Software Development Guide
ECK32-T527 core board



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

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Table of contents

Disclaimer and Copyright Notice 1
1. Overview 2
2. Development environment preparation 2
2.1. Hardware Preparation 2
2.2. Virtual machine software deployment 3
2.3. Host software installation 5
2.4. Data transmission 7
3. SDK compilation and construction 10
3.1. SDK acquisition 10
3.2. SDK directory 11
3.3. SDK configuration 13
3.4. Compile SDK 15
3.5. Compile separately 16
3.6. Compilation output 18
3.7. Image packaging 19
4. Development board adaptation 20
4.1. Solution Configuration 23
4.2. SPL 24
4.3. u-boot 26
4.4. kernel 29
4.5. File System 33
4.6. RISC-V 34
5. How to burn and update the system image 35
5.1. Make an SD card boot card 35
5.2. Mass production card burning 37
5.3. Official tool burning 40
5.4. Possible problems 43
6. References 44
7. Revision Notes 44
8. About Us 45

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Disclaimer and Copyright Notice

The information in this document is subject to change without notice. This document is provided "as is" without warranty of any kind, including any warranty of merchantabilit y, fitness for a particular purpose, or non-infringement, and any warranty otherwise expres sly stated in any proposal, specification, or sample. No liability is assumed for this docum ent, including liability for infringement of any patent arising from the use of the informati on in this document. No license, express or implied, by estoppel or otherwise, to any inte llectual property right is granted by this document.

The test data obtained in this article are all obtained by Ebyte laboratory testing, and the actual results may be slightly different.

All trade names, trademarks and registered trademarks mentioned herein are the prope rty of their respective owners and are hereby acknowledged.

The final right of interpretation belongs to Chengdu Ebyte Electronic Technology Co., Ltd.

Notice:

The contents of this manual may be changed due to product version upgrades or other reasons. Ebyt e Electronic Technology Co., Ltd. reserves the right to modify the contents of this manual without any n otice or reminder. This manual is for guidance only. Chengdu Ebyte Electronic Technology Co., Ltd. mak es every effort to provide accurate information in this manual. However, Chengdu Ebyte Electronic Techn ology Co., Ltd. does not guarantee that the contents of this manual are completely error-free, and all stat ements, information and suggestions in this manual do not constitute any express or implied warranty.

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

1. Overview

This article describes the complete process of customizing an embedded Linux system based on the Ebyte core board, including preparation for the development environment, code acquisition, bootloader and kernel porting, and customizing Rootfs to suit your application needs. We first explain how to build a system image for the EC K32-T527 core board based on the SDK source code we provide, and how to burn the built image to the development board. For users developing projects based on the EC K32-T527 core board, we focus on methods and key points for porting this system to their hardware platforms. Through practical porting cases and Buildroot-based Rootfs customization, we enable users to quickly customize a system image tailored to their hardware.

Please note that this document does not include an introduction to the buildroot project or basic knowledge of the Linux system. It is suitable for embedded Linux system developers and embedded Linux BSP developers with some development experience.

2. Development environment preparation

This chapter mainly introduces some software and hardware environments required before using the ECK32-T527 core board for development, including the necessary hardware configuration, development, and programming environment.

By reading this chapter, you will understand the installation and use of relevant hardware tools and software development and debugging tools. You will also be able to quickly build a relevant development environment to prepare for subsequent development and debugging.

Host: i7-12700 + Windows 11 22H2

Virtual Machine: Debian bullseye

2.1. Hardware Preparation

2.1.1. Debug serial port

Debug serial port: baud rate 115200, 8 data bits, 1 stop bit, no parity check , no hardware flow control. Please configure the actual serial port yourself. The configuration method is detailed below.

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2.1.2. Startup method

For users of our ECK32-T527 core board, all they need to remember is that once a bootable SD card is inserted, the system will boot from the SD card. If the SD card cannot be booted, it will boot from the onboard storage device.

2.2.Virtual machine software deployment

This chapter describes how to set up a development environment for the T527 . By reading this chapter, you'll understand the installation and use of relevant development and debugging tools. This will allow you to quickly build a development environment and prepare for subsequent development and debugging.

2.2.1. Virtual Machine Configuration

Currently, the more common virtual machine software includes VirtualBox, VMware, etc. The configuration methods of different platforms are different, and user usage habits are also different. Therefore, please configure the virtual machine to use bridging to connect to the Internet according to the virtual machine software you use, and enable the directory sharing function with Windows.

2.2.2. Configure the compilation environment

Because the T527 SDK uses buildroot version 202205, we recommend that you use the Ubuntu 2204 64-bit distribution. You can also try using a deb distribution released in a similar year, but still ensure that the version is compatible with the buildroot version.

Install the software:

```
sudo dpkg --add-architecture i386 && sudo apt update
sudo apt install --install-recommends -y \
openssh-server bash-completion tig bear curl wget bsdmainutils \
locales shellcheck netcat-openbsd vim file pv clangd-16 \
xz-utils netpbm make gcc g++ flex bison libncurses-dev libssl-dev \
bzip2 cpio unzip rsync lzop kmod fakeroot automake \
libtool device-tree-compiler bc lsof repo
sudo apt install --install-recommends -y \
```

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
busybox libc6:i386 libgcc1:i386 libstdc++6:i386
```

Create a working directory:

```
mkdir ~/work
```

2.2.3. Enable the TFTP service of the virtual machine

In the later debugging stage, you may use TFTP to obtain files from the network. Therefore, you must first install and enable the TFTP service in the virtual machine. Use the following command to install the TFTP service:

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

By default, you can only download but not upload. If you want to customize the configuration, please refer to the official document: https://help.ubuntu.com/community/TFTP . After configuration in this document, the specified path is /srv/tftp.

Add users to the tftp group and modify the shared directory permissions:

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

If the preceding operations are not performed, permission issues may occur and files cannot be written to the shared directory as a non-root user.

2.2.4. Enable NFS service for the virtual machine

In the later debugging phase, you often need to use NFS to share files. Therefore, you must first install and enable the NFS service in the virtual machine. Use the following command to install the NFS service:

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

Wait for the installation to complete. After the installation is complete, set the directory to be exported. Here, the default tftp directory is used directly. Users can modify it by themselves, or use a convenient text editor to modify the /ext/exports file. For configuration methods, refer to the man manual:

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

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
exports

    man exports
```

2.2.5. Enable the SSH service of the virtual machine

we can use the SSH client software to log in to the virtual machine under Windwos, such as using Moba X term, VSCode, and use the following command to enable the SSH service under the virtual machine:

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

The above command installs the SSH service. The SSH configuration file is /etc/ssh/sshd_config. Use the default configuration. For more configuration information, refer to the man page related to sshd_config.

2.3.Host software installation

This chapter will introduce how to build a T527 programming and debugging environment. By reading this chapter, you will understand the installation and use of related programming and debugging tools.

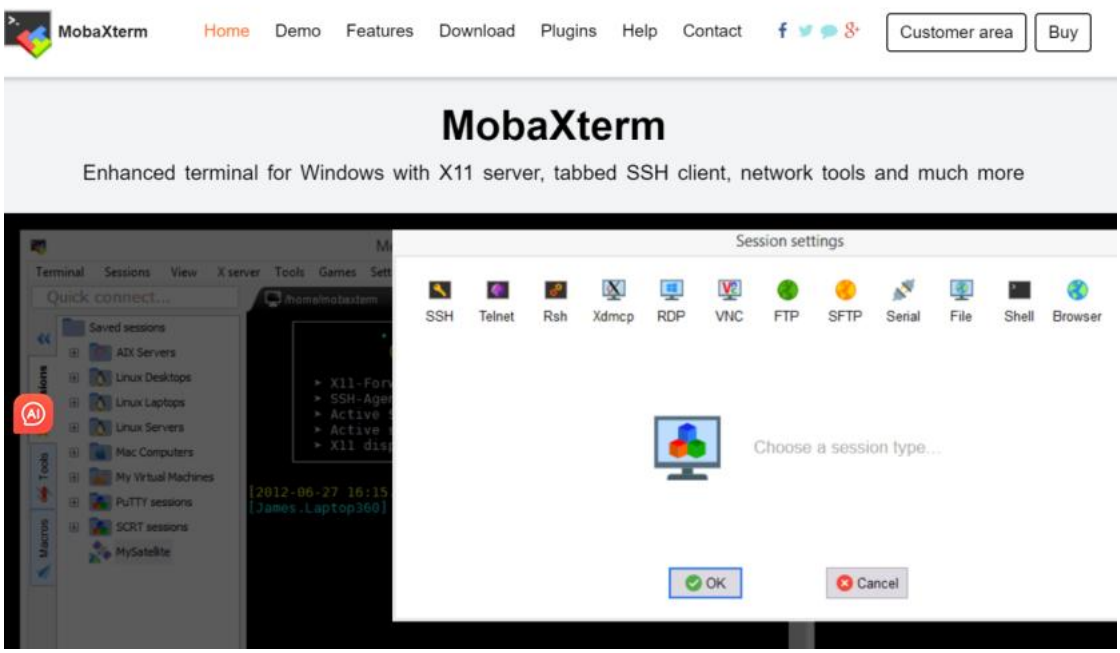
2.3.1. Moba X term

2.3.1.1. Software download and installation

MobaXterm is a terminal software that is powerful and free. It is recommended to use this software as terminal debugging software. MobaXterm software can be downloaded from its official website at https://mobaxterm.mobatek.net/ , as shown in the figure:

