

This application note describes the design security feature in Cyclone[®] III LS devices.

The design security feature is able to decrypt a configuration bitstream using an industry standard 256-bit advanced encryption standard (AES) key and AES encryption algorithm to protect your designs from the following unauthorized activities:

- Copying
- Reverse engineering
- Tampering

The following design features make the solution secure:

- Cyclone III LS devices do not support configuration file read back. This prevents attempts to read back the configuration file after it is decrypted.
- Two user-defined 256-bit sequences are required to generate the 256-bit AES key.
- The volatile 256-bit AES key is stored in the Cyclone III LS device and cleared when the external battery is removed.
- Depending on the security mode, you can configure the Cyclone III LS device using a configuration file that is encrypted with the same key, or configure with a normal configuration file for board level testing.



Not every configuration scheme supported in the Cyclone III LS devices supports the design security feature. For more information about the supported configuration schemes for the design security feature, refer to [“Supported Configuration Scheme” on page 18](#).

Security Encryption Algorithm


Cyclone III LS devices have a dedicated AES decryption block that uses the AES algorithm to decrypt configuration data using a 256-bit AES key. Before receiving the encrypted data, the 256-bit AES key must be written to configure the FPGA device.

The AES algorithm is a symmetrical block cipher that encrypts and decrypts data in blocks of 256 bits. The encrypted data is subject to a series of transformations that include byte substitutions, data mixing, data shifting, and key additions.

If the design security feature is not used, the AES decryption block is bypassed. The Cyclone III LS AES implementation is validated as conforming to the Federal Information Processing Standards FIPS-197.



For more information about the AES algorithm, refer to the *Federal Information Processing Standards Publication FIPS-197* or the *AES Algorithm (Rijndael) Information* at www.csrc.nist.gov.

 For more information about the Cyclone III LS AES validation, refer to the *Advanced Encryption Standard Algorithm Validation List* published by the National Institute of Standards and Technology (NIST) at www.csrc.nist.gov.

Volatile Key Programming Method

Cyclone III LS devices offer volatile key storage. The volatile key storage requires battery back-up but enables the key to be updated. [Table 1](#) shows the volatile key features.

Table 1. Volatile Key Features

Volatile Key Feature	Description
Key Length	256 bits
Key Programmability	Reprogrammable and erasable
External Battery	Required
Key Programming Method (1)	On-board programming (2)
Design Protection	Secure against copying, reverse engineering, and tampering (3)

Notes to [Table 1](#):

- (1) Key programming is carried out via the JTAG interface.
- (2) On-board programming is a key programming procedure in which the device is programmed on the board instead of a separate programming system.
- (3) Volatile key clear and key program JTAG instructions from the device core are supported. For more information about the JTAG instructions for volatile key clear and key program, refer to the *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Cyclone III Devices* chapter in volume 1 of the *Cyclone III Device Handbook*.

[Table 2](#) describes the two methods for volatile key on-board programming.

Table 2. Key Programming Methods

Programming Method	Programming Tool
On-Board Programming (Prototyping) (1)	EthernetBlaster communications cable, JTAG Technologies, ByteBlaster™ II cable, USB-Blaster™ download cable
On-Board Programming (Production) (2)	In-circuit tester, JTAG Technologies

Notes to [Table 2](#):

- (1) Initially used to verify proper operation of a particular method.
- (2) Used for large-volume production.

Hardware and Software Requirements

This section describes the hardware and software requirements for the Cyclone III LS design security feature.

Hardware Requirements

For successful volatile key programming, you must follow the voltage specifications of the design security feature. Table 3 shows the specifications for volatile key programming.

Table 3. Specifications for Volatile Key Programming

Parameter	Value
TCK Period	Refer to the “JTAG Specification” section in the <i>Cyclone III LS Device Data Sheet</i> chapter in volume 2 of the <i>Cyclone III Device Handbook</i>
Ambient Temperature	Refer to the specification of the operating junction temperature, T_j in the <i>Cyclone III LS Device Data Sheet</i> chapter in volume 2 of the <i>Cyclone III Device Handbook</i>
Voltage (V_{CCBAT})	1.2 V (minimum), 3.0 V (typical), 3.3 V (maximum)

V_{CCBAT} is a dedicated power supply for the volatile key storage. It is separated from other on-chip power supplies, such as V_{CCIO} or V_{CC} . V_{CCBAT} continuously supplies power to the volatile register regardless of the on-chip supply condition.

V_{CCBAT} is one of the required power-up voltages for the Cyclone III LS device to exit from power-on reset (POR). You must ensure that V_{CCBAT} is powered up to the appropriate voltage level and the device exits from POR to begin key programming or configuration. To ensure that V_{CCBAT} reaches the full rail before key programming, you must wait for 100 ms (standard POR) or 12 ms (fast POR).



You can use lithium coin-cell type batteries as the voltage supply for V_{CCBAT} . Examples of lithium coin-cell type batteries are BR1220 (–30°C to +80°C) and BR2477A (–40°C to +125°C).



For more information about the required power-up voltages for Cyclone III LS devices to exit from POR, refer to the “POR Circuitry” section in the *Configuration, Design Security, and Remote System Upgrades in Cyclone III Devices* chapter in volume 1 of the *Cyclone III Device Handbook*.

Software Requirements

To use the design security feature in Cyclone III LS devices, you must use the Quartus® II software version 9.0 SP2 and onwards. A license file is needed to enable the Cyclone III LS design security feature in the Quartus II software.



To obtain the license file for the design security feature, contact Altera Technical Support at www.altera.com/support for assistance.

How to Set Up the Design Security License File in the Quartus II Software

Perform the following steps to use and set up the design security license file:

1. Obtain a license file to enable the Cyclone III LS design security feature from Altera Technical Support at www.altera.com/support.
2. Start the Quartus II software.
3. On the Tools menu, click **License Setup**. The **Options** dialog box displays the **License Setup** options.
4. In the **License file** field, enter the location and the name of the license file, or browse to the file location and select the license file.
5. Click **OK**.

Steps for Implementing a Secure Configuration Flow

Secure configuration flow refers to the configuration flow when the design security feature is used. The secure configuration flow differs from the usual configuration flow. To implement a secure configuration flow, perform the following steps:

1. Generate the encryption key programming file and encrypt the configuration data.

The Quartus II software uses the user-defined 256-bit sequence to generate a 256-bit AES key. This key is used to encrypt the configuration file. The encrypted configuration file is stored in an external memory, such as a flash memory or configuration device.

For more information, refer to “[Step 1: Generate the Key File and Encrypt Configuration File](#)” on page 5.

2. Program the 256-bit AES key into the Cyclone III LS device.

For more information, refer to “[Step 2: Program the Volatile Key into the Cyclone III LS Device](#)” on page 12.

