

八 基于 Linux-C 的测试程序

iTOP-4412 开发板可以运行的文件系统很多，在具体的文件系统中实现特定功能前，可以使用 Linux-C 程序来测试硬件以及驱动。而且这些程序很容易移植到 Android、Qt/E 以及最小文件系统上。

特别提醒：Linux-C 程序是跨平台的，只要按照下面介绍的方法去编译，就可以将 Linux-C 的程序和 Android 系统一起运行，使用 Linux-C 的程序测试我们关注的内容。本质上，我们可以这样理解，Android 只是一个大的文件而已，以下面第一个 helloworld 为例，Linux 内核上运行着两个程序“helloworld”+“Android”。

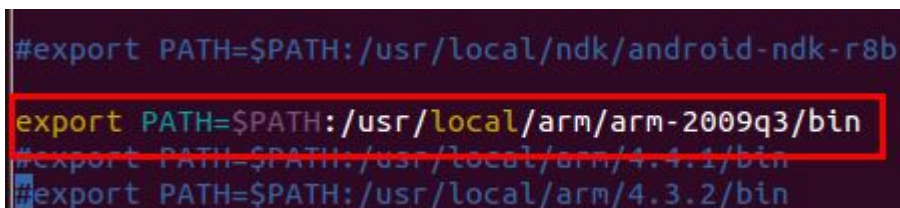
Linux-C 的测试程序源码和可执行程序在网盘目录“iTOP4412 开发板资料汇总（不含光盘内容）\iTOP-4412 开发板系统源码及镜像（其他）\小模块的测试程序”下。

8.1 测试程序的编译和运行

8.1.1 编译环境的设置

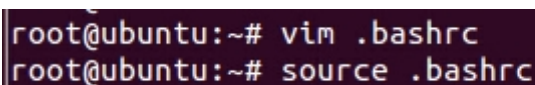
C 程序的应用程序在 Android 上运行，使用的编译器是 gcc4.4.1。编译器的安装方法参考第五章。

如下图所示，修改环境变量。



```
#export PATH=$PATH:/usr/local/ndk/android-ndk-r8b
export PATH=$PATH:/usr/local/arm/arm-2009q3/bin
#export PATH=$PATH:/usr/local/arm/4.4.1/bin
export PATH=$PATH:/usr/local/arm/4.3.2/bin
```

修改完之后，更新一下环境变量，如下图。



```
root@ubuntu:~# vim .bashrc
root@ubuntu:~# source .bashrc
```

如下图所示，输入“arm”，然后按“TAB”键，会显示后面需要用到的编译器“arm-none-linux-gnueabi-gcc-4.4.1”。

```
root@ubuntu:~# arm
arm2hpd1
arm-linux-addr2line
arm-linux-ar
arm-linux-as
arm-linux-c++
arm-linux-c++filt
arm-linux-cpp
arm-linux-g++
arm-linux-gcc
arm-linux-gcc-4.3.2
arm-linux-gcc-4.4.1
arm-none-linux-gnueabi-addr2line
arm-none-linux-gnueabi-ar
arm-none-linux-gnueabi-as
arm-none-linux-gnueabi-c++
arm-none-linux-gnueabi-c++filt
arm-none-linux-gnueabi-cpp
arm-none-linux-gnueabi-g++
arm-none-linux-gnueabi-gcc
arm-none-linux-gnueabi-gcc-4.3.2
arm-none-linux-gnueabi-gcc-4.4.1
arm-none-linux-gnueabi-gcov
```

8.1.2 编译 helloworld

程序 helloworld.c 的源码如下。

```
#include <stdio.h>

int main(void)
{
    printf("Hello World!\n");
}
```

编译 helloworld 程序，输入命令“arm-none-linux-gnueabi-gcc-4.4.1 -o helloworld helloworld.c -static”，如下图所示。

```
root@ubuntu:/home/topeet/Android-c# arm-none-linux-gnueabi-gcc-4.4.1 -o helloworld helloworld.c -static
```

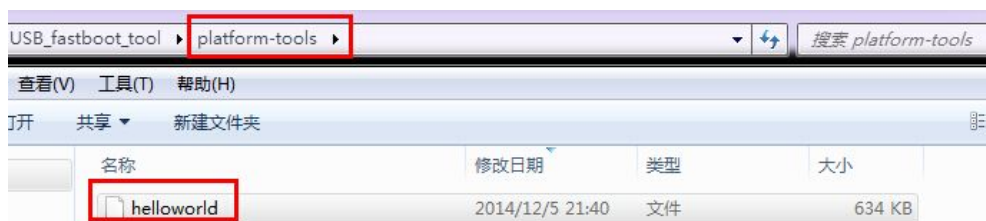
如下图，生成可执行文件 helloworld。

```
root@ubuntu:/home/topeet/Android-c# ls
helloworld helloworld.c
```

8.1.3 上传 helloworld 到开发板

开发板运行出厂自带的 Android4.0.3 系统，连接串口控制台，启动开发板，Android 运行之后接上 OTG 线。

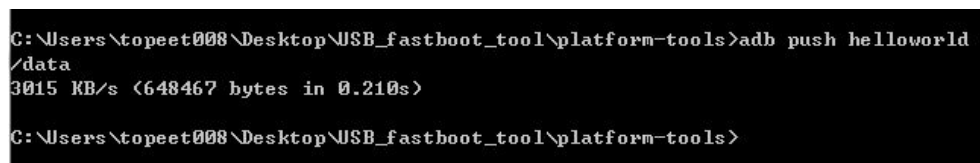
将可执行文件 helloworld 拷贝到 fastboot 烧写目录中，如下图所示，这个目录是烧写 fastboot 工具所在的目录。fastboot 工具的使用方法参考 3.6 小节。



开发板的 Android 系统运行稳定后，将 OTG 接口和电脑的 USB 连接，打开“USB_fastboot_tool\platform-tools”目录中的“cmd.exe”，如下图所示。



