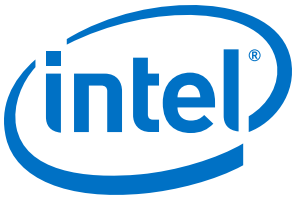


Intel Stratix 10 SoC UEFI Boot Loader User Guide

Subscribe
Send Feedback

UG-20080 | 2020.06.19
Latest document on the web: PDF | HTML

Contents

1. Overview

This document provides comprehensive information on Unified Extensible Firmware Interface (UEFI) boot loader for Intel Stratix 10 SoC.

The Intel Stratix 10 SoC provides a secure boot flow, consisting of:

- The boot ROM
- The secure device manager (SDM)
- The Secure Monitor
- The UEFI boot loader

The Intel Stratix 10 SoC secure boot flow ensures that the system boot loader is signed with a cryptographic key, validated by the firmware.

The Secure Monitor stage also implements the TrustZone* model of secure partitioning. This model divides the software environment into two isolated partitions, called the secure world and the non-secure world. The two worlds can only communicate with each other through the Secure Monitor.

The binary image of the UEFI boot loader can be stored on Quad SPI flash SD/MMC card. On board power-up, the secure device manager (SDM) loads the Secure Monitor directly onto Hard Processor System (HPS) on-chip RAM. Then the Secure Monitor loads the UEFI boot loader in HPS DDR memory.

The Secure Monitor tasks include:

- Initializing DDR SDRAM memory
- Configuring low level hardware, such as PLL, IOs, and pin MUXes, needed by nonsecure world software

The UEFI boot loader tasks include:

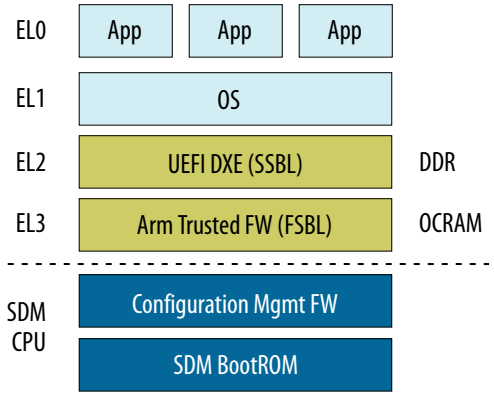
- Providing Ethernet support
- Supporting basic hardware diagnostic features
- Fetching subsequent boot software such as the operating system package or kernel image.

Note: For non-secure boot, the operating system package can include kernel image, device tree blob and filesystem. For secure boot it can be a secure kernel.

Intel Corporation. All rights reserved. Agilex, Altera, Arria, Cyclone, Enpirion, Intel, the Intel logo, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Figure 1. UEFI Boot Flow Overview



2. System Requirements

To load and execute the Intel Stratix 10 SoC Unified Extensible Firmware Interface (UEFI) boot loader, your system must meet the following requirements.

2.1. Minimum Hardware Requirements

- Linux workstation with the following configuration:
  - Serial terminal, such as Minicom for Linux
  - microSD card slot or microSD card writer or SD capable writer with SD to microSD converter

Table 1. Platform Capabilities

	Linux
Able to compile the UEFI boot loader	Yes
Able to compile the Secure Monitor	Yes

2.2. Minimum Software Requirements

- Intel® SoC FPGA Embedded Development Suite (SoC EDS) v18.1 and above
- Linaro aarch64-linux-gnu-gcc toolchain

Intel Corporation. All rights reserved. Agilex, Altera, Arria, Cyclone, Enpirion, Intel, the Intel logo, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

3. Getting Started

3.1. Installing Software Components

3.1.1. Installing the Intel SoC EDS

You must install the Intel SoC EDS on your machine.

Download the Intel SoC EDS from the Download Center for FPGAs.

3.1.2. Installing the Compiler Toolchain

You compile the UEFI boot loader and the Secure Monitor with the GNU Toolchain (EABI Release) for Arm* Processors.

You can download the GNU Toolchain from Arm's download page.

