SOM-3566-OSM Debain11 Developer Guide

V1.0

Room 02-04, 10/F, Block A, Building 8, Shenzhen International Innovation Valley, Dashi Road,
Nanshan District, Shenzhen, Guangdong, China
Emai: support@geniatech.com Tel: (+ 86) 755 86028588

REVISION HISTORY

DATE	REVISION TYPE	REVISION #	COMMENTS	INITIALS
2024/05/15	Major	1.0	Initial version	ZX&JY

1. Build Instructions

1.1. Building images

This section describes the instructions to build the Board Support Package.
Before starting the build, run the command below on the Linux Host PC to install packages used for building the BSP.

```
sudo apt-get update && sudo apt-get install git ssh make gcc libssl-dev \
liblz4-tool expect expect-dev g++ patchelf chrpath gawk texinfo chrpath \
diffstat binfmt-support qemu-user-static live-build bison flex fakeroot \
cmake gcc-multilib g++-multilib unzip device-tree-compiler ncurses-dev \
libgucharmap-2-90-dev bzip2 expat gpgv2 cpp-aarch64-linux-gnu
```

Description:
The installation command is suitable for Ubuntu22.04.
For other versions, please adopt the corresponding installation command according to the name of the installation package. If compilation encounters an error, you can install the corresponding software package according to the error message.

2. SDK compilation

2.1. Select a configuration file

```
$ ./build.sh lunch          --- select 3
Pick a defconfig:

1. rockchip_defconfig
2. rockchip_rk3566_evb2_lp4x_v10_32bit_defconfig
3. rockchip_rk3566_evb2_lp4x_v10_defconfig
4. rockchip_rk3568_evb1_ddr4_v10_32bit_defconfig
5. rockchip_rk3568_evb1_ddr4_v10_defconfig
6. rockchip_rk3568_uvc_evb1_ddr4_v10_defconfig
Which would you like? [1]:
```

2.2. U-Boot compilation

```
<SDK>#./build.sh uboot
```

2.3. Kernel compilation

```
<SDK>#./build.sh kernel
```

2.4. Recovery compilation

```
<SDK>#./build.sh recovery
```

After compiling, record `output/rockchip_rk3566_recovery/images as recovery.img in BuildRoot.

Note: Recovery.img contains the Kernel (kernel.img), so every time the kernel changes, recovery needs to be repackaged as. Recovery repackaging method is as follows:

```
<SDK>#source buildroot/envsetup.sh
<SDK>#cd buildroot
<SDK>#make recovery-reconfigure
<SDK>#cd -
<SDK>#./build.sh recovery
```

2.5. Debian compilation

The compilation time is long.

```
<SDK>#./build.sh debian
```

Note: related dependency packages need to be pre-installed.
sudo apt-get install binfmt-support qemu-user-static live-build
sudo dpkg -i ubuntu-build-service/packages/*
sudo apt-get install -f

2.6. Pack

```
<SDK>#./build.sh firmware
<SDK>#./build.sh updateimg
```

2.7. Firmware path

```
<SDK>/output/firmware/
```

2.8. System partition description

- uboot partition : uboot.img for uboot compilation.
- misc partition : for misc.img, for recovery.
- boot partition : boot.img for kernel to compile.
- recovery partition : recovery.img for recovery compilation.
- backup partition : reserved, temporarily unavailable.
- rootfs partition : rootfs.img compiled by buildroot, debian or yocto.
- oem partition : give home ambassadors and store home apps or data. mount in the /oem directory.
- userdata partition : for APP to be temporarily built or finally made, and mounted in /user data.

3. Windows brush instructions

3.1. Rockchip USB driver installation

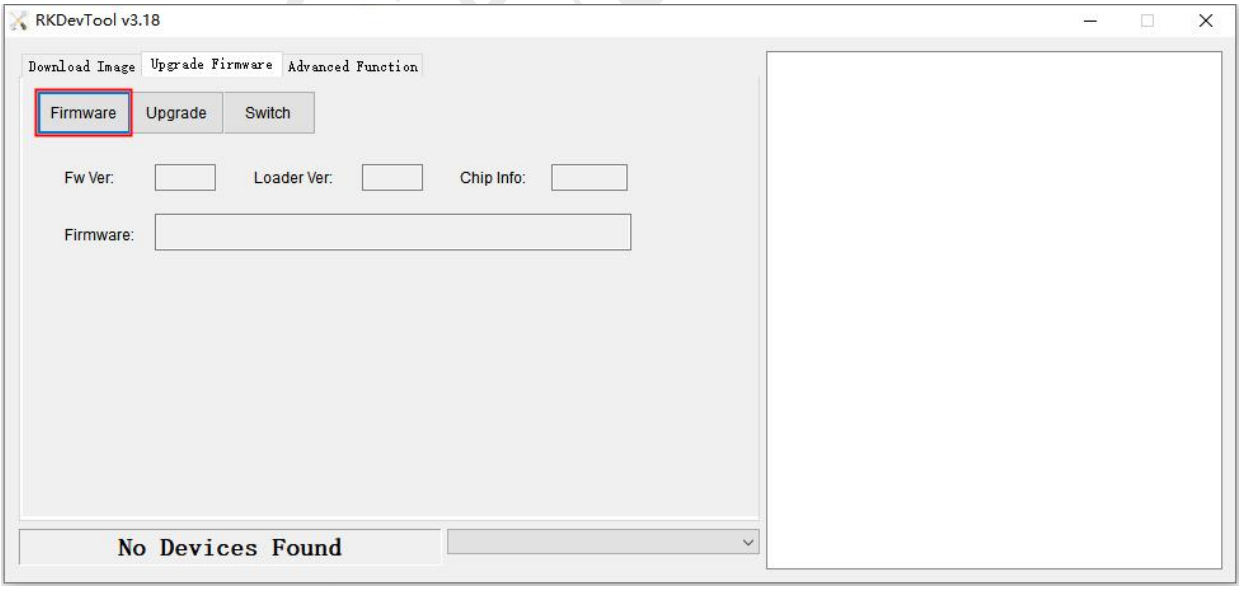
Rockchip USB driver installation aid is stored in Tools/Windows/DriverAsstant _ V5.x.zip.
Xp, win7_32, win7_64, win10_32, win10_64 and other operating systems.