如下图所示，输入命令“adb push helloworld /data”，将程序上传到开发板的“/data”目录中。



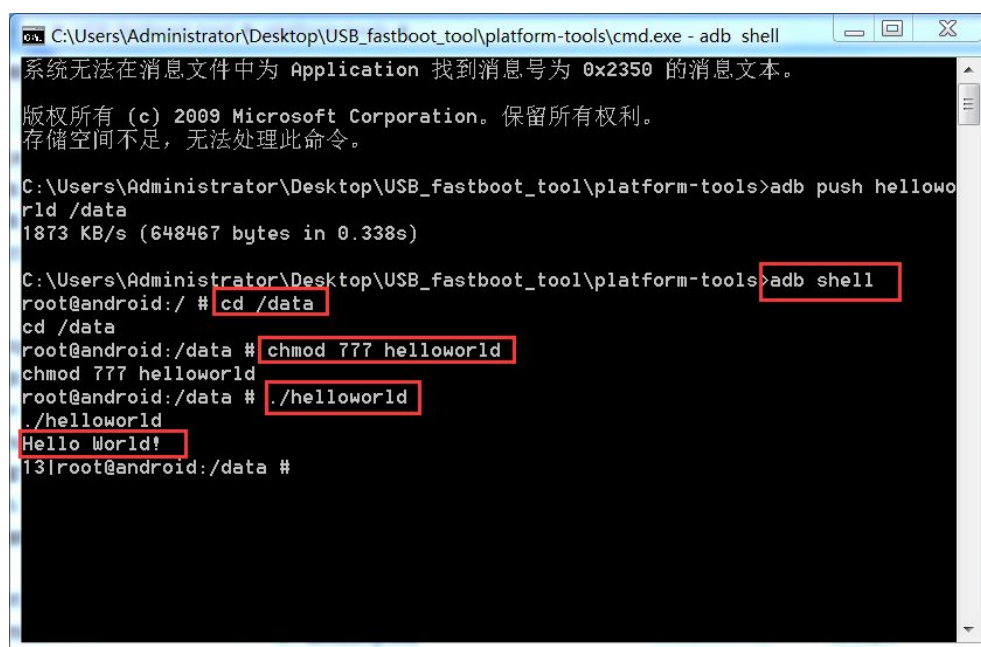
当然也可以通过手机助手、TF 卡或者 U 盘来上传可执行文件 helloworld。

8.1.4 修改程序权限和运行 helloworld

如下图所示，在超级终端中，输入“cd /data”进入“/data”目录，使用“chmod 777 helloworld”修改权限，最后输入命令“./helloworld”运行程序。超级终端中会打印出“Hello world！”，表明程序运行成功。

```
root@android:/ #  
root@android:/ # cd /data  
root@android:/data # chmod 777 helloworld  
root@android:/data # ./helloworld  
Hello World!  
13|root@android:/data #
```

除了通过控制台修改权限和运行程序，还可以通过 adb 控制台修改权限和运行程序。如下图所示，上传 helloworld 程序之后，使用命令“adb shell”打开 adb 命令行（更多 adb 命令参考 3.6.4 小节），使用“cd /data”进入 helloworld 程序上传目录“/data”，使用命令“chmod 777 helloworld”修改权限，接着使用“./helloworld”运行 helloworld 可执行程序。可以看到 adb 控制台打印“Hello World！”。



```
C:\Users\Administrator\Desktop\USB_fastboot_tool\platform-tools>cmd.exe - adb shell  
系统无法在消息文件中为 Application 找到消息号为 0x2350 的消息文本。  
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。  
存储空间不足，无法处理此命令。  
  
C:\Users\Administrator\Desktop\USB_fastboot_tool\platform-tools>adb push helloworld /data  
1873 KB/s (648467 bytes in 0.338s)  
  
C:\Users\Administrator\Desktop\USB_fastboot_tool\platform-tools>adb shell  
root@android:/ # cd /data  
cd /data  
root@android:/data # chmod 777 helloworld  
chmod 777 helloworld  
root@android:/data # ./helloworld  
./helloworld  
Hello World!  
13|root@android:/data #
```

后面的测试可以使用串口控制台运行程序，也可以使用 adb 命令运行程序。

8.2 Led 灯的测试

Led 的测试代码在网盘目录“iTOP4412 开发板资料汇总（不含光盘内容）\iTOP-4412 开发板系统源码及镜像（其他）\小模块的测试程序”。是该目录下的“iTOP-4412-Android4.0-LinuxC-Led_Vx.x”，解压之后得到 led 的二进制文件和源码。

如下图，编译 leds 测试程序，在 Ubuntu 系统中，输入编译命令 “arm-none-linux-gnueabi-gcc-4.4.1 -o leds leds.c -static”，生成 leds 可执行程序。

```
root@ubuntu:/home/topeet/Android-C# arm-none-linux-gnueabi-gcc-4.4.1 -o leds leds.c -static
root@ubuntu:/home/topeet/Android-C# ls
helloworld helloworld.c leds leds.c
root@ubuntu:/home/topeet/Android-C#
```

如下图，修改开发板/data 目录权限位 777，上传文件到开发板的 “/data”。在 cmd 命令行中，输入 adb 传文件的命令 “adb push leds /data”。

```
C:\Users\Administrator\Desktop\platform-tools>adb push leds /data
1184 KB/s (650397 bytes in 0.536s)
C:\Users\Administrator\Desktop\platform-tools>
```

如下图，修改测试程序的权限。在超级终端中，输入命令 “cd /data”，输入修改权限命令 “chmod 777 leds”。

```
root@android:/data #
root@android:/data # chmod 777 leds
root@android:/data #
```

如下图，在超级终端中，输入 “./leds” 命令运行程序。小灯会闪烁 10 次，表明程序运行成功。

```
root@android:/data #
root@android:/data # ./leds
leds light on and[ 1877.585719] LEDS_CTL DEBUG:Device Opened Success!
[ 1877.591105] debug: leds_ioctl cmd is 0
[ 1877.594035] debug: leds_ioctl cmd is 0
off 5 times
open /dev/leds success
ioctl leds 9 times
[ 1878.597949] debug: leds_ioctl cmd is 1
[ 1878.600242] debug: leds_ioctl cmd is 1
ioctl leds 8 time[ 1879.604343] debug: leds_ioctl cmd is 0
[ 1879.608409] debug: leds_ioctl cmd is 0
s
[ 1880.611983] debug: leds_ioctl cmd is 1
[ 1880.614298] debug: leds_ioctl cmd is 1
ioctl leds 7 time[ 1881.618394] debug: leds_ioctl cmd is 0
```


