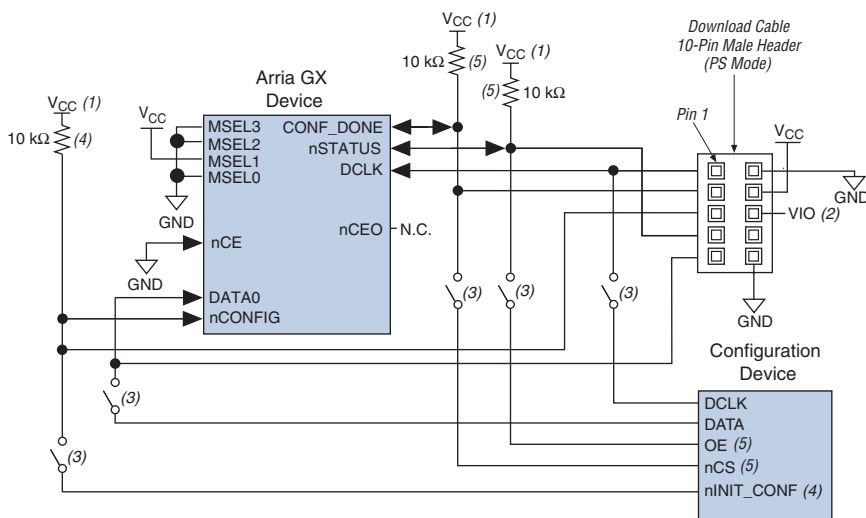
**Figure 11–28. Multi-Device PS Configuration using a USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV Cable**



**Notes to Figure 11–28:**

- (1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster ( $V_{IO}$  pin), ByteBlaster II, or ByteBlasterMV cable.
- (2) The pull-up resistors on `DATA0` and `DCLK` are only needed if the download cable is the only configuration scheme used on your board. This is to ensure that `DATA0` and `DCLK` are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on `DATA0` and `DCLK` are not needed.
- (3) Pin 6 of the header is a  $V_{IO}$  reference voltage for the MasterBlaster output driver.  $V_{IO}$  should match the device's  $V_{CCIO}$ . Refer to the *MasterBlaster Serial/USB Communications Cable User Guide* for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cables, this pin is connected to `nCE` when it is used for active serial programming, otherwise it is a no connect.

If you are using a download cable to configure device(s) on a board that also has configuration devices, electrically isolate the configuration device from the target device(s) and cable. One way of isolating the configuration device is to add logic, such as a multiplexer, that can select between the configuration device and the cable. The multiplexer chip allows bidirectional transfers on the `nSTATUS` and `CONF_DONE` signals. Another option is to add switches to the five common signals (`nCONFIG`, `nSTATUS`, `DCLK`, `DATA0`, and `CONF_DONE`) between the cable and the configuration device. The last option is to remove the configuration device from the board when configuring the device with the cable. [Figure 11–29](#) shows a combination of a configuration device and a download cable to configure an device.

**Figure 11–29. PS Configuration with a Download Cable and Configuration Device Circuit****Notes to Figure 11–29:**

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



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

- [USB-Blaster Download Cable User Guide](#)
- [MasterBlaster Serial/USB Communications Cable User Guide](#)
- [ByteBlaster II Download Cable User Guide](#)
- [ByteBlasterMV Download Cable User Guide](#)

## Passive Parallel Asynchronous Configuration

Passive parallel asynchronous (PPA) configuration uses an intelligent host, such as a microprocessor, to transfer configuration data from a storage device, such as flash memory, to the target Arria GX device.

Configuration data can be stored in RBF, HEX, or TTF format. The host system outputs byte-wide data and the accompanying strobe signals to the device. When using PPA, pull the DCLK pin high through a 10-k $\Omega$  pull-up resistor to prevent unused configuration input pins from floating.



You cannot use the Arria GX decompression feature if you are configuring your Arria GX device using PPA mode.

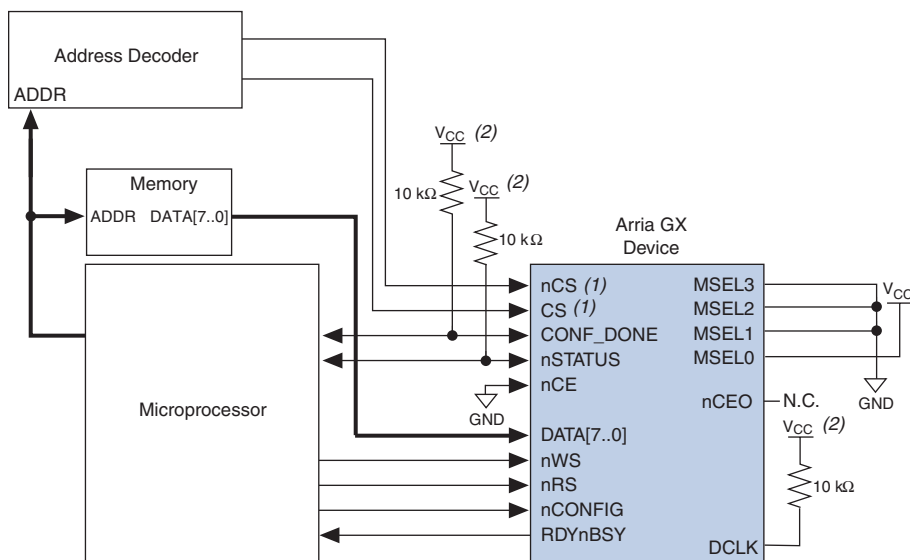
Table 11–16 shows the MSEL pin settings when using the PS configuration scheme.

<b>Table 11–16. Arria GX MSEL Pin Settings for PPA Configuration Schemes</b>				
<b>Configuration Scheme</b>	<b>MSEL3</b>	<b>MSEL2</b>	<b>MSEL1</b>	<b>MSEL0</b>
PPA	0	0	0	1
Remote System Upgrade PPA (1)	0	1	0	1

**Note to Table 11–16:**

- (1) This scheme requires that you drive the RUNLU pin to specify either remote update or local update. For more information about remote system upgrades in Arria GX devices, refer to the *Remote System Upgrades with Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

Figure 11–30 shows the configuration interface connections between the device and a microprocessor for single device PPA configuration. The microprocessor or an optional address decoder can control the device's chip select pins, nCS and CS. The address decoder allows the microprocessor to select the Arria GX device by accessing a particular address, which simplifies the configuration process. Hold the nCS and CS pins active during configuration and initialization.

