Rockchip UEFI Developer Guide

ID: RK-KF-YF-935

Release Version: V1.4.0

Release Date: 2022-08-22

Security Level: □Top-Secret □Secret □Internal ■Public

DISCLAIMER

Trademark Statement

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Preface

Overview

As a developer guide for Rockchip Buildroot/Debian/Yocto Linux system, this document is intended to help software development engineers and technical support engineers get started with the development and debugging of Rockchip Linux platform faster.

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Revision History

Date	Author	Version	Change Description
2021-11-9	YiFeng Zhao	V1.0.0	Initial version
2022-01-12	Chris Zhong	V1.1.0	Boot to Linux system
2022-04-22	Chris Zhong	V1.2.0	Add GRUB boot Mode
2022-05-18	Chris Zhong	V1.3.0	Add PCIE support
2022-08-22	Chris Zhong	V1.4.0	Add Hardware Configuration Description

Contents

1. Code Introduction

The RK3588 Linux SDK has integrated UEFI code. Please check whether there is a uefi directory in the SDK. The related directories are as follows:

1. edk2-platforms/Silicon/Rockchip

```
├── Applications  //Test Demo and Tool
├── Drivers   //General driver
├── Include   //General header file
├── Library   //General library
├── RK3568   //is only for RK3568 IP driver, etc.
├── RK3588   //is only for RK3588 IP driver, etc.
├── Rockchip.dsc.inc   //General configuration
├── Rockchip.fdf.inc   //General configuration
└── RockchipPkg.dec    //General configuration
```

2. edk2-platforms/Platform/Rockchip

```
├── DeviceTree
│   └── rk3588.dtb   // Device tree file, copied from
kernel/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4-v10-linux.dtb
├── RK3568
└── RK3588
    ├── AcpiTables
    ├── BootGraphicsResourceTableDxe
    ├── Include
    ├── Library
    ├── LogoDxe
    ├── RK3588.dec
    ├── RK3588.dsc
    ├── RK3588Dxe
    ├── RK3588.fdf
    └── RK3588GpioDxe
```

2. UEFI Compilation

RK3588 Linux SDK provides the following two compilation methods.

Execute the script in the uefi directory:

```
./make.sh rk3588 #Build rk3588 by default
```

Or use the build.sh script in the SDK root directory:

```
./build.sh uefi
```

The environment variables of EDK2 have been configured in the script. During the compilation process, u-boot will be rebuilt. After the compilation, UEFI firmware: uboot_uefi.img and RK3588_NOR_FLASH.img will be generated in the current directory. Burn this file to the uboot partition. Please select uboot_uefi.img for the board booting from eMMC, and select RK3588_NOR_FLASH.img for the board booting from SPI Nor Flash.

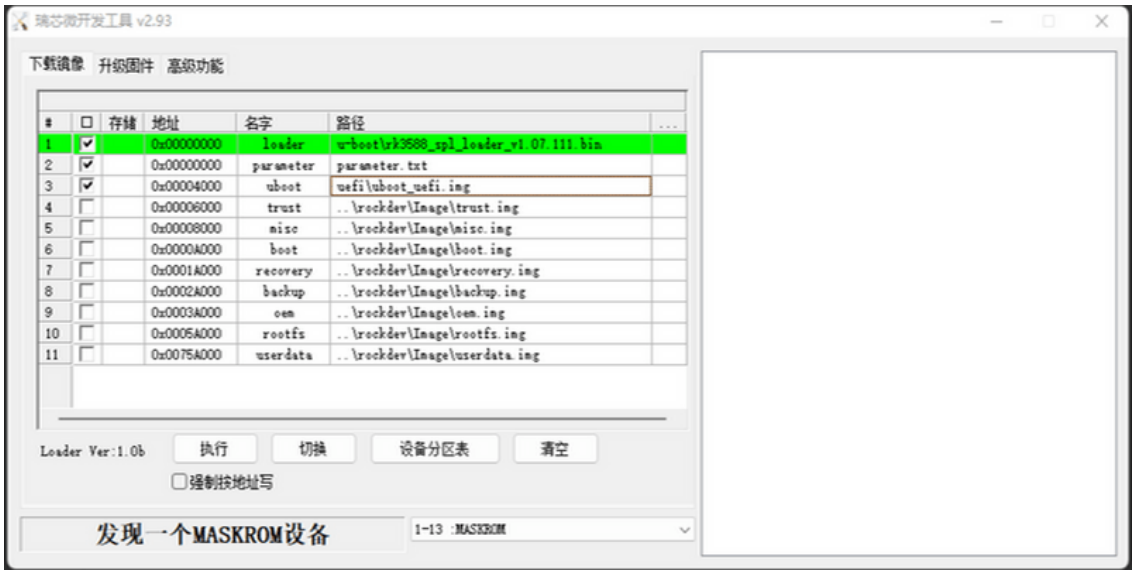
At present, RK3588 supports UEFI + DTB booting only, and does not support UEFI + ACPI booting mode. When build.sh uefi is compiling, the Kernel DTB file will be copied to the UEFI directory, so the Kernel compilation must be completed before compiling UEFI. If it is compiled with make.sh in the UEFI directory, you need to manually copy the dtb file to uefi/edk2-platforms/Platform/Rockchip/DeviceTree/rk3588.dtb:

```
$cp kernel/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4-v10-linux.dtb uefi/edk2-
platforms/Platform/Rockchip/DeviceTree/rk3588.dtb
```

