

This application note presents a design platform that integrates the various Cyclone® III LS security features with an external MAX® II device as a configuration manager.

Cyclone III LS devices extend the Cyclone architecture to include higher density, higher memory, smaller packages, and security features to protect your intellectual property (IP). These FPGAs are the first devices to implement a suite of security features at the silicon, software, and IP level on a low-power, highly-functional device platform.

The suite of security features in Cyclone III LS devices consists of Design Separation and the following Silicon security features:

- Internal oscillator
- 256-bit advanced encryption standard (AES) bitstream encryption
- JTAG port protection
- Zeroization (active clear)
- Cyclical redundancy check (CRC)

The Cyclone III LS device family offers both passive and active resistance against IP theft. AES bitstream encryption and JTAG port protection provide a passive layer of resistance against theft of confidential design files. The CRC error detection feature and an internal oscillator allow you to create an application that actively monitors designs for tamper evidence during operation.

The Cyclone III LS security features allow you to develop a secure configuration manager (an application to initialize the device to a known state) to enable passive protection against theft, and to actively monitor the design to ensure that the target operational environment does not deviate.

The configuration control and management presented in this application note form a design template that provides the standard set of security functions common to most Cyclone III LS designs.


Security services provided by this design example include:

- Unlocking the JTAG port during startup for configuration services
- Programming the contents of the AES encryption key for secure device configuration
- Re-enabling JTAG port protection
- Monitoring the device during run-time

This design example targets a Cyclone III LS FPGA Development Kit; however, you can modify the source code in the design example, which is providing with this application note, to target your specific design requirements.

Overview

Cyclone III LS device features address security at the FPGA boundary by preventing unauthorized access to the configuration bitstream files. FPGA density and complexity growth has enabled system designers to allocate larger functional blocks of the system onto the FPGA. Designs targeted to FPGAs transition from glue logic applications, which connect devices with different interface standards, to co-processing units, which provide various system functions, and to entire systems contained on a single FPGA. Because you can migrate your design functionality to the FPGA boundary, protecting the FPGA from IP theft and counterfeiting is critical. Designs targeting a modern FPGA often contain significant proprietary design information and may process confidential data, which necessitates that the security boundary used to protect the system includes the FPGA.

 For more details about the vulnerabilities of FPGA designs to counterfeiting and reverse-engineering, and the increasing requirements for secure FPGA-based designs, refer to the *Protecting the FPGA Design From Common Threats* white paper.

Elements of a Secure FPGA Design

Four broad requirements are required to create an effective deterrent against counterfeiting and IP theft:

- Resistance—the ability to resist tamper attempts.
- Detection—the ability to make the system or user aware of the tamper event.
- Response—the countermeasure that a system must take after tampering is detected.
- Evidence—the ability for authorized personnel inspecting the system to detect a tampering event.

The system designer is tasked with providing design security by satisfying the four broad requirements with a security service, which is protection against a specific type of threat. To provide an effective deterrent against IP theft, the system designer may provide multiple layers of protection against configuration data theft, or reverse engineering of the configuration image through read-back and unauthorized manipulation of a system during operation.

Cyclone III LS devices support the following security service classes:


- Passive protections
- Active protections
- Application-level protections

Passive protections protect against configuration RAM read-back. In Cyclone III LS devices, passive protections include JTAG port protection and 256-bit AES encryption on the configuration bitstream file. If enabled, these features do not require any additional response from applications running on the system.

For active protections, you can create a controller to monitor status conditions during runtime and respond by logging and correcting an unexpected status condition or clearing out any sensitive information on the FPGA. Input to the controller may include periodic health checks on the Cyclone III LS configuration, temperature, and current sensors on the system. These health checks ensure the operational

environment has not been adversely affected. Cyclone III LS devices contain two device features that you can use as configuration monitors: CRC error detection circuit and an internal oscillator. The CRC error detection circuit monitors the configuration RAM for Single Event Upsets (SEU), but can be repurposed as a mechanism to detect configuration RAM modification, deter reverse-engineering, or detect adverse system operation. You can use the internal oscillator as an independent clock source for system monitoring during runtime.

Application-level protections are additional user-design modules that enforce security policies that protect sensitive data processed by the design. Examples of application-level security policies include authentication of the design running in the system, data access control, and data flow policies between different IP modules.

 For examples of application-level protection against cloning with authentication, refer to the following white papers:

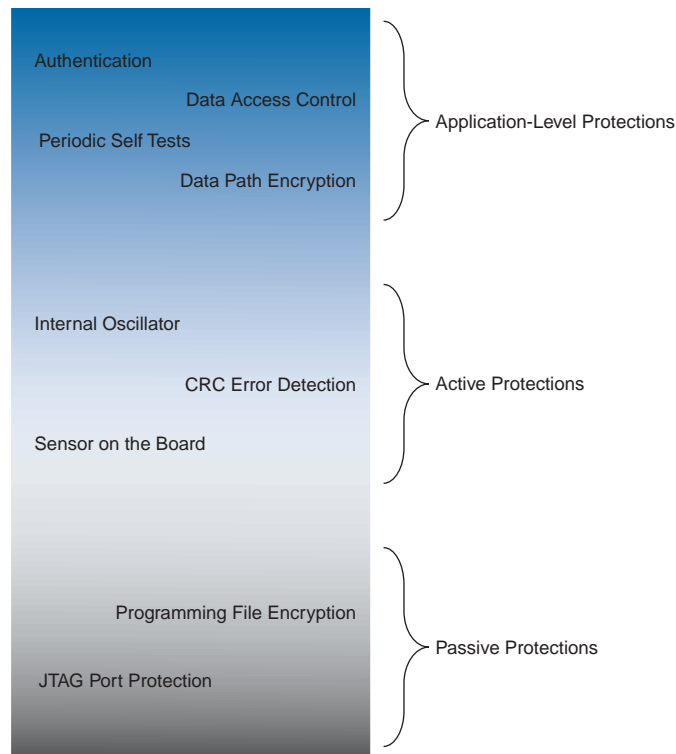
- [*FPGA Design Security Solution Using MAX II Devices*](#)
- [*An FPGA Design Security Solution Using a Secure Memory Device*](#)

