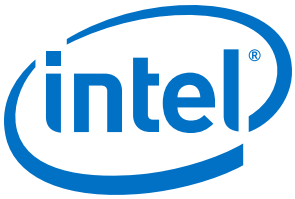


Intel® SoC FPGA Embedded Development Suite User Guide

Updated for Intel® Quartus® Prime Design Suite: **20.1**

Subscribe
Send Feedback

UG-1137 | 2020.08.07
Latest document on the web: PDF | HTML

Contents

1. Introduction to the Intel® SoC FPGA EDS....4
1.1. Tool Versions....4
1.2. Differences Between Standard and Professional Editions....4
1.3. Overview....5
1.4. Hardware and Software Development Roles....6
1.4.1. Hardware Engineer....7
1.4.2. Bare Metal and RTOS Developer....7
1.4.3. Linux Kernel and Driver Developer....7
1.4.4. Linux Application Developer....8
1.5. Hardware – Software Development Flow....8
1.6. Introduction to the Intel SoC FPGA EDS Revision History....9

2. Installing the Tools....11
2.1. Installation Folders....11
2.2. Linux....12
2.2.1. Installing SoC EDS....12
2.2.2. Installing Arm DS....12
2.2.3. Installing Linaro Bare Metal Toolchain....13
2.3. Windows....14
2.3.1. Installing SoC EDS....14
2.3.2. Installing Arm DS....15
2.3.3. Installing Cygwin....15
2.3.4. Installing MingGW....17
2.3.5. Installing Linaro Bare Metal Toolchain....21
2.4. Installing the Intel SoC FPGA EDS Document Revision History....22

3. Running the Tools....24
3.1. Linux....24
3.2. Windows....25
3.3. Running the Tools Revision History....25

4. SoC EDS Licensing....26
4.1. Getting the License....26
4.2. Activating the License....26
4.3. Intel SoC FPGA EDS Licensing Revision History....30

5. Arm Development Studio for Intel SoC FPGA Edition....32
5.1. Arm DS for Intel SoC FPGA Edition Revision History....33

6. Boot Tools User Guide....34
6.1. Introduction....34
6.2. BSP Generator....35
6.2.1. BSP Generation Flow....35
6.2.2. BSP Generator Graphical User Interface....38
6.2.3. BSP Generator Command Line Interface....38
6.2.4. BSP Files and Folders....39
6.2.5. BSP Settings....40
6.3. Bootloader Image Tool (`mkpimage`)....40
6.3.1. Operation....40

6.3.2. Header File Format....41
6.3.3. Tool Usage....42
6.3.4. Output Image Layout....43
6.4. U-Boot Image Tool (mkimage)....45
6.4.1. Tool Options....46
6.4.2. Usage Examples....46
6.5. Building the Bootloader....46
6.5.1. Building the Cyclone V SoC and Arria V SoC Preloader....46
6.5.2. Building the Intel Arria 10 SoC Bootloader....46
6.5.3. Building the Intel Stratix 10 SoC and Intel Agilex Devices....47
6.6. Boot Tools User Guide Revision History....47

7. Hardware Library....49
7.1. Feature Description....50
7.1.1. SoC Abstraction Layer (SoCAL)....50
7.1.2. HW Manager....50
7.2. Hardware Library Reference Documentation....51
7.3. System Memory Map....51
7.4. Hardware Library Revision History....52

8. HPS Flash Programmer User Guide....53
8.1. HPS Flash Programmer Command-Line Utility....54
8.2. How the HPS Flash Programmer Works....54
8.3. Using the Flash Programmer from the Command Line....54
8.3.1. HPS Flash Programmer....54
8.3.2. HPS Flash Programmer Command Line Examples....56
8.4. Supported Memory Devices....57
8.5. HPS Flash Programmer User Guide Revision History....59

9. Bare Metal Compilers....60
9.1. Arm Bare Metal Compilers....60
9.2. Linaro GCC Compiler....60
9.3. Bare Metal Compilers Revision History....61

10. SD Card Boot Utility....62
10.1. Usage Scenarios....62
10.2. Tool Options....63
10.3. SD Card Boot Utility Revision History....64

11. Linux Device Tree Generator....65
11.1. Linux Device Tree Generator Revision History....65

12. Support and Feedback....67
12.1. Support and Feedback Revision History....67

1. Introduction to the Intel® SoC FPGA EDS

The Intel® SoC FPGA Embedded Development Suite (SoC EDS) is a comprehensive tool suite for embedded software development on Intel FPGA SoC devices.

The SoC EDS contains development tools, utility programs, run-time software, and application examples that enable firmware and application software development on Intel SoC hardware platforms.

The SoC EDS is offered in two editions:

- **SoC EDS Standard Edition**—The SoC EDS Standard Edition targets the Arria® V SoC, Cyclone® V SoC, and Intel Arria 10 SoC and must be used only with FPGA projects created in Intel Quartus® Prime Standard Edition. Although the Intel FPGA SoC devices and Intel Quartus Prime Standard Edition support Intel Arria 10 SoC, Intel recommends that you use the Intel Quartus Prime Pro Edition and SoC EDS Professional Edition versions for new Intel Arria 10 SoC Projects.
- **SoC EDS Professional Edition**—The SoC EDS Professional Edition targets the Intel Arria 10 SoC, Intel Stratix® 10 SoC, and Intel Agilex™ and must be used only with FPGA projects created in Intel Quartus Prime Pro Edition.

  Note: The SoC EDS Professional Edition does not support Cyclone V SoC and Arria V SoC, as they are not supported by Intel Quartus Prime Pro Edition.

Related Information

SoC EDS Licensing on page 26

1.1. Tool Versions

Table 1. Tool Versions

Refer to this table when you want to determine the tool versions.

Tool	Version
Linaro* Bare Metal Compiler (GCC)	7.5.0 (Linaro GCC 7.5-2019.12)
Bare Metal	4.8.3 (Linaro GCC 7.2-2017.11)
Arm* Bare Metal Compiler 5	5.06 update 6
Arm Bare Metal Compiler 6	6.14
Arm Development Studio* (DS*) for Intel SoC FPGA Edition	2020.0

1.2. Differences Between Standard and Professional Editions

The main difference between the two different editions exists in the SoC devices that are supported, as mentioned in the previous section.

Things to know about installation and licensing for the two editions:

- The default installation paths are different between editions. For details on the default installation paths, refer to the "Installing the Intel SoC FPGA EDS" chapter.
- There are no licensing differences between SoC EDS editions. None of the components included with SoC EDS require a license. Only the Arm DS for Intel SoC FPGA Edition requires a license.

Related Information

Installing the Tools on page 11

1.3. Overview

The SoC EDS contains tools to help you perform the software development tasks targeting the Intel FPGA SoCs, including:

- Board bring-up
- Device driver development
- Operating system (OS) porting
- Bare Metal application development and debugging
- OS- and Linux-based application development and debugging
- Debug systems running symmetric multiprocessing (SMP)
- Debug software targeting soft IP residing on the FPGA portion of the device

Major components downloaded and installed with the SoC EDS package include:

- SoC Hardware Library (HWLIB)
- Hardware-to-software interface utilities:
  - Second stage bootloader generator
  - Linux* device tree generator(1)
- Sample applications
- Golden Hardware Reference Designs (GHRD) including:
  - FPGA hardware project
  - FPGA hardware SRAM Object File (.sof) file

  Note: The Golden Hardware Reference Designs (GHRD) included with the SoC EDS are not official releases and are intended to be used only as examples. For development purposes, use the official GHRD releases described in the *Golden System Reference Design User Manual* available on the Rocketboards website.
- Embedded command shell allowing easy invocation of the included tools
- SD Card Boot Utility
- Intel Quartus Prime Programmer and Signal Tap

Major components downloaded and installed from a separate download include:

(1) Does not support `bsp-editor`, which is provided in the Intel Quartus Prime Pro Edition software version 20.1.

- Arm Development Studio for Intel SoC FPGA Edition AE Toolkit
- Compiler tool chains:
  - Linaro Bare Metal Compiler (GCC)
  - Arm Bare Metal compiler 5
  - Arm Bare Metal compiler 6

Download information for these components are located in the *Installing the Intel SoC FPGA EDS* section.

Related Information

- Golden System Reference Design User Manual
- GitHub Repository
- Installing the Tools on page 11

1.4. Hardware and Software Development Roles

Depending on your role in hardware or software development, you need a different subset of the SoC EDS toolkit. The following table lists some typical engineering development roles and indicates the tools that each role typically requires.

For more information about each of these tools, refer to the **Intel SoC FPGA Embedded Development Suite** page.

Table 2. Hardware and Software Development Roles

This table lists typical tool usage, but your actual requirements depend on your specific project and organization.

Tool	Hardware Engineer	Bare Metal Developer	RTOS Developer	Linux Kernel and Driver Developer	Linux Application Developer
Arm DS for Intel SoC FPGA Edition Debugging	√	√	√	√	√
Arm DS for Intel SoC FPGA Edition Tracing		√	√	√	
Arm DS for Intel SoC FPGA Edition Cross Triggering		√	√	√	
Hardware Libraries		√	√	√	
Second Stage Bootloader Generator	√	√	√	√	
Flash Programmer		√	√	√	√
Bare Metal Compiler	√	√	√	√	
Linux Device Tree Generator *Note:* This tool is obsolete. For more information, refer to section *Linux Device Tree Generator*.				√	

Related Information

- SoC Embedded Development Suite web page
- *Linux Device Tree Generator* on page 65

1.4.1. Hardware Engineer

As a hardware engineer, you typically design the FPGA hardware in Platform Designer. You can use the debugger of Arm DS for Intel SoC FPGA Edition to connect to the Arm cores and test the hardware. A convenient feature of the Arm DS for Intel SoC FPGA Edition debugger is the soft IP register visibility, using Cortex Microcontroller Software Interface Standard (CMSIS) System View Description (**.svd**) files. With this feature, you can easily read and modify the soft IP registers from the Arm side.

As a hardware engineer, you may generate the Preloader for your hardware configuration. The Preloader is a piece of software that configures the HPS component according to the hardware design.

As a hardware engineer, you may also perform the board bring-up. You can use Arm DS for Intel SoC FPGA Edition debugger to verify that they can connect to the Arm and the board is working correctly.

These tasks require JTAG debugging, which is provided by the Arm DS for Intel SoC FPGA Edition, which is provided through a separate download.

For more information, refer to the *Installing the Intel SoC FPGA EDS* section.

Related Information

- Hardware – Software Development Flow on page 8
- Installing the Tools on page 11

1.4.2. Bare Metal and RTOS Developer

As a Bare Metal or a RTOS developer, you need JTAG debugging and low-level visibility into the system. The following tasks require JTAG debugging, which is enabled by the Arm DS for Intel SoC FPGA Edition.

- To compile your code and the SoC Hardware Library to control the hardware in a convenient and consistent way, use the Bare Metal compiler.
- To program the flash memory on the target board, use the Flash Programmer.

For more information, see the "Licensing" section.

Related Information

SoC EDS Licensing on page 26

1.4.3. Linux Kernel and Driver Developer

As a Linux kernel or driver developer, you may use the same tools the RTOS developers use, because you need low-level access and visibility into the system. However, you must use the Linux compiler instead of the Bare Metal compiler.

These tasks require JTAG debugging, which is enabled by the Arm DS for Intel SoC FPGA Edition, which is provided through a separate download.

For more information, refer to the *Installing the Intel SoC FPGA EDS* section.

For information about how to create a device tree, refer to the "HOWTO Create a Device Tree" web page on Rocketboards.org.

Related Information

- HOWTO Create a Device Tree
- Installing the Tools on page 11

1.4.4. Linux Application Developer

As a Linux application developer, you write code that targets the Linux OS running on the board. Because the OS provides drivers for all the hardware, you do not need low-level visibility over JTAG. Arm DS for Intel SoC FPGA Edition offers a very detailed view of the OS, showing information about the threads that are running and the drivers that are loaded.

Arm DS for Intel SoC FPGA Edition is provided through a separate download.

For more information, refer to the *Installing the Intel SoC FPGA EDS* section.

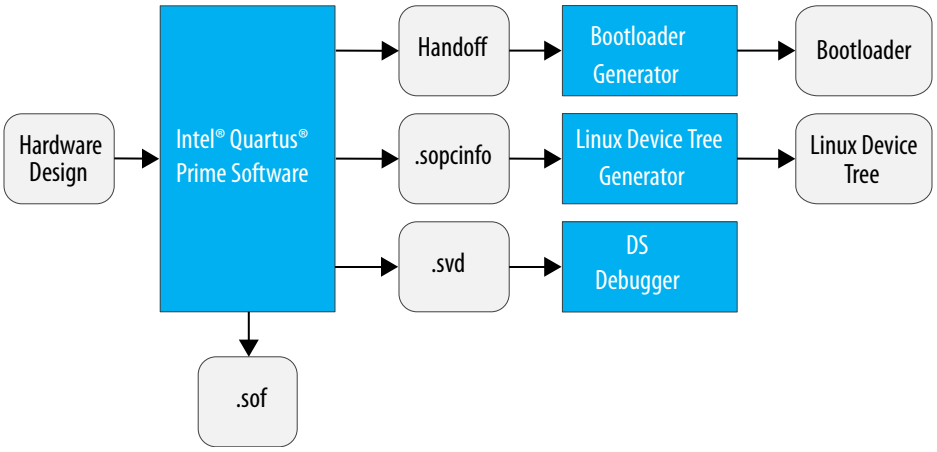
Related Information

Installing the Tools on page 11

1.5. Hardware – Software Development Flow

The Intel hardware-to-software handoff utilities allow hardware and software teams to work independently and follow their respective familiar design flows.

Figure 1. Intel Hardware-to-Software Handoff



The following handoff files are created when the hardware project is compiled:

- **Handoff** folder – contains information about how the HPS component is configured, including things like which peripherals are enabled, the pin MUXing and IOCSR settings, and memory parameters. The handoff folder is used by the second stage bootloader generator to create the preloader.

  For more information about the handoff folder, refer to the "BSP Generation Flow" section.
- **.svd** file – contains descriptions of the HPS registers and of the soft IP registers on the FPGA side implemented in the FPGA portion of the device. The **.svd** file contains register descriptions for the HPS peripheral registers and soft IP components in the FPGA portion of the SoC. This file is used by the Arm DS for Intel SoC FPGA Edition Debugger to allow you to inspect and modify these registers.
- **.sopcinfo** file – contains a description of the entire system. The SOPC Information (**.sopcinfo**) file, containing a description of the entire system, is used by the Linux device tree generator to create the device tree used by the Linux kernel.

  For more information, refer to the "Linux Device Tree Generator" section.

  Note: The soft IP register descriptions are not generated for all soft IP cores.

  Note: For Intel Stratix 10 SoC and Intel Agilex SoC, the handoff information is part of the configuration bitstream, and the bootloader has direct access to it. Because of that there is no need for a Bootloader Generator tool in this case.