8.3 Buzzer 蜂鸣器的测试

如下图，编译 buzzer 测试程序，在 Ubuntu 系统中，输入编译命令 “arm-none-linux-gnueabi-gcc-4.4.1 -o buzzer buzzer.c -static”，生成 buzzer 可执行程序。

```
root@ubuntu:/home/topeet/Android-C# arm-none-linux-gnueabi-gcc-4.4.1 -o buzzer buzzer.c -static
root@ubuntu:/home/topeet/Android-C# ls
buzzer buzzer.c helloworld helloworld.c leds leds.c
root@ubuntu:/home/topeet/Android-C#
```

如下图，上传文件到开发板的 “/data”，在 cmd 命令行中，输入 adb 传文件的命令 “adb push buzzer /data”。

```
C:\Users\Administrator\Desktop\platform-tools>adb push buzzer /data
1171 KB/s (650383 bytes in 0.542s)
C:\Users\Administrator\Desktop\platform-tools>
```

如下图，修改测试程序的权限。在超级终端中，输入命令 “cd /data”，输入修改权限命令 “chmod 777 buzzer”。

```
root@android:/data # chmod 777 buzzer
root@android:/data #
```

如下图，在超级终端中，输入 “./buzzer” 命令运行程序。蜂鸣器响 3 声，表明程序运行成功。

```
root@android:/data # ./buzzer
buzzer light on at [ 137.546190] itop4412_buzzer_ioctl: cmd = 1
and off 5 times
open /dev/buzzer_ctl success
ioctl buzzer 2 times
[ 138.549995] itop4412_buzzer_ioctl: cmd = 0
ioctl buzzer 1 times [ 139.553627] itop4412_buzzer_ioctl: cmd = 1
mes
[ 140.558478] itop4412_buzzer_ioctl: cmd = 0
ioctl buzzer 0 times [ 141.561497] itop4412_buzzer_ioctl: cmd = 1
mes
[ 142.565786] itop4412_buzzer_ioctl: cmd = 0
root@android:/data #
```

8.4 ADC 数模转换的测试

如下图，编译 adctest 测试程序，在 Ubuntu 系统中，输入编译命令 “arm-none-linux-gnueabi-gcc-4.4.1 -o adctest adctest.c -static”，生成 adctest 可执行程序。

```
root@ubuntu:/home/topeet/Android-c# arm-none-linux-gnueabi-gcc-4.4.1 -o adctest
adctest.c -static
root@ubuntu:/home/topeet/Android-c# ls
adctest  buzzertest  helloworld  ledtest
adctest.c  buzzertest.c  helloworld.c  ledtest.c
root@ubuntu:/home/topeet/Android-c#
```

如下图，在 cmd 命令行中，输入 adb 传文件的命令 “adb push adctest /data”，上传文件到开发板的 “/data”。打开 adb 命令行，进入 data 目录，修改 adctest 权限。运行 adctest 程序，控制台打印当前电阻值；然后旋转滑动变阻器的旋钮，再次运行 adctest 程序，可以看到电阻值变化了。

```
C:\Users\Administrator\Desktop\USB_fastboot_tool\platform-tools>adb push adctest
/data
2099 KB/s (649264 bytes in 0.302s)

C:\Users\Administrator\Desktop\USB_fastboot_tool\platform-tools>adb shell
root@android:/ # cd /data
cd /data
root@android:/data # chmod 777 adctest
chmod 777 adctest
root@android:/data # ./adctest
./adctest
adc ready!
open adc success!
res value is 5802
18|root@android:/data # ./adctest
./adctest
adc ready!
open adc success!
res value is 5555
18|root@android:/data #
```

8.5 串口的测试

如下图，编译串口测试程序，在 Ubuntu 系统中，输入编译命令 “arm-none-linux-gnueabi-gcc -o uart_write_read uart_write_read.c -static”，生成 uart_write_read 可执行文件”。

```
root@ubuntu:/home/topeet/Android-C# arm-none-linux-gnueabi-gcc -o uart_write_read
uart_write_read.c -static
root@ubuntu:/home/topeet/Android-C# ls
leds  leds.c  uart_write_read  uart_write_read.c
root@ubuntu:/home/topeet/Android-C#
```

将可执行文件使用命令 “adb push uart_write_read /data”，但是可能会出现如下图错误，是因为你要把文件下载的目的文件夹/data 权限受到限制。

```
d /data
adb: error: failed to copy 'uart_write_read' to '/data/uart_write_read': Permission denied
C:\Users\xunwei\Desktop\USB_fastboot_tool\platform-tools>
```

在超级终端下输入 “chmod 777 /data” ，按回车键即可。

```
-----
10|root@android:/ # chmod 777 /data
root@android:/ #
```

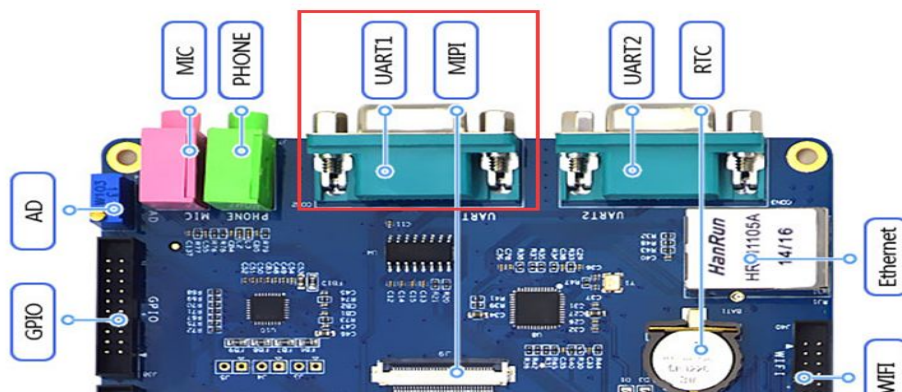
再次执行命令 “adb push uart_write_read /data ” ，可以看到文件下载成功。

```
C:\Users\xunwei\Desktop\USB_fastboot_tool\platform-tools>adb push uart_write_read /data
[100%] /data/uart_write_read
C:\Users\xunwei\Desktop\USB_fastboot_tool\platform-tools>
```