The Design Separation feature of Cyclone III LS devices allow you to design effective application-level protections. Cyclone III LS routing architecture is optimized to ensure physical isolation of IP from other partitions in the device. One of the central tenets of any security policy is data access and data flow control. The ability to physically isolate partitions of a system design allows for effective data isolation, protection against data corruption, and the design of better information control policies.

 For more information about the Design Separation flow, refer to [*AN 567: Quartus II Design Separation Flow*](#).

Figure 1 shows a summary of the types of security services.

Figure 1. FPGA Design Security Services



The example presented in this application note enables the minimum set of protections common to most Cyclone III LS designs. The two goals of this anti-tamper design example are:

- Explain the device-specific security services available for Cyclone III LS devices
- Enable the passive protections available for a Cyclone III LS device and encapsulate the security-related functions to actively monitor the device during runtime

Anti-Tamper Design Example Functional Description

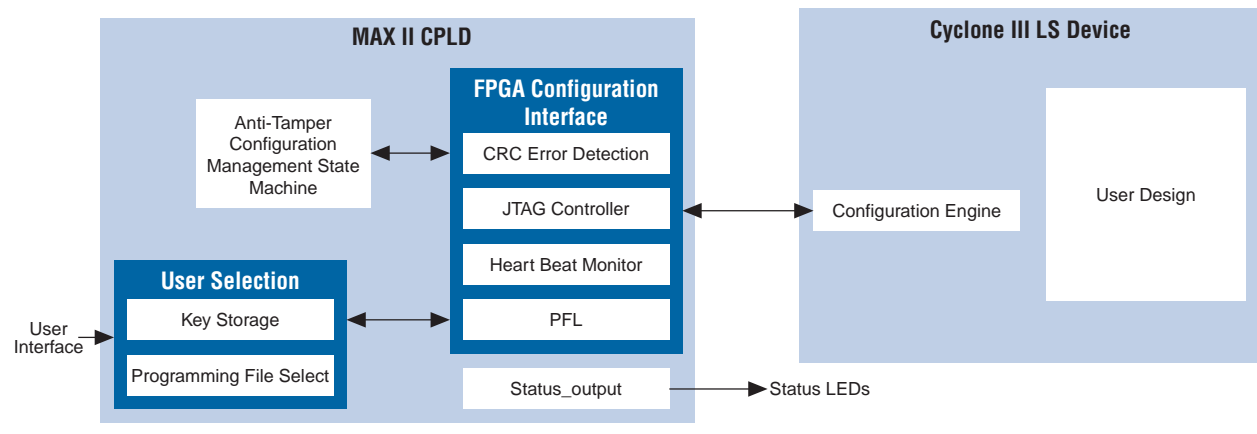
The anti-tamper design example is included as a MAX II system controller feature that ships with the Cyclone III LS FPGA Development Kit. The configuration mode is enabled when the AT_SEL dip switch (SW2.3) is on. Source code files for the anti-tamper design example are included with this application note.

The anti-tamper design example is an external configuration manager on a MAX II device that provides the following security services:

- Programs the battery-backed configuration bitstream file decryption key
- Enables JTAG port protection
- Monitors for CRC errors on the Cyclone III LS device during runtime
- Operationally checks the design programmed into the Cyclone III LS device

Figure 2 shows the functional block diagram.

Figure 2. Block Diagram



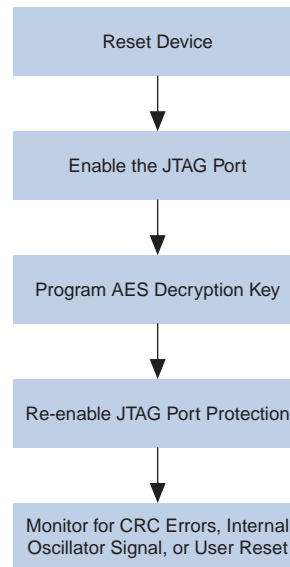
The MAX II controller configures the EP3CLS200 device with the configuration bitstream files stored in on-board flash memory. The MAX II system controller loads configuration bitstream files into the EP3CLS200 device with the Parallel Flash Loader (PFL) megafunction and a passive serial interface.

- Refer to “[Using the Anti-Tamper Design Example on the Cyclone III LS FPGA Development Kit](#)” on page 13 for instructions for using and evaluating the anti-tamper design example. For more information about the Cyclone III LS FPGA Development Kit, refer to the *Cyclone III LS FPGA Development Kit User Guide* and the *Cyclone III LS FPGA Development Board Reference Manual*.


