This datasheet describes enhanced configuration (EPC) devices.

Supported Devices

Table 1 lists the supported Altera® EPC devices.

Table 1. Altera EPC Devices

Device	Memory Size (bits)	On-Chip Decompression Support	ISP Support	Cascading Support	Reprogrammable	Recommended 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

Features

EPC devices offer the following features:

■ 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-Mb 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 or sector protection capability using the external flash interface

   ■ Supported in EPC4 and EPC16 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) with an 8-bit data output per DCLK cycle

■ Supports true n-bit concurrent configuration (n = 1, 2, 4, and 8) of Altera FPGAs

101 Innovation Drive
San Jose, CA 95134
www.altera.com

ISO
9001:2008
Registered

January 2012   Altera Corporation

Subscribe

- 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 and you can program the internal oscillator for higher frequencies of 33, 50, and 66 MHz
    - Clock synthesis supported using 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 using Jam™ Standard Test and Programming Language (STAPL)
- Supports JTAG boundary scan
- The *n*INIT_CONF pin allows private JTAG instruction to start FPGA configuration
- Internal pull-up resistor on the *n*INIT_CONF pin 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 configuration chapter in the appropriate device handbook.

Functional Description

The Altera EPC device is a single device with high speed and advanced configuration solution for high-density FPGAs. The core of an EPC 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 than one Altera FPGAs. Unused portions of the flash memory can be used to store processor code or data that can be accessed using the external flash interface after the FPGA configuration is complete.

Table 2 lists the supported EPC 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 2. Supported EPC Devices for Each Device Family (Part 1 of 3)

Device Family	Device	Data Size (Bits) [1]	EPC Devices		
			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	—	—	—

Table 2. Supported EPC Devices for Each Device Family (Part 2 of 3)

Device Family	Device	Data Size (Bits) (1)	EPC Devices		
			EPC4 (2)	EPC8 (2)	EPC16 (2)
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
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

Table 2. Supported EPC Devices for Each Device Family (Part 3 of 3)

Device Family	Device	Data Size (Bits) [1]	EPC Devices		
			EPC4 [2]	EPC8 [2]	EPC16 [2]
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
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 2:

(1) The Raw Binary File (**.rbf**) sizes are used to determine the data size for each device.

(2) These values are calculated with the compression feature of the EPC device enabled.

For more information about EPC devices, refer to the *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 3 lists the supported flash memory for all EPC devices.

Table 3. EPC Devices Flash Memory

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]
EPC16	Commercial	UBGA 88	Intel [1] or Sharp	Intel [1] or Sharp
	Industrial	UBGA 88	Intel [1] or Sharp	Intel [1]
	Military	UBGA 88	Intel [1]	Intel [1]
	Commercial/ Industrial	PQFP 100	Intel [1] or Sharp	Intel [1]

Note to Table 3:

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

The external flash interface feature is supported in EPC4 and EPC16 devices. For more information about using this feature in the EPC8 device, contact Altera Applications 24/7 Technical Support.

EPC 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. 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—an interface available after ISP and configuration

Figure 1 shows a block diagram of the EPC device.

Figure 1. EPC Device Block Diagram
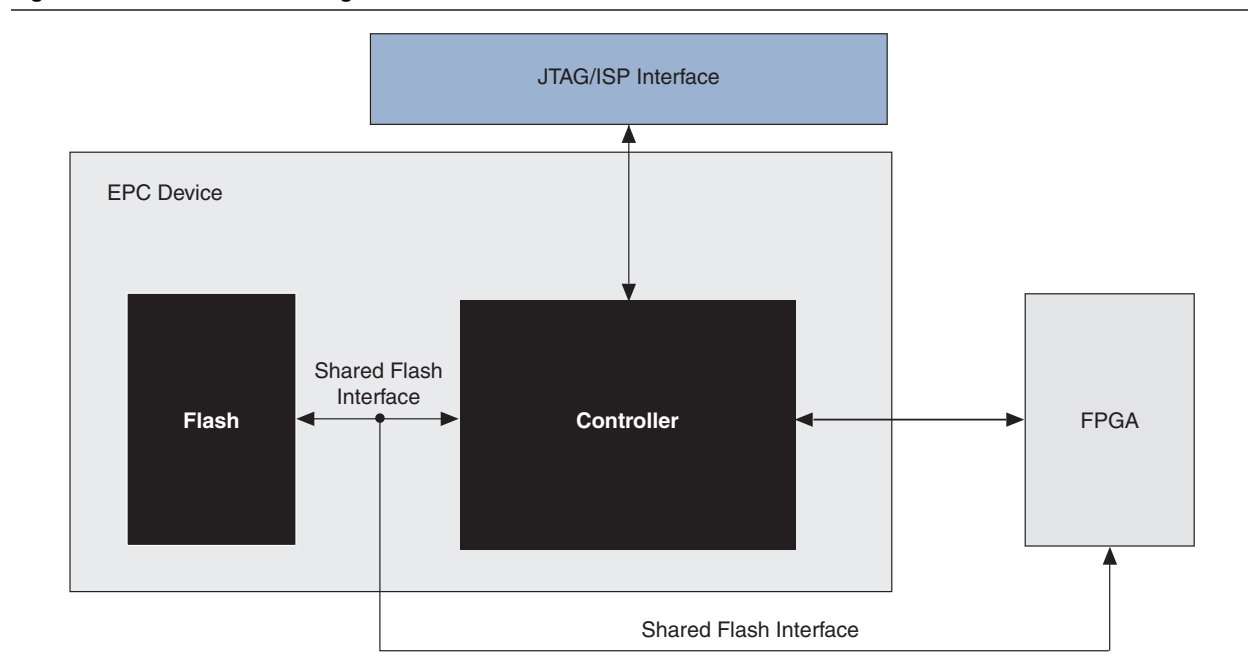
The EPC device features multiple configuration schemes. In addition to supporting the traditional passive serial (PS) configuration scheme for a single device or a serial-device chain, the EPC device features concurrent configuration and parallel configuration. With the concurrent configuration scheme, up to eight PS device chains can be configured simultaneously. In the FPP configuration scheme, 8-bits of data are clocked into the FPGA during each cycle. These configuration schemes offer significantly reduced configuration times over traditional schemes.

