Tina Linux 图形系统
开发指南

版本号: 1.3
发布日期: 2021.04.07

版本历史

版本号	日期	制/修订人	内容描述
1.0	2019.07.03	AWA1422	1. 创建
1.1	2020.06.06	AWA1422	1. 更新 MiniGUI 相关说明
1.2	2020.06.09	AWA1359	1. 添加 QT 应用的配置方法
1.3	2021.04.07	AWA1422	1. 添 MiniGUI G2D 使用说明

目　录

插　图

表　格

1 概述

本文档将介绍 Allwinner Tina Linux 中已经移植好的窗口系统，以及怎么使用，包括 MiniGUI、QT5、EFL、GTK+（WebkitGtk、Midori）、DirectFB、Wayland，整体结构如下：

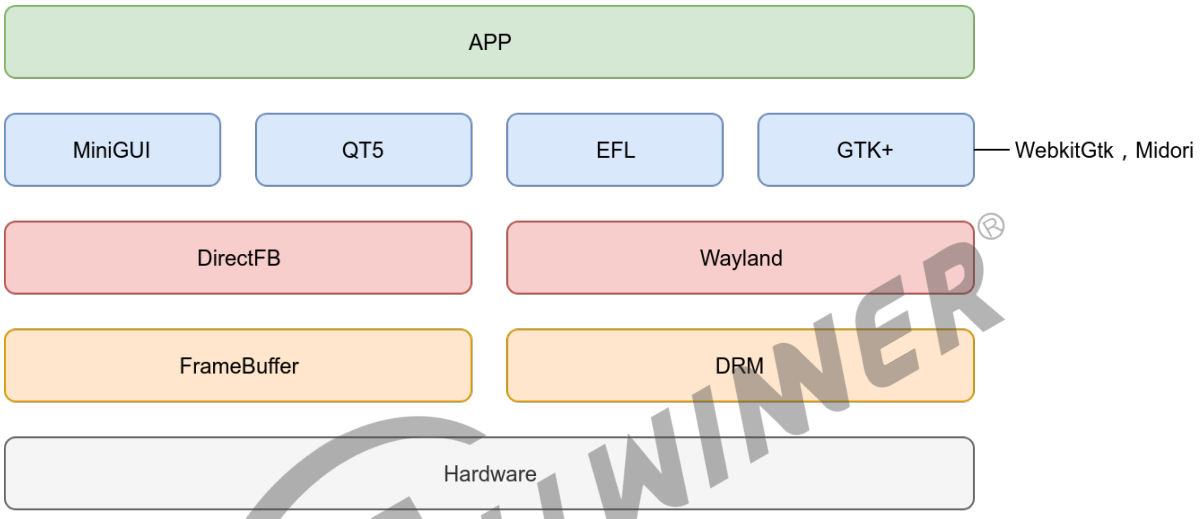

图 1-1: Tina 窗口系统框架

1.1 适用范围

Tina Linux v3.5 及以上版本。

1.2 相关人员

UI 程序开发相关人员。

2 MiniGUI

2.1 MiniGUI 说明

目前 Tina 中移植了 MiniGUI3.2 的核心库以及其组件，下表列出 MiniGUI 相关包说明：

表 2-1: MiniGUI 相关包说明

包名	说明
cell-phone-ux-demo	MiniGUI 手机界面应用
libminigui-gpl	MiniGUI 核心库
minigui-res-be	MiniGUI 资源库
mg-samples	MiniGUI 示例应用
libmdolphin	MiniGUI 浏览器核心库
mdolphin-release-home	MiniGUI 浏览器应用
mdolphin-release-tv	MiniGUI 浏览器应用
mdolphin-samples	MiniGUI 浏览器应用
libmg3d	MiniGUI 提供 3D 接口组件
libmgeff	MiniGUI 动画框架
libmgi	MiniGUI 输入法组件
libmgncs	MiniGUI 新控件集
libmgp	MiniGUI 提供打印功能组件
libmgplus	对 MiniGUI 图形绘制接口的增强
libmgutils	MiniGUI 提供对话框模板

表 2-2: 基于 MiniGUI 开发的应用

包名	说明
multimedia-test	多媒体测试 Demo，包含摄像头预览、拍照、录像、播放音、视频、浏览图片功能
r11-board	智能洗衣机 Demo，包含一些界面滑动效果，选择控件等常用功能实现
smart-music-player	智能音乐播放器 Demo，包含滑动列表实现，在 R328 和 R329 上适配

下面是 multimedia-test 应用截图：

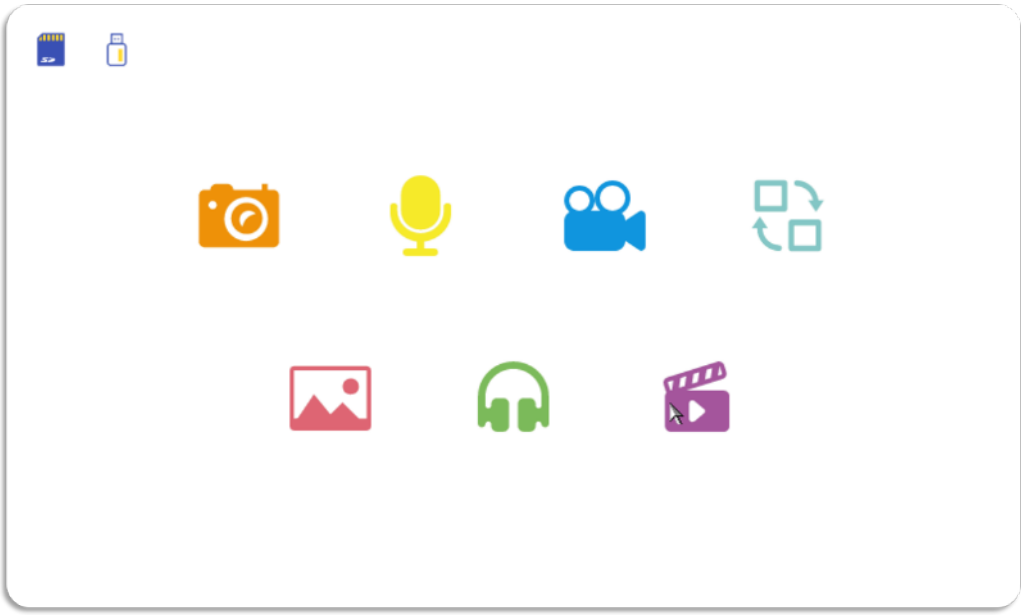

图 2-1：multimedia-test 主页截图

点击 SD 卡和 U 盘图标，可以对 SD 卡和 U 盘格式化，在拍照与录制的时候需要正确的格式，不然不能录制。蓝色的 SD 卡与 U 盘表示 SD 卡与 U 盘正确挂载，灰色的表示没有正确挂载。SD 卡与 U 盘同时挂载的时候，默认使用 SD 卡，点击相应图标进入相应功能界面。

下面是 r11-board 应用截图：

图 2-2：r11-board 主页截图

主页三个页面可以左右滑动切换下一个页面，点击不同的洗衣图片进入具体的洗衣功能界面。

图 2-3: r11-board 功能页截图

点击底部的洗涤、漂洗和脱水可以弹出滑动列表选择不同的参数，点击功能 + 按钮有旋转动画。

下面是 smart-music-player 应用截图：

图 2-4: smart-music-player 截图 1

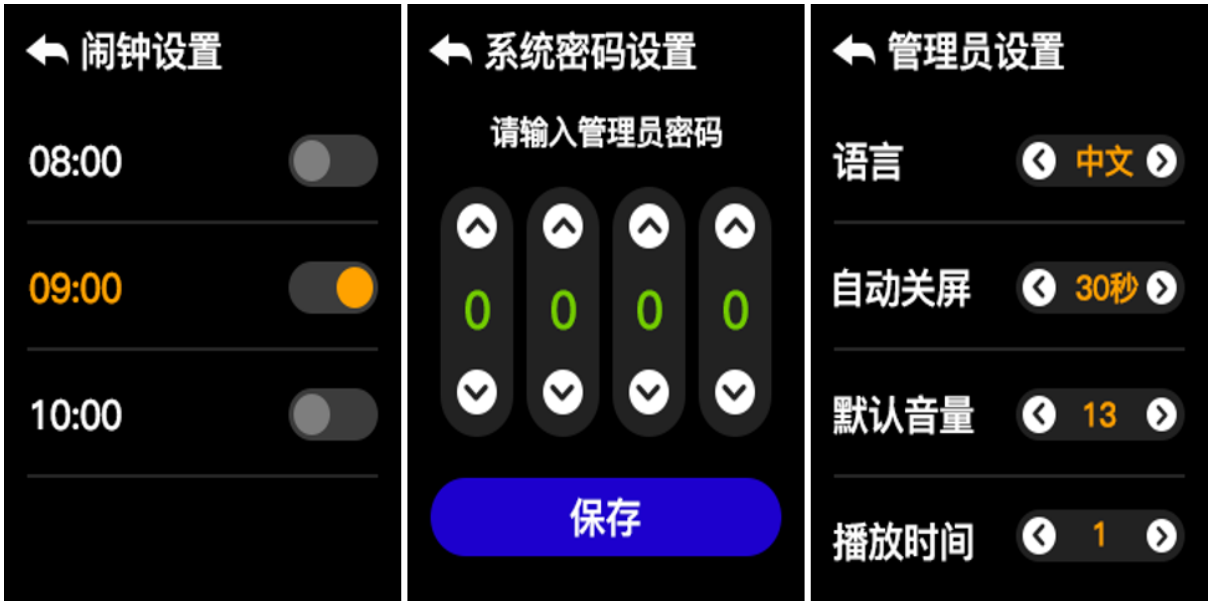

图 2-5: smart-music-player 截图 2

2.2 MiniGUI 配置

```
source build/envsetup.sh
lunch XXX平台名称
make menuconfig
```

```
Minigui --->
    <*> libminigui-gpl --->
        [*] Enabel truetype font support          (支持TTF矢量字体)
        [*] Enabel tslib support for MiniGUI      (支持触摸屏)
        [ ] Enabel g2d support for MiniGUI        (支持G2D硬件加速，需要用sunxifb显示引擎，目前只在
    R528/D1上支持)
        [ ] Enabel g2drotate support for MiniGUI      (支持G2D旋转，需要用sunxifb显示引擎，目前只
    在R528/D1上支持)
        [ ] Enabel sunxifb support for MiniGUI    (显示引擎，类似fbcon，framebuffer长度大于3时，切
    换成循环buffer)
        [ ] Enabel sunxifbion support for MiniGUI     (显示引擎，通过libuapi申请显示buffer)
        [ ] Enabel coortrans cw support for MiniGUI   (UI旋转90度)
        [ ] Enabel reduce size for MiniGUI        (裁剪一些不需要的模块)
    -*- minigui-res-be
    <*> mg-samples
```

