软件开发指南
ECK32-T527 核心板

成都亿佰特电子科技有限公司
Chengdu Ebyte Electronic Technology Co.,Ltd.

目录

免责申明和版权公告

本文中的信息，如有变更，恕不另行通知。 文档“按现状”提供，不负任何担保责任，包括对适销性、适用于特定用途或非侵权性的任何担保，和任何提案、规格或样品在他处提到的任何担保。本文档不负任何责任，包括使用本文档内信息产生的侵犯任何专利权行为的责任。本文档在此未以禁止反 言或其他方式授予任何知识产权使用许可，不管是明示许可还是暗示许可。

文中所得测试数据均为亿佰特实验室测试所得，实际结果可能略有差异。

文中提到的所有商标名称、商标和注册商标均属其各自所有者的财产，特此声明。

最终解释权归成都亿佰特电子科技有限公司所有。

注 意 ：

由于产品版本升级或其他原因，本手册内容有可能变更。亿佰特电子科技有限公司保留在没有任何通知或者提示的情况下对本手册的内容进行修改的权利。本手册仅作为使用指导，成都亿佰特电子科技有限公司尽全力在本手册中提供准确的信息，但是成都亿佰特电子科技有限公司并不确保手册内容完全没有错误，本手册中的所有陈述、信息和建议也不构成任何明示或暗示的担保。

1. 概述

本文主要介绍基于亿佰特核心板定制一个完整的嵌入式 Linux 系统的完整流程，其中包括开发环境的准备，代码的获取，以及如何进行 Bootloader, Kernel 的移植，定制适合自身应用需求的 Rootfs 等。我们首先介绍如何基于我们提供的 SDK 源代码构建适用于 ECK32-T527 核心板的系统镜像，如何将构建好的镜像烧录到开发板。针对那些基于 ECK32-T527 核心板进行项目开发的用户，我们重点介绍了将这一套系统移植到用户的硬件平台上的方法和一些要点，并通过一些实际的移植案例和基于 Buildroot 的 Rootfs 定制案例，使用户能够迅速定制适合自己硬件的系统镜像。

请注意本文档并不包含 buildroot 项目以及 Linux 系统相关基础知识的介绍，适合有一定开发经验的嵌入式 Linux 系统开发人员和嵌入式 Linux BSP 开发人员。

2. 开发环境准备

本章主要介绍基于 ECK32-T527 核心板使用开发前需要的一些软硬件环境，包括必要的硬件配置，开发、烧录所需环境介绍。

通过阅读本章节，您将了解相关硬件工具，软件开发调试工具的安装和使用。并能快速地搭建相关开发环境，为后面的开发和调试做准备。

主机：i7-12700 + win11 22H2

虚拟机：Debian bullseye

2.1. 硬件准备

2.1.1. 调试串口

调试串口：波特率 115200，数据位 8，停止位 1，无奇偶校验，无硬件流控，实际串口请自行配置，配置方式详见后文。

2.1.2. 启动方式

对于使用本司 ECK32-T527 核心板用户只需要记住的是，一旦插入可启动的 SD 卡，系统将从 SD 卡启动，SD 卡无法启动，则从板上存储设备启动。

2.2.虚拟机软件部署

本章节将介绍如何搭建 T527 的开发环境。通过阅读本章节，您将了解相关开发调试工具的安装和使用。并能快速地搭建相关开发环境，为后面的开发和调试做准备。

2.2.1. 虚拟机配置

当前较为常见的虚拟机软件有 VirtualBox，VMware 等，不同平台的配置方式有所区别，用户使用习惯亦有所区别，故请用户自行根据使用的虚拟机软件配置虚拟机使用桥接的方式联网，并打开与 windows 的目录共享功能。

2.2.2. 配置编译环境

由于 T527 的 SDK 中会使用 buildroot 202205 版本，故推荐用户使用 Ubuntu 2204 64bit 发行版，也可以尝试使用更高版本发行版，但依然请确保版本与 buildroot 版本兼容。

安装软件：

```
sudo apt install -y gcc g++ flex bison xz-utils libncurses-dev device-tree-compiler
sudo apt install -y rsync fakeroot libtool texinfo automake git wget cpio busybox
sudo apt install -y libelf-dev bc libssl-dev lzop libncursesw5-dev bear unzip make
sudo dpkg --add-architecture i386
sudo apt update
sudo apt install -y libc6:i386 libgcc1:i386 libstdc++6:i386 zlib1g:i386
```

创建工作目录：

```
mkdir ~/work
```

2.2.3. 开启虚拟机的 TFTP 服务

后面调试阶段可能会使用 TFTP 从网路获取文件，因此要先在虚拟机中安装并开启 TFTP 服务，使用如下命令安装 TFTP 服务：

```
sudo apt install tftpd-hpa -y
```

默认情况下，将分享/srv/tftp 目录下的文件，且只能下载，不能上传，如果想个性化配置，请参考官方文档：https://help.ubuntu.com/community/TFTP，本文档使用路径为/srv/tftp。

添加用户到 tftp 组并修改共享目录权限：

```
sudo usermod -a -G tftp $USER
sudo chmod 775 /srv/tftp/
sudo chgrp tftp /srv/tftp/
```

如果没有执行上述操作，可能出现权限问题，非 root 用户权限下无法写入文件到共享目录。

2.2.4. 开启虚拟机的 NFS 服务

后面调试阶段往往需要用 NFS 共享文件，因此要先在虚拟机中安装并开启 NFS 服务，使用如下命令安装 NFS 服务：

```
sudo apt install nfs-kernel-server -y
```

等待安装完成，安装完成以后设置需要导出的目录，此处直接使用 tftp 默认目录，用户可自行修改，也可使用顺手的文本编辑器修改/ext/exports 文件，配置方法参考 man 手册：

```
echo "/srv/tftp *(rw,sync,no_subtree_check,no_root_squash,fsid=root)" | sudo tee -a /etc/exports
man exports
```

2.2.5. 开启虚拟机的 SSH 服务

开启虚拟机的 SSH 服务以后我们就可以在 Windwos 下使用 ssh 客户端软件登录到虚拟机，比如使用 MobaXterm、VSCode，虚拟机下使用如下命令开启 SSH 服务：

```
sudo apt install openssh-server -y
```

上述命令安装 ssh 服务，ssh 的配置文件为/etc/ssh/sshd_config，使用默认配置即可，更多配置参考 sshd_config 相关的 man 手册。

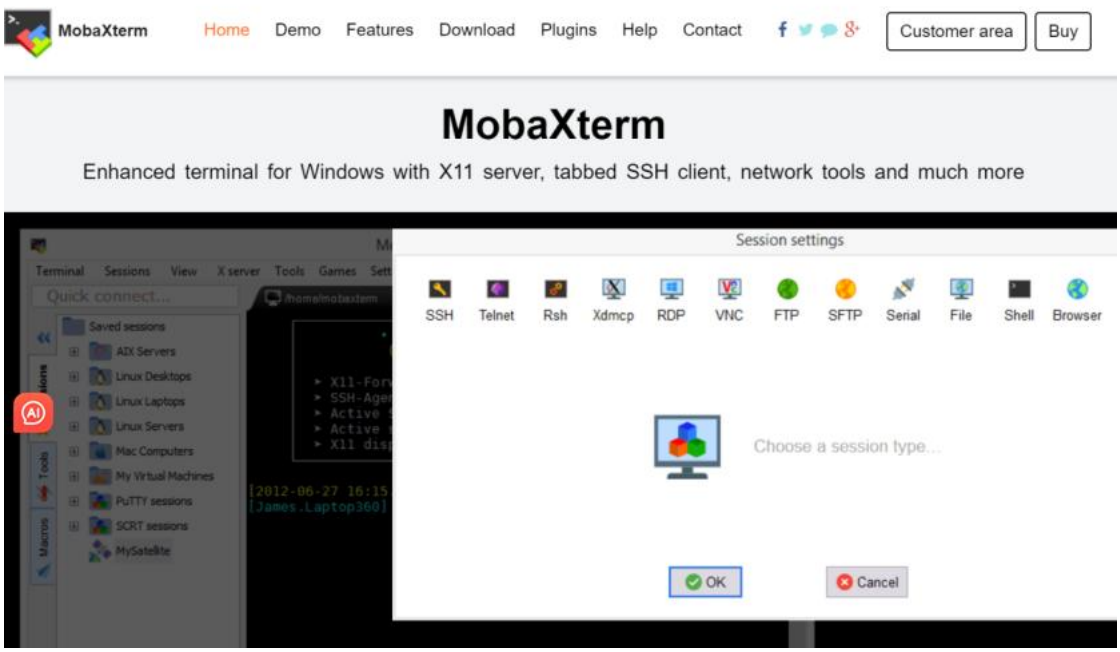
2.3.主机软件安装

本章节将介绍如何搭建 T527 的烧录调试环境。通过阅读本章节，您将了解相关烧写调试工具的安装和使用。

2.3.1. MobaXterm

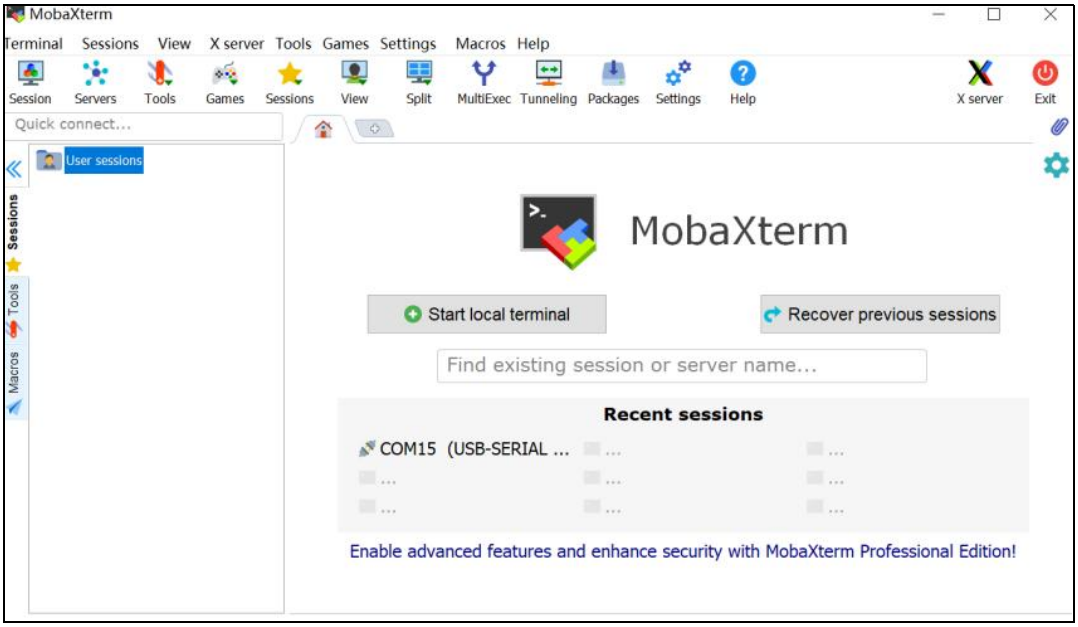
2.3.1.1. 软件下载安装

MobaXterm 是一款终端软件，功能强大而且免费，推荐使用此软件作为终端调试软件，MobaXterm 软件在其官网下载即可，地址为 https://mobaxterm.mobatek.net/，如图所示：