Anti-Tamper Design Example Configuration Sequence

The anti-tamper design example performs a configuration sequence to ensure that the MAX II system controller programs the decryption key, enables JTAG port protection, and actively monitors for configuration errors during runtime. [Figure 3](#) shows the initialization sequence.

Figure 3. Anti-Tamper Configuration Sequence



The following sections describe the steps in the configuration process and give some background information about the device features relevant to design security. The full state diagram for the Configuration Management State Machine is in [Figure 5](#) on [page 10](#).

 For more details about Cyclone III LS device features and IEEE 1149.1 in Altera devices, refer to the [Cyclone III Device Handbook](#) and [AN 39: IEEE 1149.1 \(JTAG\) Boundary Scan Testing in Altera Devices](#), respectively.

Enabling JTAG Port Instruction Access

The Cyclone III LS device powers up with limited JTAG instruction access. The device allows only the mandatory IEEE 1149.1 instructions and a FACTORY instruction to enable access to the full JTAG instruction set. Because the decryption key must be programmed over the JTAG port, the controller must execute the FACTORY instruction to issue a key program. [Table 1](#) describes the FACTORY instruction.

Table 1. FACTORY JTAG Instruction

JTAG instruction	Instruction Code	Description
FACTORY	10 1000 0001	Enables access to all other JTAG instructions (other than BYPASS, SAMPLE/PRELOAD and EXTEST instructions, which are supported upon power-up). This instruction also clears the device configuration data and advanced encryption standard (AES) volatile key.

Two conditions must be satisfied for the FACTORY instruction to successfully enable access to the full Cyclone III LS JTAG instruction set:

- The FACTORY instruction must be issued before any configuration data is loaded onto the device
- The FACTORY instruction must be issued after a power-on reset (POR); issuing the FACTORY instruction after resetting the device with `nConfig` does not enable JTAG instruction access

If you use a configuration scheme other than JTAG, you must prevent the configuration cycle from starting after POR to successfully unlock the JTAG port. You can prevent the configuration load by holding the open-drain `nSTATUS` pin low or by withholding the start byte to the Cyclone III LS device.

If the FACTORY instruction is issued after configuration starts, JTAG instruction access is not enabled; however, the Cyclone III LS device still clears the configuration data and the AES volatile key.

 Do not use the FACTORY instruction during User mode.

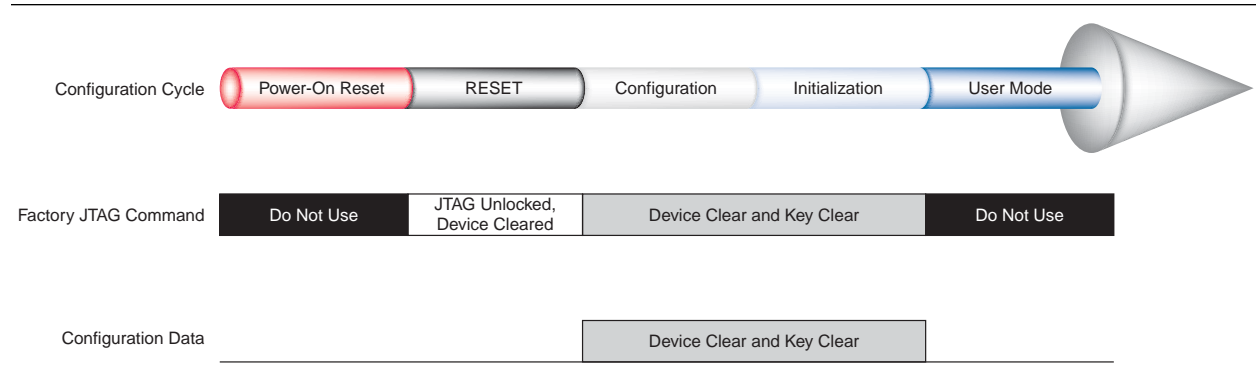
Perform the following steps to load the FACTORY instruction:

1. Power up all voltage input rails required for device configuration, including V_{ccBAT} .
2. Prevent the Cyclone III LS device from configuring.
3. Load the FACTORY JTAG instruction and return to the RTI JTAG state, while continuing to provide a clock signal on `TCK`.
4. Release `nSTATUS`. If driven low, wait for `nSTATUS` to de-assert.

The MAX II controller uses the PFL megafunction to control device configuration with a passive serial configuration scheme. To hold the device in reset after a POR cycle, the design example disables the PFL from initiating a reconfiguration.

Figure 4 shows the behavior of the FACTORY instruction to enable the JTAG instruction set access.

Figure 4. FACTORY JTAG Command Behavior During the Configuration Cycle



Configuration File Decryption

The KEY_PROG_VOL command shifts a 512-bit sequence into the FPGA. [Table 2](#) describes the instruction sequence.

Table 2. KEY_PROG_VOL JTAG Instruction

JTAG instruction	Instruction Code	Description
KEY_PROG_VOL	01 1010 1101	Targets the battery-backed AES key register as the active register chain. This register chain accepts a 512-bit value. The last value in the register chain is not routed to TDO to prevent read-back.

After the 512-bit key string is shifted in the Shift-DR JTAG state, the controller must provide a minimum of 400 clock cycles in the RTI state for the Cyclone III LS device to store the key value into the battery-backed registers.

