

This section provides information about Altera® configuration devices. The following chapters contain information about how to use these devices, feature descriptions, device pin tables, and package diagrams.

This section includes the following chapters:

- [Chapter 1, Enhanced Configuration Devices \(EPC4, EPC8, and EPC16\) Data Sheet](#)
- [Chapter 2, Altera Enhanced Configuration Devices](#)
- [Chapter 3, Serial Configuration Devices \(EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128\) Data Sheet](#)
- [Chapter 4, Configuration Devices for SRAM-Based LUT Devices Data Sheet](#)

### Revision History

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




## Features

This chapter describes the EPC4, EPC8, and EPC16 enhanced configuration devices (EPC).

- Single-chip configuration solution for Altera® ACEX® 1K, APEX™ 20K (including APEX 20K, APEX 20KC, and APEX 20KE), APEX II, Arria® GX, Cyclone®, Cyclone II, FLEX® 10K (including FLEX 10KE and FLEX 10KA), Mercury™, Stratix® II, and Stratix II GX devices
- Contains 4-, 8-, and 16-Mbit flash memories for configuration data storage
  - On-chip decompression feature almost doubles the effective configuration density
- Standard flash die and a controller die combined into single stacked chip package
- External flash interface supports parallel programming of flash and external processor access to unused portions of memory
  - Flash memory block/sector protection capability via external flash interface
  - Supported in EPC16 and EPC4 devices
- Page mode support for remote and local reconfiguration with up to eight configurations for the entire system
  - Compatible with Stratix series Remote System Configuration feature
- Supports byte-wide configuration mode fast passive parallel (FPP); 8-bit data output per DCLK cycle
- Supports true  $n$ -bit concurrent configuration ( $n = 1, 2, 4, \text{ and } 8$ ) of Altera FPGAs
- Pin-selectable 2-ms or 100-ms power-on reset (POR) time
- Configuration clock supports programmable input source and frequency synthesis
  - Multiple configuration clock sources supported (internal oscillator and external clock input pin)
  - External clock source with frequencies up to 100 MHz
  - Internal oscillator defaults to 10 MHz; Programmable for higher frequencies of 33, 50, and 66 MHz
  - Clock synthesis supported via user programmable divide counter
- Available in the 100-pin plastic quad flat pack (PQFP) and the 88-pin Ultra FineLine BGA® (UFBGA) packages
  - Vertical migration between all devices supported in the 100-pin PQFP package
- Supply voltage of 3.3 V (core and I/O)
- Hardware compliant with IEEE Std. 1532 in-system programmability (ISP) specification

- Supports ISP via Jam Standard Test and Programming Language (STAPL)
- Supports JTAG boundary scan
- *n*INIT\_CONF pin allows private JTAG instruction to start FPGA configuration
- Internal pull-up resistor on *n*INIT\_CONF always enabled
- User programmable weak internal pull-up resistors on *n*CS and OE pins
- Internal weak pull-up resistors on external flash interface address and control lines, bus hold on data lines
- Standby mode with reduced power consumption

 For more information about FPGA configuration schemes and advanced features, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

## Functional Description

The Altera enhanced configuration device is a single-device, high-speed, advanced configuration solution for very high-density FPGAs. The core of an enhanced configuration device is divided into two major blocks: a configuration controller and a flash memory. The flash memory is used to store configuration data for systems made up of one or more Altera FPGAs. Unused portions of the flash memory can be used to store processor code or data that can be accessed via the external flash interface after FPGA configuration is complete. [Table 1–1](#) summarizes the features of Altera configuration devices and the amount of configuration space they hold.

**Table 1–1.** Altera Configuration Devices

Device	Memory Size (bits)	On-Chip Decompression Support	ISP Support	Cascading Support	Reprogrammable	Operating Voltage (V)
EPC4	4,194,304	Yes	Yes	No	Yes	3.3
EPC8	8,388,608	Yes	Yes	No	Yes	3.3
EPC16	16,777,216	Yes	Yes	No	Yes	3.3

[Table 1–2](#) lists the supported configuration devices required to configure an ACEX 1K, APEX 1K, APEX 20K, APEX 20KC, APEX 20KE, APEX II, Arria GX, Cyclone, Cyclone II, FLEX 10K, FLEX 10KA, FLEX 10KE, Stratix, Stratix GX, Stratix II, Stratix II GX, or Mercury device.

**Table 1-2.** Configuration Devices Required (Part 1 of 3)

Family	Device	Data Size (Bits) (1)	EPC4 (2)	EPC8 (2)	EPC16 (2)
Arria GX	EP1AGX20C	9,640,672	—	—	1
	EP1AGX35C	9,640,672	—	—	1
	EP1AGX35D				
	EP1AGX50C	16,951,824	—	—	1
	EP1AGX50D				
	EP1AGX60C	16,951,824	—	—	1
	EP1AGX60D EP1AGX60E				
	EP1AGX90E	25,699,104	—	—	1
Stratix	EP1S10	3,534,640	1	1	1
	EP1S20	5,904,832	1	1	1
	EP1S25	7,894,144	—	1	1
	EP1S30	10,379,368	—	1	1
	EP1S40	12,389,632	—	1	1
	EP1S60	17,543,968	—	—	1
	EP1S80	23,834,032	—	—	1
Stratix GX	EP1SGX10	3,534,640	1	1	1
	EP1SGX25	7,894,144	—	1	1
	EP1SGX40	12,389,632	—	1	1
Stratix II	EP2S15	4,721,544	1	1	1
	EP2S30	9,640,672	—	1	1
	EP2S60	16,951,824	—	—	1
	EP2S90	25,699,104	—	—	—
	EP2S130	37,325,760	—	—	—
	EP2S180	49,814,760	—	—	—
Stratix II GX	EP2SGX30C	9,640,672	—	—	1
	EP2SGX30D	9,640,672	—	—	1
	EP2SGX60C	16,951,824	—	—	1
	EP2SGX60D	16,951,824	—	—	1
	EP2SGX60E	16,951,824	—	—	1
	EP2SGX90E	25,699,104	—	—	—
	EP2SGX90F	25,699,104	—	—	—
	EP2SGX130G	37,325,760	—	—	—
Cyclone	EP1C3	627,376	1	1	1
	EP1C4	924,512	1	1	1
	EP1C6	1,167,216	1	1	1
	EP1C12	2,326,528	1	1	1
	EP1C20	3,559,608	1	1	1

**Table 1-2.** Configuration Devices Required (Part 2 of 3)

Family	Device	Data Size (Bits) (1)	EPC4 (2)	EPC8 (2)	EPC16 (2)
Cyclone II	EP2C5	1,223,980	1	1	1
	EP2C8	1,983,792	1	1	1
	EP2C20	3,930,986	1	1	1
	EP2C35	7,071,234	—	1	1
	EP2C50	9,122,148	—	1	1
	EP2C70	10,249,694	—	1	1
ACEX 1K	EP1K10	159,160	1	1	1
	EP1K30	473,720	1	1	1
	EP1K50	784,184	1	1	1
	EP1K100	1,335,720	1	1	1
APEX 20K	EP20K100	993,360	1	1	1
	EP20K200	1,950,800	1	1	1
	EP20K400	3,880,720	1	1	1
APEX 20KC	EP20K200C	196,8016	1	1	1
	EP20K400C	390,9776	1	1	1
	EP20K600C	567,3936	1	1	1
	EP20K1000C	8,960,016	—	1	1
APEX 20KE	EP20K30E	354,832	1	1	1
	EP20K60E	648,016	1	1	1
	EP20K100E	1,008,016	1	1	1
	EP20K160E	1,524,016	1	1	1
	EP20K200E	1,968,016	1	1	1
	EP20K300E	2,741,616	1	1	1
	EP20K400E	3,909,776	1	1	1
	EP20K600E	5,673,936	1	1	1
	EP20K1000E	8,960,016	—	1	1
	EP20K1500E	12,042,256	—	1	1
APEX II	EP2A15	4,358,512	1	1	1
	EP2A25	6,275,200	1	1	1
	EP2A40	9,640,528	—	1	1
	EP2A70	17,417,088	—	—	1
FLEX 10K	EPF10K10	118,000	1	1	1
	EPF10K20	231,000	1	1	1
	EPF10K30	376,000	1	1	1
	EPF10K40	498,000	1	1	1
	EPF10K50	621,000	1	1	1
	EPF10K70	892,000	1	1	1
	EPF10K100	1,200,000	1	1	1

**Table 1-2.** Configuration Devices Required (Part 3 of 3)

Family	Device	Data Size (Bits) (1)	EPC4 (2)	EPC8 (2)	EPC16 (2)
FLEX 10KA	EPF10K10A	120,000	1	1	1
	EPF10K30A	406,000	1	1	1
	EPF10K50V	621,000	1	1	1
	EPF10K100A	1,200,000	1	1	1
	EPF10K130V	1,600,000	1	1	1
	EPF10K250A	3,300,000	1	1	1
FLEX 10KE	EPF10K30E	473,720	1	1	1
	EPF10K50E	784,184	1	1	1
	EPF10K50S	784,184	1	1	1
	EPF10K100B	1,200,000	1	1	1
	EPF10K100E	1,335,720	1	1	1
	EPF10K130E	1,838,360	1	1	1
	EPF10K200E	2,756,296	1	1	1
	EPF10K200S	2,756,296	1	1	1
Mercury	EP1M120	1,303,120	1	—	1
	EP1M350	4,394,032	1	—	1

**Notes to Table 1-2:**

- (1) Raw Binary Files (.rbf) were used to determine these sizes.
- (2) These values with the enhanced configuration device compression feature enabled.



For more information about additional enhanced configuration devices, refer to the Process Change Notification [PCN0506: Addition of Intel Flash Memory As Source For EPC4, EPC8 and EPC16 Enhanced Configuration Devices and Using the Intel Flash Memory-Based EPC4, EPC8 and EPC16 Devices](#) white paper.

EPC devices support three different types of flash memory. [Table 1-3](#) lists the supported flash memory for all EPC devices.

**Table 1-3.** Enhanced Configuration Devices Flash Memory (Part 1 of 2)

Device	Grade	Package	Flash Memory	
			Leaded	Lead-Fee
EPC4	Commercial	PQFP 100	Intel (1) or Micron	Intel (1) or Micron
	Industrial	PQFP 100	Intel (1) or Micron	Intel (1)
EPC8	Commercial/ Industrial	PQFP 100	Intel (1) or Sharp	Intel (1)

**Table 1-3.** Enhanced Configuration Devices Flash Memory (Part 2 of 2)

Device	Grade	Package	Flash Memory	
			Leaded	Lead-Free
EPC16	Commercial	UBGA 88	Intel (1) or Sharp	Intel (1) or Sharp
	Industrial	UBGA 88	Intel (1) or Sharp	Intel (1)
	Commercial/ Industrial	PQFP 100	Intel (1) or Sharp	Intel (1)

**Note to Table 1-3:**

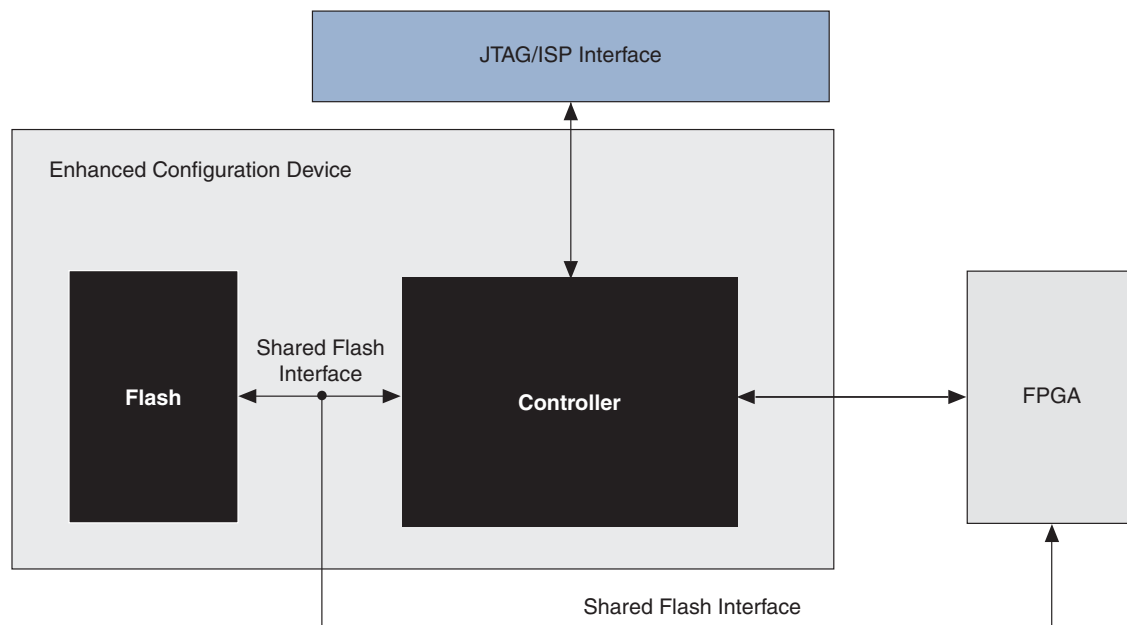
(1) For more information, refer to the Process Change Notification *PCN0506: Addition of Intel Flash Memory As Source for EPC4, EPC8 and EPC16 Enhanced Configuration Devices*.

The external flash interface is currently supported in the EPC16 and EPC4 devices. For information about using this feature in the EPC8 device, contact Altera Applications at [www.altera.com/support](http://www.altera.com/support).

Enhanced configuration devices have a 3.3-V core and I/O interface. The controller chip is a synchronous system that implements the various interfaces and features.

Figure 1-1 shows a block diagram of the enhanced configuration device. The controller chip features three separate interfaces:

- A configuration interface between the controller and the Altera FPGAs
- A JTAG interface on the controller that enables ISP of the flash memory
- An external flash interface that the controller shares with an external processor, or FPGA implementing a Nios® embedded processor (interface available after ISP and configuration)

**Figure 1-1.** Enhanced Configuration Device Block Diagram



The enhanced configuration device features multiple configuration schemes. In addition to supporting the traditional passive serial (PS) configuration scheme for a single device or a serial device chain, the enhanced configuration device features concurrent configuration and parallel configuration. With the concurrent configuration scheme, up to eight PS device chains can be configured simultaneously. In the FPP configuration scheme, 8-bits of data are clocked into the FPGA each cycle. These schemes offer significantly reduced configuration times over traditional schemes.

Furthermore, the enhanced configuration device features a dynamic configuration or page mode feature. This feature allows you to dynamically reconfigure all the FPGAs in your system with new images stored in the configuration memory. Up to eight different system configurations or pages can be stored in memory and selected using the PGM[2..0] pins. Your system can be dynamically reconfigured by selecting one of the eight pages and initiating a reconfiguration cycle.

This page mode feature combined with the external flash interface allows remote and local updates of system configuration data. The enhanced configuration devices are compatible with the Stratix Remote System Configuration feature.



For more information about Stratix Remote System Configuration, refer to the *Remote System Configuration with Stratix & Stratix GX Devices* chapter in the *Stratix Device Handbook*.

Other user programmable features include:

- Real-time decompression of configuration data
- Programmable configuration clock (DCLK)
- Flash ISP
- Programmable power-on-reset delay (PORSEL)

## FPGA Configuration

FPGA configuration is managed by the configuration controller chip. This process includes reading configuration data from the flash memory, decompressing it if necessary, transmitting configuration data via the appropriate DATA[] pins, and handling error conditions.

After POR, the controller determines the user-defined configuration options by reading its option bits from the flash memory. These options include the configuration scheme, configuration clock speed, decompression, and configuration page settings. The option bits are stored at flash address location 0x8000 (word address) and occupy 512-bits or 32-words of memory. These options bits are read using the internal flash interface and the default 10 MHz internal oscillator.

After obtaining the configuration settings, the configuration controller chip checks if the FPGA is ready to accept configuration data by monitoring the nSTATUS and CONF\_DONE lines. When the FPGA is ready (nSTATUS is high and CONF\_DONE is low), the controller begins data transfer using the DCLK and DATA[] output pins. The controller selects the configuration page to be transmitted to the FPGA by sampling its PGM[2..0] pins after POR or reset.

The function of the configuration unit is to transmit decompressed data to the FPGA, depending on the configuration scheme. The enhanced configuration device supports four concurrent configuration modes, with  $n = 1, 2, 4,$  or  $8$  (where  $n$  is the number of bits that are sent per DCLK cycle on the DATA [ $n$ ] lines). The value  $n = 1$  corresponds to the traditional PS configuration scheme. The values  $n = 2, 4,$  and  $8$  correspond to concurrent configuration of 2, 4, or 8 different PS configuration chains, respectively. Additionally, the FPGA can be configured in FPP mode, where eight bits of DATA are clocked into the FPGA per DCLK cycle. Depending on the configuration bus width ( $n$ ), the circuit shifts uncompressed configuration data to the valid DATA [ $n$ ] pins. Unused DATA [] pins drive low.

In addition to transmitting configuration data to the FPGAs, the configuration circuit is also responsible for pausing configuration whenever there is insufficient data available for transmission. This occurs when the flash read bandwidth is lower than the configuration write bandwidth. Configuration is paused by stopping the DCLK to the FPGA, when waiting for data to be read from the flash or for data to be decompressed. This technique is called “Pausing DCLK”.

The enhanced configuration device flash-memories feature a 90-ns access time (approximately 10 MHz). Hence, the flash read bandwidth is limited to about 160 megabits per second (Mbps) (16-bit flash data bus, DQ [], at 10 MHz). However, the configuration speeds supported by Altera FPGAs are much higher and translate to high configuration write bandwidths. For instance, 100-MHz Stratix FPP configuration requires data at the rate of 800 Mbps (8-bit DATA [] bus at 100 MHz). This is much higher than the 160 Mbps the flash memory can support, and is the limiting factor for configuration time. Compression increases the effective flash-read bandwidth as the same amount of configuration data takes up less space in the flash memory after compression. Since Stratix configuration data compression ratios are approximately two, the effective read bandwidth doubles to about 320 Mbps.

Finally, the configuration controller also manages errors during configuration. A CONF\_DONE error occurs when the FPGA does not de-assert its CONF\_DONE signal within 64 DCLK cycles after the last bit of configuration data is transmitted. When a CONF\_DONE error is detected, the controller pulses the OE line low, which pulls nSTATUS low and triggers another configuration cycle.

A cyclical redundancy check (CRC) error occurs when the FPGA detects corruption in the configuration data. This corruption could be a result of noise coupling on the board such as poor signal integrity on the configuration signals. When this error is signaled by the FPGA (by driving the nSTATUS line low), the controller stops configuration. If the Auto-Restart Configuration After Error option is enabled in the FPGA, it releases its nSTATUS signal after a reset time-out period and the controller attempts to reconfigure the FPGA.

After the FPGA configuration process is complete, the controller drives DCLK low and the DATA [] pins high. Additionally, the controller tri-states its internal interface to the flash memory, enables the weak internal pull-ups on the flash address and control lines, and enables bus-keep circuits on flash data lines.

The following sections briefly describe the different configuration schemes supported by the enhanced configuration device: FPP, PS, and concurrent configuration.



For detailed information about using these schemes to configure your Altera FPGA, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

## Configuration Signals

Table 1-4 lists the configuration signal connections between the enhanced configuration device and Altera FPGAs.

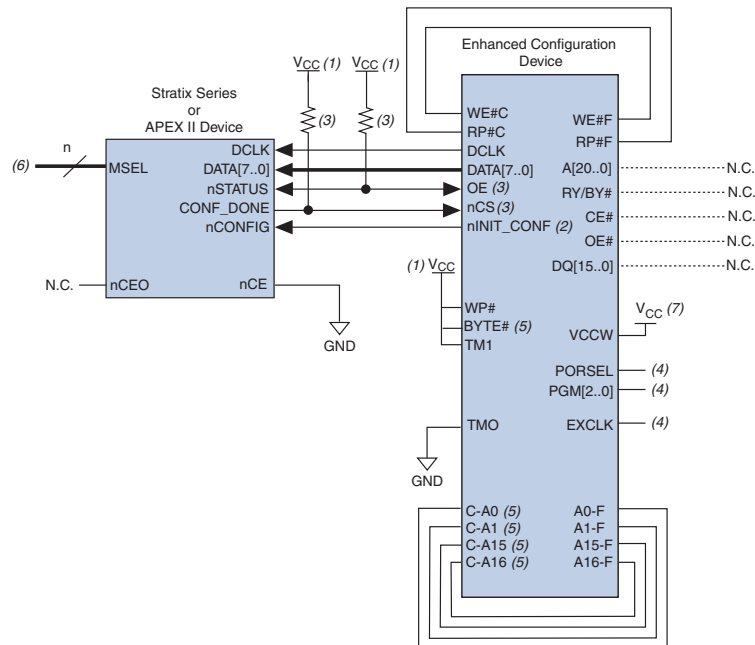
**Table 1-4.** Configuration Signals

Enhanced Configuration Device Pin	Altera FPGA Pin	Description
DATA []	DATA []	Configuration data transmitted from the configuration device to the FPGA, which is latched on the rising edge of DCLK.
DCLK	DCLK	Configuration device generated clock used by the FPGA to latch configuration data provided on the DATA [] pins.
nINIT_CONF, which	nCONFIG	Open-drain output from the configuration device that is used to start FPGA reconfiguration using the initiate configuration (INIT_CONF) JTAG instruction. This connection is not needed if the INIT_CONF JTAG instruction is not needed. If nINIT_CONF is not connected to nCONFIG, nCONFIG must be tied to V <sub>CC</sub> either directly or through a pull-up resistor.
OE	nSTATUS	Open-drain bidirectional configuration status signal, which is driven low by either device during POR and to signal an error during configuration. Low pulse on OE resets the enhanced configuration device controller.
nCS	CONF_DONE	Configuration done output signal driven by the FPGA.

## Fast Passive Parallel Configuration

Stratix series and APEX II devices can be configured using the enhanced configuration device in FPP mode. In this mode, the enhanced configuration device sends a byte of data on the DATA [7 . . 0] pins, which connect to the DATA [7 . . 0] input pins of the FPGA, per DCLK cycle. Stratix series and APEX II FPGAs receive byte-wide configuration data per DCLK cycle. Figure 1-2 shows the enhanced configuration device in FPP configuration mode. In this figure, the external flash interface is not used and hence most flash pins are left unconnected (with the few noted exceptions). For specific details about configuration interface connections including pull-up resistor values, supply voltages, and MSEL pin settings, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

Figure 1-2. FPP Configuration



## Notes to Figure 1-2:

- (1) The  $V_{CC}$  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 is not required 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.
- (4) For  $PORSEL$ ,  $PGM[]$ , and  $EXCLK$  pin connections, refer to Table 1-10.
- (5) In the 100-pin PQFP package, you must externally connect the following pins: C-A0 to F-A0, C-A1 to F-A1, C-A15 to F-A15, C-A16 to F-A16, and  $BYTE#$  to  $V_{CC}$ . Additionally, you must make the following pin connections in both 100-pin PQFP and 88-pin UFBGA packages: C-RP# to F-RP#, C-WE# to F-WE#, TM1 to  $V_{CC}$ , TMO to GND, and WP# to  $V_{CC}$ .
- (6) Connect the FPGA  $MSEL[]$  input pins to select the FPP configuration mode. For details, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.
- (7) To protect Intel Flash based EPC devices content, isolate the  $V_{CCW}$  supply from  $V_{CC}$ . For more information, refer section "Intel-Flash-Based EPC Device Protection" on page 1-15.

Multiple FPGAs can be configured using a single enhanced configuration device in FPP mode. In this mode, multiple Stratix series FPGAs, APEX II FPGAs, or both, are cascaded together in a daisy chain.

After the first FPGA completes configuration, its  $nCEO$  pin asserts to activate the  $nCE$  pin for the second FPGA, which prompts the second device to start capturing configuration data. In this setup, the FPGAs  $CONF\_DONE$  pins are tied together, and hence all devices initialize and enter user mode simultaneously. If the enhanced configuration device or one of the FPGAs detects an error, configuration stops (and simultaneously restarts) for the whole chain because the  $nSTATUS$  pins are tied together.



While Altera FPGAs can be cascaded in a configuration chain, the enhanced configuration devices cannot be cascaded to configure larger devices or chains.

- For configuration schematics and more information about multi-device FPP configuration, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

### Passive Serial Configuration

APEX 20KC, APEX 20KE, APEX 20K, APEX II, Cyclone series, FLEX 10K, and Stratix series devices can be configured using enhanced configuration devices in the PS mode. This mode is similar to the FPP mode, with the exception that only one bit of data (DATA [0]) is transmitted to the FPGA per DCLK cycle. The remaining DATA [7 . . 1] output pins are unused in this mode and drive low.

The configuration schematic for PS configuration of a single FPGA or single serial chain is identical to the FPP schematic (with the exception that only DATA [0] output from the enhanced configuration device connects to the FPGA DATA0 input pin; remaining DATA [7 . . 1] pins are left floating).

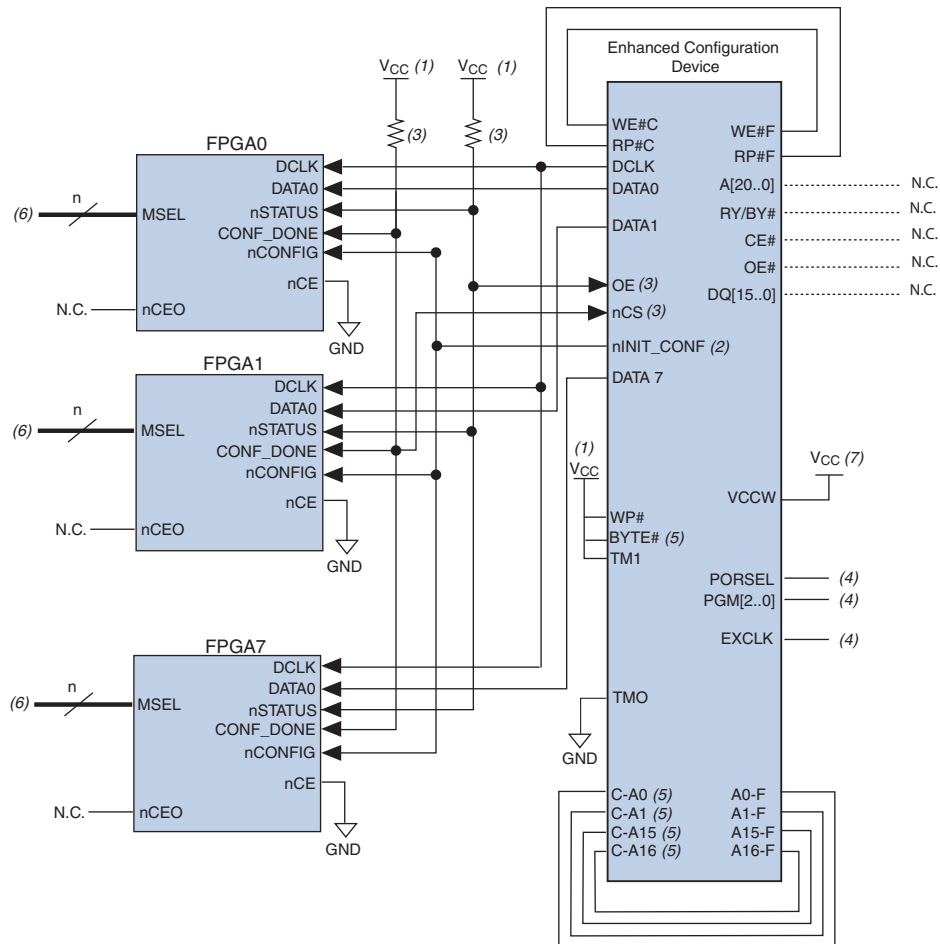
- For configuration schematics and more information about multi-device PS configuration, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

### Concurrent Configuration

Enhanced configuration devices support concurrent configuration of multiple FPGAs (or FPGA chains) in PS mode. Concurrent configuration is when the enhanced configuration device simultaneously outputs  $n$  bits of configuration data on the DATA [ $n-1$  . . 0] pins ( $n = 1, 2, 4, \text{ or } 8$ ), and each DATA [ ] line serially configures a different FPGA (chain). The number of concurrent serial chains is user-defined via the Quartus II software and can be any number from 1 to 8. For example, three concurrent chains you can select the 4-bit PS mode, and connect the least significant DATA bits to the FPGAs or FPGA chains. Leave the most significant DATA bit (DATA [3]) unconnected. Similarly, for 5-, 6-, or 7-bit concurrent chains you can select the 8-bit PS mode.

Figure 1-3 shows the schematic for configuring multiple FPGAs concurrently in the PS mode using an enhanced configuration device.

- For specific details about configuration interface connections including pull-up resistor values, supply voltages, and MSEL pin settings, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

**Figure 1-3.** Concurrent Configuration of Multiple FPGAs in PS Mode (n = 8)**Notes to Figure 1-3:**

- (1) Connect  $V_{CC}$  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 is not required 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.
- (4) For  $PORSEL$ ,  $PGM[]$ , and  $EXCLK$  pin connections, refer to Table 1-10.
- (5) In the 100-pin PQFP package, you must externally connect the following pins: C-A0 to F-A0, C-A1 to F-A1, C-A15 to F-A15, C-A16 to F-A16, and  $BYTE\#$  to  $V_{CC}$ . Additionally, you must make the following pin connections in both 100-pin PQFP and 88-pin UFBGA packages: C-RP# to F-RP#, C-WE# to F-WE#, TM1 to  $V_{CC}$ , TMO to GND, and WP# to  $V_{CC}$ .
- (6) Connect the FPGA  $MSEL[]$  input pins to select the PS configuration mode. For details, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.
- (7) To protect Intel Flash based EPC devices content, isolate the  $V_{CCW}$  supply from  $V_{CC}$ . For more information, refer section "Intel-Flash-Based EPC Device Protection" on page 1-15.


Table 1-5 summarizes the concurrent PS configuration modes supported in the enhanced configuration device.

**Table 1-5.** Enhanced Configuration Devices in PS Mode

Mode Name	Mode ( $n =$ ) (1)	Used Outputs	Unused Outputs
Passive serial mode	1	DATA0	DATA [7..1] drive low
Multi-device passive serial mode	2	DATA [1..0]	DATA [7..2] drive low
Multi-device passive serial mode	4	DATA [3..0]	DATA [7..4] drive low
Multi-device passive serial mode	8	DATA [7..0]	—

**Note to Table 1-5:**

(1) This is the number of valid DATA outputs for each configuration mode.

 For configuration schematics and more information about concurrent configurations, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

## External Flash Interface


The enhanced configuration devices support external FPGA or processor access to its flash memory. The unused portions of the flash memory can be used by the external device to store code or data. This interface can also be used in systems that implement remote configuration capabilities. Configuration data within a particular configuration page can be updated via the external flash interface and the system could be reconfigured with the new FPGA image. This interface is also useful to store Nios boot code, application code, or both.

 For more information about the Stratix remote configuration feature, refer to the *Remote System Configuration with Stratix & Stratix GX Devices* chapter in the *Stratix Device Handbook*.

The address, data, and control ports of the flash memory are internally connected to the enhanced configuration device controller and to external device pins. An external source can drive these external device pins to access the flash memory when the flash interface is available.

This external flash interface is a shared bus interface with the configuration controller chip. The configuration controller is the primary bus master. Since there is no bus arbitration support, the external device can only access the flash interface when the controller has tri-stated its internal interface to the flash. Simultaneous access by the controller and the external device will cause contention, and result in configuration and programming failures.


Since the internal flash interface is directly connected to the external flash interface pins, controller flash access cycles will toggle the external flash interface pins. The external device must be able to tri-state its flash interface during these operations and ignore transitions on the flash interface pins.

 The external flash interface signals cannot be shared between multiple enhanced configuration devices because this causes contention during in-system programming and configuration. During these operations, the controller chips inside the enhanced configuration devices are actively accessing flash memory. Therefore, enhanced configuration devices do not support shared flash bus interfaces.

The enhanced configuration device controller chip accesses flash memory during:

- FPGA configuration—reading configuration data from flash
- JTAG-based flash programming—storing configuration data in flash
- At POR—reading option bits from flash

During these operations, the external FPGA or processor must tri-state its interface to the flash memory. After configuration and programming, the enhanced configuration device's controller tri-states the internal interface and goes into an idle mode. To interrupt a configuration cycle in order to access the flash via the external flash interface, the external device can hold the FPGA's *nCONFIG* input low. This keeps the configuration device in reset by holding the *nSTATUS-OE* line low, allowing external flash access.

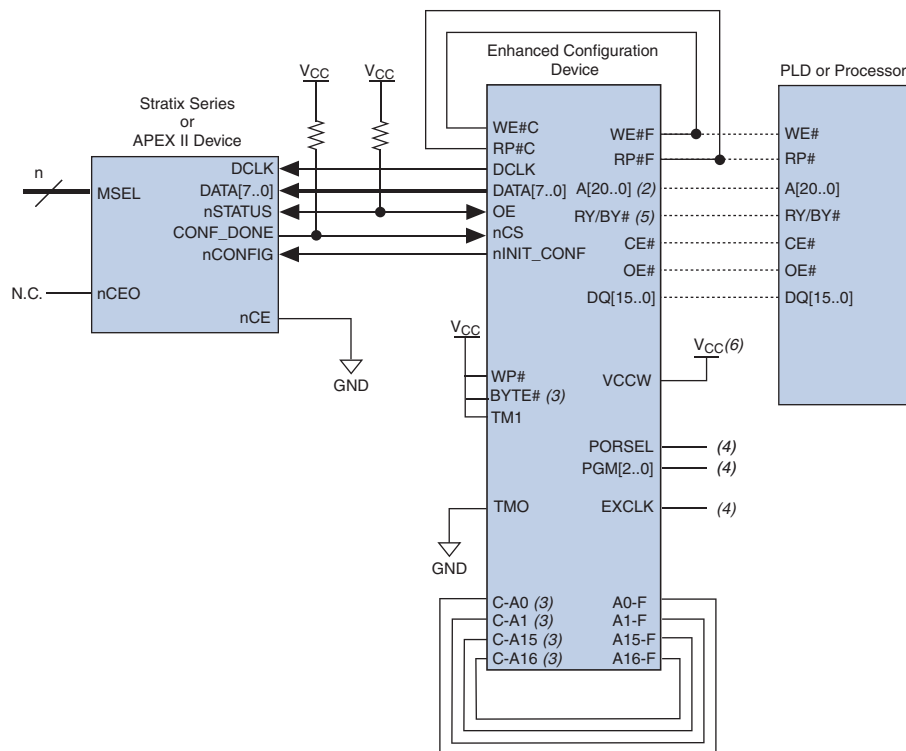
 For more information about the software support for the external flash interface feature, refer to the *Altera Enhanced Configuration Devices* chapter in volume 2 of the *Configuration Handbook*. For details about flash commands, timing, memory organization, and write protection features, refer to the following documents:

- For Micron flash-based EPC4, refer to the *Micron Flash Memory MT28F400B3 Data Sheet* at [www.micron.com](http://www.micron.com).
- For Sharp flash-based EPC16, refer to the *Sharp LHF16J06 Data Sheet Flash Memory Used in EPC16 Devices* at [www.sharpsma.com](http://www.sharpsma.com).
- For the *Intel Advanced Boot Block Flash Memory (B3) 28F008/800B3, 28F016/160B3, 28F320B3, 28F640B3 Datasheet*, visit [www.intel.com](http://www.intel.com).



