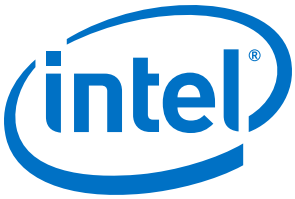


Generic Flash Programmer User Guide

Intel® Quartus® Prime Pro Edition

Updated for Intel® Quartus® Prime Design Suite: **19.4**

Subscribe
Send Feedback

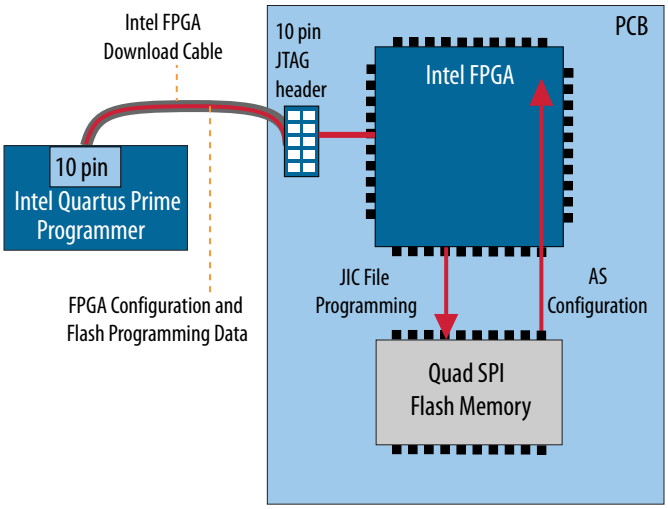
UG-20227 | 2019.12.16
Latest document on the web: PDF | HTML

Contents

1. Generic Flash Programmer User Guide Intel® Quartus® Prime Pro Edition

This document describes how to use the Generic Flash Programmer. You can use the Generic Flash Programmer to load an FPGA configuration bitstream file into a Quad SPI flash memory device. The Quad SPI flash memory device subsequently loads the configuration data into the target FPGA via Active Serial (AS) configuration. You can optionally enable bitstream compression and encryption security to reduce the size and protect the configuration bitstream files.

Figure 1. Generic Flash Programmer Configuration Example



The Generic Flash Programmer allows you to send configuration data over a download cable via a JTAG connection to the target FPGA device. The target FPGA then in turn writes the configuration data to the flash memory device. The AS configuration scheme loads the configuration data from the flash memory into the FPGA. For example, this method allows you to configure or reconfigure the FPGA from the flash memory after restoring power to the FPGA after power down.

Related Information

- Intel® Arria® 10 Core Fabric and General Purpose I/Os Handbook
- Configuration, Design Security, and Remote System Upgrades in Stratix V Devices
- Intel® Stratix® 10 Configuration User Guide
- Intel® Quartus® Prime Pro Edition User Guide: Programmer

1.1. Supported Devices and Configuration Methods

The Intel® Quartus® Prime Pro Edition Generic Flash Programmer supports the following FPGA and flash memory devices and configuration methods.

Table 1. Generic Flash Programmer FPGA and Configuration Method Support

Supported Methods	Intel Agilex™/Intel Stratix® 10 Designs	Intel Arria® 10/Intel Cyclone® 10 GX Designs
Secondary Programming File Generation	• **Programming File Generator** (recommended) • **Convert Programming File** dialog box (Intel Stratix 10 only)	**Convert Programming File** dialog box
Supported Input Files	• SRAM Object File (`.sof`) • Partial Masked SOF File (`.pmsf`) • Partial Reconfiguration Raw Binary File (`.rbf`)	SRAM Object File (`.sof`)
Supported Flash Programming Output File	JTAG Indirect Configuration File (`.jic`)	JTAG Indirect Configuration File (`.jic`)
Supported Configuration Schemes	**Active Serial x4**	• **Active Serial x4** • **Active Serial**

1.2. Quad SPI Flash Byte-Addressing

Quad SPI flash devices typically support either 3-byte addressing, 4-byte addressing, or both for programming operations. You can only configure Intel FPGAs with a flash memory device with byte addressing that is compatible with the Intel FPGA that you plan to configure.

The following table specifies the byte-addressing compatibility of Intel FPGAs for supported flash memory devices:

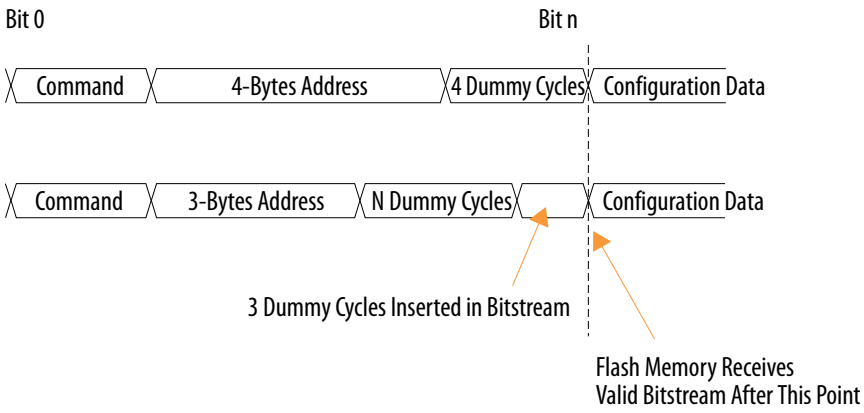
Table 2. Intel FPGA Required Flash Memory Byte Addressing

FPGA Devices	Required Flash Memory Byte Addressing
Intel Agilex devices	4-byte addressing
Intel Stratix 10 devices	4-byte addressing
Intel Cyclone 10 GX devices	4-byte addressing
Intel Arria 10 devices	4-byte addressing

Adding Dummy Clock Cycles

Flash memory devices must read either a 24-bit (3-byte) address, or 32-bit (4-byte) address before the flash device can start receiving data to write to the flash memory, or before outputting the data after the flash memory device receives a read command. Therefore, you must specify (or select a flash memory template that specifies) an appropriate dummy clock cycle value for the flash memory device, as Defining a New Flash Memory Configuration Device on page 14 describes.

Figure 2. Reading Configuration Data from Flash (3-Byte and 4-Byte Addressing)



1.3. Generic Flash Programmer Operation

The Generic Flash Programmer facilitates the generation of appropriate secondary programming files, with definition of the connected flash memory device, and the corresponding Program, Erase, Verify, Blank-Check, and Examine flows. You can then add the generated programming files to the Intel Quartus Prime Programmer for correct flash programming implementation in hardware.

You can access settings and controls for the Generic Flash Programmer from the Programmer, **Programming File Generator** (Intel Agilex and Intel Stratix 10 designs), or **Convert Programming File** dialog box (Intel Arria 10 and Intel Cyclone 10 GX designs).

These user interfaces allow you to generate required programming files, define the flash programming device, and implement your flash programming flow.

The generic flash programming flow for the **Programming File Generator** varies from the **Convert Programming File** flow. Refer to the following configuration steps for the method that you use.

Related Information

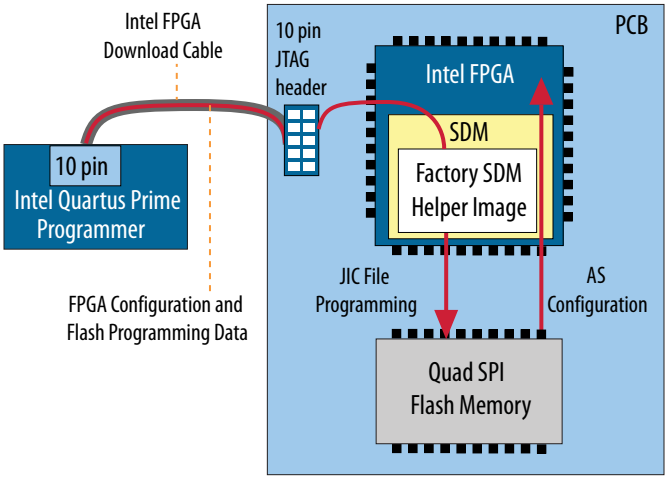
- Generic Flash Programming (Programming File Generator) on page 6
- Generic Flash Programming (Convert Programming File Dialog Box) on page 18

1.3.1. Generic Flash Programming (Programming File Generator)

You can use the **Programming File Generator** to generate the necessary secondary programming files for flash programming with Intel Agilex and Intel Stratix 10 devices.

The `.jic` secondary programming file allows you to perform JTAG programming via the FPGA Secure Device Manager (SDM) and Factory SDM helper image. The SDM controls secure access to the FPGA, and the Factory SDM helper image enables communication between the JTAG connection and the flash memory serial interface. After generating the files, you use the Intel Quartus Prime Programmer to program the flash, which in turn configures the FPGA via AS configuration.

Figure 3. Intel Stratix 10 Flash Programming Configuration



Generic Flash Programmer operation includes the following high level steps that this section describes in detail:

1. Step 1: Generate Primary Device Programming File on page 6—use the Intel Quartus Prime Assembler to generate the `.sof` FPGA configuration file.
2. Step 2: Generate Secondary Programming Files (Programming File Generator) on page 7—use the **Programming File Generator** to generate the `.jic` that you program into your flash memory device to store `.sof` configuration data.
3. Step 3: Program the Flash Memory Device on page 16—use the Intel Quartus Prime Programmer and connected Intel FPGA download cable to program the `.jic` configuration data into the flash memory device and the `.sof` into the FPGA via Active Serial JTAG configuration.

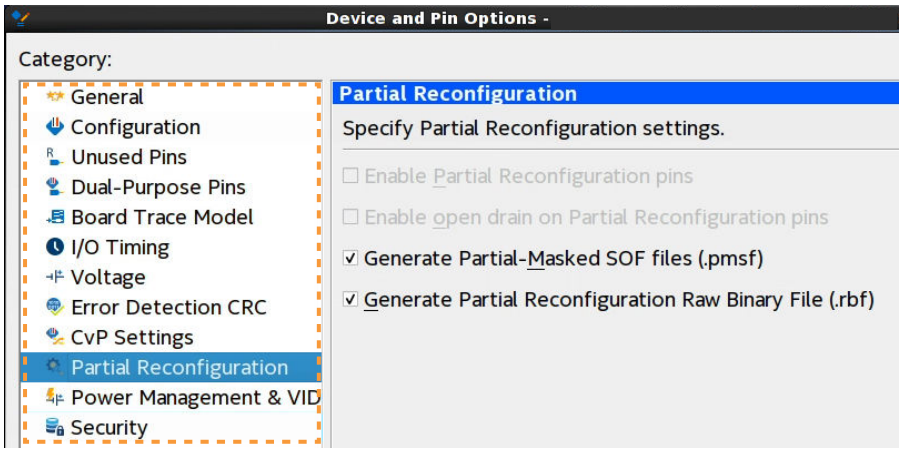
1.3.1.1. Step 1: Generate Primary Device Programming File

The Intel Quartus Prime Assembler generates the `.sof` FPGA configuration file once design compilation is complete. Prior to running the Assembler, you can specify device and pin options that impact the `.sof` and subsequent `.jic` file generation.

Follow these steps to generate a `.sof` for use in generic flash programming:

1. Before running the Assembler, click **Assignments ➤ Device ➤ Device & Pin Options** to specify options for FPGA configuration pins and other hardware settings that the `.sof` preserves. The following options are particularly relevant to generic flash programming. For option descriptions, refer to Device and Pin Options on page 40.
   - **General** tab—specify JTAG user code and configuration clock source.
   - **Configuration** tab—specify **Active Serial x4** and appropriate **Configuration Pin Options** for the FPGA. Select **Auto** for **Configuration device voltage**, which you specify with precision at a later time.
   - **Security**—specify settings to enable optional authentication and encryption of the configuration bitstream file.(1)

Figure 4. Device & Pin Options Dialog Box (Intel Stratix 10 Design)



2. To generate primary device programming files, click **Processing ➤ Start ➤ Start Assembler**, or double-click **Assembler** on the Compilation Dashboard. The Compiler confirms that prerequisite modules are complete, and launches the Assembler to generate the programming files.

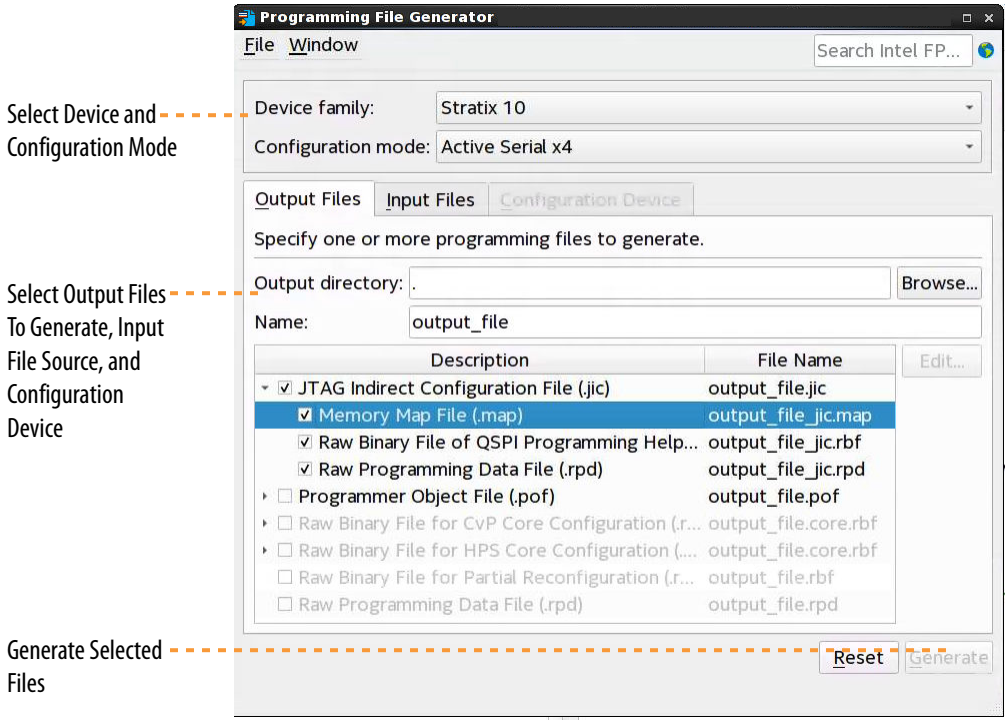
1.3.1.2. Step 2: Generate Secondary Programming Files (Programming File Generator)

You can use the **Programming File Generator** to generate secondary programming files for alternative device programming methods, such as the `.jic` for flash programming, `.rbf` for partial reconfiguration, or `.rpd` for third-party programmer configuration.

The options available in the **Programming File Generator** change dynamically, according to your device and configuration mode selection.

(1) Security options not yet available for Intel Agilex devices.