Related Information

- BSP Generation Flow on page 35
- SoC Embedded Development Suite Download Page
- Linux Device Tree Generator on page 65

1.6. Introduction to the Intel SoC FPGA EDS Revision History

Document Version	Changes
2020.08.07	Added support for Intel Quartus Prime Standard Edition and Intel Quartus Prime Pro Edition software version 20.1 releases: • Replaced Mentor Graphics* Bare Metal GCC Compiler with Linaro Bare Metal Compiler (GCC) • Updated the tool versions • Removed Linux Compiler • Updated to Arm Development Studio (DS) for Intel SoC FPGA Edition
2019.12.20	Added support for Intel Quartus Prime Standard Edition version 19.1 and Intel Quartus Prime Pro Edition version 19.3 • Added Intel Agilex support. • Removed references to SoCEDSGettingStarted wiki. • Updated tool versions, • Removed the following chapters from the *Introduction to the Intel SoC FPGA Embedded Development Suite*: — *Linux Device Tree Binary* — *Intel Stratix 10 SoC Golden Hardware Reference Design* — Removed the *Getting Started Guides* section from the document. • *Linux Kernel and Driver Developer* section: Added a link to *HOWTOCreateADeviceTree* on the Rocketboards.org website.

continued...

Document Version	Changes
2019.05.16	• *Tool Versions* section: Updated the Arm Development Studio (DS) for Intel SoC FPGA Edition version. • *Intel Stratix 10 SoC SoC Golden Hardware Reference Design* section: Updated the SoC EDS Professional Edition and Intel Quartus Prime Pro Edition versions.
2018.09.24	Updated the Tool versions for Arm Compiler 6 and Arm Development Studio (DS) for Intel SoC FPGA Edition
2018.06.18	• Updated chapter to include support for Intel Stratix 10 SoC • Replaced "Second Stage Bootloader" with "Bootloader". • For the Golden Hardware Reference Designs (GHRD), added the precompiled bootloader for Intel Stratix 10 SoC • Added the "Intel Stratix 10 SoC Golden Hardware Reference Design" section
2017.05.08	• Intel FPGA rebranding • Rebranded paths and tools for the Standard and Professional versions
2016.11.07	Maintenance release
2016.05.27	Renamed the Web and Subscription Editions to align with Quartus Prime naming
2016.02.17	Maintenance release
2015.08.06	Added Intel Arria 10 SoC support

2. Installing the Tools

The installation process consists of the following steps:

- Download and install SoC EDS
- Download and install Arm DS
- Download and install the Linaro Bare Metal Toolchain

In addition to the above, the following need to be installed on Windows machines:

- Cygwin—offers a Linux-like environment on Windows, and is required by the Embedded Command Shell
- MinGW—used as a platform to build the `Newlib` library used by the Linaro Bare Metal toolchain

2.1. Installation Folders

The default installation folder for SoC EDS, referred to as <SoC FPGA EDS installation directory> throughout this document, is shown in the following table:

Version	Host OS	Folder
Standard	Linux	`~/intelFPGA/20.1/embedded`
	Windows	`C:\intelFPGA\20.1\embedded`
Professional	Linux	`~/intelFPGA_pro/20.1/embedded`
	Windows	`C:\intelFPGA_pro\20.1\embedded`

The default installation folder for the Intel Quartus Prime Programmer, referred to as <Quartus Prime installation directory> throughout this document, is shown in the following table:

Version	Host OS	Folder
Standard	Linux	`~/intelFPGA/20.1/qprogrammer`
	Windows	`C:\intelFPGA\20.1\qprogrammer`
Professional	Linux	`~/intelFPGA_pro/20.1/qprogrammer`
	Windows	`C:\intelFPGA_pro\20.1\qprogrammer`

The default installation folder for Arm DS for Intel SoC FPGA Edition is shown in the following table:

Host OS	Folder
Linux	`/opt/arm/developmentstudio-2020.0`
Windows	`c:Program Files/Arm/Development Studio 2020.0\`

2.2. Linux

2.2.1. Installing SoC EDS

1. Download and run the SoC EDS installer from the *Intel SoC FPGA Embedded Development Suite Download Center for FPGAs* webpage.

   The current Intel Quartus Prime software version is 20.1 and the filenames are:

   - For Intel Quartus Prime Pro Edition software version 20.1: `SoCEDSProSetup-20.1.0.177-linux.run`
   - For Intel Quartus Prime Standard Edition software version 20.1: `SoCEDSSetup-20.1.0.771-linux.run`
2. Run the installer to open the **Installing Intel SoC FPGA Embedded Development Suite Pro Edition** dialog box, and click **Next** to start the **Setup Wizard**.
3. In the **License Agreement** dialog box, select **I accept the agreement** and click **Next**.
4. In the **Installation Directory** dialog box, leave the default installation path (or change it if you need it somewhere else) and click **Next**.
5. In the **Select Components** dialog box, leave the **Intel Quartus Prime Programmer and Tools** selected (unless you already have full Intel Quartus Prime installed) and click **Next**.
6. In the **Ready to install** dialog box, review the summary and click **Next**. The **Installing** dialog box shows that the tools are installing, including Intel Quartus Prime Programmer, if you selected it earlier.

   When installation completes, view this message in the final dialog box:

```
Installation is complete
```

Related Information

Intel SoC FPGA Embedded Development Suite Download Center for FPGAs webpage

2.2.2. Installing Arm DS

1. Download Arm DS installer archive from the *Arm Development Studio for Intel SoC FPGA Edition Download Center for FPGAs* website. The current Intel Quartus Prime software version is 20.1 and the filename is: `DS000-BN-00001-r20p0-00rel1.tgz`.
2. Extract the archive and run the installer as root:

```
tar xf DS000-BN-00001-r20p0-00rel1.tgz
cd DS000-BN-00001-r20p0-00rel1/
sudo ./armds-2020.0.sh
```

3. The installer displays the licensing agreement. Scroll using the space bar until the end of the license, type "yes" and press **Enter**.

```
Please answer with one of: 'yes' or 'no/quit'
Do you agree to the above terms and conditions? yes
```

4. Press **Enter** to run the platform requirements check.

```
Please answer with one of: 'yes/y' or 'no/n'
Run installation platform requirement checks? [default: yes]
– Running installation platform requirement checks
Running dependency check [succeeded]
```

5. Leave the default installation path (unless you want to change it) and press **Enter**.

```
Where would you like to install to? [default: /opt/arm/
developmentstudio-2020.0]
```

6. Press **Enter** to allow the new folder to be created:

```
Please answer with one of: 'yes/y' or 'no/n'
'/opt/arm/developmentstudio-2020.0' does not exist, create? [default: yes]
– Installing to '/opt/arm/developmentstudio-2020.0' (This may take a while…)
```

7. Press **Enter** to allow the desktop menus to be added.

```
Please answer with one of: 'yes/y' or 'no/n'
Install desktop menu item additions? [default: yes]
– Installing menu entries
```

8. Press **Enter** to allow the drivers to be installed.

```
Post install stage provides the following functions:
- Installation of USB drivers for RealView ICE and DSTREAM hardware units
Please answer with one of: 'yes/y' or 'no/n'
Run post install setup scripts? [default: yes]
– Running post install setup scripts
```

9. Successful installation results in the following message:

```
---------------------------------
Installation completed successfully
---------------------------------

To start using Arm Development Studio 2020.0 either:
- Create a suite sub-shell using /opt/arm/developmentstudio-2020.0/bin/
suite_exec
- Launch GUI tools via their desktop menu entries

The Release notes for the product can be found here: file:///opt/arm/
developmentstudio-2020.0/sw/info/readme.html
```

Related Information

Arm Development Studio for Intel SoC FPGA Edition Download Center for FPGAs

2.2.3. Installing Linaro Bare Metal Toolchain

This section shows how to install the Linaro Bare Metal toolchain for Cortex-A9, useful for compiling Bare Metal programs for Arria V SoC, Cyclone V SoC, and Intel Arria 10 SoC devices.

1. Start an Embedded Command Shell

```
~/intelFPGA_pro/20.1/embedded/embedded_command_shell.sh
```

2. Change current folder to linaro and run the installation script

```
cd $SOCEDS_DEST_ROOT/host_tools/linaro/
./install_linaro.sh
```

3. Upon successful completion, the following are installed in the `$SOCEDS_DEST_ROOT/host_tools/linaro/` folder:
   - gcc–GCC Compiler
   - newlib–Newlib library

2.3. Windows

2.3.1. Installing SoC EDS

1. Download and run the SoC EDS installer from the *Intel SoC FPGA Embedded Development Suite Download Center for FPGAs* webpage.

   The current Intel Quartus Prime software version is 20.1 and the filenames are:
   - For Intel Quartus Prime Pro Edition software version 20.1: `SoCEDSProSetup-20.1.0.177-windows.exe`
   - For Intel Quartus Prime Standard Edition software version 20.1: `SoCEDSSetup-20.1.0.771-windows.exe`
2. Run the installer to open the **Installing Intel SoC FPGA Embedded Development Suite Pro Edition** dialog box, and click **Next** to start the **Setup Wizard**.
3. In the **License Agreement** dialog box, select **I accept the agreement** and click **Next**.
4. In the **Installation Directory** dialog box, leave the default installation path (or change it if you need it somewhere else) and click **Next**.
5. In the **Select Components** dialog box, leave the **Intel Quartus Prime Programmer and Tools** selected (unless you already have full Intel Quartus Prime installed) and click **Next**.
6. In the **Ready to install** dialog box, review the summary and click **Next**. The **Installing** dialog box shows that the tools are installing, including Intel Quartus Prime Programmer, if you selected it earlier.

   The next dialog box indicates that the Intel Quartus Prime installation is complete.
7. In this dialog box, leave the driver installation options selected and click **Finish**.
8. Accept all defaults to install the Intel FPGA Download Cable II drivers.
9. In the **Welcome to the Device Driver Installation Wizard** dialog box, click **Next**.
10. In the **Completing the Device Driver Installation Wizard** dialog box, click **Finish**.
11. Accept all defaults to install FTDI CDM drivers (used to connect to USB serial ports on Development Kits).
12. In the **Welcome to the Device Driver Installation Wizard** dialog box, click **Next**.
13. In the **License Agreement** dialog box, select **I accept this agreement** and click **Next**.
14. In the **Completing the Device Driver Installation Wizard**, view this message: `The drivers were successfully installed on this computer.` and click **Finish**.

Related Information

Intel SoC FPGA Embedded Development Suite Download Center for FPGAs webpage

2.3.2. Installing Arm DS

1. Download Arm DS installer archive from the *Arm Development Studio for Intel SoC FPGA Edition Download Center for FPGAs* website. The current Intel Quartus Prime software version is 20.0 and the filename is: `DS000-BN-00000-r20p0-00rel1.zip`.
2. Extract the archive and run the `DS000-BN-00000-r20p0-00rel1\armds-2020.0.exe` installer.
3. In the **Welcome to the Arm Development Studio 2020.0 Setup Wizard** dialog box, click **Next**.
4. In the **End-User License Agreement** dialog box, select **I accept the terms in the License Agreement**, and click **Next**.
5. In the **Custom Setup** dialog box, leave the default installation location (or change it if desired) and click **Next**.
6. In the **Ready to install Arm Development Studio 2020.0** dialog box, click **Install**.
   The **Installing Arm Development Studio 2020.0** dialog box shows that Arm Development Studio 2020.0 is installing.
7. In the **Welcome to the Arm Development Studio 2020.0 Driver Installation Wizard** dialog box, click **Next**.
8. In the **Windows Security** dialog box, accept all default options for driver installation and click **Install**.
   The **Arm DS 2020.0 Driver Installation Wizard** dialog box shows: The `drivers are now installing...`
9. In the **You have completed the Arm DS 2020.0 Driver Installation Wizard** dialog box, click **Finish**.
10. In the **Installation of Arm Development Studio 2020.0 is Complete** dialog box, click **Finish** to complete installation.

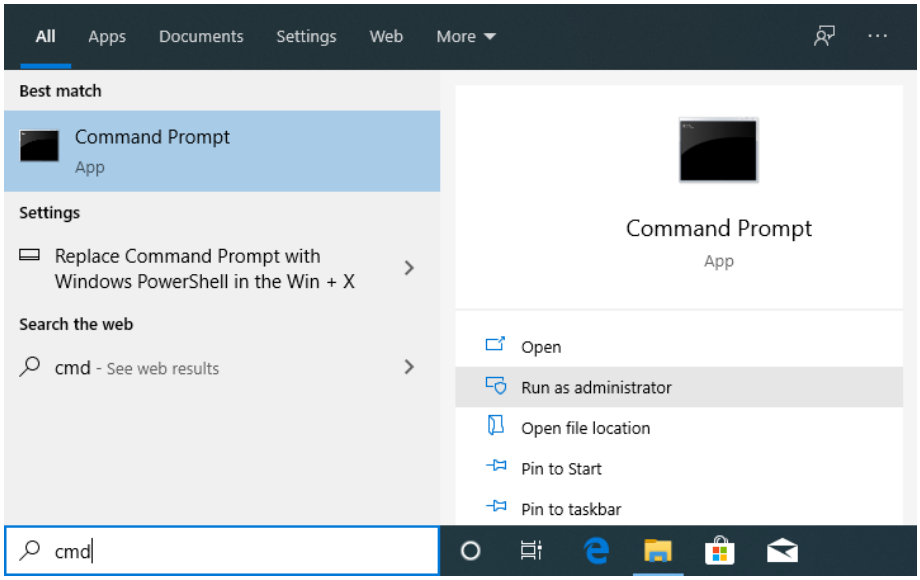
Related Information

Arm Development Studio for Intel SoC FPGA Edition Download Center for FPGAs

2.3.3. Installing Cygwin

1. Go to the *Cygwin* website and download the following installer to your computer: https://cygwin.com/setup-x86_64.exe.
2. Start a **Command Prompt** as an administrator.

