

at91sam9g25

核心板

入门指导

版本 V1.0

2018 年 07 月 21 日

版本记录

版本号	说明	时间
V1.0	初始版本	2018/07/21

目录

目录.....	3
1 概述.....	4
2 目标.....	4
2.1 软件	4
2.2 硬件	4
3 Buildroot 介绍.....	4
4 文件烧录.....	6
5 开发板基本使用.....	9
5.1 基本 GPIO 使用.....	9
5.2 串口和 RS485 模式.....	10
5.3 网络接口及 ssh	11
5.4 网络代码更新	13
5.5 液晶屏驱动	14
5.6 图像界面设计	18

1 概述

本教程将引导您使用 Buildroot 创建 at91sam9g25 的开发工具和 linux 内核，并在开发板上运行应用程序来测试相关接口。

2 目标

通过本教程，您将能够：

- 使用 buildroot 编译 at91sam9g25 的文件系统和内核
- 创建一个简单的“Hello World”应用程序，并在核心板运行
- 创建 QT 程序，并在 TFT 液晶屏上显示

本教程的软件和硬件环境要求：

2.1 软件

本参考设计的软件环境要求：

- Oracle VM VirtualBox
- USB 串口驱动程序
- tftpd 文件传输工具
- putty 终端程序(其他串口终端程序也可以)
- QT 设计工具
- SAM-BA v2.1
- QT Creator

2.2 硬件

本参考设计的硬件要求：

- 具有 4 GB RAM 和 1 GB 虚拟内存的计算机（推荐）
- at91sam9g25 开发板
- 9V 电源
- 以太网线缆

3 Buildroot 介绍

Buildroot是Linux平台上用来构建嵌入式Linux系统的框架，主要由Makefile脚本和Kconfig配置文件构成的。这些自动化工具，使得Linux嵌入式系统的构建过程不再需要过多的手工参与，因此使用者上手十分方便。

Buildroot可实现交叉编译工具的编译，linux内核编译，文件系统创建等操作。

Buildroot的配置过程如下：

- 1) 在buildroot目录下，通过“make menuconfig”指令进行相关需求的配置：

```
server@server: ~/eedesign/2021/buildroot-at91
t91/output/build/linux-custom//.config
echo 'CONFIG_DEV TMPFS=y' >> /home/server/eedesign/2021/buildroot-at91/output/bui
ld/linux-custom//.config
/bin/sed -i -e "/\\<CONFIG_DEV TMPFS_MOUNT\\>/d" /home/server/eedesign/2021/build
root-at91/output/build/linux-custom//.config
echo 'CONFIG_DEV TMPFS_MOUNT=y' >> /home/server/eedesign/2021/buildroot-at91/outp
ut/build/linux-custom//.config
/bin/sed -i -e "/\\<CONFIG_GCC_PLUGINS\\>/d" /home/server/eedesign/2021/buildroo
t-at91/output/build/linux-custom//.config
echo '# CONFIG_GCC_PLUGINS is not set' >> /home/server/eedesign/2021/buildroot-a
t91/output/build/linux-custom//.config
HOSTCC scripts/kconfig/conf.o
HOSTLD scripts/kconfig/conf
scripts/kconfig/conf --olddefconfig Kconfig
#
# configuration written to .config
#
server@server:~/eedesign/2021/buildroot-at91$ make menuconfig
*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.
server@server:~/eedesign/2021/buildroot-at91$
```

- 2) 在buildroot目录下，使用“make”指令，对第二步中的配置进行编译。

```

server@server:~/eedesign/2021/buildroot-at91
/bin/sed -i -e "/\\<CONFIG_DEV TMPFS_MOUNT\\>/d" /home/server/eedesign/2021/build
root-at91/output/build/linux-custom/.config
echo 'CONFIG_DEV TMPFS_MOUNT=y' >> /home/server/eedesign/2021/buildroot-at91/outp
ut/build/linux-custom/.config
/bin/sed -i -e "/\\<CONFIG_GCC_PLUGINS\\>/d" /home/server/eedesign/2021/buildroo
t-at91/output/build/linux-custom/.config
echo '# CONFIG_GCC_PLUGINS is not set' >> /home/server/eedesign/2021/buildroot-a
t91/output/build/linux-custom/.config
HOSTCC scripts/kconfig/conf.o
HOSTLD scripts/kconfig/conf
scripts/kconfig/conf --olddefconfig Kconfig
#
# configuration written to .config
#
server@server:~/eedesign/2021/buildroot-at91$ make menuconfig

*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.

server@server:~/eedesign/2021/buildroot-at91$
server@server:~/eedesign/2021/buildroot-at91$
server@server:~/eedesign/2021/buildroot-at91$
server@server:~/eedesign/2021/buildroot-at91$ make