**Figure 11–30. Single Device PPA Configuration Using a Microprocessor****Notes to Figure 11–30:**

- (1) If not used, the CS pin can be connected to  $V_{CC}$  directly. If not used, the nCS pin can be connected to GND directly.
- (2) The pull-up resistor should be connected to a supply that provides an acceptable input signal for the device.  $V_{CC}$  should be high enough to meet the  $V_{IH}$  specification of the I/O on the device and the external host.

During PPA configuration, it is only required to use either the nCS or CS pin. Therefore, if you are using only one chip-select input, the other must be tied to the active state. For example, nCS can be tied to ground while CS is toggled to control configuration. The device's nCS or CS pins can be toggled during PPA configuration if the design meets the specifications set for  $t_{CSSU}$ ,  $t_{WSB}$  and  $t_{CSH}$  listed in [Table 11–17 on page 11–80](#).

Upon power-up, the Arria GX devices go through a POR. The POR delay is dependent on the PORSEL pin setting. When PORSEL is driven low, the POR time is approximately 100 ms. If PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold nSTATUS low, and tri-state all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO\_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO\_pullup is driven high, the weak pull-up resistors are disabled.





The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*.

The configuration cycle consists of three stages: reset, configuration, and initialization. While `nCONFIG` or `nSTATUS` are low, the device is in reset.

Next, the microprocessor checks `nSTATUS` and `CONF_DONE`. If `nSTATUS` is not low and `CONF_DONE` is not high, the microprocessor sends the next data byte. However, if `nSTATUS` is not low and all the configuration data has been received, the device is ready for initialization. The `CONF_DONE` pin will go high one byte early in parallel configuration (FPP and PPA) modes. The last byte is required for serial configuration (AS and PS) modes. A low-to-high transition on `CONF_DONE` indicates configuration is complete and initialization of the device can begin. The open-drain `CONF_DONE` pin is pulled high by an external 10-k $\Omega$  pull-up resistor. The `CONF_DONE` pin must have an external 10-k $\Omega$  pull-up resistor in order for the device to initialize.

In Arria GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional `CLKUSR` pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Arria GX device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device.

You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the `CLKUSR` option. The **Enable user-supplied start-up clock (CLKUSR)** option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on `CLKUSR` does not affect the configuration process. After `CONF_DONE` goes high, `CLKUSR` is enabled after the time specified as  $t_{CD2CU}$ . After this time period elapses, Arria GX devices require 299 clock cycles to initialize properly and enter user mode. Arria GX devices support a `CLKUSR`  $f_{MAX}$  of 100 MHz.

An optional `INIT_DONE` pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. This **Enable INIT\_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the `INIT_DONE` pin is used, it is high because of an external 10-k $\Omega$  pull-up resistor 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 goes low. When initialization is complete, the `INIT_DONE` pin is released and pulled high. The microprocessor must be able to detect this low-to-high transition that signals the device has entered user mode. When initialization is complete, the device enters user mode. In user-mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

To ensure DATA [7 . . 0] is not left floating at the end of configuration, the microprocessor must drive them either high or low, whichever is convenient on your board. After configuration, the nCS, CS, nRS, nWS, RDYnBSY, and DATA [7 . . 0] pins can be used as user I/O pins. When choosing the PPA scheme in the Quartus II software as a default, these I/O pins are tri-stated in user mode and should be driven by the microprocessor. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the microprocessor that there is an error. If the **Auto-restart configuration after error** option-available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box-is turned on, the device releases nSTATUS after a reset time-out period (maximum of 100  $\mu$ s). After nSTATUS is released and pulled high by a pull-up resistor, the microprocessor can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 2  $\mu$ s) on nCONFIG to restart the configuration process.

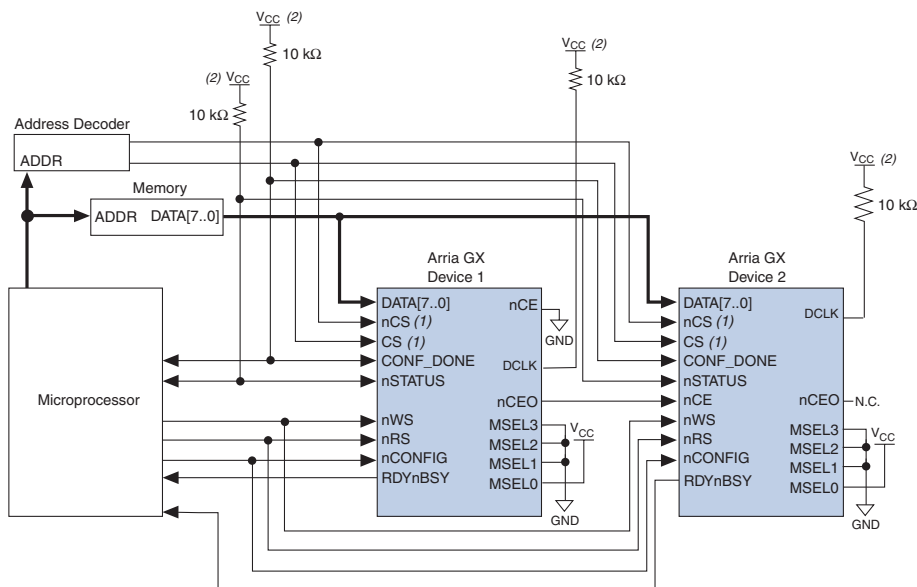
The microprocessor can also monitor the CONF\_DONE and INIT\_DONE pins to ensure successful configuration. To detect errors and determine when programming completes, monitor the CONF\_DONE pin with the microprocessor. If the microprocessor sends all configuration data but CONF\_DONE or INIT\_DONE has not gone high, the microprocessor must reconfigure the target device.



If you are using the optional CLKUSR pin and nCONFIG is pulled low to restart configuration during device initialization, ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 100  $\mu$ s).

When the device is in user-mode, a reconfiguration can be initiated by transitioning the nCONFIG pin low-to-high. The nCONFIG pin should go low for at least 2  $\mu$ s. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF\_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 11–31 shows how to configure multiple Arria GX devices using a microprocessor. This circuit is similar to the PPA configuration circuit for a single device, except the devices are cascaded for multi-device configuration.

