

Dedicated circuitry built into the Cyclone® III device family (Cyclone III and Cyclone III LS devices) consists of a cyclical redundancy check (CRC) error detection feature that can optionally check for a single-event upset (SEU) continuously and automatically.

In critical applications used in the fields of avionics, telecommunications, system control, medical, and military applications, it is important to be able to:

- Confirm the accuracy of the configuration data stored in an FPGA device
- Alert the system to an occurrence of a configuration error

This chapter describes how to activate and use the error detection CRC feature in user mode and describes how to recover from configuration errors caused by CRC error. Using the CRC error detection feature for Cyclone III device family does not impact fitting or performance.

This chapter contains the following sections:

- “Error Detection Fundamentals” on page 11-1
- “Configuration Error Detection” on page 11-2
- “User Mode Error Detection” on page 11-2
- “Automated SEU Detection” on page 11-3
- “CRC_ERROR Pin” on page 11-3
- “Table 11-2 lists the CRC_ERROR pin.” on page 11-4
- “Error Detection Block” on page 11-4
- “Error Detection Timing” on page 11-5
- “Software Support” on page 11-7
- “Recovering from CRC Errors” on page 11-10

Error Detection Fundamentals

Error detection determines if the data received through an input device is corrupted during transmission. In validating the data, the transmitter uses a function to calculate a checksum value for the data and appends the checksum to the original data frame. The receiver uses the same calculation methodology to generate a checksum for the received data frame and compares the received checksum to the transmitted checksum. If the two checksum values are equal, the received data frame is correct and no data corruption has occurred during transmission or storage.

The error detection CRC feature in Cyclone III device family puts theory into practice. In user mode, the error detection CRC feature in Cyclone III device family ensures the integrity of the configuration data.

Configuration Error Detection

In configuration mode, a frame-based CRC is stored in the configuration data and contains the CRC value for each data frame.

During configuration, Cyclone III device family calculates the CRC value based on the frame of data that is received and compares it against the frame CRC value in the data stream. Configuration continues until either the device detects an error or all the values are calculated.

For Cyclone III device family, the CRC is computed by the Quartus® II software and downloaded into the device as part of the configuration bit stream. These devices store the CRC in the 32-bit storage register at the end of the configuration mode.

User Mode Error Detection

Soft errors are changes in a configuration random-access memory (CRAM) bit state due to an ionizing particle. Cyclone III device family has built-in error detection circuitry to detect data corruption by soft errors in the CRAM cells.

This error detection capability continuously computes the CRC of the configured CRAM bits based on the contents of the device and compares it with the pre-calculated CRC value obtained at the end of the configuration. If the CRCs match, there is no error in the current configuration CRAM bits. The process of error detection continues until the device is reset (by setting *nCONFIG* to low).

The Cyclone III device family error detection feature does not check memory blocks and I/O buffers. These device memory blocks support parity bits that are used to check the contents of memory blocks for any error. The I/O buffers are not verified during error detection because the configuration data uses flip-flops as storage elements that are more resistant to soft errors. Similar flip-flops are used to store the pre-calculated CRC and other error detection circuitry option bits.

The error detection circuitry in Cyclone III device family uses a 32-bit CRC IEEE 802 standard and a 32-bit polynomial as the CRC generator. Therefore, a single 32-bit CRC calculation is performed by the device. If a soft error does not occur, the resulting 32-bit signature value is 0x000000, which results in a 0 on the output signal *CRC_ERROR*. If a soft error occurs in the device, the resulting signature value is non-zero and the *CRC_ERROR* output signal is 1.

You can inject a soft error by changing the 32-bit CRC storage register in the CRC circuitry. After verifying the failure induced, you can restore the 32-bit CRC value to the correct CRC value using the same instruction and inserting the correct value.



Be sure to read out the correct value before updating it with a known bad value.

In user mode, Cyclone III device family supports the `CHANGE_EDREG` JTAG instruction, which allows you to write to the 32-bit storage register. You can use Jam™ STAPL files (`.jam`) to automate the testing and verification process. This instruction can only be executed when the device is in user mode, and it is a powerful design feature that enables you to dynamically verify the CRC functionality in-system without having to reconfigure the device. You can then switch to use the CRC circuit to check for real errors induced by an SEU.