```

系统采用 linux5.4 内核和 buildroot2019 进行编译。

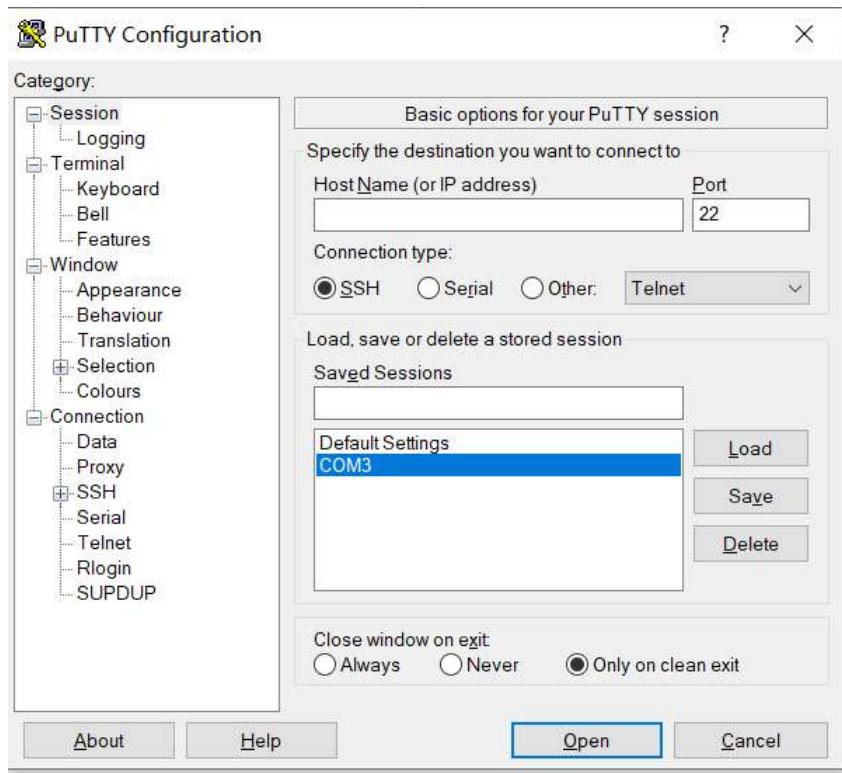
编译完成，在 output/image 目录会生成 uboot，文件系统，linux 内核 zImage，dts 等文件。

将这些文件复制到 windows 主机，后期文件烧录时会用到。

4 文件烧录

本步骤将虚拟机编译完成后 linux 内核、文件系统以及设备树烧录进实际开发板中。

- 1) 准备步骤：将/home/server/eedesign/2021/buildroot-at91/output/images 目录中所有文件共享复制到本地 PC；
- 2) 连接开发板的调试串口，开发板上电，并进入 uboot 模式。（linux 启动后，无法再烧录文件）
 - 连接好设备的调试串口至电脑串口硬件，打开 PuTTY 工具，并设置串口波特率为 115200。



- 设备上电，在3秒倒计时的时候点任意键【回车】，出现U-Boot就说明可以进行下一步烧录；

```

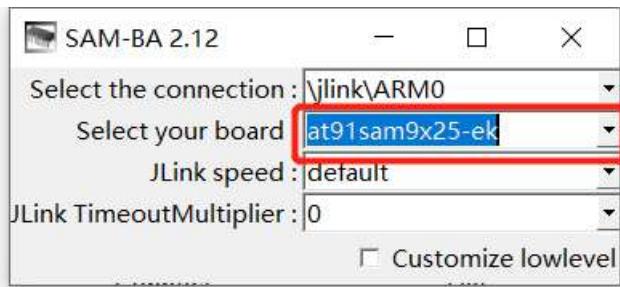
<debug_uart>
U-Boot 2019.04-linux4sam_6.1 (May 08 2021 - 11:16:39 +0800)

CPU: AT91SAM9G25
Crystal frequency:      12 MHz
CPU clock       : 400 MHz
Master clock     : 133.333 MHz
DRAM: 128 MiB
NAND: 256 MiB
MMC: Atmel mci: 0, Atmel mci: 1
Loading Environment from NAND... *** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Net:
Error: ethernet@f802c000 address not set.
eth-1: ethernet@f802c000
Hit any key to stop autoboot: 0
U-Boot>
U-Boot>
U-Boot>

```

- 3) 打开 SAM-BA v2.1 工具，选择对应的开发板，敲击回车；



- 3) 需要烧录格式化 NAND flash, 然后烧写 at91bootstrap, uboot, zImage, dts, rootfs.ubi 等文件, 可以在 sam-ba 的命令行中, 直接粘贴下面的 tcl 脚本;

