

IDO-EVB3568-V1 Android开发手册

注意事项

1 源码获取

1.1 整包源码获取

1.2 补丁获取及使用方法

2 开发环境配置

3 SDK编译

3.1 uboot编译步骤

3.2 kernel编译步骤

3.3 Android编译及固件生成步骤

3.4 固件打包



IDO-EVB3568-V1

Android开发手册

文档修订历史

版本	PCBA版本	修订内容	修订	审核	日期
V1.0	V1C	创建文档	WJY	IDO	2022/03/17
V1.1	V1D	修改开发主机环境配置说明	FYZ	IDO	2023/08/31
V1.2	V1D	补丁编译图片	FYZ	IDO	2024/08/08
V1.3	V1D	添加摄像头补丁及更新rkbin	HJT	IDO	2024/08/15

注意事项

源码的解压和编译使用普通用户即可，无需使用sudo或root用户。

1 源码获取

1.1 整包源码获取

链接: https://pan.baidu.com/s/1cc91gGOY87aLb9_DwbD0pQ

提取码: 1234

获取源码命令如下:

```
▼ Bash  
1  cat ido_evb3568_android11_211025_sdk.tgz.xa* > ido_evb3568_android11_211025_sdk.tgz  
2  tar xzvf ido_evb3568_android11_211025_sdk.tgz -C .
```

注意: ido_evb3568_android11_211025_sdk目录下有 Readme.txt 请必读。

1.2 补丁获取及使用方法

补丁文件获取链接

链接: <https://pan.baidu.com/s/1P7KzoJt2ZG4399FUO1yT7Q?pwd=1234>

提取码: 1234

patch目录下有readme.txt 有补丁及rkbin更新方法, 请必读。

2 开发环境配置

推荐编译主机配置如下:

1. Ubuntu18.04 操作系统
2. 64 位 CPU
3. 16GB 物理内存+交换内存
4. 250GB 空闲的磁盘空间

开发环境搭建, 请参考SDK根目录下

RKDocs\android\Rockchip_Developer_Guide_Android11_SDK_V1.1.5_CN.pdf文档中的 附录 A 编译开发环境搭建, 安装OpenJDK 8和一些编译依赖软件, Ubuntu18.04 通用软件安装包, 命令如下:

```

1 $ sudo apt-get update
2 $ sudo apt-get install git gnupg flex bison gperf libbsd1.2-dev libesd-jav
  a \
3 squashfs-tools build-essential zip curl libncurses-dev zlib1g-dev \
4 pngcrush schedtool libxml2 libxml2-utils xsltproc lzop libc6-dev schedtool
  g++-multilib \
5 lib32ncurses5-dev lib32readline-dev gcc-multilib libswitch-perl libssl-dev

