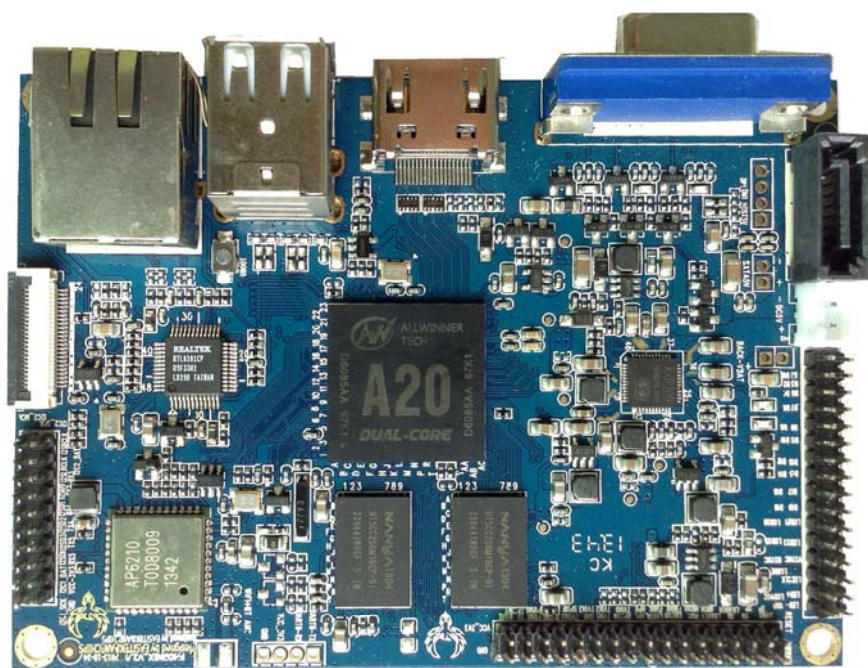


PhoenixA20 教程



目录

一、	Phoenix 开发环境搭建.....	1
1	硬件环境搭建.....	1
2	软件环境搭建.....	3
3	Phoenix源码下载.....	5
二、	Phoenix代码编译和固件烧写.....	7
1.	代码结构介绍.....	7
2.	代码编译.....	8
3.	固件烧写.....	9
三、	Phoenix调试.....	13
1.	串口调试.....	13
2.	USB调试.....	16
3.	JTAG调试.....	16
四、	Phoenix软件配置说明.....	17
1.	以太网功能的使用.....	17
2.	SATA硬盘.....	18
3.	卡使用.....	19
4.	WIFI和BT.....	19
5.	VGA和CVBS.....	23
6.	USB控制.....	24
7.	串口配置.....	24
8.	IIC配置.....	26
9.	红外配置.....	26

一、 Phoenix 开发环境搭建

1 硬件环境搭建

用户在使用 Phoenix 主板进行开发之前，先要准备好开发所需的硬件环境，只有这些硬件环境准备好后，才能对 Phoenix 进行开发。

1.1 基础硬件环境

基础硬件环境可以确保你能对 Phoenix 主板进行开发，并顺利使用板子上的多数接口。

- 1) 电脑主机（可以是笔记本或者是台式机），装有 64 位的 ubuntu，请保证内存在 4G 以上，硬盘有 100G 的剩余空间，这个主要是用于完整编译 android 系统，如果只是编译 linux，则不需要这么高的配置。编译 android 建议使用性能比较好的电脑，这样编译的时间不会太长。参考配置：四核 I5+8G DDR+500G 硬盘。



