Software Development Guide
ECK32-T527 Core Board



Chengdu Ebyte Electronic Technology Co.,Ltd.

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

Contents

Disclaimer and Copyright Notice 1
1. Introduction 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 build 9
3.1. SDK Acquisition 10
3.2. 3.2 SDK Catalog 11
3.3. SDK Configuration 12
3.4. Compiling the SDK 15
3.5. Separate compilation 16
3.6. Compilation output 17
3.7. Image Packaging 18
4. Development Board Adaptation 20
4.1. Program Configuration 23
4.2. SPL 24
4.3. u-boot 26
4.4. Kernel 29
4.5. Rootfs 32
4.6. RISC-V 34
5. How to burn an updated system image 34
5.1. Creating an SD card starter card 35
5.2. Production card burning 37
5.3. Burning with official tools 40
5.4. Problems that may be encountered 43
6. References 43
7. Revision history 43
8. About us 44

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. The document is provided "as is" without any warranty, including any warranty of merchantability, fitness for a particular purpose or non-infringement, and any warranty otherwise provided by any proposal, specification or sample. No liability is assumed for this document, including liability for infringement of any patent rights arising from the use of the information in this document. No license to any intellectual property, whether express or implied, by estoppel or otherwise, 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 property of their respective owners and are hereby acknowledged.

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

Notice:

The contents of this manual may be changed due to product version upgrades or other reasons. Yibite Electronic Technology Co., Ltd. reserves the right to modify the contents of this manual without any notice or reminder. This manual is only used as a guide. Chengdu Yibite Electronic Technology Co., Ltd. tries its best to provide accurate information in this manual, but Chengdu Yibite Electronic Technology Co., Ltd. does not ensure that the contents of the manual are completely error-free, and all statements, 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. Introduction

This article mainly introduces the complete process of customizing a complete embedded Linux system based on the EBYTE core board, including the preparation of the development environment, the acquisition of code, and how to port Bootloader, Kernel, customize the Rootsfs suitable for their own application needs, etc. We first introduce how to build a system image for the ECK32-T527 core board based on the SDK source code we provided. We firstly introduce how to build a system image for ECK32-T527 core board based on the SDK source code provided by us, and how to burn the built image to the development board. For those users who are developing projects based on ECK32-T527 core board, we focus on the methods and some key points of porting this set of system to user's hardware platform, and through some actual porting cases and customization cases of Rootfs based on Buildroot, so that users can quickly customize the system image suitable for their own hardware.

Please note that this document does not contain the introduction of buildroot project and Linux system related basics, it is suitable for embedded Linux system developers and embedded Linux BSP developers who have some development experience.

2. Development Environment Preparation

This chapter mainly introduces the ECK32-T527 core board based on the use of the development of some of the hardware and software environments needed before, including the necessary hardware configuration, development, burn-in the required environment is introduced.

By reading this chapter, you will understand the installation and use of related hardware tools, software development debugging tools. And you will be able to quickly set up the related development environment to prepare for the later development and debugging.

Host: i7-12700 + win11 22H2

Virtual Machine: Debian bullseye

2.1. Hardware preparation

2.1.1. Debugging the serial port

Debugging serial port: baud rate 115200, data bit 8, stop bit 1, no parity, no hardware flow control, the actual serial port, please configure your own configuration, see later for details of the

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

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

configuration.

2.1.2. Activation method

For users using our ECK32-T527 core board all that needs to be remembered is that once a bootable SD card is inserted, the system will boot from the SD card, and if the SD card fails to boot, it will boot from the on-board storage device.

2.2. Virtual machine software deployment

This chapter describes how to set up the T527 development environment. By reading this cha pter, you will understand the installation and use of related development and debugging tools. And you will be able to quickly set up the related development environment to prepare for the later development and debugging.

2.2.1. Virtual Machine Configuration

Currently more common virtual machine software VirtualBox, VMware, etc., the configuration of different platforms are different, the user habits are also different, so please configure the virt ual machine according to the user's own virtual machine software virtual machine to use bridged networking, and open the directory sharing function with windows.

2.2.2. Configuring the compilation environment

Since T527 will use buildroot 202205 in the SDK, it is recommended to use Ubuntu 2204 6 4bit distribution, or you can try to use a higher version of the distribution, but still please make sure the version is compatible with the buildroot version.

Install the software:

```
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
```

Create a working catalog:

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

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

```
mkdir ~/work
```

2.2.3. Turn on the TFTP service for the virtual machine

Later debugging phase may use TFTP to get files from the network, so you have to install and turn on the TFTP service in the virtual machine first, use the following command to install the TFTP service:

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

By default, the files under /srv/tftp directory will be shared and can only be downloaded, not uploaded. If you want to personalize the configuration, please refer to the official document: https://help.ubuntu.com/community/TFTP, the path used in this document is /srv/tftp.

Add users to the tftp group and change the permissions of the shared directory:

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

If you do not perform the above operation, there may be a permission problem, and you can not write files to the shared directory with non-root user privileges.

2.2.4. Enabling NFS services for virtual machines

Later debugging phase often need to use NFS to share files, so you have to install and turn on the NFS service in the virtual machine first, 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 that needs to be exported, here directly use the tftp default directory, the user can modify their own, you can also use a handy text editor to modify the /ext/exports file, the configuration method refer to man manual:

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

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

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

2.2.5. Enabling SSH service for virtual machines

After turning on the SSH service of the virtual machine, we can use the ssh client software to login to the virtual machine under Windwos, such as using MobaXterm, VSCode, and use the following command to turn on the SSH service under the virtual machine:

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

The above command installs the ssh service, the configuration file for ssh is /etc/ssh/sshd_config, just use the default configuration, for more configurations refer to the sshd_config related man pages.

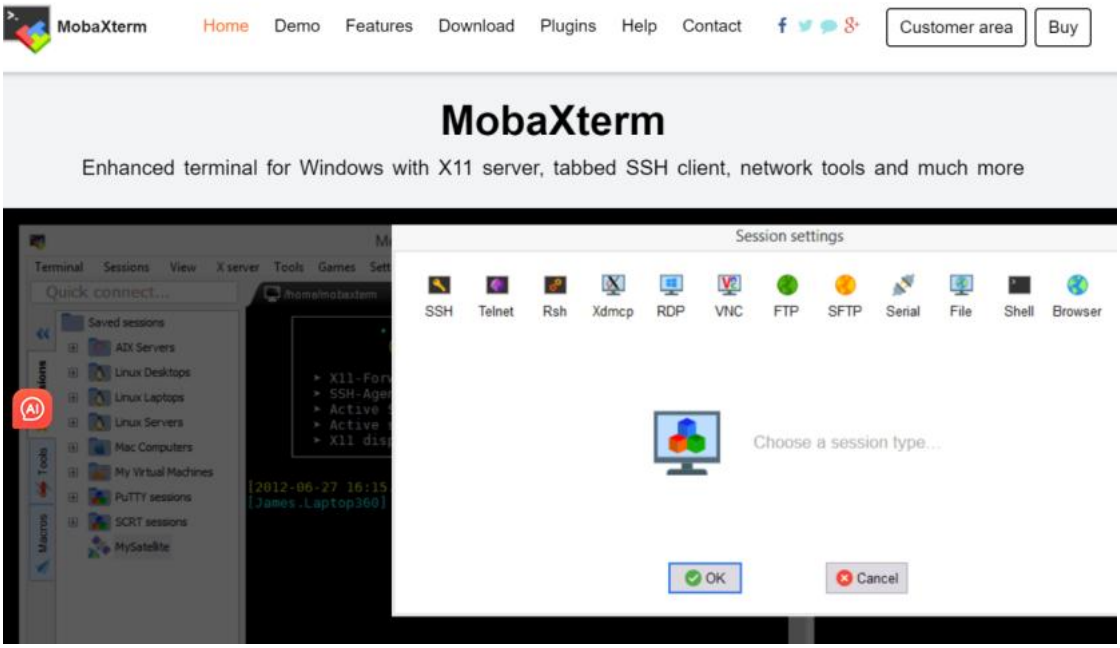
2.3. Host Software Installation

This chapter describes how to set up the burn-in debugging environment for the T527. By reading this chapter, you will understand the installation and use of related burn-in debugging tools.

