Qualcomm Technologies, Inc.

Qualcomm Linux Boot Guide

80-70014-4 Rev. AM

July 12, 2024

Contents

Figures

Tables

1 Overview

Boot refers to the process where the Qualcomm® Linux system software is launched using a predefined set of instructions stored in the read only memory (ROM).

This system software is responsible for loading and running the operating system, as well as configuring the devices after the required software has been initialized. When this process is complete, the file systems are mounted and ready for use.

To initiate an operating system, a boot software image is loaded when the machine is powered on.

This guide offers comprehensive information about the supported boot features, the architecture of a cold boot (which starts from a power-off state), and the developer interfaces for the Qualcomm Linux system software.

Additionally, this guide provides instructions on how to develop a Unified Extensible Firmware Interface (UEFI) application, configure the DeviceTree (DT) Interfaces, debug logs, and verify system bootup. For information on UEFI, see Boot loader > UEFI.

For information on the software bringup of the boot image, see Build – Additional information > Build firmware > Build BOOT.

NOTE The Qualcomm Linux platform allows you to develop applications for QCS6490 and QCS5430.

2 Getting started

This information explains how to get started with developing your software using the Qualcomm Linux platform.

Before you begin, set up your infrastructure as described in the Qualcomm Linux Build Guide. This guide also provides information about the common build workflows.

You must use the boot build for installation, compilation, and debugging boot. This enables you to analyze and resolve booting issues effectively.

Prerequisites:

- For information on supported hardware, see Getting Started in Development Kit Quick Start.
- To build a boot, see the Build Boot section.
- For instructions on how to install QDTE, see Qualcomm Device Tree Editor (QDTE)

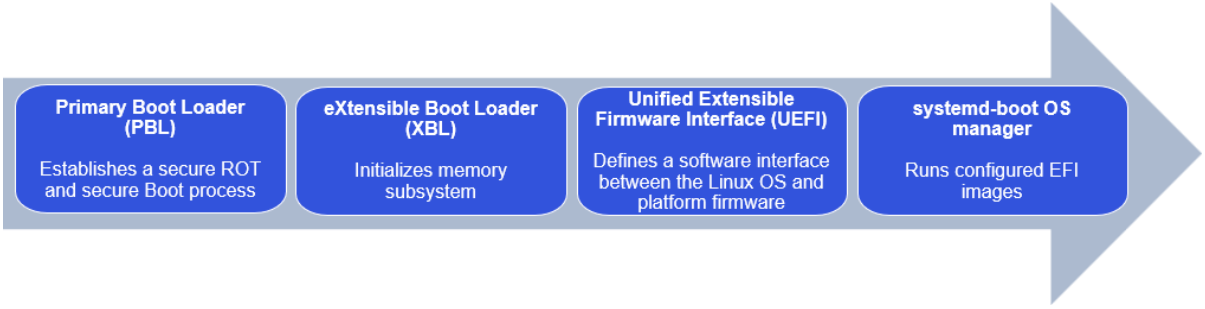
3 Architecture

The boot architecture of the Qualcomm Linux system outlines the different phases involved in the system's booting process. It offers insight into the board support package (BSP) images that are used at each stage during boot.

3.1 Boot loader

The boot loader is the software that runs when the system is powered on. It serves as an interface for loading the operating system and other required applications.

As part of the boot process, the Qualcomm Linux system includes several distinct stages of the boot loader:



Primary boot loader (PBL)

- Establishes a secure root-of-trust (RoT) and secure boot process for applications.

  For more information, see Secure boot.
- Identifies the primary storage device and loads the secondary boot loader called eXtensible boot loader (XBL).

  During the loading of the XBL image, any recoverable error causes the PBL to enter the Emergency Download (EDL) mode. For more information on how EDL mode is detected, see EDL device manager enumeration.
- Loads XBL segments into the boot internal memory.

  For more information on how PBL loads the XBL segments, see Cold boot architecture.

eXtensible Boot Loader (XBL)

- Initializes the memory subsystem, which includes buses, low-power double data rate (LPDDR), clocks, and configuration data table (CDT).
- Sets up the RAM dump to SD card, USB driver, USB charging, thermal check, power management integrated circuit (PMIC), and double data rate (DDR) functionalities.
- Loads and verifies the Qualcomm® Trusted Execution Environment (Qualcomm TEE), Qualcomm Hypervisor, and UEFI image.
- Provides the XBL configuration (XBL_CFG), a component of the cold boot flow that includes PMIC and other driver settings.
- Provides XBL with a Qualcomm-signed ELF segment to secure the external protection units (xPU).

  XBL_CFG is a standalone binary that contains platform-specific configurations and settings. The XBL_CFG driver is used by XBL to load and configure the required settings.

  The driver uses the binary to provide on-demand read access to each setting of XBL_CFG.
- Captures RAM dump if there is a crash using the following:
  - UFS on flash
  - USB on flashless

Unified Extensible Firmware Interface (UEFI)

UEFI defines a software interface between the Linux operating system (OS) and platform firmware.

The interface consists of data tables that contain platform-related information, and boot and runtime service calls that are available to Linux and boot loader. Together, these provide a standard environment for booting the Linux operating system and running the pre-boot applications.

UEFI application development and UEFI menu are currently available for users who have full access to the proprietary software shipped with Qualcomm Linux. For more information, see Boot loader in Qualcomm Linux Boot Guide - Addendum.

systemd-boot OS manager

The systemd-boot is an OS manager that runs the configured extensible firmware interface (EFI) images. It is designed to support systems with UEFI firmware exclusively and does the following:

- Loads boot entry information from the EFI system partition (ESP), typically mounted at `/efi/`, `/boot/`, or `/boot/efi/` during the Linux OS runtime and from the extended Boot Loader partition (XBOOTLDR) if it exists (mounted to `/boot/`).

  Configuration file fragments, kernels, initial RAM disk (initrd), and other EFI images must be located within ESP or XBOOTLDR.
- Reads simple and entirely generic boot loader configuration files; selects one file per boot loader entry. All the files must be located in the ESP.

To ensure that the Linux kernel can be directly executed as an EFI image, the kernel should be built with `CONFIG_EFI_STUB`.

For more information on EFI, see the following:

- For EFI boot stub, seehttps://docs.kernel.org/admin-guide/efi-stub.html.
- For a sample structure of the EFI partition, see Systemd-boot in Qualcomm Linux Yocto Guide.

3.2 Cold boot architecture

A cold boot refers to the process of starting the system from a power-off state. This process commences with the execution of the Primary boot loader (PBL).

The following figure shows the cold boot flow:

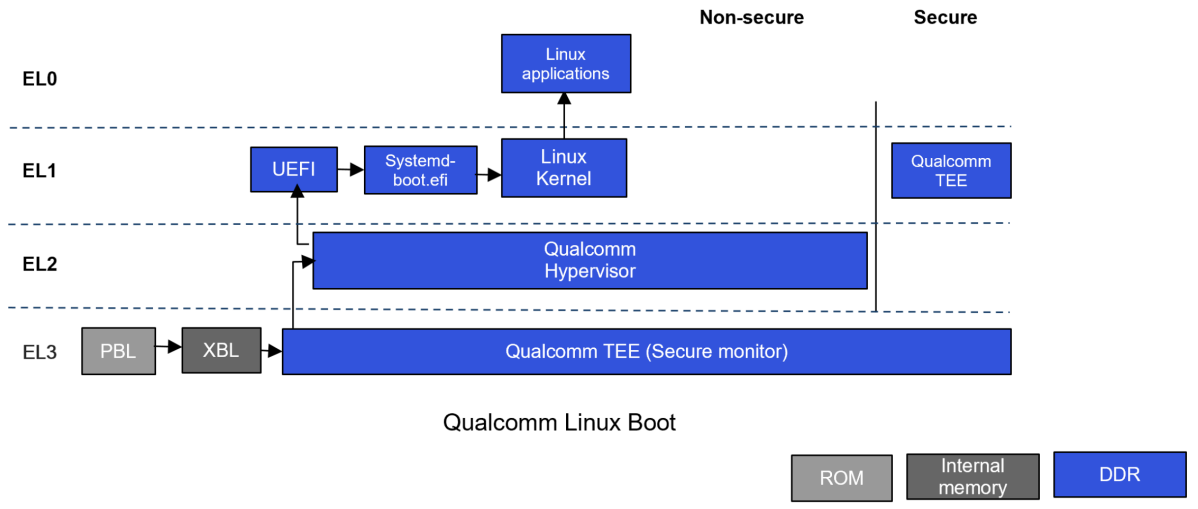


Figure 3-1 Cold boot flow

The cold boot process operates in two security modes – Secure and Nonsecure.

At the EL0, EL1, and EL2 exception levels, the boot core can operate in either Secure or Nonsecure mode while EL3 always operates in Secure mode.

For more information on the security modes, see Software Security Architecture.

The Linux OS runs in the Nonsecure EL1 state. Once the security configurations are completed in the secure EL3 state, the XBL may transition to the UEFI nonsecure EL1 state.

The following is the boot sequence that corresponds to the cold boot flow:

1. After a reset, the boot core comes out of the reset state and executes PBL. The PBL initializes the hardware clocks, CPU caches, memory management unit (MMU), and identifies the boot device based on the boot option configuration
2. The PBL loads and authenticates XBL from the boot device. The XBL runs the security configuration in EL3 Secure state and performs the following tasks:
   a. Initializes the hardware, firmware images, CPU cache, MMU, boot device, PMIC, and DDR
   b. Loads and authenticates the Qualcomm® Trusted Execution Environment (Qualcomm TEE) image from the boot device
   c. Loads and authenticates the Qualcomm Hypervisor image

d. Loads and authenticates the UEFI image

e. Executes a secure channel manager (SCM) call to jump to the Qualcomm TEE image.

NOTE SCM is the Linux kernel driver that communicates with Qualcomm TEE via a secure monitor call (SMC).

3. The Qualcomm TEE sets up the secure environment and executes the Qualcomm Hypervisor image.
4. The Qualcomm Hypervisor transfers execution to UEFI, which then loads and authenticates the systemd-boot EFI image using the Secure boot → UEFI secure boot.
5. The Systemd-boot image loads and authenticates the Linux kernel image, and transfers the execution to the Linux kernel.
6. The Linux kernel launches various Linux applications such as the bash shell.

4 Interfaces

Boot interfaces/properties are accessible through the Device Tree (DT) nodes. The DT provides a data structure that describes the system's hardware information. This information is in a format that is readable to the Image Execution Environment, which includes early firmware like XBL and UEFI. You can modify the DT properties associated with boot to customize the functionality of various device software features.

For instructions on how to modify the interfaces using the Qualcomm Device Tree editor (QDTE) tool and how to change DT properties in prebuilt device tree blob (DTB) using the QDTE tool, see Configure.

NOTE QDTE supports the option to search for property and value. To initiate a search, press **CTRL + F**.

For more information on DT, seehttps://github.com/devicetree-org.

4.1 Boot

The boot DT properties provide information on how to enable and disable user-initiated EDL in XBL. It also describes the EDL timeout and the USB cable options that can force the system into the EDL mode. These properties are used to configure the boot-related functionality.

The boot DT file path is `/boot_images/boot/Settings/Soc/<chipset>/Core/Boot/sw_boot.dtsi`.

Property name	Property description	Data type	Possible values/ value range	Device behavior
`vibration`	Turn on/off vibration at XBL exit	UINT32	▪ 0x00 ▪ 0x01	▪ ENABLE (0x01): Turns on the vibration. Default setting ▪ DISABLE (0x00): Turns off vibration
`forced_download.feature`	Enable/disable the force emergency Download mode (EDL) feature at XBL stage to flash images using USB cable along with key combination (POWER (PWR), VOL+ and VOL-keys pressed together)	UINT32	▪ 0x00 ▪ 0x01	Force EDL feature (FEDL) ▪ ENABLE (0x01): Enable user-initiated EDL in XBL. Default setting ▪ DISABLE (0x00): FEDL disabled
`forced_download.check_usb_option`	Select USB cable options to enter FEDL mode	UINT32	▪ 0x00 ▪ 0x01 ▪ 0x02	Force EDL feature configuration: ▪ CHECK_USB_D_PLUS_GROUND (0x00): The device is forced to EDL when it is powered on with a special USB cable where the D+ line is grounded. Default setting ▪ CHECK_USB_D_PLUS_HIGH_WITH_TIMEOUT (0x01): Connect to USB+ grounded during boot and within timeout, which when removed forces the device into EDL mode. ▪ Time out is provided by the `forced_download.check_dp_timeout` DT property in milliseconds (ms) ▪ CHECK_USB_REGULAR_CABLE (0x02): The device is forced to EDL when booting with a normal USB cable

4.2 PMIC

PMIC is responsible for managing all power supply requirements. This includes battery management tasks such as charging and gauging, user interfaces like flash, display, RGB, and system-on-chip (SoC) infrastructure like clocks, analog-to-digital converter (ADC), and power on (PON). You can use the DT properties to configure the PMIC XBL DT.

Certain PMIC resources can be customized in XBL through the XB_CFG image, and these customizations can be performed via a DT framework.