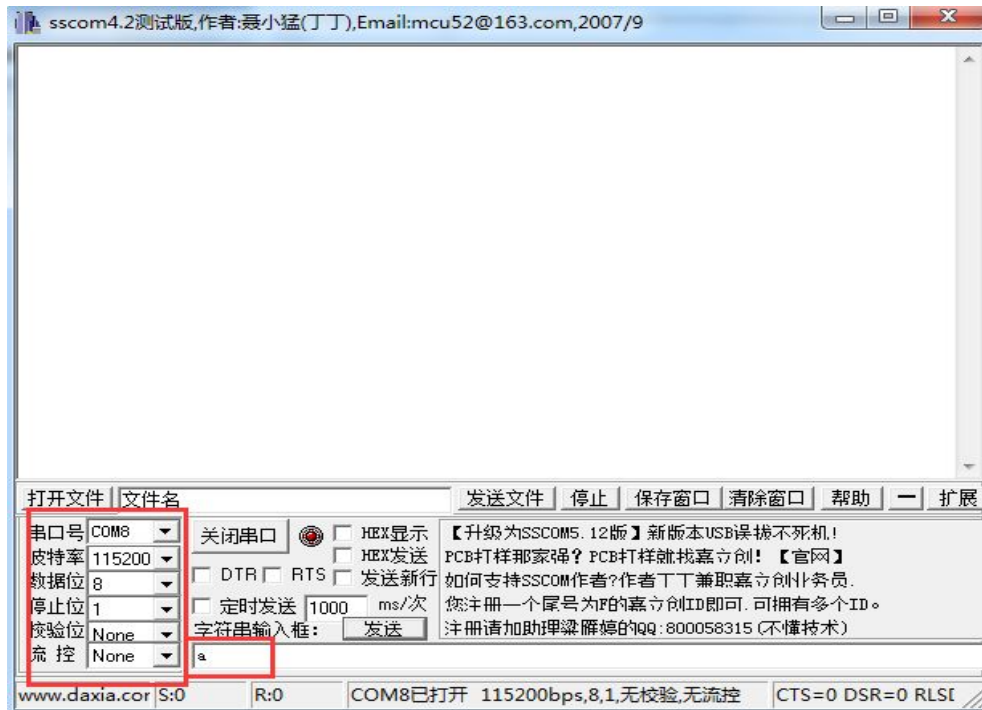
进入到 “/data” 目录下，执行命令 “chmod 777 uart_write_read” 。

```
-----
root@android:/data # chmod 777 uart_write_read
root@android:/data #
```

如下图，将 iTOP-4412 开发板红色框中的串口和上位机相连。该串口的参数和调试串口的参数相同 。



打开串口助手，设置好参数，在字符输入框中输入字符，下图中的串口号需要在设备管理器中查看。在网盘 “iTOP-4412 开发板资料汇总（不含光盘资料）\01_iTOP-4412 开发板所需 PC 软件（工具）\02_超级终端（串口调试助手）” 目录下可以得到串口助手的压缩包 “串口助手.zip” 。

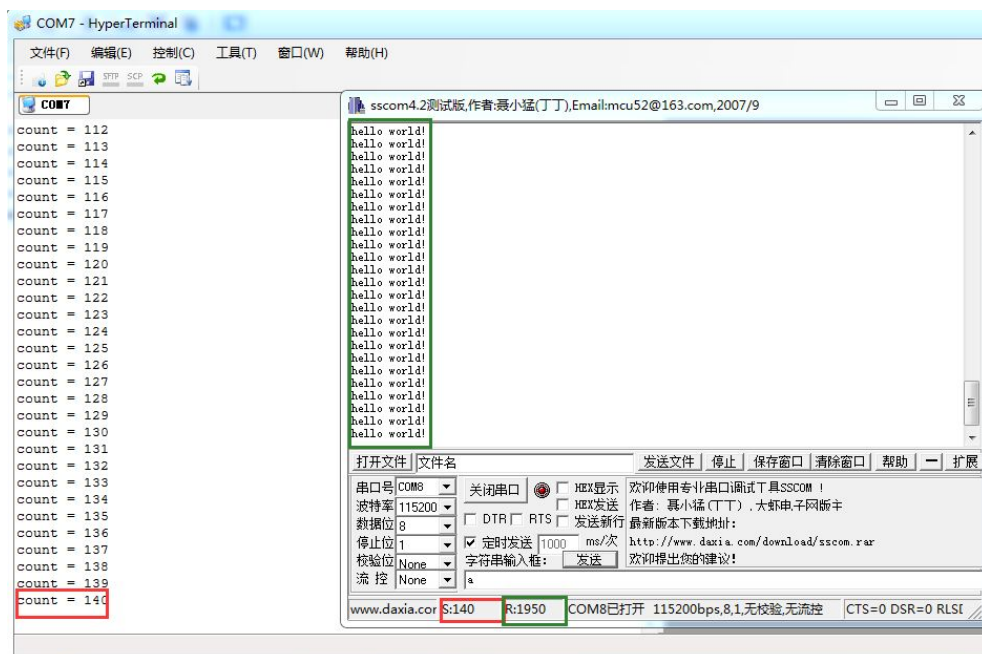


在超级终端使用命令 “./uart_write_read /dev/ttySAC3” 运行可执行文件。

```
root@android:/data # ./uart_write_read /dev/ttySAC3

uart_write_read_test start
open /dev/ttySAC3 is success
.....
```