NANDFLASH::Init

NANDFLASH::NandHeaderValue HEADER 0xc0c00405

NANDFLASH::EraseAll

NANDFLASH::SendBootFilePmecc

send_file {NandFlash} "E:/work/images/at91sam9g25ek.dtb" 0x00180000 0

send_file {NandFlash} "E:/work/images/u-boot.bin" 0x00040000 0

send_file {NandFlash} "E:/work/images/zImage" 0x00200000 0

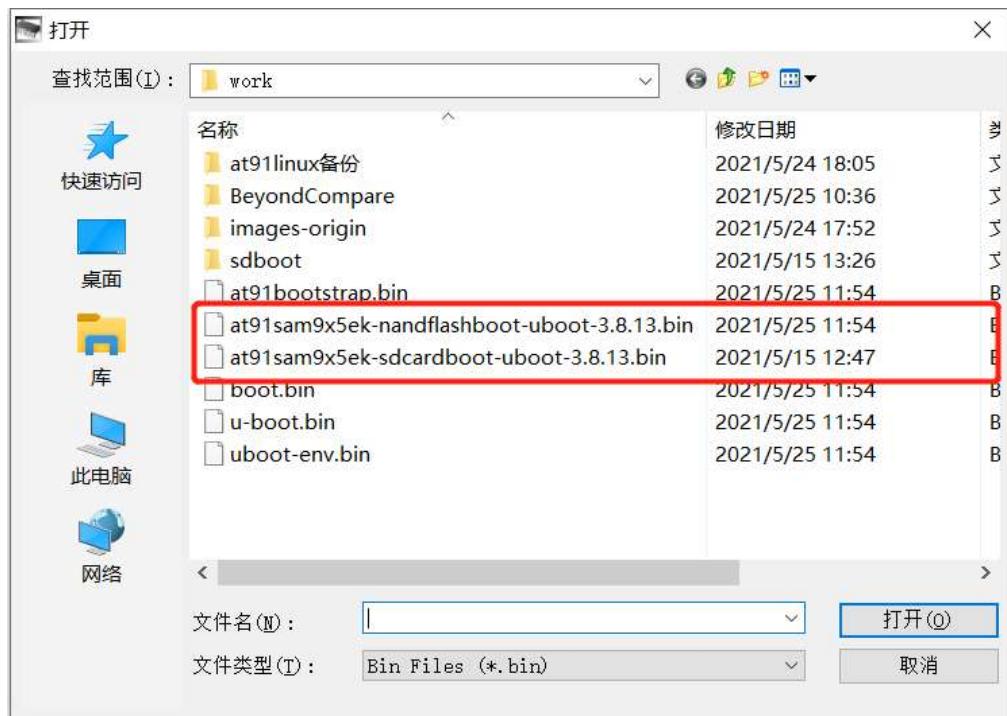
NANDFLASH::NandSetTrimffs 1

send_file {NandFlash} "E:/work/images/rootfs.ubi" 0x00800000 0

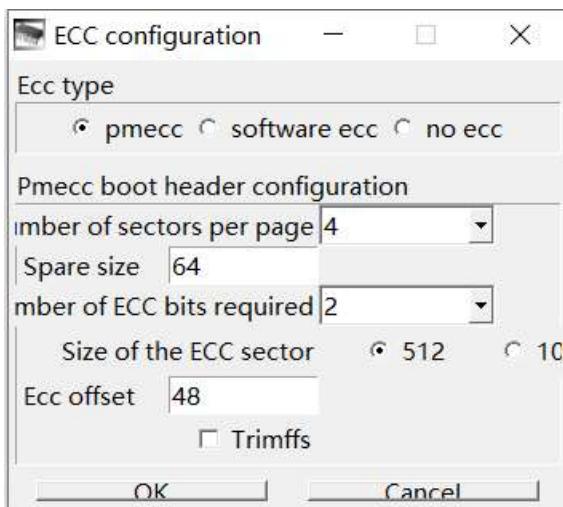
注意:

- “E:/work/images/”修改为实际的目录

- 在运行到 SendBootFilePmecc 命令时，会弹出一个选择文件框，要求选择 at91bootstrap，此时选择“at91sam9x5ek-nandflashboot-uboot-3.8.13.bin”



- 当出现 ECC 配置相关的提示框，直接回车确认；



5 开发板基本使用

5.1 基本 GPIO 使用

GPIO 在 linux 中为文件。对某一 GPIO 口进行电平修改操作时，可以通过以下步骤进行操作：

- 1) 进入需要修改的 GPIO 口目录，如 /sys/class/leds/pc4；
- 2) 使用下述指令：echo 1 > brightness # 把 1 (高电平) 重定向给 brightness。从而能够实现对 GPIO 口电平的输出控制。

```

gpio      misc      pwm      spi_master      watchdog
# cd leds/
# ls
pb18  pc4  pd21
# cd pc4
# pwd
/sys/class/leds/pc4
#
/leds/pc4
# /sys/class/leds/pc4
-sh: /sys/class/leds/pc4: Permission denied
# ^C
# ls
brightness      max_brightness      subsystem      uevent
device          power              trigger
# echo 1 > brightness
# dmesg |grep spi
atmel_spi f0000000.spi: Using dma0chan0 (tx) and dma0chan1 (rx) for DMA transfer
s
atmel_spi f0000000.spi: Atmel SPI Controller version 0x212 at 0xf0000000 (irq 27)
)
fb_st7789v spi0.0: Missing info about 'dc' gpio. Aborting.
fb_st7789v: probe of spi0.0 failed with error -22

```

5.2 串口和 RS485 模式

AT91SAM9G25 自带调试接口，开机时会输出设备启动信息（再系统启动后也可以通过 dmesg 命令查看启动记录）。

调试串口的默认波特率为 115200 8bit 无校验模式，且无硬件流控。

```

COM3 - PuTTY
<debug_uart>

U-Boot 2019.04-linux4sam_6.1 (May 08 2021 - 11:16:39 +0800)

CPU: AT91SAM9G25
Crystal frequency:      12 MHz
CPU clock :        400 MHz
Master clock :    133.333 MHz
DRAM: 128 MiB
NAND: 256 MiB
MMC: Atmel mci: 0, Atmel mci: 1
Loading Environment from NAND... *** Warning - bad CRC, using default environment

In:   serial
Out:  serial
Err:  serial
Net:
Error: ethernet@f802c000 address not set.
eth-1: ethernet@f802c000
Hit any key to stop autoboot:  0
U-Boot>
U-Boot>
U-Boot> █

```

通过串口可传输文件，如可采用 z-modem 协议来发送文件：

- 1) 在嵌入式 linux 的 shell 中输入“lrz”，进入等待文件接收状态；
- 2) 在支持 z-modem 协议的串口工具，如 windows 自带的超级终端，按照 z-modem 协议发送文件。