Furthermore, the EPC 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 the 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 EPC devices are compatible with the remote system configuration feature of the Stratix device.

For more information, 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 POR delay (PORSEL)

FPGA Configuration

FPGA configuration is managed by the configuration controller chip. This process includes reading configuration data from the flash memory, decompressing the configuration data, transmitting configuration data using 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 signals. 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 EPC 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] signals). 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 EPC 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 example, 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 the *n*STATUS signal low and triggers another configuration cycle.

A cyclic 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 *n*STATUS signal low), the controller stops configuration. If the **Auto-Restart Configuration After Error** option is enabled in the FPGA, it releases its *n*STATUS 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 the DCLK pin 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 describe the different configuration schemes supported by the EPC device—FPP, PS, and concurrent configuration schemes.

For more information, refer to the configuration chapter in the appropriate device handbook.

Configuration Signals

Table 4 lists the configuration signal connections between the EPC device and Altera FPGAs.

Table 4. Configuration Signals

EPC Device Pin	Altera FPGA Pin	Description
DATA[]	DATA[]	Configuration data transmitted from the EPC device to the FPGA, which is latched on the rising edge of DCLK.
DCLK	DCLK	EPC device generated clock used by the FPGA to latch configuration data provided on the DATA[] pins.
*n*INIT_CONF, which	*n*CONFIG	Open-drain output from the EPC 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 *n*INIT_CONF is not connected to *n*CONFIG, *n*CONFIG must be tied to V_{CC} either directly or through a pull-up resistor.
OE	*n*STATUS	Open-drain bidirectional configuration status signal, which is driven low by either the EPC device or FPGA during POR and to signal an error during configuration. Low pulse on OE resets the EPC device controller.
*n*CS	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 EPC device in the FPP configuration mode. In this mode, the EPC 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 2 shows the EPC 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 more information about configuration interface connections including the pull-up resistor values, supply voltages, and MSEL pin settings, refer to the configuration chapter in the appropriate device handbook.

Figure 2. FPP Configuration
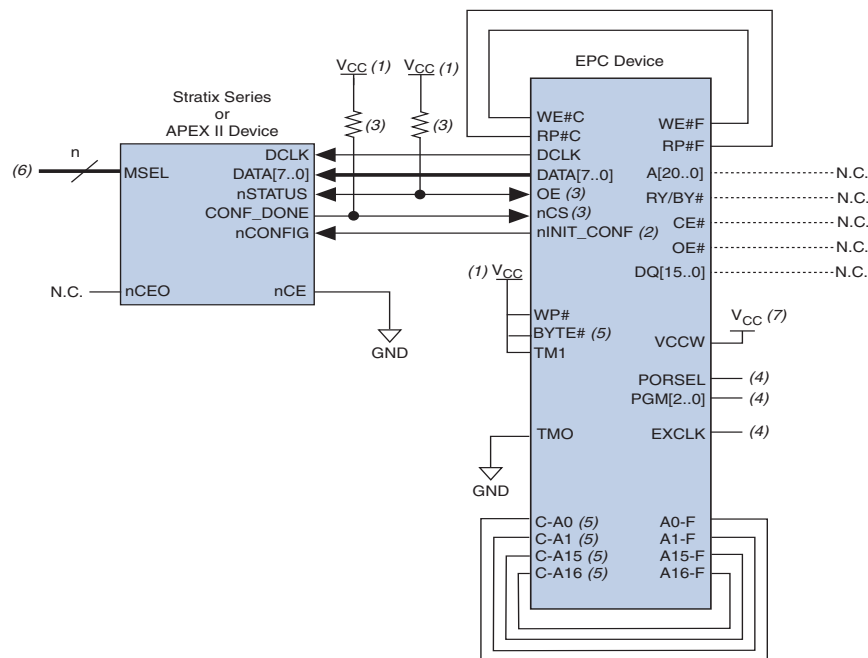
Notes to Figure 2:

(1) The V_{CC} should be connected to the same supply voltage as the EPC device.

(2) The nINIT_CONF pin is available on EPC 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 or nCONFIG signal. 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 EPC 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 10 on page 24.

(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}, TM0 to GND, and WP# to V_{CC}.

(6) Connect the FPGA MSEL[] input pins to select the FPP configuration mode. For more information, refer to the configuration chapter in the appropriate device handbook.

(7) To protect Intel Flash-based EPC devices content, isolate the V_{CCW} supply from V_{CC}. For more information, refer to "Intel Flash-Based EPC Device Protection" on page 16.

