IDO-EVB3020 Linux SDK编译手册

概述

SOM3020核心板的主芯片有两款：PX30和K3358，请根据相应的CPU型号下载相应的SDK。

一、SDK获取

链接：https://pan.baidu.com/s/1Ok_GJyzEIV23cjFq4W7tOA
提取码：1234

解压SDK
PX30：

```bash
tar -xzf px30_linux_evb3020.tar.gz -C your/target/path
```

将SDK压缩包解压到你指定的路径，加上-v 可以显示解包的具体过程。

RK3358：

```shell
cat rk3358-linux-sdk-220722.tar.gz.0* > rk3358-linux-sdk-220722.tar.gz
tar -xvf rk3358-linux-sdk-220722.tar.gz
```

二、搭建SDK编译环境

在解压的文件夹里，参考 SDK_PATH/docs/Rockchip_Developer_Guide_Linux_Software_CN.pdf，安装编译所依赖的软件包（详见4-2页）。

三、SDK编译配置选择

EVB3020的BoardConfig 文件位于 device/rockchip/px30/ 和 device/rockchip/rk3358/目录，配置文件说明如下表所示：

名称	说明
BoardConfig_ido_evb3020_lvds.mk	IDO-EVB3020（px30）开发板，搭配7寸LVDS显示屏
BoardConfig_ido_evb3020_mipi.mk	IDO-EVB3020（px30）开发板，搭配10.1寸MIPI显示屏
BoardConfig-rk3358-ido-evb3020-lvds.mk	IDO-SOM3020（rk3358）
BoardConfig-rk3358-ido-evb3020-mipi.mk	IDO-SOM3020（rk3358）

这里以PX30为例，开发板选择配置文件。
选择编译EVB3020 LVDS配置

```bash
cd px30_linux_evb3020/
./build.sh device/rockchip/px30/BoardConfig_ido_evb3020_lvds.mk
```

选择编译EVB3020 MIPI配置

```bash
cd px30_linux_evb3020/
./build.sh device/rockchip/px30/BoardConfig_ido_evb3020_mipi.mk
```

成功配置显示，以LVDS为例：

```Bash
processing option: device/rockchip/px30/BoardConfig_ido_evb3020_lvds.mk
switching to board: SDK_PATH/px30/BoardConfig_ido_evb3020_lvds.mk
```

四、Uboot 编译

```Bash
### U-Boot build command
./build.sh uboot
### or
cd u-boot/
./make.sh evb-px30
```

```
  OBJCOPY tpl/u-boot-tpl-nodtb.bin
  COPY    tpl/u-boot-tpl.bin
  LD      spl/drivers/mmc/built-in.o
  LD      spl/lib/libfdt/built-in.o
  LD      spl/drivers/clk/rockchip/built-in.o
  LD      spl/lib/built-in.o
  LD      spl/drivers/clk/built-in.o
  LD      spl/common/built-in.o
  LD      spl/drivers/core/built-in.o
  LD      spl/drivers/built-in.o
  LD      spl/u-boot-spl
  OBJCOPY spl/u-boot-spl-nodtb.bin
  CAT     spl/u-boot-spl-dtb.bin
  COPY    spl/u-boot-spl.bin
  CFGCHK  u-boot.cfg

 load addr is 0x200000!
pack input ./u-boot.bin
pack file size: 871652(851 KB)
crc = 0x3df8abb9
uboot version: U-Boot 2017.09 (Jan 05 2022 - 18:34:05)
pack uboot.img success!
pack uboot okay! Input: ./u-boot.bin
out:px30_loader_v1.14.120.bin
fix opt:px30_loader_v1.14.120.bin
merge success(px30_loader_v1.14.120.bin)
pack loader okay! Input: [redacted]/rkbin/RKBOOT/PX30MINIALL.ini
[redacted]3/u-boot
ls: cannot access 'trust*.img': No such file or directory
out:trust.img
merge success(trust.img)
[redacted]/u-boot
pack trust okay! Input: [redacted]/rkbin/RKTRUST/PX30TRUST.ini


Platform PX30 is build OK, with new .config(make evb-px30_defconfig)
[redacted]3
====Build uboot ok!====
```

成功显示Build uboot ok!