3. Burn

The ways to burn firmware for the board booting from eMMC and SPI Nor Flash are different:

- Booting from eMMC



  The board booting from eMMC requires a parameter file, so at least, the following 3 files are required to be burned:

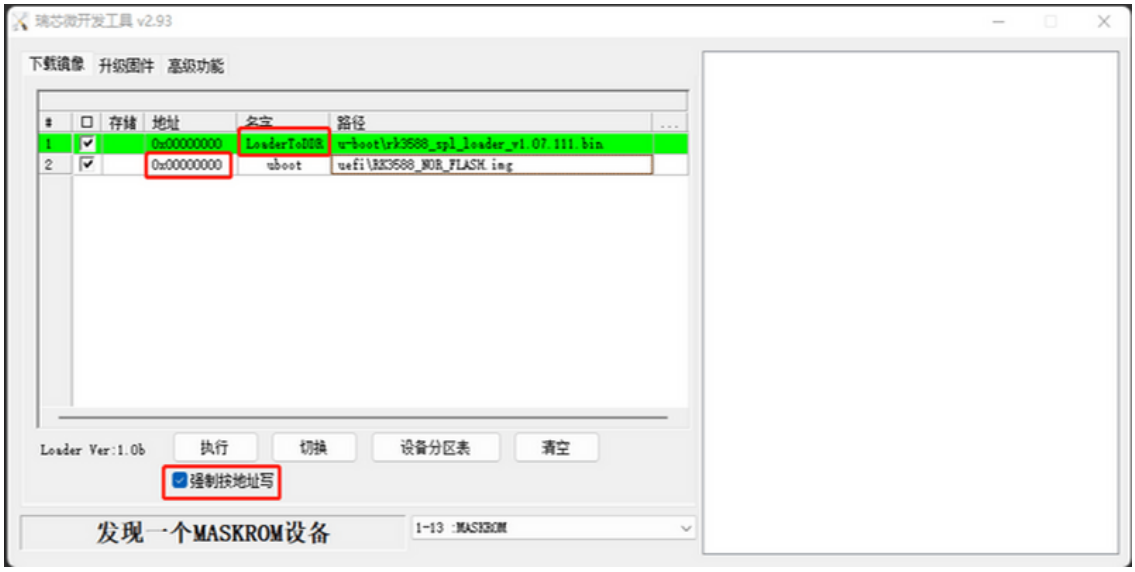
```
u-boot/rk3588_spl_loader_v1.07.111.bin
parameter.txt
uefi/uboot_uefi.img
```

The address of uboot_uefi.img is determined by the start address of uboot partition in the parameter.txt file.

The UEFI firmware burning address defined in the following parameter.txt is 0x4000, the size can not exceed 0x2000, with the unit 512 Byte.

```
FIRMWARE_VER: 1.0
MACHINE_MODEL: RK3588
MACHINE_ID: 007
MANUFACTURER: RK3588
MAGIC: 0x5041524B
ATAG: 0x00200800
MACHINE: 0xffffffff
CHECK_MASK: 0x80
PWR_HLD: 0,0,A,0,1
TYPE: GPT
CMDLINE:
mtdparts=rk29xxnand:0x00002000@0x00004000(uboot),0x00002000@0x00006000(misc),0x00
020000@0x00008000(boot),0x00040000@0x00028000(recovery),0x00010000@0x00068000(bac
kup),0x01c00000@0x00078000(rootfs),0x00040000@0x01c78000(oem),-
@0x01d18000(userdata:grow)
uuid:rootfs=614e0000-0000-4b53-8000-1d28000054a9
uuid:boot=7A3F0000-0000-446A-8000-702F00006273
```

- Booting from SPI Nor Flash



The way to burn firmware for SPI Nor Flash is different from eMMC, no parameter file is required:

```
boot/rk3588_spl_loader_v1.07.111.bin
uefi/RK3588_NOR_FLASH.img
```

Notice the red area in the picture above.