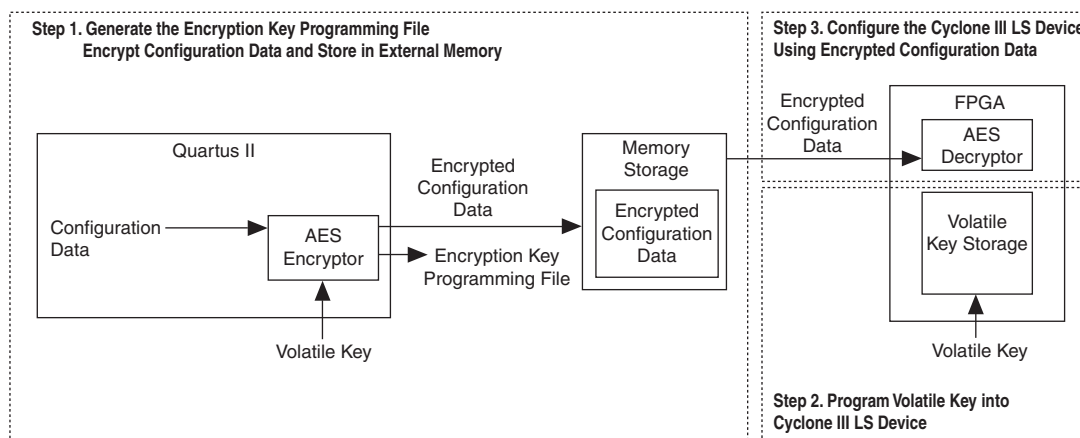
3. Configure the Cyclone III LS device.

Upon power-up, the external memory source sends the encrypted configuration file to the Cyclone III LS device. The Cyclone III LS device uses the stored security key to decrypt the file and the unencrypted data is used to configure itself.

For more information, refer to “[Step 3: Configure the Cyclone III LS Device with Encrypted Configuration File](#)” on page 16.


Figure 1 shows the secure configuration flow of the Cyclone III LS device.

Figure 1. Cyclone III LS Device Secure Configuration Flow



Step 1: Generate the Key File and Encrypt Configuration File

To use the design security feature in Cyclone III LS devices, you must generate key and configuration files that are encrypted with the same key in the Quartus II software. To generate the key, you must define two user-defined 256-bit sequences. The 256-bit AES key is generated from the two user-defined 256-bit sequences and is not saved into any Quartus II-generated configuration files. This makes it impossible to copy the key to other Cyclone III LS devices.

 The decompression feature is not supported for encrypted configuration files.


The Quartus II software generates the 256-bit key as an **.ekp** file. The **.ekp** file has different formats, depending on the hardware and system used for key programming. Table 4 shows the key file formats supported by the Quartus II software and the supported programming tools for each file format.

Table 4. Key File Formats and the Supported Programming Tools

Key File	Programming Tools
JBC (.ekp) (1)	Quartus II software with EthernetBlaster communications cable, USB-Blaster download cable, ByteBlaster II cable, and ByteBlasterMV™ download cable.
JEDEC STAPL (.jam) (2)	Quartus II software with EthernetBlaster communications cable, USB-Blaster download cable, ByteBlaster II download cable, and ByteBlaster MV download cable. Third-party programming vendors and JTAG programmer vendors.
Serial Vector Format (.svf) (2)	JTAG programmer vendors.

Notes to Table 4:

- (1) The **.ekp** file is automatically generated from the Quartus II software during the programming file conversion.
- (2) To generate the **.jam** and **.svf** files, use the **Create JAM, SVF, or ISC File** dialog box in the Quartus II software programmer.

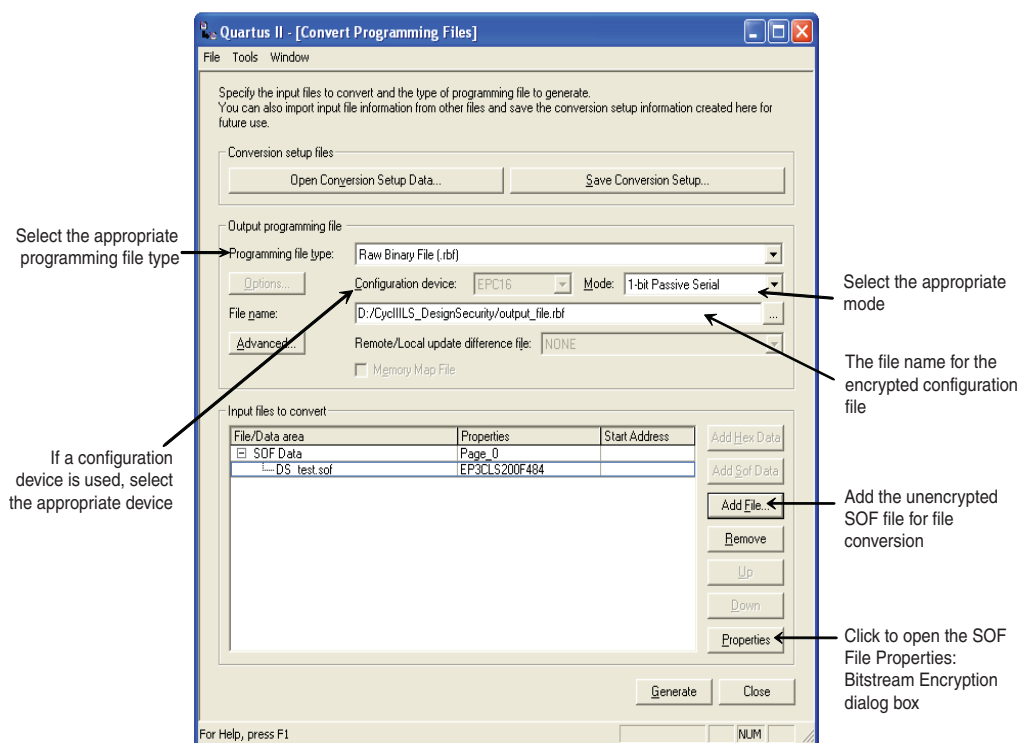
 Altera recommends that you keep the key file confidential.

How to Generate the Single-Device .ekp Key File and Encrypt the Configuration File using the Quartus II Software

To generate a single-device .ekp file and encrypt your configuration file, perform the following steps:

1. Compile your design with one of the following options to generate an unencrypted SRAM Object File (.sof):
 - On the Processing menu, click **Start Compilation**
 - On the Processing menu, point to **Start** and click **Start Assembler**
2. On the File menu, click **Convert Programming Files**. The **Convert Programming Files** dialog box appears (Figure 2).

Figure 2. Convert Programming Files Dialog Box



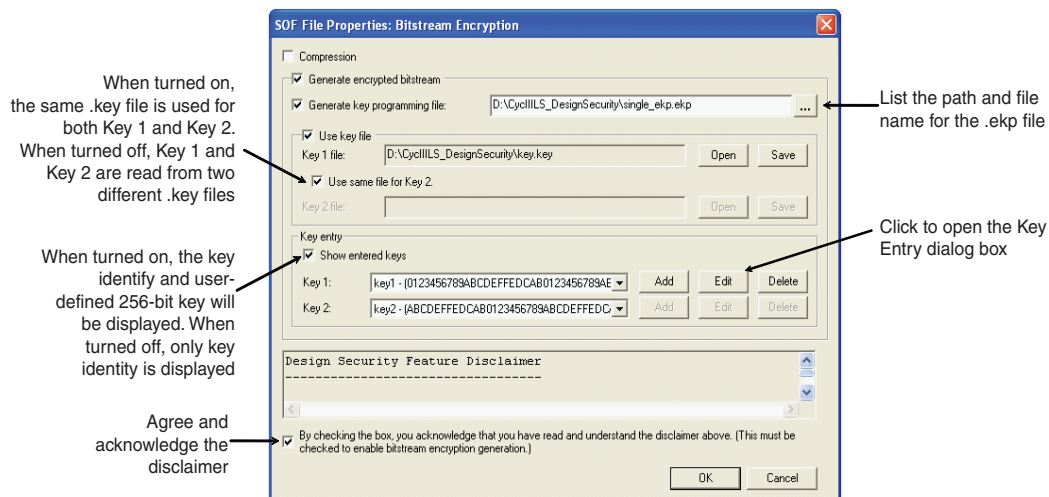
3. In the **Convert Programming Files** dialog box, make the following selections:
 - a. Select the programming file type from the **Programming file type** list.
 - b. If a configuration device is used, select the appropriate device from the **Configuration device** list.
 - c. Select the mode from the **Mode** list.
 - d. Type the filename in the **File name** field, or browse to the appropriate directory and select the file.