串口助手选择定时发送数据，如下图，其中红色框是开发板接收到的数据，绿色框中的数据是开发板发送出去的数据。



8.6 全能版 485 的测试

本节测试 RS485 使用了两块全能版，用户可以根据实际情况修改测试的例子。

如下图，编译 485 测试程序，在 Ubuntu 系统中，输入编译命令 “arm-none-linux-gnueabi-gcc-4.4.1 -o test_485 test_485.c -static”，生成 test_485 可执行程序

```
root@ubuntu:/home/topeet/Android-C#  
root@ubuntu:/home/topeet/Android-C# arm-none-linux-gnueabi-gcc-4.4.1 -o test_485  
test_485.c -static  
root@ubuntu:/home/topeet/Android-C# ls  
ADC buzzer helloworld leds test_485 Uart_ttySAC3  
ADC.c buzzer.c helloworld.c leds.c test_485.c Uart_ttySAC3.c  
root@ubuntu:/home/topeet/Android-C#
```

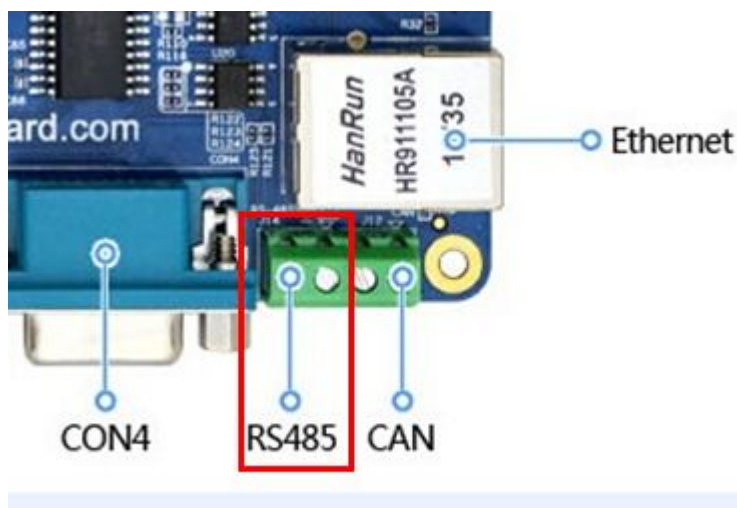
如下图，上传文件到开发板的 “/data”，在 cmd 命令行中，输入 adb 传文件的命令 “adb push test_485 /data”。

```
C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>adb push test_485 /d  
ata  
adb server is out of date. killing...  
* daemon started successfully *  
2705 KB/s (656528 bytes in 0.237s)  
C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>
```

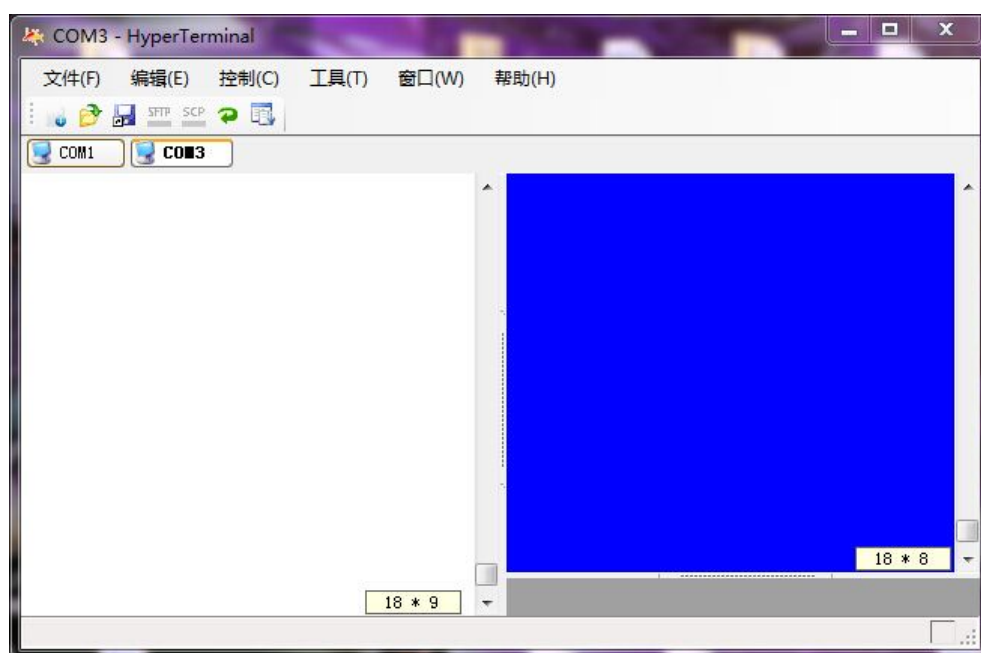
如下图，修改测试程序的权限。在超级终端中，输入命令 “cd /data”，输入修改权限命令 “chmod 777 test_485”。

```
root@android:/ #  
root@android:/ # cd /data  
root@android:/data # chmod 777 test_485  
root@android:/data #
```

这里的测试方法使用两块全能版，两块开发板的测试程序都是一样的。如下图，在网口旁的座子就是 RS485。485 有两根线，连接方式是 A 接 A，B 接 B，不用交叉。



PC 机器上，如下图，两个串口分别连接开发板的调试串口 CON2。



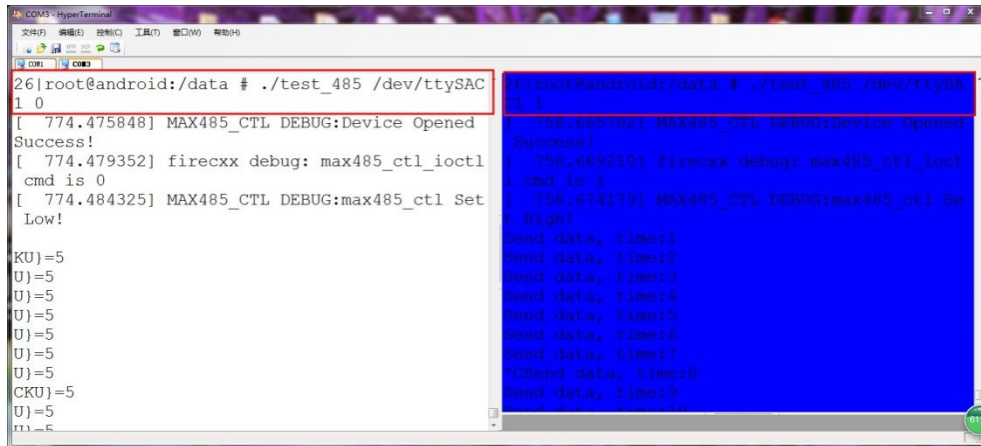
如下图，开发板启动后，在超级终端中，输入运行命令 “./test_485”。

```
26|root@android:/data # ./test_485

Usage: test_485 [uart port] [type]
              type: 0--recv, 1--send
```

