



# **D1-H Tina Linux 系统软件 开发指南**

**版本号: 0.3**  
**发布日期: 2021.04.19**

## 版本历史

版本号	日期	制/修订人	内容描述
0.1	2021.03.02	AWA1610	初始版本，D1-H 平台 Tina Linux 构建描述
0.2	2021.03.16	AWA1610	更新部分图片，统一序号
0.3	2021.04.19	AWA0985	更新适配 D1-H 部分信息



# 目 录

<b>1</b>	<b>概述</b>	<b>1</b>
1.1	编写目的	1
1.2	适用范围	1
1.3	相关人员	1
<b>2</b>	<b>Tina 系统资料</b>	<b>2</b>
2.1	概述	2
2.2	文档列表	2
<b>3</b>	<b>Tina 系统概述</b>	<b>3</b>
3.1	概述	3
3.2	系统框图	3
3.3	开发流程	4
<b>4</b>	<b>Tina 开发环境</b>	<b>5</b>
4.1	概述	5
4.2	编译环境搭建	5
4.2.1	开发主机配置	5
4.2.2	软件包配置	5
<b>5</b>	<b>Tina 系统获取</b>	<b>7</b>
5.1	概述	7
5.2	SDK 获取	7
5.3	SDK 结构	7
5.3.1	build 目录	7
5.3.2	config 目录	8
5.3.3	devices 目录	8
5.3.4	lichee 目录	9
5.3.5	package 目录	10
5.3.6	prebuilt 目录	10
5.3.7	scripts 目录	10
5.3.8	target 目录	11
5.3.9	toolchain 目录	11
5.3.10	tools 目录	11
5.3.11	out 目录	11
<b>6</b>	<b>Tina 编译打包</b>	<b>13</b>
6.1	概述	13
6.2	编译系统	13
6.3	编译 boot	13
6.4	编译内核	13
6.5	重编应用	14
6.5.1	方法一	14

6.5.2 方法二	14
6.6 其他命令	14
<b>7 Tina 系统烧写</b>	<b>16</b>
7.1 概述	16
7.2 烧录工具	16
7.3 进入烧录模式	16
<b>8 Tina uboot 定制开发</b>	<b>18</b>
8.1 概述	18
8.2 代码路径	18
8.3 uboot 功能	18
8.4 uboot 配置	18
8.4.1 defconfig 方式	19
8.4.1.1 defconfig 配置步骤	19
8.4.1.2 defconfig 配置宏介绍	19
8.4.2 menuconfig 方式	20
8.5 uboot 编译	21
8.5.1 方法一	21
8.5.2 方法二	22
8.6 uboot 的配置文件	22
8.6.1 sys_config 配置	22
8.6.1.1 sys_config.fex 结构介绍	22
8.6.1.2 sys_config.fex 配置实例	23
8.6.2 环境变量配置	23
8.6.2.1 环境变量作用	23
8.6.2.2 环境变量配置示例介绍	24
8.6.3 sys_partition.fex 分区配置	25
8.6.3.1 sys_partition.fex 分区配置介绍	25
<b>9 Tina kernel 定制开发</b>	<b>26</b>
9.1 概述	26
9.2 代码路径	26
9.3 模块开发文档	26
9.4 内核配置	26
<b>10 Tina 系统定制开发</b>	<b>28</b>
10.1 应用移植	28
10.1.1 Makefile 范例	28
10.1.2 自启动设置	30
10.1.2.1 调用自启动脚本	30
10.1.2.2 sysV 格式脚本	30
10.1.2.3 procd 格式脚本	31
10.2 应用调试	31

10.3 应用编译	33
10.4 应用安装	33
10.5 分区与挂载	33



## 插 图

3-1 Tina Linux 系统框图 . . . . .	3
3-2 Tina Linux 系统开发流程 . . . . .	4
8-1 defconfig 基本宏定义介绍图 . . . . .	20
8-2 TinaLinux_uboot_2018_menuconfig . . . . .	21
8-3 sysconfig.fex 基本结构图 . . . . .	22
8-4 uart_para 配置图 . . . . .	23
8-5 uboot 启动调用环境变量方式图 . . . . .	24
8-6 kernel cmdline 图 . . . . .	24
9-1 TinaLinux 内核配置菜单 . . . . .	27
10-1 应用配置主界面 . . . . .	32
10-2 软件包所在界面 . . . . .	32



# 1 概述

---

## 1.1 编写目的

本文档作为 Allwinner Tina Linux 系统平台开发指南，旨在帮助软件开发工程师、技术支持工程师快速上手，熟悉 Tina Linux 系统的开发及调试流程。

## 1.2 适用范围

本文适用于 D1-H（RSIC-V）平台开发

## 1.3 相关人员

本开发指南适用于 Tina 系统软件开发工程师、Tina 系统技术支持工程师，以及所有需要构建 Tina Linux 平台的相关人员。

## 2 Tina 系统资料

---

### 2.1 概述

Tina SDK 发布的文档旨在帮助开发者快速上手开发及调试，文档中涉及的内容并不能涵盖所有的开发知识和问题。文档列表也正在不断更新。

Tina SDK 提供丰富的文档资料，包括硬件参考设计文档、Flash 等基础器件支持列表、量产工具使用说明、软件开发与制定介绍文档、芯片研发手册等资料。

### 2.2 文档列表

请以全志客户服务平台最新列表为准。





## 3 Tina 系统概述

### 3.1 概述

Tina Linux 系统是基于 openwrt-14.07 的版本的软件开发包，包含了 Linux 系统开发用到的内核源码，驱动，工具、系统中间件与应用程序包。openwrt 是一个开源的嵌入式 Linux 系统自动构建框架，是由 Makefile 脚本和 Kconfig 配置文件构成的。使得用户可以通过 menuconfig 配置，编译出一个完整的可以直接烧写到机器上运行的 Linux 系统软件。

### 3.2 系统框图

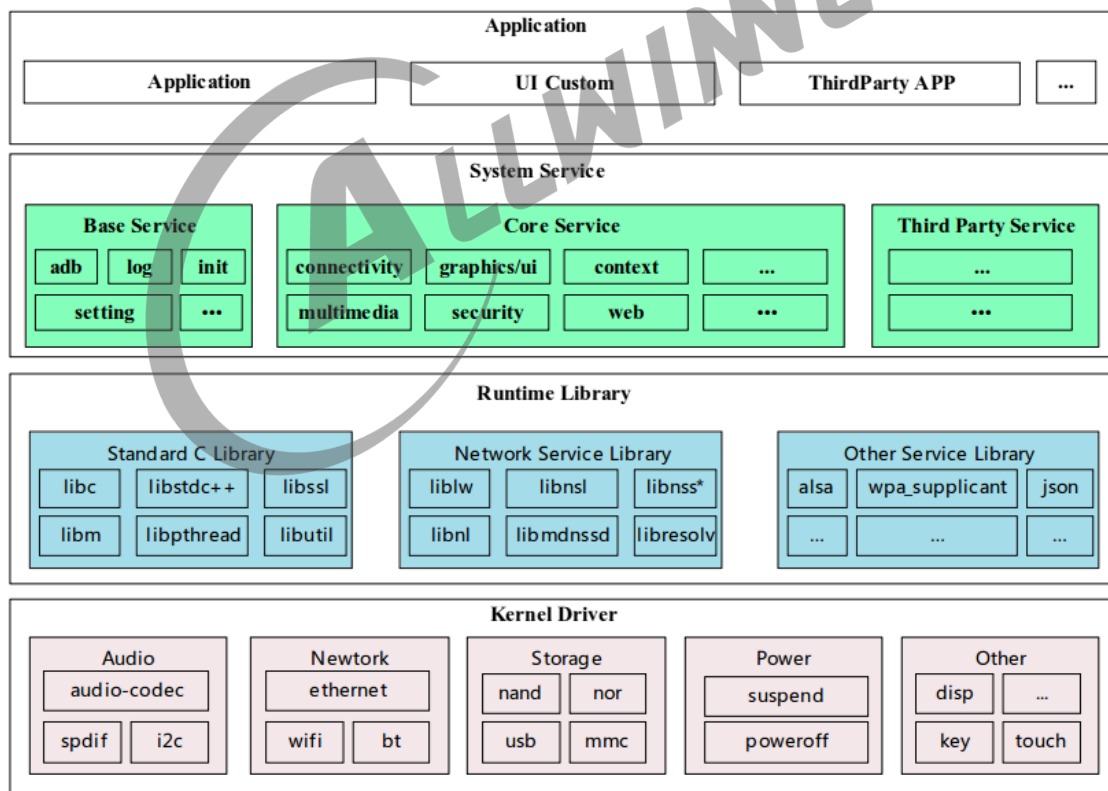


图 3-1: Tina Linux 系统框图

Tina 系统软件框图如图所示，从下至上分为 Kernel & Driver、Libraries、System Services、Applications 四个层次。各层次内容如下：