This design uses two pre-placed keys stored in user flash. You can select between the two keys and the corresponding configuration bitstream files with dip switches and push-buttons on the board.

Refer to “[Evaluating the Anti-Tamper Design Example](#)” on page 15 for details on how to select keys and configuration bitstream files.

The key value that is shifted into the device is the concatenation of the user-selected key1 and key2 values in the Quartus® II software.



The key value is shifted in the right most nibble first. For example, if the key value is 0x012CD, the resulting bitstring shifted is:

1-1-1-0--1-1-0-0--0-0-1-0--0-0-0-1--0-0-0-0



For more information about generating an encrypted configuration bitstream file, refer to [AN 589: Using the Design Security Feature in Cyclone III LS Devices](#).

Re-Enabling JTAG Port Protection

You can re-enable JTAG port protection only by issuing a POR to the Cyclone III LS device. The MAX II system controller on the Cyclone III LS FPGA Development Kit controls the voltage regulator to the V_{CCA} power rail on the Cyclone III LS device. After the key has been programmed into the device, the Configuration Management State Machine issues a POR to the FPGA by power cycling V_{CCA} and initiates a reconfiguration with the PFL megafunction.

Monitor the Device During User Mode

During User mode, the MAX II system controller monitors for configuration RAM errors or a time out on the internal oscillator.

Anti-Tamper Configuration Management State Machine

[Figure 5](#) shows the anti-tamper Configuration Management State Machine. A description of each state and the inputs to the state machine are described in [Table 3](#) and [Table 4](#), respectively.

The three logical groupings for the states are:

- Startup and Initialization—programs the encryption key and re-enables JTAG port protection.
- User_Mode_Monitor—detects CRC errors and the internal oscillator on the Cyclone III LS device.
- Critical Error State—indicates if there is a programming error, if the number of CRC errors detected is 3 or more, or if the internal oscillator stopped.

Figure 5. Anti-Tamper Configuration Management State Machine

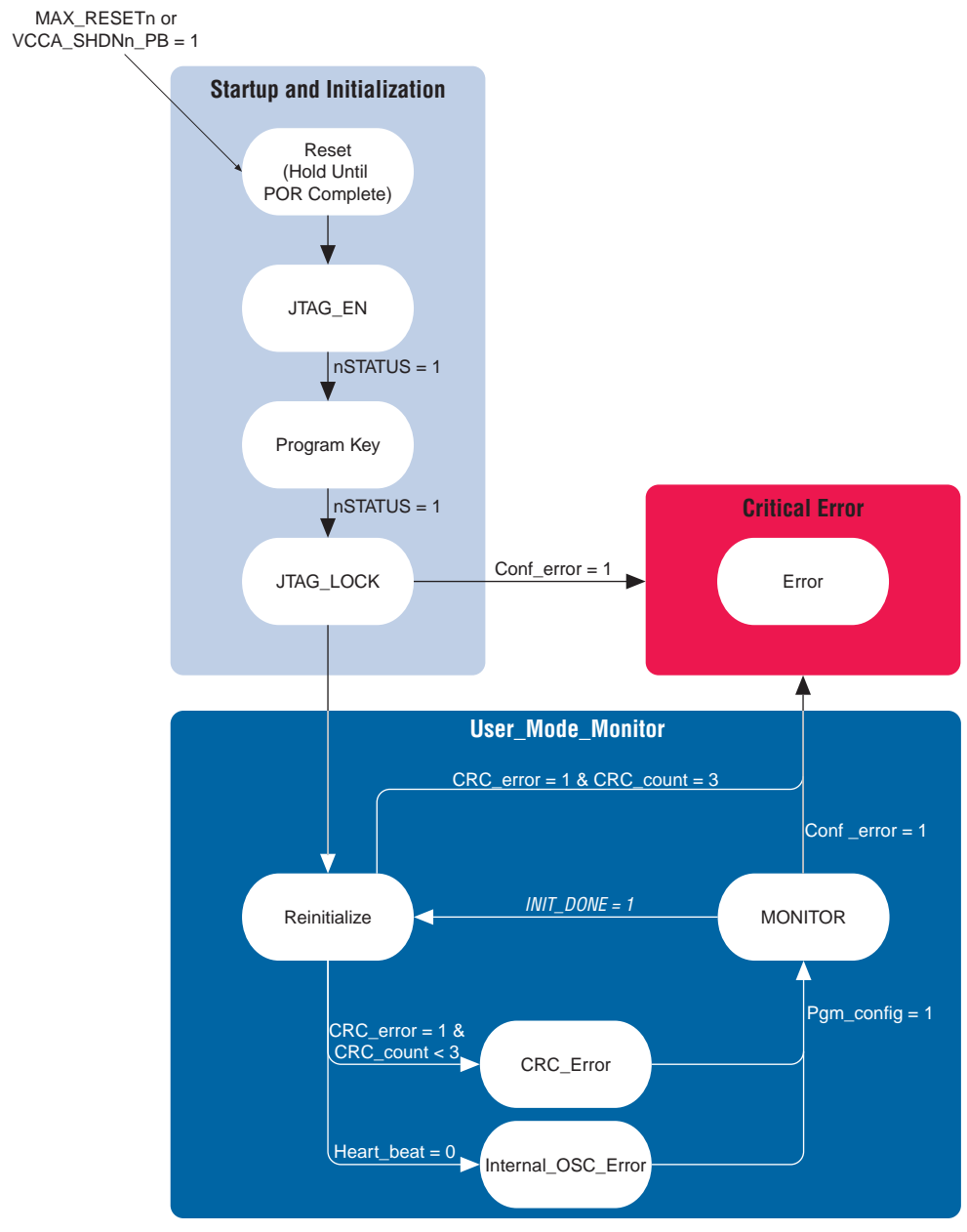


Table 3. Configuration State Machine State Description (Part 1 of 2)