2.3.1. MobaXterm

2.3.1.1. Software download and installation

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

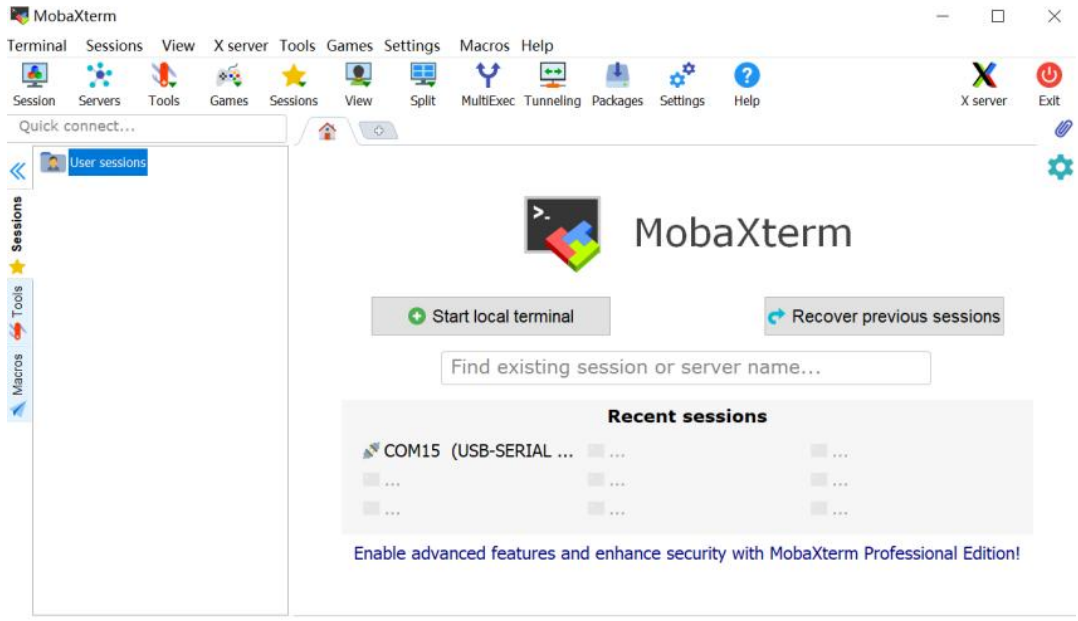


Copyright©2012–2025, Chengdu Ebyte Electronic Technology Co.,Ltd.

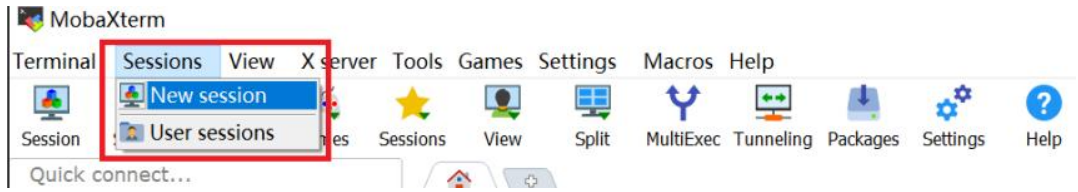
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2.3.1.2. Software use

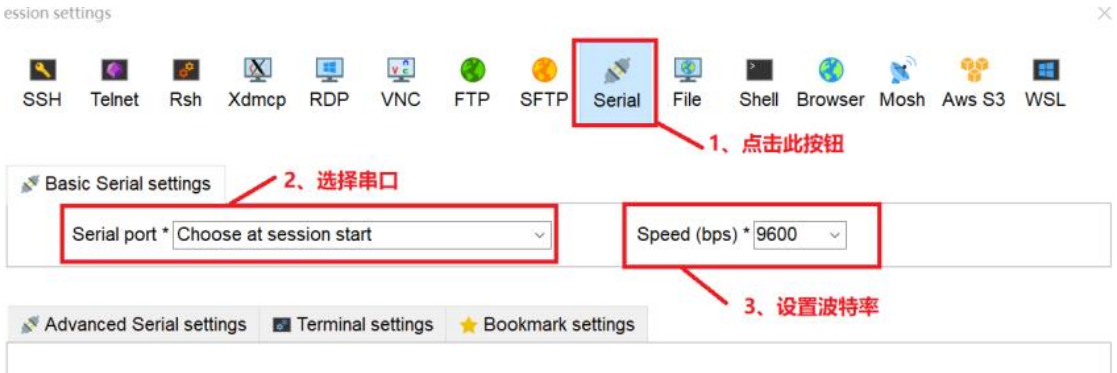
The software interface is shown in the figure:



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 purpose of u sing MobaXterm is to use it as a serial terminal. Click the "Serial" button in the picture to open the serial port setting interface, as shown in the picture:



After opening the serial port settings window, first select the serial port number to be set, so you have to use the serial port cable to connect the development board to the computer, because the default serial port baud rate used by the T527 is 115200, you need to set the baud rate to 1

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

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

15200.

2.3.2. PhoenixSuit

This tool is the official tool provided by Allwinner to burn the system from USB, the software has been stored in the directory: “04_Tools/PhoenixSuit”. For details on how to use it, see the section “Burning with Official Tools” later in this article.

2.3.3. PhoenixCard

This is the official tool provided by Allwinner for making system cards, you can choose to burn and make mass production cards and boot cards, the software has been stored in the directory: “04_Tools/PhoenixCard”. For details on how to use it, please refer to the section “Making SD card boot card” later.

2.4. Data transmission

In the previous chapters we have configured various network services in the virtual machine section, now we just need to understand how to use them on the client side with the development board.

2.4.1. TFTP client

In general, the development board only need to use the download function can be, at this time you can simply use the curl tool to download, use before you need to buildroot in the optional curl function, and then use curl to download the file (only the Linux version of curl, windows can choose to use filezilla):

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

1. /path/dist_file: where to download to, output file name name and path.

2. HOST: the ip address of the virtual machine, you need to pay attention to whether this address can be accessed.

3. src_file: directory and name of the file, note that there is no need to add the "TFTP_DIRECTORY" part of the configuration, for example, if the file is stored in `/srv/tftp/aaa/bbb/ccc`, the value of the src_file will be "aaa/bbb/ccc".

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

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

2.4.2. NFS client

Normally, development boards can mount NFS filesystems by simply using the mount command, 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 whether this address can be accessed.

2. path: the path in the NFS server you want to mount, such as /srv/tftp/aaa/bbb.

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

2.4.3. SSH

Generally speaking, we can also use SSH comes with SFTP and SCP functions for file downloads, but not many direct use of the command to upload and download, more use of third-party tools to carry out, such as the use of MobaXterm login ssh after the SFTP function, or VSCode use remote-SSH plug-in to realize the development process of the download function, the use of the way are Right-click on the file you want to download and select Download.

2.4.4. USB flash drive/SD card

For SD cards, a card reader is required, and the actual usage is the same as a USB flash drive.

2.4.4.1. Copying data from a virtual machine

Users need to use their own virtual machine software to pass the USB flash drive / SD card to the virtual machine, in general, it will be automatically mounted, in the file manager can be directly accessed, after copying the data in the file manager to uninstall the device, and then remove the device from the virtual machine can be.

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

1.Method I

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

2. Method II

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

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

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

This situation requires a little knowledge of the U disk / SD card partition, in order to specify the use of sdXY in the Y value, X value is generally 'b', but also need to judge the situation by themselves through the return information of lsblk.

Use first way to mount, without root access, do not need to specify the path, but you need to have installed the system `udisks2` program package.

To uninstall it manually, you can use the following command:

1.Method I

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

2. Method II

```
sudo umount /mnt_point
```

2.4.4.2. Copying data to the board

For development boards, they are generally streamlined and do not necessarily have the udisks2 package or a GUI interface, and if they do have an interface, they do not necessarily support automounting, but often users have root privileges, so they generally perform mount operations from the command line.

1. Mounting

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

2. Copy

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

3. Unmounting

```
umount /mnt_point
```

3. SDK compilation build

This section describes the complete process from acquisition of the SDK to compilation to ge

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

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

nerate a usable image.

3.1. SDK Acquisition

The development SDK for the T527 is included in the download file, so please extract it to the development directory by yourself.

Because the complete SDK is too large, in order to reduce the size of SDK, the SDK provided by our company is compressed by repo, and the hidden directory `.repo` is generated in the current directory after extraction.

