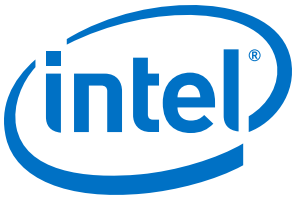


Intel® Quartus® Prime Standard Edition User Guide

Programmer

Updated for Intel® Quartus® Prime Design Suite: **18.1**

Subscribe
Send Feedback

UG-20178 | 2019.06.10
Latest document on the web: PDF | HTML

Contents

1. Programming Intel FPGA Devices

The Intel® Quartus® Prime Programmer allows you to program and configure Intel CPLD, FPGA, and configuration devices. You generate the primary device programming files in the Assembler during a full design compilation. In addition, you can generate secondary programming files for alternative programming methods. After you program the device you can test functionality on a circuit board.

1.1. Programming Flow

In the FPGA flow, device programming requires a fully compiled design that includes the programming or configuration files that the Assembler generates.

To program a device:

1. Convert the programming or configuration file to target the configuration device and, optionally, create secondary programming files.

Table 1. Programming and Configuration File Format

File Format	FPGA	CPLD	Configuration Device	Serial Configuration Device
SRAM Object File (`.sof`)	Yes	—	—	—
Programmer Object File (`.pof`)	—	Yes	Yes	Yes
JEDEC JESD71 STAPL Format File (`.jam`)	Yes	Yes	Yes	—
Jam Byte Code File (`.jbc`)	Yes	Yes	Yes	—

2. In the Intel Quartus Prime Programmer, program and configure the FPGA, CPLD, or configuration device with the appropriate programming or configuration files.

The FPGA now contains the design that you specified in the Intel Quartus Prime project.

1.1.1. 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

1.1.2. Optional Programming or Configuration Files

The Intel Quartus Prime software can generate optional programming or configuration files in formats that you can use with programming tools other than the Intel Quartus Prime Programmer.

When you compile a design in the Intel Quartus Prime software, the Assembler automatically generates either a `.sof` or `.pof` file. The Assembler also allows you to convert FPGA configuration files to programming files for configuration devices.

Related Information

AN 425: Using Command-Line Jam STAPL Solution for Device Programming

1.1.3. Secondary Programming Files

The Intel Quartus Prime software generates programming files in various formats for use with different programming tools.

Table 2. Intel Quartus Prime Software Support for Secondary File Types

File Type	Intel Quartus Prime Software Generate	Intel Quartus Prime Programmer Support
`.sof`	Yes	Yes
`.pof`	Yes	Yes
`.jam`	Yes	Yes
`.jbc`	Yes	Yes
JTAG Indirect Configuration File (`.jic`)	Yes	Yes
Serial Vector Format File (`.svf`)	Yes	—
Hexadecimal (Intel-Format) Output File (`.hexout`)	Yes	—
Raw Binary File (`.rbf`)	Yes	Yes (1)
Raw Binary File for Partial Reconfiguration (`.rbf`)	Yes	Yes (2)
Tabular Text File (`.ttf`)	Yes	—
Raw Programming Data File (`.rpd`)	Yes	—

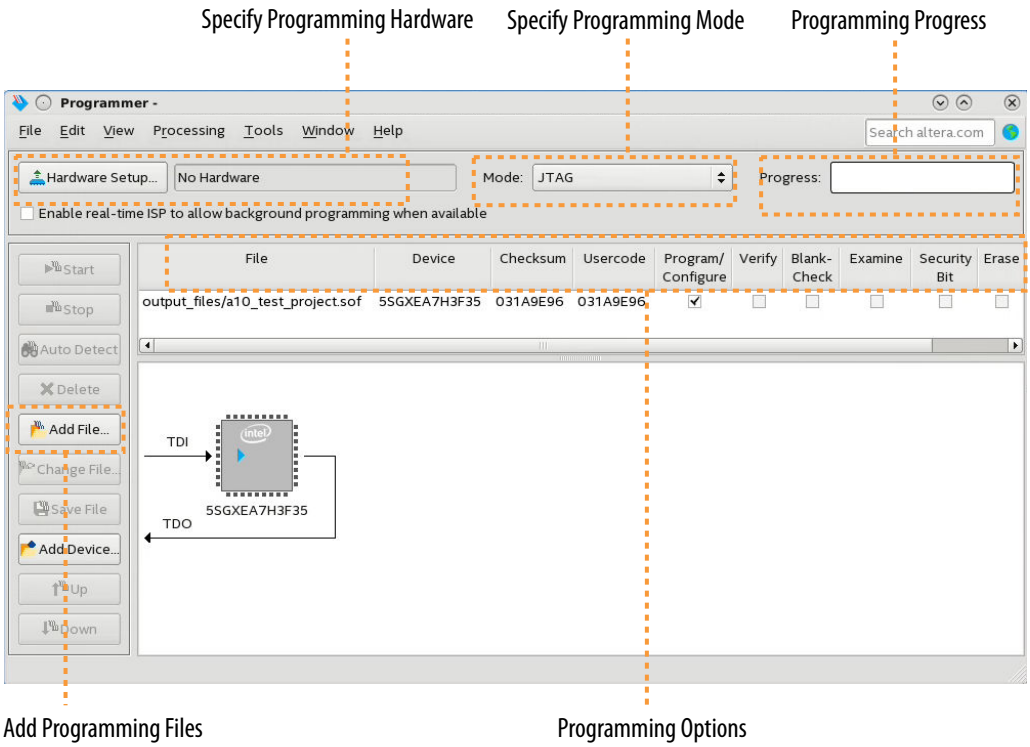
1.2. Intel Quartus Prime Programmer

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

(1) The Intel Quartus Prime Programmer supports Raw Binary (`.rbf`) File format in Passive Serial (PS) configuration mode.