- e. Under the **Input files to convert** section, click **SOF Data**.
- f. Click **Add File** to open the **Select Input File** dialog box.
- g. Browse to the unencrypted **.sof** file and click **Open**.
- h. Under the **Input files to convert** section, click on the **.sof** filename. The field is highlighted.
- i. Click **Properties**. The **SOF Files Properties: Bitstream Encryption** dialog box appears (Figure 3).
- j. In the **SOF Files Properties: Bitstream Encryption** dialog box, turn on **Generate encrypted bitstream**.
- k. Turn on **Generate key programming file** and type the **.ekp** file path and filename in the text area, or browse to and select **<filename>.ekp**.
- l. Specify the **key 1** and **key 2** from the pull-down list either with a **.key** file or the **Add** button. Key 1 and key 2 are the two user-defined 256-bit sequences that are used to generate the **.ekp** key file and encrypt the configuration bitstream. The **Add** and **Edit** buttons bring up the **Key Entry** dialog box. For more information, refer to “How to Generate the Key 1 and Key 2 from a .key File or Key Entry Dialog Box” on page 8.

The **Delete** button deletes the currently selected key from the pull-down list (Figure 3).

- m. Read the design security feature disclaimer. If you agree and acknowledge the design security feature disclaimer, turn on the acknowledgement box.
- n. Click **OK**. The **<filename>.ekp** and encrypted configuration file are generated in the same project directory.

Figure 3. SOF Files Properties: Bitstream Encryption Dialog Box



How to Generate the Key 1 and Key 2 from a .key File or Key Entry Dialog Box

The .key file is a plain text file in which each line represents a user-defined 256-bit sequence unless the line starts with “#”. The “#” symbol is used to denote comments. Each valid key line has the following format (Figure 4):

<key identity><white space><user-defined 256-bit hexadecimal sequence>


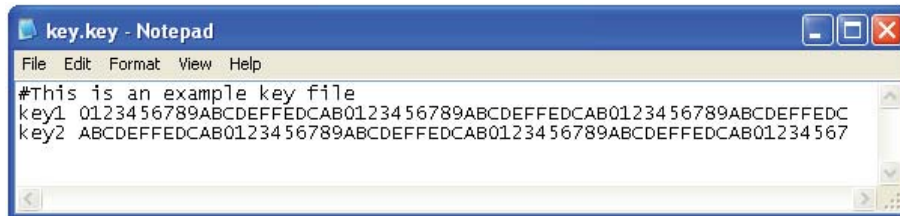
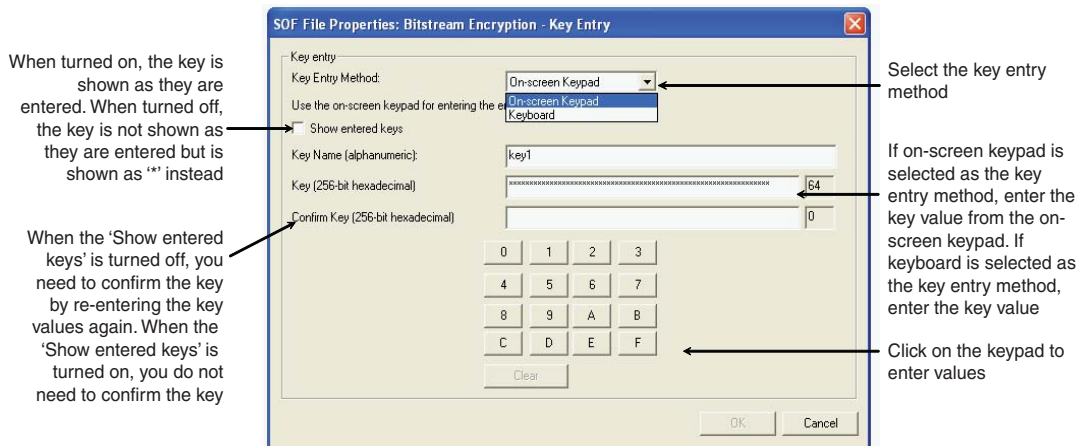
 The key identity is an alphanumeric name that is used to identify the keys (similar to the key file entry).


Figure 4. Example .key File



In the **Key entry** section (Figure 3), select **Key 1** and **Key 2** from the pull-down list or open the **Key Entry** dialog box (Figure 5) to enter the encryption key. The keys in the pull-down list can be saved to a .key file. You must click the corresponding **Save** button in the **Use key file** section to save a key and to display the standard **File** dialog box. All keys in the pull-down list are saved to the selected or created .key file.

Figure 5. Key Entry Dialog Box



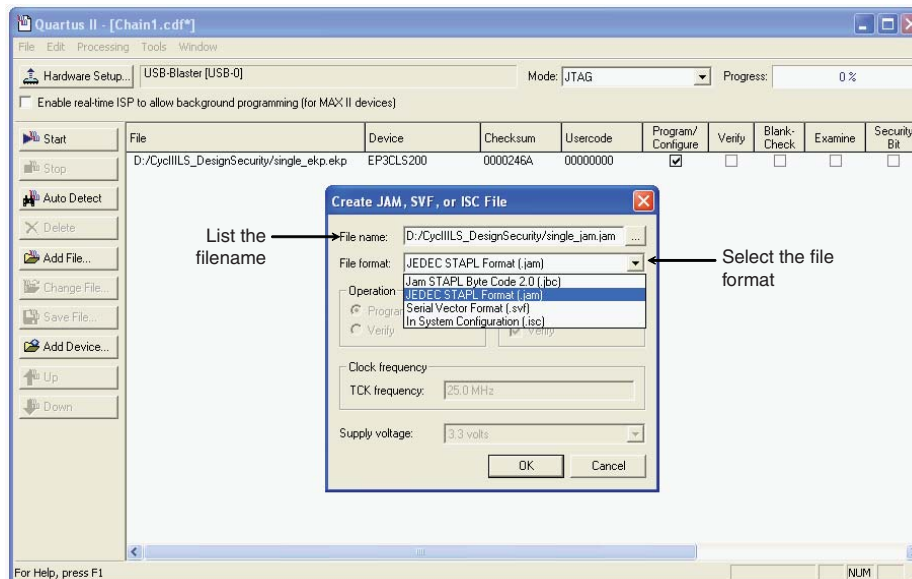
 While the on-screen keypad is being used, any attempt to use the keyboard to enter the key generates a pop-up notification and the key press is ignored.