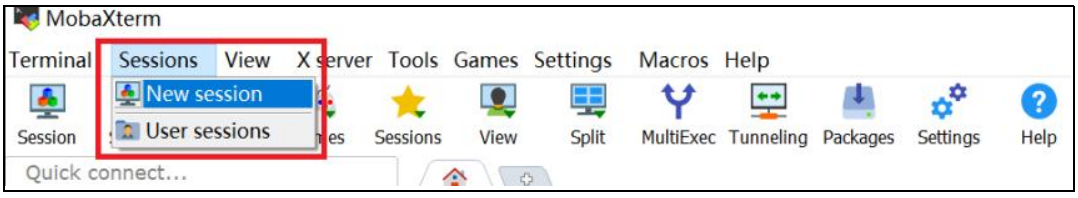


2.3.1.2. 软件使用

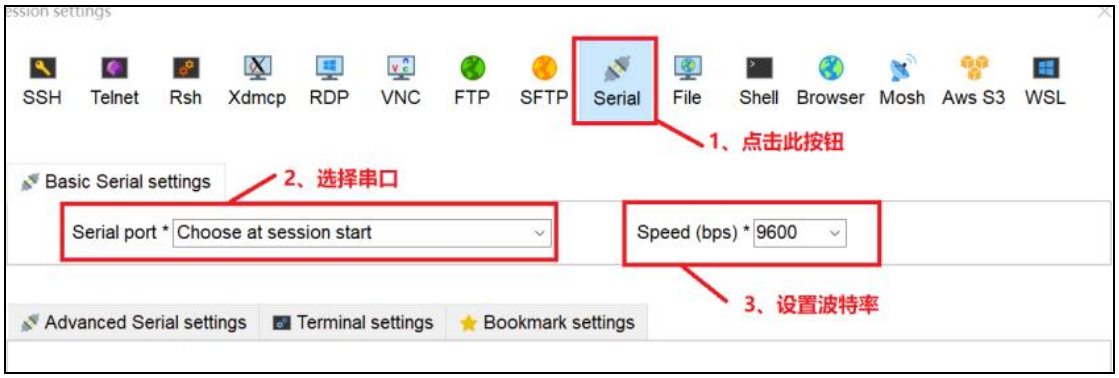
软件界面如图所示：



点击菜单栏中的“Sessions->New session”按钮，打开新建会话窗口，如图所示



建立 Serial 连接，也就是串口连接，因为我们使用 MobaXterm 的主要目的就是作为串口终端使用。点击图中的“Serial”按钮，打开串口设置界面，如图所示：



打开串口设置窗口以后先选择要设置的串口号，因此要先用串口线将开发板连接到电脑上，由于 T527 默认使用的串口波特率为 115200，需要设置波特率为 115200。

2.3.2. PhoenixSuit

该工具为全志官方提供的从 USB 烧录系统的工具，该软件已存放在目录：“04_Tools/PhoenixSuit”。具体使用方式详见后文《官方工具烧录》一节。

2.3.3. PhoenixCard

这是全志官方提供的用于制作系统卡的工具，可以选择烧录制作量产卡与启动卡，该软件已存放在目录：“04_Tools/PhoenixCard”。具体使用方式详见后文《制作 SD 卡启动卡》一节。

2.4.数据传输

在前面章节我们已经配置好了虚拟机部分各种网络服务，现在只需要了解如何在客户端与开发板的使用即可。

2.4.1. TFTP 客户端

一般情况下，开发板只需要使用下载功能即可，此时可以简单地使用 curl 工具下载，使用之前需要先在 buildroot 中选配 curl 功能，然后使用 curl 下载文件(仅限 Linux 版 curl，windows 可选择使用 filezilla)：

```
curl -o /path/dist_file tftp://HOST/src_file
```

1. /path/dist_file：要下载到哪里，输出文件名字的名字与路径。

2. HOST：虚拟机的 ip 地址，需要注意这个地址是否可以访问。

3. src_file：文件目录以及名字，需要注意的是不需要添加配置中的"TFTP_DIRECTORY"部分，比如文件存放在`/srv/tftp/aaa/bbb/ccc`，src_file 值为"aaa/bbb/ccc"。

2.4.2. NFS 客户端

一般情况下，开发板只需要简单地使用 mount 命令即可挂载 NFS 文件系统，但是需要

安装基本的支持，然后挂载 NFS：

```
mount -t nfs HOST:/path /mnt_point
```

1. HOST：虚拟机的 ip 地址，需要注意这个地址是否可以访问。
2. path：想要挂载的 NFS 服务器中的路径，比如/srv/tftp/aaa/bbb。
3. mnt_point：想要把 NFS 文件系统挂载到哪里，比如/mnt。

2.4.3. SSH

一般来说我们也可以使用 SSH 自带的 SFTP 与 SCP 功能进行文件下载，但直接使用命令上传下载的情况不多，更多使用第三方工具进行，如使用 MobaXterm 登录 ssh 后的 SFTP 功能，或者 VSCode 使用 remote-SSH 插件实现开发过程中的下载功能，使用方式都是鼠标放在需要下载的文件上右键，然后选择下载。

2.4.4. U 盘/SD 卡

对于 SD 卡，需要使用读卡器，实际用法和 U 盘一致。

2.4.4.1. 从虚拟机拷贝数据

用户需要根据自身使用的虚拟机软件先将 U 盘/SD 卡传递到虚拟机当中，一般来说会自动挂载，在文件管理器中可以直接访问，拷贝数据完毕后在文件管理器中卸载设备，然后从虚拟机中移除设备即可。

如果没有自动挂载，可以使用下方的两种指令手动挂载：

1. 方式一

```
udisksctl mount --block-device /dev/sdXY
```

2. 方式二

```
sudo mount /dev/sdXY /mnt_point
```

这种情况需要对 U 盘/SD 卡分区有一点了解，才可以指定使用 sdXY 中的 Y 值，X 值一般是'b'，但是也需要自行通过 lsblk 的返回信息判断情况。

使用方式一挂载，无需 root 权限即可访问，也不用指定路径，但是需要系统中有安装了`udisks2`这个程序包。

手动卸载对应如下命令：

1. 方式一

```
udisksctl unmount --block-device /dev/sdXY
udisksctl power-off --block-device /dev/sdXY
```

2. 方式二

```
sudo umount /mnt_point
```

2.4.4.2. 拷贝数据到开发板

对于开发板，一般会进行精简，不一定有 udisks2 包，也不一定有一个 GUI 界面，有界面也不一定支持自动挂载，但是往往用户都有 root 权限，所以一般从命令行执行 mount 操作。

1. 挂载

```
mount /dev/sdXY /mnt_point
```

2. 拷贝

```
cp /mnt_point/file /out_path/out_file
cp -ra /mnt_point/dir /out_path/out_dir
```

3. 取消挂载

```
umount /mnt_point
```

3. SDK 编译构建

本章节将描述从 SDK 的获取到编译生成一个可用镜像的完整流程。

3.1. SDK 获取

T527 的开发 SDK 已经随附在下载文件中，请用户自行解压至开发目录中。

由于完整的 SDK 过大，为了减少 SDK 大小，本司提供的 SDK 通过 repo 进行压缩，提取后在当前目录下生成隐藏目录`.repo`。

解压需要使用 repo 工具，下载 repo 工具：

```
sudo apt install -u curl
mkdir -p ~/.local/bin
curl https://storage.googleapis.com/git-repo-downloads/repo -o ~/.local/bin/repo
chmod +x ~/.local/bin/repo
```

配置 repo 工具更新源使用镜像(国内可选)：

```
echo "export REPO_URL='https://mirrors.tuna.tsinghua.edu.cn/git/git-repo'" >> ~/.bashrc
```

使能相关环境变量：

```
source ~/.profile
```

进入到项目目录以后执行如下命令从本地同步代码仓库：

```
repo sync -l
```

查看目录情况如下：

```
$ ~/work ls -A
.repo
$ ~/work repo sync -l
Checking out files: 100% (14419/14419), done.andy-2.0/toolsChecking out files:  63% (9170/14419)
Checking out files: 100% (11369/11369), done.ildChecking out files:   7% (829/11369)
Checking out files: 100% (20710/20710), done.ldroot-201902Checking out files:   6% (1368/20710)
Checking out files: 100% (2964/2964), done./device/config/commonChecking out files:  11% (350/2964)
Checking out files: 100% (72149/72149), done.evice/config/rootfs_tarChecking out files:   0% (208/72149)
Checking out files: 100% (364/364), done.uct/tina/tina-ng/buildChecking out files:  10% (38/364)
Checking out files: 100% (2781/2781), done.t/tina/tina-ng/dlChecking out files:   9% (268/2781)
Checking out files: 100% (4271/4271), done.t/tina/tina-ng/package/allwinner/wifimanagerChecking out files:  29% (1255/4271)
Checking out files: 100% (955/955), done.uct/tina/tina-ng/package/thirdparty/flutterChecking out files:  47% (453/955)
Checking out files: 100% (1498/1498), done.t/linux/external/libcedarseChecking out files:  79% (1194/1498)
Checking out files: 100% (274/274), done.a/cedarxChecking out files:  86% (238/274)
Checking out files: 100% (13/13), done.oduct/linux/external/healthdChecking out files:  53% (7/13)
Checking out files: 100% (5/5), done.product/linux/external/littlevgl-8Checking out files:  80% (4/5)
Checking out files: 100% (2934/2934), done.t/linux/external/qtChecking out files:  12% (373/2934)
Checking out files: 100% (7/7), done.product/tina/tina-ng/prebuilt/hostbuiltChecking out files:  42% (3/7)
Checking out files: 100% (15834/15834), done.tina/tina-ng/prebuilt/kernel-built/armChecking out files:   1% (217/15834)
Checking out files: 100% (1771/1771), done.e/freertosChecking out files:  28% (512/1771)
Checking out files: 100% (8244/8244), done.ct/tina/tina-rtChecking out files:   6% (496/8244)
Checking out files: 100% (860/860), done.duct/tina/rtos-toolsChecking out files:  97% (837/860)
Checking out projects: 100% (108/108), done.
repo sync has finished successfully.
$ ~/work
```

3.1.1. 可能遇到的问题

由于 repo 工具是一个 python 脚本，用户使用的 python 版本与 SDK 中的 repo 依赖版本可能不兼容，对于默认 python 是 python2 的用户，或需要配置默认 python 为 python3，若设置后依然报错，请用户到`SDK/.repo/repo`目录中执行`git pull`拉取更新的 repo 版本。