**Figure 11–31. Multi-Device PPA Configuration Using a Microprocessor****Notes to Figure 11–31:**

- (1) If not used, the CS pin can be connected to  $V_{CC}$  directly. If not used, the nCS pin can be connected to GND directly.
- (2) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.  $V_{CC}$  should be high enough to meet the  $V_{IH}$  specification of the I/O on the device and the external host.

In the multi-device PPA configuration, 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. The second device in the chain begins configuration within one clock cycle. Therefore, the transfer of data destinations is transparent to the microprocessor.

Each device's RDYnBSY pin can have a separate input to the microprocessor. Alternatively, if the microprocessor is pin limited, all the RDYnBSY pins can feed an AND gate and the output of the AND gate can feed the microprocessor. For example, if you have two devices in a PPA configuration chain, the second device's RDYnBSY pin will be high during the time that the first device is being configured. When the first device has been successfully configured, it will drive nCEO low to activate the next device in the chain and drive its RDYnBSY pin high. Therefore, because

RDYnBSY signal is driven high before configuration and after configuration before entering user-mode, the device being configured will govern the output of the AND gate.

The nRS signal can be used in the multi-device PPA chain because the Arria GX devices tri-state the DATA [7 . . 0] pins before configuration and after configuration before entering user-mode to avoid contention. Therefore, only the device that is currently being configured responds to the nRS strobe by asserting DATA7.

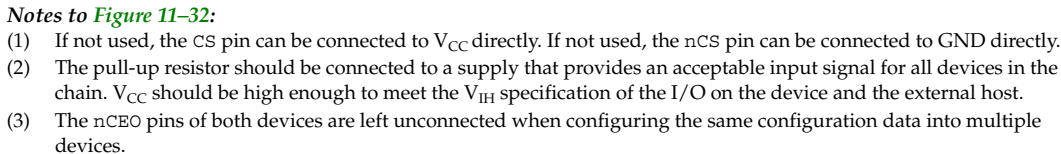
All other configuration pins (nCONFIG, nSTATUS, DATA [7 . . 0], nCS, CS, nWS, nRS, and CONF\_DONE) are connected to every device in the chain. It is not necessary to tie nCS and CS together for every device in the chain, as each device's nCS and CS input can be driven by a separate source. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DATA lines are buffered for every fourth device. Because all device CONF\_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Because all nSTATUS and CONF\_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices release their nSTATUS pins after a reset time-out period (maximum of 100  $\mu$ s). After all nSTATUS pins are released and pulled high, the microprocessor can try to reconfigure the chain without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 2  $\mu$ s) on nCONFIG to restart the configuration process.

In your system, you may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DATA [7 . . 0], nCS, CS, nWS, nRS, and CONF\_DONE) are connected to every device in the chain. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices start and complete configuration at the same time.

Figure 11–32 shows multi-device PPA configuration when both devices are receiving the same configuration data.

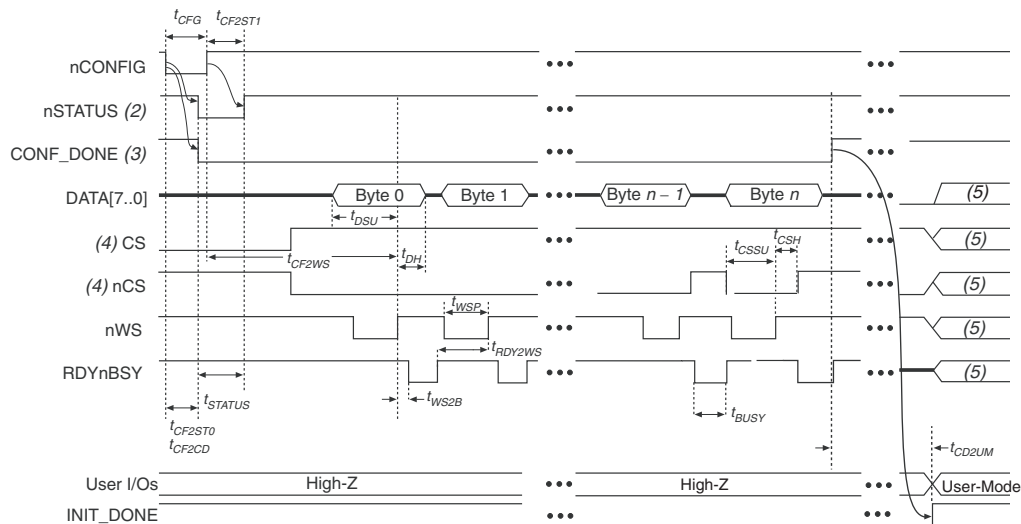


**Altera Corporation**  
**May 2008**

### PPA Configuration Timing

Figure 11–33 shows the timing waveform for the PPA configuration scheme using a microprocessor.