Figure 2. Cygwin Command Prompt



3. Change the current directory to `cygwin_setup`:
   - For Intel Quartus Prime Pro Edition software version 20.1:

     ```
     cd c:\intelFPGA_pro\20.1\embedded\cygwin_setup\
     ```

   - For Intel Quartus Prime Standard Edition software version 20.1:

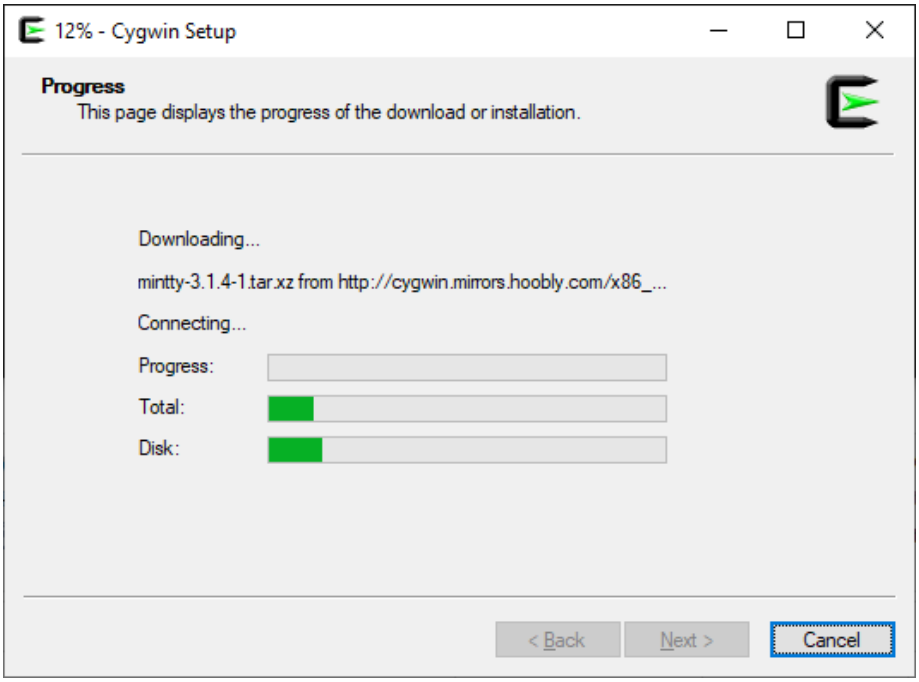
     ```
     cd c:\intelFPGA\20.1\embedded\cygwin_setup\
     ```

4. Run the `soceds-cygwin-setup.bat` executable, passing it the full path to where you downloaded the `setup-x86_64.exe` installer (in the example below, it is downloaded into your `Downloads` folder:

   ```
   soceds-cygwin-setup.bat %USERPROFILE%\Downloads\setup-x86_64.exe
   ```

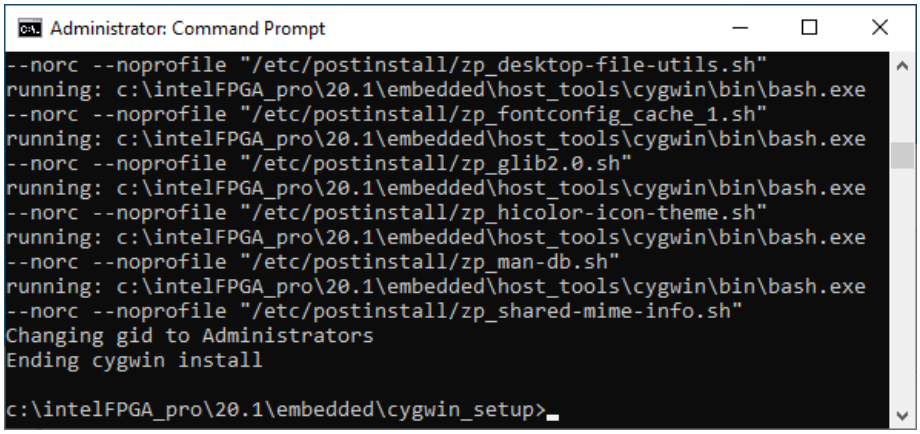
   The installer application starts:

Figure 3. Cygwin Setup



The installer application completes:

Figure 4. Administrator: Command Prompt



Related Information

Cygwin webpage

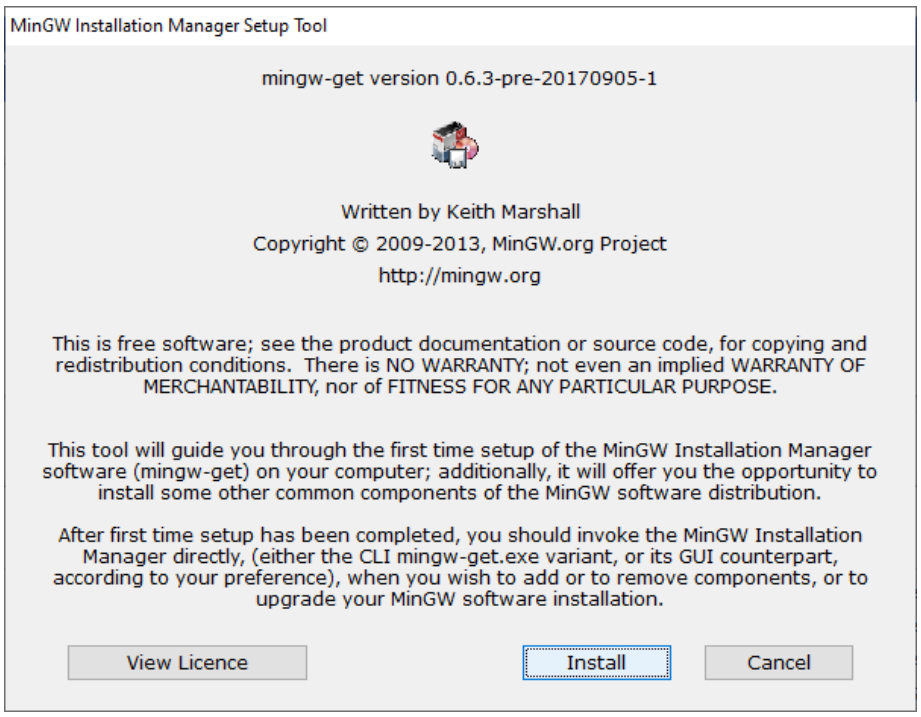
2.3.4. Installing MingGW

1. Go to the *MinGW Getting Started* website, download the following installer to your computer: https://osdn.net/projects/mingw/downloads/68260/mingw-get-setup.exe and run as an administrator.

The installer application starts.

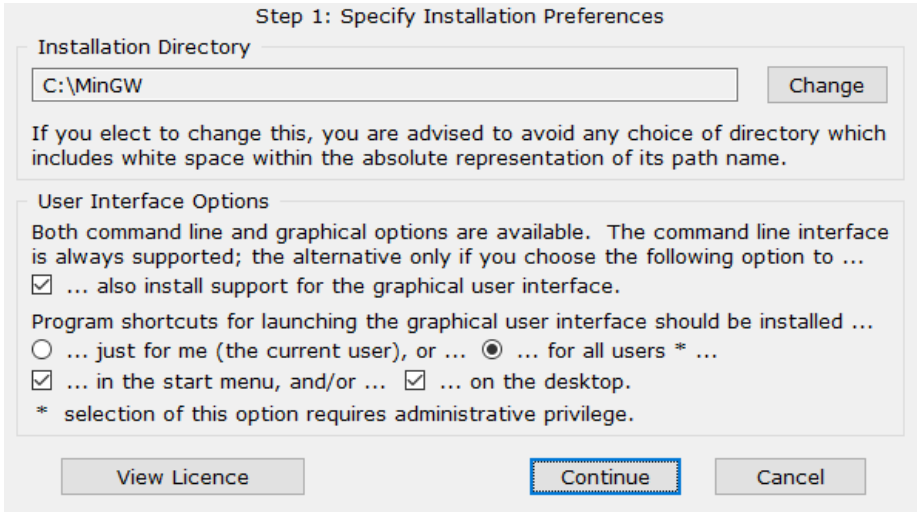
2. In the **MinGW Installation Manager Setup Tool** dialog box, click **Install**.

Figure 5. MinGW Installation Manager Setup Tool



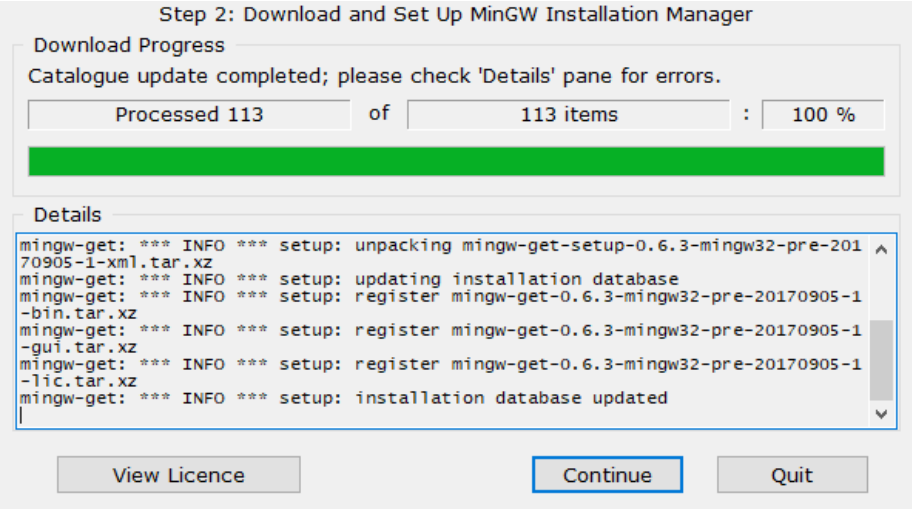
3. Leave the default settings, then click **Continue**.

Figure 6. Step 1: Specify Installation Preferences



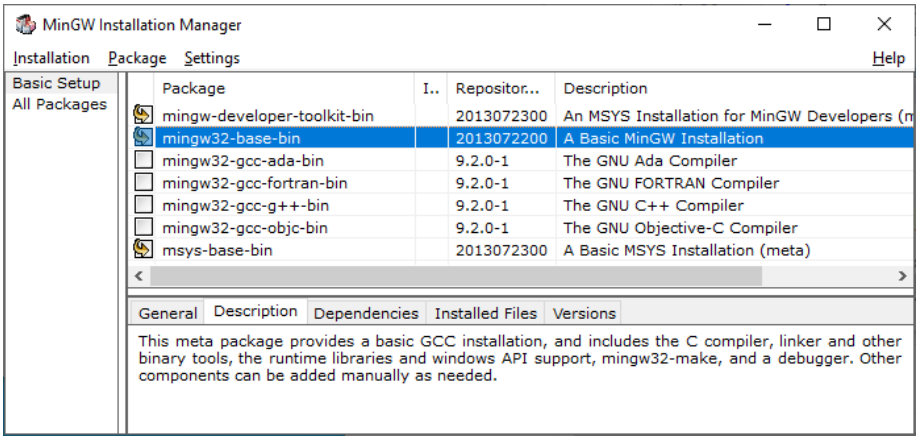
4. Click **Continue** again to proceed.

Figure 7. Step 2: Download and Set Up MinGW Installation Manager



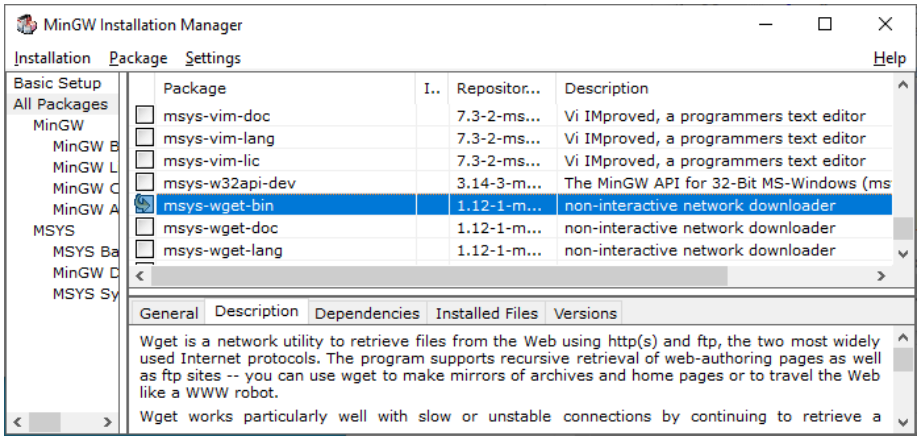
5. In the **Basic Setup** view, click on **mingw-developer-toolkit-bin** and **mingw32-base-bin** and **msys-base-bin** and select **Mark for Installation**.

Figure 8. MinGW Installation Manager: Basic Setup



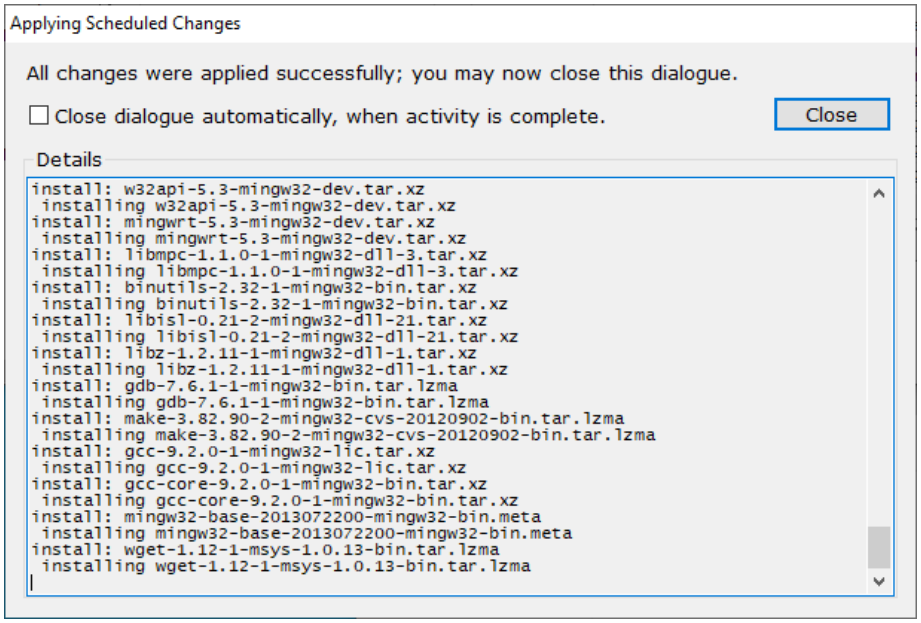
6. In the **All Packages** view, click on **msys-wget-bin** and select **Mark for Installation**.

Figure 9. MinGW Installation Manager: All Packages



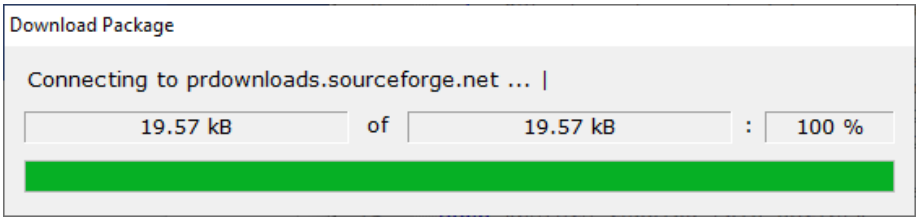
7. Select **Installation ➤ Apply Changes** from the top menu.
8. Click **Apply** to proceed.