MiniGUI 有些示例程序需要将 MiniGUI 的核心库变编译为多进程模式，因此需要进行如下的配置：

```
Minigui --->
    <*> libminigui-gpl --->
        Preferred Minigui Run Mode (ths)  --->  proc
```

几点差异化说明：

1. 目前 Tina 中集成的是 MiniGUI3.2 版本，在 64 位与 32 位的机器上都可以正常运行。
2. 如果使用的不是触摸屏，需要配置鼠标，为了正常的显示鼠标光标，需要修改如下 Make-file：

```
tina/package/minigui/libminigui-gpl/Makefile
```

把–enable-cursor=no 改成 yes，表示使用鼠标。

2.3 MiniGUI 使用

成功烧写固件后，在小机端使用 MiniGUI，需要进行如下几步：

1. 使用的是触摸屏，需要进行触摸屏校准。
2. 配置 MiniGUI.cfg 文件。

2.3.1 触摸屏校准

电容屏不需要校准，如果电容屏触摸不准确，需要把/etc/pointercal 文件删除。

电阻屏首先要确保触摸驱动正常工作，有触摸节点生成，比如说是/dev/input/event1，可以执行下面的命令，再触摸屏幕看串口有无打印。

```
cat /dev/input/event1
```

在小机端设置如下变量：

```
export TSLIB_CALIBFILE=/etc/pointercal
export TSLIB_CONFFILE=/etc/ts.conf
export TSLIB_PLUGINDIR=/usr/lib/ts
export TSLIB_CONSOLEDEVICE=none
export TSLIB_FBDEVICE=/dev/fb0
// TSLIB_TSDEVICE根据触摸屏生成的设备节点来配置
export TSLIB_TSDEVICE=/dev/input/event1
ts_calibrate
```

注意 TSLIB_TSDEVICE 需要是生成的触摸节点，按照屏幕上的提示点击完成校准，校准完成后/etc/pointerca 文件生成，保存这个校准文件，就不用每台产品都校准。

2.3.2 MiniGUI.cfg 配置

小机端/usr/local/etc/MiniGUI.cfg 文件：

```
vim usr/local/etc/MiniGUI.cfg
```

配置 MiniGUI 的 ial 和 gal 引擎，其配置文件的使用如下：

```
[system]
// GAL engine and default options
gal_engine=fbcon
// defaultmode设置显示的大小
defaultmode=800x480-32bpp

[fbcon]
// defaultmode设置显示的大小
defaultmode=800x480-32bpp

[sunxifb]
defaultmode=800x480-32bpp
// flipbuffer=1代替原来的export MG_DOUBLEBUFFER=1
flipbuffer=1
// cacheflag=1使能fb的cache，使buffer拷贝更快，在R328/R329上fb没有cache功能，需要置为0
cacheflag=1
// rotate是控制旋转的角度，使能G2D旋转后有效，当旋转角度为0与180度时，defaultmode不用改变
// 旋转角度为90与270度时，system和sunxifb的defaultmode要改成480x800-32bpp
rotate=0
```

使用触摸屏，注意 mdev 需配置成生成的触摸节点，输入引擎配置如下：

```
// IAL engine
ial_engine=tslib
mdev=/dev/input/event1
mtype=none
```

使用鼠标，输入引擎配置如下：

```
// IAL engine
ial_engine=console
mdev=/dev/input/mouse0
mtype=IMPS2
```

2.4 MiniGUI 优化

2.4.1 Double Buffer

双缓冲的目的主要是防止画面撕裂或者闪烁

1. 修改内核开启双 buffer。

修改文件 tina/lichee/linux-3.4/drivers/video/sunxi/disp2/disp/dev_disp.c

```
//fb0，注意赋值为3或者更多时，使用sunxifb引擎会切换成循环buffer，在快速滑动下可以提升一些帧率
init_para->buffer_num[0] = 2;
```

2. 在 MiniGUI 程序执行前导入环境变量。

```
export MG_DOUBLEBUFFER=1
```

注意只在使用 fbcon 引擎的时候需要导入这个环境变量，sunxifb 引擎由 flipbuffer 字段指定。

执行完 1、2 步，MiniGUI 内部就会使用双缓冲，解决界面切换时闪烁的问题。

3. 还提供了一个函数，可以在应用层控制是否使用双 buffer，比如在打开界面前打开双缓冲，打开界面之后停止使用双缓冲。

开机 framebuffer 是不带 cache 的，运行 minigui 程序的时候，如果执行了 export MG_DOUBLEBUFFER=1 或者 flipbuffer=1 并且 cacheflag=1，framebuffer 会切换成带 cache 的，默认换页的时候会刷 cache。

表 2-3: DoubleBufferEnable 函数说明

函数	说明
DoubleBufferEnable(FALSE)	framebuffer 会切换成不带 cache 的，因此不用刷 cache
DoubleBufferEnable(TRUE)	framebuffer 会切换成带 cache 的，默认换页的时候会刷 cache

DoubleBufferEnable 需要在执行 export MG_DOUBLEBUFFER=1 或者 flipbuffer=1 之后才能调用，DoubleBufferEnable 返回 0 表示调用成功，如果返回-1 表示调用失败，可能是关闭 cache 失败，也可能是 mmap framebuffer 失败，需要应用层再次调用该接口，不然显示异常或出错。

2.4.2 其他

1. 键盘换肤，可以参考《MiniGUI 更换键盘皮肤》文档。
2. 输入法更新词库，可以参考《MiniGUI 输入法更新词库》文档。
3. 文字旋转，可以参考《MiniGUI TTF 旋转字库制作并竖直显示文字》文档。
4. Ubuntu 移植 MiniGUI，可以参考《Ubuntu 64 位移植 Minigui3.2》文档。
5. 视频小窗，可以参考《minigui_per_view 视频小视窗播放》文档。

3 QT5

3.1 QT5 配置

目前 Tina 中移植了 QT5.10.1 版本，Qt 配置可以参考如下说明：

```
source build/envsetup.sh
lunch XXX平台名称
make menuconfig
```

```
Global build settings
    Binary stripping method (strip) --->    strip
Qt --->
    -*- qt5-core
    <*> qt5-examples
```

这个将原本的库的制表符信息裁剪，来减小库的大小，Qt 的某些库需要用到库的头信息 strtab 这个符号表，因此选择 strip 这种轻度的裁剪，留下 strtab 这个符号表，默认的选择是 sstrip。

为了加快编译速度，提供了不同编译工具链预编译的 QT 包：

```
tina/dl/qt-everywhere-opensource-src-5.12.9-prebuilt_glibc_32bit.tar.gz
tina/dl/qt-everywhere-opensource-src-5.12.9-prebuilt_glibc_64bit.tar.gz
tina/dl/qt-everywhere-opensource-src-5.12.9-prebuilt_musl_32bit.tar.gz
tina/dl/qt-everywhere-opensource-src-5.12.9-prebuilt_musl_64bit.tar.gz
tina/dl/qt-everywhere-opensource-src-5.12.9.tar.xz
```

如果源码编译有问题，查看 alsa-lib 配置是否选上。

通过在 make menuconfig 中选择 Qt–> qt5 use prebuilt 来判断使用哪种编译方法。

```
make menuconfig
Libraries
      --->-*- alsa-lib
Qt
      --->[*] qt5 use prebuilt
```

qt5 use prebuilt 的值会在 tina/package/qt/qt5/Makefile 文件中使用。

```
ifeq ($(CONFIG_QT5_USE_PREBUILT),y)
ifeq ($(CONFIG_USE_GLIBC),y)
ifeq ($(TARGET_ARCH),aarch64)
    PKG_MD5SUM:=b96ae8d2d55983911b7bf46896d516bc
    PKG_SOURCE:=qt-everywhere-opensource-src-$(PKG_VERSION)-prebuilt_glibc_64bit.tar.gz
    PKG_BUILD_DIR=$(COMPILE_DIR)/qt-everywhere-opensource-src-$(PKG_VERSION)-
    prebuilt_glibc_64bit
```

```
else
    PKG_MD5SUM:=6fc40f289dd51ad2bf2403ad2da85bf9
    PKG_SOURCE:=qt-everywhere-opensource-src-$(PKG_VERSION)-prebuilt_glibc_32bit.tar.gz
    PKG_BUILD_DIR=$(COMPILE_DIR)/qt-everywhere-opensource-src-$(PKG_VERSION)-
    prebuilt_glibc_32bit
endif
else ifeq ($(CONFIG_USE_MUSL),y)
ifeq ($(TARGET_ARCH),aarch64)
    PKG_MD5SUM:=b7859b3fc75a28f10047cc63f8bb2226
    PKG_SOURCE:=qt-everywhere-opensource-src-$(PKG_VERSION)-prebuilt_musl_64bit.tar.gz
    PKG_BUILD_DIR=$(COMPILE_DIR)/qt-everywhere-opensource-src-$(PKG_VERSION)-
    prebuilt_musl_64bit
else
    PKG_MD5SUM:=9d1e2d3b5673976b3277142f047d2c90
    PKG_SOURCE:=qt-everywhere-opensource-src-$(PKG_VERSION)-prebuilt_musl_32bit.tar.gz
    PKG_BUILD_DIR=$(COMPILE_DIR)/qt-everywhere-opensource-src-$(PKG_VERSION)-
    prebuilt_musl_32bit
endif
endif
else
    PKG_MD5SUM:=f177284b4d3d572aa46a34ac8f5a7f00
    PKG_SOURCE:=qt-everywhere-opensource-src-$(PKG_VERSION).tar.xz
    PKG_BUILD_DIR=$(COMPILE_DIR)/qt-everywhere-opensource-src-$(PKG_VERSION)
endif
```

3.2 QT5 platforms 选择

1. eglfs，在绘图的时候会使用 GPU 渲染 UI，如果平台有 GPU，尽量使用 eglfs。
2. libqlinuxfb，linux 标准的显示框架，会打开/dev/fb0 节点进行绘图和显示。