Multiple FPGAs can be configured using a single EPC 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 *n*CEO pin asserts to activate the *n*CE 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 EPC device or one of the FPGAs detects an error, configuration stops (and simultaneously restarts) for the whole chain because the *n*STATUS pins are tied together.

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

👣 For more information about configuration schematics and multi-device FPP configuration, refer to the configuration chapter in the appropriate device handbook.

Passive Serial Configuration

APEX 20KC, APEX 20KE, APEX 20K, APEX II, Cyclone series, FLEX 10K, and Stratix series devices can be configured using EPC 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 driven 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 EPC device connects to the FPGA DATA0 input pin and the remaining DATA[7..1] pins are left floating.

👣 For more information about configuration schematics and multi-device PS configuration, refer to the configuration chapter in the appropriate device handbook.

Concurrent Configuration

EPC devices support concurrent configuration of multiple FPGAs (or FPGA chains) in PS mode. Concurrent configuration is when the EPC device simultaneously outputs *n* bits of configuration data on the DATA[*n*-1..0] pins (*n* = 1, 2, 4, or 8), and each DATA[] line serially configures a different FPGA chain. The number of concurrent serial chains is user-defined using the Quartus II software and can be any number from 1 to 8. For example, for 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.

👣 For more information about configuration interface connections including pull-up resistor values, supply voltages, and MSEL pin settings, refer to the configuration chapter in the appropriate device handbook.

Figure 3 shows the schematic for configuring multiple FPGAs concurrently in the PS mode using an EPC device.

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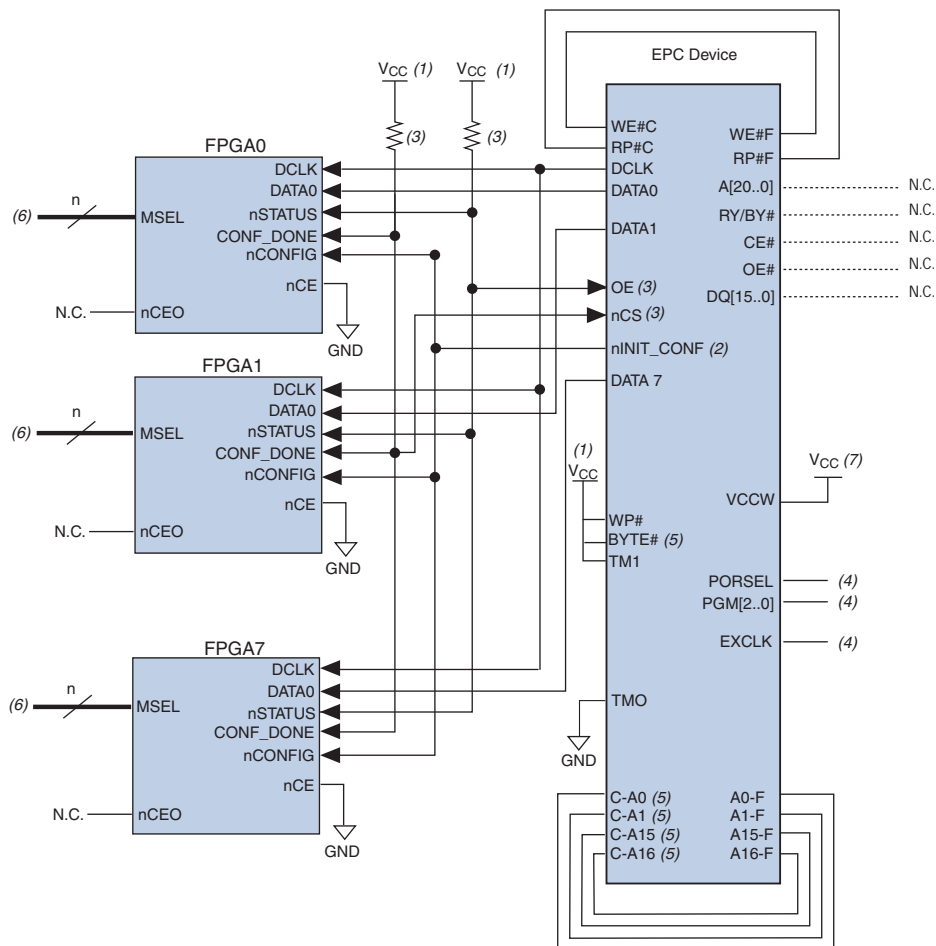
Notes to Figure 3:

(1) Connect V_{CC} to the same supply voltage as the EPC device.

(2) The nINIT_CONF pin is available on EPC 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 or nCONFIG signal. 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 EPC 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 10 on page 24.

(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}, TM0 to GND, and WP# to V_{CC}.

(6) Connect the FPGA MSEL[] input pins to select the PS configuration mode. For more information, refer to the configuration chapter in the appropriate device handbook.

(7) To protect Intel Flash based EPC devices content, isolate the V_{CCW} supply from V_{CC}. For more information, refer to "Intel Flash-Based EPC Device Protection" on page 16.

Table 5 lists the concurrent PS configuration modes supported in the EPC device.

Table 5. EPC Devices in PS Mode

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

Note to Table 5:

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

For more information about configuration schematics and concurrent configurations, refer to the configuration chapter in the appropriate device handbook.