1. Kernel&&Driver 主要提供 Linux Kernel 的标准实现。本文所述的 D1-H 平台，Tina Linux Kernel 采用 Linux5.4 内核。主要提供安全性，内存管理，进程管理，网络协议栈等基础支持；主要是通过 Linux 内核管理设备硬件资源，如 CPU 调度、缓存、内存、I/O 等。
2. Libraries 层对应一般嵌入式系统，相当于中间件层次。包含了各种系统基础库，及第三方开源程序库支持，对应用层提供 API 接口，系统定制者和应用开发者可以基于 Libraries 层的 API 开发新的应用。
3. System Services 层对应系统服务层，包含系统启动管理、配置管理、热插拔管理、存储管理、多媒体中间件等。
4. Applications 层主要是实现具体的产品功能及交互逻辑，需要一些系统基础库及第三方案程序库支持，开发者可以开发实现自己的应用程序，提供系统各种能力给到最终用户。

### 3.3 开发流程

Tina Linux 系统是基于 Linux Kernel，针对多种不同产品形态开发的 SDK。可以基于本 SDK，有效地实现系统定制和应用移植开发。



图 3-2: Tina Linux 系统开发流程

如上图所示，开发者可以遵循上述开发流程，在本地快速构建 Tina Linux 系统的开发环境和编译代码。下面将简单介绍下该流程：

（1）. 检查系统需求：在下载代码和编译前，需确保本地的开发设备能够满足需求，包括机器的硬件能力，软件系统，工具链等。目前 Tina Linux 系统只支持 Ubuntu 操作系统环境下编译，并仅提供 Linux 环境下的工具链支持，其他如 MacOS，Windows 等系统暂不支持。（2）. 搭建编译环境：开发机器需要安装的各种软件包和工具，详见开发环境章节，获知 TinaLinux 已经验证过的操作系统版本，编译时依赖的库文件等。（3）. 选择设备：在编译源码前，开发者需要先导出预定义环境变量，然后根据开发者根据的需求，选择对应的硬件板型，详见编译章节。（4）. 系统定制：开发者可以根据使用的硬件板子、产品定义，定制 U-Boot、Kernel 及 Openwrt，请参考后续章节中相关开发指南和配置的描述。（5）. 编译与打包：完成设备选择、系统定制之后执行编译命令，包括整体或模块编译以及编译清理等工作，进一步的，将生成的 boot/内核二进制文件、根文件系统、按照一定格式打包成固件。详见编译打包章节。（6）. 烧录并运行：继生成镜像文件后，将介绍如何烧录镜像并运行在硬件设备，进一步内容详见系统烧写章节。

## 4 Tina 开发环境

### 4.1 概述

嵌入式产品开发流程中，通常有两个关键的步骤，编译源码与烧写固件。源码编译需要先准备好编译环境，而固件烧写则需要厂家提供专用烧写工具。本章主要讲述这如何搭建环境来实现 Tina SDK 的编译、烧写。

### 4.2 编译环境搭建

一个典型的嵌入式开发环境包括本地开发主机和目标硬件板。

- 本地开发主机作为编译服务器，需要提供 linux 操作环境，建立交叉编译环境，为软件开发提供代码更新下载，代码交叉编译服务。
- 本地开发主机通过串口或 USB 与目标硬件板连接，可将编译后的镜像文件烧写到目标硬件板，并调试系统或应用程序。

#### 4.2.1 开发主机配置

Tina Linux SDK 是在 ubuntu14.04 开发测试的，因此我们**推荐使用 Ubuntu 14.04**主机环境进行源码编译，其他版本没有具体测试，可能需要对软件包做相应调整。

#### 4.2.2 软件包配置

编译 Tina Linux SDK 之前，需要先确定编译服务器安装了 gcc, binutils, bzip2, flex, python, perl, make, ia32-libs, find, grep, diff, unzip, gawk, getopt, subversion, libz-dev, libc headers。

ubuntu 可直接执行以下命令安装：

```
sudo apt-get install build-essential subversion git-core libncurses5-dev zlib1g-dev gawk  
flex quilt libssl-dev xsltproc libxml-parser-perl mercurial bzip2 ecj cvs unzip ia32-libs  
lib32z1 lib32z1-dev lib32stdc++6 libstdc++6 -y
```

对于 Ubuntu 16.04 以上版本，部分软件包已不再提供或者采用了其他的包，执行上述命令时，安装失败的包可先忽略，进一步执行以下命令

```
sudo apt-get install libc6:i386 libstdc++6:i386 lib32ncurses5 lib32z1
```



## 5 Tina 系统获取

### 5.1 概述

### 5.2 SDK 获取

Allwinner Tina Linux SDK 通过全志代码服务器对外发布。客户需要向业务/技术支持窗口申请 SDK 下载权限。申请需同步提供 SSH 公钥进行服务器认证授权，获得授权后即可同步代码。

### 5.3 SDK 结构

Tina Linux SDK 主要由构建系统、配置工具、工具链、工具包、目标设备应用程序、文档、脚本、linux 内核、bootloader 部分组成，下文按照目录顺序介绍相关的组成组件。

```
Tina-SDK/  
├─ build  
├─ config  
├─ Config.in  
├─ device  
├─ dl  
├─ lichee  
├─ Makefile  
├─ package  
├─ prebuilt  
├─ rules.mk  
├─ scripts  
├─ target  
├─ toolchain  
└─ tools
```

#### 5.3.1 build 目录

build 目录存放 Tina Linux 的构建系统文件，此目录结构下主要是一系列基于 Makefile 规格编写的 mk 文件。主要的功能是：

(1) . 检测当前的编译环境是否满足 Tina Linux 的构建需求。(2) . 生成 host 包 (PC 端软件包) 编译规则。(3) . 生成工具链的编译规则。(4) . 生成 target 包的编译规则。(5) . 生成 linux kernel 的编译规则。(6) . 生成系统固件的生成规则。

```
build/
├─ autotools.mk
├─ aw-upgrade.mk
├─ board.mk
├─ cmake.mk
├─ config.mk
├─ debug.mk
├─ depends.mk
├─ device.mk
├─ device_table.txt
├─ download.mk
├─ dumpvar.mk
├─ envsetup.sh
└─ .....
```

## 5.3.2 config 目录

config 目录主要存放 Tina Linux 中配置菜单的界面以及一些固定的配置项，该配置菜单基于内核的 mconf 规格书写。

```
config/
├─ Config-build.in
├─ Config-devel.in
├─ Config-images.in
├─ Config-kernel.in
├─ Config-systeminit.in
└─ top_config.in
```

## 5.3.3 devices 目录

devices 目录用于存放方案的配置文件，包括内核配置，env 配置，分区表配置，sys\_config.fex，board.dts 等。

```
device/
├─ config
│   ├── chips
│   │   └─ d1-h
│   └─ common
│       ├── cert
│       ├── debug
│       ├── dtb
│       ├── hdcp
│       ├── imagecfg
│       ├── partition
│       ├── sign_config
│       ├── toc
│       ├── tools
│       └─ version
```

