Rockchip Buildroot Developer Guide

文件标识：RK-KF-YF-A09

发布版本：V1.0.0

日期：2023-09-20

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址：福建省福州市铜盘路软件园A区18号

网址：www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

本文档介绍基于Rockchip 的arm平台，如何使用官方Buildroot发行版来构建和适配相关硬件功能的开发文档。

Buildroot是使用交叉编译，为嵌入式系统搭建一个完整的linux系统的工具，操作简单，自动。基于原生的Buildroot上Rockchip已集成相关芯片的BSP配置、各硬件模块加速功能的配置、和第三方包深度定制开发，方便客户对产品进行深度定制和二次开发。

芯片支持情况

Buildroot 版本	**SDK** 发布时间	已验芯片
2021.11	2018-06-20	RK3588、RK356X、RK3399、RK3326\PX30\RK3358、RK3308、RK3399、RK3288、RK312X、RK3036
2018.02	2022-01-15	RK356X、RK3399、RK3326\PX30\RK3358、 RK3308、RK3399、RK3288、RK312X、RK3036、RK3399PRO、 RK1808、RK1806、RV1126/RV1109、RK3328

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

日期	版本	作者	修改说明
2023-09-20	V1.0.0	Caesar Wang	初始版本

目录

Rockchip Buildroot Developer Guide

1. Buildroot第三方包下载机制
2. 坏境变量的设定
3. 编译模块和系统
4. 交叉编译工具链
   - 4.1 内部工具链端
   - 4.2 外部工具链端
5. 使用ccache Buildroot
6. 下载包位置
7. 如何在Buildroot添加新软件包
   - 7.1 在软件包目录下为您的软件创建一个目录
   - 7.2 Config.in文件
   - 7.3 .mk文件
8. 包名、配置条目名称和makefile变量关系
9. 如何从github中添加一个软件包
10. 定制模块开发
11. 桌面应用
    - 11.1 Weston应用
    - 11.2 Enlightenment桌面应用
    - 11.3 LVGL桌面应用
12. 用户和密码
13. Weston 开发
    - 13.1 配置方式
14. 具体配置
    - 14.1 屏幕区分
    - 14.2 鼠标样式及大小
    - 14.3 状态栏相关配置
    - 14.4 背景配置
    - 14.5 待机及锁屏配置
    - 14.6 显示颜色格式配置
    - 14.7 屏幕方向配置
    - 14.8 分辨率及缩放配置
    - 14.9 冻结屏幕
    - 14.10 屏幕状态配置
    - 14.11 多屏管理
    - 14.12 输入设备配置
    - 14.13 触屏校准
    - 14.14 无GPU平台配置
    - 14.15 ARM AFBC modifier配置
    - 14.16 降低UI分辨率
15. 截屏功能
16. 中文显示的支持

1. Buildroot第三方包下载机制

SDK中集成了Buildroot第三方包下载机制，确保客户能够有效下载。现在采用了更灵活的机制， 在原生dl基础上增加archives目录，预置第三方包。同时如果无法连接Google，将切换到使用国内的kgithub镜像，并优先从mk脚本指定的源下载，如果失败将尝试mk脚本和defconfig配置的镜像源，最后才是官方备份源。

在官方树中，大多数源码都是使用wget来获取；只有小部分通过他们的git，mercurial,或者svn储存库。

2. 坏境变量的设定

在SDK目录中：

```
source buildroot/envsetup.sh <config_name>
```

config_name 可以`source buildroot/envsetup.sh`列出来，选择具体平台编译。

3. 编译模块和系统

选好编译平台，接下来就可以编译每一个package，它主要是config、build、install三部分组成， 具体如下： 进入buildroot目录。

运行make 构建和安装特定的包及其依赖项。对于依赖于Buildroot基础设施的包，有许多特殊的make目标可以独立调用。像这样：

```
make <package>-<target>
```

command/target	Description
source	获取源代码（下载tarball，克隆源存储库等）
depends	构建和安装构建软件包所需的所有相依性
extract	将源代码放在包构建目录中（提取tarball，复制源代码，等等）
patch	应用补丁，如果有的话
configure	运行configure命令，如果有的话
build	运行编译命令
install-staging	目标包：在暂存目录中，运行软件包的必要安装
install	目标包：运行前面的两个安装命令 主机包：在宿主目录中运行包的安装

此外，还有一些其他有用的目标

command/target	Description
show-depends	显示构建软件包所需的依赖性
clean	运行软件包的清除命令，也可以从目标和暂存目录中卸载软件;注意，这并不是针对所有包实现的
dirclean	删除整个包构建目录
rebuild	重新运行编译命令
reconfigure	重新运行配置命令