```

结果如下图所示：

```

$ sudo apt-get update
获取:1 https://packages.microsoft.com/repos/code stable InRelease [3,590 B]
命中:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
命中:3 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy InRelease
命中:4 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy-updates InRelease
命中:5 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy-backports InRelease
已下载 3,590 B，耗时 1秒 (4,009 B/s)
正在读取软件包列表... 完成

$ sudo apt-get install git gnupg flex bison gperf libbsd1.2-dev libesd-jav \
> squashfs-tools build-essential zip curl libncurses-dev zlib1g-dev \
> pngcrush schedtool libxml2 libxml2-utils xsltproc lzop libc6-dev schedtool g++-multilib \
> lib32ncurses5-dev lib32readline-dev gcc-multilib libswitch-perl libssl-dev \
> unzip zip device-tree-compiler liblz4-tool python3-pyelftools -y
正在读取软件包列表... 完成
正在分析软件包的依赖关系树... 完成
正在读取状态信息... 完成
注意，选中 'lib32ncurses-dev' 而非 'lib32ncurses5-dev'
bison 已经是最新版 (2:3.8.2+dfsg-1build1)。
build-essential 已经是最新版 (12.9ubuntu3)。
device-tree-compiler 已经是最新版 (1.6.1-1)。
flex 已经是最新版 (2.6.4-8build2)。
lib32readline-dev 已经是最新版 (8.1.2-1)。
lzop 已经是最新版 (1.04-2build2)。
python3-pyelftools 已经是最新版 (0.27-1)。
squashfs-tools 已经是最新版 (1:4.5-3build1)。
zip 已经是最新版 (3.0-12build2)。
gperf 已经是最新版 (3.1-1build1)。
libesd-jav 已经是最新版 (0.0.7-5)。
liblz4-tool 已经是最新版 (1.9.3-2build2)。
libbsd1.2-dev 已经是最新版 (1.2.15+dfsg2-6)。
pngcrush 已经是最新版 (1.8.13-0.1)。
schedtool 已经是最新版 (1.3.0-4)。
curl 已经是最新版 (7.81.0-1ubuntu1.16)。
git 已经是最新版 (1:2.34.1-1ubuntu1.10)。
gnupg 已经是最新版 (2.2.27-3ubuntu2.1)。
lib32ncurses-dev 已经是最新版 (6.3-2ubuntu0.1)。
libc6-dev 已经是最新版 (2.35-0ubuntu3.7)。
libncurses-dev 已经是最新版 (6.3-2ubuntu0.1)。
libssl-dev 已经是最新版 (3.0.2-0ubuntu1.15)。
libxml2 已经是最新版 (2.9.13+dfsg-1ubuntu0.4)。
libxml2-utils 已经是最新版 (2.9.13+dfsg-1ubuntu0.4)。
unzip 已经是最新版 (6.0-26ubuntu3.2)。
xsltproc 已经是最新版 (1.1.34-4ubuntu0.22.04.1)。
zlib1g-dev 已经是最新版 (1:1.2.11.dfsg-2ubuntu9.2)。
下列软件包是自动安装的并且现在不需要了：
  binutils-arm-linux-gnueabi gcc-arm-linux-gnueabi gcc-11-arm-linux-gnueabi-base gcc-11-cross-base gcc-12-cross-base libasan6-armel-cross
  libatomic1-armel-cross libc6-armel-cross libc6-dev-armel-cross libgcc-11-dev-armel-cross libgcc-11-armel-cross libgomp1-armel-cross libstdc++6-armel-cross
  libubsan1-armel-cross linux-libc-dev-armel-cross
使用 'sudo apt autoremove' 来卸载它(它们)。
下列软件包将被【卸载】：
  gcc-11-arm-linux-gnueabi gcc-arm-linux-gnueabi
下列【新】软件包将被安装：
  g++-multilib gcc-multilib libswitch-perl
升级了 0 个软件包，新安装了 3 个软件包，要卸载 2 个软件包，有 9 个软件包未被升级。
需要下载 19.3 kB/20.6 kB 的归档。
解压后将会空出 53.6 MB 的空间。
获取:1 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy/universe amd64 libswitch-perl all 2.17-2.1 [18.3 kB]

```

3 SDK编译

3.1 uboot编译步骤

进入 sdk 根目录命令如下：

▼Shell

```
1 $ cd u-boot
2 $ ./make.sh rk3568
```

结果如下图所示：

```
## make rk3568_defconfig -j128
# configuration written to .config
#
arch64-linux-gnu/bin/aarch64-linux-gnu-gcc
scripts/kconfig/conf --silentoldconfig Kconfig
CHK include/config.h
CFG u-boot.cfg
GEN include/autoconf.mk.dep
CFG spl/u-boot.cfg
CFG tpl/u-boot.cfg
GEN tpl/include/autoconf.mk
GEN include/autoconf.mk
GEN spl/include/autoconf.mk
CHK include/config/u-boot.release
CHK include/generated/timestamp_autogenerated.h
UPD include/generated/timestamp_autogenerated.h
CHK include/generated/version_autogenerated.h
CHK include/generated/compile_options.h
```