**Figure 11–33. Arria GX PPA Configuration Timing Waveform Using nWS** *Note (1)*

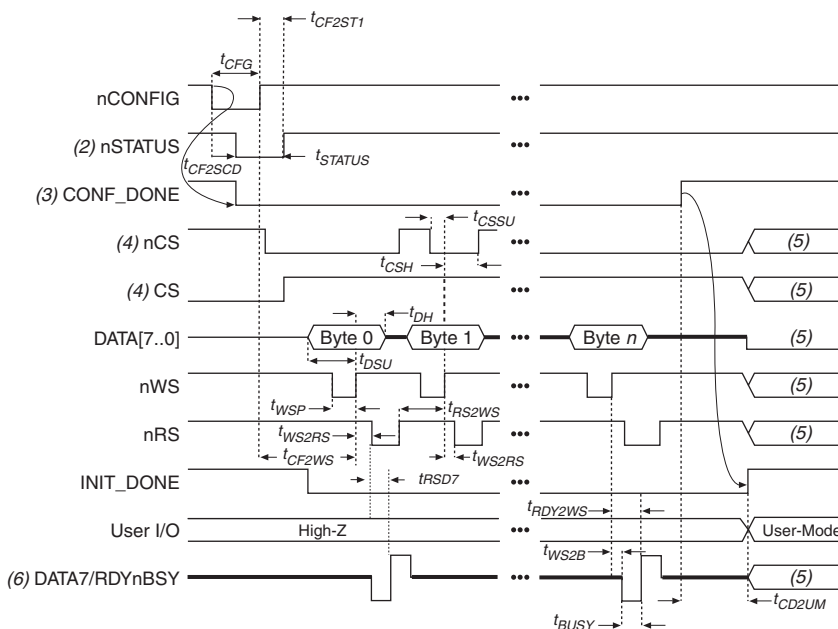


**Notes to Figure 11–33:**

- (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, Arria GX devices hold nSTATUS low for the time of the POR delay.
- (3) Upon power-up, before and during configuration, CONF\_DONE is low.
- (4) The user can toggle nCS or CS during configuration if the design meets the specification for  $t_{CSSU}$ ,  $t_{WS2P}$  and  $t_{CSH}$ .
- (5) DATA [7 . . 0], CS, nCS, nWS, nRS, and RDYnBSY are available as user I/O pins after configuration and the state of these pins depends on the dual-purpose pin settings.

Figure 11–34 shows the timing waveform for the PPA configuration scheme when using a strobed nRS and nWS signal.

**Figure 11–34. Arria GX PPA Configuration Timing Waveform Using nRS & nWS** *Note (1)*



**Notes to Figure 11–34:**

- (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, Arria GX devices hold nSTATUS low for the time of the POR delay.
- (3) Upon power-up, before and during configuration, CONF\_DONE is low.
- (4) The user can toggle nCS or CS during configuration if the design meets the specification for t\_CSSU, t\_WSP and t\_CSH.
- (5) DATA [7 . . 0], CS, nCS, nWS, nRS, and RDYNBSY are available as user I/O pins after configuration and the state of these pins depends on the dual-purpose pin settings.
- (6) DATA7 is a bidirectional pin. It is an input for configuration data input, but it is an output to show the status of RDYNBSY.

Table 11–17 defines the timing parameters for Arria GX devices for PPA configuration.

<b>Table 11–17. PPA Timing Parameters for Arria GX Devices (Part 1 of 2)</b> <i>Note (1)</i>				
Symbol	Parameter	Min	Max	Units
t <sub>CF2CD</sub>	nCONFIG low to CONF_DONE low		800	ns
t <sub>CF2ST0</sub>	nCONFIG low to nSTATUS low		800	ns



**Table 11–17. PPA Timing Parameters for Arria GX Devices (Part 2 of 2)** *Note (1)*

Symbol	Parameter	Min	Max	Units
$t_{CFG}$	nCONFIG low pulse width	2		$\mu s$
$t_{STATUS}$	nSTATUS low pulse width	10	100 (2)	$\mu s$
$t_{CF2ST1}$	nCONFIG high to nSTATUS high		100 (2)	$\mu s$
$t_{CSSU}$	Chip select setup time before rising edge on nWS	10		ns
$t_{CSH}$	Chip select hold time after rising edge on nWS	0		ns
$t_{CF2WS}$	nCONFIG high to first rising edge on nWS	100		$\mu s$
$t_{ST2WS}$	nSTATUS high to first rising edge on nWS	2		$\mu s$
$t_{DSU}$	Data setup time before rising edge on nWS	10		ns
$t_{DH}$	Data hold time after rising edge on nWS	0		ns
$t_{WSP}$	nWS low pulse width	15		ns
$t_{WS2B}$	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
$t_R$	Input rise time		40	ns
$t_F$	Input fall time		40	ns
$t_{CD2UM}$	CONF_DONE high to user mode (3)	20	100	$\mu s$
$t_{CD2CU}$	CONF_DONE high to CLKUSR enabled	40		ns
$t_{CD2UMC}$	CONF_DONE high to user mode with <b>CLKUSR</b> option on	$t_{CD2CU} + (299 \times \text{CLKUSR period})$		

**Notes to Table 11–17:**

- (1) This information is preliminary.
- (2) This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
- (3) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device.



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

## JTAG Configuration

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



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

- [IEEE 1149.1 \(JTAG\) Boundary-Scan Testing for Arria GX Devices](#) chapter in volume 2 of the *Arria GX Device Handbook*
- [JTAG Programming Support - JTAG Technologies](#)

Arria GX devices are designed such that JTAG instructions have precedence over any device configuration modes. Therefore, JTAG configuration can take place without waiting for other configuration modes to complete. For example, if you attempt JTAG configuration of Arria GX devices during PS configuration, PS configuration is terminated and JTAG configuration begins.



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

A device operating in JTAG mode uses four required pins, TDI, TDO, TMS, and TCK, and one optional pin, TRST. The TCK pin has an internal weak pull-down resistor, while the TDI, TMS, and TRST pins have weak internal pull-up resistors (typically 25 k $\Omega$ ). The TDO output pin is powered by V<sub>CCIO</sub> in I/O bank 4. All of the JTAG input pins are powered by the 3.3-V V<sub>CCPD</sub> pin. All user I/O pins are tri-stated during JTAG configuration. [Table 11–18](#) explains each JTAG pin's function.



The TDO output is powered by the V<sub>CCIO</sub> power supply of I/O bank 4.



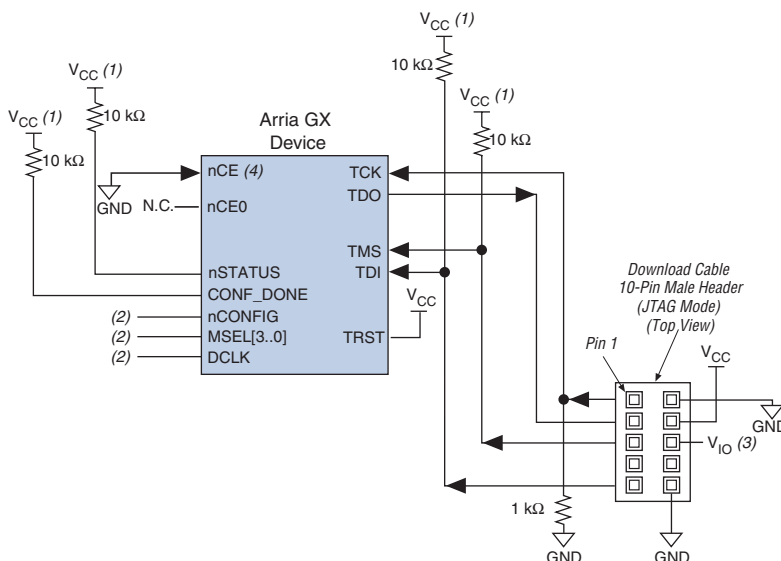
For recommendations on how to connect a JTAG chain with multiple voltages across the devices in the chain, refer to the *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

**Table 11–18. Dedicated JTAG Pins**

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

During JTAG configuration, data can be downloaded to the device on the PCB through the USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV download cable. Configuring devices through a cable is similar to programming devices in-system, except the TRST pin should be connected to V<sub>CC</sub>. This ensures that the TAP controller is not reset.