4. Booting

There are two ways to boot into the system: Android booting and Grub Booting, the following two booting modes will be introduced respectively.

4.1 Android Boot

The Android Boot mode is similar to RK3588 general Linux booting mode, the differences are:

- Replace the original uboot and boot with uboot_uefi.img and boot_uef.img, and keep other partitions unchanged;
- Add the UUID of the boot partition to the parameter.

  Run build.sh, kernel will build directly and generate boot_uefi.img for burning to the boot partition. The difference between boot_uefi.img and boot.img is that the second section packs dtb instead of resource.img.

4.1.1 Boot Partition Confirmation

The boot_uefi.img for UEFI booting requires GUID of GPT and other information to confirm the partition, so in addition to the general parameter file, the following line of information is required to specify the GUID of the boot partition:

```
uuid:boot=7A3F0000-0000-446A-8000-702F00006273
```

In addition, please confirm the offset and size of the boot partition. The detailed partition definition is the PcdAndroidBootDevicePath variable in uefi/edk2-platforms\Platform\Rockchip\RK3588\RK3588.dsc, please make sure that the UUID, offset, and size here should correspond to the parameter file one by one, otherwise UEFI will not find this partition, causing Android Boot to fail.

```
gEmbeddedTokenSpaceGuid.PcdAndroidBootDevicePath|L"VenHw(100C2CFA-B586-4198-9B4C-
1683D195B1DA)/HD(3,GPT,7A3F0000-0000-446A-8000-702F00006273,0x8000,0x20000)"
```

In the above definition of PcdAndroidBootDevicePath, the UUID of the boot partition is 7A3F0000-0000-446A-8000-702F00006273, the offset is 0x8000, and the size is 0x20000.

Through the map command of UEFI Shell, you can also see the whole partition table, and in which you will find the boot partition:

```
BLK3: Alias(s):
          VenHw(100C2CFA-B586-4198-9B4C-1683D195B1DA)/HD(3,GPT,7A3F0000-0000-446
A-8000-702F00006273,0x8000,0x20000)
```

Note: Android Boot will check whether there is Android header information in the Boot partition. Boot should be packaged in Android format instead of FIT format, otherwise it will prompt that Boot cannot be found. Using the script modified in the kernel patch will generate a boot_uefi.img file in Android format. The default Linux SDK compilation script will build boot.img in FIT format

4.2 Grub Booting

UEFI will find the /efi/boot/grubaa64.efi file of ESP partition, which is the application of GRUB. This file is is usually found in the system installation disk. The GRUB configuration file is grub.cfg, which configures information such as kernel and initrd. For more details, please refer to GRUB official documents and various system installation documents, such as "Debian_Install_Guide".

Here, we will briefly introduce how to build RK3588 kernel:

```
#Build kernel deb package
cd kernel
export CROSS_COMPILE=../prebuilts/gcc/linux-x86/aarch64/gcc-arm-10.3-2021.07-
x86_64-aarch64-none-linux-gnu/bin/aarch64-none-linux-gnu-
export ARCH=arm64
export LOCALVERSION=
export KDEB_PKGVERSION=5.10.84-1
make rockchip_linux_defconfig
make bindeb-pkg -j16

#Build dynamic libraries, which is needed in initrd
make modules
make modules_install INSTALL_MOD_PATH=$(realpath $CUR_DIR)/tmp
```

The following files will be generated in the SDK root directory, which can be used for kernel installation:

```
linux-headers-5.10.66_5.10.84-1_arm64.deb
linux-image-5.10.66_5.10.84-1_arm64.deb
linux-libc-dev_5.10.84-1_arm64.deb
```

In addition, the kernel/arch/arm64/boot/Image file in the kernel directory can be used to start the system installation disk:

```
cp kernel/arch/arm64/boot/Image /udisk/install.a64/vmlinuz
```

4.2.1 cmdline

When GRUB starts kernel, kernel will directly use the cmdline passed by GRUB instead of the cmdline carried in dts, so if you need to use ttyS as the debug serial port, you should modify grub.cfg as follows:

```
linux /install.a64/vmlinuz --- quiet
changed to
linux /install.a64/vmlinuz earlycon=uart8250,mmio32,0xfeb50000
console=ttyS2,1500000n8 --- quiet
```

If it is not specified in GRUB: console=ttyS2,1500000n8, kernel will use the default ttyFIQ0 as the debug port.