The PMIC DTSI files are at `boot_images/boot/Settings/Soc/<Chipset>/Core/PMIC/pm.dtsi, access.dtsi`

The table lists the PMIC DT properties:

Property name	Property description	Data type	Possible values/value range	Device behavior
s2-kpdpwr	Enable: PMIC power Key pin reset configuration	Boolean	▪ PM_TRUE ▪ PM_FALSE	The power key can be configured to reset or shut down the device when it is asserted for certain duration specified in “s1-ms” and “s2-ms”. Property to allow the DT setting to take effect. If PM_FALSE, the default hardware settings are programmed.
	`reset-type`: Select the reset type	UINT8	▪ PM_WARM_RESET ▪ PM_HARD_RESET ▪ PM_SHUTDOWN	▪ WARM_RESET is used for crash dump collection. ▪ HARD_RESET is used for rebooting the device. ▪ SHUTDOWN is used for shutting down the device
	`s1-ms`: Assertion time value must choose from possible values.	UINT32	▪ 0 ▪ 32 ▪ 56 ▪ 80 ▪ 128 ▪ 184 ▪ 272 ▪ 408 ▪ 608 ▪ 904 ▪ 1352 ▪ 2048 ▪ 3072 ▪ 4480 ▪ 6720 ▪ 10256	KPDPWR_N_RESET_S1_TIMER in ms.
	`s2-ms`: Assertion time value must choose from possible values.	UINT32	▪ 0 ▪ 10 ▪ 50 ▪ 100 ▪ 250 ▪ 500 ▪ 1000 ▪ 2000	KPDPWR_N_RESET_S2_TIMER in ms.
s2-kpdpwr-resin	Enable: PMIC power and Resin key combination reset configurations	Boolean	▪ PM_TRUE ▪ PM_FALSE	Both power and Resin key combinations can be configured to reset or shut down the device when both are asserted together for certain duration specified in the “s1-ms” and “s2-ms”. The property allows the DT setting to take effect. If PM_FALSE, the hardware default settings are programmed.
	`reset-type`: Select the reset type	UINT32	▪ PM_WARM_RESET ▪ PM_HARD_RESET ▪ PM_SHUTDOWN	▪ `WARM_RESET` is used for crash dump collection. ▪ `HARD_RESET` is used for rebooting the device. ▪ `SHUTDOWN` is used for shutting down the device
	`s1-ms`: Assertion time value must choose from possible values.	UINT32	▪ 0 ▪ 32	`RESIN_AND_KPDPWR_RESET_S1_TIMER` in ms

Property name	Property description	Data type	Possible values/value range	Device behavior
			▪ 56 ▪ 80 ▪ 128 ▪ 184 ▪ 272 ▪ 408 ▪ 608 ▪ 904 ▪ 1352 ▪ 2048 ▪ 3072 ▪ 4480 ▪ 6720 ▪ 10256	
	`s2-ms`: Assertion time value must choose from possible values.	UINT32	▪ 0 ▪ 10 ▪ 50 ▪ 100 ▪ 250 ▪ 500 ▪ 1000 ▪ 2000	RESIN_AND_KPDPWR_RESET_S2_TIMER in ms
s2-resin	Enable: PMIC Resin key pin used as volume down key.	Boolean	▪ PM_TRUE ▪ PM_FALSE	The resin key can be configured to reset or shut down the device when it is asserted for certain duration specified in the “s1-ms” and “s2-ms”. The property allows the DT setting to take into effect. If PM_FALSE, the hardware default settings are programmed.
	`reset-type`: Select the reset type	UINT8	▪ PM_WARM_RESET ▪ PM_HARD_RESET ▪ PM_SHUTDOWN	▪ WARM_RESET is used for crash dump collection. ▪ HARD_RESET is used for rebooting the device. ▪ SHUTDOWN is used for shutting down the device
	`s1-ms`: Assertion time value must choose from possible values.	UINT32	▪ 0 ▪ 32 ▪ 56 ▪ 80 ▪ 128 ▪ 184 ▪ 272 ▪ 408 ▪ 608 ▪ 904 ▪ 1352 ▪ 2048 ▪ 3072 ▪ 4480	RESIN_N_RESET_S1_TIMER in ms

Property name	Property description	Data type	Possible values/value range	Device behavior
			▪ 6720 ▪ 10256	
	`s2-ms`: Assertion time value must choose from possible values.	UINT32	▪ 0 ▪ 10 ▪ 50 ▪ 100 ▪ 250 ▪ 500 ▪ 1000 ▪ 2000	RESIN_N_RESET_S2_TIMER in msec
s3-reset	Enable: Configure the S3 reset	Boolean	▪ PM_TRUE ▪ PM_FALSE	▪ S3 reset is a fail-safe reset or factory reset if the device is stuck and cannot be restored with S2 reset. ▪ S3 reset is triggered when the key is asserted for the duration specified in the “timervalue-ms”. ▪ The property allows the DT setting to take effect. If PM_FALSE, the default hardware settings are programmed.
	`s3-src`: Select the reset source	UINT8	▪ PM_PON_RESET_SOURCE_KPDPWR ▪ PM_PON_RESET_SOURCE_RESIN ▪ PM_PON_RESET_SOURCE_RESIN_AND_KPDPWR ▪ PM_PON_RESET_SOURCE_RESIN_OR_KPDPWR	S3 reset source can be power key alone or Resin key alone, or a combination of both power and Resin.
	`timervalue-ms`: Assertion time value must choose from possible values.	UINT32	▪ 0 (Immediate) ▪ 8 ▪ 16 ▪ 32 ▪ 63 ▪ 125 ▪ 240 ▪ 500 ▪ 1000 ▪ 2000 ▪ 4000 ▪ 8000 ▪ 16000 ▪ 32000 ▪ 64000 ▪ 128000	RESET_S3_TIMER_n in ms

Property name	Property description	Data type	Possible values/value range	Device behavior
uvlo-config	`PMIC_INDEX`: Configure the under voltage lockout threshold of various PMICs.	UINT8	▪ PMIC_A (PMK7325) ▪ PMIC_B (PM7325) ▪ PMIC_C (PM7350C) ▪ PMIC_D (PM7325B) ▪ PMIC_E (PMR735A) ▪ PMIC_I (PM7250B) ▪ PMIC_K (PMG1110)	When the input voltage of PMIC reaches under the voltage lockout (UVLO) threshold, the device shuts down abruptly. For each PMIC present in the board, configure the UVLO threshold. The pmic_index is used to select the PMIC for which the threshold should be changed.
	`uvlo_thresh`	UINT32	mV	UVLO threshold in millivolts
	`uvlo_hyst`	UINT32	mV	UVLO hysteresis in millivolts
	`uvlo_enable`	Boolean	PM_ENABLE PM_DISABLE	The property allows the DT setting to take into effect. If PM_ DISABLE, the hardware default settings are programmed.
ovlo-config	`PMIC_INDEX`: Configure the over voltage Lockout threshold of various PMICs.	PMIC_INDEX	▪ PMIC_A (PMK7325) ▪ PMIC_B (PM7325) ▪ PMIC_C (PM7350C) ▪ PMIC_D (PM7325B) ▪ PMIC_E (PMR735A) ▪ PMIC_I (PM7250B) ▪ PMIC_K (PMG1110)	When the input voltage of PMIC reaches over the voltage lockout (OVLO) threshold, the device shuts down abruptly. For each PMIC present in the board, configure the OVLO threshold. The pmic_index is used to select the PMIC for which the threshold has to be changed.
	`ovlo_thresh`	–	mV	OVLO threshold in millivolts
	`ovlo_hyst`	–	mV	OVLO hysteresis in millivolts
	`ovlo_enable`		▪ PM_ ENABLE ▪ PM_ DISABLE	The property allows the DT setting to take into effect. If PM_ DISABLE, the hardware default settings are programmed.
Long-pwrkey-dbnc-chk	`dbnc-time-ms`: The power key debounces time to check for a valid keypress to boot up the device.	UINT32	msec	If the power key is not pressed for the configured “dbnc-time-ms” duration, the device shuts down in XBL or UEFI based on the config “chk-at” during boot up.
	`chk-at`: Check for keypress at the XBL or UEFI stage	UINT8	0, 1, 2	▪ 0 = Do not check ▪ 1 = Check-in XBL Loader stage ▪ 2 = Check-in XBL Core (UEFI)
sw-config To modify PMIC software configuration at the boot up time for various LDOs, SMPSs, Clocks, GPIOs	Verbose: Select PON log level	UINT32	▪ PM_EVENT_LOG_LEVEL_MIN ▪ PM_EVENT_LOG_LEVEL_VERBOSE	PON log indicates the various reasons for device power-on, OFF, and faults. ▪ Minimal version of PON logs printed in UART logs ▪ Verbose parsed PON logs printed in UART logs ▪ Verbose parsed PON logs along with Raw SDAM register data printed in UART logs

Property name	Property description	Data type	Possible values/value range	Device behavior
			▪ PM_EVENT_LOG_LEVEL_RAW DATA ▪ PM_EVENT_LOG_LEVEL_MAX	
	PM_LDO_SET_ENABLE: Enable the LDO	UINT32	▪ PM_TRUE ▪ PM_FALSE	See SMPS/LDO/CLK sample code
	PM_LDO_SET_VOLT: Set LDO for the voltage	UINT32	mV	
	PM_LDO_SET_MODE: Set the LDO mode	UINT32	▪ 0: Low power ▪ 1 : Bypass ▪ 3 : Normal power mode ▪ 4 : Retention mode.	
	PM_LDO_SET_PD_CTR: Set the pull down on the LDO	UINT32	▪ PM_TRUE ▪ PM_FALSE	
	PM_LDO_SET_PIN_CTRL: Enable pin control for the LDO	UINT32	▪ 0 : HWEN0 ▪ 1 : HWEN1 ▪ 2 : HWEN2	
	PM_LDO_SET_OCP_BROADCAST: Enable LDO OCP broadcast so that the device can shut down during OCP	UINT32	▪ PM_TRUE ▪ PM_FALSE	
	PM_LDO_SET_AHC: Enable LDO automatic head room control.	UINT32	▪ PM_TRUE ▪ PM_FALSE	
	PM_LDO_SET_ULS: Set the upper limit to the LDO voltage	UINT32	mV	
	PM_SMPS_SET_AHC_HR: Configure the SMPS AHC reserved head room	UINT32	▪ 0x0 : 0 mV ▪ 0x1 : 8 mV ▪ 0x2 : 16 mV ▪ 0x3 : 24 mV ▪ 0x4 : 32 mV (DEFAULT) ▪ 0x5 : 40 mV ▪ 0x6 : 48 mV ▪ 0x7 : 56 mV	
	PM_SMPS_SET_ENABLE: Enable the SMPS	UINT32	▪ PM_TRUE ▪ PM_FALSE	
	PM_SMPS_SET_VOLT: Set SMPS for the voltage	UINT32	mV	
	PM_SMPS_SET_MODE: Set the SMPS mode	UINT32	▪ 0 : Low power ▪ 2 : Auto mode ▪ 3 : Normal power mode ▪ 4 : Retention mode	

Property name	Property description	Data type	Possible values/value range	Device behavior
	`PM_SMPS_SET_PD_CTRL`: Set the pull down on the SMPS	UINT32	▪ PM_TRUE ▪ PM_FALSE	
	`PM_SMPS_SET_PIN_CTRL`: Enable Pin control for the SMPS	UINT32	▪ 0 : HWEN0 ▪ 1 : HWEN1 ▪ 2 : HWEN2	
	`PM_SMPS_SET_OCP_BROADCAST`: Enable SMPS OCP broadcast so that the device can shut down during OCP.	UINT32	▪ PM_TRUE ▪ PM_FALSE	
	`T PM_SMPS_SET_AHC`: Enable SMPS automatic head room control.	UINT32	▪ PM_TRUE ▪ PM_FALSE	
	`PM_SMPS_SET_ULS`: Set the upper limit to the SMPS voltage.	UINT32	mV	
	`PM_GPIO_SET_ENABLE`: Enable GPIO	UINT32	▪ PM_TRUE ▪ PM_FALSE	
	`PM_GPIO_SET_CFG_MODE`: Configure GPIO mode	UINT32	▪ 0 - PM_GPIO_DIG_IN ▪ 1- PM_GPIO_DIG_OUT ▪ 2- PM_GPIO_DIG_IN_OUT ▪ 3- PM_GPIO_ANA_PASS_THRU	
	`PM_GPIO_SET_OUTPUT_LVL`: Set GPIO output level	UINT32	▪ 0 - Level of pin is low ▪ 1 - Level of pin is high ▪ 2 - Level of pin is Hi-Z	
	`PM_GPIO_SET_VOLT_SRC`: Select GPIO voltage source as 1.8 V or 3.6 V	UINT32	▪ 0 - PM_GPIO_VIN0 ▪ 1 - PM_GPIO_VIN1	
	`PM_GPIO_SET_OUT_BUFF_CONFIG`: Select GPIO output buffer config.	UINT32	▪ 0 - CMOS output. ▪ 1 -Open drain NMOS output ▪ 2 - Open drain PMOS output	
	`PM_GPIO_SET_OUT_SRC_CFG`: Select GPIO output source config	UINT32	▪ 0 - Ground ▪ 1 - Paired GPIO ▪ 2 - Special function 1 ▪ 3 - Special function 2	

Property name	Property description	Data type	Possible values/value range	Device behavior
			▪ 4 - Special function 3 ▪ 5 - Special function 4 ▪ 6 - D-test 1 ▪ 7 - D-test 2 ▪ 8 - D-test 3 ▪ 9 - D-test 4	
	`PM_GPIO_SET_OUT_DRV_STR`: Select GPIO pin drive strength	UINT32	▪ 0 - Output buffer strength low. ▪ 1 - Output buffer strength medium. ▪ 2 - Output buffer strength high.	
	`PM_GPIO_SET_PULL_SEL`: Select GPIO pull settings	UINT32	▪ 0 - 30 uA constant pulls up ▪ 1 - 1.5 uA constant pulls up ▪ 2 - 31.5 uA constant pulls up ▪ 3 - 1.5 uA constant pulls up combined with 30 uA boost ▪ 4 - 10 uA constant pull down ▪ 5 - No pull	
	`PM_SET_DELAY`: Add delay between any 2 API calls	UINT32	▪ Delay in uS.	
	`PM_CLK_ENABLE`: Enable the clocks like ln_bb_clk, rf_clk.	UINT32	▪ PM_TRUE ▪ PM_FALSE	
	`PM_CLK_DRV_STR`: Select the drive strength of the clock	UINT32	▪ 0-63	
	`PM_SPMI_CLK_DATA_CFG`: Select the SPMI drive strength	UINT32	▪ 0x0: LOW10PF ▪ 0x1: MID20PF ▪ 0x2: HIGH40PF ▪ 0x3: VERYHIGH50PF	
	`PM_BUSID`: Select SPMI bus	UINT32	PM_BUSID_0	Must be PM_BUSID_0
	`PMIC_Index`: Select the PMIC	UINT32	▪ PMIC_A (PMK7325) ▪ PMIC_B (PM7325) ▪ PMIC_C (PM7350C) ▪ PMIC_D (PM7325B)	See `/boot_images/boot/Settings/ Include/pm_defines.h`

Property name	Property description	Data type	Possible values/value range	Device behavior
			▪ PMIC_E (PMR735A) ▪ PMIC_I (PM7250B) ▪ PMIC_K (PMG1110)	

	`PMIC_Peripheral`: Select the LDO, SMPS, clock, GPIO number	UINT32	▪ PM_LDO_x
			▪ PM_SMPS_x
			▪ PM_CLK_x
			▪ PM_GPIO_x
				See:
				`/boot_images/boot/Settings/Include/ pm_defines.h`
				`/boot_images/boot/QcomPkg/`
				Include`/api/pmic/pm/pm_gpio.h` (for GPIO enums)

	API parameter argument	UINT32	PM_TRUE, PM_FALSE OR Integer OR Enum	See`/boot_images/boot/Settings/Include/ pm_defines.h` `/boot_images/boot/QcomPkg/` Include`/api/pmic/pm/pm_gpio.h` (for GPIO enums)
Access (access.dtsi) To modify/enable the write access to PMIC peripheral for different sub systems like AOP, APSS, UEFI, TZ, MSS BUSID: Select SPMI bus	`UINT32`	▪ PM_BUSID_0 ▪ PM_BUSID_1	▪ PM_BUSID_0 – For SPMI0 bus slave devices ▪ PM_BUSID_1 – For SPMI1 bus slave devices	–
	`SLAVEID`: Select the slave ID	UINT32	▪ 0 - PMIC_A (PMK7325) ▪ 1 - PMIC_B (PM7325) ▪ 2 - PMIC_C (PM7350C) ▪ 3 - PMIC_D (PM7325B) ▪ 4 - PMIC_E (PMR735A) ▪ 8 - PMIC_I (PM7250B) ▪ 9 - PMIC_I (PM7250B)	–
	`PERIPH (PPID)`: Select the peripheral within PMIC	UINT32	PMIC Peripheral Register base 0x1 to 0xFF	PMIC Peripheral Register base. Example: ▪ For PM7325, GPIO_01 PPID is 0x88 ▪ For PMK7325, PON_PBS PPID is 0x08
	`OPERATION`: Add or remove the channel with permission	UINT32	▪ APPEND ▪ REMOVE ▪ APPEND_WITH_IRQ	▪ APPEND: A channel is added for the PMIC peripheral with the defined DRV having the write access permission. The default channel setting is retained. ▪ REMOVE: The channel is removed for the PMIC peripheral with the defined DRV. ▪ APPEND_WITH_IRQ: A channel is added for the PMIC peripheral with the defined DRV, which overrides the default IRQ owner DRV. In this case, the write access DRV is retained from the default setting. ▪ NOTE: There can only be one IRQ DRV owner, but multiple write-access DRVs is allowed.