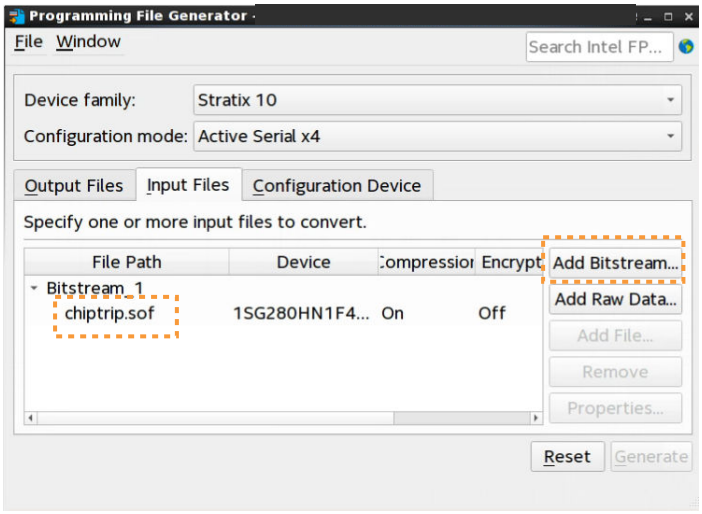
Figure 5. Programming File Generator



1. Generate the primary programming files for your design, as Step 1: Generate Primary Device Programming File on page 6 describes.
2. Click **File ➤ Programming File Generator**.
3. For **Device family**, select your target device.
4. For **Configuration mode**, select **Active Serial x4**.
5. On the **Output Files** tab, enable the checkbox for generation of the **JTAG Indirect Configuration File (.jic)**. The **Input Files** tab is now available.
6. Specify the **Output directory** and **Name** for the `.jic` file you generate.
7. On the **Input Files** tab, click **Add Bitstream** and specify the `.sof` file that contains the configuration bitstream data. To include raw data, click **Add Raw Data** and specify a Hexadecimal (Intel-Format) File (`.hex`) or Binary (`.bin`) file. To enable bitstream signing or encryption security settings, select the `.sof` file and click **Properties**, as Enabling Bitstream Authentication (Programming File Generator) on page 10 describes.(2)

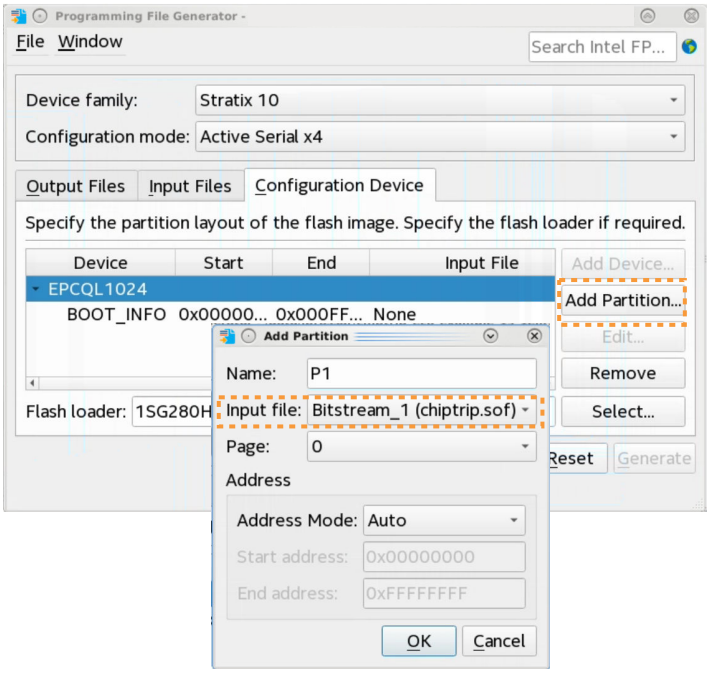
(2) Security options not yet available for Intel Agilex devices.

Figure 6. Input Files Tab



8. On the **Configuration Device** tab, click **Add Partition** to specify the `.sof` file that occupies the flash memory partition, as Add Partition Dialog Box (Programming File Generator) on page 46 describes.

Figure 7. Add Flash Partition



9. On the **Configuration Device** tab, click **Add Device** to select a supported flash memory device and predefined programming flow. When you select a predefined device, you cannot modify any setting. Alternatively, click **<<new device>>** to define a new flash memory device and programming flow, as Defining a New Flash Memory Configuration Device on page 14 describes.
10. Click the **Select** button for **Flash Loader** and select the device that controls loading of the flash memory device.
11. After you specify all options in **Programming File Generator**, the **Generate** button enables. Click **Generate** to create the files.

Related Information

- Input Files Tab Settings (Programming File Generator) on page 45
- Output Files Tab Settings (Programming File Generator) on page 45

1.3.1.2.1. Enabling Bitstream Authentication (Programming File Generator)

Bitstream authentication requires that you generate a first level signature chain (`.qky`) that includes the root key and one or more design signing keys. The root key enables the base security features and authenticates the design signing key through the public signature chain. The root key stores the SHA-256 or SHA-384 hash of the key in eFuses. You can also optionally enable firmware co-signature capability to require signing the version of configuration firmware that runs on your device. The FPGA device then can only load authenticated firmware.

Note: Refer to the *Intel Stratix 10 Device Security User Guide* for step-by-step first level signature chain key generation instructions.(3)

After you specify the `.qky` in Assembler settings, the Assembler appends the first level signature chain to the configuration `.sof` that you generate.

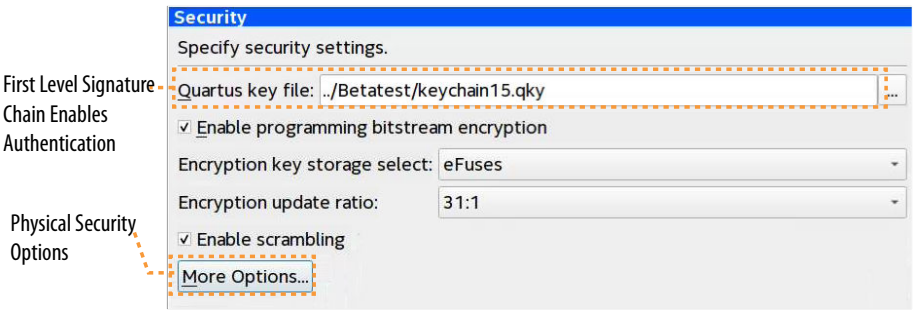
Use the **Programming File Generator** to generate the signed configuration bitstream for an `.sof` file. The JTAG Indirect Configuration File (`.jic`) and Raw Programming Data File (`.rpd`) formats are available for Active Serial (AS) configuration. The Programmer Object File (`.pof`) and Raw Binary File (`.rbf`) are available for Avalon® Streaming configuration.

Follow these steps to enable bitstream authentication:

1. Generate a first level signature chain (`.qky`) that includes the root key and one or more design signing keys, as *Intel Stratix 10 Device Security User Guide* describes.
2. To add the first level signature chain to a configuration bitstream, click **Assignments ➤ Device ➤ Device and Pin Options ➤ Security**, and then specify the first level signature chain `.qky` for the **Quartus key file** option.
3. To enable more physical device security options, click the **More Options** button on the **Security** page. More Security Options Dialog Box on page 44 describes all options.

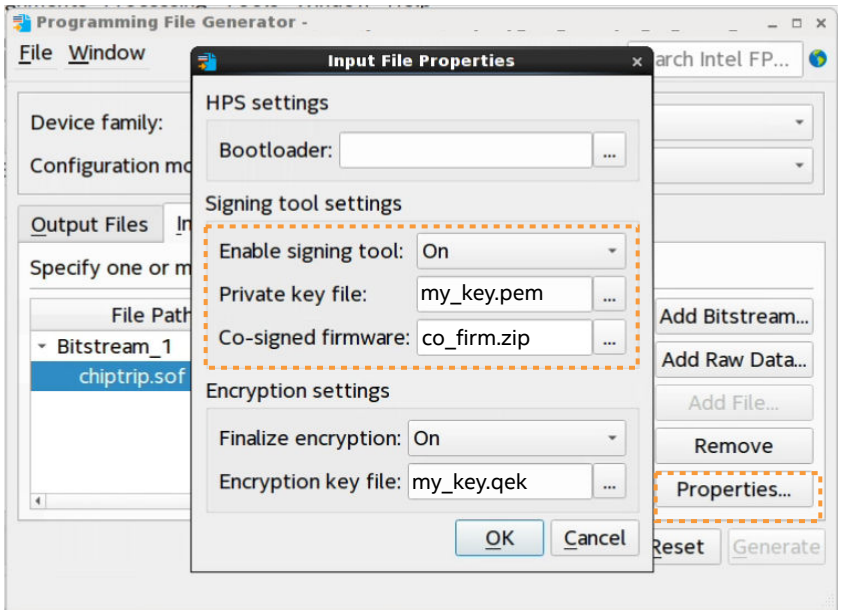
(3) Security options not yet available for Intel Agilex devices.

Figure 8. Security Tab (Device and Pin Options)



4. Generate primary device programing files in the Assembler, as Step 1: Generate Primary Device Programming File on page 6 describes. The primary device programming file now contains data to enable first level authentication.
5. To optionally enable co-signing device firmware authentication, generate a `.jic` or `.rbf` secondary programming file with the following options, as Step 2: Generate Secondary Programming Files (Programming File Generator) on page 7 describes:
   a. In **Programming File Generator**, click the **Properties** button. The **Input File Properties** dialog box appears.

Figure 9. Enabling Co-Signing Device Firmware Authentication (Intel Stratix 10 Devices)



   b. Set **Enable signing tool** to **On**.
   c. For **Private key file**, specify a design signing key Privacy Enhanced Mail Certificates file (`.pem`) for firmware co-signing. This key can be separate from the FPGA design signing key.
   d. For **Co-signed firmware**, specify a Quartus Co-Signed Firmware file (`.zip`).

e. Click **OK**.

6. Use the Programmer to configure the device with the `.jic` or `.rbf`.

Related Information

- Step 1: Generate Primary Device Programming File on page 6
- Step 2: Generate Secondary Programming Files (Programming File Generator) on page 7
- Intel Stratix 10 Device Security User Guide
  For detailed information on generating device security keys.

Specifying Additional Physical Security Settings (Programming File Generator)

Intel Stratix 10 devices can store security and other configuration settings in eFuses. You can enable additional physical security settings in eFuses to extend the level of device security protection.

To specify additional physical device security settings, follow these steps:

1. Click **Assignments ➤ Device ➤ Device and Pin Options ➤ Security**.
2. On the **Security** tab, specify the First Level Signature Chain `.qky` file that contains the root key and one or more design signing keys for the **Quartus key file** setting.
3. Click the **More Options** button and specify any of the following:

Figure 10. More Security Options Dialog Box

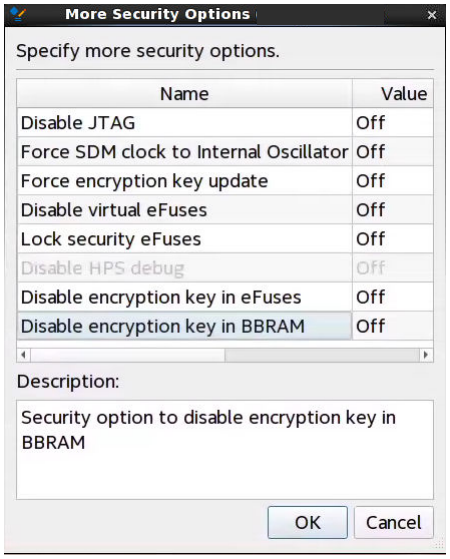


Table 3. More Security Options Dialog Box Settings

Option	Description	Values
Disable JTAG	Disables JTAG command and configuration of the device. Setting this eliminates JTAG as mode of attack, but also eliminates boundary scan functionality.	• **Off**—inactive • **On**—active until wipe of containing design • **On sticky**—active until next POR • **On check**—checks for corresponding blown fuse
Force SDM clock to internal oscillator	Disables an external clock source for the SDM. The SDM must use the internal oscillator. Using an internal oscillator is more secure than allowing an external clock source for configuration.	
Force encryption key update	Specifies that the encryption key must update by the frequency that you specify for the **Encryption update ratio** option. The default ration value is 31:1. Encryption supports up to 20 intermediate keys.	
Disable virtual eFuses	Disables the eFuse virtual programming capability.	
Lock security eFuses	Causes eFuse failure if the eFuse CRC does not match the calculated value.	
Disable HPS debug	Disables debugging through the JTAG interface to access the HPS.	
Disable encryption key in eFuses	Specifies that the device cannot use an AES key stored in eFuses. Rather, you can provides an extra level of security by storing the AES key in BBRAM.	
Disable encryption key in BBRAM	Specifies that the device cannot use AES key stored in BBRAM. Rather, you can provides an extra level of security when you store the AES key in eFuses.	

4. Click **OK**.

1.3.1.2.2. Enabling Bitstream Encryption (Programming File Generator)

To enable bitstream encryption, you must first generate a first level signature chain (`.qky`) that enables encryption options in the GUI. Next, you generate the encrypted configuration bitstream in the Assembler. Finally, you generate a secondary programming file that specifies the AES **Encryption Key file** (`.qek`) for bitstream decryption.

Follow these steps to enable bitstream encryption:

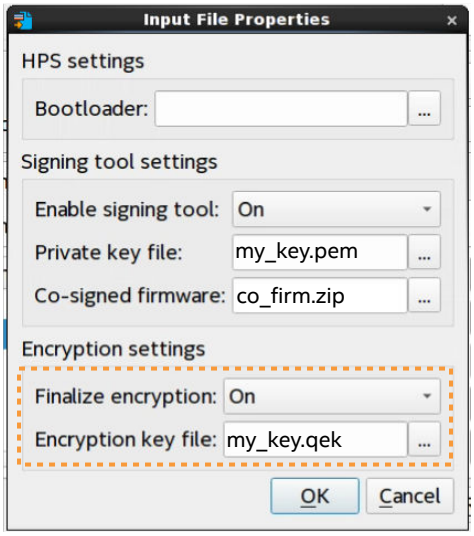
1. Generate a First Level Signature Chain that includes the root key and one or more design signing keys, as *Intel Stratix 10 Device Security User Guide* describes.
2. Click **Assignments ➤ Device ➤ Device and Pin Options ➤ Security**.
3. For the **Quartus key file** setting, specify the first level signature chain `.qky` that contains the root key and one or more design signing keys.
4. Turn on **Enable programming bitstream encryption**, and specify one or more of the following:

Table 4. Assembler Encryption Security Settings

Option	Description
Encryption key storage select	Specifies the location that stores the `.qek` key file. You can select either **Battery Backup RAM** or **eFuses** for storage.
Encryption update ratio	Specifies the ratio of configuration bits compared to the number of key updates required for bitstream decryption. You can select either **31:1** (the key must change 1 time every 31 bits) or **Disabled** (no update required). Encryption supports up to 20 intermediate keys.
Enable scrambling	Scrambles the configuration bitstream.
More Options	Opens the **More Security Options** dialog box for specifying additional physical security options.

5. Generate primary device programing files in the Assembler, as Step 1: Generate Primary Device Programming File on page 6 describes.
6. Generate a `.jic` or `.rbf` secondary programming file, as Step 2: Generate Secondary Programming Files (Programming File Generator) on page 7 describes:
   a. In the **Programming File Generator**, select the `.sof` file on the **Input Files** tab.
   b. Click the **Properties** button. The **Input File Properties** dialog box appears.