如上图，可以看到，test_485 程序需要输入参数，第一个参数是选择测试的串口，第二个是设置该口是发送数据还是接收数据。全能版连接控制 485 的串口是 ttySAC1，所以第一

个参数为 ttySAC1。运行时命令分别为 “./test_485 /dev/ttySAC1 0” 和 “./test_485 dev/ttySAC1 0”。运行后，超级终端打印信息，如下图所示，表明发送和接收都成功了。



8.7 Camera200w 以及 500w 测试

本小节测试的摄像头默认由迅为电子提供的 OV5640，其它型号可参考。

摄像头测试代码和工具都在网盘 “iTOP4412 开发板资料汇总（不含光盘内容）\iTOP-4412 开发板系统源码及镜像（其他）\小模块的测试程序\camera 测试程序” 目录下。

8.7.1 摄像头 OV5640 编译和运行

注意测试前需要先将摄像头模块查到对应接口。

编译 camera 测试程序，在 Ubuntu 系统中修改 “build.sh” 权限，输入命令

“./build.sh”，生成 camera 可执行程序，如下图。

```
root@ubuntu:/home/topeet/camera# chmod 777 build.sh
root@ubuntu:/home/topeet/camera# ./build.sh
root@ubuntu:/home/topeet/camera# ls
build.sh camera camera.cpp camera.h main.cpp
root@ubuntu:/home/topeet/camera#
```

通过命令 “adb push camera /data”，上传测试程序 camera 到开发板，如下图。

```
C:\Users\Administrator\Desktop\USB_fastboot_tool>
C:\Users\Administrator\Desktop\USB_fastboot_tool>adb push camera /data
2640 KB/s (732708 bytes in 0.271s)
C:\Users\Administrator\Desktop\USB_fastboot_tool>
```

在超级终端先使用命令“cd data”进入上传目录 data 中，再使用命令“chmod 777 camera”修改测试程序 camera 权限，如下图。

```

root@android:/ #
10|root@android:/ #
10|root@android:/ # cd data
root@android:/data # chmod 777 camera
root@android:/data #

```

在超级终端中输入运行命令“./camera /dev/video0 640x480”，摄像头启动，打印信息部分如下图。

```

[root@iTOP-4412]#
[root@iTOP-4412]# ./camera /dev/video0 640x480
[ 28.443920] s3c-fimc0: FIMC0 1 opened.
open success
***** init_device, line = 209
***** init_device, line = 220
[ 28.459558] ov5640_probe()->5194 ov5640 probe start...
[ 28.459581] ov5640_probe()->5248 error: missing soc camera link
[ 28.470479] cym: ov5640 sensor is power on
[ 28.588030] ov5640_init: version = 0x5640
[ 31.092810] [OV5640_FOCUS_AD5820_Init]Profile = 617826265
[ 31.103882] check[0]=0x0
[ 31.104940] check[1]=0xcf
[ 31.107559] check[2]=0x0
[ 31.110139] check[3]=0x8
[ 31.112578] check[4]=0xf7
[ 31.115197] check[5]=0x0
.
.
.

fun:main, line = 88
frame:48,writesize:460800
***** read_frame, line = 80
***** read_frame, line = 82
fun:main, line = 88
frame:49,writesize:460800
fun:main, line = 94
time :0.780000, rate :64.102564
[ 34.794250] ov5640_s_stream()->3572 ::stream down
[ 34.794274] ov5640_remove()->5312 ::ov5640 device removed!
[ 34.810681] cym: ov5640 sensor is power off
[ 34.835659] s3c-fimc0: fimc_runtime_suspend_cap: No capture device.
[ 34.840661] s3c-fimc0: FIMC0 0 released.

```

程序运行完毕，在当前目录生成“out.yuv”文件，该文件为生成的图片文件，如下图。


```
root@android:/data # ls
app
app-private
backup
bluetooth
camera
dalvik-cache
data
dontpanic
drm
local
lost+found
misc
out.yuv
property
resource-cache
system
user
wifi
```

使用命令 “adb pull /data/out.yuv” 将 “out.yuv” 文件上传到 PC，如下图。

```
C:\Users\Administrator\Desktop\USB_fastboot_tool>adb pull /data/out.yuv
7595 KB/s (23040000 bytes in 2.962s)
C:\Users\Administrator\Desktop\USB_fastboot_tool>
C:\Users\Administrator\Desktop\USB_fastboot_tool>
```