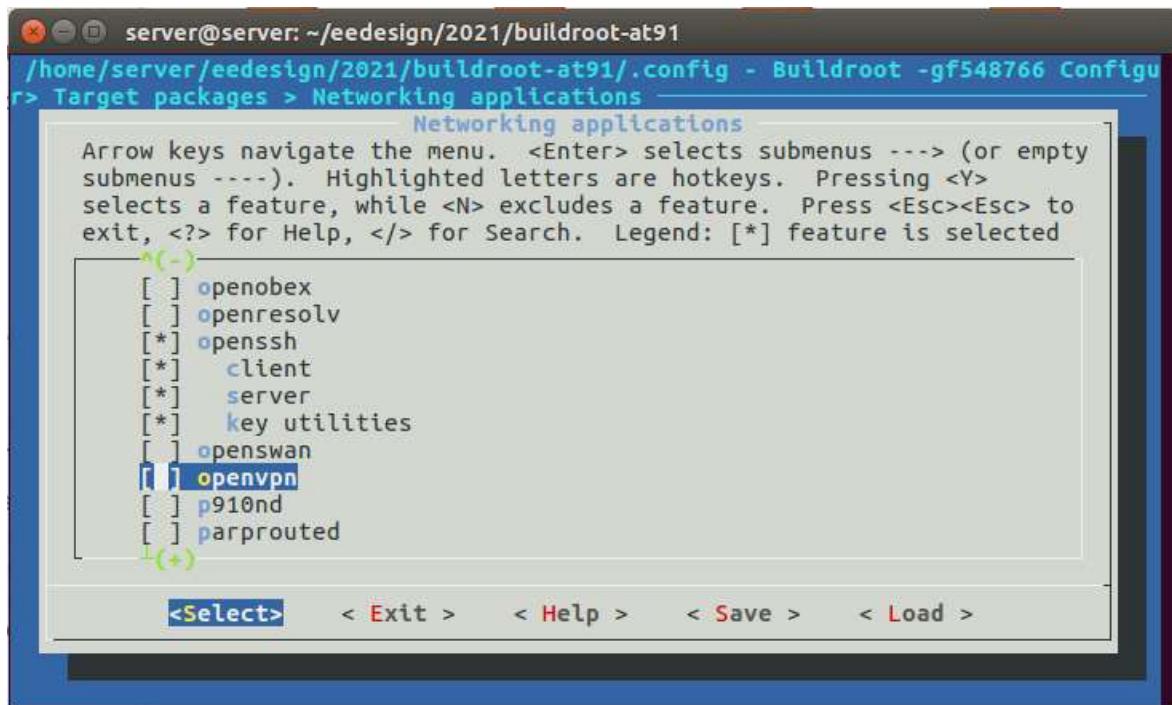
串口 1 (/dev/ttyS1) 通过 DTS 设置为 485 模式，具体修改可参考 dts 文件。

5.3 网络接口及 ssh

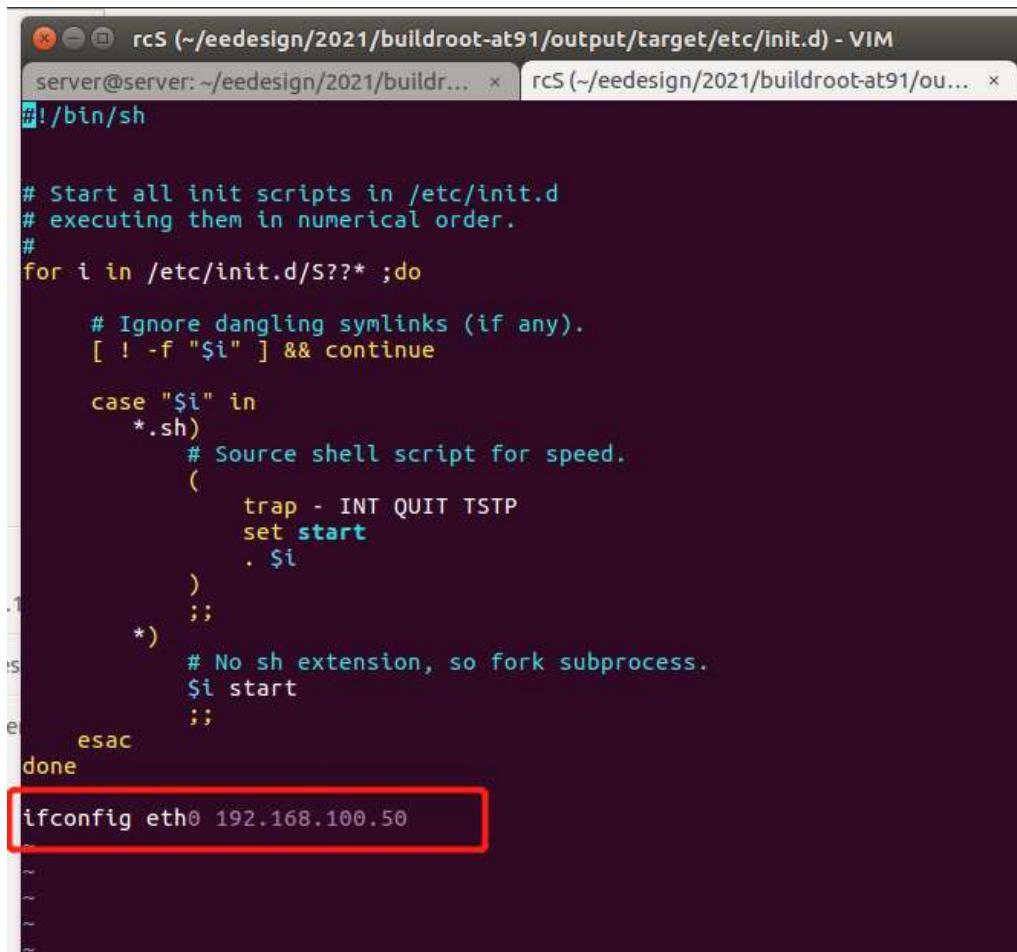
AT91sam9g25 芯片内部含有 2 个 100M MAC。在我们的底板中，有一个 100M PHY 芯片 DP83848。在 linux 内核配置时，配置加载 DP83848 驱动，即可使用 100M 以太网通信功能。