Property name	Property description	Data type	Possible values/value range	Device behavior
Charger	pmic-index-charger: Select the charger PMIC index	UINT32	PMIC_D or PMIC_I	Index of primary charger PMIC
	`Dead-battery-threshold`: Set the dead battery threshold	UINT32	2800-3300	DBC threshold in millivolts. XBL allows to boot to UEFI post the event
	`ichg-fs-cfg-enable`: Increase the charge current range up to 20 A	UINT8	▪ PM_TRUE ▪ PM_FALSE	Flag to check if the full scale charging current setting should be applied
	`ichg-fs-cfg-value`	UINT32	▪ 0: 10 A ; ▪ 1 : 20 A	Supported full-scale charging current - 0: 10 A ; 1 : 20 A
	`pm-chg-batt-cfg-enable`: Select the battery type as 1 S or 2 S	UINT8	▪ PM_TRUE ▪ PM_FALSE	Flag to check if the 1S/2S battery configuration should be applied
	`pm-chg-batt-cfg-is_1s_battery`	UINT8	▪ PM_TRUE ▪ PM_FALSE	To indicate if a 1S or 2S battery is present
	`dbg-board-id-cfg-min-id-ohms`: Minimum batt_id in ohms for debug board detection	UINT32	2000	Battery less platform with debug board, the batt_id should be within this min and max range to skip the battery charging
	`dbg-board-id-cfg-max-id-ohms`: Maximum batt_id in ohms for debug board detection	UINT32	14000	–
	`afp-cfg-too-hot-threshold`: Set the AFP threshold temperature for a hot temperature	UINT32	Temperature in degree C	If the battery temperature crosses any of the AFP thresholds, the device shuts down automatically. Example: 75 C
	`afp-cfg-too-hot-enable`: Enable/ Disable AFP for hot temperature threshold config	UINT8	▪ PM_TRUE ▪ PM_FALSE	Enable/Disable AFP for too hot temperature. If PM_FALSE, the hardware default settings are programmed.
	`afp-cfg-too-cold-threshold`: AFP threshold temp for too cold temperature	UINT32	Temperature in degree C	Example: -20 C
	`afp-cfg-too-cold-enable`: Enable/ Disable AFP for cold temperature threshold config	UINT8	▪ PM_TRUE ▪ PM_FALSE	Enable/Disable AFP for too cold temperature. If PM_FALSE, the hardware default settings are programmed
	`no-batt-cfg-enable`: Configure the battery-less platform	UINT8	▪ PM_TRUE ▪ PM_FALSE	If PM_FALSE, the device shuts down automatically if battery is not detected.
	`no-batt-cfg-boot-without-batt`: Continue to boot when the battery is not detected	UINT8	▪ PM_TRUE ▪ PM_FALSE	If PM_FALSE, the device shuts down automatically if battery is not detected.
	`no-batt-cfg-icl-value_ma`: Set the input current limit for battery less platform	UINT32	0 - 5000	Input current limit value in mA
	`float-voltage-cfg-enable`: Configure max battery voltage	UINT8	▪ PM_TRUE ▪ PM_FALSE	If PM_FALSE, the hardware default settings are programmed
	`float-voltage-cfg-value-mv`	UINT8	0 -4500	Max battery voltage for charging
	`pre-charging-current-cfg-enable`: Configure the precharge current	UINT8	▪ PM_TRUE ▪ PM_FALSE	If PM_FALSE, the hardware default settings are programmed
	`pre-charging-current-cfg-value-ma`	UINT32	0-600	Precharge current in mA when battery voltage is < Vsys_min Ex: 3 V
	`fast-charging-current-enable`: Configure the fast-charge current	UINT8	▪ PM_TRUE ▪ PM_FALSE	If PM_FALSE, the hardware default settings are programmed

Property name	Property description	Data type	Possible values/value range	Device behavior
	fast-charging-current-value-ma	UINT32	0-10000	–
	usbin-input-current-enable: Configure the USB input current limit	UINT8	▪ PM_TRUE ▪ PM_FALSE	If PM_FALSE, the hardware default settings are programmed
	usbin-input-current-value-ma	UINT32	0-5000	USB IN current configuration
	dam-cable-chg-enable: Enable charging for DAM cable	UINT8	▪ PM_TRUE ▪ PM_FALSE	If PM_FALSE, the hardware default settings are programmed. By default, the hardware disables the DAM cable charging
	dam-cable-aicl-enable: Enable AICL for DAM cable	UINT8	▪ PM_TRUE ▪ PM_FALSE	If PM_FALSE, the hardware default settings are programmed. By default, the hardware disables the DAM cable AICL.
	slave-charger-present: Indicate the Qualcomm Slave charger presence	UINT8	▪ PM_TRUE ▪ PM_FALSE	Slave charger settings

SMPS/LDO/CLK sample code

```
driver-post-init = <
      // PM_SMPS_SET_AHC_HR        PM_BUSID_0   PMIC_C  PM_SMPS_1    0x57
      // PM_SMPS_SET_ULS           PM_BUSID_0   PMIC_F  PM_SMPS_6
1375000          /*The Voltage is in microVolts*/
      // PM_LDO_SET_ULS            PM_BUSID_0   PMIC_F  PM_SMPS_6
1375000          /*The Voltage is in microVolts*/
      // PM_CLK_ENABLE             PM_BUSID_0   PMIC_A  PM_CLK_RF_1  PM_TRUE
      // PM_SMPS_SET_VOLT          PM_BUSID_0   PMIC_B  PM_SMPS_1    1200
      // PM_SMPS_SET_AHC           PM_BUSID_0   PMIC_B  PM_SMPS_1    PM_FALSE
      // PM_LDO_SET_AHC            PM_BUSID_0   PMIC_B  PM_LDO_1     PM_FALSE
      // PM_LDO_SET_VOLT           PM_BUSID_0   PMIC_B  PM_LDO_1     1200
      // PM_LDO_SET_MODE           PM_BUSID_0   PMIC_B  PM_LDO_1
PM_SW_MODE_NPM
      // PM_SMPS_SET_MODE          PM_BUSID_0   PMIC_B  PM_SMPS_1
PM_SW_MODE_NPM
      // PM_SMPS_SET_PD_CTRL       PM_BUSID_0   PMIC_B  PM_SMPS_1
PM_TRUE
      // PM_SMPS_SET_PIN_CTRL      PM_BUSID_0   PMIC_B  PM_SMPS_1
PM_TRUE
      // PM_SMPS_SET_OCP_BROADCAST PM_BUSID_0   PMIC_B  PM_SMPS_1    PM_TRUE
      // PM_LDO_SET_PD_CTRL        PM_BUSID_0   PMIC_B  PM_LDO_1
PM_TRUE
      // PM_LDO_SET_PIN_CTRL       PM_BUSID_0   PMIC_B  PM_LDO_1     PM_TRUE
      // PM_LDO_SET_OCP_BROADCAST  PM_BUSID_0   PMIC_B  PM_LDO_1     PM_TRUE
      // PM_CLK_DRV_STR            PM_BUSID_0   PMIC_A  PM_CLK_RF_1  3
      // PM_SMPS_SET_ENABLE        PM_BUSID_0   PMIC_B  PM_SMPS_1
PM_TRUE
      // PM_LDO_SET_ENABLE         PM_BUSID_0   PMIC_B  PM_LDO_1     PM_TRUE
      // PM_SPMI_CLK_DATA_CFG      PM_BUSID_0   PMIC_A  0x2
0x2              /*BUS_ID, PMIC_ID, CLK buff Config, DATA buff Config*/
```

```
    PM_DELAY(10)
  > ;
```

4.3 TLMM

The top-level mode multiplexer (TLMM) facilitates the multiplexing of general-purpose input/output (GPIO) and alternate functions driven by hardware. The DT properties for TLMM allow you to configure the GPIOs during the boot process.

File path:

- `\boot_images\boot\Settings\Soc\<Chipset>\Core\SocInfra\TLMM\<Chipset>-pinctrl.dtsi`
- `\boot_images\boot\Settings\Soc\<Chipset>\Core\SocInfra\TLMM\tlmm.dtsi`

The table lists the properties that describe how to configure the default value of a GPIO based on the requirements of the board.

Property name	Property description	Data type	Possible values/ value range	Device behavior
QCOM, sleep-config	GPIO configuration settings as defined in `<Chipset>-pinctrl.dtsi`. This is applied on device boot up.	UINT32	Pin direction: ▪ GPIO_INPUT – 0x1 ▪ GPIO_OUTPUT – 0x2 ▪ Pin pull configuration: ▪ GPIO_PULL_DOWN – 0x4 ▪ GPIO_PULL_UP – 0x8 ▪ GPIO_NO_PULL – 0x10 ▪ GPIO_KEEPER – 0x20 ▪ Pin output: ▪ GPIO_OUT_LOW – 0x40 ▪ GPIO_OUT_HIGH – 0x80 ▪ GPIO_PRG_YES – 0x100 ▪ GPIO_PRG_NO – 0x000	▪ The default value is GPIO_INPUT ▫ GPIO_INPUT: Allows the state of an input pin to be read ▫ GPIO_OUTPUT: Controls the output state of an output pin ▪ The default value is GPIO_PULL_DOWN. ▫ GPIO_PULL_DOWN – Logic 0, consider as connected to Ground ▫ GPIO_PULL_UP – Logic 1, connected to Vdd supply ▫ GPIO_NO_PULL – Floating/ High Impedance state ▫ GPIO_KEEPER – Maintain the previous state of GPIO. When a SoC enters the deepest power-saving mode, this configuration is applied. ▪ The default value is GPIO_OUT_LOW GPIO_OUT_HIGH – Logic high, consider as connected to Vdd. ▪ The default value is GPIO_PRG_NO. GPIO_PRG_YES - Ensures that any unused GPIOs remain in a low-power state after bootup Example:

Property name	Property description	Data type	Possible values/ value range	Device behavior
				(GPIO_INPUT | GPIO_PULL_DOWN | GPIO_OUT_LOW | GPIO_PRG_NO) /* PIN 10 */
Compatible	Contains a string that points to compatible chipset. This is read only. Example: compatible = "qcom,<chipset>-pinctrl"	String	–	–
reg	Represents the GPIO base address and size. This is read only. In reg property tuple, the first index contains the base address, and the second index contains the size. Example: reg = <0xf100000 0x100000>;	UINT32	–	–
ngpios	Number of GPIO pins in chipset. This is read only. Example: ngpios = <175>;	UINT32	–	–
width	Each GPIO pin has its own set of control registers. Width indicates pin to pin register offset. This is read only. Example: width = <0x1000>;	–	–	–
id	Hardware instance of GPIO Pad ID This is read only. Example: id = <0x0>;	UINT32	–	–

Property name	Property description	Data type	Possible values/ value range	Device behavior
version	GPIO driver version. This is read only. Driver 1.0 refers as 0x1. Example: `version = <0x1>;`	UINT32	–	–
gpio-controller	Identifier to represent the connected device as a GPIO device. This is read only.	String	–	–
phandles for pin configurations	Mux configuration that is used to configure the alternative functionality of GPIOs. For more information, see Pin descriptions. Example: `sdc4_data_1: sdc4_data_1 { mux = <13 3>; };` Here, 13 points to the GPIO number and 3 points to the alternate function selection.	–	`sdc4_data_1: sdc4_data_1 { mux = <13 3>; };`	On bootup, the GPIO configures to alternate functionality of GPIO.

4.4 Buses

Buses facilitate data transfer among various system components. The DT properties allow for the configuration of the Qualcomm universal peripheral (QUP) v3 serial interfaces device nodes.

The QUP v3 supports interintegrated circuit (I2C), serial peripheral interface (SPI), and universal asynchronous receiver-transmitter (UART) serial engines. The DT properties include the active and sleep settings for these serial engines.

For more information on QUP v3 and its peripheral interfaces, see Overview of peripheral devices.

The serial engine protocol and active configurations can be applied when the firmware is loaded onto the serial engine.

The following files can be configured:

- `boot_images\boot\Settings\Soc\<chipset>\Core\Buses\qup_common \<chipset>-qupv3-pinctrl.dtsi`
- `boot_images\boot\Settings\Soc\<chipset>\Core\Buses\qup_common \<chipset>-qupv3.dtsi`

The table lists the DT properties:

NOTE In `&top_qup`**k**`_se`**n**, **k** represents the QUP number and **n** represents the serial engine number.