8.7.2 检测方法

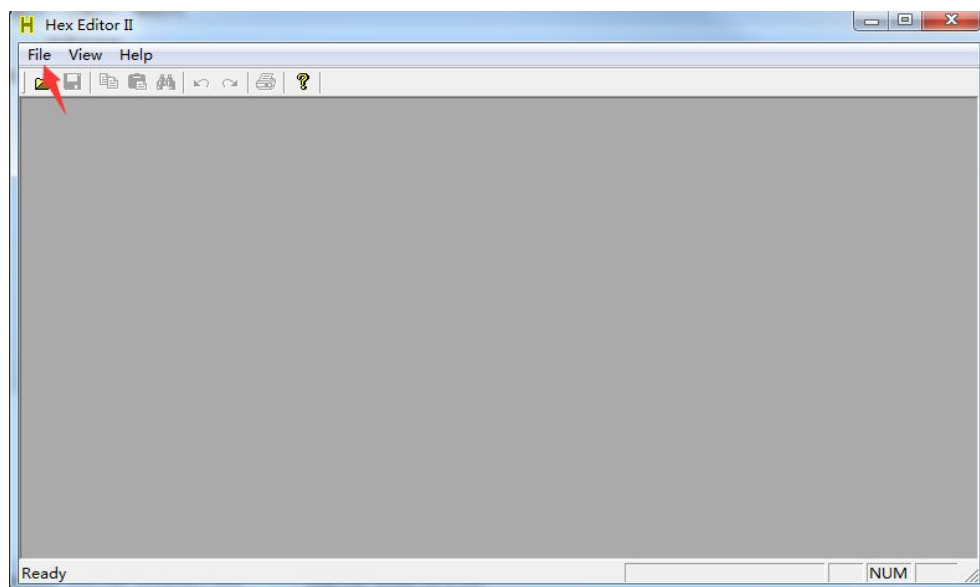
8.7.2.1 二进制文件查看器

可以通过二进制编码器查看该文件是否写入了数据，解压二进制编码器

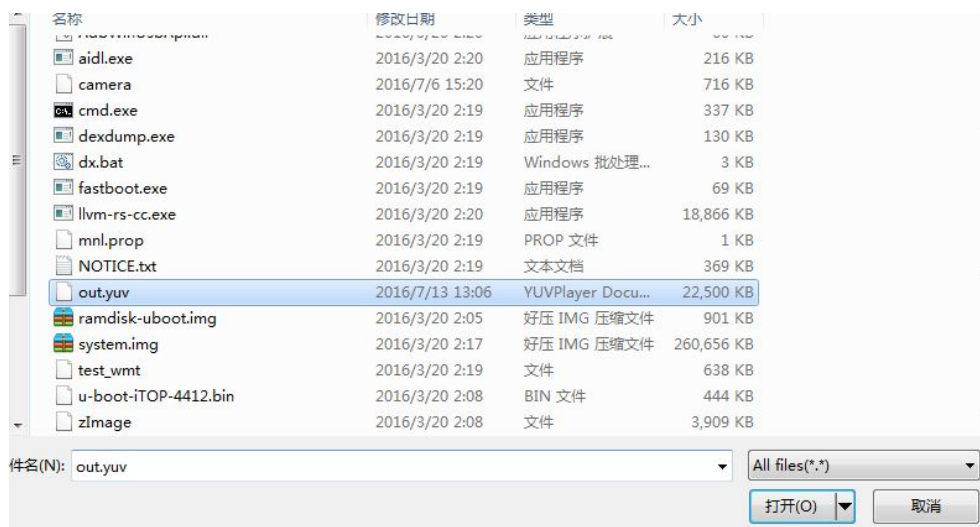
“Hex_Editor_II_2.1.0.zip”，如下图。

bconv.dat	2001/12/17 5:20	DAT 文件	58 KB
hcalc.dat	2001/12/17 5:20	DAT 文件	60 KB
Hex_Editor_II_2.1.0.rar	2016/7/13 17:36	好压 RAR 压缩文件	203 KB
HexEditor2.exe	2008/10/5 6:10	应用程序	216 KB