To decompress the SDK, you need to use the repo tool, download the repo tool:

```
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 mirrors (domestic optional):

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

Enable the relevant environment variables:

```
source ~/.profile
```

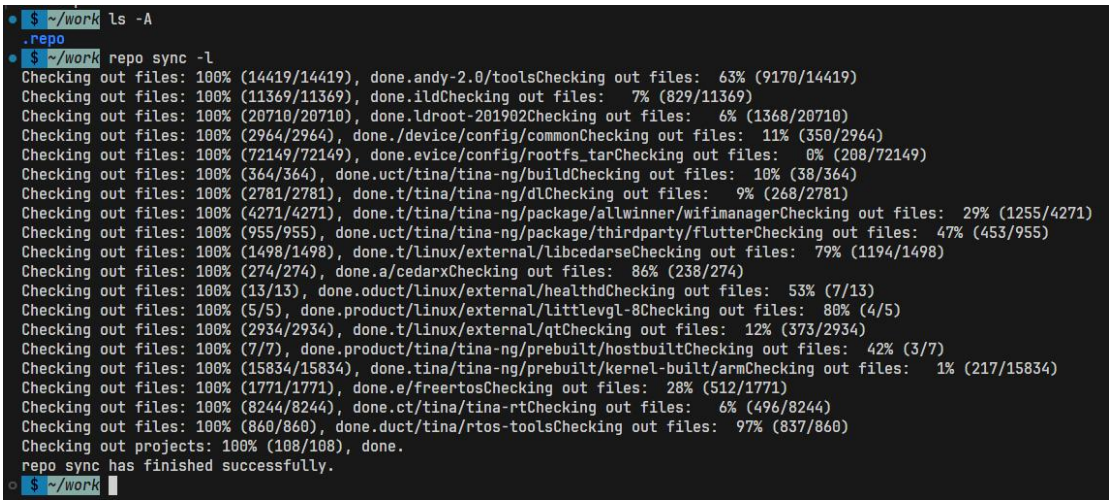
After entering the project directory, execute the following command to synchronize the code repository locally:

```
repo sync -l
```

View the catalog as follows:

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

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



3.1.1. Problems likely to be encountered

Since the repo tool is a python script, the user's python version may not be compatible with the repo dependency version in the SDK, for users whose default python is python2, or you need to configure the default python to be python3, if you still get an error after setting it up, please go to `SDK/.repo/repo` directory and Execute `git pull` to pull the updated repo version.

For some users who are not using debian/ubuntu, the above `source ~/.profile` may not add the `~/.local/bin/` directory to the environment variable, please add it as below:

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

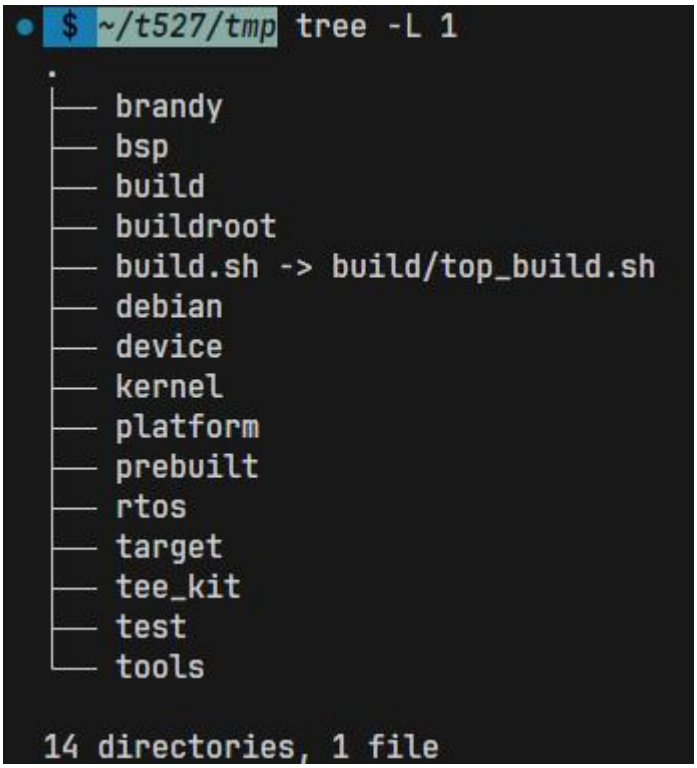
source ~/.bashrc
```

3.2. 3.2 SDK Catalog

After unpacking, the entire SDK directory is shown below:

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

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



We need to focus on these catalogs below:

Top-level directory	Subdirectory	Role
brandy	brandy-2.0/spl-pub/	SPL source code, generally not modified, the official Allwinner provided also most of the binary files.
	brandy-2.0/u-boot-2018/	U-boot source code
build		The compilation script directory, which is not normally modified.
buildroot	buildroot-202205	file system source code
	package/auto/sdk_demo/	Sample program
debian	compressed_files/	Debian sample image
device	config/chips/	A catalog of chip configurations supported by the SDK, including configuration options for several development boards.
	product	None by default, pointing to the target chip after executing the SDK configuration, in this case the T527.
kernel	linux-5.15	kernel source code
bsp	configs/linux-5.15/	Stores the 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.

3.3. SDK Configuration

Since the SDK supports a variety of chips, and also supports a variety of programs, so befor

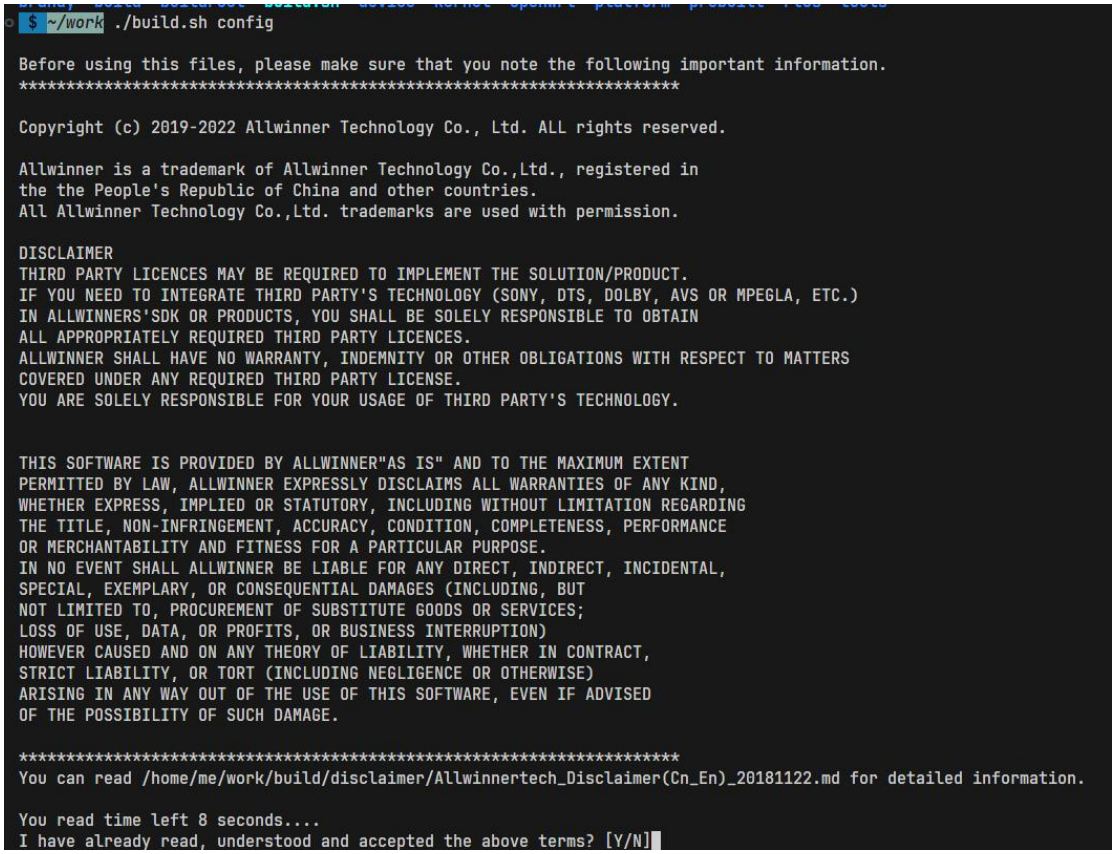
Copyright©2012–2025, Chengdu Ebyte Electronic Technology Co.,Ltd.