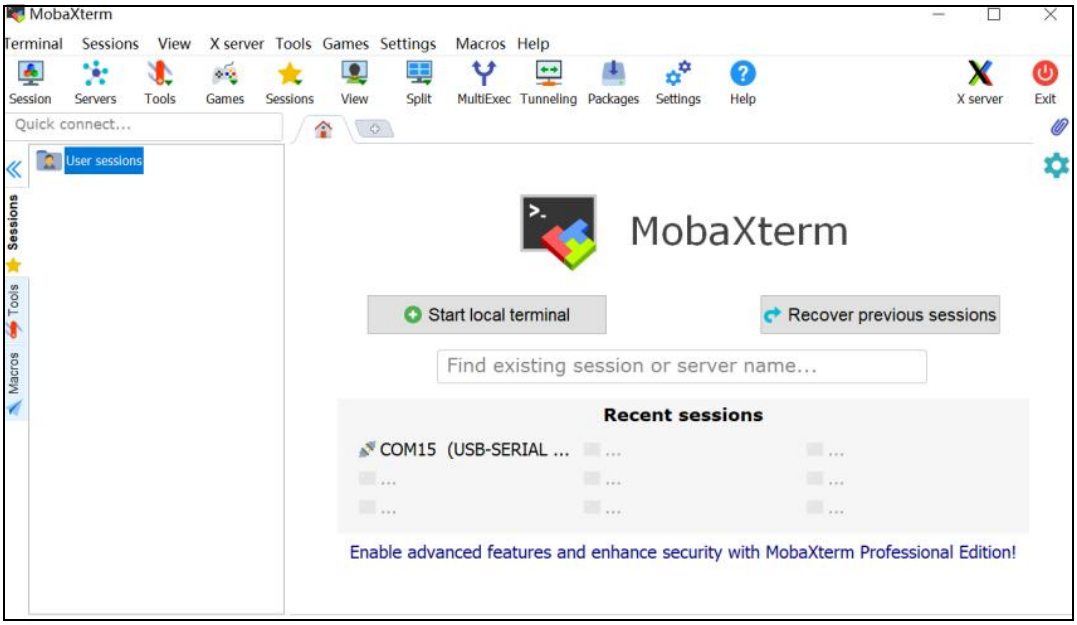
Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

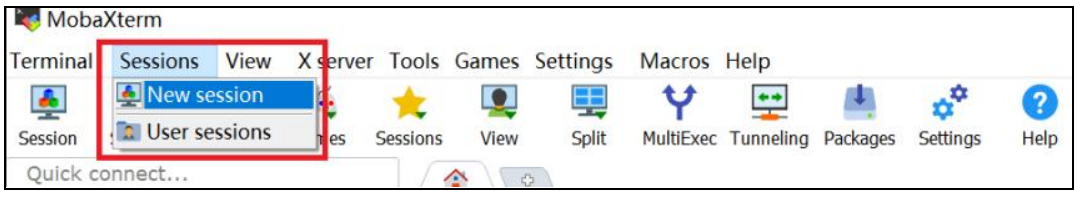


2.3.1.2. Software Usage

The software interface is shown in the figure:



Click the "Sessions->New session" button in the menu bar to open the new session window, as shown in the figure

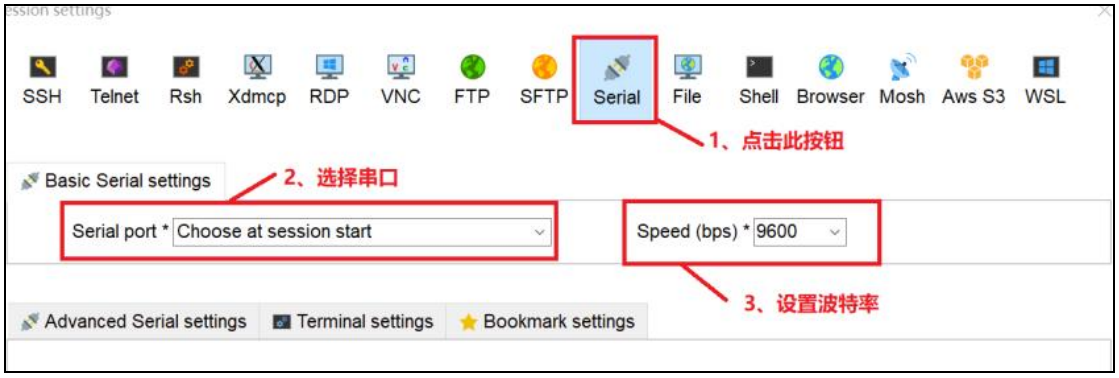


Establish a serial connection, that is, a serial port connection, because the main purpo se of using MobaXterm is to use it as a serial terminal. Click the "Serial" button in the f

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

igure to open the serial port setting interface, as shown in the figure:



After opening the serial port settings window, first select the serial port number to be set. Therefore, you must first connect the development board to the computer with a serial cable. Since the default serial port baud rate used by T527 is 115200, you need to set the baud rate to 115200.

2.3.2. PhoenixSuit

This tool is provided by Allwinner for burning the system from USB. The software is stored in the directory: " 04_Tools/PhoenixSuit " . For detailed usage, please refer to the "Official Tool Burning" section below.

2.3.3. PhoenixCard

This is an official Allwinner tool for creating system cards. You can choose to burn and create both production cards and boot cards. The software is located in the directory " 04_Tools/ PhoenixCard" . For detailed instructions, see the " Creating an SD Card Boot Card " section below.

2.4.Data transmission

In the previous chapters, we have configured various network services for the virtual machine. Now we just need to understand how to use them on the client and development board.

2.4.1. TFTP Client

In general, the development board only needs to use the download function. In this case, you can simply use the curl tool to download. Before using it, you need to select the curl function in buildroot, and then use curl to download the file (only Linux version curl, Windows can choose to use FileZilla):

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

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

1. /path/dist_file: Where to download, the name and path of the output file.

2. HOST: The IP address of the virtual machine. You need to pay attention to wheth er this address is accessible.

3. src_file: file directory and name. Note that you do not need to add the "TFTP_DI RECTORY" part of the configuration. For example, if the file is stored in `/srv/tftp/aaa/bb b/ccc`, the src_file value is "aaa/bbb/ccc".

2.4.2. NFS Client

In general, the development board only needs to use the mount command to mount t he NFS file system, but you need to install basic support and then mount NFS:

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

1. HOST: The IP address of the virtual machine. You need to pay attention to wheth er this address is accessible.

2. path: The path on the NFS server where you want to mount the file, such as /srv/ tftp/aaa/bbb.

3. mnt_point: Where you want to mount the NFS file system, such as /mnt.

2.4.3. SSH

Generally speaking, we can also use the SFTP and SCP functions that come with SS H to download files, but there are not many cases of directly using commands to upload and download. More third-party tools are used, such as the SFTP function after using Mo baXterm to log in to ssh, or VSCode using the remote-SSH plug-in to implement the do wnload function during the development process. The way to use it is to right-click the fi le to be downloaded and then select Download.

2.4.4. USB flash drive/SD card

For SD cards, you need to use a card reader, and the actual usage is the same as th at of a USB flash drive.

2.4.4.1. Copying data from a virtual machine

Users need to transfer the USB flash drive/SD card to the virtual machine according to the virtual machine software they use. Generally, it will be automatically mounted and

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

can be directly accessed in the file manager. After copying the data, uninstall the device in the file manager and then remove the device from the virtual machine.

If it is not mounted automatically, you can use the following two commands to mount it manually:

1. Method 1

```
udisksctl mount --block-device /dev/sd XY
```

2. Method 2

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

In this case, you need to have some understanding of USB drive/SD card partitions to specify the Y value in sdXY. The X value is usually 'b', but you also need to judge the situation based on the information returned by lsblk.

Use method 1 to mount. You can access it without root privileges and without specifying a path. However, you need to have the `udisks2` package installed on your system.

Manual uninstallation corresponds to the following command:

1. Method 1

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

2. Method 2

```
sudo umount /mnt_point
```

2.4.4.2. Copy data to the development board

For development boards, they are generally streamlined and may not have a udisks2 package or a GUI interface. Even if they have an interface, they may not support automatic mounting. However, users often have root permissions, so mount operations are generally performed from the command line.

1. Mount

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

2. copy

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

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

3. Unmount

```
umount /mnt_point
```

3. SDK compilation and construction

This section describes the complete process from obtaining the SDK to compiling and generating a usable image.

3.1. SDK acquisition

The T527 development SDK is included in the download file. Please unzip it to the development directory.

Since the complete SDK is too large, in order to reduce the SDK size, the SDK provided by our company is compressed through repo. After extraction, a hidden directory `.repo` is generated in the current directory.

Decompression requires the use of the repo tool, which has been installed according to the virtual machine software deployment section above. If the installed version is not suitable, you can refer to the following to install the latest repo tool. At the same time, remember to delete the repo previously installed through apt first:

```
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
```

Configure the repo tool to update the source using a mirror (optional in China):

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

Enable related environment variables:

```
source ~/.profile
```

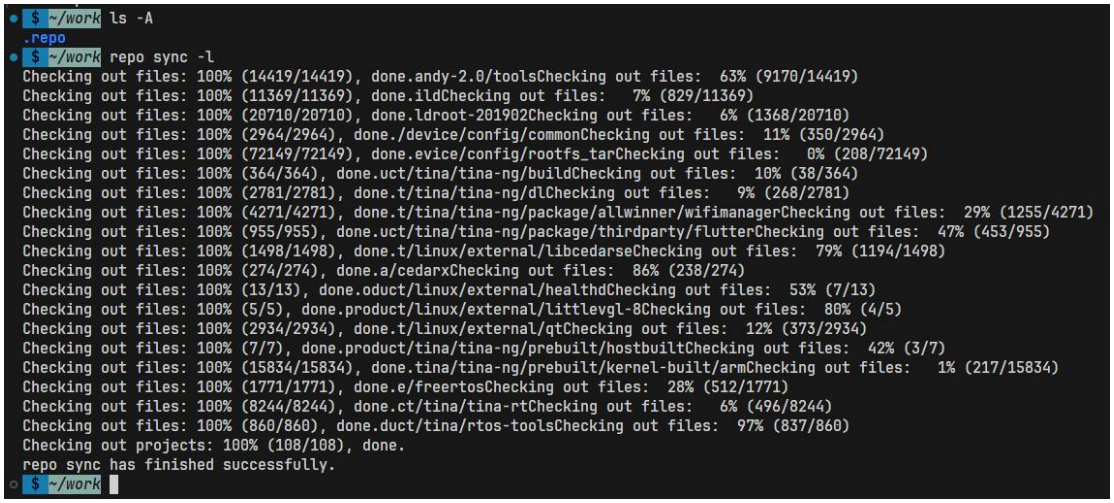
After entering the project directory, execute the following command to synchronize the code repository locally:

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
repo sync -l
```

Check the directory as follows:



3.1.1. Possible problems

Because the repo tool is a Python script, the Python version used by the user may be incompatible with the repo dependency version in the SDK. For users who use Python 2 by default, you may need to configure Python 3 by default. If an error message still occurs after setting it, go to the SDK/.repo/repo directory and run git pull to pull the latest repo version.

For users who are not using Debian/Ubuntu, the `source ~/.profile` command above may not add the `~/.local/bin/` directory to the environment variables. Please add it as follows:

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

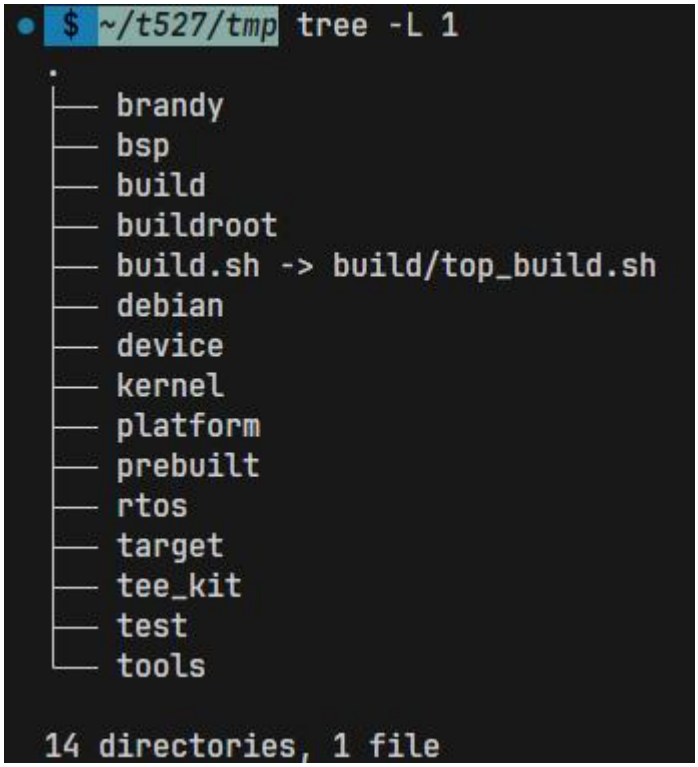
source ~/.bashrc
```

3.2. SDK directory

After decompression, the entire SDK directory is as follows:

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



We need to pay attention to the following directories:

Top-level directory	Subdirectory	effect
Brandy	brandy-2.0/spl-pub/	SPL source code is generally not modified, and most of the official files provided by Allwinner are binary files
	brandy-2.0/u-boot-2018/	U-boot source code
build		Compile script directory, generally not modified
buildroot	buildroot-20 2205	File system source code
	package/auto/sdk_demo/	Sample Program
debian	compressed_files/	Debian example image
device	config/chips/	The chip configuration directory supported by the SDK, including configuration solutions for multiple development boards
	product	By default, no, after executing SDK configuration, point to the target chip, here is T527
kernel	linux- 5.15	Kernel source code
bsp	configs/linux-5.15/	Store soc related base device tree
	drivers	Stores Allwinner related module drivers
platform		Platform extension related source code
prebuilt		Kernel, buildroot and other related compilation tools
rtos		rtos source code, mainly related to the RISC-V core
tools	tools_win	Some Windows tools
target		Overlay, matching the board name with the chip name, overwrites the final generated rootfs

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

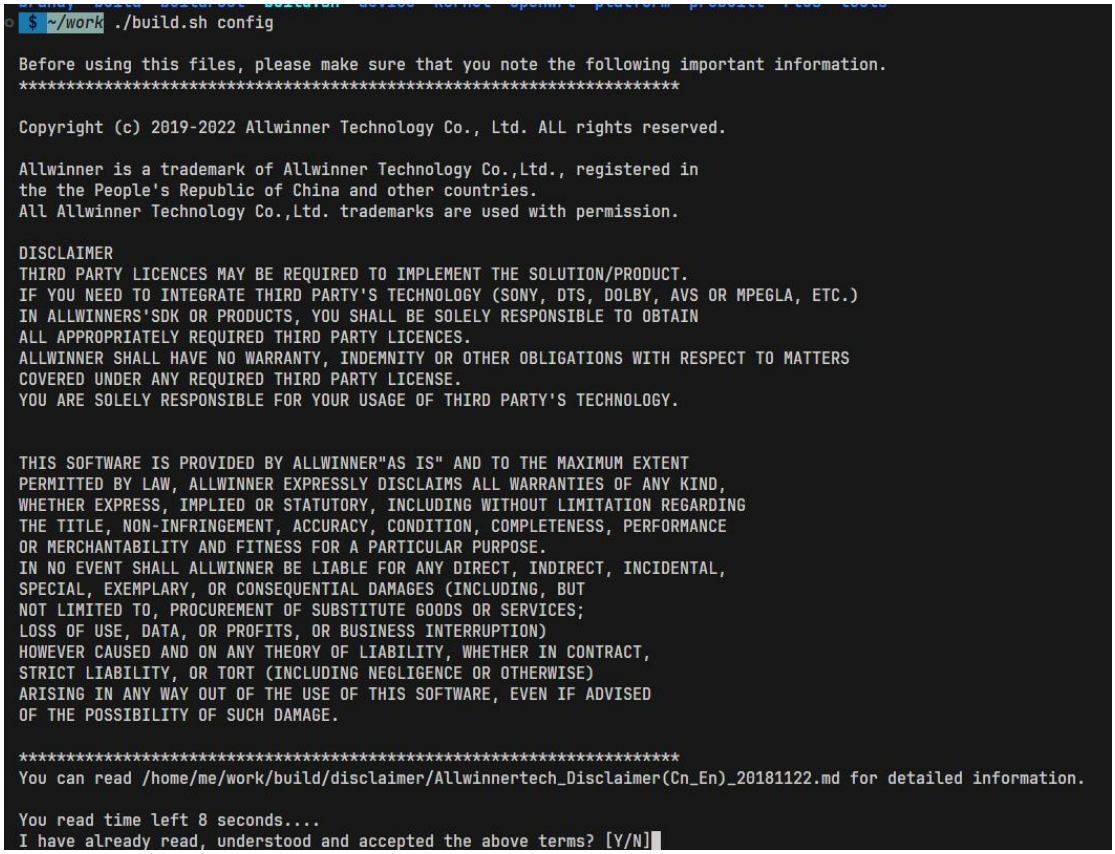
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3.3. SDK configuration

Since the SDK supports multiple chips and multiple solutions, users need to select th e configuration that matches their own development board before officially starting to com pile. The SDK also provides the following configuration methods:

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

After the execution is complete, you need to agree to some licensing certifications of Allwinner:



Press `Y` here and press Enter to confirm and proceed to the subsequent selection:

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
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]:
```

The compilation configuration used here is `device/config/chips/t527/configs/ebyte_linux_t527/buildroot/BoardConfig.mk`. After configuration is completed, the final compilation rule configuration can be viewed in the `.buildconfig` file in the SDK directory. The following lists the more important parts:

```
LICHEE_PLATFORM //Compile platform
LICHEE_LINUX_DEV //Compiled solution
LICHEE_IC // Compiled IC
LICHEE_BOARD //Compiled board configuration
LICHEE_FLASH //Compiled flash configuration
LICHEE_CHIP //Compiled chip code name
LICHEE_KERN_VER //Compiled kernel version
```

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
LICHEE_KERN_DEFCONF //Compile kernel default configuration
LICHEE_BUILDING_SYSTEM // Compiled construction system
LICHEE_BR_VER //buildroot version
LICHEE_BR_DEFCONF //buildroot default configuration
LICHEE_BR_RAMFS_CONF //buildroot's ramfs default configuration
LICHEE_BRANDY_VER //uboot version
LICHEE_BRANDY_DEFCONF //brandy's default configuration
LICHEE_CROSS_COMPILER //Cross-compilation chain used for compilation
LICHEE_CHIP_CONFIG_DIR //IC configuration directory of the compilation system
LICHEE_OUT_DIR //Compiled output directory
```

Note : For users using the nor-flash solution, the ` LICHEE_BRANDY_DEFCONF` configuration will be automatically expanded. For example, if it is configured as `sun55iw3p1_t527_defconfig` in BoardConfig.mk, the actual compilation will use the configuration file named `sun55iw3p1_t527_nor_defconfig`. If it does not exist, the compilation will fail.