平台插件的参数配置在 package/qt/qt5/files/qt-env.sh 这个文件，如下所示，默认的 platforms 是 eglfs，其中 MALI_NOCLEAR 环境变量的作用是调用 eglInitialize 函数时不清屏，不然在显示开机 logo 之后，会有一段黑屏时间，用户体验不好。

```
#!/bin/sh

export QT_QPA_PLATFORM=eglfs:size=800x480
export QT_QPA_PLATFORM_PLUGIN_PATH=/usr/lib/qt5/plugins
export QT_QPA_FONTDIR=/usr/lib/fonts
export QT_QPA_GENERIC_PLUGINS=tslib
export QT_QPA_GENERIC_PLUGINS=evdevmouse:/dev/input/event1
export QT_QPA_GENERIC_PLUGINS=evdevkeyboard:/dev/input/event2
export MALI_NOCLEAR=1
```

通常生成的平台插件在小机端的：

```
/usr/lib/qt5/plugins/platforms/libqeglfs.so
```

linuxfb 平台插件动态库为 libqlinuxfb.so。

如需更改为 linuxfb，需要修改 tina/package/qt/qt5/files/qt-env.sh 文件内容，还需要 make menuconfig 选上 qt5-drivers-linuxfb，如下所示：

```
Qt --->
    -*- qt5-core
    <*> qt5-drivers-linuxfb
    <*> qt5-examples
```

linuxfb 可以通过以下环境变量进行配置：

```
#!/bin/sh

export QT_QPA_PLATFORM=linuxfb:fb=/dev/fb0:size=800x480:mmSize=800x480:offset=0x0:tty=/dev/
    tty1
export QT_QPA_PLATFORM_PLUGIN_PATH=/usr/lib/qt5/plugins
export QT_QPA_FONTDIR=/usr/lib/fonts
export QT_QPA_GENERIC_PLUGINS=tslib
export QT_QPA_GENERIC_PLUGINS=evdevmouse:/dev/input/event1
export QT_QPA_GENERIC_PLUGINS=evdevkeyboard:/dev/input/event2
```

```
fb=/dev/fbN              // 指定帧缓冲设备;
size=<width>x<height>    // 指定屏幕大小，以像素为单位;
mmsize=<width>x<height> // 物理宽度和高度;
offset=<width>x<height> // 屏幕左上角像素偏移量
nographicsmodeswitch-    // 不要将虚拟终端切换到图形模式;
tty=/dev/ttyN            // 覆盖虚拟控制台，仅在nographicsmodeswitch未设置时使用;
```

eglfs 可以通过以下环境变量进行配置:

```
export QT_QPA_EGLFS_WIDTH=800        // 包含屏幕宽度（以像素为单位）
export QT_QPA_EGLFS_HEIGHT=480       // 包含屏幕高度（以像素为单位）
export QT_QPA_EGLFS_FB=/dev/fb0      // 覆盖帧缓冲设备，默认是/dev/fb0
export QT_QPA_EGLFS_DEPTH=32         // 覆盖屏幕的颜色深度，默认值为32
```

3.3 QT5 鼠标触摸屏配置

Qt 中使用鼠标，需要启动 udev，将鼠标设备标记为输入设备，然后 Qt 的 libinput 来处理输入事件，才能够识别鼠标。设置 udev 为自启动，默认已经将 udev 设置为自启动。

屏幕为触摸屏，因此需要 make menuconfig 选上 Qt 触摸模块 qt5-drivers-touchscreen，如下所示：

```
Qt --->
    -*- qt5-core
    <*> qt5-drivers-touchscreen
    <*> qt5-examples
```

触摸屏驱动在小机端的：

```
/usr/lib/qt5/plugins/generic/libqtslibplugin.so
```

如果触摸没效果，执行如下环境变量：

```
export QT_QPA_GENERIC_PLUGINS=tslib
```

3.4 QT5 示例运行

成功烧写固件后，在小机端使用 QT，如果使用的是电阻触摸屏，需要进行触摸屏校准，请参考本文档 2.3.1 小节。

QT 的应用示例在小机端的如下路径：

```
usr/share/qt5/examples      /这是QT自带的测试应用/
```

```
make menuconfig
Qt
  --><*> qt-easing.
  --><*> qt-textures
  --><*> qt-washing-machine
//这是新添加的三个QT应用,如果运行此应用有问题，请参照package/qt/qt-washing-machine/src/doc文档
```

运行 qt 应用需要指定插件平台，目前 QT 支持的插件平台有 eglfs 或者 linuxfb，运行示例如下所示：

```
./application -platform eglfs
./application -platform linuxfb
```

或者先执行下面的命令，导入 QT 的环境变量，再执行程序。

```
./etc/qt-env.sh
```

3.5 QT5 问题锦集

3.5.1 strip

运行 QT 的应用程序会出现如下问题，需要将 libqeglfs.so 库重新推到/usr/lib/qt5/plugins/platforms 路径下。这里如果多个插件平台库都出现这个问题，可能是由于，Tina 系统中将编译生成的库进行裁剪，使其更小，Qt 在进行动态加载的时候，需要找到库头信息中的 strtab 制表符，因此在 make menuconfig 中选择轻度裁剪模式-strip。

```
root@TinaLinux:/usr/share/qt5/examples/opengl/cube# ./cube
This application failed to start because it could not find or load the Qt platform plugin
"eglfs"
in "".

Reinstalling the application may fix this problem.
Aborted
```

图 3-1: QT5 问题锦集 strip

3.5.2 eglfs

出现下面错误，申请不上 native window 有可能是缺少 libqeglfs-mali-integration.so 这个库，需要将其 adb push 到小机端的/usr/lib/qt5/plugins/egldeviceintegrations 路径下。

```
root@TinaLinux:/usr/share/qt5/examples/opengl/cube# ./cube
qt.qpa.input: X-less xkbcommon not available, not performing key mapping
EGL Error : Could not create the egl surface: error = 0x300b

Aborted
root@TinaLinux:/usr/share/qt5/examples/opengl/cube#
```

图 3-2: QT5 问题锦集 eglfs

3.5.3 runtime

出现下面错误，传入环境变量：

```
export QT_QPA_EGLFS_INTEGRATION=none
export XDG_RUNTIME_DIR=/dev/shm
```

```
root@TinaLinux:/usr/share/qt5/examples/opengl/cube# ./cube -platform eglfs
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
EGL library doesn't support Emulator extensions
Aborted
root@TinaLinux:/usr/share/qt5/examples/opengl/cube#
```

图 3-3: QT5 问题锦集 runtime

3.5.4 触摸使用不了

出现这个原因有可能是下面步骤导致：

1. 触摸屏没有适配校准。

参考《2.3.1 触摸屏校准》/etc/ts_calibrate 进行校准。

2.qt 没有配置触摸屏的节点。

参考《3.2 QT5 platforms 选择》。

3. 如果还是不行单独执行。

```
export QT_QPA_GENERIC_PLUGINS=tslib
```

4 EFL

4.1 EFL 说明

Enlightenment Foundation Libraries (EFL) 驱动 Enlightenment，它们也可以独立使用或者构建在其他库之上以提供有用的功能并创建强大的应用程序。

核心库 EFL 在速度和大小方面都比其 GTK + 和 Qt 等的效率更高，并且具有更小的内存占用量。

目前 Tina 中移植了 EFL 1.20.6 的核心库以及其组件，下表列出 EFL 相关包说明。

表 4-1: EFL 相关包说明

包名	说明
efl	EFL 功能函数库
ephoto	依赖与 EFL 的相册应用
terminology	依赖于 EFL 的终端仿真器

下面是应用截图：

图 4-1：efl-on-wayland

4.2 EFL 配置

EFL 可以使用 Framebuffer 或者 Wayland 显示图像，如果使用 Wayland，需按照本文档第 8 小节配置好 Wayland。在 Tina 系统中，已经默认配置好了 Framebuffer。执行如下命令配置 EFL：

```
source build/envsetup.sh
lunch XXX平台名称
make menuconfig
```

```
Libraries --->
    -*- EFL
    <*> ephoto
    <*> terminology
```

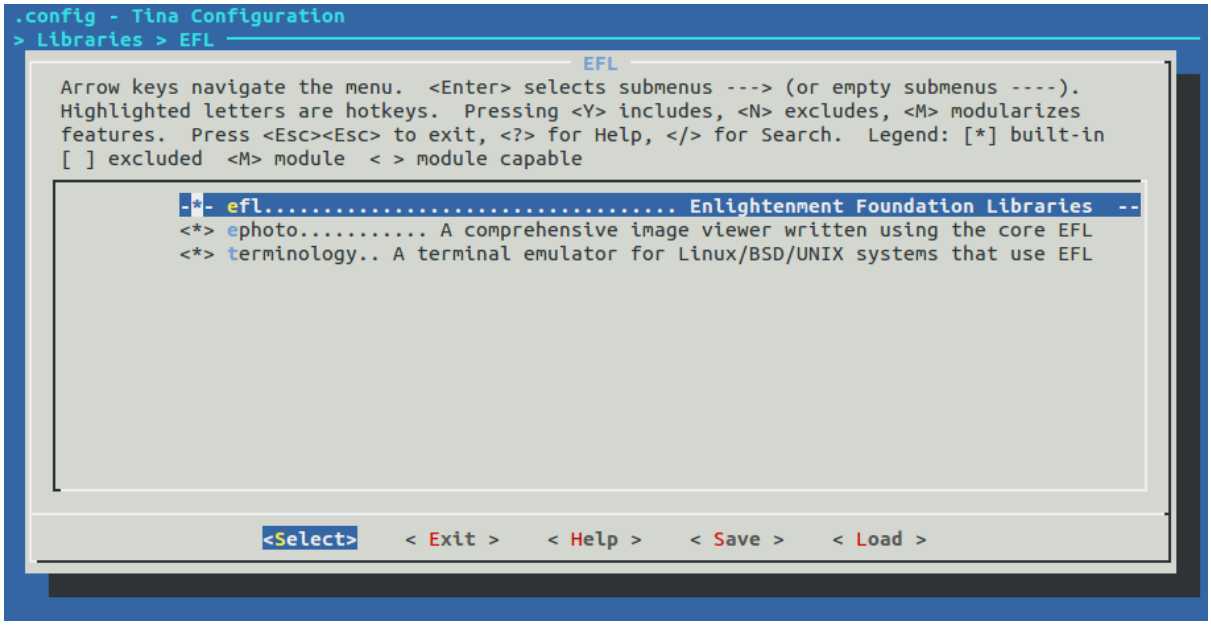

图 4-2: 配置 EFL 选项

efl 是核心库，ephoto 是一个相册应用，该应用可以选择板子里的图片进行浏览与幻灯片播放，terminology 是一个终端仿真器，类似于 ubuntu 中的终端，进入到 efl 的配置界面，可以配置 efl 支持的功能。如下图所示：