举个例子，对`rockchip-test`这个包进行编译、配置、安装，和进行修改重编等。

```
### 编译package
buildroot$ make rockchip-test-configure
buildroot$ make rockchip-test-build
buildroot$ make rockchip-test-install

### 重译package
buildroot$ make rockchip-test-reconfigure
buildroot$ make rockchip-test-rebuild
buildroot$ make rockchip-test-reinstall

### 清除package
buildroot$ make rockchip-test-dirclean
```

如果需要对配置进行修改，可运行`make menuconfig`, 比如：

```
buildroot$ make menuconfig
```

在配置工具中，您可以找到相关的帮助来描述每个菜单条目的目的。一旦配置工具完成配置，它将生成一个包含配置描述的 .config 文件。该文件将被 Makefile 用于执行所需的任务。然后，执行 make 命令。

编译Buildroot系统，直接make即可

```
buildroot$ make
```

> 保存到rootfs配置文件
> buildroot$ make update-defconfig

Buildroot编译后输出在output这个目录， 它包含几个子目录：

- images： 所有镜像（文件系统，比如ext2/4、squashfs、cpio等格式镜像）存储目录。
- build： 除了交叉编译的工具链之外的所有组件，编译（这包括运行所需的工具在主机上的Buildroot和为目标编译的包）。build/目录包含每个组件的一个子目录。
- staging： 其中的层次结构类似于根文件系统层次结构。这个目录包含了安装交叉编译工具链和为目标选择的所有用户空间包。但是，这个目录不是用来成为目标的根文件系统：它包含许多开发文件、未剥离的二进制文件和库，这些文件和库也非常多。对嵌入式系统来说是很大的。这些开发文件用于为所依赖的目标提供编译库和应用程序需要的其他库。
- target： 它几乎包含了目标的完整根文件系统：除了/dev/目录中的设备文件，所有需要的东西都是存在（Buildroot不能创建它们，因为Buildroot不会以root身份运行，也不希望以root身份运行）。此

外，它没有正确的权限（例如，busybox二进制文件的setuid）。因此，该目录不应该用于你的目标。相反，您应该使用 images/目录中构建的映像之一。如果你需要在根文件系统中用NFS挂载外部镜像，必须用root用户在images/目录中生成镜像。相比对于staging/，target/只包含运行所选目标应用程序所需的文件和库：开发文件（头文件）不存在，二进制文件被剥离。

- host：包含为主机编译的工具的安装，这些工具是正确执行Buildroot所必需的，包括交叉编译工具链。

4. 交叉编译工具链

Buildroot提供了不同的交叉编译工具链构建解决方案：

- 内部工具链后端，在配置界面中称为Buildroot工具链。
- 外部工具链后端，在配置界面中称为外部工具链。

```
$ make menuconfig
 -> Toolchain
 |     -> Toolchain type ( [=y])
```

> SDK默认使用内部工具链

4.1 内部工具链端

内部工具链后端是内置的后端，在构建之前，Buildroot自己构建了一个交叉编译的工具链。用于目标嵌入式系统的用户空间应用程序和库,这个后端是Buildroot的历史后端，并且仅限于uClibc C库的使用（i.e，glibc和eglibc C库不受此后端支持，请参阅外部工具链后端和跨界工具链后端解决方案使用glibc或eglibc）。

优点:

- 与Buildroot有良好的兼容性
- 快速，只构建必要的东西

缺点:

- 在进行清洁工作时，需要重建工具链，这需要时间。如果你想减少你的构建时间，考虑使用外部工具链后端。
- 仅限于uClibc C库

4.2 外部工具链端

外部工具链后端允许使用现有的预构建交叉编译工具链。Buildroot能识别几种常见的交叉编译工具链（从ARM的Linaro，ARM的Sourcery CodeBench，x86，x86-64，PowerPC，MIPS和SuperH，来自ADI的Blackfin工具链， Xilinx用于Microblaze的工具
链，等等。） 并且能够自动下载它们，或者它可以指向一个自定义的工具链，可以在本地下载或安装。

优点：

允许使用知名的、经过良好测试的交叉编译工具链。

- 避免交叉编译工具链的构建时间，这在嵌入式Linux系统的总体构建时间中通常非常重要。
- 不局限于uClibc:glibc和eglibc工具链得到了支持。

缺点：

- 如果你的预构建的外部工具链有缺陷，可能很难从工具链供应商那里得到解决方案，除非你构建外部工具使用交叉的工具链。

SDK目录内置交叉编译，当Buildroot外部工具链使用SDK prebuilts目录预置交叉编译，如下：

目录	说明
prebuilts/gcc/linux-x86/aarch64/gcc-arm-10.3-2021.07-x86_64-aarch64-none-linux-gnu	gcc arm 10.3.1 64位工具链
prebuilts/gcc/linux-x86/arm/gcc-arm-10.3-2021.07-x86_64-arm-none-linux-gnueabihf	gcc arm 10.3.1 32位工具链