External Flash Interface

The EPC 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 using 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 EPC device controller and 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 EPC devices because this causes contention during ISP and configuration. During these operations, the controller chips inside the EPC devices are actively accessing flash memory. Therefore, EPC devices do not support shared flash bus interfaces.

The EPC 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 EPC 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 using the external flash interface, the external device can hold the FPGA's *n*CONFIG input low. This keeps the configuration device in reset by holding the *n*STATUS-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*.

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

Figure 4. FPP Configuration with External Flash Interface [1]
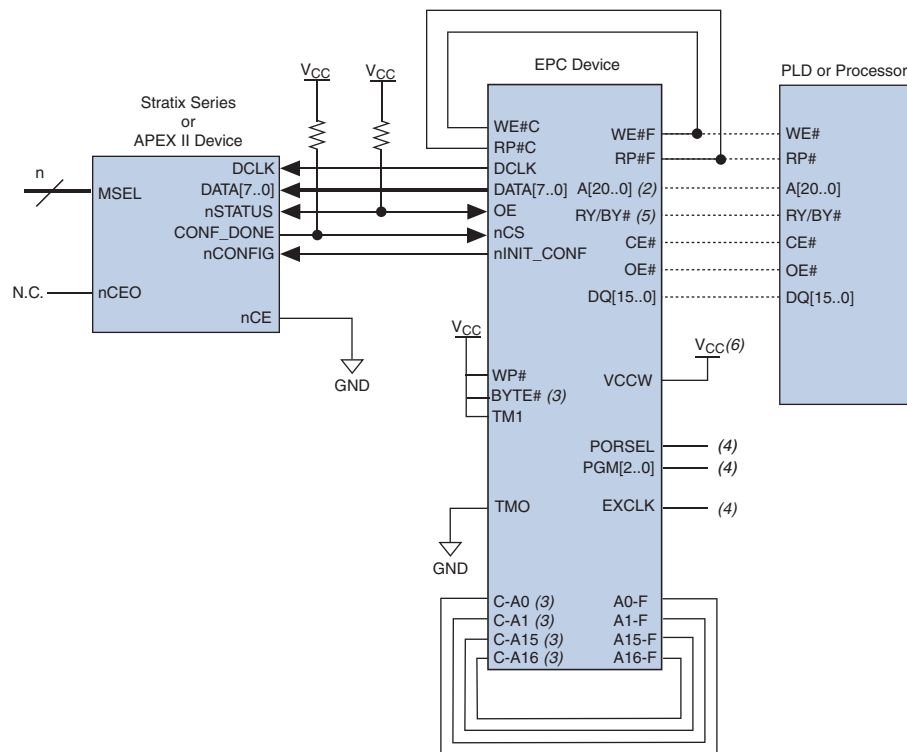
Notes to Figure 4:

(1) For external flash interface support in the EPC8 device, contact Altera Applications 24/7 Technical Support.

(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 as they are NC 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 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}, TM0 to GND, and WP# to V_{CC}.

(4) For PORSEL, PGM[], and EXCLK pin connections, refer to Table 10 on page 24.

(5) RY/BY# pin is only available for Sharp flash-based EPC8 and EPC16 devices.

(6) To protect Intel Flash based EPC devices content, isolate the V_{CCW} supply from V_{CC}. For more information, refer to "Intel Flash-Based EPC Device Protection" on page 16.

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 VCCW 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 following lists the ideal power-up sequence:

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 following lists the ideal power-down sequence:

1. Drive RP# low for 100 ns 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 connection. 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 connection, 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 EPC 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 EPC 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 EPC 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 the *Remote System Configuration with Stratix & Stratix GX Devices* chapter 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 *n*CONFIG. Page 0 is defined as the default page and the PGM[2] pin is the MSB.

☞ The PGM[2..0] input pins must not be left floating on your board. When you are not using this feature, connect the PGM[2..0] pins to GND to select the default page 000.

The EPC 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 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 EPC 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 EPC 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 EPC device is only limited by the size of the flash memory.

For more information about the page-mode feature implementation and programming file generation steps using the Quartus II software, refer to the *Altera Enhanced Configuration Devices*.

Real-Time Decompression

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

The EPC 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 bits × 66 MHz = 528 Mbps. The flash read interface, however, is limited to approximately 10 MHz (since the flash access time is ~90 ns). This translates to a flash-read bandwidth of 16 bits × 10 MHz = 160 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 bits × 10 MHz = 300 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 EPC 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 EPC 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 or 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 ~47% size reduction for these designs. Table 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 6. Stratix Compression Ratios [1]

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

Note to Table 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%). Figure 5 shows a block diagram of the clock divider unit.

Figure 5. Clock Divider Unit
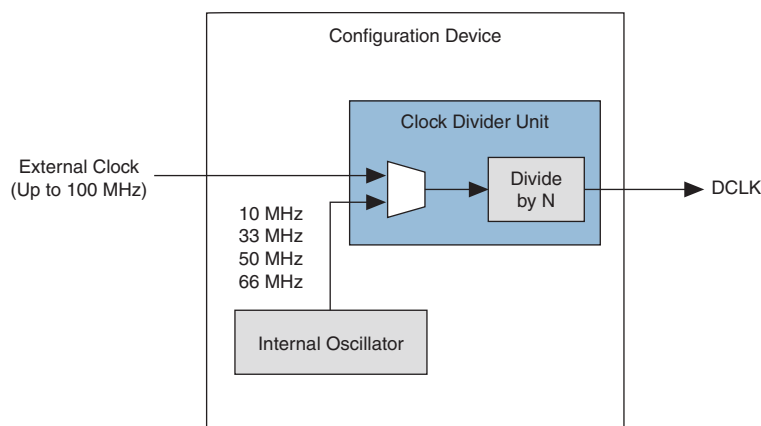
The DCLK frequency is limited by the maximum DCLK frequency the FPGA supports.