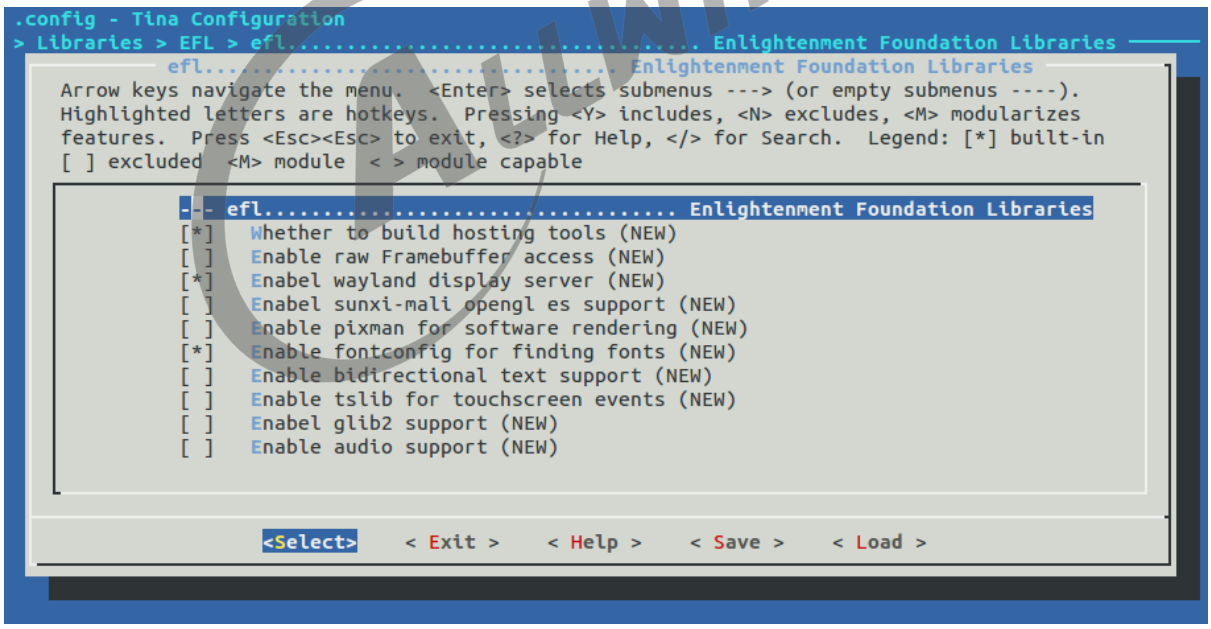

图 4-3: 配置 EFL 支持的功能选项

主要关注以下几项配置：

表 4-2: EFL 配置说明

配置	说明
Enable raw Framebuffer access	使用 framebuffer 显示 efl 的界面
Enabel wayland display server	使用 wayland 显示 efl 的界面
Enabel sunxi-mali opengl es support	使用 opengl es
Enable bidirectional text support	是否支持双向文本，从左到右，或从右到左显示文字
Enable tslib for touchscreen events	是否支持触摸

如果使用 framebuffer 显示 efl 的界面，则不需要再做其他配置什么，因为在 Tina 中默认开启 framebuffer 的，如果使用 wayland，则需要参考本文档第 8 小节配置好 wayland。

4.3 EFL 运行

成功烧写固件后，如果使用 Wayland 的话，需要保证 Weston 已经运行，在小机端使用 EFL，执行以下命令运行测试程序：

```
elementary_test
```

elementary_test 是官方的小程序，包含 efl 中各种控件的使用示例。其他两个测试程序也是这样执行：

```
ephoto
```

```
terminology
```

还可以执行 elementary_config 去配置 elf，可以配置界面渲染的模式，字体、控件的大小等等。

```
elementary_config
```

也可以手动指定渲染引擎，比如：

```
ECORE_EVAS_ENGINE=wayland_egl elementary_test
// 或者
ELM_ACCEL=gl elementary_test
// 或者
ELM_DISPLAY=wl elementary_test
```

如果想看 efl 的调试信息，可以在运行程序前加上：

```
EINA_LOG_LEVEL=4 elementary_test
```

如果是使用 wayland 显示 efl 界面的，并且想测试 opengl es，则执行：

```
ELM_ACCEL=gl elementary_test
```

然后点击 GLView Gears，GLView Many Gears，GLViewSimple 查看结果，执行 elementary_test 的时候界面可能是黑的，移动一下界面，滚动一下界面，或者最大化界面，就可以显示界面了，如果 elementary_test 没有响应，可以结束进程，再次尝试。使用 kill -9 PID 命令结束。

5 GTK+

5.1 GTK+ 说明

GTK+ 是用来创造图形界面的库，它可以运行在许多类 UNIX 系统，Windows 和 OSX。GTK+ 按照 GNU LGPL 许可证发布，这个许可证对程序来说相对宽松。GTK+ 有一个基于 C 的面向对象的灵活架构，它有对于许多其他语言的版本, 包括 C++, Objective-C, Guile/Scheme, Perl, Python, TOM, Ada95, Free Pascal 和 Eiffel。GTK+ 依赖于以下库：

1. GLib 是一个多方面用途的库, 不仅仅针对图形界面。GLib 提供了有用的数据类型、宏、类型转换，字符串工具，文件工具，主循环抽象等等。
2. GObject 是一个提供了类型系统、包括一个元类型的基础类型集合、信号系统的库。
3. GIO 是一个包括文件、设备、声音、输入输出流、网络编程和 DBus 通信的现代的易于使用的 VFS 应用程序编程接口。
4. cairo Cairo 是一个支持复杂设备输出的 2D 图形库。
5. Pango Pango 是一个国际化正文布局库。它围绕一个表现正文段落的 PangoLayout object。Pango 提供 GtkTextView、GtkLabel、GtkEntry 和其他表现正文的引擎。
6. ATK 是一个友好的工具箱。它提供了一个允许技术和图形用户界面交互的界面的集合。例如，一个屏幕阅读程序用 ATK 去发现界面上的文字并为盲人用户阅读。GTK ＋部件已经被制作方便支持 ATK 框架。
7. GdkPixbuf 是一个允许你从图像数据或图像文件创建 GdkPixbuf(“pixel buffer”) 的小的库。用一个 GdkPixbuf 与显示图像的 GtkImage 结合。
8. GDK 是一个允许 GTK ＋支持复杂图形系统的抽象层。GDK 支持 X11、wayland、Windows 和 OS X 的图形系统工具。
9. GTK+ 是 GTK+ 库本身包含的部件，确切的说是 GUI 零件，比如 GtkButton 或者 GtkTextView。

更多 GTK 应用编程可参考：示例

Gtk+(GIMP Tool Kit，GIMP 工具包) 是一个用于创造图形用户接口的图形库，下面是 GIMP on GNU/Linux 的截图：

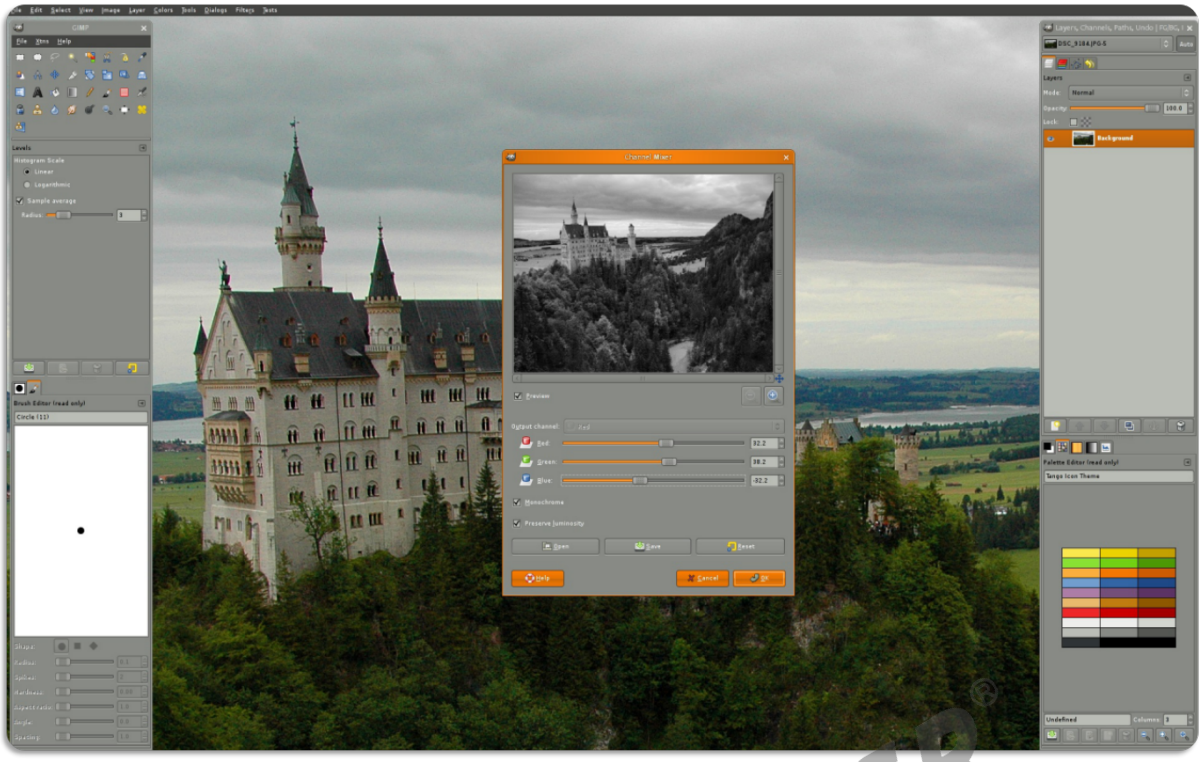

图 5-1：GTK+GIMP 运行截图

Tina 系统移植了 GTK+3 的库及其组件，对应 GTK 包及依赖说明如下：

gtk+-3.22.10.tar.xz：Gtk+3 对应源代码。

Pkgconf、gettext-full、atk、glib2、libcairo、pango、gdk-pixbuf、libepoxy、libxkb-common、libpixman、libinput、wayland、wayland-protocols、udev、libdrm、sunxi-mali：Openwrt 系统 Gtk+3 依赖包名称；对应 Makefile 位于 package/libs/libgtk3/。