5. Configuration Setting

UEFI hardware-related configurations are generally in the following files:

```
make.sh //compilation switch
edk2-platforms/Platform/Rockchip/RK3588/RK3588.dsc //variable definition
edk2-platforms/Platform/Rockchip/RK3588/RK3588.fdf //Generate firmware Image
configuration
edk2-
platforms/Platform/Rockchip/RK3588/Library/RockchipPlatfromLib/RockchipPlatfromLi
b.c //Hardware related information
```

Currently, there is no unified GPIO driver for UEFI, so the general way is to write registers directly. Usually, a GPIO needs to modify two registers, IOMUX and GPIO

5.1 SPI Nor Configuration

Please configure the IO pins used by SPI Nor Flash according to the schematic diagram. The default configuration of the SDK is FSPI_M1. If it is not the same, please modify the NorFspiIomux function in edk2-platforms/Platform/Rockchip/RK3588/Library/RockchipPlatfromLib/RockchipPlatfromLib.c:

```
void
EFIAPI
NorFspiIomux(void)
{
  /* io mux */
  MmioWrite32(NS_CRU_BASE + CRU_CLKSEL_CON78,
              (((0x3 << 12) | (0x3f << 6)) << 16) | (0x0 << 12) | (0x3f << 6));
#define FSPI_M1
#if defined(FSPI_M0)
   /*FSPI M0*/
  BUS_IOC->GPIO2A_IOMUX_SEL_L = ((0xF << 0) << 16) | (2 << 0); //FSPI_CLK_M0
  BUS_IOC->GPIO2D_IOMUX_SEL_L = (0xFFFFUL << 16) | (0x2222);
//FSPI_D0_M0,FSPI_D1_M0,FSPI_D2_M0,FSPI_D3_M0
  BUS_IOC->GPIO2D_IOMUX_SEL_H = ((0xF << 8) << 16) | (0x2 << 8); //FSPI_CS0N_M0
#elif defined(FSPI_M1)
  /*FSPI M1*/
  BUS_IOC->GPIO2A_IOMUX_SEL_H = (0xFF00UL << 16) | (0x3300);
//FSPI_D0_M1,FSPI_D1_M1
  BUS_IOC->GPIO2B_IOMUX_SEL_L = (0xF0FFUL << 16) | (0x3033);
//FSPI_D2_M1,FSPI_D3_M1,FSPI_CLK_M1
  BUS_IOC->GPIO2B_IOMUX_SEL_H = (0xF << 16) | (0x3); //FSPI_CS0N_M1
#else
  /*FSPI M2*/
  BUS_IOC->GPIO3A_IOMUX_SEL_L = (0xFFFFUL << 16) | (0x5555); //[FSPI_D0_M2-
FSPI_D3_M2]
  BUS_IOC->GPIO3A_IOMUX_SEL_H = (0xF0UL << 16) | (0x50); //FSPI_CLK_M2
  BUS_IOC->GPIO3C_IOMUX_SEL_H = (0xF << 16) | (0x2); //FSPI_CS0_M2
#endif
}
```

To store parameters in SPI Nor, "emulated non-volatile variable mode" should be disabled.

```
diff --git a/edk2-platforms/Platform/Rockchip/RK3588/RK3588.dsc b/edk2-
platforms/Platform/Rockchip/RK3588/RK3588.dsc
index 1cf0ca85ff..6301a05a83 100644
--- a/edk2-platforms/Platform/Rockchip/RK3588/RK3588.dsc
+++ b/edk2-platforms/Platform/Rockchip/RK3588/RK3588.dsc
@@ -241,7 +241,7 @@
   #
   # Make VariableRuntimeDxe work at emulated non-volatile variable mode.
   #
-  gEfiMdeModulePkgTokenSpaceGuid.PcdEmuVariableNvModeEnable|TRUE
+  gEfiMdeModulePkgTokenSpaceGuid.PcdEmuVariableNvModeEnable|FALSE
```