Property name	Property description	Data Type	Possible values/ value range	Device behavior
status = "disabled";	Status control to enable/disable serial interfaces supported for QUP v3 serial engine (I2C/SPI/ UART) protocols.	String	▪ Okay ▪ Disabled	Enables or disables the serial engine.
&top_qup**k**_se**n**_i2c_active	GPIOs active settings for I2C serial engines	UNIT32-array	Reference to a node label	This sets active GPIO configurations for the I2C protocol.
&top_qup**k**_se**n**_i2c_sleep	GPIOs sleep settings for I2C	UNIT32-array	Reference to a node label	This sets sleep GPIO configurations for the I2C protocol.
&top_qup**k**_se**n**_spi_active	GPIOs active settings for SPI	UNIT32-array	Reference to a node label	This sets active GPIO configurations for the SPI protocol.
&top_qup**k**_se**n**_spi_sleep	GPIOs sleep settings for SPI	UNIT32-array	Reference to a node label	These sets sleep GPIO configurations for the SPI protocol.
&top_qup**k**_se**n**_uart_active	GPIOs active settings for UART	UNIT32-array	Reference to a node label	This sets active GPIO configurations for the UART protocol.
&top_qup**k**_se**n**_uart_sleep	GPIOs sleep settings for UART	UNIT32-array	Reference to a node label	These sets sleep GPIO configurations for the UART protocol.

Sample configuration

```
TOP_QUP_0{
/*TOP_QUP_0_SE_0 Instance */
               TOP_QUP_0_SE_0{
               status = "disabled"; /* status to enable/disable SE0 Node */
               pinctrl-0  = <&top_qup0_se0_i2c_active>;
                pinctrl-1  = <&top_qup0_se0_i2c_sleep>;
               pinctrl-2  = <&top_qup0_se0_spi_active>;
               pinctrl-3  = <&top_qup0_se0_spi_sleep>;
               pinctrl-4  = <&top_qup0_se0_uart_active>;
               pinctrl-5  = <&top_qup0_se0_uart_sleep>;
```

4.5 USB

The DT properties for USB allow you to adjust the USB signal quality.

For USB features and architecture, see USB.

The DT file path is at `<Boot_Image>/Settings/Soc/<Chipset>/Core/WiredConnectivity/USB/usb_sw_cfg.dtsi`.

Property name	Property description	Data type	Possible values/ value range	Device behavior
path=/soc/usb0/ hs_phy_cfg	This property applies to the primary USB controller high-speed PHY (HSPHY) for configuring the USB signal quality.	UINT32-array	▪ This property is an array of address, value pairs <addr, val> ▪ addr is 4 bytes in length. It can have 1 of the 4 values, range: [0x88E306C, 0x88E3070, 0x88E3074, 0x88E3078] ▪ val is of 1 byte in length, range: 0x00 to 0xFF	By tuning this property, the USB signal quality of the primary USB HSPHY is improved.
path=/soc/usb0/ ss_phy_cfg	This property applies to the primary USB controller SuperSpeed PHY (SSPHY) for configuring the signal quality.	UINT32-array	▪ This property is an array of address, value pairs <addr, val> ▪ addr is 4 bytes in length. Range: [0x088E8000, 0x088EB000] ▪ val is of 1 byte in length, range: 0x00 to 0xFF	By tuning this property, the USB signal quality for primary USB SSPHY is improved.
path=/soc/usb1/ hs_phy_cfg	This property applies to the secondary USB controller high-speed PHY (HSPHY) for configuring the USB signal quality.	UINT32-array	▪ This property is an array of address, value pairs <addr, val> ▪ addr is 4 bytes in length. It can have one of the four values, range: [0x88E406C, 0x88E4070, 0x88E4074, 0x88E4078] ▪ val is of 1 byte in length, range: 0x00 to 0xFF	By tuning this property, the USB signal quality for the secondary USB HSPHY is improved.

4.6 Storage

To configure DT properties for storage, see Bootloader/UEFI device tree.

5 Tools

The developer tools for the boot process include Qualcomm Device Tree editor (QDTE) and Fastboot. QDTE is a GUI editor tool that can be accessed through the Qualcomm Software Center (QSC). Fastboot, on the other hand, is a diagnostic tool that offers a Recovery mode known as the Fastboot mode.

For more information on QSC, see Qualcomm Software Center User Guide.

5.1 Qualcomm Device Tree Editor (QDTE)

QDTE is a tool used to configure the DTB by editing the `xbl_config.elf` file.

You can download QDTE from QSC.

NOTE Currently, QDTE v1.3.0 is supported on Windows 10 and 11 host machines and also supported on the Ubuntu 22 Linux host machine.

QDTE offers the following features:

- Works with the `xbl_config.elf` file.
- Allows for experimentation with configuration settings in a test environment.
- Opens/reads a DTB file directly or parse an `xbl_config.elf` file and extracts the embedded DTB files. In the latter case, when the file is saved, the `xbl_config.elf` file is regenerated with signing.

NOTE Do not use the QDTE tool to create the DeviceTree source/DeviceTree source inclusion (DTS/DTSI) files that represent the input source.

For more information on how to configure QDTE, see Configure.

5.2 Fastboot

The Fastboot mode allows you to flash software images such as the boot loader and helps in system recovery.

NOTE Fastboot is supported on Windows and Linux host machines.

To enter the Fastboot mode, you can either select Fastboot from UEFI menu or use a hotkey for Fastboot.

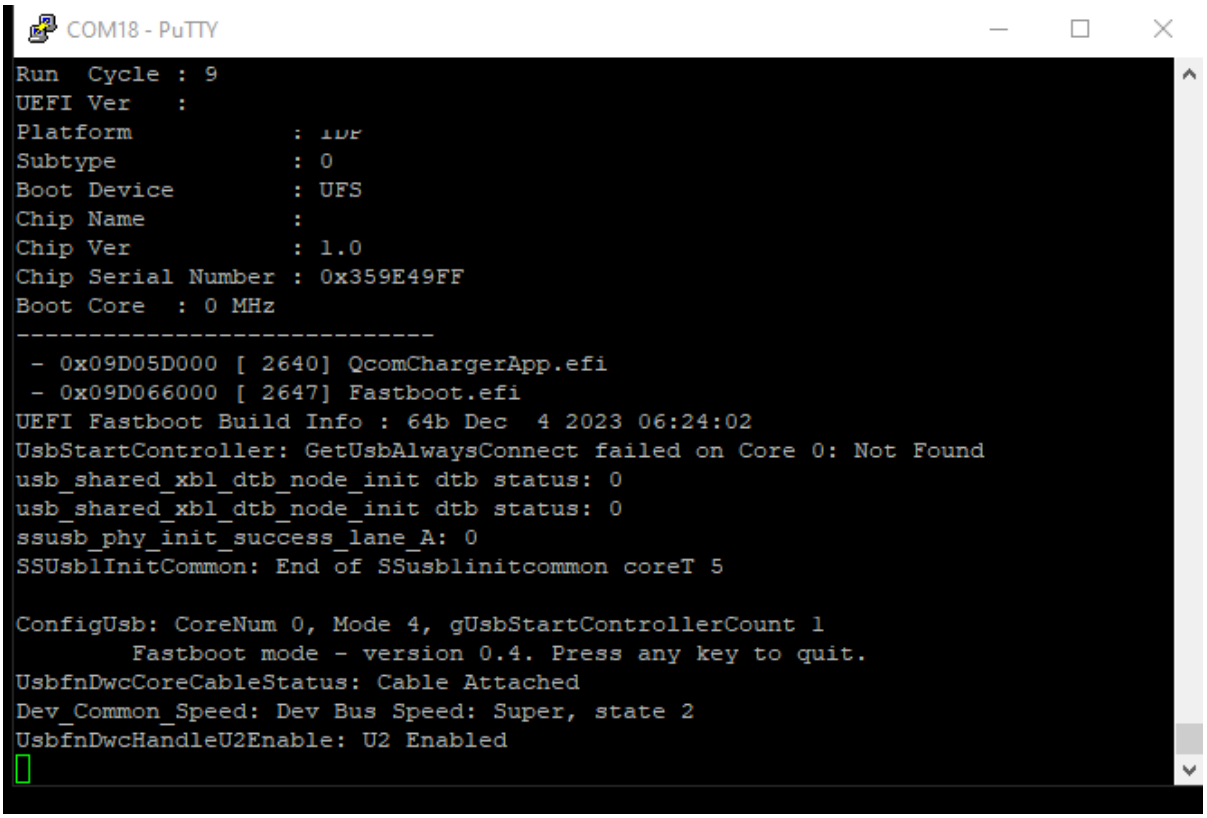
Fastboot from UEFI

This feature is currently available for users who have full access to the proprietary software shipped with Qualcomm Linux. For more information, see the Fastboot in Qualcomm Linux Boot Guide - Addendum.

Fastboot via hotkey

After powering on the device, press **VOL-** and see Status LEDs and Buttons section to switch the device into the Fastboot mode.

The figure shows the fastboot logs from the device.



Fastboot commands

When the Fastboot mode is detected, you can run the corresponding Fastboot commands.

Table 5-1 Fastboot commands

Command	Usage	Example or use case
`fastboot reboot`	Enables you to reboot the device.	A system reboot maybe required after either of the following: ▪ Flashing the software images ▪ Completing a software update
`fastboot devices`	Displays the device information if the device is connected and recognized by fastboot.	–

Table 5-1 Fastboot commands (cont.)

Command	Usage	Example or use case
`fastboot getvar <variable>`	Retrieves data from various partitions. The following are the supported `getvar` commands. ▪ `fastboot getvar partition-size:system` ▪ `fastboot getvar partition-size:xbl_a` ▪ `fastboot getvar kernel` ▪ `fastboot getvar max-download-size`	Fastboot supports multiple variables. The data from these variables can be retrieved through this command. For example, to retrieve a variable such as version, run the following Fastboot command: `fastboot getvar version`
`fastboot flash <partition name> <full path to the SW image>`	Flashes the software images.	To flash the `xbl_config.elf` image, run the following command: `fastboot flash xbl_config_a xbl_config.elf` Where: ▪ `xbl_config_a` indicates the partition label ▪ `xbl_config.elf` is the software image that is flashed
`fastboot erase`	Erases data from various partitions for a cleaner flash.	`fastboot erase xbl_config_a` erases the `xbl_config_a` partition.
`fastboot continue`	To continue with the bootup	`fastboot continue` boots up into the OS.
`fastboot oem`	Executes OEM-specific command	`fastboot oem <oem-specific command>` runs the command implemented and enabled by OEM.

Recovery with Fastboot

Fastboot serves as a recovery solution when the systemd-boot and ESP are corrupted.

The systemd-boot image boots from the EFI system partition (ESP). In such cases, Fastboot automatically initiates as a recovery option, allowing you to reflash the ESP.

6 Develop

Users with full access to the proprietary software shipped with Qualcomm Linux have the ability to develop applications based on UEFI and configure hardware platform IDs for the boards using the configuration data table (CDT). For information on how to synchronize the code for CDT and UEFI development, see Develop in Qualcomm Linux Boot Guide - Addendum.

7 Configure

The Qualcomm Device Tree editor (QDTE) is a GUI tool that is used for configuring the device tree binary (DTB) files. It can also be used to modify and configure Interfaces.

For more details on QDTE features, see Qualcomm Device Tree Editor (QDTE).

The following procedure enables you to configure the DTB files using QDTE.

NOTE This procedure is only applicable for XBL and UEFI DTB.

After installing QDTE, set up the configuration as follows:

Table 7-1 Configuration for Linux/Windows

Property	Filepath for Linux/Windows
SecTool path: Installation path to SecToolsv2, a tool used to perform secure operations on software images. For more information on SecToolsv2 for Windows, see SecTools V2: Metabuild Secure Image User Guide	Linux:`SECTOOLS="META_PATH"/<chipset>.LE.X.x/common/sectoolsv2/ext/Linux`
	Windows: `SECTOOLS="META_PATH"\<chipset>.LE.X.x\common\sectoolsv2\ext\windows`
Profile file: Used for signing the security profile XML files.	Linux:`META_PATH"/<chipset>.LE.X.x/common/sectoolsv2/kodiak_security_profile.xml`
	Windows:`META_PATH"\<chipset>.LE.X.x\common\sectoolsv2\kodiak_security_profile.xml`
Sign cmd json: Used for signing the JSON file from the QDTE installation directory. The test signing mode is used for working on a nonsecure device. For secure devices, see enable secure boot.	Linux:`/local/mnt/qcom/QDTE/test_signing_mode.json`
	Windows: `C:\Program Files (x86)\Qualcomm\QDTE\test_signing_mode.json`
Devprg file: Device programmer file.	Linux: `/prj/qct/asw/crmbuilds/crmhyd/nsid-hyd-05/BOOT.MXF.1.0.c1-00134-KODIAKLA-2/boot_images/boot/QcomPkg/SocPkg/Kodiak/Bin/LAA/DEBUG/prog_firehose_ddr.elf`
	Windows:`META_PATH"\<chipset>.LE.X.x\BOOT.MXF.1.0.c1\boot_images\boot\QcomPkg\SocPkg\Kodiak\Bin\LAA\DEBUG\prog_firehose_ddr.e`

Table 7-1 Configuration for Linux/Windows (cont.)