uboot编译成功结果如下图所示：

```
optee
arch64-linux-gnu/bin/aarch64-linux-gnu-gcc
pack loader ok.(rk356x_spl_loader_v1.10.111.bin)(0.07)
pack loader okay! Input: /
Image(no-signed, version=0): uboot.img (FIT with uboot, trust...) is ready
Image(no-signed): rk356x spl loader v1.10.111.bin (with spl, ddr, usbplug) is ready
pack uboot.img okay! Input:
Platform RK3568 is build OK, with new .config(make rk3568_defconfig -j128)
2024年 08月 08日 星期四 11:58:32 CST
```

3.2 kernel编译步骤

内核配置文件路径：kernel/arch/arm64/configs/rockchip_defconfig

设备树文件路径：kernel/arch/arm64/boot/dts/rockchip/

内核配置及设备树文件名，如下表所示：

序号	文件名	功能说明
1	ido-evb3568-v1b-hdmi.dts	显示为HDMI
2	ido-evb3568-v1b-dsi1-1200-1920.dts	显示为MIPI 1200x1920
3	ido-rk3568-evb3568-v1b-dlvs-1080p-hdmi.dts	显示为双通道LVDS 1920x1080
4	ido-rk3568-evb3568-v1b-edp-1080p-hdmi.dts	显示为eDP 2lane 1920x1080
5	ido-rk3568-evb3568-v1b-lvds-1024-600-hdmi.dts	显示为LVDS 1024x600

以编译HDMI屏内核为例，编译命令如下：

```
1 $ cd kernel
2 $ make ARCH=arm64 rockchip_defconfig rk356x_evb.config android-11.config
3 $ make ARCH=arm64 ido-evb3568-v1b-hdmi.img -j10
```

编译结果如下图所示：

```
# configuration written to .config
#
Using .config as base
Merging ./kernel/configs/rk356x_evb.config
Value of CONFIG_MALI_MIDGARD is redefined by fragment ./kernel/configs/rk356x_evb.config:
Previous value: CONFIG_MALI_MIDGARD=y
New value: # CONFIG_MALI_MIDGARD is not set
Value of CONFIG_MALI_EXPERT is redefined by fragment ./kernel/configs/rk356x_evb.config:
Previous value: CONFIG_MALI_EXPERT=y
New value: # CONFIG_MALI_EXPERT is not set
Value of CONFIG_MALI_DEBUG is redefined by fragment ./kernel/configs/rk356x_evb.config:
New value: # CONFIG_MALI_DEBUG is not set

scripts/kconfig/conf --syncconfig Kconfig
HOSTCC scripts/extract-cert
DTC arch/arm64/boot/dts/rockchip/ido-evb3568-v1b-hdmi.dtb
CALL scripts/checksyscalls.sh
CHK include/generated/compile.h
EXTRACT_CERTS
AS certs/system_certificates.o
AR certs/built-in.a
GZIP kernel/config_data.gz
CHK kernel/kheaders_data.tar.xz
fatal: 不是一个 git 仓库（或者直至挂载点 /mnt 的任何父目录）
停止在文件系统边界（未设置 GIT_DISCOVERY_ACROSS_FILESYSTEM）。
GEN kernel/kheaders_data.tar.xz
CC kernel/kheaders.o
UPD kernel/config_data.h
CC kernel/configs.o
AR kernel/built-in.a
GEN .version
CHK include/generated/compile.h
```

注意：如果编译提示选择io_domain电压，vccio4和vccio6选择1.8V，其它选择3.3V。

以上方式编译完成后，kernel目录生成的boot.img文件不能直接烧录。需要使用build.sh -K 命令来编译kernel，编译后烧写rockdev/Image-rk3568_r/boot.img。

单独编译kernel生成可直接烧录的boot.img

此处的编译方法的前提已存在rockdev/Image-rk3568_r/boot.img文件（即Android代码已经完全编译过一次或者执行过build.sh -K）。