Figure 11–35 shows JTAG configuration of a single Arria GX device.



**Notes to Figure 11–35:**

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

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.

The Quartus II software verifies successful JTAG configuration upon completion. At the end of configuration, the software checks the state of CONF\_DONE through the JTAG port. When Quartus II generates a (.jam) file for a multi-device chain, it contains instructions so that all the devices in the chain will be initialized at the same time. If CONF\_DONE is not high, the Quartus II software indicates that configuration has failed. If

CONF\_DONE is high, the software indicates that configuration was successful. After the configuration bitstream is transmitted serially via the JTAG TDI port, the TCK port is clocked an additional 299 cycles to perform device initialization.

Arria GX devices have dedicated JTAG pins that always function as JTAG pins. Not only can you perform JTAG testing on Arria GX devices before and after, but also during configuration. While other device families do not support JTAG testing during configuration, Arria GX devices support the bypass, idcode, and sample instructions during configuration without interrupting configuration. All other JTAG instructions may only be issued by first interrupting configuration and reprogramming I/O pins using the CONFIG\_IO instruction.

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

The chip-wide reset (DEV\_CLRn) and chip-wide output enable (DEV\_OE) pins on Arria 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 Arria GX devices, consider the dedicated configuration pins. [Table 11–19](#) shows how these pins should be connected during JTAG configuration.

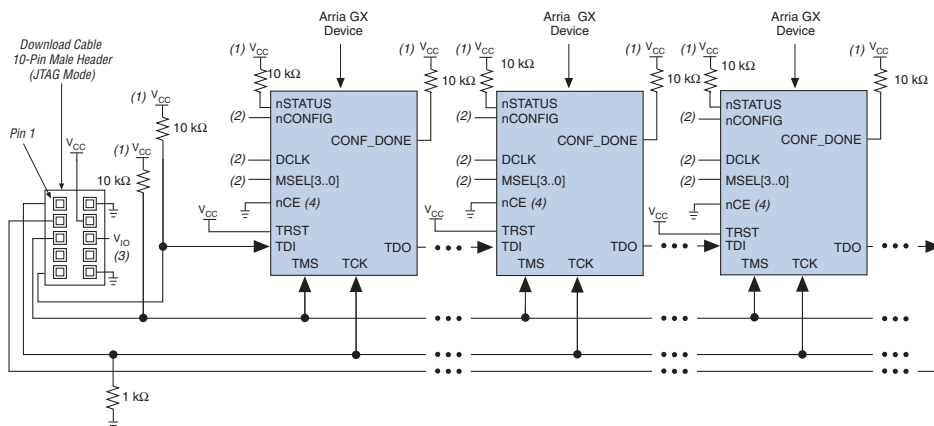
When programming a JTAG device chain, one JTAG-compatible header is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capability of the download cable. When four or more devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and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

**Table 11–19. Dedicated Configuration Pin Connections During JTAG Configuration**

Signal	Description
nCE	On all Arria 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 FPP, AS, PS, 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 Arria GX devices in the chain, nCEO can be left floating or connected to the nCE of the next device.
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, tie these pins to ground.
nCONFIG	Driven high by connecting to V <sub>CC</sub> , pulling up via a resistor, or driven high by some control circuitry.
nSTATUS	Pull to V <sub>CC</sub> via a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin should be pulled up to V <sub>CC</sub> individually.
CONF_DONE	Pull to V <sub>CC</sub> via a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each CONF_DONE pin should be pulled up to V <sub>CC</sub> 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.

JTAG-chain device programming is ideal when the system contains multiple devices, or when testing your system using JTAG BST circuitry. [Figure 11–36](#) shows multi-device JTAG configuration.

**Figure 11–36. JTAG Configuration of Multiple Devices Using a Download Cable**



**Notes to [Figure 11–36](#):**

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

The nCE pin must be connected to GND or driven low during JTAG configuration. In multi-device FPP, AS, PS, 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. In addition, the CONF\_DONE and nSTATUS signals are all shared in multi-device FPP, AS, PS, or PPA configuration chains so the devices can enter user mode at the same time after configuration is complete. When the CONF\_DONE and nSTATUS signals are shared among all the devices, every device must be configured when JTAG configuration is performed.

If you only use JTAG configuration, Altera recommends that you connect the circuitry as shown in [Figure 11–36](#), where each of the CONF\_DONE and nSTATUS signals are isolated, so that each device can enter user mode individually.

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, 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 will drive nCE of the next device low when it has successfully been JTAG configured.

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



Stratix, Arria GX, Cyclone®, and Cyclone II devices must be within the first 17 devices in a JTAG chain. All of these devices have the same JTAG controller. If any of the Stratix, Arria GX, Cyclone, and Cyclone II devices are in the 18th or after they will fail configuration. This does not affect SignalTap® II.

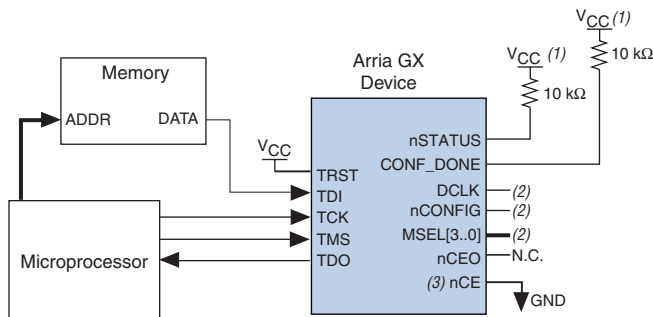


For more information on configuring multiple Altera devices in the same configuration chain, refer to the [Configuring Mixed Altera FPGA Chains](#) chapter in volume 2 of the *Configuration Handbook*.