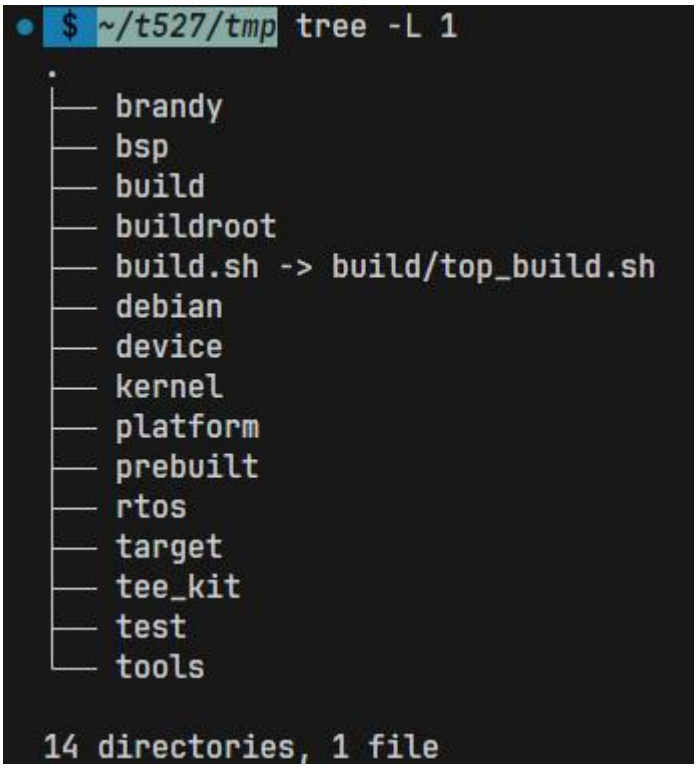
对于部分未使用 debian/ubuntu 的用户，上面`source ~/.profile`不一定会把`~/.local/bin/`目录加入到环境变量中，请按照如下方式添加：

```
echo 'export PATH="${PATH}:${HOME}/.local/bin"' >> ~/.bashrc

source ~/.bashrc
```

3.2. SDK 目录

解压完毕后，整个 SDK 目录情况如下图：



我们需要关注下面这些目录：

顶级目录	子目录	作用
brandy	brandy-2.0/spl-pub/	SPL 源码，一般不修改，全志官方提供的也大部分是二进制文件
	brandy-2.0/u-boot-2018/	U-boot 源码
build		编译脚本目录，一般不修改
buildroot	buildroot-202205	文件系统源码
	package/auto/sdk_demo/	示例程序
debian	compressed_files/	Debian 示例镜像
device	config/chips/	SDK 支持的芯片配置目录，包括多款开发板的配置方案
	product	默认没有，在执行 SDK 配置后指向目标芯片，此处为 T527
kernel	linux-5.15	内核源码
bsp	configs/linux-5.15/	存放 soc 相关 base 设备树
	drivers	存放全志相关模块驱动
platform		平台扩展相关源码
prebuilt		内核、buildroot 等相关编译工具
rtos		rtos 源码，主要与 RISC-V 核心相关
tools	tools_win	一些 windows 下的工具

3.3. SDK 配置

由于 SDK 支持多款芯片，也支持多种方案，所以在正式开始编译之前需要用户选择符合自身开发板的配置，SDK 中也提供了配置方式：

```
./build.sh config
```

执行完毕后需要同意全志的一些许可认证：

```
$ ~/work ./build.sh config

Before using this files, please make sure that you note the following important information.
*********************************************************************

Copyright (c) 2019-2022 Allwinner Technology Co., Ltd. ALL rights reserved.

Allwinner is a trademark of Allwinner Technology Co.,Ltd., registered in
the the People's Republic of China and other countries.
All Allwinner Technology Co.,Ltd. trademarks are used with permission.

DISCLAIMER
THIRD PARTY LICENCES MAY BE REQUIRED TO IMPLEMENT THE SOLUTION/PRODUCT.
IF YOU NEED TO INTEGRATE THIRD PARTY'S TECHNOLOGY (SONY, DTS, DOLBY, AVS OR MPEGLA, ETC.)
IN ALLWINNERS'SDK OR PRODUCTS, YOU SHALL BE SOLELY RESPONSIBLE TO OBTAIN
ALL APPROPRIATELY REQUIRED THIRD PARTY LICENCES.
ALLWINNER SHALL HAVE NO WARRANTY, INDEMNITY OR OTHER OBLIGATIONS WITH RESPECT TO MATTERS
COVERED UNDER ANY REQUIRED THIRD PARTY LICENSE.
YOU ARE SOLELY RESPONSIBLE FOR YOUR USAGE OF THIRD PARTY'S TECHNOLOGY.


THIS SOFTWARE IS PROVIDED BY ALLWINNER"AS IS" AND TO THE MAXIMUM EXTENT
PERMITTED BY LAW, ALLWINNER EXPRESSLY DISCLAIMS ALL WARRANTIES OF ANY KIND,
WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING WITHOUT LIMITATION REGARDING
THE TITLE, NON-INFRINGEMENT, ACCURACY, CONDITION, COMPLETENESS, PERFORMANCE
OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
IN NO EVENT SHALL ALLWINNER BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS, OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.

*********************************************************************
You can read /home/me/work/build/disclaimer/Allwinnertech_Disclaimer(Cn_En)_20181122.md for detailed information.

You read time left 8 seconds....
I have already read, understood and accepted the above terms? [Y/N]
```

在此处敲击`Y`并回车确认，进入后续选择：

```
You select Yes, Build continue....
========ACTION List: mk_config ;========
options :
All available platform:
   0. android
   1. linux
Choice [android]: 1
All available linux_dev:
   0. bsp
   1. dragonboard
   2. buildroot
   3. debian
Choice [bsp]: 2
All available ic:
   0. a523
   1. a527
   2. t527
Choice [a523]: 2
All available board:
   0. demo_linux_aiot
   1. demo_linux_aiot_nand
   2. demo_linux_aiot_spinand
   3. ebyte_linux_t527
Choice [demo_linux_aiot]: 3
All available flash:
   0. default
   1. nor
Choice [default]:
```

此处使用的编译配置为`device/config/chips/t527/configs/ebyte_linux_t527/buildroot/BoardConfig.mk`。配置完毕后，最终的编译规则配置可以在 SDK 目录下的`.buildconfig`文件查看，下面列举其中比较重要的部分：

```
LICHEE_PLATFORM //编译平台
LICHEE_LINUX_DEV //编译的方案
LICHEE_IC // 编译的 IC
LICHEE_BOARD //编译的板级配置
LICHEE_FLASH //编译的 flash 配置
LICHEE_CHIP //编译的芯片代号名称
LICHEE_KERN_VER //编译的内核版本
LICHEE_KERN_DEFCONF //编译内核的默认配置
```

```
LICHEE_BUILDING_SYSTEM  //编译的构造系统
LICHEE_BR_VER  //buildroot 的版本
LICHEE_BR_DEFCONF  //buildroot 的默认配置
LICHEE_BR_RAMFS_CONF  //buildroot 的 ramfs 默认配置
LICHEE_BRANDY_VER  //uboot 的版本
LICHEE_BRANDY_DEFCONF  //brandy 的默认配置
LICHEE_CROSS_COMPILER  //编译使用的交叉编译链
LICHEE_CHIP_CONFIG_DIR  //编译系统的 IC 配置目录
LICHEE_OUT_DIR  //编译的输出目录
```

3.4. 编译 SDK

等待 SDK 结束配置，然后开始执行编译：

```
./build.sh
```

编译完成后，生成 SPL、U-Boot、Linux 内核和 Buildroot 文件系统镜像文件。编译结束后会有如下输出：

```
Exportable Squashfs 4.0 filesystem, xz compressed, data block size 131072
        compressed data, compressed metadata, compressed fragments, no xattrs
        duplicates are removed
Filesystem size 222354.04 Kbytes (217.14 Mbytes)
        39.08% of uncompressed filesystem size (569042.73 Kbytes)
Inode table size 76414 bytes (74.62 Kbytes)
        25.35% of uncompressed inode table size (301393 bytes)
Directory table size 83916 bytes (81.95 Kbytes)
        44.15% of uncompressed directory table size (190078 bytes)
Number of duplicate files found 83
Number of inodes 8121
Number of files 6327
Number of fragments 726
Number of symbolic links  1249
Number of device nodes 0
Number of fifo nodes 0
Number of socket nodes 0
Number of directories 545
Number of ids (unique uids + gids) 1
Number of uids 1
        root (0)
Number of gids 1
        root (0)
INFO: pack rootfs ok ...
INFO: ----------------------------------------
INFO: build OK.
INFO: ----------------------------------------
```

3.5. 单独编译

3.5.1. 单独编译 u-boot

通过如下命令可以单独编译 u-boot，对于只需要修改 u-boot 阶段的时候可以节省编译时间：

```
./build.sh  bootloader
```

```
$ ~/t527/tmp ./build.sh bootloader
========ACTION List: build_bootloader ;========
options :
find: '/home/me/t527/tmp/brandy/brandy-2.0/spl': No such file or directory
find: '/home/me/t527/tmp/brandy/brandy-2.0/u-boot-bsp': No such file or directory
find: '/home/me/t527/tmp/brandy/brandy-2.0/u-boot-efex': No such file or directory
INFO: build_bootloader: brandy_path=/home/me/t527/tmp/brandy/brandy-2.0
INFO: skip build brandy.
```

3.5.2. 单独编译 kernel

通过如下命令可以单独编译内核，对于只需要修改内核阶段的时候可以节省编译时间，但是需要注意，如果修改后添加了驱动模块，则还需要同时编译 rootfs，否则找不到对应驱动模块：

```
./build.sh  kernel
```

```
$ ~/t527/tmp ./build.sh kernel
========ACTION List: build_kernel ;========
options :
INFO: build kernel ...
INFO: prepare_buildserver
INFO: Prepare toolchain ...
Setup BSP files
bsp/modules/gpu/Makefile:64: neither CONFIG_ARM nor CONFIG_ARM64 is found in .config, unsupport
2 ,/home/me/t527/tmp/kernel/linux-5.15, /home/me/t527/tmp/kernel/linux-5.15
Building kernel
15985 blocks
make[1]: Entering directory '/home/me/t527/tmp/out/t527/kernel/build'
  GEN     Makefile
  CALL    /home/me/t527/tmp/kernel/linux-5.15/scripts/atomic/check-atomics.sh
  CALL    /home/me/t527/tmp/kernel/linux-5.15/scripts/checksyscalls.sh
  CHK     include/generated/compile.h
  CC      bsp/drivers/clk/sunxi-ng/ccu_common.o
2 sun55iw3p1,/home/me/t527/tmp/kernel/linux-5.15, /home/me/t527/tmp/out/t527/kernel/build
  INSTALL /home/me/t527/tmp/out/t527/kernel/build/user_headers/include
  AR      bsp/drivers/clk/sunxi-ng/built-in.a
  AR      bsp/drivers/clk/built-in.a
  AR      bsp/drivers/built-in.a
  AR      bsp/built-in.a
  GEN     .version
  CHK     include/generated/compile.h
  UPD     include/generated/compile.h
  CC      init/version.o
  AR      init/built-in.a
  LD      vmlinux.o
```

编译完毕输出如下 log：

```
Warning: init ramdisk file '/home/me/t527/tmp/out/t527/ebyte_linux_t527/buildroot/ramfs/images/rootfs.cpio.gz' not exist on buildroot platform!
Use init ramdisk file: '/home/me/t527/tmp/kernel/linux-5.15/bsp/ramfs/ramfs_aarch64.cpio.gz'.
Copy modules to target ...
15985 blocks
85852 blocks
bootimg_build
Copy boot.img to output directory ...

sun55iw3p1 compile all(Kernel+modules+boot.img) successful


INFO: build dts ...
INFO: Prepare toolchain ...
Setup BSP files
'/home/me/t527/tmp/out/t527/kernel/staging/sunxi.dtb' -> '/home/me/t527/tmp/out/t527/ebyte_linux_t527/buildroot/sunxi.dtb'
```

3.5.3. 单独编译 buildroot

通过如下命令可以单独编译 buildroot，对于只需要修改 buildroot 阶段的时候可以节省编译时间：

```
./build.sh  buildroot_rootfs
```

```
$ ~/t527/tmp ./build.sh buildroot_rootfs
========ACTION List: build_buildroot_rootfs ;========
options :
INFO: build buildroot ...
make: Entering directory '/home/me/t527/tmp/buildroot/buildroot-202205'
```