另外，可以配置 buildroot 文件系统中的 OPENSSH 功能，开启 sshd 服务，具体配置步骤如下：

- 1) 在 buildroot 目录下输入“make menuconfig”指令，进入 Target packages → Networking applications，选中 openssh；



- 2) “make”指令编译，编译完成后，进入“/home/server/eedesign/2021/buildroot-at91/output/target /etc/ssh”目录，输入“vi sshd_config”修改 sshd 的配置文件；
- 3) 将 PermitRootLogin 设置为 yes，PermitEmptyPasswords 设置为 yes（如果没有设置 root 用户的密码）；
- 4) 进入“etc/init.d”目录，打开 rcS 文件，增加静态 IP 地址设置语句：



```
#!/bin/sh

# Start all init scripts in /etc/init.d
# executing them in numerical order.
#
for i in /etc/init.d/S??* ;do

    # Ignore dangling symlinks (if any).
    [ ! -f "$i" ] && continue

    case "$i" in
        *.sh)
            # Source shell script for speed.
            (
                trap - INT QUIT TSTP
                set start
                . $i
            )
            ;;
        *)
            # No sh extension, so fork subprocess.
            $i start
            ;;
    esac
done

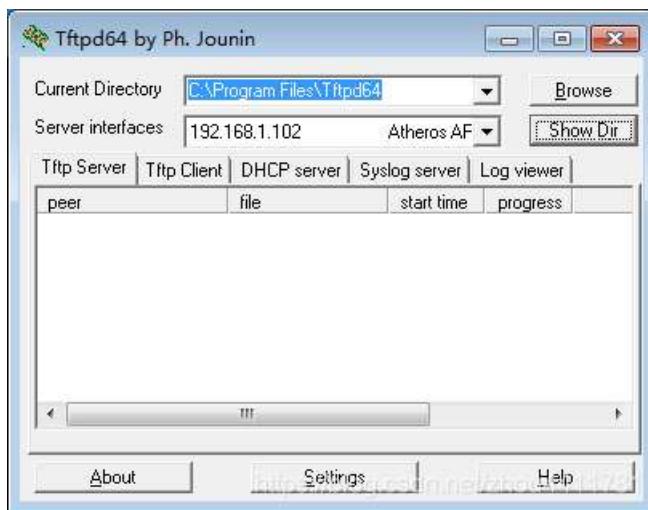
ifconfig eth0 192.168.100.50
```