5.2 GTK+ 配置

GTK 仅基于 R18 系统平台验证过，其它平台暂未验证；默认 GTK 配置成 wayland port，理论上 GTK 可以运行于所有支持 Wayland 的平台；其中 R40 使用 Wayland+FBDEV 作为显示后端，R18 使用 Wayland+DRM。

```
source build/envsetup.sh
lunch XXX平台名称
make menuconfig
```

以 R18 平台为例，主要配置项如下：

```
Libraries --->
    cairo --->
        -*- libcairo
            [*] Enable cairo postscript support
```

```
            [*] Enable cairo pdf support
            [*] Enable cairo png support
            [ ] Enable script support
            [*] Enable cairo svg support
            [ ] Enable cairo tee support
            [ ] Enable cairo xml support
    gtk3 --->
        <*> libgtk3
            [*] Broadway GDK backend
            [*] Wayland GDK backend
            [*] Install libgtk3 demo program
```

因为 Gtk+3 依赖于 Wayland，Wayland 依赖于 Weston 合成器，配置时需要选上 Weston 和 Wayland，需按照本文档第 8 小节配置好 Wayland。

5.3 GTK+ 运行

成功烧写固件后，如果使用 Wayland 的话，需要保证 Weston 已经运行，然后在小机终端运行：

```
/usr/bin/gdk-pixbuf-query-loaders --update-cache
gdk-pixbuf-query-loaders > /usr/lib/gdk-pixbuf-2.0/2.10.0/loaders.cache
```

然后运行 gtk3-demo：

```
gtk3-demo
```

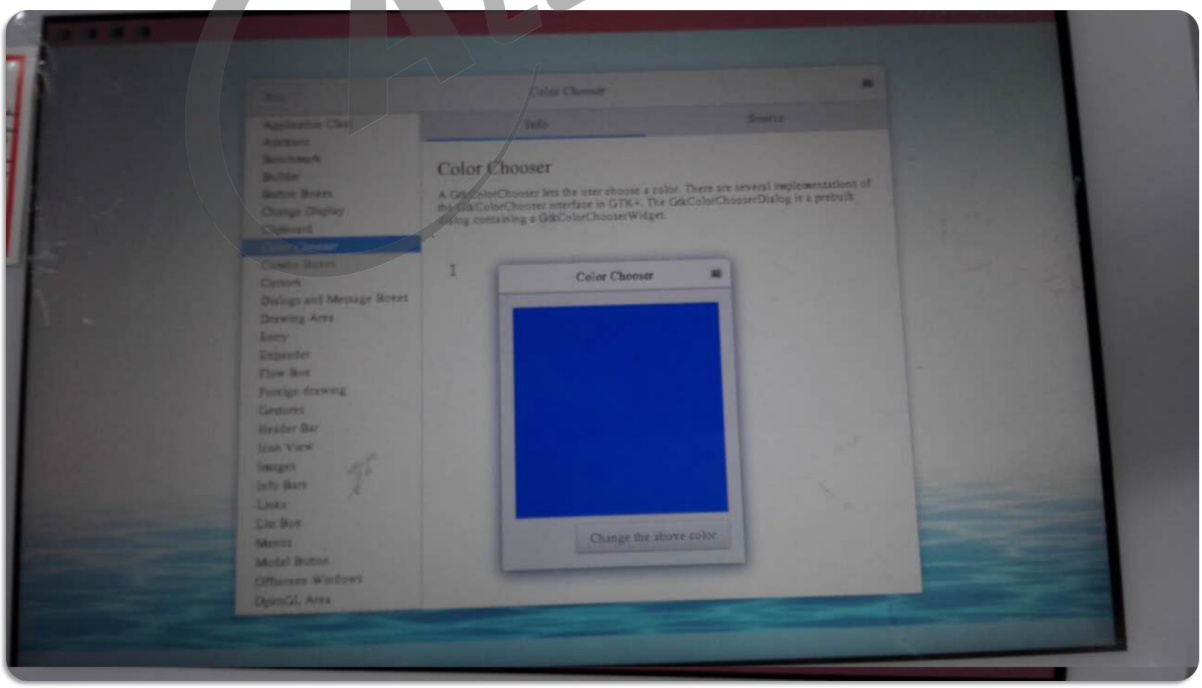

图 5-2: GTK+Demo 运行截图

5.4 GTK+ 示例

```
#include <gtk/gtk.h>

int main( int argc, char *argv[] ){
    GtkWidget *window;

    gtk_init (&argc, &argv);

    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    gtk_widget_show(window);

    gtk_main ();

    return(0);
}
```

6 WebKitGtk

6.1 WebkitGtk 说明

Tina 系统移植了 WebKitGtk 的库及其组件，对应 WebKitGtk 包及依赖说明如下：

webkitgtk-2.18.6.tar.xz、midori_0.5.11_all_.tar.bz2、package/libs/webkitgtk、package/utils/midori：WebKitGtk 和 Midori 浏览器对应源代码及 Makefile。

ruby/host、flex/host、bison/host、gperf/host、enchant、harfbuzz、icu、libjpeg、libgtk3、libsecret、libsoup、libxml2、libxslt、libsqlite3、libegl、libgles、libwebp libgles、lcms2、libtasn1、gstreamer1、gst1-libav 、gst1-plugins-bas、gst1-plugins-good、gst1-plugins-ugly、gst1-plugins-bad：Openwrt 系统 WebkitGtk 依赖包名称。

下面是 WebKitGtk 的截图：

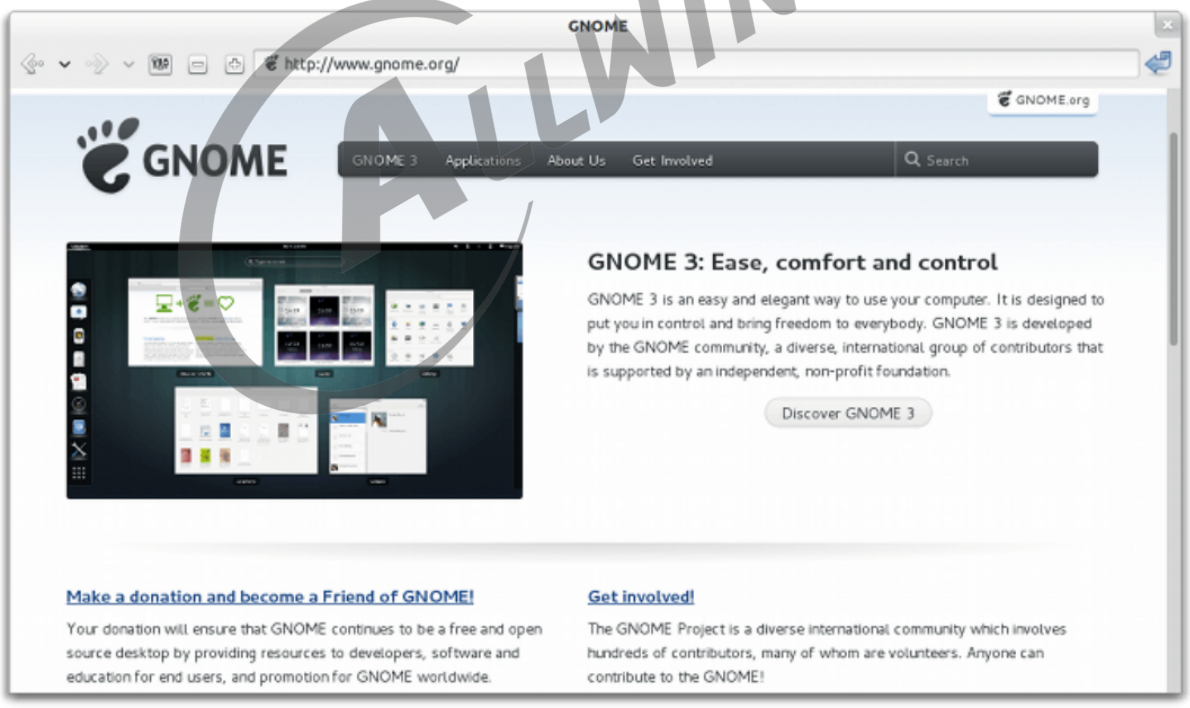

图 6-1：WebKitGtk 运行截图

6.2 WebKitGtk 配置

WebKitGtk 仅基于 R18 系统平台验证过，其它平台暂未验证；默认 WebKitGtk 配置成 wayland port，R18 使用 Wayland+DRM。

```
source build/envsetup.sh
lunch XXX平台名称
make menuconfig
```

```
Libraries --->
    WebkitGtk  --->
        <*> webkitgtk
Utilities --->
    Browser --->
        <*> midori
```

因为 WebKitGtk 依赖于 Gtk+3 和 Wayland，Wayland 依赖于 Weston 合成器，配置时需要选上 Gtk+3、Weston 和 Wayland，需按照本文档第 5 和 8 小节配置好 Gtk+3 和 Wayland。

6.3 WebKitGtk 运行

成功烧写固件后，如果使用 Wayland 的话，需要保证 Weston 已经运行，然后在小机终端运行：

```
/usr/bin/gdk-pixbuf-query-loaders --update-cache
gdk-pixbuf-query-loaders > /usr/lib/gdk-pixbuf-2.0/2.10.0/loaders.cache
```

然后运行 Midori 或 minibrowser：

```
midori
```

或者：

```
minibrowser
```

6.4 WebKitGtk 问题锦集

报错：

```
error: Package `gee-0.8' not found in specified Vala API directories or GObject-
    Introspection GIR directories
```

原因是主机环境安装了 Vala 这个工具，但是需要的是 tina 中编译出的这个工具，卸载主机 Vala 工具即可。

7 DirectFB

7.1 DirectFB 说明

DirectFB（直接帧缓冲区）是在 Linux 帧缓冲区（fbdev）抽象层之上实现的一组图形 API。

- 最大化硬件加速的实用程序。
- 支持高级图形操作，例如多种 alpha 混合模式。
- 没有内核修改没有库依赖项，libc 除外。
- 符合 MHP 规范的要求。

目前在 Tina 中，还没有对接过 GPU。

目前 Tina 中移植了 DirectFB 的核心库以及其 Demo，下表列出 DirectFB 相关包说明：

表 7-1: DirectFB 相关包说明

包名	说明
directfb	directfb 核心库
directfb-examples	directfb demo

7.2 DirectFB 配置

```
source build/envsetup.sh
lunch XXX平台名称
make menuconfig
```

```
Libraries --->
    -*- directfb
Utilities --->
    <*> directfb-examples
```