(2) The Intel Quartus Prime Programmer supports Raw Binary File for Partial Reconfiguration (`.rbf`) File format in JTAG debug mode.

Figure 1. 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.

Related Information

Programmer Page (Options Dialog Box)
In Intel Quartus Prime Help

1.2.1. 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.

1.2.2. 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.

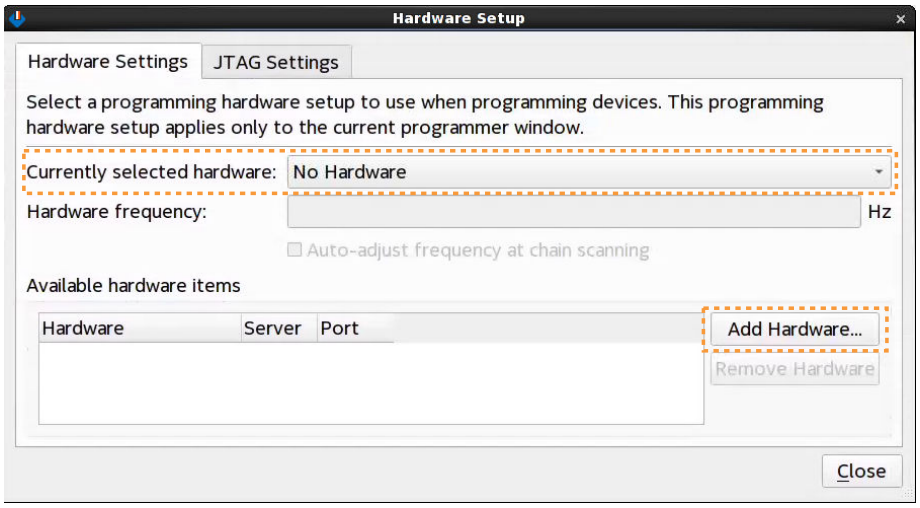
A JTAG server allows the Intel Quartus Prime Programmer to access the JTAG programming hardware connected to a remote computer through the JTAG server of that computer. The JTAG server allows you to control the programming or configuration of devices from a single computer through other computers at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

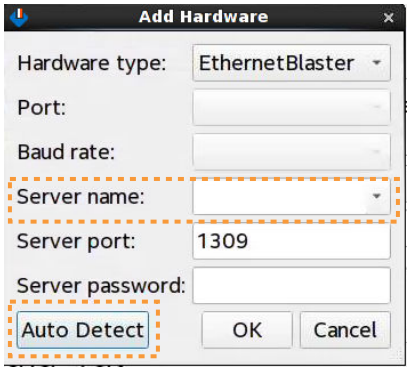
1. In the Programmer, click **Hardware Setup**.

Figure 2. 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 3. Add New Hardware



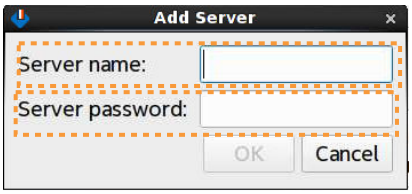
3. On the **Hardware Settings** tab, select your connected programming hardware in **Currently selected hardware**. This list is empty until you connect and add programming hardware to your system.
4. Enable or disable **Auto-adjust frequency at chain scanning** to automatically adjust the **Hardware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

1.2.3. Setting the JTAG Hardware

The JTAG server allows the Intel Quartus Prime Programmer to access the JTAG hardware. You can also access the JTAG download cable or programming hardware connected to a remote computer through the JTAG server of that computer. With the JTAG server, you can 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.

1.2.3.1. 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.

1.2.4. 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.

1.3. Programming and Configuration Modes

The following table lists the programming and configuration modes supported by Intel FPGA devices.

Table 3. Programming and Configuration Modes

Configuration Mode Supported by the Intel Quartus Prime Programmer	FPGA	CPLD	Configuration Device	Serial Configuration Device
JTAG	Yes	Yes	Yes	—
Passive Serial (PS)	Yes	—	—	—
Active Serial (AS) Programming	—	—	—	Yes
Configuration via Protocol (CvP)	Yes	—	—	—
In-Socket Programming	—	Yes (except for MAX® II CPLDs)	Yes	Yes

Related Information

Configuration via Protocol (CvP) Implementation in V-series Intel FPGAs Devices User Guide

Describes the CvP configuration mode.