Figure 11. Input File Properties



   c. Set **Finalize encryption** to **On**.
   d. Specify the AES 256-bit or 384-bit **Encryption key file** (`.qek`) to decrypt the bitstream in the SDM prior to device configuration.
7. Click **OK**.

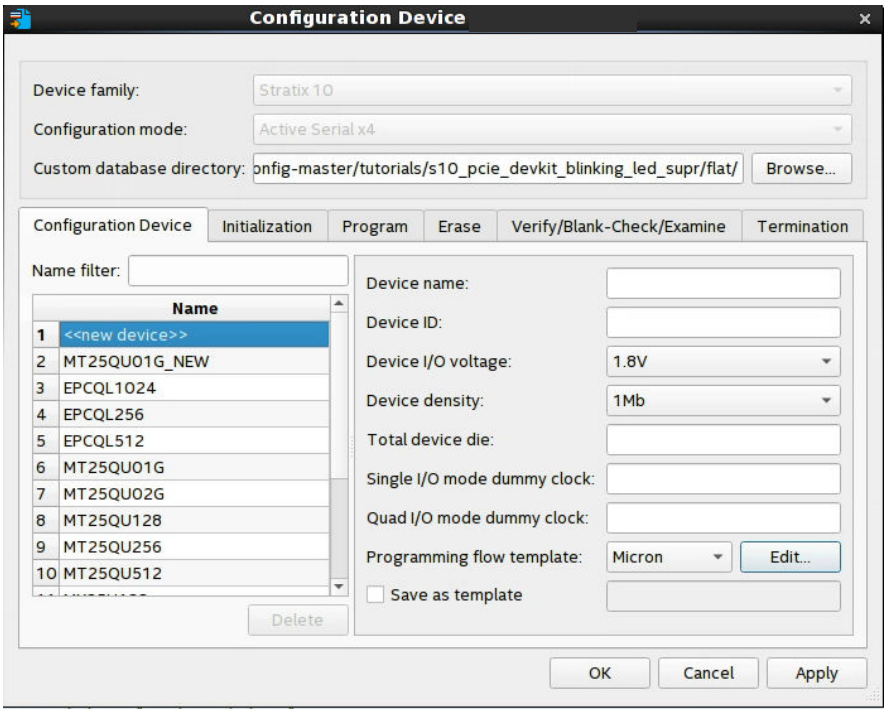
1.3.1.2.3. Defining a New Flash Memory Configuration Device

You can define and store the settings for a new flash memory device, based on the programming flow template of an existing supported flash memory device. You can customize and preserve the properties of the new flash memory device as a template for subsequent reuse, and define other flash memory devices based the template.

For Intel Agilex and Intel Stratix 10 devices, the Secure Device Manager (SDM) firmware controls the flash programming flows, and you cannot modify these flows. For Intel Arria 10 and Intel Cyclone 10 GX devices, you can modify and preserve the default programming flows, as Modifying Programming Flows on page 25 describes.

When you define a new flash memory device, Intel Quartus Prime software stores the collection of settings in an `.xml` file automatically in the **Custom database directory** location that you can specify.

Figure 12. Define New Flash Device (Intel Stratix 10 Example)



Follow these steps to define a new flash memory device:

1. Perform one of the following to generate a `.jic` file for flash programming:
   - Step 2: Generate Secondary Programming Files (Programming File Generator) on page 7
   - Step 2: Generate Secondary Programming Files (Convert Programming Files) on page 19
2. For the **Configuration Device** option, select **<<new device>>**. The settings on this and other tabs become available.
3. For **Programming flow template**, select an existing flash memory device template for the new device initial settings, or define a flash programming flow for a different flash memory vendor based on an existing template.
4. Specify the remaining settings on the **Configuration Device** tab:

Table 5. Configuration Device Tab Settings

Option	Description
Device name	Specify a unique name for the flash not already listed in the **Name** column. The **Name** must not contain any empty string (space) or special characters (except "_").
Device ID	Specify the 3-byte ID that the Programmer Auto-Detect operation uses to detect the flash programming device, such as `0x20 0xBB 0x21`.
Device I/O voltage	Specify **1.8V** or **3.0/3.3V** to match your memory device specification.
Device density	Select the total density that corresponds with your flash memory device size.
Total device die	Specify the total number of die for a stacked device (where applicable).
Single I/O mode dummy clock	Specify the Fast Read dummy clock cycle for flash device in single I/O protocol. The programming file generation uses this setting to determine if the configuration requires bit shifting to compensate for the actual dummy clock cycle during Active Serial configuration.
Quad I/O mode dummy clock	Specify the Fast Read dummy clock cycle for flash device in Quad I/O protocol. The programming file generation uses this setting to determine if the configuration requires bit shifting to compensate for the actual dummy clock cycle during Active Serial configuration.
Custom database directory	Specifies the location of the `.xml` file that preserves a flash memory device definition and flow information. The default location is the project directory or current working directory. *Note:* When you specify a non-default folder for the **Custom database directory** location, place the `.sof` and `.jic` files in the same folder as the `.xml` file to avoid missing a defined flash database or corruption of the `.jic` file.
Save as template	Enable this option and specify a unique name to save the current flash memory device definition as a template for later use. Programming File Generator saves the template in the **Custom database directory**. Click the **Edit** button to delete any templates you save.

5. For supported FPGA devices, optionally modify any of the default programming flows for the flash memory device, as Modifying Programming Flows on page 25 describes.

   Note: When you modify a programming flow, all `.jic` files using this programming flow are affected. For example, you can define a new `micron_1gb` flow, and then use this device to define the `micro_1gb_partA.jic` file. Later, you modify the`micron_1gb` flow, and then use this flow to create `micro_1gb_partB.jic`. In this example, `micro_1gb_partA` programming flow reflects the latest modifications to `micron_1gb`.

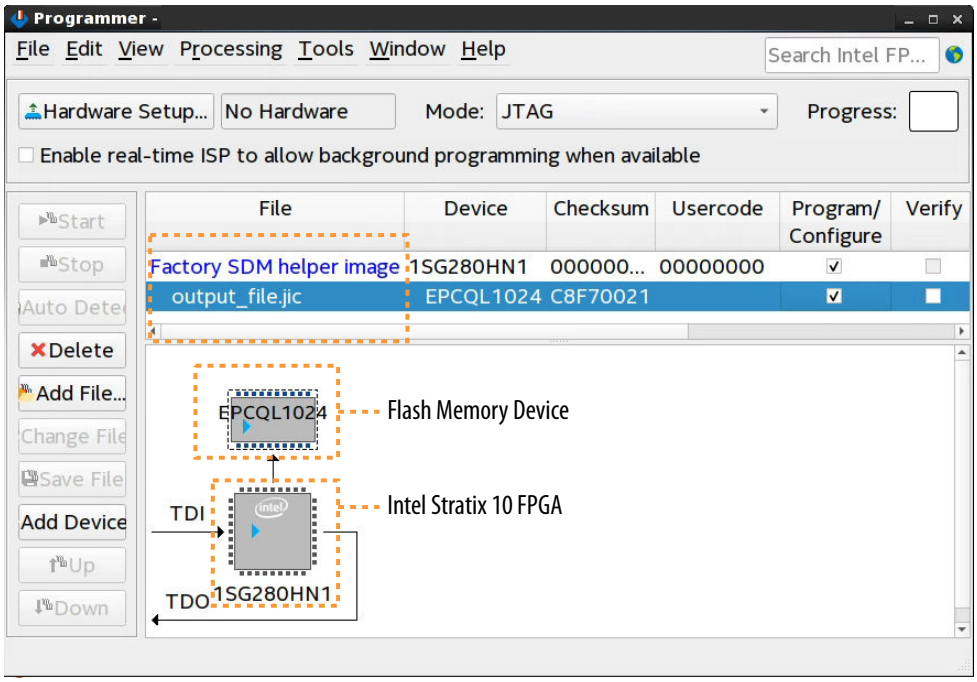
1.3.1.3. Step 3: Program the Flash Memory Device

Follow these steps to program the flash memory device using the Intel Quartus Prime Programmer and Intel FPGA download cable over a JTAG connection via Active Serial programming.

1. Generate a `.jic` file, as Step 2: Generate Secondary Programming Files (Programming File Generator) on page 7 describes.
2. Open the Intel Quartus Prime software, and then click **Tools ➤ Programmer**.
3. In the Programmer, click **Hardware Setup**, and then select a connected Intel FPGA Download Cable.

4. Click **Add File**, and then select the `.jic` file you generate in Step 1.

Figure 13. JIC Loaded for Flash Programming in Programmer



5. Enable the **Program/Configure** option for the `.jic` file. The FPGA **Device** row **Program/Configure** option automatically enables, and displays the **Factory SDM helper image** as the FPGA configuration data. This entry indicates that the programming flow first configures the SDM with the Helper Image, which subsequently controls the programming of the flash memory device.
6. Click **Start** and wait for the progress bar to reach 100%. The programming flow executes according to your specifications in the `.jic` file.

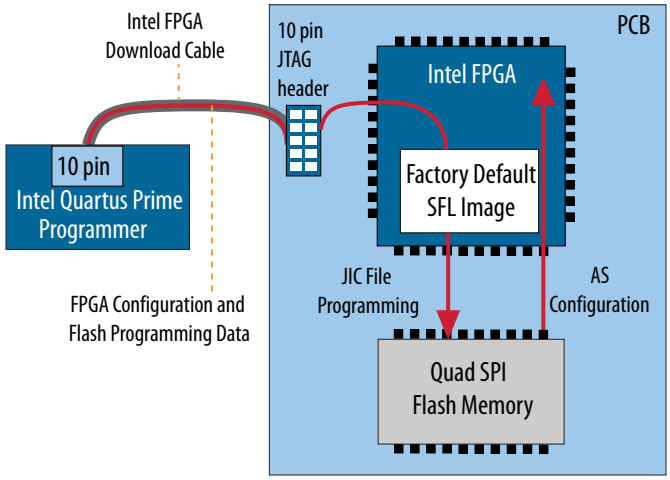
1.3.2. Generic Flash Programming (Convert Programming File Dialog Box)

In generic flash programming with the **Convert Programming File** dialog box, you generate the necessary device programming and configuration files, and perform JTAG programming of the flash memory via the Factory default SFL image.

The Factory default SFL image enables communication between the JTAG pins and the flash memory device's serial interface. The SFL is an instance of the Serial Flash Loader Intel FPGA IP that is optimized for this function.

After generating the files, you use the Intel Quartus Prime Programmer to program the flash, which in turn configures the FPGA via AS configuration.

Figure 14. Flash Programming Configuration (Intel Arria 10 and Intel Cyclone 10 GX Example)



Generic flash programming with the **Convert Programming File** dialog box includes the following high level steps that this section describes in detail:

1. Step 1: Generate Primary Device Programming Files on page 18—use the Intel Quartus Prime Assembler to generate the `.sof` FPGA configuration file.
2. Step 2: Generate Secondary Programming Files (Convert Programming Files) on page 19—use the **Convert Programming File** dialog box to generate the `.jic` that you program into your flash memory device to store `.sof` configuration data.
3. Step 3: Program the Flash Memory Device on page 26—use the Intel Quartus Prime Programmer and connected Intel FPGA download cable to program the `.jic` configuration data into the flash memory device and the `.sof` into the FPGA via Active Serial JTAG configuration.

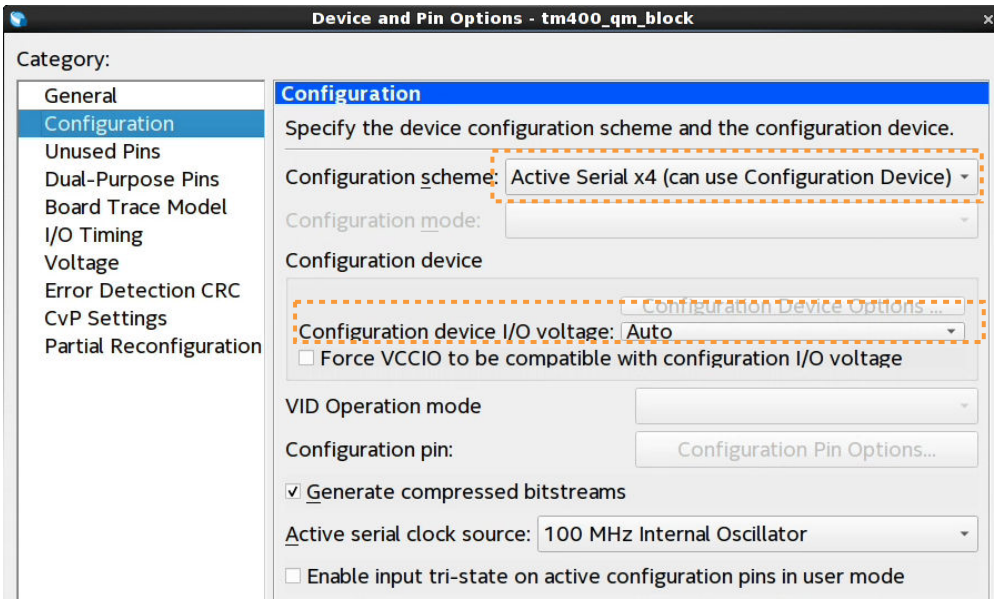
1.3.2.1. Step 1: Generate Primary Device Programming Files

The Intel Quartus Prime Assembler generates the `.sof` FPGA configuration file once design compilation is complete. Prior to running the Assembler, you can specify device and pin options that impact the `.sof` and subsequent `.jic` file generation.

Follow these steps to generate a `.sof` file for use in generic flash programming:

1. Before running the Assembler, click **Assignments ➤ Device ➤ Device & Pin Options** to specify options for FPGA configuration pins and other hardware settings that the `.sof` file preserves. The following options are particularly relevant to generic flash programming. For option descriptions, refer to Device and Pin Options on page 40.
   - **General** tab—specify the **JTAG user code**, configuration clock source, and options for specific configuration pins.
   - **Configuration** tab—specify **Active Serial** or **Active Serial x4** for the FPGA **Configuration scheme**. Select **Auto** for **Configuration device I/O voltage**, which you can specify with precision at a later time.