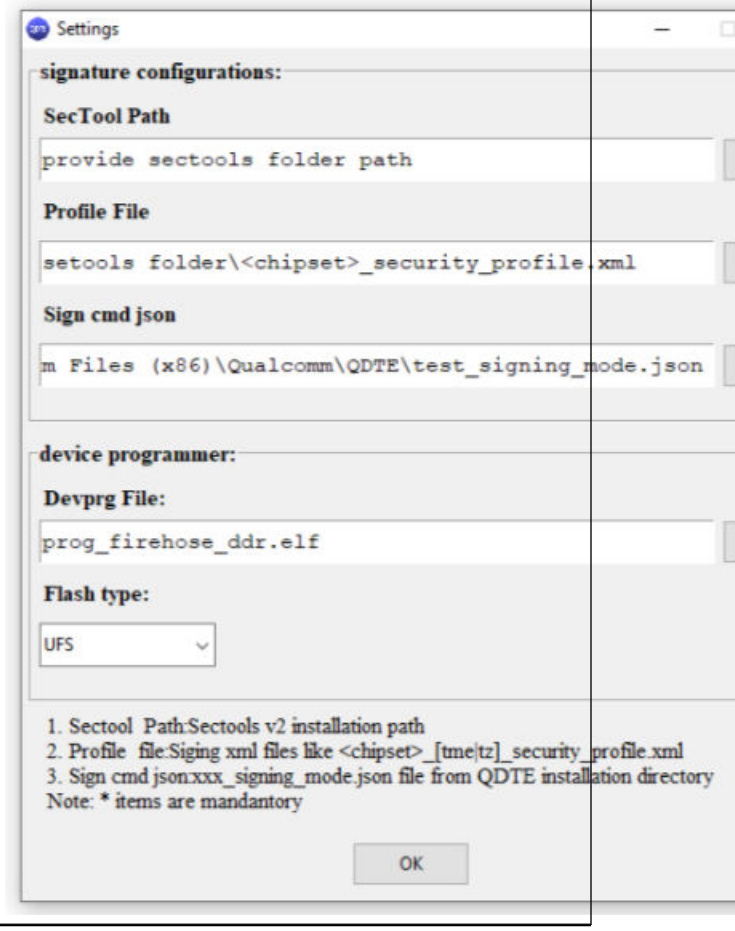
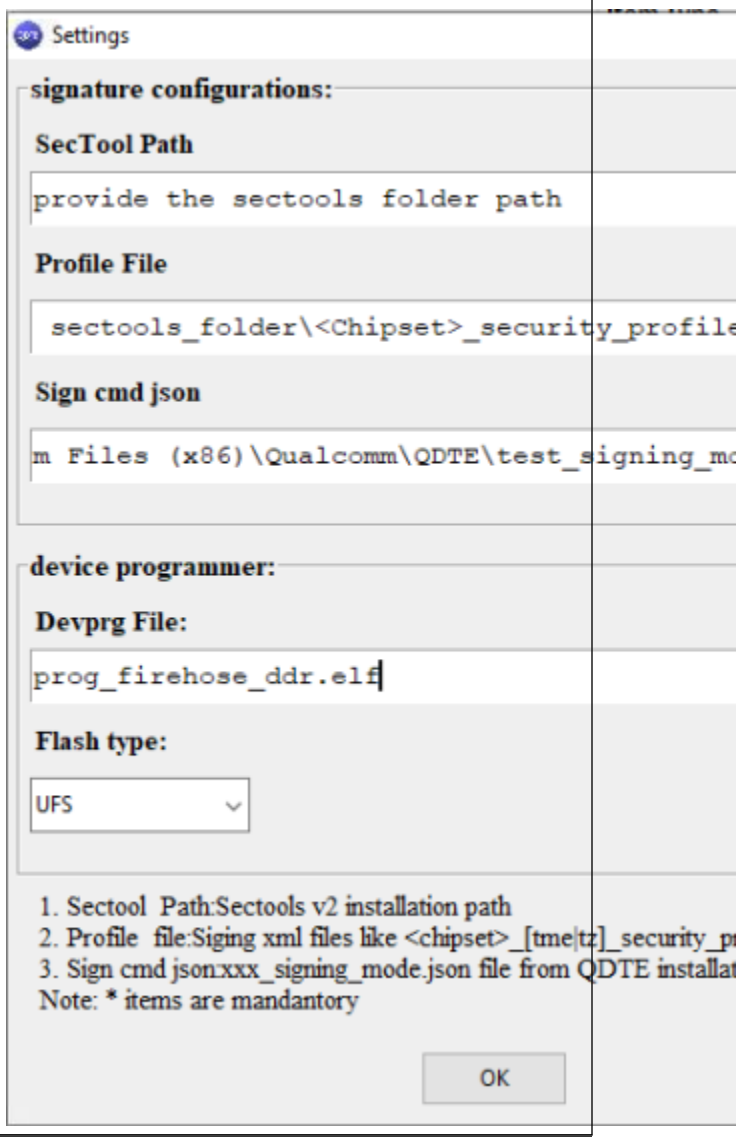
Property	Filepath for Linux/Windows
Flash type: Type of storage, which in this case is UFS.	Linux: 

Table 7-1 Configuration for Linux/Windows (cont.)

Property	Filepath for Linux/Windows
	Windows: 

1. To open the XBLConfig ELF file, navigate to **File** > **Open DTB Elf** > **From Build**.
   - If the file is invalid or corrupted, an error is generated. To prevent this, ensure to download the valid boot binaries including `xbl_config.elf`, and use them with XBL config.

     NOTE The boot component is released in the binary and is part of the firmware binary.
   - If the file loads successfully, a tree view appears, populated with the DTB elements.

The XBLConfig helper window appears containing helpful instructions for the following:

- The name of the file being edited.
- A list of DTBs found in the XBLConfig ELF.
- An option to save the modified XBLConfig, along with a console window that displays the progress of the updates.

In the background, the QDTE tool invokes the XBLConfig toolchain to disassemble the ELF file. It may take a few seconds before a success message appears in the console.

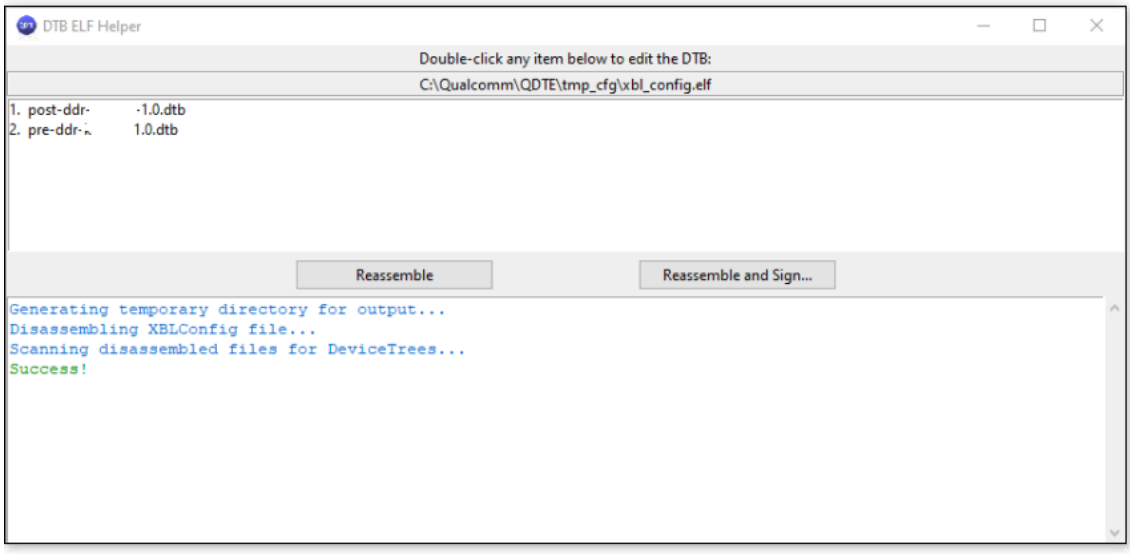
- Once the XBLConfig has been successfully disassembled, a list of DTBs found in the XBLConfig is displayed, accompanied by a `Success!` message.
- The console window uses color-coding to indicate the severity of messages:
  - Blue: Informational
  - Green: Success
  - Orange: Warning
  - Red: Error
  - Black: Miscellaneous debugging information output by the XBLConfig toolchain

2. To edit the XBLConfig ELF file, double-click it.

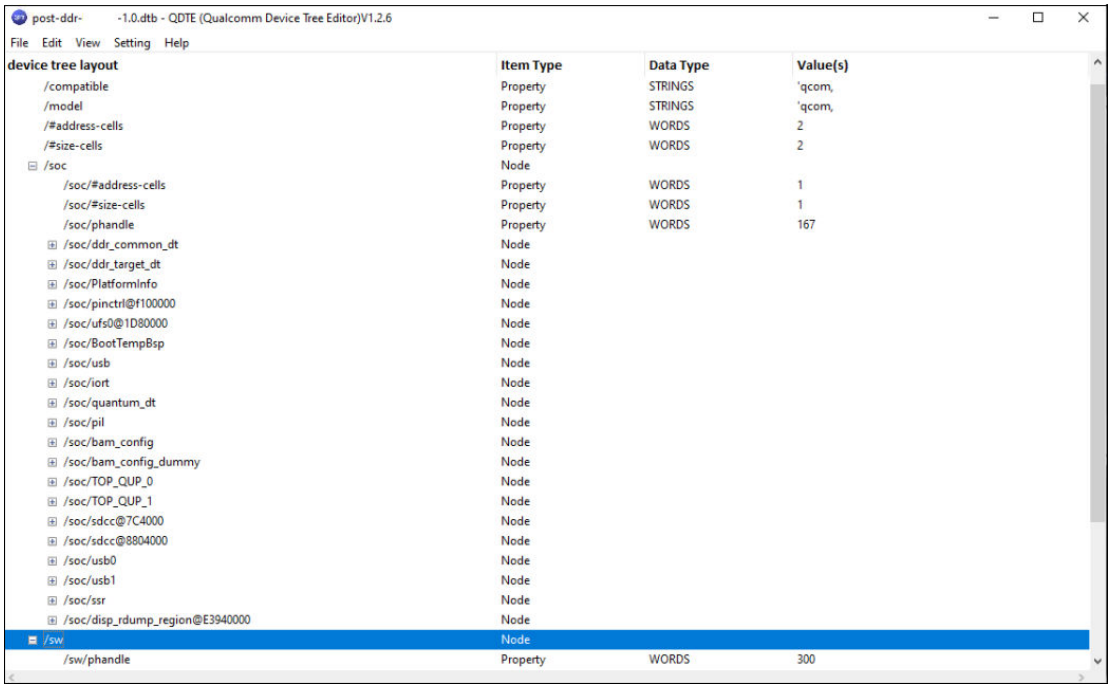
   The file is displayed on the primary QDTE interface.

   You can edit the DTB just like any other DTB file on the system. Changes are autosaved as you modify theDTB file and when the XBLConfig ELF is re-assembled. Due to this autosave feature, the file cannot be reloaded from the file system.

   The figure shows an XBLConfig Helper window displaying the disassembled results of an XBLConfig file containing two DTBs.



- The figure illustrates an example file opened in a tree view with all four columns populated:



  While nodes do not have values for **Data Type** and **Value(s)**, the properties do.

  - The node name and property can only contain a subset of characters, specifically: abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789,._+?#-.
  - Additionally, the node name can contain @.
  - It is important to note that a node cannot have two children with identical names, regardless of whether the children are nodes or properties.

- If the file undergoes changes on the disk while the QDTE tool is open, you can select **File** > **Reload File** to discard all changes and reload the file from the disk.

3. To edit a property, you can either right-click and select **Edit** or double-click to open a dialog box. This dialog box allows you to edit both the property name and value.

   When editing a property value, the dialog box is the only part of the GUI that responds to user actions or inputs.

   For WORDS and BYTES types, the dialog box allows you to view and edit the values in both decimal and hexadecimal formats.

   If you edit the value in one format. The value in the other format is automatically updated. WORDS and BYTES are treated as unsigned integers with the following maximum values:

   - A single WORDS value can have a maximum value of 4294967295.
   - A single BYTES value can have a maximum value of 255.
   - A single STRINGS value contains allowed characters in a string enclosed between two single quotes.

   To use a single quote in a value, it should be preceded by a backslash (\). To use a backslash in a value, it should be followed by another backslash (\ becomes \\). Other unrecognized backslash escape sequences remain unaffected in the string.

If you must input multiple values, delimit with a space. This method is applicable for WORDS, BYTES, and STRINGS types.

To edit the name of a property, modify the part of the property path text field that comes after the final forward slash. Remember, only the property name can be modified. The rest of the path should remain unchanged.

- To add items to the tree, right-click on a node. The following items are displayed:

Tree item	Description
Add New Child	Adds a child in the tree.
Node	Prompts you for the name of the new node and adds it. Invalid node names result in an error.
Property of Type	Opens a menu that allows you to choose the type of property to add. Adding a property displays the **Edit** dialog box corresponding to the type of the new property. On confirmation, a new property of the specified type is added.

- All items in the DeviceTree, except for the root node, can be removed. To delete an item, right-click on it and select **Delete**. Deleting a node will also remove its child nodes, if any.

4. To view the DTB file as a binary hexadecimal dump, go to **View** > **Raw**. This displays the exported DTB file.

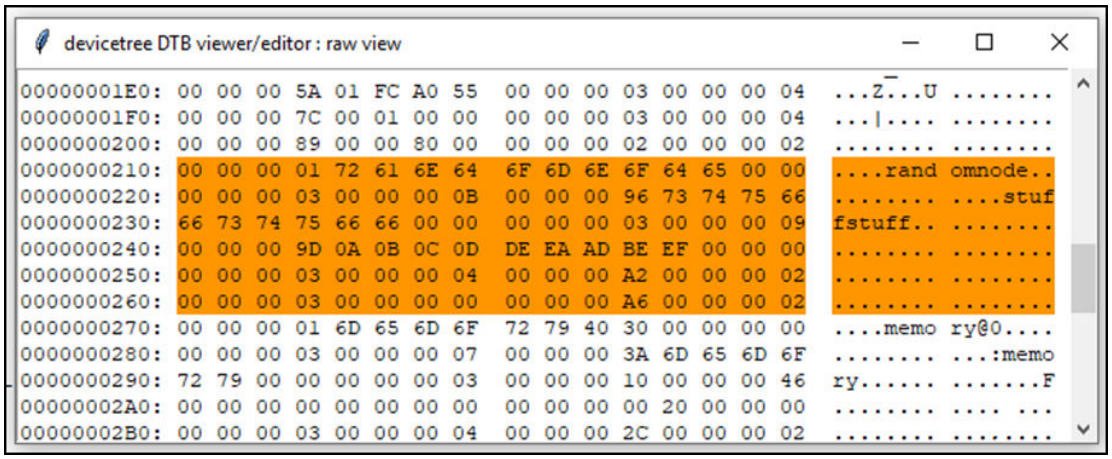
   When debugging or reading through the raw view, you have the option to highlight specific items in the DeviceTree.

   a. To do this, right-click any item in the Tree view.

   b. Click **Highlight** and then choose a color.

      - If the default options are not suitable, the **Custom** option allows you to choose any color.
      - When you highlight an item, all its child items (including those added later) are also highlighted. The highlights include the 4-bytestart and end markers and the data in between.
      - For properties, the string table does not display property names.
      - If you highlight the root node, every item in the structure block of the DeviceTree is highlighted, except for the header, memory reservation block, and strings block.

The following example of the raw view shows the offset, hexadecimal value, and corresponding character columns, with a node highlighted in orange.



To remove all highlights, go to **View** > **Clear Highlights**.

5. To save the modified DTBs, you can either go to **File** > **Save** (or press **CTRL + S**) to overwrite the existing file.

   Or go to **File** > **Save Copy As** (or press **CTRL + SHIFT + S**) to save a copy of the DTB to a new file.

   When saving DTB files, elements that cannot be modified in this GUI, such as the memory reservation block, remains unchanged from the input file. The output binary should match the raw view.

   You can also save your changes by selecting **File** > **Export to DTS**. The system prompts you to specify a file to which the DTS file should be exported.

   When recompiled with the DeviceTree compiler (DTC) or similar tools, this source file should generate a structure similar to that produced when exporting the DTB directly.

   - The advantage of this method is that it allows you to modify parameters that cannot be changed in this tool.
   - The downside is that the file must be recompiled.
   - The `DATA TYPE` displayed in the exported DTS files can differ from the actual data type in the original DTSI files. For example, when read from QDTE, the UIN16 value is displayed as two UIN8 values.