State	Function
RESET	This is the power-up state. The anti-tamper IP must wait in this state until power rails have reached steady state.
JTAG_EN	This state instructs the JTAG controller to issue the FACTORY instruction. The JTAG controller must wait in the RTI JTAG state and continue to provide TCK clock cycles to the EP3CLS200 device until nSTATUS de-asserts. nSTATUS asserts high after the FACTORY instruction completes successfully. Key material is cleared.

Table 3. Configuration State Machine State Description (Part 2 of 2)

State	Function
PROG_KEY	This state starts the programming process for the AES decryption key. KEY_PROG_VOL targets the 512-bit key register. After a new 512-bit key is shifted in, the controller must wait 400 TCK clock cycles in the RTI state. nSTATUS asserts high after the key programs successfully.
JTAG_LOCK	This state re-enables JTAG port protection by issuing a POR to the Cyclone III LS device.
MONITOR	This state monitors the Cyclone III LS User mode design. The Security_0 status LED is asserted here.
CRC_Error	This state clears the EP3CLS200 device in response to a CRC error by pulling the nCONFIG pin low. The CRC_Error LED is asserted here.
Internal_OSC_Error	This state clears the EP3CLS200 device in response to a time out from the heartbeat monitor. The Security_1 LED is asserted here.
Reinitialize	This state instructs the PFL to reconfigure the EP3CLS200 FPGA.
ERROR	This state indicates that a programming error occurred, the number of consecutive CRC error events that happen has reached a threshold of 3 errors, or the internal oscillator has stopped. Only the MAX_RESETh or the VCCA_SHDNn push-buttons will exit this state.


Table 4. Inputs to the Configuration State Machine


Inputs to Anti-Tamper Configuration Management State Machine	Description
MAX_RESETh	Resets the MAX II system controller. This signal is sourced from the MAX_RESET push-button on the Cyclone III LS FPGA Development Kit.
VCCA_SHDNn_PB	Resets the MAX II system controller. This signal is sourced from the VCCA_SHDNn push-button on the Cyclone III LS FPGA Development Kit.
FPGA_nSTATUS	Open drain configuration status pin from the EP3CLS200. You can drive this signal low to hold the device in reset.
FPGA_CONF_DONE	Indicates that the FPGA has been successfully configured. This signal is sourced from the EP3CLS200 device CONF_DONE pin.
CRC_ERROR	Indicates a CRC error. This signal asserts high if the CRC error detection pin on the EP3CLS200 FPGA asserts high or if the CRC_ERROR push-button is asserted.
Conf_error	Indicates a configuration error. This signal is sourced from the PFL megafunction.
Heartbeat	Signal from the heartbeat monitor circuit. A logic 0 indicates that the internal oscillator on the EP3CLS200 FPGA has timed out.
CRC_count	Count value from the CRC detection monitor. The threshold for CRC errors before the state machine goes into the ERROR state is set to 3.

CRC Error Detection

Cyclone III LS devices contain a 32-bit CRC value in the configuration RAM of the device. During User mode, the CRC error detection circuitry continuously calculates a CRC value based on the contents of the configuration RAM array and compares it to the expected value. If an error occurs, the CRC error pin asserts high, and the MAX II system controller samples the pin to report a CRC error.

The **CRC_ERROR** push-button on the Cyclone III LS FPGA Development Kit simulates a CRC error in the EP3CLS200 FPGA.


 The **CRC_ERROR** push-button does not cause an actual CRC error on the EP3CL200 FPGA. The **CRC_ERROR** push-button only tests the ability of the MAX II system controller to detect a CRC error on the EP3CL200 FPGA.

 For more information about the CRC error detection feature in Cyclone III devices, refer to the *SEU Mitigation in Cyclone III Devices* chapter in the *Cyclone III Device Handbook*.

JTAG controller

The JTAG controller issues the FACTORY instruction and the Key Program command.

There are two 512-bit key values stored in the MAX II user flash memory that the JTAG controller uses to program the key value into the FPGA. The key value sent is determined by a dip switch setting (SW2.2) on the Cyclone III LS FPGA Development Kit.

 The MAX II system controller has exclusive control over the JTAG chain when the Anti-Tamper (AT) mode is active. The JTAG chain configuration is controlled by the JTAG_AT_SEL signal on the analog switch U23. During AT mode, the MAX II system controller drives the JTAG_AT_SEL signal to logic 0 when the MAX II system controller requires JTAG resources for the EP3CL200 FPGA. The JTAG chain configuration returns to the default configuration when the MAX II system controller does not require a JTAG communication link or when the AT mode is inactive.

 For more information about the JTAG chain configuration on the Cyclone III LS FPGA Development Kit, refer to *Cyclone III LS FPGA Development Board Reference Manual*.