其中，config/chips/d1-h 存放 D1-H 平台相关的配置，其目录结构如下：

```
d1-h
├─ bin
├─ boot-resource
│   └─ boot-resource
│       └─ bat
├─ configs
│   ├── default
│   └─ nezha
│       ├── linux -> linux-5.4
│       └─ linux-5.4
└─ tools
```

- bin 目录存放编译 boot 等 bin 文件，当 Tina SDK 构建或重新编译 boot 时，对应的文件会被替换。快捷跳转命令：cbin。
- boot-resource 目录存放开机动画等资源。
- tools 目录存放方案构建时需要的工具
- configs 目录存放该 CHIP 平台对应的多个硬件方案配置文件。其中，default 为公共配置，nezha 对应硬件 nezha 板的方案配置，若存在更多个硬件方案，便会在该目录下新建对应的方案目录。若公共配置目录 default 和方案配置目录中，存在相同的配置文件时，优先使用方案配置。快捷跳转命令：cconfigs（该命令会跳转到该目录下 linux 目录）。

以 nezha 方案为例，简述方案配置目录下，具体内容：

```
d1-h/configs/nezha/
├─ board.dts -> linux-5.4/board.dts
├─ env.cfg
├─ linux -> linux-5.4
├─ linux-5.4
│   ├── board.dts
│   └─ config-5.4
├─ sys_config.fex
└─ sys_partition.fex
```

board.dts 板级 dts 配置文件，符合 linux 内核 dts 配置格式及合并规则。

env.cfg 环境变量配置文件，Uboot 将此环境变量传递给内核。

linux/config-5.4 Linux5.4 内核配置文件，配置方案下默认 linux 内核功能。

sys\_config.fex 打包阶段根据 sys\_config 配置更新 boot0, uboot, optee 等 bin 文件的头部等信息，例如更新 dram 参数、uart 参数等。

sys\_partition.fex 分区配置文件。

### 5.3.4 lichee 目录

lichee 目录主要存放 bootloader，内核等代码。

```
lichee/  
├─ brandy-2.0  
│   └─ tools  
│       └─ u-boot-2018  
└─ linux-5.4
```

快捷跳转命令：ckernel, cboot。

### 5.3.5 package 目录

package 目录存放目标 (target) 机器上的软件包源码和编译规则，目录按照目标软件包的功能进行分类。

```
package/  
├─ add-rootfs-demo  
├─ admin  
├─ allwinner  
│   ...  
├─ utils  
└─ wayland
```

### 5.3.6 prebuilt 目录

prebuild 目录存放预编译交叉编译器，目录结构如下。gcc/riscv 即为编译 D1-H 所用的工具链目录

```
prebuilt/  
├─ gcc  
│   └─ linux-x86  
│       ├── host  
│       └─ riscv  
│           └─ toolchain-thead-glibc
```

### 5.3.7 scripts 目录

scripts 目录用于存放一些构建编译相关的脚本。

一般指定解释器为#!/bin/bash的脚本是 host#!/bin/sh的脚本是 target 端工具。

```
scripts/  
├─ add_initramfs.sh  
├─ arm-magic.sh  
└─ ...
```



### 5.3.8 target 目录

target 目录用于存放目标板相关的配置以及 sdk 和 toolchain 生成的规格。

```
target/
├── allwinner
├── Config.in
├── imagebuilder
├── Makefile
├── sdk
└── toolchain
```

快捷跳转命令：cdevice。

### 5.3.9 toolchain 目录

toolchain 目录包含交叉工具链构建配置、规则。

```
toolchain/
├── binutils
├── fortify-headers
├── gcc
├── gdb
├── glibc
├── insight
├── kernel-headers
├── musl
└── wrapper
```

### 5.3.10 tools 目录

tools 目录用于存放 host 端工具的编译规则。

```
tools/
├── autoconf
├── automake
├── aw_tools
├── b43-tools
└── .....
```

### 5.3.11 out 目录

out 目录用于保存编译相关的临时文件和最终镜像文件，编译后自动生成此目录，以编译 d1-h-nezha 方案为例说明。

```
out/  
├─ d1-h-nezha  
└─ host
```

其中，host 目录用于存放 host 端的工具以及一些开发相关的文件。

d1-h-nezha 目录为方案对应的目录。方案目录下的结构如下：

```
out/d1-h-nezha/  
├─ boot.img  
├─ compile_dir  
├─ d1-h-nezha-boot.img  
├─ d1-h-nezha-Image  
├─ d1-h-nezha-uImage  
├─ image  
├─ md5sums  
├─ packages  
├─ rootfs.img  
├─ sha256sums  
├─ staging_dir  
├─ tina_d1-h-nezha_uart0.img  
└─ usr.img
```

- tina\_d1-h-nezha\_uart0.img 为最终固件包（系统镜像），串口信息通过串口输出。若使用 pack -d，则生成的固件包为 xxx\_card0.img，串口信息转递到 tf 卡座输出。
- boot.img 为最终烧写到系统 boot 分区的数据，可能为 boot.img 格式也可能为 uImage 格式。
- rootfs.img 为最终烧写到系统 rootfs 分区的数据，该分区默认为 squashfs 格式。
- d1-h-nezha-Image 为内核的 Image 格式镜像，用于进一步生成 uImage。
- d1-h-nezha-uImage 为内核的 uImage 格式镜像，若配置为 uImage 格式，则会拷贝成 boot.img。
- d1-h-nezha-boot.img 为内核的 boot.img 格式镜像，若配置为 boot.img 格式，则会拷贝成 boot.img。
- compile\_dir 为 sdk 编译 host，target 和 toolchain 的临时文件目录，存有各个软件包的源码。
- staging\_dir 为 sdk 编译过程中保存各个目录结果的目录。
- packages 目录保存的是最终生成的 ipk 软件包。

快捷跳转命令：cout。

## 6 Tina 编译打包

### 6.1 概述

### 6.2 编译系统

```
1. source build/envsetup.sh
2. lunch
3. make [-jN]
4. pack [-d]
```

其中，

步骤1 建立编译环境，导出编译变量。

步骤2 提示需要选择你想要编译的方案。

步骤3 参数N为并行编译进程数量，依赖编译服务器CPU核心数，如4核PC，可"make -j4"

步骤4 打包固件，-d参数使生成固件包串口信息转到tf卡座输出。

编译完成后系统镜像会打包在out/<board>/目录下

### 6.3 编译 boot

命令	命令有效目录	作用
mboot	tina 下任意目录	编译 boot0 和 uboot
mboot0	tina 下任意目录	编译 boot0
muboot	tina 下任意目录	编译 uboot

### 6.4 编译内核

命令	命令有效目录	作用
mkkernel	tina 下任意目录	编译内核

## 6.5 重编应用

请确保进行过一次固件的编译，确保 SDK 基础已经编译，才能单独重编应用包。重编应用包应用场景一般为：**只修改了应用，不想重新烧写固件，只需要安装应用安装包即可**。请确保在编译前已加载 tina 环境：

```
$ source build/envsetup.sh
$ lunch
```

### 6.5.1 方法一

当在应用包的目录（包括其子目录）中，可执行

```
$ mm [-B]
=> B参数则先clean此应用临时文件再编译
```

示例：假设软件包路径为：tina/package/Utils/rwcheck，则：

```
$ cd tina/package/Utils/rwcheck
$ mm -B
```

编译出应用安装包保存路径为：

```
tina/out/<方案>/packages/base
```

### 6.5.2 方法二

当在 tina 的根目录，可执行：

```
$ make <应用包的路径>/clean, ==>清空应用包临时文件
$ make <应用包的路径>/install, ==>编译软件包
或者
$ make <应用包的路径>/{clean, install}, ==>先清空临时文件再编译
```

示例：假设软件包的路径为：tina/package/Utils/rwcheck，则：

```
$ cd tina
$ make package/Utils/rwcheck/{clean, install}
```