可从以下百度云网盘地址下载工具链：

工具链下载

SDK若需要编译单个模块或者第三方应用，需交叉编译环境进行配置。比如RK3562其交叉编译工具位于`buildroot/output/rockchip_rk3562/host/usr`目录下，需要将工具的bin/目录和`aarch64-buildroot-linux-gnu/bin/`目录设为环境变量，在顶层目录执行自动配置环境变量的脚本：

```
source  buildroot/envsetup.sh rockchip_rk3562
```

输入命令查看：

```
cd buildroot/output/rockchip_rk3562/host/usr/bin
./aarch64-linux-gcc --version
```

此时会打印如下信息：

```
aarch64-linux-gcc.br_real (Buildroot) 11.3.0
```

> 外部工具链如果需要SDK自带的工具链，
> 可集成`configs/rockchip/toolchain/*`相关配置。

5. 使用ccache Buildroot

默认SDK编译Buildroot都开启ccache，ccache是一个编译器缓存。它存储由每个编译过程产生的对象文件，并且能够跳过未来的编译相同的源文件（具有相同的编译器和相同的参数）通过使用预先存在的对象文件。在几乎是从头开始构建的，它可以很好地加快构建过程。
ccache支持集成在Buildroot中。你必须使能Build options— Enable compiler cache 选项。这将自动构建ccache并将其用于每一个主机和目标编译。缓存位于$HOME/.buildroot-ccache。它存储在Buildroot输出目录之外，这样它就可以由独立的Buildroot构建共享。如果你想要删除缓存，只需删除这个目录。
您可以通过运行make ccache-stats来获得关于缓存的统计信息（它的大小、命中数、遗漏等）。

6. 下载包位置

由Buildroot下载的各种tarball都存储在BR2_DL_DIR中，默认情况下是dl目录。如果你想要保留一个完整版本的Buildroot，它是已知的与相关的tarball一起工作，你可以做一个
这个目录的副本。这将允许您使用完全相同的版本重新生成工具链和目标文件系统。
如果你维护几个Buildroot trees，最好有集成一个预下载包位置。SDK可通过设置`BUILDROOT_DL_DIR`环境变量换了来实现。如果有设置，那么BR2_DL_DIR在Buildroot中的值就会被覆盖。

```
$ export BUILDROOT_DL_DIR <shared download location>

在~/.bashrc中添加上面一行设置环境变量
```

下载位置也可以设置在.config文件中，使用BR2_DL_DIR选项。这个值会被BUILDROOT_DL_DIR环境变量覆盖写。

7. 如何在Buildroot添加新软件包

如何将新软件包（用户空间库或应用程序）集成到Buildroot中。可参考
`builroot/package/rockchip/*`相关包添加。

7.1 在软件包目录下为您的软件创建一个目录

例如 `package/rockchip`

7.2 Config.in文件

然后，创建一个Config.in文件。这个文件将包含与我们的rockchip软件相关的选项描述在配置工具中使用和显示。它基本上应该包含:

```
menuconfig BR2_PACKAGE_ROCKCHIP
        bool "Rockchip Platform"

if BR2_PACKAGE_ROCKCHIP

config BR2_PACKAGE_ROCKCHIP_HAS_ISP1
        bool
        help
          Has Rockchip ISP1
```

7.3 .mk文件

这是最难的部分。创建一个名为rockchip.mk的文件。它描述了如何下载、配置、构建、安装等等。根据包类型的不同，.mk文件必须以不同的方式编写，使用不同的基础结构：

- makefile用于一般软件包（不使用自动工具或CMake）：这些都是基于一个类似于使用的基础设施对于基于自动工具的包，但是需要开发人员做更多的工作。他们指定了应该做什么配置、编译、安装和清理包。这个基础结构必须用于所有的包，不使用自动工具作为他们的构建系统。在将来，可能会为其他构建而编写其他专门的基础设施系统。

- 基于自动工具的软件（autoconf、automake等）的makefile：我们为这些包提供了专门的基础设施，因为自动工具是一个非常常见的构建系统。这个基础设施必须用于依赖于他们用构建系统自动工具的新软件包。
- 基于cmak基软件的makefile：我们为这些包提供了一个专用的基础设施，因为CMake是越来越多的常用的构建系统，具有标准化的行为。这个基础结构必须用于依赖于CMake的新软件包。