How to Generate the Single-Device .jam or .svf Key File

To generate the single-device .jam or .svf key file, perform the following steps:

1. On the Tools menu, click **Programmer**. The **Programmer** dialog box appears.
2. In the **Mode** list, select **JTAG** as the programming mode.
3. Click **Hardware Setup**. The **Hardware Setup** dialog box appears.
 - a. In the currently selected hardware list, select the programming cable.
 - b. Click **Done**.
4. Click **Add File**. The **Select Programmer File** dialog box appears.
 - a. Type <filename>.ekp in the **File name** field.
 - b. Click **Open**.
5. Highlight the .ekp file you have added and turn on **Program/Configure**.
6. On the File menu, point to **Create/Update** and click **Create JAM, SVF, or ISC File**. The **Create JAM, SVF, or ISC File** dialog box appears (Figure 6).
7. Select the file format required (**JEDEC STAPL Format [.jam]** or **Serial Vector Format [.svf]**) for the .ekp file in the **File format** field.
8. Type the filename in the **File name** field, or browse to and select the file.
9. Click **OK** to generate the .jam or .svf file.

Figure 6. Create .jam or .svf Key File from .ekp File



How to Generate the Single-Device .ekp File and Encrypt the Configuration File Using the Quartus II Software with the Command-Line Interface

A command-line interface allows you to generate a single-device .ekp file and encrypt a Raw Binary File (.rbf). The command-line interface uses the Quartus II software command-line executable, `quartus_cpf`.



For more information about the available options in the `quartus_cpf`, run `quartus_cpf -help=option` from the Quartus II software command line help.

To generate a single-device .ekp file, the following syntax or options are required:

- `--key / -k <path to key file>:<key identity>`
- A .sof file (user design)
- An .ekp file (the required encryption key programming filename)

```
quartus_cpf --key <keyfile>:<keyid1>:<keyid2> <input_sof_file>
<output_ekp_keyfile>
```

Example 1 shows the command for generating an .ekp file from two sets of keys that are stored in two different key files: key 1 in `key1.key` and key 2 in `key2.key`.

Example 1. Two Different .key Files

```
quartus_cpf --key D:\CIIILS_DS\key1.key:key1 --key
D:\CIIILS_DS\key2.key:key2 D:\CIIILS_DS\test.sof
D:\CIIILS_DS\test.ekp
```

Example 2 shows the command for generating an .ekp file from two sets of keys that are stored in the same key file: key 1 and key 2 in `key12.key`.

Example 2. Same .key File

```
quartus_cpf --key D:\CIIILS_DS\key12.key:key1:key2
D:\CIIILS_DS\test.sof D:\CIIILS_DS\test.ekp
```

To generate an encrypted single-device .rbf file, the following syntax or options are required:

- `-c/--convert`
- A .sof file (user design)
- A .rbf file (the required encrypted .rbf filename)

```
quartus_cpf -c --key <keyfile>:<keyid1>:<keyid2>
<input_sof_file> <output_rbf_file>
```

Example 3 shows the command for generating an encrypted .rbf file from a .sof file.

Example 3. Generate encrypted .rbf file

```
quartus_cpf -c --key D:\CIIILS_DS\key12.key:key1:key2
D:\CIIILS_DS\test.sof D:\CIIILS_DS\test.rbf
```

How to Generate the Multi-Device Key File Using the Quartus II Software

Perform the following steps to generate a multi-device .ekp file:


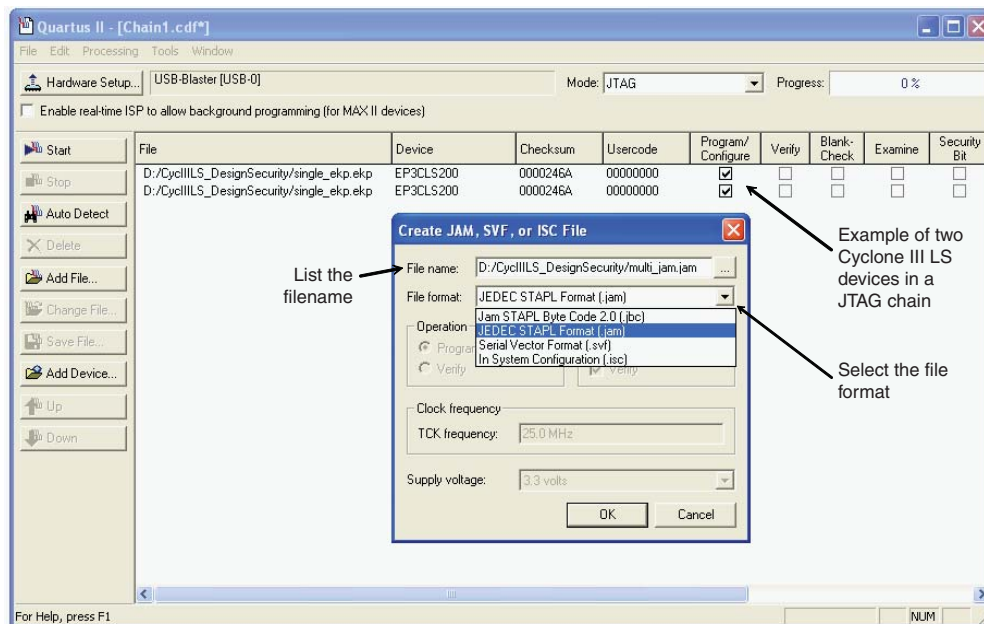
1. Repeat Steps 1–3 in “How to Generate the Single-Device .jam or .svf Key File” on page 9.
 2. Click **Add File**. The **Select Programmer File** dialog box appears.
 - a. Select the single-device .ekp file, and type `<single_ekp>.ekp` in the **File name** field.
 - b. Click **Open**.
-  For the correct sequence of devices in the same JTAG chain, you can use the **Auto-Detect** option in the Quartus II software programmer. If one of the FPGA devices does not need to be key-programmed, you do not need to replace the device with the `<single_ekp>.ekp` file in the Quartus II software programmer.
3. Repeat Step 2 for each device in the same chain. Ensure that the right device sequence is used when adding the .ekp files to the programmer window.
 4. Highlight all the .ekp files you have added and click **Program/Configure**.
 5. On the File menu, point to **Create/Update** and click **Create JAM, SVF, or ISC File**. The **Create JAM, SVF, or ISC File** dialog box appears (Figure 7).
 6. Select the required file format (.jam or .svf), for all the .ekp files in the **File format** field.
 7. Type the filename in the **File name** field, or browse to and select the file.
 8. Click **OK** to generate the .jam or .svf file.

Figure 7. Create Multi-Device Key File



Step 2: Program the Volatile Key into the Cyclone III LS Device

Before programming the volatile key into the Cyclone III LS device, ensure that the FPGA can be configured successfully with an unencrypted configuration file. The volatile key is a reprogrammable and erasable key. Before you program the Cyclone III LS device with the volatile key, you must provide an external battery to retain the volatile key. Cyclone III LS devices with successful volatile key programmed can accept both encrypted and unencrypted configuration bitstreams. This enables the use of unencrypted configuration bitstreams for board-level testing.

Any attempt to configure a Cyclone III LS device containing the volatile key with a configuration file encrypted with the wrong key causes the configuration to fail. If this occurs, the nSTATUS signal from the FPGA pulses low and continues to reset itself.

You can program the volatile key into the Cyclone III LS device using on-board prototyping tools listed in [Table 2 on page 2](#).

Volatile Key Programming Using the Quartus II Software

You can perform the volatile key programming using a programming cable (EthernetBlaster, ByteBlaster II, or USB-Blaster) and the Quartus II software. Connect the programming cable to the programming cable header as shown in [Figure 8](#).