- Linux: `gcc-arm-8.3-2019.03-x86_64-aarch64-linux-gnu.tar.xz`

3.2. Building the Secure Monitor

As security becomes more and more important, a secured boot solution becomes a requirement in the embedded world. To ensure comprehensive security and a trusted platform, secure partitioning is required. The Intel Stratix 10 device achieves secure partitioning by implementing the TrustZone model with Arm Trusted Firmware (ATF). The TrustZone model splits the computing environment into two isolated worlds, the secure world and normal world, which are linked by a software monitor called the Secure Monitor. The two worlds have separated logical address space and peripherals. Communication between the two worlds is only possible by calling the privileged Secure Monitor call (SMC) instruction.

The full secure boot solution is:

- BootRom
- Secure Device Manager
- Secure Monitor
- Uboot/UEFI
- Hypervisor
- OS

Secure Monitor mode is a privileged mode and is always secure regardless of the state of the NS bit. The Secure Monitor is code that runs in Secure Monitor mode and processes switches to and from the Secure world. The overall security of the software relies on the security of this code along with the Secure boot code.

Intel Corporation. All rights reserved. Agilex, Altera, Arria, Cyclone, Enpirion, Intel, the Intel logo, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Related Information

www.trustedfirmware.org
General information about Arm Trusted Firmware

3.2.1. User Configuration

You can find all platform configurations in `arm-trusted-firmware/plat/intel/soc/stratix10/include/socfpga_plat_def.h`.

For user configuration, you must modify the boot sources based on your preferences. You select `BOOT_SOURCE_SDMMC` if boot from SDMMC or select `BOOT_SOURCE_QSPI` if boot from QSPI.

```
#define BOOT_SOURCE BOOT_SOURCE_SDMMC
```

Note: To change the boot filename or offset, you can change the `#define` in this file.

3.2.2. Getting the Arm Trusted Firmware Source Code

The ATF source is at GitHub. To get the ATF source code, simply run the following steps:

1. Open a terminal.
2. Create a new directory to check out the ATF source code from GitHub.
3. Change to this working directory and clone the ATF source from the Git trees as follows:

```
$ git clone https://github.com/altera-opensource/arm-trusted-firmware
```

4. When completed, change to the `arm-trusted-firmware` folder and perform a Git check out as follows:

```
$ cd arm-trusted-firmware
$ git checkout socfpga_v2.1
```

Related Information

- Building the ATF on page 7
- Compiling the UEFI Source Code with the Linaro Tool Chain on page 10
- Running the Secure Monitor on page 13

3.2.3. Building the ATF

This section describes how to build the ATF with the Linaro GCC compiler.

To start building the ATF with the Linaro GCC compiler, simply run the following steps:

1. Change your directory to the ATF source code location as follows:

```
$ cd arm-trusted-firmware
```

2. Set the GCC path and environment variable CROSS_COMPILE to Linaro cross compile as follows:

```
export PATH=<your gcc directory>/\
gcc-arm-8.3-2019.03-x86_64-aarch64-linux-gnu/bin/:$PATH
$ export ARCH=arm64
$ export CROSS_COMPILE=aarch64-linux-gnu-
```

3. Remove the build tree completely as follows:

```
$ make realclean
```

4. Build the ATF by using the following command:

```
$ make PLAT=stratix10 bl2 bl31
```

5. The following messages appear when the ATF build is successful:

```
  AS      lib/cpus/aarch64/cortex_a53.S
  AS      bl31/aarch64/bl31_entrypoint.S
  AS      bl31/aarch64/crash_reporting.S
  AS      bl31/aarch64/ea_delegate.S
  AS      bl31/aarch64/runtime_exceptions.S
  AS      lib/cpus/aarch64/dsu_helpers.S
  AS      plat/common/aarch64/platform_mp_stack.S
  AS      lib/el3_runtime/aarch64/cpu_data.S
  AS      lib/cpus/aarch64/cpu_helpers.S
  AS      lib/locks/exclusive/aarch64/spinlock.S
  AS      lib/psci/aarch64/psci_helpers.S
  AS      lib/el3_runtime/aarch64/context.S
  AS      lib/cpus/aarch64/wa_cve_2017_5715_bpiall.S
  AS      lib/cpus/aarch64/wa_cve_2017_5715_mmu.S
  AS      common/aarch64/debug.S
  AS      lib/aarch64/cache_helpers.S
  AS      lib/aarch64/misc_helpers.S
  AS      plat/common/aarch64/platform_helpers.S
  AS      drivers/ti/uart/aarch64/16550_console.S
  AS      plat/intel/soc/common/aarch64/plat_helpers.S
  PP      bl31/bl31.ld.S
  LD      build/stratix10/release/bl31/bl31.elf
  BIN     build/stratix10/release/bl31.bin

Built build/stratix10/release/bl31.bin successfully

  OD      build/stratix10/release/bl31/bl31.dump
```