Table 11-1 lists the `CHANGE_EDREG` JTAG instructions.

Table 11-1. CHANGE_EDREG JTAG Instruction

JTAG Instruction	Instruction Code	Description
<code>CHANGE_EDREG</code>	00 0001 0101	This instruction connects the 32-bit CRC storage register between TDI and TDO. Any precomputed CRC is loaded into the CRC storage register to test the operation of the error detection CRC circuitry at the <code>CRC_ERROR</code> pin.



After the test completes, to clear the CRC error and restore the original CRC value, power cycle the device or perform the following procedure:

1. After the configuration completes, use JTAG instruction `CHANGE_EDREG` to shift out the correct precomputed CRC value and load the wrong CRC value to the CRC storage register. The `CRC_ERROR` pin will be asserted and shows that a CRC error is detected.
2. Use JTAG instruction `CHANGE_EDREG` to shift in the correct precomputed CRC value. The `CRC_ERROR` pin is deasserted and shows that the error detection CRC circuitry is working.

Automated SEU Detection

Cyclone III device family offers on-chip circuitry for automated checking of SEU detection. Applications that require the device to operate error-free at high elevations or in close proximity to earth's North or South Pole require periodic checks to ensure continued data integrity. The error detection cyclic redundancy code feature controlled by the **Device and Pin Options** dialog box in the Quartus II software uses a 32-bit CRC circuit to ensure data reliability and is one of the best options for mitigating SEU.

You can implement the error detection CRC feature with existing circuitry in Cyclone III device family, eliminating the need for external logic. The CRC is computed by the device during configuration and checked against an automatically computed CRC during normal operation. The `CRC_ERROR` pin reports a soft error when configuration CRAM data is corrupted, and you must decide whether to reconfigure the FPGA by strobing the `nCONFIG` pin low or ignore the error.


CRC_ERROR Pin


A specific error detection pin, `CRC_ERROR`, is required to monitor the results of the error detection circuitry during user mode.

Table 11-2 lists the CRC_ERROR pin.

Table 11-2. CRC_ERROR Pin Description

Device	CRC_ERROR Pin Type	Description
Cyclone III	Dedicated Output or Open Drain Output (Optional)	By default, the Quartus II software sets the CRC_ERROR pin as a dedicated output. If the CRC_ERROR pin is used as a dedicated output, you must ensure that the V_{CCIO} of the bank in which the pin resides meets the input voltage specification of the system receiving the signal. Optionally, you can set this pin to be an open-drain output by enabling the option in the Quartus II software from the Error Detection CRC tab of the Device and Pin Options dialog box. Using the pin as an open-drain provides an advantage on the voltage leveling. To use this pin as open-drain, you can tie this pin to V_{CCIO} of Bank 1 through a 10-k Ω pull-resistor. Alternatively, depending on the voltage input specification of the system receiving the signal, you can tie the pull-up resistor to a different pull-up voltage.
Cyclone III LS	Open Drain Output	To use the CRC_ERROR pin, you can either tie this pin to V_{CCIO} through a 10-k Ω pull-up resistor, or depending on input voltage specification of the system receiving the signal, you can tie this pin to a different pull-up voltage.

 For more information about the CRC_ERROR pin information for Cyclone III device family, refer to the Cyclone III [Pin-Out Files for Altera Devices](#) page on the Altera® website.

 WYSIWYG is an optimization technique that performs optimization on VQM (Verilog Quartus Mapping) netlist in the Quartus II software.

Error Detection Block

Table 11-3 lists the types of CRC detection to check the configuration bits.

Table 11-3. Types of CRC Detection to Check the Configuration Bits

First Type of CRC Detection	Second Type of CRC Detection
<ul style="list-style-type: none"> ■ CRAM error checking ability (32-bit CRC) during user mode, for use by the CRC_ERROR pin. ■ There is only one 32-bit CRC value, and this value covers all the CRAM data. 	<ul style="list-style-type: none"> ■ 16-bit CRC embedded in every configuration data frame. ■ During configuration, after a frame of data is loaded into the device, the pre-computed CRC is shifted into the CRC circuitry. ■ Simultaneously, the CRC value for the data frame shifted-in is calculated. If the pre-computed CRC and calculated CRC values do not match, nSTATUS is set low. ■ Every data frame has a 16-bit CRC. Therefore, there are many 16-bit CRC values for the whole configuration bit stream. ■ Every device has a different length of configuration data frame.

This section focuses on the first type—the 32-bit CRC when the device is in user mode.

Error Detection Registers

There are two sets of 32-bit registers in the error detection circuitry that store the computed CRC signature and pre-calculated CRC value. A non-zero value on the signature register causes the CRC_ERROR pin to set high.

Figure 11-1 shows the block diagram of the error detection block and the two related 32-bit registers: the signature register and the storage register.

Figure 11-1. Error Detection Block Diagram

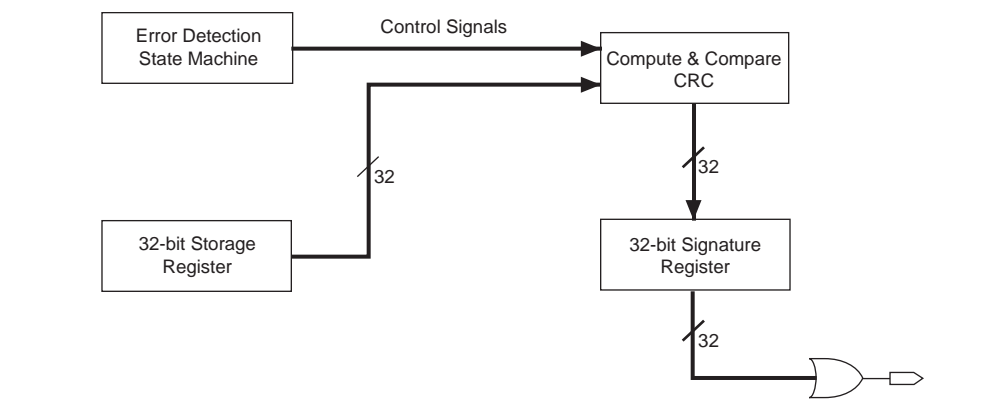


Table 11-4 lists the registers shown in Figure 11-1.

Table 11-4. Error Detection Registers

Register	Function
32-bit signature register	<p>This register contains the CRC signature. The signature register contains the result of the user mode calculated CRC value compared against the pre-calculated CRC value. If no errors are detected, the signature register is all zeros. A non-zero signature register indicates an error in the configuration CRAM contents.</p> <p>The <code>CRC_ERROR</code> signal is derived from the contents of this register.</p>
32-bit storage register	<p>This register is loaded with the 32-bit pre-computed CRC signature at the end of the configuration stage. The signature is then loaded into the 32-bit CRC circuit (called the Compute and Compare CRC block, as shown in Figure 11-1) during user mode to calculate the CRC error. This register forms a 32-bit scan chain during execution of the <code>CHANGE_EDREG</code> JTAG instruction. The <code>CHANGE_EDREG</code> JTAG instruction can change the content of the storage register. Therefore, the functionality of the error detection CRC circuitry is checked in-system by executing the instruction to inject an error during the operation. The operation of the device is not halted when issuing the <code>CHANGE_EDREG</code> instruction.</p>

Error Detection Timing

When the error detection CRC feature is enabled through the Quartus II software, the device automatically activates the CRC process upon entering user mode after configuration and initialization is complete.

The `CRC_ERROR` pin is driven low until the error detection circuitry has detected a corrupted bit in the previous CRC calculation. After the pin goes high, it remains high during the next CRC calculation. This pin does not log the previous CRC calculation. If the new CRC calculation does not contain any corrupted bits, the `CRC_ERROR` pin is driven low. The error detection runs until the device is reset.

The error detection circuitry runs off an internal configuration oscillator with a divisor that sets the maximum frequency.

Table 11-5 lists the minimum and maximum error detection frequencies.