Figure 15. Device & Pin Options Dialog Box



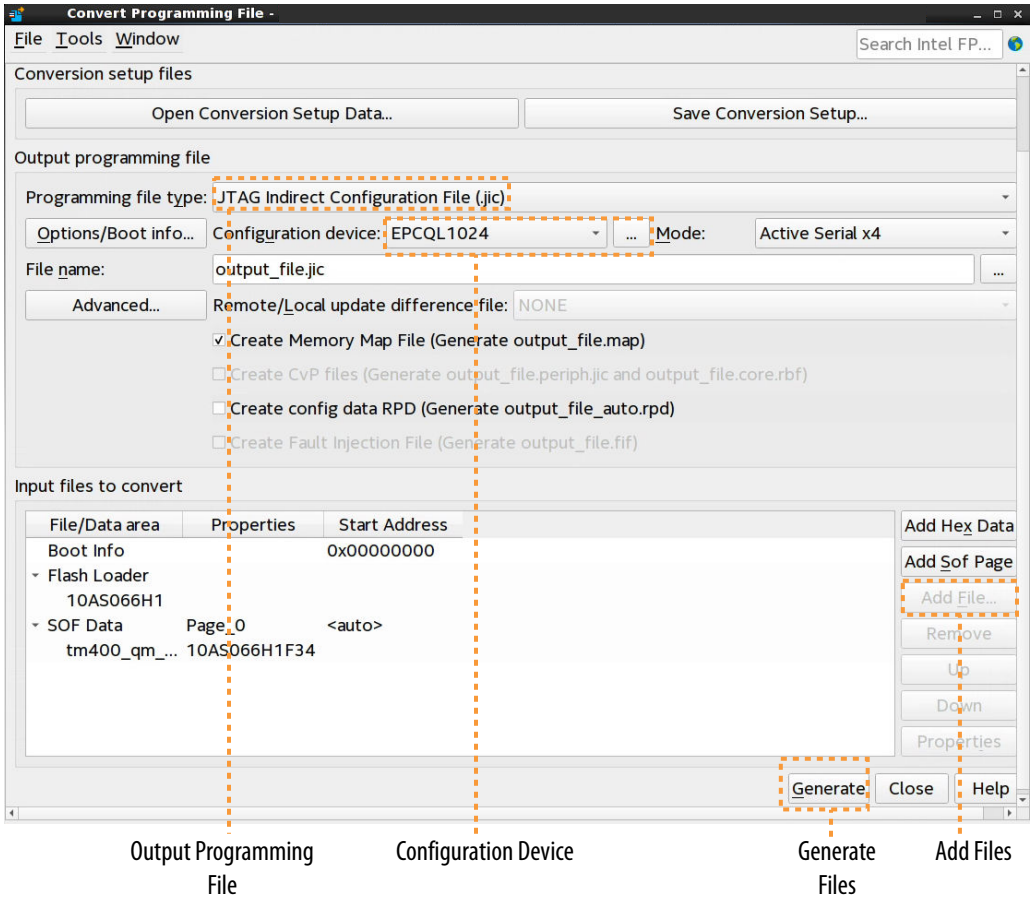
2. To generate primary device programming files, click **Processing ➤ Start ➤ Start Assembler**. The Compiler confirms that prerequisite modules are complete, and launches the Assembler to generate the programming files.

1.3.2.2. Step 2: Generate Secondary Programming Files (Convert Programming Files)

You can use the **Convert Programming File** dialog box to generate secondary programming files for alternative device programming methods. For example, generating the `.jic` file for flash programming, the `.rbf` file for partial reconfiguration, or the `.rpd` file for a third-party programmer configuration.

The options available in the **Convert Programming File** dialog box change dynamically, according to your device and configuration mode selection.

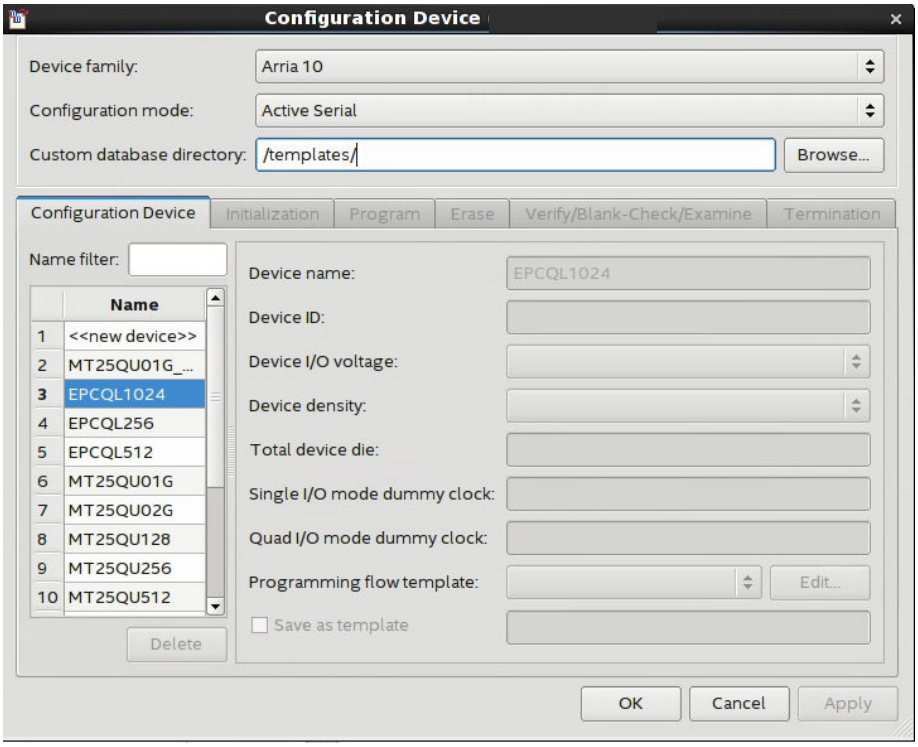
Figure 16. Convert Programming File Dialog Box



1. Generate the primary programming files for your design, as Step 1: Generate Primary Device Programming Files on page 18 describes.
2. Click **File ➤ Convert Programming Files**.
3. Under **Output programming file**, select the **JTAG Indirect Configuration File (.jic)** as the **Programming file type** that you generate. The Generic Flash Programmer supports only this file type.
4. Specify the **File name** and output directory **(...)** for the `.jic` file you generate.
5. For the configuration **Mode**, select **Active Serial x4** or **Active Serial**.

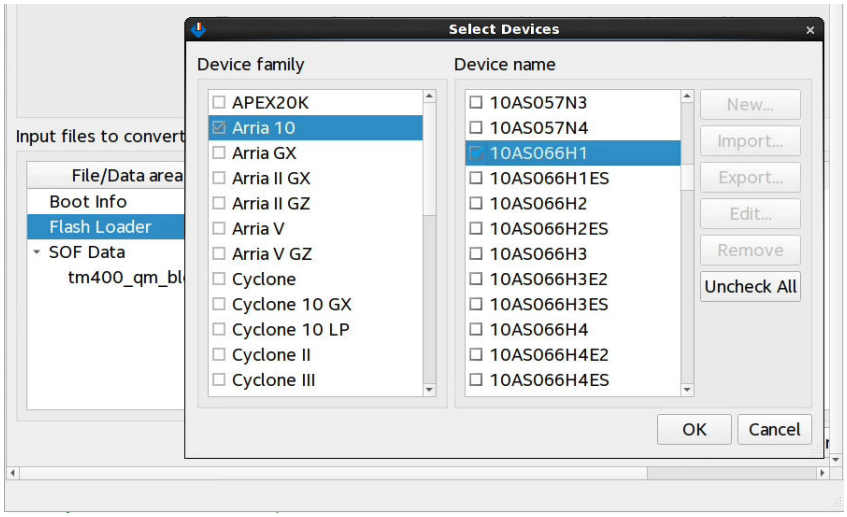
   Note: Intel Stratix 10 devices support only **Active Serial x4**.
6. To specify the **Configuration device**, click the **(...)** button to select a supported flash memory device and predefined programming flow. When you select a predefined device, you cannot modify any setting. Alternatively, click **<<new device>>** to define a new flash memory device and programming flow, as Defining a New Flash Memory Configuration Device on page 14, and Modifying Programming Flows on page 25 describe.

Figure 17. Configuration Device Dialog Box



7. Under **Input files to convert**, select the **SOF Data** item, and then click the **Add File** button. Specify the `.sof` file that contains the configuration bitstream data. To include raw data, click **Add Hex Data** and specify a `.hex` file.
8. To enable bitstream compression or encryption security settings, select the `.sof` file and click **Properties**, as Enabling Bitstream Encryption or Compression for Intel Arria 10 and Intel Cyclone 10 GX Devices on page 22 describes.
9. Select the **Flash Loader** text, and then click the **Add Device** button. Select the device that controls loading of the flash device.

Figure 18. Selecting the Flash Loader Device



10. After you specify all options in the **Convert Programming File** dialog box, click the **Generate** button to create the files.

Related Information

- Convert Programming File Dialog Box on page 47
- Compression and Encryption Settings (Convert Programming File) on page 48

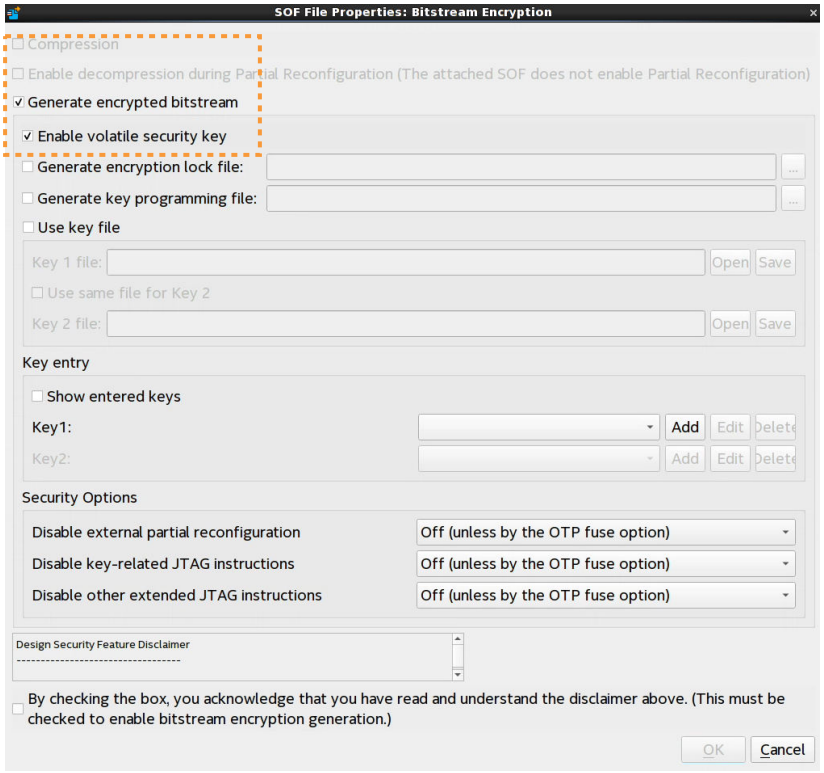
1.3.2.2.1. Enabling Bitstream Encryption or Compression for Intel Arria 10 and Intel Cyclone 10 GX Devices

You can optionally enable bitstream file encryption that requires a user-defined 256-bit security key to access the configuration bitstream. Alternatively, you can enable bitstream compression to reduces the size of your programming file to minimize file transfer and storage requirements. The compression reduces configuration file size by 30% to 55% (depending on the design). File compression and encryption options are mutually exclusive for Intel Arria 10 and Intel Cyclone 10 GX devices.

Follow these steps to enable bitstream file compression or encryption for Intel Arria 10 and Intel Cyclone 10 GX devices:

1. Generate a `.jic` file for flash programming, as this document describes.
2. In the **Convert Programming File** dialog box, select the `.sof` file under **Input files to convert**.
3. Click the **Properties** button. The **SOF File Properties: Bitstream Encryption** dialog box appears.

Figure 19. Enabling Bitstream Compression or Encryption (Intel Arria 10 and Intel Cyclone 10 GX Designs)



4. To enable compression, turn on the **Compression** option. All encryption options disable as these options are mutually exclusive.
5. To enable bitstream file encryption:
   a. Turn off the **Compression** option.
   b. Turn on the **Generate encrypted bitstream** option.
   c. Specify options for programming file key decryption, and **Security Options**, as Compression and Encryption Settings (Convert Programming File) on page 48 describes.
6. Click **OK**.

Related Information

- Compression and Encryption Settings (Convert Programming File) on page 48
- Intel Arria 10 Core Fabric and General Purpose I/Os Handbook
  For detailed device security configuration steps.
- Intel Cyclone 10 GX Core Fabric and General Purpose I/Os Handbook
  For detailed device security configuration steps.

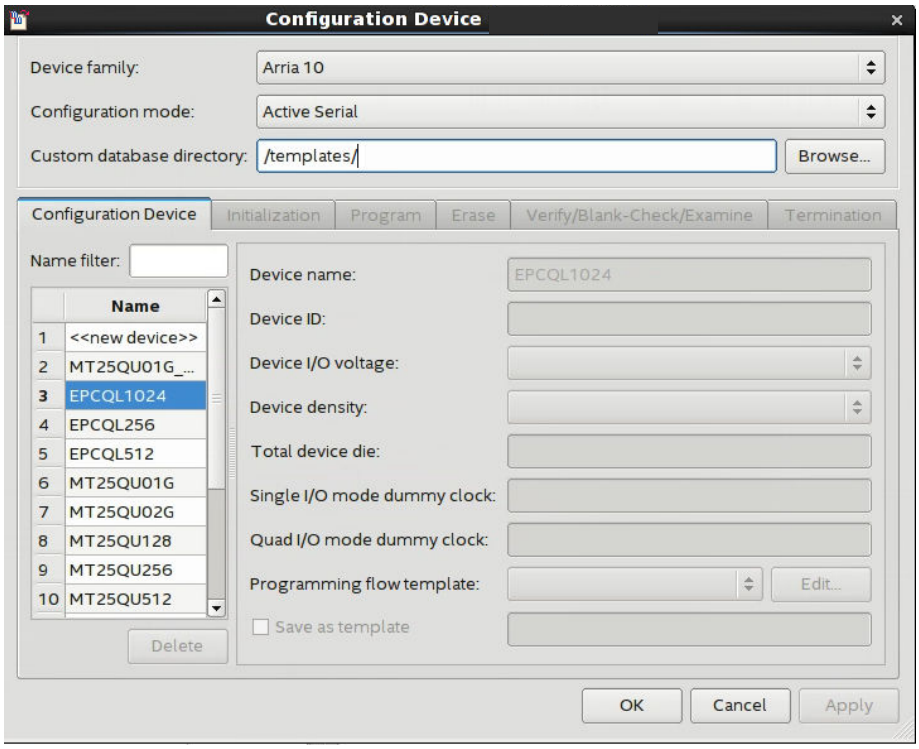
1.3.2.2.2. Defining a New Flash Memory Device

You can define and store the settings for a new flash memory device, based on the programming flow template of an existing supported flash memory device. You can customize and preserve the properties of the new flash memory device for subsequent reuse, and define other flash memory devices based on one you define.

For Intel Stratix 10 devices, the Secure Device Manager (SDM) firmware controls the flash programming flows, and you cannot modify these flows. For Intel Arria 10 and Intel Cyclone 10 GX devices, you can modify and preserve the default programming flows, as Modifying Programming Flows on page 25 describes.

When you define a new flash memory device, Intel Quartus Prime software stores the collection of settings in an `.xml` file automatically in the **Custom database directory** location that you can specify.