Figure 10. Schedule of Pending Actions



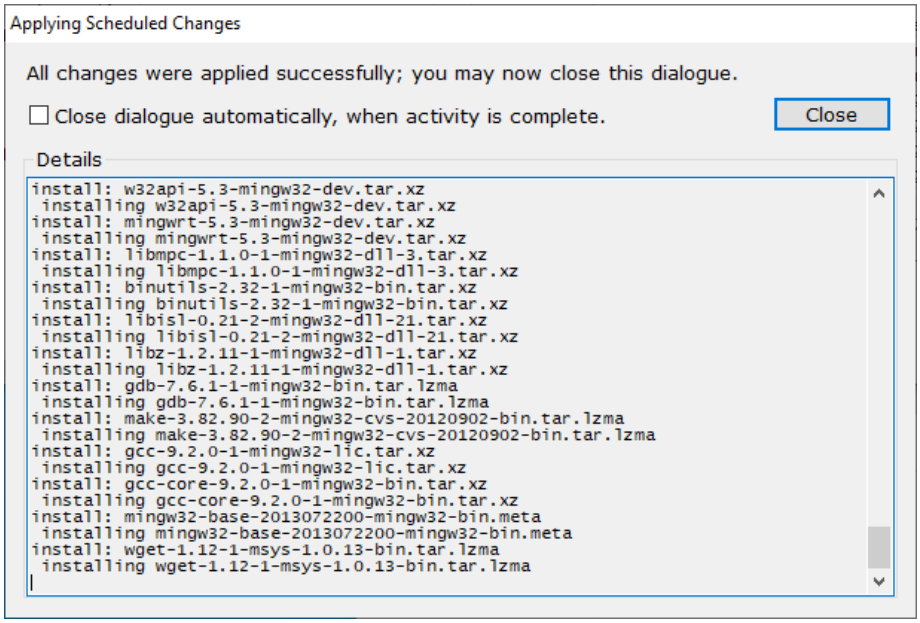
Installer downloads all necessary packages.

Figure 11. Download Package



9. After the installer applied all changes, click **Close**.

Figure 12. Applying Scheduled Changes



10. Select **Installation ➤ Quit** from the top menu.

Related Information

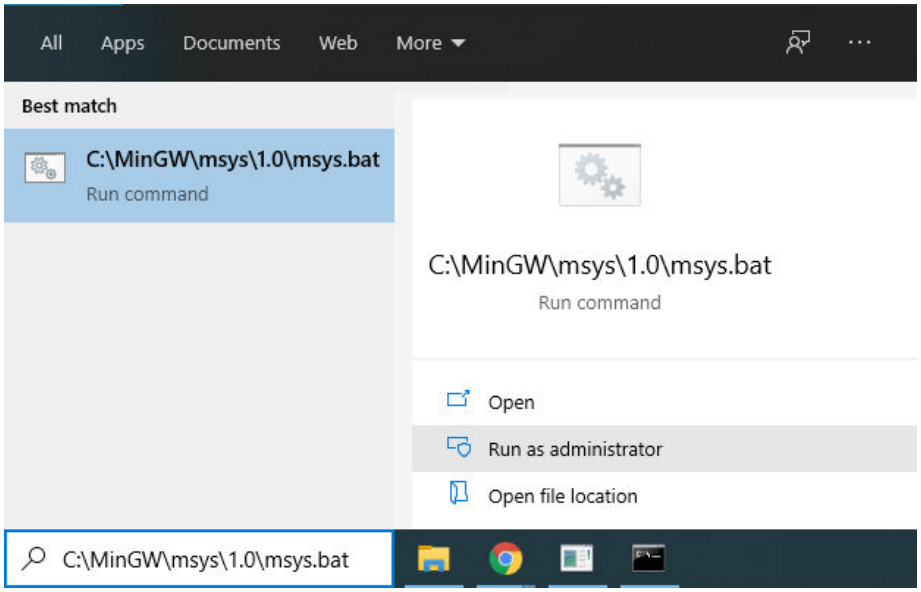
MinGW Getting Started webpage

2.3.5. Installing Linaro Bare Metal Toolchain

This section shows how to install the Linaro Bare Metal toolchain for Cortex-A9, useful for compiling Bare Metal programs for Arria V SoC, Cyclone V SoC, and Intel Arria 10 SoC.

1. Run `C:\MinGW\msys\1.0\msys.bat` as an administrator.

Figure 13. Run Command



2. In the Msys console, go to the linaro folder:
   - For Intel Quartus Prime Pro Edition software version 20.1:
     ```
     cd c:/intelFPGA_pro/20.1/embedded/host_tools/linaro
     ```
   - For Intel Quartus Prime Standard Edition software version 20.1:
     ```
     cd c:/intelFPGA/20.1/embedded/host_tools/linaro
     ```
3. Run the installer:
   ```
   ./install_linaro.sh
   ```
4. The Linaro toolchain and the `newlib` library are downloaded and compiled.
5. Upon successful completion, the following are installed in the `linaro` folder:
   - gcc–GNU Compiler
   - newlib–Newlib library

2.4. Installing the Intel SoC FPGA EDS Document Revision History

Document Version	Changes
2020.08.07	Added support for Intel Quartus Prime Standard Edition and Intel Quartus Prime Pro Edition software version 20.1 releases:

continued...

Document Version	Changes
	• Added the following new sections: — *Installing Linaro Bare Metal Toolchain* for both Linux and Windows — *Installing Cygwin* for Windows — *Installing MingGW* for Windows • The following sections were updated for Windows and duplicated for Linux: — *Installing SoC EDS* — *Installing Arm DS*
2019.12.20	Supported with Intel Quartus Prime Standard Edition version 19.1 and Intel Quartus Prime Pro Edition version 19.3 • Updated steps in the *Installing Cygwin on Windows** section.
2019.05.16	• *Installation Folders* section: Updated the path names for the 19.1 release • Created two new sections called *Installing Cygwin on Windows* and *Installing the SoC EDS on Windows*.
2018.09.24	Updated the path names for the 18.1 release.
2018.06.18	Maintenance release
2018.05.07	Maintenance release
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	Updated the installation path to 15.1
2015.08.06	Added Intel Arria 10 SoC support

3. Running the Tools

The tools provided with SoC EDS need to be run from an Embedded Command Shell. Additionally, the Arm DS tools need to be run from an Arm DS shell.

3.1. Linux

Information about starting the embedded command shell, Arm DS shell, and Arm DS Eclipse for Linux.

Starting Embedded Command Shell

For Intel Quartus Prime Pro Edition software version 20.1:

```
~/intelFPGA_pro/20.1/embedded/embedded_command_shell.sh
```

For Intel Quartus Prime Standard Edition software version 20.1:

```
~/intelFPGA/20.1/embedded/embedded_command_shell.sh
```

Starting Arm DS Shell

1. Start an Embedded Command Shell.

For Intel Quartus Prime Pro Edition software version 20.1:

```
~/intelFPGA_pro/20.1/embedded/embedded_command_shell.sh
```

For Intel Quartus Prime Standard Edition software version 20.1:

```
~/intelFPGA/20.1/embedded/embedded_command_shell.sh
```

2a. If Arm Compiler 5 is desired:

```
/opt/arm/developmentstudio-2020.0/bin/suite_exec -t "Arm Compiler 5" bash
```

2b. If Arm Compiler 6 is desired (only needed by the Intel Quartus Prime Pro Edition software version 20.1

```
/opt/arm/developmentstudio-2020.0/bin/suite_exec -t "Arm Compiler 6" bash
```

Starting Arm DS Eclipse

1. Start an Embedded Command Shell.
2. Start an Arm DS Shell with the desired compiler selected.
3. Run the `armds_ide` command.

   Helpful parameters for `armds_ide` command are:

- `-nosplash`: speeds up loading by not displaying the splash screen
- `-data <workspace-name>`: allows the workspace folder to be specified

3.2. Windows

Information about starting the embedded command shell, Arm DS shell, and Arm DS Eclipse for Windows.

Starting Embedded Command Shell

You can start an Embedded Command Shell through navigation:

For Intel Quartus Prime Pro Edition software version 20.1: **Start menu ➤ Intel FPGA 20.1 Pro Edition ➤ SoC EDS Command Shell**

For Intel Quartus Prime Standard Edition software version 20.1: **Start menu ➤ Intel FPGA 20.1 ➤ SoC EDS Command Shell**

Starting Arm DS Shell

1. Start Embedded Command Shell.
2. Run the `cmdsuite` program.

```
/cygdrive/c/Program\ Files/Arm/Development\ Studio\ 2020.0/bin/cmdsuite.exe
```

3. Run the `bash` program to get back to Embedded Command Shell colors.
4. If needed, run the `select_toolchain` command to select between **Arm Compiler 5** and **Arm Compiler 6**.

Starting Arm DS Eclipse

1. Start an Embedded Command Shell.
2. Start an Arm DS Shell with the desired compiler selected.
3. Run the `armds_ide` command.

   Helpful parameters for `armds_ide` command are:

   - `-nosplash`: speeds up loading by not displaying the splash screen
   - `-data <workspace-name>`: allows the workspace folder to be specified

3.3. Running the Tools Revision History

Document Version	Changes
2020.08.07	Maintenance release
2019.12.20	Added support for Intel Quartus Prime Standard Edition and Intel Quartus Prime Pro Edition software version 20.1 releases: • Replaces the *Embedded Command Shell* section

4. SoC EDS Licensing

The only SoC EDS component which requires a license is the Arm DS for Intel SoC FPGA Edition. Two license options are available:

- Arm DS for Intel SoC FPGA Edition License
- 30-day Evaluation of Arm DS for Intel SoC FPGA Edition License

For information about Arm DS for Intel SoC FPGA Edition features, editions, and licensing, refer to *Arm DS for Intel SoC FPGA Edition Toolkit*.

Related Information

Arm DS for Intel SoC FPGA Edition Toolkit

4.1. Getting the License

Follow the steps below to get the License:

- For **Arm DS for Intel SoC FPGA Edition License**—If you have purchased a SoC development kit or a stand-alone license for DS for Intel SoC FPGA Edition AE, then you have already received an Arm license serial number. This is a 15-digit alphanumeric string with two dashes in between. Use this serial number to activate your license in DS for Intel SoC FPGA Edition, as shown in the *Activating the License* section.
- For **30-Day Evaluation of Arm DS for Intel SoC FPGA Edition License**—If you want to evaluate the Arm DS for Intel SoC FPGA Edition, you can get a 30-Day Evaluation activation code from the *Arm Development Studio for Intel SoC FPGA Devices* webpage by clicking on the **Evaluate/Activate** button.

Related Information

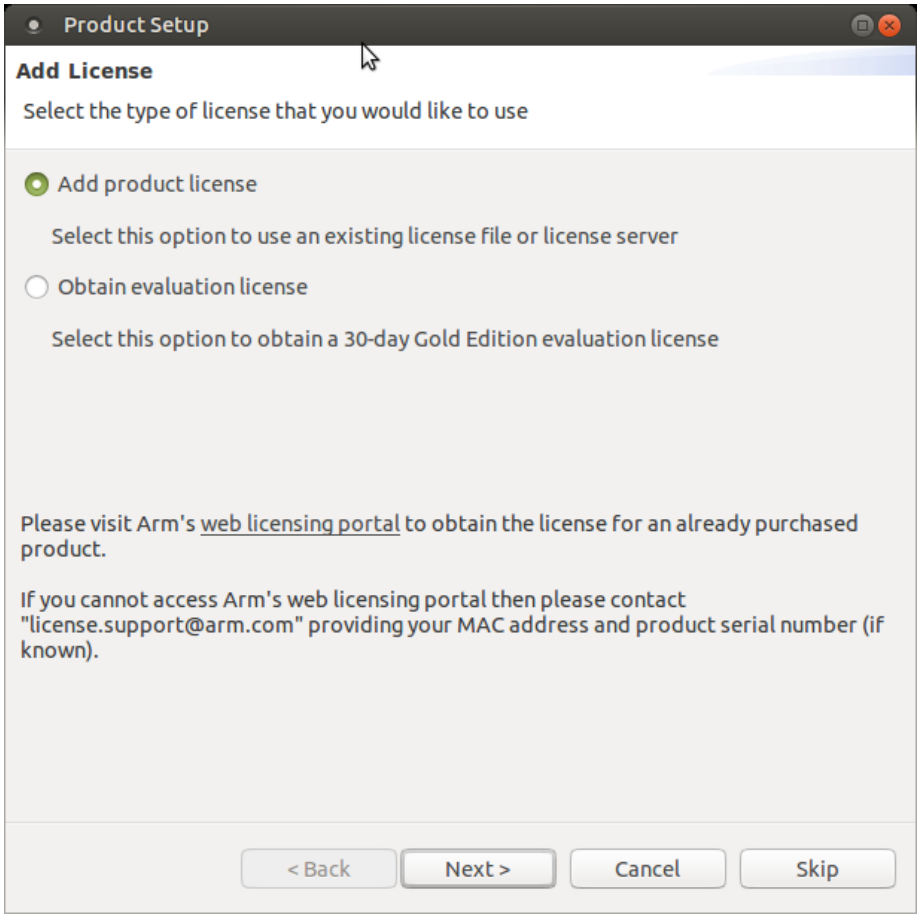
Arm Development Studio for Intel SoC FPGA Devices webpage

4.2. Activating the License

This section presents the steps required for activating and applying your license.

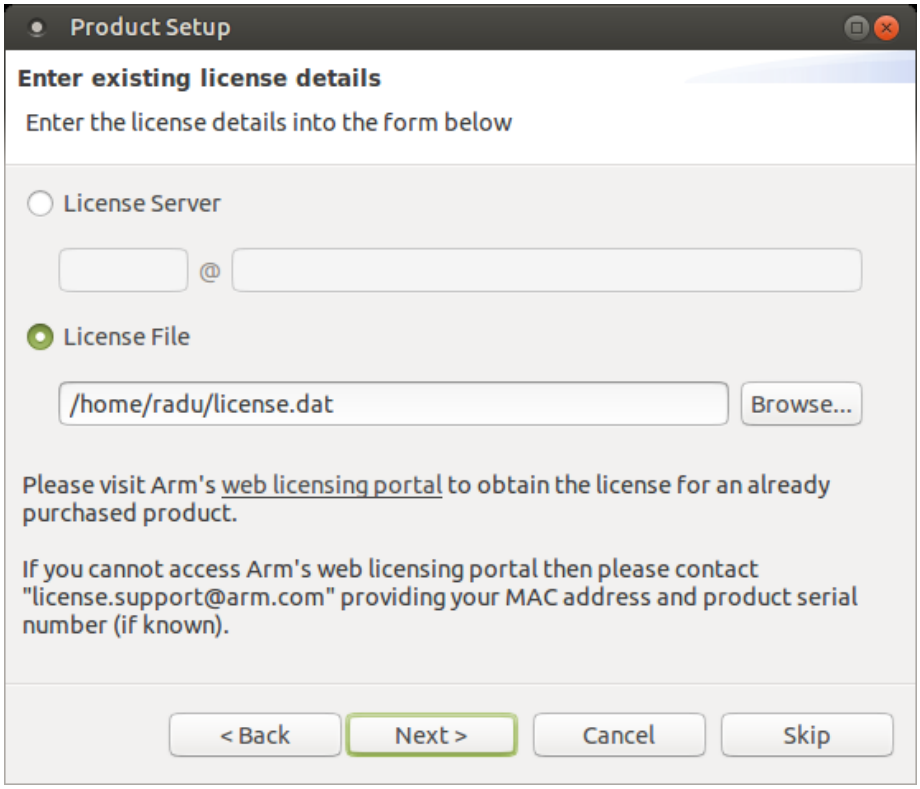
1. Go to *Arm Licensing Portal* to generate a license file based on the license code you received with your Development Kit.
2. Start the Arm DS for Intel SoC FPGA Edition IDE.
3. The first time you run this, the tool asks you to add a product license. Select **Add product license** and click **Next**.