1.4. Enabling Bitstream File Compression and Security

You can optionally enable bitstream compression (and decompression) to reduce the size of your programming bitstream 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.

You can separately enable generation of encryption key programming files and user-defined 256-bit security key to protect and authenticate the configuration bitstream. Encryption of the bitstream also offers side-channel protection from non-intrusive attack.

Related Information

AN 556: Using the Design Security Features in Intel FPGAs

1.5. Convert Programming Files Tool

The Intel Quartus Prime software provides the **Convert Programming Files** tool (**File ➤ Convert Programming Files**), which allows you to convert programming files from one file format to another.

The **Convert Programming Files** tool supports the following design families:

Table 4. Device Families that the Convert Programming Files Tool Supports

Software	Version	Supported Device Families
Intel Quartus Prime Standard Edition	18.1	• Intel Arria® 10 • Intel MAX 10 • Intel Cyclone® 10 LP • MAX V • Cyclone V, Arria V, Stratix® V.

The **Convert Programming Files** tool also allows you to configure multiple devices with an external host, such as a microprocessor or CPLD. For example, you can combine multiple `.sof` files into one `.pof` file.

To save time in subsequent conversions, click **Save Conversion Setup** to write the conversion specifications in a Conversion Setup File (`.cof`).

To load a `.cof` setup in the **Convert Programming Files** dialog box, click **Open Conversion Setup Data** .

For example, to store the FPGA data in configuration devices, you can convert the `.sof` data to another format, such as `.pof`, `.hexout`, `.rbf`, `.rpd`, or `.jic`, and then program the configuration device.

Example 1. Conversion Setup File Contents

```
<?xml version="1.0" encoding="US-ASCII" standalone="yes"?>
<cof>
    <output_filenameoutput_file.pof</output_filename>
    <n_pages>1</n_pages>
    <width>1</width>
    <mode>14</mode>
    <sof_data>
```

```
        <user_name>Page_0</user_name>
        <page_flags>1</page_flags>
        <bit0>
            <sof_filename>/users/user1/template/output_files/template_test.sof</
sof_filename>
        </bit0>
    </sof_data>
    <version>7</version>
    <create_cvp_file>0</create_cvp_file>
    <create_hps_iocsr>0</create_hps_iocsr>
    <auto_create_rpd>0</auto_create_rpd>
    <options>
        <map_file>1</map_file>
    </options>
    <MAX10_device_options>
        <por>0</por>
        <io_pullup>1</io_pullup>
        <auto_reconfigure>1</auto_reconfigure>
        <isp_source>0</isp_source>
        <verify_protect>0</verify_protect>
        <epof>0</epof>
        <ufm_source>0</ufm_source>
    </MAX10_device_options>
    <advanced_options>
        <ignore_epcs_id_check>0</ignore_epcs_id_check>
        <ignore_condone_check>2</ignore_condone_check>
        <plc_adjustment>0</plc_adjustment>
        <post_chain_bitstream_pad_bytes>-1</post_chain_bitstream_pad_bytes>
        <post_device_bitstream_pad_bytes>-1</post_device_bitstream_pad_bytes>
        <bitslice_pre_padding>1</bitslice_pre_padding>
    </advanced_options>
</cof>
```

Related Information

Convert Programming Files Dialog Box
In Intel Quartus Prime Help

1.5.1. Debugging the Configuration

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 5. Advanced Options Settings

Option Setting	Description	Values
Disable EPCS ID check	Directs the FPGA to skips the EPCS silicon ID verification. Applies to single and multi device AS configuration modes on all devices.	Default setting is ON (EPCS 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. The CONF_DONE error check is disabled by default for Stratix V, Arria V, and Cyclone V devices for AS-PS multi device configuration mode.	Default setting is OFF (AS mode CONF_DONE error check is enabled).

continued...

Option Setting	Description	Values
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 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)	—	—	—

continued...

(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

Failure Symptoms	Disable EPCS ID Check	Disable AS Mode CONF_DONE Error Check	PLC Settings	Post-Chain Bitstream Pad Bytes	Post-Device Bitstream Pad Bytes	Bitslice Padding Value
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

1.5.2. Converting Programming Files for Partial Reconfiguration

The **Convert Programming File** dialog box supports the following programming file generation and option for Partial Reconfiguration:

- Partial-Masked SRAM Object File (.pmsf) output file generation, with .msf and .sof as input files.
- .rbf for Partial Reconfiguration output file generation, with a .pmsf as the input file.

  Note: The .rbf for Partial Reconfiguration file is only for Partial Reconfiguration.
- Providing the **Enable decompression during Partial Reconfiguration** option to enable the option bit for bitstream decompression during Partial Reconfiguration, when converting a full design .sof to any supported file type.

1.5.2.1. Generating .pmsf using a .msf and a .sof

To generate the .pmsf in the **Convert Programming Files** dialog box:

1. In the **Convert Programming Files** dialog box, under the **Programming file type** field, select **Partial-Masked SRAM Object File (.pmsf)**.
2. In **File name**, specify the necessary output file name.
3. In the **Input files to convert** field, add necessary input files to convert. You can add only a .msf and .sof.
4. Click **Generate**.