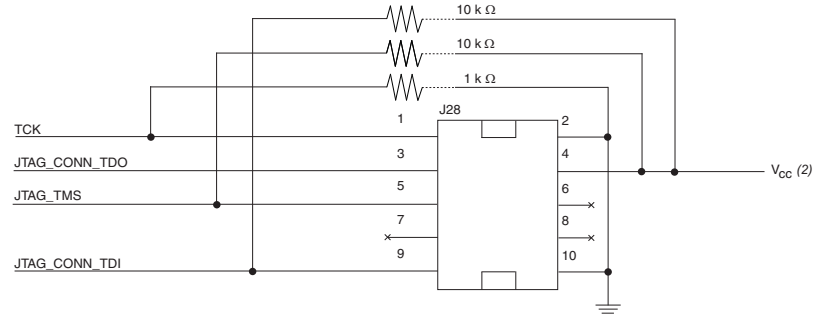

 For more information about connecting the programming cable, refer to the respective programming cable user guide in the [Programming Cables](#) web page.


Figure 8. Programming Cable Header *(Note 1)*



Notes to Figure 8:

- (1) The EthernetBlaster, ByteBlaster II, and USB-Blaster headers are identical for security key programming.
- (2) This voltage refers to V_{CCA} when the JTAG pins are powered up at 2.5/3.0/3.3-V V_{CCIO} , and refers to V_{CCIO} when the JTAG pins are powered up at 1.5/1.8-V V_{CCIO} .

 To perform volatile key programming using the Quartus II software through the EthernetBlaster, you must check the firmware version of the EthernetBlaster. Verify that the JTAG firmware build number is 101 or greater. If the version precedes build number 101, apply the firmware upgrade ([EBFW100101.tar.gz](#)).

 For more information about firmware upgrade instructions, refer to the [EthernetBlaster Communications Cable User Guide](#).

How to Perform Single-Device or Multi-Device Volatile Key Programming Using the Quartus II Software

To perform single-device volatile key programming using the Quartus II software, perform the following steps:

1. In the Tools menu, click **Programmer**. The programmer window appears (Figure 9 on page 14).
2. In the **Mode** list, select **JTAG** as the programming mode (Figure 9 on page 14).
3. Click **Hardware Setup**. The **Hardware Setup** dialog box appears.
 - a. In the currently selected hardware list, select the programming cable that is currently used.
 - b. Click **Done**.
4. Click **Add File**. The **Select Programmer File** dialog box appears.
 - a. Single-device key programming using an **.ekp** file:
 - i. Type *<filename>.ekp* in the **File name** field.
 - ii. Click **Open**.
 - iii. Highlight the **.ekp** file you added and click **Program/Configure** (Figure 9 on page 14).
 - b. Multi-device key programming using **.ekp** files:
 - i. Type *<filename>.ekp* in the **File name** field.
 - ii. Click **Open**.
 - iii. Repeat Steps **i – ii** for the number of devices in the same chain.
 - iv. Highlight the **.ekp** files you added and click **Program/Configure** (Figure 10 on page 14).
 - c. Multi-device key programming using a **.jam** file:
 - i. Type *<filename>.jam* in the **File name** field.
 - ii. Click **Open**.
 - iii. Highlight the **.jam** file you added and click **Program/Configure** (Figure 11 on page 15).
5. Click **Start** to program the key. The Quartus II software message window provides information about the success or failure of the key programming operation.



For the correct sequence of the devices in the same JTAG chain, you can use the **Auto-Detect** option in the Quartus II programmer.



Unlike the Stratix® III, Stratix IV, and Arria® II GX devices, you do not need to turn on the **Configure volatile design security key** in the **Programmer Options** dialog box to perform the Cyclone III LS devices volatile key programming. This option is ignored in Cyclone III LS devices.

Figure 9. Single-Device Key Programming using .ekp File

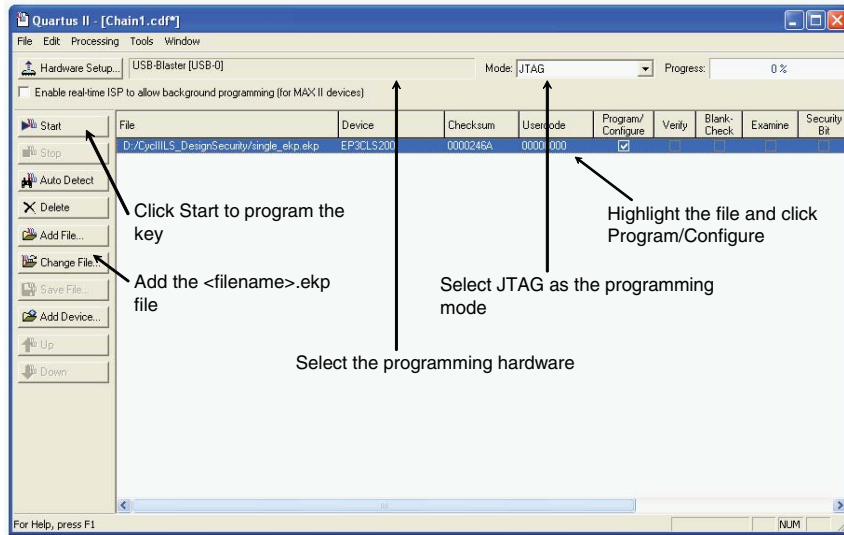


Figure 10. Multi-Device Key Programming using .ekp Files

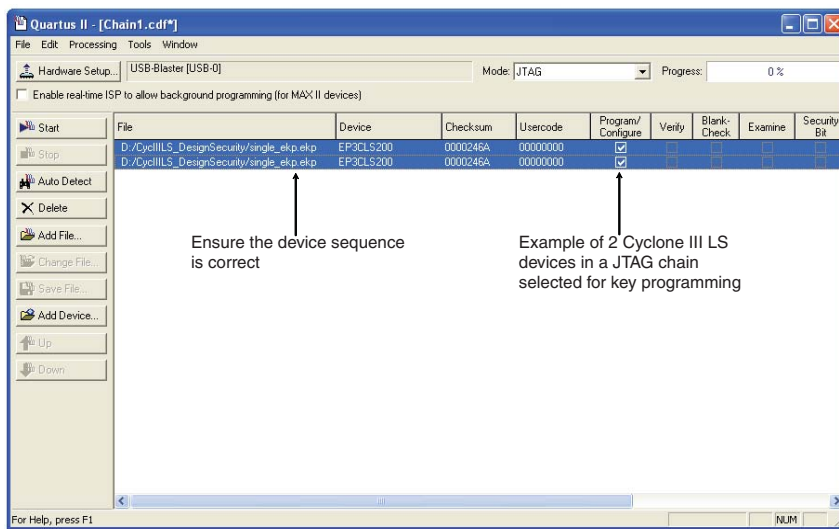
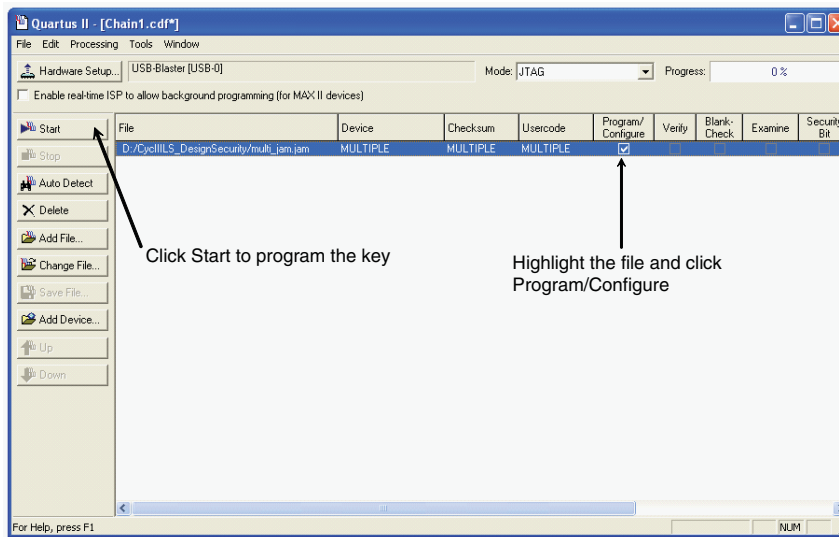