Figure 14. Add License



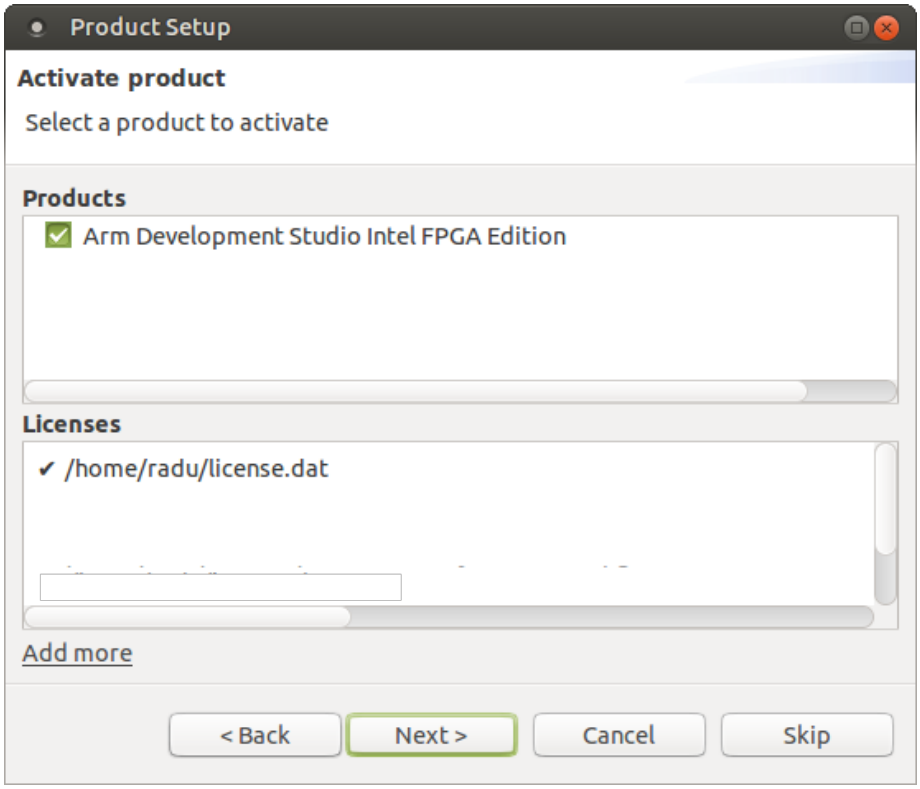
4. Select **License File** and browse to the place where you stored your generated license. Click **Next**.

Figure 15. Enter Existing License Details



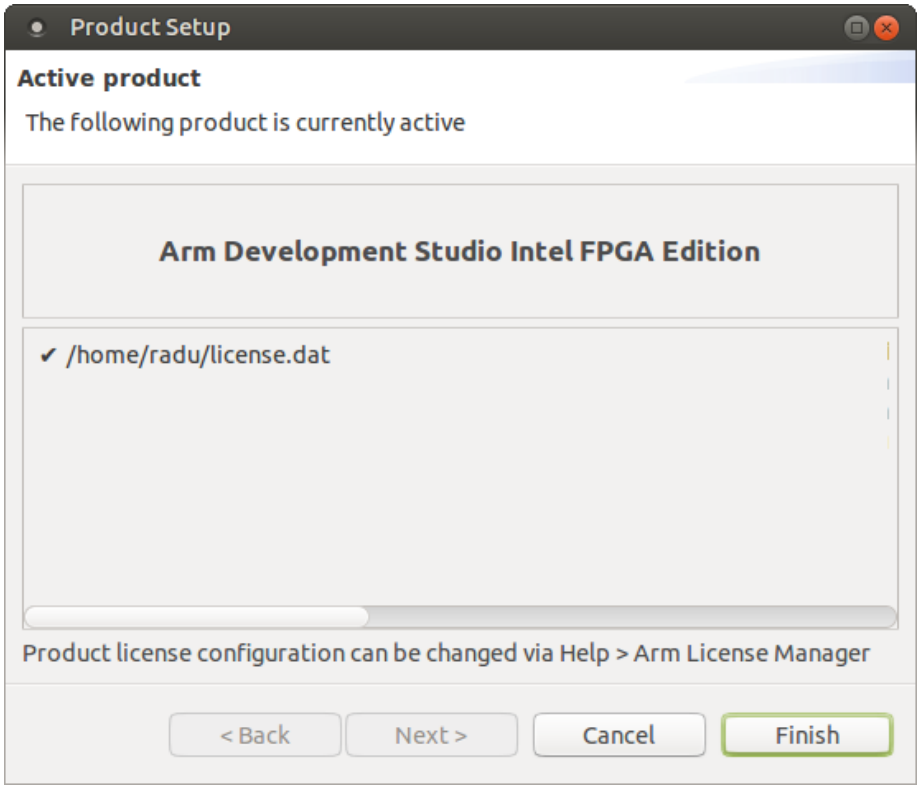
5. The **Arm DS for Intel SoC FPGA Edition** license is automatically selected. Click **Next** to activate it.

Figure 16. Activate Product



6. The License is activated. Click **Finish** to complete the process.

Figure 17. Active Product



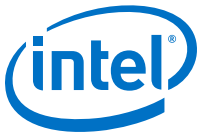
Related Information

- Getting the License on page 26
- Arm Software Licensing webpage

4.3. Intel SoC FPGA EDS Licensing Revision History

Document Version	Changes
2020.08.07	Maintenance release
2020.07.31	Added support for Intel Quartus Prime Standard Edition and Intel Quartus Prime Pro Edition software version 20.1 releases: • Removed information about Arm DS for Intel SoC FPGA Edition Community Edition License.
2019.12.20	Added support for Intel Quartus Prime Standard Edition version 19.1 and Intel Quartus Prime Pro Edition version 19.3 • Maintenance release
2019.05.16	Maintenance release
2018.09.24	Maintenance release
2018.06.18	Maintenance release
2017.05.08	• Intel FPGA rebranding • Rebranded paths and tools for the Standard and Professional versions

continued...



Document Version	Changes
2016.11.07	Maintenance release
2016.05.27	Renamed the Web and Subscription Editions to align with Quartus Prime naming
2016.02.17	Maintenance release
2015.08.06	Added Intel Arria 10 SoC support

5. Arm Development Studio for Intel SoC FPGA Edition

The Arm DS for Intel SoC FPGA Edition is a device-specific exclusive offering from Intel and is a powerful Eclipse-based comprehensive Integrated Development Environment (IDE).

For the current version, refer to Table 1 on page 4.

Some of the most important provided features are:

- File editing, supporting syntax highlighting and source code indexing
- Build support, based on makefiles
- Bare Metal debugging
- Linux application debugging
- Linux kernel and driver debugging
- Multicore debugging
- Access to HPS peripheral registers
- Access to FPGA soft IP peripheral registers
- Tracing of program execution through Program Trace Macrocells (PTM)
- Tracing of system events through System Trace Macrocells (STM)
- Cross-triggering between HPS and FPGA
- Connecting to the target using Intel FPGA Download Cable II

The Arm Development Studio for Intel SoC FPGA Edition is a complex tool with many features and options. It is the only SoC EDS component that requires a license.

Note: Arm DS is downloaded and installed separately.

For information about the licensing options and how to obtain them, refer to the *Intel SoC FPGA EDS Licensing* section.

For examples of using Arm DS for Intel SoC FPGA Edition, refer to the *SoC EDS and Arm Development Studio* webpage on RocketBoards.

You can access the Arm Development Studio for Intel SoC FPGA Edition reference material from Eclipse, by navigating to **Help ➤ Help Contents ➤ Arm Developer Studio Documentation** or on the Arm website.

You can also access additional material from Eclipse, by navigating to **Help ➤ Tutorials and Videos**.

Related Information

- Tool Versions on page 4

- SoC EDS Licensing on page 26
- SoC EDS and Arm Development Studio webpage on RocketBoards
  For more information about Arm DS for Intel SoC FPGA Edition
- Running the Tools on page 24
- Online Arm Development Studio for Intel SoC FPGA Edition Documentation
  For more information, refer to the Arm Development Studio for Intel SoC FPGA Edition reference documentation located on the ARM website.
- Arm Development Studio for Intel SoC FPGA Devices

5.1. Arm DS for Intel SoC FPGA Edition Revision History

Document Version	Changes
2020.08.07	Added support for Intel Quartus Prime Standard Edition and Intel Quartus Prime Pro Edition software version 20.1 releases: • Moved *Starting Eclipse Arm DS for Intel SoC FPGA Edition* to the *Running the Tools* section. • Removed *Bare Metal Project Management* and *Debugging* sections.
2019.12.20	Added support for Intel Quartus Prime Standard Edition version 19.1 and Intel Quartus Prime Pro Edition version 19.3 • Added *Running the Tools* section to replace the *Embedded Command Shell*
2019.12.20	Added support for Intel Quartus Prime Standard Edition version 19.1 and Intel Quartus Prime Pro Edition version 19.3 • Removed references to the *Getting Started Guides* section. • *Accessing Debug Configurations*: Added a link to the *Debugging the HPS Boot Loader Using the Arm DS for Intel SoC FPGA Edition* section in the *Intel Stratix 10 SoC FPGA Boot User Guide*
2019.05.16	Maintenance release
2018.09.24	Maintenance release
2018.06.18	Maintenance release
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	Updated default size in ETF Settings
2015.08.06	Added Intel Arria 10 SoC support

6. Boot Tools User Guide

6.1. Introduction

The boot flow for all the Intel SoC devices includes a bootloader. The bootloader has two stages called First Stage Bootloader (FSBL) and Second Stage Bootloader (SSBL). The FSBL is also known as Preloader.

The FSBL needs to fit into the on-chip RAM (OCRAM) and has limited functionality. It performs the tasks of initializing the HPS, bringing up the DDRAM and then loading and executing the next stage in the boot process. The next stage can be the SSBL, the end application, or an Operating System (OS).

The SSBL resides in DDRAM and therefore can have a larger size. Besides the ability to load and execute the next stage in the booting process, it typically offers a lot more functionality such as filesystem support, networking services, and command line interface.

Figure 18. Cyclone V SoC, Arria V SoC, and Intel Arria 10 SoC Typical Boot Flow

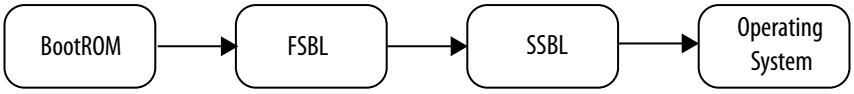
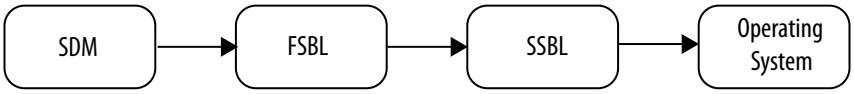


Figure 19. Intel Stratix 10 SoC and Intel Agilex Typical Boot Flow



On Cyclone V SoC, Arria V SoC and Intel Arria 10 SoC devices, the very first code run by HPS is the BootROM. For Intel Stratix 10 SoC and Intel Agilex devices, the very first code run by HPS is the FSBL which is loaded by Secure Device Manager (SDM).

This chapter presents the tools that are used to enable the bootloader management:

- **BSP Generator** – enables you to get the Intel Quartus Prime handoff information and use it for the initial configuration of the bootloader for Cyclone V SoC, Arria V SoC and Intel Arria 10 SoC devices.
- **Bootloader Image Tool (`mkpimage`)** — enables you to add the BootROM-required header on top of the bootloader for Cyclone V SoC, Arria V SoC and Intel Arria 10 SoC devices.
- **U-Boot Image Tool (mkimage)**— enables you to add the bootloader-required header on top of the files loaded by bootloader.

This chapter also includes instructions on how to build the bootloader for Intel Stratix 10 SoC and Intel Agilex devices. For instructions on how to build the bootloader for Cyclone V SoC, Arria V SoC and Intel Arria 10 SoC, please visit "Building Bootloader" on RocketBoards.org.

6.2. BSP Generator

The BSP Generator is used for the initial configuration of the Bootloader based on the Intel Quartus Prime handoff information for the following Intel SoC devices: Cyclone V SoC, Arria V SoC and Intel Arria 10 SoC.

Note: The BSP Generator is not used for the Intel Stratix 10 SoC and Intel Agilex devices. For Intel Stratix 10 SoC and Intel Agilex devices, the settings are available in the Intel Quartus Prime project, and passed along to the Bootloader by the SDM.

The BSP Generator allows you to create a new BSP with the default settings, based on the handoff information from Intel Quartus Prime.

The generated BSP includes a makefile, which outputs a message pointing you to visit "Building Bootloader" on RocketBoards.org to get details on how to build the bootloader.

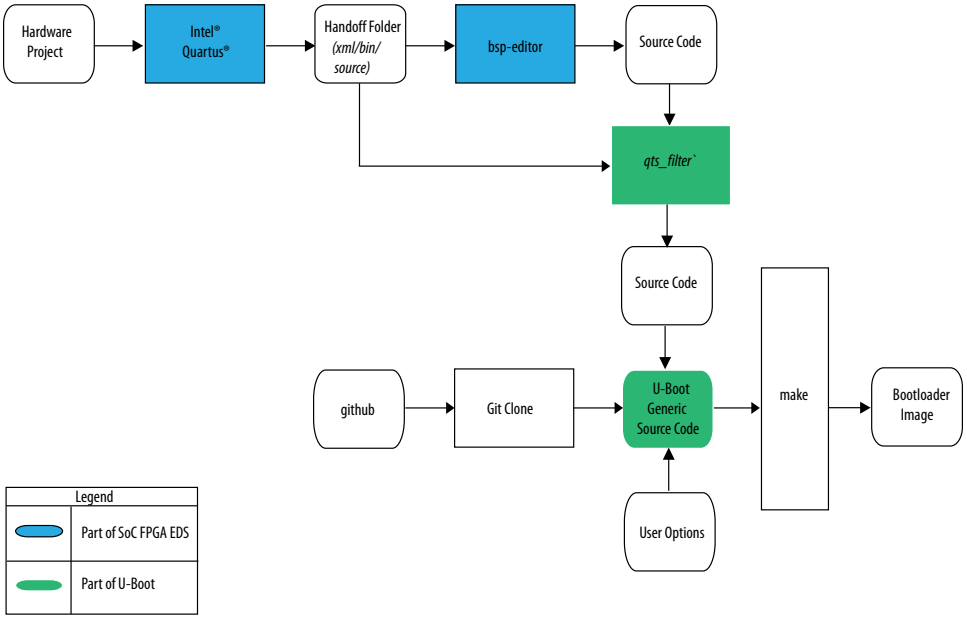
6.2.1. BSP Generation Flow

This section presents the BSP generation flow for the Cyclone V SoC, Arria V SoC, and Intel Arria 10 SoC bootloader. While the flows are similar, there are some important differences.