Figure 11–37 shows JTAG configuration of a Arria GX device with a microprocessor.

**Figure 11–37. JTAG Configuration of a Single Device Using a Microprocessor**



**Notes to Figure 11–37:**

- (1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.  $V_{CC}$  should be high enough to meet the  $V_{IH}$  specification of the I/O on the device.
- (2) The nCONFIG, MSEL[3..0] pins should be connected to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect nCONFIG to  $V_{CC}$ , and MSEL[3..0] to ground. Pull DCLK either high or low, whichever is convenient on your board.
- (3) nCE must be connected to GND or driven low for successful JTAG configuration.

## Jam STAPL

Jam 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.

The Jam Player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine.



For more information on JTAG and Jam STAPL in embedded environments, refer to [AN 122: Using Jam STAPL for ISP & ICR via an Embedded Processor](#).

## Device Configuration Pins

Table 11–20 describes the connections and functionality of all the configuration related pins on Arria GX devices and summarizes the Arria GX pin configuration.

<b>Table 11–20. Arria GX Configuration Pin Summary (Part 1 of 2)</b> <i>Note (1)</i>					
Bank	Description	Input/Output	Dedicated	Powered By	Configuration Mode
3	PGM[2..0]	Output		(2)	PS, FPP, PPA, RU, LU
3	ASDO	Output		(2)	AS
3	nCSO	Output		(2)	AS
3	CRC_ERROR	Output		(2)	Optional, all modes
3	DATA0	Input		(3)	All modes except JTAG
3	DATA[7..1]	Input		(3)	FPP, PPA
3	DATA7	Bidirectional		(2), (3)	PPA
3	RDYnBSY	Output		(2)	PPA
3	INIT_DONE	Output		Pull-up	Optional, all modes
3	nSTATUS	Bidirectional	Yes	Pull-up	All modes
3	nCE	Input	Yes	(3)	All modes
3	DCLK	Input	Yes	(3)	PS, FPP
		Output		(2)	AS
3	CONF_DONE	Bidirectional	Yes	Pull-up	All modes
8	TDI	Input	Yes	VCCPD	JTAG
8	TMS	Input	Yes	VCCPD	JTAG
8	TCK	Input	Yes	VCCPD	JTAG
8	TRST	Input	Yes	VCCPD	JTAG
8	nCONFIG	Input	Yes	(3)	All modes
8	VCCSEL	Input	Yes	VCCINT	All modes
8	CS	Input		(3)	PPA
8	CLKUSR	Input		(3)	Optional
8	nWS	Input		(3)	PPA
8	nRS	Input		(3)	PPA
8	RUnLU	Input		(3)	PS, FPP, PPA, RU, LU
8	nCS	Input		(3)	PPA
7	PORSEL	Input	Yes	VCCINT	All modes
7	nIO_PULLUP	Input	Yes	VCCINT	All modes
7	PLL_ENA	Input	Yes	(3)	Optional

**Table 11–20. Arria GX Configuration Pin Summary (Part 2 of 2)** *Note (1)*

Bank	Description	Input/Output	Dedicated	Powered By	Configuration Mode
7	nCEO	Output	Yes	(2), (4)	All modes
4	MSEL [3 . . 0]	Input	Yes	VCCINT	All modes
4	TDO	Output	Yes	(2), (4)	JTAG

**Notes to Table 11–20:**

- (1) Total number of pins is 41, total number of dedicated pins is 19.
- (2) All outputs are powered by VCCIO except as noted.
- (3) All inputs are powered by VCCIO or VCCPD, based on the VCCSEL setting, except as noted.
- (4) An external pull-up resistor may be required for this configuration pin because of the multivolt I/O interface. Refer to the *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook* for pull-up or level shifter recommendations for nCEO and TDO.

Figure 11–38 shows the I/O bank locations.

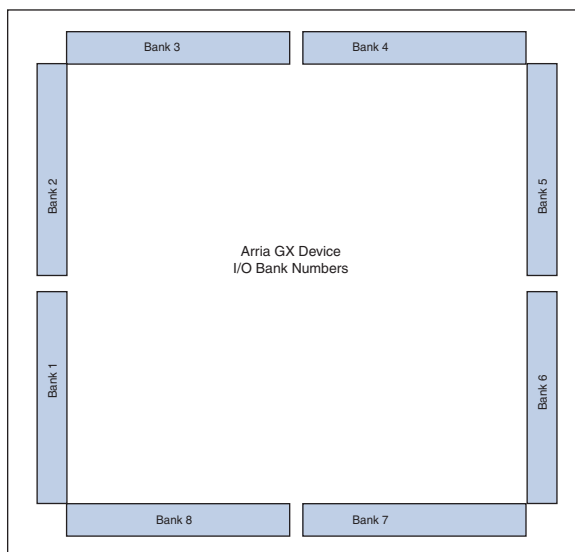
**Figure 11–38. Arria GX I/O Bank Numbers**

Table 11–21 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 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 1 of 10)**

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
V <sub>CCPD</sub>	N/A	All	Power	<p>Dedicated power pin. This pin is used to power the I/O pre-drivers, the JTAG input pins, and the configuration input pins that are affected by the voltage level of V<sub>CCSEL</sub>.</p> <p>This pin must be connected to 3.3-V. V<sub>CCPD</sub> must ramp-up from 0-V to 3.3-V within 100 ms. If V<sub>CCPD</sub> is not ramped up within this specified time, your Arria GX device will not configure successfully. If your system does not allow for a V<sub>CCPD</sub> ramp-up time of 100 ms or less, you must hold nCONFIG low until all power supplies are stable.</p>
V <sub>CCSEL</sub>	N/A	All	Input	<p>Dedicated input that selects which input buffer is used on the PLL_ENA pin and the configuration input pins; nCONFIG, DCLK (when used as an input), nSTATUS (when used as an input), CONF_DONE (when used as an input), DEV_OE, DEV_CLRn, DATA[7..0], RUnLU, nCE, nWS, nRS, CS, nCS, and CLKUSR. The 3.3-V/2.5-V input buffer is powered by V<sub>CCPD</sub>, while the 1.8-V/1.5-V input buffer is powered by V<sub>CCIO</sub>.</p> <p>The V<sub>CCSEL</sub> input buffer has an internal 5-kΩ pull-down resistor that is always active. The V<sub>CCSEL</sub> input buffer is powered by V<sub>CCINT</sub> and must be hardwired to V<sub>CCPD</sub> or ground. A logic high selects the 1.8-V/1.5-V input buffer, and a logic low selects the 3.3-V/2.5-V input buffer. For more information, refer to the “V<sub>CCSEL</sub> Pin” section.</p>