Figure 20. Define New Flash Device (Intel Arria 10 Example)



Follow these steps to define a new flash memory device:

1. Perform one of the following to generate a `.jic` file for flash programming:

- Step 2: Generate Secondary Programming Files (Programming File Generator) on page 7
- Step 2: Generate Secondary Programming Files (Convert Programming Files) on page 19

2. For the **Configuration Device** option, select **<<new device>>**. The settings on this and other tabs become available.
3. For **Programming flow template**, select an existing flash memory device template for the new device initial settings. For flash memory device support, refer to Supported Devices and Configuration Methods on page 4.
4. Specify the remaining settings on the **Configuration Device** tab:

Table 6. Configuration Device Tab Settings

Option	Description
Device name	Specify a unique name for the flash not already listed in the **Name** column. The **Name** must not contain any empty string (space) or special characters (except "_").
Device ID	Specify the 3-byte ID that the Programmer Auto-Detect operation uses to detect the flash programming device, such as `0x20 0xBB` or `0x21`.
Device I/O voltage	Specify **1.8V** or **3.0/3.3V** to match your memory device specification.
Device density	Select the total density that corresponds with your flash memory device size.
Total device die	Specify the total number of die for a stacked device (where applicable).
Single I/O mode dummy clock	Specify the Fast Read dummy clock cycle for a flash device in single I/O protocol. The programming file generation uses this setting to determine if the configuration requires bit shifting to compensate for the actual dummy clock cycle during Active Serial configuration.
Quad I/O mode dummy clock	Specify the Fast Read dummy clock cycle for the flash device in Quad I/O protocol. The programming file generation uses this setting to determine if the configuration requires bit shifting to compensate for the actual dummy clock cycle during Active Serial configuration.
Custom database directory	Specifies the location of the `.xml` file that preserves a flash memory device definition and flow information. The default location is the project directory or current working directory. *Note:* When you specify a non-default folder for the **Custom database directory** location, place the `.sof` and `.jic` files in the same folder as the `.xml` file to avoid missing a defined flash database or corruption of the `.jic` file.
Save as template	Enable this option and specify a unique name to save the current flash memory device definition as a template for later use. Programming File Generator saves the template in the **Custom database directory**. Click the **Edit** button to delete any templates you save.

5. For supported FPGA devices, optionally modify any of the default programming flows for the flash memory device, as Modifying Programming Flows on page 25 describes.

1.3.2.2.3. Modifying Programming Flows

You can modify and preserve the flows for Initialization, Program, Erase, Verify/Blank-Check/Examine, and Termination operations for a flash memory device for supported devices. For each operation, you can drag and drop **Actions** (such as **Read Register**) into locations in the flow to match the programming requirements of your flash memory device.

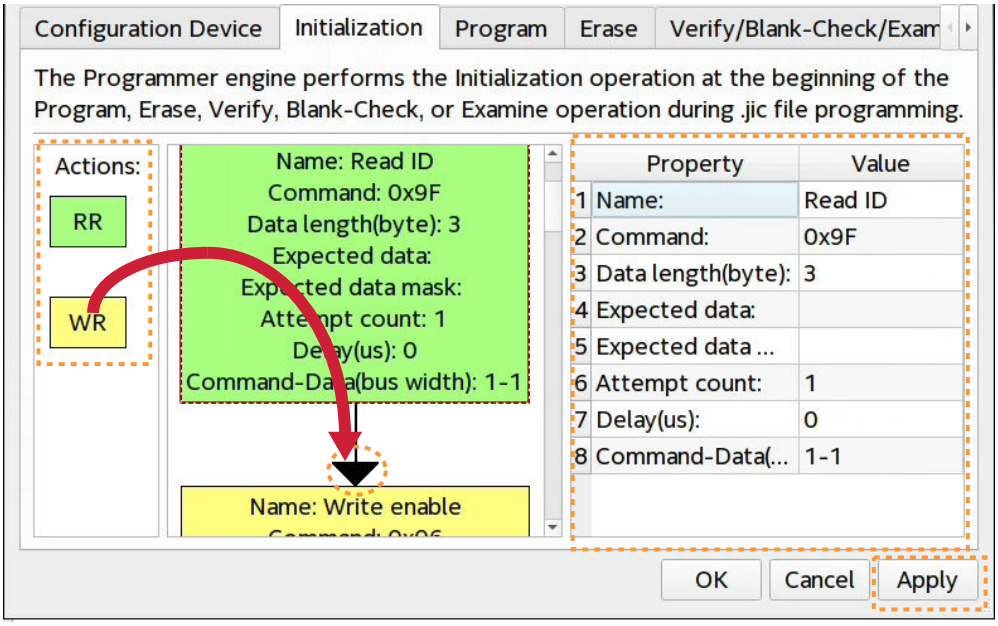
Note: For Intel Stratix 10 devices, the Secure Device Manager (SDM) firmware controls the flash programming flows, and you cannot modify these flows.

Modifying a programming flow affects all `.jic` files that use that programming flow.

Follow these steps to modify a default programming flow:

1. Define a new configuration device, as Defining a New Flash Memory Configuration Device on page 14 describes.
2. In the **Configuration Device** dialog box, click an available **Action** on the **Initialization**, **Program**, **Erase**, **Verify/Blank-Check/Examine**, and **Termination** tabs.
3. With the **Action** selected, drag the action to an arrow point in the flow graphic. The arrow icon changes to indicate when the placement is legal, at which point you can release the mouse to place the **Action** in the flow. Alternatively, right-click in the flow graphic to add, delete, copy, or paste an **Action** from the menu.

Figure 21. Modifying Initialization Flow



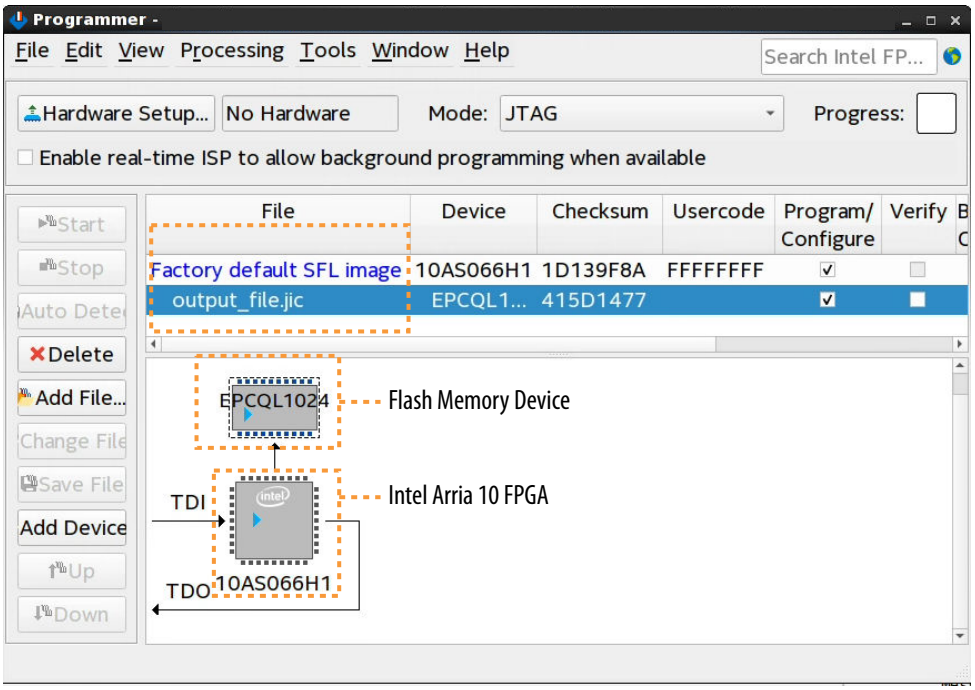
4. To modify the properties of an **Action**, click the action in the flow graphic. The editable properties appear in the adjacent pane, as Programming Flow Action Properties on page 37 describes.
5. When you are satisfied with the modifications, click **Apply**. The flow preserves with your new configuration device definition, and applies during the flash device programming.

1.3.2.3. Step 3: Program the Flash Memory Device

Follow these steps to program the flash memory device using the Intel Quartus Prime Programmer and Intel FPGA download cable over a JTAG connection via Active Serial programming.

1. Generate a `.jic` file, as Step 2: Generate Secondary Programming Files (Convert Programming Files) on page 19 describes.
2. Open the Intel Quartus Prime software, and then click **Tools ➤ Programmer**.
3. In the Programmer, click **Hardware Setup**, and then select a connected Intel FPGA Download Cable.
4. Click **Add File**, and then select the `.jic` file you generate at the previous step.

Figure 22. JIC Loaded for Flash Programming in Programmer



5. Enable the **Program/Configure** option for the `.jic` file. The FPGA **Device** row **Program/Configure** option automatically enables, and displays the **Factory default SFL image** as the FPGA configuration data. This entry indicates that the programming flow first configures the FPGA with the SFL image, which subsequently controls the programming of the flash memory device.
6. Click **Start** and wait for the progress bar to reach 100%. The programming flow executes according to your specifications in the `.jic` file.

1.3.2.4. Full Erase of Flash Memory Sectors

When performing flash memory erase operations via JTAG and a `.jic` file, the Intel Quartus Prime Programmer erases only the flash memory sectors that the `.jic` specifies.

For example, if you specify a `.jic` file containing only a 13.6Mbits FPGA image on an EPCQ64A device, the Programmer erases only the bottom 13.6Mbits, and does not erase the remaining 50.4Mbits of data.

To erase the entire flash memory device contents, do not specify a `.jic` file for flash programming. Rather, manually add the flash device to the associated FPGA device chain by following these steps:

1. In the Programmer, right-click the target FPGA device, and then click **Edit ➤ Attach Flash Device**.
2. Select the appropriate flash device from the list. The Factory Default Serial Flash Loader loads for the FPGA automatically.
3. In the Programmer, enable the **Erase** checkbox, and click **Start** to start the erase operation.

1.4. Generic Flash Programmer Flow Templates (Intel Stratix 10 devices)

For Intel Stratix 10 devices, the Secure Device Manager (SDM) firmware controls the flash programming flows, and you cannot modify the flow templates. Each flow template is identical for all supported flash memory devices for Intel Stratix 10 devices.

The following describe the templates for each programming flow:

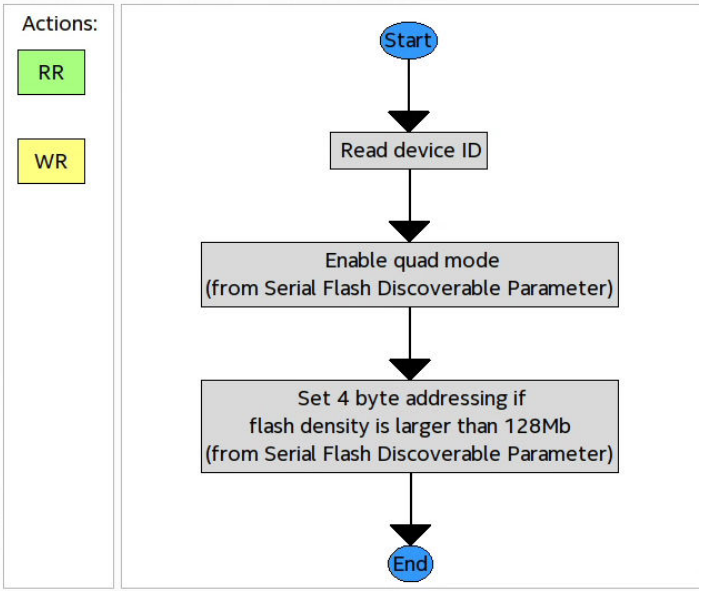
Related Information

- Initialization Flow Template (Intel Stratix 10 Devices) on page 28
- Program Flow Template (Intel Stratix 10 Devices) on page 29
- Erase Flow Template (Intel Stratix 10 Devices) on page 30
- Verify/Blank-Check/Examine Flow Template (Intel Stratix 10 Devices) on page 31
- Termination Flow Template (Intel Stratix 10 Devices) on page 31

1.4.1. Initialization Flow Template (Intel Stratix 10 Devices)

The Initialization flow template for Intel Stratix 10 devices has the following **Actions**.

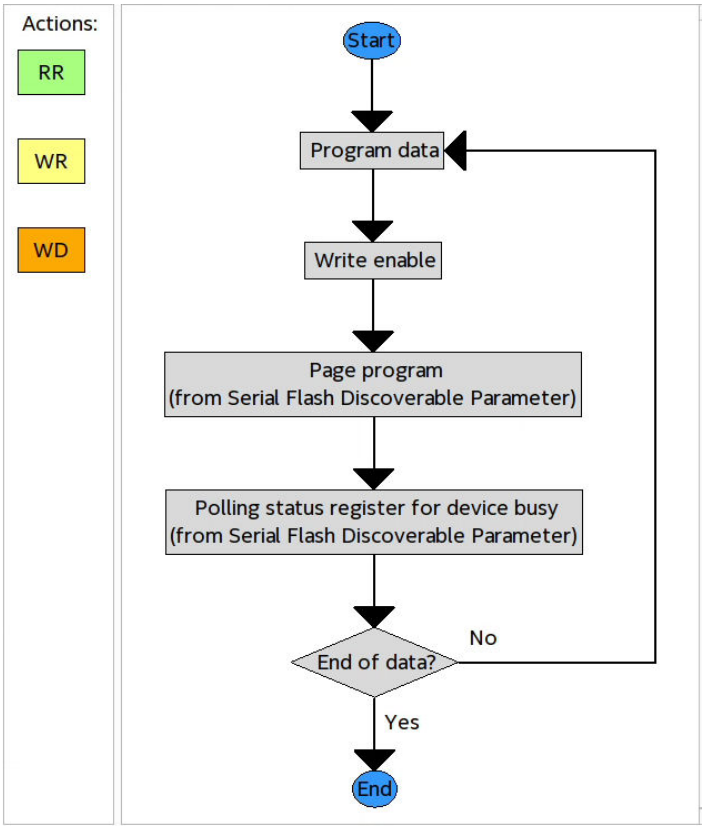
Figure 23. Initialization Flow Template



1.4.2. Program Flow Template (Intel Stratix 10 Devices)

The Program flow template for Intel Stratix 10 devices has the following **Actions**.

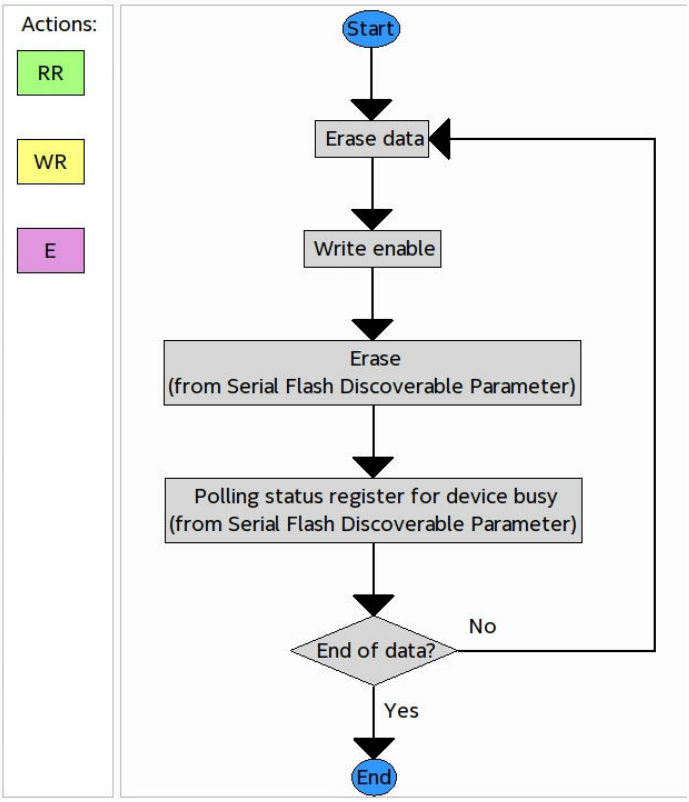
Figure 24. Program Flow Template



1.4.3. Erase Flow Template (Intel Stratix 10 Devices)

The Erase flow template for Intel Stratix 10 devices has the following **Actions**.