3.4. Compile SDK

Wait for the SDK to finish configuration, then start compiling:

```
./build.sh
```

After the compilation is complete, SPL, U-Boot, Linux kernel and Buildroot file system image files are generated. After the compilation is complete, the following output will be displayed:

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
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. Compile separately

3.5.1. Compile u-boot separately

You can compile u-boot separately by using the following command, which can save compilation time when you only need to modify the u-boot stage:

```
./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. Compile the kernel separately

The kernel can be compiled separately using the following command, which can save compilation time when only the kernel stage needs to be modified. However, it should be noted that if a driver module is added after the modification, the rootfs also needs to be compiled at the same time, otherwise the corresponding driver module cannot be found:

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
./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
```

After compilation, the following log is output:

```
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. Compile buildroot separately

You can compile buildroot separately by running the following command, which can save compilation time when you only need to modify the buildroot stage:

```
./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'
```

After compilation, the following log is output:

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
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. Compilation output

After the compilation is complete, you will find an out directory in the SDK director y, which stores all the compiled output of the SDK.

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

The kernel directory is a soft link to the t527/kernel directory, which is the complete compiled output of the kernel. The toolchain directory is the cross-compilation tool chain used in the compilation process. We mainly look at the t527 directory:

```
$ ~/t527/tmp/out/t527 tree -L 2
.
├── ebyte_linux_t527
│   ├── buildroot
│   └── debian
└── kernel
    ├── build
    └── staging

6 directories, 0 files
$ ~/t527/tmp/out/t527
```

Go to the out/t527/ebyte_linux_t527/buildroot directory:

```
$ ~/t527/tmp/out/t527/ebyte_linux_t527/buildroot ls
all_modules  boot.img   dtbo.img  lib            rootfs.squashfs  System.map
arisc        buildroot  dtc       ramfs.cpio.gz  rootfs.ubifs     vmlinux
bImage       dist       dts_dep   rootfs.ext4    sunxi.dtb        vmlinux.tar.bz2
$ ~/t527/tmp/out/t527/ebyte_linux_t527/buildroot
```

We mainly need to focus on the following files:

- boot.img: kernel image, including kernel, device tree, Android image format
- sunxi.dtb: The compiled kernel device tree file will eventually be packaged into boot.img.
- vmlinux: The compiled kernel file will eventually be packaged into boot.img.

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

- rootfs. *: The root file system generated by buildroot. Usually the suffix is ext4.

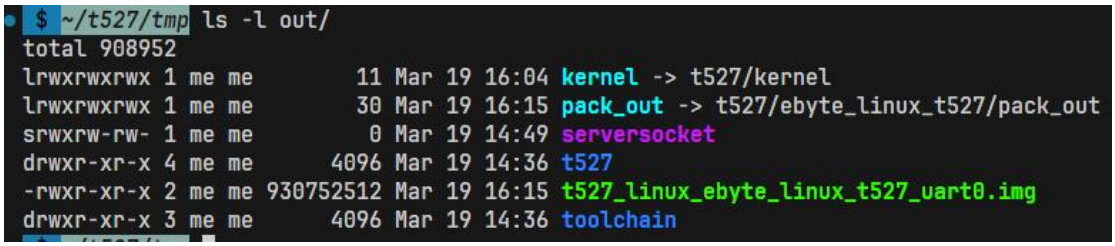
3.7. Image packaging

After the previous compilation, various compiled files are output, but most of them c annot be used directly and need to be packaged into a final usable image:

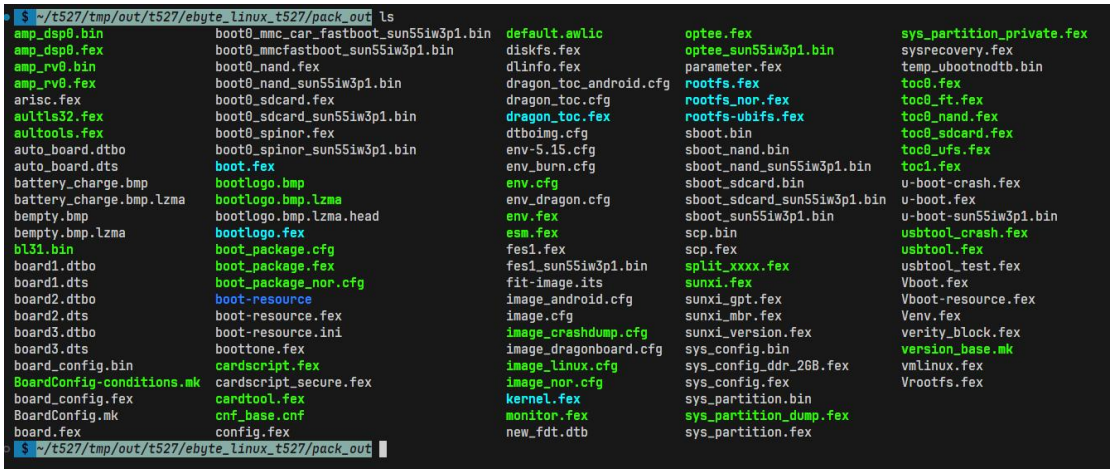
```
./build.sh pack
```

After the packaging is completed, check the out directory and you will find an additi onal pack_out directory and an img file. The img file is the image file that needs to be burned to the development board, and the pack_out directory is the intermediate file for p ackaging the image, some of which can be used in the end.

Let's look at the pack_out directory again:



You can find that it is a soft link from t527/<board>/pack_out:



There is a lot of content, so we mainly focus on the following files:

- boot_package.fex : The packaged u-boot image, including the contents of which c an be viewed in boot_package.cfg
- boot-resource.fex : Resources used during the startup process, such as logo image s.
- env.fex: u-boot environment variable configuration, copy of SDK/device/product/co

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

nfigs/<board>/[buildroot]/env.cfg.

- ❖ boot.fex: A soft link to the kernel image boot.img compiled above.
- ❖ rootfs*.fex: The soft link of the buildroot root file system compiled above.

3.7.1. Possible problems

The following situations may occur during the packaging process:

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
```

If this happens, it means that the user-configured partition size is not suitable for the final output file. You can find the partition corresponding to the filename in the figure in `SDK/device/product/configs/<board>/[buildroot]/sys_partition.fex, and then modify the partition size to adapt to the corresponding file. The minimum partition size is described in dl_file_size in the figure above:

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

4. Development board adaptation

In order to adapt to the user's new hardware platform, the user needs to have a certain understanding of the CPU chip manual and CPU chip user guide, and also need to have a certain understanding of the SDK modifications. From the description of the SDK directory above, we can know that the development board/solution configuration is stored in S

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

DK/device/product/configs/:



In this level of directories, except for the default directory, all other directories can be considered to be different development board/solution configurations. The contents in the default directory are used as default configuration items when the required items cannot be found in the corresponding board-level configuration. Generally, they do not need to be modified and should not be modified.

Still taking ebyte_linux_t527 as an example:

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
$ ~/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
```

The main files we need to pay attention to here are the following:

- BoardConfig.mk: Compile option configuration required by the SDK.
- board.dts : Soft link of linux-5.15/board.dts, used for the kernel's device tree file.
- buildroot: Detailed configuration used by buildroot to build the root file system.
- bsp_defconfig: kernel-related configuration.
- sys_config.fex : System parameter configuration. Old kernel versions play the role of device tree. However, in Linux 5.4 and later versions, only a few modules (such as DDR and serial port debug level) are retained. It is mostly used in the

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

SPL and u-boot stages.

- uboot-board.dts: device tree file in the u-boot stage.
- sys_partition.fex : Mirror partition configuration.
- env.cfg: environment variables in the u-boot stage.

We can find that some files with the same name as those in the upper directory also exist in the buildroot directory. This is an overwriting relationship, and the overwriting relationship is as follows:

1. bsp, buildroot , debian and other system solutions are parallel directories and will not overwrite each other.

2. The BoardConfig.mk, env.cfg, and sys_partition.fex files in bsp, buildroot , and other system solutions will overwrite the BoardConfig.mk, env.cfg, and sys_partition.fex files in the default directory.

3. In the same directory, BoardConfig_nor.mk and sys_partition_nor.fex will overwrite BoardConfig.mk and sys_partition.fex. Note that this only happens when the nor solution is selected in the SDK configuration.

4. If the BSP, Buildroot , or other system solutions do not have BoardConfig.mk, env.cfg, or sys_partition.fex configured, the BoardConfig.mk, env.cfg, and sys_partition.fex in the default directory will be used.

5. Note that BoardConfig.mk is included by the system solution's BoardConfig.mk. If the system solution and the upper layer have the same properties, they will be overwritten; otherwise, they will be considered added.

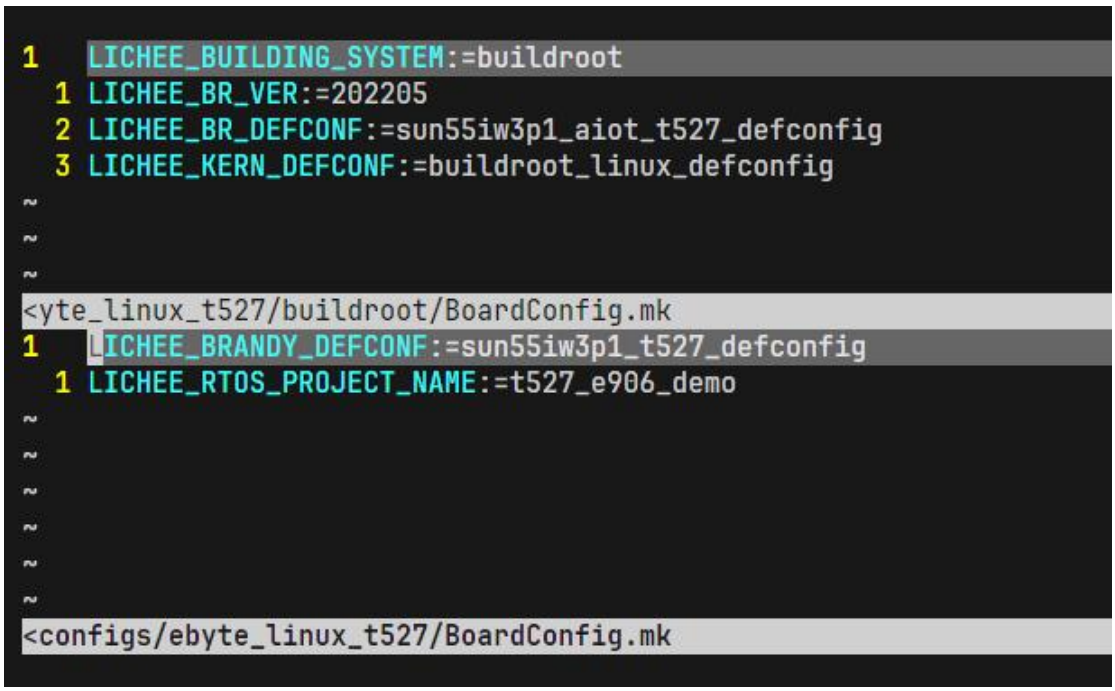
4.1. Solution Configuration

From the above information, we can understand that when the SDK is compiled, it will be initialized according to the configuration of the BoardConfig.mk file and decide which scheme to start compiling and packaging.

Taking the buildroot solution of ebyte_linux_t527 as an example, the content of the BoardConfig.mk file is as follows:

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



We mainly focus on the following:

- LICHEE_KERN_DEFCONF: The name of the kernel defconfig file, which is located in the linux-5.4 directory.
- LICHEE_BR_VER : The buildroot version, which will automatically search for the corresponding version in the SDK/buildroot/ directory.
- LICHEE_BR_DEFCONF : The default configuration of buildroot, corresponding to the path where the buildroot version source code is located.
- LICHEE_BRANDY_DEFCONF : This is the default configuration of u-boot, in SDK/ brandy / brandy-2.0/u-boot-2018 .

4.2. SPL

From the above information, we can understand that for the SPL stage, in most cases we only need to modify the sys_config.fex file.

This file is used for SPL and u-boot configuration starting from kernel version 5.4 or higher. SDK will make some configurations for the current development board based on this file configuration when compiling/packaging. The main parts we need to modify are as follows:

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.2.1. Target board platform related information