**Table 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 2 of 10)**

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
PORSEL	N/A	All	Input	<p>Dedicated input which selects between a POR time of 12 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 12 ms and a logic low selects POR time of about 100 ms.</p> <p>The PORSEL input buffer is powered by <math>V_{CCINT}</math> and has an internal 5-k<math>\Omega</math> pull-down resistor that is always active. The PORSEL pin should be tied directly to <math>V_{CCPD}</math> or GND.</p>
nIO_PULLUP	N/A	All	Input	<p>Dedicated input that chooses whether the internal pull-up resistors on the user I/O pins and dual-purpose I/O pins (nCS0, nASDO, DATA[7..0], nWS, nRS, RDYnBSY, nCS, CS, RUnLU, 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-up resistors, while a logic low turns them on.</p> <p>The nIO-PULLUP input buffer is powered by <math>V_{CCPD}</math> and has an internal 5-k<math>\Omega</math> pull-down resistor that is always active. The nIO-PULLUP can be tied directly to <math>V_{CCPD}</math> or use a 1-k<math>\Omega</math> pull-up resistor or tied directly to GND.</p>
MSEL[3..0]	N/A	All	Input	<p>4-bit configuration input that sets the Arria GX device configuration scheme. Refer to <a href="#">Table 11–1</a> for the appropriate connections.</p> <p>These pins must be hard-wired to <math>V_{CCPD}</math> or GND.</p> <p>The MSEL[3..0] pins have internal 5-k<math>\Omega</math> pull-down resistors that are always active.</p>

**Table 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 3 of 10)**

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCONFIG	N/A	All	Input	<p>Configuration control input. Pulling this pin low during user-mode will cause the device to lose its configuration data, enter a reset state, tri-state all I/O pins. Returning this pin to a logic high level will initiate a reconfiguration.</p> <p>If your configuration scheme uses an enhanced configuration device or EPC2 device, nCONFIG can be tied directly to V<sub>CC</sub> or to the configuration device's nINIT_CONF pin.</p>
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.</p> <p>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 will cause the configuration device to attempt to configure the device, but because the device ignores transitions on nSTATUS in user-mode, the device 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. The external 10-kΩ pull-up resistor should be used only when the internal pull-up resistor is not used.</p>

**Table 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 4 of 10)**

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nSTATUS (continued)				<p>If VCCPD and VCCIO are not fully powered up, the following could occur:</p> <ul style="list-style-type: none"> <li>• VCCPD and VCCIO are powered high enough for the nSTATUS buffer to function properly, and nSTATUS is driven low. When VCCPD and VCCIO are ramped up, POR trips, and nSTATUS is released after POR expires.</li> <li>• VCCPD and VCCIO are not powered high enough for the nSTATUS buffer to function properly. In this situation, nSTATUS might appear logic high, triggering a configuration attempt that would fail because POR did not yet trip. When VCCPD and VCCIO are powered up, nSTATUS is pulled low because POR did not yet trip. When POR trips after VCCPD and VCCIO are powered up, nSTATUS is released and pulled high. At that point, reconfiguration is triggered and the device is configured.</li> </ul>

**Table 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 5 of 10)**

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
CONF_DONE	N/A	All	Bidirectional open-drain	<p>Status output. The target device 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-k<math>\Omega</math> pull-up resistor in order for the device to initialize.</p> <p>Driving CONF_DONE low after configuration and initialization does not affect the configured device.</p> <p>The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external 10-k<math>\Omega</math> pull-up resistors should not be used on these pins. The external 10-k<math>\Omega</math> pull-up resistor should be used only when the internal pull-up resistor is not used.</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 device.</p>



**Table 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 6 of 10)**

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCEO	N/A	All	Output	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. The nCEO pin is powered by V <sub>CCIO</sub> in I/O bank 7. For recommendations on how to connect nCEO in a chain with multiple voltages across the devices in the chain, refer to the <i>Arria GX Architecture</i> chapter in volume 1 of the <i>Arria GX Device Handbook</i> .
ASDO	N/A in AS mode I/O in non-AS mode	AS	Output	Control signal from the Arria GX device to the serial configuration device in AS mode used to read out configuration data.  In AS mode, ASDO has an internal pull-up resistor that is always active.
nCSO	N/A in AS mode I/O in non-AS mode	AS	Output	Output control signal from the Arria GX device to the serial configuration device in AS mode that enables the configuration device.  In AS mode, nCSO has an internal pull-up resistor that is always active.

**Table 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 7 of 10)**

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DCLK	N/A	Synchronous configuration schemes (PS, FPP, AS)	Input (PS, FPP) Output (AS)	<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 device on the rising edge of DCLK.</p> <p>In AS mode, DCLK is an output from the Arria GX device that provides timing for the configuration interface. In AS mode, DCLK has an internal pull-up resistor (typically 25 k<math>\Omega</math>) that is always active.</p> <p>In PPA mode, DCLK should be tied high to V<sub>CC</sub> to prevent this pin from floating.</p> <p>After configuration, this pin is tri-stated. In schemes that use a configuration device, DCLK will be 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.</p>
DATA0	I/O	PS, FPP, PPA, AS	Input	<p>Data input. In serial configuration modes, bit-wide configuration data is presented to the target device on the DATA0 pin.</p> <p>The V<sub>IH</sub> and V<sub>IL</sub> levels for this pin are dependent on the input buffer selected by the VCCSEL pin. Refer to the section “<a href="#">VCCSEL Pin</a>” on page 11–9 for more information.</p> <p>In AS mode, DATA0 has an internal pull-up resistor that is always active.</p> <p>After configuration, DATA0 is available as a user I/O pin and the state of this pin depends on the <b>Dual-Purpose Pin</b> 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 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 8 of 10)**