## 6.6 其他命令

命令	命令有效目录	作用
make	tina 根目录	编译整个 sdk
make menuconfig	tina 根目录	启动软件包配置界面
make kernel_menuconfig	tina 根目录	启动内核配置界面
croot	tina 下任意目录	快速切换到 tina 根目录
cconfigs	tina 下任意目录	快速切换到方案的 bsp 配置目录
cdevice	tina 下任意目录	快速切换到方案配置目录
cgeneric	tina 下任意目录	快速切换到方案 generic 目录
cout	tina 下任意目录	快速切换到方案的输出目录
cboot	tina 下任意目录	快速切换到 bootloader 目录
cgrep	tina 下任意目录	在 c / c++ / h 文件中查找字符串
mininstall path/to/package/	tina 根目录	编译并安装软件包
mclean path/to/package/	tina 根目录	clean 软件包
mm [-B]	软件包目录	编译软件包，-B 指编译前先 clean
pack	tina 根目录	打包固件
m	tina 下任意目录	make 的快捷命令，编译整个 sdk
p	tina 下任意目录	pack 的快捷命令，打包固件

## 7 Tina 系统烧写

### 7.1 概述

本章节主要介绍如何将构建完成的镜像文件 (image) 烧写并运行在硬件设备上的流程。

SDK 中的烧录工具不再更新，后续会删除，请优先选择从全志客户服务平台下载最新烧录工具。

windows 工具均集成在 APST 中，下载安装 APST 即可，APST 的工具均自带文档。

### 7.2 烧录工具

Tina 提供的几种镜像烧写工具介绍如表所示，用户可以选择合适的烧写方式进行烧写。

工具	运行系统	描述
PhoenixSuit	windows	分分区升级及整个固件升级工具
PhoenixCard	windows	卡固件制作工具
PhoenixUSBpro	windows	量产升级工具，支持 USB 一拖 8 烧录
LiveSuit	ubuntu	固件升级工具

对于 ubuntu：

- 64bit 主机使用 LiveSuitV306\_For\_Linux64.zip。
- 32bit 主机使用 LiveSuitV306\_For\_Linux32.zip。

具体烧录工具和使用说明，请到全志客户服务平台下载。

### 7.3 进入烧录模式

设备需进入烧录模式，以下几种情况会进入烧录模式：

1. BROM 无法读取到 boot0，例如新换的 flash 不包含数据，或者上电时短路 flash 阻断通信。

2. 在串口中按 2 进入烧录。即，在串口工具输出框中，按住键盘的'2'，不停输出字符'2'，上电启动。boot0 检测到此字符，会跳到烧录模式。
3. 在 uboot 控制台，执行 efex。
4. 在 linux 控制台，执行 reboot efex。
5. adb 可用的情况下，可使用 adb shell reboot efex，或点击烧录工具上的“立即烧录”按钮。
6. 当板子有 FEL 按键时，按住 FEL 按键上电。
7. 制作特殊的启动卡，从卡启动再进入烧录模式。



## 8 Tina uboot 定制开发

### 8.1 概述

本章节简单介绍 uboot 基本配置、功能裁剪、编译打包、常用命令的使用，帮助客户了解 Tina 平台 uboot 框架，为 boot 定制开发提供基础。

Tina SDK 中，D1-H 平台使用版本是 uboot-2018，放在路径 lichee/ 目录下。

### 8.2 代码路径

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

### 8.3 uboot 功能

TinaSDK 中，bootloader/uboot 在内核运行之前运行，可以初始化硬件设备、建立内存空间映射图，从而将系统的软硬件环境带到一个合适状态，为最终调用 linux 内核准备好正确的环境。在 Tina 系统平台中，除了必须的引导系统启动功能外，uboot 还提供烧写、升级等其它功能。

- 引导内核能从存储介质（nand/mmc/spinor）上加载内核镜像到 DRAM 指定位置并运行。
- 量产 & 升级包括卡量产，USB 量产，私有数据烧录，固件升级。
- 开机提示信息开机能显示启动 logo 图片 (BMP 格式)。
- Fastboot 功能实现 fastboot 的标准命令，能使用 fastboot 刷机。

### 8.4 uboot 配置

以 uboot-2018 为例，各项功能可以通过 defconfig 或配置菜单 menuconfig 进行开启或关闭，具体配置方法如下：



## 8.4.1 defconfig 方式

defconfig 是 mconf 工具的配置文件。在 Tina SDK 中存在 uboot, kernel, tina 等多种 defconfig, 本节介绍 uboot 的 defconfig。

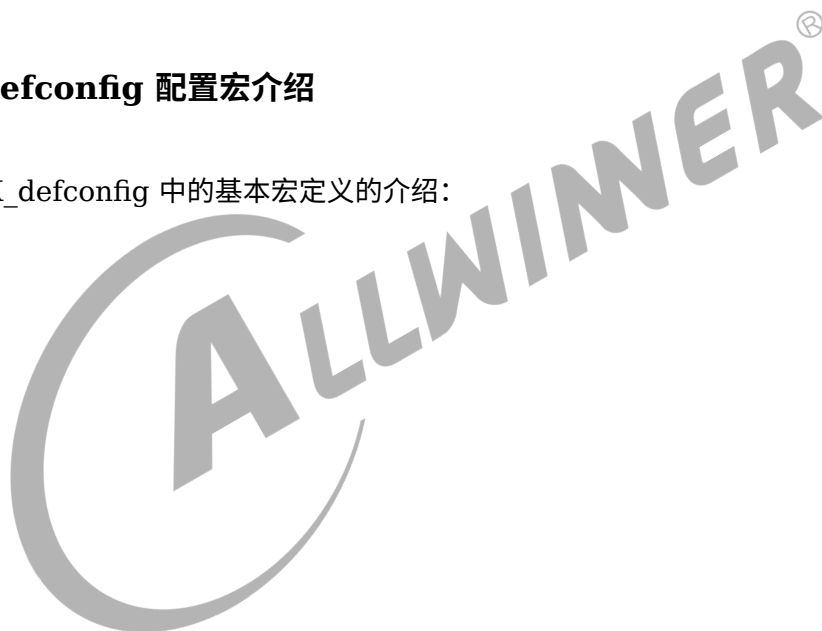
uboot 下的 deconfig 存放在 `${TinaSDK}/lichee/brandy2.0/u-boot-2018/configs/` 目录下, 如 D1-H 的配置文件为 `sun20iw1p1_defconfig` (若是 spinor 方案, 则为 `sun20iw1p1_nor_defconfig`)。

### 8.4.1.1 defconfig 配置步骤

1. `vim ${TinaSDK}/lichee/brandy2.0/u-boot-2018/configs/sun20iw1p1_defconfig`
2. 打开 `sun20iw1p1_defconfig/sun20iw1p1_nor_defconfig` 后, 在相应的宏定义前去掉或添加 `"#"` 即可将相应功能开启或关闭。

### 8.4.1.2 defconfig 配置宏介绍

如下图是 XXX\_defconfig 中的基本宏定义的介绍:



宏定义	宏定义功能说明	宏定义默认状态 (开启/关闭)
CONFIG_SUNXI_NAND	支持 nand 驱动的开关	(sun8iw18p1_nor_defconfig 中默认关闭, sun8iw18p1_defconfig 中默认开启)
CONFIG_SUNXI_SPINOR	支持 spinor 驱动的开关	开启
CONFIG_SUNXI_USB	支持 usb 驱动的开关	开启
CONFIG_SUNXI_EFEX	支持 usb 烧录功能的开关	开启
CONFIG_SUNXI_BURN	支持烧 key 功能的开关	开启
CONFIG_SUNXI_FASTBOOT	支持 fastboot 功能的开关	开启
CONFIG_EFI_PARTITION	支持 gpt 功能的开关	开启
CONFIG_ANDROID_BOOT_IMAGE	支持 android 固件启动功能的开关	开启
CONFIG_SUNXI_SFRITE	支持量产功能的总开关	开启 <sup>②</sup>
CONFIG_SUNXI_SECURE_STORAGE	支持安全存储的开关	开启
CONFIG_SUNXI_SECURE_BOOT	支持安全启动的开关	开启
CONFIG_SUNXI_KEYBOX	支持信用链的开关	开启
CONFIG_CMD_SUNXI_EFEX	shell 命令支持跳烧写命令的开关	开启
CONFIG_CMD_SUNXI_BURN	shell 命令支持烧 key 命令的开关	开启
CONFIG_CMD_GPT	shell 命令支持 gpt 命令的开关	开启
CONFIG_CMD_FAT	shell 命令支持 fat 命令的开关	开启
CONFIG_CMD_FASTBOOT	shell 命令支持 fastboot 命令的开关	开启
CONFIG_CMD_SUNXI_DMA	shell 命令支持 dma 测试命令的开关	开启
CONFIG_CMD_PART	shell 命令支持查看分区信息命令的开关	开启
CONFIG_CMD_SF	shell 命令支持 spi flash 读写命令的开关	开启