编译的原理：在kernel目录下将编译生成的 kernel.img 和 resource.img 替换到旧的 boot.img 中，命令如下：

```
1 $ cd kernel
2 $ make ARCH=arm64 rockchip_defconfig rk356x_evb.config android-11.config
3 $ make ARCH=arm64 BOOT_IMG=../rockdev/Image-rk3568_r/boot.img ido-evb3568-v1b-hdmi.img -j10
```

kernel编译结果，如下图所示：

```
Building modules, stage 2.
MODPOST 3 modules
found ./arch/arm64/boot/dts/rockchip/.ido-evb3568-v1b-hdmi.dtb.dts.tmp
found ./arch/arm64/boot/dts/rockchip/.ido-evb3568-v1b-hdmi.dtb.dts.tmp
found ./arch/arm64/boot/dts/rockchip/.ido-evb3568-v1b-hdmi.dtb.dts.tmp
found ./arch/arm64/boot/dts/rockchip/.ido-evb3568-v1b-hdmi.dtb.dts.tmp
found ./arch/arm64/boot/dts/rockchip/.ido-evb3568-v1b-hdmi.dtb.dts.tmp
found ./arch/arm64/boot/dts/rockchip/.ido-evb3568-v1b-hdmi.dtb.dts.tmp
found ./arch/arm64/boot/dts/rockchip/.ido-evb3568-v1b-hdmi.dtb.dts.tmp
Image: resource.img (with ido-evb3568-v1b-hdmi.dtb logo.bmp logo_kernel.bmp) is ready
Image: boot.img (with Image resource.img) is ready
Image: zboot.img (with Image.lz4 resource.img) is ready
```

使用此方法编译出kernel/boot.img文件可以直接用于烧录至boot分区。

3.3 Android编译及固件生成步骤

▼ Shell

```
1 $ source build/envsetup.sh
2 $ lunch rk3568_r-userdebug
3 $ make -j10
```

结果如下图所示：

```
/ido_evb3568_android11_211025_sdk$ source build/envsetup.sh
/ido_evb3568_android11_211025_sdk$ lunch rk3568_r-userdebug

=====
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=11
TARGET_PRODUCT=rk3568_r
TARGET_BUILD_VARIANT=userdebug
TARGET_BUILD_TYPE=release
TARGET_ARCH=arm64
TARGET_ARCH_VARIANT=armv8-a
TARGET_CPU_VARIANT=cortex-a55
TARGET_2ND_ARCH=arm
TARGET_2ND_ARCH_VARIANT=armv8-2a
TARGET_2ND_CPU_VARIANT=cortex-a55
HOST_ARCH=x86_64
HOST_2ND_ARCH=x86
HOST_OS=linux
HOST_OS_EXTRA=Linux-5.4.0-90-generic-x86_64-Ubuntu-18.04.6-LTS
HOST_CROSS_OS=windows
HOST_CROSS_ARCH=x86
HOST_CROSS_2ND_ARCH=x86_64
HOST_BUILD_TYPE=release
BUILD_ID=RQ3A.210705.001
OUT_DIR=out
=====
```