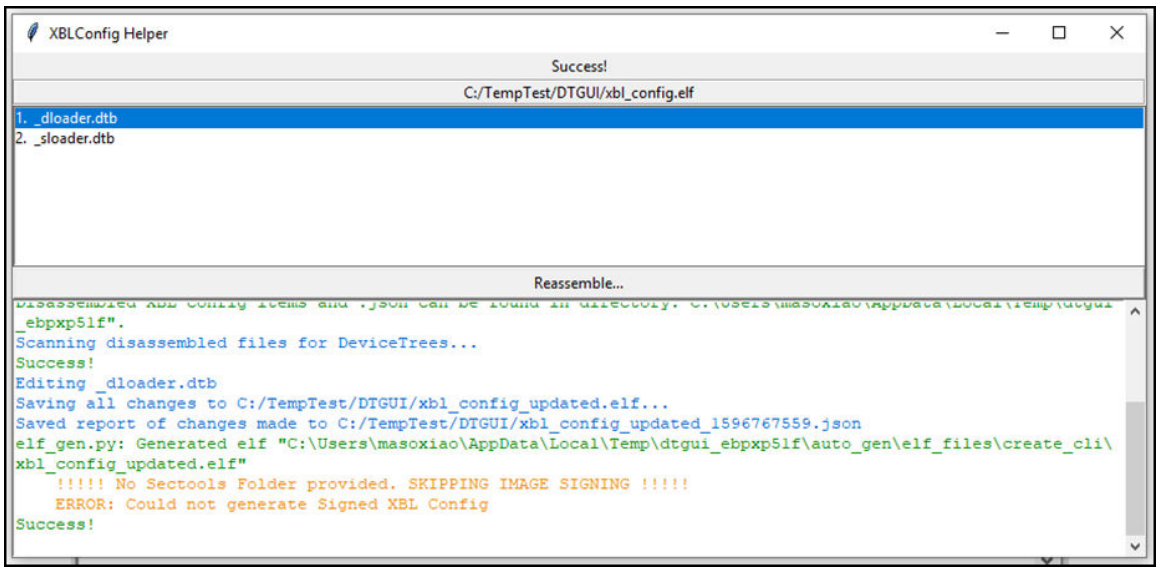
   To generate a summary of the changes, select **File** > **Export Change Report**. This will create a JSON file listing all changes made to the DT in the session, along with the result of the DTB after the operation.

6. To save the modified DTB files within the main QDTE, navigate to the XBLConfig Helper window, and click **Reassemble and Sign**. This saves your modified XBLConfig file.

7. Next, choose a name for the output file.

   The XBLConfig toolchain is then invoked to re-assemble the ELF.

   NOTE If an error message appears indicating that a signed XBLConfig was not generated, it is downgraded to a warning, and a success message is displayed instead.


XBLConfig Helper
Success!
C:/TempTest/DTGUI/xbl_config.elf
1. _dloader.dtb
2. _sloader.dtb
Reassemble...
_ebpxp51f".
Scanning disassembled files for DeviceTrees...
Success!
Editing _dloader.dtb
Saving all changes to C:/TempTest/DTGUI/xbl_config_updated.elf...
Saved report of changes made to C:/TempTest/DTGUI/xbl_config_updated_1596767559.json
elf_gen.py: Generated elf "C:\Users\masoxiao\AppData\Local\Temp\dtgui_ebpxp51f\auto_gen\elf_files\create_cli\
xbl_config_updated.elf"
!!!!! No Sectools Folder provided. SKIPPING IMAGE SIGNING !!!!!
ERROR: Could not generate Signed XBL Config
Success!

8 Debug

The universal asynchronous receiver/transmitter (UART) logs enable you to monitor and debug data exchanged via a UART interface between software and a device.

The sample UART logs can help you identify the various stages of system bootup. In the event of a crash, these logs can assist you to do the following:

- Determine the stage at which the crash occurred
- Gain a high-level understanding of the execution process

8.1 PBL/XBL log

The sample primary boot loader/eXtensible boot loader (PBL/XBL) log indicates the start and end of the PBL and XBL.

For more information on PBL and XBL, see Boot loader.

The table lists log entries that can help verify the start and end of the boot loaders:

Table 8-1 Checking PBL/XBL log

Log entry	Description
`S - Boot Interface: UFS`	Indicates the booting of software images from UFS as boot media is stored in UFS
`S - PBL Patch Ver: 1`	Indicates the start of PBL
`S - 69043 - PBL, End`	Indicates the end of PBL
`B - 79574 - SBL1, Start`	Indicates the start of XBL **NOTE** Both SBL1 and XBL refer to the same secondary boot loader.
`B - 1394338 - SBL1, End`	Indicates the end of XBL
`D - 1318210 - SBL1, Delta`	Indicates the time difference in micro seconds between start and end of XBL

The following is a sample log:

```
Format: Log Type - Time(microsec) - Message - Optional Info
Log Type: B - Since Boot(Power On Reset),  D - Delta,  S - Statistic
S - QC_IMAGE_VERSION_STRING=BOOT.<XX.x.x.xx-xxxxx-XXXXX-x>
S - IMAGE_VARIANT_STRING=Soc<XXXXXXX>
S - OEM_IMAGE_VERSION_STRING=ip-<xx-xxx-xxx-xx>
S - Boot Interface: UFS
```

```
S - Secure Boot: Off
S - Boot Config @ 0x00786070 = 0x000000c1
S - JTAG ID @ 0x00786130 = 0x001980e1
S - OEM ID @ 0x00786138 = 0x00000000
S - Serial Number @ 0x00786134 = 0xf744c1e9
S - OEM Config Row 0 @ 0x007841c0 = 0x0000000000000000
S - OEM Config Row 1 @ 0x007841c8 = 0x0000000000000000
S - Feature Config Row 0 @ 0x00784148 = 0x0000000000000000
S - Feature Config Row 1 @ 0x00784150 = 0x0000000000000000
S - Core 0 Frequency, 1516 MHz
S - PBL Patch Ver: 1
D -      6626 - pbl_apps_init_timestamp
D -     31574 - bootable_media_detect_timestamp
D -       853 - bl_elf_metadata_loading_timestamp
D -       704 - bl_hash_seg_auth_timestamp
D -      6621 - bl_elf_loadable_segment_loading_timestamp
D -      4638 - bl_elf_segs_hash_verify_timestamp
D -     17185 - bl_sec_hash_seg_auth_timestamp
D -       811 - bl_sec_segs_hash_verify_timestamp
D -        31 - pbl_populate_shared_data_and_exit_timestamp
S -     69043 - PBL, End
B -     79574 - SBL1, Start
B -    210297 - SBL1 BUILD @ <XXXXXXX>
B -    214445 - usb: usb_shared_xbl_dtb_node_init dtb status , 0x3
B -    223717 - usb: HS: device tree Not available , 0x3
B -    224114 - usb: eud_serial_upd , 0xf744c1e9
D -    229421 - sbl1_hw_init
D -       519 - media_init:1
D -         0 - smss_load_cancel
B -    239455 - SMSS -  Image Load, Start
D -      3111 - SMSS -  Image Loaded, Delta - (0 Bytes)
D -      1037 - Auth Metadata
D -      5825 - sbl1_xblconfig_init
B -    254522 - XBL Config -  Image Load, Start
B -    258640 - DTB  Found: [pre-ddr][7601f201000000][0]
D -      4453 - boot_pre_ddr_dtb_load
B -    267851 - UFS Device Tree Settings
B -    271206 - UFS Init Speed: HS Enabled 1, Gear 4 2 Lane Rate 2
B -    280874 - UFS Perf Speed: HS Enabled 1, Gear 4 2 Lane Rate 2
B -    287096 - UFS Timeouts(us): fDeviceInit 2500000, UTRD Poll 30000000
D -     45323 - media_init:2
B -    387868 - UFS INQUIRY ID: SAMSUNG KM2L9001CM-B518 0700
B -    389485 - UFS Boot LUN: 1
B -    400953 - UFS GEAR: 3
D -     95099 - media_init:3
D -        30 - shrm_load_cancel
B -    410835 - SHRM -  Image Load, Start
```

```
D -       702 - Auth Metadata
D -      1403 - Segments hash check
D -     12230 - SHRM -  Image Loaded, Delta - (35752 Bytes)
D -         0 - boot_default_cdt_init
D -       488 - boot_cdt_init
B -    435540 - CDT -  Image Load, Start
B -    438407 - CDT Version:3,Platform ID:32,Major ID:1,Minor ID:0,Subtype:2
D -     16928 - sbl1_hw_platform_pre_ddr
D -         0 - devcfg init
B -    464942 - PMIC A:2.0 B:1.0 C:2.2 I:1.0
B -    466772 - PM: PM8 0x4
B -    469181 - PM: xVdd reset
B -    471835 - PM: PON by SYSOK
B -    724497 - PM: SET_VAL:Skip
B -    724527 - PM: Verifying PON-Trigger specific configurations & current
PON-Trigger
B -    733281 - PM: All PON-Trigger specific configs verified. Proceeding to
BOOT
B -    742736 - PM: PSI: b0x06_v0x3c
B -    745450 - PM: Device Init # SPMI Transn: 15422
D -    291031 - pm_device_init, Delta
B -    751154 - pm_driver_init, Start
B -    762286 - PM: Driver Init # SPMI Transn: 500
D -      7655 - pm_driver_init, Delta
B -    766983 - PM: CHG Type in CHG init : 0
B -    771375 - PM: Battery ID: 7612Ohm
B -    774547 - PM: debug board connected
B -    778207 - vsense_init, Start
D -         0 - vsense_init, Delta
D -    339312 - sbl1_hw_pre_ddr_init
D -         0 - boot_dload_handle_forced_dload_timeout
D -      2928 - sbl1_load_ddr_training_data
B -    803034 - Pre_DDR_clock_init, Start
D -        61 - Pre_DDR_clock_init, Delta
D -     12902 - sbl1_ddr_set_params
B -    814746 - sbl1_ddr_init, Start
B -    818162 - LP4 DDR detected
B -    832772 - eCDT MRR - Data Starting Address: 0x09066D00
D -     15250 - sbl1_ddr_init, Delta
B -    838445 - DSF version = 262.0.46
B -    841830 - Manufacturer ID = 1, Device Type = 7
B -    845399 - Rank 0 size = 2048 MB, Rank 1 size = 4096 MB
B -    850248 - Row Hamming DDR
B -    855769 - Row Hammer Check : DRAM supports unlimited MAC Value :
MR_RH[OP2:0 = 0] & MR_RH[OP3 = 1] for CH0 & CS0
B -    864431 - Row Hammer Check : DRAM supports unlimited MAC Value :
MR_RH[OP2:0 = 0] & MR_RH[OP3 = 1] for CH0 & CS1
```

```
B -    875136 - Row Hammer Check : DRAM supports unlimited MAC Value :
MR_RH[OP2:0 = 0] & MR_RH[OP3 = 1] for CH1 & CS0
B -    885842 - Row Hammer Check : DRAM supports unlimited MAC Value :
MR_RH[OP2:0 = 0] & MR_RH[OP3 = 1] for CH1 & CS1
D -     81893 - sbl1_ddr_init
D -        31 - boot_pre_ddi_entry
B -    904325 - do_ddr_training, Start
B -    945530 - DDR: Start of DDR Training Restore
B -    949068 - Current DDR Freq = 1709 MHz
B -    950166 - Max enabled DDR Freq = 2092 MHz
B -    954192 - DDR: End of DDR Training Restore
D -     51057 - do_ddr_training, Delta
D -     58834 - sbl1_do_ddr_training
D -       518 - boot_ddi_entry
B -    970022 - Pimem init cmd, entry
D -      9211 - Pimem init cmd, exit
B -    982466 - External heap init, Start
B -    985546 - External heap init, End
D -     22112 - sbl1_post_ddr_init
D -        30 - sbl1_hw_init_secondary
B -    996252 - DDR -  Image Load, Start
B -   1001559 - DTB  Found: [post-ddr][7601f201000000][2001010000000000]
D -      9486 - boot_post_ddr_dtb_load
D -       762 - boot_fedl_check
B -   1013789 - APDP -  Image Load, Start
D -      2959 - APDP -  Image Loaded, Delta - (0 Bytes)
D -         0 - boot_dload_dump_security_regions
D -         0 - ramdump_load_cancel
B -   1030137 - RamDump -  Image Load, Start
D -      3325 - RamDump -  Image Loaded, Delta - (0 Bytes)
D -         0 - boot_update_abnormal_reset_status
D -         0 - boot_cache_set_memory_barrier
D -         0 - boot_smem_debug_init
D -       458 - boot_smem_init
D -         0 - boot_smem_alloc_for_minidump
D -        61 - boot_smem_store_pon_status
D -        30 - sbl1_hw_platform_smem
D -        61 - boot_ddr_share_data_to_aop
D -       488 - boot_clock_init_rpm
D -         0 - boot_vsense_copy_to_smem
D -         0 - boot_populate_ram_partition_table
D -        30 - boot_populate_ddr_details_shared_table
D -         0 - sbl1_tlmm_init
D -         0 - sbl1_efs_handle_cookies
B -   1092906 - OEM_MISC -  Image Load, Start
D -       762 - Auth Metadata
D -       274 - Segments hash check
```

```
D -     10645 - OEM_MISC -  Image Loaded, Delta - (5048 Bytes)
B -   1106875 - QTI_MISC -  Image Load, Start
D -      5704 - QTI_MISC -  Image Loaded, Delta - (0 Bytes)
B -   1122308 - PM: PM Total Mem Allocated: 2340
D -      5520 - sbl1_pm_aop_pre_init_wrapper
B -   1126883 - AOP -  Image Load, Start
D -       763 - Auth Metadata
D -      1677 - Segments hash check
D -     13085 - AOP -  Image Loaded, Delta - (200732 Bytes)
B -   1143292 - QSEE Dev Config -  Image Load, Start
D -       854 - Auth Metadata
D -       610 - Segments hash check
D -     13024 - QSEE Dev Config -  Image Loaded, Delta - (53248 Bytes)
B -   1165344 - QSEE -  Image Load, Start
D -     17873 - Auth Metadata
D -     21594 - Segments hash check
D -     82441 - QSEE -  Image Loaded, Delta - (3900768 Bytes)
B -   1251811 - DTB: invalid vibration prop value
B -   1256722 - DTB: Vibration Enabled
D -     10217 - sbl1_hw_play_vibr
B -   1264865 - SEC -  Image Load, Start
D -      3142 - SEC -  Image Loaded, Delta - (0 Bytes)
B -   1271758 - CPUCPFW -  Image Load, Start
D -     17598 - Auth Metadata
D -     17385 - Segments hash check
D -     48007 - CPUCPFW -  Image Loaded, Delta - (171180 Bytes)
B -   1328793 - QHEE -  Image Load, Start
D -     17843 - Auth Metadata
D -      6954 - Segments hash check
D -     27664 - QHEE -  Image Loaded, Delta - (1478736 Bytes)
B -   1359781 - APPSBL -  Image Load, Start
D -       671 - Auth Metadata
D -     10889 - Segments hash check
D -     25315 - APPSBL -  Image Loaded, Delta - (2564048 Bytes)
D -         0 - sbl1_save_appsbl_index
B -   1394338 - SBL1, End
D -   1318210 - SBL1, Delta
```

8.2 UEFI log

The UEFI log provides information about the start and end of UEFI.