Table 11-5. Minimum and Maximum Error Detection Frequencies

Device Type	Error Detection Frequency	Maximum Error Detection Frequency	Minimum Error Detection Frequency	Valid Divisors (2 ⁿ)
Cyclone III device family	80 MHz/2 ⁿ	80 MHz	312.5 kHz	0, 1, 2, 3, 4, 5, 6, 7, 8

You can set a lower clock frequency by specifying a division factor in the Quartus II software (for more information, refer to “Software Support” on page 11-7). The divisor is a power of two (2), where *n* is between 0 and 8. The divisor ranges from one through 256. Refer to Equation 11-1.

Equation 11-1. Error Detection Frequency

$$\text{Error detection frequency} = \frac{80 \text{ MHz}}{2^n}$$

CRC calculation time depends on the device and the error detection clock frequency.

Table 11-6 lists the estimated time for each CRC calculation with minimum and maximum clock frequencies for Cyclone III device family.

Table 11-6. CRC Calculation Time

Device		Minimum Time (ms) ⁽¹⁾	Maximum Time (s) ⁽²⁾
Cyclone III	EP3C5	5	2.29
	EP3C10	5	2.29
	EP3C16	7	3.17
	EP3C25	9	4.51
	EP3C40	15	7.48
	EP3C55	23	11.77
	EP3C80	31	15.81
Cyclone III LS	EP3CLS70	42	21.24
	EP3CLS100	42	21.24
	EP3CLS150	79	40.27
	EP3CLS200	79	40.27

Notes to Table 11-6:

- (1) The minimum time corresponds to the maximum error detection clock frequency and may vary with different processes, voltages, and temperatures (PVT).
- (2) The maximum time corresponds to the minimum error detection clock frequency and may vary with different PVT.

Software Support

Enabling the CRC error detection feature in the Quartus II software generates the CRC_ERROR output to the optional dual purpose CRC_ERROR pin.

To enable the error detection feature using CRC, perform the following steps:

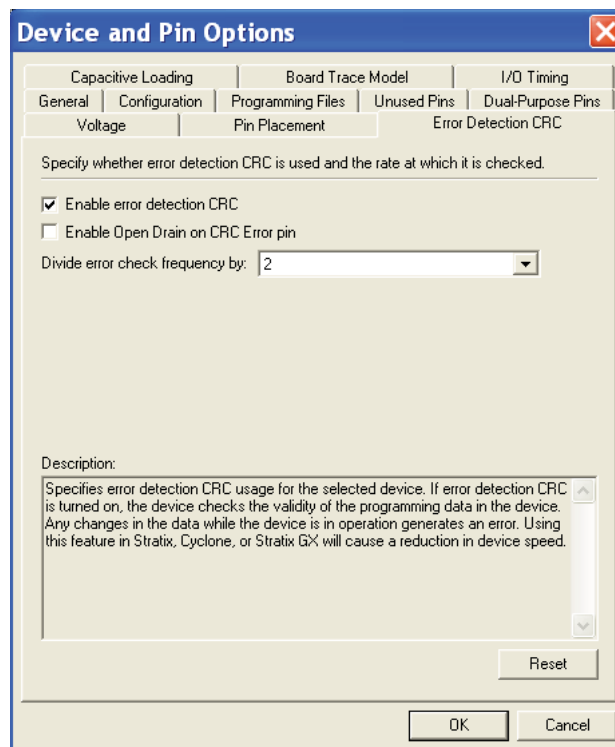
1. Open the Quartus II software and load a project using Cyclone III device family.
2. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
3. In the Category list, select **Device**. The **Device** page appears.
4. Click **Device and Pin Options**, as shown in [Figure 11-2](#).
5. In the **Device and Pin Options** dialog box, click the **Error Detection CRC** tab.
6. Turn on **Enable error detection CRC**.
7. In the **Divide error check frequency by** box, enter a valid divisor as documented in [Table 11-5](#) on page 11-6.




The divisor value divides down the frequency of the configuration oscillator output clock. This output clock is used as the clock source for the error detection process.