- 2) 带 VGA 或者 HDMI 接口的显示器



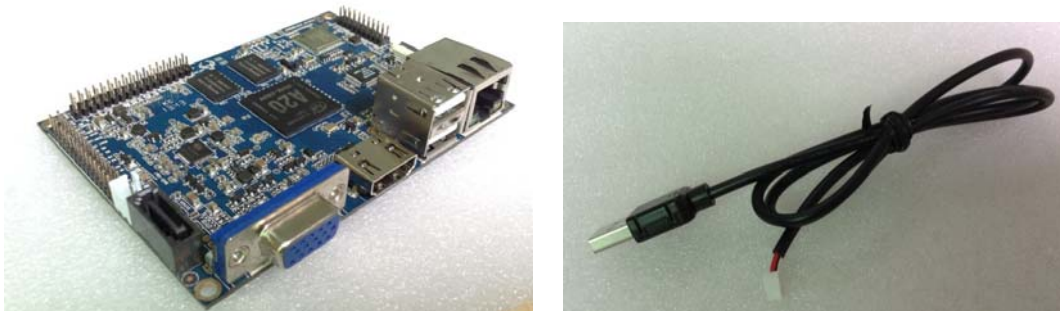
3) HDMI 或者 VGA 数据线



4) USB 线（标配，不需额外购买）和 USB 转串口线



5) Phoenix 主板和电源线（标配）



以上就是开发所必需的基础硬件环境，配备好这些后，Phoenix 开发所需要的硬件环境就基本完整了，可以使用 Phoenix 进行 android 和 linux 的开发。

1.2 可选硬件环境

可选的硬件环境可以让你使用 Phoenix 更多功能，并对 Phoenix 进行更广义的开发。但是这些硬件并不是必需的，如果你不需要 Phoenix 的某些功能，那你可以不需要配这些硬件设备。

1) 带数据线和电源线的 2.5 寸硬盘。Phoenix 集成了硬盘接口，用户可以直接使用 2.5 英寸

的硬盘，硬盘也可以从主板直接供电。



- 2) 蓝牙音箱和无线鼠标，蓝牙音箱可以用来验证蓝牙功能，当然你也可以使用其他蓝牙设备，鼠标则可以用来做用户交互。



- 3) TF 卡和 U 盘
- 4) 其他配件。Phoenix 后面会为用户提供多种扩展子板，以方便用户验证 Phoenix 的更多的功能。

2 软件环境搭建

2.1 操作系统安装

安装 ubuntu 操作系统，详细安装过程请 Google 相应教程，同时你需要安装 windows 操作系统，windows 操作系统主要是用来安装固件升级工具，目前 A20 并没有提供 linux 的固件升级工具。你可以使用虚拟机，这样可以同时运行 ubuntu 和 windows 操作系统，如果你有两个电脑，那你需要至少有一台电脑安装 ubuntu。Ubuntu 最好选择 12.04 以上版本，而且必须是 64 位的操作系统，这个是编译 android 的基本条件。

2.2 Linux 工具链的安装

- 1) 安装 JDK

```
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java6-installer
```

2) 安装平台支持软件集

```
$ sudo apt-get install libgl1-mesa-dri:i386
$ sudo apt-get install git gnupg flex bison gperf build-essential \
zip curl libc6-dev libncurses5-dev:i386 x11proto-core-dev \
libx11-dev:i386 libreadline6-dev:i386 libgl1-mesa-glx:i386 \
libgl1-mesa-dev g++-multilib mingw32 tofrodos \
python-markdown libxml2-utils xsltproc zlib1g-dev:i386

$ sudo ln -s /usr/lib/i386-linux-gnu/mesa/libGL.so.1 /usr/lib/i386-linux-gnu/libGL.so
```

3) 安装交叉编译工具链

编译工具链已经集成在 Android SDK 中，工具链位于 Android SDK 中的
lichee/boot/config/gcc-linaro/

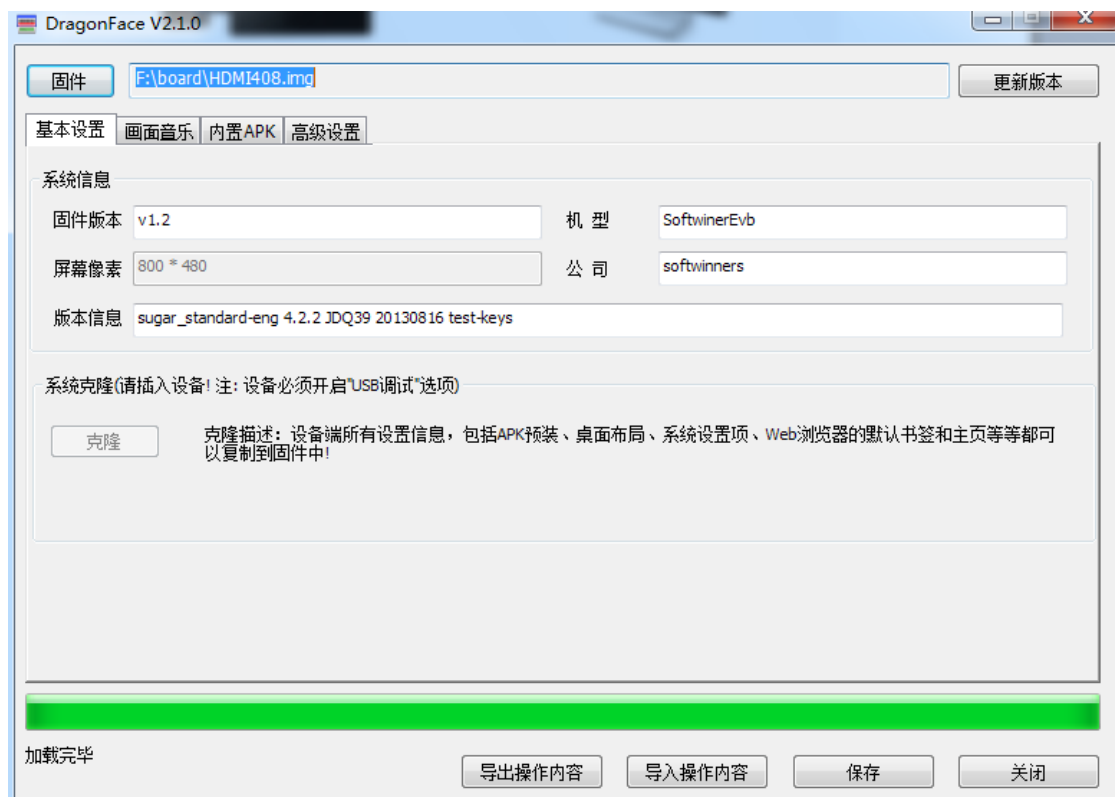
2.3 windows 工具安装

Windows 主要用来安装固件升级工具 **phoenixsuit**，该工具安装非常方便，下载安装包，直接点击安装即可，安装过程不再详述。安装后打开软件后的界面如下：



安装完固件升级工具后，还可以再安装一些常用的工具，比如串口调试工具，可以从网上下载免费的串口工具 **putty** 等。你还可以安装一些文本编辑工具。

如果用户不想自己编译固件，可以安装固件修改工具 **dragonface** 对固件进行一些简单的修改，从而满足需求一些特殊的需求。**Dragonface** 有绿色版，不需要安装，下载下来后可以直接使用，界面如下：



3 Phoenix 源码下载

Phoenix 源代码都是放在 github 中托管，主要是方便不同地区的人都可以访问并下载到代码。源代码分为两部分，android4.2 源码和 linux 源码，如果你只是进行 linux 开发，则没有必要下载 android 代码。下面详细说明如何下载代码：

3.1 linux 源代码下载

用户如果是第一次使用 repo 来下载代码，则在下载代码之前需要安装 repo 工具，具体操作如下：

```
$ mkdir ~/bin
$ PATH=~/bin:$PATH
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
```

安装完成后，就可以使用 repo 来下载 PhoenixA20 的代码了。用户可以通过下面的命令下载 linux 的相关代码：

```
$ mkdir lichee
$ cd lichee
```

```
$ repo init -u https://github.com/qubir/phoenixA20_linux_repo.git  
$ repo sync
```