Heartbeat Monitor

In anti-tamper design example, the Cyclone III LS internal oscillator should be routed to the HEARTBEAT signal. The MAX II system controller implements a watchdog timer circuit with the HEARTBEAT signal. After the watchdog timer times out, an error is asserted to the anti-tamper Configuration Management State Machine.

Parallel Flash Loader

The EP3CL200 FPGA is configured with a passive serial interface. The MAX II controller uses the Parallel Flash Loader (PFL) megafunction to load the configuration bitstream files to the EP3CL200 FPGA.

 For more information about the Parallel Flash Loader (PFL), refer to *AN 386: Using the Parallel Flash Loader with the Quartus II Software*.

Using the Anti-Tamper Design Example on the Cyclone III LS FPGA Development Kit

Figure 6 shows the Cyclone III LS device Development Kit. Table 5 and Table 6 list the switches, buttons, and status LEDs that are associated with the security features on the Cyclone III LS FPGA Development Kit.

Figure 6. User Interface for Anti-Tamper Design Example

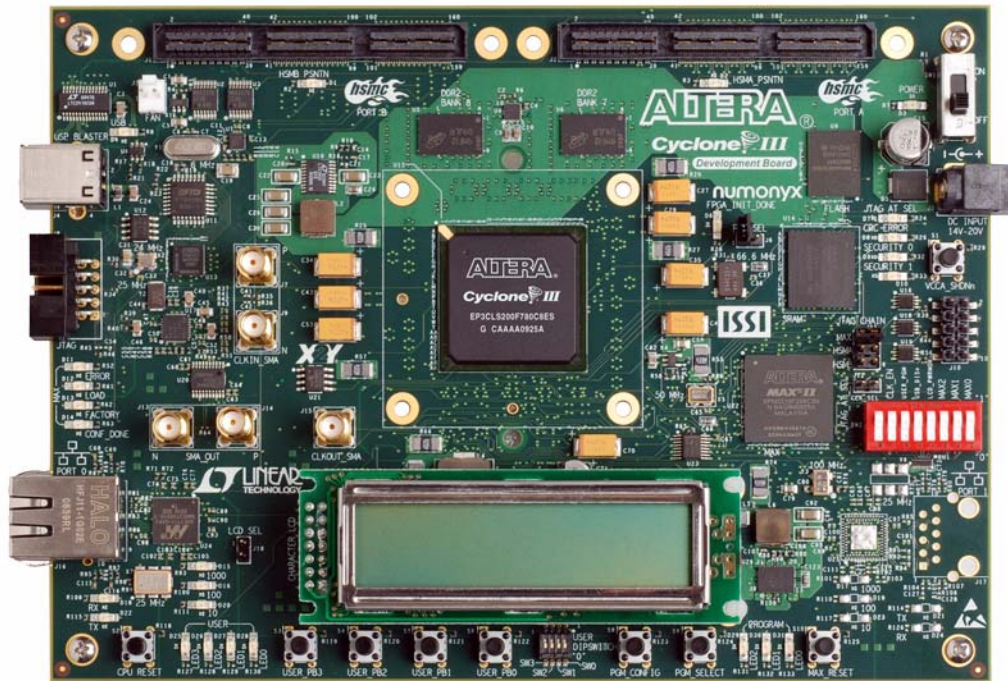


Table 5. MAX II DIP Switches Related to Security Features

Switch	Signal Name	Description	Default Value
SW2.2	MAX1	When the anti-tamper mode is active, SW2.2 selects the pre-placed key. "0" selects the pre-placed key associated with the Page_1 configuration bitstream file included with this design example. "1" selects the pre-placed key associated with the Page_2 configuration bitstream file included with this design example.	0
SW2.3	AT_ACTIVE	"1" enables the anti-tamper design example.	0
SW4.4 (1)	JTAG_SECURE	This switch determines whether or not the MAX II system controller issues the FACTORY instruction upon power-up to enable access to the full JTAG instruction set. "0" issues the FACTORY instruction during POR of the EP3CL200 FPGA.	0

Note to Table 5:

(1) Not used in the anti-tamper design example.

Table 6. LEDs and Push-Buttons Used for the Anti-Tamper Design Example

Signal Names	Description
LEDs	
JTAG_AT_SEL	Illuminated when the MAX II system controller has control of the JTAG chain.
CRC_ERROR_MAX	Illuminated when a CRC error occurs on the FPGA.
SECURITY_LED0 , SECURITY_LED1	Status LEDs for the anti-tamper design example. Security 0, On: Indicates that the anti-tamper design example is in the Monitor state. Security 1, On: Indicates that the anti-tamper design example is in the Internal_OSC_error state. Security 1, Security 0, Blinking: Indicates that the anti-tamper design example is in the critical error state.
PGM_LED[2 : 0]	Indicates that the MAX II system controller will load a configuration bitstream file at the next re-configuration. One-hot encoding.
Push Buttons	
PGM_SEL	Changes the configuration bitstream file that the MAX II system controller will load at the next reconfiguration (based on the PGM_LEDs).
PGM_CONFIG	Initiates reconfiguration during the CRC_Error and Internal_OSC_Error states in the anti-tamper design example.
MAX_RESETh	Resets the MAX II system controller; Reinitiates the startup and initialization states in the anti-tamper design example.
VCCA_SHDNh	Initiates POR to the EP3CLS200 FPGA.
CRC_ERROR_PB	Forces the Configuration Management State Machine into the CRC_Error state.