Android编译成功结果如下图所示：

```
[ 99% 295/296] finishing build rules ...
packages/apps/Camera2/Android.mk: warning: "Camera2 (app:platform) should not link against libjni_jpegutil (native:ndk:libc++:static)"
[100% 363/363] Target super fs image for debug: out/target/product/rk3568_r/super.img
2024-08-08 13:04:13 - build_super_image.py - INFO : Building super image from info dict...
2024-08-08 13:04:13 - sparse_img.py - INFO : Total of 241990 4096-byte output blocks in 18 input chunks.
2024-08-08 13:04:13 - sparse_img.py - INFO : Total of 30217 4096-byte output blocks in 3 input chunks.
2024-08-08 13:04:13 - sparse_img.py - INFO : Total of 59057 4096-byte output blocks in 7 input chunks.
2024-08-08 13:04:13 - sparse_img.py - INFO : Total of 74228 4096-byte output blocks in 6 input chunks.
2024-08-08 13:04:13 - sparse_img.py - INFO : Total of 154 4096-byte output blocks in 2 input chunks.
2024-08-08 13:04:13 - common.py - INFO : Running: "lpmake --metadata-size 65536 --super-name super --metadata-slots 2 --device super:3263168512 --group rockchip_dynamic_partitions:3258974208 --partition system:readonly:991191040:rockchip_dynamic_partitions --image system=out/target/product/rk3568_r/system.img --partition system_ext:readonly:123768832:rockchip_dynamic_partitions --image system_ext=out/target/product/rk3568_r/system_ext.img --partition vendor:readonly:241897472:rockchip_dynamic_partitions --image vendor=out/target/product/rk3568_r/vendor.img --partition product:readonly:304037888:rockchip_dynamic_partitions --image product=out/target/product/rk3568_r/product.img --partition odm:readonly:630784:rockchip_dynamic_partitions --image odm=out/target/product/rk3568_r/odm.img --sparse --output out/target/product/rk3568_r/super.img"
2024-08-08 13:04:33 - common.py - INFO : lpmake I 08-08 13:04:13 837 837 builder.cpp:1031 [liblp]Partition system will resize from 0 bytes to 991191040 bytes
lpmake I 08-08 13:04:13 837 837 builder.cpp:1031 [liblp]Partition system_ext will resize from 0 bytes to 123768832 bytes
lpmake I 08-08 13:04:13 837 837 builder.cpp:1031 [liblp]Partition vendor will resize from 0 bytes to 241897472 bytes
lpmake I 08-08 13:04:13 837 837 builder.cpp:1031 [liblp]Partition product will resize from 0 bytes to 304037888 bytes
lpmake I 08-08 13:04:13 837 837 builder.cpp:1031 [liblp]Partition odm will resize from 0 bytes to 630784 bytes
2024-08-08 13:04:33 - build_super_image.py - INFO : Done writing image out/target/product/rk3568_r/super.img
#### build completed successfully (06:29 (mm:ss)) ####
```

版本选择：rk3568_r-userdebug

需要编译内核选择的屏幕修改Android主显和系统方向，如下表所示：

屏幕	系统配置参数修改
HDMI	- device/rockchip/rk356x/rk3568_r/rk3568_r.mk PRODUCT_PROPERTY_OVERRIDES += vendor.hwc.device.primary=HDMI-A-1 - device/rockchip/rk356x/BoardConfig.mk SF_PRIMARY_DISPLAY_ORIENTATION := 0
MIPI	- ddevice/rockchip/rk356x/rk3568_r/rk3568_r.mk PRODUCT_PROPERTY_OVERRIDES += vendor.hwc.device.primary=DSI - device/rockchip/rk356x/BoardConfig.mk SF_PRIMARY_DISPLAY_ORIENTATION := 270
eDP	- ddevice/rockchip/rk356x/rk3568_r/rk3568_r.mk PRODUCT_PROPERTY_OVERRIDES += vendor.hwc.device.primary=eDP - device/rockchip/rk356x/BoardConfig.mk SF_PRIMARY_DISPLAY_ORIENTATION := 0
LVDS	- ddevice/rockchip/rk356x/rk3568_r/rk3568_r.mk PRODUCT_PROPERTY_OVERRIDES += vendor.hwc.device.primary=LVDS - device/rockchip/rk356x/BoardConfig.mk SF_PRIMARY_DISPLAY_ORIENTATION := 0

3.4 固件打包

编译完成后，执行 SDK 根目录下的 mkimage.sh 脚本生成固件，所有烧写所需的各分区镜像将都 rockdev/Image-rk3568_r/目录下，命令如下：