编译完毕输出如下 log：

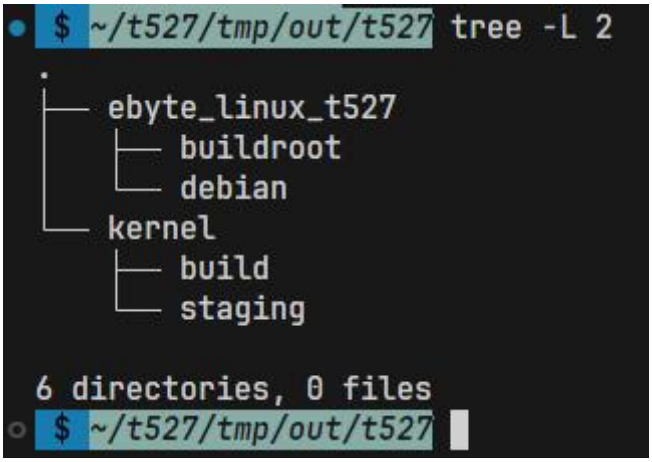
```
make[1]: Leaving directory '/home/me/t527/tmp/buildroot/package/auto/sdk_demo/mem_test'
make: Leaving directory '/home/me/t527/tmp/buildroot/package/auto/sdk_demo'
build auto finish
INFO: copy the config files form device ...
make: Entering directory '/home/me/t527/tmp/platform'
Makefile:35: "----------1--------"
Makefile:36: /home/me/t527/tmp/platform
make: Nothing to be done for 'INSTALL_FILES'.
make: Leaving directory '/home/me/t527/tmp/platform'
INFO: build buildroot OK.
```

3.6. 编译输出

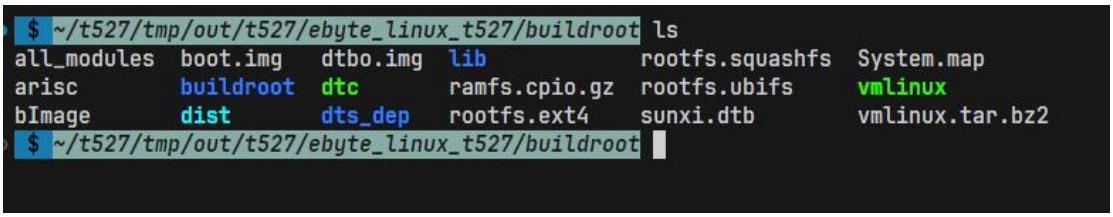
在编译完毕以后会发现在 SDK 目录中出现了一个 out 目录，这里面存放了 SDK 的全部编译输出。

```
$ ~/t527/tmp/out ls
kernel  serversocket  t527  toolchain
$ ~/t527/tmp/out
```

其中 kernel 目录为 t527/kernel 目录的软连接，是内核的完整编译输出，toolchain 目录为编译过程中使用到交叉编译工具链，我们主要来看 t527 目录：



进入到 out/t527/ebyte_linux_t527/buildroot 目录：



我们主要需要关注其中下面这几个文件：

- boot.img：内核镜像，包含 kernel，设备树，安卓镜像格式
- sunxi.dtb：编译出来的内核设备树文件，最终会打包到 boot.img 中。
- vmlinux：编译出来的内核文件，最终会打包到 boot.img 中。
- rootfs.*：buildroot 生成的根文件系统。通常后缀为 ext4。

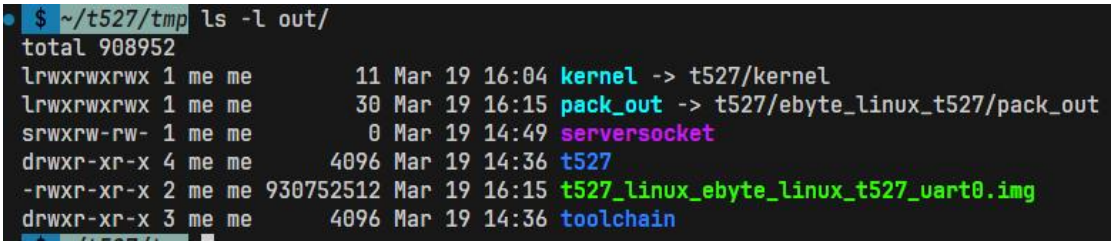
3.7. 镜像打包

经过之前的编译后，输出了各种编译文件，但是大多无法直接使用，需要打包成最终可用的镜像：

```
./build.sh pack
```

等待打包结束后查看 out 目录，会发现多出来了一个 pack_out 目录与一个 img 文件，该 img 文件就是最终需要烧录到开发板上的镜像文件，而 pack_out 目录则是打包该镜像的中间文件，其中部分文件也是最终可以用到的。

再来看 pack_out 目录：



可以发现他是一个从 t527/<board>/pack_out 过来的软连接：

```
$ ~/t527/tmp/out/t527/ebyte_linux_t527/pack_out ls
amp_dsp0.bin                boot0_mmc_car_fastboot_sun55iw3p1.bin  default.awlic              optee.fex                  sys_partition_private.fex
amp_dsp0.fex                boot0_mmcfastboot_sun55iw3p1.bin       diskfs.fex                 optee_sun55iw3p1.bin       sysrecovery.fex
amp_rv0.bin                 boot0_nand.fex                         dlinfo.fex                 parameter.fex              temp_ubootnodtb.bin
amp_rv0.fex                 boot0_nand_sun55iw3p1.bin              dragon_toc_android.cfg     rootfs.fex                 toc0.fex
arisc.fex                   boot0_sdcard.fex                       dragon_toc.cfg             rootfs_nor.fex             toc0_ft.fex
aultls32.fex                boot0_sdcard_sun55iw3p1.bin            dragon_toc.fex             rootfs-ubifs.fex           toc0_nand.fex
aultools.fex                boot0_spinor.fex                       dtboimg.cfg                sboot.bin                  toc0_sdcard.fex
auto_board.dtbo             boot0_spinor_sun55iw3p1.bin            env-5.15.cfg               sboot_nand.bin             toc0_ufs.fex
auto_board.dts              boot.fex                               env_burn.cfg               sboot_nand_sun55iw3p1.bin  toc1.fex
battery_charge.bmp          bootlogo.bmp                           env.cfg                    sboot_sdcard.bin           u-boot-crash.fex
battery_charge.bmp.lzma     bootlogo.bmp.lzma                      env_dragon.cfg             sboot_sdcard_sun55iw3p1.bin u-boot.fex
bempty.bmp                  bootlogo.bmp.lzma.head                 env.fex                    sboot_sun55iw3p1.bin       u-boot-sun55iw3p1.bin
bempty.bmp.lzma             bootlogo.fex                           esm.fex                    scp.bin                    usbtool_crash.fex
bl31.bin                    boot_package.cfg                       fes1.fex                   scp.fex                    usbtool.fex
board1.dtbo                 boot_package.fex                       fes1_sun55iw3p1.bin        split_xxxx.fex             usbtool_test.fex
board1.dts                  boot_package_nor.cfg                   fit-image.its              sunxi.fex                  Vboot.fex
board2.dtbo                 boot-resource                          image_android.cfg          sunxi_gpt.fex              Vboot-resource.fex
board2.dts                  boot-resource.fex                      image.cfg                  sunxi_mbr.fex              Venv.fex
board3.dtbo                 boot-resource.ini                      image_crashdump.cfg        sunxi_version.fex          verity_block.fex
board3.dts                  boottone.fex                           image_dragonboard.cfg      sys_config.bin             version_base.mk
board_config.bin            cardscript.fex                         image_linux.cfg            sys_config_ddr_2GB.fex     vmlinux.fex
BoardConfig-conditions.mk   cardscript_secure.fex                  image_nor.cfg              sys_config.fex             Vrootfs.fex
board_config.fex            cardtool.fex                           kernel.fex                 sys_partition.bin
BoardConfig.mk              cnf_base.cnf                           monitor.fex                sys_partition_dump.fex
board.fex                   config.fex                             new_fdt.dtb                sys_partition.fex
$ ~/t527/tmp/out/t527/ebyte_linux_t527/pack_out
```

其中内容比较多，我们主要关注下面这几个文件：

- boot_package.fex：打包后的 u-boot 镜像，包含内容可以查看 boot_package.cfg
- boot-resource.fex：启动过程中用到的资源，如 logo 图片。
- env.fex：u-boot 的环境变量配置，SDK/device/product/configs/<board>/[buildroot]/env.cfg 的复制。
- boot.fex：上文编译出来的内核镜像 boot.img 的软连接。
- rootfs*.fex：上文编译出来的 buildroot 根文件系统的软连接。

3.7.1. 可能遇到的问题

在打包过程中可能会遇到以下情况：

3.7.1.1. dl file XXXX size too large

```
mbr size = 16384
mbr magic softw411
disk name=boot-resource
disk name=env
disk name=env-redund
disk name=boot
disk name=rootfs
ERROR: dl file rootfs.fex size too large
ERROR: filename = rootfs.fex
ERROR: dl_file_size = 2097152 sector
ERROR: part_size = 97152 sector
update_for_part_info -1
ERROR: update mbr file fail
ERROR: update_mbr failed
```

若出现此种情况，表示用户配置的分区大小不适配最终输出文件，可以在`SDK/device/

product/configs/<board>/[buildroot]/sys_partition.fex 中查找图中 filename 对应文件对应分区，然后修改该分区大小以适配相应文件，分区大小最小值如上图中 dl_file_size 描述：

```
[partition]
    name         = rootfs
    size         = 16777216
    downloadfile = "rootfs.fex"
    user_type    = 0x8000
```

4. 开发板适配

为了适配用户新的硬件平台，用户需要对 CPU 芯片手册，CPU 芯片用户指南有一定了解，同时需要对 SDK 修改内容有一定了解，从上面 SDK 个目录的描述中可以知道，在 SDK/device/product/configs/中存放的是开发板/方案配置：

```
$ ~/t527/ebyte/device/product/configs tree -L 1
.
├── default
├── demo_linux_aiot
├── demo_linux_aiot_nand
├── demo_linux_aiot_spinand
└── ebyte_linux_t527
```

这一级目录中，除了 default 目录以外，可以认为都是不同的开发板/方案配置，而 default 目录中的内容是作为对应板级配置中找不到需要项的时候的默认配置项，一般不用修改，也不应该修改。

依然以 ebyte_linux_t527 为例：

```
$ ~/t527/ebyte/device/product/configs/ebyte_linux_t527 tree
.
├── arisc.config
├── bin
│   ├── amp_dsp0.bin
│   └── amp_rv0.bin
├── BoardConfig.mk
├── board.dts -> linux-5.15/board.dts
├── boottone.fex
├── buildroot
│   ├── BoardConfig.mk
│   ├── env.cfg
│   └── sys_partition.fex
├── debian
│   ├── env.cfg
│   └── sys_partition.fex
├── dtbo
│   ├── auto_board.dtbo
│   ├── auto_board.dts
│   ├── board1.dtbo
│   ├── board1.dts
│   ├── board2.dtbo
│   ├── board2.dts
│   ├── board3.dtbo
│   ├── board3.dts
│   └── dtboimg.cfg
├── linux-5.15
│   ├── board.dts
│   ├── board_standard.dts
│   ├── bsp_defconfig
│   ├── env-5.15.cfg
│   └── sys_partition.fex
├── sys_config_ddr_2GB.fex
├── sys_config.fex
└── uboot-board.dts
```