8. 包名、配置条目名称和makefile变量关系

Buildroot中，有一些关联：

- the package name，它是包目录名(以及*.mk文件的名称);
- 配置条目名称在 Config.in 文件中声明；
- makefile变量前缀

必须使用以下规则来保持这些要素之间的一致性：

- 包目录和*mk名字是包名本身(例如:package/foo-bar_boo/foo-bar_boo.mk);
- 目标名称是包名本身（如：foo-barboo）;
- 配置条目是大写的包名。 - 字符替换为 _ ，前缀为BR2_PACKAGE_。(例如:BR2_PACKAGE_FOO_BAR_BOO);
- 后缀.mk变量的以大写的包名为前缀，字符替换为_(例如：FOO_BAR_BOO_VERSION);

9. 如何从github中添加一个软件包

在github上管理的软件包一般没有发布包的下载地址。然而，从github仓库上直接下载软件包也是可以的。

```
FOO_VERSION = v1.0 # tag or (abbreviated) commit ID
FOO_SITE = http://github.com/<user>/<package>/tarball/$(FOO_VERSION)
```

> 注意
> • FOO_VERSION可以是标签也可以是提交ID
> • github生成的tarball名称与Buildroot的默认值匹配（例如：foo-1234567.tar.gz），所以它不需要在.mk文件中指定。
> •当使用提交ID作为版本时，通常情况下，SHA1的前7个字符就足够了。

10. 定制模块开发

参考/buildroot/configs/rockchip* 配置开关，各模块可自行定制开发。

```
buildroot$ tree -L 2 configs/rockchip/
configs/rockchip/
├── base  # 通用基础包
│   ├── base.config
│   ├── common.config
```

```
│   ├── kernel.config
│   ├── recovery.config
│   └── tiny.config
├── benchmark.config # benchmark跑分测试工具
├── chips # 相关芯片配置
│   ├── rk3036_arm.config
│   ├── rk3036.config
│   ├── rk312x_arm.config
│   ├── rk312x.config
│   ├── rk3288_arm.config
│   ├── rk3288.config
│   ├── rk3308_aarch64.config
│   ├── rk3308_arm.config
│   ├── rk3308.config
│   ├── rk3326_aarch64.config
│   ├── rk3326_arm.config
│   ├── rk3326.config
│   ├── rk3399_aarch64.config
│   ├── rk3399_arm.config
│   ├── rk3399.config
│   ├── rk3399pro_aarch64.config
│   ├── rk3399pro_arm.config
│   ├── rk3399pro.config
│   ├── rk3399pro_npu_aarch64.config
│   ├── rk3399pro_npu_arm.config
│   ├── rk3399pro_npu.config
│   ├── rk3528_aarch64.config
│   ├── rk3528.config
│   ├── rk3562_aarch64.config
│   ├── rk3562_arm.config
│   ├── rk3562.config
│   ├── rk3566_rk3568_aarch64.config
│   ├── rk3566_rk3568_arm.config
│   ├── rk3566_rk3568.config
│   ├── rk3588_aarch64.config
│   ├── rk3588_arm.config
│   └── rk3588.config
├── chromium.config # chromium浏览器
├── debug.config # 调试工具配置
├── electric.config # 电力配置
├── font # 字体配置
│   ├── chinese.config
│   └── font.config
├── fs # 文件系统格式配置
│   ├── e2fs.config
│   ├── exfat.config
│   ├── ntfs.config
│   ├── ubifs.config
│   └── vfat.config
├── gdb.config # gdb调试
├── gpu # GPU配置
│   └── gpu.config
├── libcamera.config # libcamera配置
├── locale
│   ├── chinese.config
│   └── locale.config
├── multimedia # 多媒体配置
│   ├── audio.config
```

```
│   ├── camera.config
│   ├── gst
│   ├── mpp.config
│   └── rockit.config
├── npu2.config # npu配置
├── powermanager.config #电源管理配置
├── rknn_demo.conf
├── tee_aarch64_v2.config
├── test.config
├── toolchain # 工具链配置
│   ├── arm_10_aarch64.config
│   ├── arm_10_armhf.config
│   └── arm_8_armhf.config
├── weston.config # Weston配置
├── wifibt # RKWIFIBT网络配置
│   ├── bt.config
│   ├── network.config
│   └── wireless.config
└── x11.config # X11配置
...
```

11. 桌面应用

Buildroot默认使用基于wayland协议的weston drm来作为显示后端。如下图：

这些 Weston 应用提供了一些状态栏和背景等基础功能配置，如Chromium浏览器、Terminal终端、Launchers配置，摄像头预览，多路视频，GPU，鼠标等demo。如需更多Demo可以通过/etc/xdg/weston/weston.ini.d/* 配置添加即可。更多Weston参数配置和使用参考 Weston 开发。