5.4 网络代码更新

可通过tftp协议，实现网络数据传输。如将windows平台设置为tftp服务器，在开发板上，通过tftp客户端下载文件或数据。

具体步骤如下：

- 1) 配置开发板IP地址，和tftp服务器端保证网络可达（如在一个局域网子网）；
- 2) 配置传输文件夹路径以及绑定的网口；



3) 在 Linux shell 下上输入指令，`tftp -g -r ifconfig.txt 192.168.1.102`

即可从 ssh 服务器 192.168.1.102，下载 ifconfig.txt 到开发板

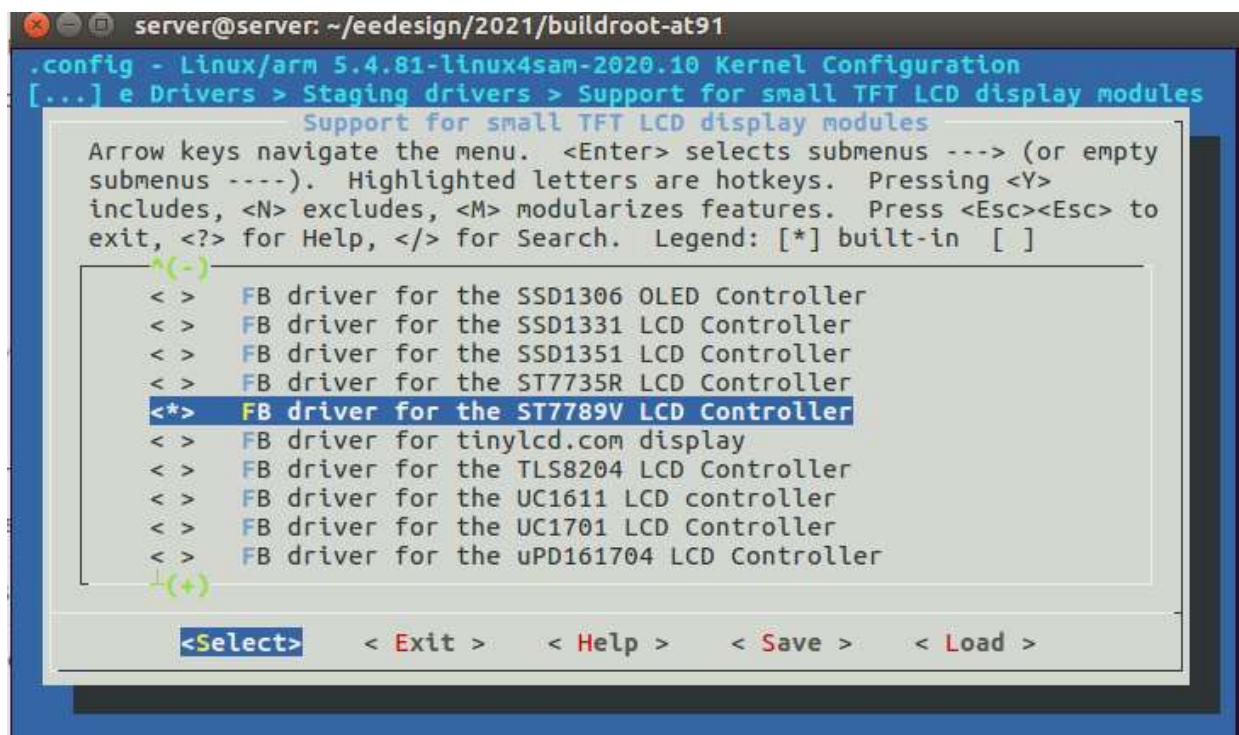
#所传第一个参数是传输的文件名字，第二个是 PC 端的 IP 地址。

5.5 液晶屏驱动

以 ST7789 SPI 接口的液晶驱动芯片为例，对 Linux 与液晶显示相关的驱动进行说明。

ST7789 采用 SPI 接口进行通信。linux 在与 ST7789 进行通信时，采用“frame buffer”先将图像数据进行缓存，也就是 ST7789 的驱动含有 SPI 驱动和 FB 驱动。

1) 在 buildroot-at91 目录下，输入“make linux-menuconfig”，依次进入 Device Driver-> Staging drivers ->Support for small TFT LCD display modules-> FB driver for the ST7789V LCD controller；单击空格打上“*”，退出保存；



2) 修改 fb_st7789v.c 文件，在 init_display 函数中，对 st7789 芯片的初始化进行修改（修复屏幕颜色问题）：

```
//fb_st7789v.c 文件部分代码
static int init_display(struct fbtft_par *par)
{
    par->fbftops.reset(par);
    mdelay(50);
    write_reg(par,0x36,0x00);
    write_reg(par,0x3A,0x05);
    write_reg(par,0xB2,0x0C,0x0C,0x00,0x33,0x33);
    write_reg(par,0xB7,0x35);
    write_reg(par,0xBB,0x19);
    write_reg(par,0xC0,0x2C);
    write_reg(par,0xC2,0x01);
    write_reg(par,0xC3,0x12);
    write_reg(par,0xC4,0x20);