6.2.1.1. Cyclone V SoC and Arria V SoC Flow

For Cyclone V SoC and Arria V SoC, the BSP Generator is required to be run first, to convert some of the handoff files into C source code. After that, you need to clone the U-Boot source code from github. Then you use an U-Boot script called `qts-filter` to extract the information from the handoff folder, and the set of files created by the BSP Generator, and put them in the U-Boot source code folder. Then you use the U-Boot `makefile` to build the SPL image. The SPL image can then be downloaded to a Flash device or FPGA RAM to be used for booting HPS.

Figure 20. Arria V SoC/ Cyclone V SoC BSP Generator Flow



The hardware handoff information contains various settings that you entered when creating the hardware design in Platform Designer and Intel Quartus Prime Standard Edition. These include the following:

- Pin-muxing for the HPS dedicated pins
- I/O settings for the HPS dedicated pins:
  - Voltage
  - Slew rate
  - Pull up/ down
- Configuration of the bridges between HPS and FPGA
- Clock tree settings:
  - PLL settings
  - Clock divider settings
  - Clock gating settings
- DDR settings:
  - Technology
  - Width
  - Speed

The handoff settings are output from the Intel Quartus Prime Standard Edition compilation and are located in the **<quartus project directory>/ hps_isw_handoff/<hps entity name>** directory (where <**hps entity name** > is the HPS component name in Platform Designer).

You must update the hardware handoff files and regenerate the BSP each time a hardware change impacts the HPS, such as after pin multiplexing or pin assignment changes.

For complete instructions on how to build the bootloader for Cyclone V SoC and Arria V SoC, please visit "Building Bootloader" on RocketBoards.org.

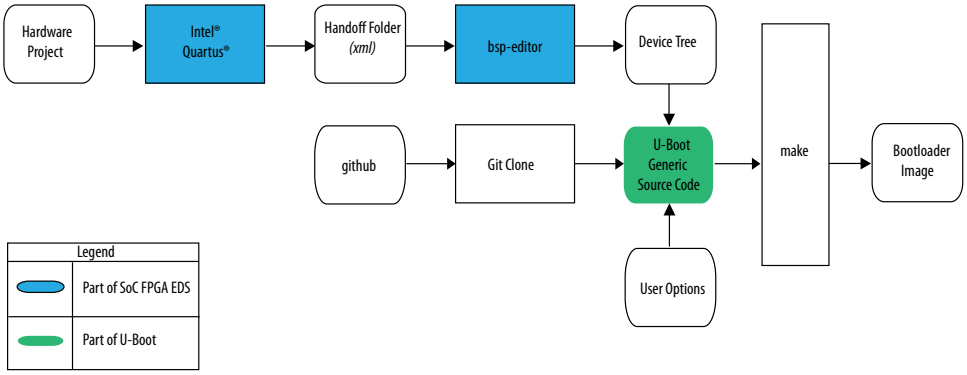
6.2.1.2. Intel Arria 10 SoC Flow

For Intel Arria 10 SoC, you run the BSP Generator first, and it creates a Device Tree with default settings based on the Intel Quartus Prime handoff information. There is no source code generated, because all customization is encapsulated in the Bootloader Device Tree.

Note: The device tree is board specific and may need to be edited to reflect customer designs

The next step is to clone the U-Boot source code from GitHub, and copy the generated device tree into U-Boot source code. For complete instructions, go to the Building Bootloader web page on RocketBoards.org. Then the U-Boot is compiled using the `make` utility and it creates the combined bootloader image, which contains both the bootloader executable and the bootloader device tree. You can download the combined image to a Flash device or FPGA RAM to use for booting the HPS.

Figure 21. Intel Arria 10 SoC BSP Generator Flow



The hardware handoff information contains various settings that you entered when creating the hardware design in Platform Designer.. These include the following:

- Pin-muxing for the HPS dedicated pins
- I/O settings for the HPS dedicated pins:
  - Voltage
  - Slew rate
  - Pull up/ down
- Pin-muxing for the shared pins
- Configuration of the bridges between HPS and FPGA
- Clock tree settings:
  - PLL settings
  - Clock divider settings
  - Clock gating settings

The handoff settings are output from the Intel Quartus Prime Standard Edition compilation and are located in the **<quartus project directory>/hps_isw_handoff** directory.

The user must run the BSP Generator and re-generate the Bootloader device tree each time a hardware change results in a change of the above parameters.

However, you does not have to always recompile the Bootloader whenever a hardware setting is changed. The Bootloader needs to be recompiled only when changing the boot source.

6.2.2. BSP Generator Graphical User Interface

The BSP generator GUI is deprecated, and you do not need to use it anymore. The settings are now customized directly in the bootloader files, and not from the BSP Generator. Only the `bsp-create-settings` command line tool needs to be ran, but without customized options, as they are not taken into consideration.

6.2.3. BSP Generator Command Line Interface

The BSP command-line tools can be invoked from the embedded command shell, and provide the following feature:

bsp-create-settings—a tool that converts the Intel Quartus Prime handoff information to source code for Cyclone V SoC and Arria V SoC, and to a Device Tree for Intel Arria 10 SoC.

Note: The following tools are depreciated and not used anymore:

- **bsp-update-settings**
- **bsp-query-settings**
- **bsp-generate-files**

6.2.3.1. bsp-create-settings

The **bsp-create-settings** tool converts the Intel Quartus Prime handoff information to source code for Cyclone V SoC and Arria V SoC, and to a Device Tree for Intel Arria 10 SoC.

Table 3. User Parameters: bsp-create-settings

Option	Required	Description
`--type <bsp-type>`	Yes	This option specifies the type of BSP. Allowed values are: • "spl" for Cyclone V SoC/Arria V SoC Preloader • "uboot" and "uefi" for Intel Arria 10 SoC Bootloader
`--settings <filename>`	Yes	This option specifies the path to a BSP settings file. The file is created with default settings.Intel recommends that you name the BSP settings file `settings.bsp`.
`--preloader-settings-dir <directory>`	Yes	This option specifies the path to the hardware handoff files.
`--bsp-dir <directory>`	Yes	This option specifies the path where the BSP files are generated. When specified, `bsp-create-settings` generates the files after the settings file has been created.Intel recommends that you always specify this parameter with `bsp-create-settings`.
`--set <name> <value>`	No	This option sets the BSP setting <name> to the value <value>. Multiple instances of this option can be used with the same command. Refer to ***BSP Settings*** for a complete list of available setting names and descriptions.

6.2.4. BSP Files and Folders

The files and folders created with the BSP Generator are stored in the location you specified in **BSP target directory** in the **New BSP** dialog box.

For Cyclone V SoC/Arria Preloader BSPs, the generated files include the following:

- **settings.bsp** – file containing all BSP settings
- **Makefile** – makefile used to compile the Preloader and create the preloader image; for more information, refer to Preloader Compilation
- **preloader.ds** – deprecated, and not used
- **generated** – folder containing files generated from the hardware handoff files

For Intel Arria 10 SoC Bootloader BSPs the generated files include the following:

- **settings.bsp** – file containing all BSP settings
- **Makefile** – makefile used to compile the Bootloader, convert the Bootloader device tree file to binary, and create the combined Bootloader and Bootloader Device Tree image; for more information, refer to Preloader Compilation
- **config.mk** – deprecated, and not used
- **devicetree.dts** – Bootloader device tree, containing the Bootloader customization details, derived from the handoff files and you settings

6.2.5. BSP Settings

The BSP settings are deprecated and are not used anymore. Instead, the bootloader is configured manually, by selecting a different configuration, changing the selected configuration through "`make menuconfig`", editing the Device Tree or editing the source code.

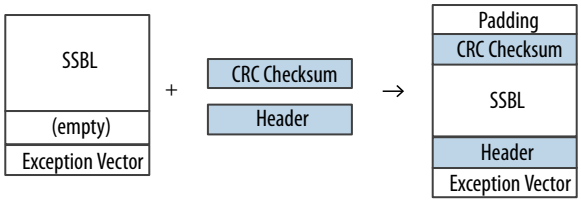
6.3. Bootloader Image Tool (`mkpimage`)

The Bootloader Image Tool (`mkpimage`) creates an Intel BootROM-compatible image of the Arria V SoC and Cyclone V SoC Preloader or Intel Arria 10 SoC Bootloader. The tool can also decode the header of previously generated images.

The `mkpimage` tool makes the following assumptions:

- The input file format is raw binary. You must use the **objcopy** utility provided with the GNU Compiler Collection (GCC) tool chain from the Mentor Graphics website to convert other file formats, such as Executable and Linking Format File (**.elf**), Hexadecimal (Intel-Format) File (**.hex**), or S-Record File (**. srec**), to a binary format.
- The output file format is binary.
- The tool always creates the output image at the beginning of the binary file. If the image must be programmed at a specific base address, you must supply the address information to the flash programming tool.
- The output file contains only Preloader or Bootloader images. Other images such as Linux, SRAM Object File (.sof) and user data are programmed separately using a flash programming tool or related utilities in the U-boot on the target system.

Figure 22. Basic Operation of the `mkpimage` Tool



6.3.1. Operation

The `mkpimage` tool runs on a host machine. The tool generates the header and CRC checksum and inserts them into the output image with the bootloader program image and its exception vector.

For certain flash memory tools, the position of the bootloader images must be aligned to a specific block size; the `mkpimage` tool generates any padding data that may be required.

The `mkpimage` tool optionally decodes and validates header information when given a pre-generated bootloader image.

As illustrated, the binary bootloader image is an input to the `mkpimage` tool. The compiler leaves an empty space between the bootloader exception vector and the program. The `mkpimage` tool overwrites this empty region with header information and calculates a checksum for the whole image.

When necessary, the `mkpimage` tool appends the padding data to the output image.

The `mkpimage` tool can operate with either one or four input files. Operation on four input files consists in processing each file individually, then concatenating the four resulted images.

6.3.2. Header File Format

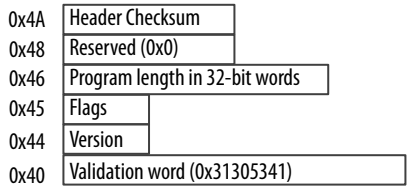
The `mkpimage` header file format has two versions:

- Version 0, used for Cyclone V SoC and Arria V SoC FSBL (Preloader)
- Version 1, used for Intel Arria 10 SoC Bootloader

For Version 0, used for Cyclone V SoC and Arria V SoC Preloader, the header includes the following:

- Validation word (0x31305341)
- Version field (set to 0x0)
- Flags field (set to 0x0)
- Program length measured by the number of 32 bit words in the Preloader program
- 16-bit checksum of the header contents (0x40 – 0x49)

Figure 23. Header Format Version 0

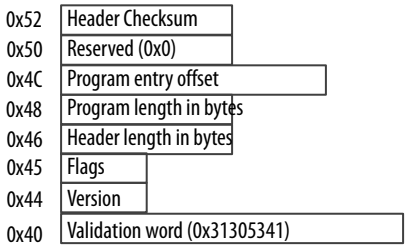


For Version 1, used for Intel Arria 10 SoC Bootloader, the header includes the following:

- Validation word (0x31305341).
- Version field (set to 0x1).
- Flags field (set to 0x0).
- Header length, in bytes, set to 0x14 (20 bytes).

- Total program length (including the exception vectors and the CRC field) in bytes. For an image to be valid, length must be a minimum of 0x5C (92 bytes) and a maximum of 0x32000 (200KiB).
- Program entry offset relative to the start of header (0x40) and should be 32-bit word-aligned. Default is 0x14, any value smaller than that is invalid.
- 16-bit checksum of the header contents (0x40 – 0x51):

Figure 24. Header Format Version 1



The header checksum for both versions of the `mkpimage` header is the CRC checksum of the byte value from offset 0x0 to (n*4)-4 bytes where n is the program length.

The CRC is a standard CRC32 with the polynomial:

```
x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1
```

There is no reflection of the bits and the initial value of the remainder is 0xFFFFFFFF and the final value is exclusive OR-ed with 0xFFFFFFFF.

6.3.3. Tool Usage

The mkimage tool has three usage models:

- Single image creation
- Quad image creation
- Single or quad image decoding

If an error is found during the make image process, the tool stops and reports the error. Possible error conditions include:

- For Cyclone V SoC and Arria V SoC Preloaders: input image is smaller than 81 bytes or larger than 60 KB.
- For Intel Arria 10 SoC Bootloaders: input image is smaller than 92 bytes or larger than 200 KB.

Type `mkpimage` to launch the tool. The `--help` option provides a tool description and tool usage and option information.

```
$ mkpimage --help
mkpimage version 19.1

Description: This tool creates an Intel BootROM-compatible image of Second
Stage Boot Loader (SSBL). The input and output files are in binary format.
It can also decode and check the validity of previously generated image.

Usage:
 Create quad image:
     mkpimage [options] -hv <num> -o <outfile> <infile> <infile> <infile>
<infile>
```

```
 Create single image:
     mkpimage [options] -hv <num> -o <outfile> <infile>
 Decode single/quad image:
     mkpimage -d [-a <num>] <infile>

Options:
 -a (--alignment) <num>       : Address alignment in kilobytes for output image
                                (64, 128, 256, etc.), default to 64 for header
                                version 0 and 256 for header version 1,
                                override if the NAND flash has a different
                                block size. If outputting a single image, value
                                of '0' is permitted to specify no flash block
                                padding (needed for SSBL image encryption).
 -d (--decode)                : Flag to decode the header information from
                                input file and display it
 -f (--force)                 : Flag to force decoding even if the input file
                                is an unpadded image
 -h (--help)                  : Display this help message and exit
 -hv (--header-version) <num> : Header version to be created (Arria/Cyclone V
SoC =
                                0, Intel
          Arria 10 SoC = 1)
 -o (--output) <outfile>      : Output file, relative and absolute path
                                supported
 -off (--offset) <num>        : Program entry offset relative to start of
                                header (0x40), default to 0x14. Used for header
                                version 1 only
 -v (--version)               : Display version and exit
```

6.3.4. Output Image Layout

6.3.4.1. Base Address

The bootable SSBL image must be placed at 0x0 for NAND and QSPI flash. For SD/MMC the image can also be placed at 0x0, but typically the image is placed at offset 0x0 in a custom partition of type 0xA2. The custom partition does not have a filesystem on it. The BootROM is able to locate the partition using the MBR (Master Boot Record) located at 0x0 on the SD/MMC card.

The `mkpimage` tool always places the output image at the start of the output binary file, regardless of the target flash memory type. The flash programming tool is responsible for placing the image at the desired location on the flash memory device.

6.3.4.2. Size

For Cyclone V SoC and Arria V SoC, a single Preloader has a maximum 60 KB image size. You can store up to four preloader images in flash. If the BootROM does not find a valid preloader image at the first location, it attempts to read an image from the next location and so on. To take advantage of this feature, program four preloader images in flash.

For Intel Arria 10 SoC, a single Bootloader has a maximum 200 KB image size. You can store up to four Bootloader images in flash. If the BootROM does not find a valid Bootloader image at the first location, it attempts to read the next one and so on. To take advantage of this feature, program four Bootloader images in flash.