Figure 1-4 shows an FPP configuration schematic with the external flash interface in use.

Figure 1-4. FPP Configuration with External Flash Interface (Note 1)



**Notes to Figure 1-4:**

- (1) For external flash interface support in EPC8 enhanced configuration device, contact Altera Applications.
- (2) Pin A20 in EPC16 devices, pins A20 and A19 in EPC8 devices, and pins A20, A19, and A18 in EPC4 devices should be left floating. These pins should not be connected to any signal; they are no-connect pins.
- (3) In the 100-pin PQFP package, you must externally connect the following pins: C-A0 to F-A0, C-A1 to F-A1, C-A15 to F-A15, C-A16 to F-A16, and BYTE# to VCC. Additionally, you must make the following pin connections in both 100-pin PQFP and 88-pin UFBGA packages: C-RP# to F-RP#, C-WE# to F-WE#, TM1 to VCC, TMO to GND, and WP# to VCC.
- (4) For PORSEL, PGM [], and EXCLK pin connections, refer to Table 1-10.
- (5) RY/BY# pin is only available for Sharp flash-based EPC8 and EPC16.
- (6) To protect Intel Flash based EPC devices content, isolate the VCCW supply from VCC. For more information, refer section "Intel-Flash-Based EPC Device Protection" on page 1-15.

**Intel-Flash-Based EPC Device Protection**

In the absence of the lock bit protection feature in the EPC4, EPC8, and EPC16 devices with Intel flash, Altera recommends four methods to protect the Intel Flash content in EPC4, EPC8, and EPC16 devices. Any method alone is sufficient to protect the flash. The methods are listed here in the order of descending protection level:

1. Using an RP# of less than 0.3 V on power-up and power-down for a minimum of 100 ns to a maximum 25 ms disables all control pins, making it impossible for a write to occur.

2. Using  $V_{PP} < V_{PPLK}$ , where the maximum value of  $V_{PPLK}$  is 1 V, disables writes.  $V_{PP} < V_{PPLK}$  means programming or writes cannot occur.  $V_{PP}$  is a programming supply voltage input pin on the Intel flash.  $V_{PP}$  is equivalent to the  $V_{CCW}$  pin on EPC devices.
3. Using a high  $CE\#$  disables the chip. The requirement for a write is a low  $CE\#$  and low  $WE\#$ . A high  $CE\#$  by itself prevents writes from occurring.
4. Using a high  $WE\#$  prevent writes because a write only occurs when the  $WE\#$  is low.

Performing all four methods simultaneously is the safest protection for the flash content.

The ideal power-up sequence is as follows:

1. Power-up  $V_{CC}$ .
2. Maintain  $V_{PP} < V_{PPLK}$  until  $V_{CC}$  is fully powered up.
3. Power-up  $V_{PP}$ .
4. Drive  $RP\#$  low during the entire power-up process.  $RP\#$  must be released high within 25 ms after  $V_{PP}$  is powered up.



$CE\#$  and  $WE\#$  must be high for the entire power-up sequence.

The ideal power-down sequence is as follows:

1. Drive  $RP\#$  low for 100ns before power-down.
2. Power-down  $V_{PP} < V_{PPLK}$ .
3. Power-down  $V_{CC}$ .
4. Drive  $RP\#$  low during the entire power-down process.



$CE\#$  and  $WE\#$  must be high for the entire power-down sequence.

The  $RP\#$  pin is not internally connected to the controller. Therefore, an external loop-back connection between  $C-RP\#$  and  $F-RP\#$  must be made on the board even when you are not using the external device to the  $RP\#$  pin with the loop-back. Always tri-state  $RP\#$  when the flash is not in use.

If an external power-up monitoring circuit is connected to the  $RP\#$  pin with the loop-back, use the following guidelines to avoid contention on the  $RP\#$  line:

- The power-up sequence on the 3.3-V supply should complete within 50 ms of power-up. The 3.3-V  $V_{CC}$  should reach the minimum  $V_{CC}$  before 50 ms and  $RP\#$  should then be released.
- $RP\#$  should be driven low by the power-up monitoring circuit during power-up. After power-up,  $RP\#$  should be tri-stated externally by the power-up monitoring circuit.

If the preceding guidelines cannot be completed within 50 ms, then the  $OE$  pin must be driven low externally until  $RP\#$  is ready to be released.

## Dynamic Configuration (Page Mode)


The dynamic configuration (or page mode) feature allows the enhanced configuration device to store up to eight different sets of designs for all the FPGAs in your system. You can then choose which page (set of configuration files) the enhanced configuration device should use for FPGA configuration.

Dynamic configuration or the page mode feature enables you to store a minimum of two pages: a factory default or fail-safe configuration, and an application configuration. The fail-safe configuration page could be programmed during system production, while the application configuration page could support remote or local updates. These remote updates could add or enhance system features and performance. However, with remote update capabilities comes the risk of possible corruption of configuration data. In the event of such a corruption, the system could automatically switch to the fail-safe configuration and avoid system downtime.

The enhanced configuration device page mode feature works with the Stratix Remote System Configuration feature, to enable intelligent remote updates to your systems.

 For more information about remotely updating Stratix FPGAs, refer to *Remote System Configuration with Stratix & Stratix GX Devices* in the *Stratix Device Handbook*.


The three PGM[2..0] input pins control which page is used for configuration, and these pins are sampled at the start of each configuration cycle when OE goes high. The page mode selection allows you to dynamically reconfigure the functionality of your FPGA by switching the PGM[2..0] pins and asserting nCONFIG. Page 0 is defined as the default page and the PGM[2] pin is the most significant bit (MSB).

 The PGM[2..0] input pins must not be left floating on your board, regardless of whether this feature is used or not. When this feature is not used, connect the PGM[2..0] pins to GND to select the default page 000.

The enhanced configuration device pages are dynamically sized regions in memory. The start address and length of each page is programmed into the option-bit space of the flash memory during initial programming. All subsequent configuration cycles will sample the PGM[] pins and use the option-bit information to jump to the start of the corresponding configuration page. Each page must have configuration files for all FPGAs in your system that are connected to that enhanced configuration device.

For example, if your system requires three configuration pages and includes two FPGAs, each page will store two SRAM Object Files (.sof) for a total of six .sof in the configuration device.

Furthermore, all enhanced configuration device configuration schemes (PS, FPP, and concurrent PS) are supported with the page-mode feature. The number of pages, devices, or both, that can be configured using a single enhanced configuration device is only limited by the size of the flash memory.

 For detailed information about the page-mode feature implementation and programming file generation steps using the Quartus II software, refer to the *Altera Enhanced Configuration Devices* chapter in volume 2 of the *Configuration Handbook*.

## Real-Time Decompression

Enhanced configuration devices support on-chip real time decompression of configuration data. FPGA configuration data is compressed by the Quartus II software and stored in the enhanced configuration device. During configuration, the decompression engine inside the enhanced configuration device will decompress or expand configuration data. This feature increases the effective-configuration density of the enhanced configuration device up to 7, 15, or 30 Mbits in the EPC4, EPC8, and EPC16, respectively.

The enhanced configuration device also supports a parallel 8-bit data bus to the FPGA to reduce configuration time. However, in some cases, the FPGA data-transfer time is limited by the flash-read bandwidth. For example, when configuring an APEX II device in FPP (byte-wide data per cycle) mode at a configuration speed of 66 MHz, the FPGA write bandwidth is equal to  $8 \text{ bits} \times 66 \text{ MHz} = 528 \text{ Mbps}$ . The flash read interface, however, is limited to approximately 10 MHz (since the flash access time is  $\sim 90 \text{ ns}$ ). This translates to a flash-read bandwidth of  $16 \text{ bits} \times 10 \text{ MHz} = 160 \text{ Mbps}$ . Hence, the configuration time is limited by the flash-read time.

When configuration data is compressed, the amount of data that needs to be read out of the flash is reduced by about 50%. If 16 bits of compressed data yields 30 bits of uncompressed data, the flash-read bandwidth increases to  $30 \text{ bits} \times 10 \text{ MHz} = 300 \text{ Mbps}$ , reducing overall configuration time.

You can enable the controller's decompression feature in the Quartus II software, **Configuration Device Options** window by turning on **Compression Mode**.



The decompression feature supported in the enhanced configuration devices is different from the decompression feature supported by the Stratix II FPGAs and the Cyclone series. When configuring Stratix II FPGAs or the Cyclone series using enhanced configuration devices, Altera recommends enabling decompression in Stratix II FPGAs or the Cyclone series only for faster configuration.

The compression algorithm used in Altera devices is optimized for FPGA configuration bitstreams. Since FPGAs have several layers of routing structures (for high performance and easy routability), large amounts of resources go unused. These unused routing and logic resources as well as un-initialized memory structures result in a large number of configuration RAM bits in the disabled state. Altera's proprietary compression algorithm takes advantage of such bitstream qualities.

The general guideline for effectiveness of compression is the higher the device logic/routing utilization, the lower the compression ratio (where the compression ratio is defined as the original bitstream size divided by the compressed bitstream size).

For Stratix designs, based on a suite of designs with varying amounts of logic utilization, the minimum compression ratio was observed to be 1.9 or a  $\sim 47\%$  size reduction for these designs. [Table 1-6](#) lists sample compression ratios from a suite of Stratix designs. These numbers serve as a guideline (not a specification) to help you allocate sufficient configuration memory to store compressed bitstreams.

**Table 1-6.** Stratix Compression Ratios *(Note 1)*

Item	Minimum	Average
Logic Utilization	98%	64%
Compression Ratio	1.9	2.3
% Size Reduction	47%	57%

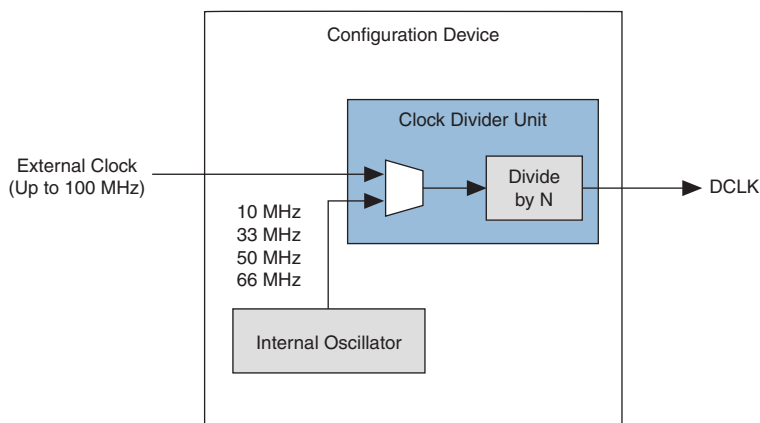
**Note to Table 1-6:**

(1) These numbers are preliminary. They are intended to serve as a guideline, not a specification.


## Programmable Configuration Clock

The configuration clock (DCLK) speed is user programmable. One of two clock sources can be used to synthesize the configuration clock; a programmable oscillator or an external clock input pin (EXCLK). The configuration clock frequency can be further synthesized using the clock divider circuitry. This clock can be divided by the N counter to generate your DCLK output. The N divider supports all integer dividers between 1 and 16, as well as a 1.5 divider and a 2.5 divider. The duty cycle for all clock divisions other than non-integer divisions is 50% (for the non-integer dividers, the duty cycle will not be 50%). Refer to [Figure 1-5](#) for a block diagram of the clock divider unit.

**Figure 1-5.** Clock Divider Unit



The DCLK frequency is limited by the maximum DCLK frequency the FPGA supports.

 The maximum DCLK input frequency supported by the FPGA is specified in the appropriate FPGA family chapter in the *Configuration Handbook*.

The controller chip features a programmable oscillator that can output four different frequencies. The various settings generate clock outputs at frequencies as high as 10, 33, 50, and 66 MHz, as listed in [Table 1-7](#).

**Table 1-7.** Internal Oscillator Frequencies

Frequency Setting	Min (MHz)	Typ (MHz)	Max (MHz)
10	6.4	8.0	10.0
33	21.0	26.5	33.0
50	32.0	40.0	50.0
66	42.0	53.0	66.0

Clock source, oscillator frequency, and clock divider (N) settings can be made in the Quartus II software, by accessing the **Configuration Device Options** inside the **Device Settings** window or the **Convert Programming Files** window. The same window can be used to select between the internal oscillator and the external clock (EXCLK) input pin as your configuration clock source. The default setting selects the internal oscillator at the 10 MHz setting as the clock source, with a divide factor of 1.



For more information about making the configuration clock source, frequency, and divider settings, refer to the *Altera Enhanced Configuration Devices* chapter in volume 2 of the *Configuration Handbook*.

## Flash In-System Programming (ISP)

The flash memory inside enhanced configuration devices can be programmed in-system via the JTAG interface and the external flash interface. JTAG-based programming is facilitated by the configuration controller in the enhanced configuration device. External flash interface programming requires an external processor or FPGA to control the flash.



The enhanced configuration device flash memory supports 100,000 erase cycles.

### JTAG-based Programming

The IEEE Std. 1149.1 JTAG Boundary Scan is implemented in enhanced configuration devices to facilitate the testing of its interconnection and functionality. Enhanced configuration devices also support the ISP mode. The enhanced configuration device is compliant with the IEEE Std. 1532 draft 2.0 specification.

The JTAG unit of the configuration controller communicates directly with the flash memory. The controller processes the ISP instructions and performs the necessary flash operations. The enhanced configuration devices support a maximum JTAG TCK frequency of 10 MHz.

During JTAG-based ISP, the external flash interface is not available. Before the JTAG interface programs the flash memory, an optional JTAG instruction (PENDCFG) can be used to assert the FPGA's *nCONFIG* pin (via the *nINIT\_CONF* pin). This will keep the FPGA in reset and terminate any internal flash access. This function prevents contention on the flash pins when both JTAG ISP and an external FPGA or processor try to access the flash simultaneously. The *nINIT\_CONF* pin is released when the initiate configuration (*nINIT\_CONF*) JTAG instruction is updated. As a result, the FPGA is configured with the new configuration data stored in flash.

An initiate configuration (*nINIT\_CONF*) JTAG instruction can be added to your programming file in the Quartus II software by enabling the **Initiate configuration after programming** option in the **Programmer options** window (Options menu).

### Programming via External Flash Interface

This method allows parallel programming of the flash memory (using the 16-bit data bus). An external processor or FPGA acts as the flash controller and has access to programming data (via a communication link such as UART, Ethernet, and PCI). In addition to the program, erase, and verify operations, the external flash interface supports block/sector protection instructions.



For information about protection commands, areas, and lock bits, refer to the appropriate flash data sheets.

- For Micron flash-based EPC4, refer to the *Micron Flash Memory MT28F400B3 Data Sheet* at [www.micron.com](http://www.micron.com).
- For Sharp flash-based EPC16, refer to the *Sharp LHF16J06 Data Sheet Flash Memory Used in EPC16 Devices* at [www.sharpsma.com](http://www.sharpsma.com).
- For the *Intel Advanced Boot Block Flash Memory (B3) 28F008/800B3, 28F016/160B3, 28F320B3, 28F640B3 Datasheet*, visit [www.intel.com](http://www.intel.com).

External flash interface programming is only allowed when the configuration controller has relinquished flash access (by tri-stating its internal interface). If the controller has not relinquished flash access (during configuration or JTAG-based ISP), you must hold the controller in reset before initiating external programming. The controller can be reset by holding the FPGA *nCONFIG* line at a logic low level. This keeps the controller in reset by holding the *nSTATUS-OE* line low, allowing external flash access.



If initial programming of the enhanced configuration device is done in-system via the external flash interface, the controller must be kept in reset by driving the FPGA *nCONFIG* line low to prevent contention on the flash interface.

## Pin Description

Table 1-8 through Table 1-10 describe the enhanced configuration device pins. These tables include configuration interface pins, external flash interface pins, JTAG interface pins, and other pins.

**Table 1-8.** Configuration Interface Pins (Part 1 of 2)

Pin Name	Pin Type	Description
DATA [7..0]	Output	Configuration data output bus. DATA changes on each falling edge of DCLK. DATA is latched into the FPGA on the rising edge of DCLK.
DCLK	Output	The DCLK output pin from the enhanced configuration device serves as the FPGA configuration clock. DATA is latched by the FPGA on the rising edge of DCLK.



**Table 1–8.** Configuration Interface Pins (Part 2 of 2)

Pin Name	Pin Type	Description
$nCS$	Input	The $nCS$ pin is an input to the enhanced configuration device and is connected to the FPGA's $CONF\_DONE$ signal for error detection after all configuration data is transmitted to the FPGA. The FPGA will always drive $nCS$ and $OE$ low when $nCONFIG$ is asserted. This pin contains a programmable internal weak pull-up resistor of $6K\ \Omega$ that can be disabled/enabled in the Quartus II software through the <b>Disable <math>nCS</math> and <math>OE</math> pull-ups on configuration device</b> option.
$nINIT\_CONF$	Open-Drain Output	The $nINIT\_CONF$ pin can be connected to the $nCONFIG$ pin on the FPGA to initiate configuration from the enhanced configuration device via a private JTAG instruction. This pin contains an internal weak pull-up resistor of $6K\ \Omega$ that is always active. The $INIT\_CONF$ 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 pull-up resistor.
$OE$	Open-Drain Bidirectional	This pin is driven low when POR is not complete. A user-selectable 2-ms or 100-ms counter holds off the release of $OE$ during initial power up to permit voltage levels to stabilize. POR time can be extended by externally holding $OE$ low. $OE$ is connected to the FPGA $nSTATUS$ signal. After the enhanced configuration device controller releases $OE$ , it waits for the $nSTATUS$ - $OE$ line to go high before starting the FPGA configuration process. This pin contains a programmable internal weak pull-up resistor of $6K\ \Omega$ that can be disabled/enabled in the Quartus II software through the <b>Disable <math>nCS</math> and <math>OE</math> pull-ups on configuration device</b> option.

**Table 1–9.** External Flash Interface Pins (Part 1 of 3)

Pin Name	Pin Type	Description
$A[20..0]$	Input	<p>These pins are the address input to the flash memory for read and write operations. The addresses are internally latched during a write cycle.</p> <p>When the external flash interface is not used, leave these pins floating (with a few exceptions (1)). These flash address, data, and control pins are internally connected to the configuration controller.</p> <p>In the 100-pin PQFP package, four address pins (<math>A_0</math>, <math>A_1</math>, <math>A_{15}</math>, <math>A_{16}</math>) are not internally connected to the controller. These loop-back connections must be made on the board between the <math>C-A[]</math> and <math>F-A[]</math> pins even when not using the external flash interface. All other address pins are connected internal to the package.</p> <p>All address pins are connected internally in the 88-pin UFBGA package.</p> <p>Pin <math>A_{20}</math> in EPC16 devices, pins <math>A_{20}</math> and <math>A_{19}</math> in EPC8 devices, and pins <math>A_{20}</math>, <math>A_{19}</math>, and <math>A_{18}</math> in EPC4 devices are no-connects. These pins should be left floating on the board.</p>
$DQ[15..0]$	Bidirectional	<p>This is the flash data bus interface between the flash memory and the controller. The controller or an external source drives <math>DQ[15..0]</math> during the flash command and the data write bus cycles. During the data read cycle, the flash memory drives the <math>DQ[15..0]</math> to the controller or external device.</p> <p>Leave these pins floating on the board when the external flash interface is not used.</p>
$CE\#$	Input	<p>Active low flash input pin that activates the flash memory when asserted. When it is high, it deselected the device and reduces power consumption to standby levels. This flash input pin is internally connected to the controller.</p> <p>Leave this pin floating on the board when the external flash interface is not used.</p>



**Table 1-9.** External Flash Interface Pins (Part 2 of 3)

Pin Name	Pin Type	Description
RP# (1)	Input	<p>Active low flash input pin that resets the flash when asserted. When high, it enables normal operation. When low, it inhibits write operation to the flash memory, which provides data protection during power transitions.</p> <p>This flash input is not internally connected to the controller. Hence, an external loop-back connection between C-RP# and F-RP# must be made on the board even when you are not using the external flash interface.</p> <p>When using the external flash interface, connect the external device to the RP# pin with the loop back. Always tri-state RP# when the flash is not in use.</p>
OE#	Input	<p>Active-low flash-control input that is asserted by the controller or external device during flash read cycles. When asserted, it enables the drivers of the flash output pins.</p> <p>Leave this pin floating on the board when the external flash interface is not used.</p>
WE# (1)	Input	<p>Active-low flash-write strobe asserted by the controller or external device during flash write cycles. When asserted, it controls writes to the flash memory. In the flash memory, addresses and data are latched on the rising edge of the WE# pulse.</p> <p>This flash input is not internally connected to the controller. Hence, an external loop-back connection between C-WE# and F-WE# must be made on the board even when you are not using the external flash interface.</p> <p>When using the external flash interface, connect the external device to the WE# pin with the loop back.</p>
WP#	Input	<p>Usually tied to V<sub>CC</sub> or ground on the board. The controller does not drive this pin because it could cause contention.</p> <p>Connection to V<sub>CC</sub> is recommended for faster block erase/programming times and to allow programming of the flash-bottom boot block, which is required when programming the device using the Quartus II software.</p> <p>This pin should be connected to V<sub>CC</sub> even when the external flash interface is not used.</p>
V <sub>CCW</sub>	Supply	<p>Block erase, full-chip erase, word write, or lock-bit configuration power supply.</p> <p>Connect this pin to the 3.3-V V<sub>CC</sub> supply, even when you are not using the external flash interface.</p>
RY/BY#	Open-Drain Output	<p>Flash asserts this pin when a write or erase operation is complete. This pin is not connected to the controller. RY/BY# is only available in Sharp flash-based EPC8 and EPC16. (2)</p> <p>Leave this pin floating when the external flash interface is not used.</p>

**Table 1–9.** External Flash Interface Pins (Part 3 of 3)

Pin Name	Pin Type	Description
BYTE#	Input	Flash byte-enable pin and is only available for enhanced configuration devices in the 100-pin PQFP package.  This pin must be connected to $V_{CC}$ on the board even when you are not using the external flash interface (the controller uses the flash in 16-bit mode). For Intel flash-based EPC device, this pin is connected to the $V_{CCQ}$ of the Intel flash die internally. Therefore, <b>BYTE#</b> must be connected directly to $V_{CC}$ without using any pull-up resistor.

**Notes to Table 1–9:**

- (1) These pins can be driven to 12 V during production testing of the flash memory. Since the controller cannot tolerate the 12-V level, connections from the controller to these pins are not made internal to the package. Instead they are available as two separate pins. You must connect the two pins at the board level (for example, on the printed circuit board (PCB), connect the **C-WE#** pin from controller to **F-WE#** pin from the flash memory).
- (2) For more information, refer to the Process Change Notification *PCN0506: Addition of Intel Flash Memory As Source For EPC4, EPC8 and EPC16 Enhanced Configuration Devices* and *Using the Intel Flash Memory-Based EPC4, EPC8 and EPC16* white paper.

**Table 1–10.** JTAG Interface Pins and Other Required Controller Pins

Pin Name	Pin Type	Description
TDI	Input	JTAG data input pin. Connect this pin to $V_{CC}$ if the JTAG circuitry is not used.
TDO	Output	JTAG data output pin. Do not connect this pin if the JTAG circuitry is not used (leave floating).
TCK	Input	JTAG clock pin. Connect this pin to GND if the JTAG circuitry is not used.
TMS	Input	JTAG mode select pin. Connect this pin to $V_{CC}$ if the JTAG circuitry is not used.
PGM[2..0]	Input	These three input pins select one of the eight pages of configuration data to configure the FPGAs in the system.  Connect these pins on the board to select the page specified in the Quartus II software when generating the enhanced configuration device POF. <b>PGM[2]</b> is the MSB. The default selection is page 0; <b>PGM[2..0]</b> = 000. These pins must not be left floating.
EXCLK	Input	Optional external clock input pin that can be used to generate the configuration clock ( <b>DCLK</b> ).  When an external clock source is not used, connect this pin to a valid logic level (high or low) to prevent a floating-input buffer. If <b>EXCLK</b> is used, toggling the <b>EXCLK</b> input pin after the FPGA enters user mode will not effect the enhanced configuration device operation.
PORSEL	Input	This pin selects a 2-ms or 100-ms POR counter delay during power up. When <b>PORSEL</b> is low, POR time is 100 ms. When <b>PORSEL</b> is high, POR time is 2 ms. This pin must be connected to a valid logic level.
TMO	Input	For normal operation, this test pin must be connected to GND.
TM1	Input	For normal operating, this test pin must be connected to $V_{CC}$ .

## Power-On Reset

The POR circuit keeps the system in reset until power-supply voltage levels have stabilized. The POR time consists of the  $V_{CC}$  ramp time and a user-programmable POR delay counter. When the supply is stable and the POR counter expires, the POR circuit releases the OE pin. The POR time can be further extended by an external device by driving the OE pin low.



Do not execute JTAG or ISP instructions until POR is complete.

The enhanced configuration device supports a programmable POR delay setting. You can set the POR delay to the default 100-ms setting or reduce the POR delay to 2 ms for systems that require fast power-up. The PORSEL input pin controls this POR delay; a logic-high level selects the 2-ms delay, while a logic-low level selects the 100-ms delay.

The enhanced configuration device can enter reset under the following conditions:

- The POR reset starts at initial power-up during  $V_{CC}$  ramp-up or if  $V_{CC}$  drops below the minimum operating condition anytime after  $V_{CC}$  has stabilized
- The FPGA initiates reconfiguration by driving  $nSTATUS$  low, which occurs if the FPGA detects a CRC error or if the FPGA's  $nCONFIG$  input pin is asserted
- The controller detects a configuration error and asserts OE to begin re-configuration of the Altera FPGA (for example, when CONF\_DONE stays low after all configuration data has been transmitted)

## Power Sequencing

Altera requires that you power-up the FPGA's  $V_{CCINT}$  supply before the enhanced configuration device's POR expires.

Power up needs to be controlled so that the enhanced configuration device's OE signal goes high after the CONF\_DONE signal is pulled low. If the EEPIC device exits POR before the FPGA is powered up, the CONF\_DONE signal will be high because the pull-up resistor is holding this signal high. When the enhanced configuration device exits POR, OE is released and pulled high by a pull-up resistor. Since the enhanced configuration device samples the  $nCS$  signal on the rising edge of OE, it detects a high level on CONF\_DONE and enters an idle mode. DATA and DCLK outputs will not toggle in this state and configuration will not begin. The enhanced configuration device will only exit this mode if it is powered down and then powered up correctly.




To ensure the enhanced configuration device enters configuration mode properly, you must ensure that the FPGA completes power-up before the enhanced configuration device exits POR.

The pin-selectable POR time feature is useful for ensuring this power-up sequence. The enhanced configuration device has two POR settings, 2 ms when PORSEL is set to a high level and 100 ms when PORSEL is set to a low level. For more margin, the 100-ms setting can be selected to allow the FPGA to power-up before configuration is attempted.

Alternatively, a power-monitoring circuit or a power-good signal can be used to keep the FPGA's `nCONFIG` pin asserted low until both supplies have stabilized. This ensures the correct power up sequence for successful configuration.

## Programming and Configuration File Support

The Quartus II software provides programming support for the enhanced configuration device and automatically generates the `.pof` for the EPC4, EPC8, and EPC16 devices. In a multi-device project, the software can combine the `.sof` for multiple ACEX 1K, APEX 20K, APEX II, Cyclone series, FLEX 10K, Mercury, and Stratix series FPGAs into one programming file for the enhanced configuration device.

 For details about generating programming files, refer to the *Altera Enhanced Configuration Devices* chapter and the *Software Settings* section in volume 2 of the *Configuration Handbook*.

Enhanced configuration devices can be programmed in-system through the industry-standard 4-pin JTAG interface. The ISP feature in the enhanced configuration device provides ease in prototyping and updating FPGA functionality.

After programming an enhanced configuration device in-system, FPGA configuration can be initiated by including the enhanced configuration device's JTAG `INIT_CONF` instruction (Table 1-11).

The ISP circuitry in the enhanced configuration device is compliant with the IEEE Std. 1532 specification. The IEEE Std. 1532 is a standard that allows concurrent ISP between devices from multiple vendors.

**Table 1-11.** Enhanced Configuration Device JTAG Instructions (Part 1 of 2) (Note 1)

JTAG Instruction	OPCODE	Description
SAMPLE/ PRELOAD	00 0101 0101	Allows a snapshot of the state of the enhanced configuration device pins to be captured and examined during normal device operation and permits an initial data pattern output at the device pins.
EXTEST	00 0000 0000	Allows the external circuitry and board-level interconnections to be tested by forcing a test pattern at the output pins and capturing results at the input pins.
BYPASS	11 1111 1111	Places the 1-bit bypass register between the TDI and the TDO pins, which allow the BST data to pass synchronously through a selected device to adjacent devices during normal device operation.
IDCODE	00 0101 1001	Selects the device IDCODE register and places it between TDI and TDO, allowing the device IDCODE to be serially shifted out to TDO. The device IDCODE for all enhanced configuration devices is the same and shown below: 0100A0DDh
USERCODE	00 0111 1001	Selects the USERCODE register and places it between TDI and TDO, allowing the USERCODE to be serially shifted out the TDO. The 32-bit USERCODE is a programmable user-defined pattern.

**Table 1-11.** Enhanced Configuration Device JTAG Instructions (Part 2 of 2) (Note 1)

JTAG Instruction	OPCODE	Description
INIT_CONF	00 0110 0001	This function initiates the FPGA re-configuration process by pulsing the <i>n</i> INIT_CONF pin low, which is connected to the FPGA <i>n</i> CONFIG pin. After this instruction is updated, the <i>n</i> INIT_CONF pin is pulsed low when the JTAG state machine enters Run-Test/Idle state. The <i>n</i> INIT_CONF pin is then released and <i>n</i> CONFIG is pulled high by the resistor after the JTAG state machine goes out of Run-Test/Idle state. The FPGA configuration starts after <i>n</i> CONFIG goes high. As a result, the FPGA is configured with the new configuration data stored in flash via ISP. This function can be added to your programming file (.pof, .jam, .jbc) in the Quartus II software by enabling the <b>Initiate configuration after programming</b> option in the <b>Programmer options</b> window (Options menu).
PENDCFG	00 0110 0101	This optional function can be used to hold the <i>n</i> INIT_CONF pin low during JTAG-based ISP of the enhanced configuration device. This feature is useful when the external flash interface is controlled by an external FPGA/processor. This function prevents contention on the flash pins when both the controller and external device try to access the flash simultaneously. Before the enhanced configuration device's controller can access the flash memory, the external FPGA/processor needs to tri-state its interface to flash. This can be ensured by resetting the FPGA using the <i>n</i> INIT_CONF, which drives the <i>n</i> CONFIG pin and keeps the external FPGA/processor in the "reset" state. The <i>n</i> INIT_CONF pin is released when the initiate configuration (INIT_CONF) JTAG instruction is issued.

**Note to Table 1-11:**

(1) Enhanced configuration device instruction register length is 10 and boundary scan length is 174.



For more information about the enhanced configuration device JTAG support, refer to the BSDL files provided at the Altera website.

Enhanced configuration devices can also be programmed by third-party flash programmers or on-board processors using the external flash interface. Programming files (.pof) can be converted to an Intel HEX format file (.hexout) using the Quartus II **Convert Programming Files** utility, for use with the programmers or processors.

You can also program the enhanced configuration devices using the Quartus II software, the Altera Programming Unit (APU), and the appropriate configuration device programming adapter. Table 1-12 lists which programming adapter to use with each enhanced configuration device.

**Table 1-12.** Programming Adapters

Device	Package	Adapter
EPC16	88-pin UFBGA	PLMUEPC-88
	100-pin PQFP	PLMQEPC-100
EPC8	100-pin PQFP	PLMQEPC-100
EPC4	100-pin PQFP	PLMQEPC-100

## IEEE Std. 1149.1 (JTAG) Boundary-Scan

The enhanced configuration device provides JTAG BST circuitry that complies with the IEEE Std. 1149.1-1990 specification. JTAG boundary-scan testing can be performed before or after configuration, but not during configuration.

Figure 1-6 shows the timing requirements for the JTAG signals.

**Figure 1-6.** JTAG Timing Waveforms

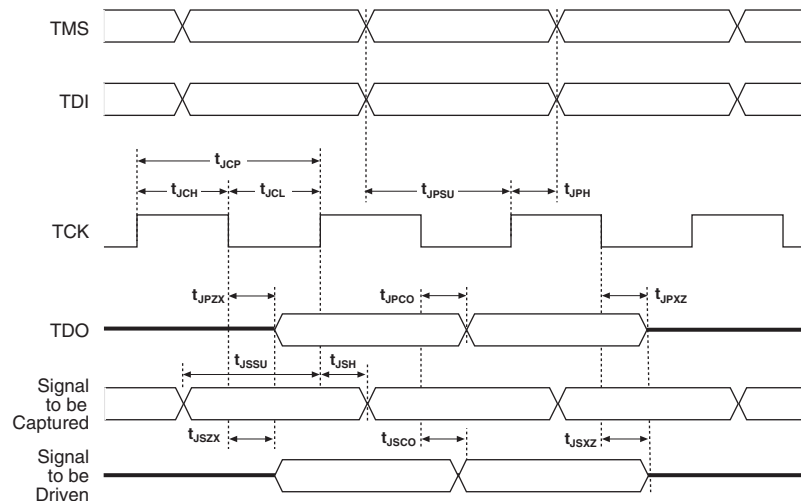


Table 1-13 lists the timing parameters and values for the enhanced configuration device.