Figure 11. Multi-Device Key Programming using .jam Files





How to Perform Single-Device or Multi-Device Volatile Key Programming Using the Quartus II Software with Command-Line Interface

To perform single-device volatile key programming using the Quartus II software command-line interface, perform the following steps:

1. Open the command line prompt window.
2. To determine the programming cable port number that is connected to the JTAG server, type `quartus_jli -n` at the command line prompt.
3. With the **.jam** file generated in “[Step 1: Generate the Key File and Encrypt Configuration File](#)” on page 5, perform volatile programming to a single-device or multi-device volatile key programming with the following command-line:
 - Single-device volatile key programming:
`quartus_jli -c<n> single_jam.jam -aKEY_CONFIGURE`
 - Multi-device volatile key programming:
`quartus_jli -c<n> multi_jam.jam -aKEY_CONFIGURE`

`<n>` is the port number returned with the `-n` option.

 The Quartus II software command-line executable provides information about the success or failure of the key programming operation.

 For more information about `quartus_jli` command line, refer to the “Using the Command-Line Executable in the Quartus II Software” section in [AN 425: Using Command-Line Jam STAPL Solution for Device Programming](#).

Volatile Key Programming Using JTAG Technologies

You can perform the security programming for your design using a .svf file and a JT 37xx boundary-scan controller in combination with a JT 2147 QuadPod system.

- For more information about procedures for JTAG programming, refer to the JTAG technologies website at www.jtag.com.

Step 3: Configure the Cyclone III LS Device with Encrypted Configuration File

To configure the protected Cyclone III LS device with the encrypted configuration file, the encrypted configuration data is sent to the Cyclone III LS device. The FPGA decrypts the configuration data with the previously stored security key and uses the unencrypted data to configure itself. Only configuration files encrypted using the correct security key are accepted by the FPGA for successful configuration. A stolen encrypted file is useless without a correct security key.

Security Mode Verification

Cyclone III LS devices support the KEY_VERIFY JTAG instruction (Table 5) that allows you to verify the existing security mode of the device. You can use .jam files to automate the security mode verification steps to check if the volatile key has been successfully programmed.

- For more information about the available security modes in Cyclone III LS devices, refer to the “Available Security Modes” section in the *Configuration, Design Security, and Remote System Upgrades in Cyclone III Devices* chapter in volume 1 of the *Cyclone III Device Handbook*.

Table 5. KEY_VERIFY JTAG Instruction

JTAG Instruction	Instruction Code	Description
KEY_VERIFY	00 0001 0011	Connects the key verification scan register between TDI and TDO

- You must issue the FACTORY instruction to enable the access of non-mandatory JTAG instructions before you issue the KEY_VERIFY instruction. For more information regarding the FACTORY instruction, refer to *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Cyclone III Devices* chapter in volume 1 of the *Cyclone III Device Handbook*.

The key verification scan register in the Cyclone III LS device is powered by V_{CCBAT} and contains only one volatile key verify bit that indicates if a 256-bit AES key has been programmed to the device. This bit is set after the 256-bit AES key is successfully programmed. By reading out the content of the volatile key verify bit, the security mode of the device can be verified as defined in Table 6.

Table 6. Volatile Key Verify Bit

Volatile Key Verify Bit	Security Mode	Description
0	No Key Operation	Indicates that there is no key programmed in the Cyclone III LS device. In this mode, only unencrypted configuration bitstreams are allowed to configure the device.
1	Volatile Key	Indicates that the Cyclone III LS device is key programmed. This mode accepts both encrypted and unencrypted configuration bitstreams.

Example 4 shows a .jam file used to execute the KEY_VERIFY JTAG instruction to verify the security mode of a single Cyclone III LS device.

Example 4. Example of .jam File to Verify the Cyclone III LS Security Mode

```
'Key Verification in JAM format
BOOLEAN verify_reg;

IRSCAN 10, $013;
WAIT 100 USEC;
DRSCAN 1, $0, CAPTURE verify_reg;

PRINT "Security Mode Verification for Single Cyclone III
LS Device ";
IF (INT(verify_reg) == 0) THEN PRINT "Security Mode: No
Key Operation";
IF (INT(verify_reg) == 1) THEN PRINT "Security Mode:
Volatile Key";
```

Supported Configuration Scheme

The design security feature is available in all configuration methods except in JTAG-based configuration. You can use the design security feature in fast passive parallel (FPP) mode (when using an external controller, such as a MAX[®] II device or a microprocessor and a flash memory), or in AS and PS configuration schemes.

Table 7 summarizes the configuration schemes in Cyclone III LS devices that support the design security feature.

Table 7. Security Configuration Schemes

Configuration Scheme	Configuration Method	Design Security
FPP	MAX II device or microprocessor and flash memory	✓ (1)
AS	Serial configuration device	✓
PS	MAX II device or microprocessor and flash memory	✓
	Download cable	✓ (2)
JTAG	Download cable	— (3)

Notes to Table 7:

- (1) The host system must send a DCLK signal that is 4x the data rate.
- (2) The design security feature is not supported using a .sof file.
- (3) Volatile key programming only.



The decompression feature is not supported when the design security feature is enabled. However, you can use the design security feature with remote system upgrade features.



You must specify the configuration schemes used by setting the MSEL [3 . . 0] pin settings on the Cyclone III LS device. For more information on the MSEL pins setting, refer to the “Configuration Scheme” section in the *Configuration, Design Security, and Remote System Upgrades in Cyclone III Devices* chapter in volume 1 of the *Cyclone III Device Handbook*.

You can either perform a boundary-scan test (BST) or use the SignalTap[®] II logic analyzer to analyze functional data in the Cyclone III LS device. When using the SignalTap II logic analyzer, you must first configure the device with an encrypted configuration file using the PS, FPP, or AS configuration mode. Your design must contain at least one instance of the SignalTap II logic analyzer. After the SignalTap II logic analyzer window is opened in the Quartus II software, scan the devices chain and the SignalTap II logic analyzer is now ready to acquire data over the JTAG interface.



You must issue the FACTORY instruction to enable the access of non-mandatory JTAG instructions before you use the SignalTap II logic analyzer to acquire data over the JTAG interface. For more information regarding the FACTORY instruction, refer to *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Cyclone III Devices* chapter in volume 1 of the *Cyclone III Device Handbook*.