在这里面我们主要需要注意的文件有下面几个：

- BoardConfig.mk：SDK 需要使用的编译选项配置。
- board.dts：linux-5.15/board.dts 的软连接，用于内核的设备树文件。
- buildroot：buildroot 构建根文件系统使用的详细配置。
- bsp_defconfig：内核相关的配置。
- sys_config.fex：系统参数配置，旧版本内核起到设备树的作用，但是在 Linux5.4 以后版本，仅保留少部分模块 (如 ddr、串口 debug 等级) 配置功能，且多用于 SPL 和 u-boot 阶段。

- uboot-board.dts：u-boot 阶段的设备树文件。
- sys_partition.fex：镜像分区配置。
- env.cfg：u-boot 阶段的环境变量。

我们可以发现，在 buildroot 目录下也存在部分上一级目录中同名的文件，这是一种覆盖关系，其覆盖关系如下：

1. bsp、buildroot、debian 等系统方案，属于并行目录，不会相互覆盖。

2. bsp、buildroot 等系统方案的 BoardConfig.mk、env.cfg、sys_partition.fex 会覆盖 default 目录的 BoardConfig.mk、env.cfg、sys_partition.fex。

3. 同级别目录的，BoardConfig_nor.mk、env_nor.cfg、sys_partition_nor.fex 会覆盖 BoardConfig.mk、env.cfg、sys_partition.fex。注意，仅当在 SDK 配置中选择 nor 方案才会。

4. 当 bsp、buildroot 等系统方案没有配置 BoardConfig.mk、env.cfg、sys_partition.fex，会使用 default 目录的 BoardConfig.mk、env.cfg、sys_partition.fex。

5. 注意，BoardConfig.mk 是包含关系，由系统方案的 BoardConfig.mk 往上层包含，如果系统方案和上层有相同属性，才会覆盖，否则视为增加。

4.1. 方案配置

从上面的信息可以了解到，SDK 编译时，会根据 BoardConfig.mk 文件的配置进行初始化，决定依照何种方案开始编译打包。

以 ebyte_linux_t527 的 buildroot 方案为例，BoardConfig.mk 文件内容如下：

```
1   LICHEE_BUILDING_SYSTEM:=buildroot
  1 LICHEE_BR_VER:=202205
  2 LICHEE_BR_DEFCONF:=sun55iw3p1_aiot_t527_defconfig
  3 LICHEE_KERN_DEFCONF:=buildroot_linux_defconfig
~
~
~
<yte_linux_t527/buildroot/BoardConfig.mk
1   LICHEE_BRANDY_DEFCONF:=sun55iw3p1_t527_defconfig
  1 LICHEE_RTOS_PROJECT_NAME:=t527_e906_demo
~
~
~
~
~
~
<configs/ebyte_linux_t527/BoardConfig.mk
```

我们主要关注其中这几项：

- LICHEE_KERN_DEFCONF：内核 defconfig 文件名称，该文件位于 linux-5.4 目录中。
- LICHEE_BR_VER：buildroot 的版本，会自动去 SDK/buildroot/目录中寻找对应版本。
- LICHEE_BR_DEFCONF：buildroot 的默认配置，在对应 buildroot 版本源码所在路径。
- LICHEE_BRANDY_DEFCONF：这是 u-boot 的默认配置，在 SDK/brandy/brandy-2.0/u-boot-2018 中。

4.2. SPL

从上面的信息可以了解到，对于 SPL 阶段，我们大部分情况下只需要修改 sys_config.fex 文件即可。

该文件在 kernel 版本大于等于 5.4 版本开始，主要用于 SPL，u-boot 相关配置，SDK 在编译/打包时会根据该文件配置针对当前开发板做一些配置，我们主要需要修改的部分有下面几个：

4.2.1. 目标板平台相关信息

```
;-------------------------------------------------------------------------------
; debug_mode=0 表示关闭启动时 Boot0/SPL 的打印 log，默认为 8。
; 0：关闭所有打印
; 1：只显示关键节点打印
; 4：打印错误信息
; 8：打印所有 log 信息
;-------------------------------------------------------------------------------
; earseflag 表示是否执行烧写是是否执行擦除操作
; 0x0：不进行擦除。
; 0x01：进行擦除，private，secure storage、user data、mbr 分区保留，其他擦除。
; 0x11：进行擦除，private，secure storage 分区保留，其他擦除。
; 0x12：强制擦除(整个 flash)。
```

```
;-------------------------------------------------------------------------
[platform]
eraseflag    = 1
debug_mode   = 1
```

4.2.2. 目标板存储设备信息

```
;-------------------------------------------------------------------------
;storage_type    = boot medium, 0-nand, 1-card0, 2-card2, -1(default)auto scan
;-------------------------------------------------------------------------
[target]
storage_type     = -1      ; 推荐使用自动识别方式，或根据实际使用的存储设备配置
```

4.2.3. 目标板调试串口信息

```
;-------------------------------------------------------------------------
; 描述 gpio 的形式：Port:端口+组内序号<功能分配><内部电阻状态><驱动能力><输出电平状态>
;-------------------------------------------------------------------------
[uart_para]
uart_debug_port = 0
uart_debug_tx    = port:PB09<2><1><default><default>
uart_debug_rx    = port:PB10<2><1><default><default>
```

此处应该针对用户自己的底板配置来确定使用哪一个串口作为调试串口，建议使用串口0，否则还需要修改 buildroot 构建的根文件系统中相关的内容。但是对于串口 0 实际使用哪些 GPIO 不作要求，可以通过查看《T527_Datasheet_V0.91.pdf》文档查找哪些引脚可以复用为串口 0。

4.2.4. 其他需求

默认情况下不推荐修改 SPL，如确实需要在 SPL 阶段执行一些特殊修改的用户，请参考《Linux_SPL-PUB_开发指南.pdf》文档。

4.3. u-boot

4.3.1. 设备树

uboot-board.dts 文件是 uboot 阶段初期使用设备树文件，只作为初始阶段使用，当 u-boot 执行到 sunxi_replace_fdt_v2 函数后，就会开始使用 kernel 阶段的设备树信息，所以该文件中只需要配置极少部分情况即可。

4.3.2. 环境变量

env.cfg 文件是 u-boot 阶段的环境变量文件，用户可自行配置，但是其中有几个地方比较特殊：

4.3.2.1. bootcmd 变量

```
int sunxi_update_bootcmd(void)
{
    char  boot_commond[128];
    int   storage_type = get_boot_storage_type();
    memset(boot_commond, 0x0, 128);
    strncpy(boot_commond, env_get("bootcmd"), sizeof(boot_commond)-1);
    debug("base bootcmd=%s\n", boot_commond);

    if ((storage_type == STORAGE_SD) || (storage_type == STORAGE_EMMC) ||
        (storage_type == STORAGE_EMMC3) || (storage_type == STORAGE_EMMC0)) {
        sunxi_str_replace(boot_commond, "setargs_nand", "setargs_mmc");
        debug("bootcmd set setargs_mmc\n");
    } else if (storage_type == STORAGE_NOR) {
        sunxi_str_replace(boot_commond, "setargs_nand", "setargs_nor");
    } else if (storage_type == STORAGE_NAND) {
#ifdef CONFIG_SUNXI_UBIFS
#ifndef CONFIG_MACH_SUN8IW18
        if (nand_use_ubi()) {
            sunxi_str_replace(boot_commond, "setargs_nand", "setargs_nand_ubi");
            debug("bootcmd set setargs_nand_ubi\n");
        }
#endif
#else
        debug("bootcmd set setargs_nand\n");
#endif
    }
```

上面的代码存在于 SDK/brandy/brandy-2.0/u-boot-2018/board/sunxi/board_helper.c 文件中，意思是对于 bootcmd 变量的值，会根据实际的启动存储器设备来自动切换，切换规则是改变"setargs_nand"到对应环境，所以用户可以保持使用下面这样的配置，由 u-boot 自动切换。

```
bootcmd=run  setargs_nand  boot_normal
```

4.3.2.2. mmc_root 变量

```
int sunxi_update_partinfo(void)

        memset(part_name, 0, PARTITION_NAME_MAX_SIZE);
        if (storage_type == STORAGE_NAND) {
            sprintf(part_name, "nand0p%d", index);
        } else if (storage_type == STORAGE_NOR) {
            sprintf(part_name, "mtdblock%d", index);
        } else {
            sprintf(part_name, "mmcblk0p%d", index);
        }

        temp_offset = strlen((char*)info.name) + strlen(part_name) + 2;
        if (temp_offset >= PARTITION_SETS_MAX_SIZE) {
            printf("partition_sets is too long, please reduces "
                   "partition name\n");
            break;
        }
        sprintf(partition_index, "%s@%s:", info.name, part_name);
        if (root_partition && strncmp(root_partition, (char *)info.name, sizeof(info.name)) == 0)
            sprintf(root_part_name, "/dev/%s", part_name);
        if (blkoops_partition && strncmp(blkoops_partition, (char *)info.name, sizeof(info.name)) == 0
            sprintf(blkoops_part_name, "/dev/%s", part_name);
        offset += temp_offset;
        partition_index = partition_sets + offset;

    }
    partition_sets[offset - 1] = '\0';
    partition_sets[PARTITION_SETS_MAX_SIZE - 1] = '\0';
    env_set("partitions", partition_sets);

#ifdef CONFIG_SUNXI_DM_VERITY
    if (*root_part_name != 0) {
        printf("set root to dm-0\n");
        env_set("verity_dev", root_part_name);
        if (storage_type == STORAGE_NAND)
            env_set("nand_root", "/dev/dm-0");
        else if (storage_type == STORAGE_NOR)
            env_set("nor_root", "/dev/dm-0");
        else
            env_set("mmc_root", "/dev/dm-0");
    }
#else
    if (*root_part_name != 0) {
        printf("set root to %s\n", root_part_name);
        if (storage_type == STORAGE_NAND)      gerrit-sdk, 3个月前 • da-t113-tina5.0-v1.0 release
            env_set("nand_root", root_part_name);
        else if (storage_type == STORAGE_NOR)
            env_set("nor_root", root_part_name);
        else
            env_set("mmc_root", root_part_name);
```

上面的代码存在于 SDK/brandy/brandy-2.0/u-boot-2018/board/sunxi/board_helper.c 文件中，意思是对于 xxx_root 变量的值，会根据实际的启动存储器设备来自动切换，切换规则是改变"xxx_root"到 root_part_name 值，一般情况下应该交由 u-boot 自动切换，。