8. Click **OK**.

Figure 11-2. Enabling the Error Detection CRC Feature in the Quartus II Software



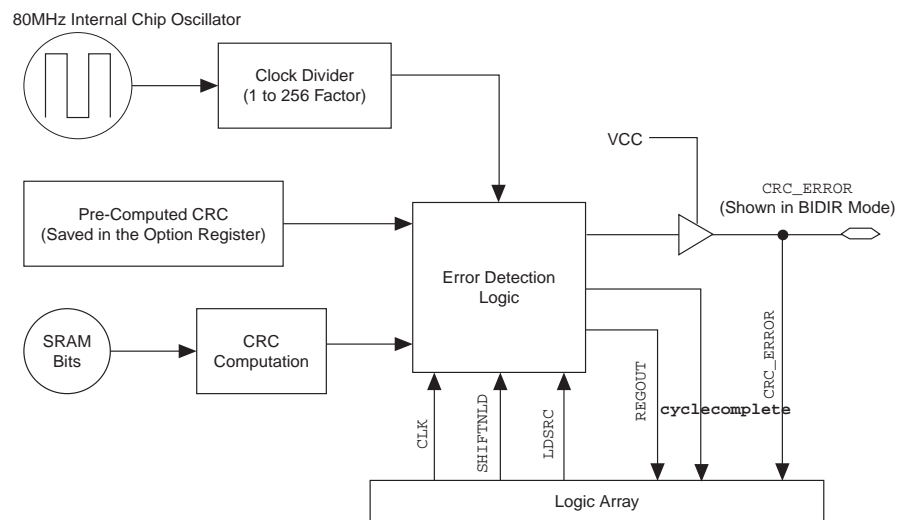
 For Cyclone III LS devices, the “Enable Open Drain on CRC Error Pin” option is not available because the Quartus II software sets the `CRC_ERROR` pin for the Cyclone III LS device as open drain output by default.


Accessing Error Detection Block Through User Logic

The error detection circuit stores the computed 32-bit CRC signature in a 32-bit register. This signature is read out by user logic from the core. The `<device>_crcblock` primitive is a WYSIWYG component used to establish the interface from user logic to the error detection circuit. The `<device>_crcblock` primitive atom contains the input and output ports that must be included in the atom. To access the logic array, the `<device>_crcblock` WYSIWYG atom must be inserted into your design.

Figure 11-3 shows the error detection block diagram in FPGA devices and shows the interface that the WYSIWYG atom enables in your design.

Figure 11-3. Error Detection Block Diagram



 The user logic is affected by the soft error failure, thus reading out the 32-bit CRC signature through the `regout` should not be relied upon to detect a soft error. You should rely on the `CRC_ERROR` output signal itself, because this `CRC_ERROR` output signal cannot be affected by a soft error.

To enable the `<device>_crcblock` WYSIWYG atom, you must name the atom for each Cyclone III device family accordingly.

Table 11-7 lists the name of the WYSIWYG atom for Cyclone III device family.

Table 11-7. WYSIWYG Atoms

Device	WYSIWYG Atom
Cyclone III	cycloneiii_crcblock
Cyclone III LS	cycloneiiiils_crcblock



To enable the `cycloneiii_crcblock` primitive in version 8.0 SP1 or earlier of the Quartus II software, turn on the error detection CRC feature in the **Device and Pins Options** dialog box. This is not required when you are using version 8.1 and later of the Quartus II software.

Example 11-1 shows an example of how to define the input and output ports of a WYSIWYG atom in a Cyclone III LS device.

Example 11-1. Error Detection Block Diagram

```
cycloneiiiils_crcblock<crcblock_name>
(
  .clk(<clock source>),
  .shiftnld(<shiftnld source>),
  .ldsrc(<ldsrc source>),
  .crcerror(<crcerror out destination>),
  .regout(<output destination>),
  .cyclecomplete(<cyclecomplete destination>),
);
```

Table 11-8 lists the input and output ports that must be included in the atom. The input and output ports of the atoms for Cyclone III device family are similar, except for the `cyclecomplete` port which is for Cyclone III LS devices only.

Table 11-8. CRC Block Input and Output Ports (Part 1 of 2)

Port	Input/Output	Definition
<code><crcblock_name></code>	Input	Unique identifier for the CRC block, and represents any identifier name that is legal for the given description language (For example Verilog HDL, VHDL, AHDL). This field is required.
<code>.clk(<clock source></code>	Input	This signal designates the clock input of this cell. All operations of this cell are with respect to the rising edge of the clock. Whether it is the loading of the data into the cell or data out of the cell, it always occurs on the rising edge. This port is required.
<code>.shiftnld (<shiftnld source>)</code>	Input	This signal is an input into the error detection block. If <code>shiftnld=1</code> , the data is shifted from the internal shift register to the <code>regout</code> at each rising edge of <code>clk</code> . If <code>shiftnld=0</code> , the shift register parallel loads either the pre-calculated CRC value or the update register contents depending on the <code>ldsrc</code> port input. This port is required.

