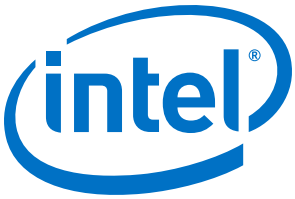


Intel® Quartus® Prime Pro Edition User Guide

Programmer

Updated for Intel® Quartus® Prime Design Suite: **20.4**

Subscribe
Send Feedback

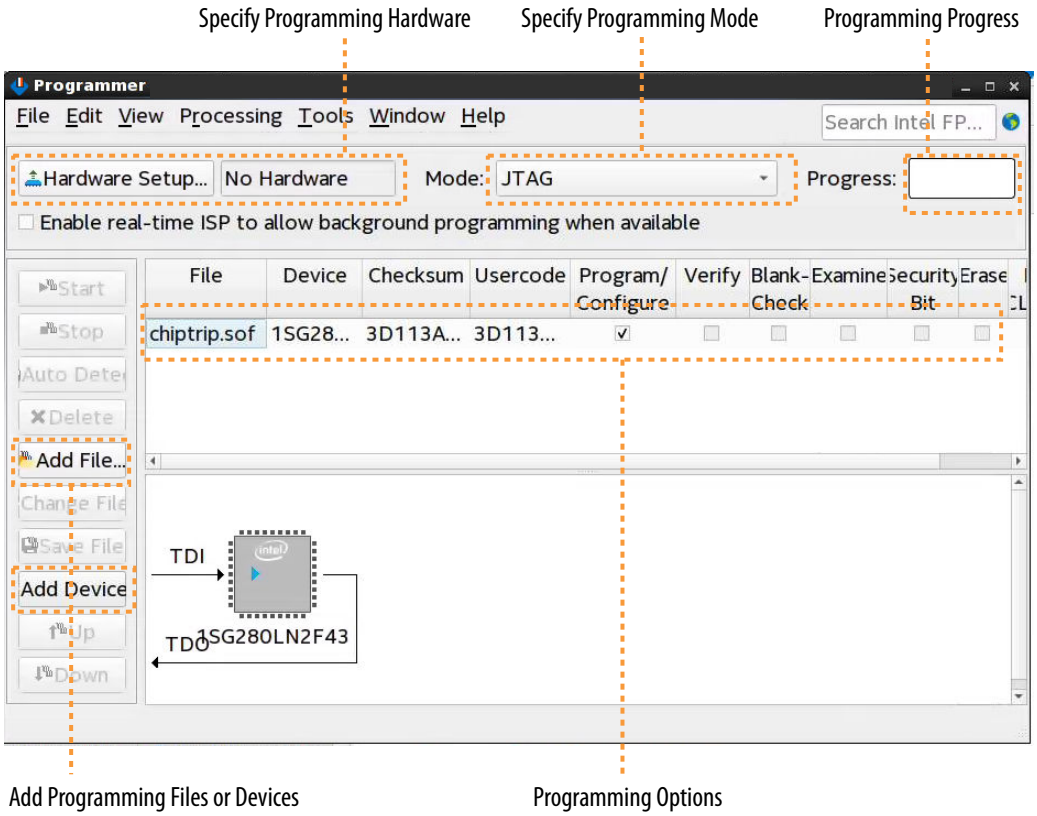
UG-20134 | 2020.12.14
Latest document on the web: PDF | HTML

Contents

1. Intel® Quartus® Prime Programmer User Guide

The Intel® Quartus® Prime Programmer allows you to program and configure Intel CPLD, FPGA, and configuration devices. Following full design compilation, you generate the primary device programming files in the Assembler, and then use the Programmer to load the programming file to a device. This user guide details Intel FPGA programming file generation and use of the Intel Quartus Prime Programmer.

Figure 1. Intel Quartus Prime Programmer



Related Information

- Generating Primary Device Programming Files on page 5
- Generating Secondary Programming Files on page 6
- Enabling Bitstream Security for Intel Stratix 10 Devices on page 18
- Using the Intel Quartus Prime Programmer on page 39
- Programming with Flash Loaders on page 45

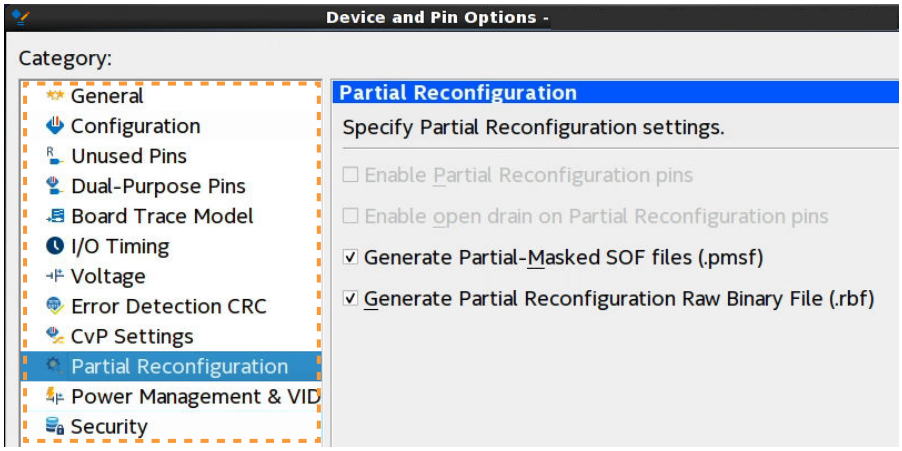
1.1. Generating Primary Device Programming Files

By default, the Compiler's Assembler module generates the primary device programming files at the end of full compilation. Alternatively, you can start the Assembler independently any time after design place and route to generate primary device programming files, such as SRAM Object Files (`.sof`) for configuration of Intel FPGAs.

Follow these steps to generate primary device programming files:

1. To specify programming options that enable features in the primary device programming file, such as **Configuration**, **Error Detection CRC**, and device **Security** options, click **Assignments ➤ Device ➤ Device & Pin Options**. Device & Pin Options Dialog Box on page 49 describes all options.(1)

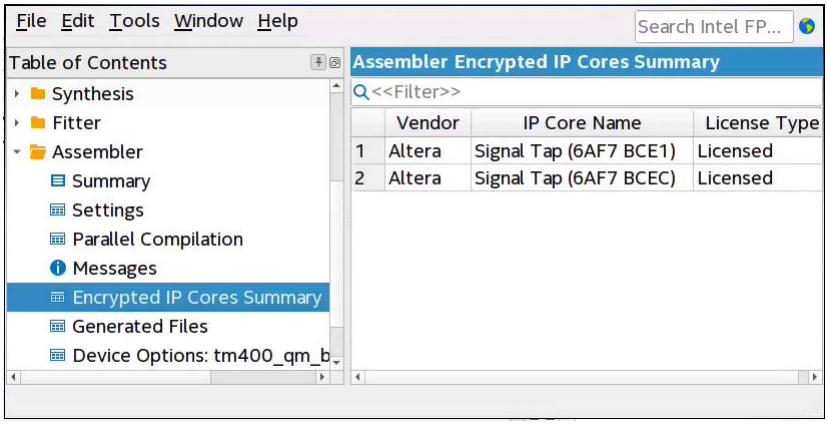
Figure 2. Device & Pin Options Dialog Box (Intel Stratix® 10 Design)



2. To generate primary device programming files, click **Processing ➤ Start ➤ Start Assembler**, or double-click **Assembler** on the Compilation Dashboard. The Assembler generates the programming files according to the options you specify.
3. After running the Assembler, view detailed information about programming file generation, including the programming file Summary and Encrypted IP information in the Assembler report folder in the Compilation Report.

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

Figure 3. Assembler Reports



Note: Each successive release of the Intel Quartus Prime software typically includes:

- Added support for new features in supported FPGA devices.
- Added support for new devices.
- Efficiency and performance improvements.
- Improvements to compilation time and resource use of the design software.

Due to these improvements, different versions of the Intel Quartus Prime Pro Edition, Intel Quartus Prime Standard Edition, and Intel Quartus Prime Lite Edition software can produce different programming files from release to release.

1.2. Generating Secondary Programming Files

After generating primary device programming files, you can optionally generate one or more derivative programming files for alternative device configurations, such as flash programming, partial reconfiguration, remote system update, Configuration via Protocol (CvP), or hard processor system (HPS) core configuration.

You can use the **Programming File Generator** or **Convert Programming Files** dialog box to generate secondary programming files:

- The **Programming File Generator** supports advanced programming features and is optimized for Intel Agilex, Intel Stratix® 10, Intel MAX® 10, and Intel Cyclone® 10 LP devices.
- The **Convert Programming Files** dialog box supports all devices released prior to Intel Stratix 10 devices.

Table 1. Secondary Programming File Generators

	Programming File Generator	Convert Programming Files	
Device Support	• Intel Agilex • Intel Stratix 10 • Intel MAX 10 • Intel Cyclone 10 LP	• Intel Arria® 10 • Intel Cyclone 10GX and LP • Intel MAX 10	APEX20K, Arria II GX and GZ, Arria V, Cyclone, Cyclone II, Cyclone III and LS, Cyclone IV E and GX, Cyclone V,

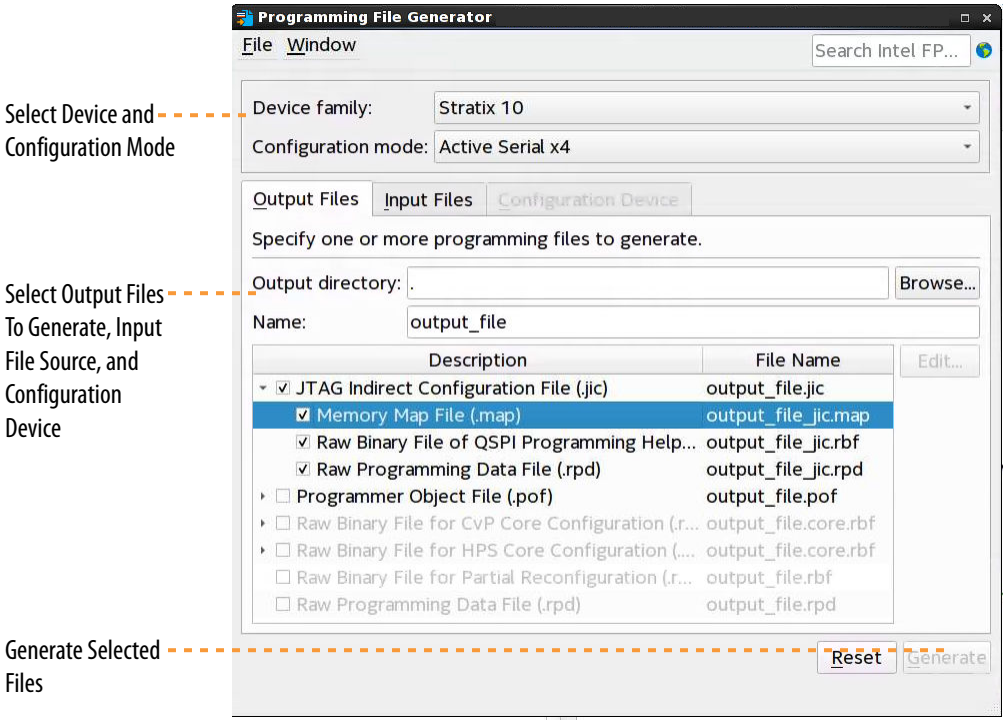
	Programming File Generator	Convert Programming Files	
			HardCopy® III, HardCopy II, HardCopy IV, MAX V, Stratix, Stratix II, Stratix III, Stratix IV, Stratix V

1.2.1. Generating Secondary Programming Files (Programming File Generator)

Follow these steps to generate secondary programming files for alternative device programming methods with the **Programming File Generator**.

1. Generate the primary programming files for your design, as Generating Primary Device Programming Files on page 5 describes.
2. Click **File ➤ Programming File Generator**.
3. For **Device family**, select your target device. The options available in the **Programming File Generator** change dynamically, according to your device and configuration mode selection.
4. For **Configuration mode**, select an Active Serial mode that your device supports. Configuration Modes (Programming File Generator) on page 10 describes all modes.
5. On the **Output Files** tab, enable the checkbox for generation of the file you want to generate. The **Input Files** tab is now available. Secondary Programming Files (Programming File Generator) on page 10 describes all output files.
6. Specify the **Output directory** and **Name** for the file you generate. Output Files Tab Settings (Programming File Generator) on page 58 describes all options.

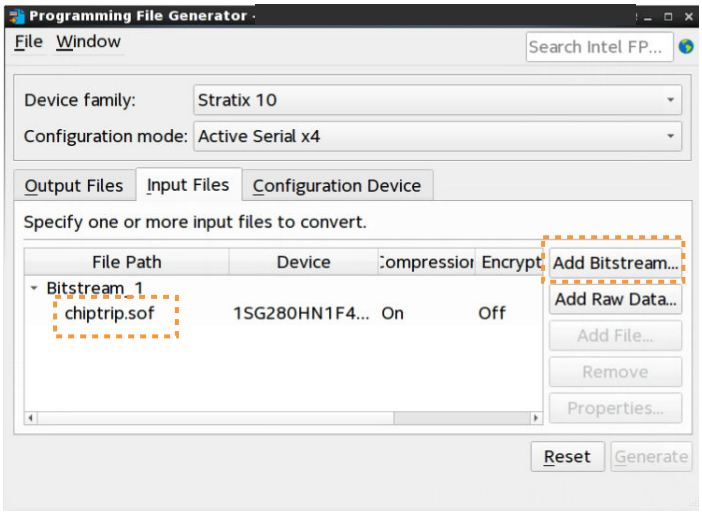
Figure 4. Programming File Generator



7. To specify a `.sof` file that contains the configuration bitstream data, on the **Input Files** tab, click **Add Bitstream**. To include raw data, click **Add Raw Data** and specify a Hexadecimal (Intel-Format) File (`.hex`) or Binary (`.bin`) file.
8. To specify bitstream authentication or encryption security settings for the file, select the `.sof` and click **Properties**, as Enabling Bitstream Authentication (Programming File Generator) on page 19 describes.(2)

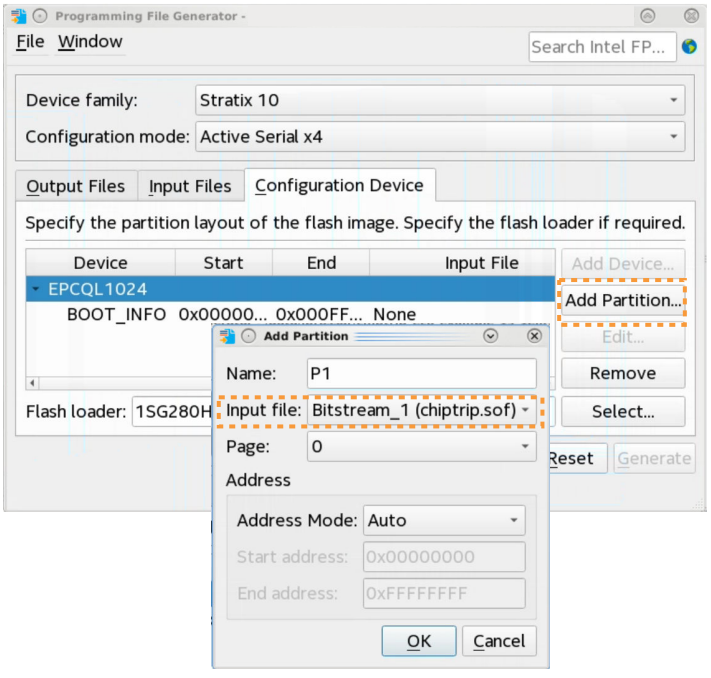
(2) Security options not yet available for Intel Agilex devices.

Figure 5. Input Files Tab



9. To specify the `.sof` file that occupies the flash memory partition, click **Add Partition** on the **Configuration Device** tab. Add Partition Dialog Box (Programming File Generator) on page 60 describes all options.

Figure 6. Add Flash Partition



10. To select a supported flash memory device and predefined programming flow, click **Add Device** on the **Configuration Device** tab. Alternatively, click **<<new device>>** to define a new flash memory device and programming flow. Configuration Device Tab Settings on page 59 describes all settings.
11. Click the **Select** button for **Flash Loader** and select the device that controls loading of the flash memory device. Select Devices (Flash Loader) Dialog Box on page 63 describes all settings.
12. After you specify all options in **Programming File Generator**, the **Generate** button enables. Click **Generate** to create the files.

1.2.1.1. Configuration Modes (Programming File Generator)

Select one of the following **Configuration modes** in **Programming File Generator** for generation of secondary programming files:

Table 2. Programming File Generator Configuration Modes

Programming Mode	Description	Supports Devices
Active Serial x4	For storing configuration data in a low-cost serial configuration device with non-volatile memory and four-pin interface. Serial configuration devices provide a serial interface to access the configuration data. During device configuration, Intel Stratix 10 devices read the configuration data through the serial interface, decompress the data if necessary, and configure their SRAM cells.	• Intel Agilex • Intel Stratix 10
AVST x8	The Avalon®-ST configuration mode uses an external host, such as a microprocessor or Intel MAX 10 device. The external host controls the transfer of configuration data from an external storage such as flash memory to the FPGA. The design that controls the configuration process resides in the external host. You can use the PFL II IP core with an Intel MAX 10 device as the host to read configuration data from a flash memory device that configures an Intel Stratix 10 FPGA.	
AVST x16		
AVST x32		
1-Bit Passive Serial	An external controller passes configuration data to one or more FPGA devices via a serial data stream. The FPGA device is a slave device with a 5-wire interface to the external controller. The external controller can be an intelligent host such as a microcontroller or CPU.	Intel Cyclone 10 LP
Active Serial	Stores configuration data in a low-cost serial configuration device with non-volatile memory and four-pin interface.	
Internal Configuration	Uses a `.pof` file for internal configuration of the Intel MAX 10 device's Configuration Flash Memory (CFM) and User Flash Memory (UFM) via a download cable Intel Quartus Prime Programmer.	Intel MAX 10

1.2.1.2. Secondary Programming Files (Programming File Generator)

After generating primary device programming files, you can generate the following secondary device programming files with the **Programming File Generator** for alternative device configuration modes:

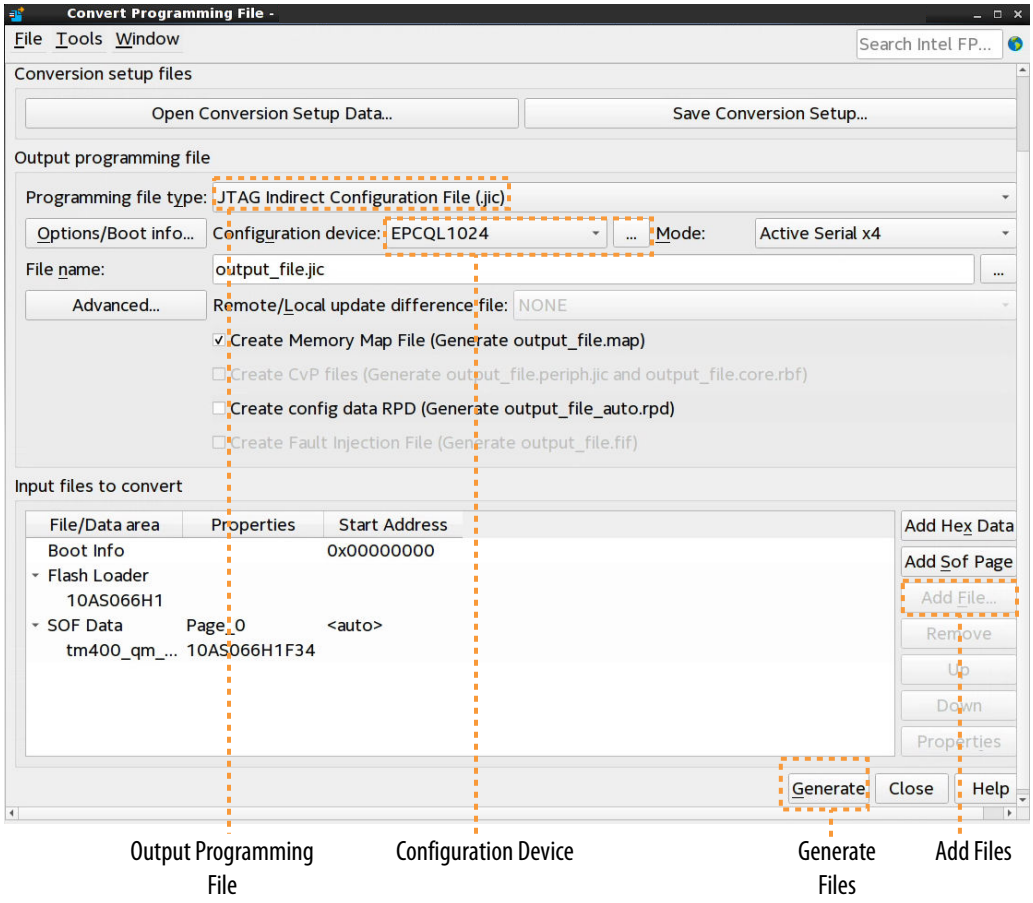
Table 3. Programming File Generator Output File Types

Programming File Type	Extension	Description
Hexadecimal (Intel-Format) Output File for SRAM	`.hexout`	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.
JTAG Indirect Configuration File	`.jic`	Proprietary Intel FPGA file type that stores serial flash programming data for programming via Intel FPGA JTAG pins. This method only supports Active Serial configuration. Before programming the flash, the Programmer first configures the FPGA with the Serial Flash Helper Design.
Map File	`.map`	A text file containing the byte addresses of pages and data stored in the memory of a configuration device for
Programmer Object File	`.pof`	A binary file used by the Programmer to program a flash memory device via active serial header, or to program a flash memory device via the Parallel Flash Loader Intel FPGA IP.
Raw Binary File	`.rbf`	Configuration bitstream file for use with a third-party data source, partial reconfiguration, or HPS data source. Supports Passive Serial (PS) and Avalon-Streaming (AVST) modes.
Raw Binary File for CvP Core Configuration	`.rbf`	A binary file that containing logic that is programmed by configuration (CRAM) for CvP phase 2. The core bitstream is in `.rbf` format.
Raw Binary File for HPS Core Configuration		A binary file that containing logic that is programmed by configuration (CRAM) for HPS configuration phase 2. The core bitstream is in `.rbf` format.
Raw Programming Data File	`.rpd`	Stores data for configuration with third-party programming hardware. You generate Raw Programming Data Files from a `.pof` or `.sof`. The `.rpd` file is a subset of a `.pof` or `.jic` that includes only device-specific binary programming data for Active Serial configuration scheme with EPCS or EPCQ serial configuration devices and remote system update.
Tabular Text File	`.ttf`	A TTF contains the decimal equivalent of a Raw Binary File (.rbf).

1.2.2. Generating Secondary Programming Files (Convert Programming File Dialog Box)

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.

Figure 7. Convert Programming File Dialog Box



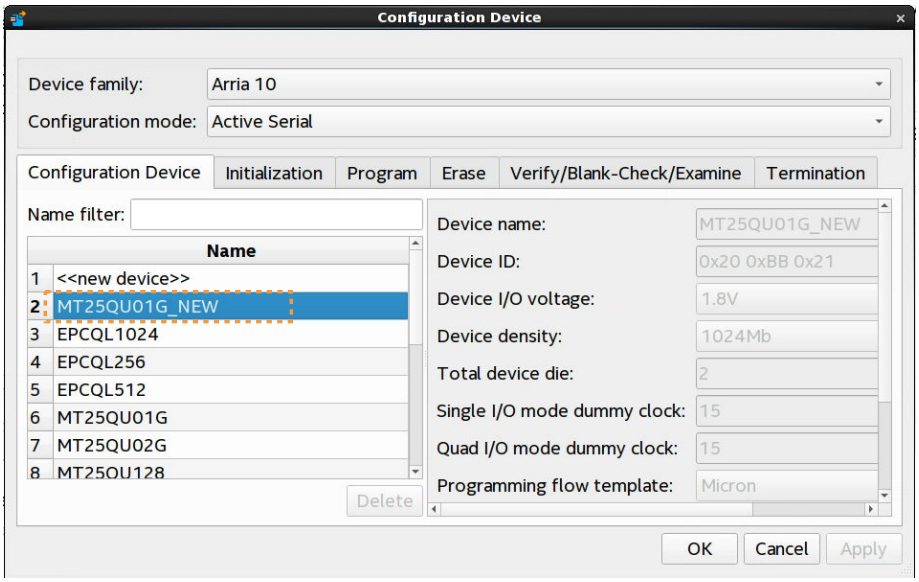
The options available in the **Convert Programming File** dialog box change dynamically, according to your device and configuration mode selection.

1. Generate the primary programming files for your design, as Generating Primary Device Programming Files on page 5 describes.
2. Click **File ➤ Convert Programming Files**.
3. Under **Output programming file**, select the **Programming file type** that you want to generate. Secondary Programming Files (Convert Programming Files) on page 14 describes all file options.
4. Specify the **File name** and output directory **(...)** for the file that you generate.
5. For the configuration **Mode**, select **Active Serial x4** or **Active Serial**. Configuration Modes (Convert Programming Files) on page 15 describes all modes.

   Note: Intel Stratix 10 devices support only **Active Serial x4**.

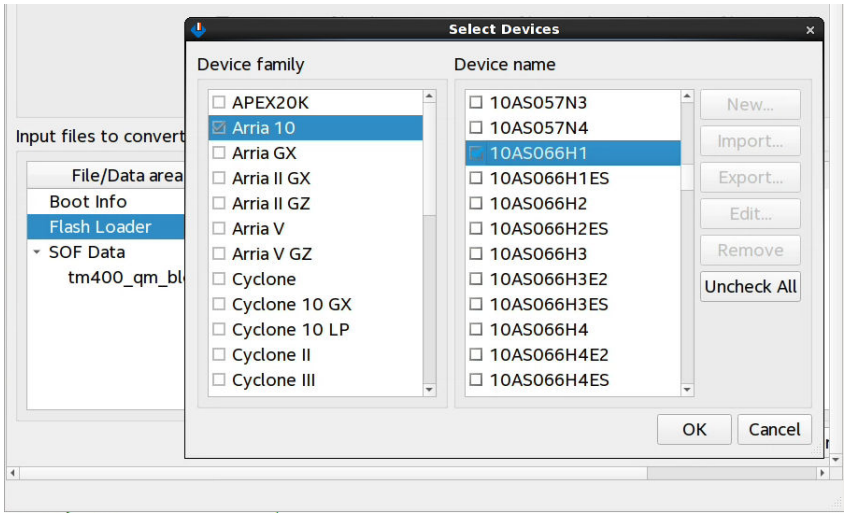
6. To specify the **Configuration device**, click the **(...)** button to select a supported flash memory device and predefined programming flow. When you select a predefined device, you cannot modify any setting. Alternatively, click **<<new device>>** to define a new flash memory device and programming flow. Configuration Device Tab Settings on page 59 describes all settings.

Figure 8. 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 23 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 9. 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.

1.2.2.1. Secondary Programming Files (Convert Programming Files)

After generating primary device programming files, you can generate the following secondary device programming files with the **Convert Programming Files** dialog box for alternative device configuration modes:

Table 4. Output File Types

Programming File Type	Extension	Description
CvP Files	`.jic/.rbf`	Files required for CvP configuration.
Hexadecimal (Intel-Format) Output File for SRAM	`.hexout`	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.
JTAG Indirect Configuration File	`.jic`	Proprietary Intel FPGA file type that stores serial flash programming data for programming via Intel FPGA JTAG pins. This method only supports Active Serial configuration. Before programming the flash, the Programmer first configures the FPGA with the Serial Flash Helper Design.
Memory Map File	`.map`	Contains the byte addresses of pages and HEX data stored in the memory of an EPC4, EPC8, or EPC16 configuration device. The MAP File stores the start and end addresses of the **Main Block Data** and **Bottom Boot Data** items, and the start and end addresses of pages within the **Main Block Data** item.
Partial-Masked SRAM Object Files	`.pmsf`	Contains the partial-mask bits for configuration of a PR region. The `.pmsf` file contains all the information for creating PR bitstreams.
Merged Mask Setting File	`.msf`	Contains the mask bits for the static region in a PR design.
Programmer Object File	`.pof`	A binary file that contains the data for programming non-volatile Intel MAX 10, MAX V, MAX II, or flash memory devices that can configure Intel FPGA devices. A Programmer consists of a remote

continued...

Programming File Type	Extension	Description
		update enabled `.pof` and additional remote update enabled `.sof` that you used to program configuration devices in remote update configuration mode.
Raw Binary File	`.rbf`	Configuration bitstream file for use with a third-party data source, partial reconfiguration, or HPS data source. Supports Passive Serial (PS) and Avalon-Streaming (AVST) modes.
Raw Programming Data File	`.rpd`	Stores data for configuration with third-party programming hardware. You generate Raw Programming Data Files from a `.pof` or `.sof`. The `.rpd` file is a subset of a `.pof` or `.sof` that includes only device-specific binary programming data for Active Serial configuration scheme with EPCS or EPCQ serial configuration devices and remote system update. The `.rpd` file content has one bit swapped in comparison with the output file.
Tabular Text File	`.ttf`	A TTF contains the decimal equivalent of a Raw Binary File (.rbf).

1.2.2.2. Configuration Modes (Convert Programming Files)

Select one of the following **Configuration modes** in **Convert Programming Files** for generation of secondary programming files:

Table 5. Convert Programming Files Configuration Modes

Programming Mode	Description
1-Bit/2-Bit/4-Bit/8-Bit Passive Serial	An external controller passes configuration data to one or more FPGA devices via a serial data stream. The FPGA device is a slave device with a 5-wire interface to the external controller. The external controller can be an intelligent host such as a microcontroller or CPU, or the Intel Quartus Prime Programmer, or an EPC2 or EPC16 configuration device.
Active Parallel	Supports configuration devices using commodity 16-bit parallel flash memories to control the configuration interface.
Active Serial	For storing configuration data in a low-cost serial configuration device with non-volatile memory. Serial configuration devices provide a serial interface to access the configuration data. During device configuration, the device reads the configuration data through the serial interface, decompresses the data if necessary, and configures their SRAM cells.
Active Serial x4	
AVST x8/x16/x32	The Avalon-ST configuration mode uses an external host, such as a microprocessor or Intel MAX 10 device. The external host controls the transfer of configuration data from an external storage such as flash memory to the FPGA. The design that controls the configuration process resides in the external host. You can use the PFL II IP core with an Intel MAX 10 device as the host to read configuration data from a flash memory device that configures an FPGA.
Passive Parallel Synchronous	An external controller, such as a CPU, loads the design data into a device via a common data bus. Data is latched by the device on the first rising edge of a CPU-driven clock signal. The next eight falling clock edges serialize this latched data within the device. The device latches the next 8-bit byte of data on every eighth rising edge of the clock signal until the device is completely configured.
Passive Parallel Asynchronous	An external controller, such as a CPU, loads the design data into a device via a common data bus. The device accepts a parallel byte of input data. Intelligent communication between the external controller and the device allows the external controller to configure the device.
Internal Configuration	Uses a `.pof` file for internal configuration of the Intel MAX 10 device's Configuration Flash Memory (CFM) and User Flash Memory (UFM) via a download cable Intel Quartus Prime Programmer.

1.2.2.3. Debugging the Configuration (Convert Programming Files)

Click the **Advanced** option in the **Convert Programming Files** dialog box to debug the file conversion configuration. Only choose advanced settings that apply to the design's target Intel FPGA device.

Changes in the **Advanced Options** dialog box affect `.pof`, `.jic`, `.rpd`, and `.rbf` file generation.

The following table describes the **Advanced Options** settings:

Table 6. Advanced Options Settings

Option Setting	Description	Values
Disable EPCS/EPCQ ID check	Directs the FPGA to skips the EPCS/EPCQ silicon ID verification. Applies to single and multi device AS configuration modes on all devices.	Default setting is ON (EPCS/EPCQ ID check is enabled).
Disable AS mode CONF_DONE error check	Directs the FPGA to skip the CONF_DONE error check. Applies to single- and multi-device (AS) configuration modes on all devices.	Default setting is OFF (AS mode CONF_DONE error check is enabled).
Program Length Count adjustment	Specifies the offset you can apply to the computed PLC of the entire bitstream. Applies to single- and multi-device (AS) configuration modes on all FPGA devices.	Integer (Default = 0)
Post-chain bitstream pad bytes	Specifies the number of pad bytes appended to the end of an entire bitstream.	If the bitstream of the last device is uncompressed, default value is 0. Otherwise, default is 2
Post-device bitstream pad bytes	Specifies the number of pad bytes appended to the end of the bitstream of a device. Applies to all single-device configuration modes on all FPGA devices.	Zero or positive integer. Default is 0
Bitslice Padding Value	Specifies the padding value used to prepare bitslice configuration bitstreams, such that all bitslice configuration chains simultaneously receive their final configuration data bit. Use only in 2, 4, and 8-bit PS configuration mode, when you use an EPC device with the decompression feature enabled. Applies to all FPGA devices that support enhanced configuration devices.	0 or 1 Default value is 1

The following table lists possible symptoms of a failing configuration, and describes the advanced options necessary for configuration debugging.

Failure Symptoms	Disable EPCS/ EPCQ ID Check	Disable AS Mode CONF_DONE Error Check	PLC Settings	Post-Chain Bitstream Pad Bytes	Post-Device Bitstream Pad Bytes	Bitslice Padding Value
Configuration failure occurs after a configuration cycle.	—	Yes	Yes	Yes (3)	Yes (4)	—
Decompression feature is enabled.	—	Yes	Yes	Yes (3)	Yes (4)	—
Encryption feature is enabled.	—	Yes	Yes	Yes (3)	Yes (4)	—
CONF_DONE stays low after a configuration cycle.	—	Yes	Yes (5)	Yes (3)	Yes (4)	—
CONF_DONE goes high momentarily after a configuration cycle.	—	Yes	Yes (6)	—	—	—
FPGA does not enter user mode even though CONF_DONE goes high.	—	—	—	Yes (3)	Yes (4)	—
Configuration failure occurs at the beginning of a configuration cycle.	Yes	—	—	—	—	—
EPCS128	Yes	—	—	—	—	—
Failure in .pof generation for EPC device using Intel Quartus Prime Convert Programming File Utility when the decompression feature is enabled.	—	—	—	—	—	Yes

(3) Use only for multi-device chain

(4) Use only for single-device chain

(5) Start with positive offset to the PLC settings

(6) Start with negative offset to the PLC settings

1.3. Enabling Bitstream Security for Intel Stratix 10 Devices

Intel Stratix 10 devices provide flexible and robust security features to protect sensitive data, intellectual property, and device hardware from physical and remote attacks. The Intel Stratix 10 device architecture supports bitstream authentication and encryption security features. The Assembler applies bitstream compression automatically to reduce file size whenever you use authentication or encryption.

- **Bitstream Authentication**—verifies that the configuration bitstream and firmware are from a trusted source. Enable additional co-signing device firmware authentication to ensure that only signed firmware runs on the HPS or FPGA, and to authorize HPS JTAG debugging. Enable authentication security by specifying a first level signature chain file (`.qky`) for the **Quartus Key File** option (**Device and Pin Options** dialog box), as Enabling Bitstream Authentication (Programming File Generator) on page 19 describes.[7]
- **Bitstream Encryption**—protects proprietary or sensitive data from view or extraction in the configuration bitstream using an Advanced Encryption Standard (AES) 256-bit or 384-bit security key. Encryption also provides side-channel protection from non-intrusive attack. You can store the owner AES key in eFuses or BBRAM. Enable encryption by turning on the **Enable programming bitstream encryption** option (**Device and Pin Options** dialog box), as Enabling Bitstream Encryption (Programming File Generator) on page 22 describes.

Table 7. Intel Stratix 10 Bitstream Authentication Files

Term	Description	Extension
First Level Signature Chain Key File	File you generate that specifies 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.	`.qky`
Root Key File	File you generate that anchors the first level signature chain to a known root key. The FPGA calculates the hash of the root entry and checks if it matches the expected hash. The Assembler appends the root key to the programming file and stores the key in eFuses.	`.qky`
Design Signing Key File	File you generate and append to the root key that authenticates the bitstream in the SDM to allow configuration of the device with the pending bitstream. Use separate design signing keys for the FPGA and HPS for highest security.	`.pem`
Firmware Co-signing Key File	Files provided in `<install>\common\devinfo\programmer \firmware` that includes the owner signature and firmware file that you use to sign the firmware to run on the FPGA or HPS.	`.zip`
Signed HPS Certificate File	Specifies a secure HPS debug certificate that permits access to the JTAG interface for HPS debugging. A secure HPS debug certificate is valid until you power down or reconfigure the device.	`.cert`

Note: Intel Arria 10 and Intel Cyclone 10 GX devices do not support bitstream authentication.

Related Information

Intel Stratix 10 Device Security User Guide
For detailed device security configuration steps.

[7] Bitstream authentication is available only for Intel Stratix 10 devices that include the AS (Advanced Security) ordering code suffix and all Intel Stratix 10 DX devices.

1.3.1. 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.(8)

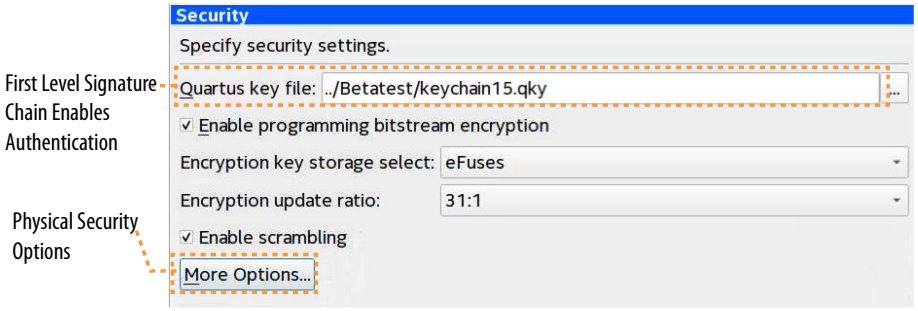
After you specify the `.qky` in Assembler settings, the Assembler appends the first level signature chain to the configuration `.sof` that you generate.

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 57 describes all options.

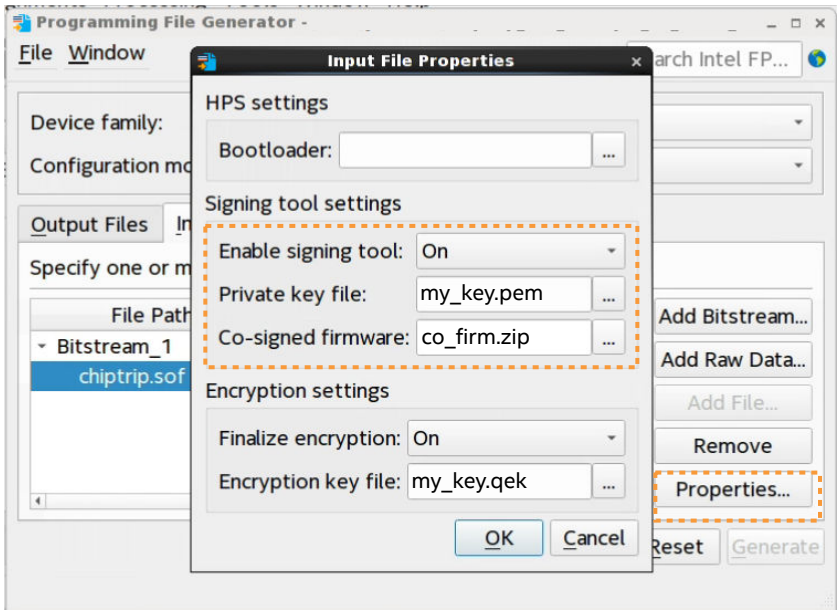
Figure 10. Security Tab (Device and Pin Options)



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

4. Generate primary device programing files in the Assembler, as Generating Primary Device Programming Files on page 5 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 Generating 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 11. 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

- Device & Pin Options Dialog Box on page 49
- Specifying Additional Physical Security Settings (Programming File Generator) on page 21
- Intel Stratix 10 Device Security User Guide
  For detailed information on generating device security keys.

1.3.2. 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 12. More Security Options Dialog Box

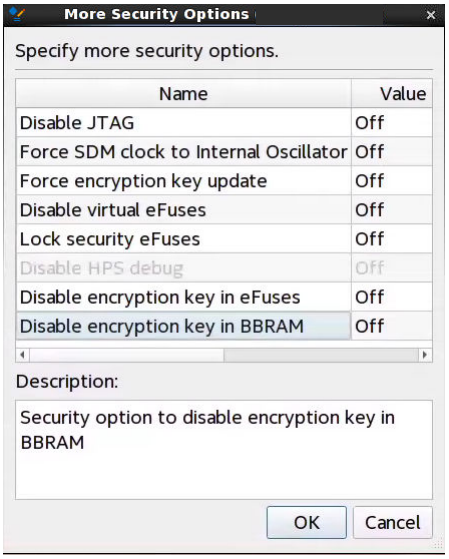


Table 8. More Security Options Dialog Box Settings

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.	

continued...

Option	Description	Values
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**.

Related Information

Enabling Bitstream Authentication (Programming File Generator) on page 19

1.3.3. 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:

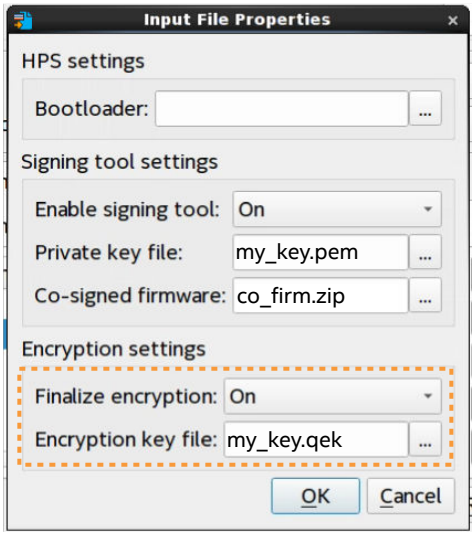
1. Generate a First Level Signature Chain that includes the root key and one or more design signing keys, as *Intel Stratix 10 Device Security User Guide* describes.
2. Click **Assignments ➤ Device ➤ Device and Pin Options ➤ Security**.
3. For the **Quartus key file** setting, specify the first level 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 9. 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 Generating Primary Device Programming Files on page 5 describes.
6. Generate a `.jic` or `.rbf` secondary programming file, as Generating 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 13. 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**.

Related Information

Input Files Tab Settings (Programming File Generator) on page 58

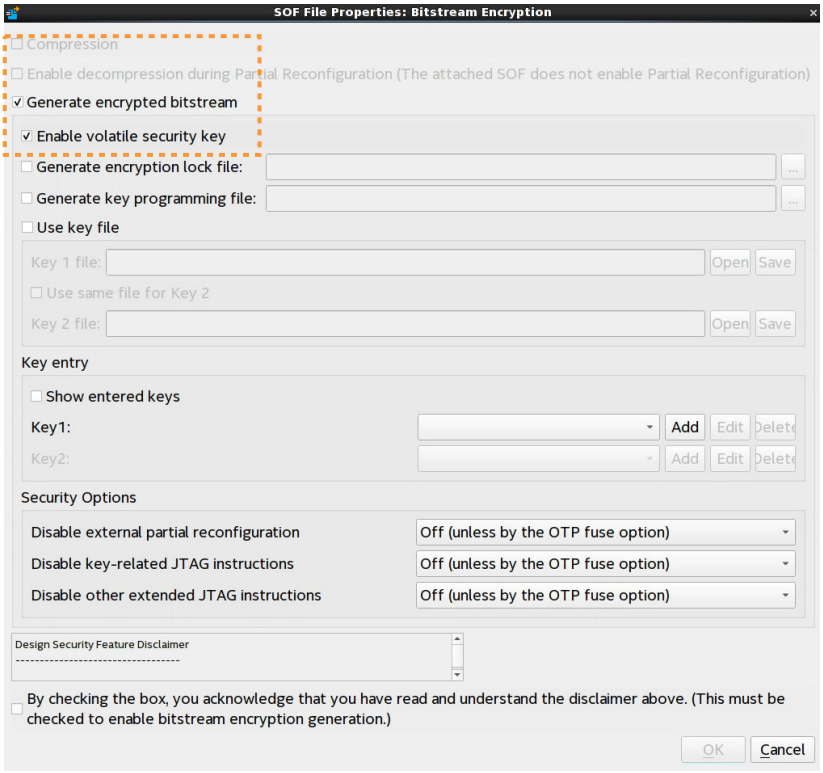
1.4. Enabling Bitstream Encryption or Compression for Intel Arria 10 and Intel Cyclone 10 GX Devices

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 14. 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 61 describes.
6. Click **OK**.

Related Information

- 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.

1.5. Generating Programming Files for Partial Reconfiguration

The following sections describe generation of bitstream and other files for partial reconfiguration.

1.5.1. Generating PR Bitstream Files

For Intel Stratix 10 designs, the Assembler generates a configuration `.rbf` automatically at the end of compilation. For Intel Arria 10 and Intel Cyclone 10 GX designs, use any of the following methods to process the PR bitstreams and generate the Raw Binary File (`.rbf`) file for reconfiguration.

Note: The Intel Quartus Prime Pro Edition software supports compilation of PR designs for Intel Agilex devices, but does not yet support Intel Agilex PR bitstream generation.

Generating PR Bitstreams During Compilation

Follow these steps to generate the `.rbf` file during compilation:

1. Add the following assignments to the revision `.qsf` to automatically generate the required PR bitstreams following compilation:

```
set_global_assignment -name GENERATE_PR_RBF_FILE ON
set_global_assignment -name ON_CHIP_BITSTREAM_DECOMPRESSION OFF
```

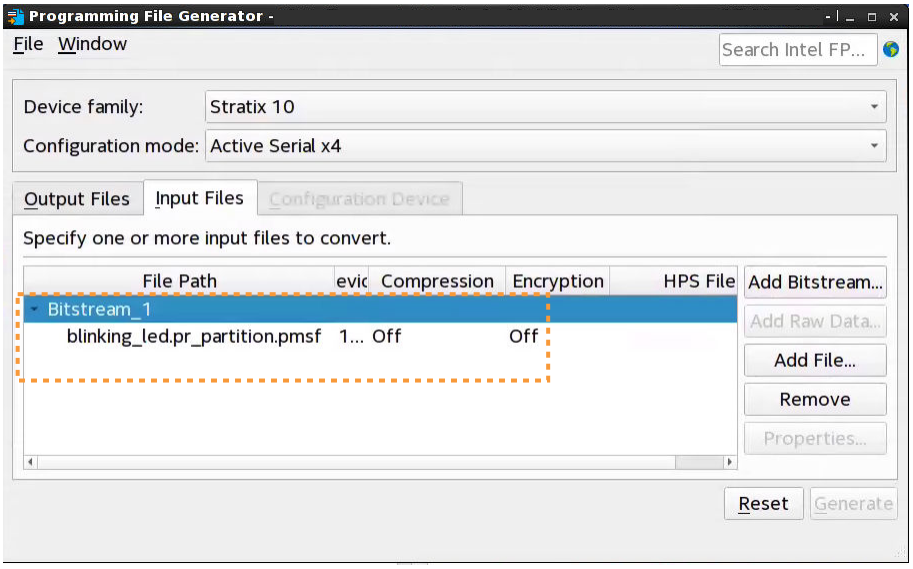
2. To compile the revision and generate the `.rbf`, click **Processing ➤ Start Compilation**.

Generating PR Bitstreams with Programming File Generator

Follow these steps to generate the `.rbf` for PR programming with the **Programming File Generator**:

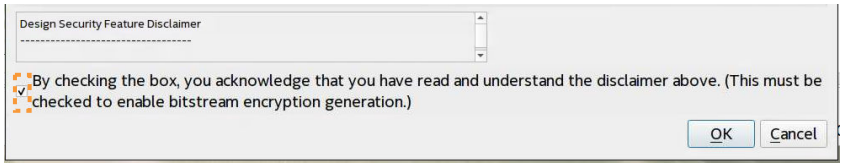
1. Click **File ➤ Programming File Generator**. The **Programming File Generator** appears.
2. Specify the target **Device family** and the **Configuration mode** for partial reconfiguration.
3. On the **Output File** tab, specify the **Output directory**, file **name**, and enable the **Raw Binary File for Partial Reconfiguration (.rbf)** file type.
4. To add the input `.pmsf` file to convert, click the **Input Files** tab, click **Add Bitstream**, and specify the `.pmsf` that you generated in the Assembler.

Figure 15. Adding Bitstream File



5. On the **Input Files** tab, select the bitstream `.pmsf` file and click **Properties**. Specify any of the following options for the `.rbf`:
   - **Enable compression**—generates compressed PR bitstream files to reduce file size.
   - **Enable encryption**—generates encrypted independent bitstreams for base image and PR image. You can encrypt the PR image even if your base image has no encryption. The PR image can have a separate encryption key file (`.ekp`). You can also specify other **Security settings**.
   - If you turn on **Enable encryption**, you must also acknowledge the **Design Security Feature Disclaimer** by checking the box.

Figure 16. Design Security Feature Disclaimer



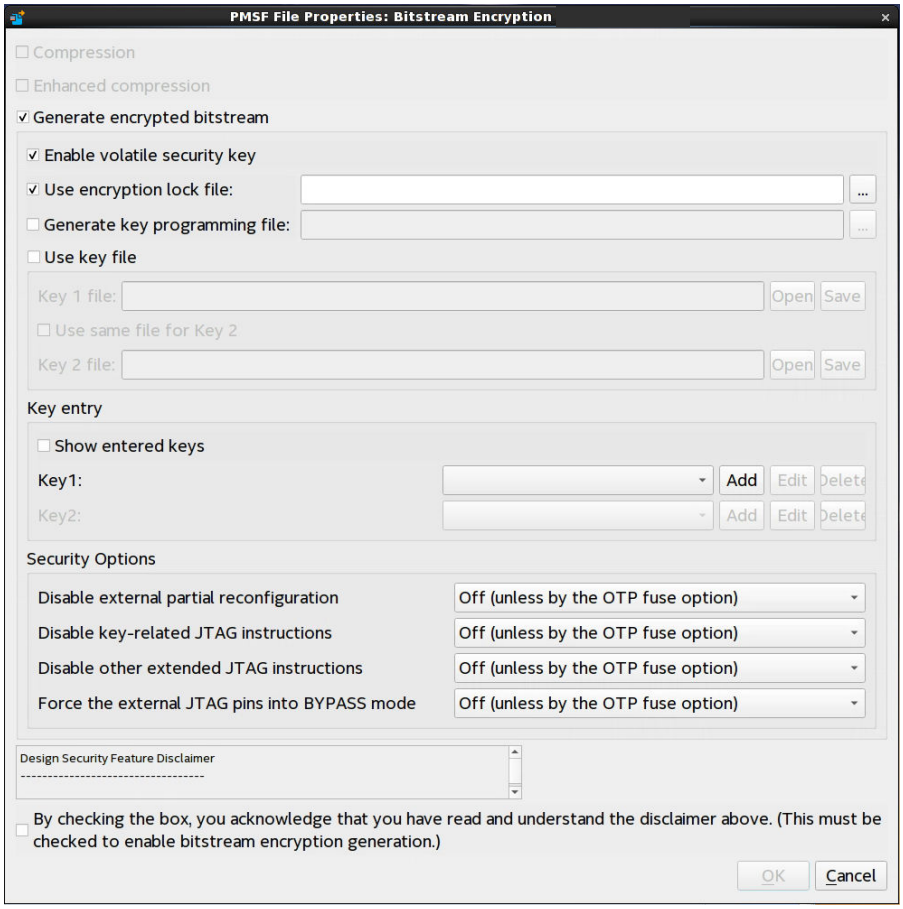
6. Click **OK**.
7. In **Programming File Generator**, click **Generate**. The PR bitstream files generate according to your specifications.

Generating PR Bitstreams with Convert Programming Files Dialog Box

Follow these steps to generate the `.rbf` with the **Convert Programming Files** dialog box:

1. Click **File ➤ Convert Programming Files**. The **Convert Programming Files** dialog box appears.
2. Specify the output file name and **Programming file type** as **Raw Binary File for Partial Reconfiguration (.rbf)**.
3. To add the input `.pmsf` file to convert, click **Add File**.
4. Select the newly added `.pmsf` file, and click **Properties**.
5. Enable or disable any of the following options and click **OK**:
   - **Compression**—enables compression on PR bitstream.
   - **Enhanced compression**—enables enhanced compression on PR bitstream.
   - **Generate encrypted bitstream**—generates encrypted independent bitstreams for base image and PR image. You can encrypt the PR image even if your base image has no encryption. The PR image can have a separate encryption key file (`.ekp`). If you enable **Generate encrypted bitstream**, enable or disable the **Enable volatile security key**, **Use encryption lock file**, and **Generate key programming file** options.
6. Click **Generate**. The PR bitstream files generate according to your specifications.

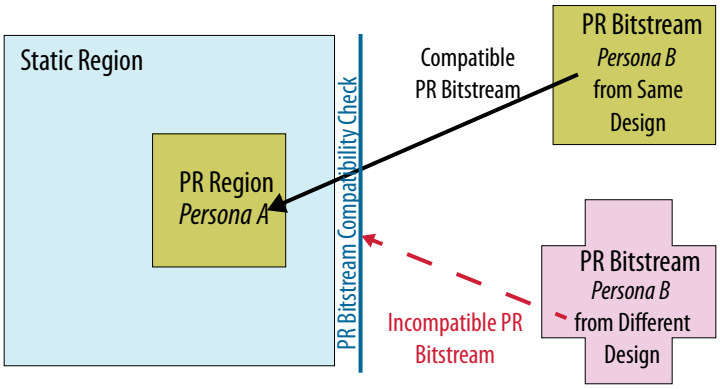
Figure 17. PMSF File Properties Bitstream Encryption



1.5.2. Partial Reconfiguration Bitstream Compatibility Checking

Partial reconfiguration bitstream compatibility checking verifies the compatibility of the reconfiguration bitstream to prevent configuration with an incompatible PR bitstream. The following sections describe PR bitstream compatibility check support.

Figure 18. PR Bitstream Compatibility Checking



Intel Stratix 10 and Intel Agilex PR Bitstream Compatibility Checking

For Intel Stratix 10 and Intel Agilex designs, PR bitstream compatibility checking is automatically enabled in the Compiler and in the Secure Device Manager (SDM) firmware by default. The following limitations apply to PR designs *if PR bitstream compatibility checking is enabled*:

- The firmware allows up to a total of 32 PR regions, irrespective of the number of hierarchical partial reconfiguration layers.
- Your PR design can have up to six hierarchical partial reconfiguration layers.
- Your PR design, when there is no hierarchy, can have up to 32 regions.
- Your PR design can have up to 15 child PR regions of any parent PR region (if it is hierarchical). Child PR regions count towards the total limit of 32 PR regions.

The Compiler generates an error if your PR design exceeds these limits when PR bitstream compatibility checking is enabled.

If you require more PR regions than this limitation allows, or otherwise want to disable PR bitstream compatibility checking , you can add the following assignment to the `.qsf` file:

```
set_global_assignment -name ENABLE_PR_POF_ID OFF
```

When you set this assignment to off, the limit of 32 total regions does not apply in the Compiler.

Note: If you require the PR bitstream authentication feature for your design, you must enable PR bitstream compatibility checking by setting the global assignment `ENABLE_PR_POF ID` to `ON`. The default setting is `ON`.

Intel Arria 10 and Intel Cyclone 10 GX PR Bitstream Compatibility Checking

For Intel Arria 10 and Intel Cyclone 10 GX designs, you enable or disable PR bitstream compatibility checking by turning on the **Enable bitstream compatibility check** option when instantiating the Partial Reconfiguration Controller Intel Arria 10/Cyclone 10 FPGA IP from the IP Catalog.

The PR IP verifies the partial reconfiguration PR Bitstream file (`.rbf`). When you enable the bitstream compatibility check, the PR `.pof` ID is encoded as the 71st word of the PR bitstream. If the PR IP detects an incompatible bitstream, then the PR IP stops the PR operation, and the status output reports an error.

When you turn on **Enable bitstream compatibility check**, the PR Controller IP core creates a **PR bitstream ID** and displays the bitstream ID in the configuration dialog box. For bitstream compatibility checking with hierarchical PR designs, refer to additional steps in *AN 806: Hierarchical Partial Reconfiguration Tutorial for Intel Arria 10 GX FPGA Development Board*.

Related Information

AN 806: Hierarchical Partial Reconfiguration Tutorial for Intel Arria 10 GX FPGA Development Board

1.5.3. Raw Binary Programming File Byte Sequence Transmission Examples

The raw binary programming file (`.rbf`) file contains the device configuration data in little-endian raw binary format. The following example shows transmitting the `.rbf` byte sequence 02 1B EE 01 in x32 mode:

Table 10. Writing to the PR control block or SDM in x32 mode

In x32 mode, the first byte in the file is the least significant byte of the configuration double word, and the fourth byte is the most significant byte.

Double Word = 01EE1B02			
LSB: BYTE0 = 02	BYTE1 = 1B	BYTE2 = EE	MSB: BYTE3 = 01
D[7..0]	D[15..8]	D[23..16]	D[31..24]
0000 0010	0001 1011	1110 1110	0000 0001

1.5.4. Generating a Merged .pmsf File from Multiple .pmsf Files

Use a single merged `.rbf` file to reconfigure two PR regions simultaneously. To merge two or more `.pmsf` files:

1. Open the **Convert Programming Files** dialog box.
2. Specify the output file name and programming file type as **Merged Partial-Mask SRAM Object File (.pmsf)**.
3. In the **Input files to convert** dialog box, select **PMSF Data**.
4. To add input files, click **Add File**. You must specify two or more files for merging.
5. To generate the merged file, click **Generate**.

Alternatively, to merge two or more `.pmsf` files from the Intel Quartus Prime shell, type the following command:

```
quartus_cpf --merge_pmsf=<number of merged files> <pmsf_input_file_1> \
     <pmsf_input_file_2> <pmsf_input_file_etc> <pmsf_output_file>
```

For example, to merge two `.pmsf` files, type the following command:

```
quartus_cpf --merge_pmsf=<2> <pmsf_input_file_1> <pmsf_input_file_2> \
     <pmsf_output_file>
```

1.6. Generating Programming Files for Intel FPGA Devices with Hard Processor Systems

When generating programming files for Intel FPGA devices with a Hard Processor System (HPS), you must first determine what boot flow you want to use for the device: HPS Boot First or FPGA Configuration First.

Depending on the boot flow that you want to use, follow the instructions in one of the following sections:

- Generating Programming Files for HPS Boot First Boot Flows on page 31
- Generating Programming Files for FPGA Configuration First Boot Flows on page 33

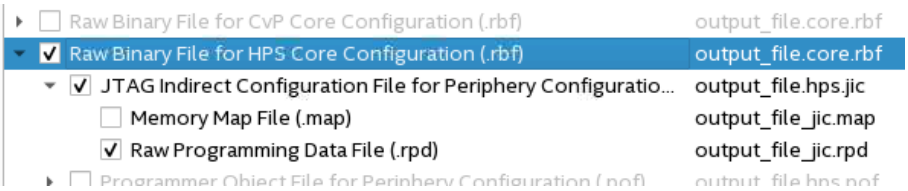
1.6.1. Generating Programming Files for HPS Boot First Boot Flows

In HPS Boot First boot flows, the HPS I/O and EMIF are configured and the HPS is booted before configuring the FPGA I/O and core.

Configuring the HPS I/O for the first time and then loading the HPS FSBL is called "Phase 1 configuration". The subsequent configuration of FPGA I/O and core by HPS is called "Phase 2 configuration".

To generate programming files for HPS Boot First boot flows:

1. Generate the primary programming files for your design, as Generating Primary Device Programming Files on page 5 describes.
2. Click **File ➤ Programming File Generator**.
3. For **Device family**, select your target device. The options available in the **Programming File Generator** change dynamically, according to your device and configuration mode selection.
4. For **Configuration mode**, select an Active Serial mode that your device supports. Configuration Modes (Programming File Generator) on page 10 describes all modes.
5. On the **Output Files** tab, select **Raw Binary File for HPS Core Configuration (.rbf)**, then select the following files:
   - **JTAG Indirect Configuration File for Periphery Configuration (.jic)**
   - **Raw Programming Data File (.rpd)**

   Secondary Programming Files (Programming File Generator) on page 10 describes all output files.
6. On the **Output Files** tab, select **Raw Programming Data File (.rpd)** and click **Edit**.
7. Optional: **Optional:** In the **RPD Properties** dialog box, set **Bit swap** to **On**.

   Important: This step may or may not be required, depending on the external programmer that you use.
8. Specify the **Output directory** and **Name** for the file you generate. Output Files Tab Settings (Programming File Generator) on page 58 describes all options.
9. On the **Input Files** tab, click **Add Bitstream** to add your `.sof` file or files.
10. For each `.sof` file that you add, edit their properties as follows:
    a. Select the `.sof` file and click **Properties**.
    b. In the **Bootloader** field of **Input File Properties** dialog box, add the U-Boot First State Boot Loader (FSBL) file. Ensure that the file is an Intel-format hexadecimal (`.hex`) file.
11. To add other data, such as U-Boot Second Stage Boot Loader (SSBL) file or Phase 2 bitstreams:

a. Click **Add Raw Data** and specify an Intel-format hexadecimal (`.hex`) file.

b. Select the file you added and click **Properties**.

c. In the **Input File Properties** dialog box, set the **Bit swap** field to **On**.

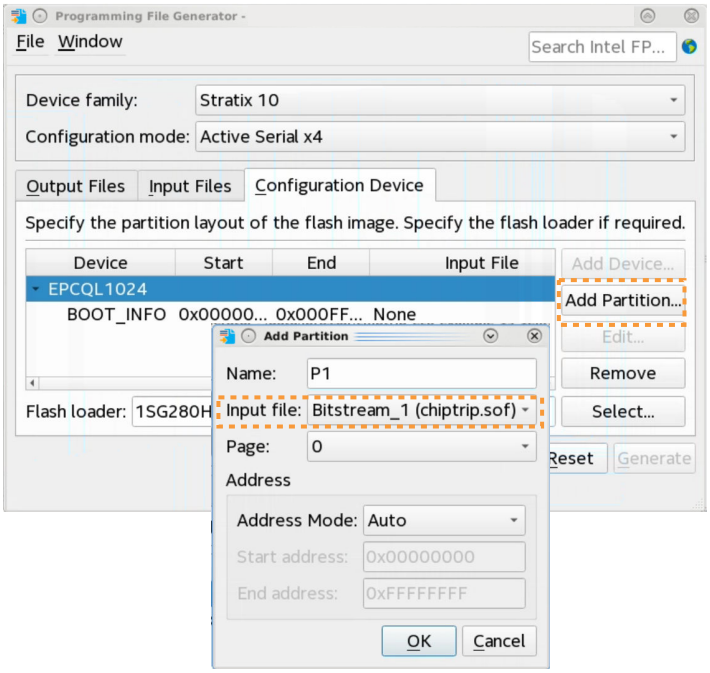
Important: Your Phase 1 and Phase 2 bitstreams are subject to the following restrictions:

- Your Phase 1 and Phase 2 bitstreams must be generated by the same version of Intel Quartus Prime, including any applied patches or updates.
- If your Phase 1 and Phase 2 bitstreams are generated from different Intel Quartus Prime Pro Edition projects, review the following Knowledge Base article for additional steps that might be required:

  Why does FPGA configuration fail from Linux / u-boot fail on Intel Stratix 10 10 SX devices when I use phase 2 bitstreams generated from different Quartus Projects?

12. To specify the `.sof` file that occupies the flash memory partition, click **Add Partition** on the **Configuration Device** tab. Add Partition Dialog Box (Programming File Generator) on page 60 describes all options.

Figure 19. Add Flash Partition



13. To select a supported flash memory device and predefined programming flow, click **Add Device** on the **Configuration Device** tab. Alternatively, click **<<new device>>** to define a new flash memory device and programming flow. Configuration Device Tab Settings on page 59 describes all settings.

14. Click the **Select** button for **Flash Loader** and select the device that controls loading of the flash memory device. Select Devices (Flash Loader) Dialog Box on page 63 describes all settings.
15. After you specify all options in **Programming File Generator**, the **Generate** button enables. Click **Generate** to create the files.
16. **Optional:** Export your settings to PFG setting file (`.pfg`) so that you can use these settings again with the `quartus_pfg` command line tool.

    For details, refer to quartus_pfg Command Line Tool on page 36.

Related Information

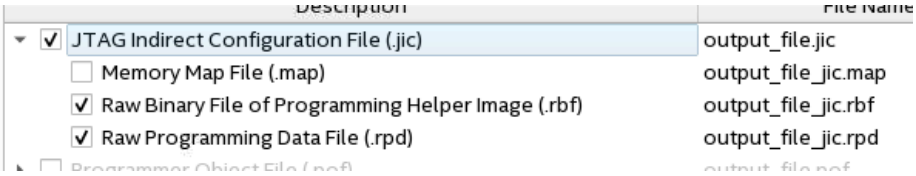
- Intel Stratix 10 Configuration User Guide
- Intel Stratix 10 SoC FPGA Boot User Guide
- Intel Stratix 10 Hard Processor System Remote System Update User Guide
- Intel Arria 10 SoC FPGA Boot User Guide

1.6.2. Generating Programming Files for FPGA Configuration First Boot Flows

In FPGA Configuration First boot flows, the FPGA core and periphery are configured first. After that, the HPS can optionally be booted.

To generate programming files for FPGA Configuration First boot flows

1. Generate the primary programming files for your design, as Generating Primary Device Programming Files on page 5 describes.
2. Click **File ➤ Programming File Generator**.
3. For **Device family**, select your target device. The options available in the **Programming File Generator** change dynamically, according to your device and configuration mode selection.
4. For **Configuration mode**, select an Active Serial mode that your device supports. Configuration Modes (Programming File Generator) on page 10 describes all modes.
5. On the **Output Files** tab, select **JTAG Indirect Configuration File (.jic)**, then select the following files:
   - **Raw Binary File of Programming Helper Image (.rbf)**
   - **Raw Programming Data File (.rpd)**



   Secondary Programming Files (Programming File Generator) on page 10 describes all output files.
6. On the **Output Files** tab, select **Raw Programming Data File (.rpd)** and click **Edit**.
7. Optional: **Optional:** In the **RPD Properties** dialog box, set **Bit swap** to **On**.

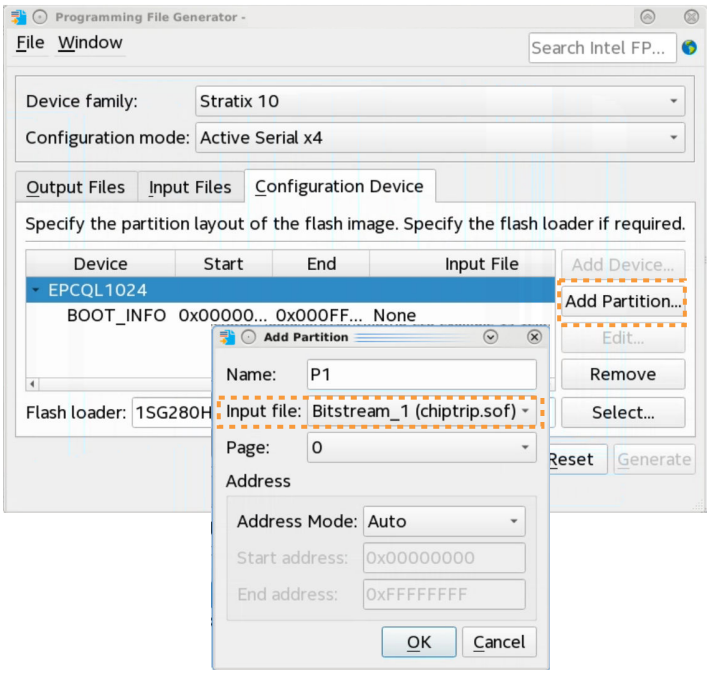
Important: This step may or may not be required, depending on the external programmer that you use.

8. Specify the **Output directory** and **Name** for the file you generate. Output Files Tab Settings (Programming File Generator) on page 58 describes all options.
9. On the **Input Files** tab, click **Add Bitstream** to add your `.sof` file and then edit its properties:
   a. Select the `.sof` file and click **Properties**.
   b. In the **Bootloader** field of **Input File Properties** dialog box, add the U-Boot First State Boot Loader (FSBL) hex file (`.hex`).
10. To add other data, such as a U-Boot Second Stage Boot Loader (SSBL) file or Phase 2 bitstreams:
    a. Click **Add Raw Data** and specify an Intel-format hexadecimal (`.hex`) file.
    b. Select the file you added and click **Properties**.
    c. In the **Input File Properties** dialog box, set the **Bit swap** field to **On**.

    Important: Your Phase 1 and Phase 2 bitstreams are subject to the following restrictions:
    - Your Phase 1 and Phase 2 bitstreams must be generated by the same version of Intel Quartus Prime, including any applied patches or updates.
    - If your Phase 1 and Phase 2 bitstreams are generated from different Intel Quartus Prime Pro Edition projects, review the following Knowledge Base article for additional steps that might be required:

      Why does FPGA configuration fail from Linux / u-boot fail on Intel Stratix 10 SX devices when I use phase 2 bitstreams generated from different Quartus Projects?
11. To specify the `.sof` file that occupies the flash memory partition, click **Add Partition** on the **Configuration Device** tab. Add Partition Dialog Box (Programming File Generator) on page 60 describes all options.

Figure 20. Add Flash Partition



12. To select a supported flash memory device and predefined programming flow, click **Add Device** on the **Configuration Device** tab. Alternatively, click **<<new device>>** to define a new flash memory device and programming flow. Configuration Device Tab Settings on page 59 describes all settings.
13. Click the **Select** button for **Flash Loader** and select the device that controls loading of the flash memory device. Select Devices (Flash Loader) Dialog Box on page 63 describes all settings.
14. After you specify all options in **Programming File Generator**, the **Generate** button enables. Click **Generate** to create the files.
15. Optional: **Optional:** Export your settings to a PFG setting file (`.pfg`) so that you can use these settings again with the `quartus_pfg` command line tool.

    For details, refer to quartus_pfg Command Line Tool on page 36.

Related Information

- Intel Stratix 10 Configuration User Guide
- Intel Stratix 10 SoC FPGA Boot User Guide
- Intel Stratix 10 Hard Processor System Remote System Update User Guide
- Intel Arria 10 SoC FPGA Boot User Guide

1.7. Scripting Support

The Intel Quartus Prime software allows generating programming files from the command line. You can incorporate these commands to scripted flows.

1.7.1. quartus_pfg Command Line Tool

The Programing File Generator is also available as the `quartus_pfg` executable. You can specify conversion settings in the command line or through a PFG setting file (`.pfg`). This ability is useful for advanced designs that require multiple images or multiple user data files (HEX/RBF), because you define the settings once in the GUI and then export for subsequent use in the command line.

By default, `quartus_pfg` converts files using the AVSTx8 configuration scheme. To use a different flow, specify the proper operation mode, as the following command shows:

```
quartus_pfg -c  -o device=MT25QU512 -o mode=ASX4 -o flash_loader=1SG280HN3S3 \
      project.sof project.jic
```

To export PFG settings to a `.pfg` file, click **File ➤ Save**. The Programming File Generator only saves settings that are consistent.

For more information about the `quartus_pfg` executable, type the following in the command line:

```
quartus_pfg --help
```

Differences Between GUI and Command Line Tool

The command line tool supports single image conversion only.

1.7.2. quartus_cpf Command Line Tool

The Convert Programming Files tool is also available as the `quartus_cpf` command line executable. You can specify conversion settings in the command line or with a conversion setup file (`.cof`).

For help with the `quartus_cpf` executable, type the following at the command line:

```
quartus_cpf --help
```

1.7.2.1. Generating a Partial-Mask SRAM Object File using a Mask Settings File and a SRAM Object File

- To generate a `.pmsf` file with the `quartus_cpf` executable, type the following in the command line:

  ```
  quartus_cpf -p <pr_revision.msf> <pr_revision.sof> <new_filename.pmsf>
  ```

Note: The `-p` option is available for designs targeting Intel Arria 10 and Intel Cyclone 10 GX device families.

Related Information

Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration
In *Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration*

1.8. Generating Programming Files Revision History

Document Version	Intel Quartus Prime Version	Changes
2020.12.14	20.4	• Added new "Generating Programming Files for the HPS Flash Programmer" section.
2020.10.13	20.3	• Added more details about alternate configuration schemes to "quartus_pfg Command Line Tool" topic.
2020.05.08	19.4	• Added note about programming file differences to "Generating Primary Device Programming Files" topic.
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 Security for Intel Stratix 10 Devices" topic. • 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. • 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.
2019.06.10	19.1	• Added links to *Generic Flash Programmer User Guide*. • Added flash programming details to "Generating Secondary Programming Files" and created separate topics for **Programming File Generator** and **Convert Programming Files** dialog box. • Added new "Enabling Bitstream Encryption or Co-Signing (Programming File Generator)" topic. • Added new "Enabling Bitstream Compression or Encryption (Convert Programming File)" topic. • Updated screenshots for latest GUI.
2019.04.01	19.1	• Retitled and reorganized topics to improve flow of information. • Added "Programming File Generator Configuration Modes" topic. • Added "Convert Programming File Configuration Modes" topic. • Added "Generating Programming Files for Partial Reconfiguration." • Added "Generating PR Bitstreams Files." • Added "Partial Reconfiguration Bitstream Compatibility Checking." • Added "Raw Binary Programming File Byte Sequence Transmission Examples." • Added "Generating a Merged .pmsf File from Multiple .pmsf Files."
2018.10.09	18.1	• Added MAX V to the list of devices that the Programming File Generator tool supports. • Added table : *Device Families that the Convert Programming Files Tool Supports*.

continued...

Document Version	Intel Quartus Prime Version	Changes
2018.09.24	18.1	• Added topic: *quartus_cpf Command Line Tool*. • Stated that the Convert Programming Files dialog box is a legacy tool that supports file conversion for older device families. • In topic: *Output File Types*, specified that the list includes file types generated by the Converting Programming Files tool.
2018.08.07	18.0	Reverted document title to *Programmer User Guide: Intel Quartus Prime Pro Edition*.
2018.06.27	18.0	• Created the new chapter with information from the *Programming Devices* chapter. • Included information about the Programming File Generator tool.

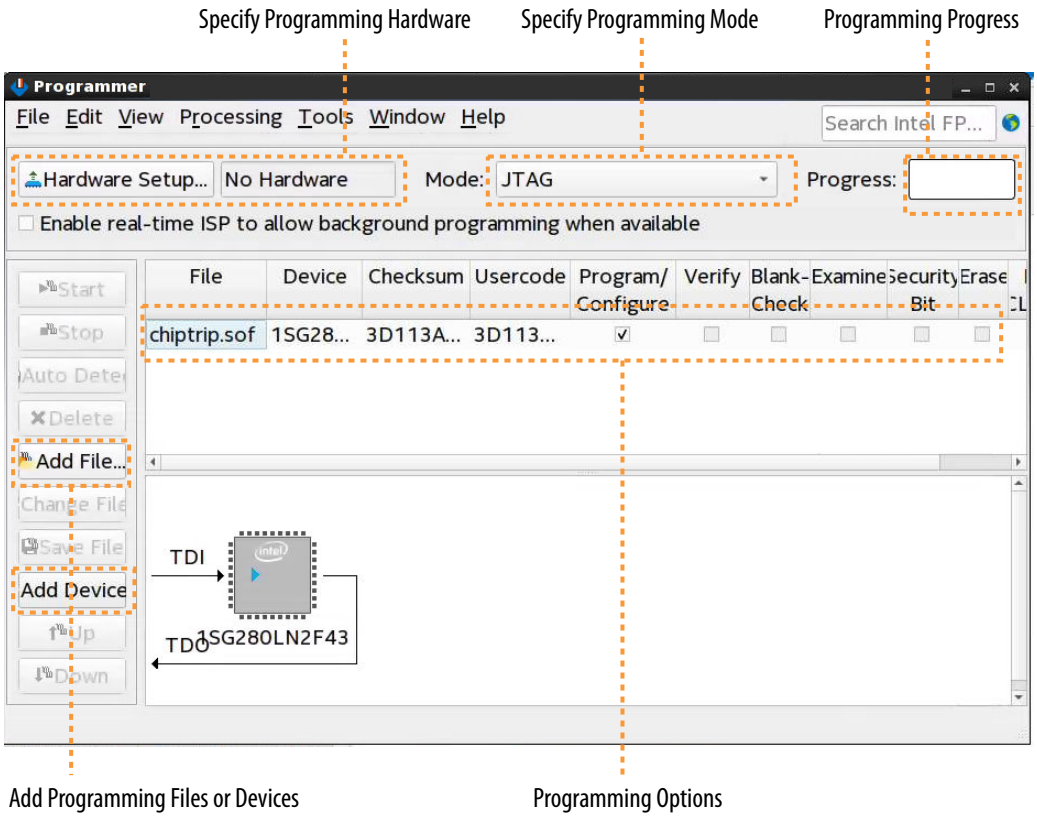
2. Using the Intel Quartus Prime Programmer

Use the Intel Quartus Prime Programmer and connected communication cable to program or configure Intel CPLD, FPGA, and configuration devices. This chapter describes how to setup and use the Intel Quartus Prime Programmer.

2.1. Intel Quartus Prime Programmer

Access the integrated Programmer by clicking **Tools ➤ Programmer** in the Intel Quartus Prime software.

Figure 21. Intel Quartus Prime Programmer



Prior to programming or configuration, you generate and specify the primary programming files, setup the programming hardware, and set the configuration mode in the Programmer.

2.2. Programming and Configuration Modes

The current version of the Intel Quartus Prime Programmer supports the following programming and configuration modes in the Programmer's **Mode** list. Select a configuration mode to setup and run that type of programming or configuration.

Table 11. Programming and Configuration Modes

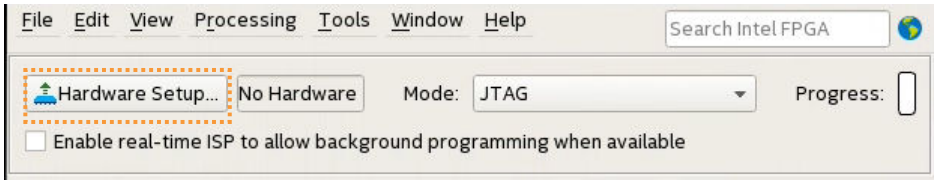
Programming or Configuration Mode	Description
JTAG	A configuration method that configures one or more devices through the Joint Test Action Group (JTAG) Boundary-Scan Test (BST) circuitry.
In-Socket Programming	Configuration device programming or testing via the Altera Programming Unit (APU).
Passive Serial	An external controller passes configuration data to one or more configuration devices via a serial data stream. The device is treated as a slave device with a 5-wire interface to the external controller. The external controller can be an intelligent host such as a microcontroller or CPU, or the Intel Quartus Prime Programmer. The external controller can also be a serial configuration device.
Active Serial Programming	The active serial memory interface block loads design data into one or more devices. The active serial memory interface block controls the configuration process, and configures all of the devices in the chain using the configuration data stored in an EPCS1, EPCS4, EPCS16, EPCS64, EPCQ, EPCQL, and third-party QSPI serial configuration devices.

2.3. Basic Device Configuration Steps

Basic FPGA Device Configuration over JTAG involves opening the Intel Quartus Prime Programmer, connecting to a device on a development kit or board, and loading the configuration SRAM Object File (`.sof`) into the SRAM of the FPGA. The following steps describe the basic JTAG device configuration flow:

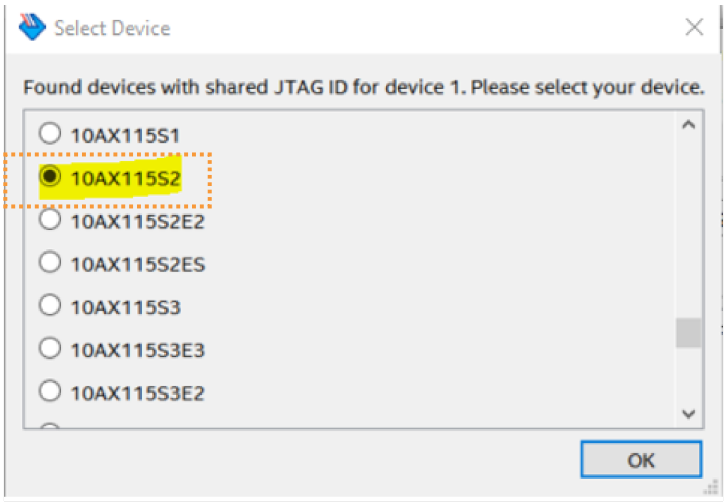
1. To run the Assembler to generate primary programming files, click **Processing ➤ Start ➤ Start Assembler**. The Compiler runs any prerequisite stages and generates programming files according to your specifications, as Generating Primary Device Programming Files on page 5 describes.
2. To open the Programmer, click **Tools ➤ Programmer**.
3. Connect the board cables. For JTAG device configuration, connect the JTAG USB cable to the board, and connect the power cable attached to the board to a power source.
4. Turn on power to the board.
5. In the Programmer, select **JTAG** for the programming **Mode**, as Programming and Configuration Modes on page 40 describes.
6. Click **Hardware Setup**. In the **Hardware** list, select connected programming hardware, as Specifying the Programming Hardware Setup on page 42 describes.

Figure 22. Hardware Setup



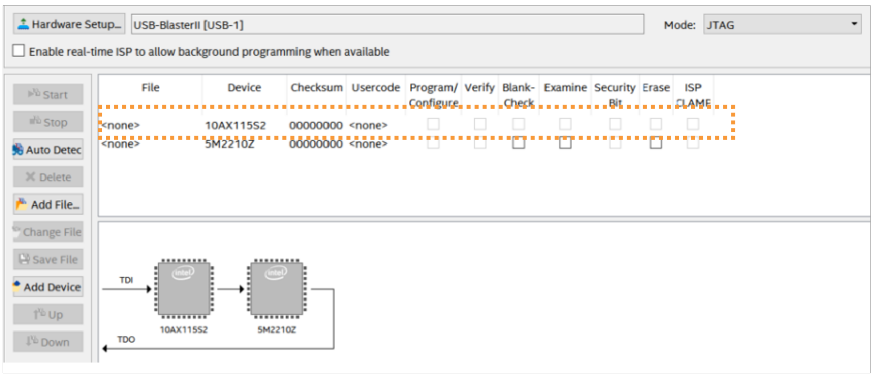
7. In the **Found Devices** list, select the device that matches your design and click **OK**.

Figure 23. Select Device



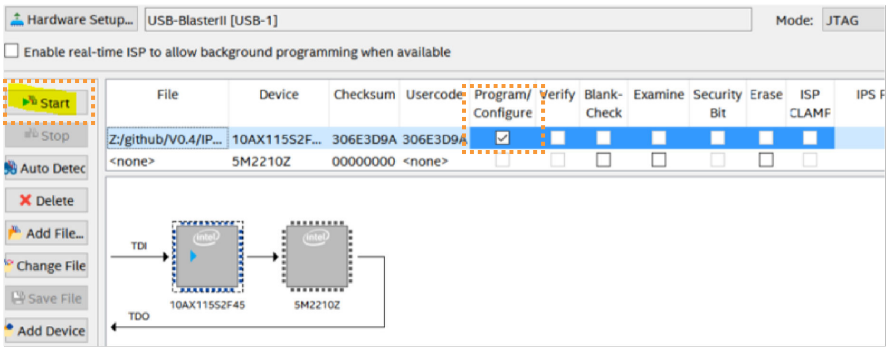
8. Right-click the row in the file list, and then click **Change File**.

Figure 24. Programmer Window



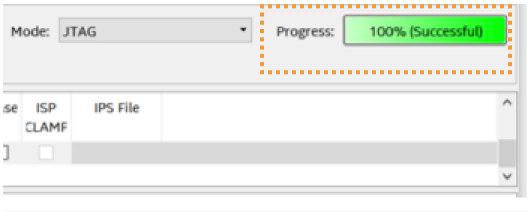
9. Browse to select the `.sof` file.
10. Enable the **Program/Configure** option for the row.

Figure 25. Program/Configure Option



11. Click **Start**. The progress bar reaches 100% when device configuration is complete. The device is now fully configured and in operation.

Figure 26. Programming Successful



Note: If device configuration fails, confirm that the device you select for configuration matches the device you specify during `.sof` file generation.

2.4. Specifying the Programming Hardware Setup

Before you can program or configure a device, you must specify an appropriate hardware setup. The Programmer's **Hardware Setup** dialog box allows you to add and remove programming hardware or JTAG servers from the current programming setup. You can specify a hardware setup for device programming or configuration, or configure a local JTAG server.

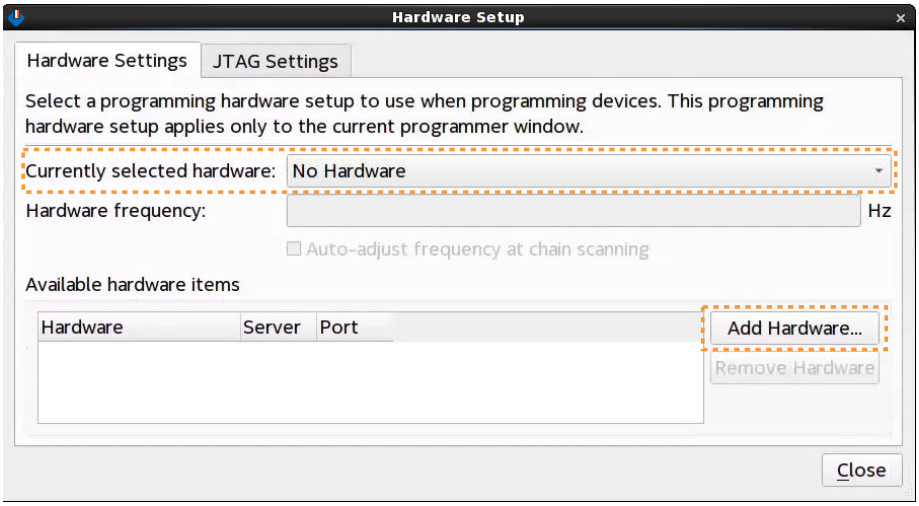
A JTAG server allows the Intel Quartus Prime Programmer to access the JTAG programming hardware connected to a remote computer through the JTAG server of that computer. The JTAG server allows you to control the programming or configuration of devices from a single computer through other computers at remote locations. The JTAG server uses the TCP/IP communications protocol.

Selecting Device Programming Hardware

Follow these steps to select device programming hardware in the Programmer:

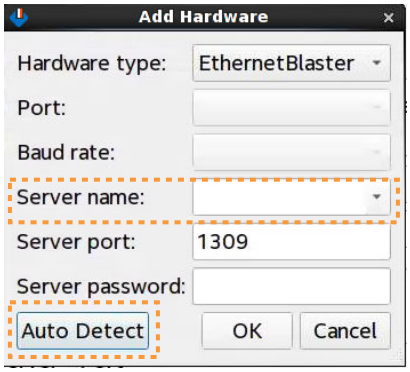
1. In the Programmer, click **Hardware Setup**.

Figure 27. Hardware Setup Dialog Box



2. To add new programming hardware, click **Add Hardware** on the **Hardware Settings** tab. In the **Add Hardware** dialog box, click **Auto Detect** to detect your programming hardware, or specify the properties of your programming hardware.

Figure 28. Add New Hardware



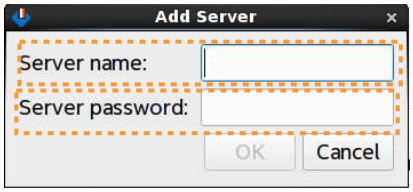
3. On the **Hardware Settings** tab, select your connected programming hardware in **Currently selected hardware**. This list is empty until you connect and add programming hardware to your system.
4. Enable or disable **Auto-adjust frequency at chain scanning** to automatically adjust the **Hardware frequency** according to the frequency at chain scanning.
5. Click **Close**. The setup appears as the current hardware setup.

Selecting a JTAG Server for Device Programming

Follow these steps to select a JTAG server for device programming in the Programmer:

1. In the Programmer, click **Hardware Setup**.
2. On the **JTAG Settings** tab, click **Add Server**. In the **JTAG Settings** dialog box, specify the **Server name** and **Server password**.

Figure 29. JTAG Settings



3. Under **JTAG Servers**, select the JTAG server that you want to access for programming.
4. Click **Close**. The setup appears as the current hardware setup.

2.4.1. JTAG Chain Debugger Tool

The JTAG Chain Debugger tool allows you to test the JTAG chain integrity and detect intermittent failures of the JTAG chain. You access the tool by clicking **Tools ➤ JTAG Chain Debugger** on the Intel Quartus Prime software.

In addition, the tool allows you to shift in JTAG instructions and data through the JTAG interface, and step through the test access port (TAP) controller state machine for debugging purposes.

2.4.2. Editing the Details of an Unknown Device

When the Intel Quartus Prime Programmer automatically detects devices with shared JTAG IDs, the Programmer prompts you to specify the device in the JTAG chain. If the Programmer does not prompt you to specify the device, you must manually add each device in the JTAG chain to the Programmer, and define the instruction register length of each device.

To edit the details of an unknown device, follow these steps:

1. Double-click the unknown device listed under the device column.
2. Click **Edit**.
3. Change the device **Name**.
4. Specify the **Instruction register length**.
5. Click **OK**.
6. Save the `.cdf` file.

2.4.3. Running JTAG Daemon with Linux

The JTAGD daemon is the Linux version of a JTAG server. The JTAGD daemon allows a remote machine to program or debug boards connected to a Linux host over the network. The JTAGD daemon also allows programs to share JTAG resources.

Running the JTAGD daemon prevents:

- The JTAGD server from exiting after two minutes of idleness.
- The JTAGD server from not accepting connections from remote machines, which might lead to an intermittent failure.

To run JTAGD as a daemon:

1. Create an `/etc/jtagd` directory.
2. Set the permissions of this directory and the files in the directory to allow read/write access.
3. Execute `jtagd` (with no arguments) from the `quartus/bin` directory.

The JTAGD daemon is now running and does not terminate when you log off.

2.5. Programming with Flash Loaders

Parallel and serial configuration devices do not support the JTAG programming interface. However, you can use a flash loader to program configuration devices in-system via the JTAG interface. The flash loader allows an FPGA to function as a bridge between the JTAG interface and the configuration device. The Intel Quartus Prime software supports various parallel and serial flash loaders for programming bitstream storage and configuration via flash memory devices.

Refer to the following documents for step-by-step flash programming instructions.

Related Information

- Generic Serial Flash Interface Intel FPGA IP Core User Guide
- Intel Parallel Flash Loader IP Core User Guide
- Generic Flash Programmer User Guide
- Customizable Flash Programmer User Guide

2.5.1. Specifying Flash Partitions

Flash partitions allow you to store bitstreams or raw data.

Note: The **Programming File Generator** supports defining flash partitions only for `.jic` or `.pof` programming files.

To create flash partitions in the **Configuration Devices** tab:

1. Select the device and click **Add Partition**.
2. In the **Add Partition** dialog box, define the following parameters, and then click **OK**:

Table 12. 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:

continued...

Setting	Description
	• **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**.

The partition associated to the device appears in the device list.

3. If you want to change the parameters of a partition, click the partition and then click **Edit**.
4. If you want to remove a partition, click the partition and then click **Remove**.
5. After specifying the settings for all flash partitions, click **Generate.**

2.5.2. 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.

2.6. Verifying the Programming File Source with Project Hash

Intel Quartus Prime programming files support the project hash property, which identifies the source project from which programming files generate.

During compilation, the Intel Quartus Prime software generates a unique project hash, and embeds this hash value in the programming files (`.sof`). You can verify the source of programming files by matching the project and programming file hash values.

The project hash does not change for different builds of the Intel Quartus Prime software, or when you install a software update. However, if you upgrade any IP with a different build or patch, the project hash changes.

2.6.1. Obtaining Project Hash for Intel Arria 10 Devices

To obtain the project hash value of a `.sof` programming file for a design, use the `quartus_asm` command-line executable (`quartus_asm.exe` in Windows) with the `--project_hash` option.

```
quartus_asm --project_hash <sof-file>
```

Example 1. Output of Project Hash Command:

In this example, the programming file is `worm.sof`.

```
Info: *******************************************************************
Info: Running Quartus Prime Assembler
Info: Version 17.0.0 Build 288 04/12/2017 SJ Pro Edition
Info: Copyright (C) 2017  Intel Corporation. All rights reserved.
Info: Your use of Intel Corporation's design tools, logic functions
Info: and other software and tools, and its AMPP partner logic
Info: functions, and any output files from any of the foregoing
Info: (including device programming or simulation files), and any
Info: associated documentation or information are expressly subject
Info: to the terms and conditions of the Intel Program License
Info: Subscription Agreement, the Intel Quartus Prime License Agreement,
Info: the Intel MegaCore Function License Agreement, or other
Info: applicable license agreement, including, without limitation,
Info: that your use is for the sole purpose of programming logic
Info: devices manufactured by Intel and sold by Intel or its
Info: authorized distributors.  Please refer to the applicable
Info: agreement for further details.
Info: Processing started: Fri Apr 14 18:01:47 2017
Info: Command: quartus_asm -t project_hash.tcl worm.sof
Info: Quartus(args): worm.sof
0x1ffdc3f47c57bbe0075f6d4cb2cb9deb
Info (23030): Evaluation of Tcl script project_hash.tcl was successful
Info: Quartus Prime Assembler was successful. 0 errors, 0 warnings
Info: Peak virtual memory: 1451 megabytes
Info: Processing ended: Fri Apr 14 18:01:56 2017
Info: Elapsed time: 00:00:09
Info: Total CPU time (on all processors): 00:00:04
```

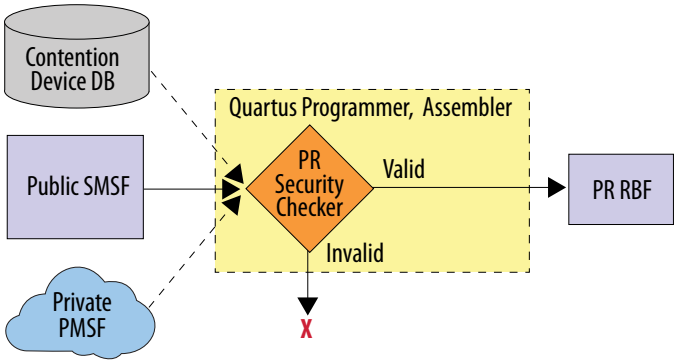
2.7. Using PR Bitstream Security Verification (Intel Stratix 10 Designs)

PR bitstream security verification requires a separate license and `.qsf` setting to enable. After you license and enable PR bitstream verification, the Compiler generates both a public Secure Mask Settings File (`.smsf`) and private Partially Masked Settings File (`.pmsf`) for each PR region during the base compilation.

The `.pmsf` contains comprehensive information that the Programmer requires to generate the PR bitstream for a Client region, including the actual bit settings, a region mask, and all the auxiliary bit masks. The `.smsf` contains a region ownership mask and comprehensive information to detect a peek or poke attack by the PR region's persona.

Thereafter, the Programmer requires both the private `.pmsf` and public `.smsf` to generate the PR bitstream for this PR region, ensuring that the PR persona can only change bits that the persona owns. The Platform Owner may or may not release `.smsf` files to third-party Clients as part of the PR region collateral. The Platform Owner uses the `.smsf` to generate the PR bitstream from Client's `.pmsf` for this PR region with the Programmer.

Figure 30. PR Bitstream Security Validation in Programmer



Follow these steps to license, enable, and use PR bitstream security verification:

1. Obtain the license file to enable generation of `.smsf` files for PR regions during base compilation, and to perform PR bitstream security verification during PR bitstream generation in the Programmer. To obtain the license, login or register for a My-Intel account, and then submit an Intel Premier Support case requesting the license key.
2. To add the license file to the Intel Quartus Prime Pro Edition software, click **Tools ➤ License Setup** and specify the feature **License File**.
3. To enable PR security validation features, add the following line to the project `.qsf`:

```
set_global_assignment -name PR_SECURITY_VALIDATION on
```

4. Compile the base revision.
5. Following base compilation, view the Assembler reports to view the generated `.smsf` files required for bitstream generation for each PR region.
6. The Platform Owner may release `.smsf` files to third-party clients as part of the PR region collateral. The Client provides the private `.pmsf` to the Platform Owner to verify PR security of the PR Persona configuration and generate validated PR bitstreams.
7. To validate PR security of the Client's `.pmsf`, the Platform Owner specifies the `.smsf` and corresponding `.pmsf` files at the Programmer command line to generate the validated PR bitstreams:

```
quartus_cpf -c –-smsf=<smsf_file> <pmsf_file> <output_file>
```

Related Information

- Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration
  In *Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration*
- My-Intel.com

2.8. Stand-Alone Programmer

The free Stand-Alone Programmer is available and has the same full functionality as the Intel Quartus Prime Programmer.

The Stand-Alone Programmer is useful when programming devices on a workstation that does not have an Intel Quartus Prime software license. The Stand-Alone Programmer does not require a separate Intel Quartus Prime software license. Download the Stand-Alone Programmer from the Download Center on the Intel website.

Related Information

Download Center for FPGAs

2.8.1. Stand-Alone Programmer Memory Consumption

The following operations increase memory usage in the Stand-Alone Programmer:

- Auto-detect
- Adding programming files to the flash memory
- Manually attaching the flash in the Programmer

In Windows systems, the Stand-Alone Programmer has the following memory limitations:

Table 13. Stand-Alone Programmer Memory Limitations

Application	Maximum Flash Device Size	Flash Device Operation Using PFL
64-bit Stand-Alone Programmer	Up to 2 Gb	Multiple Flash Device

2.9. Programmer Settings Reference

The following topics describe Intel Quartus Prime settings that impact programming and programming file generation.

2.9.1. Device & Pin Options Dialog Box

The following tables describe **Device & Pin Option** settings that impact generation of primary and secondary programming files. To access these settings, click **Assignments ➤ Device ➤ Device & Pin Options**.

Table 14. 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**.

Option	Description
Options *Note:* Not supported for Intel Agilex or 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. • **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.

continued...

Option	Description
	• **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: • **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.

Table 15. 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.

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.

continued...

Option	Description
	• **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**.
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 16. Unused Pin Options

Allow you specify the reserve state of all the unused pins on the device. To access, click **Assignments ➤ Device ➤ Device and Pin Options ➤ Unused Pins**. Disabled options are unavailable for the current device or configuration mode.

Option	Description
Reserve all unused pins	• **As input tri-stated**—the pins reserve as tri-state input pins. • **As output driving ground**—the pins reserve as output pins and drive the ground signal. • **As output driving an unspecified signal**—the pins reserve as output pins and drive any signal. • **As input tri-stated with bus-hold circuitry**—the pins reserve as tri-state input pins with bus-hold circuitry. • **As input tri-stated with weak pull-up**—the pins reserve as tri-state input pins with weak pull-up resistors.

Table 17. Dual-Purpose Pin Options

Allow you to specify whether the associated dual-purpose pin is reserved, and the reservation purpose. To access, click **Assignments ➤ Device ➤ Device and Pin Options ➤ Dual-Purpose Pins**. Disabled options are unavailable for the current device or configuration mode.

Option	Description
Dual-purpose pins	• **Use as regular I/O**—the dual-purpose pin is not reserved. Rather the I/O pin is in in user mode. • **Use as programming pin**—the nCE0 pin is reserved as a dedicated programming pin. • **As input tri-stated**—the dual-purpose pin is reserved as an input pin.

Option	Description
	• **As output driving ground**—the dual-purpose pin is reserved as an output pin and drives the ground signal. • **As output driving an unspecified signal**—the dual-purpose pin is reserved as an output pin and drives any signal. • **Compiler configured**—the Compiler automatically selects the best reserve setting for the dual-purpose pin, considering the current configuration scheme, and whether the pins are only used for configuration. If your design uses the Active Parallel configuration scheme and the Programmer does not communicate directly with the parallel flash device in user mode, you should reserve all dual-purpose pins connected to the parallel flash device as Compiler configured.

Table 18. Board Trace Model Options

For Intel Cyclone 10 GX designs only, allows you to specify the board trace, termination, and capacitive load parameters for each I/O standard. The board trace model parameters then apply to all output or bidirectional pins that you assign with the specified I/O standard. Board trace model parameters do not apply if you assign them to anything other than an output or bidirectional pin. You can create board trace model assignments for individual output or bidirectional pins in the Pin Planner. To access, click **Assignments ➤ Device ➤ Device and Pin Options ➤ Board Trace Model**. Disabled options are unavailable for the current device or configuration mode.

Option	Description
I/O standard	Specifies the supported I/O standard, such as **Differential 1.8-V SSTL Class II**.
Board trace model	Lists the board trace model parameters, with units, and values for the specified **I/O standard**. You can change the value of each parameter. The board trace model assignments apply to all output and bidirectional pins with the specified I/O standard assigned to them.

Table 19. I/O Timing Options

Allow you to specify the node at which output I/O timing terminates. To access, click **Assignments ➤ Device ➤ Device and Pin Options ➤ I/O Timing**. Disabled options are unavailable for the current device or configuration mode.

Option	Description
Default timing I/O endpoint	Specify **Near end** or **Far end**.

Table 20. Voltage Options

Allow you to specify the default I/O bank voltage for pins on the target device. Also displays the core voltage of the device or other internal voltage information. To access, click **Assignments ➤ Device ➤ Device and Pin Options ➤ Voltage**. Disabled options are unavailable for the current device or configuration mode.

Option	Description
Default I/O standard	Specify **1.2 V**, **1.5 V**, **1.8 V**, **2.5 V**, **3.0 LVTTL**, or **3.0 LVCMOS**.

Table 21. Error Detection CRC Options

Allow you to specify whether to use error detection cyclic redundancy check (CRC) and the value by which you want to divide the error check frequency for the currently selected device. To access, click **Assignments ➤ Device ➤ Device and Pin Options ➤ Error Detection CRC**. Disabled options are unavailable for the current device or configuration mode.

Option	Description
Enable Error Detection CRC_ERROR pin	Enables error detection CRC and CRC_ERROR pin usage for the targeted device. This check determines the validity of the programming data in the device. Any changes in the data while the device is in operation generates an error.

continued...

Option	Description
	Note: Not available for Intel Agilex or Intel Stratix 10 devices.
Enable Open Drain on CRC Error pin	Sets the CRC_ERROR pin as an open-drain pin. This action decouples the voltage level of the CRC_ERROR pin from VCCIO voltage. When you turn on this option, you must connect a pull-up resistor to the CRC_ERROR pin. *Note:* Not available for Intel Agilex or Intel Stratix 10 devices.
Enable error detection check	Enables error detection CRC checking to verify the validity of programming data in the device, and reports any changes in the data while the device is in operation.
Minimum SEU interval	Specifies the minimum time interval between two checks of the same bit. Setting to 0 means check as frequently as possible. Setting to a large value saves power. The unit of interval is millisecond. The maximum allowed number of intervals is 10000.
Enable internal scrubbing	Specifies use of internal scrubbing to correct any detected single error or double adjacent error within the core configuration memory while the device is still running.
Generate SEU sensitivity map file	Generates a Single Event Upset Sensitivity Map file. This file allows you to enable the Advanced SEU detection feature.
Allow SEU fault injection	Allows the injection of fault patterns to test for SEU.

Table 22. CvP Settings

Specifies the configuration mode for Configuration via Protocol (CvP). To access, click **Assignments ➤ Device ➤ Device and Pin Options ➤ CvP Settings**. Disabled options are unavailable for the current device or configuration mode.

Option	Description
Configuration via protocol	In **Initialization and update** mode, the periphery image stores in an external configuration device and loads the image into the FPGA through a conventional configuration scheme. The core image stores in a host memory and loads into the FPGA through the PCIe link. In **Core initialization** mode, the periphery image stores in an external configuration device and loads into the FPGA through the conventional configuration scheme. The core image is stores in a host memory and is loads into the FPGA through the PCIe link. In **Core update** mode, the FPGA device is initialized after initial system power up by loading the full configuration image from the external local configuration device to the FPGA. You can use the PCIe link to perform one or more FPGA core image update through this mode. In the **Off** mode, CvP is turned off.
Enable CvP_CONFDONE pin	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. *Note:* Not available for Intel Agilex or Intel Stratix 10 devices.
Enable open drain on CvP_CONFDONE pin	Enables the open drain on the CvP_CONFDONE pin. *Note:* Not available for Intel Agilex or Intel Stratix 10 devices.

Table 23. Partial Reconfiguration Options

Specifies generation of secondary programming files that partial reconfiguration requires. To access, click **Assignments ➤ Device ➤ Device and Pin Options ➤ Partial Reconfiguration**. Disabled options are unavailable for the current device or configuration mode.

Option	Description
Enable partial reconfiguration pins	Allows you to enable the PR_REQUEST, PR_READY, PR_ERROR, PR_DONE, DCLK, and DATA[31..0] pins. These pins are needed to support partial reconfiguration (PR) with an external host. An external host uses the PR_REQUEST pin to request partial reconfiguration, the PR_READY pin to determine if the device is ready to receive programming data, the PR_ERROR pin to externally monitor programming errors, and the PR_DONE pin to indicate the device finished programming. If this option is turned off, these pins are not available as PR pins when the device operates in user mode and the dual-purpose programming pins are available as user I/O pins.

continued...

Option	Description
	Note: Not available for Intel Agilex or Intel Stratix 10 devices.
Enable open drain on partial reconfiguration pins	Allows you to specify an open drain on the PR_READY, PR_ERROR, PR_DONE Partial Reconfiguration pins. *Note:* Not available for Intel Agilex or Intel Stratix 10 devices.
Generate Partial-Masked SOF files	Generates a Partial-Masked SRAM Object file (`.pmsf`) containing both configuration data and region definitions that can be used to re-configure a device region. If this option is turned on, the `.pmsf` generates instead of a Mask Settings file (`.msf`).
Generate Partial Reconfiguration RBF	Generates a Partial Reconfiguration Raw Binary File (`.rbf`) containing configuration data that an intelligent external controller can use to reconfigure the portion of target device.

Table 24. Power Management & VID Options

For Intel Stratix 10 and Intel Agilex devices only, specifies options for managing power, such as the bus speed mode and the slave address of the voltage regulator when in PMBus Master mode. To access, click **Assignments ➤ Device ➤ Device and Pin Options ➤ Power Management & VID Options**. Disabled options are unavailable for the current device or configuration mode.

Option	Description
Bus speed mode	Specifies the bus speed mode (for example, **100 KHz** or **400 KHz**) in PMBus Master mode.
Slave device type	Specifies the slave device type when the target FPGA device is in PMBus master mode. Available options are **ED8401**, **EM21XX**, **EM22XX**, **ISL82XX**, **LTM4677**, and **Other**.
Device address in PMBus Slave mode	Specifies the starting `00` device address when in PMBus Slave mode.
PMBus device 0 slave address through **PMBus device 7 slave address**	Specifies 7-bit hexadecimal value (without leading prefix 0x). For example, 7F for the slave address of a voltage regulator when in PMBus Master mode. You must specify a non-zero address.
Voltage output format	Specifies the **Auto discovery**, **Direct format**, or **Linear format** output voltage format when in PMBus Master mode
Direct format coefficient (m,b,R)	Specifies direct format coefficient m, b, or R when in PMBus Master mode. Signed integer between -32768 and 32767. Coefficient m is the slope coefficient. Coefficient b is the offset. Coefficient R is the exponent. Refer to the PMBus device manufacturer product documentation for these values. You must set this parameter when output voltage format of PMBus device is **Direct format** or **Auto discovery** format. You must specify a non-zero address when the output voltage format of PMBus device is in **Direct format**.
Linear format N	Specifies linear format N when in PMBus Master mode. Signed integer between -16 and 15. This is the exponent for the mantissa for the output voltage related command when VOUT format is set to **Linear format**. Refer to the PMBus device manufacturer product documentation for these values. You must specify a non-zero value for **Linear format**.
Translated voltage value unit	Specifies the **Volts** or **Millivolts** output voltage format when in PMBus Master mode.
Enable PAGE command	The FPGA PMBus master uses PAGE command to set all output channels on registered regulator modules to respond to `VOUT_COMMAND`.

Table 25. 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.

Table 26. 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.

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. 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.

continued...

Option	Values	Description
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.

(9)

Related Information

Enabling Bitstream Authentication (Programming File Generator) on page 19

2.9.2. More Security Options Dialog Box

Table 27. 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.	
Lock security eFuses	Causes eFuse failure if the eFuse CRC does not match the calculated value.	

continued...

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

Option	Description	Values
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.	

(10)

2.9.3. 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.

The following output file options are available:

Table 28. 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)**

2.9.4. 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.

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

The following input file settings are available:

Table 29. 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.

Related Information

Enabling Bitstream Encryption (Programming File Generator) on page 22

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

Table 30. 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.

(11)

2.9.6. Configuration Device Tab Settings

The **Configuration Device** tab allows you to specify the properties of an existing or new flash memory device that you want to program. Click **Add Device** to select a programming template for a predefined flash memory, or to click **<<new device>>** and then define a new flash memory device.

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

The following settings are available:

Table 31. 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. *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.

2.9.7. 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 32. 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:

continued...

Setting	Description
	• **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**.

2.9.8. 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 33. 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.

2.9.9. 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 34. 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.

2.9.10. 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 35. 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**.

2.9.11. 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 36. 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.

2.10. Scripting Support

In addition to the Intel Quartus Prime Programmer GUI, you can access programmer functionality from the command line and from scripts with the Intel Quartus Prime command-line executable `quartus_pgm.exe` (or `quartus_pgm` in Linux).

The following command programs a device:

```
quartus_pgm –c usbblasterII –m jtag –o bpv;design.pof ←
```

Where:

-c usbblasterII specifies the Intel FPGA Download Cable II

-m jtag specifies the JTAG programming mode

-o bpv represents the blank-check, program, and verify operations

design.pof represents the `.pof` containing the design logic

The Programmer automatically executes the erase operation before programming the device.

For Linux terminal, use:

```
quartus_pgm –c usbblasterII –m jtag –o bpv\;design.pof
```

Related Information

Intel Quartus Prime Scripting
In *Intel Quartus Prime Help*

2.10.1. The jtagconfig Debugging Tool

You can use the `jtagconfig` command-line utility to check the devices in a JTAG chain and the user-defined devices. The `jtagconfig` command-line utility is similar to the auto detect operation in the Intel Quartus Prime Programmer.

For more information about the `jtagconfig` utility, use the help available at the command prompt:

```
jtagconfig [–h | --help]
```

Note: The help switch does not reference the `-n` switch. The `jtagconfig -n` command shows each node for each JTAG device.

Related Information

Command Line Scripting
In *Intel Quartus Prime Pro Edition User Guide: Scripting*

2.11. Using the Intel Quartus Prime Programmer Revision History

Table 37. Document Revision History

Date	Intel Quartus Prime Version	Changes
2020.05.26	19.4.0	• Corrected descriptions of "Bus Speed Mode" and "Slave Device Type" power options.
2019.09.30	19.3.0	• 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.

continued...

Date	Intel Quartus Prime Version	Changes
2019.06.10	19.1.0	• Updated "Programming with Flash Loaders" topic to reflect new Generic Flash Programmer. • Removed references to obsolete 32-bit stand-alone Programmer. • Added "Erasing Flash Memory Sectors" topic describing complete erase of flash memory. • Added new "Programmer Settings Reference" section containing the following new GUI reference topics: — "Device & Pin Options Dialog Box" — "Input Files Tab Settings (Programming File Generator)" — "Output Files Tab Settings (Programming File Generator)" — "Configuration Device Tab Settings (Programming File Generator)" — "Add Partition Dialog Box (Programming File Generator)" — "Bitstream Compression, Authentication, and Encryption Settings (Programming File Generator)" — "Convert Programming Files Dialog Box" — "Bitstream Compression and Encryption Settings (Convert Programming File)" — "SOF Data Properties Dialog Box" — "Select Devices (Flash Loader) Dialog Box"
2019.04.01	19.1.0	• Added new "Using PR Bitstream Security Verification" topic. • Added new "Basic Device Configuration Steps" topic. • Added new "Programming and Configuration Modes" topic. • Retitled and reorganized topics to improve flow of information. • Added enhanced diagram of Programmer to "Intel Quartus Prime Programmer" topic. • Updated screenshots.
2018.10.09	18.1.0	• Created topic: *Stand-Alone Programmer Memory Limitations* from content in topic: *Stand-Alone Programmer*. • Removed outdated support information.
2018.08.07	18.0.0	Reverted document title to *Programmer User Guide: Intel Quartus Prime Pro Edition*.
2018.06.27	18.0.0	• Moved information about programming file generator to new chapter: *Generating Programming Files*.
2018.05.07	18.0.0	• First release as part of the stand-alone *Programmer User Guide*
2017.05.08	17.0.0	• Added Project Hash feature.
2016.10.31	16.1.0	• Implemented Intel rebranding.
2015.11.02	15.1.0	Changed instances of Quartus II to Intel Quartus Prime software.
2015.05.04	15.0.0	Added Conversion Setup File (.cof) description and example.
December 2014	14.1.0	Updated the Scripting Support section to include a Linux command to program a device.
June 2014	14.0.0	• Added Running JTAG Daemon. • Removed Cyclone III and Stratix III devices references. • Removed MegaWizard Plug-In Manager references. • Updated Secondary Programming Files section to add notes about the Quartus II Programmer support for **.rbf** files.
November 2013	13.1.0	• Converted to DITA format. • Added JTAG Debug Mode for Partial Reconfiguration and Configuring Partial Reconfiguration Bitstream in JTAG Debug Mode sections.

continued...

Date	Intel Quartus Prime Version	Changes
November 2012	12.1.0	• Updated Table 18–3 on page 18–6, and Table 18–4 on page 18–8. • Added "Converting Programming Files for Partial Reconfiguration" on page 18–10, "Generating .pmsf using a .msf and a .sof" on page 18–10, "Generating .rbf for Partial Reconfiguration Using a .pmsf" on page 18–12, "Enable Decompression during Partial Reconfiguration Option" on page 18–14 • Updated "Scripting Support" on page 18–15.
June 2012	12.0.0	• Updated Table 18–5 on page 18–8. • Updated "Quartus II Programmer GUI" on page 18–3.
November 2011	11.1.0	• Updated "Configuration Modes" on page 18–5. • Added "Optional Programming or Configuration Files" on page 18–6. • Updated Table 18–2 on page 18–5.
May 2011	11.0.0	• Added links to Quartus II Help. • Updated "Hardware Setup" on page 21–4 and "JTAG Chain Debugger Tool" on page 21–4.
December 2010	10.1.0	• Changed to new document template. • Updated "JTAG Chain Debugger Example" on page 20–4. • Added links to Quartus II Help. • Reorganized chapter.
July 2010	10.0.0	• Added links to Quartus II Help. • Deleted screen shots.
November 2009	9.1.0	No change to content.
March 2009	9.0.0	• Added a row to Table 21–4. • Changed references from "JTAG Chain Debug" to "JTAG Chain Debugger". • Updated figures.

3. Using the HPS Flash Programmer

The Intel Quartus Prime software and Intel Quartus Prime Programmer include the hard processor system (HPS) flash programmer. Hardware designs, such as HPS, incorporate flash memory on the board to store FPGA configuration data or HPS program data. The HPS flash programmer programs the data into a flash memory device connected to an Intel FPGA SoC. The programmer sends file contents over an Intel download cable, such as the Intel FPGA Download Cable II, to the HPS and instructs the HPS to write the data to the flash memory.

The HPS flash programmer programs the following content types to flash memory:

- HPS software executable files — Many systems use flash memory to store non-volatile program code or firmware. HPS systems can boot from flash memory.

  Note: The HPS Flash Programmer is mainly intended to be used for programming the Preloader image to QSPI or NAND flash. Because of the low speed of operation, it is not recommended to be used for programming large files.
- FPGA configuration data — At system power-up, the FPGA configuration controller on the board or HPS read FPGA configuration data from the flash memory to program the FPGA. The configuration controller or HPS may be able to choose between multiple FPGA configuration files stored in flash memory.
- Other arbitrary data files — The HPS flash programmer programs a binary file to any location in a flash memory for any purpose. For example, a HPS program can use this data as a coefficient table or a sine lookup table.

The HPS flash programmer programs the following memory types:

- Quad serial peripheral interface (QSPI) Flash
- Open NAND Flash Interface (ONFI) compliant NAND Flash

Note: The HPS Flash Programmer does not support Intel Stratix 10 SoC or Intel Agilex devices. It is your responsibility to program the flash connected to HPS. Several options are possible:

- Use a bus switch to route the QSPI signals to an external master that does the programming.
- Use software running on HPS to do the programming. For example, U-Boot can be loaded with an Arm* debugger or System Console, and then used to program the flash.

Note: On Intel Stratix 10 SoC or Intel Agilex devices, the HPS can have access to the QSPI flash memory connected to SDM. This flash is programmed using the Intel Quartus Prime Programmer tool that is part Intel Quartus Prime Pro Edition.

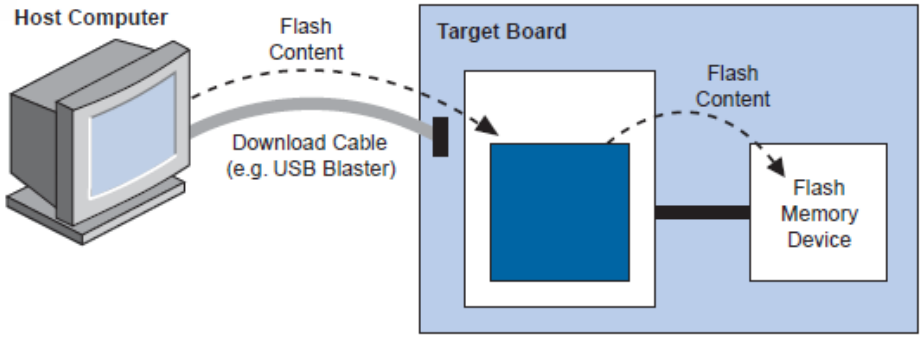
3.1. HPS Flash Programmer Command-Line Utility

Run the HPS flash programmer from the command line. The HPS flash programmer is located in the <*Quartus Prime installation directory*>/quartus/bin directory.