> 注意： 第三方UI框架在未经商业授权的情况下可能涉及侵权风险。如果在Rockchip平台上使用Buildroot QT/Enlightenment/Minigui/LVGL等UI框架，需要获得第三方授权和技术支持，因为Rockchip官方不提供相关技术支持和维护。

11.1 Weston应用

基于Wayland上使用的weston客户端相关应用：

```
/usr/bin/#
├── weston-calibrator
├── weston-clickdot
├── weston-cliptest
├── weston-confine
├── weston-content_protection
├── weston-debug
├── weston-dnd
├── weston-editor
├── weston-eventdemo
├── weston-flower
├── weston-fullscreen
├── weston-image
├── weston-multi-resource
├── weston-presentation-shm
```

```
├── weston-resizor
├── weston-scaler
├── weston-screenshooter
├── weston-simple-damage
├── weston-simple-dmabuf-egl
├── weston-simple-dmabuf-v4l
├── weston-simple-egl
├── weston-simple-shm
├── weston-simple-touch
├── weston-smoke
├── weston-stacking
├── weston-subsurfaces
├── weston-tablet
├── weston-terminal
├── weston-touch-calibrator
├── weston-transformed
```

11.2 Enlightenment桌面应用

相关配置开启即可，比如下：

```
BR2_PACKAGE_EFL=y
BR2_PACKAGE_LUAJIT=y
BR2_PACKAGE_EFL_GSTREAMER1=y
BR2_PACKAGE_ENLIGHTENMENT=y
BR2_PACKAGE_EFL_WAYLAND=y
```

实际运行如下图：

11.3 LVGL桌面应用

相关配置开启即可，比如下：

```
BR2_PACKAGE_LVGL=y
BR2_PACKAGE_LVGL_COLOR_DEPTH=32
BR2_PACKAGE_LV_DRIVERS=y
BR2_PACKAGE_LV_DRIVERS_USE_DRM=y
BR2_PACKAGE_LVGL_DEMO=y
BR2_PACKAGE_RK_DEMO=y
BR2_PACKAGE_LVGL_DEMO_USE_DRM=y
BR2_PACKAGE_FREETYPE=y
```

12. 用户和密码

用户：root
密码：rockchip

13. Weston 开发

Weston是Wayland开源显示协议的官方参考实现，Rockchip Buildroot SDK的显示服务默认使用Weston drm后端。

13.1 配置方式

Buildroot SDK中Weston的配置方式有以下几种：

a、启动参数

即启动Weston时命令所带参数，如weston --tty=2，位于/etc/init.d/S49weston，对应SDK代码中位置为：buildroot/package/weston/S49weston

b、weston.ini配置文件

位于/etc/xdg/weston/weston.ini及/etc/xdg/weston/weston.ini.d/下的.ini文件，对应SDK代码中位置如：buildroot/board/rockchip/common/base/etc/xdg/weston/weston.ini

参考：https://fossies.org/linux/weston/man/weston.ini.man

c、特殊环境变量

此类环境变量一般设置在/etc/profile.d/weston.sh，对应SDK代码中位置为：buildroot/package/weston/weston.sh.

d、动态配置文件

对于drm后端显示功能，Buildroot SDK中的Weston提供一些动态配置支持，默认路径为/tmp/.weston_drm.conf，可以通过环境变量WESTON_DRM_CONFIG指定。

e、udev rules

Weston中输入设备的部分配置需要通过udev rules。

14. 具体配置

14.1 屏幕区分

Weston使用output(head) name区分屏幕设备。具体信息获取可以通过Weston启动log：

```
# weston&
[02:11:29.746] DRM: head 'DSI-1' found ...
```

14.2 鼠标样式及大小

Weston支持在weston.ini配置文件的shell段设置鼠标样式和大小，如

```
    # /etc/xdg/weston/weston.ini

    [shell]
    cursor-theme=whiteglass # Buildroot SDK支持comix/obsidian/xcursor/xcursor-
transparent鼠标主题包
    cursor-size=24
```

14.3 状态栏相关配置

Weston支持在weston.ini配置文件的shell段设置状态栏的背景色、位置、缩放，以及在launcher段设置快捷启动程序，如：

```
    # /etc/xdg/weston/weston.ini

    [shell]
    panel-color=0x90ff0000
    # 颜色格式为ARGB8888

    panel-position=bottom
    # top|bottom|left|right|none，none为禁止

    panel-scale=4
    # 缩放为4倍

    [launcher]
    icon=/usr/share/icons/gnome/24x24/apps/utilities-terminal.png
    # 图标路径

    path=/usr/bin/gnome-terminal
    # 快捷启动命令
```