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

e the official start of the compilation requires the user to select the configuration that meets their own development board, the SDK also provides a configuration method:

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

After execution you need to agree to some license certifications from Allwinner:



Tap `Y` here and enter to confirm and go 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 build configuration used here is `device/config/chips/t527/configs/ebyte_linux_t527/buildroot /BoardConfig.mk`. Once configured, the final build rule configuration can be viewed in the `.build config` file in the SDK directory, the more important parts of which are listed below:

```
LICHEE_PLATFORM // Compilation platform
LICHEE_LINUX_DEV // Compiled scheme
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
LICHEE_KERN_DEFCONF // Compiled kernel default configuration
```

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

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

```
LICHEE_BUILDING_SYSTEM // Built build system
LICHEE_BR_VER // Buildroot's version
LICHEE_BR_DEFCONF // Buildroot's default configuration
LICHEE_BR_RAMFS_CONF // Buildroot's ramfs default Configuration
LICHEE_BRANDY_VER // version of uboot
LICHEE_BRANDY_DEFCONF // default configuration for brandy
LICHEE_CROSS_COMPILER // cross-compilation chain used for compilation
LICHEE_CHIP_CONFIG_DIR // IC configuration for compilation system Directory
LICHEE_OUT_DIR // Output directory for compilation
```

3.4. Compiling the SDK

Wait for the SDK to finish configuring and then start executing the compilation:

```
./build.sh
```

After compilation is complete, SPL, U-Boot, Linux kernel and Buildroot filesystem image files are generated. The following output is produced at the end of the compilation:

```
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: ----------------------------------------
```

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

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

3.5. Separate compilation

3.5.1. Compiling u-boot separately

The following command allows you to compile u-boot separately, which saves compilation ti me 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. Compiling the kernel separately

The kernel can be compiled separately with the following command, which can save compilat ion time when only the kernel phase needs to be modified, but note that if a driver module is ad ded after the modification, the rootfs needs to be compiled at the same time, otherwise the corres ponding driver module cannot be found:

```
./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
```

The following log is output after compilation:

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

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

```
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. Build buildroot separately

Buildroot can be compiled separately with the following command, which saves compilation ti me when only the buildroot phase needs to be modified:

```
./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'
```

The following log is output after compilation:

```
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

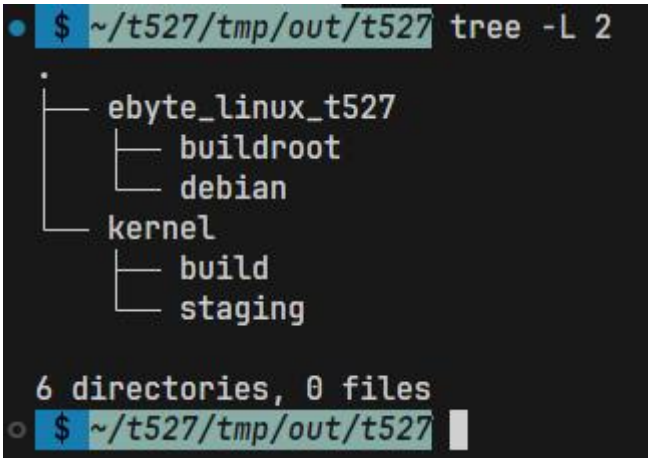
After compiling, you will find an out directory in the SDK directory, which holds all the co mpiled output of the SDK.

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

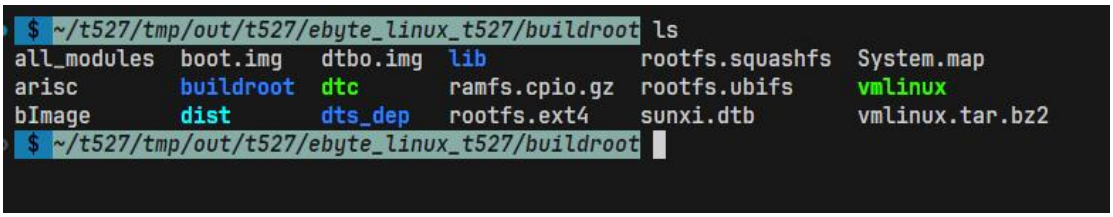
The kernel directory is a soft link to the t527/kernel directory, which is the complete compila tion output of the kernel, and the toolchain directory is the cross-compilation toolchain used in th e compilation process, so let's focus on the t527 directory:

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

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



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



We need to focus on the following files:

- boot.img: kernel image, including kernel, device tree, android image format
- sunxi.dtb: compiled kernel device tree files, will eventually be packaged into boot.img.
- vmlinux: compiled kernel file, will eventually be packaged into boot.img.
- rootfs.*: root filesystem generated by buildroot. Usually suffixed with ext4.

3.7. Image Packaging

After the previous compilation, various compiled files were output, but most of them could n ot be used directly and needed to be packaged into a final usable image:

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

Waiting for the end of the package to view the out directory, you will find an additional pac k_out directory and an img file, the img file is the final image needs to be burned to the develo pment board, while the pack_out directory is to pack the image of the intermediate files, some of these files can also be used eventually.

Let's look at the pack_out directory again:

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

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

```
$ ~/t527/tmp ls -l out/
total 908952
lrwxrwxrwx 1 me me        11 Mar 19 16:04 kernel -> t527/kernel
lrwxrwxrwx 1 me me        30 Mar 19 16:15 pack_out -> t527/ebyte_linux_t527/pack_out
srwxrw-rw- 1 me me         0 Mar 19 14:49 serversocket
drwxr-xr-x 4 me me      4096 Mar 19 14:36 t527
-rwxr-xr-x 2 me me 930752512 Mar 19 16:15 t527_linux_ebyte_linux_t527_uart0.img
drwxr-xr-x 3 me me      4096 Mar 19 14:36 toolchain
```

He can be found as a soft link coming from t527/<board>/pack_out:

```
$ ~/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
```

There are more contents, we mainly focus on the following files:

- boot_package.fex: packaged u-boot image, contains the contents of the boot_package.cfg
- boot-resource.fex: boot process used in the resources such as the logo image.
- env.fex: environment variable configuration for u-boot, copy of SDK/device/product/configs/<board>/[buildroot]/env.cfg.
- boot.fex: soft link to boot.img, the kernel image compiled above.
- rootfs*.fex: soft link to the buildroot root filesystem compiled above.

3.7.1. Problems likely to be encountered

The following situations may be encountered during the packing process:

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

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

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 partition size configured by the user is not suitable for the final output file, you can find the partition corresponding to the corresponding file in `SDK/device/product/configs/<board>/[buildroot]/sys_partition.fex, and then modify the size of the partition to fit the corresponding file, the minimum value of partition size is as described in dl_file_size in the above figure. Then modify the partition size to fit the corresponding file. The minimum partition size is described in dl_file_size in the above figure:

```
[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, CPU chip user guide, and at the same time need to have a certain understanding of the contents of the SDK modification, from the above description of the SDK directory can be known, in the SDK/device/product/configs/ in the storage of the development of the board/program configuration:

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

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

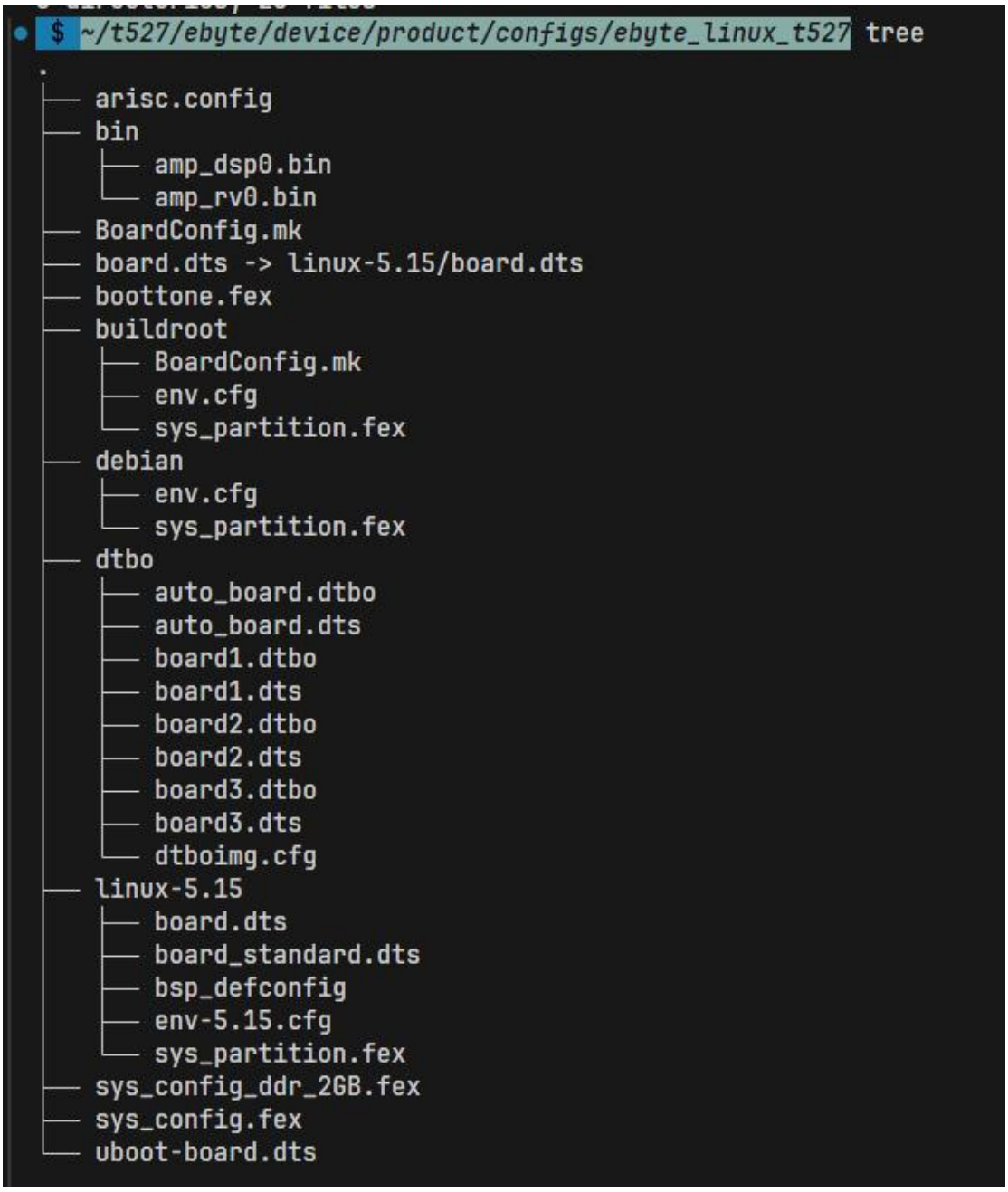


In this level of directory, except for the default directory, it can be considered that they are all different development board/program configurations, while the content in the default directory is used as the default configuration item when the corresponding board-level configuration cannot find the required item, which generally does not need to be modified, and should not be modified.

Take 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



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

- ❖ BoardConfig.mk: the configuration of compilation options that the SDK needs to use.
- ❖ board.dts: linux-5.15/board.dts softlink for the kernel's device tree file.
- ❖ buildroot: detailed configuration used by buildroot to build the root filesystem.
- ❖ bsp_defconfig: kernel-related configuration.
- ❖ sys_config.fex: system parameter configuration, old version of the kernel plays the role of th e device tree, but in Linux5.4 and later versions, only a small number of modules (such as ddr, serial port debug level) to retain the configuration of the function, and mostly used in S

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

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

PL and u-boot stage.

- uboot-board.dts: device tree file for u-boot phase.
- sys_partition.fex: mirrored partition configuration.
- env.cfg: environment variables for the u-boot phase.

We can find that some files with the same name in the previous directory also exist in the buildroot directory, which is an override relationship as follows:

1. bsp, buildroot, debian, etc. are parallel directories and do not overwrite each other.

2. boardConfig.mk, env.cfg, sys_partition.fex of bsp, buildroot, and other system scenarios will overwrite boardConfig.mk, env.cfg, and sys_partition.fex of the default directory.

3. of the same level of directories. BoardConfig_nor.mk, env_nor.cfg, sys_partition_nor.fex will override BoardConfig.mk, env.cfg, sys_partition.fex. Note that only if the nor scheme is selected in the SDK configuration.

4. When the system scheme such as bsp, buildroot is not configured with BoardConfig.mk, env.cfg, sys_partition.fex, the BoardConfig.mk, env.cfg, sys_partition.fex in the default directory will be used.

5. Note that the BoardConfig.mk is an inclusion relationship, from the BoardConfig.mk of the system solution to the upper level, if the system solution and the upper level have the same attributes, it will only be overwritten, otherwise it is regarded as an addition.

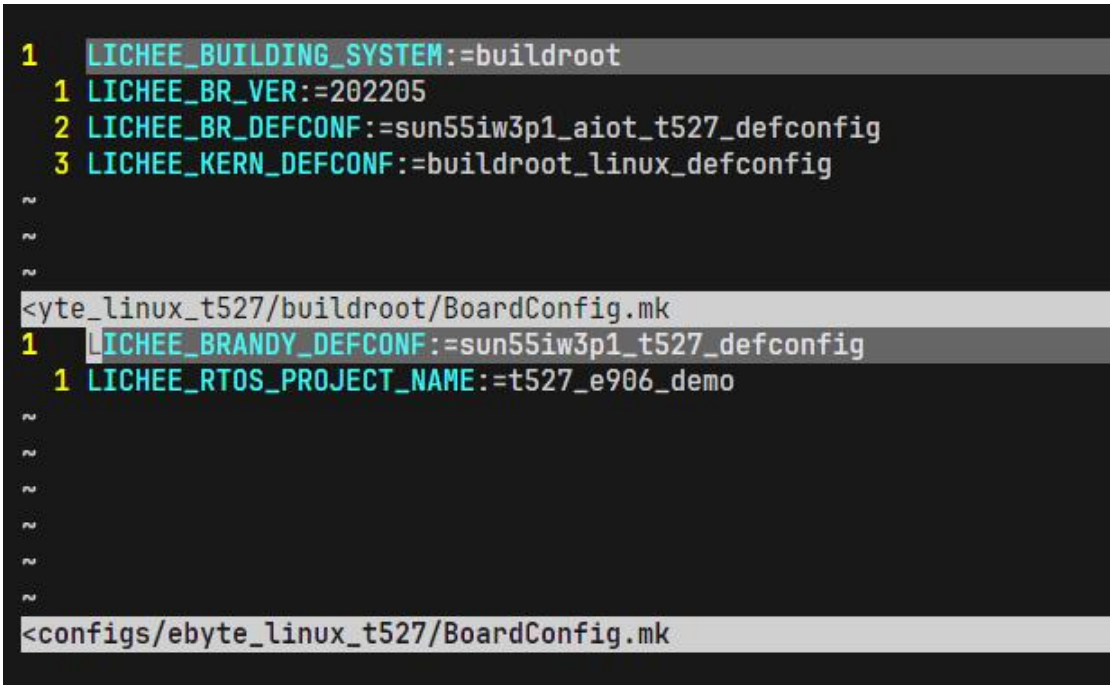
4.1. Program Configuration

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.

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

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

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



We mainly focus on these items:

- LICHEE_KERN_DEFCONF: the name of the kernel defconfig file, which is located in the linux-5.4 directory.
- LICHEE_BR_VER: the version of buildroot, it will automatically go to SDK/buildroot/ directory to find the corresponding version.
- LICHEE_BR_DEFCONF: default configuration of buildroot, in the path where the source code of the corresponding buildroot version is located.
- LICHEE_BRANDY_DEFCONF: this is the default configuration for 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 phase, we only need to modify the sys_config.fex file in most cases.

This file in the kernel version greater than or equal to the beginning of version 5.4, mainly used for SPL, u-boot related configuration, SDK in the compilation/packaging will be based on the configuration of the file for the current development board to do some configuration, we need to modify the main part of the following:

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 Information

```
;-------------------------------------------------------------------------------
; debug_mode=0 means turn off printing logs for Boot0/SPL at startup, default is 8.
; 0: turn off all printing
; 1: show only key node printing
; 4: print error messages
; 8: print all log messages
;-------------------------------------------------------------------------------
; earseflag Indicates whether to perform burn-write is whether to perform erase operation
; 0x0: do not perform erase.
; 0x01: Erase, private, secure storage, user data, mbr partition retained, other erase.
; 0x11: Perform erase, private, secure storage partition reserved, others erased.
; 0x12: force erase (whole 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     ; Recommended to use the automatic identification method, or according to the actual use of the storage device configuration
```

4.2.3. Target Board Debug Serial Port Information

```
;-------------------------------------------------------------------------------
; Describe the form of the gpio: Port:Port + group serial number <function assignment> <inte
```

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

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

```
rnal resistor 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>
```

Here you should determine which serial port to use as the debugging port for your own base board configuration, and it is recommended to use serial port 0, otherwise you need to modify th e contents of the root file system built by buildroot. But for the serial port 0 actually use which GPIO is not required, you can view the "T527_Datasheet_V0.91.pdf" document to find out which pins can be multiplexed to serial port 0.

4.2.4. Other requirements

It is not recommended to modify SPL by default, if you do need to perform some special m odifications in the SPL phase, please refer to the document "Linux_SPL-PUB_Development_Guide. pdf".

4.3. u-boot

4.3.1. Device tree

The uboot-board.dts file is the device tree file used at the beginning of the uboot stage, and is only used as the initial stage. When u-boot executes to the sunxi_replace_fdt_v2 function, it wi ll start to use the device tree information of the kernel stage, so only a very small part of the sit uation needs to be configured in this file.

4.3.2. Environmental variables

The env.cfg file is the environment variable file for the u-boot stage and can be configured b y the user, but there are a few places in it that are special:

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

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

4.3.2.1. 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 SDK/brandy/brandy-2.0/u-boot-2018/board/sunxi/board_helper.c file, which means that the value of the bootcmd variable will be automatically switched according to the actual boot memory device, and the switching rule is to change “setargs_nand” to the corresponding environment. The switching rule is to change “setargs_nand” to the corresponding environment, so users can keep using the following configuration, which is automatically switched by u-boot.

```
bootcmd=run setargs_nand boot_normal
```

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

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 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, and the switching rule is to change the value of “xxx_root” to “root_part_name”. The switching rule is to change the value of “xxx_root” to the value of “root_part_name”, which in general should be left to u-boot to switch automatically.

```
[    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.

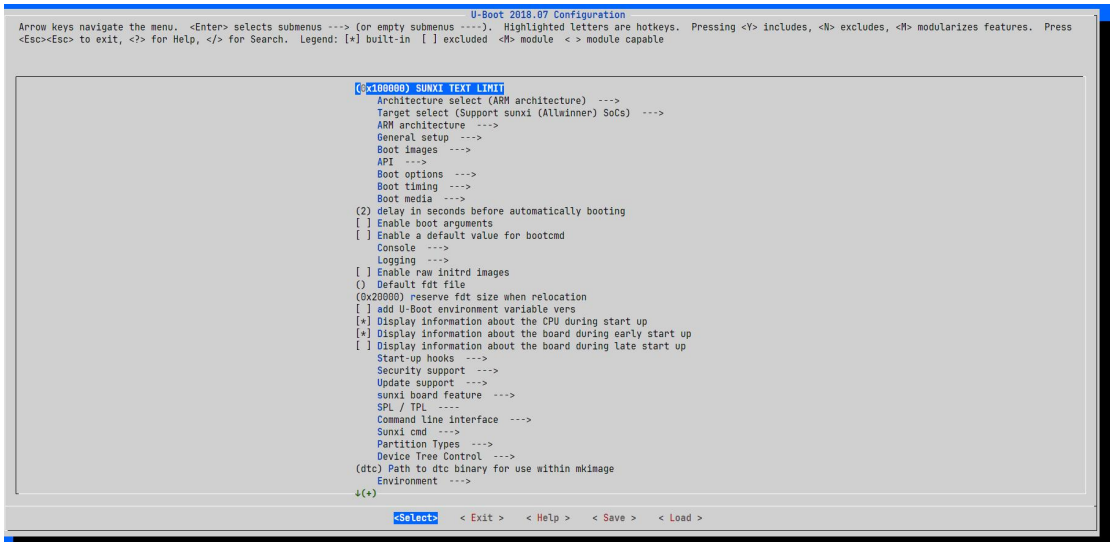
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.3.3. defconfig

The defconfig configuration file for the u-boot stage is based on the value of the LICHEE_BRANDY_DEFCONF entry in the BoardConfig.mk file, which in the buildroot scenario of evb1_auto is brandy/brandy-2.0/u-boot-2018/ configs/sun55iw3p1 _t527_defconfig. it is generally not recommended to modify it, if users do need to modify it, please follow the steps below to enter the menuconfig interface:

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

make menuconfig
```



Note that if defconfig is reconfigured, what was just configured via menuconfig will be overwritten.

4.4. Kernel

Kernel-related configuration options are stored in the <SDK>/device/product/configs/<board>/linux-5.15 directory, which mainly consists of two files, one for board.dts, and one for bsp_defconfig. board.dts will be softwired during compilation to SDK/device/product/configs/<board>/board.dts, and ultimately generated SDK/out/kernel/staging/sunxi.dtb.

4.4.1. Device tree

From the above we know that the current board device tree configuration file is board.dts, for the board.dts file, most of the nodes have the corresponding description of the description of the explanation for the user's reference, but there are some places more special:

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

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

1. spi-nand 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.

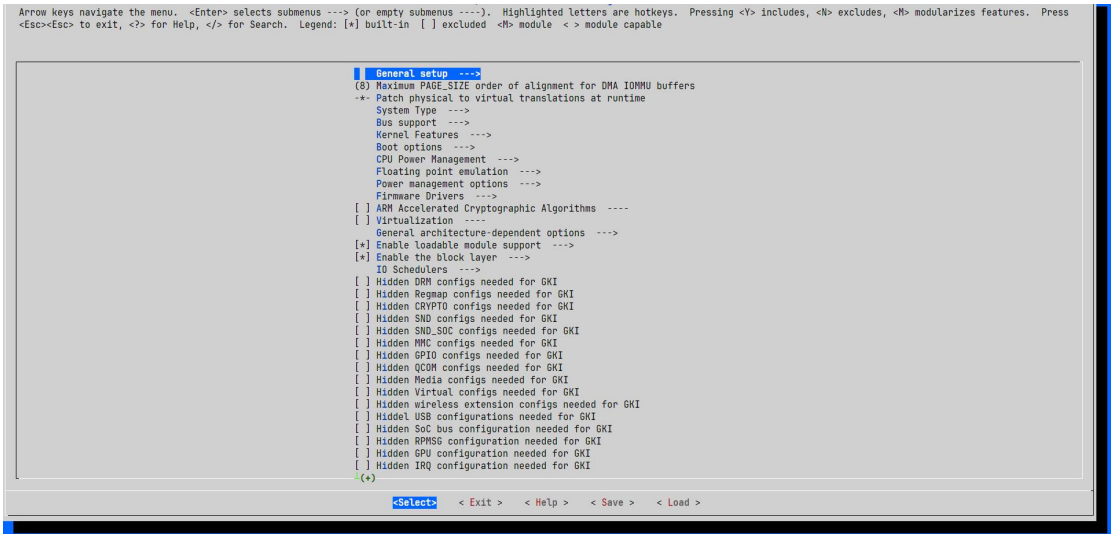
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

The above code exists in 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 based on the information of the corresponding storage device at boot time, so users may find that Although the device node of the corresponding storage device is configured as `status = “disabled”`, it can still enter the system normally.

4.4.2. defconfig

From the above, we can know that the board-level defconfig file is determined based on the LICHEE_KERN_DEFCONF value in the BoardConfig.mk file, and the configuration file must exist in the <SDK>/device/product/configs/ <board>/linux-5.15 directory.

Here 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, it can be used in the top level directory of the SDK the following way to enter the menuconfig interface to configure:



After the changes are made, users can update and save the new configuration to the bsp_defconfig file with the following command:

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

4.5. Rootfs

4.5.1. Buildroot

From the above, we can know that the board-related configurations related to buildroot are st

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

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

ored in the <SDK>/device/product/configs/<board>/[buildroot]/BoardConfig.mk file, which is mainly the source code files of LICHEE_BR_VER and LICHEE_BR_ DEFCONF. Here we take ebyte_linux_t527 as an example, the version of buildroot used is 202205, its source code file is located in SDK/buildroot/buildroot-202205, and its build configuration file is located in the buildroot source directory under configs/ sun55iw3p1_aiot_t527_defconfig.

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

4.5.2. Unable to log in

For users who are using buildroot version 202205, if you cannot log in or if you log in and find that you need to type the same content twice in a row before it is displayed once in the terminal, such as typing `aa` and the terminal displays `a`, please modify the following file: `<SDK>/buildroot/config/buildroot/buildroot/ post_build.sh`, and add the following contents 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
```

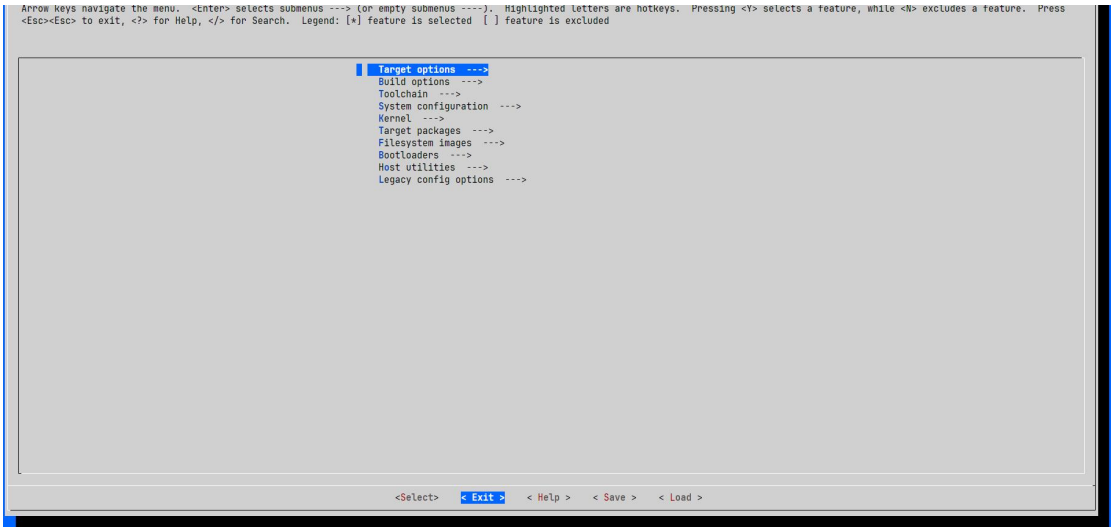
4.5.3. defconfig

For most users, in addition to the kernel needs to be adapted to the baseboard, but also need to modify the root filesystem to adapt to their own needs, it is recommended to use the menuconfig way to modify the configuration, and further adapt to their own program needs. You can enter buildroot's menuconfig interface with the following command:

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

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

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



After the modification, users can update and save the new configuration to the sun55iw3p1_aiot_t527_defconfig file with the following command:

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

4.5.4. Partition

The sys_partition.fex file describes the partition of the packaged image, the main part that the user needs to modify is the size of the rootfs partition and the root filesystem that the partition should be put into, which should not be smaller than the actual root filesystem size, and should not be larger than the physical usable size of the storage device.

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

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

4.6. RISC-V

For RISC-V porting use, customers can refer to “FreeRTOS_RTOS_System_Development_Guide.pdf”, which will not be repeated here.

5. How to burn an updated system image

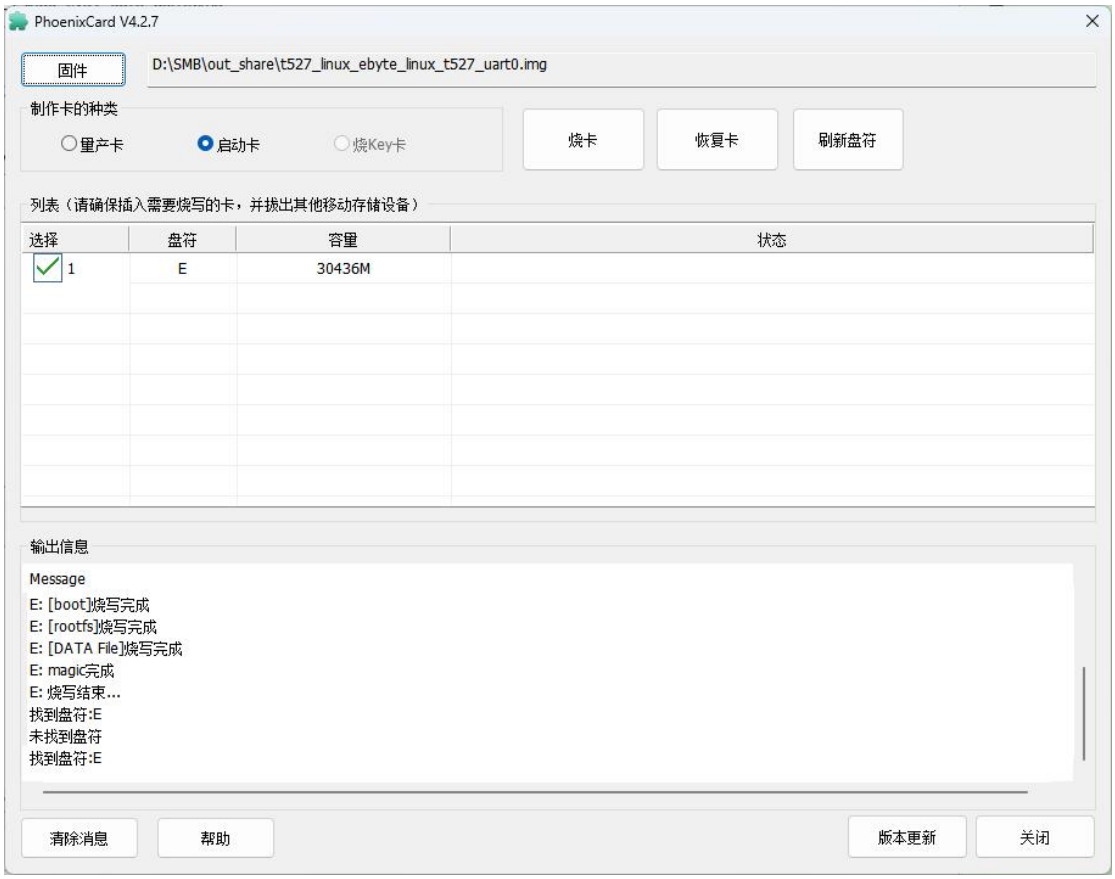
Copyright©2012–2025, Chengdu Ebyte Electronic Technology Co.,Ltd.

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

This chapter focuses on the burning update method based on the ECK32-T527 core board.

5.1. Creating an SD card starter card

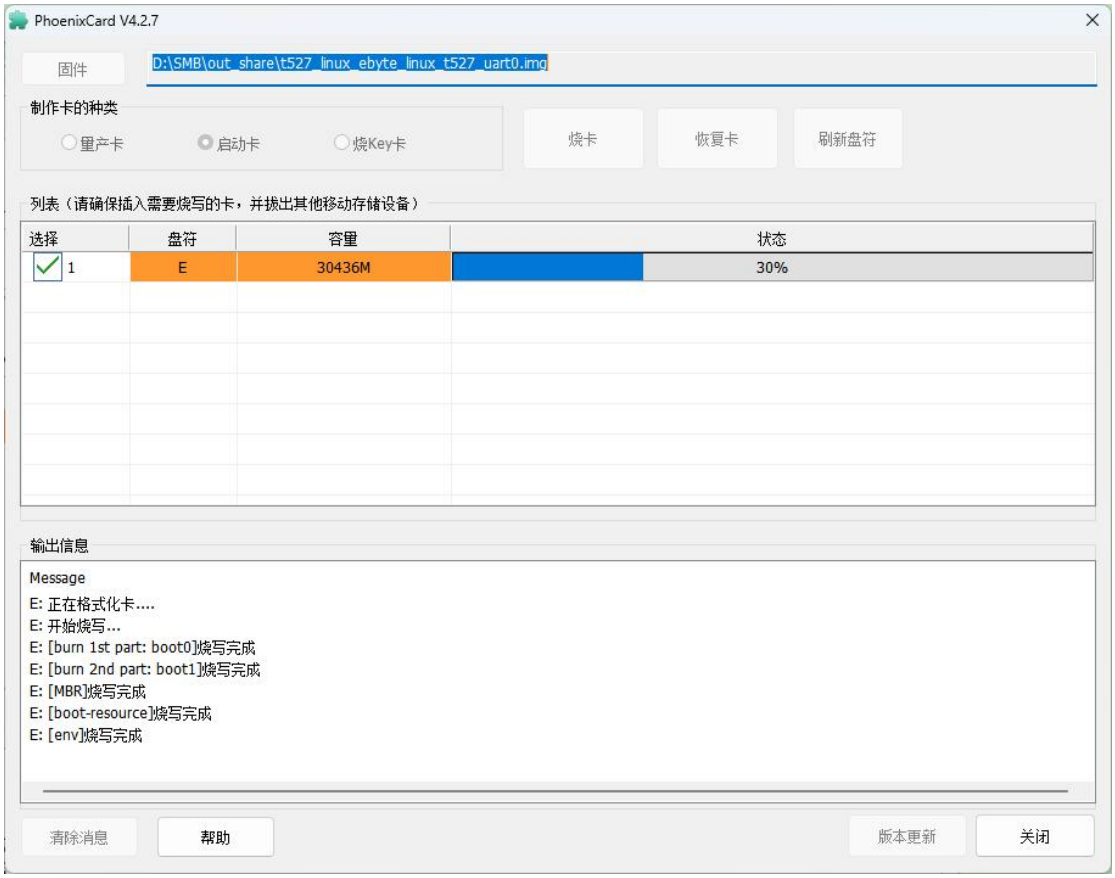
First open the PhoenixCard program with the following interface:



Users should select the firmware to be burned in order, select the card type as “boot card”, and then select the SD card device to be burned, the software will automatically scan the availabl e removable 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 proces s:

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

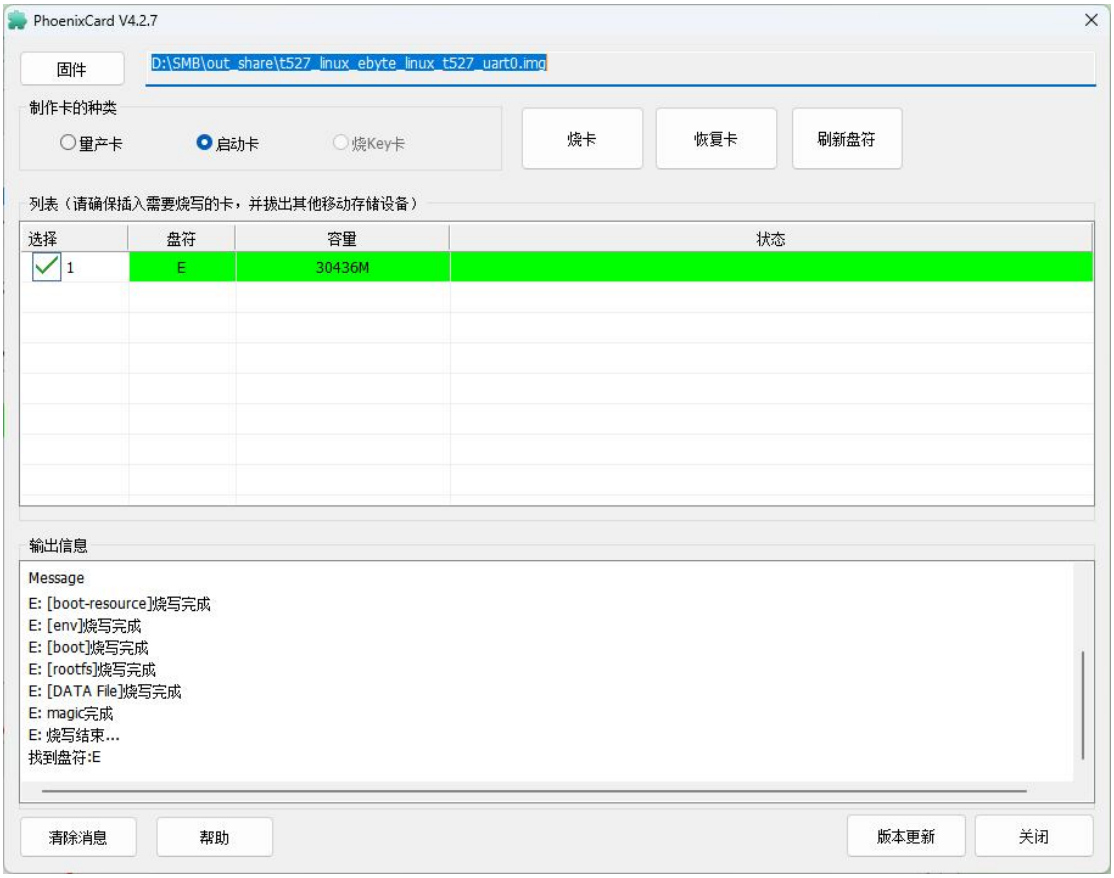
Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



The interface is as follows, you can check whether you have successfully made the boot card through the output message 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. Production card burning

In the production of SD card boot card, we will also find that there is a production card pro duction categories, the way available through the production of an SD card to carry out the pipeli ne operation of the burning function, the user does not need to use a separate image of the firm ware, direct use of the boot card stage can be used directly to correctly use the image can be.

After the production is completed from the production card to start the system, connected to the debugging serial port we can view all the print information of the burning process:

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

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

```
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
```

The formal burning of the system begins after the system boots into the u-boot phase.

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

LRI 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
```

After the similar content in the above figure, users can disconnect the power supply of the development board, pull out the production card, and then re-power on, the serial terminal will print the following startup information to make sure that it is started from the eMMC:

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

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

```
[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. Burning with official tools

This section describes how to use PhoenixSuit, the official Allwinner tool, to implement the USB fast burn-in feature.

5.3.1. Entering burn mode

Before using it we need to know how to get the small machine into burn-in mode, according to the official documentation, there are four ways as follows:

1. Press and hold the fel button at boot time
2. Connect the serial port at boot time and press and hold the number key "2"
3. Go to the u-boot console and enter: "efex"

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

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

4. Go to the system and enter: "reboot efex" at the system console

The general common way is to power on the device and hold down the fel button, which requires the user to pull out the corresponding pins and connect them to a button when making the base board, this way does not require the storage device to have an available system on it. For way 2, it should be noted that this function is realized in the boot firmware provided by Allwinner officially, if the user configures the wrong serial port description in sys_config.fex, way 2 will be invalidated, but there is no need to worry about it, you can also re-burn the correct firmware by making a mass production card.

5.3.2. Burning Tool Configuration

Let's take a look at the burn-in tool. After installing and opening the tool, the interface is as follows:



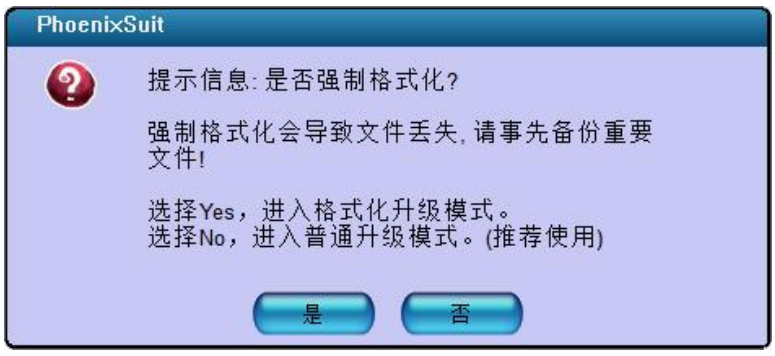
Click on the “One Click Flashing” item on the top to go to the image selection.

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

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



Users do not have to care about the immediate upgrade button, as long as you open the soft ware, select a good burn image, usb cable is normally connected, once the small machine to be b urned into the burn mode, it will automatically recognize and ask for the burn situation:

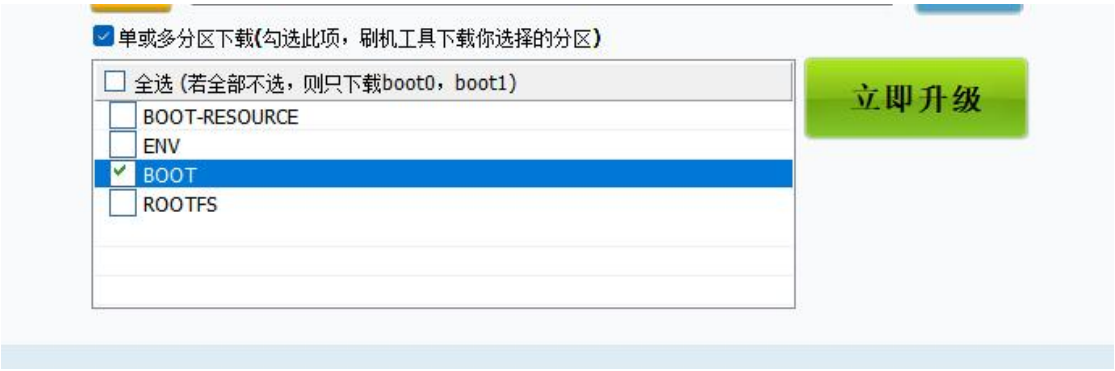


Users can choose whether to format the upgrade or not as needed.

At the same time, the user can also burn part of the content individually by configuring the following figure:

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

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



This text indicates that only boot0, uboot, and kernel parts are burned, and the file system is not updated.

5.4. Problems that may be encountered

5.4.1. Creating a boot card to find a non-storage device

When you use this tool to burn boot card, you may find that the tool can not find the corresponding storage device after accessing the SD card, please make sure that the SD card is occupied by VMware, if it is occupied, please unoccupy it and return the right to use it to the host, if it is still unavailable, please try to replace the USB interface, or check whether it is occupied by other programs.

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 history

Version	Contents of the modification	Modify Time	Preparation of texts	Calibrate	Vetting
V1.0	Initial draft	2025-3-19	WYQ	WFX	WFX

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

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