For more information about the maximum DCLK input frequency supported by the FPGA, refer to the configuration chapter in the appropriate device 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. Table 7 lists the internal oscillator frequencies.

Table 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*.

Flash In-System Programming (ISP)

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

☞ The EPC device flash memory supports 100,000 erase cycles.

JTAG-based Programming

The IEEE Std. 1149.1 JTAG Boundary Scan is implemented in EPC devices to facilitate the testing of its interconnection and functionality. EPC devices also support the ISP mode. The EPC 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. EPC devices support the 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 (using 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.

You can add an initiate configuration (*nINIT_CONF*) JTAG instruction 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 using 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 using a communication link such as UART, Ethernet, and PCI. In addition to the program, erase, and verify operations, the external flash interface supports block or sector protection instructions.

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 *n*CONFIG line at a logic low level. This keeps the controller in reset by holding the *n*STATUS-OE line low, allowing external flash access.

☞   If initial programming of the EPC device is done in-system using the external flash interface, the controller must be kept in reset by driving the FPGA *n*CONFIG line low to prevent contention on the flash interface.

Pin Description

Table 8 through Table 10 list the EPC device pins. These tables include configuration interface pins, external flash interface pins, JTAG interface pins, and other pins.

Table 8. Configuration Interface Pins

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 EPC device serves as the FPGA configuration clock. DATA is latched by the FPGA on the rising edge of DCLK.
nCS	Input	The nCS pin is an input to the EPC 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 6 KΩ that can be disabled or enabled in the Quartus II software through the **Disable nCS and OE pull-ups on configuration device** 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 EPC device using a private JTAG instruction. This pin contains an internal weak pull-up resistor of 6K Ω 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 EPC 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 6 KΩ that can be disabled or enabled in the Quartus II software through the **Disable nCS and OE pull-ups on configuration device** option.

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

Pin Name	Pin Type	Description
A[20..0]	Input	These pins are the address input to the flash memory for read and write operations. The addresses are internally latched during a write cycle. 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. In the 100-pin PQFP package, four address pins (A0, A1, A15, A16) are not internally connected to the controller. These loop-back connections must be made on the board between the C-A[] and F-A[] pins even when you are not using the external flash interface. All other address pins are connected internal to the package. All address pins are connected internally in the 88-pin UFBGA package. Pin A20 in EPC16 devices, pins A20 and A19 in EPC8 devices, and pins A20, A19, and A18 in EPC4 devices are NC pins. These pins should be left floating on the board.
DQ[15..0]	Bidirectional	This is the flash data bus interface between the flash memory and the controller. The controller or an external source drives DQ[15..0] during the flash command and the data write bus cycles. During the data read cycle, the flash memory drives the DQ[15..0] to the controller or external device. Leave these pins floating on the board when the external flash interface is not used.
CE#	Input	Active low flash input pin that activates the flash memory when asserted. When it is high, it deselects the device and reduces power consumption to standby levels. This flash input pin is internally connected to the controller. Leave this pin floating on the board when the external flash interface is not used.
RP# [1]	Input	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. 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. 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.
OE#	Input	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. Leave this pin floating on the board when the external flash interface is not used.
WE# [1]	Input	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. 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. When using the external flash interface, connect the external device to the WE# pin with the loop back.

Table 9. External Flash Interface Pins (Part 2 of 2)

Pin Name	Pin Type	Description
WP#	Input	Usually tied to V_{CC} or GND on the board. The controller does not drive this pin because it could cause contention. Connection to V_{CC} is recommended for faster block erase or programming times and to allow programming of the flash-bottom boot block, which is required when programming the device using the Quartus II software. This pin should be connected to V_{CC} even when the external flash interface is not used.
V_{CCW}	Supply	Block erase, full-chip erase, word write, or lock-bit configuration power supply. Connect this pin to the 3.3-V V_{CC} supply, even when you are not using the external flash interface.
RY/BY#	Open-Drain Output	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)* Leave this pin floating when the external flash interface is not used.
BYTE#	Input	Flash byte-enable pin and is only available for EPC 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, BYTE# must be connected directly to V_{CC} without using any pull-up resistor.

Notes to Table 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 PCB, connect the C-WE# pin from controller to F-WE# pin from the flash memory).

(2) For more information, refer to the *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 10. JTAG Interface Pins and Other Required Controller Pins   (Part 1 of 2)

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 this pin 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 EPC device POF. PGM[2] is the MSB. The default selection is page 0; PGM[2..0] =000. These pins must not be left floating.