Serial FlashLoader Support with Encryption Enabled

The SerialFlash Loader (SFL) is an in-system programming solution for serial configuration devices. You can instantiate the SFL block into your design and have the flexibility to update your design stored in the serial configuration device without reprogramming the configuration device via the AS interface.

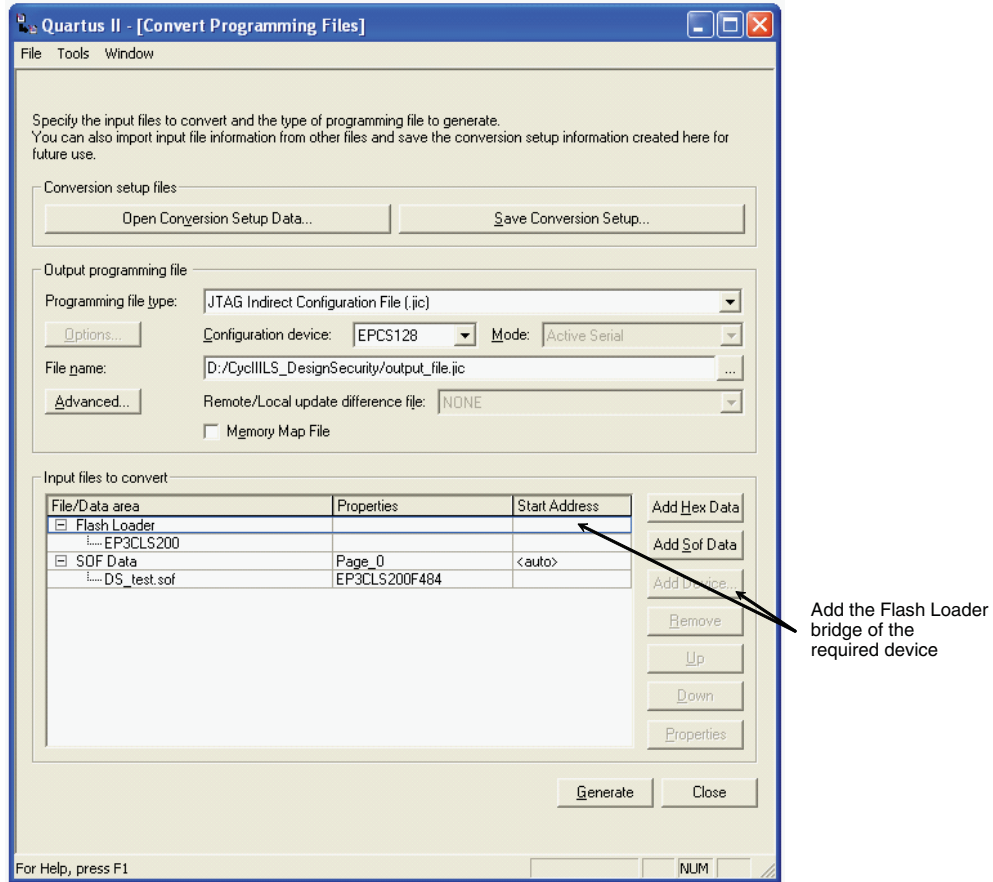


For more information about instantiating the SFL megafunction, refer to the “Instantiating SFL Megafunction in the Quartus II Software” section in *AN 370: Using the Serial FlashLoader With the Quartus II Software*.


You can use the SFL solution for your application as long as the JTAG interface of the FPGA is accessible. To use the SFL megafunction with the encryption feature enabled in a single FPGA device chain, perform the following steps:

1. Instantiate the SFL megafunction in the Cyclone III LS device top-level design.
2. Compile your design with one of the following options. An unencrypted **.sof** is generated.
 - On the Processing menu, click **Start Compilation**
 - On the Processing menu, point to **Start** and click **Start Assembler**
3. Follow these steps to convert a **.sof** to a **.jic** file:
 - a. On the File menu, click **Convert Programming Files**. The **Convert Programming Files** dialog box appears (Figure 12).
 - b. In the **Convert Programming Files** dialog box, select the **JTAG Indirect Configuration File (.jic)** from the **Programming file type** list.
 - c. Select the serial configuration device from the **Configuration device** list.
 - d. Type the filename in the **File name** field, or browse to and select the file.
 - e. Under the **Input files to convert** section, click **SOF Data**.
 - f. Click **Add File** to open the **Select Input File** dialog box.
 - g. Browse to the unencrypted **.sof** file and click **Open**.
 - h. Under the **Input files to convert** section, click on the **.sof** filename. The field is highlighted. Encrypt the **.sof** file by following Step 2 in “How to Generate the Single-Device .ekp Key File and Encrypt the Configuration File using the Quartus II Software” on page 6.
 - i. Click **Flash Loader** and then click **Add Device** (Figure 12). The **Select Devices** page appears.
 - j. Select the target device that you are using to program the serial configuration device. Click **OK**.
 - k. Click **Generate** to generate the **.jic** file.

Figure 12. Create a .jic File



4. Program the serial configuration device with the encrypted .jic file.

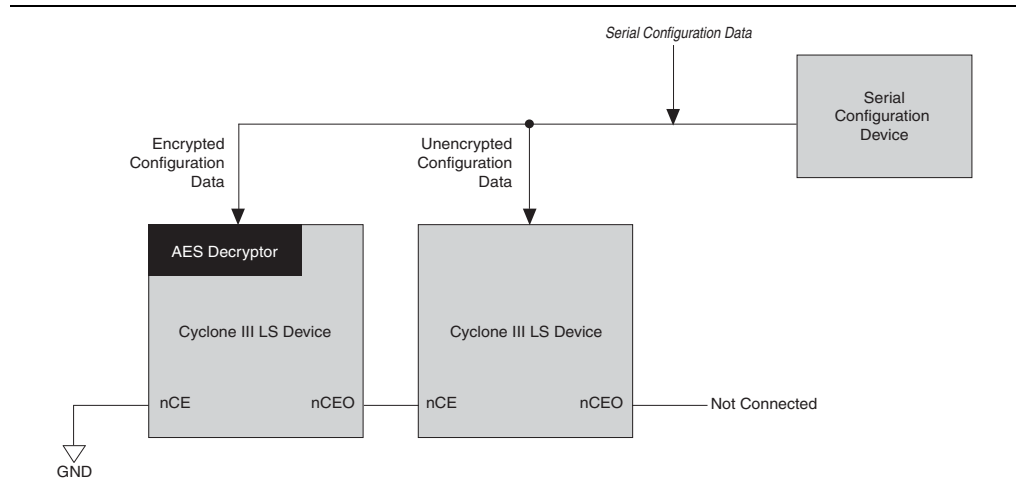
 For more information about programming the serial configuration device or devices with the .jic file that you have created, refer to the procedure in the “Programming Serial Configuration Devices Using the Quartus II Programmer and .jic Files” section in *AN 370: Using the Serial FlashLoader With the Quartus II Software*.

5. Follow the steps in “How to Perform Single-Device or Multi-Device Volatile Key Programming Using the Quartus II Software” on page 13 to program the key to the Cyclone III LS device.
6. The encrypted Cyclone III LS device is configured by the programmed serial configuration device.

Considerations When Choosing a Configuration Scheme

As shown in [Figure 13](#), in a serial configuration scheme (AS or PS), you can cascade a series of Cyclone III LS devices that accept encrypted data using the design security feature along with devices that do not use the design security feature, such as other Altera devices that accept unencrypted data in the same configuration chain.