1.5.2.2. Generating a .rbf for Partial Reconfiguration from a .pmsf file

After generating the .pmsf file, you convert the .pmsf file into a .rbf file with the **Convert Programming Files** dialog box.

To generate the `.rbf` for Partial Reconfiguration:

1. In the **Convert Programming Files** dialog box, in the **Programming file type** field, select **Raw Binary File for Partial Reconfiguration (.rbf)**.
2. In the **File name** field, specify the output file name.
3. In the **Input files to convert** field, add input files to convert.

   You can add only one `.pmsf` file.
4. Select the `.pmsf`, and click **Properties**.
   The **PMSF File Properties** dialog box appears.
5. Make your selection either by turning on or turning off the following options:
   - **Compression option**—This option enables compression on Partial Reconfiguration bitstream. If you turn on this option, then you must turn on the **Enable decompression during Partial Reconfiguration** option.
   - **Enable SCRUB mode option**—The default of this option is based on AND/OR mode. This option is valid only when Partial Reconfiguration masks in your design are not overlapped vertically. Otherwise, you cannot generate the `.rbf` for Partial Reconfiguration.
   - **Write memory contents option**—This option is a workaround for initialized RAM/ROM in a Partial Reconfiguration region.

   For more information about these options refer to *Design Planning for Partial Reconfiguration* in *Intel Quartus Prime Standard Edition User Guide: Partial Reconfiguration*.
6. Click **OK**.
7. Click **Generate**.

1.5.2.3. Enable Decompression During Partial Reconfiguration Option

You can turn on the **Enable decompression during Partial Reconfiguration** option in the **SOF File Properties: Bitstream Encryption** dialog box, which you can access from the **Convert Programming File** dialog box. This option is available when converting a `.sof` to any supported programming file types listed in *Secondary Programming Files*.

This option is hidden for other targeted devices that do not support Partial Reconfiguration. To view this option in the **SOF File Properties: Bitstream Encryption** dialog box, the `.sof` must be targeted on an Intel FPGA device that supports Partial Reconfiguration.

If you turn on the **Compression** option when generating the `.rbf` for Partial Reconfiguration, then you must turn on the **Enable decompression during Partial Reconfiguration** option.

1.6. 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
- Customizable Flash Programmer User Guide
- Generic Flash Programmer User Guide

1.7. JTAG Debug Mode for Partial Reconfiguration

The JTAG debug mode allows you to configure partial reconfiguration bitstream through the JTAG interface. Use this feature to debug PR bitstream and eventually helping you in your PR design prototyping. This feature is available for internal and external host. Using the JTAG debug mode forces the Data Source Controller to be in x16 mode.

During JTAG debug operation, the JTAG command sent from the Intel Quartus Prime Programmer ignores and overrides most of the Partial Reconfiguration IP core interface signals (`clk`, `pr_start`, `double_pr`, `data[]`, `data_valid`, and `data_read`).

Note: The TCK is the main clock source for PR IP core during this operation.

You can view the status of Partial Reconfiguration operation in the messages box and the Progress bar in the Intel Quartus Prime Programmer. The PR_DONE, PR_ERROR, and CRC_ERROR signals are monitored during PR operation and reported in the Messages box at the end of the operation.

The Intel Quartus Prime Programmer can detect the one or more PR_DONE instructions in plain or compressed PR bitstream and, therefore, can handle single or double PR cycle accordingly. However, only single PR cycle is supported for encrypted Partial Reconfiguration bitstream in JTAG debug mode (provided that the specified device is configured with the encrypted base bitstream which contains the PR IP core in the design).

Note: Configuring an incompatible PR bitstream to the specified device may corrupt your design, including the routing path and the PR IP core placed in the static region. When this issue occurs, the PR IP core stays in an undefined state, and the Intel Quartus Prime Programmer is unable to reset the IP core. As a result, the Intel Quartus Prime Programmer generates the following error when you try to configure a new PR bitstream:

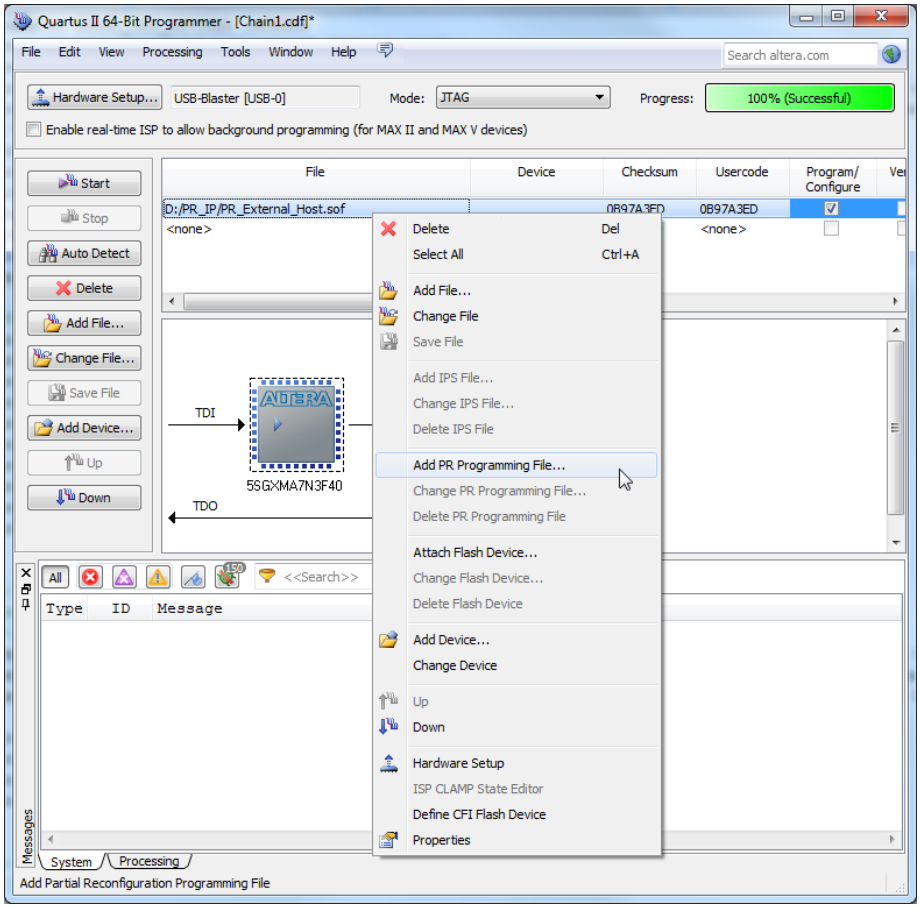
```
Error (12897): Partial Reconfiguration status: Can't reset the PR megafunction.
This issue occurred because the design was corrupted by an incompatible PR
bitstream in the previous PR operation. You must reconfigure the device with a
good design.
```

1.7.1. Configuring Partial Reconfiguration Bitstream in JTAG Debug Mode

To configure the Partial Reconfiguration bitstream in JTAG debug mode, follow these steps:

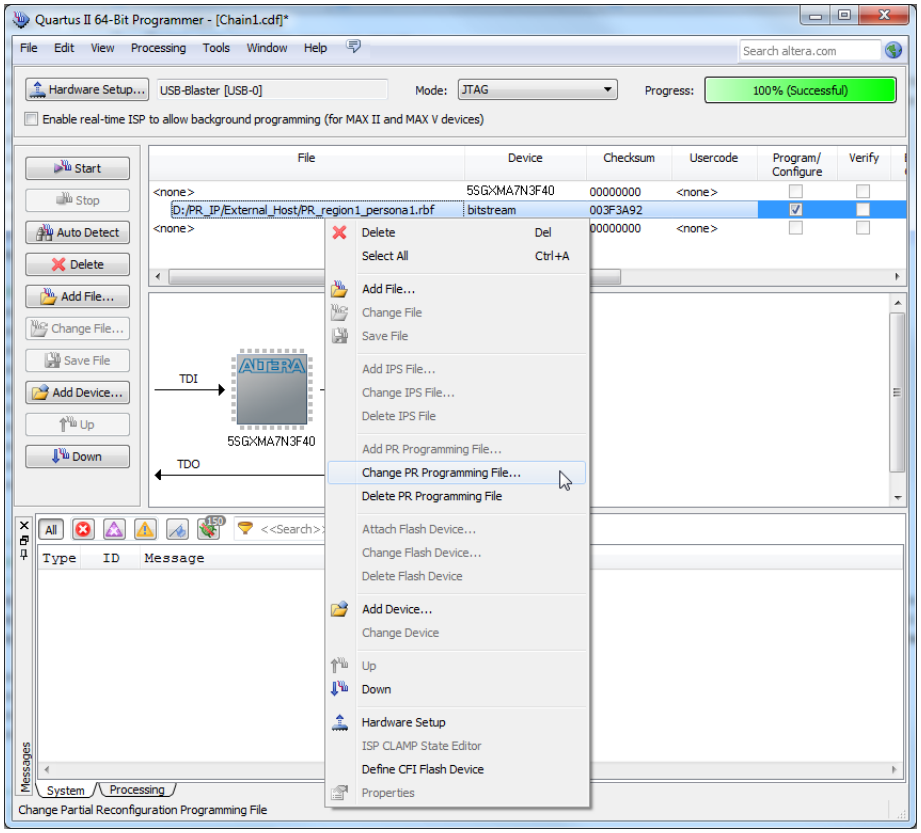
1. In the Intel Quartus Prime Programmer GUI, right click a highlighted base bitstream (in `.sof`) and then click **Add PR Programming File** to add the PR bitstream (`.rbf`).