图 8-1: defconfig 基本宏定义介绍图

## 8.4.2 menuconfig 方式

通过 menuconfig 方式配置的方法步骤如下

```
cd /TinaSDK/lichee/brandy2.0/u-boot-2018/  
make sun20iwlpl_defconfig # 主要用于生成.config, 若刚编译过该平台, 则可以跳过此步骤。  
make menuconfig
```

执行上述命令会弹出 menuconfig 配置菜单, 如下图所示, 此时即可对各模块功能进行配置, 配置方法 menuconfig 配置菜单窗口中有说明。

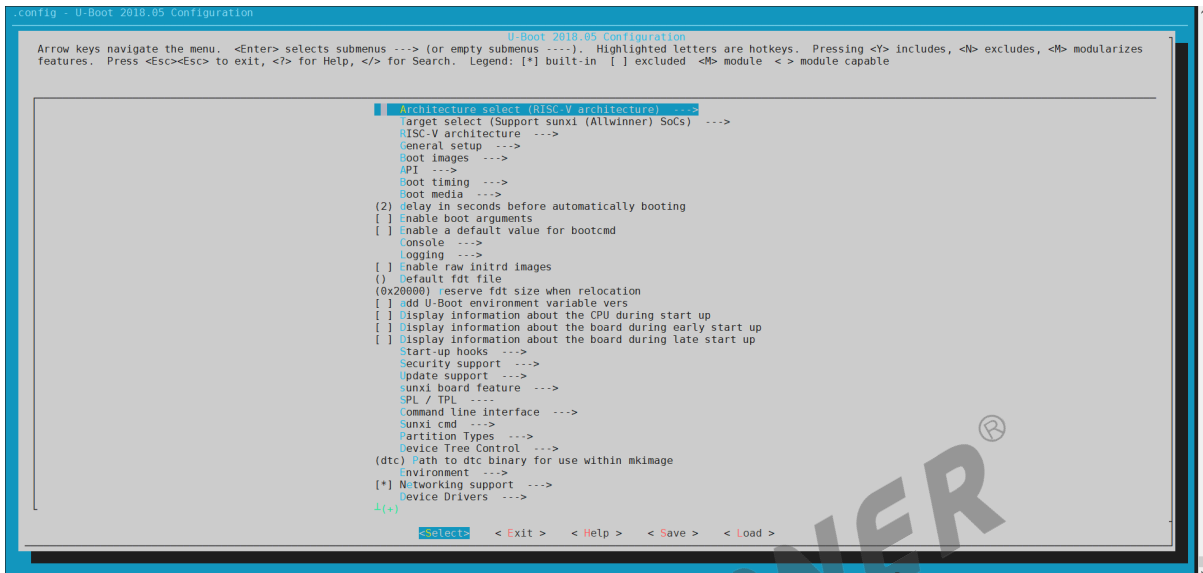


图 8-2: TinaLinux\_uboot\_2018\_menuconfig

## 8.5 uboot 编译

### 8.5.1 方法一

在 tina 目录下, 即可编译 uboot。

```
source build/envsetup.sh(见详注1)  
lunch (见详注2)  
muboot(见详注3)  
详注:  
1 加载环境变量及tina提供的命令。  
2 输入编号, 选择方案。  
3 编译uboot, 编译完成后自动更新uboot binary到TinaSDK/target/allwinner/$(BOARD)-common/bin/。  
4 采用此种方法时, 使用make menuconfig修改的配置无效, 因为muboot命令会使用过deconfig默认配置覆盖.config。  
。
```

## 8.5.2 方法二

```
source build/envsetup.sh(见详注1)
lunch (见详注2)
cboot(见详注3)
make XXX_config(见详注4)
make menuconfig (可跳过此步)
make -j
详注：
1 加载环境变量及tina提供的命令。
2 输入编号，选择方案。
3 跳转到Uboot源码目录。
4 选择方案配置，如果是使用norflash，运行make XXX_nor_config。
5 执行编译uboot的动作。
```

## 8.6 uboot 的配置文件

### 8.6.1 sys\_config 配置

sys\_config.fex 的作用主要是：打包阶段根据 sys\_config 配置更新 boot0, uboot, optee 等 bin 文件的头部等信息，例如更新 dram 参数、uart 参数等其文档存放路径：

```
TinaSDK/device/config/chips/$(CHIP)/configs/$(BOARD)/sys_config.fex
```

#### 8.6.1.1 sys\_config.fex 结构介绍

sys\_config.fes 主要由主键和子键构成，主键是某项功能或模块的主标识，由 [] 括起，子键是对该功能或模块中各个参数的配置项，如下图所示，dram\_para 是主键，dram\_clk、dram\_type 和 dram\_zp 是子键。

```
*****
;dram select configuration
;select_mode      :      dram模式选择,      0:不进行自动识别
;                  1:gpio识别模式(dram_para, dram_para1-15, 共16组有效)
;                  2:gpadc识别模式(dram_para, dram_para1-7, 共8组有效)
;                  3:1个IO+gpadc识别模式(dram_para, dram_para1-15, 共16组有效)。
;                  其中IO配置优先级按select_gpio0>select_gpio1>select_gpio2>select_gpio3
;                  有效值(0-3)
;gpadc_channel    :      选择gpadc通道
;select_gpio1-4   :      选择gpio pin
*****

[dram_select_para]
select_mode       = 0
gpadc_channel     = 1
select_gpio0      = port:PB7<0><1><default><default>
select_gpio1      = port:PB4<0><1><default><default>
select_gpio2      = port:PH1<0><1><default><default>
select_gpio3      = port:PH0<0><1><default><default>
```

图 8-3: sysconfig.fex 基本结构图

(上图仅供参考，请以实际工程及模块说明文档为准)。

### 8.6.1.2 sys\_config.fex 配置实例

[uart\_para]: 串口配置项，uart\_para 配置项是 uboot 串口打印调试时用到的重要配置

```
-----  
;uart configuration  
;uart_type --- 2 (2 wire), 4 (4 wire), 8 (8 wire, full function)  
-----  
[uart_para]  
uart_debug_port = 0  
uart_debug_tx   = port:PF02<3><1><default><default>  
uart_debug_rx   = port:PF04<3><1><default><default>
```

图 8-4: uart\_para 配置图

上图中的 uart\_debug\_port=0 表示使用的是 uart0，uart\_debug\_tx/uart\_debug\_rx 配置的 gpio 口（PF02/PF04）需要根据对应的 GPIO DATASHEET 进行配置。

(上图仅供参考，请以实际工程及模块说明文档为准)。

### 8.6.2 环境变量配置

uboot 的环境变量就是一个个的键值对，操作接口为：getenv(), setenv(), saveenv()。环境变量的形式：

```
boot_normal=sunxi_flash read 40007800 boot;boota 4000780\  
boot_recovery=sunxi_flash read 40007800 recovery;boota 4000780\  
boot_fastboot= fastboot
```

#### 8.6.2.1 环境变量作用

可以把一些参数信息或者命令序列定义在该环境变量中。在环境变量中定义 UBOOT 命令序列，可以把 UBOOT 各个功能模块按顺序组合在一起执行，从而完成某个重要功能。