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].</p> <p>The <math>V_{IH}</math> and <math>V_{IL}</math> levels for this pin are dependent on the input buffer selected by the VCCSEL pin. Refer to the section “<a href="#">VCCSEL Pin</a>” on page 11–9 for more information.</p> <p>In serial configuration schemes, they function as user I/O pins during configuration, which means they are tri-stated.</p> <p>After PPA or FPP configuration, DATA[7..1] are available as user I/O pins and the state of these pin depends on the <b>Dual-Purpose Pin</b> settings.</p>
DATA7	I/O	PPA	Bidirectional	<p>In the PPA configuration scheme, the DATA7 pin presents the RDY<sub>n</sub>BSY signal after the nRS signal has been strobed low.</p> <p>The <math>V_{IH}</math> and <math>V_{IL}</math> levels for this pin are dependent on the input buffer selected by the VCCSEL pin. Refer to the section “<a href="#">VCCSEL Pin</a>” on page 11–9 for more information.</p> <p>In serial configuration schemes, it functions as a user I/O pin 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 <b>Dual-Purpose Pin</b> 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 pin during configuration, which means it is tri-stated.</p> <p>After PPA configuration, nWS is available as a user I/O pins and the state of this pin depends on the <b>Dual-Purpose Pin</b> settings.</p>

**Table 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 9 of 10)**

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 pin and the state of this pin depends on the <b>Dual-Purpose Pin</b> 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 will drive 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 pin during configuration, which means it is tri-stated.</p> <p>After PPA configuration, RDYnBSY is available as a user I/O pin and the state of this pin depends on the <b>Dual-Purpose Pin</b> settings.</p>

**Table 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 10 of 10)**

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 ground while CS is toggled to control configuration.</p> <p>In non-PPA schemes, it functions as a user I/O pin during configuration, which means it is tri-stated.</p> <p>After PPA configuration, nCS and CS are available as user I/O pins and the state of these pins depends on the <b>Dual-Purpose Pin</b> settings.</p>
RUnLU	N/A if using Remote System Upgrade I/O if not	Remote System Upgrade 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 pin is available as general-purpose user I/O pin.</p> <p>When using remote system upgrade in AS mode, set the RUnLU pin to high because AS does not support local update</p>
PGM [2 . . 0]	N/A if using Remote System Upgrade I/O if not using	Remote System Upgrade in FPP, PS, or PPA	Output	<p>These output pins select one of eight pages in the memory (either flash or enhanced configuration device) when using a remote system upgrade 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 11–22 describes the optional configuration pins. If these optional configuration pins are not enabled in the Quartus II software, they are available as general-purpose user I/O pins. Therefore, during configuration, these pins function as user I/O pins and are tri-stated with weak pull-up resistors.

**Table 11–22. Optional Configuration Pins**

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

Table 11–23 describes the dedicated JTAG pins. JTAG pins must be kept stable before and during configuration to prevent accidental loading of JTAG instructions. The TDI, TMS, and TRST have weak internal pull-up resistors (typically 25 k $\Omega$ ) while TCK has a weak internal pull-down resistor. If you plan to use the SignalTap embedded logic array analyzer, you need to connect the JTAG pins of the Arria GX device to a JTAG header on your board.

**Table 11–23. Dedicated JTAG Pins**

Pin Name	User Mode	Pin Type	Description
TDI	N/A	Input	<p>Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. The TDI pin is powered by the 3.3-V <math>V_{CCPD}</math> supply.</p> <p>If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to <math>V_{CC}</math>.</p>
TDO	N/A	Output	<p>Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. The TDO pin is powered by <math>V_{CCIO}</math> in I/O bank 4. For recommendations on connecting a JTAG chain with multiple voltages across the devices in the chain, refer to the chapter <a href="#">IEEE 1149.1 (JTAG) Boundary Scan Testing in Arria GX Devices</a> chapter in volume 2 of the <i>Arria GX Device Handbook</i>.</p> <p>If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected.</p>
TMS	N/A	Input	<p>Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. The TMS pin is powered by the 3.3-V <math>V_{CCPD}</math> supply.</p> <p>If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to <math>V_{CC}</math>.</p>
TCK	N/A	Input	<p>The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. The TCK pin is powered by the 3.3-V <math>V_{CCPD}</math> supply.</p> <p>If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting TCK to GND.</p>
TRST	N/A	Input	<p>Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. The TRST pin is powered by the 3.3-V <math>V_{CCPD}</math> supply.</p> <p>If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting the TRST pin to GND.</p>

## Conclusion

Arria GX devices can be configured in a number of different schemes to fit your system's need. In addition, configuration data decompression and remote system upgrade support supplement the Arria GX configuration solution.

## Referenced Documents

This chapter references the following documents:

- *Altera Enhanced Configuration Devices* chapter in volume 2 of the *Configuration Handbook*
- *AN 122: Using Jam STAPL for ISP & ICR via an Embedded Processor*
- *AN 418: SRunner: An Embedded Solution for Serial Configuration*
- *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook*
- *Arria GX Device Handbook*
- *ByteBlaster II Download Cable User Guide*
- *ByteBlasterMV Download Cable User Guide*
- *Configuration Devices for SRAM-Based LUT Devices Data Sheet* chapter in volume 2 of the *Configuration Handbook*
- *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*
- *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*
- *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* chapter in volume 2 of the *Configuration Handbook*
- *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Arria GX Devices*
- *Jam Programming Support - JTAG Technologies*
- *MasterBlaster Serial/USB Communications Cable User Guide*
- *Remote System Upgrades With Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*
- *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*
- *Software Settings* section in volume 2 of the *Configuration Handbook*
- *USB-Blaster USB Port Download Cable User Guide*



## Document Revision History

Table 11–24 shows the revision history for this chapter.

<b>Table 11–24. Document Revision History</b>		
<b>Date and Document Version</b>	<b>Changes Made</b>	<b>Summary of Changes</b>
May 2008 v1.3	Updated: <ul style="list-style-type: none"> <li>• Table 11–2</li> <li>• Figure 11–6</li> <li>• Figure 11–7</li> </ul>	—
	Minor text edits.	—
August 2007 v1.2	Added the “Referenced Documents” section.	—
	Minor text edits.	—
June 2007 v1.1	Updated $t_{CF2CK}$ in Figures 11–6 and 11–7.	—
May 2007 v1.0	Initial Release	—