7.3 DirectFB 运行

在小机端可以执行一些 df_ 开头的测试用例，比如 df_andi，df_dok：

```
df_andi
```

图 7-1：DirectFB 运行截图

8 Wayland

8.1 Wayland 说明

Wayland 是一套 display server(Wayland compositor) 与 client 间的通信协议，而 Weston 是 Wayland compositor 的参考实现，定位于在 Linux 上替换 X 图形系统。

目前 Tina 中移植了 Wayland 的核心库以及其组件，下表列出 Wayland 相关包说明：

表 8-1: Wayland 相关包说明

包名	作用
glmark2	使用 Wayland 作为运行后端的 GPU 测试程序，或者使用 FBDEV 进行显示
wayland	编译 Weston 需要用到的主机端工具
wayland-protocols	Wayland 协议，相当于插件
weston	核心库

8.2 Wayland 配置

8.2.1 menuconfig

Wayland 目前可以在 R18 与 R40 上运行，其他平台暂未测试，其中在 R40 只能使用 FBDEV 作为运行后端，在 R18 上可以使用 DRM 与 FBDEV。执行如下命令进行配置：

```
source build/envsetup.sh
lunch XXX平台名称
make menuconfig
```

```
Wayland --->
    glmark2
        [ ] Enabel fbdev support
        [*] Enabel wayland support
    wayland
    wayland-protocols
    weston --->
        [ ] Enabel dbus support
        [ ] Enabel weston-launch linux pam support
        [*] Enabel opengl es support
        [ ] Enabel fbdev compositor support
```

```
        [*] Enabel drm compositor support
        [ ] Enabel lcms supports support
        [ ] Enabel junit xml support
        [ ] Enabel demo clients install
```

如下图所示：

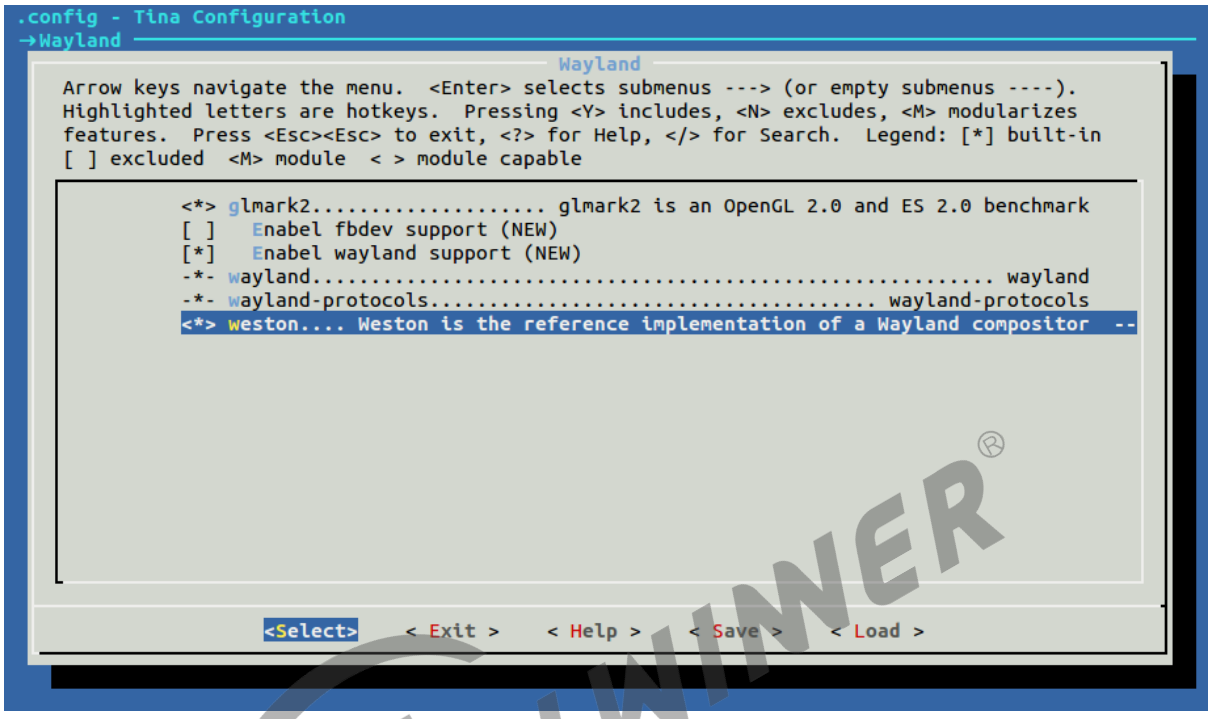

图 8-1: Wayland 选项

glmark2 是使用 GPU 的跑分测试程序，可以在 R18 上使用 DRM 作为 Wayland 后端的时候使用，除此之外还可以使用 FBDEV 进行显示并测试 GPU 性能。wayland，wayland-protocols 在编译 weston 的时候用到，进入到 weston 的配置界面，可以配置 weston 支持的功能。如下图所示：

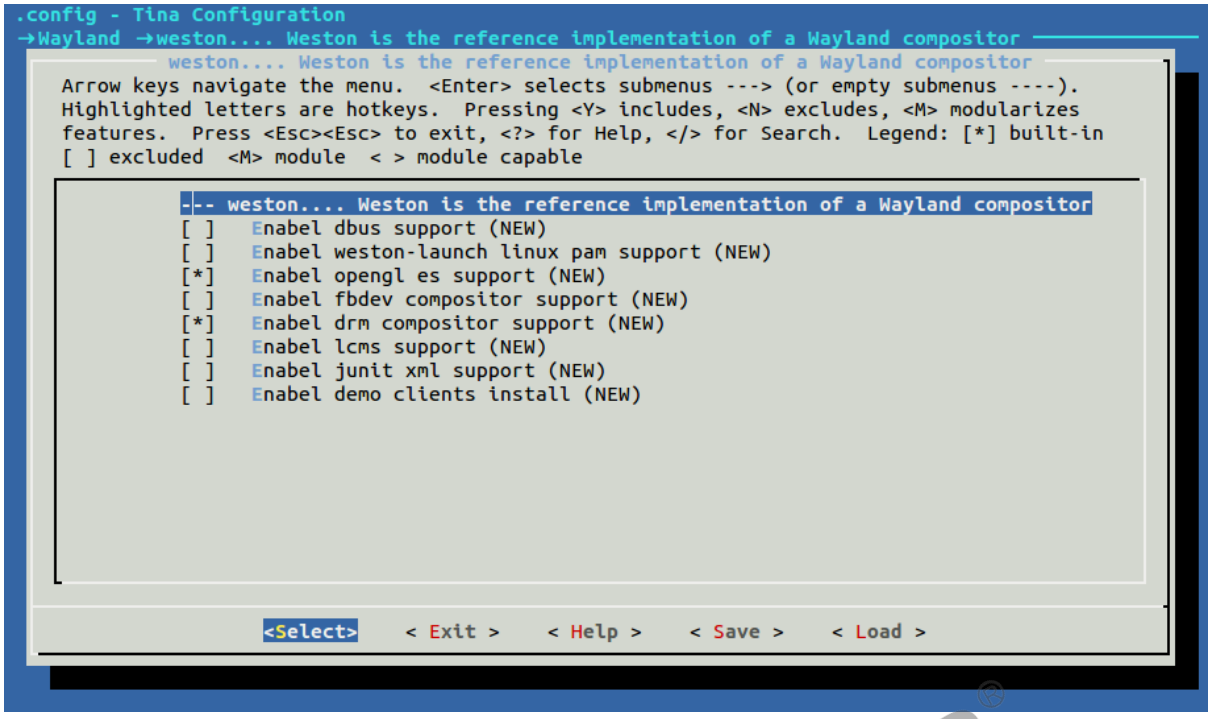

图 8-2: Weston 选项

主要关注以下几项配置：

表 8-2: Wayland 配置说明

选项	说明
Enabel opengl es support	只能在使用 DRM 作为 weston 后端的时候选上，支持 opengl es GPU 加速
Enabel fbdev compositor support	使用 framebuffer 作为显示引擎
Enabel drm compositor support	使用 DRM 作为显示引擎
Enabel demo clients install	编译出 Wayland 测试 Demo

如果使用 FBDEV，需要选上 kmod-mali-utgard-km 与 kmod-sunxi-disp：

```
Kernel modules --->
    Video Support --->
        <*> kmod-mali-utgard-km
        <*> kmod-sunxi-disp
        < > kmod-sunxi-drm
```

如果使用 DRM 与 GPU 加速，需要选上 kmod-mali-utgard-km 与 kmod-sunxi-drm：

```
Kernel modules --->
    Video Support --->
        <*> kmod-mali-utgard-km
        < > kmod-sunxi-disp
        <*> kmod-sunxi-drm
```

选择 DRM 与 GPU 加速的如下图所示：

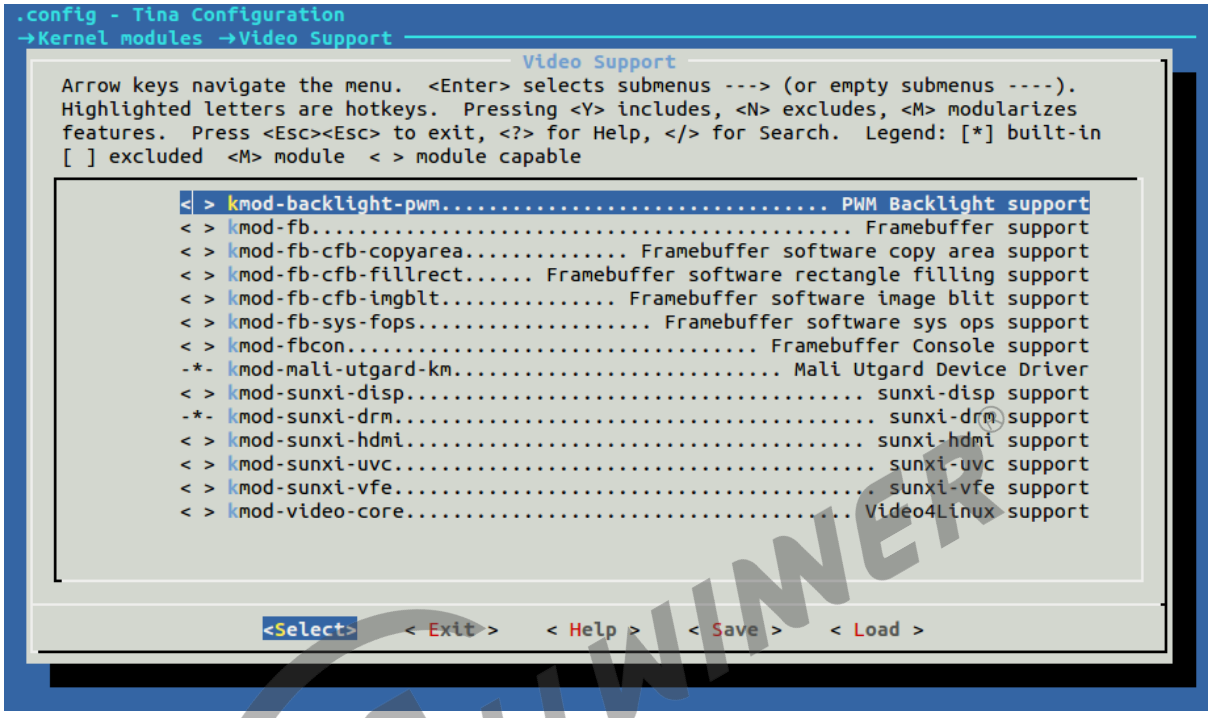

图 8-3: Kernel modules 配置

注意 FBDEV 与 DRM 是互斥的，不能同时选择。

如果需要 Weston 开机自启动，需要修改 tina/package/wayland/weston 目录下的 Makefile，把 $(CP) ./weston $(1)/etc/init.d 的注释去除即可。