14.4 背景配置

Weston支持在weston.ini配置文件的shell段设置背景图案、颜色，以及在desktop-launcher段设置快捷启动程序，如

```
    # /etc/xdg/weston/weston.ini

    [shell]
    background-image=/usr/share/backgrounds/gnome/Aqua.jpg
    # 背景图案（壁纸）绝对路径

    background-type=tile
    # scale|scale-crop|tile

    background-color=0xff002244
    # 颜色格式为ARGB8888，未设置背景图案时生效


    [desktop-launcher]
    icon=/usr/share/icons/hicolor/256x256/apps/chromium.png
    # 图标路径
```

```
path=/usr/bin/chromium www.baidu.com
# 快捷启动命令

displayname=chromium
# 显示名称
```

14.5 待机及锁屏配置

Weston的超时待机时长可以在启动参数中配置，也可以在weston.ini的core段配置，如：

```
# /etc/init.d/S49weston
start_weston()
{
  /usr/bin/weston --idle-time=0&   # 0为禁止待机，单位为秒
}
```

或者

```
# /etc/xdg/weston/weston.ini

[core]
idle-time=10
```

Weston的锁屏可以在weston.ini的shell段配置，如：

```
# /etc/xdg/weston/weston.ini

[shell]
locking=false
# 禁止锁屏

lockscreen-icon=/usr/share/icons/gnome/256x256/actions/lock.png
# 解锁按钮图案

lockscreen=/usr/share/backgrounds/gnome/Garden.jpg
# 锁屏界面背景
```

14.6 显示颜色格式配置

Buildroot SDK内Weston目前默认显示格式为ARGB8888，对于某些低性能平台，可以在weston.ini的core段配置为RGB565，如：

```
# /etc/xdg/weston/weston.ini

[core]
gbm-format=rgb565
# xrgb8888|argb8888|rgb565|xrgb2101010
```

也可以在weston.ini的output段单独配置每个屏幕的显示格式，如：

```
# /etc/xdg/weston/weston.ini

[output]
name=LVDS-1

gbm-format=rgb565
# xrgb8888|argb8888|rgb565|xrgb2101010
```

14.7 屏幕方向配置

Weston的屏幕显示方向可以在weston.ini的output段配置，如

```
# /etc/xdg/weston/weston.ini

[output]
name=LVDS-1

transform=rotate-90
# normal|rotate-90|rotate-180|rotate-270|flipped|flipped-90|flipped-
180|flipped-270
```

如果需要动态配置屏幕方向，可以通过动态配置文件，如：

```
echo "output:all:rotate90" > /tmp/.weston_drm.conf # 所有屏幕旋转90度
echo "output:eDP-1::rotate180" > /tmp/.weston_drm.conf # eDP-1旋转180度
```

14.8 分辨率及缩放配置

Weston的屏幕分辨率及缩放可以在weston.ini的output段配置，如：

```
# /etc/xdg/weston/weston.ini

[output]
name=LVDS-1

mode=1280x800
# 需为屏幕支持的有效分辨率

scale=2
# 需为整数倍数，支持应用内部实现缩放
```

如需要缩放到特定分辨率(物理分辨率不变)，可以通过WESTON_DRM_VIRTUAL_SIZE环境变量配置所有屏幕的大小，如：

```
# /etc/profile.d/weston.sh
export WESTON_DRM_VIRTUAL_SIZE=1024x768
```

如果需要动态配置分辨率及缩放，可以通过动态配置文件，如：

```
    echo "output:HDMI-A-1:mode=800x600" > /tmp/.weston_drm.conf # 修改HDMI-A-1分辨率为800x600
    echo "output:eDP-1:rect=<10,20,410,620>" > /tmp/.weston_drm.conf # eDP-1显示到(10,20)位置，大小缩放为400x600
    echo "output:HDMI-A-1:size=1920x1080" > /tmp/.weston_drm.conf # 缩放HDMI-A-1到1080p，物理分辨率不变
```

以上缩放时，如果硬件VOP显示模块不支持缩放，则需要依赖RGA处理。

14.9 冻结屏幕

在启动Weston时，开机logo到UI显示之间存在短暂切换黑屏。如需要防止黑屏，可以通过以下方式短暂冻结Weston屏幕内容：

使用定制--warm-up运行参数在UI启动后开始显示

```
    # /etc/init.d/S49weston
    start_weston()
    {
      /usr/bin/weston --warm-up&
    }
```