In addition, due to EMMC and SPI_ M0 pin is reused, and if SPI Nor is enabled and FSPI_M0 is configured, EMMC communication errors will occur, please disable EMMC:

```
diff --git a/edk2-platforms/Platform/Rockchip/RK3588/RK3588.fdf b/edk2-
platforms/Platform/Rockchip/RK3588/RK3588.fdf
index 24c01a8463..f936c8ba34 100644
--- a/edk2-platforms/Platform/Rockchip/RK3588/RK3588.fdf
+++ b/edk2-platforms/Platform/Rockchip/RK3588/RK3588.fdf
@@ -279,7 +279,7 @@ READ_LOCK_STATUS   = TRUE
   #INF Silicon/Synopsys/DesignWare/Drivers/DwEmmcDxe/DwEmmcDxe.inf
   INF Silicon/Rockchip/Drivers/MmcDxe/MmcDxe.inf
   #INF Silicon/Rockchip/Drivers/DwEmmcDxe/DwEmmcDxe.inf
-  INF Silicon/Rockchip/Drivers/SdhciHostDxe/SdhciHostDxe.inf
+  #INF Silicon/Rockchip/Drivers/SdhciHostDxe/SdhciHostDxe.inf
```

5.2 PCIE Configuration

Check whether the compilation switch ROCKCHIP_PCIE30 is enabled in make.sh. If it is not enabled, please add the following modifications manually:

```
diff --git a/make.sh b/make.sh
index f8f8d68cee..3e35971a9a 100755
--- a/make.sh
+++ b/make.sh
@@ -13,6 +13,7 @@ case "$1" in
         *)
                 CHIP=3588
+                FLAGS+=" -D ROCKCHIP_PCIE30"
                 ;;
 esac
 echo Start to build rk$CHIP UEFI
```

In addition, please pay attention to the power supply and reset configuration of PCIE. The configuration function is in:

edk2-platforms/Platform/Rockchip/RK3588/Library/RockchipPlatfromLib/RockchipPlatfromLib.c

Please modify the register according to the schematic diagram and TRM.

```
void
EFIAPI
```

```
Pcie30IoInit(void)
{
    /* Set reset and power IO to gpio output mode */
    MmioWrite32(0xFD5F808C, 0xf << (8 + 16)); /* gpio4b6 to gpio mode -> reset */
    MmioWrite32(0xFEC50008, 0x40004000); /* output */

    MmioWrite32(0xFD5F8070, 0xf << (12 + 16)); /* gpio3c3 to gpio mode -> power
*/
    MmioWrite32(0xFEC4000c, 0x80008); /* output */
}

void
EFIAPI
Pcie30PowerEn(void)
{
    MmioWrite32(0xFEC40004, 0x80008); /* output high to enable power */
}

void
EFIAPI
Pcie30PeReset(BOOLEAN enable)
{
    if(enable)
        MmioWrite32(0xFEC50000, 0x40000000); /* output low */
    else
        MmioWrite32(0xFEC50000, 0x40004000); /* output high */
}
```

5.3 Display Configuration

The display function is disabled by default, if you want to turn it on, please modify the FLAGS in make.sh

```
diff --git a/make.sh b/make.sh
index f8f8d68cee..3e35971a9a 100755
--- a/make.sh
+++ b/make.sh
@@ -13,6 +13,7 @@ case "$1" in
         *)
                 CHIP=3588
+                FLAGS+=" -D ROCKCHIP_VOPEN"
                 ;;
 esac
 echo Start to build rk$CHIP UEFI
```

5.4 USB Configuration

The host function of USB2.0 and Type-c is enabled by default, just enable VBUS power supply, if customer's hardware is different from EVB, please modify UsbPortPowerEnable in:

edk2-platforms/Platform/Rockchip/RK3588/Library/RockchipPlatfromLib/RockchipPlatfromLib.c function:

```
#define GPIO4_BASE         0xFEC50000
```

```
#define GPIO_SWPORT_DR_L   0x0000
#define GPIO_SWPORT_DR_H   0x0004
#define GPIO_SWPORT_DDR_L  0x0008
#define GPIO_SWPORT_DDR_H  0x000C

void
EFIAPI
UsbPortPowerEnable (void)
{
  /* enable usb host vbus supply */
  MmioWrite32(GPIO4_BASE + GPIO_SWPORT_DR_L, (0x0100UL << 16) | 0x0100); /*
GPIO4_B0 */
  MmioWrite32(GPIO4_BASE + GPIO_SWPORT_DDR_L, (0x0100UL << 16) | 0x0100);
  /* enable usb otg0 vbus supply */
  MmioWrite32(GPIO4_BASE + GPIO_SWPORT_DR_H, (0x0100UL << 16) | 0x0100); /*
GPIO4_D0 */
  MmioWrite32(GPIO4_BASE + GPIO_SWPORT_DDR_H, (0x0100UL << 16) | 0x0100);
}
```