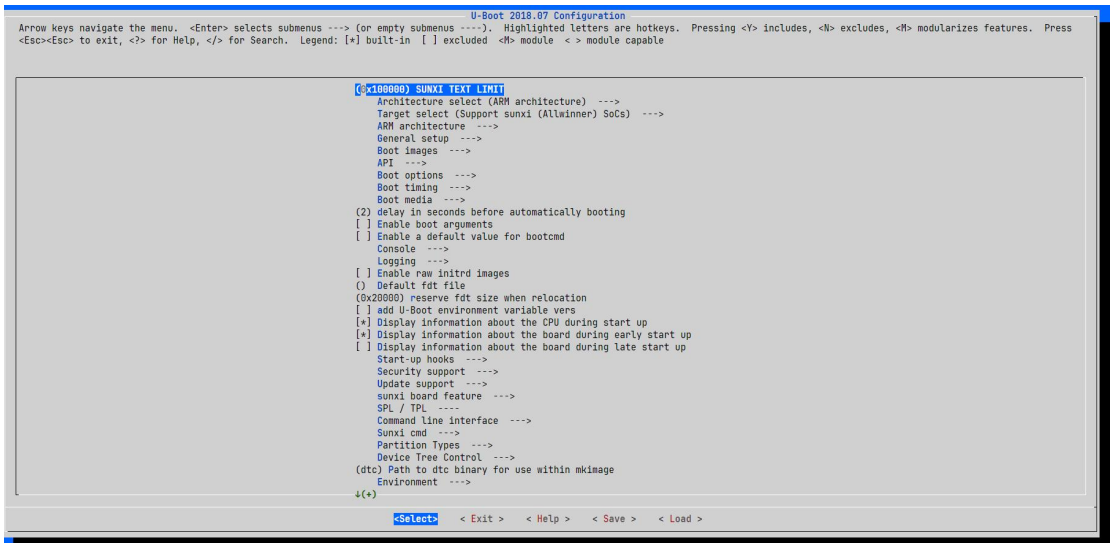
```
[    0.584401] 001: cfg80211: failed to load regulatory.db
[    0.606698] 001: Waiting for root device /dev/mmcblk0p5...
[    0.779385] 001: usb 1-1: new high-speed USB device number 2 using sunxi-ehci
[    1.031897] 001: hub 1-1:1.0: USB hub found
[    1.035936] 001: hub 1-1:1.0: 4 ports detected
[    1.120311] 000: [SNDCODEC][sunxi_check_hs_detect_status][191]:plugin --> switch:3
[    1.389589] 000: random: fast init done
[    1.489365] 001: usb 1-1.3: new high-speed USB device number 3 using sunxi-ehci
[    1.919379] 001: usb 1-1.4: new high-speed USB device number 4 using sunxi-ehci
[    2.222961] 001: cdc_ether 1-1.4:2.0 eth1: register 'cdc_ether' at usb-sunxi-ehci-1.4, CDC Ethernet Device, 00:e0:99:d3:98:19
[    4.649368] 001:
[    4.649368] 001: insmod_device_driver
[    4.649368] 001:
[    4.649661] 001: sunxi_usb_udc 4100000.udc-controller: 4100000.udc-controller supply udc not found, using dummy regulator
```

4.3.3. defconfig

u-boot 阶段的 defconfig 配置文件是根据 BoardConfig.mk 文件中 LICHEE_BRANDY_DE

FCONF 项的值来确定的，在 evb1_auto 的 buildroot 方案中，该文件为 brandy/brandy-2.0/u-boot-2018/configs/sun55iw3p1_t527_defconfig。一般不推荐修改，如果用户确实需要修改，请按照如下步骤进入 menuconfig 界面：

```
cd brandy/brandy-2.0/u-boot-2018
make menuconfig
```



需要注意的是，如果重新配置了 defconfig，则刚才通过 menuconfig 配置的会被覆盖掉。

4.4. 内核

与内核相关的配置选项都存放在<SDK>/device/product/configs/<board>/linux-5.15 目录中，其中主要就是两个文件，一个为 board.dts，一个为 bsp_defconfig。其中 board.dts 在编译过程中会软连接到 SDK/device/product/configs/<board>/board.dts，并最终生成 SDK/out/kernel/staging/sunxi.dtb

4.4.1. 设备树

从上面我们知道，当前的板级设备树配置文件是 board.dts，对于 board.dts 文件，大部分节点都有相应的注释描述解释供用户参考，但是有部分地方较为特殊：

1. spi-nand 节点
2. sdc0 节点
3. sdc2 节点

```
int sunxi_update_fdt_para_for_kernel(void)
#endif
    /* fix nand&sdmmc */
    switch (storage_type) {
    case STORAGE_NAND:
        fdt_enable_node("nand0", 1);
        fdt_enable_node("spi0", 0);
        break;
    case STORAGE_SPI_NAND:
#ifdef CONFIG_SUNXI_UBIFS
#ifdef CONFIG_MACH_SUN8IW18
        if (nand_use_ubi() == 0) {
            /* sun8iw18 aw-nftl config */
            fdt_enable_node("spinand", 1);
            fdt_enable_node("spi0", 0);
        }
#else
        /* sun50iw11 config */
        fdt_enable_node("spi0", 1);
        fdt_enable_node("/soc/spi/spi-nand", 1);
#endif
#else
        /* legacy config */
        fdt_enable_node("spinand", 1);      gerrit-sdk
        fdt_enable_node("spi0", 0);
#endif
        break;
    case STORAGE_EMMC:
        ret = fdt_enable_node("sunxi-mmc2", 1);
        if (ret)
            fdt_enable_node("mmc2", 1);
#ifdef CONFIG_SUNXI_SDMMC
        mmc_set_mmcblckx("sunxi-mmc2");
#endif
        break;
    case STORAGE_EMMC0:
        ret = fdt_enable_node("sunxi-mmc0", 1);
        if (ret)
            fdt_enable_node("mmc0", 1);
        break;
    case STORAGE_EMMC3:
        ret = fdt_enable_node("sunxi-mmc3", 1);
        if (ret)
            fdt_enable_node("mmc3", 1);
        break;
    case STORAGE_SD:
```

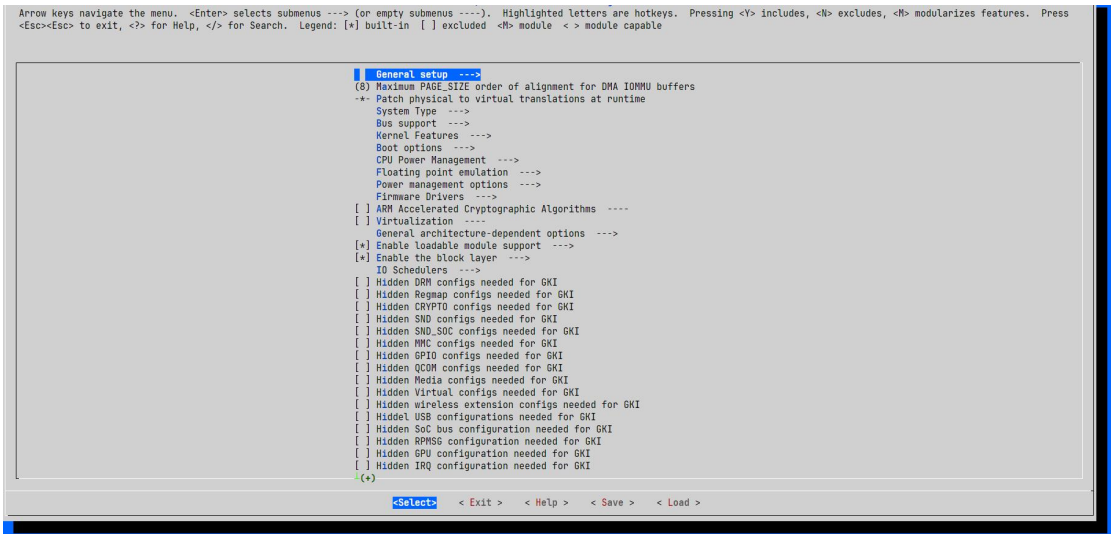
上面的代码存在于 SDK/brandy/brandy-2.0/u-boot-2018/board/sunxi/board_helper.c 文件中，意思是 u-boot 会根据启动时对应的存储设备信息来自动使能/失能 kernel 设备树中的部分节点，所以用户可能会发现虽然对应存储设备的设备节点配置为`status = "disabled"`依然可以正常进入系统。

4.4.2. defconfig

从上面可以知道，板级的 defconfig 文件是根据 BoardConfig.mk 文件中 LICHEE_KERN_DEFCONF 值确定，配置文件一定存在于<SDK>/device/product/configs/<board>/linux-5.15 目录中。

此处以 ebyte_linux_t527 为例，其内核 defconfig 文件为<SDK>/device/product/configs/ebyte_linux_t527/linux-5.15/bsp_defconfig。若用户需要修改配置，可在 SDK 顶层目录中使用下面的方式进入 menuconfig 界面进行配置：

```
./build.sh  menuconfig
```



修改完毕以后用户可以通过下面命令更新并保存新的配置到 bsp_defconfig 文件：

```
./build.sh  saveconfig
```

4.5. 文件系统

4.5.1. Buildroot

从上文可以知道，与 buildroot 有关的板级相关配置存储在<SDK>/device/product/configs/<board>/[buildroot]/BoardConfig.mk 文件中，主要是 LICHEE_BR_VER 与 LICHEE_BR_DEFCONF。此处我们以 ebyte_linux_t527 为例，使用的 buildroot 版本为 202205，其源码文件

位于 SDK/buildroot/buildroot-202205，其编译配置文件位于 buildroot 源码目录下的 configs/sun55iw3p1_aiot_t527_defconfig。

更多的 buildroot 相关使用资料请查阅 buildroot 官方手册：https://buildroot.org/downloads/manual/manual.html。

4.5.2. 无法登录

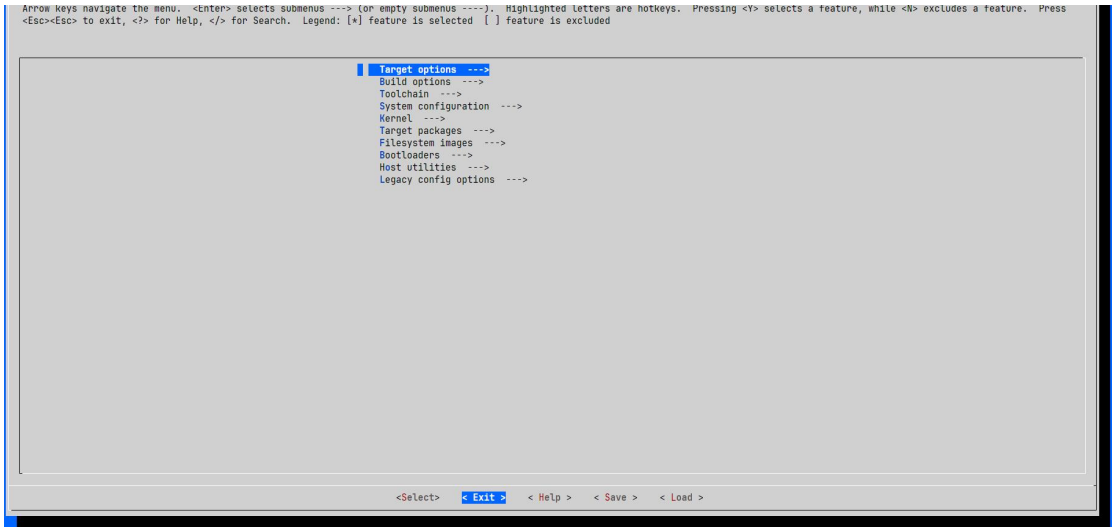
对于使用 buildroot 版本为 202205 的用户，如果出现无法登录或登录后发现需要连续输入两次相同内容才会在终端显示一次，如输入`aa`，终端显示`a`，请修改如下文件：`<SDK>/buildroot/config/buildroot/buildroot/post_build.sh`，在其中添加如下内容：

```
+               # change ttyAS0 to ttyS0
+               if grep -q "ttyAS0" ${TARGET_DIR}/etc/inittab; then
+                       echo "found ttyAS0, change it to ttyS0!"
+                       sed -i 's/ttyAS0/ttyS0/g' ${TARGET_DIR}/etc/inittab
+               fi
+
                #commented ttyS0, insert /bin/sh
                grep "#ttyS0::respawn:" ${TARGET_DIR}/etc/inittab >/dev/null
```