The anti-tamper design example shipped with the Cyclone III LS FPGA Development Kit contains two pre-placed keys stored in user flash memory on the MAX II system controller. The anti-tamper design example allows you to select between two images stored in the on-board flash memory. The encrypted configuration bitstream files corresponding to the pre-placed keys are provided with the anti-tamper design example files. However, the configuration bitstream files are not programmed by default in the user flash memory.

Design Example Contents

The source code for the anti-tamper design example, the key files, and the supporting configuration bitstream files are shipped on the CD packaged with the Cyclone III LS FPGA Development Kit. All content related to the anti-tamper design example is located in the following directory:

`<install_directory>\kits\cycloneIIIS_3cls200_fpga\examples\max2\at_example`

Table 7 lists the contents of the example directory.

Table 7. Files Included with the Anti-Tamper Design Example (Part 1 of 2)

File/Folder name	Description
fpga_based_pfl.sof	This SRAM Object File (.sof) is a Parallel Flash Loader configuration bitstream file that loads Encrypted_images.pof .
Encrypted_images.pof	The Programmer Object File (.pof) contains two encrypted images for Page_1 and Page_2 of the on-board flash that corresponds to the pre-placed keys on the MAX II system controller.

Table 7. Files Included with the Anti-Tamper Design Example (Part 2 of 2)

File/Folder name	Description
encryption_keys (Folder)	Contains key1.key and key2.key. Each file contains a 256-bit key string. Page_1 is encrypted with key1 concatenated with key2. Page_2 is encrypted with key2 concatenated with key1.
Page1_project_file (Folder)	Contains the Quartus II Archive File (.qar) for the project in Page_1 of Encrypted_images.pof .
Page2_project_file (Folder)	Contains the .qar for the project in Page_2 of Encrypted_images.pof

Evaluating the Anti-Tamper Design Example

Before you evaluate the anti-tamper design example, load the configuration bitstream file that corresponds with the pre-placed key. Instructions for loading the on-board flash memory are included in [“Loading the On-Board Flash Memory”](#).

To enable the anti-tamper design example, perform the following steps:

1. Select the key with dip switch SW2.2.
2. To enable the anti-tamper design example, set dip switch SW2.3 to logic 1.
3. Reset the anti-tamper design example with the **MAX_RESET** or **VCCA_SHDNn** push-buttons. The anti-tamper design example attempts to load the factory image. A watchdog time out occurs on the factory image.
4. Select the programming file that corresponds to the key that you selected with the **PGM_SEL** push-button. The **PGM_SEL** push-button toggles the PGM LEDs.
5. Reinitialize the device with the **PGM_CONFIG** push-button. When PGM_LED1 is illuminated, Page_1 is loaded. When PGM_LED2 is illuminated, Page_2 is loaded.
6. If the configuration is successful, the Security 0: LED will be illuminated, indicating that the anti-tamper configuration state machine is in the monitor state.

During the monitor state, you can issue a CRC error with the **CRC_ERROR** push-button.

On both FPGA images, the **USER_PB0** push-button disables the internal oscillator, which causes a watchdog time out.

Loading the On-Board Flash Memory

The on-board flash memory supports up to three FPGA configuration bitstream files. Page_0 is the factory image loaded in the flash memory. Page_1 and Page_2 are user flash images.

If you erase the factory image, refer to the [Cyclone III LS FPGA Development Kit User Guide](#) to restore the FPGA image (factory image) shipped with the Cyclone III LS FPGA Development Kit.

Perform the following steps to program the on-board flash memory with the encrypted configuration bitstream files that correspond with the pre-placed keys:

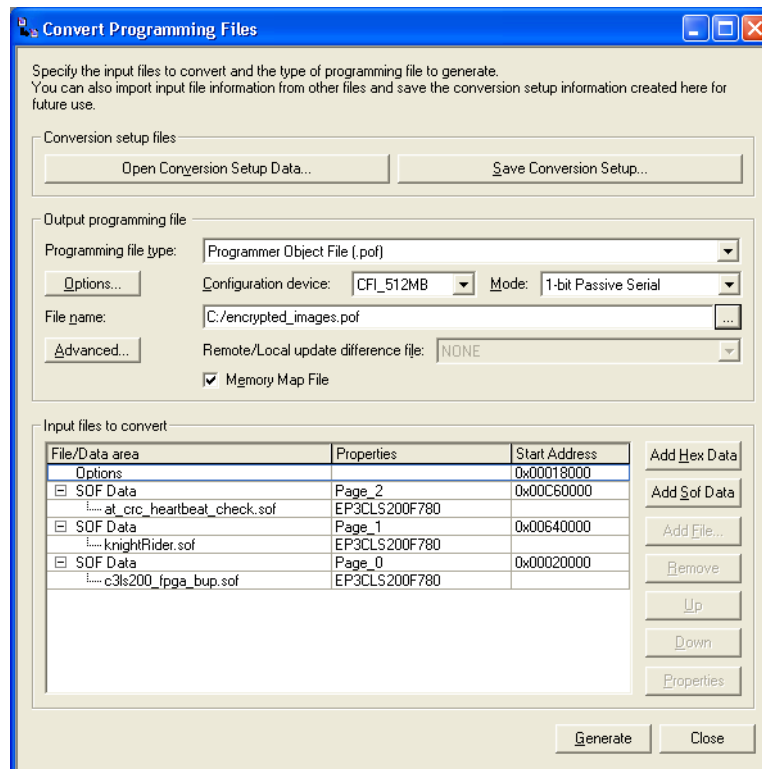
1. Ensure that SW2.3 and SW2.4 are set to logic 0 to inactivate the anti-tamper design example. The device powers up with the JTAG port unlocked.
2. Power cycle the Cyclone III LS FPGA Development Kit.