Table 10. JTAG Interface Pins and Other Required Controller Pins (Part 2 of 2)

Pin Name	Pin Type	Description
EXCLK	Input	Optional external clock input pin that can be used to generate the configuration clock (DCLK). 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 EXCLK is used, toggling the EXCLK input pin after the FPGA enters user mode will not effect the EPC device operation.
PORSEL	Input	This pin selects a 2-ms or 100-ms POR counter delay during power up. When PORSEL is low, POR time is 100 ms. When PORSEL is high, POR time is 2 ms. This pin must be connected to a valid logic level.
TM0	Input	For normal operation, this test pin must be connected to GND.
TM1	Input	For normal operation, 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 EPC 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 EPC device enters 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 *n*STATUS low, which occurs if the FPGA detects a CRC error or if the FPGA's *n*CONFIG input pin is asserted

■ The controller detects a configuration error and asserts OE to begin reconfiguration 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 EPC device's POR expires.

Power up needs to be controlled so that the EPC device's OE signal goes high after the CONF_DONE signal is pulled low. If the EPC 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 EPC device exits POR, OE is released and pulled high by a pull-up resistor. Since the EPC 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 EPC device will only exit this mode if it is powered down and then powered up correctly.

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

The pin-selectable POR time feature is useful for ensuring this power-up sequence. The EPC 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 EPC device and automatically generates the **.pof** for the EPC4, EPC8, and EPC16 devices. In a multi-device project, the Quartus II 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 EPC device.

For more information about generating programming files, refer to the *Altera Enhanced Configuration Devices*.

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

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

The ISP circuitry in the EPC 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 11. JTAG Instructions for EPC Devices [1]

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

Note to Table 11:

(1) Instruction register length for the EPC device is 10 and boundary scan length is 174.

For more information about the EPC device JTAG support, refer to the Configuration Devices BSDL Files page.

EPC 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 a Hexadecimal (Intel-Format) File (**.hexout**) using the Quartus II **Convert Programming Files** utility, for use with the programmers or processors.

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

Table 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 EPC device provides JTAG BST circuitry that complies with the IEEE Std. 1149.1-1990 specification. JTAG BST can be performed before or after configuration, but not during configuration.

Figure 6 shows the timing requirements for the JTAG signals.

Figure 6. JTAG Timing Waveforms
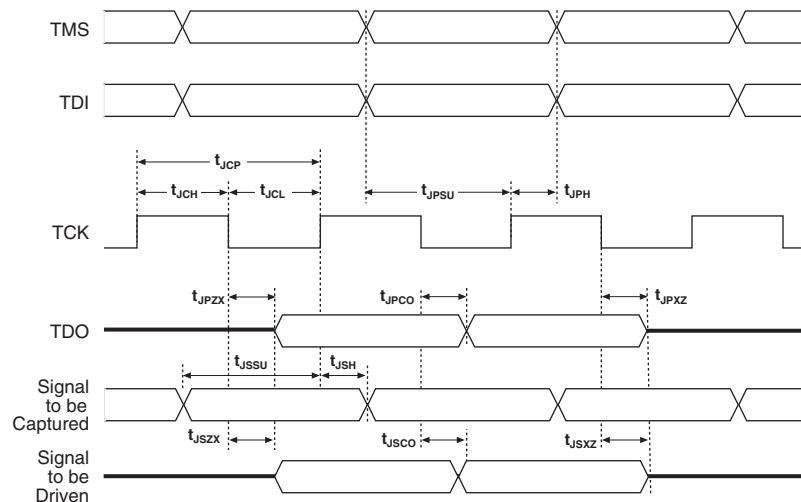
Table 13 lists the timing parameters and values for the EPC device.

Table 13. JTAG Timing Parameters and Values   (Part 1 of 2)

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

Table 13. JTAG Timing Parameters and Values (Part 2 of 2)

Symbol	Parameter	Min	Max	Unit
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 7 shows the configuration timing waveform when you are using an EPC device.

Figure 7. Configuration Timing Waveform Using an EPC Device
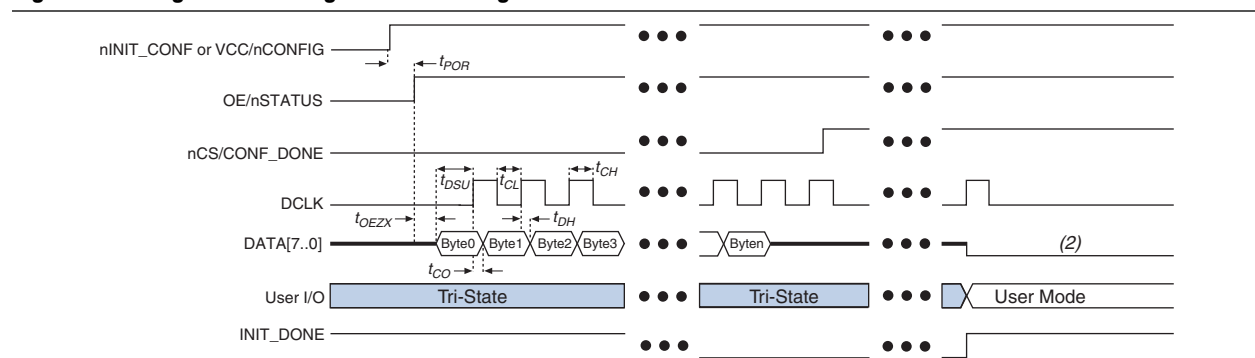
Notes to Figure 7:

(1) The EPC device drives `DCLK` low after configuration.