Figure 5. Adding PR Programming File



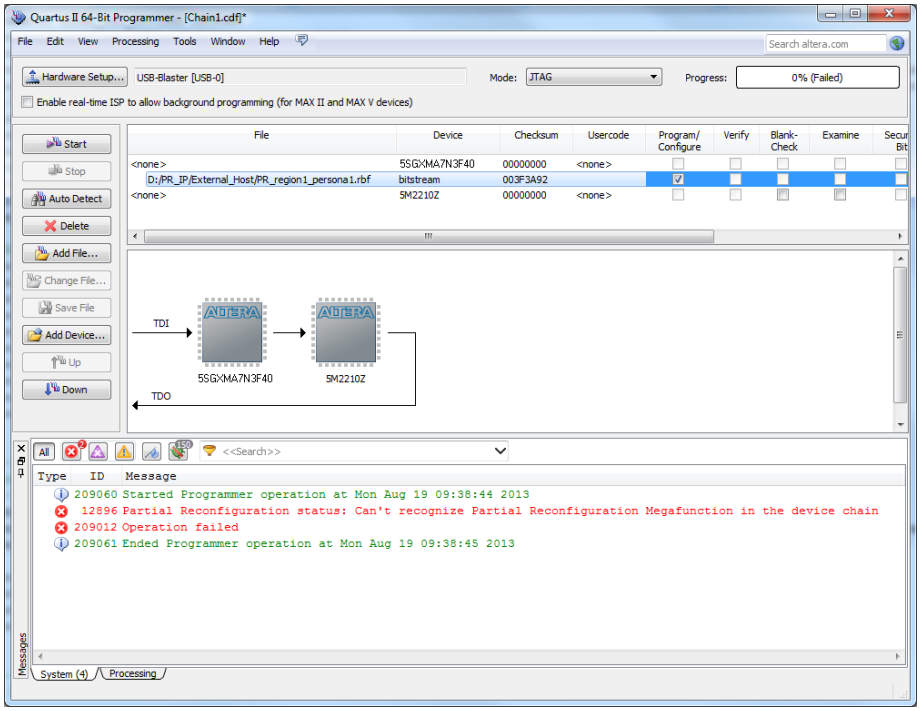
2. After adding the PR bitstream, you can change or delete the Partial Reconfiguration programming file by clicking **Change PR Programming File** or **Delete PR Programming File**.

Figure 6. Change PR Programming File or Delete PR Programming File



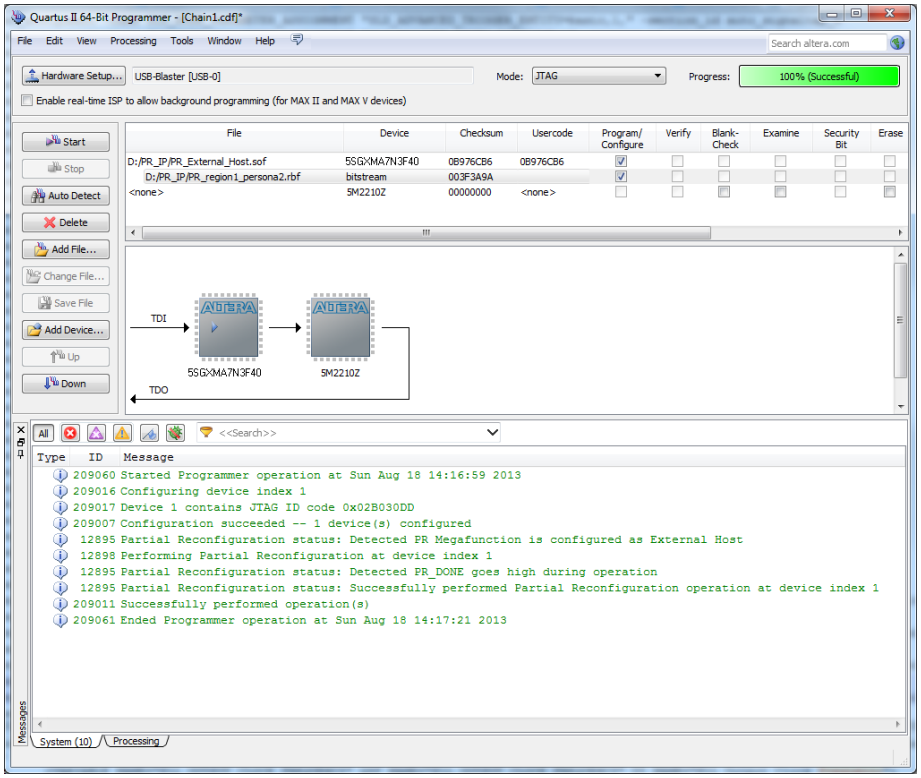
3. Click **Start** to configure the PR bitstream. The Intel Quartus Prime Programmer generates an error message if the specified device does not contain the PR IP core in the design (you must instantiate the Partial Reconfiguration IP core in your design to use the JTAG debug mode).