3.2. How the HPS Flash Programmer Works

The HPS flash programmer is divided into a host and a target. The host portion runs on your computer and sends flash programming files and programming instructions over a download cable to the target. The target portion is the HPS in the SoC. The target accepts the programming data flash content and required information about the target flash memory device sent by the host. The target writes the data to the flash memory device.

Figure 31. HPS Flash Programmer



The HPS flash programmer determines the type of flash to program by sampling the boot select (BSEL) pins during cold reset; you do not need to specify the type of flash to program.

3.3. Using the Flash Programmer from the Command Line

3.3.1. HPS Flash Programmer

The HPS flash programmer utility can erase, blank-check, program, verify, and examine the flash. The utility accepts a Binary File with a required "**.bin**" extension.

The HPS flash programmer command-line syntax is:

```
quartus_hps <options> <file.bin>
```

Note: The HPS flash programmer uses byte addressing.

Table 38. HPS Flash Programmer Parameters

Option	Short Option	Required	Description
`--addr`	`-a`	Yes (if the start address is not 0)	This option specifies the start address of the operation to be performed.
`--cable`	`-c`	Yes	This option specifies what download cable to use. To obtain the list of programming cables, run the command "jtagconfig". It lists the available cables, like in the following example: `jtagconfig` • Intel FPGA Download Cable [USB-0] • Intel FPGA Download Cable [USB-1] • Intel FPGA Download Cable [USB-2] The "`-c`" parameter can be the number of the programming cable, or its name. The following are valid examples for the above case: • `-c 1` • `-c "Intel FPGA Download Cable [USB-2]"`
`--device`	`-d`	Yes (if there are multiple HPS devices in the chain)	This option specifies the index of the HPS device. The tool automatically detects the chain and determine the position of the HPS device; however, if there are multiple HPS devices in the chain, the targeted device index must be specified.
`--boot`	N/A	Yes	Option to reconfigure the HPS IOCSR and PINMUX before starting flash programming. For the Intel Quartus Prime HPS Flash Programmer options: • Warm/cold reset HPS (BootROM) so that BootROM can reconfigure the setting. FPGA (for Intel Arria 10) is `nconfig`. • Explicitly configure dedicated I/O and PINMUX Available options: • 1 - Set Breakpoint to halt CPU, warm reset HPS [not recommended](12) • 2 - Set Watchpoint to halt CPU, warm reset HPS(13) • 3 - Explicitly configure dedicated IO and PINMUX(14) • 16 - Cold reset HPS(15) Cyclone V and Intel Arria 10 support values: 1, 2 and 16. Intel Arria 10 supports values: 2, 3 and 16 *Note:* For the first three options, add up integer of 16, so that the HPS cold reset is performed Available options for a cold reset before the flow: • 17 - Cold reset HPS + breakpoint(16)

continued...

(12) Warm reset HPS for BootROM to configure dedicated IO and PINMUX, and use a breakpoint to halt CPU.

(13) Warm reset HPS for BootROM to configure dedicated IO and PINMUX, and use a watchpoint to halt CPU.

(14) Explicitly configure dedicated IO and PINMUX.

(15) Bit-wise on top of the flow, if you set the bit, the tool will perform cold reset first

(16) Cold reset HPS for BootROM to configure dedicated IO and PINMUX, and use a breakpoint to halt CPU.

Option	Short Option	Required	Description
			• 18 - Cold reset HPS + watchpoint(17) • 19 - Cold reset HPS + configure dedicated IO and PINMUX(18)
`--exit_xip`	N/A	Yes (if the QSPI flash device has been put into XIP mode)	This option exits the QSPI flash device from XIP mode. A non-zero value has to be specified for the argument. For example, `quartus_hps -c <cable> -o <operation> --exit_xip=0x80`.
`--operation`	`-o`	Yes	This option specifies the operation to be performed. The following operations are supported: • I—Read IDCODE of SOC device and discover Access Port • S—Read Silicon ID of the flash • E—Erase flash • B—Blank-check flash • P—Program flash • V—Verify flash • EB—Erase and blank-check flash • BP—Program <*BlankCheck*> flash • PV—Program and verify flash • BPV—Program (blank-check) and verify flash • X—Examine flash *Note:* The program begins with erasing the flash operation before programming the flash by default.
`--size`	`-s`	No	This option specifies the number of bytes of data to be performed by the operation. `size` is optional.
Note: The following options: `repeat` and `interval` must be used together and are optional. The HPS BOOT flow supports up to four images where each image is identical. These options duplicate the operation data; therefore you do not need embedded software to create a large file containing duplicate images.			
`--repeat`	`-t`	No	`repeat`—specifies the number of duplicate images for the operation to perform.
`--interval`	`-i`	No	`interval`—specifies the repeated address. The default value is 64 kilobytes (KB).

3.3.2. HPS Flash Programmer Command Line Examples

Type `quartus_hps --help` to obtain information about usage. You can also type `quartus_hps --help=<option>` to obtain more details about each option. For example "quartus_hps --help=o".

Example 2. Program File to Address 0 of Flash

`quartus_hps –c 1 –o P input.bin` programs the input file (**input.bin**) into the flash, starting at flash address 0 using a cable 1.

(17) Cold reset HPS for BootROM to configure dedicated IO and PINMUX, and use a watchpoint to halt CPU.

(18) Cold reset HPS, then explicitly configure dedicated IO and PINMUX.

Example 3. Program First 500 Bytes of File to Flash (Decimal)

`quartus_hps –c 1 –o PV –a 1024 –s 500 input.bin` programs the first 500 bytes of the input file (**input.bin**) into the flash, starting at flash address 1024, followed by a verification using a cable 1.

Note: Without the prefix "0x" for the flash address, the tool assumes it is decimal.

Example 4. Program First 500 Bytes of File to Flash (Hexadecimal)

`quartus_hps –c 1 –o PV –a 0x400 –s 500 input.bin` programs the first 500 bytes of the input file (**input.bin**) into the flash, starting at flash address 1024, followed by a verification using a cable *1*.

Note: With the prefix 0x, the tool assumes it is hexadecimal.

Example 5. Program File to Flash Repeating Twice at Every 1 MB

`quartus_hps –c 1 –o BPV –t 2 –i 0x100000 input.bin` programs the input file (**input.bin**) into the flash, using a cable 1. The operation repeats itself twice at every 1 megabyte (MB) of the flash address. Before the program operation, the tool ensures the flash is blank. After the program operation, the tool verifies the data programmed.

Example 6. Erase Flash on the Flash Addresses

`quartus_hps –c 1 –o EB input.bin` erases the flash on the flash addresses where the input file (**input.bin**) resides, followed by a blank-check using a cable 1.

Example 7. Erase Full Chip

`quartus_hps –c 1 –o E` erases the full chip, using a cable 1. When no input file (**input.bin**) is specified, it erases all the flash contents.

Example 8. Erase Specified Memory Contents of Flash

`quartus_hps –c 1 –o E –a 0x100000 –s 0x400000` erases specified memory contents of the flash. For example, 4 MB worth of memory content residing in the flash address, starting at 1 MB, are erased using a cable 1.

Example 9. Examine Data from Flash

`quartus_hps –c 1 –o X –a 0x98679 –s 56789 output.bin` examines 56789 bytes of data from the flash with a 0x98679 flash start address, using a cable 1.

3.4. Supported Memory Devices

Table 39. QSPI Flash

Flash Device	Manufacturer	Device ID	DIE #	Density (Mb)	Voltage
M25P40	Micron	0x132020	1	4	3.3
N25Q064	Micron	0x17BA20	1	64	3.3
N25Q128	Micron	0x18BA20	1	128	3.3
N25Q128	Micron	0x18BB20	1	128	1.8
N25Q256	Micron	0x19BA20	1	256	3.3

continued...

Flash Device	Manufacturer	Device ID	DIE #	Density (Mb)	Voltage
N25Q256	Micron	0x19BB20	1	256	1.8
MT25QL512	Micron	0x20BA20	1	512	3.3
N25Q512	Micron	0x20BA20	2	512	3.3
MT25QU512	Micron	0x20BB20	1	512	1.8
N25Q512A	Micron	0x20BB20	2	512	1.8
N25Q00AA	Micron	0x21BA20	4	1024	3.3
MT25QU01G	Micron	0x21BB20	2	1024	1.8
N25Q00AA	Micron	0x21BB20	4	1024	1.8
MT25QL02G	Micron	0x22BA20	4	2048	3.3
MT25QU02G	Micron	0x22BB20	4	2048	1.8
S25FL128S	Cypress	0x182001	1	128 (64KB Sectors)	3.3
S25FL128S	Cypress	0x182001	1	128 (256KB Sectors)	3.3
S25FL256S	Cypress	0x190201	1	256 (64KB Sectors)	3.3
S25FL256S	Cypress	0x190201	1	256 (256KB Sectors)	3.3
S25FL512S	Cypress	0x200201	1	512	3.3
MX25L12835E	Macronix	0x1820C2	1	128	3.3
MX25L25635	Macronix	0x1920C2	1	256	3.3
MX66L51235F	Macronix	0x1A20C2	1	512	3.3
MX66L1G45	Macronix	0x1B20C2	1	1024	3.3
MX66U51235	Macronix	0x3A25C2	1	512	1.8
GD25Q127C	GigaDevice	0x1840C8	1	128	3.3

Table 40. ONFI Compliant NAND Flash

Manufacturer	MFC ID	Device ID	Density (Gb)
Micron	0x2C	0x68	32
Micron	0x2C	0x48	16
Micron	0x2C	0xA1	8
Micron	0x2C	0xF1	8

Note: The above table contains just examples of supported devices. The HPS Flash Programmer supports all ONFI compliant NAND flash devices that are supported by the HPS QSPI Flash Controller.

Note: The HPS Flash Programmer supports Intel Arria 10 devices. For more information, refer to the "Supported Flash Devices for Intel Arria 10" web page.

Related Information

Supported Flash Devices for Intel Arria 10 SoCs

3.5. HPS Flash Programmer User Guide Revision History

Document Version	Intel Quartus Prime Version	Changes
2020.12.14	20.4	Moved HPS Flash Programmer User Guide from *Intel SoC FPGA Embedded Development Suite User Guide* to *Intel Quartus Prime Pro Edition User Guide: Programmer*.

Document Version	Changes
2020.08.07	Maintenance release
2020.05.29	Added descriptions for the bit-wise values not displayed in help output
2019.12.20	Supported with Intel Quartus Prime Standard Edition version 19.1 and Intel Quartus Prime Pro Edition version 19.3 • Maintenance release
2019.05.16	*HPS Flash Programmer*: Documented --boot=18 for QSPI programming
2018.09.24	Added supported memory devices in the "QSPI Flash" table; and updated voltages for several flash devices.
2018.06.18	• Updated chapter to include support for Intel Stratix 10 • Updated chapter to include support for Intel Quartus Prime Pro Edition
2017.05.08	• Intel FPGA rebranding • Rebranded paths and tools for the Standard and Professional versions
2016.11.07	Maintenance release
2016.05.27	Maintenance release
2016.02.17	Added QSPI Flash part number to the QSPI Flash table in the "Supported Memory Devices" chapter
2015.08.06	Added Intel Arria 10 support

A. Intel Quartus Prime Pro Edition User Guide: Programmer Document Archive

If an Intel Quartus Prime version is not listed, the user guide for the previous Intel Quartus Prime version applies.

Intel Quartus Prime Version	User Guide
20.3	Intel Quartus Prime Pro Edition User Guide: Programmer
19.4	Intel Quartus Prime Pro Edition User Guide: Programmer
19.3	Intel Quartus Prime Pro Edition User Guide: Programmer
19.1	Intel Quartus Prime Pro Edition User Guide: Programmer
18.1	Intel Quartus Prime Pro Edition User Guide: Programmer
18.0	Intel Quartus Prime Pro Edition User Guide: Programmer

B. Intel Quartus Prime Pro Edition User Guides

Refer to the following user guides for comprehensive information on all phases of the Intel Quartus Prime Pro Edition FPGA design flow.

Related Information

- Intel Quartus Prime Pro Edition User Guide: Getting Started
  Introduces the basic features, files, and design flow of the Intel Quartus Prime Pro Edition software, including managing Intel Quartus Prime Pro Edition projects and IP, initial design planning considerations, and project migration from previous software versions.
- Intel Quartus Prime Pro Edition User Guide: Platform Designer
  Describes creating and optimizing systems using Platform Designer, a system integration tool that simplifies integrating customized IP cores in your project. Platform Designer automatically generates interconnect logic to connect intellectual property (IP) functions and subsystems.
- Intel Quartus Prime Pro Edition User Guide: Design Recommendations
  Describes best design practices for designing FPGAs with the Intel Quartus Prime Pro Edition software. HDL coding styles and synchronous design practices can significantly impact design performance. Following recommended HDL coding styles ensures that Intel Quartus Prime Pro Edition synthesis optimally implements your design in hardware.
- Intel Quartus Prime Pro Edition User Guide: Design Compilation
  Describes set up, running, and optimization for all stages of the Intel Quartus Prime Pro Edition Compiler. The Compiler synthesizes, places, and routes your design before generating a device programming file.
- Intel Quartus Prime Pro Edition User Guide: Design Optimization
  Describes Intel Quartus Prime Pro Edition settings, tools, and techniques that you can use to achieve the highest design performance in Intel FPGAs. Techniques include optimizing the design netlist, addressing critical chains that limit retiming and timing closure, optimizing device resource usage, device floorplanning, and implementing engineering change orders (ECOs).
- Intel Quartus Prime Pro Edition User Guide: Programmer
  Describes operation of the Intel Quartus Prime Pro Edition Programmer, which allows you to configure Intel FPGA devices, and program CPLD and configuration devices, via connection with an Intel FPGA download cable.
- Intel Quartus Prime Pro Edition User Guide: Block-Based Design
  Describes block-based design flows, also known as modular or hierarchical design flows. These advanced flows enable preservation of design blocks (or logic that comprises a hierarchical design instance) within a project, and reuse of design blocks in other projects.

- Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration
  Describes Partial Reconfiguration, an advanced design flow that allows you to reconfigure a portion of the FPGA dynamically, while the remaining FPGA design continues to function. Define multiple personas for a particular design region, without impacting operation in other areas.
- Intel Quartus Prime Pro Edition User Guide: Third-party Simulation
  Describes RTL- and gate-level design simulation support for third-party simulation tools by Aldec*, Cadence*, Mentor Graphics*, and Synopsys* that allow you to verify design behavior before device programming. Includes simulator support, simulation flows, and simulating Intel FPGA IP.
- Intel Quartus Prime Pro Edition User Guide: Third-party Synthesis
  Describes support for optional synthesis of your design in third-party synthesis tools by Mentor Graphics*, and Synopsys*. Includes design flow steps, generated file descriptions, and synthesis guidelines.
- Intel Quartus Prime Pro Edition User Guide: Third-party Logic Equivalence Checking Tools
  Describes support for optional logic equivalence checking (LEC) of your design in third-party LEC tools by OneSpin*.
- Intel Quartus Prime Pro Edition User Guide: Debug Tools
  Describes a portfolio of Intel Quartus Prime Pro Edition in-system design debugging tools for real-time verification of your design. These tools provide visibility by routing (or "tapping") signals in your design to debugging logic. These tools include System Console, Signal Tap logic analyzer, system debugging toolkits, In-System Memory Content Editor, and In-System Sources and Probes Editor.
- Intel Quartus Prime Pro Edition User Guide: Timing Analyzer
  Explains basic static timing analysis principals and use of the Intel Quartus Prime Pro Edition Timing Analyzer, a powerful ASIC-style timing analysis tool that validates the timing performance of all logic in your design using an industry-standard constraint, analysis, and reporting methodology.
- Intel Quartus Prime Pro Edition User Guide: Power Analysis and Optimization
  Describes the Intel Quartus Prime Pro Edition Power Analysis tools that allow accurate estimation of device power consumption. Estimate the power consumption of a device to develop power budgets and design power supplies, voltage regulators, heat sink, and cooling systems.
- Intel Quartus Prime Pro Edition User Guide: Design Constraints
  Describes timing and logic constraints that influence how the Compiler implements your design, such as pin assignments, device options, logic options, and timing constraints. Use the Interface Planner to prototype interface implementations, plan clocks, and quickly define a legal device floorplan. Use the Pin Planner to visualize, modify, and validate all I/O assignments in a graphical representation of the target device.
- Intel Quartus Prime Pro Edition User Guide: PCB Design Tools
  Describes support for optional third-party PCB design tools by Mentor Graphics* and Cadence*. Also includes information about signal integrity analysis and simulations with HSPICE and IBIS Models.
- Intel Quartus Prime Pro Edition User Guide: Scripting
  Describes use of Tcl and command line scripts to control the Intel Quartus Prime Pro Edition software and to perform a wide range of functions, such as managing projects, specifying constraints, running compilation or timing analysis, or generating reports.