或者

```
    # /etc/init.d/S49weston
    start_weston()
    {
      export WESTON_FREEZE_DISPLAY=/tmp/.weston_freeze # 设置特殊配置文件路径
      touch /tmp/.weston_freeze # 冻结显示
      /usr/bin/weston&
      sleep 1 && rm /tmp/.weston_freeze& # 1秒后解冻
    }
```

又或者

```
    # /etc/init.d/S49weston
    start_weston()
    {
      echo "output:all:freeze" > /tmp/.weston_drm.conf # 冻结显示
      /usr/bin/weston&
      ...
      sleep 1 && \
      echo "output:all:unfreeze" > /tmp/.weston_drm.conf& # 1秒后解冻
    }
```

14.10 屏幕状态配置

DRM框架支持强制配置屏幕状态:

```
echo on > /sys/class/drm/card0-HDMI-A-1/status # 强制HDMI-A-1为接入状态
#on|off|detect，detect为热拔插
```

如果需要更具体的动态屏幕状态配置，可以通过动态配置文件，如：

```
echo "output:DSI-1:off" > /tmp/.weston_drm.conf #关闭DSI（非拔出）
echo "output:HDMI-A-1:freeze" > /tmp/.weston_drm.conf #冻结HDMI-A-1
echo "output:eDP-1:on" > /tmp/.weston_drm.conf #开启eDP
echo "compositor:state:off" > /tmp/.weston_drm.conf #显示休眠
echo "compositor:state:sleep" > /tmp/.weston_drm.conf #显示休眠，触屏唤醒
echo "compositor:state:freeze" > /tmp/.weston_drm.conf #冻结显示
echo "compositor:state:on" > /tmp/.weston_drm.conf #显示唤醒
```

14.11 多屏管理

Buildroot SDK的Weston支持多屏镜像同显、多屏异显、屏幕位置配置及热拔插等功能。

镜像模式缩放时，如果硬件VOP显示模块不支持缩放，则需要依赖RGA处理。

相关配置通过环境变量设置，如：

```
# /etc/profile.d/weston.sh
export WESTON_DRM_PRIMARY=HDMI-A-1 # 指定主显为HDMI-A-1
export WESTON_DRM_SINGLE_HEAD=1 # 强制单显
export WESTON_DRM_MIRROR=1 # 使用镜像模式（多屏同显），不设置此环境变量即为异显
export WESTON_DRM_KEEP_RATIO=1 # 镜像模式下缩放保持纵横比，不设置此变量即为强制全屏
export WESTON_DRM_HEAD_MODE=primary # 只使能主显
export WESTON_DRM_HEAD_MODE=internal # 只使能内置显示器
export WESTON_DRM_HEAD_MODE=external # 只使能外置显示器
export WESTON_DRM_HEAD_MODE=external-dual # 使能所有显示器，优先外置显示器
export WESTON_DRM_HEAD_FALLBACK=1 # 未匹配到显示器时，使能任意一个有效显示器
export WESTON_DRM_MASTER=1 # 允许关闭未使用的屏幕

export WESTON_OUTPUT_FLOW=horizontal # 默认水平排列
export WESTON_OUTPUT_FLOW=vertical # 默认垂直排列
export WESTON_OUTPUT_FLOW=same-as # 所有显示器默认位置(0,0)
```

也支持在weston.ini的output段单独禁用指定屏幕：

```
# /etc/xdg/weston/weston.ini

[output]
name=LVDS-1

mode=off
# off|current|preferred|<WIDTHxHEIGHT@RATE>
```

也可以通过动态配置文件，如：

```
echo "output:HDMI-A-1:pos=100,200" > /tmp/.weston_drm.conf # 设置HDMI显示位置
echo "output:HDMI-A-1:prefer" > /tmp/.weston_drm.conf # 设置应用默认显示HDMI上
echo "output:HDMI-A-1:primary" > /tmp/.weston_drm.conf # 设置主显为HDMI
```

14.12 输入设备配置

Weston服务默认需要至少一个输入设备，如无输入设备，则需要在weston.ini中的core段特殊设置：

```
# /etc/xdg/weston/weston.ini

[core]
require-input=false
```

Weston中如存在多个屏幕，需求绑定把输入设备和屏幕，则可通过udev规则配置WL_OUTPUT，如：

```
# /lib/udev/rules.d/99-goodix-ts.rules
ATTRS{name}=="goodix-ts", ENV{WL_OUTPUT}="HDMI-A-1"
```

或者配置WL_SEAT：

```
# /lib/udev/rules.d/99-goodix-ts.rules
ATTRS{name}=="goodix-ts", ENV{WL_SEAT}="seat1"
```

```
# /etc/xdg/weston/weston.ini

[output]
name=LVDS-1

seat=seat1
```