For more information on UEFI, see Unified Extensible Firmware Interface (UEFI).

The table lists log entries that can help you verify the start and end of UEFI.

Table 8-2 Checking UEFI log

Log entry	Description
`UEFI Start [ 1568]`	Indicates the start of UEFI
`[ 3086] UEFI End`	Indicates the end of UEFI

The subsequent example log uses the debug build flavor.

To use the build flavor, replace the 'RELEASE' string with 'DEBUG' in the boot build command. For more information, see Build – Additional information > Build firmware > Build BOOT.

```
UEFI Start     [ 1568]
 - 0x09FC01000 [ 1571] Sec.efi
ASLR        : ON
DEP         : ON (RTB)
Timer Delta : +11 mS
RAM Entry 0 : Base 0x0080000000  Size 0x003A800000
RAM Entry 1 : Base 0x00C0000000  Size 0x0040000000
RAM Entry 2 : Base 0x0100000000  Size 0x0100000000
Total Available RAM : 6056 MB (0x017A800000)
Total Installed RAM : 6144 MB (0x0180000000)
Multithread   : ON (Lib ver 1.2)
CPU Cores     : 8 (init 2)
Init 1 aux cores of 7
Init CPU core 1
CONF File   : uefiplatLA.cfg
UEFI Ver    : 6.0.231205.BOOT.<XXX.x.x.Xx-xxxxx-XXXXXXX-x>
Build Info  : 64b <XXXX>
Boot Device : UFS
PROD Mode   : FALSE
Retail      : FALSE

  > Scheduler up on Core 1
DTB config    : client[0]..trace[0]..verbose[0]
 - 0x09F0CF000 [ 1630] DxeCore.efi
Loading DxeCore at 0x009F0CF000 EntryPoint=0x009F0D0000
HOBLIST address in DXE = 0x9EEC0018
FV Hob            0x9FC00000 - 0x9FE70FFF
FV Hob            0x9F10E000 - 0x9F47AFFF
FV2 Hob           0x9F10E000 - 0x9F47AFFF
                  631008B0-B2D1-410A-8B49-2C5C4D8ECC7E -
00000000-0000-0000-0000-000000000000
 - 0x09F069000 [ 1633] EnvDxe.efi
 - 0x09F05F000 [ 1634] RscRtDxe.efi
 - 0x09F057000 [ 1634] SCHandlerRtDxe.efi
 - 0x09EB94000 [ 1634] RuntimeDxe.efi
 - 0x09F049000 [ 1635] ArmCpuDxe.efi
 - 0x09F040000 [ 1636] ArmGicDxe.efi
```

```
 - 0x09F039000 [ 1636] MetronomeDxe.efi
 - 0x09F031000 [ 1636] ArmTimerDxe.efi
 - 0x09F027000 [ 1636] SmemDxe.efi

 - 0x09EFCD000 [ 1637] DALSys.efi
 - 0x09EFC4000 [ 1637] HWIODxeDriver.efi
 - 0x09EFB9000 [ 1637] ChipInfo.efi
 - 0x09EFB1000 [ 1638] PlatformInfoDxeDriver.efi
 - 0x09EF8D000 [ 1638] HALIOMMU.efi
 - 0x09EF79000 [ 1687] ULogDxe.efi
 - 0x09EF71000 [ 1687] CmdDbDxe.efi
 - 0x09EF69000 [ 1688] PwrUtilsDxe.efi
 - 0x09EF56000 [ 1688] NpaDxe.efi
 - 0x09EF46000 [ 1689] RpmhDxe.efi
 - 0x09EF3A000 [ 1689] VcsDxe.efi
 - 0x09EF0C000 [ 1690] ClockDxe.efi
 >>> Cluster 0:        0 Hz
 >>> Cluster 2:        0 Hz
 - 0x09EEFF000 [ 1697] ICBDxe.efi
 - 0x09EEF6000 [ 1699] ShmBridgeDxeLA.efi
 - 0x09EEE7000 [ 1700] ScmDxeLA.efi
 - 0x09EEDE000 [ 1702] DALTLMM.efi
 - 0x09EED4000 [ 1702] SPMI.efi
 - 0x09EECA000 [ 1707] I2C.efi
 - 0x09EB8B000 [ 1707] ResetRuntimeDxe.efi
 - 0x09EB66000 [ 1708] PmicDxe.efi
PM0: 47, PM1: 63, PM2: 49, PM8: 46,
Module cannot re-initialize DAL module environment
 - 0x09EB5B000 [ 1726] DiskIoDxe.efi
 - 0x09EB4E000 [ 1726] PartitionDxe.efi
 - 0x09EB46000 [ 1726] EnglishDxe.efi
 - 0x09EB3B000 [ 1727] FvSimpleFileSystem.efi
 - 0x09EB1F000 [ 1727] SdccDxe.efi
 - 0x09EAFE000 [ 1728] UFSDxe.efi
UFS INQUIRY ID: SAMSUNG KM2L9001CM-B518 0700
UFS Boot LUN: 1
 Protective MBR validation might be needed.
 Protective MBR validation might be needed.
 - 0x09E80F000 [ 1757] Fat.efi
 - 0x09D9B0000 [ 1757] TzDxeLA.efi
 - 0x09D90F000 [ 1759] VariableDxe.efi
ConfigStorPartitions: Status:Success
<VariablePolicyInitialize:179> variable policy engine disabled
 - 0x09E807000 [ 1877] QcomWDogDxe.efi
HW Wdog Setting from PCD : Disabled
 - 0x09D927000 [ 1878] WatchdogTimer.efi
 - 0x09D6D4000 [ 1878] SecurityStubDxe.efi
```

```
 - 0x09E821000 [ 1878] EmbeddedMonotonicCounter.efi
 - 0x09D91E000 [ 1880] RealTimeClock.efi
 - 0x09D6AD000 [ 1881] DevicePathDxe.efi
 - 0x09D6CC000 [ 1881] CapsuleRuntimeDxe.efi
 - 0x09D666000 [ 1882] HiiDatabase.efi
 - 0x09D6C3000 [ 1882] FontDxe.efi
 - 0x09D58E000 [ 1887] QcomBds.efi
 - 0x09D63F000 [ 1895] VerifiedBootDxe.efi
 - 0x09D65A000 [ 1896] DDRInfoDxe.efi
 - 0x09D626000 [ 1897] FeatureEnablerDxe.efi
QseeResponse->result = 0xFFFFFFFF
Status = 0x7
QseeResponse->result = 0xFFFFFFFF
Status = 0x7
Image partition label not found
EFI_QseecomStartApp: Load from partition (featenabler_a)Failed:
Status(0x800000000000000E), appId(0)
QseeResponse->result = 0xFFFFFFFF
Status = 0x7
QseeResponse->result = 0xFFFFFFFF
Status = 0x7
Image partition label not found
EFI_QseecomStartApp: Load from partition (featenabler)Failed:
Status(0x800000000000000E), appId(0)
Image partition label not found
LoadImageFromPartitionUsingGuid Failed: 14
EFI_QseecomStartApp: Load Failed:  Status(0x800000000000000E)
Failed to start featenabler_a TA, status = 14
 - 0x09D4EB000 [ 1901] DisplayDxe.efi
DisplayDxe: SW renderer mode enabled!
DisplayDxe: Panel ID:0x00000000 [LCD]
DisplayDxe: Resolution 640x480 (1 intf)
 - 0x09D5DA000 [ 1912] FvDxe.efi
 - 0x09D60B000 [ 1912] PILProxyDxe.efi
 - 0x09D555000 [ 1913] PILDxe.efi
 - 0x09D5CF000 [ 1934] IPCCDxe.efi
 - 0x09D571000 [ 1934] GlinkDxe.efi
smem_alloc_ex: SMEM alloc_ex failed with err=-3! smem_type=478, remote=3,
size=32, flags=0x40000000. - 0x09D4DB000 [ 1935] CPRDxe.efi
 - 0x09D586000 [ 1935] PrintDxe.efi
 - 0x09D540000 [ 1936] SPI.efi
 - 0x09D4C2000 [ 1936] PmicGlinkDxe.efi
 - 0x09D4D1000 [ 1937] UsbPwrCtrlDxe.efi
 - 0x09D483000 [ 1937] QcomChargerDxeLA.efi
 - 0x09D54D000 [ 1947] ChargerExDxe.efi
 - 0x09D46A000 [ 1947] UsbfnDwc3Dxe.efi
 - 0x09D4AB000 [ 1948] XhciPciEmulation.efi
```

```
 - 0x09D43B000 [ 1948] XhciDxe.efi
 - 0x09D429000 [ 1949] UsbBusDxe.efi
 - 0x09D41C000 [ 1949] UsbKbDxe.efi
 - 0x09D4B6000 [ 1950] UsbMassStorageDxe.efi
 - 0x09D406000 [ 1950] UsbConfigDxe.efi
UsbConfigInit: USB RUMI 0, ver 65536, sub 1
UsbConfigLibOpenProtocols: PMI version (0x0)
UsbConfigLibOpenProtocols: gPmicNpaClientHS2 cannot be created
UsbConfigInit: Dual Role Enabled on Port Number: 0
UsbConfigInit: after setting role
UsbConfigInit: UsbConfigInit, not start on port: 0, mode 0
UsbConfigInit: Dual Role Enabled on Port Number: 1
UsbConfigInit: after setting role
UsbConfigInit: UsbConfigInit, not start on port: 1, mode 0
UsbConfigInit: UsbPwrCtrl No. of Ports = 1

UsbConfigPortsQueryConnectionChange: usbport->connectstate: ATT
ConnectSts : Attach, Data Role : UFP (DEVICE Mode), Lane : CC1, CoreNum : 0
HandlePortPartnerXtach: Cable Attach core 0, portmode 1, dualmode 1
 - 0x09D455000 [ 1954] ButtonsDxe.efi
ButtonsDxeTest: Keypress SDAM data payload 0
 - 0x09D462000 [ 1956] TsensDxe.efi
 - 0x09D3FE000 [ 1958] LimitsDxe.efi
 - 0x09D3D7000 [ 1962] GpiDxe.efi
 - 0x09D390000 [ 2010] UCDxe.efi
 - 0x09D3C6000 [ 2014] RngDxe.efi
 - 0x09D3BE000 [ 2015] SimpleTextInOutSerial.efi
 - 0x09D386000 [ 2016] ConPlatformDxe.efi
 - 0x09D369000 [ 2016] ConSplitterDxe.efi
 - 0x09D35D000 [ 2017] GraphicsConsoleDxe.efi
 - 0x09D37C000 [ 2018] ASN1X509Dxe.efi
 - 0x09D3B3000 [ 2019] SecRSADxe.efi
 - 0x09D348000 [ 2019] DtPlatformDxe.efi
DtPlatformLoadDtb XXXxxxx-rb3.dtb is loaded
DtPlatformDxeEntryPoint: no DT/ACPI preference found, defaulting to DT
 - 0x09D2A6000 [ 2030] SoftSKUDxe.efi
SoftSKUDxeInitialize: SoftSKU not supported for this chip
Error: Image at 0009D2A6000 start failed: Unsupported
 - 0x09D2A6000 [ 2033] MinidumpTADxe.efi
Image partition label not found
LoadImageFromPartitionUsingGuid Failed: 14
EFI_QseecomStartApp: Load Failed:  Status(0x800000000000000E)
MinidumpTALib:LoadImageFromPartition(mdcompress) failed: 0xNot Found
MinidumpTADxe: Minidump TA loading failed.
Error: Image at 0009D2A6000 start failed: Not Found
 - 0x09D294000 [ 2037] UsbMsdDxe.efi
 - 0x09D287000 [ 2038] UsbDeviceDxe.efi
```

```
 - 0x09D2A7000 [ 2038] UsbInitDxe.efi
 - 0x09D27E000 [ 2039] ParserDxe.efi
 - 0x09D277000 [ 2040] SerialPortDxe.efi
BDS Entry      [ 2040]
Disp init wait [ 2040]
------------------------------
Platform Init  [ 2057] BDS
Boot Cycle : 1
Run  Cycle : 1
UEFI Ver   : 6.0.231205.BOOT.<XXX.x.x.Xx-xxxxx-XXXXXXX-x>
Platform           : IOT
Subtype            : 1
Boot Device        : UFS
Chip Name          : <XXXxxxx>
Chip Ver           : 1.0
Chip Serial Number : <0xXxxxxxx>
Boot Core  : 0 MHz
------------------------------
 - 0x09D21C000 [ 2114] QcomChargerApp.efi
 Protective MBR validation might be needed.
 Protective MBR validation might be needed.
ERROR: A bit not set !
AppsProcClkMHz is zero
Firmware update failed - Status: Unsupported
Firmware provisioning failed
Platform Init End : 2282
------------------------------
[QcomBds] BootOrder not found
 Protective MBR validation might be needed.
 Protective MBR validation might be needed.
[QcomBds] Enumerating non-removable boot options
[QcomBds] Adding new boot/driver option Boot0000, Description: Non-removable
Media
Booting option 0:(Boot0000) "Non-removable Media"
UEFI Total : 886 ms
POST Time      [ 2454] OS Loader
 - 0x1C22E0000 [ 2456]
 - 0x1D1F80000 [ 2655]
 - 0x1FDD00000 [ 2688]
EFI stub: Booting Linux Kernel...
EFI stub: Loaded initrd from LINUX_EFI_INITRD_MEDIA_GUID device path
EFI stub: Using DTB from configuration table
EFI stub: Exiting boot services...
Start EBS        [ 2717]
MDPUpdateDynamicClocks: Clock management not supported!
MDPUpdateCoreClockAndBandwidth: MDPUpdateDynamicClocks failed!
MDPLib: MDPUpdateCoreClockAndBandwidth failed!
```

```
MDPUpdateDynamicClocks: Clock management not supported!
MDPUpdateCoreClockAndBandwidth: MDPUpdateDynamicClocks failed!
MDPLib: MDPUpdateCoreClockAndBandwidth failed!
MDPLib: SW Render enabled, no Clocks disabled need
MDPLib: SW Render enabled, no Clocks disabled need
MDPLib: SW Render enabled, no Clocks disabled need
MDPLib: SW Render enabled, no Clocks disabled need
Successfully synced all UEFI tables
Sync Duration = 40 ms
Warning: Clearing A-bit !
App Log Flush : 315 ms
ScmArmV8ExitBootServicesHandler, Status = 0x0.
Exit EBS
[ 3086] UEFI End
```

8.3 Linux log

The sample log signifies the start of the Linux boot up process.