例如，如果执行了上述提到的 boot\_normal 环境变量对应的命令，Uboot 则会先调用 sunxi\_flash 命令从存储介质的 boot 分区上加载内核到 DRAM 的 0x40007800 位置；然后调用 boota 命令完成内核的引导。

uboot 启动时调用环境变量方式下如图所示：

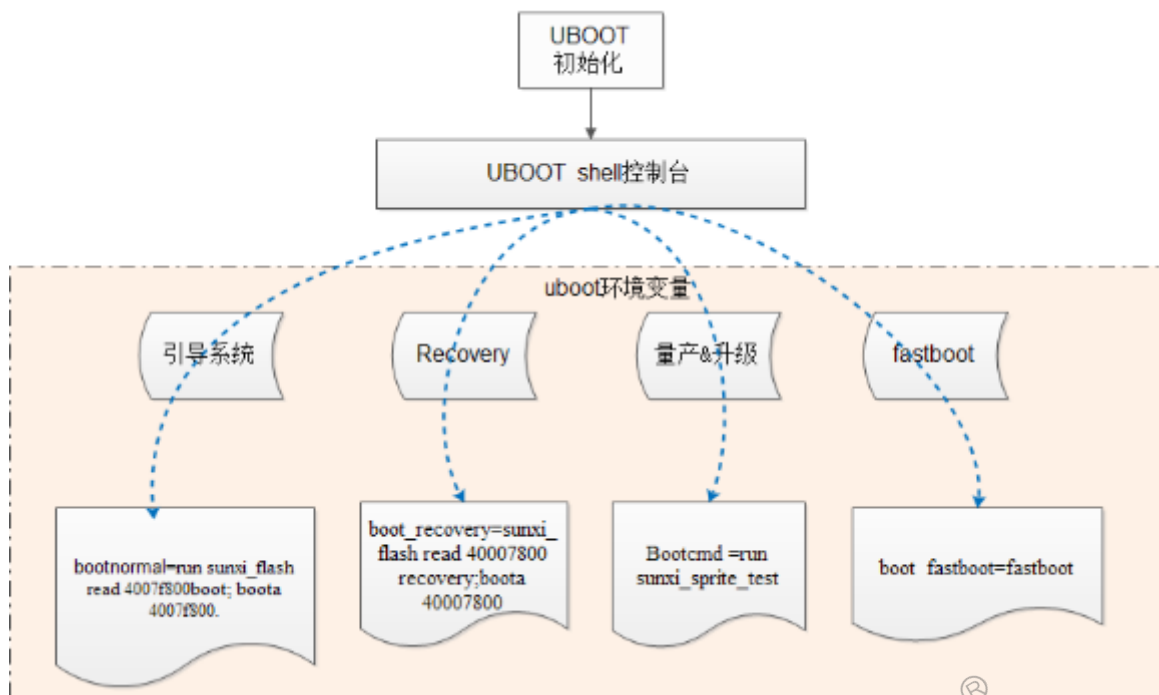


图 8-5: uboot 启动调用环境变量方式图

### 8.6.2.2 环境变量配置示例介绍

TinaSDK 中，环境变量配置文件保存在 TinaSDK/target/allwinner/\$(BOARD)/configs/env.cfg 文件，用户使用的时候，可能会看到 env-4.4.cfg、env-4.9.cfg 等文件，env-xxx 后缀数字表示在不同内核版本上的配置。打开后其内容示例如下，

- bootdelay=0，改环境变量 bootdelay（即 boot 启动时 log 中的倒计时延迟时间）值的大小，为便于调试，bootdelay 的值一般不要等于 0，这样在小机上电后按下任意键才能进入 uboot shell 命令状态。
- boot\_normal=sunxi\_flash read 40007800 boot;boota 4000780，设置启动内核命令，即将 boot 分区读到内存 0x40007800 地址处，然后从内存 0x40007800 地址处启动内核。
- Setargs\_nand=setenv bootargs earlyprintk=\${earlyprintk}.....，设置内核相关环境变量，该变量在启动至内核的 log 中会打印处理，即 cmdline 如下图：

```
[ 0.000000] Kernel command line: earlyprintk=sunxi-uart,0x05000c00 initcall_debug=0 console=ttyS0,115200 log
level=8 root=/dev/mmcblk0p7 rootwait init=/init partitions=boot-resource@mmcblk0p2:env@mmcblk0p5:boot@mmcblk0p6
:rootfs@mmcblk0p7:UDISK@mmcblk0p1 cma=256M androidboot.mode=normal androidboot.hardware=sun50iw6p1 boot_type=2
```

图 8-6: kernel cmdline 图

(上图仅供参考，请以实际工程及模块说明文档为准)。

- loglevel=8，设置内核 log 打印等级。

## 8.6.3 sys\_partition.fex 分区配置

分区配置文件是一个规划磁盘分区的文件，烧录过程会按照该分区配置文件将各分区数据烧录至 flash 中。

TinaSDK 中，分区配置文件路径 TinaSDK/target/allwinner/\$(BOARD)/configs/sys\_partition.fex。有些方案可以看到 sys\_partition.fex、sys\_partition\_nor.fex 两个分区配置文件，若是打包 Tina 非 nor 固件，则使用的是 sys\_partition\_linux.fex 配置文件，若是打包 nor 固件，则使用的是 sys\_partition\_nor.fex。

### 8.6.3.1 sys\_partition.fex 分区配置介绍

一个分区的属性，包含名称、分区大小、下载文件与用户属性。以下是文件中所描述的一个分区的属性：

- name，分区名称由用户自定义。当用户在定义一个分区的时候，可以把这里改成自己希望的字符串，但是长度不能超过 16 个字节。
- size，定义该分区的大小，以扇区的单位 (1 扇区 = 512bytes，如上图给 env 分区分配了 32768 个扇区，即  $32768 * 512 / 1024 / 1024 = 16M$ )，注意，为了字节对齐，这里分配的扇区大小应当能整除 128。
- downloadfile，下载文件的路径和名称。可以使用相对路径，相对是指相对于 image.cfg 文件所在分区。也可以使用绝对路径。
- user\_type，提供给操作系统使用的属性。目前，每个操作系统在读取分区的时候，会根据用户属性来判断当前分区是不是属于自己的然后才进行操作。这样设计的目的是为了避免在多系统同时存在的时候，A 操作系统把 B 操作系统的系统分区进行了不应该的读写操作，导致 B 操作系统无法正常工作。

更具体的说明，可参考《TinaLinux 存储管理开发指南》。

## 9 Tina kernel 定制开发

### 9.1 概述

本章节简单介绍 kernel 基本配置、功能裁剪、常用命令的使用，帮助客户了解 Tina 平台 linux 内核，为内核定制开发提供基础。

目前 Tina SDK 中，D1-H 平台使用的是 linux-5.4 版本内核。

### 9.2 代码路径

```
${TinaSDK} /lichee/linux-5.4
```

### 9.3 模块开发文档

详阅 BSP 开发文档，文档目录包括常用内核模块使用与开发说明。

### 9.4 内核配置

客户在定制化产品时，通常需要更改 linux 内核配置，在 TinaSDK 中，打开内核配置的方式如下，

```
croot
make kernel_menuconfig
```