6. The table below lists the Secure Monitor output files.

Table 2. Descriptions of Secure Monitor Files

File Path and Name	Description
\build\stratix10\release\bl31.bin	Generated binary file
\build\stratix10\release\bl31\bl31.elf	Generated elf file
\build\stratix10\release\bl2.bin	Generated binary file
\build\stratix10\release\bl2\bl2.elf	Generated elf file

3.3. Building the UEFI Boot Loader

To build a UEFI boot loader, you obtain the UEFI source code and compile the UEFI source with the supported toolchain.

The Unified Extensible Firmware Interface (UEFI) is a standardized firmware specification that simplifies and secures platform initialization and firmware bootstrap operations. UEFI is currently developed and supported by representatives from more than 250 industry-leading technology companies. Arm and the Linaro Enterprise Group are also promoting the use of UEFI on Arm architecture, because the UEFI specification helps standardize the boot process for Arm processor-based platforms.

UEFI technology is future-proofed through standardization of firmware design rather than proprietary firmware design. UEFI specifications promote business and technological efficiency, improve performance and security, facilitate interoperability between devices, platforms and systems and comply with next-generation technologies. The UEFI specification is peer-reviewed and published, allowing developers to write firmware once per platform and reuse it without much modification. This reuse results in cost and time savings during boot loader development.

This framework uses the BSD license, permitting you to optionally commercialize your implementation with minimal legal issues.

You can compile the UEFI source code either in a Windows or in a Linux system.

3.3.1. Prerequisites

Building the UEFI requires additional Linux packages. Depending on your Linux distribution, the command to install the packages is different:

If you are using a Ubuntu distribution, type:

```
$ sudo apt-get install uuid-dev build-essential
```

If you using a Fedora distribution, type:

```
$ sudo yum install uuid-devel libuuid-devel
```

For building UEFI, the Python package is required. If Python is not already available on your system, running the commands from the SoC EDS Embedded Command Shell provides the required Python dependency.

3.3.2. Obtaining the UEFI Source Code

The UEFI source code is located in GitHub. The following steps show you how to get the UEFI source code.

1. Open a terminal.
2. Clone the UEFI source from the Git trees.

   ```
   $ git clone https://github.com/altera-opensource/uefi-socfpga edk2
   ```

3. When completed, change to the edk2 folder and perform a Git check out.

   ```
   $ cd edk2
   $ git checkout socfpga_udk201905
   ```

3.3.3. Obtaining the edk2 Platform Source Code

The edk2 platforms source code is located in GitHub. To get the edk2 platforms source code:

```
$ git clone https://github.com/altera-opensource/edk2-platforms-socfpga edk2-platforms
```

```
$ cd edk2-platforms
```

```
$ git checkout socfpga_udk201905
```

3.3.4. Compiling the UEFI Source Code with the Linaro Tool Chain

This section explains how to compile the UEFI source code with the Linaro toolchain in a Linux system:

1. Open a terminal and enter the following command:

```
$ cd <your directory that contain edk2 and edk2-platforms>
$ export PATH=<your gcc directory>/\
gcc-arm-8.3-2019.03-x86_64-aarch64-linux-gnu/bin/:$PATH
$ export CROSS_COMPILE= aarch64-linux-gnu-
$ export ARCH=arm64
$ export GCC48_AARCH64_PREFIX=aarch64-linux-gnu-
```

2. Set up the EDK_TOOLS_PATH:

```
$ export EDK_TOOLS_PATH=$PWD/edk2/BaseTools
```

3. Set up the a PACKAGES_PATH to point to the location of the repositories:

```
$ export PACKAGES_PATH= $PWD/edk2:$PWD/edk2-platforms/
```