如果选择了 DRM 作为显示引擎，还可以把 DRM 的测试 Demo 给选上，选项如下：

```
Libraries --->
    libdrm --->
        [*] install libdrm test programs
        [ ] Enable support for vc4's API
```

weston 依赖的库 cairo 也可以配置，其中 Enable cairo pdf support 与 Enable cairo png support 是必须选择上的，不然编译的时候会报错，如果编译 GTK+ 的话，需要多选择一些，参考本文档第 5.2 小节。

```
Libraries --->
    cairo --->
        -*- libcairo
            [ ] Enable cairo postscript support
            [*] Enable cairo pdf support
```

```
            [*] Enable cairo png support
            [ ] Enable script support
            [ ] Enable cairo svg support
            [ ] Enable cairo tee support
            [ ] Enable cairo xml support
```

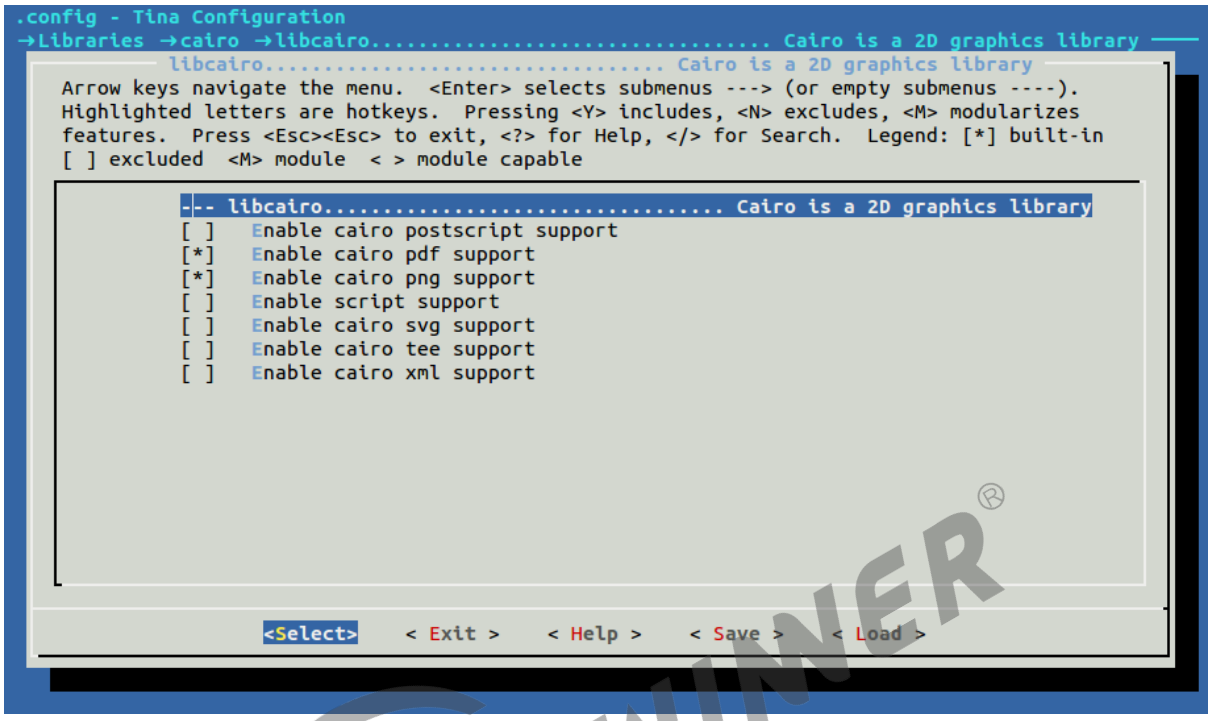

图 8-4: Cairo 选项

8.2.2 kernel_menuconfig

8.2.2.1 FBDEV

如果 menuconfig 选择的是使用 FBDEV 作为后端，已经选择 kmod-sunxi-disp，R18 平台会自动配置下面的选项，不用再执行这一小节的步骤，其他平台暂未实现自动配置，下面的主要是记录使用 FBDEV 作为后端，需要的内核配置。其他平台内核已经默认配置使用 FBDEV。

执行以下命令，以 R18 的为例：

```
make kernel_menuconfig
```

选上 Framebuffer Console Support(sunxi)、DISP Driver Support(sunxi-disp2)、Framebuffer Console support 与 Transform Driver Support(sunxi)：

```
Device Drivers --->
    Graphics support --->
        Frame buffer Devices --->
            <*> Support for frame buffer devices --->
                Video support for sunxi --->
```

```
                    [*] Framebuffer Console Support(sunxi)
                    <*> DISP Driver Support(sunxi-disp2)
        Console display driver support --->
            <*> Framebuffer Console support
    Character devices --->
        <*> Transform Driver Support(sunxi)
```