执行完后，shell 控制台会跳出配置菜单。如下图所示 (本节以 Linux-4.9 为例，实际为 Linux-5.4，请以实际工程为准)，



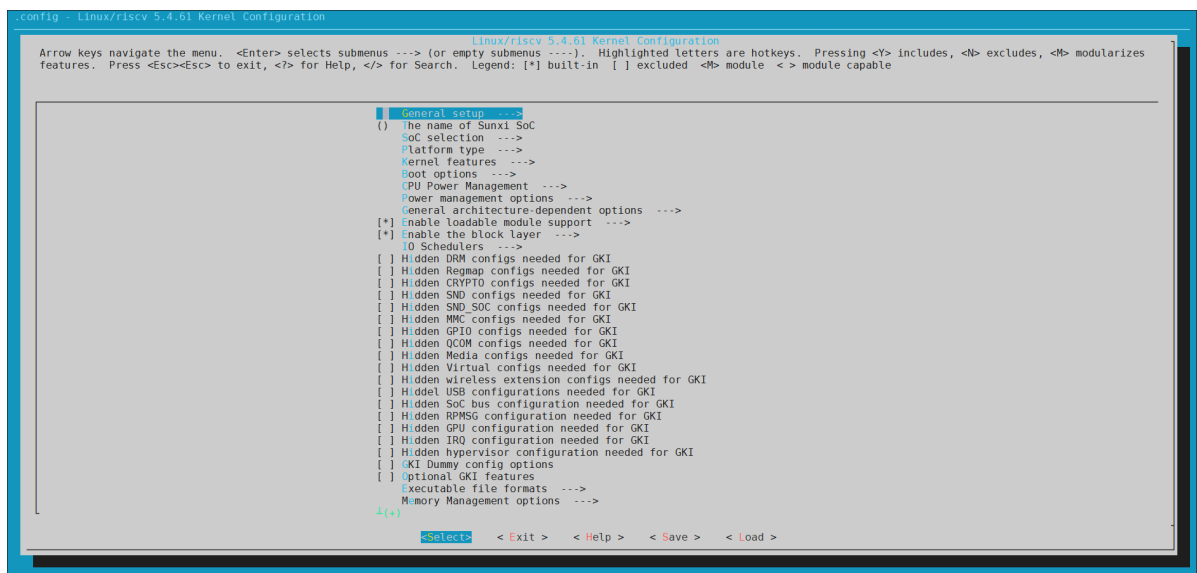


图 9-1: TinaLinux 内核配置菜单



## 10 Tina 系统定制开发

### 10.1 应用移植

在 Tina Linux SDK 中一个软件包目录下通常包含如下两个目录和一个文件：

```
package/<分类>/<软件包名>/Makefile
package/<分类>/<软件包名>/patches/ [可选]
package/<分类>/<软件包名>/files/ [可选]
```

其中，  
patches 保存补丁文件，在编译前会自动给源码打上所有补丁  
files 保存软件包的源码，在编译时会对应源码覆盖源码中的源文件  
Makefile 编译规则文件，

#### 10.1.1 Makefile 范例

该 Makefile 的功能是软件源码的准备，编译和安装的过程，提供给 Tina Linux 识别和管理软件包的接口，软件的编译逻辑是由软件自身的 Makefile 决定，理论上和该 Makefile(该 Makefile 只执行 make 命令和相关参数) 无实质关系。

```
include $(TOPDIR)/rules.mk

PKG_NAME:=bridge                # 软件包名称
PKG_VERSION:=1.0.6              # 软件包版本 详见1
PKG_RELEASE:=1                  # Makefile版本

PKG_SOURCE:=bridge-utils-$(PKG_VERSION).tar.gz # 软件包文件名 (dl目录下)
PKG_SOURCE_URL:=@SF/bridge      # 下载地址 详见2
PKG_MD5SUM:=9b7dc52656f5cbec846a7ba3299f73bd # 软件包MD5值 详见3
PKG_CAT:=zcat                   # 解压方法

PKG_BUILD_DIR:=$(BUILD_DIR)/bridge-utils-$(PKG_VERSION)

include $(INCLUDE_DIR)/package.mk

define Package/bridge            # Package/<包名> 详见4
    SECTION:=net                 # 类别
    CATEGORY:=Base system        # 在menuconfig中路径
    TITLE:=Ethernet bridging configuration utility # 在menuconfig中描述
    URL:=http://bridge.sourceforge.net/ # 下载地址
    # [可选]
    MAINTAINER:=                 # 该包维护人员联系方式
    # [可选]
    DEPENDS:=                     # 该包依赖关系 详见5
    # [可选]
```

```
BUILDONLY:=                                # 固定编译 详见6
endif

# [可选]
define Package/<包名>/conffiles
    # 指定软件包依赖的配置文件，一个配置一行
endif

# [可选]
define Package/<包名>/description
    # 添加详细的文字描述信息
endif

define Package/<包名>/install
    # 指定的软件包安装动作 #详见7
endif

# [可选]
define Build/Configure
    # 软件包配置 详见8
endif

# [可选]
define Build/<包名>/Prepare
    # 编译前的Prepare阶段，可以用于解压源码，给源码打补丁等
endif

# [可选]
define Build/<包名>/Compile
    # 指定编译动作 详见9
endif

$(eval $(call BuildPackage,<包名>)) # 调用编译软件包的宏BuildPackage, 详见10
```

详注：

- 1.如果是开源软件，软件包版本建议与下载软件包的版本一致。
- 2.以PKG开头的变量主要告诉编译系统去哪里下载软件包。
- 3.md5sum用于校验下载下来的软件包是否正确，如果正确，在编译该软件的时候，就会在PKG\_BUILD\_DIR下找到该软件包的源码。
- 4.Package/<name>: <name>用来指定该Package的名字，该名字会在配置系统中显示。
- 5.使用依赖包的名字<name>来指定依赖关系，如果是扩展包，前面添加一个“+”号，如果是内核版本依赖使用@LINUX\_2\_<minor version>。
- 6.如果该值为1，该包将不会出现在配置菜单中，但会作为固定编译，可选。
- 7.内置的几个关键字如下：  
INSTALL\_DIR相当于install -d -m 0755  
INSTALL\_BIN相当于install -m 0755  
INSTALL\_DATA相当于install -m 0644  
INSTALL\_CONF相当于install -m 0600
- 8.在开源软件中一般用来生成Makefile，其中参数可以通过CONFIGURE\_VARS来传递。
- 9.在开源软件中一般相当于执行make，其中有两个参数可以使用:MAKE\_FLAGS和MAKE\_VARS。
- 10.该Makefile的所有define部分都是为该宏的参数做的定义.上层Makefile通过调用此宏进行编译。

## 10.1.2 自启动设置

在 Tina Linux 中支持两种格式的初始化脚本，一种是 busybox 式或者 sysV 式的初始化脚本，一种是 procd 式的初始化脚本。一般我们把由初始化脚本启动的应用叫做服务。

初始化脚本以 shell 脚本的编程语言组织，shell 脚本作为基础知识在此不展开说明。一般情况下，初始化脚本源码保存在软件的 files 目录，且后缀为 “.init”，例如：

```
tina/package/system/fstools/files/fstab.init
```

在 Makefile 的 install 中把初始化脚本安装到 target 端的/etc/init.d 中，例如：

```
define Package/block-mount/install
    $(INSTALL_DIR) $(1)/etc/init.d/
    $(INSTALL_BIN) ./files/fstab.init $(1)/etc/init.d/fstab
endef
```

### 10.1.2.1 调用自启动脚本

- 手动调用方式在启动的时候会有太多的 log，且 log 信息已被 logd 守护进程收集，不利于我们调试初始化脚本，此时可通过 target 端的命令行手动调用的形式来调试，例如：

```
root@TinaLinux: /# /etc/init.d/fstab start
```

### 10.1.2.2 sysV 格式脚本

sysV 式的初始化脚本保存在 target 端的/etc/init.d/目录下，实现开机自启动。下例以最小内容的初始化脚本作示例讲解，核心是实现 start/stop 函数：

```
#!/bin/sh    /etc/rc.common
# Example script
# Copyright (C) 2007 OpenWrt.org

START=10
STOP=15
DEPEND=xxxx

start() {
    #commands to launch application
}
stop() {
    #commands to kill application
}
```

注意：

START=10, 指明开机启动优先级(序列) [数值越小, 越先启动], 取值范围0-99。  
STOP=15, 指明关机停止优先级(序列) [数值越小, 越先关闭], 取值范围0-99。

DEPEND=xxxx，指明初始化脚本会并行执行，通过此项配置确保执行的依赖。

在 rc.common 中提供了一个 init 脚本的功能模板，模板中包括如下几个组成部分：

名称	属性	功能
start	必须实现	启动一个服务
stop	必须实现	停止一个服务
reload	可选实现	重启一个服务
enable	可选实现	重新加载服务
disable	可选实现	禁用服务