(2) The EPC device drives `DATA[]` high after configuration.

Table 14 lists the timing parameters when you are using the EPC devices.

Table 14. EPC Device Configuration Parameters (Part 1 of 2)

Symbol	Parameter	Condition	Min	Typ	Max	Unit
f_{DCLK}	`DCLK` frequency	40% duty cycle	—	—	66.7	MHz
t_{DCLK}	`DCLK` period	—	15	—	—	ns
t_{HC}	`DCLK` duty cycle high time	40% duty cycle	6	—	—	ns
t_{LC}	`DCLK` duty cycle low time	40% duty cycle	6	—	—	ns
t_{CE}	`OE` to first `DCLK` delay	—	40	—	—	ns
t_{OE}	`OE` to first `DATA` available	—	40	—	—	ns
t_{OH}	`DCLK` rising edge to `DATA` change	—	[1]	—	—	ns
t_{CF} [2]	`OE` assert to `DCLK` disable delay	—	277	—	—	ns
t_{DF} [2]	`OE` assert to `DATA` disable delay	—	277	—	—	ns
t_{RE} [3]	`DCLK` rising edge to `OE`	—	60	—	—	ns
t_{LOE}	`OE` assert time to assure reset	—	60	—	—	ns
f_{ECLK}	`EXCLK` input frequency	40% duty cycle	—	—	100	MHz

Table 14. EPC Device Configuration Parameters (Part 2 of 2)

Symbol	Parameter	Condition	Min	Typ	Max	Unit
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 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 EPC 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 15 through Table 19 list information about absolute maximum ratings, recommended operating conditions, DC operating conditions, supply current values, and pin capacitance data for the EPC devices.

Table 15. Absolute Maximum Rating for EPC Devices

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 16. Recommended Operating Conditions for EPC Devices

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
		For military use [1]	−55	125	C
T_R	Input rise time	—	—	20	ns
T_F	Input fall time	—	—	20	ns

Note to Table 16:

(1) Applicable for UBGA88 package of the EPC16 device only.

Table 17. DC Operating Conditions for EPC Devices

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	μA
I_{OZ}	Tri-state output off-state current	$V_O = V_{CC}$ or ground	−10	—	10	μA

Table 18. I_{CC} Supply Current Values for EPC Devices

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

Note to Table 18:

(1)   For the V_{CCW} supply current information, refer to the appropriate flash memory data sheet at **www.altera.com**.

Table 19. Capacitance for EPC Devices

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

Package

The EPC16 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.

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

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

**Figure 8. PCB Routing for 88-Pin UFBGA Package** [1]
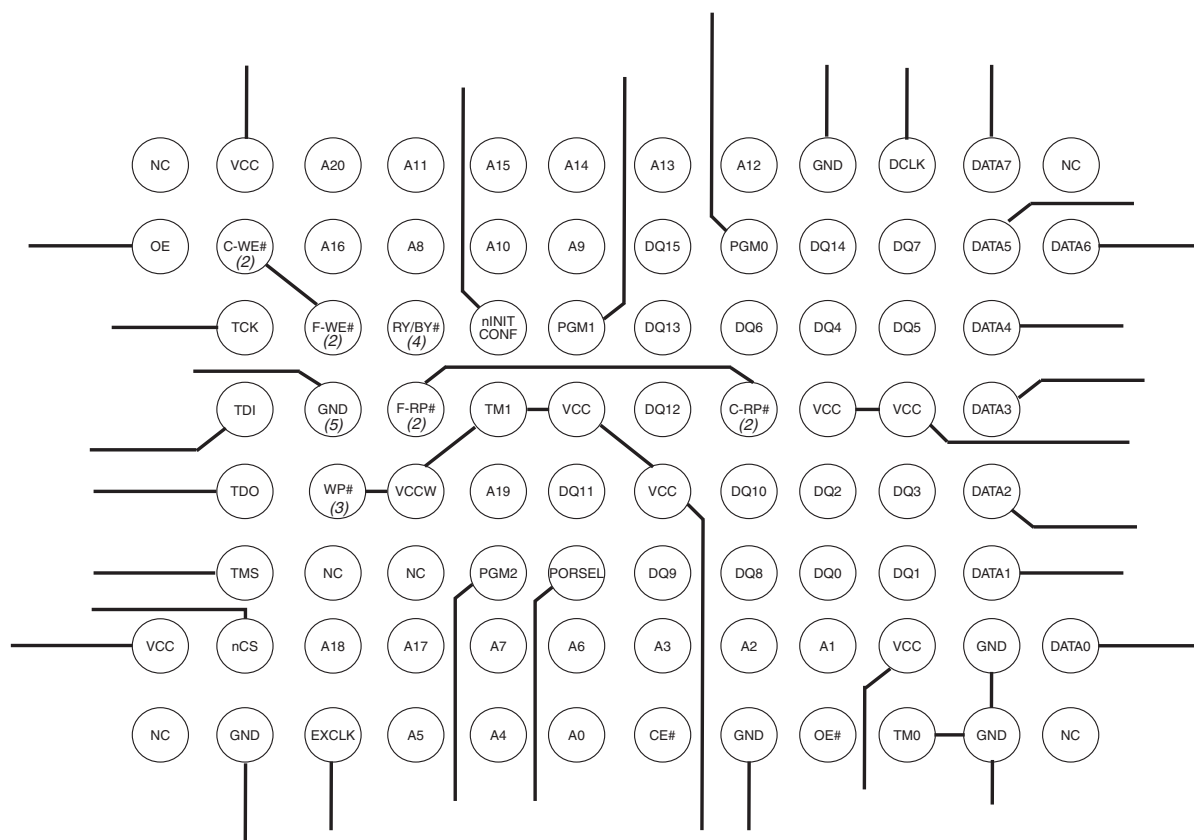
Notes to Figure 8:

(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 EPC devices.