```
;-------------------------------------------------------------------------------
; debug_mode=0 means turning off the printing log of Boot0/SPL during startup, the default is 8.
; 0: Turn off all printing
; 1: Only print key nodes
; 4 : Print error message
; 8: Print all log information
;-------------------------------------------------------------------------------
; earseflag indicates whether to perform programming or erasing operation
; 0x0: Do not erase.
; 0x01: Erase, private, secure storage, user data, mbr partitions are retained, and others are erased.
; 0x11: Erase, private and secure storage partitions are retained, and others are erased.
; 0x12: Force erase (entire flash).
;-------------------------------------------------------------------------------
[platform]
eraseflag = 1
debug_mode = 1
```

4.2.2. Target board storage device information

```
;-------------------------------------------------------------------------------
;storage_type = boot medium, 0-nand, 1-card0, 2-card2, -1( default )auto scan
;-------------------------------------------------------------------------------
[target]
storage_type = -1; It is recommended to use automatic identification or configure according to the actual storage device used
```

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.2.3. Target board debugging serial port information

```
;-----------------------------------------------------------------------------
; Describe the form of gpio: Port: port + group number <function allocation><internal resistance status><drive capability><output level status>
;-----------------------------------------------------------------------------
[uart_para]
uart_debug_port = 0
uart_debug_tx = port:PB09<2><1><default><default>
uart_debug_rx = port:PB10<2><1><default><default>
```

The user should determine which serial port to use as the debug port based on their own board configuration. We recommend using serial port 0. Otherwise, you will need to modify the relevant content in the root file system created by buildroot. However, there is no specific requirement for which GPIOs serial port 0 actually uses. You can refer to the "T527_Datasheet_V0.91.pdf" document to find out which pins can be reused as serial port 0.

4.2.4. Other requirements

By default, it is not recommended to modify SPL. If you really need to perform some special modifications in the SPL stage, please refer to the "Linux_SPL-PUB_Development Guide.pdf" document.

4.3. u-boot

4.3.1. Device Tree

uboot-board.dts file is the device tree file used in the early stage of uboot and is only used in the initial stage. When u-boot executes the sunxi_replace_fdt_v2 function, it will start to use the device tree information of the kernel stage. Therefore, only a few situations need to be configured in this file.

4.3.2. Environment variables

The env.cfg file is the environment variable file for the u-boot stage. Users can confi

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

gure it by themselves, but there are several special features:

4.3.2.1. bootcmd variables

```
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
    }
```

The above code exists in the SDK/brandy/brandy-2.0/u-boot-2018/board/sunxi/board_helper.c file. It means that the value of the bootcmd variable will be automatically switched according to the actual boot memory device. The switching rule is to change "setargs_nand" to the corresponding environment, so users can keep using the following configuration and let u-boot automatically switch.

```
bootcmd=run setargs_nand boot_normal
```

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.3.2.2. mmc_root variable

```
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);
```

The above code exists in the SDK/brandy/brandy-2.0/u-boot-2018/board/sunxi/board_helper.c file, which means that the value of the xxx_root variable will be automatically switched according to the actual boot memory device. The switching rule is to change "xxx_root" to the root_part_name value. Under normal circumstances, it should be automatically switched by u-boot.

```
[    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
```

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

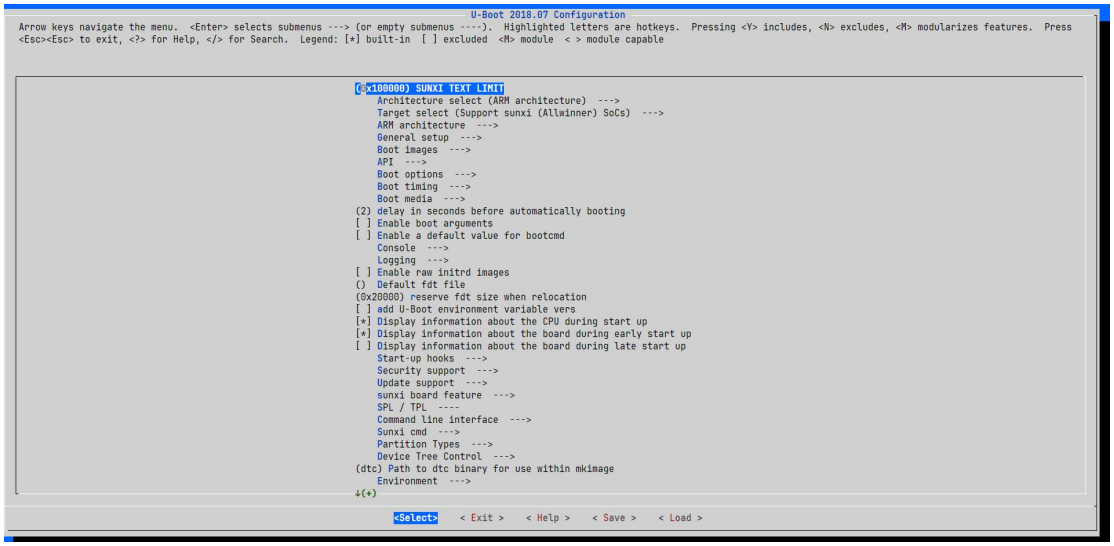
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.3.3. defconfig

The defconfig configuration file at the u-boot stage is determined by the value of the LICHEE_BRANDY_DE FCONF item in the BoardConfig.mk file. In the evb1_auto buildroot solution, this file is brandy/brandy-2.0/u-boot-2018/configs/sun55iw3p1_t527_defconfig. Modification is generally not recommended. If the user does need to modify it, please follow the steps below to enter the menuconfig interface:

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

make menuconfig
```



It should be noted that if defconfig is reconfigured, the configuration just made through menuconfig will be overwritten.

4.4. kernel

The kernel-related configuration options are stored in the <SDK>/device/product/configs/<board>/linux-5.15 directory, which mainly consists of two files: board.dts and bsp_defconfig. During the compilation process, board.dts will be soft-linked to SDK/device/product/configs/<board>/board.dts, and finally SDK/out/kernel/staging/sunxi.dtb will be generated.

4.4.1. Device Tree

From the above, we know that the current board-level device tree configuration file is board.dts. For the board.dts file, most nodes have corresponding comments and explanations for users' reference, but some places are more special:

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

1. spi0 node
2. sdc0 node
3. sdc2 node

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
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:
```

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

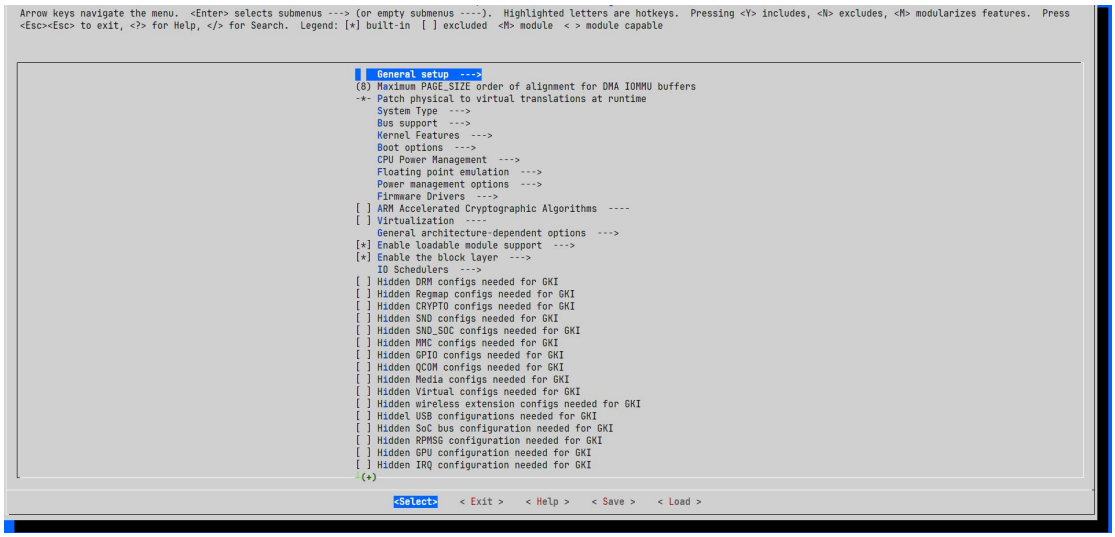
The above code exists in the SDK/brandy/brandy-2.0/u-boot-2018/board/sunxi/board_helper.c file, which means that u-boot will automatically enable/disable some nodes in the kernel device tree according to the corresponding storage device information at startup. Therefore, users may find that even if the device node of the corresponding storage device is configured as `status = "disabled"`, they can still enter the system normally.

4.4.2. defconfig

As can be seen from the above, the board-level defconfig file is determined by the LICHEE_KERN_DEFCONF value in the BoardConfig.mk file. The configuration file must exist in the <SDK>/device/product/configs/<board>/linux-5.15 directory.

Here we take ebyte_linux_t527 as an example. Its kernel defconfig file is <SDK>/device/product/configs/ebyte_linux_t527/linux-5.15/bsp_defconfig. If the user needs to modify the configuration, he can use the following method to enter the menuconfig interface in the SDK top-level directory to configure it:

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



After the modification is completed, the user can update and save the new configuration to the bsp_defconfig file using the following command:

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

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.5. File System

4.5.1. Buildroot

As mentioned above, board-level configuration for buildroot is stored in the <SDK>/device/product/configs/<board>/[buildroot]/BoardConfig.mk file, primarily LICHEE_BR_VER and LICHEE_BR_DEFCONF . Here, we use the ebyte_linux_t527 as an example, using buildroot version 202205. Its source code is located in SDK/buildroot/buildroot-202205, and its compilation configuration file is located in configs/sun55iw3p1_aiot_t527_defconfig in the buildroot source directory.

For more information about buildroot, please refer to the official buildroot manual: https://buildroot.org/downloads/manual/manual.html .

4.5.2. Unable to log in

For users using buildroot version 202205, if you are unable to log in or find that you need to enter the same content twice in a row before it is displayed in the terminal, for example, if you enter `aa`, the terminal will display `a`. Please modify the following file: `<SDK>/buildroot/config/buildroot/buildroot/post_build.sh` and add the following content to it:

```
+ # 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

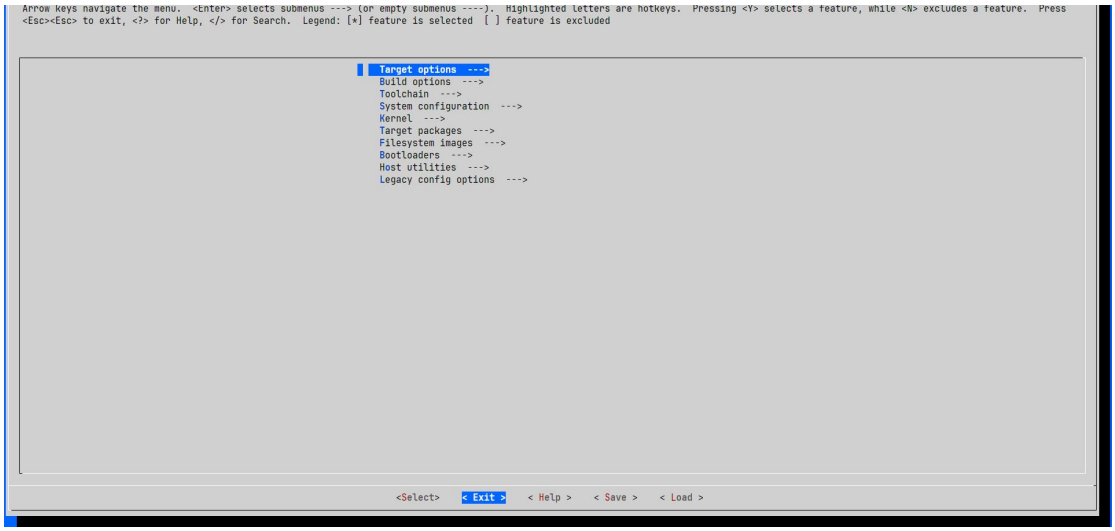
For most users, in addition to adapting the kernel to the baseboard, they also need to modify the root file system to adapt to their own needs. It is recommended to use menu

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

config to modify the configuration to further adapt to their own solution requirements. Yo u can enter the menuconfig interface of buildroot with the following command:

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



After the modification is completed, the user can update and save the new configurati on to the sun55iw3p1_aiot_t527_defconfig file using the following command:

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

4.5.4. Partition

sys_partition.fex file describes the partitioning of the packaged image. The main parts that users need to modify are the rootfs partition size and the root file system that shoul d be placed in the partition. It must not be smaller than the actual root file system size, nor larger than the physically available size of the storage device.

For eMMC or SD card storage devices, the rootfs partition should look like the follo wing:

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

4.6. RISC-V

For RISC-V porting, customers can refer to the FreeRTOS_RTOS_System_Developmen

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

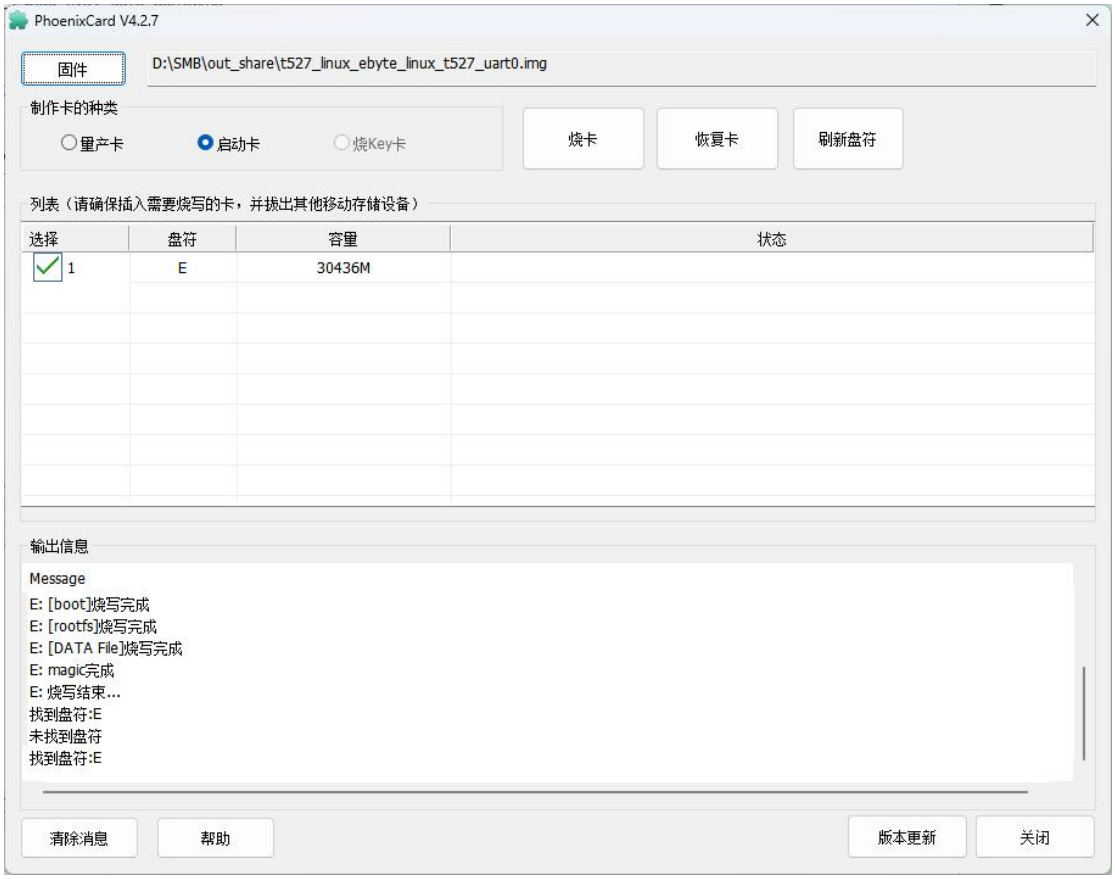
t Guide.pdf, which will not be described in detail here.

5. How to burn and update the system image

This chapter mainly introduces the burning and updating method based on the ECK32-T527 core board .

5.1. Make an SD card boot card

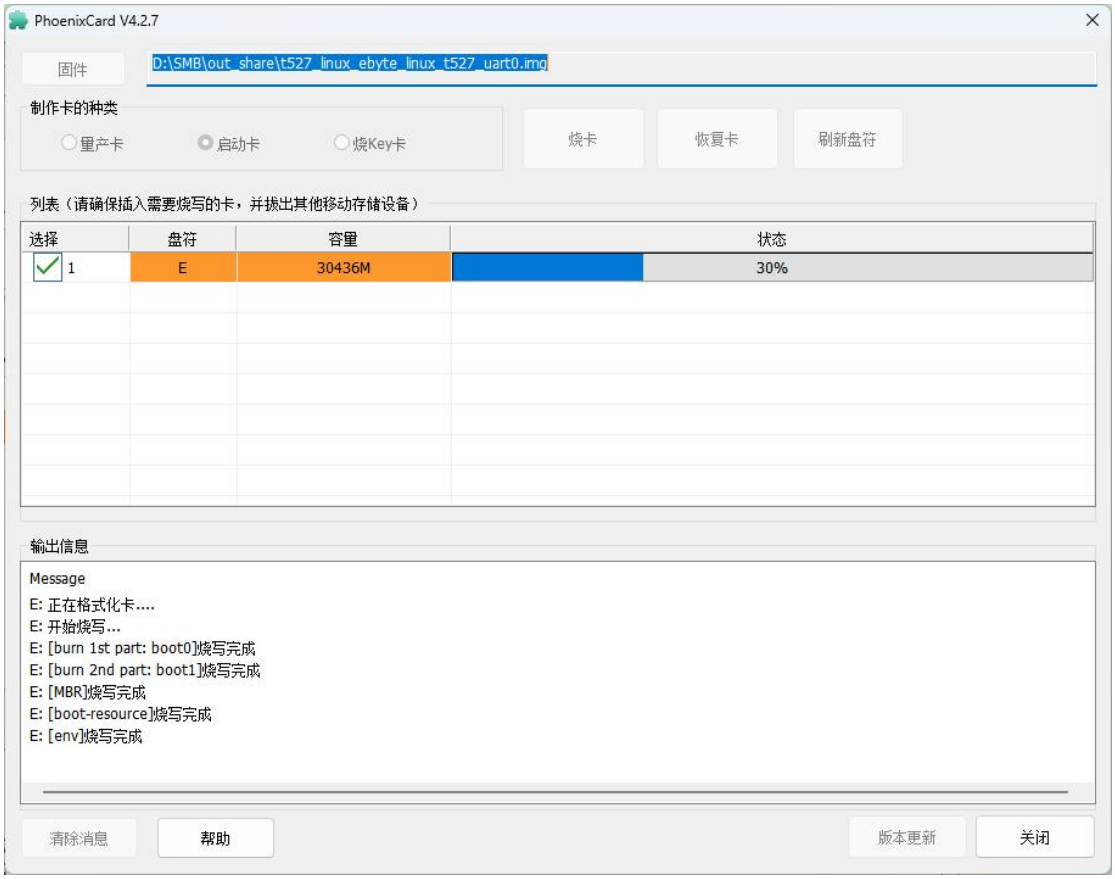
First open the PhoenixCard program, the interface is as follows:



The user should select the firmware to be burned, select the card type as "Startup Card", and then select the SD card device to be burned. The software will automatically scan the available mobile storage devices. Please be careful not to select the wrong device. Finally, click the "Burn Card" button. The current progress will be displayed in real time during the burning process:

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

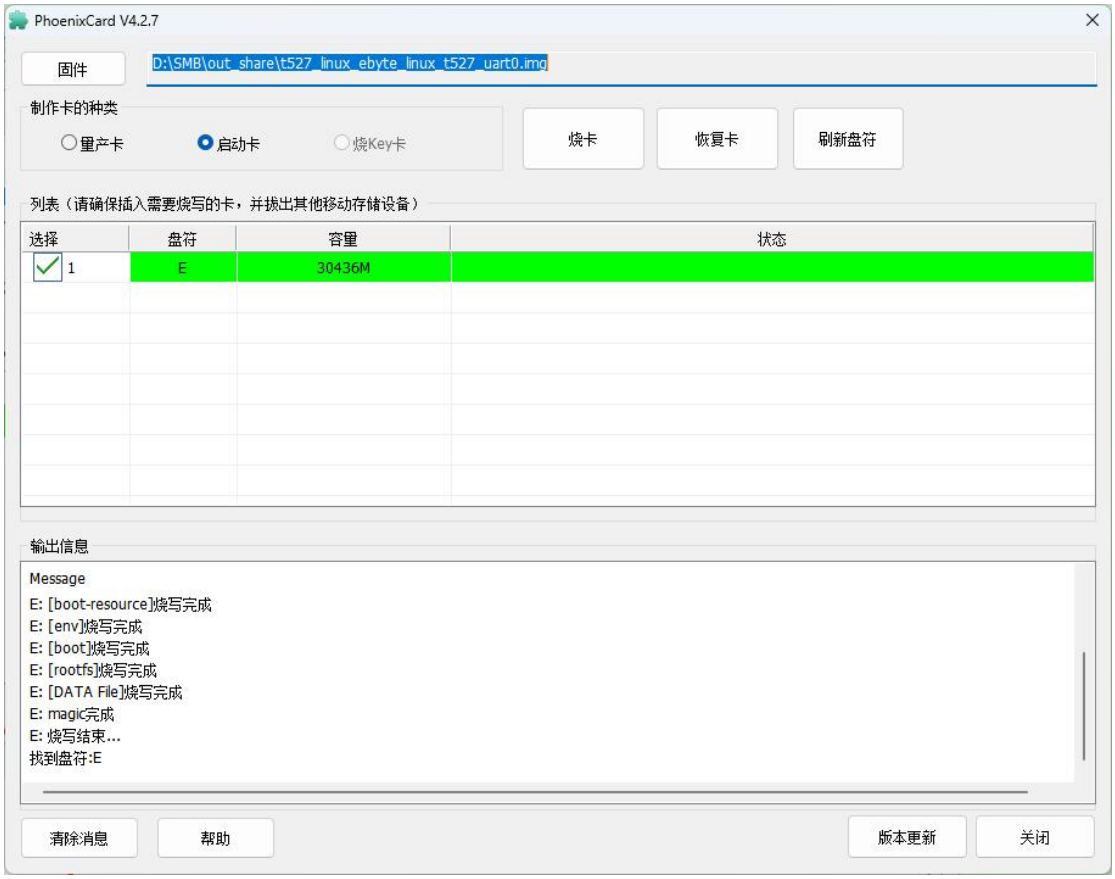
Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



After the flashing is completed, the interface is as follows. You can check whether the startup card is successfully created through the output information at the bottom of the software interface:

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



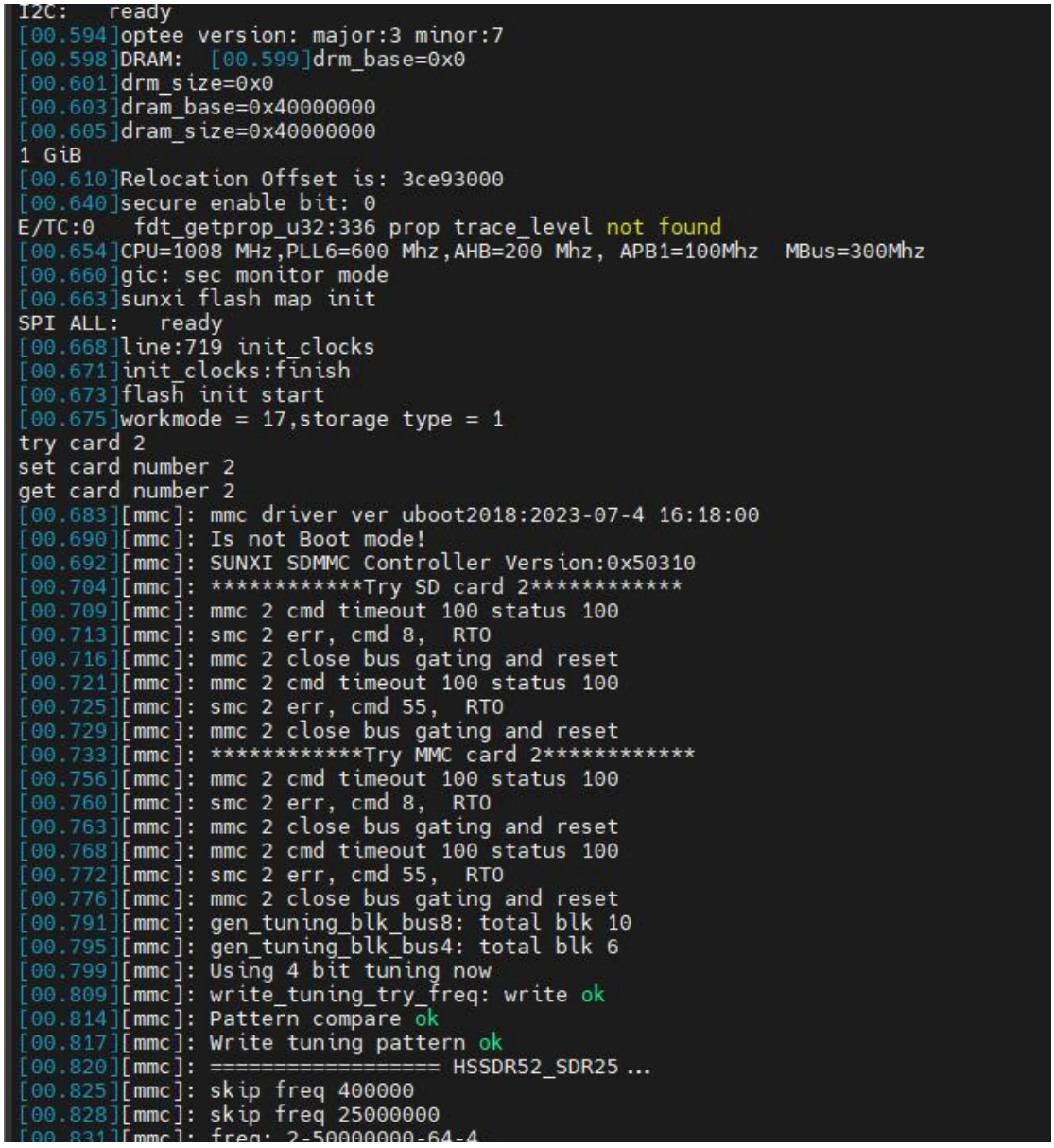
5.2. Mass production card burning

When making an SD card boot card, we will also find that there is a type of mass-p roduced card. This method can be used to perform pipeline programming by making an S D card. Users do not need to use a separate image firmware, and can directly use the im age that can be used correctly in the boot card stage.

After the production is completed, start the system from the mass production card an d connect the debug serial port to view all the printed information during the burning pro cess:

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



After the system boots into the u-boot stage, the system begins to be officially burned.