3.2. Windows burner makes

The SDK provides Windows writing tools (version requires V3.15 or above), which are located in Cheng Genlu:

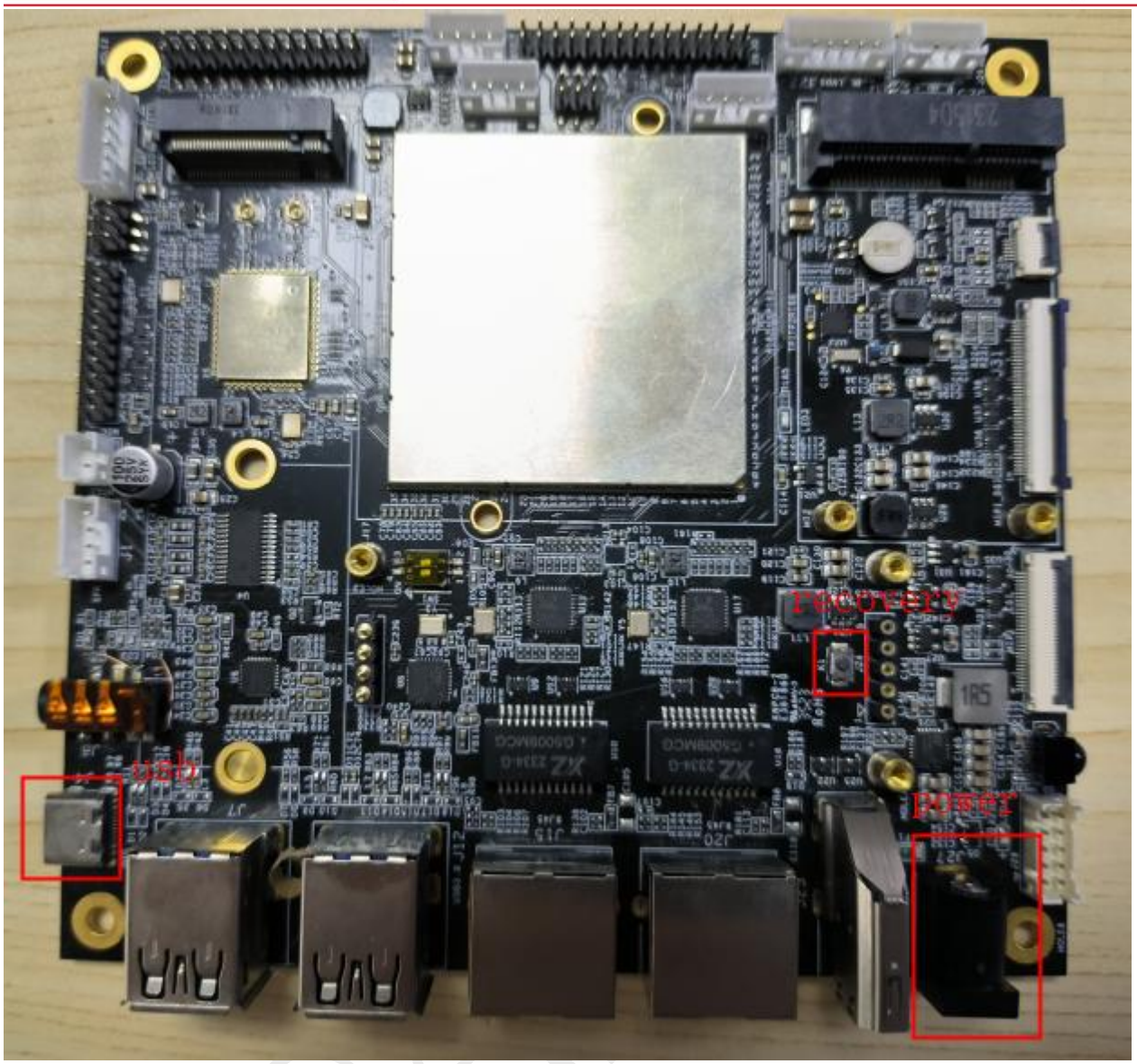
```
tools/
 ├— windows/RKDevTool
```

As shown in the figure below, after being compiled into the corresponding firmware, the device needs to enter the MASKROM or BootROM burning mode. After connecting the USB download cable, press and hold "Mask ROM" and press the reset key "RST" to release, then it can enter the Mask ROM mode. After loading the corresponding path compiled into the firmware, click "Execute" to enter the burning mode.