    write_reg(par,0xC6,0x0F);
    write_reg(par,0xD0,0xA4,0xA1);
    write_reg(par,0xE0,0xD0,0x04,0x0D,0x11,0x13,0x2B,0x3F,0x54,0x4C,0x18,0x0D,0x0B,0x1F,0x23);
    write_reg(par,0xE1,0xD0,0x04,0x0C,0x11,0x13,0x2C,0x3F,0x44,0x51,0x2F,0x1F,0x20,0x23);
    write_reg(par,0x21);
```

```
//write_reg(par,0x20);
write_reg(par,0x11);
mdelay(50);
write_reg(par,0x29);
mdelay(200);
return 0;
}
```

- 5) 修改“/home/server/eedesign/2021/buildroot-at91/output/build/linux-custom/arch/arm/boot/dts/”目录下的 at91sam9g25ek.dts 文件，该文件主要增加了 st7789 spi 接口说明，以及复位和命令/数据选择引脚说明：

```
//at91sam9g25ek-new.dts 部分代码
&spi0 {
    status = "okay";
    st7789v@0 {
        status = "okay";
        compatible = "sitronix,st7789v";
        reg = <0>;
        spi-max-frequency = <32000000>;
        rotate = <0>;
        spi-cpol;
        spi-cpha;
        //bgr;
        fps = <30>;
        buswidth = <8>;
        reset-gpios = <&pioB 17 GPIO_ACTIVE_HIGH>;
        dc-gpios = <&pioC 31 GPIO_ACTIVE_LOW>;
        debug = <0>;
    };
};
```

- 6) 修改 at91sam9x5cm.dtsi 文件，增加 GPIO，用于屏幕背光控制：

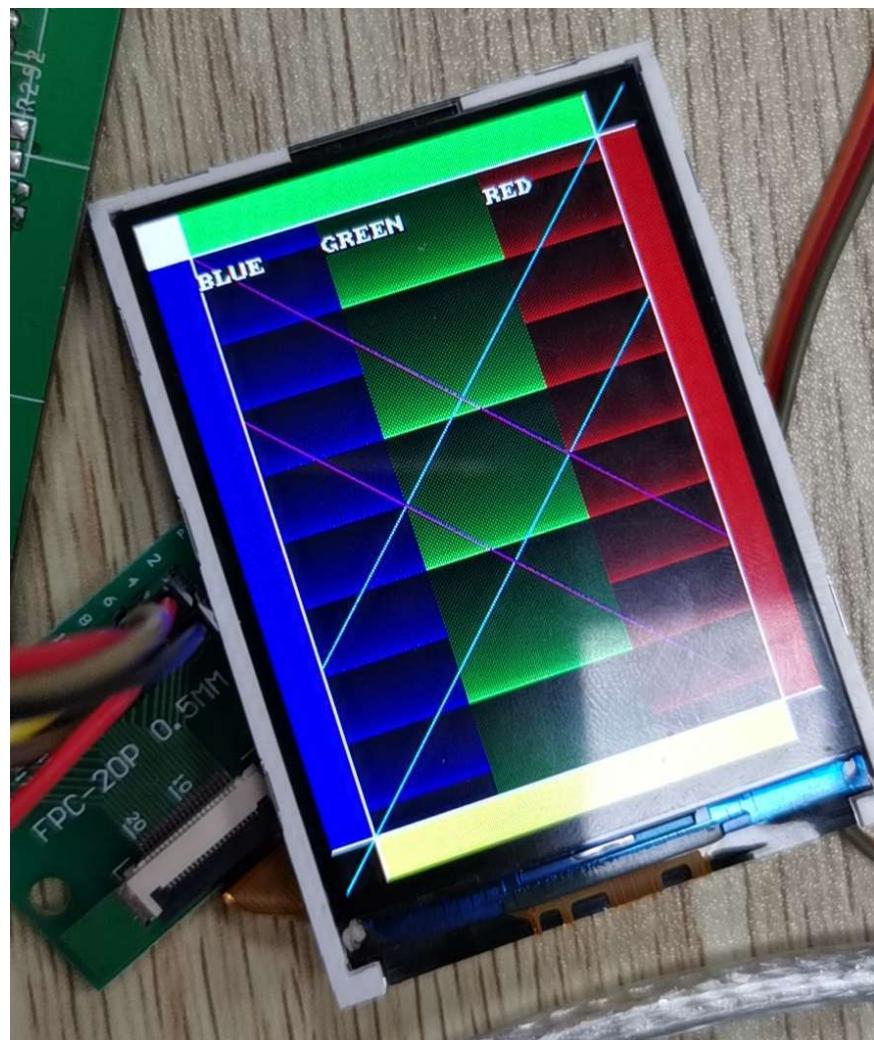
```
pc4 {
    label = "pc4";
    gpios = <&pioC 4 GPIO_ACTIVE_LOW>;
```

```
};
```