至此，你只需要耐心等待几十分钟，这个取决于你的网络环境。下载完成后，执行下面的操作：

```
$ mv buildroot/build_linux.sh build.sh
```

3.2 android 源代码下载

(未完待续)

以上两个代码都是放在 PhoenixA20 私有的帐号上，熟悉 A20 开发的用户可以从其他渠道获取代码，linux-sunxi 是一个不错的选择。

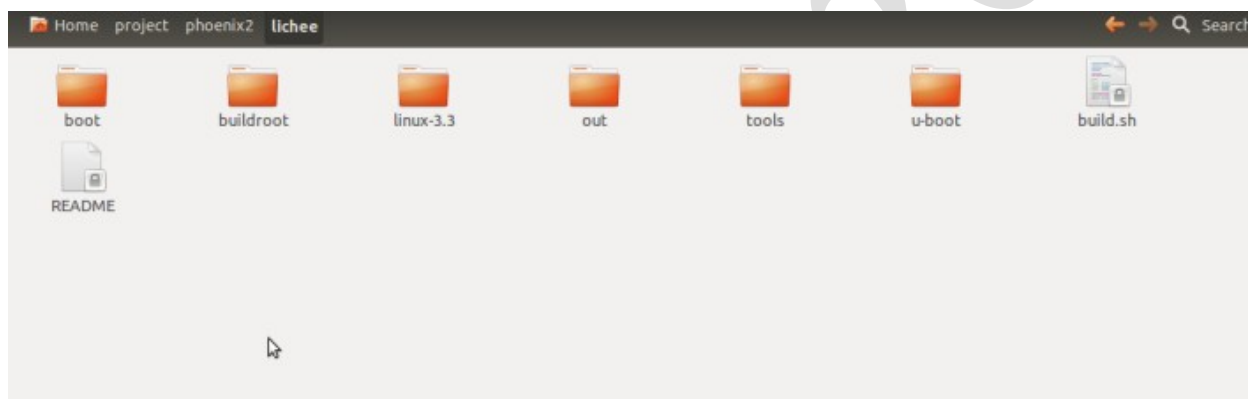
至此，Phoenix 的开发环境搭建完毕。

二、 Phoenix代码编译和固件烧写

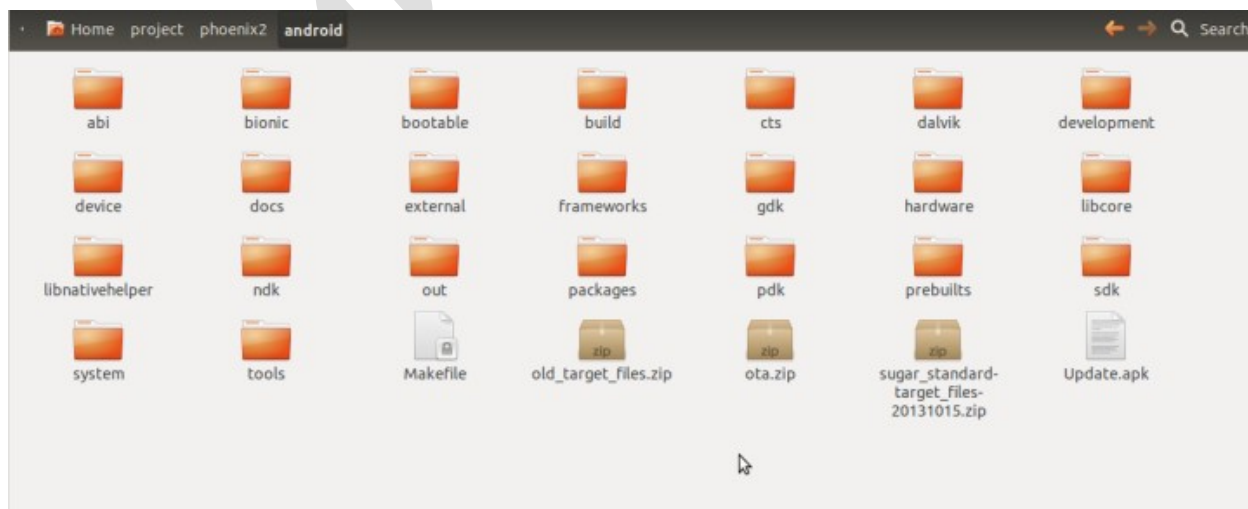
1. 代码结构介绍

Phoenix 代码下载完后，将在同一个目录中呈现出两个目录，一个是 android 目录，一个是 lichee 目录。其中 android 目录存放 android 的所有源代码，lichee 目录则存放 linux 源码以及一些工具链等。必须注意，android 和 lichee 必须放在同一个目录下，比如都放在 phoenix 目录，这个主要是编译 android 的时候需要从 lichee 中拷贝一些文件，而相关的脚本已经把拷贝文件的路径写成固定的，所以如果不放在一起的话，将导致无法从 lichee 的输出目录拷贝生成的内核文件和设备驱动，导致编译出来的 android 无法使用。

1.1 lichee 代码结构介绍



1.2 android 代码结构介绍



2. 代码编译

2.1 编译 android 代码

如果要想编译生成自定义的 android 固件，你需要同时编译 linux 和 android 的源代码，下面将详细描述如何编译这些代码，并生成固件。

1) 编译 lichee 代码

使当前目录为 lichee 的根目录。然后执行下面的命令：

```
cd lichee
./build.sh -p sun7i_android
```

或者：

```
. buildroot/scripts/mksetup.sh    #导入环境变量，根据提示选择对应选项
mklichee
```

即完成了一次 lichee 的编译（根据服务器配置，耗时至少 10 分钟），编译成功，屏幕上出现

```
INFO:build u-boot OK.
...
INFO:build rootfs OK.
INFO:build lichee OK.
```

2) 编译 android 代码

编译完 lichee 后，将目录切换到 android 目录，然后进行下面的操作：