3. Program the EP3CLS200F780 device with the Quartus II Programmer and the `fpga_based_pfl.sof`.
4. After you program the EP3CLS200F780 device, use the **Auto-Detect** command on the chain. A 512-MB CFI flash module is attached to the EP3CLS200F780 FPGA.
5. Target the CFI flash module with the `encrypted_images.pof`.
6. In the Program/configure check boxes, select Page_1, Page_2, and OPTION bits. Do not select Leave Page_0. Click **Start** to program the selected flash sectors.

Generating Your Own Configuration Images

Use the **Convert Programming Files** dialog box to create and load a `.pof` with multiple configuration bitstream files into the on-board flash memory. The anti-tamper design example uses Page_1 and Page_2 of the on-board flash memory to store two configuration bitstream files. [Figure 7](#) shows the settings for creating a `.pof` to program flash memory.

Figure 7. Generating a `.pof` to Program Flash Memory



Perform the following steps to generate a `.pof` for the Cyclone III LS FPGA Development Kit:

1. On the File menu, click **Convert Programming Files**. The **Convert Programming Files** dialog box appears.

- Under **Output programming file**, set the options listed in [Table 8](#).

Table 8. Output Programming File Settings

Setting	Value
Programming file type	Programmer Object File (.pof)
Configuration device	CFI_512 MB
Mode	1-bit Passive Serial

- In the **Options** dialog box, set the **Option bit address** to **0x180000**.
- Click **Add Sof Data** twice to add two SOF data pages.
- Select **Page_0** and click **Add File**. Select the SOF image. The Page_0 .sof is a place holder and is not programmed into the on-board flash.
- Select **Page_1** and **Page_2** and click **Add File**. Place the SOF images that the anti-tamper design example should target.
- Set the start addresses for each SOF Data page to the following:
 - **Page_0: 0x00020000**
 - **Page_1: 0x00640000**
 - **Page_2: 0x00C60000**

To set the start addresses for each SOF Data page, open the **Properties** dialog box by selecting the SOF Data field and clicking **Properties**.

- Encrypt the SOF images in Page_1 and Page_2 by selecting the .sof name under **Input files to convert** and clicking the **Properties** button.



For more information about generating an encrypted configuration bitstream file, refer to [AN 589: Using the Design Security Feature in Cyclone III LS Devices](#).

Changing the Pre-Placed Key Value

The pre-placed key is placed on the MAX II controller design with the on-chip user flash memory. To modify the pre-placed key, perform the following steps:

- Modify a copy of the **key.mif**, located in the `<install_directory>/examples/max2/at_example/encryption_keys` directory.
- Replace the **key.mif** in the MAX II system controller Quartus II project with the **key.mif** you modified. The MAX II system controller project archive is located in the `<install_directory>/examples/max2` folder.
- Run the Assembler.
- Program the MAX II system controller and power-cycle the Cyclone III LS FPGA Development Kit.

The **key.mif** is a 1-kbit file that contains two encryption keys. The encryption key material is a 512-bit hex string stored in little endian format. An example of a **.mif** containing two decryption keys is shown in [Figure 8](#). The first keystring in the **.mif** is:

```
0x01234567890123456789012345678901234567890123456789012345678901234567980123abcdefabcdefabcdefab
cdefabcdefabcdefabcdefabcdefabcdefabcdef
```

Figure 8. .mif Containing the Encryption Key

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	CD	AB	EF	CD	AB	EF	CD	AB
8	EF	CD	AB	EF	CD	AB	EF	CD
16	AB	EF	CD	AB	EF	CD	AB	EF
24	CD	AB	EF	CD	AB	EF	CD	AB
32	23	01	98	67	45	23	01	89
40	67	45	23	01	89	67	45	23
48	01	89	67	45	23	01	89	67
56	45	23	01	89	67	45	23	01
64	23	01	98	67	45	23	01	89
72	67	45	23	01	89	67	45	23
80	01	89	67	45	23	01	89	67
88	45	23	01	89	67	45	23	01
96	CD	AB	EF	CD	AB	EF	CD	AB
104	EF	CD	AB	EF	CD	AB	EF	CD
112	AB	EF	CD	AB	EF	CD	AB	EF
120	CD	AB	EF	CD	AB	EF	CD	AB

Conclusions

The Cyclone III LS FPGA device features enable security services that protect against theft and reverse engineering of proprietary design files. This anti-tamper design example provides a template to enable a passive and active protection boundary around the FPGA.

Revision History

Table 9 shows the revision history for this application note.

Table 9. Revision History

Date and Version	Changes Made
October 2009, v1.0	Initial release.



101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support
www.altera.com/support

Copyright © 2009 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