如下图所示：

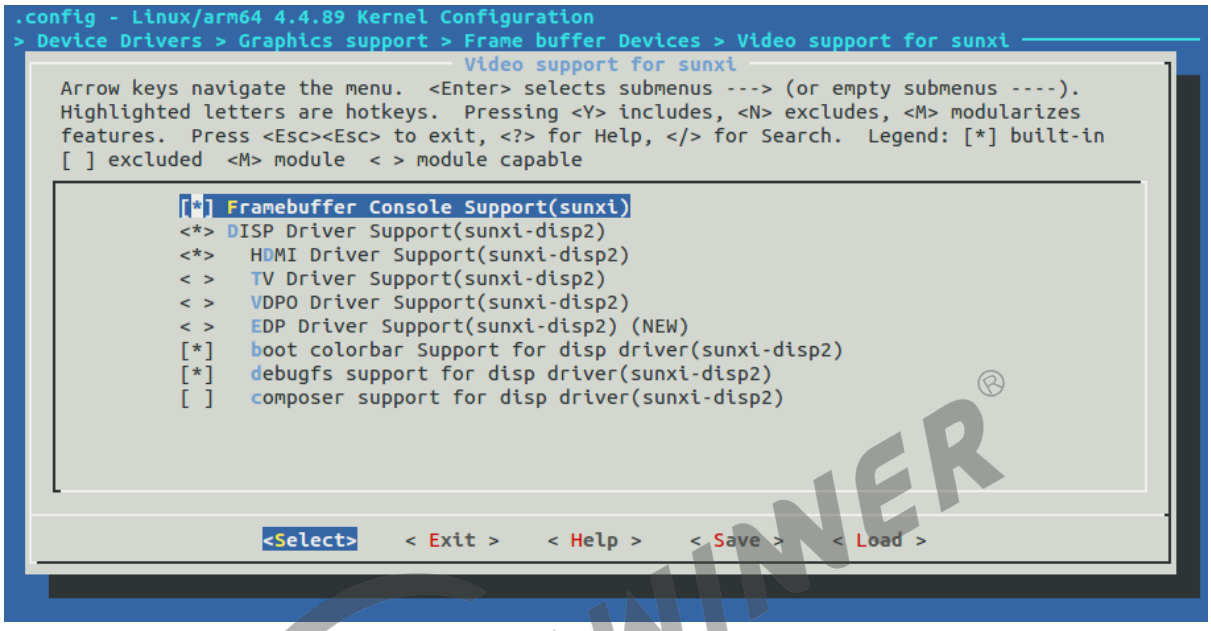

图 8-5: Video support for sunxi 选项

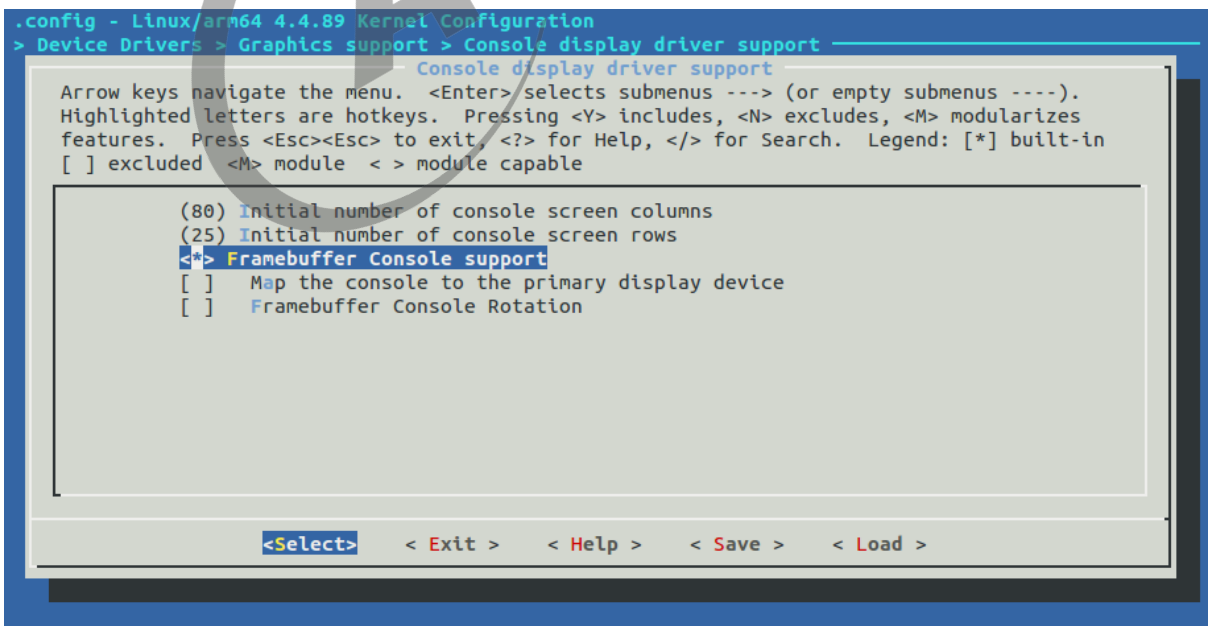

图 8-6: Console display driver support 选项

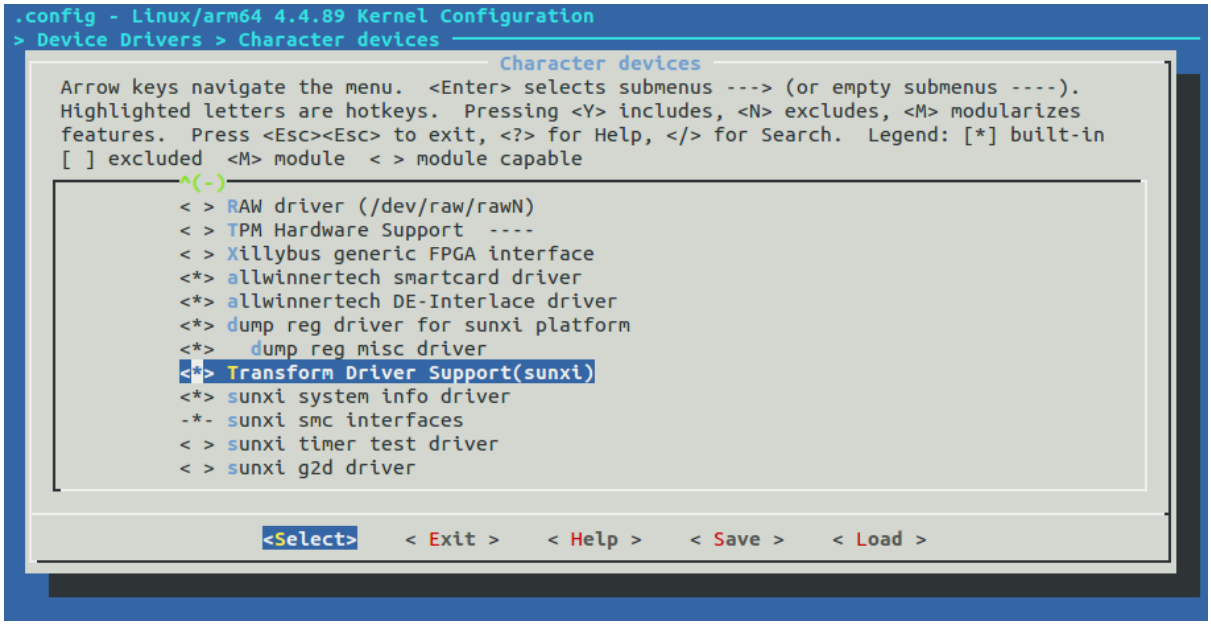

图 8-7: Character devices 选项

8.2.2.2 DRM

如果 menuconfig 选择的是使用 DRM 作为后端，由于内核中默认使用 FBDEV，所以先要取消原本的配置，再选择上 DRM 的配置，在 menuconfig 的配置中取消 kmod-sunxi-disp，选上 kmod-sunxi-drm，R18 平台会自动配置下面的选项，不用在执行这一小节的步骤，其他平台暂未实现自动配置。

执行以下命令，以 R18 的为例。现阶段只有 R18 支持 DRM：

```
make kernel_menuconfig
```

取消选择 Framebuffer Console Support(sunxi)、DISP Driver Support(sunxi-disp2)、Framebuffer Console support 与 Transform Driver Support(sunxi)：

```
Device Drivers --->
    Graphics support --->
        Frame buffer Devices --->
            < > Support for frame buffer devices --->
                Video support for sunxi --->
                    [ ] Framebuffer Console Support(sunxi)
                    < > DISP Driver Support(sunxi-disp2)
        Console display driver support --->
            < > Framebuffer Console support
    Character devices --->
        < > Transform Driver Support(sunxi)
```

选上 DRM 配置：

```
Device Drivers --->
    Graphics support --->
        <*> Direct Rendering Manager (XFree86 4.1.0 and higher DRI support)
        <*> DRM Support for Allwinnertech SoC A and R Series
```

如下图所示：

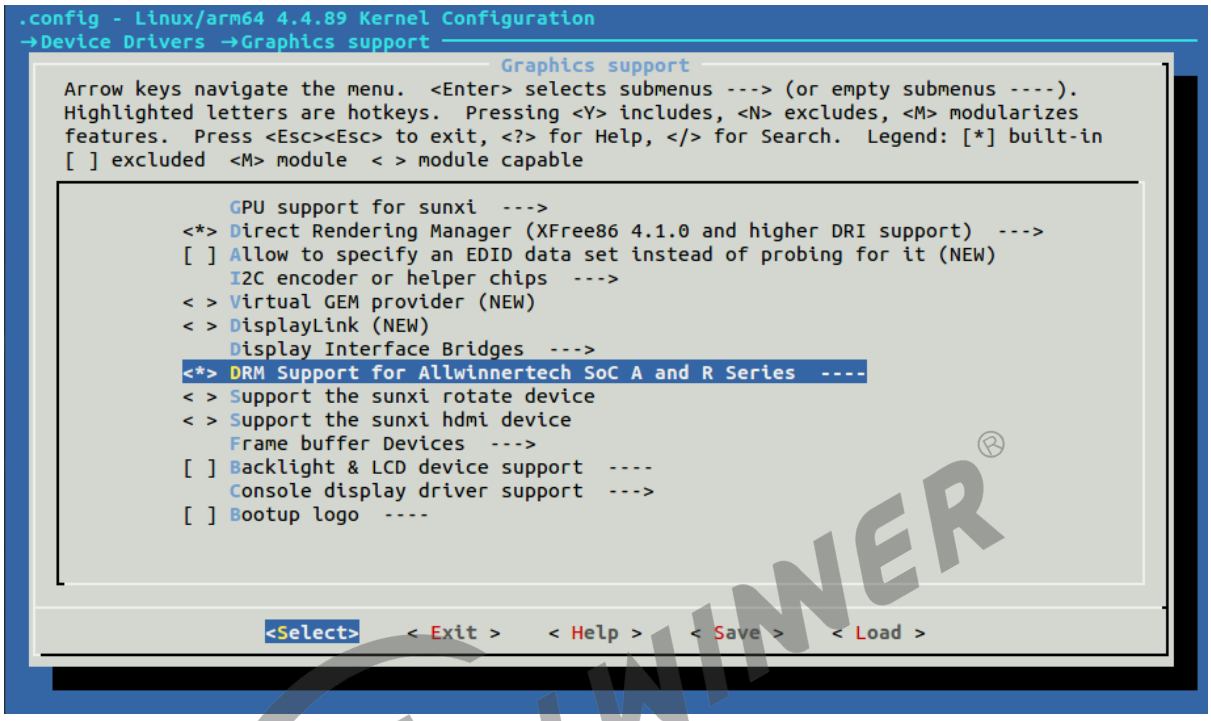

图 8-8: Graphics support 选项

DRM 配置好之后，以前可能编译过不使用 DRM 的固件，需要先把 out 删除掉，并且需要清理之前内核编译的文件，不然可能会遇到一些编译问题，在内核目录下执行：

```
make clean
```

8.3 Wayland 使用

8.3.1 weston 运行

成功烧写固件后，在小机端使用 Wayland，需要执行以下命令：

```
chmod 0700 /dev/shm/
export XDG_RUNTIME_DIR=/dev/shm
export XDG_CONFIG_HOME=/etc/xdg
weston --backend=drm-backend.so --tty=1 --idle-time=0 &
// 或者
weston --backend=fbdev-backend.so --tty=1 --idle-time=0 &
```

如果没有/dev/shm/文件夹，手动创建即可：

```
mkdir /dev/shm/
```

需要开启调试的话，运行 weston 前执行以下命令：

```
export MESA_DEBUG=1
export EGL_LOG_LEVEL=debug
export LIBGL_DEBUG=verbose
export WAYLAND_DEBUG=1
```

如果有编译 Wayland Demo 的话，运行 weston 之后，可以运行 Demo，在/usr/bin 下：

wayland-scanner、weston-calibrator、weston-clickdot、weston-cliptest、weston-confine、weston-dnd、weston-eventdemo、weston-flower、weston-fullscreen、weston-image、weston-info、weston-multi-resource、weston-presentation-shm、weston-resizor、weston-scaler、weston-simple-damage、weston-simple-dmabuf-intel、weston-simple-dmabuf-v4l、weston-simple-egl、weston-simple-shm、weston-simple-touch、weston-smoke、weston-stacking、weston-subsurfaces、weston-terminal、weston-transformed。

GPU 跑分测试程序可以执行以下命令，前提是编译了 glmark2：

```
glmark2-es2-wayland
```

鼠标、键盘等输入设备，插上就可以使用。如果没有反应的话，确定是否编译了鼠标，键盘的驱动。

8.3.2 weston.ini

weston.ini 是 Wayland 的桌面配置文件，比如说想要去掉背景与状态栏，则可以修改以下的参数值。注释掉 background-image，background-color 改成黑色 0xff000000，panel-position 改成 none：

```
vi /etc/xdg/weston.ini
```

```
[shell]
# background-image=/usr/share/weston/background.png
background-color=0xff000000
panel-position=none
```

如果需要旋转屏幕的话：

```
# [output]
[output]
# name=LVDS1, mipi屏DSI-1
name=DSI-1
# mode=1680x1050, 修改成对应的分辨率
```

```
mode=480*800
# transform=90，旋转的角度
transform=90
```

更多具体参数，请参考weston.ini（5）- Arch 手册页。

8.4 Wayland 问题锦集

报错：

```
no "wayland-egl" found
```

原因可能是在之前已经编译过了没有 wayland 的图形系统，GPU 库被编译成不支持 wayland 的库，在配置 weston 的时候一定要把 Enabel opengl es support 选择上，在 tina/package/libs/gpu-um/目录下执行 mm -B 重新编译 GPU 的库，如果还报 no “wayland-egl” found，可以删除 tina/out/目录再重新编译。

著作权声明

商标声明

免责声明