在 shell 里面可以使用如下的命令来操作相关的服务。

```
$ root@TinaLinux:/# /etc/init.d/exmple restart|start|stop|reload|enable|disable
```

### 10.1.2.3 procd 格式脚本

以下例的初始化脚本作示例讲解，主要是实现函数 start\_service：

```
#!/bin/sh /etc/rc.common
USE_PROCD=1
PROG=xxxx
START=10
STOP=15
DEPEND=xxxx

start_service() {
    procd_open_instance
    procd_set_param command $PROG -f
    .....
    procd_close_instance
}
```

详细的介绍可以参考：<https://wiki.openwrt.org/inbox/procd-init-scripts>。

## 10.2 应用调试

新添加的软件默认配置为不使能，此时需要手动配置使能软件包。通过在 tina 的根目录执行 make menuconfig 进入软件包的配置界面：

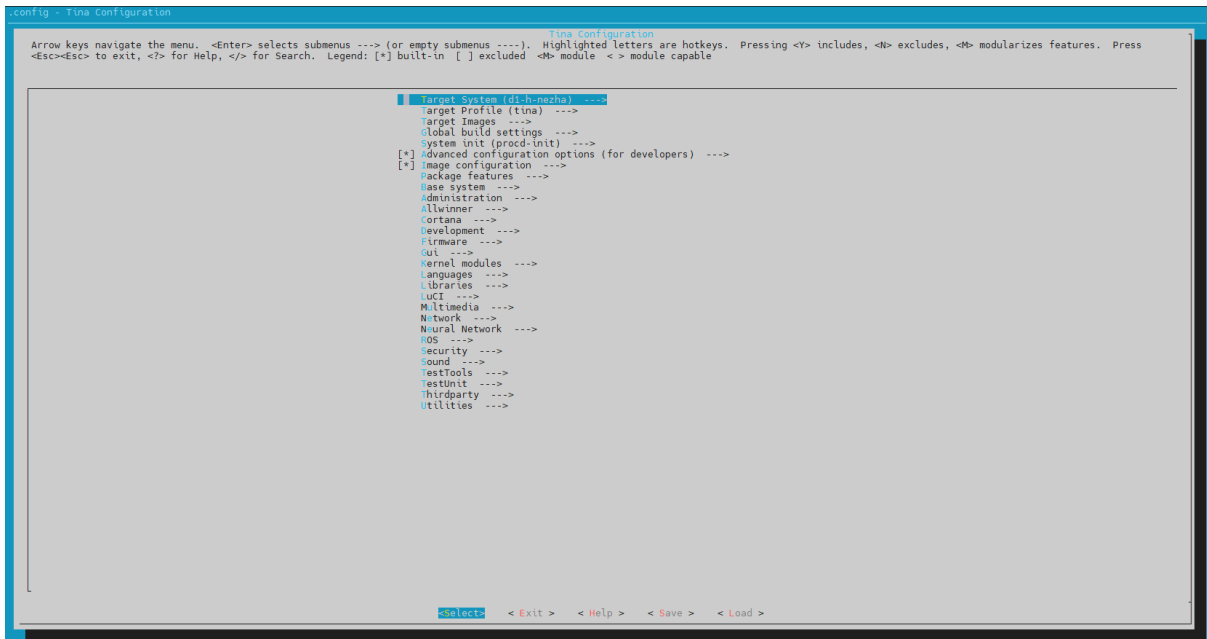


图 10-1: 应用配置主界面

软件包的所在路径与软件包的 Makefile 中的定义有关，以 stress-ng 为例，在 Makefile 中定义为：

```
define Package/stress-ng
SECTION:=utils
CATEGORY:=Allwinner
TITLE:=stress-ng is a more powerful stress utility
URL:=http://kernel.ubuntu.com/~cking/stress-ng/
endef
```

此时，只需要在 menuconfig 界面中进入 Allwinner 选项，即可找到 stress-ng 的软件包。

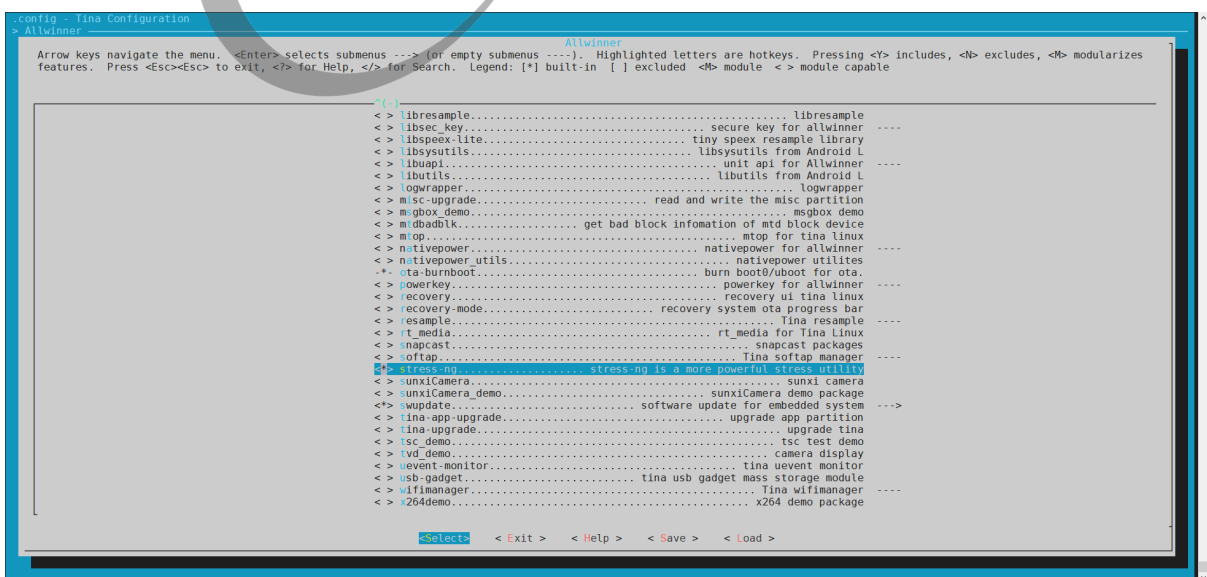


图 10-2: 软件包所在界面

前缀符号含义：

[\*] 或 <\*> : 编译进入SDK  
[ ] 或 <> : 不包含

支持操作：

Y 或 y : 选择包含  
N 或 n : 取消选择

## 10.3 应用编译

详见上文重编应用章节。

## 10.4 应用安装

### (1) . 获取安装包

安装包一般位于目录：

tina/out/<方案>/packages/base

安装包命名格式为：

<应用名>\_<应用版本>-<应用释放版本>\_sunxi.ipk

### (2) . 安装应用包

通过 adb 推送安装包到小机：

\$ adb push <安装包路径> <推送到小机路径>

安装应用包：

\$ opkg install <安装包路径>

## 10.5 分区与挂载

- 升级分区

分区	功能
boot 分区	存内核镜像
rootfs 分区	基础系统镜像分区，包含 /lib, /bin, /etc 等
recovery 分区	存放恢复系统镜像，仅大容量方案有，详见 OTA 文档
extend 分区	存放恢复系统镜像及 rootfs 的 usr 部分，仅小容量方案有，详见 OTA 文档

- 不升级分区

分区	功能
private 分区	存储 SN 号分区
misc 分区	系统状态、刷机状态分区
UDISK 分区	用户数据分区，一般挂载在 /mnt/UDISK
overlayfs 分区	存储 overlayfs 覆盖数据

- 默认挂载点

分区	挂载点	备注
/dev/by-name/boot	/boot	
/dev/by-name/boot-res	/boot-res	
/dev/by-name/UDISK	/mnt/UDISK	用户数据分区
/dev/mmcbk0 或 /dev/mmcbk0p1	/mnt/SDCARD	Tf 卡挂载点
/dev/by-name/rootfs_data	/overlay	存储 overlayfs 覆盖数据






## 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。