6.3.4.3. Address Alignment

For Cyclone V SoC and Arria V SoC, every Preloader image has to be aligned to a 64KB boundary, except for NAND devices. For NAND devices, each Preloader image has to be aligned to the greater of 64 KB or NAND block size.

For Intel Arria 10 SoC, every Bootloader image has to be aligned to a 256 KB boundary, except for NAND devices. For NAND devices, each Bootloader image has to be aligned to the greater of 256 KB or NAND block size.

The following tables present typical image layouts, that are used for QSPI, SD/MMC and NAND devices with NAND erase block size equal or less to 64 KB (for Cyclone V SoC/Arria V SoC) or 256 KB (for Intel Arria 10 SoC).

Table 4. Typical Arria V SoC/Cyclone V SoC Preloader Image Layout

Offset	Image
0x30000	Preloader Image 3
0x20000	Preloader Image 2
0x10000	Preloader Image 1
0x00000	Preloader Image 0

Table 5. Typical Intel Arria 10 SoC Bootloader Image Layout

Offset	Image
0xC0000	Bootloader Image 3
0x80000	Bootloader Image 2
0x40000	Bootloader Image 1
0x00000	Bootloader Image 0

The `mkpimage` tool is unaware of the target flash memory type. If you do not specify the block size, the default is 64 KB.

6.3.4.3.1. NAND Flash

Each Preloader or Bootloader image occupies an integer number of blocks. A block is the smallest entity that can be erased, so updates to a particular boot image do not impact the other images.

For example for Cyclone V SoC and Arria V SoC, a single Preloader image has a maximum size of 64 KB. But it the NAND block is 128 KB, then the Preloader images will need to be located at 128 KB intervals.

6.3.4.3.2. Serial NOR Flash

Each QSPI boot image occupies an integer number of sectors unless subsector erase is supported; this ensures that updating one image does not affect other images.

6.3.4.3.3. SD/MMC

The master boot record, located at the first 512 bytes of the device memory, contains partition address and size information. The Preloader and Bootloader images are stored in partitions of type 0xA2. Other items may be stored in other partition types according to the target file system format.

You can use the fdisk tool to set up and manage the master boot record. When the fdisk tool partitions an SD/MMC device, the tool creates the master boot record at the first sector, with partition address and size information for each partition on the SD/MMC.

6.3.4.4. Padding

The mkimage tool inserts a CRC checksum in the unused region of the image. Padding fills the remaining unused regions. The contents of the padded and unused regions of the image are undefined.

Related Information

- Arria V Hard Processor System Technical Reference Manual
  For more information, please refer to the Booting and Configuration chapter.
- Cyclone V Hard Processor System Technical Reference Manual
  For more information, please refer to the Booting and Configuration chapter.
- Arria 10 Hard Processor System Technical Reference Manual
  For more information, please refer to the Booting and Configuration chapter.

6.4. U-Boot Image Tool (mkimage)

Both the FSBL and SSBL require the presence of the U-Boot image header at the beginning of the next stage boot image. Depending on usage scenario, other items that are loaded by either Preloader or Bootloader may also require the presence of the U-Boot image header.

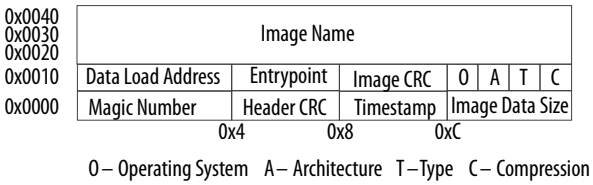
The mkimage utility is delivered with SoC EDS and can be used to append the U-Boot image header to the next stage boot image or any other required files.

Figure 25. mkimage Header

The header consists of the following items:

- Image magic number - determines if the image is a valid boot image
- Image data size - the length of the image to be copied
- Data load address - the entry point of the boot image (not used for items that are not bootable images)
- Operating system - determines the type of image
- Image name - the name of the boot image
- Image CRC - the checksum value of the boot image

Figure 26. mkimage Header Layout



6.4.1. Tool Options

mkimage invokes the mkimage tool and the --help option provides the tool description and option information.

```
$ mkimage --help
Usage: mkimage -l image
          -l ==> list image header information
       mkimage [-x] -A arch -O os -T type -C comp -a addr -e ep -n name -d
data_file[:data_file...] image
          -A ==> set architecture to 'arch'
          -O ==> set operating system to 'os'
          -T ==> set image type to 'type'
          -C ==> set compression type 'comp'
          -a ==> set load address to 'addr' (hex)
          -e ==> set entry point to 'ep' (hex)
          -n ==> set image name to 'name'
          -d ==> use image data from 'datafile'
          -x ==> set XIP (execute in place)
       mkimage [-D dtc_options] [-f fit-image.its|-F] fit-image
          -D => set options for device tree compiler
          -f => input filename for FIT source
Signing / verified boot not supported (CONFIG_FIT_SIGNATURE undefined)
       mkimage -V ==> print version information and exit
```

6.4.2. Usage Examples

Example 1. Creating a U-boot Image

```
mkimage –A arm –T firmware –C none –O u-boot –a 0x08000040 –e 0 –n "U-Boot
2014.10 for SOCFGPA board" –d u-boot.bin u-boot.img
```

Example 2. Creating a Bare Metal Application Image

```
mkimage -A arm -O u-boot -T standalone -C none -a 0x02100000 -e 0 -n "Bare
Metal image" -d hello_world.bin hello_world.img
```

6.5. Building the Bootloader

6.5.1. Building the Cyclone V SoC and Arria V SoC Preloader

For complete details on how to build the Cyclone V SoC and Arria V SoC bootloaders, please visit "Building Bootloader" on RocketBoards.org.

6.5.2. Building the Intel Arria 10 SoC Bootloader

For complete details on how to build the Intel Arria 10 SoC bootloaders, please visit "Building Bootloader" on RocketBoards.org.

6.5.3. Building the Intel Stratix 10 SoC and Intel Agilex Devices

On Intel Stratix 10 SoC and Intel Agilex devices, the SDM loads the FSBL from the configuration bitstream. The configuration bitstream contains the hardware handoff data, which is made available to the FSBL. Because of this, there is no need for a Bootloader Generator tool to pass additional information to the Bootloader building process.

For information about how to build the Intel Stratix 10 SoC U-Boot, refer to the *Building Bootloader* webpage on RocketBoards.

For information about ATF and UEFI, refer to *Intel Stratix 10 SoC UEFI Boot Loader User Guide*.

For information about the Intel Stratix 10 SoC bootloaders, refer to the *Intel Stratix 10 SoC FPGA Boot User Guide*.

Related Information

- Intel Stratix 10 SoC UEFI Boot Loader User Guide
- Intel Stratix 10 SoC FPGA Boot User Guide
- Building Bootloader

6.6. Boot Tools User Guide Revision History

Document Version	Changes
2020.08.07	Maintenance release.
2019.12.20	Supported with Intel Quartus Prime Standard Edition version 19.1 and Intel Quartus Prime Pro Edition version 19.3 • Added support for Intel Agilex • Provided instruction on how to build the bootloader now that the BSP generator GUI is deprecated • Removed the following sections: — *Using `.tcl` Scripts* — *Cyclone V SoC and Arria V SoC BSP Settings* — *U-Boot Build Flow* — *UEFI Build Flow* • Moved the following sections to the *Boot Tools User Guide Professional Edition*: — *Intel Arria 10 SoC BSP Settings* — *Intel Arria 10 SoC Main BSP Settings Group* — *Intel Arria 10 SoC Bootloader MPU Firewall BSP Settings Group* — *Intel Arria 10 SoC Bootloader L3 Firewall BSP Settings Group* — *Intel Arria 10 SoC Bootloader FPGA-to-SDRAM Firewall BSP Settings Group* • Removed the following BSP Generator Command Line Interfaces that were deprecated: — `bsp-update-settings` — `bsp-query-settings` — `bsp=generate-files` • Provided instruction on how to configure the bootloader manually now that the BSP settings are deprecated • Provided details on how to build the bootloaders by pointing to "Building Bootloader" on RocketBoards.org

continued...

Document Version	Changes
2019.05.16	• *Tool Usage* section: Updated the `mkpimage` version to 19.1. • *Building the Intel Stratix 10 SoC Bootloader* section: In step 1, updated the path for the 19.1 release.
2018.09.24	Updated the path names, in the "Building the Intel Stratix 10 SoC Bootloader" section, to coincide with the 18.1 release.
2018.06.18	• Updated chapter to include support for Intel Stratix 10 SoC • Replaced "Second Stage Bootloader" with "Bootloader". • Replaced "SSBL" with "FSBL" • Added the Intel Stratix 10 SoC Typical Boot Flow figure • Added the "Building the Intel Stratix 10 SoC Bootloader" section
2017.05.08	• Intel FPGA rebranding • Rebranded paths and tools for the Standard and Professional versions
2016.11.7	Added an "UEFI Build Flow" section
2016.05.27	Added a link to the Intel Arria 10 SoC Secure Boot User Guide
2016.02.17	• Updated the help definition for `mkpimage` in the "Tools Usage" and "Tool Options" sections • Updated the Intel Arria 10 SoC Main BSP Settings Group table
2015.08.06	Added Intel Arria 10 SoC support

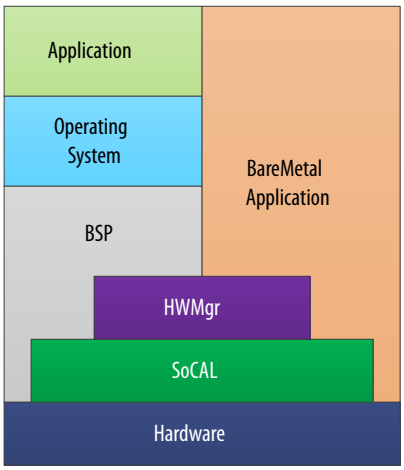
Related Information

Boot Tools User Guide Professional Edition

7. Hardware Library

The Intel SoC FPGA Hardware Library (HWLIB) was created to address the needs of low-level software programmers who require full access to the configuration and control facilities of SoC FPGA hardware. An additional purpose of the HWLIB is to mitigate the complexities of managing the operation of a sophisticated, multi-core application processor and its integration with hardened IP peripheral blocks and programmable logic in a SoC architecture.

Figure 27. Hardware Library



Within the context of the SoC hardware and software ecosystem, the HWLIB is capable of supporting software development in conjunction with full featured operating systems or standalone BareMetal programming environments. The relationship of the HWLIB within a complete SoC HW/SW environment is illustrated in the above figure.

The HWLIB provides a symbolic register abstraction layer known as the SoC Abstraction Layer (SoCAL) that enables direct access and control of HPS device registers within the address space. This layer is necessary for enabling several key stakeholders (boot loader developers, driver developers, BSP developers, debug agent developers, and board bring-up engineers) requiring a precise degree of access and control of the hardware resources.

The HWLIB also deploys a set of Hardware Manager (HW Manager) APIs that provides more complex functionality and drivers for higher level use case scenarios.

The HWLIB has been developed as a source code distribution. The intent of this model is to provide a useful set of out-of-the-box functionality and to serve as a source code reference implementation that a user can tailor accordingly to meet their target system requirements.

The capabilities of the HWLIB are expected to evolve and expand over time particularly as common use case patterns become apparent from practical application in actual systems.

In general, the HWLIB assumes to be part of the system software that is executing on the Hard Processor System (HPS) in privileged supervisor mode and in the secure state.

The anticipated HWLIB clients include:

- BareMetal application developers
- Custom preloader and boot loader software developers
- BSP developers
- Diagnostic tool developers
- Software driver developers
- Debug agent developers
- Board bring-up engineers
- Other developers requiring full access to SoC FPGA hardware capabilities

7.1. Feature Description

This section provides a description of the operational features and functional capabilities present in the HWLIB. An overview and brief description of the HWLIB architecture is also presented.

The HWLIB is a software library architecturally comprised of two major functional components:

- SoC Abstraction Layer (SoCAL)
- HW Manager

7.1.1. SoC Abstraction Layer (SoCAL)

The SoC Abstraction Layer (SoCAL) presents the software API closest to the actual HPS hardware. Its purpose is to provide a logical interface abstraction and decoupling layer to the physical devices and registers that comprise the hardware interface of the HPS.

The SoCAL provides the benefits of:

- A logical interface abstraction to the HPS physical devices and registers including the bit-fields comprising them.
- A loosely coupled software interface to the underlying hardware that promotes software isolation from hardware changes in the system address map and device register bit field layouts.

7.1.2. HW Manager

The HW Manager component provides a group of functional APIs that address more complex configuration and operational control aspects of selected HPS resources.

The HW Manager functions have the following characteristics:

- Functions employ a combination of low level device operations provided by the SoCAL executed in a specific sequence to effect a desired operation.
- Functions may employ cross functional (such as from different IP blocks) device operations to implement a desired effect.
- Functions may have to satisfy specific timing constraints for the application of operations and validation of expected device responses.
- Functions provide a level of user protection and error diagnostics through parameter constraint and validation checks.

The HW Manager functions are implemented using elemental operations provided by the SoCAL API to implement more complex functional capabilities and services. The HW Manager functions may also be implemented by the compound application of other functions in the HW Manager API to build more complex operations (for example, software controlled configuration of the FPGA).

7.2. Hardware Library Reference Documentation

Reference documentation for the SoCAL and HW Manager are distributed as part of the Intel SoC FPGA EDS. The documentation is in HTML format and is located at <SoC EDS `installation directory>/ip/altera/hps/doc`. You can view this documentation with any web browser.

7.3. System Memory Map

The addresses of the HPS hard IP modules are accessible through the provided SoCAL macros. SoCAL also provides macros for accessing the individual registers and register fields of the HPS hard IP modules.

For the FPGA IP modules, the macros for accessing IP registers and register fields are usually part of the IP deliverables. However, the actual IP modules-based addresses are often changed at system integration time, in the Platform Designer (Standard) tool.

The tool "sopc-create-header-files" can be used to create a C include file with the bases addresses of all the IP modules residing in the FPGA fabric.

Note: The tool is part of Intel Quartus Prime Standard Edition, and not of Intel SoC FPGA EDS. The tool can be invoked from the SoC EDS Embedded Command Shell once Intel Quartus Prime Standard Edition is installed.

The basic usage of the tool is to invoke it with the .sopcinfo file as a single parameter. For example, in order to generate the include files for the Intel Arria 10 SoC GHRD, the following command can be executed in that folder: `sopc-create-header-files ghrd_10as066n2.sopcinfo`.

The tool creates a separate include file for each of the masters in the system, showing the system addresses from that master's point of view. Use the file **<hps_component_name>_a9_0.h** for HPS software development, as it shows the system addresses from the HPS point of view.

The following example demonstrates how to use the tool to generate only the file that shows the HPS A9 Core 0 point of view:

```
sopc-create-header-files ghrd_10as066n2.sopcinfo --module\
arria10_hps_0_arm_a9_0 --single arria10_hps_0_arm_a9_0.h
```

This creates just one file, called "arria10_hps_0_arm_a9_0.h".