或者通过动态配置文件，如：

```
echo "output:HDMI-A-1:input=*" > /tmp/.weston_drm.conf # 匹配所有设备
echo "output:HDMI-A-1:input=" > /tmp/.weston_drm.conf # 禁用输入
echo "output:HDMI-A-1:input=event6" > /tmp/.weston_drm.conf
echo "output:HDMI-A-1:input=goodix*" > /tmp/.weston_drm.conf
echo "output:HDMI-A-1:input=goodix-ts" > /tmp/.weston_drm.conf
```

输入设备的vendor id、product id及设备名可通过evtest工具查询，如：

```
# evtest /dev/input/event8 | head -3
Input driver version is 1.0.1
Input device ID: bus 0x18 vendor 0xdead product 0xbeef version 0x28bb
Input device name: "goodix-ts"
```

14.13 触屏校准

Weston如果需要校准触屏，可以通过WESTON_TOUCH_CALIBRATION环境变量，如：

```
    # /etc/profile.d/weston.sh
    export WESTON_TOUCH_CALIBRATION="1.013788 0.0 -0.061495 0.0 1.332709
-0.276154"
```

或者通过udev规则配置，如：

```
    # /lib/udev/rules.d/99-goodix-ts.rules
    ATTRS{name}=="goodix-ts", ENV{LIBINPUT_CALIBRATION_MATRIX}="1.013788 0.0
-0.061495 0.0 1.332709 -0.276154"
```

校准参数的获取可以使用Weston校准工具: weston-calibrator，工具运行后会生成若干随机点，依次点击后输出校准参数，如：Final calibration values: 1.013788 0.0 -0.061495 0.0 1.332709 -0.276154

也可以直接使用Weston提供的另一个校准工具：weston-touch-calibrator，需要配置:

```
    # /etc/xdg/weston/weston.ini

    [libinput]
    touchscreen_calibrator=true
    calibration_helper=/bin/weston-calibration-helper.sh
```

14.14 无GPU平台配置

SDK中的Weston默认使用GPU进行渲染合成加速，对于无GPU或GPU性能不足的平台，也可以选用RGA替代进行加速。

具体配置需要Buildroot SDK开启BR2_PACKAGE_LINUX_RGA以及BR2_PACKAGE_WESTON_DEFAULT_PIXMAN。

14.15 ARM AFBC modifier配置

当芯片支持AFBC时，SDK中的Weston支持使用基于GPU的AFBC压缩格式进行显示。

具体配置：

```
    # /etc/profile.d/weston.sh
    # export WESTON_DISABLE_ATOMIC=1
    export WESTON_ALLOW_GBM_MODIFIERS=1
```

14.16 降低UI分辨率

当UI性能或DDR带宽不足时，可以降低UI的分辨率。

具体配置：

```
    echo "output:HDMI-A-1:down-scale=0.5" >> /tmp/.weston_drm.conf # UI缩放0.5倍
```

具体可参考Weston开发文档

`<SDK>/docs/cn/Linux/Graphics/Rockchip_Developer_Guide_Buildroot_Weston_CN.pdf`

15. 截屏功能

```
buildroot:/# killall weston
buildroot:/# weston --debug&
buildroot:/# weston-screenshooter

就可以截图当前画面
```

16. 中文显示的支持

weston如下配置:

```
$ cat /etc/xdg/weston/weston.ini
[terminal]
font=Source Han Sans CN Medium
font-size=14
term=xterm-256color
```

把buildroot的configs/rockchip/locale/chinese.config导入或者开启以下配置:

```
BR2_TOOLCHAIN_GLIBC_GCONV_LIBS_COPY=y
BR2_PACKAGE_BUSYBOX_UNICODE=y
# BR2_ENABLE_LOCALE_PURGE is not set
BR2_GENERATE_LOCALE="zh_CN.UTF-8"
```

buildroot中环境变量的设定:

```
root@rk3588:/# cat /etc/profile.d/lang.sh
export LANG=zh_CN.utf8
```

buildroot中相关提交如下:

```
buildroot$ git log --oneline
c476944fee configs: locale.config: Disable BR2_ENABLE_LOCALE_PURGE
f9654d67c8 localedef: Sync with 2018 SDK
2cbb75a54c coreutils: Support bypassing Unicode when printing
15340338dc busybox: Support bypassing Unicode when printing
5e9b8ba00c configs: rockchip: Add locale.config
863eed048e busybox: Support enabling unicode

device/rockchip$ git log --oneline
5c42211 post-rootfs.sh: Support setting LANG environment
```