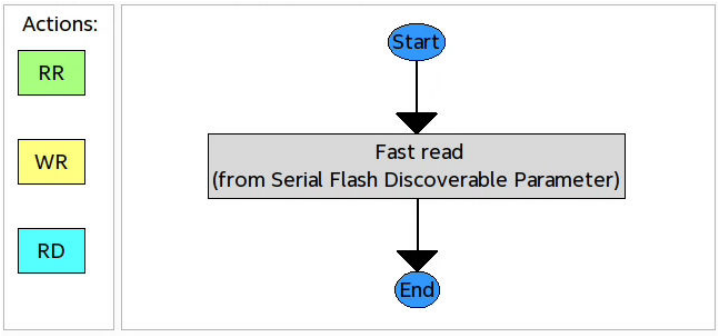
Figure 25. Erase Flow Template



1.4.4. Verify/Blank-Check/Examine Flow Template (Intel Stratix 10 Devices)

The Verify/Blank-Check/Examine flow template for Intel Stratix 10 devices has the following **Actions**.

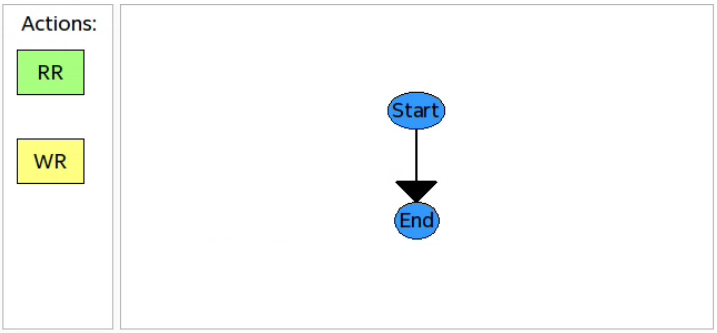
Figure 26. Verify/Blank-Check/Examine Flow Template



1.4.5. Termination Flow Template (Intel Stratix 10 Devices)

The Termination flow template for Intel Stratix 10 devices has no actions, as the following figure shows.

Figure 27. Termination Flow Template



1.5. Generic Flash Programmer Flow Templates (Intel Arria 10 and Intel Cyclone 10 GX)

For supported devices, you can modify and preserve the default programming flows, as Modifying Programming Flows on page 25 describes.

The following describe the templates for each programming flow:

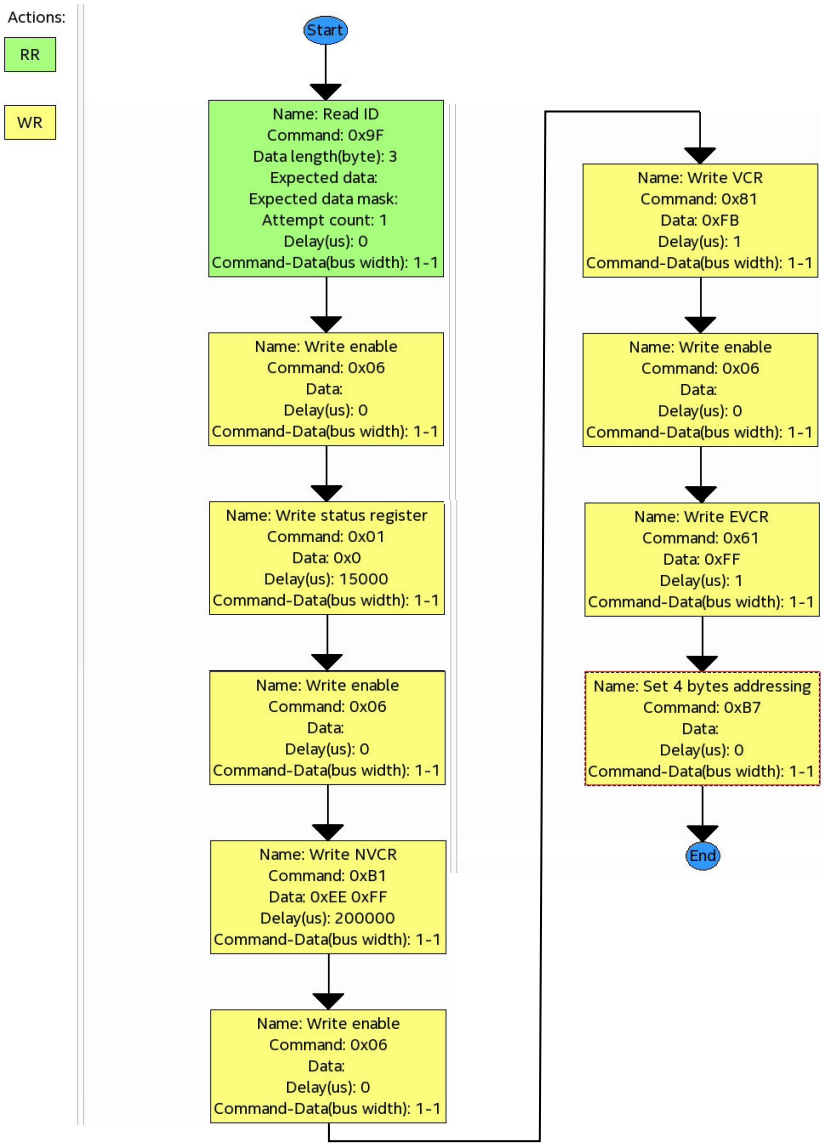
Related Information

- Initialization Flow Templates (Intel Arria 10 and Intel Cyclone 10 GX) on page 32
- Program Flow Template (Intel Arria 10 and Intel Cyclone 10 GX) on page 33
- Erase Flow Template (Intel Arria 10 and Intel Cyclone 10 GX) on page 34
- Verify/Blank-Check/Examine Flow Template (Intel Arria 10 and Intel Cyclone 10 GX) on page 35
- Termination Flow Template (Intel Arria 10 and Intel Cyclone 10 GX) on page 36
- Programming Flow Action Properties on page 37

1.5.1. Initialization Flow Templates (Intel Arria 10 and Intel Cyclone 10 GX)

The Initialization flow template has the following **Actions**.

Figure 28. Initialization Flow Template (Micron Example)

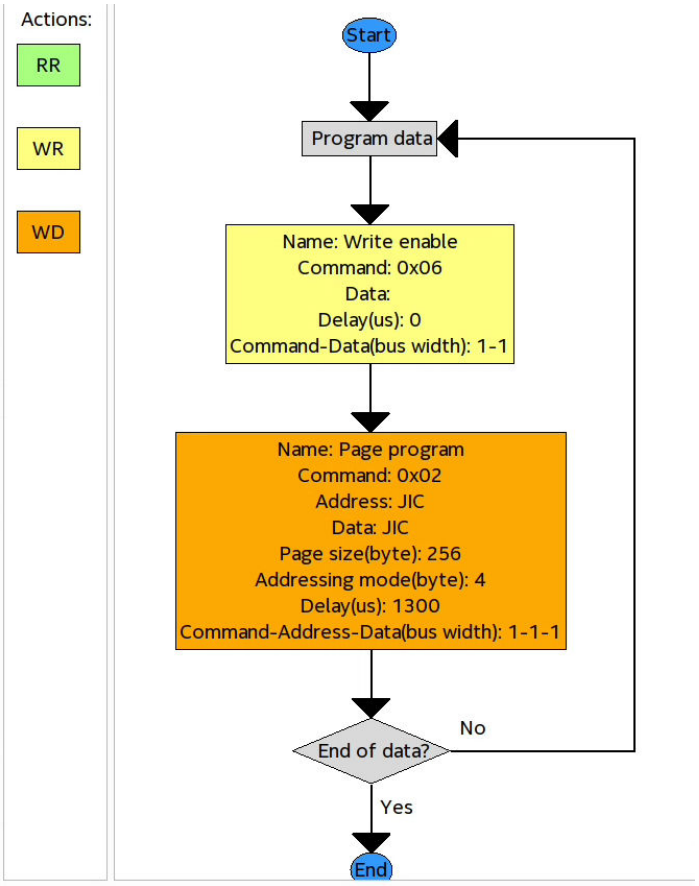


Note: Example and **Action** properties are for reference only as specific actions vary by flash memory device.

1.5.2. Program Flow Template (Intel Arria 10 and Intel Cyclone 10 GX)

The Program flow template has the following **Actions**.

Figure 29. Program Flow Template

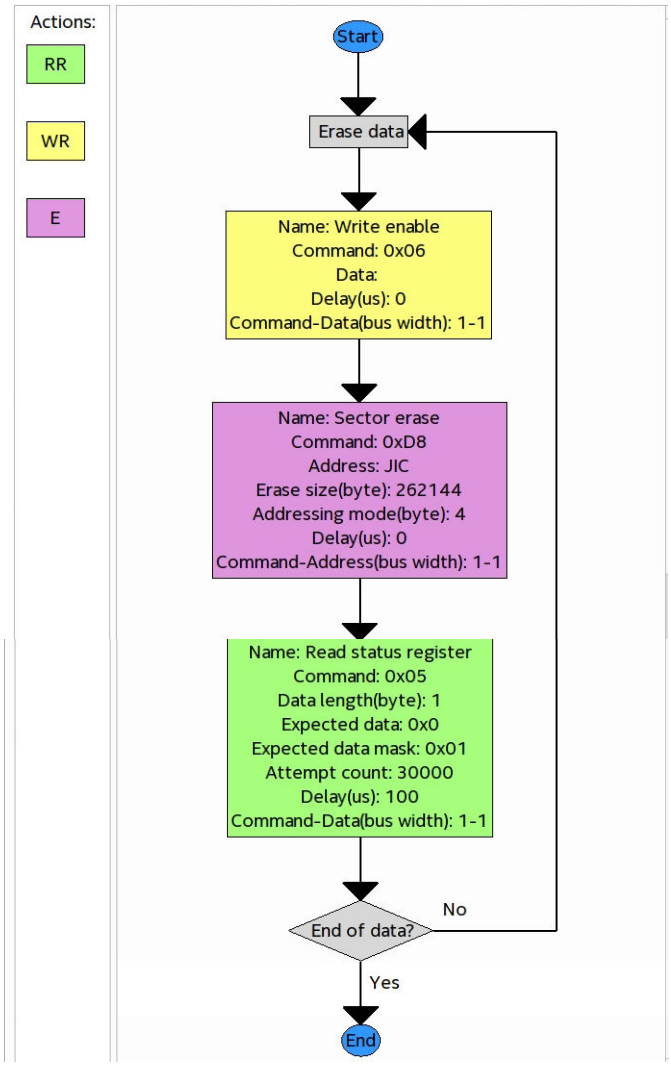


Note: **Action** properties are for reference only as specific properties vary by application.

1.5.3. Erase Flow Template (Intel Arria 10 and Intel Cyclone 10 GX)

The Erase flow template has the following **Actions**.

Figure 30. Erase Flow Template

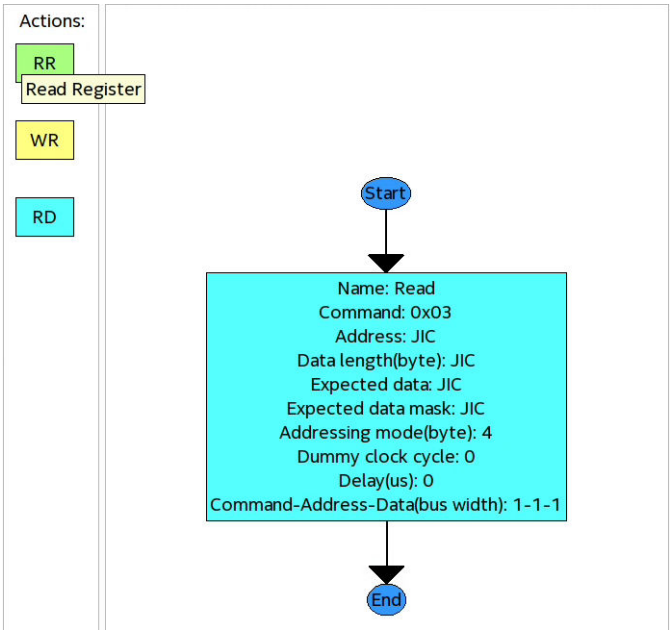


Note: **Action** properties are for reference only as specific properties vary by application.

1.5.4. Verify/Blank-Check/Examine Flow Template (Intel Arria 10 and Intel Cyclone 10 GX)

The Verify/Blank-Check/Examine flow template has the following **Actions**.

Figure 31. Verify/Blank-Check/Examine Flow Template

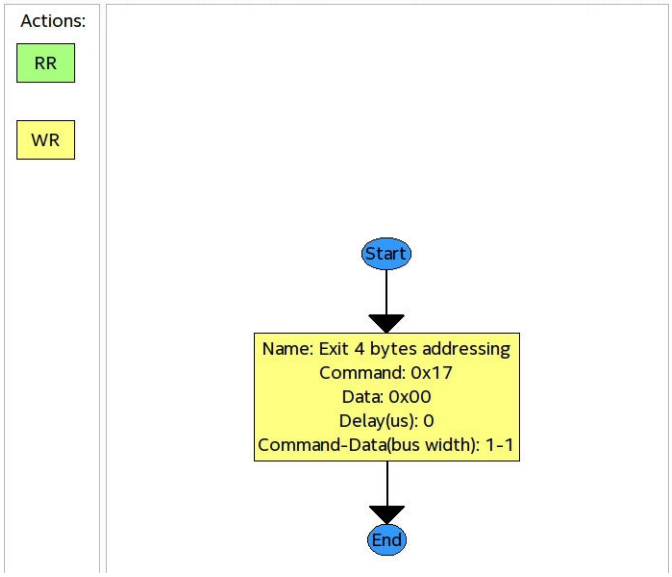


Note: **Action** properties are for reference only as specific properties vary by application.

1.5.5. Termination Flow Template (Intel Arria 10 and Intel Cyclone 10 GX)

The Termination flow template has the following **Actions**.

Figure 32. Termination Flow Template



Note: **Action** properties are for reference only as specific properties vary by application.