You can also run "`sopc-create-header-files --help`" for more details about the tool, or refer to Intel Quartus Prime Standard Edition documentation.

7.4. Hardware Library Revision History

Document Version	Changes
2020.08.07	Maintenance release
2019.12.20	Added support for Intel Quartus Prime Standard Edition version 19.1 and Intel Quartus Prime Pro Edition version 19.3 • Maintenance release
2019.05.16	Maintenance release
2018.09.24	Updated the location and format to find the reference documentation for the SoCAL and HW Manager.
2018.06.18	• Updated chapter to include support for Intel Stratix 10 SoC • Replaced "Second Stage Bootloader" with "Bootloader". • Updated information about the Toolchains
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	• Updated the HW Library figure • Added a new section called "System Memory Map"
2015.08.06	Added Intel Arria 10 SoC support

8. HPS Flash Programmer User Guide

The Intel Quartus Prime software and Intel Quartus Prime Programmer include the 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 the Intel Stratix 10 SoC devices. It is your responsibility to program the flash connected to HPS. Several options are possible:

- Use a bus switch to route the EPCQ-L/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, 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 of both the Intel Quartus Prime Pro Edition and Intel SoC FPGA Embedded Development Suite.

8.1. HPS Flash Programmer Command-Line Utility

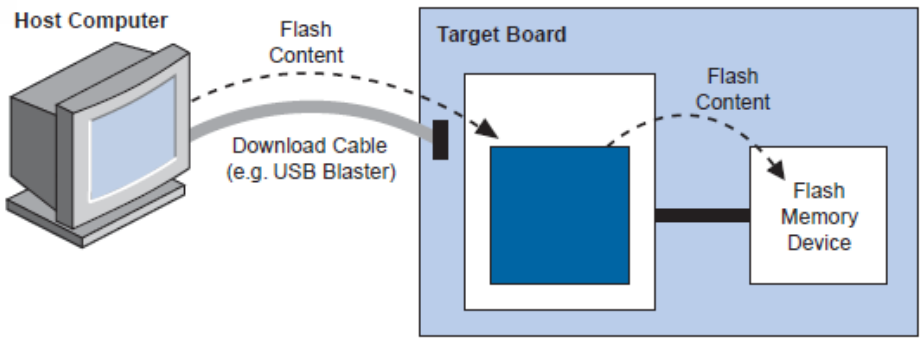
You can run the HPS flash programmer directly from the command line.

- For the Quartus Prime software, the HPS flash programmer is located in the <Quartus Prime installation directory>/quartus/bin directory.
- For the SoC EDS software, the HPS flash programmer is located in the <SoC EDS installation directory>/ qprogrammer/bin directory.

8.2. 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 28. 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.

8.3. Using the Flash Programmer from the Command Line

8.3.1. 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 6. 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 SoC) is `nconfig`. • Explicitly configure dedicated I/O and PINMUX Available options: • 1 - Set Breakpoint to halt CPU, warm reset HPS [not recommended](2) • 2 - Set Watchpoint to halt CPU, warm reset HPS(3) • 3 - Explicitly configure dedicated IO and PINMUX(4) • 16 - Cold reset HPS(5) Cyclone V SoC and Intel Arria 10 SoC support values: 1, 2 and 16. Intel Arria 10 SoC 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(6) • 18 - Cold reset HPS + watchpoint(7) • 19 - Cold reset HPS + configure dedicated IO and PINMUX(8)

continued...

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

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

(4) Explicitly configure dedicated IO and PINMUX.

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

Option	Short Option	Required	Description
`--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 Sislicon 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).

8.3.2. 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 3. 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 *M*.

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

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

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

Example 4. 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 *M*.

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

Example 5. 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 *M*.

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

Example 6. 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 *M*. 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 7. 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 *M*.

Example 8. Erase Full Chip

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

Example 9. 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 *M*.

Example 10. 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 *M*.

8.4. Supported Memory Devices

Table 7. 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 8. 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 the Cyclone V SoC, Arria V SoC and Intel Arria 10 SoC devices. For more information, refer to the "Supported Flash Devices for Cyclone V SoC and Arria V SoC" and the "Supported Flash Devices for Intel Arria 10 SoC" web pages.

Related Information

- Supported Flash Devices for Cyclone V and Arria V SoC
- Supported Flash Devices for Arria 10 SoC

8.5. HPS Flash Programmer User Guide Revision History

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 SoC • 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 SoC support

9. Bare Metal Compilers

The following bare metal compilers are available::

- Arm Bare Metal Compiler 5 (ARMCC)
- Arm Bare Metal Compiler 6 (LLVM-based)
- Linaro Bare Metal Compiler 7.5.0 (GCC based)

9.1. Arm Bare Metal Compilers

The Arm Bare Metal compilers are part of the Arm DS for Intel SoC FPGA Edition, which is provided as a separate download. For more information about installation, refer to the *Installing the Intel SoC FPGA EDS* section.

The Arm Bare Metal Compiler 5 targets 32-bit platforms (Cortex-A9 on Cyclone V SoC, Arria V SoC and Intel Arria 10 SoC).

The Arm Bare Metal Compiler 6 targets both 32-bit platforms and 64-bit platforms (Cortex-A53 on Intel Stratix 10 SoC).

The Arm Bare Metal compilers documentation is integrated in Arm DS for Intel SoC FPGA Edition, like with all the Arm DS for Intel SoC FPGA Edition components.

Related Information

Installing the Tools on page 11

9.2. Linaro GCC Compiler

The Linaro GCC compiler is provided as an additional download. For information about installation, refer to the *Installing the Intel SoC FPGA EDS* section.

The GCC-based compiler, **arm-eabi-gcc**, has the following features:

- Targets 32-bit platforms (Cortex-A9 on Cyclone V SoC, Arria V SoC and Intel Arria 10 SoC).
- Assumes bare metal operation
- Uses the standard Arm embedded-application binary interface (EABI) conventions.

The Bare Metal compiler is installed in the following folder:

```
<SoC EDS installation directory>/host_tools/linaro/gcc
```

The Embedded Command Shell sets the correct environment PATH variables for the bare metal compilation tools to be invoked. You can open the shell from the <*SoC EDS installation directory*>. After starting the shell, commands like **arm-eabi-gcc** can be

invoked directly. When the Arm DS for Intel SoC FPGA Edition IDE is started from the embedded command shell, it inherits the environment settings, and it can call these compilation tools directly.

You can also use the full path to the compilation tools:

```
<SoC EDS installation directory>/host_tools/linaro/gcc
```

The bare metal compiler is installed with full documentation, located at:

```
<SoC EDS installation directory>/host_tools/linaro/gcc/share/doc
```

The documentation is offered in HTML format.

Among the provided documents are:

- Compiler manual
- Assembler manual
- Linker manual
- Binutils manual
- GDB manual
- Libraries Manual

Related Information

- Installing the Tools on page 11
- Differences Between Standard and Professional Editions on page 4

9.3. Bare Metal Compilers Revision History

Document Version	Changes
2020.08.07	Added support for Intel Quartus Prime Standard Edition and Intel Quartus Prime Pro Edition software version 20.1 releases: • Updated path to the compilation tools
2019.12.20	Added support for Intel Quartus Prime Standard Edition version 19.1 and Intel Quartus Prime Pro Edition version 19.3 • Removed reference to SoC EDS Getting Started Guides
2019.05.16	Maintenance release
2018.09.24	Maintenance release
2018.06.18	• Updated chapter to include support for Intel Stratix 10 SoC • Added information that the Arm Bare Metal Compiler 6 targets both 32-bit and 64-bit platforms
2017.05.08	• Intel FPGA rebranding • Rebranded paths and tools for the Standard and Professional versions
2016.11.07	Added a "Mentor Code Sourcery Compiler" section
2016.05.27	Maintenance release
2016.02.17	Updated the compiler version for Mentor Graphics Sourcery™ CodeBench Lite Edition
2015.08.06	Added Intel Arria 10 SoC support

10. SD Card Boot Utility

The SoC EDS SD card boot utility is a tool for updating the boot software on an SD card.

The Preloader is typically stored in a custom partition (with type = 0xA2) on the SD card. Optionally, the next boot stage (usually the Bootloader) can also be stored on the same custom partition.

Since it is a custom partition without a file-system, the Preloader, Bootloader, or both cannot be updated by copying the new file to the card; and a software tool is needed.

The SD card boot utility allows you to update the Preloader, Bootloader, or both on a physical SD card or a disk image file. The utility is not intended to create a new bootable SD card or disk image file from scratch. In order to do that, it is recommended to use **fdisk** on a Linux host OS.

Note: The SD Card Boot Utility tool is not needed for Intel Stratix 10 SoC devices.

10.1. Usage Scenarios

This utility is intended to update boot software on that resides on an existing:

- Existing SD card
- Existing disk image file

The tool supports updating the following:

- Cyclone V SoC and Arria V SoC Preloader—The Preloader file needs to have the `mkpimage` header, as required by the bootROM.
- Cyclone V SoC and Arria V SoC Bootloader—The Bootloader file needs to have the `mkimage` header, as required by the Preloader.
- Intel Arria 10 SoC Bootloader—The Bootloader needs to have the `mkpimage` header, as required by the bootROM.

Note: Both `mkpimage` and `mkimage` tools are delivered as part of SoC EDS.

The tool only updates the custom partition that stores the Intel SoC boot code. The rest of the SD card or disk image file is not touched. This includes the Master Boot Record (MBR) and any other partitions (such as FAT and EXT3) and free space.

Warning: The users of this tool need administrative or root access to their computer to use this tool to write to physical SD cards. These rights are not required when only working with disk image files. Please contact the IT department if you do not have the proper rights on your PC.

10.2. Tool Options

The utility is a command line program. The table describes all the command line options; and the figure shows the `--help` output from the tool.

Table 9. Command Line Options

Command line Argument	Required?	Description
-p filename	Optional	Specifies Preloader file to write
-b filename	Optional	Specifies Cyclone V or Arria V SoC Bootloader file to write
-B filename	Optional	Specifies Intel Arria 10 SoC Bootloader file to write
-a write	Required	Specifies action to take. Only "write" action is supported. Example: "-a write"
disk_file	Required(unless -d option is used)	Specifies disk image file or physical disk to write to. A disk image file is a file that contains all the data for a storage volume including the partition table. This can be written to a physical disk later with another tool. For physical disks in Linux, just specify the device file. For example: /dev/ mmcblk0 For physical disks in Windows, specify the physical drive path such as \\.\physicaldrive2 or use the drive letter option(-d) to specify a drive letter. The drive letter option is the easiest method in Windows
-d	Optional	specify disk drive letter to write to. Example: "-d E". When using this option, the disk_file option cannot be specified.
-h	Optional	Displays help message and exits
--version	Optional	Displays script version number and exits

Sample Output from Utility

```
$ alt-boot-disk-util --help
Altera Boot Disk Utility
Copyright (C) 1991-2014 Altera Corporation

Usage:
#write preloader to disk
    alt-boot-disk-util -p preloader -a write disk_file

#write bootloader to disk
    alt-boot-disk-util -b bootloader -a write disk_file

#write BOOTloader and PREloader to disk
    alt-boot-disk-util -p preloader -b bootloader -a write disk_file

#write Arria10 Bootloader to disk
    alt-boot-disk-util -B a10bootloader -a write disk_file

#write BOOTloader and PREloader to disk drive 'E'
    alt-boot-disk-util -p preloader -b bootloader -a write -d E
```

```
Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -b FILE, --bootloader=FILE
                        bootloader image file'
  -p FILE, --preloader=FILE
                        preloader image file'
  -a ACTION, --action=ACTION
                        only supports 'write' action'
  -B FILE, --a10-bootloader=FILE
                        arria10 bootloader image file
  -d DRIVE, --drive=DRIVE
                        specify disk drive letter to write to
```

10.3. SD Card Boot Utility Revision History

Document Version	Changes
2020.08.07	Maintenance release
2020.07.31	Maintenance release
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	Maintenance release
2018.09.24	Maintenance release
2018.06.18	• Updated chapter to indicate that the SD Card Boot Utility is not needed for Intel Stratix 10 SoC devices
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	Maintenance release
2015.08.06	Added Intel Arria 10 SoC support

11. Linux Device Tree Generator

A device tree is a data structure that describes the underlying hardware to an operating system - primarily Linux. By passing this data structure to the OS kernel, a single OS binary may be able to support many variations of hardware. This flexibility is particularly important when the hardware includes an FPGA.

The Linux Device Tree Generator (DTG) tool is part of SoC EDS and is used to create Linux device trees for the Cyclone V SoC, Arria V SoC, and Intel Arria 10 SoC systems that contain FPGA designs created using Platform Designer. The generated device tree describes the HPS peripherals, selected FPGA Soft IP, and peripherals that are board dependent. The Intel Arria 10 SoC Bootloader also has an associated Device Tree called Bootloader Device Tree that is created and managed by the BSP Editor tool.

Warning: The Linux Device Tree Generator is obsolete and does not support Intel Stratix 10 SoC and newer devices. However, it is tested with and supports only the Linux kernel version targeted by the associated GSRD. Intel does not recommend that you use the Linux Device Tree Generator if your design targets a different Linux kernel version. Instead, Intel recommends that you manage the Device Tree manually by using the Device Tree files provided by the kernel as a baseline, and by adding the FPGA IP and board information manually.

Refer to *HOWTOCreateADeviceTree*, which supports all kernels.

Related Information

HOWTO Create a Device Tree

11.1. Linux Device Tree Generator Revision History

Document Version	Changes
2020.08.07	Added support for Intel Quartus Prime Standard Edition and Intel Quartus Prime Pro Edition software version 20.1 releases: • Renamed to *Linux Device Tree Generator* because this is the only tool available • Removed the *Linux Compiler* section.
2019.12.20	Added support for Intel Quartus Prime Standard Edition version 19.1 and Intel Quartus Prime Pro Edition version 19.3 •
2019.05.16	Maintenance release
2018.09.24	Maintenance release
2018.06.18	• Updated chapter to include support for Intel Stratix 10 SoC • Added information about using the Linaro compiler instead of the Linux compiler • Updated the chapter to indicate that the Linux Device Tree Generator does not support the Intel Stratix 10 SoC devices.

continued...

Document Version	Changes
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	Maintenance release
2015.08.06	Added Intel Arria 10 SoC support

12. Support and Feedback

Intel values your feedback. Please contact your Intel TSFAE or submit a service request at myIntel FPGA Sign In to report software bugs, potential enhancements, or obtain any additional information.

Related Information

myIntel FPGA Sign In

12.1. Support and Feedback Revision History

Document Version	Changes
2020.08.07	Maintenance release
2019.12.20	Added support for Intel Quartus Prime Standard Edition version 19.1 and Intel Quartus Prime Pro Edition version 19.3 • Removed reference to SoC EDS Getting Started Guides
2019.05.16	Maintenance release
2018.09.24	Maintenance release
2018.06.18	Updated chapter to include support for Intel Stratix 10 SoC
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	Maintenance release
2015.08.06	Added Intel Arria 10 SoC support