4.5.3. defconfig

对于大多数用户，除了内核需要适配底板以外，还需要修改根文件系统以适配自己的使用需求，推荐使用 menuconfig 的方式修改配置，进一步的适配自己的方案需求。可以通过下面的命令进入 buildroot 的 menuconfig 界面：

```
./build.sh buildroot_menuconfig
```



修改完毕以后用户可以通过下面命令更新并保存新的配置到 sun55iw3p1_aiot_t527_defconfig 文件：

```
./build.sh  buildroot_saveconfig
```

4.5.4. 分区

sys_partition.fex 文件描述了打包后镜像的分区情况，用户主要需要修改的部分为 rootfs 分区大小以及该分区应该放入的根文件系统，不得小于实际根文件系统大小，也不得大于存储设备物理可用大小。

对于 eMMC 或 SD 卡存储设备，rootfs 分区应该大致如下面的情况：

```
[partition]
    name         = rootfs
    size         = 16777216
    downloadfile = "rootfs.fex"
    user_type    = 0x8000
```

4.6. RISC-V

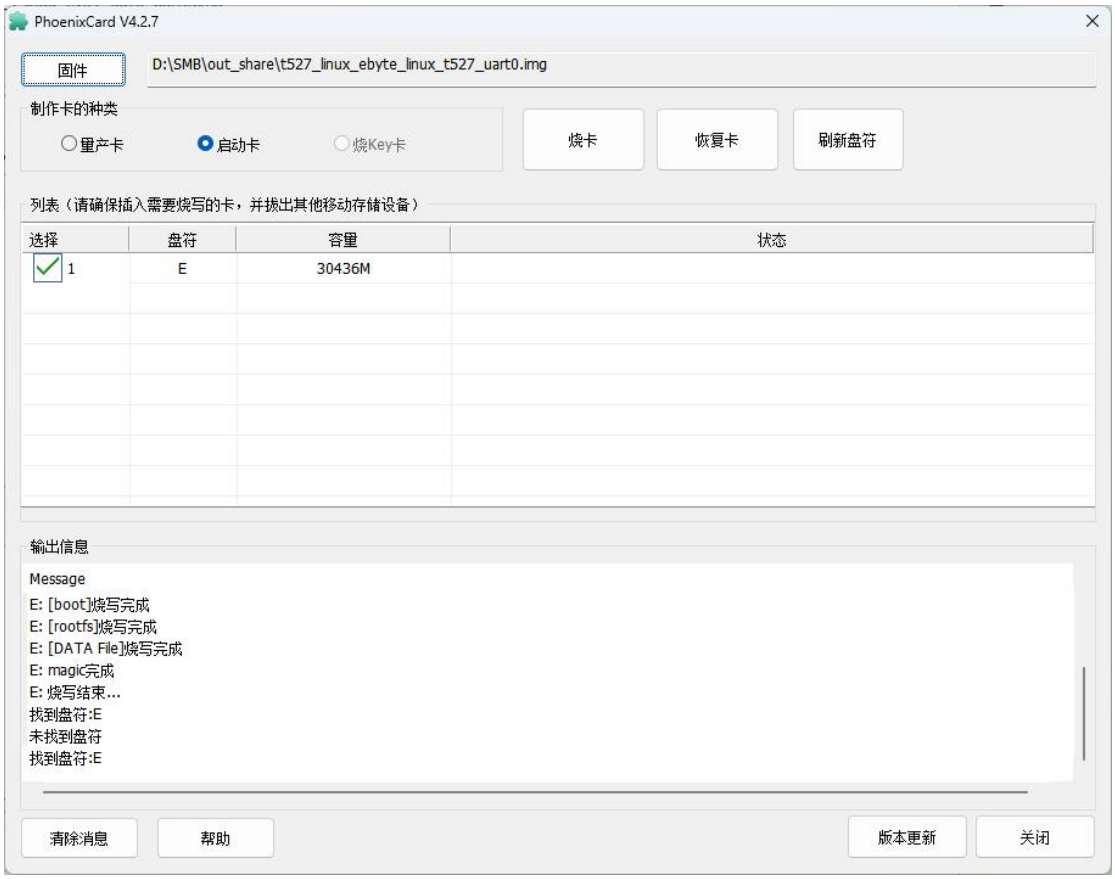
对于 RISC-V 移植使用，客户可以参考《FreeRTOS_RTOS_系统_开发指南.pdf》，此处不再赘述。

5. 如何烧录更新系统镜像

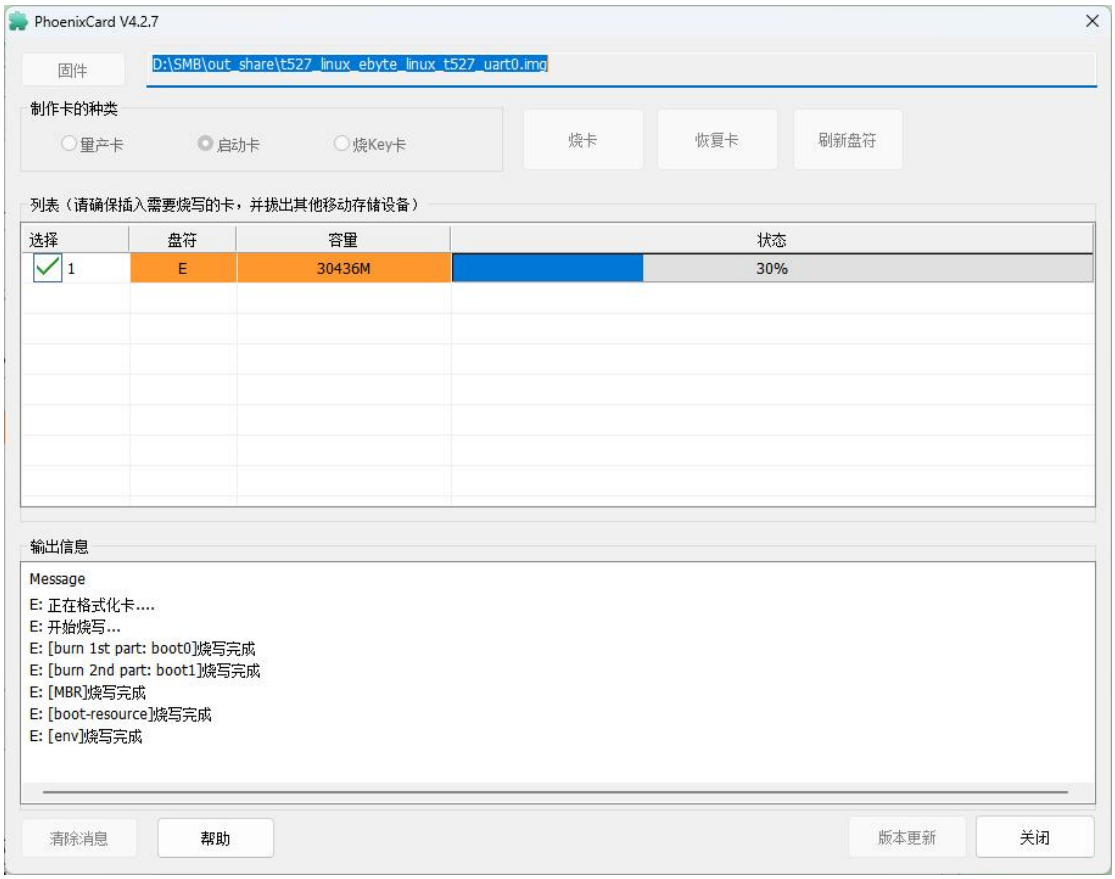
本章主要介绍基于 ECK32-T527 核心板的烧录更新方式。

5.1. 制作 SD 卡启动卡

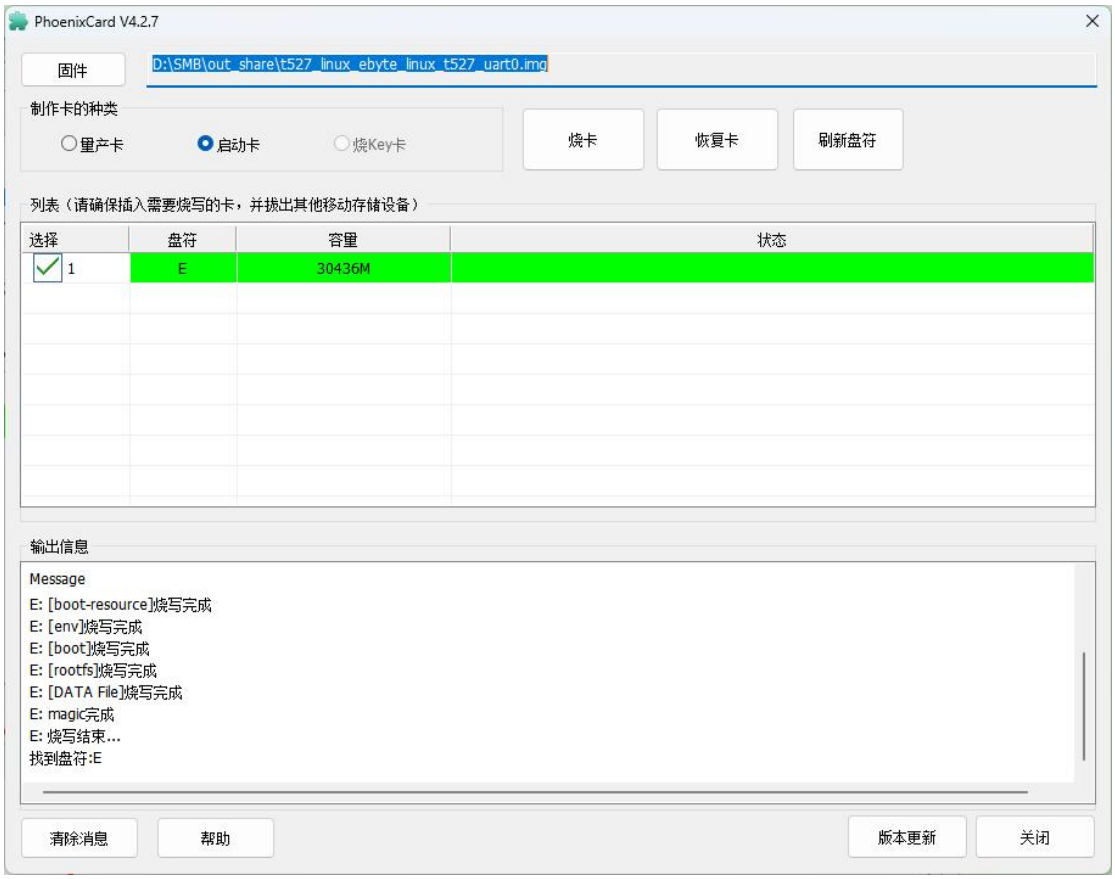
首先打开 PhoenixCard 程序，界面如下：



用户请依次选择待烧写固件，选择卡的种类为“启动卡”，而后选择待烧写的 SD 卡设备，该软件会自动扫描可用的移动存储设备，请注意不要选择了错误的设备。最后点击“烧卡”按钮，烧卡过程中会实时显示当前进度：