1.5.6. Programming Flow Action Properties

The available programming flow **Actions** have the following properties:

Table 7. Read Register (RR) Action Properties

Symbol	Action	Property	Description
Read Register (RR)	Performs a read operation that returns a list of data register bytes.	**Name**	Displays the name of the command that executes as part of the **Action**, such as **Read ID** and **Read flag status register**.
		Command	Hex value that executes the command, such as `0x70`.
		Data length (byte)	Expected length of returned data. To read according to the `.jic` file, enter the value as `JIC`.
		Expected data	(Optional) For use with **Expected data mask**. The following are example entries: • 1 Byte: `0xAB` • 2 Bytes: `0xAB 0xCD` • 3 Bytes: `0xAB 0xCD 0xEF` To compare the actual read back data against the `.jic` file, enter the value as `JIC`.
		Expected data mask	(Optional) Specify this value if you specify the **Expected data**. The default value is zero. The operation compares the actual read back register values against the expected value and mask value, and reports error on mismatch. The following are example entries: • 1 Byte: `0xFF` • 2 Bytes: `0xFF 0xFF`
		Delay (us)	Delay after **Command** executes.
		Attempt count	The number of times to repeat the command. You can use the **Expected data**, **Expected data mask**, **Delay** and **Attempt count** to create a polling operation on a register value.
		Command-Data (bus width)	Ratio of command to data bus widths.

Table 8. Write Register (WR) Action Properties

Action	Description	Property	Description
Write Register (WR)	Performs a write operation to a volatile or non-volatile register.	**Name**	Displays the name of the command that executes as part of the **Action**, such as **Write enable** or **Write status register**.
		Command	Hex value that executes the command, such as `0x70`.
		Data	Data that you want to write into the register. If the command doesn't require data (for example, Write Enable), leave this property empty. The following are example entries:

continued...

Action	Description	Property	Description
			• 1 Byte: 0xAB • 2 Bytes: 0xAB 0xCD
		Delay (us)	Delay after **Command** executes.
		Command-Data (Bus Width)	Ratio of command to data bus widths.

Table 9. Write Data (WD) Action Properties

Action	Description	Property	Description
Write Data (WD)	Performs a write operation of data into the flash memory.	**Name**	Displays the name of the command that executes as part of the **Action**, such as **Page program**.
		Command	Hex value that executes the command, such as 0x70.
		Address	Specifies the flash memory address that you want to start writing. To flash the .jic file, enter the value as JIC.
		Data	Data that you want to program into the flash device.(4) To flash the .jic file, enter the value as JIC.
		Page size (byte)	Page size of this flash memory device. Typical value is 256 for most flash devices.
		Addressing mode (byte)	Specifies the number of bytes the flash memory device expects for an address (switch between 3 or 4 bytes). The mode must match the addressing mode that you specify in the Initialization flow.
		Delay (us)	Delay after **Command** executes.
		Command-Address-Data (bus width)	Ratio between command, address, and data bus widths.

Table 10. Read Data (RD) Action Properties

Symbol	Action	Property	Description
Read Data (WD)	Performs a read operation of data from the flash memory.	**Name**	Displays the name of the command that executes as part of the **Action**, such as **Read**.
		Command	Hex value that executes the command, such as 0x70.
		Address	Specifies the flash memory address that you want to start reading. To read according to the .jic file, enter the value as JIC.
		Data length (byte)	Expected length of returned data. To read according to the .jic file, enter the value as JIC.

continued...

(4) You can specify the **Data** value in hex, decimal, or a mixture of both. The max entry cannot exceed 32727. For hex entry, 1 byte (0xAA) equals 5 characters for a max entry of 6553 hex characters. The Data value you specify must match the number of bytes for the page size of 256.

Symbol	Action	Property	Description
		Expected data	(Optional) For use with **Expected data mask**. The following are example entries: • 1 Byte: `0xAB` • 2 Bytes: `0xAB 0xCD` • 3 Bytes: `0xAB 0xCD 0xEF` To compare the actual read back data against the `.jic` file, enter the value as `JIC`.
		Expected data mask	If you specify the **Expected data mask**, then the operation compares the actual read back data against the expected value and mask value, and reports an error on mismatch. If you want to compare against the `.jic` file, enter the value as `JIC`.
		Addressing mode (byte)	Specifies the number of bytes the flash memory device expects for an address (switch between 3 or 4 bytes). The mode must match the addressing mode that you specify in the Initialization flow.
		Dummy clock cycle	Specifies the number of dummy clock cycles subsequent to the **Command**.
		Delay (us)	Delay after **Command** executes.
		Command-Address-Data (bus width)	Ratio between command, address, and data bus widths.

Table 11. Erase (E) Action Properties

Symbol	Action	Property	Description
Erase (E)	Performs an erase of data from the flash memory.	**Name**	Displays the name of the command that executes as part of the **Action**, such as **Erase sector**.
		Command	Hex value that executes the command, such as `0x70`.
		Address	Specifies the flash memory address that you want to start erasing. The address must be aligned with **Erase size**. If you want to flash the `.jic` file, enter the value as `JIC`.
		Erase size	Specifies the erase size (in Bytes).
		Addressing mode	Specifies the number of bytes the flash memory device expects for an address (switch between 3 or 4 bytes). The mode must match the addressing mode that you specify in the Initialization flow.
		Delay	Delay after **Command** executes.
		Command-Address (Bus Width)	Ratio of command to address bus widths.

1.6. Generic Flash Programmer Settings Reference

The following topics describe Generic Flash Programmer settings.

Related Information

- Device and Pin Options on page 40

1.6.1. Device and Pin Options

The following tables describe **Device & Pin Option** settings that impact Generic Flash Programmer. To access, click **Assignments ➤ Device ➤ Device & Pin Options**.

General Device Options

Allow you to specify basic device configuration options that are independent of a specific configuration scheme. To access these settings, click **Assignments ➤ Device ➤ Device and Pin Options ➤ General**.

Table 12. General Device Options

Option	Description
Options *Note:* Not supported for Intel Stratix 10 devices.	• **Auto-restart configuration after error**—restarts the configuration process automatically if a data error is encountered. If this option is turned off, you must externally direct the device to restart the configuration process if an error occurs. This option is available for passive serial and active serial configuration schemes. • **Release clears before tri-states**—releases the clear signal on registered logic cells and I/O cells before releasing the output enable override on tri-state buffers. If this option is turned off, the output enable signals are released before the clear overrides are released. • **Enable user-supplied start-up clock (CLKUSR)**—uses a user-supplied clock on the CLKUSR pin for initialization. When turned off, external circuitry is required to provide the initialization clock on the DCLK pin in the Passive Serial and Passive Parallel Synchronous configuration schemes; in the Passive Parallel Asynchronous configuration scheme, the device uses an internal initialization clock. • **Enable device-wide reset (DEV_CLRn)**—enables the DEV_CLRn pin, which allows all registers of the device to be reset by an external source. If this option is turned off, the DEV_CLRn pin is disabled when the device operates in user mode and is available as a user I/O pin. • **Enable device-wide output enable (DEV_OE)**—enables the DEV_OE pin when the device is in user mode. If this option is turned on, all outputs on the chip operate normally. When the pin is disabled, all outputs are tri-stated. If this option is turned off, the DEV_OE pin is disabled when the device operates in user mode and is available as a user I/O pin. • **Enable INIT_DONE output**—enables the INIT_DONE pin, which allows you to externally monitor when initialization is complete and the device is in user mode. If this option is turned off, the INIT_DONE pin is disabled when the device operates in user mode and is available as a user I/O pin.

continued...

Option	Description
	• **Enable JTAG Pin Sharing**—enables the JTAG pin sharing feature. The JTAGEN pin is enables and becomes a dedicated input pin in user mode. JTAG pins (TDO, TCK, TDI, and TMS pins) are available as test pins when the JTAGEN pin is pull low. JTAG pins are dedicated when the JTAGEN pin is high. If this option is turned off, the JTAGEN pin is disabled when the device operates in user mode and is available as a user I/O pin. JTAG pins are retained as dedicated JTAG pins. • **Enable nCONFIG, nStatus, and CONF_DONE pins**—enables the major configuration pins, nCONFIG, nSTATUS, and CONF_DONE pin in user mode. If this option is turned off, the nCONFIG, nSTATUS, and CONF_DONE pins are disabled when the device operates in user mode and are available as user I/O pins. • **Enable OCT_DONE** —enables the OCT_DONE pin, which controls whether the INIT_DONE pin is gated by OCT_DONE pin. If this option is turned off, the INIT_DONE pin is not gated by the OCT_DONE pin. • **Enable security bit support**—enables the security bit support, which prevents data in a device from being obtained and used to program another device. This option is available for supported device (MAX® II, and MAX V) families. • **Set unused TDS pins to GND**—sets the unused temperature sensing diode TSD pins, TEMPDIODEp and TEMPDIODEn to GND in the pin. By default, TSD pins are available for connection to an external temperature sensing device; however, you must manually connect the pins to GND if they are not connected. When turned on, this option updates the information in the .pin file and does not affect FPGA behavior. • **Enable CONFIG_SEL pin**—enables the BOOT_SEL pin in user mode. If this option is turned off, the BOOT_SEL pin is disabled when the device operates in user mode and is available as a user I/O pin. • **Enable nCEO pin**—enables the nCEO pin. This pin should be connected to the nCE of the succeeding device when multiple devices are being programmed. If this option is turned off, the nCEO pin is disabled when the device operates in user mode and is available as a user I/O pin. • **Enable autonomous PCIe HIP mode**—releases the PCIe HIP after periphery configuration, before device core configuration completes. This option only takes effect if CvP mode is disabled. • **Enable the HPS early release of HPS IO**—releases the HPS shared I/O bank after the IOCSR programming.
Auto usercode	Sets the JTAG user code to match the checksum value of the device programming file. The programming file is a .pof for non-volatile devices, or an .sof for SRAM-based devices. If you turn on this option, the **JTAG user code** option is not available.
JTAG user code	Specifies a hexadecimal number for the device selected for the current Compiler settings. The JTAG user code is an extension of the option register. This data can be read with the JTAG USERCODE instruction. If you turn on **Auto usercode**, this option is not available.
In-system programming clamp state	Allows you to specify the state that the pins take during in-system programming for used pins that do not have an in-system programming clamp state assignment. Unused pins and dedicated inputs must always be tri-stated for in-system programming. Used pins are tri-stated by default during in-system programming, which electrically isolates the device from other devices on the board. At times, however, in order to prevent system damage you may want to specify the logic level for used pins during in-system programming. The following settings are available:

continued...

Option	Description
	• **Tri-state**—the pins are tri-stated. • **High**—the pins drive VCCIO. • **Low**—the pins drive GND. • **Sample and Sustain**—the pins drive the level captured during the SAMPLE/PRELOAD JTAG instruction.
Configuration clock source	Specifies the clock source for device initialization (the duration between CONF_DONE signal went high and before INIT_DONE signal goes high). For AS x1 or AS x4 configuration mode, you can select either **Internal Oscillator** or **CLKUSR** pin only. The DCLK pin is an illegal option for AS mode. In 14 nm device families, only **Internal Oscillator** or **OSC_CLK_1** pins are available.
Device initialization clock source	Specifies the clock source for device initialization (the duration between CONF_DONE signal went high and before INIT_DONE signal goes high). For AS x1 or AS x4 configuration mode, you can select either **Internal Oscillator** or **CLKUSR** pin only. The DCLK pin is an illegal option for AS mode. In 14 nm device families, only **Internal Oscillator** or **OSC_CLK_1** pins are available.

Configuration Options

Allow you to specify the configuration scheme, configuration device and pin options, serial clock source, and other options for subsequent device configuration with your programming bitstream. To access these settings, click **Assignments ➤ Device ➤ Device and Pin Options ➤ Configuration**. Disabled options are unavailable for the current device or configuration mode.

Table 13. Configuration Options

Option	Description
Configuration scheme	Specifies the scheme of configuration for generation of appropriate primary and secondary programming files, such as **Active Serial x4**. Only options appropriate for the current **Configuration Scheme** are available.
Configuration Device	Allows you to specify options for an external configuration device that stores and loads configuration data. • **Configuration device I/O voltage**—specifies the VCCIO voltage of the configuration pins for the current configuration scheme of the target device. This option is available for supported device families. • **Force VCCIO voltage to be compatible with configuration I/O voltage**—forces the VCCIO voltage of the configuration pins to be the same as the configuration device I/O voltage. If you turn off this option, the VCCIO voltage of the configuration pins may vary depending on the I/O standards used in the I/O banks containing the configuration pins. This option is available for supported device families.
Configuration Pin Options	Enables or disables operation of specific device configuration pins for status monitoring, SEU error detection, CvP, and other configuration pin options.
Generate compressed bitstreams	Generates compressed bitstreams and enables bitstream decompression in the target device.
Active serial clock source	Specifies the configuration clock source for Active Serial programming. Options range from 12.5 MHz to 100 MHz.
VID Operation Mode	Enables Voltage IDentification logic in the target device with selected operation mode. The available options are **PMBus Master** or **PMBus Slave**.

continued...

Option	Description
HPS/FPGA configuration order	For hard processor system (HPS) configuration, specifies the order of configuration between the HPS and FPGA. The options are **HPS First**, **After INIT_DONE**, and **When requested by FPGA**.
HPS debug access port	• **Disabled**—the HPS JTAG is not enabled. • **HPS Pins**—the HPS JTAG is routed to the HPS dedicated I/O. • **SDM Pins**—the HPS JTAG is chained to the FPGA JTAG.
Disable Register Power-Up Initialization	Specifies whether the Assembler generates a bit stream with register power-up initialization.

Table 14. Assembler Security Settings

For Intel Stratix 10 devices, specifies settings for programming bitstream authentication, encryption, scrambling, and other eFuse enabled security options. To access these settings, click **Assignments ➤ Device ➤ Device and Pin Options ➤ Security**. Disabled options are unavailable for the current device or configuration mode.

Option	Description
Quartus Key File	Specifies the first level signature chain file (`.qky`) that you generate. This chain includes the root key (`.pem`) and one or more design signing keys (`.pem`) required to sign the bitstream and allow access to the FPGA when using authentication or encryption.
Encryption key storage select	Specifies the location that stores the `.qek` key file. You can select either **Battery Backup RAM** or **eFuses** for storage.
Encryption update ratio	Specifies the ratio of configuration bits compared to the number of key updates required for bitstream decryption. You can select either **31:1** (the key must change 1 time every 31 bits) or **Disabled** (no update required). Encryption supports up to 20 intermediate keys.
Enable scrambling	Scrambles the configuration bitstream.
More Options	Opens the **More Security Options** dialog box for specifying additional physical security options.

Configuration PIN Dialog Box

For Intel Stratix 10 devices, allows you to enable or disable specific configuration pins. For example, you can enable the CvP_CONFDONE pin, which indicates that the device finished core programming in Configuration via Protocol mode. To access these settings, click **Assignments ➤ Device ➤ Device and Pin Options ➤ Configuration Pin Options**. Disabled options are unavailable for the current device or configuration mode.