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

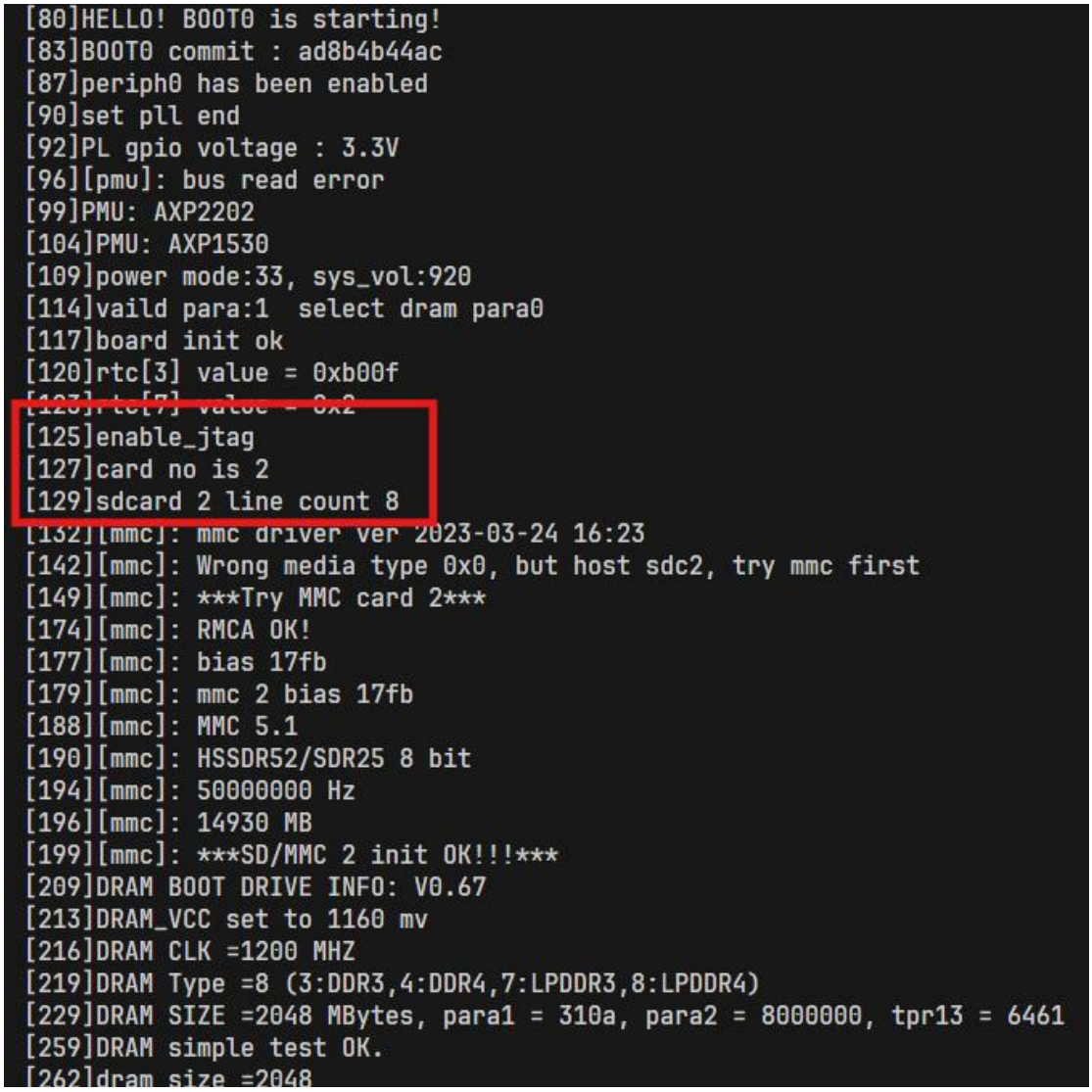
Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
[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
```

If a message similar to the one above appears, the user can disconnect the power supply of the development board, remove the mass production card, and then power it back on. The serial terminal will print the following startup information to confirm that it is booting from the eMMC:

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



5.3. Official tool burning

This section describes how to use the official Allwinner tool PhoenixSuit to implemen t USB fast burning.

5.3.1. Enter burning mode

Before using, we need to understand how to put the device into programming mode. According to the official documentation, there are four ways:

1. Press and hold the FEL button while powering on
2. Connect the serial port when booting, press and hold the number key "2"
3. Enter the u-boot console and enter: "efex"
4. Enter the system and enter in the system console: "reboot efex"

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

1. The commonly used method is to press and hold the FEL button while powering on the computer. This requires the user to pull out the corresponding pins and connect th em to a button when making the baseboard. This method does not require a working syst em on the storage device. Regarding the second method, please note that this function is i mplemented in the official boot firmware provided by Allwinner . If the user configures t he wrong serial port description in sys_config.fex, the second method will fail. However, t here is no need to worry, as you can re-burn the correct firmware by making a mass-pro duced card.

5.3.2. Burning tool configuration

Let's look at the flashing tool. After installing and opening the tool, the interface is a s follows:



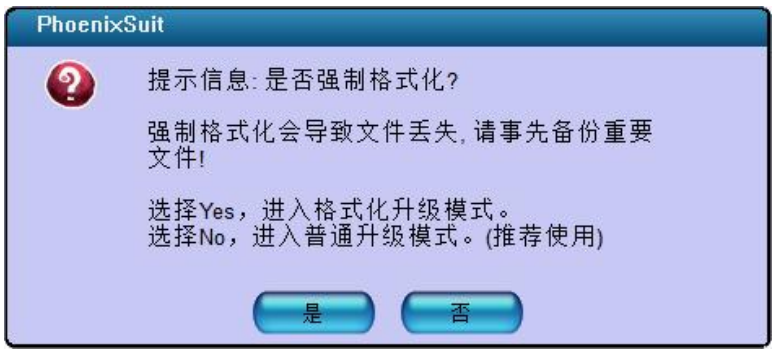
Click the "One-click flash" item above to enter the image selection.

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Users do not need to worry about the upgrade button. Just open the software, select t he image to be burned, and connect the USB cable normally. Once the device to be burn ed enters the burning mode, it will automatically identify and ask about the burning statu s:

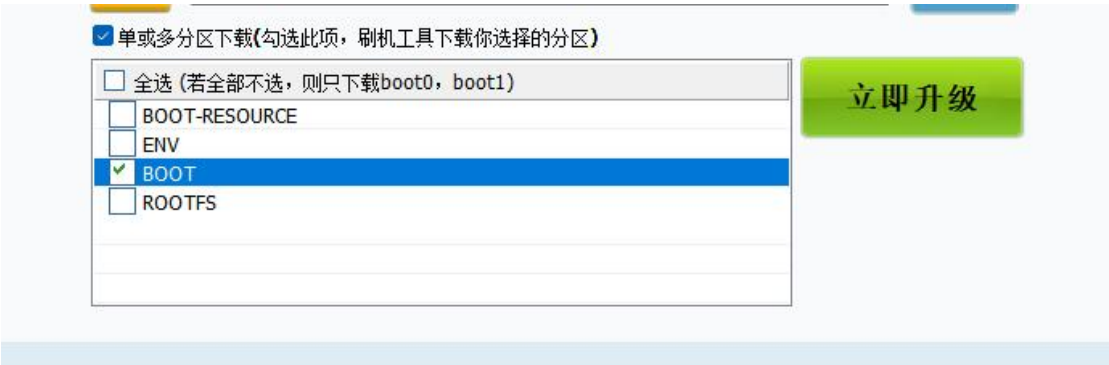


Users can choose whether to format and upgrade as needed.

At the same time, users can also burn some content separately through the following configuration:

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



For example, here it means that only boot0, uboot, and kernel are burned, and the file system is not updated.

5.4. Possible problems

5.4.1. Cannot find storage device when making startup card

When using this tool to burn a startup card, you may find that the tool cannot find the corresponding storage device after inserting the SD card. First, determine whether the SD card is occupied by VMware. If so, release the SD card and return its usage rights to the host. Then, enter the Windows system's Disk Management tool to check whether the device exists. If so, repartition the device so that the system can recognize the new drive letter. If there is no drive letter, it will not be found. If it still cannot be used, try changing the USB port or check whether it is occupied by another program.

5.4.2. nor-flash configuration

Our core board does not provide SPI-NOR devices. If the user provides them in the baseboard, you can refer to `device/config/chips/t527/configs/pi_ebyte_t527/bsp/` to implement it. In addition, you need to modify the following files according to the materials used:

1. brandy/brandy-2.0/u-boot-2018 /drivers/mtd/spi/spi-nor-ids.c
2. bsp/drivers/mtd/spi-nor-5.15/<material manufacturer>.c

Note: Using the NOR solution does not support display and NPU functions.

5.4.3. PK group pins cannot be controlled as GPIO

This is due to a function conflict between the PK group and the VIN. You can solve this by adding the following configuration to the device tree:

```
&pd1 {
```

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
/* pk */
pd1_vi@A523_PCK_VI {
ppu-always-on;
};

/* pd */
pd1_vo0@A523_PCK_VO0 {
ppu-always-on;
};

/* pj */
pd1_vo1@A523_PCK_VO1 {
ppu-always-on;
};
};
```

6. References

- T527_Datasheet_V0.91.pdf
- T527_User_Manual_V0.92.pdf
- Allwinner Development Community: https://bbs.aw-ol.com/
- Linux kernel open source community: https://www.kernel.org/
- Debian wiki: https://wiki.debian.org/
- Buildroot-manual: https://buildroot.org/downloads/manual/manual.html

7. Revision Notes

Revision Notes Table

Version	Modifications	Modification time	prepared by	Proofreading	Approval
V1.0	First Draft	2025-3-19	WQ	WFX	WFX
V1.1	Updated pi board related description	2025-7-30	WQ	WFX	WFX

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

8. About Us

Technical support: service@cdebyte.com

Documents and RF Setting download link: www.cdebyte.com

Thank you for using Ebyte products! Please contact us with any questions or suggestions: info@cdebyte.com

---

Web: www.cdebyte.com

Address: B5 Mould Park, 199# Xiqu Ave, High-tech District, Sichuan, China

Chengdu Ebyte Electronic Technology Co.,Ltd.

Copyright©2012-2025, Chengdu Ebyte Electronic Technology Co., Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000