LCD 液晶屏的硬件连接关系如下表：

LCD 液晶屏	Linux 系统
LEDK	PC4
LEDA	VCC5V
RESET	PB17
CS	PA14
SCL	PA13
RS	PC31
SDA	PA12

进入 Linux shell 后，输入“fb-test”指令，会出现 RGB 字母提示以及对应的三种颜色，如下图；

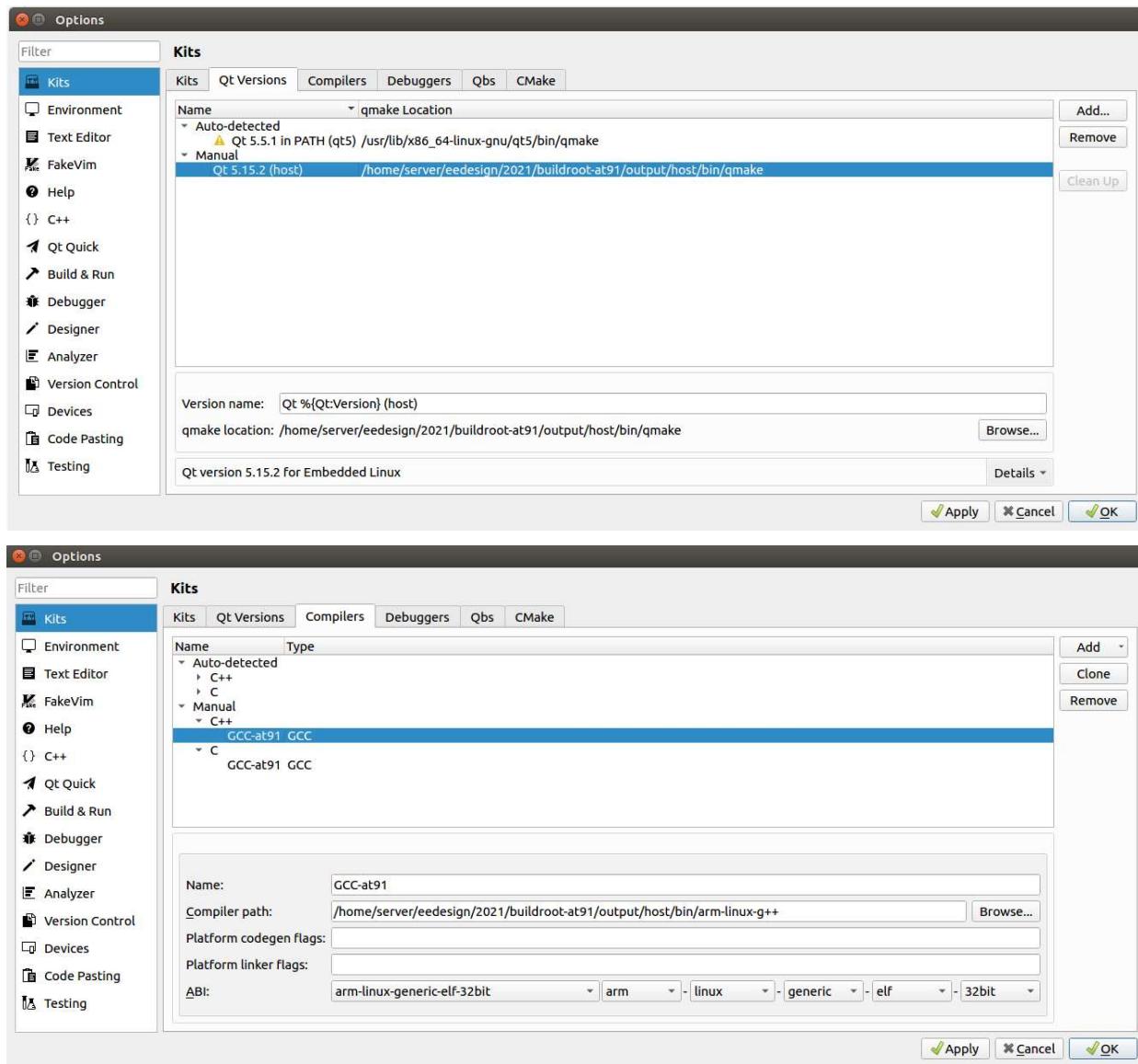


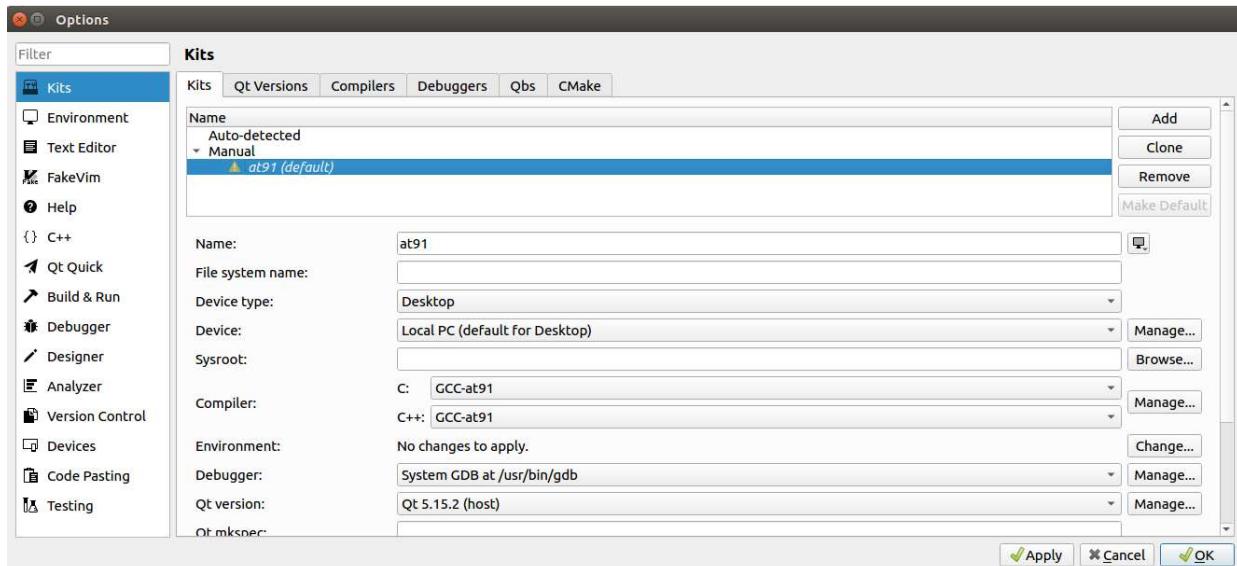
5.6 图像界面设计

QT 可以进行图像界面的设计，Buildroot 里面指定编译 QT 库，并加载到开发板的文件系统中。

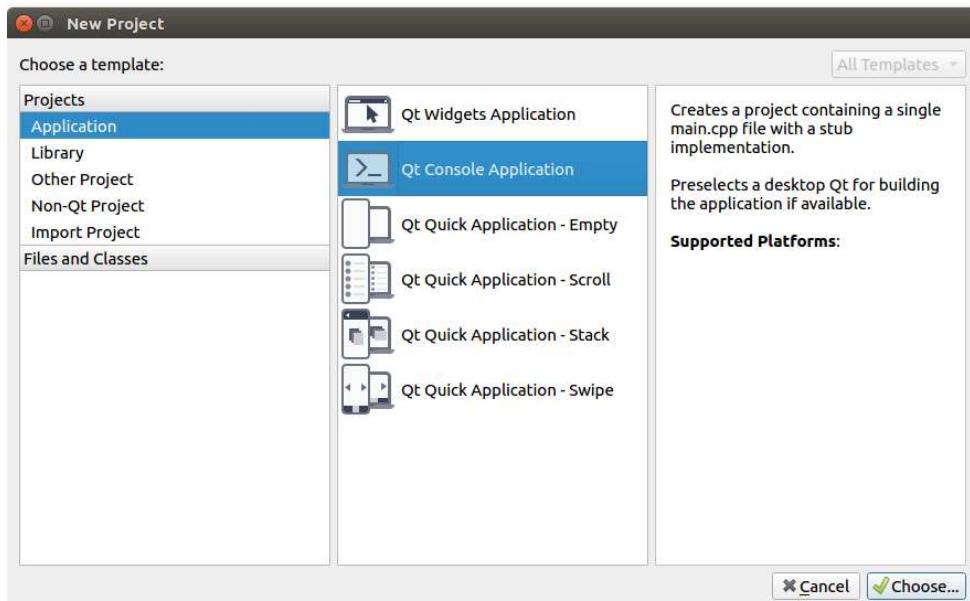
创建了一个简单的“Hello World”应用程序，并使其在核心板上运行：

1) 预先配置好 QT 交叉编译链：





2) 创建新工程



3) 在工程页面输入“hello world”语句

```

main.cpp*
1 #include <QCoreApplication>
2
3 int main(int argc, char *argv[])
4 {
5     QCoreApplication a(argc, argv);
6     printf("hello world!\n");
7
8     return a.exec();
9 }
10

```

4) 编译



- 5) 利用 tftpd 文件传输工具将构建生成的 QT 文件（此处是 untitled2）传输进 Linux 开发板中，输入“tftp -g -r untitled2 192.168.100.200”传输，输入“./untitled2”运行上述文件：



如果 QT 设计的程序含有图像界面，且需要在屏幕上显示，则需要配置相应的环境变量。

```
export QT_QPA_PLATFORM=linuxfb;fb=linuxfb;fb=/dev/fb0:size=240x320:mmSize=240x320:offset=0x0:tty=/dev/tty1
```