Table 15. Configuration PIN Dialog Box

Option	Values	Description
USE PWRMGT_SCL output	**SDM_1O0|SDM_IO14**	This is a required PMBus interface for the power management when the VID operation mode is the PMBus Master or PMBus Slave mode. Disable this pin for a non-SmartVID device. Intel recommends using the SDM_IO14 pin for this function.
Use PWRMGT_SDA output	**SDM_1O11|SDM_1O12|SDM_1O16**	This is a required PMBus interface for the power management when the VID operation mode is the PMBus Master or PMBus Slave mode. Disable this pin for a non-SmartVID device.

continued...

Option	Values	Description
		Intel recommends using the SDM_IO11 pin for this function.
Use PWRMGT_ALERT output	**SDM_1O0|SDM_1O12**	This is a required PMBus interface for the power management that is used only in the PMBus Slave mode. Disable this pin for a non-SmartVID device. Intel recommends using the SDM_IO12 pin for this function.
USE CONF_DONE output	**SDM_100, SDM_1010 - SDM_1016**	Implement CONF_DONE using appropriate configuration pin resource.
USE INIT_DONE output	**SDM_100, SDM_1010 - SDM_1016**	Enables the INIT_DONE pin, which allows you to externally monitor when initialization is completed and the device is in user mode. If this option is turned off, the INIT_DONE pin is disabled when the device operates in user mode and is available as a user I/O pin.
USE CVPCONF_DONE output	**SDM_100, SDM_1010 - SDM_1016**	Enables the CVP_CONFDONE pin, which indicates that the device finished core programming in Configuration via Protocol mode. If this option is turned off, the CVP_CONFDONE pin is disabled when the device operates in user mode and is available as a user I/O pin.
USE SEU_ERROR output	**SDM_100, SDM_1010 - SDM_1016**	Enables the SEU_ERROR pin for use in single event upset error detection.
USE UIB CATTRIP output	**SDM_100, SDM_1010 - SDM_1016**	Enables UIB_CATTRIP output to indicate an extreme over-temperature conditioning resulted from UIB usage.
USE HPS cold nreset	**SDM_100, SDM_1010 - SDM_1016**	An optional reset input that cold resets only the HPS and is configured for bidirectional operation.
Direct to factory image	**SDM_100, SDM_1010 - SDM_1016**	If this pin asserted then device loads the factory image as the first image after boot without attempting to load any application image.
USE DATA LOCK output	**SDM_100, SDM_1010 - SDM_1016**	Output to indicate DIBs on both die in the same package is ready for data transfer.

1.6.2. More Security Options Dialog Box

Table 16. More Security Options Dialog Box

For Intel Stratix 10 devices, specifies additional configuration bitstream physical security settings. To access these settings, click **Assignments ➤ Device ➤ Device and Pin Options ➤ Security ➤ More Settings** button. Disabled options are unavailable for the current device or configuration mode.

Option	Description	Values
Disable JTAG	Disables JTAG command and configuration of the device. Setting this eliminates JTAG as mode of attack, but also eliminates boundary scan functionality.	• **Off**—inactive • **On**—active until wipe of containing design • **On sticky**—active until next POR • **On check**—checks for corresponding blown fuse
Force SDM clock to internal oscillator	Disables an external clock source for the SDM. The SDM must use the internal oscillator. Using an internal oscillator is more secure than allowing an external clock source for configuration.	
Force encryption key update	Specifies that the encryption key must update by the frequency that you specify for the **Encryption update ratio** option. The default ration value is 31:1. Encryption supports up to 20 intermediate keys.	
Disable virtual eFuses	Disables the eFuse virtual programming capability.	

continued...

Option	Description	Values
Lock security eFuses	Causes eFuse failure if the eFuse CRC does not match the calculated value.	
Disable HPS debug	Disables debugging through the JTAG interface to access the HPS.	
Disable encryption key in eFuses	Specifies that the device cannot use an AES key stored in eFuses. Rather, you can provides an extra level of security by storing the AES key in BBRAM.	
Disable encryption key in BBRAM	Specifies that the device cannot use AES key stored in BBRAM. Rather, you can provides an extra level of security when you store the AES key in eFuses.	

(5)

1.6.3. Input Files Tab Settings (Programming File Generator)

The **Input Files** tab allows you to specify the `.sof`, `.pmsf`, or `.rbf` file that contains the configuration bitstream data required to generate one or more secondary programming files. The **Input Files** tab and options change dynamically, according to your **Output Files** tab selections.

The following input file settings are available:

Table 17. Input File Settings

Setting	Description
Add Bitstream	Click this button to specify a `.sof`, `.pmsf`, or `.rbf` as input for generation of the secondary programming file you select in **Output Files**. Depending on the target device, the Intel Quartus Prime software may allow you to add multiple SOF files.
Add Raw Data	Click this button to specify a `.hex` or `.bin` file that contains raw programming data as input for generation of the secondary programming file you select in **Output Files**.
Remove	Removes the file you select from the **Input Files** tab.
Properties	Displays the properties of the item you select in the **Input Files** tab.

1.6.4. Output Files Tab Settings (Programming File Generator)

The **Output Files** tab allows you to specify the type of secondary programming file that you want to generate (output) with the **Programming File Generator**. The **Programming File Generator** converts a primary programming file (for example, `.sof`) into a programming file for alternative programming methods (for example, a `.jic` for flash programming). The **Output Files** tab and options change dynamically according to your selections.

(5) Security options not yet available for Intel Agilex devices.

The following output file options are available:

Table 18. Output File Options

Setting	Description
Device family	Specifies the FPGA device family you are targeting for configuration. **Programming File Generator** supports only Intel Agilex, Intel Stratix 10, Intel MAX 10, and Intel Cyclone 10 LP devices.
Configuration mode	Specifies the method of FPGA configuration, such as **Active Serial x4**, **AVST x8**, **AVST x16**, or **AVST x32**. Generic Flash Programmer supports only **Active Serial x4**.
Output directory and **Name**	Specifies the name and location of the file you generate. By default, this location is in the top-level project directory.
File Types	Allows you to enable the type of secondary programming file that you want to generate. Generic Flash Programmer supports only **JTAG Indirect Configuration File (.jic)**. The available options include: • **JTAG Indirect Configuration File (.jic)** • **Programmer Object File (.pof)** • **Raw Binary File for CvP Core Configuration (.rbf)** • **Raw Binary File for HPS Core Configuration (.rbf)** • **Raw Binary File for Partial Reconfiguration (.rbf)** • **Raw Programming Data File (.rpd)**

1.6.5. Add Partition Dialog Box (Programming File Generator)

To open in the **Programming File Generator**, click the **Configuration Device** tab, select a device from the list, and click **Add Partition**.

Allows you to specify the attributes of a new partition. The following settings are available:

Table 19. Add Partition Dialog Box Settings

Setting	Description
Name	Name that you give to the partition.
Input file	Input file to program into the flash partition.
Page	Configuration devices can store multiple configuration bitstreams in flash memory, called pages. CFI configuration devices can store up to eight configuration bitstreams. Intel Hyperflex™ devices can store up to four configuration bitstreams, including the factory image. In Intel Hyperflex devices, with the remote system update feature enabled, **Page** represents the parity.
Address Mode	The options are: • **Auto**—automatically allocates a block in the flash device to store the data. • **Block**—specify the start and end address of the flash partition. • **Start**—specify the start address of the partition. The tool assigns the end address of the partition based on the input data size.
Start address	Specifies the start address of the partition. Only enabled when **Address Mode** is **Block** or **Start**.
End address	Specifies the end address of the partition. Only enabled when **Address Mode** is **Block**.

1.6.6. Bitstream Co-Signing Security Settings (Programming File Generator)

Table 20. Input File Properties Dialog Box (Programming File Generator)

Allows you to specify options for bitstream authentication, co-signing, and encryption security. To access, select an `.sof` or `.rbf` in the **Input files** tab in the **Programming File Generator**, and click **Properties**.

Option	Description
Bootloader	Specifies an ASCII text file in Intel hexadecimal format that contains configuration data for programming a parallel data source, such as a configuration device or a mass storage device. The parallel data source in turn configures an SRAM-based Intel device
Enable signing tool	Enables the signing tool that checks for a required Privacy Enhanced Mail Certificates file (`.pem`) for the **Private key file**, and a Quartus Co-Signed Firmware file (`.zip`) for the **Co-signed firmware** option.
Private key file	Specifies the private `.pem` file required to sign the configuration bitstream when using the signing tool. If your `.pem` is password-protected, you are prompted to enter the password.
Co-signed firmware	Specifies the firmware source (`.zip`) required to include the signed firmware in the configuration bitstream.
Finalize encryption	Finalizes the configuration bitstream encryption.
Encryption key file	Specifies the Encryption Key File (`.qek`) required to decrypt the configuration bitstream file.

(6)

1.6.7. Convert Programming File Dialog Box

Allows you to convert or combine one or more secondary programming files that support alternative device configuration schemes, such as flash programming, partial reconfiguration, or remote system update.

Table 21. Convert Programming File Dialog Box Settings

Setting	Description
Programming file type	Allows you to specify a secondary programming file format for conversion of a primary programming file. The Generic Flash Programmer supports only the `.jic` file type.
Configuration device	Allows you to select a predefined or define a new configuration device. Click the (**...**) button to define a new device and programming flow.
Mode	Allows you to select the method of device configuration. The Generic Flash Programmer supports only the **Active Serial** or **Active Serial x4** modes.
Output file	Specifies the location of the files that **Convert Programming File** generates. By default this location is the top-level project directory.
Input files to convert	Specifies one or more primary programming files for conversion or combination into one or more secondary programming files for alternative programming methods.

(6) Security options not yet available for Intel Agilex devices.

1.6.8. Compression and Encryption Settings (Convert Programming File)

The compression and encryption settings allow you to specify options for compression and encryption key security for the device configuration SRAM Object File (`.sof`). To access these settings, select the `.sof` in the **Input files to convert** list in the **Convert Programming File** dialog box, and click **Properties**.

Table 22. SOF File Properties: Bitstream Encryption Dialog Box (Convert Programming Files)

Allows you to specify options for compression and encryption key security for the device configuration SRAM Object File (`.sof`). To access, select an `.sof` in the **Input files to convert** list in the **Convert Programming Files** dialog box, and click **Properties**.

Option	Description
Compression	Applies compression to the bitstream to reduce the size of your programming file. The Intel Quartus Prime Assembler can generate a compressed bitstream image that reduces configuration file size by 30% to 55% (depending on the design). The FPGA device receives the compressed configuration bitstream, and then can decompress the data in real-time during configuration. This option is unavailable whenever **Generate encrypted bitstream** is enabled.
Enable decompression during partial reconfiguration	Enables the option bit for bitstream decompression during Partial Reconfiguration.
Generate encrypted bitstream	Generates an encrypted bitstream configuration image. You then generate and specify an encryption key file (`.ekp`) for device configuration. This option is unavailable whenever **Compression** is enabled.
Enable volatile security key	Allows you to encrypt the `.sof` file with volatile (enabled) or non-volatile (disabled) security key.
Generate encryption lock file	Specifies the name of the encryption lock file (`.elk`) that **Convert Programming Files** generates.
Generate key programming file	Specifies the name of the key programming file (`.key`) that **Convert Programming Files** generates.
Use key file	• **Key 1 file**—specifies the name of Key 1 `.key` file. • **Key 2 file**—specifies the name of Key 2 `.key` file.
Key entry	Specifies the keys for bitstream decryption.
Security options	The following options allow you to enable or disable features that impact device security for the configuration bitstream. • **Disable partial reconfiguration**—disables use of partial reconfiguration for the bitstream. • **Disable key-related JTAG instructions**—disables use of key-related JTAG instructions for the bitstream. • **Disable other extended JTAG instructions**—disables use of other JTAG instructions for the bitstream. • **Force the external JTAG pins into BYPASS mode**—forces the external JTAG pins into BYPASS mode. You can specify **Off**, **Turns On Until the Next Full Configuration**, **Turns on until the next Power-On-Reset event**, **Turns on by blowing the corresponding fuses**,
Design Security Feature Disclaimer	Acknowledges required acceptance of Design Security Disclaimer.

Related Information

Enabling Bitstream Encryption or Compression for Intel Arria 10 and Intel Cyclone 10 GX Devices on page 22

1.6.9. SOF Data Properties Dialog Box (Convert Programming File)

Allows you to define flash memory pages that store configuration data. To access from the **Convert Programming File** dialog box, click the **SOF Data** item and click the **Properties** button.

The following settings are available:

Table 23. SOF Data Properties Dialog Box Settings

Setting	Description
Pages	Configuration devices can store multiple configuration bitstreams in flash memory, called pages. CFI configuration devices can store up to eight configuration bitstreams. Some Intel FPGA devices can store multiple configuration bitstreams, including the factory image.
Address mode for selected pages	The options are: • **Auto**—automatically allocates a block in the flash device to store the data. • **Block**—specify the start and end address of the flash partition. • **Start**—specify the start address of the partition. The tool assigns the end address of the partition based on the input data size.
Start address	Specifies the start address of the partition. Only enabled when **Address Mode** is **Block** or **Start**.
End address	Specifies the end address of the partition. Only enabled when **Address Mode** is **Block**.

1.6.10. Select Devices (Flash Loader) Dialog Box

Allows you to select the device that controls loading of configuration data into a flash memory device. To access from the **Programming File Generator**, click the **Select** button for **Flash loader** in the **Configuration Device** tab. To access from the **Convert Programming File** dialog box, select the **Flash Loader** item and click **Add Device**.

The following settings are available:

Table 24. Flash Loader (Select Devices Dialog Box)

Option	Description
Device family	Specifies the family of the flash loader device.
Device name	Specifies the name of the flash loader device.

1.7. Generic Flash Programmer User Guide Revision History

Document Version	Intel Quartus Prime Version	Changes
2019.12.16	19.4	• Added programming file generation support for Intel Agilex devices. • Noted Intel Agilex security feature limitations.
2019.09.30	19.3	• Added new "Enabling Bitstream Authentication (Programming File Generator)" topic. • Added new "Specifying Additional Physical Security Settings (Programming File Generator)" topic. • Added new "Enabling Bitstream Encryption (Programming File Generator)" topic.

continued...

Document Version	Intel Quartus Prime Version	Changes
		• Updated name of “Authentication and Encryption” tab to “Security” tab. • Added footnote about programming file support for Intel Agilex devices. • Described new **More Security Settings** dialog box. • Updated "Defining a New Flash Memory Configuration Device" for new default flow template location. • Updated "Device & pin Options" topic to reflect new Security settings tab. • Updated "Configuration Device Tab" topic to reflect Custom database directory option. • Referenced compilation support for Intel Agilex devices. • Added "More Security Options Dialog Box" topic. • Added new “Bitstream Co-Signing Security Settings” topic. • Updated "SOF File Properties: Bitstream Encryption Dialog Box" topic. • Added new steps to “Full Erase of Flash Memory Sectors” topic.
2019.06.19	19.1	Corrected document title.
2019.05.15	19.1	First public release.

1.8. Generic Flash Programmer Document Archive

If the table does not list a software version, the user guide for the previous software version applies.

Intel Quartus Prime Version	User Guide
19.3	Generic Flash Programmer User Guide
19.1	Generic Flash Programmer User Guide