The table lists a log entry that guides you on how to confirm the successful bootup of Linux.

Table 8-3 Checking Linux log

Log entry	Description
[0.000000] Booting Linux on physical CPU 0x0000000000 [0x412fd050] [0.000000] Linux version 6.6.0 (oe-user@oe-host) (aarch64-qcom-linux-gcc (GCC) 11.4.0, GNU ld (GNU Binutils) 2.38.20220708) #1 SMP PREEMPT <dayXX> <monthXX> [0.000000] KASLR enabled [0.000000] Machine model: Qualcomm Technologies, Inc. XXXxxxx-addons XXx platform [0.000000] efi: EFI v2.7 by Qualcomm Technologies, Inc.	Indicates the start of the Linux bootup

The following log is a sample Linux bootup log:

```
[    0.000000] Booting Linux on physical CPU 0x0000000000 [0x412fd050]
[    0.000000] Linux version 6.6.0 (oe-user@oe-host) (aarch64-qcom-linux-gcc
(GCC) 11.4.0, GNU ld (GNU Binutils) 2.38.20220708) #1 SMP PREEMPT <dayXX>
<monthXX> <timexxx> UTC <yearxxxx>
[    0.000000] KASLR enabled
[    0.000000] Machine model: Qualcomm Technologies, Inc. XXXxxxx-addons XXx
platform
[    0.000000] efi: EFI v2.7 by Qualcomm Technologies, Inc.
[    0.000000] efi: MEMATTR=0x9d233018 INITRD=0x9d228718 RNG=0x9d222018
MEMRESERVE=0x9d228218
[    0.000000] random: crng init done
[    0.000000] Reserved memory: created CMA memory pool at
```

```
0x00000000ff000000, size 12 MiB
[    0.000000] OF: reserved mem: initialized node adsp-heap, compatible id
shared-dma-pool
[    0.000000] OF: reserved mem: 0x00000000ff000000..0x00000000ffbfffff
(12288 KiB) map reusable adsp-heap
.....................
.....................
.....................
.....................
[   23.653629] qnoc-sc7280 1580000.interconnect: sync_state() pending due to
3d00000.qcom,kgsl-3d0
[   23.662613] qnoc-sc7280 1580000.interconnect: sync_state() pending due to
3d00000.gpu
[   23.670698] qnoc-sc7280 9100000.interconnect: sync_state() pending due to
3d00000.gpu
[   23.678776] qnoc-sc7280 9100000.interconnect: sync_state() pending due to
3d00000.qcom,kgsl-3d0
[   23.687743] qnoc-sc7280 1580000.interconnect: sync_state() pending due to
aa00000.video-codec
[   23.696524] qnoc-sc7280 1740000.interconnect: sync_state() pending due to
aa00000.video-codec
[   23.705308] qnoc-sc7280 1500000.interconnect: sync_state() pending due to
aa00000.video-codec
[   23.714081] qnoc-sc7280 9100000.interconnect: sync_state() pending due to
aa00000.video-codec
[   23.722895] qcom-rpmhpd 18200000.rsc:power-controller: sync_state()
pending due to aa00000.video-codec
[   33.874665] refgen: disabling
..................
```

9 Examples

The examples illustrates typical scenarios encountered during system bootup.

NOTE For information on fastboot detection and commands, see Fastboot.

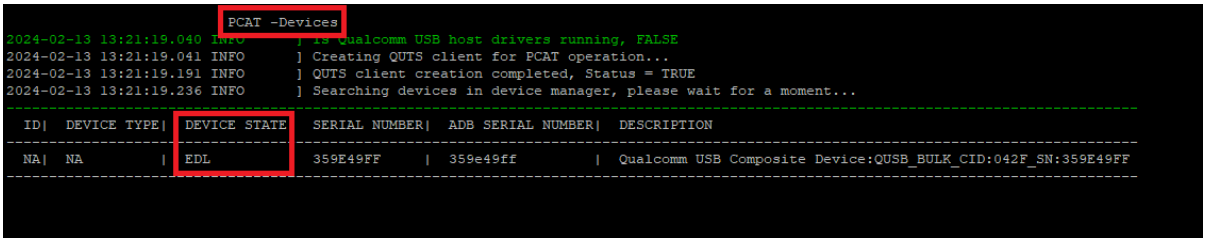
9.1 Enumeration of EDL device manager

If the storage is either empty or corrupted, the device switches to the EDL mode upon power on.

To verify if the device is in EDL mode, run the following command:

```
PCAT-Devices
```

The figure shows the device state as EDL:



When EDL is detected, load the build using Qualcomm Product Configuration Assistant Tool (PCAT).

9.2 Selection of boot device

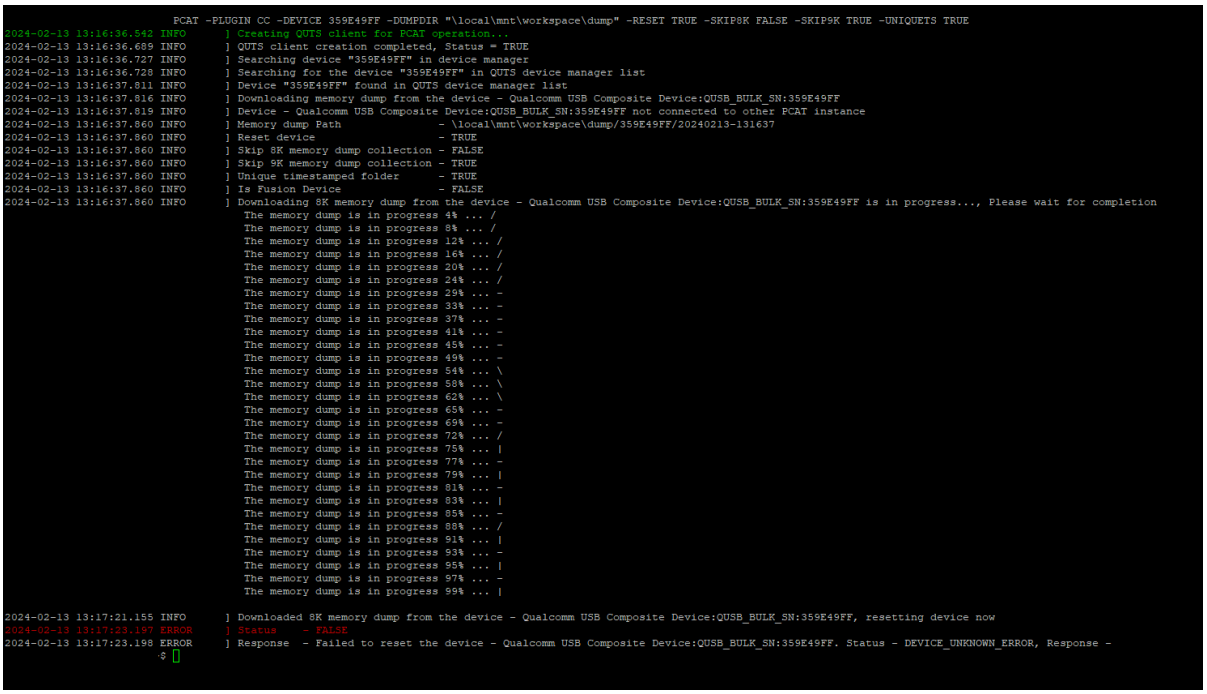
Users with full access to the proprietary software shipped with Qualcomm Linux can use the BDS menu. This menu offers various use cases as part of the test package. For more information, see the Boot device selection in Qualcomm Linux Boot Guide - Addendum.

9.3 Detection of RAM dump

In the event of a crash, PCAT collects the RAM dump and displays the logs for debugging purposes.

For information on how to collect and parse the RAM dump, see Debug Linux Kernel space > Collect and Parse RAM dump collection.

The figure illustrates the driver identifying the crash and displaying the USB logs:



10 References

Related documents:

Title	Document number
Document	
Qualcomm Linux Build Guide	80-70014-254
Dev Kit User Guide	80-70014-251
Qualcomm Linux Security Guide	80-70014-11
Resources	
TianocoreEDK2 open-source implementation	http://www.tianocore.org/edk2/.
systemd-boot	https://www.freedesktop.org/software/systemd/man/latest/systemd-boot.html
EFI boot stub	https://docs.kernel.org/admin-guide/efi-stub.html
UEFI specification – DXE phase	https://uefi.org/specs/PI/1.8/V2_Overview.html
Qualcomm package manager (QPM)	https://qpm.qualcomm.com/#/main/tools/details/QPM3

Acronyms

Acronym	Definition
AOP	Always-on processor
APPS	Applications
BDS	Boot device selection
cDSP	Compute DSP
CDT	Configuration data table
CLK	Clock
CPUCP	CPU subsystem control processor
DDR	Double data rate
DEVCFG	Device configuration
DSC	Description
DT	DeviceTree
DXE	Driver execution environment
EDL	Emergency download
EFI	Extensible firmware interface
ESP	EFI system partition

FDF	Forms data format
GPIO	General-purpose input/output
IMEM	Internal memory
Initrd	Initial RAM disk
ISP	Image signal processor
LPASS	Low-power audio subsystem
MMU	Memory management unit
NPU	Neural processing unit
NSP	Neural signal processor
PBL	Primary boot loader
PCAT	Product configuration assistant tool
PCD	Platform configuration database
PMIC	Power management integrated circuit
QDTE	Qualcomm DeviceTree editor
QSC	Qualcomm Software center
Qualcomm TEE	Qualcomm® trusted execution environment (TEE)
RoT	Root of trust
TZ	TrustZone
UART	Universal asynchronous receiver/transmitter
UEFI	Unified extensible Firmware interface
UFS	Universal flash storage
XBL	Extensible boot loader, referred as secondary boot loader (SBL) and SBL1 occasionally in this guide
XBL_CFG	XBL configuration
XBOOTLDR	Extended Boot Loader partition
xPU	eXternal protection unit

LEGAL INFORMATION

Your access to and use of this material, along with any documents, software, specifications, reference board files, drawings, diagnostics and other information contained herein (collectively this "Material"), is subject to your (including the corporation or other legal entity you represent, collectively "You" or "Your") acceptance of the terms and conditions ("Terms of Use") set forth below. If You do not agree to these Terms of Use, you may not use this Material and shall immediately destroy any copy thereof.

1) Legal Notice.

This Material is being made available to You solely for Your internal use with those products and service offerings of Qualcomm Technologies, Inc. ("**Qualcomm Technologies**"), its affiliates and/or licensors described in this Material, and shall not be used for any other purposes. If this Material is marked as "**Qualcomm Internal Use Only**", no license is granted to You herein, and You must immediately (a) destroy or return this Material to Qualcomm Technologies, and (b) report Your receipt of this Material to qualcomm.support@qti.qualcomm.com. This Material may not be altered, edited, or modified in any way without Qualcomm Technologies' prior written approval, nor may it be used for any machine learning or artificial intelligence development purpose which results, whether directly or indirectly, in the creation or development of an automated device, program, tool, algorithm, process, methodology, product and/or other output. Unauthorized use or disclosure of this Material or the information contained herein is strictly prohibited, and You agree to indemnify Qualcomm Technologies, its affiliates and licensors for any damages or losses suffered by Qualcomm Technologies, its affiliates and/or licensors for any such unauthorized uses or disclosures of this Material, in whole or part.

Qualcomm Technologies, its affiliates and/or licensors retain all rights and ownership in and to this Material. No license to any trademark, patent, copyright, mask work protection right or any other intellectual property right is either granted or implied by this Material or any information disclosed herein, including, but not limited to, any license to make, use, import or sell any product, service or technology offering embodying any of the information in this Material.

THIS MATERIAL IS BEING PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESSED, IMPLIED, STATUTORY OR OTHERWISE. TO THE MAXIMUM EXTENT PERMITTED BY LAW, QUALCOMM TECHNOLOGIES, ITS AFFILIATES AND/OR LICENSORS SPECIFICALLY DISCLAIM ALL WARRANTIES OF TITLE, MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, SATISFACTORY QUALITY, COMPLETENESS OR ACCURACY, AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MOREOVER, NEITHER QUALCOMM TECHNOLOGIES, NOR ANY OF ITS AFFILIATES AND/OR LICENSORS, SHALL BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY EXPENSES, LOSSES, USE, OR ACTIONS HOWSOEVER INCURRED OR UNDERTAKEN BY YOU IN RELIANCE ON THIS MATERIAL.

Certain product kits, tools and other items referenced in this Material may require You to accept additional terms and conditions before accessing or using those items.

Technical data specified in this Material may be subject to U.S. and other applicable export control laws. Transmission contrary to U.S. and any other applicable law is strictly prohibited.

Nothing in this Material is an offer to sell any of the components or devices referenced herein.

This Material is subject to change without further notification.

In the event of a conflict between these Terms of Use and the *Website Terms of Use* on www.qualcomm.com or the *Qualcomm Privacy Policy* referenced on www.qualcomm.com, these Terms of Use will control. In the event of a conflict between these Terms of Use and any other agreement (written or click-through, including, without limitation any non-disclosure agreement) executed by You and Qualcomm Technologies or a Qualcomm Technologies affiliate and/or licensor with respect to Your access to and use of this Material, the other agreement will control.

These Terms of Use shall be governed by and construed and enforced in accordance with the laws of the State of California, excluding the U.N. Convention on International Sale of Goods, without regard to conflict of laws principles. Any dispute, claim or controversy arising out of or relating to these Terms of Use, or the breach or validity hereof, shall be adjudicated only by a court of competent jurisdiction in the county of San Diego, State of California, and You hereby consent to the personal jurisdiction of such courts for that purpose.

2) Trademark and Product Attribution Statements.

Qualcomm is a trademark or registered trademark of Qualcomm Incorporated. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the U.S. and/or elsewhere. The Bluetooth® word mark is a registered trademark owned by Bluetooth SIG, Inc. Other product and brand names referenced in this Material may be trademarks or registered trademarks of their respective owners.

Snapdragon and Qualcomm branded products referenced in this Material are products of Qualcomm Technologies, Inc. and/or its subsidiaries. Qualcomm patented technologies are licensed by Qualcomm Incorporated.