Figure 7. Starting PR Bitstream Configuration



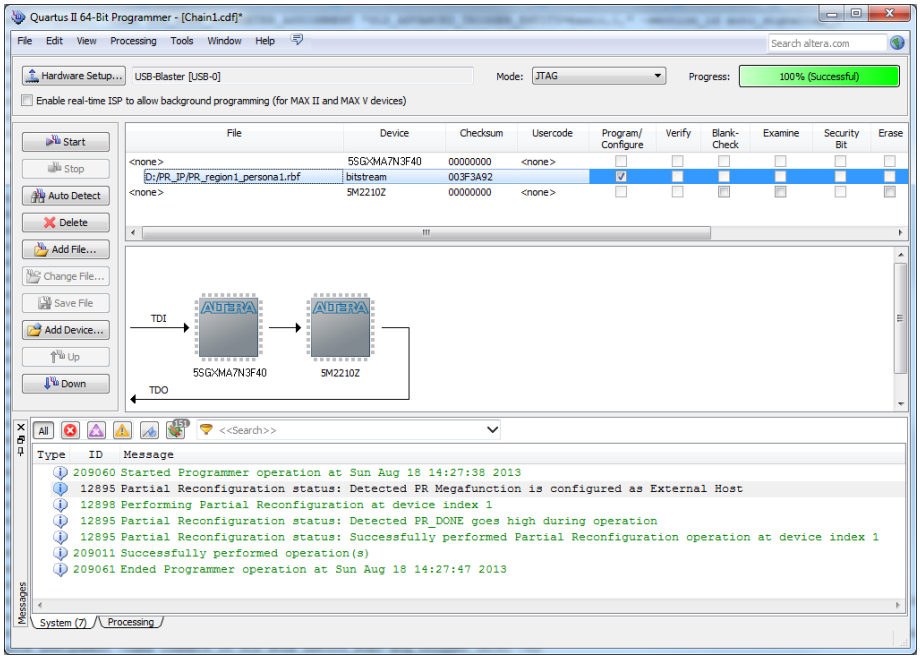
4. Configure the valid `.rbf` in JTAG debug mode with the Intel Quartus Prime Programmer.

Figure 8. Configuring Valid .rbf



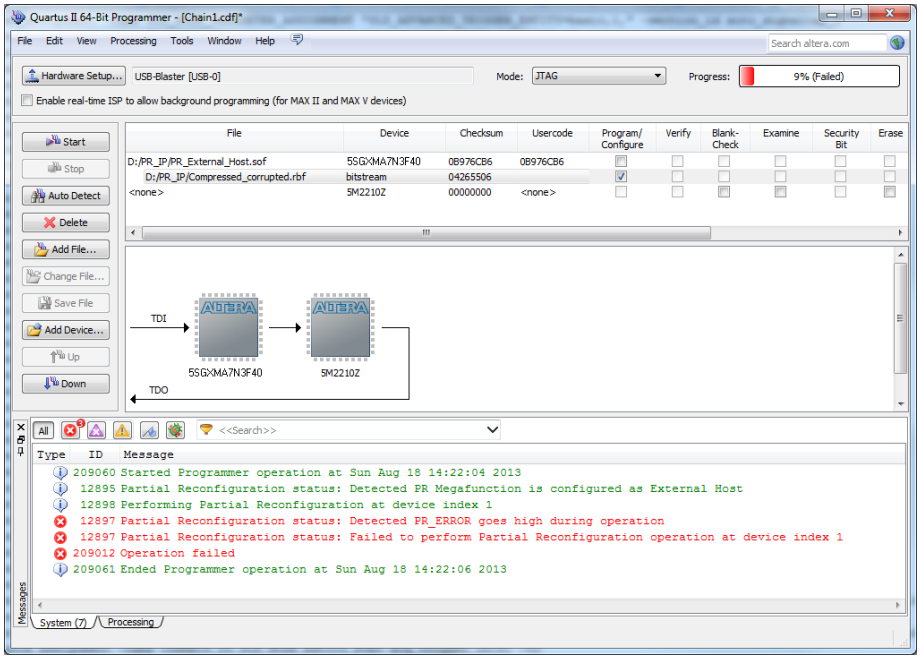
5. The JTAG debug mode is also supported if the PR IP core is pre-programmed on the specified device.

Figure 9. Partial Reconfiguration IP Core Successfully Pre-programmed



6. The Intel Quartus Prime Programmer reports error when you try to configure the corrupted `.rbf` in JTAG debug mode.

Figure 10. Configuring Corrupted .rbf



1.8. 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 is available for Arria V, Stratix V, Cyclone V, Intel MAX 10, and Intel Arria 10 device families.

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.

1.8.1. Obtaining Project Hash for Arria V, Stratix V, Cyclone V and Intel MAX 10 Devices

To obtain the project hash value of a `.sof` programming file for a design targeted to Arria V, Stratix V, Cyclone V, and Intel MAX 10 devices, use the following command, which dumps out metadata information that includes the project hash.

```
quartus_cpf --info <sof-file-name>
```

Example 2. Output of Project Hash Extraction

In this example, the programming file name is `cb_intosc.sof`.

```
File: cb_intosc.sof
        File CRC: 0x0000
        Creator: Quartus Prime Compiler
        Version 17.0.0 Internal Build 565 02/09/2017 SJ Standard Edition
        Comment: UNIX
        Device: 5SGSMD5K2F40
        Data checksum: 0x02534E5A
        JTAG usercode: 0x02534E5A
        Project Hash:  0x556e737065636966696564
```

1.8.2. 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 3. Output of Project Hash Command