五、Kernel 编译

内核 dts 位于 /kernel/arch/arm64/boot/dts/rockchip/ 目录

名称	说明
px30-linux-ido-evb3020-v1c-lvds.dts	IDO EVB3020 V1C开发板，搭配7寸LVDS显示屏设备树
px30-linux-ido-evb3020-v1c-mipi.dts	IDO EVB3020 V1C开发板，搭配10.1寸MIPI显示屏设备树

编译方法

```Bash
### kernel build command
./build.sh kernel
### or
cd kernel/
make ARCH=arm64 px30_linux_ido_evb3020_defconfig
make ARCH=arm64 px30-linux-ido-evb3020-v1c-lvds.img -j10
```

```
  SYSMAP  System.map
  OBJCOPY arch/arm64/boot/Image
  LZ4C    arch/arm64/boot/Image.lz4
  Image:  kernel.img is ready
  CHK     include/config/kernel.release
  CHK     include/generated/uapi/linux/version.h
  CHK     include/generated/utsrelease.h
  CHK     scripts/mod/devicetable-offsets.h
  CHK     include/generated/timeconst.h
  CHK     include/generated/bounds.h
  CHK     include/generated/asm-offsets.h
  CALL    scripts/checksyscalls.sh
make[2]: 'include/generated/vdso-offsets.h' is up to date.
  Building modules, stage 2.
  MODPOST 0 modules
  Image:  resource.img (with px30-linux-ido-evb3020-v1b-lvds.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
====Build kernel ok!====
```

成功显示Build kernel ok!

六、recovery 编译

```Bash
./build.sh recovery
```

```
2022-01-06T12:05:25 >>> host-xz 5.2.3 Patching libtool
2022-01-06T12:05:25 >>> host-xz 5.2.3 Configuring
2022-01-06T12:05:30 >>> host-xz 5.2.3 Building
2022-01-06T12:05:34 >>> host-xz 5.2.3 Installing to host directory
2022-01-06T12:05:37 >>> host-squashfs 3de1687d7432ea9b302c2db9521996f506c140a3 Extracting
2022-01-06T12:05:37 >>> host-squashfs 3de1687d7432ea9b302c2db9521996f506c140a3 Patching
2022-01-06T12:05:38 >>> host-squashfs 3de1687d7432ea9b302c2db9521996f506c140a3 Configuring
2022-01-06T12:05:38 >>> host-squashfs 3de1687d7432ea9b302c2db9521996f506c140a3 Building
2022-01-06T12:05:41 >>> host-squashfs 3de1687d7432ea9b302c2db9521996f506c140a3 Installing to host directory
2022-01-06T12:05:44 >>>   Finalizing target directory
2022-01-06T12:05:44 >>>   Sanitizing RPATH in target tree
2022-01-06T12:05:45 >>>   Copying overlay board/rockchip/common/base
2022-01-06T12:05:45 >>>   Copying overlay board/rockchip/common/recovery
2022-01-06T12:05:45 >>>   Executing post-build script build/post.sh
2022-01-06T12:05:45 >>>   Generating root filesystem image rootfs.cpio
2022-01-06T12:05:47 >>>   Generating root filesystem image rootfs.ext2
2022-01-06T12:05:49 >>>   Generating root filesystem image rootfs.squashfs
2022-01-06T12:05:49 >>>   Generating root filesystem image rootfs.tar
Done in 33min 37s
log saved on [redacted]/br.log
====Build rockchip_px30_recovery ok!====
pack recovery.img...done.
you take 44:03.48 to build recovery
====Build recovery ok!====
```

成功显示Build recovery ok！不过编译时间比较长，请耐心等待。

七、Buildroot 编译

Bash

```
./build.sh buildroot
```

```
2022-01-06T16:11:44 >>> host-makedevs  Installing to host directory
2022-01-06T16:11:50 >>> host-lz4 v1.7.5 Extracting
2022-01-06T16:11:50 >>> host-lz4 v1.7.5 Patching
2022-01-06T16:11:50 >>> host-lz4 v1.7.5 Configuring
2022-01-06T16:11:51 >>> host-lz4 v1.7.5 Building
2022-01-06T16:12:03 >>> host-lz4 v1.7.5 Installing to host directory
2022-01-06T16:12:10 >>> host-lzo 2.10 Extracting
2022-01-06T16:12:10 >>> host-lzo 2.10 Patching
2022-01-06T16:12:10 >>> host-lzo 2.10 Configuring
2022-01-06T16:12:24 >>> host-lzo 2.10 Building
2022-01-06T16:12:31 >>> host-lzo 2.10 Installing to host directory
2022-01-06T16:12:37 >>> host-squashfs 3de1687d7432ea9b302c2db9521996f506c140a3 Extracting
2022-01-06T16:12:37 >>> host-squashfs 3de1687d7432ea9b302c2db9521996f506c140a3 Patching
2022-01-06T16:12:37 >>> host-squashfs 3de1687d7432ea9b302c2db9521996f506c140a3 Configuring
2022-01-06T16:12:37 >>> host-squashfs 3de1687d7432ea9b302c2db9521996f506c140a3 Building
2022-01-06T16:12:44 >>> host-squashfs 3de1687d7432ea9b302c2db9521996f506c140a3 Installing to host directory
2022-01-06T16:12:50 >>>   Finalizing target directory
2022-01-06T16:12:54 >>>   Sanitizing RPATH in target tree
2022-01-06T16:13:09 >>>   Copying overlay board/rockchip/common/base
2022-01-06T16:13:09 >>>   Copying overlay board/rockchip/common/wifi
2022-01-06T16:13:09 >>>   Copying overlay board/rockchip/common/qt
2022-01-06T16:13:09 >>>   Copying overlay board/rockchip/px30/fs-overlay-64/
2022-01-06T16:13:09 >>>   Executing post-build script build/post.sh
2022-01-06T16:13:09 >>>   Generating root filesystem image rootfs.cpio
2022-01-06T16:13:39 >>>   Generating root filesystem image rootfs.ext2
2022-01-06T16:13:42 >>>   Generating root filesystem image rootfs.squashfs
2022-01-06T16:13:45 >>>   Generating root filesystem image rootfs.tar
Done in 2h 15min 46s
log saved on [redacted]/br.log. pack buildroot image at: [redacted]/buildroo
t/output/rockchip_px30_64/images/rootfs.ext4
you take 2:21:19 to build buildroot
====Build buildroot ok!====
```