Note: Before writing, you need to install the latest USB driver. The driver details are as follows:

```
<SDK>/tools/windows/DriverAssitant_v5.12.zip
```


recovery
usb
power

4. SDK development

4.1. U-Boot development

The U-Boot startup process of the platform is as follows, and only some important steps are listed:

```
start.s
    //assembly environment
    => IRQ/FIQ/lowlevel/vbar/errata/cp15/gic          // ARM architecture-related lowlevel initialization
    => _main
        => stack                                      //Prepare the stack needed by C environment.
        //[Phase] C environment is initialized, and a series of function calls are initiated.
        => board_init_f: init_sequence_f[]
            initf_malloc
            arch_cpu_init                             //[lowlevel initialization of SOC]
            serial_init                               //Serial port initialization
            dram_init                                 //[Get ddr capacity information]
            reserve_mmu                               //reserve the memory to the low address from the end of ddr.
            reserve_video
            reserve_uboot
            reserve_malloc
            reserve_global_data
            reserve_fdt
            reserve_stacks
            dram_init_banksize
            sysmem_init
            setup_reloc                               //Determine the address of U-boot to reloc.
        //assembly environment
        => relocate_code                              //relocation of U-Boot code by assembly.
        //[Phase] C environment is initialized, and a series of function calls are initiated.
        => board_init_r: init_sequence_r[]
            initr_caches                              //Enable MMU and I/Dcache
            initr_malloc
            bidram_initr
            sysmem_initr
            initr_of_live                             //initialize of_live
            initr_dm                                  //Initialize the dm framework
            board_init                                //[Platform initialization, core part]
                board_debug_uart_init                 //configuration of serial iomux and clk
                init_kernel_dtb                       //[Cut to kernel dtb]!
                clks_probe                            //Initialize the system frequency
                regulators_enable_boot_on             //Initialize the system power.
                io_domain_init                        //io-domain initialization
                set_armclk_rate                       //__weak, ARM frequency boost (realized only when the platform needs it)
                dvfs_init                             //Frequency modulation and voltage regulation of wide temperature core
                rk_board_init                         // __weak, implemented by each specific platform.
            console_init_r
```

```
board_late_init                 //[Platform Lateinitialization]
    rockchip_set_ethaddr        //Set the mac address
    rockchip_set_serialno       //set serialno
    setup_boot_mode             //parsing "reboot xxx" command,
                                //Identify key and loader burning mode, recovery
    charge_display              // U-Boot charging
    rockchip_show_logo          //Display boot logo
    soc_clk_dump                //print clk tree
    rk_board_late_init          // __weak, implemented by each specific platform.
run_main_loop                   //[Enter command mode or execute startup command]
```

4.2. keyboard shortcuts

The platform provides serial key combination to trigger some events for debugging and writing. Press:

- Ctrl+c: enter u-boot command mode;
- Ctrl+d: enter loader burning mode;
- Ctrl+b: enter maskrom burning mode;
- Ctrl+f: enter fastboot mode;
- Ctrl+m: print bidram/system information;
- Ctrl+i: enable the kernel initcall_debug;
- Ctrl+p: print cmdline information;
- Ctrl+s: "starting kernel ..." followed by u-boot command;

4.3. Introduction to DTS

<sdk>/u-boot/arch/arm/dts/
├—rk3568-evb.dts
| ├— rk3568.dtsi
| └— rk3568-u-boot.dtsi

4.4. Introduction to CONFIGS

<sdk>/u-boot/configs/rk3568_defconfig

4.5. Kernel development

This section briefly introduces some common configuration modifications of the kernel, mainly DTS configuration, to help customers make some simple modifications faster and more conveniently.

4.6. Introduction to DTS

Linux kernel supports multi-platforms to store DTS and DTS of RK platforms in:
<sdk>/kernel/arch/arm64/boot/dts/rockchip
├—rk3566-evb2-lp4x-v10-linux.dts
| ├— rk3566-evb2-lp4x-v10.dtsi
| └— rk3568-linux.dtsi
| ├— rk3566.dtsi
| └— rk3566-evb.dtsi
| ├— rk3568-evb.dtsi
| └— rk3568.dtsi

4.7. Introduction to CONFIGS

<sdk>/kernel/arch/arm64/configs/rockchip_linux_defconfig