(5) Pin D3 is a NC pin for Intel Flash-based EPC16.

Package Layout Recommendation

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

Figure 9. EPC Device PCB Footprint Specifications for 100-Pin PQFP Packages [1], [2]
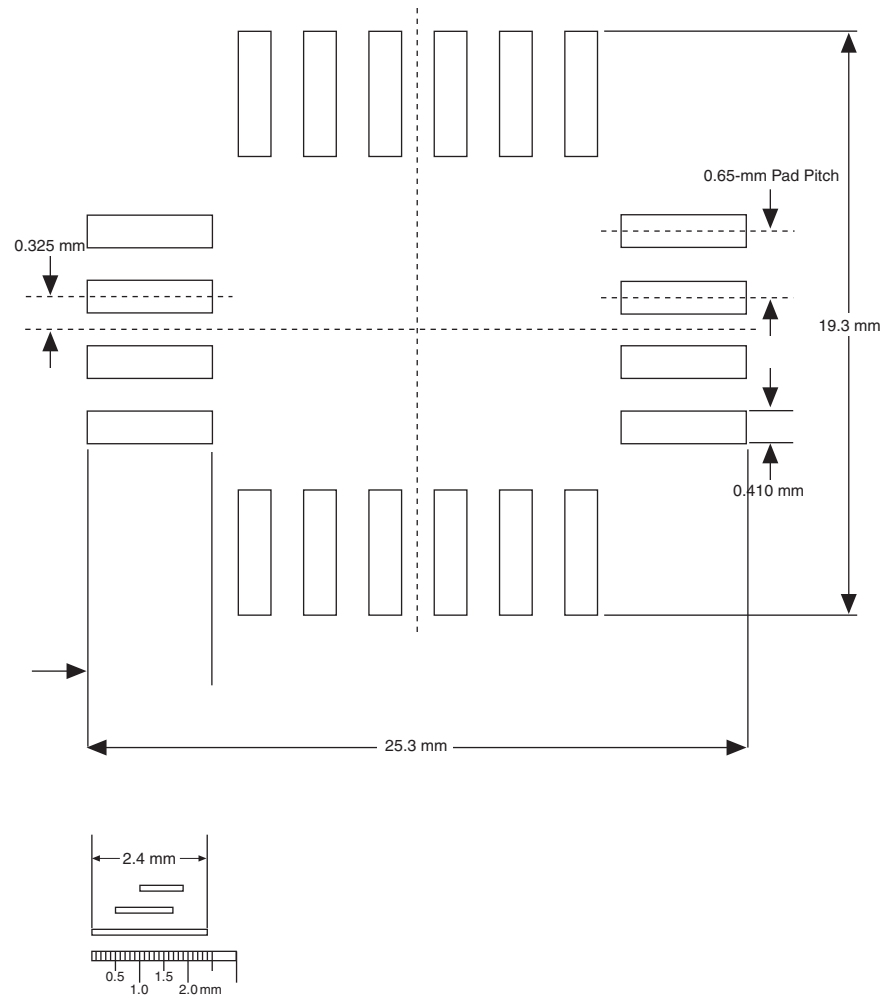
Notes to Figure 9:

(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 more information about package outline drawings, refer to the Package and Thermal Resistance page.

Device Pin-Outs

For more information, refer to the Configuration Devices Pin-Out Files page.

Document Revision History

Table 20 lists the revision history for this document.

Table 20. Document Revision History

Date	Version	Changes
January 2012	3.0	Minor text edits.
June 2011	2.9	Updated Table 1–3 and Table 1–16.
December 2009	2.8	■ Added Table 1–1 and Table 1–2. ■ Updated Table 1–17 and Table 1–18. ■ Removed "Referenced Documents" section.
October 2008	2.7	■ Updated Table 2–1, Table 2–7, and Table 2–8. ■ Updated Figure 2–2, Figure 2–3, and Figure 2–4. ■ Updated "JTAG-based Programming" section. ■ Added "Intel-Flash-Based EPC Device Protection" section. ■ Updated new document format.
May 2008	2.6	Minor textual and style changes. Added "Referenced Documents" section.
February 2008	2.5	Updated Table 2–18 with information about EPC16UI88AA.
May 2007	2.4	Added "Intel-Flash-Based EPC Device Protection" section.
April 2007	2.3	Added document revision history.
October 2005	2.2	Made changes to content.
July 2004	2.0	■ Added Stratix II and Cyclone II device information throughout chapter. ■ Updated VCCW connection in Figure 2–2, Figure 2–3, and Figure 2–4. ■ Updated (Note 2) of Figure 2–2, Figure 2–3, and Figure 2–4. ■ Updated (Note 4) of Table 2–12. ■ Updated unit of ICC0 in Table 2–16. ■ Added ICCW to Table 2–16.
September 2003	1.0	Initial Release.