Table 11-8. CRC Block Input and Output Ports (Part 2 of 2)

Port	Input/Output	Definition
<code>.ldsrc</code> (<ldsrc source>)	Input	This signal is an input into the error detection block. If <code>ldsrc=0</code> , the pre-computed CRC register is selected for loading into the 32-bit shift register at the rising edge of <code>clk</code> when <code>shiftnld=0</code> . If <code>ldsrc=1</code> , the signature register (result of the CRC calculation) is selected for loading into the shift register at the rising edge of <code>clk</code> when <code>shiftnld=0</code> . This port is ignored when <code>shiftnld=1</code> . This port is required.
<code>.crcerror</code> (<crcerror indicator output>)	Output	This signal is the output of the cell that is synchronized to the internal oscillator of the device (80-MHz internal oscillator) and not to the <code>clk</code> port. It asserts high if the error block detects that a SRAM bit has flipped and the internal CRC computation has shown a difference with respect to the pre-computed value. This signal must be connected either to an output pin or a bidirectional pin. If it is connected to an output pin, you can only monitor the <code>CRC_ERROR</code> pin (the core cannot access this output). If the <code>CRC_ERROR</code> signal is used by core logic to read error detection logic, this signal must be connected to a <code>BIDIR</code> pin. The signal is fed to the core indirectly by feeding a <code>BIDIR</code> pin that has its output enable port connected to <code>VCC</code> (Figure 11-3 on page 11-8).
<code>.regout</code> (<registered output>)	Output	This signal is the output of the error detection shift register synchronized to the <code>clk</code> port, to be read by core logic. It shifts one bit at each cycle, so you should clock the <code>clk</code> signal 31 cycles to read out the 32 bits of the shift register.
<code>.cyclecomplete</code> (<cyclone complete indicator output>)	Output	This signal is for <code>cycloneiils_crcblock</code> only. This output signal is synchronized to the internal oscillator of the device (80-MHz internal oscillator), and not to the <code>clk</code> port. The signal asserts high for one clock cyclone every time an error detection cyclone completes.

Recovering from CRC Errors

The system that the Altera FPGA resides in must control device reconfiguration. After detecting an error on the `CRC_ERROR` pin, strobing the `nCONFIG` low directs the system to perform the reconfiguration at a time when it is safe for the system to reconfigure the FPGA.

When the data bit is rewritten with the correct value by reconfiguring the device, the device functions correctly.

While soft errors are uncommon in Altera devices, certain high-reliability applications might require a design to account for these errors.

Document Revision History

Table 11-9 lists the revision history for this document.

Table 11-9. Document Revision History (Part 1 of 2)

Date	Version	Changes
December 2011	2.3	<ul style="list-style-type: none"> ■ Updated “User Mode Error Detection” on page 11-2. ■ Update hyperlinks. ■ Minor text edits.
December 2009	2.2	Minor changes to the text.

Table 11-9. Document Revision History (Part 2 of 2)

Date	Version	Changes
July 2009	2.1	Made minor correction to the part number.
June 2009	2.0	<ul style="list-style-type: none"> ■ Updated chapter part number. ■ Updated “Introduction” on page 11-1. ■ Updated Table 11-6 on page 11-6 and Table 11-8 on page 11-9. ■ Updated Figure 11-2 on page 11-7. ■ Updated “Accessing Error Detection Block Through User Logic” on page 11-8.
October 2008	1.3	<ul style="list-style-type: none"> ■ Added chapter “Accessing Error Detection Block through User Logic” to document. ■ Updated chapter to new template.
May 2008	1.2	<ul style="list-style-type: none"> ■ Updated Table 11-2.
July 2007	1.1	<ul style="list-style-type: none"> ■ Added chapter TOC to document.
March 2007	1.0	Initial release.