成功显示Build buildroot ok！编译时间比较长，同样需要耐心等待。

八、自动编译

```Bash
./build.sh all
```

all –build uboot, kernel, rootfs, recovery image
简而言之，就是一条命令自动执行上面的全部编译过程

九、固件打包

执行下方命令，将编译好的各分区固件打包至 rockdev 目录

```Bash
./build.sh firmware
```

```
tune2fs 1.43.9 (8-Feb-2018)
Setting maximal mount count to -1
Setting interval between checks to 0 seconds
create uboot.img...done.
create trust.img...done.
create loader...done.
create boot.img...done.
                                                    3/device/rockchip/px30/parameter.txt
0x00002000@0x00004000(uboot),0x00002000@0x00006000(trust),0x00002000@0x00008000(misc),0x00010000@0x0000a000(boot),0x00010000@0x0
001a000(recovery),0x00010000@0x0002a000(backup),0x00020000@0x0003a000(oem),0x00da0000@0x0005a000(rootfs),-@0x00dfa000(userdata:g
row)
 Image: image in rockdev is ready
Make image ok!
```

执行下方命令，将各个分区镜像合成一个完整的烧录包

```Bash
./build.sh updateimg
```

```
processing option: updateimg
Make update.img
start to make update.img...
Android Firmware Package Tool v1.65
------ PACKAGE ------
Add file: ./package-file
Add file: ./Image/MiniLoaderAll.bin
Add file: ./Image/parameter.txt
Add file: ./Image/trust.img
Add file: ./Image/uboot.img
Add file: ./Image/misc.img
Add file: ./Image/boot.img
Add file: ./Image/recovery.img
Add file: ./Image/rootfs.img
Add file: ./Image/oem.img
Add file: ./Image/userdata.img
Add CRC...
Make firmware OK!
------ OK ------
********RKImageMaker ver 1.66********
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.
Make update image ok!
```

执行完后可在rockdev 目录获得一个用于烧录的完整固件包文件 update.img

```
@wt_rd_server:~/                                  $ ls rockdev/
boot.img           oem.img        rootfs.ext4  uboot.img
MiniLoaderAll.bin  parameter.txt  rootfs.img   update.img
misc.img           recovery.img   trust.img    userdata.img
```

如需要打包Ubuntu文件系统的固件，请更新网盘SDK中的Ubuntu-rootfs目录下的文件系统。

1. SDK下新建Ubuntu目录

```
mkdir ubuntu
```

2. 将文件系统拷贝至Ubuntu目录下

```
                                                  /px30/px30_linux/ubuntu/
                ubuntu20.04-Desktop-evb3020.img  ubuntu-base-20.04-arm64-evb3020.img
```

3. 链接Ubuntu文件系统

```
ln -rsf ubuntu/ubuntu-base-20.04-arm64-evb3020.img  rockdev/rootfs.img
```

4. 执行脚本打包系统

```
./build.sh updateimg
```

最终版在rockdev目录下生成update.img。

十、更多

其他编译命令，可通过执行 ./build.sh -h 查看帮助。

```
$ ./build.sh -h
Usage: build.sh [OPTIONS]
Available options:
BoardConfig*.mk    -switch to specified board config
uboot              -build uboot
spl                -build spl
kernel             -build kernel
modules            -build kernel modules
rootfs             -build default rootfs, currently build buildroot as default
buildroot          -build buildroot rootfs
ramboot            -build ramboot image
multi-npu_boot     -build boot image for multi-npu board
yocto              -build yocto rootfs
debian             -build debian9 stretch rootfs
distro             -build debian10 buster rootfs
pcba               -build pcba
recovery           -build recovery
all                -build uboot, kernel, rootfs, recovery image
cleanall           -clean uboot, kernel, rootfs, recovery
firmware           -pack all the image we need to boot up system
updateimg          -pack update image
otapackage         -pack ab update otapackage image
save               -save images, patches, commands used to debug
allsave            -build all & firmware & updateimg & save
```