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

```
Info: ******************************************************************
Info: Running Quartus Prime Assembler
Info: Version 17.1.0 Build 569 08/23/2017 SJ Standard 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 Aug 25 18:22:53 2017
Info: Command: quartus_asm --project_hash one_wire.sof
Info: Quartus(args): one_wire.sof
Info: Using INI file /data/test_asm/dis_all/quartus.ini
0x0e43694a1ffaf5da6088f900ffb0f7b6
Info (23030): Evaluation of Tcl script /tools/quartuskit/17.1std/quartus/
common/tcl/apps/qasm/project_hash.tcl was successful
Info: Quartus Prime Assembler was successful. 0 errors, 0 warnings
Info: Peak virtual memory: 1123 megabytes
Info: Processing ended: Fri Aug 25 18:22:59 2017
Info: Elapsed time: 00:00:06
Info: Total CPU time (on all processors): 00:00:03
```

1.9. 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

About Intel Quartus Prime Scripting
In Intel Quartus Prime Help

1.9.1. 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 Standard Edition Handbook Volume 2*
- Command Line Scripting
  In *Intel Quartus Prime Standard Edition User Guide: Scripting*

1.9.2. 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>
```

1.9.3. Generating Raw Binary File for Partial Reconfiguration using a .pmsf

You can generate a `.rbf` for Partial Reconfiguration with the `quartus_cpf` command by typing the following command:

```
quartus_cpf –o foo.txt –c <pr_revision.pmsf> <pr_revision.rbf>
```

Note: You must run this command in the same directory where the files are located.

1.10. Programming Intel FPGA Devices Revision History

The following revision history applies to this chapter:

Document Version	Intel Quartus Prime Version	Changes
2019.06.10	18.1.1	• Added links to *Generic Flash Programmer User Guide*. • Updated "Programming with Flash Loaders" topic to reflect new Generic Flash Programmer. • Removed references to obsolete 32-bit stand-alone Programmer. • Updated screenshots for latest GUI.
2018.09.24	18.1.0	• Initial release in Intel Quartus Prime Standard Edition User Guide. • Renamed topic: *Project Hash* to *Verifying if Programming Files Correspond to a Compilation of the Same Source Files*. •
2017.11.06	17.1.0	• Updated Project Hash feature.
2017.05.08	17.0.0	• Added Project Hash feature.
2015.11.02	15.1.0	Changed instances of *Quartus II* to *Intel Quartus Prime*.
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.
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.

continued...

Document Version	Intel Quartus Prime Version	Changes
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.

Related Information

Documentation Archive

For previous versions of the *Intel Quartus Prime Handbook*, search the documentation archives.

1.11. 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
18.1.0	Intel Quartus Prime Standard Edition User Guide: Programmer

A. Intel Quartus Prime Standard Edition User Guides

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

Related Information

- Intel Quartus Prime Standard Edition User Guide: Getting Started
  Introduces the basic features, files, and design flow of the Intel Quartus Prime Standard Edition software, including managing Intel Quartus Prime Standard Edition projects and IP, initial design planning considerations, and project migration from previous software versions.
- Intel Quartus Prime Standard Edition User Guide: Platform Designer
  Describes creating and optimizing systems using Platform Designer (Standard), a system integration tool that simplifies integrating customized IP cores in your project. Platform Designer (Standard) automatically generates interconnect logic to connect intellectual property (IP) functions and subsystems.
- Intel Quartus Prime Standard Edition User Guide: Design Recommendations
  Describes best design practices for designing FPGAs with the Intel Quartus Prime Standard Edition software. HDL coding styles and synchronous design practices can significantly impact design performance. Following recommended HDL coding styles ensures that Intel Quartus Prime Standard Edition synthesis optimally implements your design in hardware.
- Intel Quartus Prime Standard Edition User Guide: Design Compilation
  Describes set up, running, and optimization for all stages of the Intel Quartus Prime Standard Edition Compiler. The Compiler synthesizes, places, and routes your design before generating a device programming file.
- Intel Quartus Prime Standard Edition User Guide: Design Optimization
  Describes Intel Quartus Prime Standard 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, and optimization of device resource usage.
- Intel Quartus Prime Standard Edition User Guide: Programmer
  Describes operation of the Intel Quartus Prime Standard 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 Standard 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 Standard 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 Standard 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 Standard Edition User Guide: Debug Tools
  Describes a portfolio of Intel Quartus Prime Standard 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, Transceiver Toolkit, In-System Memory Content Editor, and In-System Sources and Probes Editor.
- Intel Quartus Prime Standard Edition User Guide: Timing Analyzer
  Explains basic static timing analysis principals and use of the Intel Quartus Prime Standard 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 Standard Edition User Guide: Power Analysis and Optimization
  Describes the Intel Quartus Prime Standard 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 Standard 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 Pin Planner to visualize, modify, and validate all I/O assignments in a graphical representation of the target device.
- Intel Quartus Prime Standard 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 Standard Edition User Guide: Scripting
  Describes use of Tcl and command line scripts to control the Intel Quartus Prime Standard Edition software and to perform a wide range of functions, such as managing projects, specifying constraints, running compilation or timing analysis, or generating reports.