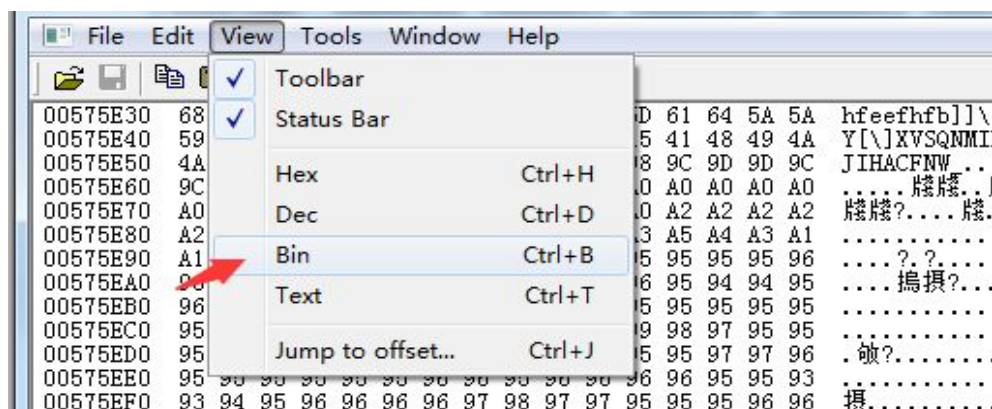
点击上图红色框中的 “HexEditor2.exe”，进入软件，界面如下图。



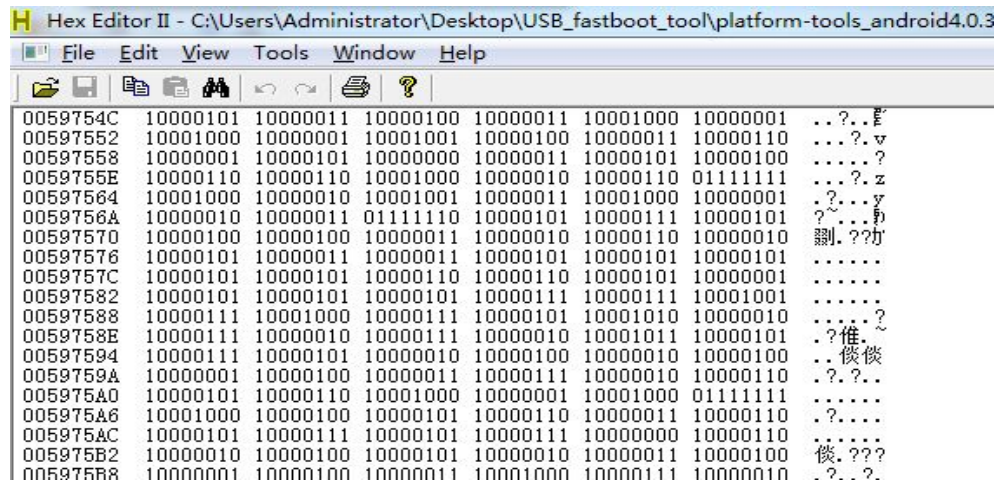
单机上图 File->Open 打开 “out.yuv” 如下图。



点击软件菜单栏 View -> Bin，选择 Bin 查看文件二进制信息，如下图。

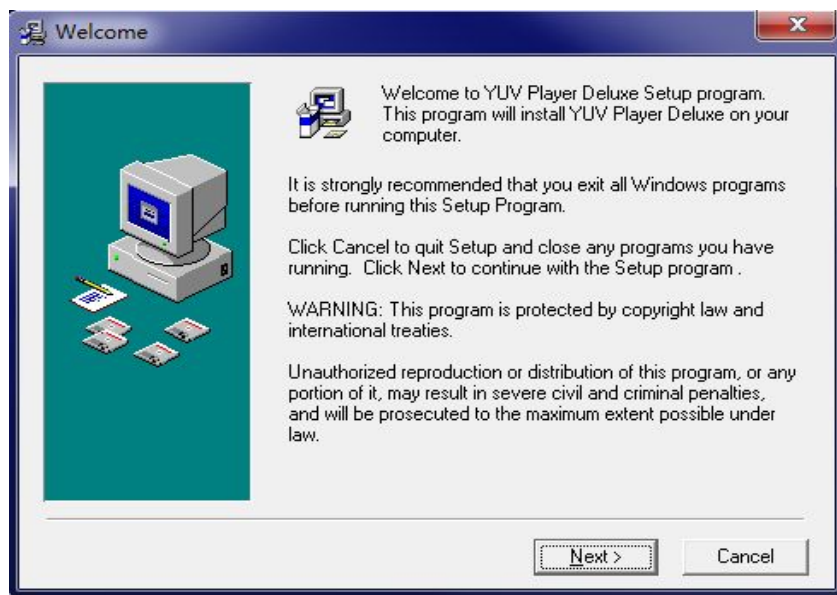


“out.yuv” 的部分二进制如下图，说明该文件写入了数据。



8.7.2.2 播放器 YUVPlayer

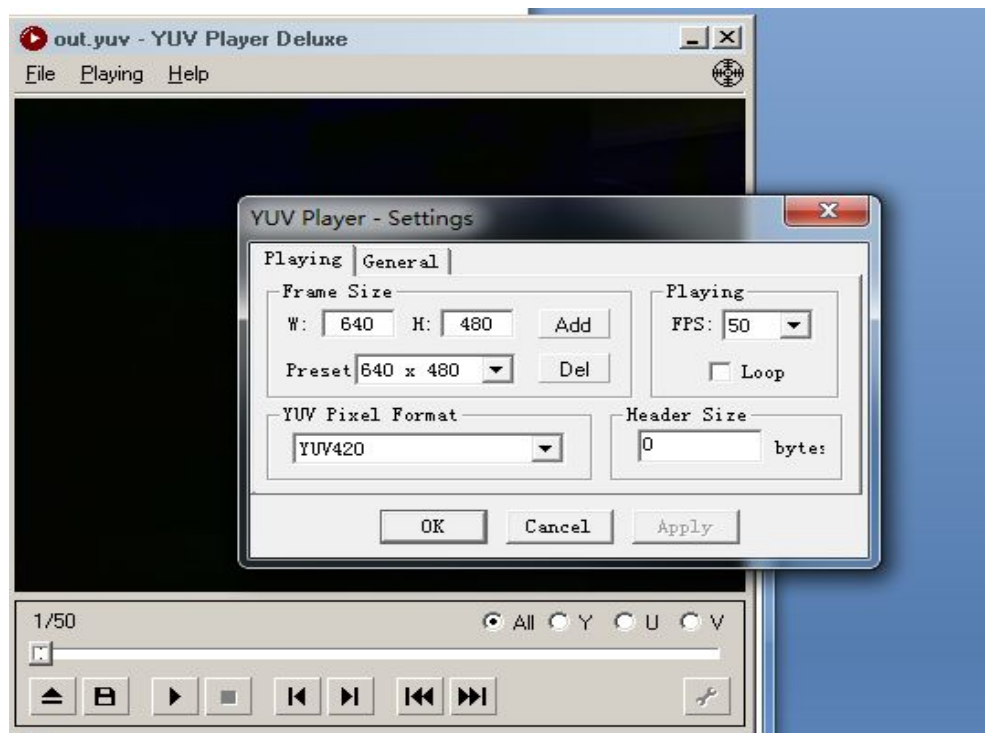
可以通过直接打开 “.yuv” 格式文件，显示图片内容来确定测试是否成功，解压 “YUV 视频流文件播放-查看器.zip”，得到安装包，点击 “YUVPlayer.exe” 开始安装，如下图。



安装完成之后，打开“YUVPlayer.exe”，如下图。



点击上图 File->Open 找到开发板上传到 PC 的 “out.yuv” 文件并打开，设置参数 640x480 如下图。



点击 OK，OV5640 拍摄的图片如下图。



8.8 扩展文档

8.8.1 智能网关

文档全称是“iTOP-4412-实战教程-智能网关_V1.0”，文档详细介绍了智能网关的硬件和实现方法。

8.8.2 串口文件传输工具的移植

文档全称是“iTOP-4412-实战教程-串口工具 lrzsz 移植_V1.0”，文档介绍了串口工具 lrzsz 的移植，以及如何使用 lrzsz 通过串口控制台来传小的测试例程。

8.8.3 GPS 模块测试文档

文档全称是“iTOP-4412-MiniLinux-GPS 使用文档_V1.0”，文档介绍了如何使用 c 程序测试 GPS。

8.8.4 SSH 的移植

文档全称是“iTOP-4412-实战教程-ssh 服务器移植到 arm 开发板_V1.0”，文档介绍了如何将 SSH 功能移植到 4412 开发板上。

8.8.5 Linux-C 程序调用 shell 脚本

文档全称是“iTOP-开发板-linux-C 调用 shell 命令使用文档”，文档介绍了 C 语言程序如何调用 shell 脚本。

8.8.6 GPS 模块的数据格式介绍

文档全称“iTOP-实战-GPS 模块的数据格式_V1.0”，文档介绍了 GPS 模块的数据格式。

8.8.7 modbus 移植和使用

文档全称“iTOP-开发板-modbus 移植和使用文档_V1.0”，文档介绍了 modbus 移植到 linux-arm 平台的方法和测试例程的使用。

8.8.8 python 移植到 linux-arm

文档全称“iTOP-4412-linux 系统-python 移植教程_V1.0”，文档介绍了如何将 python 移植到 linux-arm 平台，以及几个简单例程。