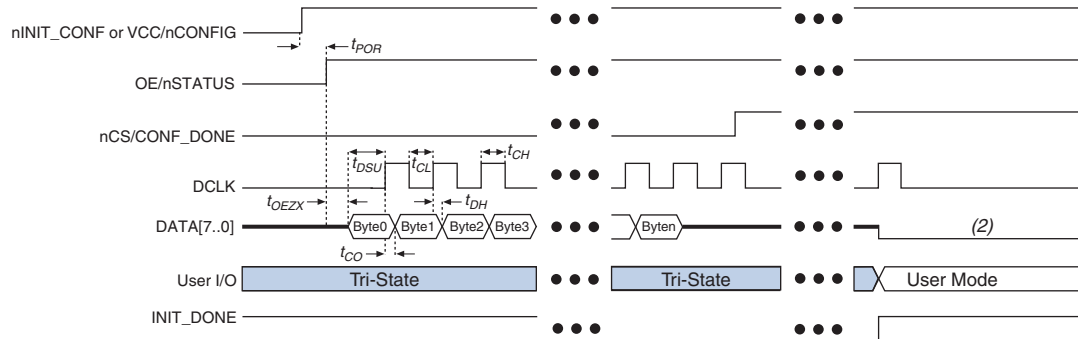
**Table 1-13.** JTAG Timing Parameters and Values

Symbol	Parameter	Min	Max	Unit
$t_{JCP}$	TCK clock period	100	—	ns
$t_{JCH}$	TCK clock high time	50	—	ns
$t_{JCL}$	TCK clock low time	50	—	ns
$t_{JPSU}$	JTAG port setup time	20	—	ns
$t_{JPH}$	JTAG port hold time	45	—	ns
$t_{JPCO}$	JTAG port clock output	—	25	ns
$t_{JPZX}$	JTAG port high impedance to valid output	—	25	ns
$t_{JPXZ}$	JTAG port valid output to high impedance	—	25	ns
$t_{JSSU}$	Capture register setup time	20	—	ns
$t_{JSH}$	Capture register hold time	45	—	ns
$t_{JSCO}$	Update register clock to output	—	25	ns
$t_{JSZX}$	Update register high-impedance to valid output	—	25	ns
$t_{JSXZ}$	Update register valid output to high impedance	—	25	ns

## Timing Information

Figure 1-7 shows the configuration timing waveform when using an enhanced configuration device.

**Figure 1-7.** Configuration Timing Waveform Using an Enhanced Configuration Device



**Notes to Figure 1-7:**

- (1) The enhanced configuration device will drive DCLK low after configuration.
- (2) The enhanced configuration device will drive DATA [] high after configuration.

Table 1-14 defines the timing parameters when using the enhanced configuration devices.

For more information about the flash memory (external flash interface) timing, refer to the appropriate flash data sheet on the Altera website at [www.altera.com](http://www.altera.com).

- For Micron flash-based EPC4, refer to the *Micron MT28F400B3 Data Sheet Flash Memory Used in EPC4 Devices* at [www.micron.com](http://www.micron.com).
- For Sharp flash-based EPC16, refer to the *Sharp LHF16J06 Data Sheet Flash Memory Used in EPC16 Devices* at [www.sharpsma.com](http://www.sharpsma.com).
- For Intel flash-based EPC4 and EPC16, refer to Intel Flash 28F016B3 at [www.intel.com](http://www.intel.com).

**Table 1-14.** Enhanced Configuration Device Configuration Parameters (Part 1 of 2)

Symbol	Parameter	Condition	Min	Typ	Max	Unit
f <sub>DCLK</sub>	DCLK frequency	40% duty cycle	—	—	66.7	MHz
t <sub>DCLK</sub>	DCLK period	—	15	—	—	ns
t <sub>HC</sub>	DCLK duty cycle high time	40% duty cycle	6	—	—	ns
t <sub>LC</sub>	DCLK duty cycle low time	40% duty cycle	6	—	—	ns
t <sub>CE</sub>	OE to first DCLK delay	—	40	—	—	ns
t <sub>OE</sub>	OE to first DATA available	—	40	—	—	ns
t <sub>OH</sub>	DCLK rising edge to DATA change	—	(1)	—	—	ns
t <sub>CF</sub> (2)	OE assert to DCLK disable delay	—	277	—	—	ns
t <sub>DF</sub> (2)	OE assert to DATA disable delay	—	277	—	—	ns
t <sub>RE</sub> (3)	DCLK rising edge to OE	—	60	—	—	ns

**Table 1-14.** Enhanced Configuration Device Configuration Parameters (Part 2 of 2)

Symbol	Parameter	Condition	Min	Typ	Max	Unit
$t_{LOE}$	OE assert time to assure reset	—	60	—	—	ns
$f_{ECLK}$	EXCLK input frequency	40% duty cycle	—	—	100	MHz
$t_{ECLK}$	EXCLK input period	—	10	—	—	ns
$t_{ECLKH}$	EXCLK input duty cycle high time	40% duty cycle	4	—	—	ns
$t_{ECLKL}$	EXCLK input duty cycle low time	40% duty cycle	4	—	—	ns
$t_{ECLKR}$	EXCLK input rise time	100 MHz	—	—	3	ns
$t_{ECLKF}$	EXCLK input fall time	100 MHz	—	—	3	ns
$t_{POR}^{(4)}$	POR time	2 ms	1	2	3	ms
		100 ms	70	100	120	ms

**Notes to Table 1-14:**

- (1) To calculate  $t_{OH}$ , use the following equation:  $t_{OH} = 0.5$  (DCLK period) - 2.5 ns.
- (2) This parameter is used for CRC error detection by the FPGA.
- (3) This parameter is used for CONF\_DONE error detection by the enhanced configuration device.
- (4) The FPGA  $V_{CCINT}$  ramp time should be less than 1-ms for 2-ms POR, and it should be less than 70 ms for 100-ms POR.

## Operating Conditions

Table 1-15 through Table 1-19 provide information about absolute maximum ratings, recommended operating conditions, DC operating conditions, supply current values, and pin capacitance data for the enhanced configuration devices.

**Table 1-15.** Enhanced Configuration Device Absolute Maximum Rating

Symbol	Parameter	Condition	Min	Max	Unit
$V_{CC}$	Supply voltage	With respect to ground	-0.2	4.6	V
$V_I$	DC input voltage	With respect to ground	-0.5	3.6	V
$I_{MAX}$	DC $V_{CC}$ or ground current	—	—	100	mA
$I_{OUT}$	DC output current, per pin	—	-25	25	mA
$P_D$	Power dissipation	—	—	360	mW
$T_{STG}$	Storage temperature	No bias	-65	150	C
$T_{AMB}$	Ambient temperature	Under bias	-65	135	C
$T_J$	Junction temperature	Under bias	—	135	C

**Table 1-16.** Enhanced Configuration Device Recommended Operating Conditions

Symbol	Parameter	Condition	Min	Max	Unit
$V_{CC}$	Supplies voltage for 3.3-V operation	—	3.0	3.6	V
$V_I$	Input voltage	With respect to ground	-0.3	$V_{CC} + 0.3$	V
$V_O$	Output voltage	—	0	$V_{CC}$	V
$T_A$	Operating temperature	For commercial use	0	70	C
		For industrial use	-40	85	C
$T_R$	Input rise time	—	—	20	ns
$T_F$	Input fall time	—	—	20	ns



**Table 1–17.** Enhanced Configuration Device DC Operating Conditions

Symbol	Parameter	Condition	Min	Typ	Max	Unit
$V_{CC}$	Supplies voltage to core	—	3.0	3.3	3.6	V
$V_{IH}$	High-level input voltage	—	2.0	—	$V_{CC} + 0.3$	V
$V_{IL}$	Low-level input voltage	—	—	—	0.8	V
$V_{OH}$	3.3-V mode high-level TTL output voltage	$I_{OH} = -4$ mA	2.4	—	—	V
	3.3-V mode high-level CMOS output voltage	$I_{OH} = -0.1$ mA	$V_{CC} - 0.2$	—	—	V
$V_{OL}$	Low-level output voltage TTL	$I_{OL} = -4$ mA DC	—	—	0.45	V
	Low-level output voltage CMOS	$I_{OL} = -0.1$ mA DC	—	—	0.2	V
$I_I$	Input leakage current	$V_I = V_{CC}$ or ground	-10	—	10	$\mu$ A
$I_{OZ}$	Tri-state output off-state current	$V_O = V_{CC}$ or ground	-10	—	10	$\mu$ A

**Table 1–18.** Enhanced Configuration Device  $I_{CC}$  Supply Current Values

Symbol	Parameter	Condition	Min	Typ	Max	Unit
$I_{CC0}$	Current (standby)	—	—	50	150	$\mu$ A
$I_{CC1}$	$V_{CC}$ supply current (during configuration)	—	—	60	90	mA
$I_{CCW}$	$V_{CCW}$ supply current	—	—	(1)	(1)	—

**Note to Table 1–18:**(1) For  $V_{CCW}$  supply current information, refer to the appropriate flash memory data sheet at [www.altera.com](http://www.altera.com).**Table 1–19.** Enhanced Configuration Device Capacitance

Symbol	Parameter	Condition	Min	Max	Unit
CIN	Input pin capacitance	—	—	10	pF
COUT	Output pin capacitance	—	—	10	pF

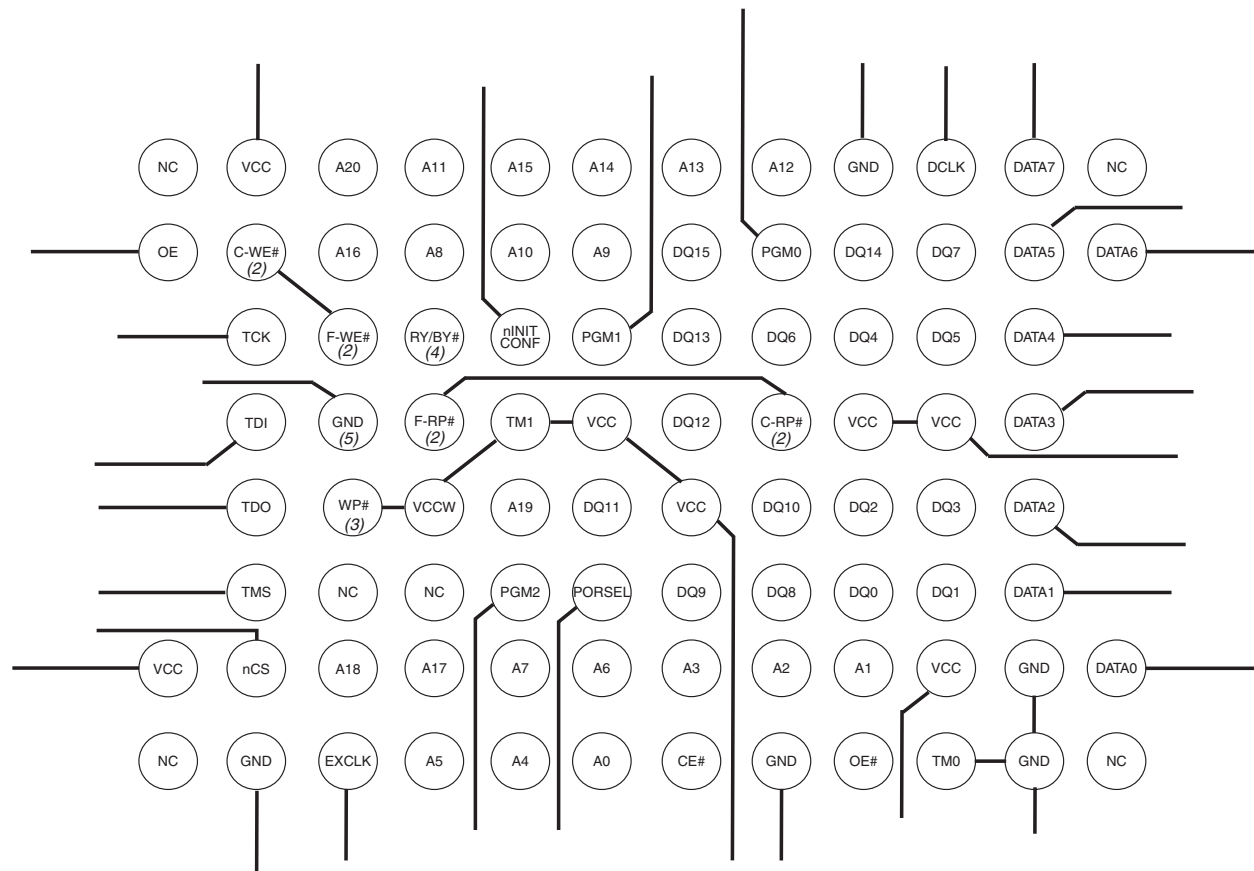
## Package

The EPC16 enhanced configuration device is available in both the 88-pin UFBGA package and the 100-pin PQFP package. The UFBGA package, which is based on 0.8-mm ball pitch, maximizes board space efficiency. A board can be laid out for this package using a single PCB layer. The EPC8 and EPC4 devices are available in the 100-pin PQFP package.

Enhanced configuration devices support vertical migration in the 100-pin PQFP package.

Figure 1-8 shows the PCB routing for the 88-pin UFBGA package. The Gerber file for this layout is on the Altera website.

**Figure 1-8.** PCB Routing for 88-Pin UFBGA Package (Note 1)



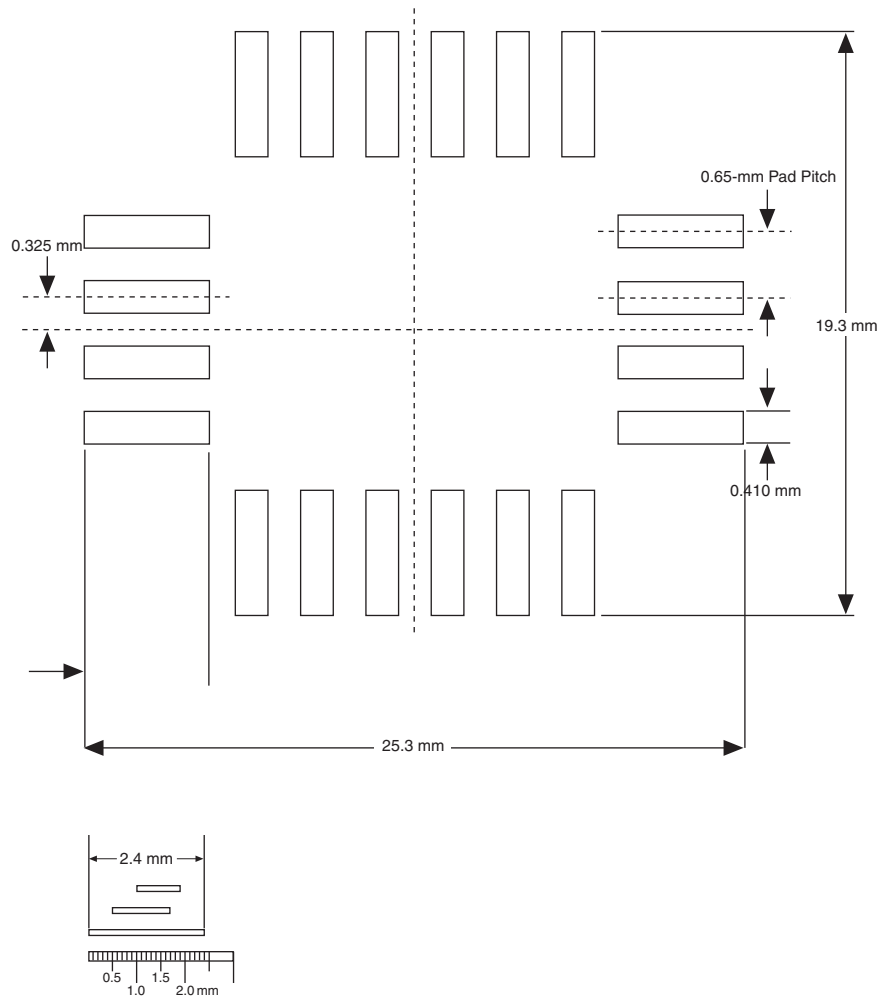
#### Notes to Figure 1-8:

- (1) If the external flash interface feature is not used, then the flash pins should be left unconnected because they are internally connected to the controller unit. The only pins that need external connections are  $WP\#$ ,  $WE\#$ , and  $RP\#$ . If the flash is being used as an external memory source, then the flash pins should be connected as outlined in the pin descriptions section.
- (2)  $F-RP\#$  and  $F-WE\#$  are pins on the flash die.  $C-RP\#$  and  $C-WE\#$  are pins on the controller die.  $C-WE\#$  and  $F-WE\#$  should be connected together on the PCB.  $F-RP\#$  and  $C-RP\#$  should also be connected together on the PCB.
- (3)  $WP\#$  (write protection pin) should be connected to a high level (3.3 V) to be able to program the flash bottom boot block, which is required when programming the device using the Quartus II software.
- (4)  $RY/BY\#$  is only available in Sharp flash-based enhanced configuration devices.
- (5) Pin D3 is a  $NC$  pin for Intel Flash-based EPC16.

## Package Layout Recommendation


Sharp flash-based EPC16 and EPC8 enhanced configuration devices in the 100-pin PQFP packages have different package dimensions than other Altera 100-pin PQFP devices (including the Micron flash-based EPC4, Intel flash-based EPC16, EPC8 and EPC4). Figure 1-9 shows the 100-pin PQFP PCB footprint specifications for enhanced configuration devices that allows for vertical migration between all devices. These footprint dimensions are based on vendor-supplied package outline diagrams.

**Figure 1-9.** Enhanced Configuration Device PCB Footprint Specifications for 100-Pin PQFP Packages (Note 1), (2)



**Notes to Figure 1-9:**

- (1) Used 0.5-mm increase for front and back of nominal foot length.
- (2) Used 0.3-mm increase to maximum foot width.

 For package outline drawings, refer to the *Altera Device Package Information Data Sheet*.

## Device Pin-Outs

 For pin-out information, refer to *Altera Configuration Devices Pin-Out Files*.

## Chapter Revision History

Table 1–20 lists the revision history for this chapter.

**Table 1–20.** Chapter Revision History

Date	Version	Changes Made
December 2009	2.8	<ul style="list-style-type: none"> <li>■ Added Table 1–1 and Table 1–2.</li> <li>■ Updated Table 1–17 and Table 1–18.</li> <li>■ Removed “Referenced Documents” section.</li> </ul>
October 2008	2.7	<ul style="list-style-type: none"> <li>■ Updated Table 2–1, Table 2–7, and Table 2–8.</li> <li>■ Updated Figure 2–2, Figure 2–3, and Figure 2–4.</li> <li>■ Updated “JTAG-based Programming” section.</li> <li>■ Added “Intel-Flash-Based EPC Device Protection” section.</li> <li>■ Updated new document format.</li> </ul>
May 2008	2.6	<ul style="list-style-type: none"> <li>■ Minor textual and style changes. Added “Referenced Documents” section.</li> </ul>
February 2008	2.5	<ul style="list-style-type: none"> <li>■ Updated Table 2–18 with information about EPC16UI88AA.</li> </ul>
May 2007	2.4	<ul style="list-style-type: none"> <li>■ Added “Intel-Flash-Based EPC Device Protection” section.</li> </ul>
April 2007	2.3	<ul style="list-style-type: none"> <li>■ Added document revision history.</li> </ul>
October 2005	2.2	<ul style="list-style-type: none"> <li>■ Made changes to content.</li> </ul>
July 2004	2.0	<ul style="list-style-type: none"> <li>■ Added Stratix II and Cyclone II device information throughout chapter.</li> <li>■ Updated VCCW connection in Figure 2–2, Figure 2–3, and Figure 2–4.</li> <li>■ Updated (Note 2) of Figure 2–2, Figure 2–3, and Figure 2–4.</li> <li>■ Updated (Note 4) of Table 2–12.</li> <li>■ Updated unit of ICC0 in Table 2–16.</li> <li>■ Added ICCW to Table 2–16.</li> </ul>
September 2003	1.0	<ul style="list-style-type: none"> <li>■ Initial Release.</li> </ul>

This chapter describes the latest enhanced configuration device flash memory standard with a feature-rich configuration controller. A single-chip configuration solution provides you with several new and advanced features that significantly reduce configuration times. This chapter discusses the hardware and software implementation of enhanced configuration device features such as concurrent and dynamic configuration, data compression, clock division, and an external flash memory interface. Enhanced configuration devices include EPC4, EPC8, and EPC16 devices.

## Concurrent Configuration

Configuration data is transmitted from the enhanced configuration device to the SRAM-based device on the DATA lines. The DATA lines are outputs on the enhanced configuration devices, and inputs to the SRAM-based devices.

These DATA lines correspond to the `Bit $n$`  lines in the **Convert Programming Files** window in the Altera® Quartus® II software. For example, if you specify a SRAM Object File (`.sof`) to use `Bit0` in the Quartus II software, that `.sof` will be transmitted on the DATA [0] line from the enhanced configuration device to the SRAM-based device.

## Supported Schemes and Guidelines


There are several different ways to configure Altera SRAM-based programmable logic devices (PLDs) with enhanced configuration devices:

- 1-bit passive serial (PS)
- 2-bit PS
- 4-bit PS
- 8-bit PS
- Fast passive parallel (FPP)

Additionally, you can use these configuration schemes in conjunction with the dynamic configuration option (previously called page mode operation) for sophisticated configuration setups.

FPP configuration mode uses the eight DATA [7 . . 0] lines from the enhanced configuration device, which can be used to configure APEX™ II and Stratix® series devices. To decrease configuration time, FPP configuration provides eight bits of configuration data per clock cycle to the target device.


 For more information about configuration schemes, refer to the *Enhanced Configuration Devices (EPC4, EPC8, and EPC16) Data Sheet*, or *Configuring Stratix & Stratix GX Devices*.

 For more information about additional enhanced configuration devices, refer to the Process Change Notification *PCN0506: Addition of Intel Flash Memory As Source For EPC4, EPC8 and EPC16 Enhanced Configuration Devices* and the *Using the Intel Flash Memory Based EPC4, EPC8, and EPC16 Devices* white paper.

## Concurrent Configuration Using $n$ -Bit PS Modes

The  $n$ -bit ( $n = 1, 2, 4, \text{ or } 8$ ) PS configuration mode allows enhanced configuration devices to concurrently configure SRAM-based devices or device chains. In addition, these devices do not have to be the same device family or density; they can be any combination of Altera SRAM-based 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.

The Quartus II software only allows the selection of  $n$ -bit PS configuration modes. 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 2-1](#) to select the appropriate configuration mode for the fastest configuration times.

 Mode selection has an impact on the amount of memory used, as described in [“Calculating the Size of Configuration Space”](#) on page 2-14.

**Table 2-1.** Recommended Configuration Using  $n$ -Bit PS Modes

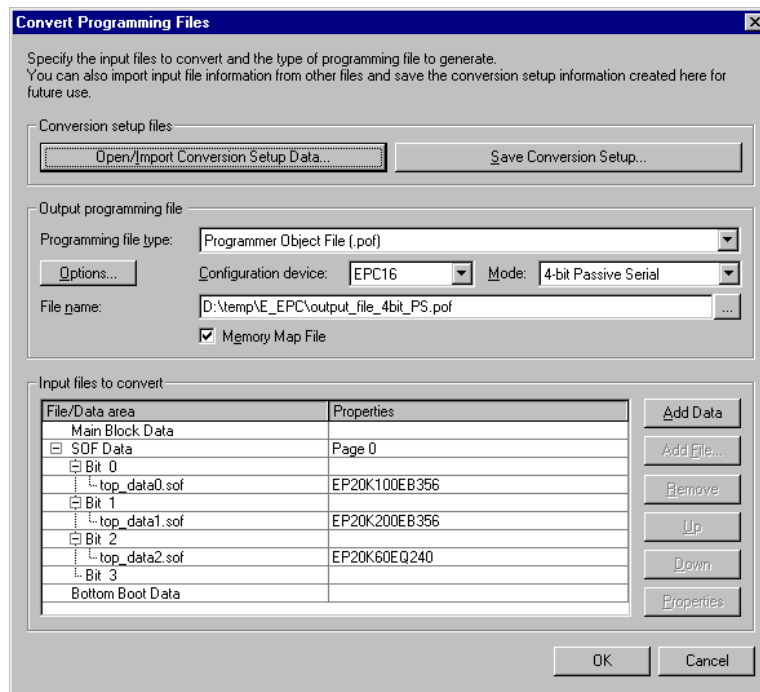
Number of Devices (1)	Recommended Configuration Mode
1	1-bit PS
2	2-bit PS
3	4-bit PS
4	4-bit PS
5	8-bit PS
6	8-bit PS
7	8-bit PS
8	8-bit PS

**Note to Table 2-1:**

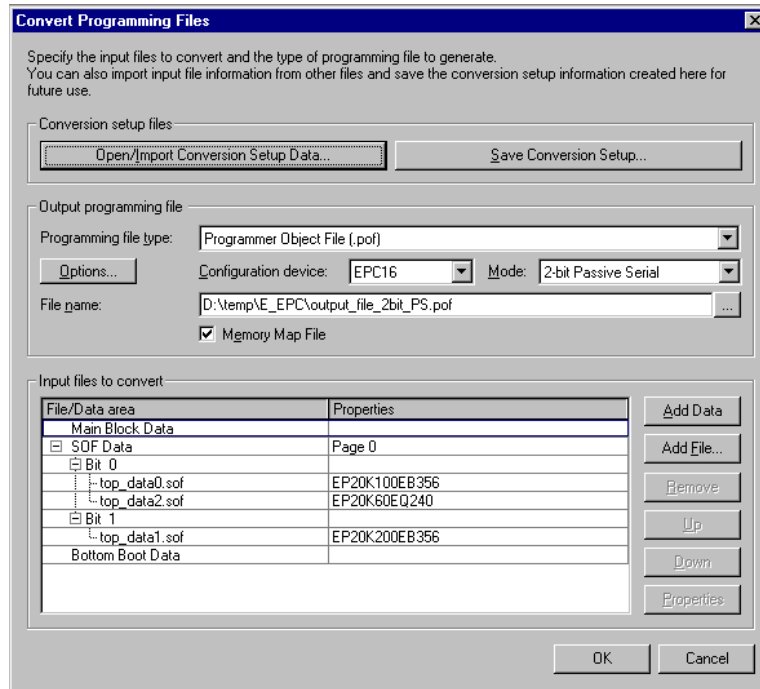
(1) Assume that each DATA line is only configuring one device, not a daisy chain of devices.

For example, if you configure three SRAM-based devices, you would use the 4-bit PS mode. For the DATA0, DATA1, and DATA2 lines, the corresponding .sof data will be transmitted from the configuration device to the SRAM-based PLD. For DATA3, you can leave the corresponding Bit3 line blank in the Quartus II software. On the printed circuit board (PCB), leave the DATA3 line from the enhanced configuration device unconnected. [Figure 2-1](#) shows the **Quartus II Convert Programming Files** window (Tools menu) setup for this scheme.

**Figure 2-1.** Software Settings for Configuring Devices Using *n*-Bit PS Modes

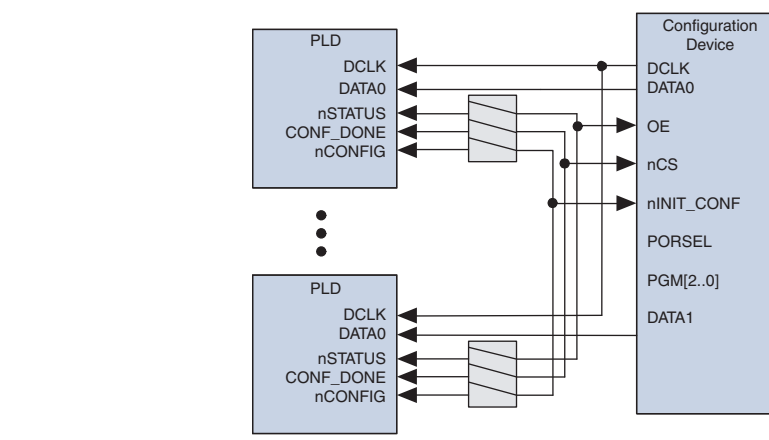


Alternatively, you can daisy chain two SRAM-based devices to one DATA line while the other DATA lines drive one device each. For example, you could use the 2-bit PS mode to drive two SRAM-based devices with DATA Bit 0 (EP20K100E and EP20K60E devices) and the third device (the EP20K200E device) with DATA Bit 1. This 2-bit PS configuration scheme requires less space in the configuration flash memory, but may increase the total system configuration time as shown in [Figure 2-2](#).

**Figure 2-2.** Daisy Chaining Two SRAM-Based Devices to One DATA Line

## Design Guidelines

For debugging, Altera recommends keeping the control lines such as *nSTATUS*, *nCONFIG*, and *CONF\_DONE* between each PLD and the configuration device separate. You can keep control lines separate with a switch to manage which control signals are fed back into the enhanced configuration device. Figure 2-3 shows an example of the connections between the enhanced configuration device and the targeted PLDs.

**Figure 2-3.** Example of Using Debugging Switches for Control Lines



## Dynamic Configuration (Page Mode) Implementation Overview

Pages in enhanced configuration devices allow you to organize and store various configurations for entire systems that use one or more Altera PLDs. This dynamic configuration (or page mode) feature allows systems to dynamically reconfigure their PLDs with different configuration files.

You can use different pages to store configuration files that support different standards (for example, I/O standards, memory). Alternatively, the different pages can place the system in different modes. For instance, page 0 could contain a configuration `.sof` for the PLD that only processes data packets; page 1 could contain a configuration file for the same PLD that processes data and voice packets.

With the ability to dynamically switch pages, you can also configure Altera devices with various revisions for debugging without having to reprogram the configuration device. For example, you can configure a device that is on “stand-by” to perform another function and then reconfigure it back with the original configuration file.

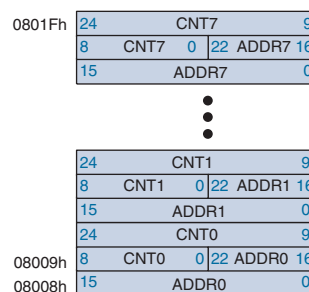
A page is a section of the flash memory space that contains configuration data for all PLDs in the system. One page stores one system configuration regardless of the number of PLDs in the system. The size of each page is dynamic and can change each time the enhanced configuration device is reprogrammed. Enhanced configuration devices support a maximum of eight pages of configuration data, or eight system configurations. The number of pages is also limited to the density of the configuration device.



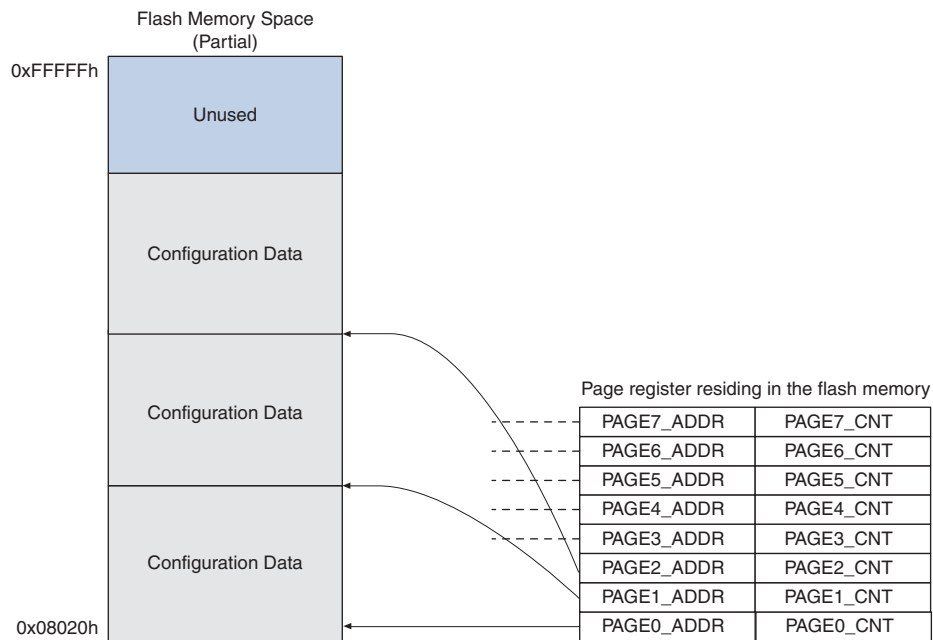
The number of pages required in a system is not dependent on the number of PLDs in the system, but depends on the number of unique system configurations.

External page mode input pins `PGM [2 . . 0]` determine which page to use during PLD configuration, and page pointers determine the data location. Each page pointer consists of a starting address register and a length count register. The word-addressable starting address register (23 bits) is used to determine where the page begins in the flash memory. The count register (25 bits) determines the length of the page counted in nibbles (group of 4 bits equaling half of a byte). [Figure 2-4](#) shows a block diagram of the option-bit space and its address locations.

**Figure 2-4.** Option-Bit Memory Map




For example, a page for the EPC16 configuration device must start between word addresses `0x08020h` and `0xFFFFFh` and cannot overlap with other pages. Refer to [Figure 2-5](#) for an EPC16 page mode example using three pages.

**Figure 2-5.** EPC16 Page Mode Implementation Example

During configuration, different pages are selected by the PGM[2..0] pins. These pins are used to select one out of eight pages (or eight system configurations). PGM[2..0] pins are sampled once before the configuration data is sent to the target PLDs.


Setting the PGM[2..0] pins to select an incorrect page (for example, a page that is non-existent or a blank page) causes the enhanced configuration device to enter an erroneous state. The only way to recover from this state is to set the PGM[2..0] pins to select a valid page and then power cycle the board.

 To ensure proper configuration, only set the PGM[2..0] pins to select valid pages.