4. Set up the WORKSPACE:

```
$ export WORKSPACE = $PWD
```

5. Set up the build environment:

```
$ . edk2/edksetup.sh
```

6. Build BaseTools (ensure the python tools are installed):

```
$ make -C edk2/BaseTools
```

7. Compile the UEFI bootloader by entering the following command:

```
$ build -a AARCH64 -p Platform/Intel/Stratix10/Stratix10SoCPkg.dsc -t GCC48
-b DEBUG -y report.log -j build.log -Y PCD -Y LIBRARY -Y FLASH -Y DEPEX -Y
BUILD_FLAGS -Y FIXED_ADDRESS
```

8. Your terminal displays a "Build Done" message after the UEFI is successfully compiled.

3.3.5. UEFI Generated Files

Compiling the UEFI source code creates the following files in the `/Build/Stratix10SoCPkg/RELEASE_GCC48` folder:

Table 3. UEFI Generated Files

File	Description
`INTELSTRATIX10_EFI.fd`	This file is the UEFI bootloader to boot UEFI shell and enable ethernet feature or run a UEFI application

3.3.6. Generating the FIP

FIP is the payload that ATF's BL2 loads into RAM and executed. The FIP contains the binary for BL31 and UEFI bootloader, and a container that BL2 recognizes.

To build the FIP, follow these commands:

```
$ export ARCH = ARM64
```

```
$ export CROSS_COMPILE= aarch64-linux-gnu-
```

```
$ cd <your arm-trusted-firmware source code directory>
```

Build the FIP by using the following command:

```
$ make fip BL33= <your UEFI build workspace>/Build/Stratix10SoCPKG/\
DEBUG_GCC48/FV/INTELSTRATIX10_EFI.fd fip PLAT=stratix10
```

4. Running UEFI on Intel Stratix 10 Hardware

4.1. Running on a Physical Board with ATF and UEFI Bootloader

This section describes how to run the Secure Monitor on a physical board.

4.1.1. Generate a .sof file with ATF

1. Get a `.sof` file from the $SOCEDS_DEST_ROOT installation directory.
2. Convert the binary file `bl2.bin`, generated in *Building the ATF*.

```
$ aarch64-linux-gnu-objcopy -I binary -O ihex - \
-change-addresses 0xffe00000 bl2.bin bl2.hex
```

3. Include the bootloader into the `.sof` file as follows:

```
$ quartus_pfg -c -o hps_path=bl2.hex \
ghrd_1sx280lu2f50e2vg.sof ghrd_1sx280lu2f50e2vg_hps.sof
```

```
$ quartus_pfg -c -o hps_path=bl2.hex ghrd_1sx280lu2f50e2vg.sof ghrd_1sx280lu2f50e2vg_hps.sof
Info: ********************************************************************
Info: Running Quartus Prime Programming File Generator
    Info: Version 20.3.0 Internal Build 15 03/05/2020 SC Pro Edition
    Info: Copyright (C) 2020  Intel Corporation. All rights reserved.
    Info: Your use of Intel Corporation's design tools, logic functions
    Info: and other software and tools, and any partner logic
    Info: functions, and any output files from any of the foregoing
    Info: (including device programming or simulation files), and any
    Info: associated documentation or information are expressly subject
    Info: to the terms and conditions of the Intel Program License
    Info: Subscription Agreement, the Intel Quartus Prime License Agreement,
    Info: the Intel FPGA IP License Agreement, or other applicable license
    Info: agreement, including, without limitation, that your use is for
    Info: the sole purpose of programming logic devices manufactured by
    Info: Intel and sold by Intel or its authorized distributors.  Please
    Info: refer to the applicable agreement for further details, at
    Info: https://fpgasoftware.intel.com/eula.
    Info: Processing started: Tue Mar 10 15:26:53 2020
    Info: System process ID: 34910
Info: Command: quartus_pfg -c -o hps_path=bl2.hex ghrd_1sx280lu2f50e2vg.sof ghrd_1sx280lu2f50e2vg_hps.sof
Add bootloader data in the SOF
Info: Quartus Prime Programming File Generator was successful. 0 errors, 0 warnings
    Info: Peak virtual memory: 1748 megabytes
    Info: Processing ended: Tue Mar 10 15:27:00 2020
    Info: Elapsed time: 00:00:07
    Info: System process ID: 34910
```

Related Information

Building the ATF on page 7

4.1.2. Creating an SD Card Image

1. Generate UEFI Bootloader and FIP as in *Building the UEFI Boot Loader* and *Generating the FIP*.
2. Build Linux and root file system based on the instructions in Rocketboard.
3. Build the SD card image:

Intel Corporation. All rights reserved. Agilex, Altera, Arria, Cyclone, Enpirion, Intel, the Intel logo, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

a. Get the `make_image` python script and make it executable:

```
$ wget https://releases.rocketboards.org/release/2019.10/gsrd/tools/
make_sdimage.py
```

```
$ chmod +x make_sdimage.py
```

b. Prepare the fat partition contents:

```
$ mkdir fat && cd fat
```

```
$ cp <your linux image folder>/linux-socfpga/arch/arm64/boot/Image
```

```
$ cp <your device tree folder>/linux-socfpga/arch/arm64/boot/dts/altera/
socfpga_stratix10_socdk.dtb
```

c. Prepare the root file system partition contents:

```
$ mkdir rootfs && cd rootfs
```

```
$ tar xf <your rootfs directory>/gsrd-console-image-*.tar.xz
```

d. Create the SD card image:

```
$ sudo ./make_sdimage.py -f -P fip.bin,num=3,format=raw,size=10M,
type=A2 -P rootfs/\
*,num=2,format=ext3,size=1500M -P
Image,socfpga_stratix10_socdk.dtb,num=1,format=fat32,size=500M -s 2G -n
sdimage.img
```

Note: If you already have an SD image with A2 partition, you can replace the FIP file with the command below:

```
$ sudo dd if =arm-trusted-firmware/build/stratix10/release/fip.bin
of=/dev/sdx3
```

Related Information

- Compiling the UEFI Source Code with the Linaro Tool Chain on page 10
- Building the UEFI Boot Loader on page 8

4.1.3. Running the Secure Monitor

1. Power up the board after the SD card is inserted.
2. Open Quartus programmer and program the board with the `.sof` file generated in *Generating a .sof File with ATF*.

The board boots up from the ATF and automatically loads UEFI bootloader to boot UEFI shell.

Related Information

Generate a .sof file with ATF on page 12

4.1.4. Debugging with DS

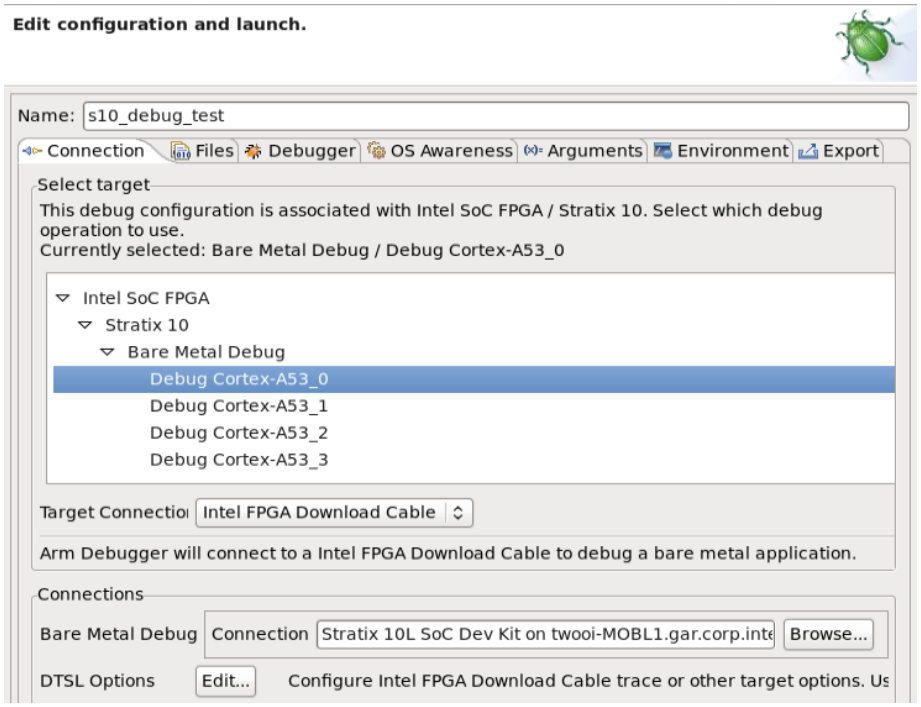
This section describes how to load ATF and UEFI bootloader to the physical board through DS.