Figure 13. Multi-Device Cyclone III LS Active Serial Configuration



When using FPP, the design security feature must be either enabled or disabled for all Cyclone III LS devices in the chain. The design security feature cannot be selectively enabled for individual devices in the chain because of the relationship between the DATA and DCLK signal, which is discussed in [“DCLK Considerations When Using the FPP Configuration Scheme”](#) on page 21. If the chain contains devices that do not support the design security feature, Altera recommends that you use a serial configuration scheme.

However, the DATA and DCLK signal relationship in FPP mode when using the decompression feature is the same as when using the design security feature. If all the devices in the chain use the decompression feature, the design security feature can be selectively enabled or disabled for each device.

DCLK Considerations When Using the FPP Configuration Scheme

The FPP configuration scheme with the design security feature enabled requires that an external host be used, such as a MAX II device or a microprocessor to control the flow of the DATA and DCLK signal to Cyclone III LS devices.

DCLK is the clock source that is used to clock the configuration process. Configuration data is received on the DATA [7 . . 0] pins. If you are using the Cyclone III LS design security feature with the FPP configuration scheme, the external host must be able to send a DCLK frequency that is 4x the data rate.

The 4x DCLK signal does not require an additional pin and is sent on the DCLK pin. The maximum DCLK frequency supported in Cyclone III LS devices is 100 MHz and produces a maximum data rate shown in [Equation 1](#):

Equation 1. Maximum DCLK Frequency

$$\frac{100 \text{ MHz}}{4} \times 8 \text{ bits} = 200 \text{ Mbps}$$

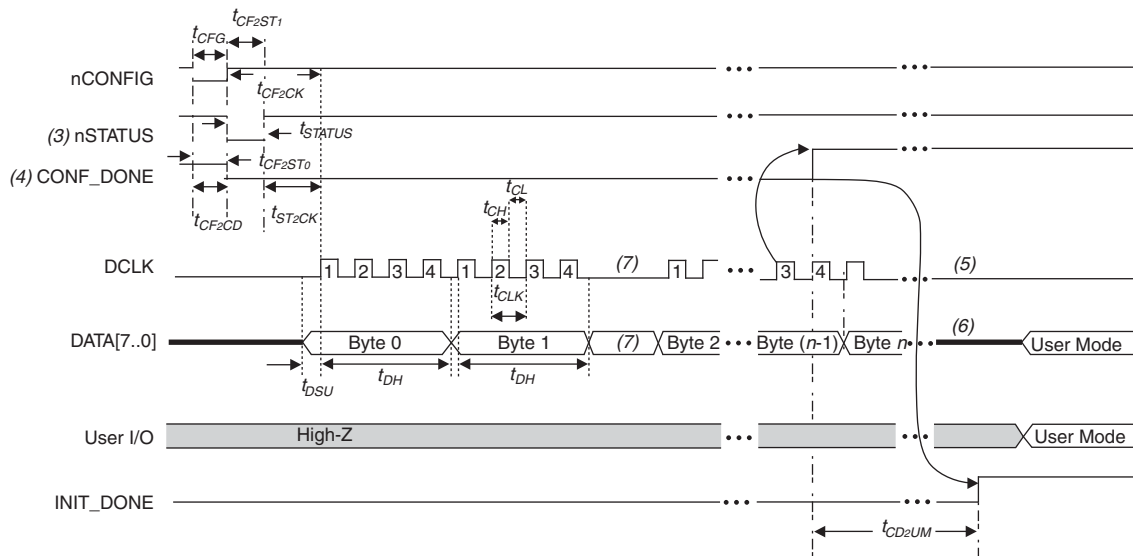
When using the design security feature, the first configuration data byte is latched on the Cyclone III LS device on the first DCLK rising edge. Subsequent data bytes are latched four clock cycles after each previous data byte. The DCLK speed must be below the frequency specified in [Equation 1](#) to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

If you must stop DCLK while using the design security feature with the FPP configuration scheme, it can only be stopped three clock cycles after the last data byte was latched on the Cyclone III LS device. This gives the configuration circuit enough clock cycles to process the last byte of latched configuration data. When the clock restarts, data must be present on the DATA [7 . . 0] pins prior to sending the first DCLK rising edge.

Timing Waveform with Design Security Feature Enabled

[Figure 14](#) shows the timing waveform for the FPP configuration that uses a MAX II device or a microprocessor as an external host. This waveform shows the timing when the design security feature is enabled.

Figure 14. FPP Configuration Timing Waveform with the Design Security Feature Enabled (Note 1), (2)



Notes to Figure 14:

- (1) This timing waveform should be used when the design security feature is used.
- (2) The beginning of this waveform shows the device in user mode. In user mode, nCONFIG, nSTATUS, and CONF_DONE are at logic-high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
- (3) Upon power-up, Cyclone III LS devices hold nSTATUS low during the POR delay.
- (4) Upon power-up, before and during configuration, CONF_DONE is low.
- (5) DCLK should not be left floating after configuration. It should be driven high or low, whichever is more convenient.
- (6) DATA [7 . . 0] pins are available as user I/O pins after configuration, and the state of these pins depends on the dual-purpose pin settings.
- (7) If needed, DCLK can be paused. When DCLK restarts, the external host must provide data on the DATA [7 . . 0] pins prior to sending the first DCLK rising edge.

Table 8 shows the FPP timing parameters for Cyclone III LS devices with the design security feature enabled.

Table 8. FPP Timing Parameters for Cyclone III LS with the Design Security Feature Enabled (Note 1), (3)

Symbol	Parameter	Minimum	Maximum	Units
t_{CF2CD}	nCONFIG low to CONF_DONE Low	—	500	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—	500	ns
t_{CFG}	nCONFIG low pulse width	500	—	ns
t_{STATUS}	nSTATUS low pulse width	45	685 (2)	μ s
t_{CF2ST1}	nCONFIG high to nSTATUS high	—	685 (2)	μ s
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	685 (2)	—	μ s
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2	—	μ s
t_{DSU}	Data setup time before rising edge on DCLK	5	—	ns
t_{DH}	Data hold time after rising edge on DCLK	22.5	—	ns
t_{CH}	DCLK high time	3.2	—	ns
t_{CL}	DCLK low time	3.2	—	ns
t_{CLK}	DCLK period	7.5	—	ns
f_{MAX}	DCLK frequency	—	100	MHz
t_{CD2UM}	CONF_DONE high to user mode (3)	300	650	μ s
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 x maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	$t_{CD2CU} + (3,192 \times \text{CLKUSR period})$	—	—

Notes to Table 8:

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

US Export Controls

The US export controls for the Cyclone III LS devices are generally classified under the US Export Control Classification Numbers (ECCN) 3A001.a.7 or 3A991.d. Although Cyclone III LS devices perform decryption, the export control classification of the devices does not change as the decryption capability is only used to protect the configuration bitstream. Altera's Quartus II software development tools (version 9.0 or later), which encrypt the configuration bitstream, are formally classified under US ECCN 5D002 c.1 and subject to export under license exception ENC as a "retail" commodity to most countries. You may contact opexp_imp@altera.com for any export-related questions.

Document Revision History

Table 10 shows the revision history for this document.

Table 10. Document Revision History

Date and Revision	Changes Made	Summary of Changes
September 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. RSDS and PPDS are registered trademarks of National Semiconductor. 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.