Within each page, you can store as many configuration files as your system needs. There is no limitation to the length of a page except for the physical limitation determined by the size of the flash memory (for example, 0xFFFFFFFFh for EPC16 devices). However, all pages must be contiguous.

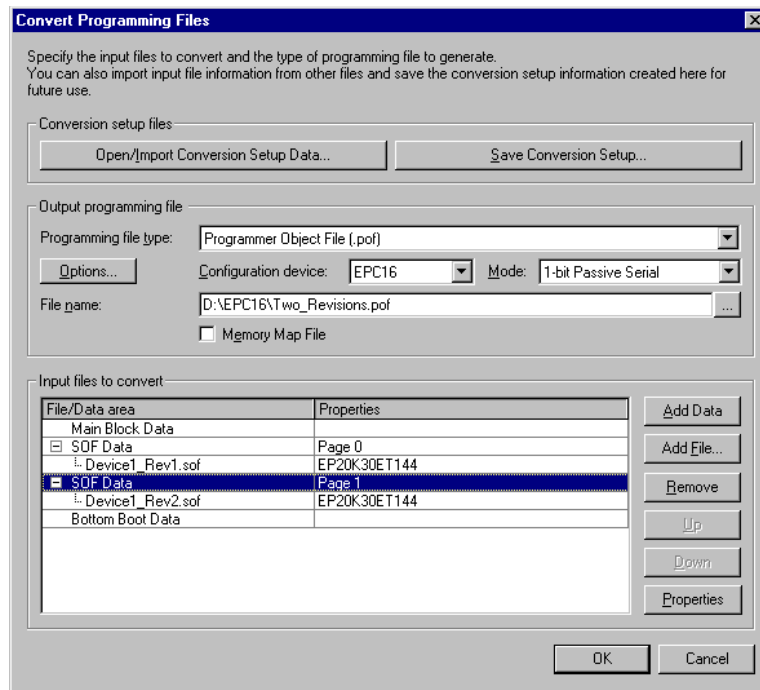
## Software Implementation (Convert Programming Files)

The **Convert Programming Files** window (Tools menu) in the Quartus II software allows you to create enhanced configuration device programmer object files (.pof) and enable the dynamic configuration feature.

 Passive parallel asynchronous (PPA) and passive parallel synchronous (PPS) configuration modes are not supported by enhanced configuration devices. If you choose one of these modes, the Quartus II software reports an error message when the .pof is generated.

In the **Convert Programming Files** window, there are **SOF Data** entries (.sof), located in the **Input files to convert** dialog box. Each **SOF Data** entry refers to a unique system configuration. **Figure 2-6** shows the setup for a system that has one APEX device and uses two pages, 0 and 1. Each page has a different version of the configuration file for the same APEX device.

**Figure 2-6.** Using Page Mode Example



To set the page pointer to point to a particular page or **SOF Data** entry, select **SOF Data** and click **Properties**. Clicking **Properties** launches the **SOF Data Properties** window where you can select page pointers to point to the chosen **SOF Data**. If you do not use the **SOF Data Properties** window to make changes, the default page is 0. Each **SOF Data** entry for your configuration device must have a unique page number. **Figure 2-7** shows page pointer 1 assigned to the **SOF Data** section containing **Device1\_Rev2.sof** (**Figure 2-6**).

**Figure 2-7.** Software Setting for Selecting Pages

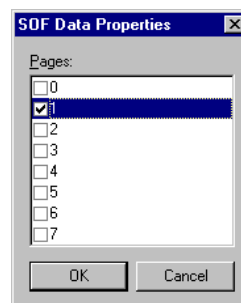
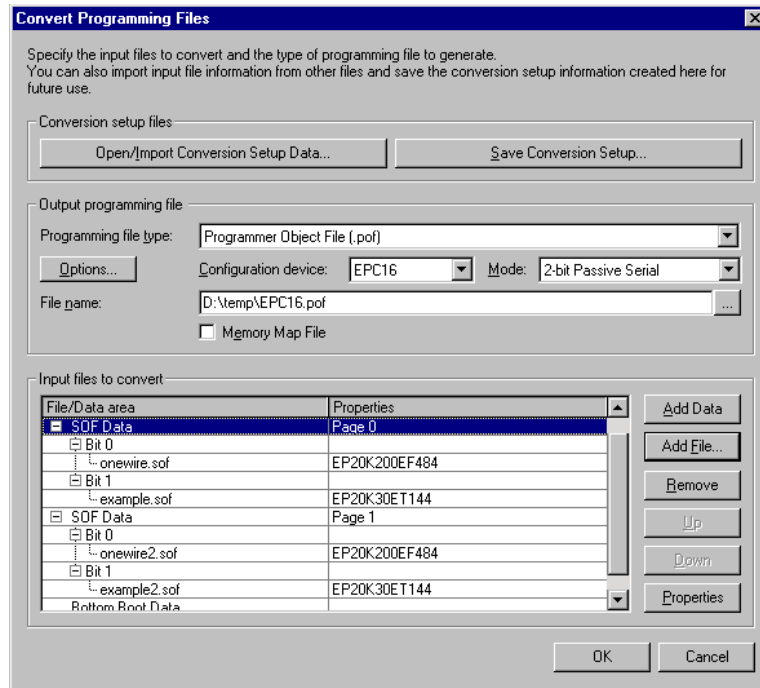


Figure 2–8 shows a more complex setup that uses the 2-bit PS configuration mode to concurrently configure two different APEX devices with multiple pages storing two revisions of each design. Two configurations for the entire system requires four configuration files (the number of devices multiplied by the number of unique system configurations).

**Figure 2–8.** Concurrent Configuration of Two Devices with Two System Configurations



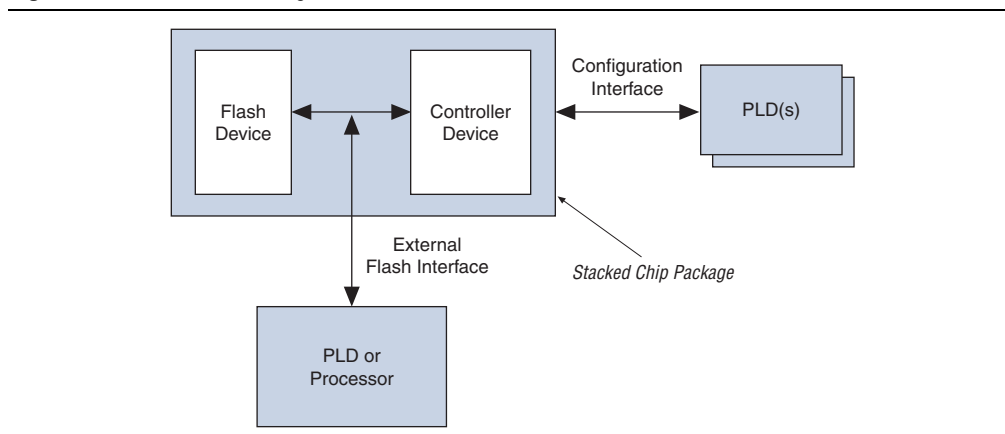
By selecting the **Memory Map File** option, the Quartus II Memory Map output file (**.map**) is generated, describing the flash memory address locations. This information is typically useful when using the external flash interface feature.

## External Flash Memory Interface

Enhanced configuration devices support an external flash interface that allows devices external to the controller access to the enhanced configuration device's flash memory. You can use the flash memory to store boot or application code for processors, or as general-purpose memory for processors and PLDs.

Figure 2-9 shows the interfaces available on the enhanced configuration device.


**Figure 2-9.** Enhanced Configuration Device Interfaces



Applications that require remote update capabilities for on-board programmable logic (Stratix series devices), and applications that use soft-embedded processor cores (Nios<sup>®</sup> embedded processor) typically use the external flash memory interface feature.

For soft-embedded processor core applications, the controller configures the programmable logic with the configuration data stored in the flash memory. On successful configuration, the embedded processor uses the external flash interface to boot up and run code from the same flash memory, eliminating the need for a stand-alone flash memory device.

For applications requiring remote system configuration capabilities, a processor or PLD can use the external flash interface to store an updated configuration image into a new page in flash memory (the external flash interface coupled with dynamic configuration). You can obtain new configuration data from a local intelligent host or through the Internet. Reconfiguring the system with the new page updates the system configuration.

 For more information about implementing remote and local system updates with enhanced configuration devices, refer to the *Using Remote System Configuration with Stratix & Stratix GX Devices* chapter in the *Stratix Handbook*, or the *Remote System Upgrades with Stratix II & Stratix II GX Devices* chapter in the *Stratix II Handbook*.

Currently, EPC4 and EPC16 configuration devices support the external flash interface. For support of this feature in other enhanced configuration devices, contact Altera Applications.

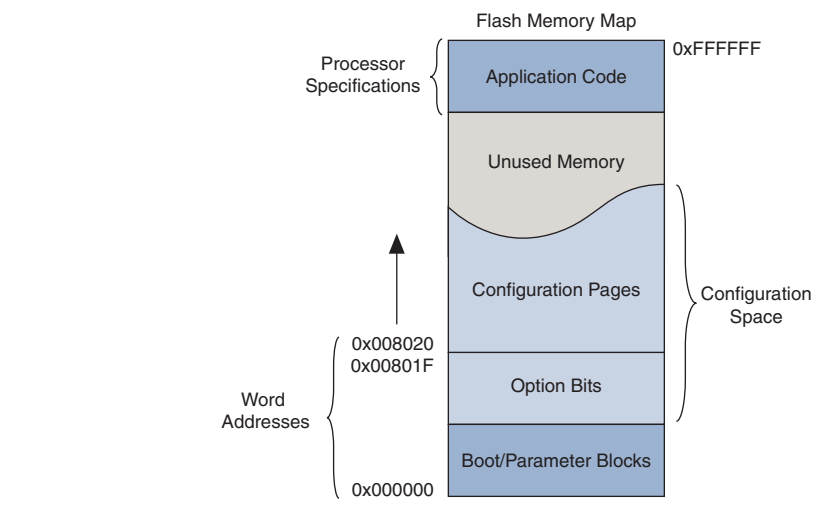
## Flash Memory Map

You can divide an enhanced configuration device's flash memory into two categories: logical (configuration and processor space) and physical (flash data block boundaries). Configuration space consists of portions of memory used to store configuration option bits and configuration data. Processor space consists of portions of memory used to store boot and application code.

## Logical Divisions

In all enhanced configuration devices, configuration option bits are stored ranging from word address 0x008000 to 0x00801F (byte address 0x010000 to 0x01003F). These bits are used to enable various controller features such as configuration mode selection, compression mode selection, and clock divider selection. In all enhanced configuration devices, configuration data is stored starting from word address location 0x008020 or byte address 0x010040. The ending address of configuration space is not fixed and depends on the number and density of PLDs configured using the enhanced configuration device as well as the number of pages. All remaining address locations above the configuration space are available for processor application code. The boot space spans addresses 0x000000 to 0x007FFF. Both boot and application code spaces are intended for use by an external processor or PLD. [Figure 2-10](#) shows the flash memory map inside an EPC16 device.

**Figure 2-10.** EPC16 Flash Memory Map



## Physical Divisions

Conversely, physical divisions are flash data blocks that can be individually written to and erased. For instance, the Sharp flash-based EPC16 device contains 16-Mbit Sharp flash memory that is divided into 2 boot blocks, 6 parameter blocks, and 31 main data blocks. These physical divisions vary from one flash memory or vendor to another and must be considered if the external flash interface is used to erase or write flash memory. These divisions are not significant if the interface is used as a read-only interface after initial programming.



For detailed information about enhanced configuration device flash memories, refer to the following documents:


- For Micron flash-based EPC4, refer to the *Micron Flash Memory MT28F400B3 Data Sheet* at [www.micron.com](http://www.micron.com).
- For Sharp flash-based EPC16, refer to the *Sharp LHF16J06 Data Sheet Flash Memory Used in EPC16 Devices* at [www.sharpsma.com](http://www.sharpsma.com).
- For the *Intel Advanced Boot Block Flash Memory (B3) 28F008/800B3, 28F016/160B3, 28F320B3, 28F640B3 Datasheet*, visit [www.intel.com](http://www.intel.com).

## Interface Availability and Connections

Flash memory ports are shared between the internal controller and the external device. A processor or PLD can use the external flash interface to access flash memory only when the controller is not using the interface. Therefore, the internal controller is the primary master of the bus, while the external device is the secondary master.

Flash memory ports (address, data, and control) are internally connected to the controller device. Additionally, these ports are connected to pins on the package providing the external interface. During in-system programming of the enhanced configuration device as well as configuration of the PLDs, the controller uses the internal interface to flash memory, rendering the external interface unavailable. External devices should tri-state all connections (address, data, and control) for the entire duration of in-system programming and configuration to prevent contention.


On completion of in-system programming and configuration, the internal controller tri-states its interface to the flash memory and enables weak internal pull-up resistors on address and control lines as well as bus-hold circuits on the data lines. The internal flash interface is now disabled and the external flash interface is available.

 If you do not use the external flash interface feature, most flash-related pins must be left unconnected on the board to avoid contention. There are a few exceptions to this guideline outlined in the data sheet and pin-out tables.

 For detailed schematics, refer to the *Enhanced Configuration Devices (EPC4, EPC8, and EPC16) Data Sheet*.


## Quartus II Software Support

You can use the **Convert Programming Files** window to generate flash memory programming files. You can program flash memory in-system using JTAG or through the external flash interface. Select the **.pof** when programming the flash memory in-system. You can also convert this **.pof** to a Jam™ standard test and programming language (STAPL) file (**.jam**) or Jam Byte-Code file (**.jbc**) for in-system programming. When programming the flash memory through the external flash interface, you can create a **.hexout** from this window.

 The **.hexout** used for programming enhanced configuration devices is different from the **.hexout** configuration file generated for SRAM PLDs.

Along with PLD configuration files, you can program processor boot and application code into flash memory through the **Convert Programming Files** window. You can add a **.hex** file containing boot code to the **Bottom Boot Data** section of the window. Similarly, you can add a **.hex** file containing application code to the **Main Block Data** section. You can store these files in the flash memory using relative or absolute addressing. For selecting the type of addressing, highlight the **Bottom Boot Data** or **Main Block Data** section and click **Properties (Convert Programming Files window)**.


Relative addressing mode allows the Quartus II software to pick the location of the file in memory. For example, the Quartus II software always stores boot code starting at address location 0x000000. This data increases to higher addresses.

 The maximum boot file size for the EPC16 configuration device is 32K words or 64 Kbytes. The boot code is limited to the boot and flash memory parameter blocks.

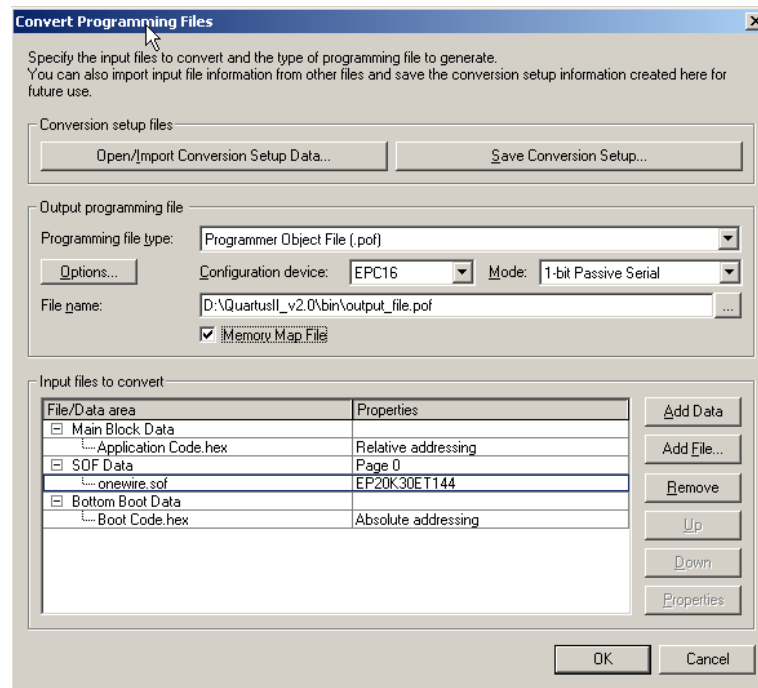
When you select relative addressing mode for **Main Block Data**, the Quartus II software aligns the last byte of information with the highest address (for example, 0x1FFFFFF). Therefore, the starting address is dependent on the size of the **.hex** file. You can easily obtain the starting address of the application code with the **.map** file discussed in this section.

Conversely, the absolute addressing mode forces the Quartus II software to store the boot or application **.hex** file data in address locations specified inside the **.hex** file itself. When this mode is selected, create **.hex** files with the correct offsets and ensure there is no overlap with addresses used for storing configuration data.

**Figure 2-11** shows a screen shot of the **Convert Programming Files** window setup to create a **.pof** and **.map** file for an enhanced configuration device.

 Only one **.hex** file can be added to the **Bottom Boot Data** and **Main Block Data** sections of this window.

**Figure 2-11.** Storing Boot and Application Code in Flash Memory



You can use the **Quartus II Convert Programming Files** window to create two files specific to the external flash interface feature—the **.hexout** and the **.map** files. The **.hexout** contains an image of the flash memory and the **.map** file contains memory map information. The **.hexout** can be used by an external processor or PLD to program the flash memory via the external flash interface. The **.map** file contains starting and ending addresses for boot code, configuration page data, and application code.



You can use the **.hexout** to program blank enhanced configuration devices, update portions of the flash memory (for example, a new configuration page), or both. This file uses the Intel hexadecimal file format and contains 16 Mbits or 2 Mbytes of data. [Table 2-2](#) lists the format of the **.map** file.


**Table 2-2.** File Format (.map) *(Note 1)*

Block	Start Address	End Address
BOTTOM BOOT	0x00000000	0x0000001F
OPTION BITS	0x00010000	0x0001003F
PAGE 0	0x00010040	0x0001AD7F
MAIN	0x001FFFE0	0x001FFFFF

**Note to Table 2-2:**

(1) All the addresses in this file are byte addresses.

To perform partial flash memory updates, select the relevant portions of the **.hexout** using memory map information provided in the **.map** file.

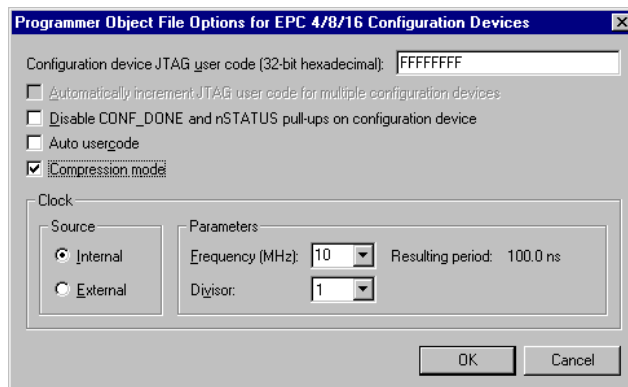
 Configuration data and processor space data could exist within the same physical data block. In such cases, erasing the physical data block would affect both configuration and processor data, requiring you to update both. You can avoid this situation by storing application data starting from the next available whole data block.

## Data Compression

Enhanced configuration devices support an efficient compression algorithm that compresses configuration data by 1.9× for typical designs, effectively doubling the size of the device. To select the right density for enhanced configuration devices, you should pre-calculate the total size of uncompressed configuration space.

By clicking **Options** (Convert Programming Files window), you can turn on the **Compression mode** option in the Programming Object File Options window with **pdf** selected as the programming file type, as shown in [Figure 2-12](#).

**Figure 2-12.** Selecting Compression Mode



## Calculating the Size of Configuration Space

When using 1-bit PS configuration mode to serially configure multiple devices, all configuration data is transmitted through the same DATA line and the devices are daisy-chained together. Therefore, the total size of the uncompressed configuration data is equal to the sum of the SRAM-based device's configuration file size multiplied by the number of pages used.

When using  $n$ -bit PS configuration mode to concurrently configure multiple devices, each SRAM-based device has its own DATA line from the enhanced configuration devices. The total size of the uncompressed configuration space is equal to the size of the largest device's configuration file size multiplied by  $n$  (where  $n = 1, 2, 4, \text{ or } 8$ ), which is then multiplied by the number of pages used. For example, if three devices are concurrently configured using 4-bit PS configuration mode, the total size of the uncompressed configuration space is equal to the size of the largest device's configuration file multiplied by four.

When using FPP configuration mode, the total size of the uncompressed configuration space is equal to the sum of the SRAM-based device's configuration file size multiplied by the number of pages used.

For configuration file sizes of SRAM-based devices, refer to the *Configuration Handbook*.

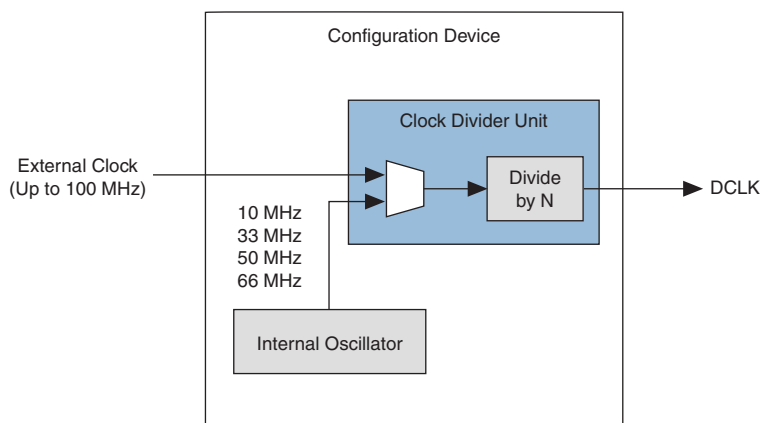
## Clock Divider

The clock divider value specifies the clock frequency divisor, which is used to determine the DCLK frequency, or how fast the data is clocked into the SRAM-based device. You must consider the maximum DCLK input frequency of the targeted SRAM device family while selecting the clock input and divider settings. For DCLK timing specifications of SRAM-based devices, refer to the *Configuration Handbook*.

## Settings and Guidelines

Enhanced configuration devices can use either the internal oscillator or an external clock source to clock data into SRAM-based devices, as shown in [Figure 2-13](#). The enhanced configuration device's internal oscillator runs at nominal speeds of 10, 33, 50, or 66 MHz. The minimum and maximum speeds are shown in the *Enhanced Configuration Device Data Sheet*. Additionally, the enhanced configuration device can accept an external clock source running at speeds of up to 100 MHz.

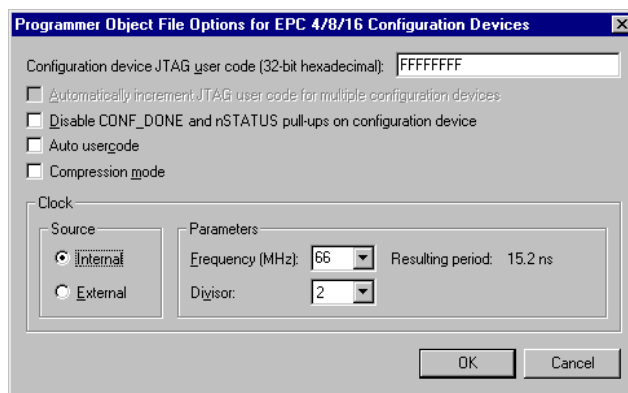
Figure 2-13. Clock Divider Unit in Enhanced Configuration Devices



## Software Implementations

You can select the clock source and the clock speed in the **Programming Object File Options** window with **pof** selected as the programming file type (Convert Programming Files window), as shown in Figure 2-14. You can type the appropriate external clock frequency in the **Frequency (MHz)** drop-down menu, and select any value from the divisor list regardless of the clock source setting.

Figure 2-14. Software for Setting Clock Source and Clock Divisor



## Conclusion

The enhanced configuration device is a single-chip configuration solution that provides increased configuration flexibility and faster time-to-market. Features such as data compression, multiple clock sources, clock division, and parallel or concurrent programming significantly reduce configuration times, while the dynamic configuration mode and the external flash interface take intelligent system configuration to a higher level.

## Chapter Revision History

Table 2-3 lists the revision history for this chapter.

**Table 2-3.** Chapter Revision History

Date	Version	Changes Made
December 2009	2.5	<ul style="list-style-type: none"> <li>■ Removed “Referenced Documents” section.</li> </ul>
October 2008	2.4	<ul style="list-style-type: none"> <li>■ Added “Referenced Documents” section.</li> <li>■ Updated new document format.</li> </ul>
April 2007	2.3	<ul style="list-style-type: none"> <li>■ Added document revision history.</li> </ul>
October 2005	2.2	<ul style="list-style-type: none"> <li>■ Technical content added.</li> </ul>
July 2004	2.0	<ul style="list-style-type: none"> <li>■ Added text regarding pointing to an incorrect page after <a href="#">Figure 2-8</a>.</li> <li>■ Renamed <b>.hexpof</b> to <b>.hexout</b> throughout chapter.</li> </ul>
September 2003	1.0	<ul style="list-style-type: none"> <li>■ Initial Release.</li> </ul>

## Features

The serial configuration devices provide the following features:

- 1-, 4-, 16-, 64-, and 128-Mbit flash memory devices that serially configure Arria® series, Cyclone® series, all device families in the Stratix® series except the Stratix device family, and FPGAs using the active serial (AS) configuration scheme
- Easy-to-use four-pin interface
- Low cost, low-pin count, and non-volatile memory
- Low current during configuration and near-zero standby mode current
- 2.7-V to 3.6-V operation
- EPCS1 and EPCS4 available in 8-pin small outline integrated circuit (SOIC) package. EPCS16 available in 8-pin or 16-pin SOIC packages. EPCS64 and EPCS128 available in 16-pin SOIC package
- Enables the Nios® processor to access unused flash memory through AS memory interface
- Re-programmable memory with more than 100,000 erase/program cycles
- Write protection support for memory sectors using status register bits
- In-system programming support with SRunner software driver
- In-system programming support with USB Blaster™, EthernetBlaster, or ByteBlaster™ II download cables
- Additional programming support with the Altera® Programming Unit (APU) and programming hardware from BP Microsystems, System General, and other vendors
- Delivered with the memory array erased (all the bits set to 1)



The term “serial configuration devices” used in this document refers to Altera EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128.

## Functional Description

With SRAM-based devices that support active serial configuration, configuration data must be reloaded each time the device powers up, the system reconfigures, or when new configuration data is required. Serial configuration devices are flash memory devices with a serial interface that can store configuration data for FPGA devices that support active serial configuration and reload the data to the device upon power-up or reconfiguration. [Table 3-1](#) summarizes the features of the Altera configuration devices and the amount of configuration space they hold.

**Table 3-1.** Altera Configuration Devices *(Note 1), (2)*

Device	Memory Size (bits)	On-Chip Decompression Support	ISP Support	Cascading Support	Reprogrammable	Operating Voltage (V)
EPCS1	1,048,576	No	Yes	No	Yes	3.3
EPCS4	4,194,304	No	Yes	No	Yes	3.3
EPCS16	16,777,216	No	Yes	No	Yes	3.3
EPCS64	67,108,864	No	Yes	No	Yes	3.3
EPCS128	134,217,728	No	Yes	No	Yes	3.3

**Notes to Table 3-1:**

- (1) To program these devices using Altera Programming Unit or Master Programming Unit, refer to [Altera Programming Hardware Data Sheet](#).
- (2) The EPCS device can be re-programmed in system with Byte Blaster II download cable or an external microprocessor using SRunner. For more information about SRunner, refer to the [AN418, SRunner: An Embedded Solution for EPCS Programming](#).

For an 8-pin SOIC package, you can migrate vertically from the EPCS1 to the EPCS4 or EPCS16 because the EPCS devices are offered in the same device package. Similarly, for a 16-pin SOIC package, you can migrate vertically from the EPCS16 to the EPCS64 or EPCS128.

Use the compression ratio calculation to determine the FPGA device to fit the EPCS.

**Example 3-1.** Compression Ratio Calculation

EP4SGX530 = 189,000,000 bits

EPCS128 = 134,217,728 bits

Preliminary data indicates that compression typically reduces the configuration bitstream size by 35% to 55%. We take the worst case that is 35% compression.

$189,000,000 \text{ bits} \times 0.65 = 122,850,000 \text{ bits}$   
It fits EPCS128 device.

With the new data-decompression feature in Arria series, Cyclone series, and all device families in the Stratix series except the Stratix device family, you can use smaller serial configuration devices to configure larger FPGAs.



Serial configuration devices cannot be cascaded.

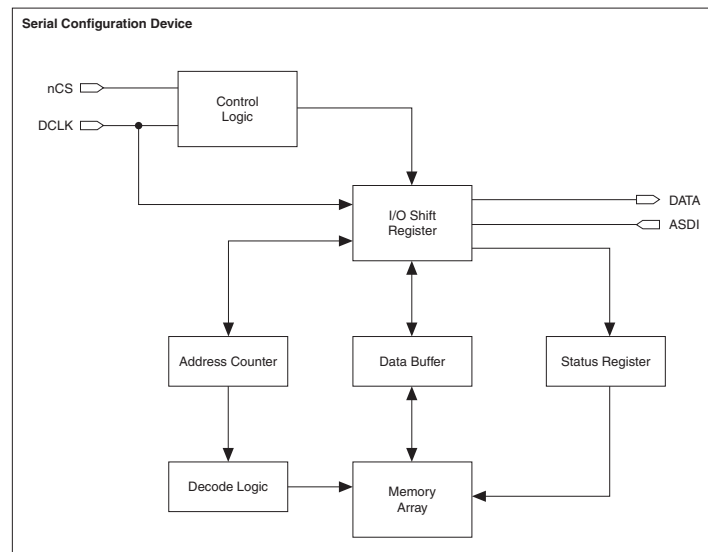


For more information about the FPGA decompression feature, refer to the configuration chapter in the appropriate device handbook.

The serial configuration devices are designed to configure the Cyclone series and all device families in the Stratix series except the Stratix device family. It cannot configure other existing Altera FPGA device families.

Figure 3-1 shows the serial configuration device block diagram.

Figure 3-1. Serial Configuration Device Block Diagram



## Accessing Memory in Serial Configuration Devices

You can access the unused memory locations of the serial configuration device to store or retrieve data through the Nios processor and SOPC Builder. SOPC Builder is an Altera tool for creating bus-based (especially microprocessor-based) systems in Altera devices. SOPC Builder assembles library components such as processors and memories into custom microprocessor systems.

SOPC Builder includes the EPCS device controller core, which is an interface core specifically designed to work with the serial configuration device. With this core, you can create a system with a Nios embedded processor that allows software access to any memory location within the serial configuration device.

 For more information about accessing memory within the serial configuration device, refer to the [Active Serial Memory Interface Data Sheet](#).


## Active Serial FPGA Configuration

The following Altera FPGAs support Active Serial (AS) configuration scheme with serial configuration devices:

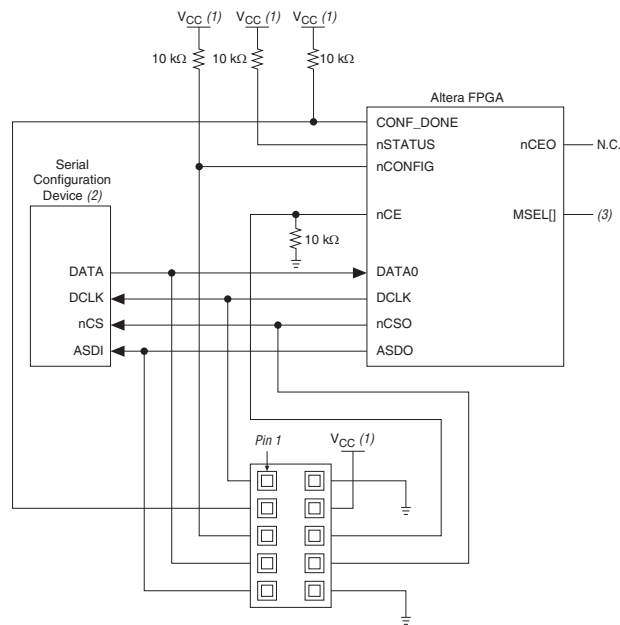
- Arria series
- Cyclone series
- all device families in the Stratix series except the Stratix device family

 This section is only relevant for FPGAs that support the AS configuration scheme.

There are four signals on the serial configuration device that interface directly with the FPGA's control signals. The serial configuration device signals DATA, DCLK, ASDI, and nCS interface with DATA0, DCLK, ASDO, and nCSO control signals on the FPGA, respectively. Figure 3-2 shows a serial configuration device programmed via a download cable, which configures an FPGA in AS mode. Figure 3-3 shows a serial configuration device programmed using the APU or a third-party programmer configuring an FPGA in AS configuration mode.

 For more information about the serial configuration device pin description, refer to Table 3-23.

**Figure 3-2.** Altera FPGA Configuration in AS Mode (Serial Configuration Device Programmed Using Download Cable)  
(Note 1), (4)

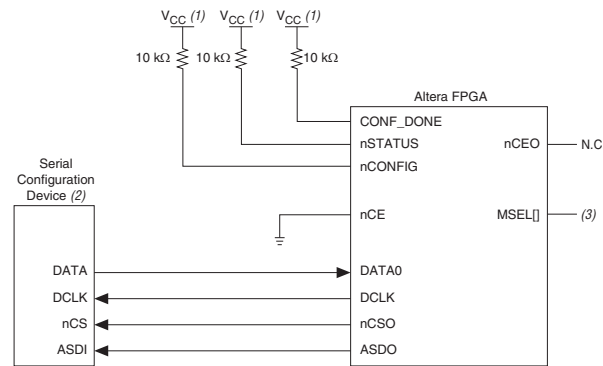


**Notes to Figure 3-2:**

- (1) For the  $V_{CC}$  value, refer to the respective FPGA family handbook Configuration chapter.
- (2) Serial configuration devices cannot be cascaded.
- (3) Connect the FPGA MSEL[] input pins to select the AS configuration mode. For details, refer to the respective FPGA family chapter in the *Configuration Handbook*.
- (4) For more information about configuration pin I/O requirements in an AS scheme for an Altera FPGA, refer to the respective FPGA family handbook Configuration chapter.




**Figure 3-3.** Altera FPGA Configuration in AS Mode (Serial Configuration Device Programmed by APU or Third-Party Programmer) (Note 1), (4)




**Notes to Figure 3-3:**

- (1) For the  $V_{CC}$  value, refer to the respective FPGA family handbook Configuration chapter.
- (2) Serial configuration devices cannot be cascaded.
- (3) Connect the FPGA  $MSEL[]$  input pins to select the AS configuration mode. For details, refer to the respective FPGA family chapter in the *Configuration Handbook*.
- (4) For more information about configuration pin I/O requirements in an AS scheme for an Altera FPGA, refer to the respective FPGA family handbook Configuration chapter..

The FPGA acts as the configuration master in the configuration flow and provides the clock to the serial configuration device. The FPGA enables the serial configuration device by pulling the  $nCS$  signal low via the  $nCSO$  signal (refer to Figure 3-2 and Figure 3-3). Subsequently, the FPGA sends the instructions and addresses to the serial configuration device via the  $ASDO$  signal. The serial configuration device responds to the instructions by sending the configuration data to the FPGA's  $DATA0$  pin on the falling edge of  $DCLK$ . The data is latched into the FPGA on the next  $DCLK$  signal's falling edge.

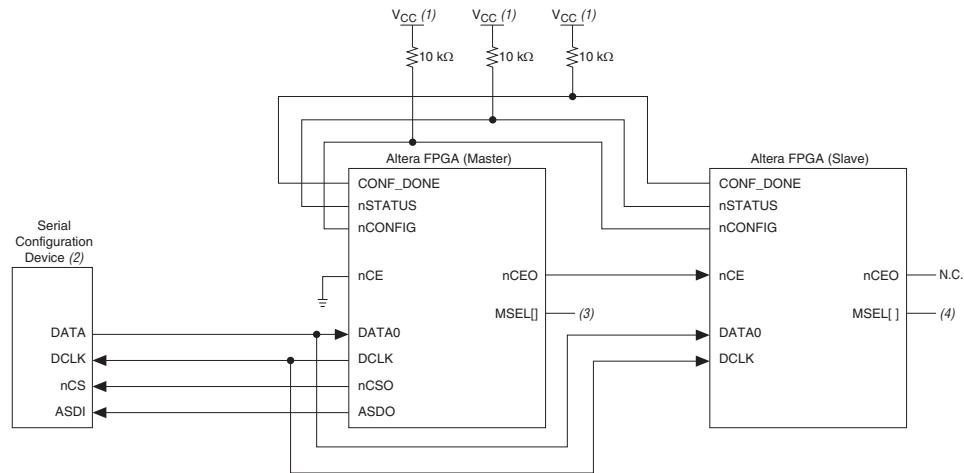
 Before the FPGA enters configuration mode, ensure that  $V_{CC}$  of the EPCS is ready. If it is not, you must hold  $nCONFIG$  low until all power rails of EPCS are ready.

The FPGA controls the  $nSTATUS$  and  $CONF\_DONE$  pins during configuration in AS mode. If the  $CONF\_DONE$  signal does not go high at the end of configuration or if the signal goes high too early, the FPGA will pulse its  $nSTATUS$  pin low to start reconfiguration. Upon successful configuration, the FPGA releases the  $CONF\_DONE$  pin, allowing the external 10-k $\Omega$  resistor to pull this signal high. Initialization begins after the  $CONF\_DONE$  goes high. After initialization, the FPGA enters user mode.

 For more information about configuring the FPGAs in AS mode or other configuration modes, refer to the Configuration chapter in the appropriate device handbook.

Multiple devices can be configured by a single EPCS device. However, serial configuration devices cannot be cascaded. Refer to [Table 3-1](#) to ensure the programming file size of the cascaded FPGAs does not exceed the capacity of a serial configuration device. [Figure 3-4](#) shows the AS configuration scheme with multiple FPGAs in the chain. The first FPGA is the configuration master and has its MSEL [ ] pins set to AS mode. The following FPGAs are configuration slave devices and have their MSEL [ ] pins set to PS mode.

**Figure 3-4.** Multiple Devices in AS Mode (*Note 1*), (*5*)



**Notes to Figure 3-4:**

- (1) For the  $V_{CC}$  value, refer to the respective FPGA family handbook Configuration chapter.
- (2) Serial configuration devices cannot be cascaded.
- (3) Connect the FPGA MSEL [ ] input pins to select the AS configuration mode. For details, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.
- (4) Connect the FPGA MSEL [ ] input pins to select the PS configuration mode. For details, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.
- (5) For more information about configuration pin I/O requirements in an AS scheme for an Altera FPGA, refer to the respective FPGA family handbook Configuration chapter.

## Serial Configuration Device Memory Access

This section describes the serial configuration device's memory array organization and operation codes. Timing specifications for the memory are provided in the "Timing Information" section.

### Memory Array Organization

Table 3-2 provides details about the memory array organization in EPCS128, EPCS64, EPCS16, EPCS4, and EPCS1.

**Table 3-2.** Memory Array Organization in Serial Configuration Devices

Details	EPCS128	EPCS64	EPCS16	EPCS4	EPCS1
Bytes (bits)	16,777,216 bytes (128 Mbits)	8,388,608 bytes (64 Mbits)	2,097,152 bytes (16 Mbits)	524,288 bytes (4 Mbits)	131,072 bytes (1 Mbit)
Number of sectors	64	128	32	8	4
Bytes (bits) per sector	262,144 bytes (2 Mbits)	65,536 bytes (512 Kbits)	65,536 bytes (512 Kbits)	65,536 bytes (512 Kbits)	32,768 bytes (256 Kbits)
Pages per sector	1,024	256	256	256	128
Total number of pages	65,536	32,768	8,192	2,048	512
Bytes per page	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes

Table 3-3 through Table 3-7 list the address range for each sector in EPCS128, EPCS64, EPCS16, EPCS4, and EPCS1.

**Table 3-3.** Address Range for Sectors in EPCS128 (Part 1 of 3)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
63	H'FC0000	H'FFFFFF
62	H'F80000	H'FBFFFF
61	H'F40000	H'F7FFFF
60	H'F00000	H'F3FFFF
59	H'EC0000	H'EFFFFF
58	H'E80000	H'EBFFFF
57	H'E40000	H'E7FFFF
56	H'E00000	H'E3FFFF
55	H'DC0000	H'DFFFFFF
54	H'D80000	H'DBFFFF
53	H'D40000	H'D7FFFF
52	H'D00000	H'D3FFFF
51	H'CC0000	H'FFFFFF
50	H'C80000	H'CBFFFF
49	H'C40000	H'C7FFFF
48	H'C00000	H'C3FFFF

**Table 3-3.** Address Range for Sectors in EPCS128 (Part 2 of 3)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
47	H'BC0000	H'BFFFFFF
46	H'B80000	H'BBFFFFFF
45	H'B40000	H'B7FFFFFF
44	H'B00000	H'B3FFFFFF
43	H'AC0000	H'AFFFFFF
42	H'A80000	H'ABFFFFFF
41	H'A40000	H'A7FFFFFF
40	H'A00000	H'A3FFFFFF
39	H'9C0000	H'9FFFFFF
38	H'980000	H'9BFFFFFF
37	H'940000	H'97FFFFFF
36	H'900000	H'93FFFFFF
35	H'8C0000	H'8FFFFFF
34	H'880000	H'8BFFFFFF
33	H'840000	H'87FFFFFF
32	H'800000	H'83FFFFFF
31	H'7C0000	H'7FFFFFF
30	H'780000	H'7BFFFFFF
29	H'740000	H'77FFFFFF
28	H'700000	H'73FFFFFF
27	H'6C0000	H'6FFFFFF
26	H'680000	H'6BFFFFFF
25	H'640000	H'67FFFFFF
24	H'600000	H'63FFFFFF
23	H'5C0000	H'5FFFFFF
22	H'580000	H'5BFFFFFF
21	H'540000	H'57FFFFFF
20	H'500000	H'53FFFFFF
19	H'4C0000	H'4FFFFFF
18	H'480000	H'4BFFFFFF
17	H'440000	H'47FFFFFF
16	H'400000	H'43FFFFFF
15	H'3C0000	H'3FFFFFF
14	H'380000	H'3BFFFFFF
13	H'340000	H'37FFFFFF
12	H'300000	H'33FFFFFF
11	H'2C0000	H'2FFFFFF
10	H'280000	H'2BFFFFFF

**Table 3-3.** Address Range for Sectors in EPCS128 (Part 3 of 3)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
9	H'240000	H'27FFFF
8	H'200000	H'23FFFF
7	H'1C0000	H'1FFFFF
6	H'180000	H'1BFFFF
5	H'140000	H'17FFFF
4	H'100000	H'13FFFF
3	H'0C0000	H'0FFFFF
2	H'080000	H'0BFFFF
1	H'040000	H'07FFFF
0	H'000000	H'03FFFF

**Table 3-4.** Address Range for Sectors in EPCS64 (Part 1 of 4)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
127	H'7F0000	H'7FFFFF
126	H'7E0000	H'7EFFFF
125	H'7D0000	H'7DFFFF
124	H'7C0000	H'7CFFFF
123	H'7B0000	H'7BFFFF
122	H'7A0000	H'7AFFFF
121	H'790000	H'79FFFF
120	H'780000	H'78FFFF
119	H'770000	H'77FFFF
118	H'760000	H'76FFFF
117	H'750000	H'75FFFF
116	H'740000	H'74FFFF
115	H'730000	H'73FFFF
114	H'720000	H'72FFFF
113	H'710000	H'71FFFF
112	H'700000	H'70FFFF
111	H'6F0000	H'6FFFFF
110	H'6E0000	H'6EFFFF
109	H'6D0000	H'6DFFFF
108	H'6C0000	H'6CFFFF
107	H'6B0000	H'6BFFFF
106	H'6A0000	H'6AFFFF
105	H'690000	H'69FFFF
104	H'680000	H'68FFFF

**Table 3-4.** Address Range for Sectors in EPCS64 (Part 2 of 4)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
103	H'670000	H'67FFFF
102	H'660000	H'66FFFF
101	H'650000	H'65FFFF
100	H'640000	H'64FFFF
99	H'630000	H'63FFFF
98	H'620000	H'62FFFF
97	H'610000	H'61FFFF
96	H'600000	H'60FFFF
95	H'5F0000	H'5FFFFF
94	H'5E0000	H'5EFFFF
93	H'5D0000	H'5DFFFF
92	H'5C0000	H'5CFFFF
91	H'5B0000	H'5BFFFF
90	H'5A0000	H'5AFFFF
89	H'590000	H'59FFFF
88	H'580000	H'58FFFF
87	H'570000	H'57FFFF
86	H'560000	H'56FFFF
85	H'550000	H'55FFFF
84	H'540000	H'54FFFF
83	H'530000	H'53FFFF
82	H'520000	H'52FFFF
81	H'510000	H'51FFFF
80	H'500000	H'50FFFF
79	H'4F0000	H'4FFFFF
78	H'4E0000	H'4EFFFF
77	H'4D0000	H'4DFFFF
76	H'4C0000	H'4CFFFF
75	H'4B0000	H'4BFFFF
74	H'4A0000	H'4AFFFF
73	H'490000	H'49FFFF
72	H'480000	H'48FFFF
71	H'470000	H'47FFFF
70	H'460000	H'46FFFF
69	H'450000	H'45FFFF
68	H'440000	H'44FFFF
67	H'430000	H'43FFFF
66	H'420000	H'42FFFF

**Table 3-4.** Address Range for Sectors in EPCS64 (Part 3 of 4)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
65	H'410000	H'41FFFF
64	H'400000	H'40FFFF
63	H'3F0000	H'3FFFFF
62	H'3E0000	H'3EFFFF
61	H'3D0000	H'3DFFFF
60	H'3C0000	H'3CFFFF
59	H'3B0000	H'3BFFFF
58	H'3A0000	H'3AFFFF
57	H'390000	H'39FFFF
56	H'380000	H'38FFFF
55	H'370000	H'37FFFF
54	H'360000	H'36FFFF
53	H'350000	H'35FFFF
52	H'340000	H'34FFFF
51	H'330000	H'33FFFF
50	H'320000	H'32FFFF
49	H'310000	H'31FFFF
48	H'300000	H'30FFFF
47	H'2F0000	H'2FFFFF
46	H'2E0000	H'2EFFFF
45	H'2D0000	H'2DFFFF
44	H'2C0000	H'2CFFFF
43	H'2B0000	H'2BFFFF
42	H'2A0000	H'2AFFFF
41	H'290000	H'29FFFF
40	H'280000	H'28FFFF
39	H'270000	H'27FFFF
38	H'260000	H'26FFFF
37	H'250000	H'25FFFF
36	H'240000	H'24FFFF
35	H'230000	H'23FFFF
34	H'220000	H'22FFFF
33	H'210000	H'21FFFF
32	H'200000	H'20FFFF
31	H'1F0000	H'1FFFFF
30	H'1E0000	H'1EFFFF
29	H'1D0000	H'1DFFFF
28	H'1C0000	H'1CFFFF

**Table 3-4.** Address Range for Sectors in EPCS64 (Part 4 of 4)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
27	H'1B0000	H'1BFFFF
26	H'1A0000	H'1AFFFF
25	H'190000	H'19FFFF
24	H'180000	H'18FFFF
23	H'170000	H'17FFFF
22	H'160000	H'16FFFF
21	H'150000	H'15FFFF
20	H'140000	H'14FFFF
19	H'130000	H'13FFFF
18	H'120000	H'12FFFF
17	H'110000	H'11FFFF
16	H'100000	H'10FFFF
15	H'0F0000	H'0FFFFF
14	H'0E0000	H'0EFFFF
13	H'0D0000	H'0DFFFF
12	H'0C0000	H'0CFFFF
11	H'0B0000	H'0BFFFF
10	H'0A0000	H'0AFFFF
9	H'090000	H'09FFFF
8	H'080000	H'08FFFF
7	H'070000	H'07FFFF
6	H'060000	H'06FFFF
5	H'050000	H'05FFFF
4	H'040000	H'04FFFF
3	H'030000	H'03FFFF
2	H'020000	H'02FFFF
1	H'010000	H'01FFFF
0	H'000000	H'00FFFF

**Table 3-5.** Address Range for Sectors in EPCS16 (Part 1 of 2)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
31	H'1F0000	H'1FFFFFF
30	H'1E0000	H'1EFFFF
29	H'1D0000	H'1DFFFF
28	H'1C0000	H'1CFFFF
27	H'1B0000	H'1BFFFF
26	H'1A0000	H'1AFFFF



**Table 3-5.** Address Range for Sectors in EPCS16 (Part 2 of 2)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
25	H'190000	H'19FFFF
24	H'180000	H'18FFFF
23	H'170000	H'17FFFF
22	H'160000	H'16FFFF
21	H'150000	H'15FFFF
20	H'140000	H'14FFFF
19	H'130000	H'13FFFF
18	H'120000	H'12FFFF
17	H'110000	H'11FFFF
16	H'100000	H'10FFFF
15	H'0F0000	H'0FFFFF
14	H'0E0000	H'0EFFFF
13	H'0D0000	H'0DFFFF
12	H'0C0000	H'0CFFFF
11	H'0B0000	H'0BFFFF
10	H'0A0000	H'0AFFFF
9	H'090000	H'09FFFF
8	H'080000	H'08FFFF
7	H'070000	H'07FFFF
6	H'060000	H'06FFFF
5	H'050000	H'05FFFF
4	H'040000	H'04FFFF
3	H'030000	H'03FFFF
2	H'020000	H'02FFFF
1	H'010000	H'01FFFF
0	H'000000	H'00FFFF

**Table 3-6.** Address Range for Sectors in EPCS4 (Part 1 of 2)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
7	H'70000	H'7FFFF
6	H'60000	H'6FFFF
5	H'50000	H'5FFFF
4	H'40000	H'4FFFF
3	H'30000	H'3FFFF
2	H'20000	H'2FFFF

**Table 3-6.** Address Range for Sectors in EPCS4 (Part 2 of 2)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
1	H'10000	H'1FFFF
0	H'00000	H'0FFFF

**Table 3-7.** Address Range for Sectors in EPCS1

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
3	H'18000	H'1FFFF
2	H'10000	H'17FFF
1	H'08000	H'0FFFF
0	H'00000	H'07FFF

## Operation Codes

This section describes the operations that can be used to access the memory in serial configuration devices. The DATA, DCLK, ASDI, and nCS signals access the memory in serial configuration devices. All serial configuration device operation codes, addresses and data are shifted in and out of the device serially, with the most significant bit (MSB) first.

The device samples the active serial data input on the first rising edge of the DCLK after the active low chip select (nCS) input signal is driven low. Shift the operation code (MSB first) serially into the serial configuration device through the active serial data input (ASDI) pin. Each operation code bit is latched into the serial configuration device on the rising edge of the DCLK.

Different operations require a different sequence of inputs. While executing an operation, you must shift in the desired operation code, followed by the address bytes, data bytes, both, or neither. The device must drive nCS high after the last bit of the operation sequence is shifted in. [Table 3-8](#) lists the operation sequence for every operation supported by the serial configuration devices.

For the read byte, read status, and read silicon ID operations, the shifted-in operation sequence is followed by data shifted out on the DATA pin. You can drive the nCS pin high after any bit of the data-out sequence is shifted out.

For the write byte, erase bulk, erase sector, write enable, write disable, and write status operations, drive the nCS pin high exactly at a byte boundary (drive the nCS pin high a multiple of eight clock pulses after the nCS pin is driven low); otherwise, the operation is rejected and is not executed.

All attempts to access the memory contents while a write or erase cycle is in progress will not be granted, and the write or erase cycle will continue unaffected.

**Table 3-8.** Operation Codes for Serial Configuration Devices

Operation	Operation Code (1)	Address Bytes	Dummy Bytes	Data Bytes	DCLK $f_{MAX}$ (MHz)
Write enable	0000 0110	0	0	0	25
Write disable	0000 0100	0	0	0	25
Read status	0000 0101	0	0	1 to infinite (2)	25
Read bytes	0000 0011	3	0	1 to infinite (2)	20
Read silicon ID (4)	1010 1011	0	3	1 to infinite (2)	25
Fast read	0000 1011	3	1	1 to infinite (2)	40
Write status	0000 0001	0	0	1	25
Write bytes	0000 0010	3	0	1 to 256 (3)	25
Erase bulk	1100 0111	0	0	0	25
Erase sector	1101 1000	3	0	0	25
Read Device Identification (5)	1001 1111	0	2	1 to infinite (2)	25

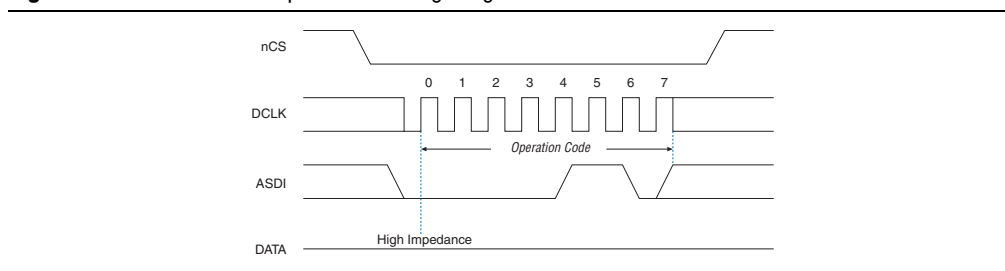
**Notes to Table 3-8:**

- (1) The MSB is listed first and the least significant bit (LSB) is listed last.
- (2) The status register, data or silicon ID are read out at least once on the DATA pin and will continuously be read out until  $nCS$  is driven high.
- (3) Write bytes operation requires at least one data byte on the DATA pin. If more than 256 bytes are sent to the device, only the last 256 bytes are written to the memory.
- (4) Read silicon ID operation is available only for EPCS1, EPCS4, EPCS16, and EPCS64.
- (5) Read Device Identification operation is available only for EPCS128.

**Write Enable Operation**

The write enable operation code is  $b'0000\ 0110$ , and the MSB is listed first. The write enable operation sets the write enable latch bit, which is bit 1 in the status register. Always set the write enable latch bit before write bytes, write status, erase bulk, and erase sector operations. Figure 3-5 shows the timing diagram for the write enable operation. Figure 3-7 and Figure 3-8 show the status register bit definitions.

**Figure 3-5.** Write Enable Operation Timing Diagram



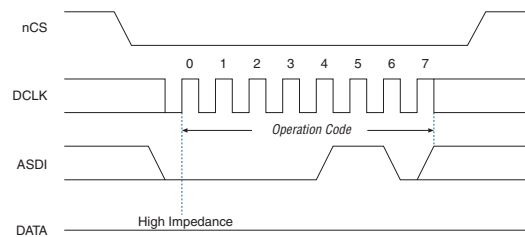
## Write Disable Operation

The write disable operation code is  $b'0000\ 0100$ , with the MSB listed first. The write disable operation resets the write enable latch bit, which is bit 1 in the status register. To prevent the memory from being written unintentionally, the write enable latch bit is automatically reset when implementing the write disable operation as well as under the following conditions:

- Power up
- Write bytes operation completion
- Write status operation completion
- Erase bulk operation completion
- Erase sector operation completion

Figure 3-6 shows the timing diagram for the write disable operation.

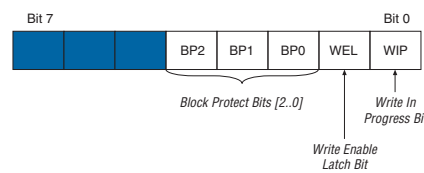
**Figure 3-6.** Write Disable Operation Timing Diagram



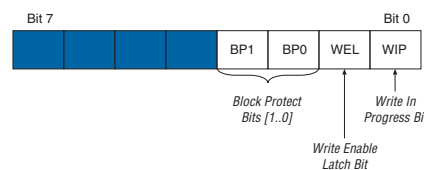
## Read Status Operation

The read status operation code is  $b'0000\ 0101$ , with the MSB listed first. You can use the read status operation to read the status register. Figure 3-7 and Figure 3-8 show the status bits in the status register of the serial configuration devices.

**Figure 3-7.** EPCS4, EPCS16, EPCS64, and EPCS128 Status Register Status Bits



**Figure 3-8.** EPCS1 Status Register Status Bits



Setting the write in progress bit to 1 indicates that the serial configuration device is busy with a write or erase cycle. Resetting the write in progress bit to 0 means no write or erase cycle is in progress.

Resetting the write enable latch bit to 0 indicates that no write or erase cycle will be accepted. Set the write enable latch bit to 1 before every write bytes, write status, erase bulk, and erase sector operation.

The non-volatile block protect bits determine the area of the memory protected from being written or erased unintentionally. Table 3-9 through Table 3-13 list the protected area in the serial configuration devices with reference to the block protect bits. The erase bulk operation is only available when all the block protect bits are 0. When any of the block protect bits are set to 1, the relevant area is protected from being written by write bytes operations or erased by erase sector operations.

**Table 3-9.** Block Protection Bits in EPCS1

Status Register Content		Memory Content	
BP1 Bit	BPO Bit	Protected Area	Unprotected Area
0	0	None	All four sectors: 0 to 3
0	1	Sector 3	Three sectors: 0 to 2
1	0	Two sectors: 2 and 3	Two sectors: 0 and 1
1	1	All sectors	None

**Table 3-10.** Block Protection Bits in EPCS4

Status Register Content			Memory Content	
BP2 Bit	BP1 Bit	BPO Bit	Protected Area	Unprotected Area
0	0	0	None	All eight sectors: 0 to 7
0	0	1	Sector 7	Seven sectors: 0 to 6
0	1	0	Sectors 6 and 7	Six sectors: 0 to 5
0	1	1	Four sectors: 4 to 7	Four sectors: 0 to 3
1	0	0	All sectors	None
1	0	1	All sectors	None
1	1	0	All sectors	None
1	1	1	All sectors	None

**Table 3-11.** Block Protection Bits in EPCS16 (Part 1 of 2)

Status Register Content			Memory Content	
BP2 Bit	BP1 Bit	BPO Bit	Protected Area	Unprotected Area
0	0	0	None	All sectors (32 sectors 0 to 31)
0	0	1	Upper 32nd (Sector 31)	Lower 31/32nds (31 sectors: 0 to 30)
0	1	0	Upper sixteenth (two sectors: 30 and 31)	Lower 15/16ths (30 sectors: 0 to 29)
0	1	1	Upper eighth (four sectors: 28 to 31)	Lower seven-eighths (28 sectors: 0 to 27)
1	0	0	Upper quarter (eight sectors: 24 to 31)	Lower three-quarters (24 sectors: 0 to 23)
1	0	1	Upper half (sixteen sectors: 16 to 31)	Lower half (16 sectors: 0 to 15)

**Table 3–11.** Block Protection Bits in EPCS16 (Part 2 of 2)

Status Register Content			Memory Content	
BP2 Bit	BP1 Bit	BPO Bit	Protected Area	Unprotected Area
1	1	0	All sectors (32 sectors: 0 to 31)	None
1	1	1	All sectors (32 sectors: 0 to 31)	None

**Table 3–12.** Block Protection Bits in EPCS64

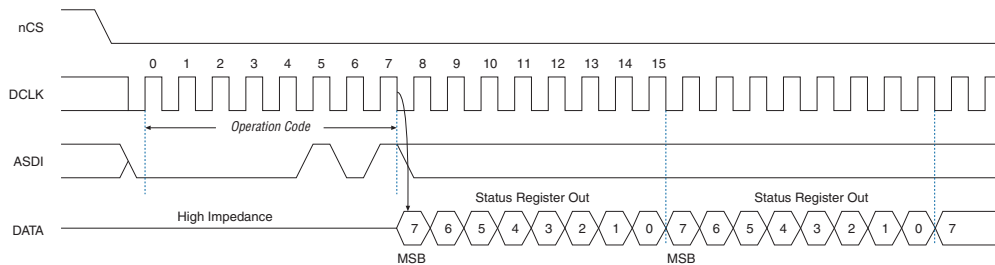
Status Register Content			Memory Content	
BP2 Bit	BP1 Bit	BPO Bit	Protected Area	Unprotected Area
0	0	0	None	All sectors (128 sectors: 0 to 127)
0	0	1	Upper 64th (2 sectors: 126 and 127)	Lower 63/64ths (126 sectors: 0 to 125)
0	1	0	Upper 32nd (4 sectors: 124 to 127)	Lower 31/32nds (124 sectors: 0 to 123)
0	1	1	Upper sixteenth (8 sectors: 120 to 127)	Lower 15/16ths (120 sectors: 0 to 119)
1	0	0	Upper eighth (16 sectors: 112 to 127)	Lower seven-eighths (112 sectors: 0 to 111)
1	0	1	Upper quarter (32 sectors: 96 to 127)	Lower three-quarters (96 sectors: 0 to 95)
1	1	0	Upper half (64 sectors: 64 to 127)	Lower half (64 sectors: 0 to 63)
1	1	1	All sectors (128 sectors: 0 to 127)	None

**Table 3–13.** Block Protection Bits in EPCS128

Status Register Content			Memory Content	
BP2 Bit	BP1 Bit	BPO Bit	Protected Area	Unprotected Area
0	0	0	None	All sectors (64 sectors: 0 to 63)
0	0	1	Upper 64th (1 sector: 63)	Lower 63/64ths (63 sectors: 0 to 62)
0	1	0	Upper 32nd (2 sectors: 62 to 63)	Lower 31/32nds (62 sectors: 0 to 61)
0	1	1	Upper 16th (4 sectors: 60 to 63)	Lower 15/16ths (60 sectors: 0 to 59)
1	0	0	Upper 8th (8 sectors: 56 to 63)	Lower seven-eighths (56 sectors: 0 to 55)
1	0	1	Upper quarter (16 sectors: 48 to 63)	Lower three-quarters (48 sectors: 0 to 47)
1	1	0	Upper half (32 sectors: 32 to 63)	Lower half (32 sectors: 0 to 31)
1	1	1	All sectors (64 sectors: 0 to 63)	None

You can read the status register at any time, even while a write or erase cycle is in progress. When one of these cycles is in progress, you can check the write in progress bit (bit 0 of the status register) before sending a new operation to the device. The device can also read the status register continuously, as shown in [Figure 3–9](#).

**Figure 3-9.** Read Status Operation Timing Diagram



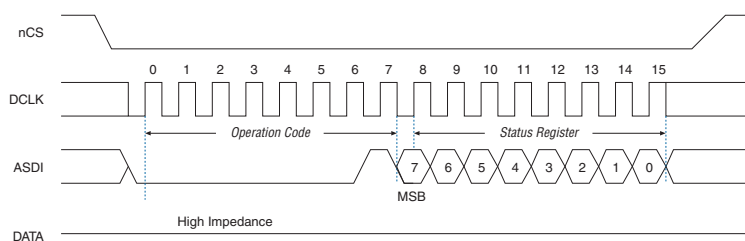
### Write Status Operation

The write status operation code is  $b'0000\ 0001$ , with the MSB listed first. Use the write status operation to set the status register block protection bits. The write status operation has no effect on the other bits. Therefore, you can implement this operation to protect certain memory sectors, as defined in [Table 3-9](#) through [Table 3-13](#). After setting the block protect bits, the protected memory sectors are treated as read-only memory. You must execute the write enable operation before the write status operation so the device sets the status register's write enable latch bit to 1.

The write status operation is implemented by driving  $nCS$  low, followed by shifting in the write status operation code and one data byte for the status register on the ASDI pin. [Figure 3-10](#) shows the timing diagram for the write status operation.  $nCS$  must be driven high after the eighth bit of the data byte has been latched in, otherwise, the write status operation is not executed.

Immediately after  $nCS$  drives high, the device initiates the self-timed write status cycle. The self-timed write status cycle usually takes 5 ms for all serial configuration devices and is guaranteed to be less than 15 ms (refer to  $t_{WS}$  in [Table 3-16](#)). You must account for this delay to ensure that the status register is written with desired block protect bits. Alternatively, you can check the write in progress bit in the status register by executing the read status operation while the self-timed write status cycle is in progress. The write in progress bit is 1 during the self-timed write status cycle, and 0 when it is complete.

**Figure 3-10.** Write Status Operation Timing Diagram

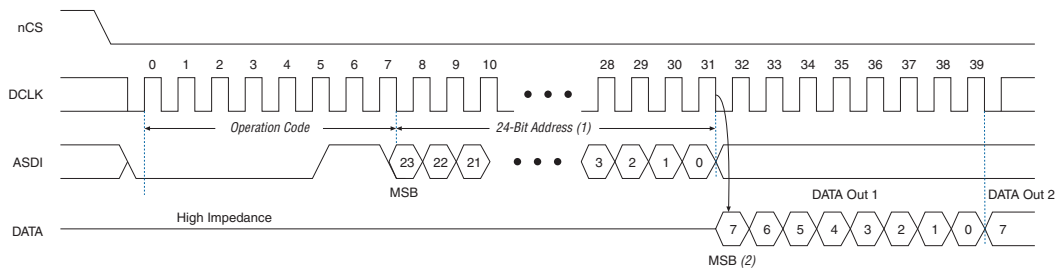


## Read Bytes Operation

The read bytes operation code is  $b'0000\ 0011$ , with the MSB listed first. To read the memory contents of the serial configuration device, the device is first selected by driving  $nCS$  low. Then, the read bytes operation code is shifted in followed by a 3-byte address ( $A[23..0]$ ). Each address bit must be latched in on the rising edge of the DCLK. After the address is latched in, the memory contents of the specified address are shifted out serially on the DATA pin, beginning with the MSB. For reading Raw Programming Data files (.rpd), the content is shifted out serially beginning with the LSB. Each data bit is shifted out on the falling edge of DCLK. The maximum DCLK frequency during the read bytes operation is 20 MHz. Figure 3–11 shows the timing diagram for the read bytes operation.

The first byte address can be at any location. The device automatically increments the address to the next higher address after shifting out each byte of data. Therefore, the device can read the whole memory with a single read bytes operation. When the device reaches the highest address, the address counter restarts at  $0x000000$ , allowing the memory contents to be read out indefinitely until the read bytes operation is terminated by driving  $nCS$  high. The device can drive  $nCS$  high any time after data is shifted out. If the read bytes operation is shifted in while a write or erase cycle is in progress, the operation is not executed and has no effect on the write or erase cycle in progress.

**Figure 3–11.** Read Bytes Operation Timing Diagram



### Notes to Figure 3–11:

- (1) Address bit  $A[23]$  is a don't-care bit in EPCS64. Address bits  $A[23..21]$  are don't-care bits in EPCS16. Address bits  $A[23..19]$  are don't-care bits in EPCS4. Address bits  $A[23..17]$  are don't-care bits in the EPCS1.
- (2) For .rpd files, the read sequence shifts out the LSB of the data byte first.



### Fast Read Operation

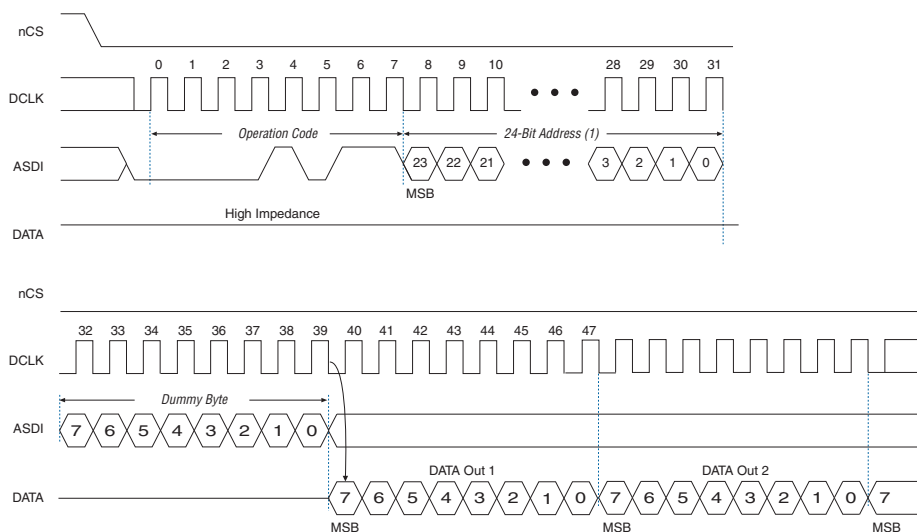
The device is first selected by driving  $nCS$  low. The fast read instruction code is followed by a 3-byte address ( $A_{23}-A_0$ ) and a dummy byte, each bit being latched-in during the rising edge of DCLK. Then the memory contents, at that address, is shifted out on DATA, each bit being shifted out, at a maximum frequency of 40 MHz, during the falling edge of DCLK.

The instruction sequence is shown in Figure 3-12.

The first byte addressed can be at any location. The address is automatically incremented to the next higher address after each byte of data is shifted out. The whole memory can, therefore, be read with a single fast read instruction. When the highest address is reached, the address counter rolls over to 000000h, allowing the read sequence to be continued indefinitely.

The fast read instruction is terminated by driving  $nCS$  high at any time during data output. Any fast read instruction is rejected during the Erase, Program, or Write operations without any effect on the operation that is in progress .

**Figure 3-12.** FAST\_READ Operation Timing Diagram



**Note to Figure 3-12:**

- (1) Address bit  $A[23]$  is a don't-care bit in EPCS64. Address bits  $A[23..21]$  are don't-care bits in EPCS16. Address bits  $A[23..19]$  are don't-care bits in EPCS4. Address bits  $A[23..17]$  are don't-care bits in the EPCS1.

### Read Silicon ID Operation

The read silicon ID operation code is  $b'1010\ 1011$ , with the MSB listed first. Only EPCS1, EPCS4, EPCS16, and EPCS64 support this operation. It reads the serial configuration device's 8-bit silicon ID from the DATA output pin. If this operation is shifted in during an erase or write cycle, it is ignored and has no effect on the cycle that is in progress.

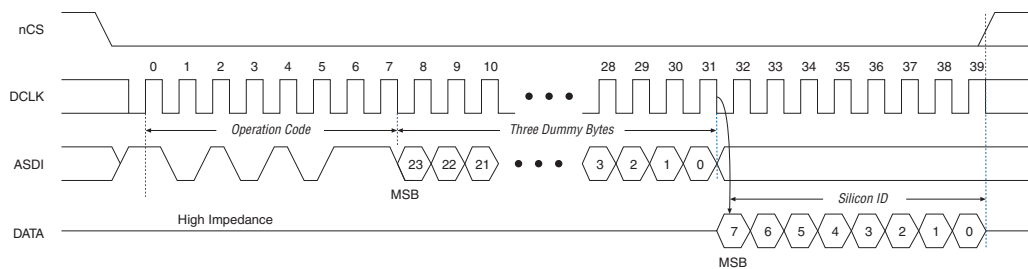
Table 3-14 lists the serial configuration device silicon IDs.

**Table 3-14.** Serial Configuration Device Silicon ID

Serial Configuration Device	Silicon ID (Binary Value)
EPCS1	$b'0001\ 0000$
EPCS4	$b'0001\ 0010$
EPCS16	$b'0001\ 0100$
EPCS64	$b'0001\ 0110$

The device implements the read silicon ID operation by driving  $nCS$  low then shifting in the read silicon ID operation code followed by three dummy bytes on ASDI. The serial configuration device's 8-bit silicon ID is then shifted out on the DATA pin on the falling edge of DCLK, as shown in Figure 3-13. The device can terminate the read silicon ID operation by driving  $nCS$  high after the silicon ID has been read at least once. Sending additional clock cycles on DCLK while  $nCS$  is driven low can cause the silicon ID to be shifted out repeatedly.

**Figure 3-13.** Read Silicon ID Operation Timing Diagram (Note 1)



**Note to Figure 3-13:**

- (1) Only EPCS1, EPCS4, EPCS16, and EPCS64 support Read Silicon ID operation.

### Read Device Identification Operation

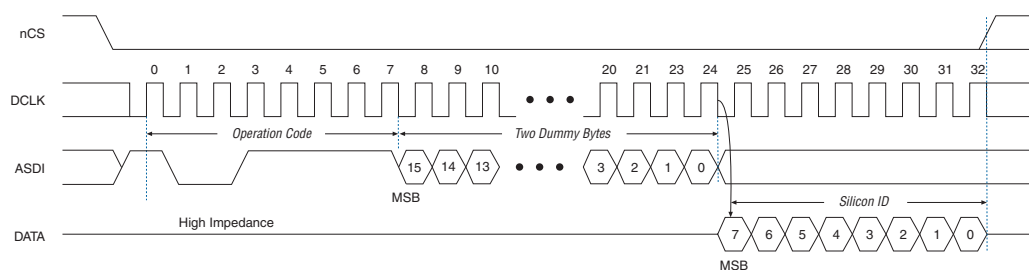
The read device identification operation code is  $b'1001\ 1111$ , with the MSB listed first. Only EPCS128 supports this operation. It reads the serial configuration device's 8-bit device identification from the DATA output pin. If this operation is shifted in during an erase or write cycle, it is ignored and has no effect on the cycle that is in progress. Table 3-15 shows the serial configuration device identification.

**Table 3-15.** Serial Configuration Device Identification

Serial Configuration Device	Silicon ID (Binary Value)
EPCS128	$b'0001\ 1000$

The device implements the read device identification operation by driving  $nCS$  low then shifting in the read device identification operation code followed by two dummy bytes on ASDI. The serial configuration device's 16-bit device identification is then shifted out on the DATA pin on the falling edge of DCLK, as shown in Figure 3-14. The device can terminate the read device identification operation by driving  $nCS$  high after reading the device identification at least once.

**Figure 3-14.** Read Device Identification Operation Timing Diagram (Note 1)



**Note to Figure 3-14:**

(1) Only EPCS128 supports read device identification operation.

### Write Bytes Operation

The write bytes operation code is  $b'0000\ 0010$ , with the MSB listed first. The write bytes operation allows bytes to be written to the memory. The write enable operation must be executed prior to the write bytes operation to set the write enable latch bit in the status register to 1.

The write bytes operation is implemented by driving  $nCS$  low, followed by the write bytes operation code, three address bytes and a minimum one data byte on ASDI. If the eight least significant address bits ( $A[7..0]$ ) are not all 0, all sent data that goes beyond the end of the current page is not written into the next page. Instead, this data is written at the start address of the same page (from the address whose eight LSBs are all 0). Drive  $nCS$  low during the entire write bytes operation sequence, as shown in Figure 3-15.

If more than 256 data bytes are shifted into the serial configuration device with a write bytes operation, the previously latched data is discarded and the last 256 bytes are written to the page. However, if less than 256 data bytes are shifted into the serial configuration device, they are guaranteed to be written at the specified addresses and the other bytes of the same page are unaffected.

If the design must write more than 256 data bytes to the memory, it needs more than one page of memory. Send the write enable and write bytes operation codes followed by three new targeted address bytes and 256 data bytes before a new page is written.

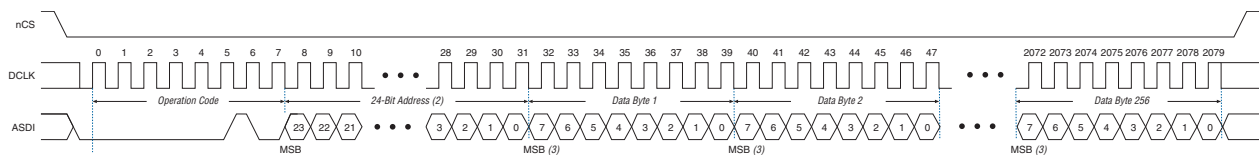
$nCS$  must be driven high after the eighth bit of the last data byte has been latched in. Otherwise, the device will not execute the write bytes operation. The write enable latch bit in the status register is reset to 0 before the completion of each write bytes operation. Therefore, the write enable operation must be carried out before the next write bytes operation.

The device initiates the self-timed write cycle immediately after  $nCS$  is driven high. Refer to  $t_{WB}$  in Table 3-16 on page 3-27 for the self-timed write cycle time for the respective EPCS devices. Therefore, you must account for this amount of delay before another page of memory is written. Alternatively, you can check the status register's write in progress bit by executing the read status operation while the self-timed write cycle is in progress. The write in progress bit is set to 1 during the self-timed write cycle, and 0 when it is complete.



The bytes of serial configuration devices memory must be erased to all 1 or 0xFF before write bytes operation is implemented. This can be achieved by either using the erase sector instruction in a sector, or the erase bulk instruction throughout the entire memory.

**Figure 3-15.** Write Bytes Operation Timing Diagram (Note 1)



**Notes to Figure 3-15:**

- (1) Use the erase sector or the erase bulk instruction to initialize the memory bytes of the serial configuration devices to all 1 or 0xFF before implementing the write bytes operation.
- (2) Address bit  $A[23]$  is a don't-care bit in EPCS64. Address bits  $A[23..21]$  are don't-care bits in EPCS16. Address bits  $A[23..19]$  are don't-care bits in EPCS4. Address bits  $A[23..17]$  are don't-care bits in EPCS1.
- (3) For .rpd files, write the LSB of the data byte first.

### Erase Bulk Operation

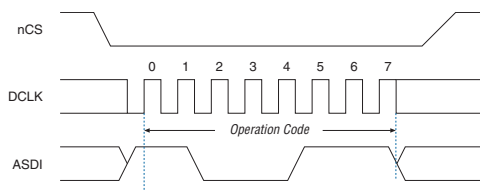
The erase bulk operation code is  $b'1100\ 0111$ , with the MSB listed first. The erase bulk operation sets all memory bits to 1 or 0xFF. Similar to the write bytes operation, the write enable operation must be executed prior to the erase bulk operation so that the write enable latch bit in the status register is set to 1.

You can implement the erase bulk operation by driving  $nCS$  low and then shifting in the erase bulk operation code on the ASDI pin.  $nCS$  must be driven high after the eighth bit of the erase bulk operation code has been latched in. Figure 3-16 shows the timing diagram.

The device initiates the self-timed erase bulk cycle immediately after  $nCS$  is driven high. Refer to  $t_{EB}$  in Table 3-16 for the self-timed erase bulk cycle time for the respective EPCS devices.

You must account for this delay before accessing the memory contents. Alternatively, you can check the write in progress bit in the status register by executing the read status operation while the self-timed erase cycle is in progress. The write in progress bit is 1 during the self-timed erase cycle and 0 when it is complete. The write enable latch bit in the status register is reset to 0 before the erase cycle is complete.

**Figure 3-16.** Erase Bulk Operation Timing Diagram



### Erase Sector Operation

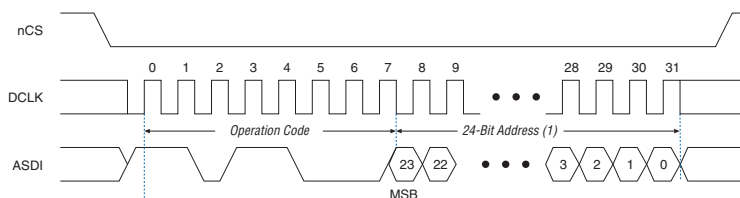
The erase sector operation code is  $b'1101\ 1000$ , with the MSB listed first. The erase sector operation allows the user to erase a certain sector in the serial configuration device by setting all bits inside the sector to 1 or  $0xFF$ . This operation is useful for users who access the unused sectors as general purpose memory in their applications.

The write enable operation must be executed prior to the erase sector operation so that the write enable latch bit in the status register is set to 1.

The erase sector operation is implemented by first driving  $nCS$  low, then shifting in the erase sector operation code and the three address bytes of the chosen sector on the ASDI pin. The three address bytes for the erase sector operation can be any address inside the specified sector. (Refer to Table 3-3 through Table 3-7 for sector address range information.) Drive  $nCS$  high after the eighth bit of the erase sector operation code has been latched in. Figure 3-17 shows the timing diagram.

Immediately after the device drives  $nCS$  high, the self-timed erase sector cycle is initiated. Refer to  $t_{ES}$  in Table 3-16 for the self-timed erase sector cycle time for the respective EPCS devices. You must account for this amount of delay before the memory contents can be accessed. Alternatively, you can check the write in progress bit in the status register by executing the read status operation while the erase cycle is in progress. The write in progress bit is 1 during the self-timed erase cycle and 0 when it is complete. The write enable latch bit in the status register resets to 0 before the erase cycle is complete.

**Figure 3-17.** Erase Sector Operation Timing Diagram



**Note to Figure 3-17:**

- (1) Address bit  $A[23]$  is a don't-care bit in EPCS64. Address bits  $A[23..21]$  are don't-care bits in EPCS16. Address bits  $A[23..19]$  are don't-care bits in EPCS4. Address bits  $A[23..17]$  are don't-care bits in EPCS1.

## Power and Operation

This section describes the power modes, power-on reset (POR) delay, error detection, and initial programming state of serial configuration devices.

### Power Mode

Serial configuration devices support active power and standby power modes. When  $nCS$  is low, the device is enabled and is in active power mode. The FPGA is configured while in active power mode. When  $nCS$  is high, the device is disabled but could remain in active power mode until all internal cycles have completed (such as write or erase operations). The serial configuration device then goes into stand-by power mode. The  $I_{CC1}$  parameter specifies the  $V_{CC}$  supply current when the device is in active power mode and the  $I_{CC0}$  parameter specifies the current when the device is in stand-by power mode (refer to [Table 3-21](#)).

### Power-On Reset

During initial power-up, a POR delay occurs to ensure the system voltage levels have stabilized. During AS configuration, the FPGA controls the configuration and has a longer POR delay than the serial configuration device.



For the POR delay time, refer to the configuration chapter in the appropriate device handbook.

### Error Detection

During AS configuration with the serial configuration device, the FPGA monitors the configuration status through the  $nSTATUS$  and  $CONF\_DONE$  pins. If an error condition occurs ( $nSTATUS$  drives low) or if the  $CONF\_DONE$  pin does not go high, the FPGA will begin reconfiguration by pulsing the  $nSTATUS$  and  $nCS0$  signals, which controls the chip select pin on the serial configuration device ( $nCS$ ).

After an error, configuration automatically restarts if the **Auto-Restart Upon Frame Error** option is turned on in the Quartus® II software. If the option is turned off, the system must monitor the  $nSTATUS$  signal for errors and then pulse the  $nCONFIG$  signal low to restart configuration.

## Timing Information

Figure 3-18 shows the timing waveform for write operation to the serial configuration device.

Figure 3-18. Write Operation Timing

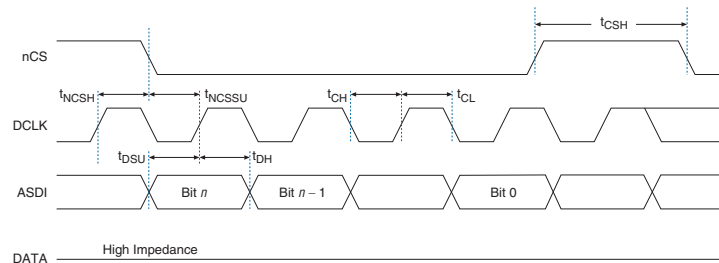


Table 3-16 defines the serial configuration device timing parameters for write operation.

Table 3-16. Write Operation Parameters

Symbol	Parameter	Min	Typ	Max	Unit
$f_{WCLK}$	Write clock frequency (from FPGA, download cable, or embedded processor) for write enable, write disable, read status, read silicon ID, write bytes, erase bulk, and erase sector operations	—	—	25	MHz
$t_{CH}$	DCLK high time	20	—	—	ns
$t_{CL}$	DCLK low time	20	—	—	ns
$t_{NCSU}$	Chip select ( $nCS$ ) setup time	10	—	—	ns
$t_{NCSH}$	Chip select ( $nCS$ ) hold time	10	—	—	ns
$t_{DSU}$	Data ( $ASDI$ ) in setup time before rising edge on DCLK	5	—	—	ns
$t_{DH}$	Data ( $ASDI$ ) hold time after rising edge on DCLK	5	—	—	ns
$t_{CSH}$	Chip select high time	100	—	—	ns
$t_{WB} (1)$	Write bytes cycle time for EPCS1, EPCS4, EPCS16, and EPCS64	—	1.5	5	ms
	Write bytes cycle time for EPCS128	—	2.5	7	ms
$t_{WS} (1)$	Write status cycle time	—	5	15	ms
$t_{EB} (1)$	Erase bulk cycle time for EPCS1	—	3	6	s
	Erase bulk cycle time for EPCS4	—	5	10	s
	Erase bulk cycle time for EPCS16	—	17	40	s
	Erase bulk cycle time for EPCS64	—	68	160	s
	Erase bulk cycle time for EPCS128	—	105	250	s
$t_{ES} (1)$	Erase sector cycle time for EPCS1, EPCS4, EPCS16, and EPCS64	—	2	3	s
	Erase sector cycle time for EPCS128	—	2	6	s

Note to Table 3-16:

(1) These parameters are not shown in Figure 3-18.

Figure 3-19 shows the timing waveform for the serial configuration device's read operation.

Figure 3-19. Read Operation Timing

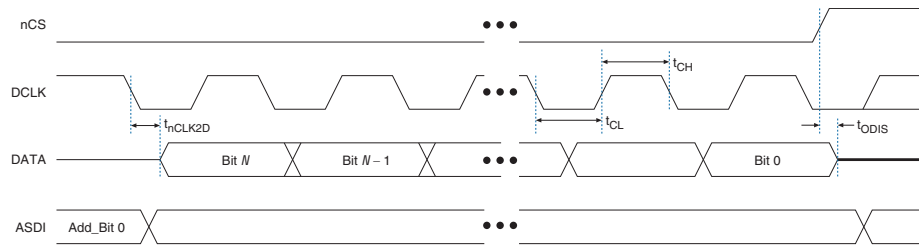



Table 3-17 defines the serial configuration device timing parameters for read operation.

Table 3-17. Read Operation Parameters

Symbol	Parameter	Min	Max	Unit
$f_{\text{RCLK}}$	Read clock frequency (from FPGA or embedded processor) for read bytes operation	—	20	MHz
$t_{\text{CH}}$	DCLK high time	25	—	ns
$t_{\text{CL}}$	DCLK low time	25	—	ns
$t_{\text{ODIS}}$	Output disable time after read	—	15	ns
$t_{\text{nCLK2D}}$	Clock falling edge to data	—	8	ns

 Existing batches of EPCS1 and EPCS4 manufactured on 0.15  $\mu\text{m}$  process geometry support AS configuration up to 40 MHz. However, batches of EPCS1 and EPCS4 manufactured on 0.18  $\mu\text{m}$  process geometry support only up to 20 MHz. EPCS16, EPCS64, and EPCS128 are not affected.


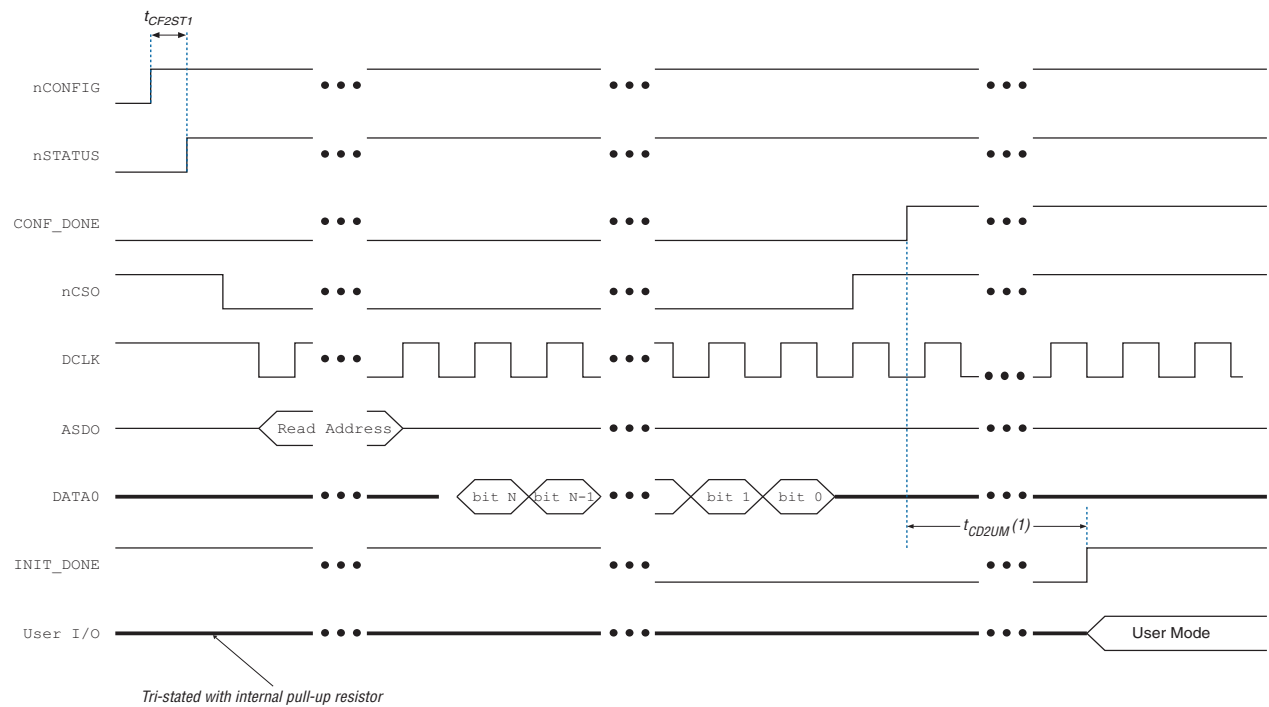
 For information about product traceability and transition date to differentiate between 0.15  $\mu\text{m}$  process geometry and 0.18  $\mu\text{m}$  process geometry EPCS1 and EPCS4, refer to the Process Change Notification [PCN 0514: Manufacturing Changes on EPCS Family](#).



Figure 3-20 shows the timing waveform for FPGA AS configuration scheme using a serial configuration device.

Figure 3-20. AS Configuration Timing



**Note to Figure 3-20:**


(1)  $t_{CD2UM}$  is a FPGA dependent parameter. For more information, refer to the respective device configuration chapters.

 For more information about the timing parameters in Figure 3-20, refer to the respective FPGA family handbook Configuration chapter.

## Programming and Configuration File Support

The Quartus II software provides programming support for serial configuration devices. After selecting the serial configuration device, the Quartus II software automatically generates the Programmer Object File (.pof) to program the device. The software allows users to select the appropriate serial configuration device density that most efficiently stores the configuration data for a selected FPGA.


The 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 that you can customize to fit in different embedded systems. The SRunner can read .rpd files and write to the serial configuration devices. The programming time is comparable to the Quartus II software programming time. Note that writing and reading the .rpd file to the EPCS is different from other data and address bytes. The LSB of .rpd bytes must be shifted out first during the read bytes instruction and the LSB of .rpd bytes must be shifted in first during the write bytes instruction. This is because the FPGA reads the LSB of the .rpd data first during the configuration process.


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

Serial configuration devices can be programmed using the APU with the appropriate programming adapter (PLMSEPC-8) via the Quartus II software, USB Blaster, EthernetBlaster, or the ByteBlaster II download cable via the Quartus II software. In addition, many third-party programmers, such as BP Microsystems and System General, offer programming hardware that supports serial configuration devices.

During in-system programming of a serial configuration device via the USB Blaster, EthernetBlaster, or ByteBlaster II download cable, the cable pulls *nCONFIG* low to reset the FPGA and overrides the 10-k $\Omega$  pull-down resistor on the FPGA's *nCE* pin (refer to [Figure 3-2](#)). The download cable then uses the four interface pins (*DATA*, *nCS*, *ASDI*, and *DCLK*) to program the serial configuration device. When the programming is complete, the download cable releases the serial configuration device's four interface pins and the FPGA's *nCE* pin, and pulses *nCONFIG* to start configuration.

The FPGA can program the serial configuration device in-system using the JTAG interface with the Serial FlashLoader. This solution allows you to indirectly program the serial configuration device using the same JTAG interface that is used to configure the FPGA.

 For more information about the Serial FlashLoader, refer to *AN 370: Using the Serial FlashLoader with the Quartus II Software*.

 For more information about programming and configuration support, refer to the following documents:

- [Altera Programming Hardware Data Sheet](#)
- [Programming Hardware Manufacturers](#)
- [USB-Blaster Download Cable User Guide](#)
- [ByteBlaster II Download Cable User Guide](#)
- [EthernetBlaster Communications Cable User Guide](#)

## Operating Conditions

Table 3–18 through Table 3–22 provide information about absolute maximum ratings, recommended operating conditions, DC operating conditions, and capacitance for serial configuration devices.

**Table 3–18.** Absolute Maximum Ratings (Note 1)

Symbol	Parameter	Condition	Min	Max	Unit
$V_{CC}$	Supply voltage for EPCS1, EPCS4, and EPCS16	With respect to ground	–0.6	4.0	V
	Supply voltage for EPCS64 and EPCS128	With respect to ground	–0.2	4.0	V
$V_I$	DC input voltage for EPCS1, EPCS4, and EPCS16	With respect to ground	–0.6	4.0	V
	DC input voltage for EPCS64 and EPCS128	With respect to ground	–0.5	4.0	V
$I_{MAX}$	DC $V_{CC}$ or GND current	—	—	15	mA
$I_{OUT}$	DC output current per pin	—	–25	25	mA
$P_D$	Power dissipation	—	—	54	mW
$T_{STG}$	Storage temperature	No bias	–65	150	°C
$T_{AMB}$	Ambient temperature	Under bias	–65	135	°C
$T_J$	Junction temperature	Under bias	—	135	°C

**Table 3–19.** Recommended Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
$V_{CC}$	Supply voltage	(2)	2.7	3.6	V
$V_I$	Input voltage	Respect to GND	–0.3	$0.3 + V_{CC}$	V
$V_O$	Output voltage	—	0	$V_{CC}$	V
$T_A$	Operating temperature	For commercial use	0	70	°C
		For industrial use	–40	85	°C
		For extended industrial temperature	–40	125	°C
$t_R$	Input rise time	—	—	5	ns
$t_F$	Input fall time	—	—	5	ns

**Table 3–20.** DC Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
$V_{IH}$	High-level input voltage for EPCS1, EPCS4, and EPCS16	—	$0.6 \times V_{CC}$	$V_{CC} + 0.4$	V
	High-level input voltage for EPCS64 and EPCS128	—	$0.6 \times V_{CC}$	$V_{CC} + 0.2$	V
$V_{IL}$	Low-level input voltage	—	-0.5	$0.3 \times V_{CC}$	V
$V_{OH}$	High-level output voltage	$I_{OH} = -100 \mu\text{A}$ (3)	$V_{CC} - 0.2$	—	V
$V_{OL}$	Low-level output voltage	$I_{OL} = 1.6 \text{ mA}$ (3)	—	0.4	V
$I_I$	Input leakage current	$V_I = V_{CC}$ or GND	-10	10	$\mu\text{A}$
$I_{OZ}$	Tri-state output off-state current	$V_O = V_{CC}$ or GND	-10	10	$\mu\text{A}$

**Table 3–21.**  $I_{CC}$  Supply Current

Symbol	Parameter	Conditions	Min	Max	Unit
$I_{CC0}$	$V_{CC}$ supply current (standby) for EPCS1, EPCS4, and EPCS16	—	—	50	$\mu\text{A}$
	$V_{CC}$ supply current (standby) for EPCS64 and EPCS128	—	—	100	$\mu\text{A}$
$I_{CC1}$	$V_{CC}$ supply current (during active power mode) for EPCS1, EPCS4, and EPCS16	—	5	15	mA
	$V_{CC}$ supply current (during active power mode) for EPCS64 and EPCS128	—	5	20	mA

**Table 3–22.** Capacitance (Note 4)

Symbol	Parameter	Conditions	Min	Max	Unit
$C_{IN}$	Input pin capacitance	$V_{IN} = 0 \text{ V}$	—	6	pF
$C_{OUT}$	Output pin capacitance	$V_{OUT} = 0 \text{ V}$	—	8	pF

**Notes to Table 3–18 through Table 3–22:**

- (1) For more information, refer to the *Operating Requirements for Altera Devices Data Sheet*.
- (2) Maximum  $V_{CC}$  rise time is 100 ms.
- (3) The  $I_{OH}$  parameter refers to high-level TTL or CMOS output current; the  $I_{OL}$  parameter refers to low-level TTL or CMOS output current.
- (4) Capacitance is sample-tested only at  $T_A = 25 \times \text{C}$  and at a 20-MHz frequency.

## Pin Information

As shown in Figure 3–21 and Figure 3–22, the serial configuration device is an 8-pin or 16-pin device. The control pins on the serial configuration device are: serial data output (DATA), active serial data input (ASDI), serial clock (DCLK), and chip select (*nCS*). Table 3–23 lists the serial configuration device's pin descriptions.

Figure 3–21 shows the Altera serial configuration device 8-pin SOIC package and its pin-out diagram.

**Figure 3–21.** Altera Serial Configuration Device 8-Pin SOIC Package Pin-Out Diagram

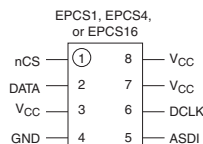
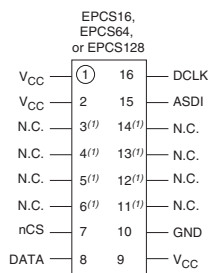


Figure 3–22 shows the Altera serial configuration device 16-pin SOIC package and its pin-out diagram.

**Figure 3–22.** Altera Serial Configuration Device 16-Pin SOIC Package Pin-Out Diagram



**Note to Figure 3–22:**

- (1) These pins can be left floating or connected to  $V_{CC}$  or GND, whichever is more convenient on the board.

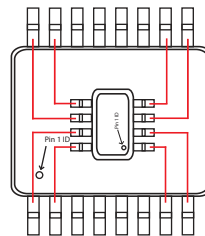
**Table 3–23.** Serial Configuration Device Pin Description (Part 1 of 2)

Pin Name	Pin Number in 8-Pin SOIC Package	Pin Number in 16-Pin SOIC Package	Pin Type	Description
DATA	2	8	Output	The DATA output signal transfers data serially out of the serial configuration device to the FPGA during read/configuration operation. During read/configuration operations, the serial configuration device is enabled by pulling <i>nCS</i> low. The DATA signal transitions on the falling edge of DCLK.
ASDI	5	15	Input	The AS data input signal is used to transfer data serially into the serial configuration device. It receives the data that should be programmed into the serial configuration device. Data is latched on the rising edge of DCLK.

**Table 3-23.** Serial Configuration Device Pin Description (Part 2 of 2)


Pin Name	Pin Number in 8-Pin SOIC Package	Pin Number in 16-Pin SOIC Package	Pin Type	Description
$\overline{nCS}$	1	7	Input	The active low chip select input signal toggles at the beginning and end of a valid instruction. When this signal is high, the device is deselected and the $DATA$ pin is tri-stated. When this signal is low, it enables the device and puts the device in an active mode. After power up, the serial configuration device requires a falling edge on the $\overline{nCS}$ signal before beginning any operation.
DCLK	6	16	Input	DCLK is provided by the FPGA. This signal provides the timing of the serial interface. The data presented on $ASDI$ is latched to the serial configuration device on the rising edge of DCLK. Data on the $DATA$ pin changes after the falling edge of DCLK and is latched into the FPGA on the next falling edge.
$V_{CC}$	3, 7, 8	1, 2, 9	Power	Power pins connect to 3.3 V.
GND	4	10	Ground	Ground pin.

As shown in [Figure 3-21](#) and [Figure 3-22](#), the serial configuration device is an 8-pin or 16-pin device. In order to take advantage of vertical migration from EPCS1 to EPCS128, Altera recommends a layout for serial configuration devices.

**Figure 3-23.** Layout Recommendation for Vertical Migration from EPCS1 to EPCS128

## Package

EPCS1 and EPCS4 available in 8-pin small outline integrated circuit (SOIC) package. EPCS16 available in 8-pin and 16-pin small outline integrated circuit (SOIC) packages. EPCS64 and EPCS128 available in 16-pin small outline integrated circuit (SOIC) package.

 For more information about Altera device packaging including mechanical drawing and specifications for this package, refer to the [Altera Device Package Information Data Sheet](#).

## Ordering Code

Table 3–24 lists the ordering codes for serial configuration devices.

**Table 3–24.** Serial Configuration Device Ordering Codes

Device	Ordering Code (1)
EPCS1	EPCS1SI8 EPCS1SI8N
EPCS4	EPCS4SI8 EPCS4SI8N
EPCS16	EPCS16SI16N EPCS16SI8N
EPCS64	EPCS64SI16N
EPCS128	EPCS128SI16N

**Note to Table 3–24:**

(1) N: Lead free.

## Chapter Revision History

Table 3–25 lists the revision history for this chapter.

**Table 3–25.** Chapter Revision History (Part 1 of 3)

Date	Version	Changes Made
December 2009	3.3	<ul style="list-style-type: none"> <li>■ Updated “Features” and “Functional Description” sections.</li> <li>■ Added “Fast Read Operation” section.</li> <li>■ Removed Table 4–2 to Table 4–9, Table 4-26, and Table 4–33.</li> <li>■ Updated Table 3–1.</li> <li>■ Updated Figure 3–2.</li> <li>■ Removed “Referenced Documents” section.</li> </ul>
October 2008	3.2	<ul style="list-style-type: none"> <li>■ Updated “Introduction”, “Active Serial FPGA Configuration”, “Operation Codes”, “Read Status Operation”, “Read Device Identification Operation”, and “Package” sections.</li> <li>■ Updated Table 4–10, Table 4–25, Table 4–26, and Table 4–32.</li> <li>■ Updated Figure 4–5, Figure 4–13, and Figure 4–19.</li> <li>■ Added Figure 4–22.</li> <li>■ Added Table 4–33.</li> <li>■ Updated new document format.</li> </ul>
May 2008	3.1	<ul style="list-style-type: none"> <li>■ Updated Table 4–3, Table 4–6, Table 4–7, Table 4–28, and Table 4–29.</li> <li>■ Deleted Note 5 to Table 4–31.</li> <li>■ Added “Referenced Documents” section.</li> </ul>

**Table 3–25.** Chapter Revision History (Part 2 of 3)

Date	Version	Changes Made
August 2007	3.0	<ul style="list-style-type: none"> <li>■ Updated “Introduction” section.</li> <li>■ Updated “Functional Description” section.</li> <li>■ Updated Table 4–1 through Table 4–4 and Table 4–7 through Table 4–9 to with EPCS128 information.</li> <li>■ Added Table 4–6 on Arria GX.</li> <li>■ Added notes to Figure 4–3.</li> <li>■ Added notes to Figure 4–4.</li> <li>■ Updated Table 4–10 with EPCS128 information.</li> <li>■ Added new Table 4–11 on address range for sectors in EPCS128 device.</li> <li>■ Updated Table 4–16 with information about “Read Device Identification” and added (Note 5).</li> <li>■ Added new Table 4–21 on block protection bits in EPCS128.</li> <li>■ Added notes to Figure 4–12.</li> <li>■ Added new section “Read Device Identification Operation” with Table 4–23 and Figure 4–13.</li> <li>■ Updated “Write Bytes Operation”, “Erase Bulk Operation” and “Erase Sector Operation” sections.</li> <li>■ Updated Table 4–24 to include EPCS128 information.</li> <li>■ Updated (Note 1) to Table 4–26.</li> <li>■ Updated VCC and VI information to include EPCS128 in Table 4–27.</li> <li>■ Updated VIH information to include EPCS128 in Table 4–29.</li> <li>■ Updated ICC0 and ICC1 information to include EPCS128 in Table 4–30.</li> <li>■ Updated Figure 4–21 and Table 4–34 with EPCS128 information.</li> </ul>
April 2007	2.0	<ul style="list-style-type: none"> <li>■ Updated “Introduction” section.</li> <li>■ Updated “Functional Description” section and added handpara note.</li> <li>■ Added Table 4–4, Table 4–6, and Table 4–7.</li> <li>■ Updated “Active Serial FPGA Configuration” section and its handpara note.</li> <li>■ Added notes to Figure 4–2.</li> <li>■ Updated Table 4–26 and added (Note 1).</li> <li>■ Updated Figure 4–20.</li> <li>■ Updated Table 4–34.</li> </ul>
January 2007	1.7	<ul style="list-style-type: none"> <li>■ Removed reference to PLMSEPC-16 in “Programming and Configuration File Support”.</li> <li>■ Updated DCLK pin information in Table 4–32.</li> </ul>
October 2006	1.6	<ul style="list-style-type: none"> <li>■ Updated Figure 4–19.</li> <li>■ Updated Table 4–30 and Table 4–32.</li> </ul>
August 2005	1.5	<ul style="list-style-type: none"> <li>■ Updated table 4-4 to include EPCS64 support for Cyclone devices.</li> </ul>
August 2005	1.4	<ul style="list-style-type: none"> <li>■ Updated tables.</li> <li>■ Minor text updates.</li> </ul>
February 2005	1.3	<ul style="list-style-type: none"> <li>■ Updated hot socketing AC specifications.</li> </ul>



**Table 3-25.** Chapter Revision History (Part 3 of 3)

<b>Date</b>	<b>Version</b>	<b>Changes Made</b>
October 2003	1.2	<ul style="list-style-type: none"><li>■ Added Serial Configuration Device Memory Access section.</li><li>■ Updated timing information in Tables 4-10 and 4-11 section.</li><li>■ Updated timing information in Tables 4-16 and 4-17.</li></ul>
July 2003	1.1	<ul style="list-style-type: none"><li>■ Minor updates.</li></ul>
May 2003	1.0	<ul style="list-style-type: none"><li>■ Added document to the <i>Cyclone Device Handbook</i>.</li></ul>



## Features

This chapter describes configuration devices for SRAM-based LUT Devices.

- Configuration device family for configuring Altera® ACEX® 1K, APEX™ 20K (including APEX 20K, APEX 20KC, and APEX 20KE), APEX II, Arria® GX, Cyclone®, Cyclone II, FLEX® 10K (including FLEX 10KE and FLEX 10KA) Mercury™, Stratix®, Stratix GX, Stratix II, and Stratix II GX devices.
- Easy-to-use 4-pin interface to Altera FPGAs
- Low current during configuration and near-zero standby current
- 5.0-V and 3.3-V operation
- Programming support with the Altera Programming Unit (APU) and programming hardware from Data I/O, BP Microsystems, and other third-party programmers
- Available in compact plastic packages
  - 8-pin plastic dual in-line package (PDIP)
  - 20-pin plastic J-lead chip carrier (PLCC) package
  - 32-pin plastic thin quad flat pack (TQFP) package
- EPC2 device has reprogrammable Flash configuration memory
  - 5.0-V and 3.3-V in-system programmability (ISP) through the built-in IEEE Std. 1149.1 JTAG interface
  - Built-in JTAG boundary-scan test (BST) circuitry compliant with IEEE Std. 1149.1
  - Supports programming through Serial Vector Format Files (.svf), Jam™ Standard Test and Programming Language (STAPL) Files (.jam), Jam STAPL Byte-Code Files (.jbc), and the Quartus® II and MAX+PLUS® II softwares via the USB Blaster, MasterBlaster™, ByteBlaster™ II, EthernetBlaster, or ByteBlasterMV™ download cable
  - nINIT\_CONF pin allows INIT\_CONF JTAG instruction to begin FPGA configuration
  - Can be programmed with Programmer Object Files (.pof) for EPC1 and EPC1441 devices
  - Available in 20-pin PLCC and 32-pin TQFP packages

 For detailed information about enhanced configuration devices, refer to *Enhanced Configuration Devices (EPC4, EPC8 and EPC16) Data Sheet*.

 For detailed information about serial configuration devices, refer to *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64 and EPCS128) Data Sheet*.

## Functional Description

With SRAM-based devices, configuration data must be reloaded each time the device powers up, the system initializes, or when new configuration data is needed. Altera configuration devices store configuration data for SRAM-based ACEX 1K, APEX 20K, APEX II, Arria GX, Cyclone, Cyclone II, FLEX 10K, FLEX 6000, Mercury, Stratix, Stratix GX, Stratix II, and Stratix II GX devices. [Table 4-1](#) lists Altera configuration devices and their features.

**Table 4-1.** Altera Configuration Devices

Device	Memory Size (Bits)	ISP Support	Cascaded Support	Reprogrammable	Operating Voltage
EPC1	1,046,496	No	Yes	No	5.0 or 3.3 V
EPC2	1,695,680	Yes	Yes	Yes	5.0 or 3.3 V
EPC1064	65,536	No	No	No	5.0 V
EPC1064V	65,536	No	No	No	3.3 V
EPC1213	212,942	No	Yes	No	5.0 V
EPC1441	440,800	No	No	No	5.0 or 3.3 V

[Table 4-2](#) lists the supported configuration devices required to configure the ACEX 1K, APEX 20K, APEX 20KC, APEX 20KE, APEX II, Cyclone, Cyclone II, FLEX 10K, FLEX 10KA, FLEX 10KE, FLEX 6000/A, FLEX 8000A, Mercury, Stratix, Stratix GX, or Stratix II device.

**Table 4-2.** Configuration Devices Required (Part 1 of 4)

Family	Device	Data Size (Bits) (1)	EPC1064/ 1064V	EPC1213	EPC1441	EPC1	EPC2
ACEX 1K	EP1K10	159,160	—	—	1	1	1
	EP1K30	473,720	—	—	—	1	1
	EP1K50	784,184	—	—	—	1	1
	EP1K100	1,335,720	—	—	—	—	1
APEX 20K	EP20K100	993,360	—	—	—	1	1
	EP20K200	1,950,800	—	—	—	—	2
	EP20K400	3,880,720	—	—	—	—	3
APEX 20KC	EP20K200C	1,968,016	—	—	—	—	2
	EP20K400C	3,909,776	—	—	—	—	3
	EP20K600C	5,673,936	—	—	—	—	4
	EP20K1000C	8,960,016	—	—	—	—	6

**Table 4-2.** Configuration Devices Required (Part 2 of 4)

Family	Device	Data Size (Bits) (1)	EPC1064/ 1064V	EPC1213	EPC1441	EPC1	EPC2
APEX 20KE	EP20K30E	354,832	—	—	1	1	1
	EP20K60E	648,016	—	—	—	1	1
	EP20K100E	1,008,016	—	—	—	1	1
	EP20K160E	1,524,016	—	—	—	—	1
	EP20K200E	1,968,016	—	—	—	—	2
	EP20K300E	2,741,616	—	—	—	—	2
	EP20K400E	3,909,776	—	—	—	—	3
	EP20K600E	5,673,936	—	—	—	—	4
	EP20K1000E	8,960,016	—	—	—	—	6
	EP20K1500E	12,042,256	—	—	—	—	8
APEX II	EP2A15	1,168,688	—	—	—	—	3
	EP2A25	1,646,544	—	—	—	—	4
	EP2A40	2,543,016	—	—	—	—	6
	EP2A70	4,483,064	—	—	—	—	11
Cyclone	EP1C3	627,376	—	—	—	1	1
	EP1C4	925,000	—	—	—	1	1
	EP1C6	1,167,216	—	—	—	1 (2)	1
	EP1C12	2,326,528	—	—	—	—	1 (2)
	EP1C20	3,559,608	—	—	—	—	2 (2)
Cyclone II	EP2C5	1,265,792	—	—	—	—	1
	EP2C8	1,983,536	—	—	—	—	2
	EP2C20	3,892,496	—	—	—	—	3
	EP2C35	6,848,608	—	—	—	—	5
	EP2C50	9,951,104	—	—	—	—	6
	EP2C70	14,319,216	—	—	—	—	9
FLEX 10K	EPF10K10	118,000	—	—	1	1	1
	EPF10K20	231,000	—	—	1	1	1
	EPF10K30	376,000	—	—	1	1	1
	EPF10K40	498,000	—	—	—	1	1
	EPF10K50	621,000	—	—	—	1	1
	EPF10K70	892,000	—	—	—	1	1
	EPF10K100	1,200,000	—	—	—	—	1
FLEX 10KA	EPF10K10A	120,000	—	—	1	1	1
	EPF10K30A	406,000	—	—	1	1	1
	EPF10K50V	621,000	—	—	—	1	1
	EPF10K100A	1,200,000	—	—	—	—	1
	EPF10K130V	1,600,000	—	—	—	—	1
	EPF10K250A	3,300,000	—	—	—	—	2

**Table 4-2.** Configuration Devices Required (Part 3 of 4)

Family	Device	Data Size (Bits) (1)	EPC1064/ 1064V	EPC1213	EPC1441	EPC1	EPC2
FLEX 10KE	EPF10K30E	473,720	—	—	—	1	1
	EPF10K50E	784,184	—	—	—	1	1
	EPF10K50S	784,184	—	—	—	1	1
	EPF10K100B	1,200,000	—	—	—	—	1
	EPF10K100E	1,335,720	—	—	—	—	1
	EPF10K130E	1,838,360	—	—	—	—	2
	EPF10K200E	2,756,296	—	—	—	—	2
	EPF10K200S	2,756,296	—	—	—	—	2
FLEX 6000/A	EPF6010A	260,000	—	—	1	1	—
	EPF6016 (5.0V) / EPF6016A	260,000	—	—	1	1	—
	EPF6024A	398,000	—	—	1	1	—
FLEX 8000A	EPF8282A / EPF8282AV (3.3 V)	40,000	1	1	1	1	—
	EPF8452A	64,000	1	1	1	1	—
	EPF8636A	96,000	—	1	1	1	—
	EPF8820A	128,000	—	1	1	1	—
	EPF81188A	192,000	—	1	1	1	—
	EPF81500A	250,000	—	—	1	1	—
Mercury	EP1M120	1,303,120	—	—	—	—	1
	EP1M350	4,394,032	—	—	—	—	3
Stratix	EP1S10	3,534,640	—	—	—	—	3 (3)
	EP1S20	5,904,832	—	—	—	—	4
	EP1S25	7,894,144	—	—	—	—	5
	EP1S30	10,379,368	—	—	—	—	7
	EP1S40	12,389,632	—	—	—	—	8
	EP1S60	17,543,968	—	—	—	—	11
	EP1S80	23,834,032	—	—	—	—	15
	Stratix GX	EP1SGX10	3,534,640	—	—	—	—
EP1SGX25		7,894,144	—	—	—	—	5
EP1SGX40		12,389,632	—	—	—	—	8

**Table 4-2.** Configuration Devices Required (Part 4 of 4)

Family	Device	Data Size (Bits) (1)	EPC1064/ 1064V	EPC1213	EPC1441	EPC1	EPC2
Stratix II	EP2S15	5,000,000	—	—	—	—	3
	EP2S30	10,100,000	—	—	—	—	7
	EP2S60	17,100,000	—	—	—	—	11
	EP2S90	27,500,000	—	—	—	—	17
	EP2S130	39,600,000	—	—	—	—	24
	EP2S180	52,400,000	—	—	—	—	31

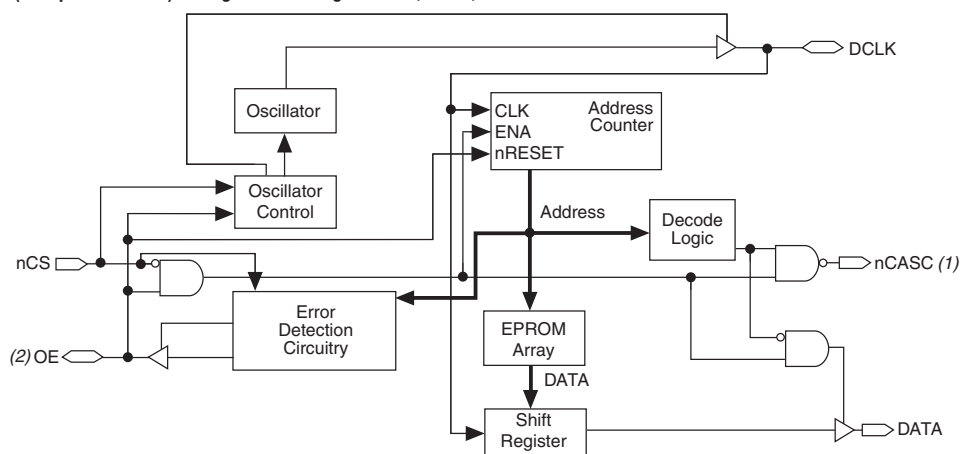
**Notes to Table 4-2:**

- (1) Raw Binary Files (.rbf) were used to determine these sizes.
- (2) This is with the Cyclone series compression feature enabled.
- (3) EP1S10 ES devices requires four EPC2 devices.

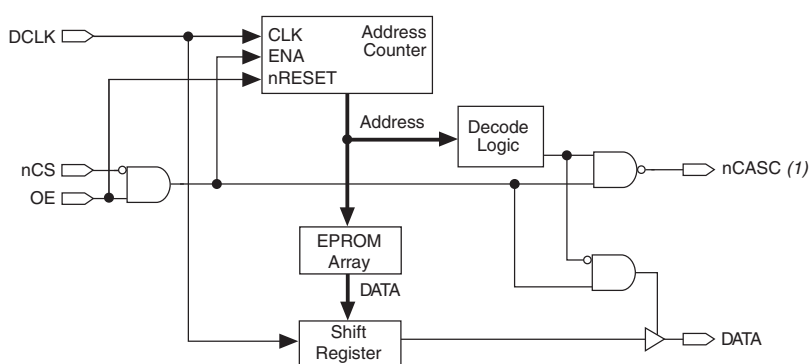
Figure 4-1 shows the configuration device block diagram.

**Figure 4-1.** Configuration Device Block Diagram

**FPGA (except FLEX 8000) Configuration Using an EPC2, EPC1, or EPC1441**



**FLEX 8000 Device Configuration Using an EPC1, EPC1441, EPC1213, EPC1064, or EPC1064V**



**Notes to Figure 4-1:**

- (1) The EPC1441 devices do not support data cascading. The EPC1, EPC2, and EPC1213 devices support data cascading.
- (2) The OE pin is a bidirectional open-drain pin.

## Device Configuration

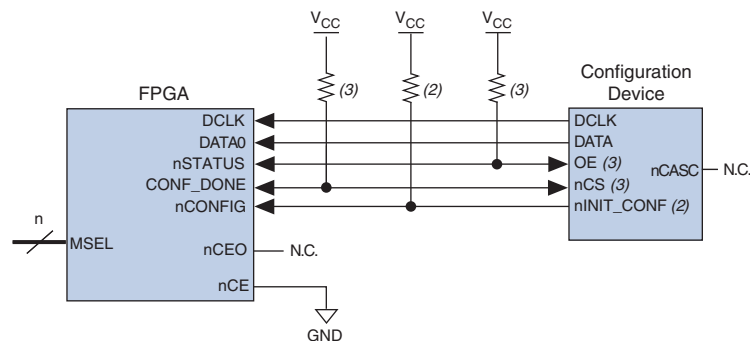
The EPC1, EPC2, and EPC1441 devices store configuration data in its EPROM array and serially clock data out using an internal oscillator. The OE, *nCS*, and DCLK pins supply the control signals for the address counter and the DATA output tri-state buffer. The configuration device sends a serial bitstream of configuration data to its DATA pin, which is routed to the DATA0 input of the FPGA.

The control signals for configuration devices (OE, *nCS*, and DCLK) interface directly with the FPGA control signals (*nSTATUS*, CONF\_DONE, and DCLK, respectively). All Altera FPGAs can be configured by a configuration device without requiring an external intelligent controller.

 An EPC2 device cannot configure FLEX 8000 or FLEX 6000 devices. Refer to [Table 4-2 on page 4-2](#) for the configuration devices that support FLEX 8000 and FLEX 6000 devices.

[Figure 4-2](#) shows the basic configuration interface connections between the configuration device and the Altera FPGA. For specific details about configuration interface connections, including pull-up resistor values, supply voltages and MSEL pin setting, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

**Figure 4-2.** Altera FPGA Configured Using an EPC1, EPC2, or EPC1441 Configuration Device (*Note 1*)



### Notes to Figure 4-2:

- (1) For specific details about configuration interface connections refer to the FPGA family chapter in the *Configuration Handbook*.
- (2) The *nINIT\_CONF* pin (available on EPC2 devices) has an internal pull-up resistor that is always active. This means an external pull-up resistor is not required 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 or not available, *nCONFIG* must be pulled to  $V_{CC}$  either directly or through a resistor.
- (3) EPC2 devices have internal programmable pull-up resistors on OE and *nCS*. 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 EPC2 device allows the user to begin configuration of the FPGA via an additional pin, *nINIT\_CONF*. The *nINIT\_CONF* pin of the EPC2 device can be connected to the *nCONFIG* of the FPGA, which allows the *INIT\_CONF* JTAG instruction to begin FPGA configuration. The *INIT\_CONF* JTAG instruction causes the EPC2 device to drive *nINIT\_CONF* low, which in turn pulls *nCONFIG* low. Pulling *nCONFIG* low on the




FPGA will reset the device. When the JTAG state machine exits this state, *nINIT\_CONF* is released and pulled high by an internal 1-k $\Omega$  resistor, which in turn pulls *nCONFIG* high to begin configuration. If its functionality is not used, the *nINIT\_CONF* pin does not need to be connected and *nCONFIG* of the FPGA must be pulled to  $V_{CC}$  either directly or through a resistor.

The EPC2 device's *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 configuration device's *OE* and *nCS* pins control the tri-state buffer on its *DATA* output pin, and enable the address counter and oscillator. When *OE* is driven low, the configuration device resets the address counter and tri-states its *DATA* pin. The *nCS* pin controls the *DATA* output of the configuration device. If *nCS* is held high after the *OE* reset pulse, the counter is disabled and the *DATA* output pin is tri-stated. If *nCS* is driven low after the *OE* reset pulse, the counter and *DATA* output pin are enabled. When *OE* is driven low again, the address counter is reset and the *DATA* output pin is tri-stated, regardless of the state of *nCS*.

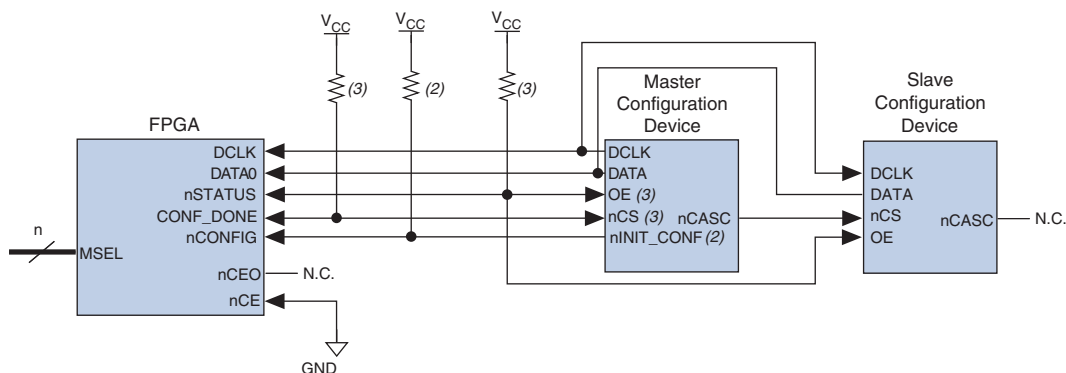
If the FPGA's configuration data exceeds the capacity of a single EPC1 or EPC2 configuration device, multiple EPC1 or EPC2 devices can be cascaded together. If multiple EPC1 or EPC2 devices are required, the *nCASC* and *nCS* pins provide handshaking between the configuration devices.

 EPC1441 and EPC1064/V devices cannot be cascaded.

When configuring ACEX 1K, APEX 20K, APEX II, Arria GX, Cyclone, Cyclone II, FLEX 10K, Mercury, Stratix, Stratix GX, Stratix II, and Stratix II GX devices with cascaded EPC1 or EPC2 devices, the position of the EPC1 or EPC2 device in the chain determines its mode of operation. The first configuration device in the chain is the master, while subsequent configuration devices are slaves. The *nINIT\_CONF* pin of the master EPC2 device can be connected to the *nCONFIG* of the FPGAs, which allows the *INIT\_CONF* JTAG instruction to begin FPGA configuration. The *nCS* pin of the master configuration device is connected to the *CONF\_DONE* of the FPGAs, while its *nCASC* pin is connected to *nCS* of the next slave configuration device in the chain. Additional EPC1 or EPC2 devices can be chained together by connecting *nCASC* to *nCS* of the next slave EPC1 or EPC2 device in the chain. The last device's *nCS* input comes from the previous device, while its *nCASC* pin is left floating. All other configuration pins (*DCLK*, *DATA*, and *OE*) are connected to every device in the chain.

Figure 4-3 shows the basic configuration interface connections between a configuration device chain and the Altera FPGA.

 For specific details about configuration interface connections, including pull-up resistor values, supply voltages and *MSEL* pin setting, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

**Figure 4-3.** Altera FPGA Configured Using Two EPC1 or EPC2 Configuration Devices (Note 1)**Notes to Figure 4-3:**


- (1) For specific details about configuration interface connections refer to the appropriate FPGA family chapter in the *Configuration Handbook*.
- (2) The *nINIT\_CONF* pin (available on EPC2 devices) has an internal pull-up resistor that is always active. This means an external pull-up resistor is not required 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 or not available, *nCONFIG* must be pulled to *V<sub>CC</sub>* either directly or through a resistor.
- (3) EPC2 devices have internal programmable pull-up resistors on *OE* and *nCS*. 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.

When the first device in a configuration device chain is powered-up or reset, its *nCS* pin is driven low because it is connected to the *CONF\_DONE* of the FPGA. Because both *OE* and *nCS* are low, the first device in the chain will recognize it is the master device and will control configuration. Since the slave devices' *nCS* pin is fed by the previous devices' *nCASC* pin, its *nCS* pin will be high upon power-up and reset. In the slave configuration devices, the *DATA* output is tri-stated and *DCLK* is an input. During configuration, the master device supplies the clock through *DCLK* to the FPGA and to any slave configuration devices. The master EPC1 or EPC2 device also provides the first stream of data to the FPGA during multi-device configuration. After the master EPC1 or EPC2 device finishes sending configuration data, it tri-states its *DATA* pin to avoid contention with other configuration devices. The master EPC1 or EPC2 device will also drive its *nCASC* pin low, which pulls the *nCS* pin of the next device low. This action signals the slave EPC1 or EPC2 device to start sending configuration data to the FPGAs.

The master EPC1 or EPC2 device clocks all slave configuration devices until configuration is complete. When all configuration data is transferred and the *nCS* pin on the master EPC1 or EPC2 device is driven high by the FPGA's *CONF\_DONE* pin, the master EPC1 or EPC2 device then goes into zero-power (idle) state. The master EPC2 device drives *DATA* high and *DCLK* low, while the EPC1 and EPC1441 device tri-state *DATA* and drive *DCLK* low.

If *nCS* on the master EPC1 or EPC2 device is driven high before all configuration data is transferred, the master EPC1 or EPC2 device drives its *OE* signal low, which in turn drives the FPGA's *nSTATUS* pin low, indicating a configuration error. Additionally, if the configuration device sends its data and detects that *CONF\_DONE* has not gone high, it recognizes that the FPGA has not configured successfully. EPC1 and EPC2 devices wait for 16 *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. Configuration automatically restarts if the **Auto-restart configuration on error** option is turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box or the MAX+PLUS II software's **Global Project Device Options** dialog box (Assign menu).

 For more information about FPGA configuration and configuration interface connections between configuration devices and Altera FPGAs, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.


## Power and Operation

This section describes Power-On Reset (POR) delay, error detection, and 3.3-V and 5.0-V operation of Altera configuration devices.

### Power-On Reset (POR)

During initial power-up, a POR delay occurs to permit voltage levels to stabilize. When configuring an FPGA with an EPC1, EPC2, or EPC1441 device, the POR delay occurs inside the configuration device, and the POR delay is a maximum of 200 ms. When configuring a FLEX 8000 device with an EPC1213, EPC1064, or EPC1064V device, the POR delay occurs inside the FLEX 8000 device, and the POR delay is typically, 100 ms, with a maximum of 200 ms.

During POR, the configuration device drives its OE pin low. This low signal delays configuration because the OE pin is connected to the target FPGA's *nSTATUS* pin. When the configuration device completes POR, it releases its open-drain OE pin, which is then pulled high by a pull-up resistor.

 The FPGA should be powered up before the configuration device exits POR to avoid the master configuration device from entering slave mode.

If the FPGA is not powered up before the configuration device exits POR, the CONF\_DONE/*nCS* line will be high because of the pull-up resistor. When the configuration device exits POR and releases OE, it sees *nCS* high, which signals the configuration device to enter slave mode. Therefore, configuration will not begin (the DATA output is tri-stated and DCLK is an input pin in slave mode).

### Error Detection Circuitry

The EPC1, EPC2, and EPC1441 configuration devices have built-in error detection circuitry for configuring ACEX 1K, APEX 20K, APEX II, Arria GX, Cyclone, Cyclone II, FLEX 10K, FLEX 6000, Mercury, Stratix, Stratix GX, Stratix II, or Stratix II GX devices.

Built-in error-detection circuitry uses the *nCS* pin of the configuration device, which monitors the CONF\_DONE pin on the FPGA. If *nCS* on the master EPC1 or EPC2 device is driven high before all configuration data is transferred, the master EPC1 or EPC2 device drives its OE signal low, which in turn drives the FPGA's *nSTATUS* pin low, indicating a configuration error. Additionally, if the configuration device sends is generated its data and detects that CONF\_DONE has not gone high, it recognizes that the FPGA has not configured successfully. EPC1 and EPC2 devices wait for 16 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. Configuration automatically restarts if the **Auto-restart configuration on error** option is turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box or the MAX+PLUS II software's **Global Project Device Options** dialog box (Assign menu).

In addition, if the FPGA detects a cyclical redundancy code (CRC) error in the received data, it will flag the error by driving nSTATUS low. This low signal on nSTATUS will drive the OE pin of the configuration device low, which will reset the configuration device. CRC checking is performed when configuring all Altera FPGAs.

### 3.3-V or 5.0-V Operation

The EPC1, EPC2, and EPC 1441 configuration device may be powered at 3.3 V or 5.0 V. For each configuration device, an option must be set for 5.0-V or 3.3-V operation.

For EPC1 and EPC1441 configuration devices, 3.3-V or 5.0-V operation is controlled by a programming bit in the .pof. The **Low-Voltage mode** option in the **Options** tab of the **Configuration Device Options** dialog box in the Quartus II software or the **Use Low-Voltage Configuration EPROM** option in the **Global Project Device Options** dialog box (Assign menu) in the MAX+PLUS II software sets this parameter. For example, EPC1 devices are programmed automatically to operate in 3.3-V mode when configuring FLEX 10KA devices, which have a V<sub>CC</sub> voltage of 3.3 V. In this example, the EPC1 device's VCC pin is connected to a 3.3-V power supply.

For EPC2 devices, this option is set externally by the VCCSEL pin. In addition, the EPC2 device has an externally controlled option, set by the VPPSEL pin, to adjust the programming voltage to 5.0 V or 3.3 V. The functions of the VCCSEL and VPPSEL pins are described below. These pins are only available in the EPC2 devices.

- VCCSEL pin – For EPC2 configuration devices, 5.0-V or 3.3-V operation is controlled by the VCCSEL option pin. The device functions in 5.0-V mode when VCCSEL is connected to GND; the device functions in 3.3-V mode when VCCSEL is connected to V<sub>CC</sub>.
- VPPSEL pin – The EPC2 V<sub>PP</sub> programming power pin is normally tied to V<sub>CC</sub>. For EPC2 devices operating at 3.3 V, it is possible to improve in-system programming times by setting V<sub>PP</sub> to 5.0 V. For all other configuration devices, V<sub>PP</sub> must be tied to V<sub>CC</sub>. The EPC2 device's VPPSEL pin must be set in accordance with the EPC2 VPP pin. If the VPP pin is supplied by a 5.0-V supply, VPPSEL must be connected to GND; if the VPP pin is supplied by a 3.3-V power supply, VPPSEL must be connected to V<sub>CC</sub>.

Table 4-3 describes the relationship between the  $V_{CC}$  and  $V_{PP}$  voltage levels and the required logic level for VCCSEL and VPPSEL. A logic level of high means the pin should be connected to  $V_{CC}$ , while a low logic level means the pin should be connected to GND.

**Table 4-3.** VCCSEL and VPPSEL Pin Functions on the EPC2

VCC Voltage Level (V)	VPP Voltage Level (V)	VCCSEL Pin Logic Level	VPPSEL Pin Logic Level
3.3	3.3	High	High
3.3	5.0	High	Low
5.0	5.0	Low	Low

At 3.3-V operation, all EPC2 inputs are 5.0-V tolerant, except DATA, DCLK, and nCASC. The DATA and DCLK pins are used only to interface between the EPC2 device and the FPGA it is configuring. The voltage tolerances of all EPC2 pins at 5.0 V and 3.3 V are listed in Table 4-4.

**Table 4-4.** EPC2 Input and Bidirectional Pin Voltage Tolerance

Pin	5.0-V Operation		3.3-V Operation	
	5.0-V Tolerant	3.3-V Tolerant	5.0-V Tolerant	3.3-V Tolerant
DATA	✓	✓	—	✓
DCLK	✓	✓	—	✓
nCASC	✓	✓	—	✓
OE	✓	✓	✓	✓
nCS	✓	✓	✓	✓
VCCSEL	✓	✓	✓	✓
VPPSEL	✓	✓	✓	✓
nINIT_CONF	✓	✓	✓	✓
TDI	✓	✓	✓	✓
TMS	✓	✓	✓	✓
TCK	✓	✓	✓	✓

If an EPC1, EPC2, or EPC1441 configuration device is powered at 3.3 V, the nSTATUS and CONF\_DONE pull-up resistors must be connected to 3.3 V. If these configuration devices are powered at 5.0 V, the nSTATUS and CONF\_DONE pull-up resistors can be connected to 3.3 V or 5.0 V.

## Programming and Configuration File Support

The Quartus II and MAX+PLUS II softwares provide programming support for Altera configuration devices. During compilation, the Quartus II and MAX+PLUS II softwares automatically generates a **.pof**, which can be used to program the configuration devices. In a multi-device project, the software can combine the programming files for multiple ACEX 1K, APEX 20K, APEX II, Arria GX, Cyclone, Cyclone II, FLEX 10K, Mercury, Stratix, Stratix GX, Stratix II, and Stratix II GX devices into one or more configuration devices. The software allows you to select the appropriate configuration device to most efficiently store the data for each FPGA.

All Altera configuration devices are programmable using Altera programming hardware in conjunction with the Quartus II or MAX+PLUS II software. In addition, many third part programmers offer programming hardware that supports Altera configuration devices.



An EPC2 device can be programmed with a **.pof** generated for an EPC1 or EPC1441 device. An EPC1 device can be programmed using a **.pof** generated for an EPC1441 device.

EPC2 configuration devices can be programmed in-system through its industry-standard 4-pin JTAG interface. ISP capability in the EPC2 devices provides ease in prototyping and FPGA functionality. When programming multiple EPC2 devices in a JTAG chain, the Quartus II and MAX+PLUS II softwares and other programming methods employ concurrent programming to simultaneously program multiple devices and reduce programming time. EPC2 devices can be programmed and erased up to 100 times.

After programming an EPC2 device in-system, FPGA configuration can be initiated by the EPC2 `INIT_CONF` JTAG instruction. Refer to [Table 4-6 on page 4-14](#).



For more information about programming and configuration support, refer to the following documents:

- [Altera Programming Hardware Data Sheet](#)
- [USB-Blaster Download Cable User Guide](#)
- [MasterBlaster Serial/USB Communications Cable User Guide](#)
- [ByteBlaster II Download Cable User Guide](#)
- [ByteBlasterMV Download Cable User Guide](#)
- [BitBlaster Serial Download Cable Data Sheet](#)

You can also program configuration devices using the Quartus II or MAX+PLUS II software with the APU, and the appropriate configuration device programming adapter.

Table 4-5 shows which programming adapter to use with each configuration device.

**Table 4-5.** Programming Adapters

Device	Package	Adapter
EPC2	20-pin J-Lead	PLMJ1213
	32-pin TQFP	PLMT1213
EPC1	8-pin DIP	PLMJ1213
	20-pin J-Lead	PLMJ1213
EPC1441	8-pin DIP	PLMJ1213
	20-pin J-Lead	PLMJ1213
	32-pin TQFP	PLMT1064

The following steps explain how to program Altera configuration devices using the Quartus II software and the APU:

1. Choose the **Quartus II Programmer** (Tools menu).
2. Load the appropriate **.pof** by clicking **Add**. The **Device** column displays the device for the current programming file.
3. Insert a blank configuration device into the programming adapter's socket.
4. Turn on the **Program/Configure**. You can also turn on **Verify** to verify the contents of a programmed device against the programming data loaded from a programming file.
5. Click **Start**.
6. After successful programming, you can place the configuration device on the PCB to configure the FPGA device.