1. Ensure that you have installed DS. Launch eclipse using the following command:

```
$ armds_ide &
```

2. Set up new debug connection:

Figure 2. Step Illustration



3. After configuration is complete, connect to the target.

   Note: You must program the board with `ghrd_1sx280lu2f50e2vg_hps_debug.sof` before connecting to the target.

4. In the DS command console, you may load a debug script with the following contents to download the ATF and UEFI bootloader to physical board.

```
# Load BL2 using debugger
restore <path to ATF>/build/stratix10/debug/bl2.bin binary 0xffe00000
add-symbol-file <path to ATF>/build/stratix10/debug/bl2/bl2.elf

# Initial State
delete
```

```
set $pc = 0xffe00000

# Skips watchdog
hbreak watchdog_init
continue
wait 5s
set $r0 = 0
set $pc = $lr

# Skips image authentication
# BL31
hbreak load_auth_image
continue
wait 5s
set $r0 = 0
set $pc = $lr

# BL33 (UEFI)
continue
wait 5s
set $r0 = 0
set $pc = $lr

# Load BL31 & BL33 using debugger
# BL31_BASE = 0x1000
restore <path to ATF>/build/stratix10/debug/bl31.bin binary 0x1000
add-symbol-file <path to ATF>/build/stratix10/debug/bl31/bl31.elf

# PLAT_NS_IMAGE_OFFSET = 0x10000000
restore <path to UEFI>/Build/Stratix10SoCPkg/DEBUG_GCC48/FV/
INTELSTRATIX10_EFI.fd binary 0x10000000
continue
```

4.2. Booting Linux

This section shows you how to boot Linux after UEFI enters the UEFI shell.

4.2.1. Booting from the UEFI Shell

1. Boot the board up to the UEFI shell, as described in *Running the Secure Monitor*.
2. Once the UEFI shell is loaded, enter the following command to boot Linux:

```
$ Image dtb=socfpga_stratix10_socdk.dtb console=ttyS0,115200 root=/dev/mmcb
```

Note: Make sure Linux image and dtb is stored in the SD card.

```
/HD(3,MBR,0xAAA5000E,0x3E8802,0x5001)
Press ESC in 5 seconds to skip startup.nsh or any other key to continue.
Shell>
Shell>
Shell> Image dtb=socfpga_stratix10_socdk.dtb console=ttyS0,115200 root=/dev/mmcb
lk0p2 rw rootwait
Loading driver at 0x000364F0000 EntryPoint=0x00037734EEC
Loading driver at 0x000364F0000 EntryPoint=0x00037734EEC
EFI stub: Booting Linux Kernel...
EFI stub: EFI_RNG_PROTOCOL unavailable, no randomness supplied
EFI stub: Using DTB from command line
EFI stub: Exiting boot services and installing virtual address map...
[    0.000000] Booting Linux on physical CPU 0x0000000000 [0x410fd034]
[    0.000000] Linux version 5.4.13-altera (oe-user@oe-host) (gcc version 9.2.0 (GCC)) #1 SMP PREEMPT
:13:01 UTC 2020
[    0.000000] Machine model: SoCFPGA Stratix 10 SoCDK
[    0.000000] efi: Getting EFI parameters from FDT:
[    0.000000] efi: EFI v2.70 by EDK II
```

Related Information

Running the Secure Monitor on page 13

5. Document Revision History for Intel Stratix 10 SoC UEFI Boot Loader User Guide

Document Version	Changes
2020.06.19	Updated the following sections: • *Minimum Hardware Requirements* • *Minimum Software Requirements* • *Installing the Compiler Toolchain* • *User Configuration* • *Getting the Arm Trusted Firmware Source Code* • *Building the ATF* • *Obtaining the UEFI Source Code* • *Obtaining the edk2 Platform Source Code* • *Compiling the UEFI Source Code with the Linaro Tool Chain* • *UEFI Generated Files* • *Generate a .sof file with ATF* • *Creating an SD Card Image* • *Debugging with DS* • *Booting from the UEFI Shell*
2019.03.28	• Added a new section: *Building the Secure Monitor* to describe new boot stage and secure boot. • Updated section: *UEFI Generated Files*. • Added a new section: *Running UEFI on Intel Stratix 10 Hardware*.
2017.06.19	Initial release.

Intel Corporation. All rights reserved. Agilex, Altera, Arria, Cyclone, Enpirion, Intel, the Intel logo, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered