



T3 系列 Linux-5.10 SDK 开发指南

版本号: 1.0
发布日期: 2022.11.22

版本历史

版本号	日期	制/修订人	内容描述
1.0	2022.11.22	KPA0501	初始版本

目 录

1 前言	1
1.1 文档简介	1
1.2 目标读者	1
1.3 适用范围	1
1.4 相关术语	1
2 目录结构	2
2.1 brandy	2
2.2 bsp	2
2.3 kernel	3
2.4 device/config/chips/t3_*	3
2.5 platform	6
2.6 buildroot	6
2.7 build	7
2.8 tools	7
3 开发环境配置	9
3.1 Linux 服务器开发环境搭建	9
3.1.1 硬件配置	9
3.1.2 系统版本	9
3.1.3 网络环境	10
3.1.4 软件包	10
3.2 Windows PC 环境搭建	11
3.2.1 开发工具安装	11
3.2.2 开发板驱动安装	11
3.2.3 烧录软件安装	12
3.3 开发板介绍	12
3.3.1 使用准备	13
3.3.2 开发板供电	13
3.3.3 串口连接	13
3.3.4 USB 调试连接	14
4 编译代码和打包固件	16
4.1 编译基础	16
4.1.1 基本编译命令	16
4.1.2 编译命令	16
4.2 编译示例	18
4.2.1 基本编译示例	18
4.2.2 编译 nand/nor 板型配置示例	19

4.2.2.1 rawnand 方案的 buildroot 系统编译配置	19
4.2.2.2 spinand 方案的 buildroot 系统编译配置	19
4.2.2.3 spinor 方案的 bsp 系统编译配置	19
4.2.3 AB 系统与 Recovery 系统编译	20
4.2.4 单独编译系统分区	20
4.2.4.1 单独编译 boot0	20
4.2.4.2 单独编译 u-boot-2018	20
4.2.4.3 单独编译 kernel	20
4.2.4.4 单独编译 buildroot	21
4.2.5 单独编译 buildroot 软件包	21
4.2.6 OTA 包的编译方法	21
4.2.6.1 AB 升级包的编译	21
4.2.6.2 Recovery 升级包的编译	22
4.2.7 工具链简介	22
4.2.7.1 Kernel 工具链	22
4.2.7.2 Buildroot 工具链	23
5 常用修改介绍	24
5.1 defconfig 配置修改	24
5.2 内核驱动修改	24
5.3 buildroot 软件包修改	25
5.3.1 添加 buildroot 自研软件包	25
5.3.2 配置 buildroot 开机自启动	25
6 固件烧写	26
6.1 USB 烧录	26
6.1.1 运行 PhoenixSuit	26
6.1.2 连接设备	26
6.1.3 选择 img 文件	27
6.1.4 触发烧录模式	28
6.2 SD 卡烧录	29
6.2.1 制作升级卡	29
6.2.2 插入平台上电升级	30
6.2.3 制作启动卡	31
7 系统调试	32
7.1 串口调试	32
7.2 ADB 调试	33
7.2.1 adb 简介	33
7.2.2 运行 ADB	33
7.2.3 adb 常用命令	33
8 常见问题	35
8.1 编译 uboot 但未更新 uboot 的相关 bin 文件	35

8.2 部分编译命令不支持	35
9 附录	36
9.1 在线帮助文档	36



插图

图 3-1	交叉编译环境	9
图 3-2	开发板驱动	11
图 3-3	T3 P3 开发板介绍	13
图 3-4	串口接口	14
图 3-5	USB-type-c 接口	15
图 6-1	连接设备	27
图 6-2	选择固件	28
图 6-3	PhoenixSuit 用户指南	29
图 6-4	制作升级卡	30
图 6-5	SD 卡槽	31
图 7-1	串口连接	32

1 前言

1.1 文档简介

本文档用于介绍全志科技 T3 系列芯片的 Tina5.0 linux-5.10 SDK。

Tina5.0 系统是基于 linux 的 SDK 开发包，它集成了 BSP、构建系统、linux 应用、测试系统、独立 IP、工具和文档，既可作为 BSP、IP 的开发、验证和发布平台，也可作为嵌入式 Linux 系统。

📖 说明

Linux-5.10 内核版本以上的方案将统一采用 **Tina5.0** 平台架构开发，**Linux-5.10** 内核版本以下的方案继续沿用 **Longan Linux** 系统。此文档将重点阐述 **Tina5.0** 平台上的环境配置编译，请重点关注 **T3** 系列 **Linux-5.10 SDK** 编译章节。

1.2 目标读者

T3、T3-C、T3-Pro 平台开发人员。

1.3 适用范围

本文档适用于 T3 系列 Linux-5.10 V1.0 版本 SDK。

1.4 相关术语

表 1-1: 术语介绍

术语	说明
Tina5.0	全志提供的 Linux SDK
APST	全志量产软件中心
PhoenixSuit	全志平台 USB 烧录软件
PhoenixCard	全志平台卡烧录软件

2 目录结构

Tina5.0 平台主要由 brandy、bsp、build、kernel、buildroot 组成。其中 brandy 包含 boot0 与 uboot2018；bsp 是独立于内核的驱动程序仓库；kernel 为 linux 内核；buildroot 为应用层构建系统。

```
.
├── brandy
├── bsp
├── build
├── buildroot
├── build.sh -> build/top_build.sh
├── device
├── kernel
├── platform
├── prebuilt
├── test
└── tools
```

2.1 brandy

Tina5.0 Linux-5.10 平台使用 brandy2.0 版本，其目录主要结构如下：

```
brandy/brandy-2.0/
├── build.sh -> tools/build.sh
├── spl-pub
├── tools
└── u-boot-2018
```

说明

默认的代码编译流程中只有更新 **uboot** 相关代码或者切换编译板型才会重新编译 **boot0** 与 **uboot**。

2.2 bsp

Tina5.0 Linux-5.10 平台使用 BSP 独立仓库架构，主要目标是降低 BSP 模块驱动代码与内核原生改代码间的耦合度，从而使 BSP 模块可以便捷地适配各种内核版本，方便产品内核升级。

其目录主要结构如下：

```
bsp/
├── configs
├── drivers
├── include
└── Kconfig
```


- Makefile
- modules
- platform
- ramfs



BSP 独立仓库介绍可参考文档：《Linux_BSP 独立仓库使用指南》。

2.3 kernel

Tina5.0 Linux-5.10 平台使用的内核版本是 linux-5.10.149，其目录主要结构如下：

```
kernel/linux-5.10/
├── arch
├── block
├── bsp -> ../../bsp
├── certs
├── COPYING
├── CREDITS
├── crypto
├── Documentation
├── drivers
├── fs
├── include
├── init
├── ipc
├── Kbuild
├── Kconfig
├── kernel
├── lib
├── LICENSES
├── MAINTAINERS
├── Makefile
├── mm
├── net
├── README
├── samples
├── scripts
├── security
├── sound
├── tools
├── usr
└── virt
```



默认目录结构跟标准的 **linux** 内核一致，请谨慎修改，建议优先修改 **bsp** 目录下的代码，方便后续更新内核版本，**bsp** 仓库介绍可参考文档：《Linux_BSP 独立仓库使用指南》。

2.4 device/config/chips/t3_*

芯片配置目录，包含多个板级配置，每个板级配置都有不同的 board.dts, sys_config.fex 等配置文件。

device/config/chips/t3_*

```

├─ bin
│   ├── boot0_nand_sun8iw11p1.bin          # rawnand&spinand flash的boot0文件
│   ├── boot0_sdcard_sun8iw11p1.bin        # emmc flash的boot0文件
│   ├── boot0_spinor_sun8iw11p1.bin        # spinor flash的boot0文件
│   ├── fes1_sun8iw11p1.bin
│   ├── optee_sun8iw11p1.bin
│   ├── sboot_sun8iw11p1.bin
│   ├── u-boot-spinor-secure-sun8iw11p1.bin
│   ├── u-boot-spinor-sun8iw11p1.bin
│   └── u-boot-sun8iw11p1.bin
├─ boot-resource
│   ├── boot-resource
│   │   ├── bat
│   │   ├── bootlogo.bmp                  # 开机启动logo 支持32bit与24bit的bmp图片
│   │   ├── fastbootlogo.bmp
│   │   ├── font24.sft
│   │   └── font32.sft
│   └── boot-resource.ini
├─ configs
│   ├── default
│   │   ├── autobuild.mk
│   │   ├── BoardConfig.mk
│   │   ├── boot_package.cfg
│   │   ├── boot_package_nor.cfg
│   │   ├── default.awlic
│   │   ├── diskfs.fex
│   │   ├── dragon_toc_android.cfg
│   │   ├── dragon_toc.cfg
│   │   ├── env_burn.cfg
│   │   ├── env.cfg
│   │   ├── env_dragon.cfg
│   │   ├── env_ubifs.cfg
│   │   ├── image_android.cfg
│   │   ├── image.cfg
│   │   ├── image_dragonboard.cfg
│   │   ├── image_linux.cfg
│   │   ├── image_nor.cfg
│   │   ├── linux-5.10                    # linux-5.10默认配置，优先级低于板级目录下的配置文件
│   │   │   ├── bsp_defconfig              # emmc&rawnand flash的bsp方案的kernel config配置
│   │   │   ├── bsp_fast_boot_defconfig   # 快启方案的kernel config配置
│   │   │   ├── bsp_nor_defconfig          # spinor flash的bsp方案的kernel config配置
│   │   │   ├── bsp_spinand_defconfig      # spinand flash的bsp方案的kernel config配置
│   │   │   └── dragonboard_defconfig      # emmc&rawnand flash的dragonboard方案的kernel
│   ├── config配置
│   │   └── dragonboard_spinand_defconfig # spinand flash的dragonboard方案的kernel
│   └── config配置
│       ├── S50module
│       ├── overlay.fex
│       ├── parameter.fex
│       ├── sys_partition.fex
│       ├── sysrecovery.fex
│       ├── verity_block.fex
│       └── version_base.mk
├─ p3                                # emmc方案配置，支持bsp/dragonboard/buildroot方案
│   ├── BoardConfig.mk                # bsp方案的版型配置
│   ├── board.dts -> linux-5.10/board.dts
│   ├── bsp                            # bsp方案配置目录
│   └── env.cfg                        # bsp方案的env配置，优先级高于default目录下的env.cfg

```

```

├── buildroot                                # buildroot方案配置目录
│   ├── BoardConfig.mk                      # buildroot方案的版型配置
│   ├── env-ab.cfg                          # AB系统方案的env配置
│   ├── env.cfg                            # buildroot方案的env配置，默认AB系统
│   ├── env-recovery.cfg                   # Recovery系统方案的env配置
│   ├── swupdate                           # OTA方案的系统配置
│   ├── sys_partition-ab.fex               # AB系统方案的分区表配置
│   ├── sys_partition.fex                  # buildroot方案的分区表配置，默认AB系统
│   └── sys_partition-recovery.fex          # recovery系统方案的分区表配置
├── dragonboard                             # dragonboard方案配置目录
│   ├── BoardConfig.mk                     # dragonboard方案的版型配置
│   ├── env.cfg                           # dragonboard方案的env配置
│   ├── sys_partition.fex                  # dragonboard方案的分区表配置
│   └── test_config.fex                    # dragonboard方案的测试方案配置
├── dtbo                                    # dtbo配置目录
├── linux-5.10                             # linux-5.10内核配置目录
│   ├── board.dts                          # linux-5.10内核的dts设备树配置
│   ├── buildroot_bsp_defconfig            # buildroot方案的kernel config配置
│   └── buildroot_bsp_recovery_defconfig    # buildroot方案下recovery系统的kernel
config配置
├── sys_config.fex                          # p3板型的sys_config.fex配置，仅支持boot0的配置
├── uboot-board.dts                         # p3板型的uboot阶段使用的dts设备树配置
├── p3-nor                                  # spinor方案配置，支持bsp方案
│   ├── BoardConfig_nor.mk                 # bsp方案的版型配置
│   ├── board.dts -> linux-5.10/board.dts
│   ├── bsp                               # bsp方案配置目录
│   ├── dtbo                              # dtbo配置目录
│   ├── linux-5.10                         # linux-5.10内核配置目录
│   ├── sys_config.fex                     # p3-spinor板型的sys_config.fex配置，仅支持boot0的配置
│   ├── sys_partition_nor.fex              # p3-spinor板型的分区表配置
│   └── uboot-board.dts                    # p3-spinor板型的uboot阶段使用的dts设备树配置
├── p3-rawnand-ubi                          # rawnand方案配置，支持bsp/dragonboard/buildroot方案
│   ├── BoardConfig.mk                     # bsp方案的版型配置
│   ├── board.dts -> linux-5.10/board.dts
│   ├── bsp                               # bsp方案配置目录
│   ├── buildroot                         # buildroot方案配置目录
│   ├── dragonboard                       # dragonboard方案配置目录
│   ├── dtbo                              # dtbo配置目录
│   ├── linux-5.10                         # p3-rawnand板型的linux-5.10内核配置目录
│   ├── sys_config.fex                     # p3-rawnand板型的sys_config.fex配置，仅支持boot0的配置
│   ├── sys_partition.fex                  # p3-rawnand板型的分区表配置
│   └── uboot-board.dts                    # p3-rawnand板型的uboot阶段使用的dts设备树配置
├── p3-spinand-ubi                          # spinand方案配置，支持bsp/dragonboard/buildroot方案
│   ├── BoardConfig.mk                     # bsp方案的版型配置
│   ├── board.dts -> linux-5.10/board.dts
│   ├── bsp                               # bsp方案配置目录
│   ├── buildroot                         # buildroot方案配置目录
│   ├── dragonboard                       # dragonboard方案配置目录
│   ├── dtbo                              # dtbo配置目录
│   ├── linux-5.10                         # p3-spinand板型的linux-5.10内核配置目录
│   ├── sys_config.fex                     # p3-spinand板型的sys_config.fex配置，仅支持boot0的配置
│   ├── sys_partition.fex                  # p3-spinand板型的分区表配置
│   └── uboot-board.dts                    # p3-spinand板型的uboot阶段使用的dts设备树配置
├── dtbo
├── tools
│   ├── cardscript.fex
│   └── readme.txt

```



说明

linux-4.9 以后内核版本的 **SDK**, **sys_config.fex** 只涉及 **boot0** 的修改, 如 **DDR** 参数等; **uboot-board.dts** 给 **uboot** 模块提供支持; **board.dts** 给 **kernel** 模块提供支持。**board.dts** 在编译过程中会覆盖内核目录下的 **dtsti** 文件 (**bsp/configs/linux-5.10/sun8iw11p1.dtsi**), 因此对于所有版型的公共 **dts** 配置, 可以直接配置在 **sun8iw11p1.dtsi** 文件中。

2.5 platform

应用层软件包, 一般跨项目的共性软件包可以放置在此目录下。

```
platform
├── allwinner
│   ├── display
│   │   ├── libgpu                # GPU的mali.so库
│   │   └── libuapi
│   ├── power
│   │   └── healthd
│   ├── system
│   │   ├── ota-burnboot          # OTA升级相关代码
│   │   └── rpbuf
│   ├── usb
│   │   ├── adbd                # adbd相关代码
│   │   ├── mtp
│   │   └── setusbconfig
│   ├── utils
│   │   ├── boot-play
│   │   ├── cpu_monitor          # cpu_monitor相关代码
│   │   ├── libsocket_db
│   │   ├── mtop                 # mtop相关代码
│   │   └── uevent-monitor
│   ├── wireless
│   │   ├── btmanager           # bt相关代码
│   │   ├── common
│   │   ├── firmware            # wifi/bt的固件包
│   │   └── wifimanager          # wifi相关代码
└── Makefile -> build/Makefile
```

2.6 buildroot

buildroot 软件包, 平台默认支持 buildroot-202205 版本。

```
├── buildroot-202205            # buildroot原生代码
│   ├── arch
│   ├── board
│   ├── boot
│   ├── build.sh
│   ├── CHANGES
│   ├── Config.in
│   ├── Config.in.legacy
│   └── configs
```

```

├── COPYING
├── DEVELOPERS
├── dl # buildroot原生软件包
├── docs
├── fs
├── linux
├── Makefile
├── Makefile.legacy
├── output
├── package # buildroot原生软件包编译配置
├── README
├── support
├── system
├── toolchain
├── utils
├── config # buildroot自研软件包编译配置
├── buildroot
├── package # buildroot方案自研软件包
├── auto # buildroot方案模块测试demo
├── cedarx # aw多媒体cedarx框架代码
├── libcedarc # aw多媒体cedarc框架代码

```

2.7 build

编译脚本目录。

```

build$ tree -L 1
.
├── bin # 编译生成rootfs所用的工具
├── bsp.sh
├── buildbase.sh
├── createkeys # 生成安全密钥的脚本
├── disclaimer
├── envsetup.sh
├── getvmlinux.sh
├── hook
├── Makefile
├── mkcmd.sh
├── mkcommon.sh
├── mkkernel.sh
├── pack # pack使用的脚本
├── parser.sh
├── quick.sh
├── shflags
├── swapsdc.sh
├── swupdate # 生成OTA包的脚本
├── top_build.sh

```

2.8 tools

编译打包工具，tools_win 是 PC 端烧录等工具目录。

```
tools/  
├── build  
├── codecheck  
├── pack  
├── tools_dev  
└── tools_win
```

3 开发环境配置

本章主要介绍了如何在本地搭建编译环境来编译 T3 系列 Linux-5.10 SDK 源代码。PC 编译环境建议使用 Ubuntu 14.04(64 bit) linux 环境进行编译。

一个典型的嵌入式开发环境通常包括 linux 服务器、Windows PC 和目标硬件板。linux 服务器上建立交叉编译开发环境，为软件开发提供代码更新下载，代码交叉编译服务。

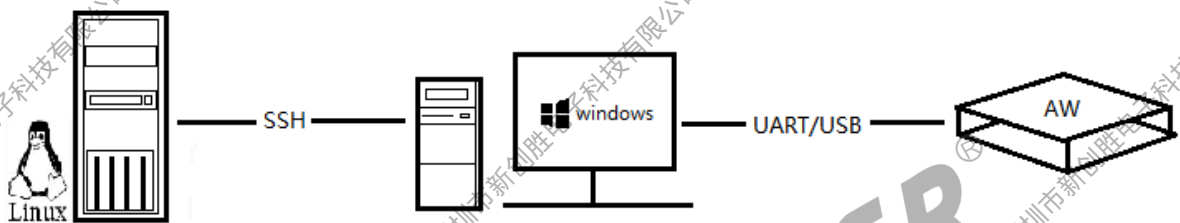


图 3-1: 交叉编译环境

Windows PC 和 Linux 服务器共享程序，Windows PC 并安装 SecureCRT 或 puTTY，通过网络远程登陆到 Linux 服务器，在 linux 服务器上进行交叉编译和代码的开发调试。

Windows PC 通过串口和 USB 与目标开发板连接，可将编译后的镜像文件烧写到目标开发板，并调试系统或应用程序。

3.1 Linux 服务器开发环境搭建

3.1.1 硬件配置

推荐 64 位系统，硬盘空间大于 30G。如果您需要进行多个系统的构建，请预留更大的硬盘空间。

3.1.2 系统版本

本 SDK 开发环境安装如下版本 linux 系统，在默认情况下，SDK 均以此 linux 系统进行编译。

Ubuntu 14.04.5 LTS:


```
Linux version 3.19.0-80-generic (buildd@lcy01-33) (gcc version 4.8.4 (Ubuntu 4.8.4-2ubuntu1~14.04.3) ) #88~14.04.1-Ubuntu SMP Fri Jan 13 14:54:07 UTC 2017
```

说明

注意：如用其他版本的 *linux*，请自行处理可能出现的软件包或环境设置问题。

3.1.3 网络环境

请自行安装 *nfs*、*samba*、*ssh* 等网络组件，并自行配置网络。

3.1.4 软件包

注意

下面的命令请手动安装，并确认每一个都成功安装。

除了 *gcc*，*ncurses*，*bison*，*autoconf*，*wget*，*patch*，*texinfo*，*zlib*，*dos2unix* 之外，还需要安装一些额外的软件包。配置好网络环境之后，则可以通过如下命令安装编译 SDK 需要的软件包：

```
sudo apt-get install git
sudo apt-get install gnupg
sudo apt-get install flex
sudo apt-get install bison
sudo apt-get install gperf
sudo apt-get install build-essential
sudo apt-get install zip
sudo apt-get install curl
sudo apt-get install libc6-dev
sudo apt-get install libncurses5-dev:i386
sudo apt-get install x11proto-core-dev
sudo apt-get install libx11-dev:i386
sudo apt-get install libreadline6-dev:i386
sudo apt-get install libgl1-mesa-glx:i386
sudo apt-get install libgl1-mesa-dev
sudo apt-get install g++-multilib
sudo apt-get install mingw32
sudo apt-get install tofrodos
sudo apt-get install python-markdown
sudo apt-get install libxml2-utils
sudo apt-get install xsltproc
sudo apt-get install zlib1g-dev:i386
sudo apt-get install gawk
sudo dpkg-reconfigure dash 选择no
sudo ln -s /usr/lib/i386-linux-gnu/mesa/libGL.so.1 /usr/lib/i386-linux-gnu/libGL.so
```

若编译遇到报错，请再根据报错信息，安装对应的软件包。

3.2 Windows PC 环境搭建

本节介绍 Windows PC 端需要的环境配置。

3.2.1 开发工具安装

请自行选择 SourceInsight, Notepad++, Qt Creator, VS code 等 IDE 或其它编辑软件, 以及 Xshell 或 puTTY 等串口通讯软件。

3.2.2 开发板驱动安装

一般在 Windows7 的环境下, 当目标板设备上电并插上 USB 线之后, 会自动安装 USB 设备驱动程序。如果安装成功, 则会在 Windows 管理器中出现下图中红色椭圆形标识的设备 Android Phone。



图 3-2: 开发板驱动

有些电脑在设备上电并插上 USB 线之后, 自动安装 USB 设备驱动程序会失败。推荐使用驱动人生等软件, 自动检索安装驱动程序。

Windows10 系统一般是自带 ADB 驱动的, 如果无法识别到 ADB 设备, 请手动更新驱动程序 (使用 PhoenixSuit 目录下的 Drivers/ADB_Driver 文件夹)。

3.2.3 烧录软件安装

请从 APST 平台下载最新版本的 PhoenixSuit、PhoenixCard 软件。当 sdk 编译打包后，就可以通过 PhoenixSuit 烧录，详细步骤将在后文介绍。

APST 下载方法：登录全志“一号通”平台，点击页面上面的“开发工具”，点击“Windows 工具下载”，安装下载的“.msi”安装包。

烧录软件下载：启动 APST 软件并登录（需要管理员权限），点击左侧的“全部”，查看可用工具并下载，点击“运行”启动工具软件。

如果终端客户没有 APST 权限，请找代理商导出所需的最新版工具的压缩包，可免安装使用。

首次运行 APST 时，会检测当前机器是否包含全志驱动程序，如果驱动程序不存在，会自动进行驱动程序的安装（USB 烧录驱动，不同于 ADB 驱动）。

(请注意 Linux-5.10 SDK 要使用新的 PhoenixSuit(v1.19) 和 PhoenixCard(4.2.8)，否则可能会导致无法进行 USB 烧录、无法卡升级等问题。最新版本的工具请使用 APST 下载)

3.3 开发板介绍

AW T3 公版 P3 板型如下：

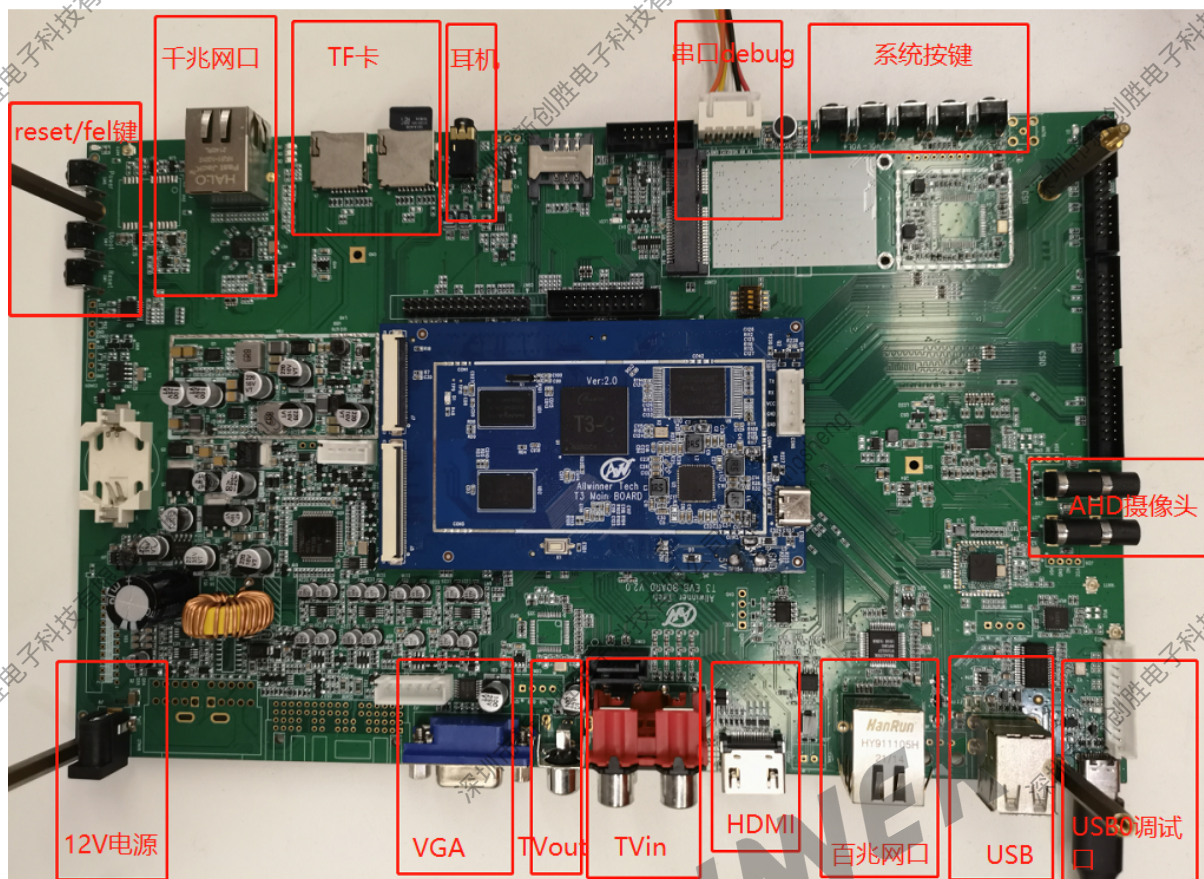


图 3-3: T3 P3 开发板介绍

本手册重点关注 DCIN12V（连接 12V 直流电源），CPU 调试串口（连接串口通讯），USB-OTG type-c 端口（用于烧录和 ADB）。

3.3.1 使用准备

请检查串口硬件工具以及串口连接线、12V 直流电源、以及 USB 烧录线等是否就绪。

3.3.2 开发板供电

请使用 12V 直流电源为开发板供电，供电电流推荐 2A 左右。

3.3.3 串口连接

默认的调试串口用的是 uart0，电压为 3.3v，连接如下图：

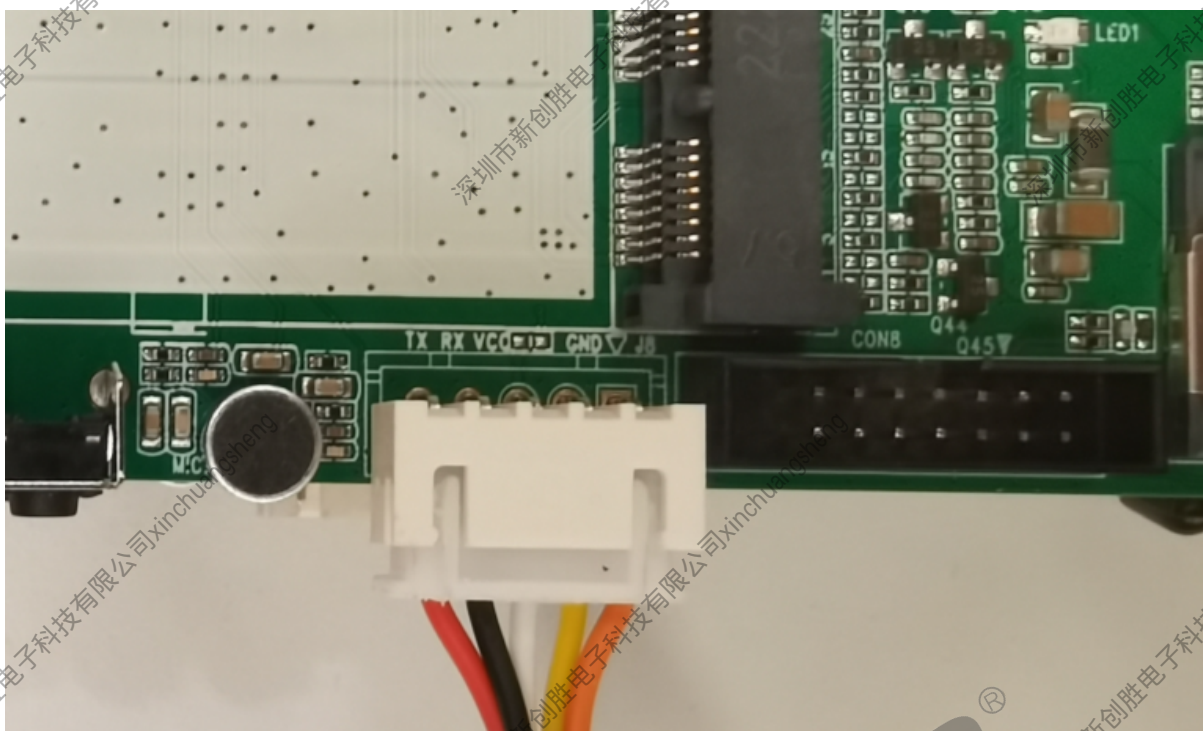


图 3-4: 串口接口

3.3.4 USB 调试连接

请使用 USB Micro 数据线，连接开发板和 windows PC 和 usb 端口。



图 3-5: USB-type-c 接口

4 编译代码和打包固件

本章介绍全编译和部分编译的详细步骤。编译完成后，通过打包，生成最终的 img。

4.1 编译基础

4.1.1 基本编译命令

进入 SDK 顶级目录，执行如下命令即可。

初次进行 SDK 环境配置需要进行多个选项的选择，详情见下文“编译示例”章节。

```
$ source build/envsetup.sh      #进行sdk环境配置（重要）每次开新的shell都请做此操作，再进行开发
$ ./build.sh config             #进行sdk config配置
$ ./build.sh                    #编译整个sdk
$ ./build.sh pack               #打包固件
```

4.1.2 编译命令

可以执行 source build/envsetup.sh 命令进行查看编译命令。

```
$ source build/envsetup.sh
NOTE: The SDK(/Tina5.0) was successfully loaded.
load buildroot,dragonboard,bsp...ok
Invoke . build/quick.sh from your shell to add the following functions to your environment:
  croot          - Changes directory to the top of the tree
  cbasp          - Changes directory to the bsp
  ckernel        - Changes directory to the kernel
  cbrandy        - Changes directory to the brandy
  cboot          - Changes directory to the uboot
  cbr            - Changes directory to the buildroot
  cchips         - Changes directory to the board
  cconfigs       - Changes directory to the board's config
  cbin           - Changes directory to the board's bin
  cdtls          - Changes directory to the kernel's dts
  ckernelout     - Changes directory to the kernel output
  cout           - Changes directory to the product's output
Usage: build.sh [args]
  build.sh       - default build all
  build.sh bootloader - only build bootloader
  build.sh kernel   - only build kernel
  build.sh buildroot_rootfs - only build buildroot
  build.sh menuconfig - edit kernel menuconfig
  build.sh saveconfig - save kernel menuconfig
```

```

build.sh recovery_menuconfig - edit recovery menuconfig
build.sh recovery_saveconfig - save recovery menuconfig
build.sh buildroot_menuconfig - edit buildroot menuconfig
build.sh buildroot_saveconfig - save buildroot menuconfig
build.sh clean - clean all
build.sh distclean - distclean all
build.sh pack - pack firmware
build.sh pack_debug - pack firmware with debug info output to card0
build.sh pack_secure - pack firmware with secureboot
Usage: pack [args]
pack - pack firmware
pack -d - pack firmware with debug info output to card0
pack -s - pack firmware with secureboot
pack -sd - pack firmware with secureboot and debug info output to
card0

```

表 4-1: 编译命令说明

类别	命令	说明
整体编译	<code>./build.sh config</code>	编译配置，弹出编译选择
	<code>./build.sh</code>	根据编译配置，编译 SDK
	<code>./build.sh clean</code>	清除过程文件和目标文件
	<code>./build.sh distclean</code>	清除所有生成的文件
局部编译	<code>./build.sh bootloader</code>	单独编译 bootloader，包含 boot0 与 uboot
	<code>./build.sh kernel</code>	单独编译 kernel
	<code>./build.sh dragonboard</code>	单独编译 dragonboard
	<code>./build.sh buildroot_rootfs</code>	单独编译 buildroot
修改配置	<code>./build.sh menuconfig</code>	配置内核的 defconfig
	<code>./build.sh saveconfig</code>	保存内核的 defconfig
	<code>./build.sh recovery_menuconfig</code>	配置内核 recovery 系统的 defconfig
	<code>./build.sh recovery_saveconfig</code>	保存内核 recovery 系统的 defconfig
	<code>./build.sh buildroot_menuconfig</code>	配置 buildroot 系统的 defconfig
	<code>./build.sh buildroot_saveconfig</code>	保存 buildroot 系统的 defconfig
打包	<code>./build.sh pack</code>	打包命令，调试串口为 uart0
	<code>./build.sh pack_debug</code>	打包命令，调试串口为 card0
	<code>./build.sh pack_secure</code>	打包命令，生成 secure 固件，调试串口为 uart0
	<code>./build.sh pack_debug_secure</code>	打包命令，生成 secure 固件，调试串口为 card0

说明

Linux-5.10 内核上暂不支持再内核目录下通过 `make ARCH=arm menuconfig` 修改配置，请使用 `./build.sh menuconfig` 进行。如需更新 `defconfig` 文件，请使用 `./build.sh menuconfig` 与 `./build.sh saveconfig` 进行更新 `defconfig`。

4.2 编译示例

4.2.1 基本编译示例

以 T3 buildroot p3 EMMC 板型为例，完整的编译步骤如下：

```
$ ./build.sh config # 选择板型配置项
=====ACTION List: mk_config ;=====
options :
All available platform:
    0. android
    1. linux # 选择Linux固件
Choice [linux]: 1
All available linux_dev:
    0. bsp # 选择bsp方案
    1. dragonboard # 选择dragonboard方案
    2. buildroot # 选择buildroot方案
Choice [bsp]: 2
All available ic:
    0. t3 # 选择IC类型，编译的固件与IC必须保持一致，否则无法烧录
Choice [t3]: 0
All available board:
    0. p3-nor # 选择p3-spinor类型机器
    1. p3-rawnand-ubi # 选择p3-rawnand类型机器
    2. p3-spinand-ubi # 选择p3-spinand类型机器
    3. p3 # 选择p3-emmc类型机器
Choice [p3]: 3
All available flash:
    0. default # p3-rawnand/p3-spinand/p3-emmc类型机器选default
    1. nor # p3-spinor类型机器需要选择nor，其他flash类型请勿选择此配置
Choice [default]: 0
Setup BSP files
...
INFO: prepare_buildserver

$ source build/envsetup.sh # 设置环境变量，可使用快速跳转命令、简化编译命令
$ ./build.sh # 一键编译
$ ./build.sh pack # 打包生成固件
```

最终生成的 img 路径：out/t3/p3/buildroot/t3_linux_p3_uart0.img。

📖 说明

若方案为 **T3-C** 方案，则 **All available ic** 需要选择 **t3_c**。

若方案为 **T3-Pro** 方案，则 **All available ic** 需要选择 **t3_pro**。

4.2.2 编译 nand/nor 板型配置示例

4.2.2.1 rawnand 方案的 buildroot 系统编译配置

```
$ ./build.sh config          # 选择板型配置项
依次选择如下选项：
linux、buildroot、t3、p3-rawnand-ubi、default
$ source build/envsetup.sh    # 设置环境变量，可使用快速跳转命令、简化编译命令
$ ./build.sh                  # 一键编译
$ ./build.sh pack             # 打包生成固件
```

最终生成的 img 路径：out/t3/p3/buildroot/t3_linux_p3-rawnand-ubi_uart0.img。



rawnand 方案公版上采用 **128M** 容量进行测试。

4.2.2.2 spinand 方案的 buildroot 系统编译配置

```
$ ./build.sh config          # 选择板型配置项
依次选择如下选项：
linux、buildroot、t3、p3-spinand-ubi、default
$ source build/envsetup.sh    # 设置环境变量，可使用快速跳转命令、简化编译命令
$ ./build.sh                  # 一键编译
$ ./build.sh pack             # 打包生成固件
```

最终生成的 img 路径：out/t3/p3/buildroot/t3_linux_p3-spinand-ubi_uart0.img。



spinand 方案公版上采用 **128M** 容量进行测试。

4.2.2.3 spinor 方案的 bsp 系统编译配置

```
$ ./build.sh config          # 选择板型配置项
依次选择如下选项：
linux、bsp、t3、p3-nor、nor
$ source build/envsetup.sh    # 设置环境变量，可使用快速跳转命令、简化编译命令
$ ./build.sh                  # 一键编译
$ ./build.sh pack             # 打包生成固件
```

最终生成的 img 路径：out/t3/p3/buildroot/t3_linux_p3-nor_uart0.img。



spinor 方案公版上采用 **32M** 容量进行测试，由于 **flash** 大小限制，暂不支持 **buildroot** 系统，仅支持最小 **rootfs**，文件位置：**device/config/rootfs_tar/target-arm-linaro-5.3.tar.bz2**。

4.2.3 AB 系统与 Recovery 系统编译

bsp 方案/dragonboard 方案/spinor 方案中，所有板型均默认使用非 AB 方案，且不带 recovery 分区。

buildroot 方案中，公版 emmc 板型默认配置为 AB 系统分区，公版 rawnand/spinand 板型默认使用非 AB 方案，且不带 recovery 分区。

对于 buildroot 方案编译，如需使用另外一套系统分区方案，可以使用对应方案的 env.cfg 文件以及 sys_partition.fex 文件即可。

以 buildroot 方案下 T3 P3 板型为例，切换到 Recovery 系统编译时，执行以下命令即可。

```
$ cd device/config/chips/t3/configs/p3/buildroot
$ cp env-recovery.cfg env.cfg # 使用recovery系统的env配置
$ cp sys_partition-recovery.fex sys_partition.fex #使用recovery系统的分区表
$ cd -
$ ./build.sh config # 选择板型配置项
依次选择如下选项：
linux、buildroot、t3、p3、default
$ source build/envsetup.sh # 设置环境变量，可使用快速跳转命令、简化编译命令
$ ./build.sh # 一键编译标准系统
$ ./build.sh recovery # 编译recovery系统
$ ./build.sh pack # 打包生成固件
```

4.2.4 单独编译系统分区

4.2.4.1 单独编译 boot0

```
tina5.0/brandy/brandy-2.0$ ./build.sh -o spl-pub -p sun8iw1lp1
```

4.2.4.2 单独编译 u-boot-2018

```
tina5.0$ build.sh bootloader
```

4.2.4.3 单独编译 kernel

```
tina5.0$ build.sh kernel
```

4.2.4.4 单独编译 buildroot

```
tina5.0$ build.sh buildroot_rootfs
```

说明

更多编译命令请参考**编译命令**。

4.2.5 单独编译 buildroot 软件包

以 mtop 软件包为例，使用 buildroot 标准编译方式进行模块编译。

```
进入buildroot目录
$ cd buildroot/buildroot-202205
删除上次mtop编译生成的中间文件
$ make mtop-dirclean
强制重新编译mtop
$ make mtop-rebuild
编译完成后会自动把生成的bin文件拷贝到out目录下的rootfs中
```

此时可以手动push进小机端进行测试，或者重新执行/build.sh即可更新rootfs包

说明

mtop 的源码位置：**platform/allwinner/utls/mtop/***

mtop 的 **buildroot** 编译配置位置：**buildroot/config/buildroot/allwinner/utls/mtop/***

mtop 编译生成的中间文件位置：**out/t3/p3/buildroot/buildroot/build/mtop/***

mtop 编译最终生成的 **rootfs** 下的 **bin** 位置：**out/t3/p3/buildroot/buildroot/target/bin/mtop**

说明

buildroot 的自研软件包每次修改源码后不会自动编译，所以最好手动模块编译一次。

4.2.6 OTA 包的编译方法

OTA 升级方案的详细介绍可参考《A40i 系列 & T3 系列_Linux-5.10_OTA_开发指南》。

说明

emmc/rawnand/spinand 支持 **AB** 系统升级与 **Recovery** 系统升级两种方式，**spinor** 暂不支持 **OTA** 升级。

4.2.6.1 AB 升级包的编译

如果是 emmc 板型，默认是 ab 分区配置，则不需要修改 env 与分区表，可直接开始编译固件。

如果是 rawnand 和 spinand 板型，默认非 ab 分区配置，则需要修改 env.cfg 和 sys_partition.fex 配置文件，然后开始编译固件。

```
$ cd device/config/chips/t3/configs/p3-xxx/buildroot
$ cp env-ab.cfg env.cfg # 使用ab系统的env配置
$ cp sys_partition-ab.fex sys_partition.fex # 使用ab系统的分区表
```

完整的 AB 升级包的编译命令可参考如下：

```
$ ./build.sh config # 选择板型配置项
依次选择如下选项：
linux、buildroot、t3、p3、default
$ source build/envsetup.sh # 设置环境变量，可使用快速跳转命令、简化编译命令
$ ./build.sh # 一键编译
$ ./build.sh pack # 打包生成固件
$ ./swupdate_pack_swu -ab # 生成OTA完整升级包
```

最终生成的 OTA 升级包路径：out/t3/p3/buildroot/swupdate/buildroot_t3_p3-ab.swu。

说明

AB 系统升级命令参考：`/sbin/swupdate_cmd.sh -i /mnt/UDISK/update.swu -e stable,now_A_next_B`

4.2.6.2 Recovery 升级包的编译

完整的 Recovery 升级包的编译命令可参考如下：

```
$ cd device/config/chips/t3/configs/p3/buildroot
$ cp env-recovery.cfg env.cfg # 使用recovery系统的env配置
$ cp sys_partition-recovery.fex sys_partition.fex #使用recovery系统的分区表
$ cd -
$ ./build.sh config # 选择板型配置项
依次选择如下选项：
linux、buildroot、t3、p3、default
$ source build/envsetup.sh # 设置环境变量，可使用快速跳转命令、简化编译命令
$ ./build.sh # 一键编译标准系统
$ ./build.sh recovery # 一键编译内核recovery系统，用于生成recovery.img
$ ./build.sh pack # 打包生成固件
$ swupdate_pack_swu -recovery # 生成OTA完整升级包
```

最终生成的 OTA 升级包路径：out/t3/p3/buildroot/swupdate/buildroot_t3_p3-recovery.swu。

说明

Recovery 系统升级命令参考：`/sbin/swupdate_cmd.sh -i /mnt/UDISK/update.swu -e stable,upgrade_recovery`

`./build.sh recovery` 命令编译出来的 `recovery.img` 为进入 `recovery` 模式后启动的内核。进入 `recovery` 系统后加载的 `rootfs` 为 `buildroot` 制作出来的 `initramfs` 包，文件路径：`bsp/ramfs/rootfs_recovery_32bit.cpio.gz`。

详细介绍可参考文档《A40i 系列 & T3 系列 Linux-5.10 OTA_开发指南》。

4.2.7 工具链简介

4.2.7.1 Kernel 工具链

Kernel 工具链位于：

```
/prebuilt/kernelbuilt/arm/gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi.tar.xz
```

说明

bsp 方案/dragonboard 方案/buildroot 方案的 kernel 编译均使用此工具链进行编译

4.2.7.2 Buildroot 工具链

Buildroot 工具链位于：

```
/buildroot/buildroot-202205/dl/toolchain-external-linaro-armsf/gcc-linaro-7.3.1-2018.05-x86_64_arm-linux-gnueabi.tar.xz
```

说明

Buildroot-202205 版本目前仅支持软浮点工具链编译。

5 常用修改介绍

5.1 defconfig 配置修改

内核 defconfig 修改，建议使用如下命令：

```
tina5.0$ ./build.sh menuconfig      # 进入内核menuconfig界面
tina5.0$ ./build.sh saveconfig      # 根据配置项的修改保存到defconfig文件中
```

内核的 recovery 系统的 defconfig 修改，建议使用如下命令：

```
tina5.0$ ./build.sh recovery_menuconfig # 进入内核recovery系统的menuconfig界面
tina5.0$ ./build.sh recovery_saveconfig # 根据配置项的修改保存到defconfig文件中
```

buildroot 的 defconfig 修改，建议使用如下命令：

```
tina5.0$ ./build.sh buildroot_menuconfig # 进入buildroot系统的menuconfig界面
tina5.0$ ./build.sh buildroot_saveconfig # 根据配置项的修改保存到defconfig文件中
```

5.2 内核驱动修改

由于 linux-5.10 内核采用 bsp 独立仓库机制，建议驱动开发时优先修改 SDK 中 bsp 目录下的驱动代码。

T3 系列 Linux-5.10 SDK 中内核代码与 Linux 社区代码基本保持一致，对应 Linux 社区版本为 Linux-5.10.149，即发布时 SDK 内核代码已合入了 linux 社区 linux-5.10.y 分支上 Linux-5.10.149 之前的所有提交。

SDK 中内核代码后期维护可以自行同步 linux 社区的内核代码，如需更新社区 linux 的内核补丁，可手动下载社区内核代码，并合入到 SDK 中进行编译验证。

linux 社区仓库地址：<https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/>，对应代码分支：linux-5.10.y。



说明

BSP 独立仓库介绍可参考文档：《Linux_BSP 独立仓库使用指南》。

5.3 buildroot 软件包修改

5.3.1 添加 buildroot 自研软件包

以 gpu_um_pub 为例

1) 给 buildroot 增加编译规则，即新建 buildroot/config/buildroot/gpu_um_pub 目录，并创建 gpu_um_pub.mk 与 Config.in。

📖 说明

注意：**gpu_um_pub.mk** 中的 **GPU_UM_PUB_INSTALL_TARGET_CMDS** 需要与 **Config.in** 中的 **BR2_PACKAGE_GPU_UM_PUB** 名称保持一致，即 **GPU_UM_PUB** 需要一致，否则编译不会进入 ***_INSTALL_TARGET_CMDS** 中。

2) 把软件包添加到 buildroot 编译脚本中。

2.1) 修改 buildroot/config/buildroot/Config.in，增加如下信息：

```
source "../config/buildroot/gpu_um_pub/Config.in"
```

2.2) 修改 buildroot/config/buildroot/platform.mk，增加如下信息：

```
include ${PLATFORM_PATH}/gpu_um_pub/gpu_um_pub.mk
```

3) 修改 buildroot 的 defconfig 后，开始编译即可。

5.3.2 配置 buildroot 开机自启动

模块位置：buildroot/allwinner/system/busybox-init-base-files。

以 adbd 开机自启动为例：

1) 新建 S50adb_start 文件，并实现 start 与 stop 方法，分别对应开机自启动运行，以及关机退出进程。

2) 配置 busybox-init-base-files.mk，把新建的 S50adb_start 文件拷贝到 rootfs 中的 /etc/init.d/ 文件夹下。

📖 说明

原理介绍：

buildroot 原生开机运行的模块为 **package/initscripts**，加载 **rootfs** 后首先会运行 **/etc/init.d/rcS** 脚本，这个脚本会遍历 **/etc/init.d/** 文件夹下的 **S** 开头的文件，并依次执行他们。

所以如果期望开机自启动其他进程，可新建 **Sxx** 的文件，拷贝至 **/etc/init.d/** 即可实现。

6 固件烧写

本章介绍如何将编译好的固件，烧写到开发板的步骤。烧录软件前文已经介绍安装方式。

6.1 USB 烧录

这种烧录方式方便开发人员进行软件的开发以及调试，具体步骤如下：

6.1.1 运行 PhoenixSuit

启动 PhoenixSuit，目前最新版本为 1.19，旧版本工具可能不支持 Linux-5.10 平台。

6.1.2 连接设备

开发板上电开机，用 microUSB 线连接到电脑，查看是否检测到设备。检测到设备之后的界面如下，下方会提示设备连接成功：



图 6-1: 连接设备


注意：如未检测到设备，可能驱动未正常安装。可以使用 **PhoenixSuit** 安装目录、**APST** 安装目录、**SDK** 的 **tools/tools_win** 等目录下的驱动手动安装。

6.1.3 选择 img 文件

点击【一键刷机】图标，选择编译生成的 **img** 文件。



图 6-2: 选择固件

注意：选择完文件后，若 PhoenixSuit 是 v1.13 以上的版本，可以点“立即升级”进行升级烧录。

6.1.4 触发烧录模式

进入烧录模式可使用以下任一方法：

方法一：连接 adb 线之后，点击 PhoenixSuit 软件中的“立即升级”按钮后，进入烧录模式。

方法二：连接串口，输入 reboot efex 命令，系统自动重启后，进入烧录模式。

方法三：连接串口，机器上电之前按住键盘“s”键，进入 uboot 模式，再输入命令 efex，系统自动重启后，进入烧录模式。

方法四：连接串口，机器上电之前按住键盘“2”键，再对机器上电后，进入烧录模式。

方法五：硬件上对 fe1 脚进行对地短接，再对机器上电后，进入烧录模式。

注意：开发板供电请务必使用 12V 直流电源供电，请勿使用 type-c USB 线给机器供电。

注意：在开始菜单，或者安装目录，可以找到《PhoenixSuit 用户指南.pdf》，里面介绍了烧录软件的整个流程。更具体的信息请参考该文档。



图 6-3: PhoenixSuit 用户指南

说明

请等待烧录软件提示烧录成功后再断开电源或者 USB 线，否则会导致固件烧录不完整，系统无法启动。出现烧录失败问题请检查固件与板子硬件上连接的 flash 是否匹配。

6.2 SD 卡烧录

此种方式常用于量产或售后软件升级。

6.2.1 制作升级卡

从 APST 下载并打开 PhoenixCard 软件（目前最新版本为 4.2.8），并安装 APST 中提供的“VS Runtime Collection”，制作 sd 升级卡的相关信息可以点 PhoenixCard 软件的“帮助”查看。参考下图，首先选择固件，然后选择“量产卡”，再选择盘符，最后点击“烧卡”，开始制作启动卡。

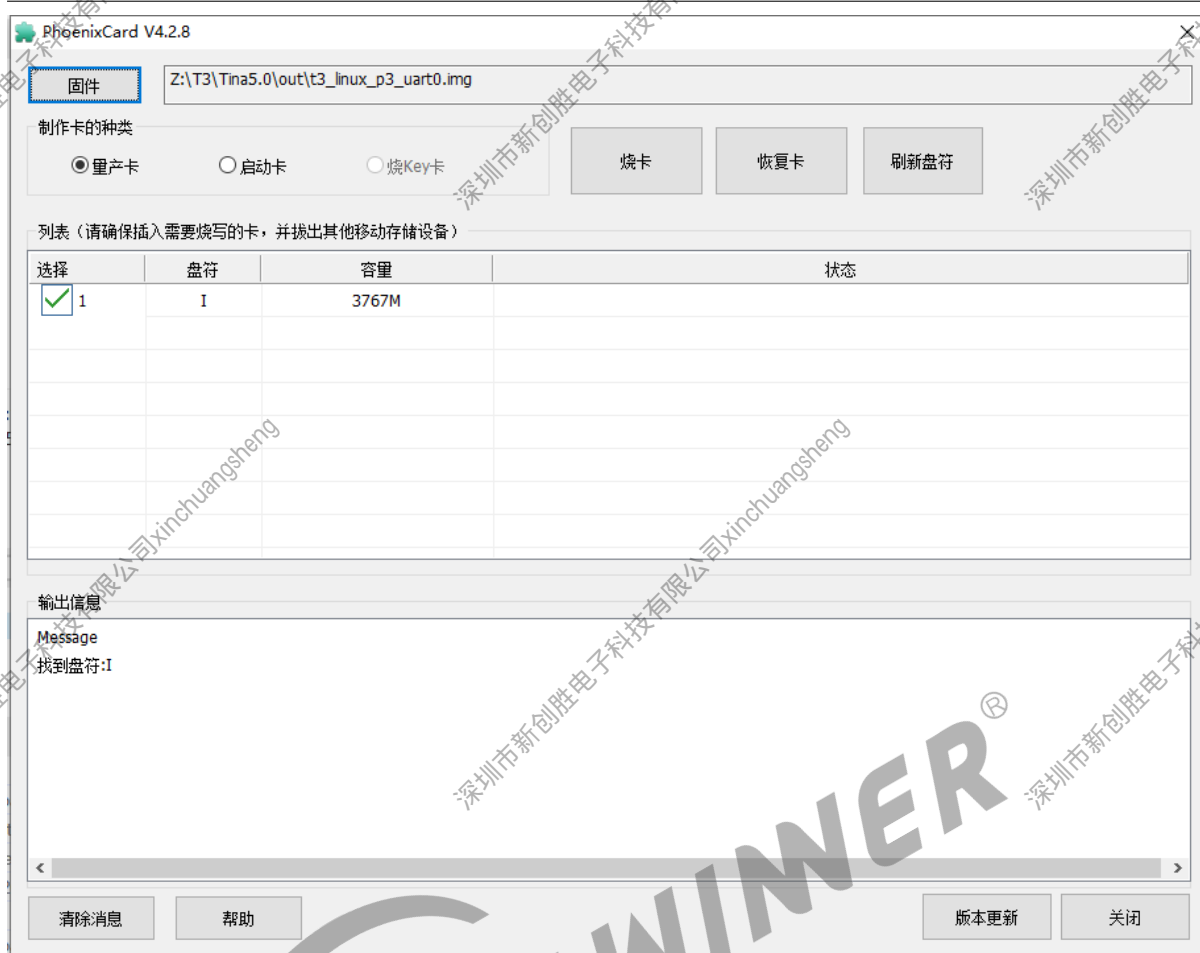


图 6-4：制作升级卡

如果卡里面已经有多个分区，第一次操作可能会失败，重新点击“烧卡”即可，如果多次烧卡失败，请检查是否安装了“VS Runtime Collection”。

(注意：若是 v4.2 以下的软件，则有可能无法进行正常的卡升级/卡烧录动作。)

6.2.2 插入平台上电升级

卡烧录好后，插入机器 TF 卡槽，如下图所示：

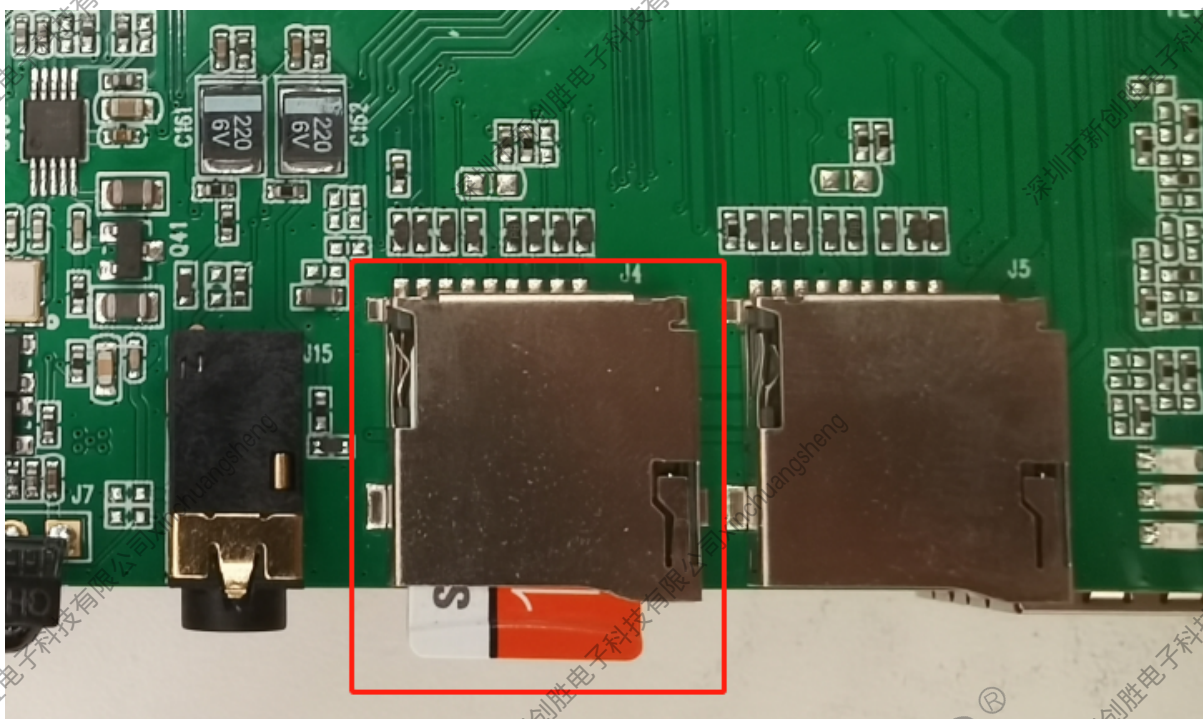


图 6-5: SD 卡槽

重新上电，机器就会自动升级了，可以看到屏幕上有进度条，调试串口有相应输出。整个过程大概 1~2 分钟，具体视固件大小而定。

升级完毕后，串口会打印提示信息，并自动关机，此时可以拔掉 TF 卡，然后重新上电即可。

注：PhoenixCard 软件执行完后，Windows 系统可能会提示无法识别磁盘询问是否格式化，请不要做格式化操作，因为 Linux 的分区 Windows 系统不能识别，如果被 Windows 系统格式化则 PhoenixCard 软件写入数据会全部丢失，这个卡也就白做了。

6.2.3 制作启动卡

打开 PhoenixCard 软件，首先选择固件，然后选择“启动卡”，再选择盘符，最后点击“烧卡”，如果卡里面已经有多个分区，第一次操作可能会失败，重新点击“烧卡”即可，如果多次烧卡失败，请检查是否安装了“VS Runtime Collection”。

使用 PhoenixCard 软件制作的量产卡或者启动卡，里面有多个分区，而且部分分区不能被 Windows 系统所识别，文件浏览器中看到的容量会偏小，如果要恢复作为存储卡，使用 PhoenixCard 软件的“恢复卡”功能进行格式化即可，卡会被格式化为单分区的，容量也会正常显示。也可使用 Windows 自带的磁盘分区工具进行分区删除操作，只是略微繁琐。

7 系统调试

支持串口和 adb 方式来和 windows PC 通讯。

7.1 串口调试

通过 windows PC 端串口通讯工具，连接开发板串口。

配置参数：波特率：115200，数据位：8，奇偶校验位：无，停止位：1。

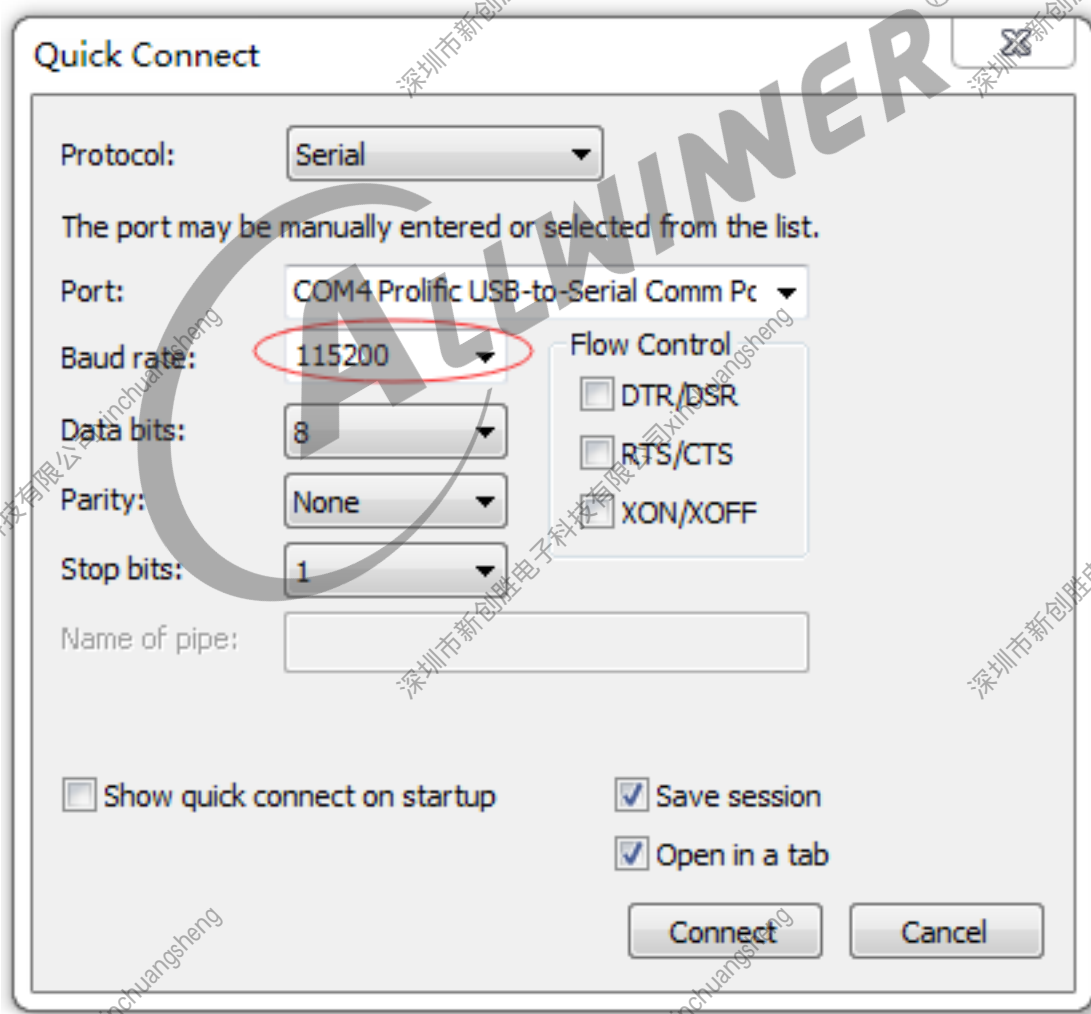


图 7-1：串口连接

7.2 ADB 调试

7.2.1 adb 简介

adb 全称为 Android Debug Bridge，是 Android SDK 里的一个工具，用于操作管理 Android 模拟器或真实的 Android 设备。其主要功能有：

- 运行设备的 shell（命令行）；
- 管理模拟器或设备的端口映射；
- 在计算机和设备之间上传/下载文件。

7.2.2 运行 ADB

Windows PC 端的 adb 使用方法和 adb 应用程序，请自行从网络搜索。

上电会自动加载 adb 脚本，如果有问题，请检查 usb0 的当前模式，并手动切换到 device 模式。

```
# cat /sys/devices/platform/soc@1c00000/soc@1c00000:usb@1c00000/usb_device
[ 226.411530]
[ 226.411530] rmmmod_device_driver
[ 226.411530]
[ 226.418331] rmmmod_device_driver()223 WARN: get power supply failed
[ 226.426754]
[ 226.426754] insmod_device_driver
[ 226.426754]
device_chose finished!
```

如果发现 PhoenixSuit 工具已经提示设备已经连接成功，则说明已经可以了，如果没有提示，请再次执行/etc/init.d/S50adb_start，并检查接线是否正常。

adb 连接成功后就可以在 Windows PC 直接通过 adb 来更新平台的应用程序或者库文件，不用重新烧录固件。

7.2.3 adb 常用命令

- pc 端查看当前连接的设备：

```
adb devices
```

- PC 端进入设备 shell 命令行模式：

```
adb shell
```

- 将电脑上的文件上传至设备：

```
adb push <local path> <remote path>
```

- 下载设备里的文件到电脑：

```
adb pull <remote path> <local path>
```


8 常见问题

本章主要介绍平台环境搭建的常见问题及解决办法。

8.1 编译 uboot 但未更新 uboot 的相关 bin 文件

目前 build 脚本中在编译 bootloader 时，会优先检测 uboot 相关代码是否有改动，如果没有改动则会跳过 bootloader 的编译，所以如需强制编译 bootloader，可以更新 brandy/brandy-2.0/u-boot-2018 目录下任意文件的时间戳，重新编译即可。

```
INFO: build arisc
INFO: build_bootloader: brandy_path=~/.Tina5.0/brandy/brandy-2.0
INFO: skip build brandy.
INFO: build kernel ...
INFO: prepare_buildserver
INFO: Prepare toolchain ...
```

8.2 部分编译命令不支持

部分编译命令，如 croot/cbsp 等无法使用，可能是因为没有使能环境变量，需要先回到 SDK 根目录执行如下命令：

```
source build/envsetup.sh
```

9 附录

9.1 在线帮助文档

makefile 帮助文档：<http://www.gnu.org/software/make/manual/make.html>




著作权声明

版权所有 © 2022 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明

、 **全志科技** （不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。