The following steps explain how to program Altera configuration devices using the MAX+PLUS II software and the APU:

1. Open the MAX+PLUS II Programmer.
2. Load the appropriate **.pof** using the **Select Programming File** dialog box (File menu). By default, the **Programmer** loads the current project's **.pof**. The **Device** field displays the device for the current programming file.
3. Insert a blank configuration device into the programming adapter's socket.
4. Click **Program**.
5. After successful programming, you can place the configuration device on the PCB to configure the FPGA device.

If you are cascading EPC1 or EPC2 devices, you must generate multiple **.pof**. The first device **.pof** will have the same name as the project, while the second device **.pof** will have the same name as the first, but with a "\_1" extension (e.g., **top\_1.pof**).



## IEEE Std. 1149.1 (JTAG) Boundary-Scan Testing

The EPC2 provides JTAG BST circuitry that complies with the IEEE Std. 1149.1-1990 specification. JTAG boundary-scan testing can be performed before or after configuration, but not during configuration. The EPC2 device supports the JTAG instructions shown in [Table 4-6](#).

**Table 4-6.** EPC2 JTAG Instructions

JTAG Instruction	OPCODE	Description
SAMPLE/PRELOAD	00 0101 0101	Allows a snapshot of a signal at the device pins to be captured and examined during normal device operation, and permits an initial data pattern output at the device pins.
EXTEST	00 0000 0000	Allows the external circuitry and board-level interconnections to be tested by forcing a test pattern at the output pins and capturing results at the input pins.
BYPASS	11 1111 1111	Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through a selected device to adjacent devices during normal device operation.
IDCODE	00 0101 1001	Selects the device IDCODE register and places it between TDI and TDO, allowing the device IDCODE to be serially shifted out of TDO. The device IDCODE for the EPC2 configuration device is shown below: 0000 0001000000000010 00001101110 1
USERCODE	00 0111 1001	Selects the USERCODE register and places it between TDI and TDO, allowing the USERCODE to be serially shifted out of TDO. The 32-bit USERCODE is a programmable user-defined pattern.
INIT_CONF	00 0110 0001	This function initiates the FPGA re-configuration process by pulsing the <i>n</i> INIT_CONF pin low, which is connected to the FPGAs <i>n</i> CONFIG pins. After this instruction is updated, the <i>n</i> INIT_CONF pin is pulsed low when the JTAG state machine enters the Run-Test/Idle state. The <i>n</i> INIT_CONF pin is then released and <i>n</i> CONFIG is pulled high by the resistor after the JTAG state machine goes out of Run-Test/Idle state. The FPGA configuration starts after <i>n</i> CONFIG goes high. As a result, the FPGA is configured with the new configuration data stored in the configuration device. This function can be added to your programming file (.pof, .jam, .jbc) in the Quartus II software by enabling the <b>Initiate configuration after programming</b> option in the <b>Programmer options</b> window (Options menu). This instruction is also used by the MAX+PLUS II software, .jam files, and .jbc files.
ISP Instructions	—	These instructions are used when programming an EPC2 device via JTAG ports with a USB Blaster, MasterBlaster, ByteBlaster II, EthernetBlaster, or ByteBlaster MV download cable, or using a .jam, .jbc, or .svf via an embedded processor.

 For more information, refer to [Application Note 39 \(IEEE 1149.1 \(JTAG\) Boundary-Scan Testing in Altera Devices\)](#) or the EPC2 BSDL files on the Altera web site.



Figure 4-4 shows the timing requirements for the JTAG signals.

Figure 4-4. EPC2 JTAG Waveforms

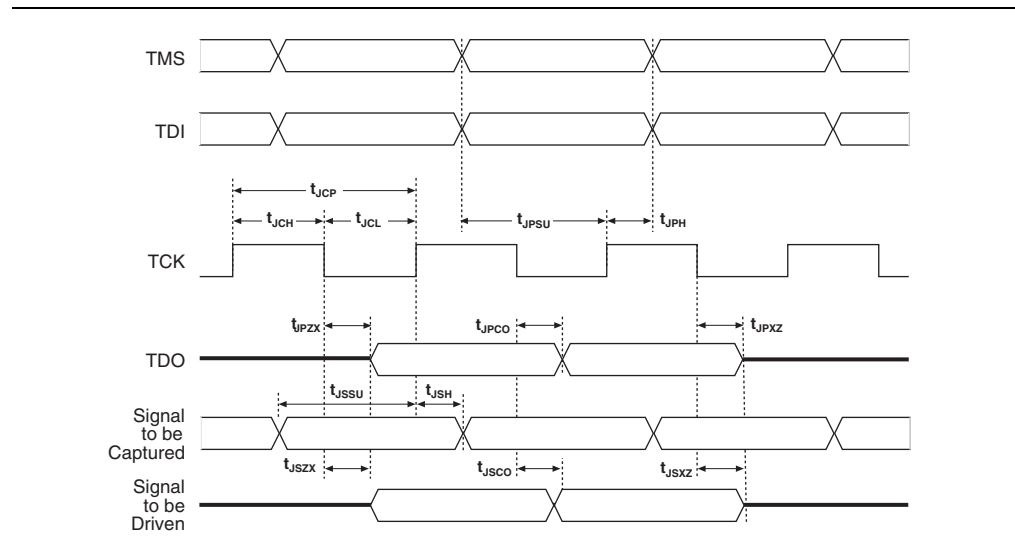


Table 4-7 shows the timing parameters and values for configuration devices.

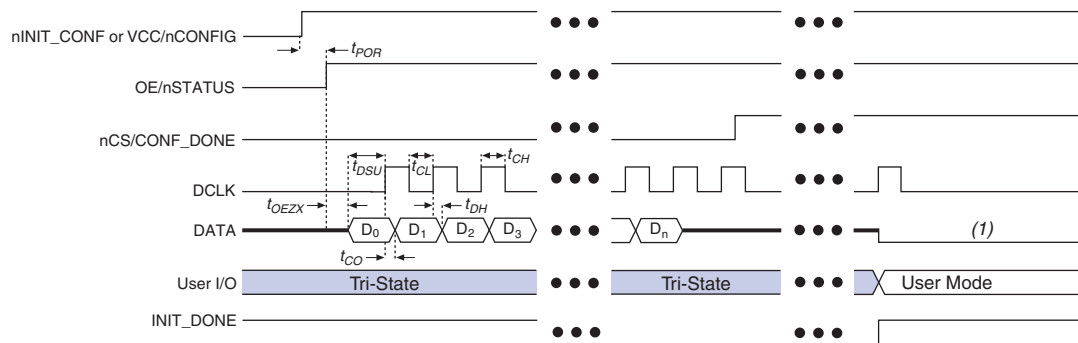
Table 4-7. JTAG Timing Parameters and Values

Symbol	Parameter	Min	Max	Unit
$t_{JCP}$	TCK clock period	100	—	ns
$t_{JCH}$	TCK clock high time	50	—	ns
$t_{JCL}$	TCK clock low time	50	—	ns
$t_{JPSU}$	JTAG port setup time	20	—	ns
$t_{JPH}$	JTAG port hold time	45	—	ns
$t_{JPCO}$	JTAG port clock to output	—	25	ns
$t_{JPZX}$	JTAG port high impedance to valid output	—	25	ns
$t_{JPXZ}$	JTAG port valid output to high impedance	—	25	ns
$t_{JSSU}$	Capture register setup time	20	—	ns
$t_{JSH}$	Capture register hold time	45	—	ns
$t_{JSCO}$	Update register clock to output	—	25	ns
$t_{JSZX}$	Update register high-impedance to valid output	—	25	ns
$t_{JSXZ}$	Update register valid output to high impedance	—	25	ns

## Timing Information

Figure 4-5 shows the timing waveform when using a configuration device.

**Figure 4-5.** Timing Waveform Using a Configuration Device



**Note to Figure 4-5:**

- (1) The EPC2 device will drive DCLK low and DATA high after configuration. The EPC1 and EPC1441 device will drive DCLK low and tri-state DATA after configuration.

Table 4-8 defines the timing parameters when using EPC2 devices at 3.3 V.

**Table 4-8.** Timing Parameters when Using EPC2 devices at 3.3 V

Symbol	Parameter	Min	Typ	Max	Units
$t_{POR}$	POR delay (1)	—	—	200	ms
$t_{OEZX}$	OE high to DATA output enabled	—	—	80	ns
$t_{CE}$	OE high to first rising edge on DCLK	—	—	300	ns
$t_{DSU}$	Data setup time before rising edge on DCLK	30	—	—	ns
$t_{DH}$	Data hold time after rising edge on DCLK	0	—	—	ns
$t_{CO}$	DCLK to DATA out	—	—	30	ns
$t_{CDOE}$	DCLK to DATA enable/disable	—	—	30	ns
$f_{CLK}$	DCLK frequency	5	7.7	12.5	MHz
$t_{MCH}$	DCLK high time for the first device in the configuration chain	40	65	100	ns
$t_{MCL}$	DCLK low time for the first device in the configuration chain	40	65	100	ns
$t_{SCH}$	DCLK high time for subsequent devices	40	—	—	ns
$t_{SCL}$	DCLK low time for subsequent devices	40	—	—	ns
$t_{CASC}$	DCLK rising edge to nCASC	—	—	25	ns
$t_{CCA}$	nCS to nCASC cascade delay	—	—	15	ns
$t_{OEW}$	OE low pulse width (reset) to guarantee counter reset	100	—	—	ns
$t_{OEC}$	OE low (reset) to DCLK disable delay	—	—	30	ns
$t_{NRCAS}$	OE low (reset) to nCASC delay	—	—	30	ns

**Note to Table 4-8:**

- (1) During initial power-up, a POR delay occurs to permit voltage levels to stabilize. Subsequent reconfigurations do not incur this delay.

Table 4-9 defines the timing parameters when using EPC1 and EPC1441 devices at 3.3 V.

**Table 4-9.** Timing Parameters when Using EPC1 and EPC1441 Devices at 3.3 V

Symbol	Parameter	Min	Typ	Max	Units
t <sub>POR</sub>	POR delay (1)	—	—	200	ms
t <sub>OEZX</sub>	OE high to DATA output enabled	—	—	80	ns
t <sub>CE</sub>	OE high to first rising edge on DCLK	—	—	300	ns
t <sub>DSU</sub>	Data setup time before rising edge on DCLK	30	—	—	ns
t <sub>DH</sub>	Data hold time after rising edge on DCLK	0	—	—	ns
t <sub>CO</sub>	DCLK to DATA out	—	—	30	ns
t <sub>CDOE</sub>	DCLK to DATA enable/disable	—	—	30	ns
f <sub>CLK</sub>	DCLK frequency	2	4	10	MHz
t <sub>MCH</sub>	DCLK high time for the first device in the configuration chain	50	125	250	ns
t <sub>MCL</sub>	DCLK low time for the first device in the configuration chain	50	125	250	ns
t <sub>SCH</sub>	DCLK high time for subsequent devices	50	—	—	ns
t <sub>SCL</sub>	DCLK low time for subsequent devices	50	—	—	ns
t <sub>CASC</sub>	DCLK rising edge to nCASC	—	—	25	ns
t <sub>CCA</sub>	nCS to nCASC cascade delay	—	—	15	ns
t <sub>OEw</sub>	OE low pulse width (reset) to guarantee counter reset	100	—	—	ns
t <sub>OEC</sub>	OE low (reset) to DCLK disable delay	—	—	30	ns
t <sub>NRCAS</sub>	OE low (reset) to nCASC delay	—	—	30	ns

**Note to Table 4-9:**

- (1) During initial power-up, a POR delay occurs to permit voltage levels to stabilize. Subsequent reconfigurations do not incur this delay.

Table 4-10 defines the timing parameters when using EPC1, EPC2, and EPC1441 devices at 5.0 V.

**Table 4-10.** Timing Parameters when Using EPC1, EPC2 and EPC1441 Devices at 5.0 V (Part 1 of 2)

Symbol	Parameter	Min	Typ	Max	Units
t <sub>POR</sub>	POR delay (1)	—	—	200	ms
t <sub>OEZX</sub>	OE high to DATA output enabled	—	—	50	ns
t <sub>CE</sub>	OE high to first rising edge on DCLK	—	—	200	ns
t <sub>DSU</sub>	Data setup time before rising edge on DCLK	30	—	—	ns
t <sub>DH</sub>	Data hold time after rising edge on DCLK	0	—	—	ns
t <sub>CO</sub>	DCLK to DATA out	—	—	20	ns
t <sub>CDOE</sub>	DCLK to DATA enable/disable	—	—	20	ns
f <sub>CLK</sub>	DCLK frequency	6.7	10	16.7	MHz
t <sub>MCH</sub>	DCLK high time for the first device in the configuration chain	30	50	75	ns
t <sub>MCL</sub>	DCLK low time for the first device in the configuration chain	30	50	75	ns

**Table 4-10.** Timing Parameters when Using EPC1, EPC2 and EPC1441 Devices at 5.0 V (Part 2 of 2)

Symbol	Parameter	Min	Typ	Max	Units
$t_{SCH}$	DCLK high time for subsequent devices	30	—	—	ns
$t_{SCL}$	DCLK low time for subsequent devices	30	—	—	ns
$t_{CASC}$	DCLK rising edge to $nCASC$	—	—	20	ns
$t_{CCA}$	$nCS$ to $nCASC$ cascade delay	—	—	10	ns
$t_{OEw}$	OE low pulse width (reset) to guarantee counter reset	100	—	—	ns
$t_{OEC}$	OE low (reset) to DCLK disable delay	—	—	20	ns
$t_{NRCAS}$	OE low (reset) to $nCASC$ delay	—	—	25	ns

**Note to Table 4-10:**

(1) During initial power-up, a POR delay occurs to permit voltage levels to stabilize. Subsequent reconfigurations do not incur this delay.

Table 4-11 defines the timing parameters when using EPC1, EPC1064, EPC1064V, EPC1213, and EPC1441 devices when configuring FLEX 8000 device.

**Table 4-11.** FLEX 8000 Device Configuration Parameters Using EPC1, EPC1064, EPC1064V, EPC1213, and EPC1441 Devices

Symbol	Parameter	EPC1064V		EPC1064 EPC1213		EPC1 EPC1441		Unit
		Min	Max	Min	Max	Min	Max	
$t_{OEZX}$	OE high to DATA output enabled	—	75	—	50	—	50	ns
$t_{CSZX}$	$nCS$ low to DATA output enabled	—	75	—	50	—	50	ns
$t_{CSXZ}$	$nCS$ high to DATA output disabled	—	75	—	50	—	50	ns
$t_{CSS}$	$nCS$ low setup time to first DCLK rising edge	150	—	100	—	50	—	ns
$t_{CSH}$	$nCS$ low hold time after DCLK rising edge	0	—	0	—	0	—	ns
$t_{DSU}$	Data setup time before rising edge on DCLK	75	—	50	—	50	—	ns
$t_{DH}$	Data hold time after rising edge on DCLK	0	—	0	—	0	—	ns
$t_{CO}$	DCLK to DATA out delay	—	100	—	75	—	75	ns
$t_{CK}$	Clock period	240	—	160	—	100	—	ns
$f_{CK}$	Clock frequency	—	4	—	6	—	8	MHz
$t_{CL}$	DCLK low time	120	—	80	—	50	—	ns
$t_{CH}$	DCLK high time	120	—	80	—	50	—	ns
$t_{XZ}$	OE low or $nCS$ high to DATA output disabled	—	75	—	50	—	50	ns
$t_{OEw}$	OE pulse width to guarantee counter reset	150	—	100	—	100	—	ns
$t_{CASC}$	Last DCLK + 1 to $nCASC$ low delay	—	90	—	60	—	50	ns
$t_{CKXZ}$	Last DCLK + 1 to DATA tri-state delay	—	75	—	50	—	50	ns
$t_{CEOUT}$	$nCS$ high to $nCASC$ high delay	—	150	—	100	—	100	ns

## Operating Conditions

Table 4-12 through Table 4-19 provide information about absolute maximum ratings, recommended operating conditions, DC operating conditions, and capacitance for configuration devices.

**Table 4-12.** Absolute Maximum Ratings (Note 1)

Symbol	Parameter	Conditions	Min	Max	Unit
$V_{CC}$	Supply voltage	With respect to ground (2)	-2.0	7.0	V
$V_I$	DC input voltage	With respect to ground (2)	-2.0	7.0	V
$I_{MAX}$	DC $V_{CC}$ or ground current	—	—	50	mA
$I_{OUT}$	DC output current, per pin	—	-25	25	mA
$P_D$	Power dissipation	—	—	250	mW
$T_{STG}$	Storage temperature	No bias	-65	150	°C
$T_{AMB}$	Ambient temperature	Under bias	-65	135	°C
$T_J$	Junction temperature	Under bias	—	135	°C

**Table 4-13.** Recommended Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
$V_{CC}$	Supply voltage for 5.0-V operation	(3), (4)	4.75 (4.50)	5.25 (5.50)	V
	Supply voltage for 3.3-V operation	(3), (4)	3.0 (3.0)	3.6 (3.6)	V
$V_I$	Input voltage	With respect to ground	-0.3	$V_{CC} + 0.3$ (5)	V
$V_O$	Output voltage	—	0	$V_{CC}$	V
$T_A$	Operating temperature	For commercial use	0	70	°C
		For industrial use	-40	85	°C
$t_R$	Input rise time	—	—	20	ns
$t_F$	Input fall time	—	—	20	ns

**Table 4-14.** DC Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
$V_{IH}$	High-level input voltage	—	2.0	$V_{CC} + 0.3$ (5)	V
$V_{IL}$	Low-level input voltage	—	-0.3	0.8	V
$V_{OH}$	5.0-V mode high-level TTL output voltage	$I_{OH} = -4$ mA DC (6)	2.4	—	V
	3.3-V mode high-level CMOS output voltage	$I_{OH} = -0.1$ mA DC (6)	$V_{CC} - 0.2$	—	V
$V_{OL}$	Low-level output voltage	$I_{OL} = 4$ mA DC (6)	—	0.4	V
$I_I$	Input leakage current	$V_I = V_{CC}$ or ground	-10	10	μA
$I_{OZ}$	Tri-state output off-state current	$V_O = V_{CC}$ or ground	-10	10	μA

**Table 4-15.** EPC1064, EPC1064V, and EPC1213 Device  $I_{CC}$  Supply Current Values

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$I_{CC0}$	$V_{CC}$ supply current (standby)	—	—	100	200	$\mu\text{A}$
$I_{CC1}$	$V_{CC}$ supply current (during configuration)	—	—	10	50	mA

**Table 4-16.** EPC2 Device Values

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$I_{CC0}$	$V_{CC}$ supply current (standby)	$V_{CC} = 5.0\text{ V or }3.3\text{ V}$	—	50	100	$\mu\text{A}$
$I_{CC1}$	$V_{CC}$ supply current (during configuration)	$V_{CC} = 5.0\text{ V or }3.3\text{ V}$	—	18	50	mA
$R_{CONF}$	Configuration pins	Internal pull up (OE, nCS, nINIT_CONF)	—	1	—	$\text{k}\Omega$

**Table 4-17.** EPC1 Device  $I_{CC}$  Supply Current Values

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$I_{CC0}$	$V_{CC}$ supply current (standby)	—	—	50	100	$\mu\text{A}$
$I_{CC1}$	$V_{CC}$ supply current (during configuration)	$V_{CC} = 5.0\text{ V}$	—	30	50	mA
		$V_{CC} = 3.3\text{ V}$	—	10	16.5	mA

**Table 4-18.** EPC1441 Device  $I_{CC}$  Supply Current Values

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$I_{CC0}$	$V_{CC}$ supply current (standby)	—	—	30	60	$\mu\text{A}$
$I_{CC1}$	$V_{CC}$ supply current (during configuration)	$V_{CC} = 5.0\text{ V}$	—	15	30	mA
$I_{CC1}$	$V_{CC}$ supply current (during configuration)	$V_{CC} = 3.3\text{ V}$	—	5	10	mA

**Table 4-19.** Capacitance (Note 7)


Symbol	Parameter	Conditions	Min	Max	Unit
$C_{IN}$	Input pin capacitance	$V_{IN} = 0\text{ V}, f = 1.0\text{ MHz}$	—	10	pF
$C_{OUT}$	Output pin capacitance	$V_{OUT} = 0\text{ V}, f = 1.0\text{ MHz}$	—	10	pF

**Notes to Table 4-12 through Table 4-19:**

- For more information, refer to the Operating Requirements for Altera Devices Data Sheet.
- The minimum DC input is -0.3 V. During transitions, the inputs may undershoot to -2.0 V or overshoot to 7.0 V for input currents less than 100 mA and periods shorter than 20 ns under no-load conditions.
- Numbers in parentheses are for industrial temperature range devices.
- Maximum  $V_{CC}$  rise time is 100 ms.
- Certain EPC2 pins may be driven to 5.75 V when operated with a 3.3-V  $V_{CC}$ . Refer to Table 4-4 on page 4-11.
- The  $I_{OH}$  parameter refers to high-level TTL or CMOS output current; the  $I_{OL}$  parameter refers to low-level TTL or CMOS output current.
- Capacitance is sample-tested only.

## Pin Information

Table 4-20 describes EPC1, EPC2, and EPC1441 pin functions during device configuration.

 For pin information about enhanced configuration devices, refer to the *Enhanced Configuration Devices (EPC4, EPC8 and EPC16) Data Sheet*.

 For pin information about serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64 and EPCS128) Data Sheet*.

**Table 4-20.** EPC1, EPC2 and EPC1441 Pin Functions During Configuration (Part 1 of 3)

Pin Name	Pin Number			Pin Type	Description
	8-Pin PDIP (1)	20-Pin PLCC	32-Pin TQFP (2)		
DATA	1	2	31	Output	<p>Serial data output. The DATA pin connects to the DATA0 of the FPGA. DATA is latched into the FPGA on the rising edge of DCLK.</p> <p>The DATA pin is tri-stated before configuration and when the nCS pin is high. After configuration, the EPC2 device will drive DATA high, while the EPC1 and EPC1441 device will tri-state DATA.</p>
DCLK	2	4	2	Bidirectional	<p>Clock output when configuring with a single configuration device or when the configuration device is the first (master) device in a chain. Clock input for the next (slave) configuration devices in a chain. The DCLK pin connects to the DCLK of the FPGA.</p> <p>Rising edges on DCLK increment the internal address counter and present the next bit of data on the DATA pin. The counter is incremented only if the OE input is held high, the nCS input is held low, and all configuration data has not been transferred to the target device.</p> <p>After configuration or when OE is low, the EPC1, EPC2 and EPC1441 device will drive DCLK low.</p>
OE	3	8	7	Open-Drain Bidirectional	<p>Output enable (active high) and reset (active low). The OE pin connects to the nSTATUS of the FPGA.</p> <p>A low logic level resets the address counter. A high logic level enables DATA and the address counter to count. If this pin is low (reset) during configuration, the internal oscillator becomes inactive and DCLK drives low. Refer to <a href="#">“Error Detection Circuitry” on page 4-9</a>.</p> <p>The OE pin has an internal programmable 1-kΩ resistor in EPC2 devices. If internal pull-up resistors are use, external pull-up resistors should not be used on these pins. The internal pull-up resistors can be disabled through the <b>Disable nCS and OE pull-ups on configuration device</b> option.</p>

**Table 4-20.** EPC1, EPC2 and EPC1441 Pin Functions During Configuration (Part 2 of 3)

Pin Name	Pin Number			Pin Type	Description
	8-Pin PDIP (1)	20-Pin PLCC	32-Pin TQFP (2)		
<i>nCS</i>	4	9	10	Input	<p>Chip select input (active low). The <i>nCS</i> pin connects to the <i>CONF_DONE</i> of the FPGA.</p> <p>A low input allows <i>DCLK</i> to increment the address counter and enables <i>DATA</i> to drive out. If the EPC1 or EPC2 is reset (<i>OE</i> pulled low) while <i>nCS</i> is low, the device initializes as the master device in a configuration chain. If the EPC1 or EPC2 device is reset (<i>OE</i> pulled low) while <i>nCS</i> is high, the device initializes as a slave device in the chain.</p> <p>The <i>nCS</i> pin has an internal programmable 1-k<math>\Omega</math> resistor in EPC2 devices. If internal pull-up resistors are use, external pull-up resistors should not be used on these pins. The internal pull-up resistors can be disabled through the <b>Disable <i>nCS</i> and <i>OE</i> pull-ups on configuration device</b> option.</p>
<i>nCASC</i>	6	12	15	Output	<p>Cascade select output (active low).</p> <p>This output goes low when the address counter has reached its maximum value. When the address counter has reached its maximum value, the configuration device has sent all its configuration data to the FPGA. In a chain of EPC1 or EPC2 devices, the <i>nCASC</i> pin of one device is connected to the <i>nCS</i> pin of the next device, which permits <i>DCLK</i> to clock data from the next EPC1 or EPC2 device in the chain. For single EPC1 or EPC2 devices and the last device in the chain, <i>nCASC</i> is left floating.</p> <p>This pin is only available in EPC1 and EPC2 devices, which support data cascading.</p>
<i>nINIT_CONF</i>	N/A	13	16	Open-Drain Output	<p>Allows the <i>INIT_CONF</i> JTAG instruction to initiate configuration. The <i>nINIT_CONF</i> pin connects to the <i>nCONFIG</i> of the FPGA.</p> <p>If multiple EPC2 devices are used to configure a FPGA, the <i>nINIT_CONF</i> of the first EPC2 pin is tied to the FPGA's <i>nCONFIG</i> pin, while subsequent devices' <i>nINIT_CONF</i> pins are left floating.</p> <p>The <i>INIT_CONF</i> pin has an internal 1-k<math>\Omega</math> pull-up resistor that is always active in EPC2 devices.</p> <p>This pin is only available in EPC2 devices.</p>
<i>TDI</i>	N/A	11	13	Input	<p>JTAG data input pin. Connect this pin to <i>V<sub>CC</sub></i> if the JTAG circuitry is not used.</p> <p>This pin is only available in EPC2 devices.</p>
<i>TDO</i>	N/A	1	28	Output	<p>JTAG data output pin. Do not connect this pin if the JTAG circuitry is not used.</p> <p>This pin is only available in EPC2 devices.</p>



**Table 4-20.** EPC1, EPC2 and EPC1441 Pin Functions During Configuration (Part 3 of 3)

Pin Name	Pin Number			Pin Type	Description
	8-Pin PDIP (1)	20-Pin PLCC	32-Pin TQFP (2)		
TMS	N/A	19	25	Input	JTAG mode select pin. Connect this pin to $V_{CC}$ if the JTAG circuitry is not used. This pin is only available in EPC2 devices.
TCK	N/A	3	32	Input	JTAG clock pin. Connect this pin to ground if the JTAG circuitry is not used. This pin is only available in EPC2 devices.
VCCSEL	N/A	5	3	Input	Mode select for $V_{CC}$ supply. $V_{CCSEL}$ must be connected to ground if the device uses a 5.0-V power supply ( $V_{CC} = 5.0$ V). $V_{CCSEL}$ must be connected to $V_{CC}$ if the device uses a 3.3-V power supply ( $V_{CC} = 3.3$ V). This pin is only available in EPC2 devices.
VPPSEL	N/A	14	17	Input	Mode select for $V_{PP}$ . $V_{PPSEL}$ must be connected to ground if $V_{PP}$ uses a 5.0-V power supply ( $V_{PP} = 5.0$ V). $V_{PPSEL}$ must be connected to $V_{CC}$ if $V_{PP}$ uses a 3.3-V power supply ( $V_{PP} = 3.3$ V). This pin is only available in EPC2 devices.
VPP	N/A	18	23	Power	Programming power pin. For the EPC2 device, this pin is normally tied to $V_{CC}$ . If the EPC2 $V_{CC}$ is 3.3 V, $V_{PP}$ can be tied to 5.0 V to improve in-system programming times. For EPC1 and EPC1441 devices, $V_{PP}$ must be tied to $V_{CC}$ . This pin is only available in EPC2 devices.
VCC	7, 8	20	27	Power	Power pin.
GND	5	10	12	Ground	Ground pin. A 0.2- $\mu$ F decoupling capacitor must be placed between the VCC and GND pins.

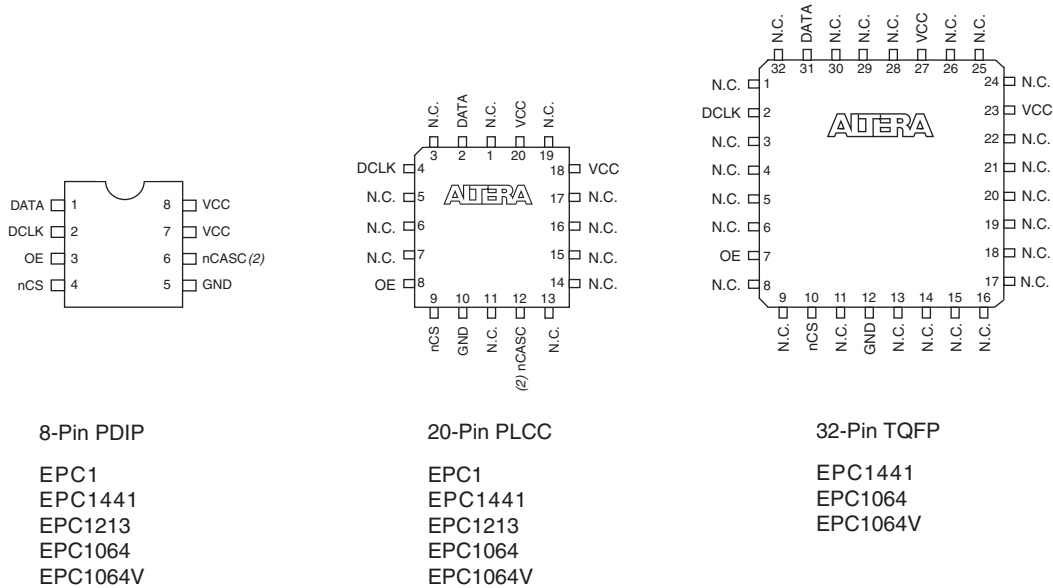
**Notes to Table 4-20:**

- (1) This package is available for EPC1 and EPC1441 devices only.
- (2) This package is available for EPC2 and EPC1441 devices only.

# Package

Figure 4-6 and Figure 4-7 show the configuration device package pin-outs.

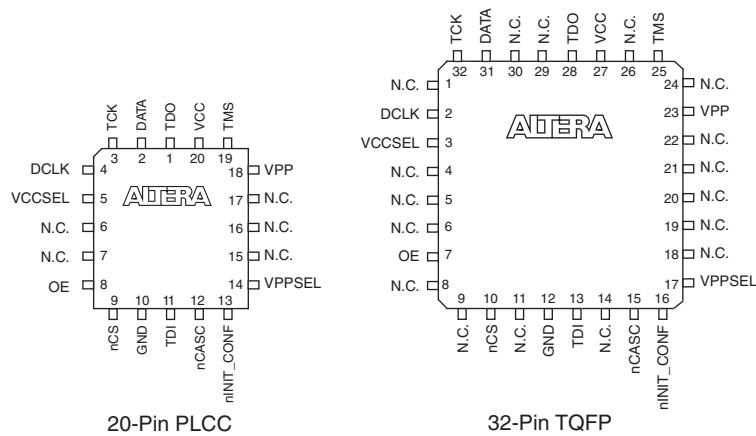
**Figure 4-6.** EPC1, EPC1064, EPC1064V, EPC1213, and EPC1441 Package Pin-Out Diagrams (Note 1)



**Notes to Figure 4-6:**

- (1) EPC1 and EPC1441 devices are one-time programmable devices. ISP is not available in these devices.
- (2) The nCASC pin is available on EPC1 devices, which allows them to be cascaded. On the EPC1441 devices, nCASC is a reserved pin and should be left unconnected.

**Figure 4-7.** EPC2 Package Pin-Out Diagrams



For package outlines and drawings, refer to the [Altera Device Package Information Data Sheet](#).

## Ordering Codes

Table 4–21. shows the ordering codes for the EPC1, EPC2, and EPC1441 configuration devices.

**Table 4–21.** Configuration Device Ordering Codes

Device	Package	Temperature	Ordering Code
EPC2	32-pin TQFP	Commercial	EPC2TC32
EPC2	32-pin TQFP	Industrial	EPC2TI32
EPC2	20-pin PLCC	Commercial	EPC2LC20
EPC2	20-pin PLCC	Industrial	EPC2LI20
EPC1	20-pin PLCC	Commercial	EPC1LC20
EPC1	20-pin PLCC	Industrial	EPC1LI20
EPC1	8-pin PDIP	Commercial	EPC1PC8
EPC1	8-pin PDIP	Industrial	EPC1PI8
EPC1441	32-pin TQFP	Commercial	EPC1441TC32
EPC1441	32-pin TQFP	Industrial	EPC1441TI32
EPC1441	20-pin PLCC	Commercial	EPC1441LC20
EPC1441	20-pin PLCC	Industrial	EPC1441LI20
EPC1441	8-pin PDIP	Commercial	EPC1441PC8
EPC1441	8-pin PDIP	Industrial	EPC1441PI8

## Chapter Revision History

Table 4–22 lists the revision history for this chapter.

**Table 4–22.** Chapter Revision History

Date	Version	Changes Made
December 2009	2.4	<ul style="list-style-type: none"> <li>■ Updated “Features” section.</li> <li>■ Removed “Referenced Documents” section.</li> </ul>
October 2008	2.3	<ul style="list-style-type: none"> <li>■ Updated “Features” and “IEEE Std. 1149.1 (JTAG) Boundary-Scan Testing” sections.</li> <li>■ Updated Table 5–2 and Table 5–16.</li> <li>■ Added “Referenced Documents” section.</li> <li>■ Updated new document format.</li> </ul>
April 2007	2.2	<ul style="list-style-type: none"> <li>■ Added document revision history.</li> </ul>
July 2004	2.0	<ul style="list-style-type: none"> <li>■ Added Stratix II and Cyclone II device information throughout chapter.</li> </ul>
September 2003	1.0	<ul style="list-style-type: none"> <li>■ Initial Release.</li> </ul>