正式烧写完毕后界面如下，可以通过软件界面下方的输出信息查看是否确实制作启动卡成功：



5.2. 量产卡烧录

在制作 SD 卡启动卡的时候我们还会发现其中存在一个量产卡的制作种类，该种方式可用通过制作一张 SD 卡来进行流水线操作的烧写功能，用户无需使用单独的镜像固件，直接使用启动卡阶段可以正确使用的镜像即可。

制作完毕以后从该量产卡启动系统，连接调试串口我们可以查看烧写过程全部的打印信息：

```
I2C:   ready
[00.594]optee version: major:3 minor:7
[00.598]DRAM:  [00.599]drm_base=0x0
[00.601]drm_size=0x0
[00.603]dram_base=0x40000000
[00.605]dram_size=0x40000000
1 GiB
[00.610]Relocation Offset is: 3ce93000
[00.640]secure enable bit: 0
E/TC:0   fdt_getprop_u32:336 prop trace_level not found
[00.654]CPU=1008 MHz,PLL6=600 Mhz,AHB=200 Mhz, APB1=100Mhz  MBus=300Mhz
[00.660]gic: sec monitor mode
[00.663]sunxi flash map init
SPI ALL:   ready
[00.668]line:719 init_clocks
[00.671]init_clocks:finish
[00.673]flash init start
[00.675]workmode = 17,storage type = 1
try card 2
set card number 2
get card number 2
[00.683][mmc]: mmc driver ver uboot2018:2023-07-4 16:18:00
[00.690][mmc]: Is not Boot mode!
[00.692][mmc]: SUNXI SDMMC Controller Version:0x50310
[00.704][mmc]: ************Try SD card 2************
[00.709][mmc]: mmc 2 cmd timeout 100 status 100
[00.713][mmc]: smc 2 err, cmd 8,  RTO
[00.716][mmc]: mmc 2 close bus gating and reset
[00.721][mmc]: mmc 2 cmd timeout 100 status 100
[00.725][mmc]: smc 2 err, cmd 55,  RTO
[00.729][mmc]: mmc 2 close bus gating and reset
[00.733][mmc]: ************Try MMC card 2************
[00.756][mmc]: mmc 2 cmd timeout 100 status 100
[00.760][mmc]: smc 2 err, cmd 8,  RTO
[00.763][mmc]: mmc 2 close bus gating and reset
[00.768][mmc]: mmc 2 cmd timeout 100 status 100
[00.772][mmc]: smc 2 err, cmd 55,  RTO
[00.776][mmc]: mmc 2 close bus gating and reset
[00.791][mmc]: gen_tuning_blk_bus8: total blk 10
[00.795][mmc]: gen_tuning_blk_bus4: total blk 6
[00.799][mmc]: Using 4 bit tuning now
[00.809][mmc]: write_tuning_try_freq: write ok
[00.814][mmc]: Pattern compare ok
[00.817][mmc]: Write tuning pattern ok
[00.820][mmc]: ================== HSSDR52_SDR25 ...
[00.825][mmc]: skip freq 400000
[00.828][mmc]: skip freq 25000000
[00.831][mmc]: freq: 2-50000000-64-4
```

在系统引导进入 u-boot 阶段后开始正式烧写系统。

```
[61.690]successed in writting part dsp0
origin_verify value = 2e679ba6, active_verify value = 2e679ba6
[61.708]successed in verify part dsp0
[61.711]successed in download part dsp0
[61.716]successed in downloading part
uboot size = 0x168000
storage type = 2
sunxi_sprite_deal_uboot ok
[61.880]successed in downloading uboot
[61.886][mmc]: write mmc 2 info ok
dram para[0] = 318
dram para[1] = 3
dram para[2] = 7b7bfb
dram para[3] = 1
dram para[4] = 1102
dram para[5] = 4000000
dram para[6] = 1c70
dram para[7] = 42
dram para[8] = 18
dram para[9] = 0
dram para[10] = 4a2195
dram para[11] = 2423190
dram para[12] = 8b061
dram para[13] = b4787896
dram para[14] = 0
dram para[15] = 48484848
dram para[16] = 48
dram para[17] = 1620121e
dram para[18] = 0
dram para[19] = 0
dram para[20] = 0
dram para[21] = 770000
dram para[22] = 2
dram para[23] = b4056103
dram para[24] = 0
dram para[25] = 0
dram para[26] = 0
dram para[27] = 0
dram para[28] = 0
dram para[29] = 0
dram para[30] = 0
dram para[31] = 0
storage type = 2
[61.951]successed in downloading boot0
CARD OK
[61.955]sprite success
sprite_next_work=3
next work 3
SUNXI_UPDATE_NEXT_ACTION_SHUTDOWN
[64.963][mmc]: mmc exit start
[64.985][mmc]: mmc 2 exit ok
```

出现上图类似内容后用户可以断开开发板供电，拔出量产卡，而后重新上电即可，串口终端会打印如下启动信息用于确定是从 eMMC 上启动的：

```
[80]HELLO! BOOT0 is starting!
[83]BOOT0 commit : ad8b4b44ac
[87]periph0 has been enabled
[90]set pll end
[92]PL gpio voltage : 3.3V
[96][pmu]: bus read error
[99]PMU: AXP2202
[104]PMU: AXP1530
[109]power mode:33, sys_vol:920
[114]vaild para:1  select dram para0
[117]board init ok
[120]rtc[3] value = 0xb00f
[125]enable_jtag
[127]card no is 2
[129]sdcard 2 line count 8
[132][mmc]: mmc driver ver 2023-03-24 16:23
[142][mmc]: Wrong media type 0x0, but host sdc2, try mmc first
[149][mmc]: ***Try MMC card 2***
[174][mmc]: RMCA OK!
[177][mmc]: bias 17fb
[179][mmc]: mmc 2 bias 17fb
[188][mmc]: MMC 5.1
[190][mmc]: HSSDR52/SDR25 8 bit
[194][mmc]: 50000000 Hz
[196][mmc]: 14930 MB
[199][mmc]: ***SD/MMC 2 init OK!!!***
[209]DRAM BOOT DRIVE INFO: V0.67
[213]DRAM_VCC set to 1160 mv
[216]DRAM CLK =1200 MHZ
[219]DRAM Type =8 (3:DDR3,4:DDR4,7:LPDDR3,8:LPDDR4)
[229]DRAM SIZE =2048 MBytes, para1 = 310a, para2 = 8000000, tpr13 = 6461
[259]DRAM simple test OK.
[262]dram size =2048
```

5.3. 官方工具烧录

本章节描述如何使用全志官方工具 PhoenixSuit 实现 USB 快速烧录功能。

5.3.1. 进入烧录模式

使用之前我们需要了解如何让小机进入烧写模式，按照官方文档，有如下四种方式：

1. 开机时按住 fel 按键
2. 开机时连接串口，按住数字键"2"
3. 进入 u-boot 控制台输入："efex"
4. 进入系统，在系统控制台输入："reboot efex"

1. 一般常用方式为开机按住 fel 按键，这就需要用户在制作底板时拉出相应引脚并连接到一个按键上，这种方式无需存储设备上有可用系统。对于方式二，需要注意，这个功能是

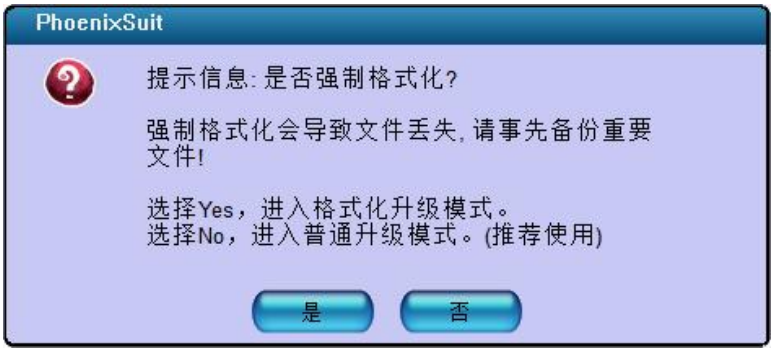
在全志官方提供的 boot 固件中实现的，如果用户在 sys_config.fex 中配置了错误的串口描述，方式二将失效，但是无需担心，还可以通过制作量产卡的方式重新烧写正确的固件。

5.3.2. 烧录工具配置

再来看烧写工具，安装并打开该工具后，界面如下：

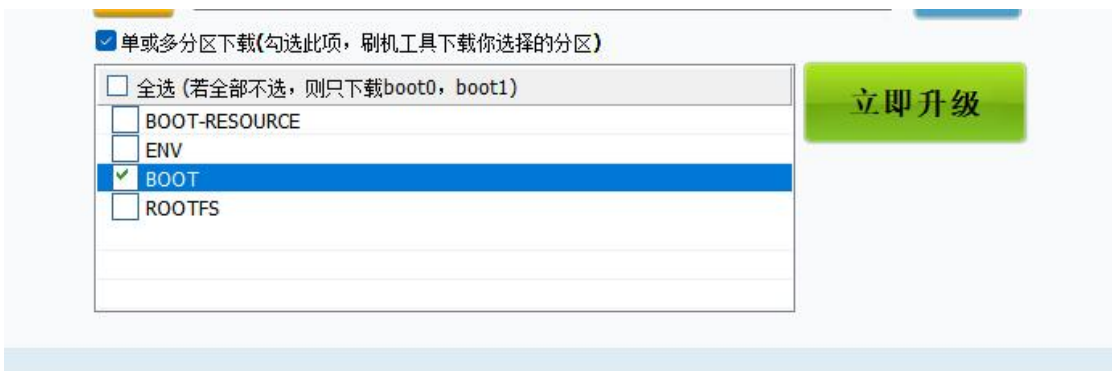
点击上方的“一键刷机”项，进入到镜像选择。

用户不用管立即升级按钮，只要在打开该软件，选择好烧写镜像，usb 连线正常连接后，一旦待烧写小机进入烧写模式，会自动识别并询问烧写情况：



用户按需选择是否格式化升级。

与此同时，用户还可以通过如下图配置实现单独烧录部分内容：



如此处表示只烧录 boot0，uboot，内核三部分，不更新文件系统。

5.4. 可能遇到的问题

5.4.1. 制作启动卡找不存储设备

用户在使用该工具烧写启动卡时，可能发现接入 SD 卡后该工具找不到相应存储设备，请确定该 SD 卡是否被 VMware 占用，若被占用请解除占用归还使用权给 host 主机，若依然不可用请尝试更换 USB 接口，或查看是否被其他程序占用。

6. 参考资料

- T527_Datasheet_V0.91.pdf
- T527_User_Manual_V0.92.pdf
- 全志开发社区：https://bbs.aw-ol.com/
- Linux kernel 开源社区：https://www.kernel.org/
- Debian wiki：https://wiki.debian.org/
- Buildroot-manual：https://buildroot.org/downloads/manual/manual.html

7. 修订说明

修订说明表

版本	修改内容	修改时间	编制	校对	审批
V1.0	初稿	2025-3-19	WYQ	WFX	WFX

8. 关于我们

销售热线：4000-330-990
技术支持：support@cdebyte.com 官方网站：https://www.ebyte.com
公司地址：四川省成都市高新西区西区大道 199 号 B5 栋

成都亿佰特电子科技有限公司
Chengdu Ebyte Electronic Technology Co.,Ltd.