▼ Shell |

```
1 $ ./mkimage.sh
```

结果如下图所示：

```
.../ido_evb3568_android11_211025_sdk$ ./mkimage.sh
TARGET_PRODUCT=rk3568_r
TARGET_BASE_PARAMETER_IMAGE==device/rockchip/common/baseparameter/v2.0/baseparameter.img
HIGH_RELIABLE_RECOVERY_OTA=
BOARD_AVB_ENABLE=false
system filesystem is ext4
create dtbo.img...
done.
create resource.img...
done.
create boot.img...
done.
create boot-debug.img...
done.
skip copy images: vendor_boot.img
skip copy images: vendor_boot-debug.img
create recovery.img...
done.
create super.img...
done.
create vbmeta.img...
BOARD_AVB_ENABLE is false, use default vbmeta.img
create misc.img.... done.
create uboot.img...
u-boot/trust.img not found! Please make it from u-boot first!
create loader...
create config.cfg...
create baseparameter...done.
```

将所有分区镜像合并成单个的镜像，命令如下：

▼ Bash |

```
1 $ cd RKTools/linux/Linux_Pack_Firmware/rockdev/
2 $ ./mkupdate_rk356x.sh
```

结果如下图所示：

```

start to make update.img...
Android Firmware Package Tool v2.0
----- PACKAGE -----
Add file: ./package-file
package-file,Add file: ./package-file done,offset=0x800,size=0x2ba,userspace=0x1
Add file: ./Image/MiniLoaderAll.bin
bootloader,Add file: ./Image/MiniLoaderAll.bin done,offset=0x1000,size=0x719c0,userspace=0xe4
Add file: ./Image/parameter.txt
parameter,Add file: ./Image/parameter.txt done,offset=0x73000,size=0x282,userspace=0x1
Add file: ./Image/uboot.img
uboot,Add file: ./Image/uboot.img done,offset=0x73800,size=0x400000,userspace=0x800
Add file: ./Image/misc.img
misc,Add file: ./Image/misc.img done,offset=0x473800,size=0xc000,userspace=0x18
Add file: ./Image/boot.img
boot,Add file: ./Image/boot.img done,offset=0x47f800,size=0x2114800,userspace=0x4229
Add file: ./Image/dtbo.img
dtbo,Add file: ./Image/dtbo.img done,offset=0x2594000,size=0x26f,userspace=0x1
Add file: ./Image/vbmeta.img
vbmeta,Add file: ./Image/vbmeta.img done,offset=0x2594800,size=0x1000,userspace=0x2
Add file: ./Image/recovery.img
recovery,Add file: ./Image/recovery.img done,offset=0x2595800,size=0x44af000,userspace=0x895e
Add file: ./Image/baseparameter.img
baseparameter,Add file: ./Image/baseparameter.img done,offset=0x6a44800,size=0x100000,userspace=0x200
Add file: ./Image/super.img
super,Add file: ./Image/super.img done,offset=0x6b44800,size=0x62232dc8,userspace=0xc4466
Add CRC...
Make firmware OK!
----- OK -----
*****rkImageMaker ver 2.0*****
Generating new image, please wait...
Writing head info...
Writing boot file...
Writing firmware...
Generating MD5 data...
MD5 data generated successfully!
New image generated successfully!
Making update.img OK.

```

执行mkupdate_rk356x.sh 命令后会将各分区镜像合并成一个update.img 的镜像文件，如下图所示：

```

afptool mkupdate_px30.sh mkupdate_rk3328.sh mkupdate_rk356x_box.sh package-file-box readme.txt
gen-package-file.sh mkupdate_rk3126c.sh mkupdate_rk3368.sh mkupdate_rk356x.sh package-file-rk356x rkImageMaker
Image mkupdate_rk322x.sh mkupdate_rk3399pro.sh mkupdate.sh package-file-rk356x-ab unpack.sh
mkupdate_ab_rk356x_box_32.sh mkupdate_rk3288.sh mkupdate_rk3399.sh package-file package-file-rk356x-box update.img
mkupdate_ab rk356x_box.sh mkupdate rk3326.sh mkupdate rk356x_box 32.sh package-file-ab package-file-rk356x-box-ab

```