```
. build/envsetup.sh    #导入环境变量
lunch                  #根据自己的开发平台，选择方案
extract-bsp            #拷贝内核和模块到 android 中
make -j4               #-j 开启多核编译，一般为 cpu 数量的一半
```

编译完成之后，将会在相应的产品目录（out/target/product/wing-xxx）会生成：boot.img, recovery.img . system.img 3 个包。

3) 固件打包

至此 android 的编译就完成了，此时你可以直接调用命令

```
$ pack
```

将会在 lichee/tools/pack 目录下生成 sun7i_android_xxx.img 文件，这就是我们要的 android 固件，可以通过烧写工具直接烧写到 Phoenix 中。

2.2 编译 linux 代码

如果只需要编译 linux 固件，可以使用下面的命令：

```
cd lichee
```

```
./build.sh -p sun7i
```

```
./build.sh pack
```

根据提示选择相应的参数，即可编译打包固件。

如果某些选项没有选上，可能会导致编译出错，可输入以下命令清除，重新编译

```
./build.sh -m clean
```

```
./build.sh -p sun7i
```

2.3 编译常见问题

1) can't find lz

编译时出现以下打印信息：

```
/usr/bin/ld: cannot find -lz
```

```
collect2: ld returned 1 exit status
```

```
make: *** [out/host/linux-x86/obj/EXECUTABLES/aapt_intermediates/aapt] Error 1
```

原因是缺少 lib32z1-dev，安装即可:sudo apt-get install lib32z1-dev

2) You must install 'makeinfo' on your build machine

解决方法: sudo apt-get install texinfo

3) msgfmt not found

解决方法: sudo apt-get install gettext

2.4 局部编译和打包

不用每次编译都对系统进行完全编译，你可以只编译有修改的部分。boot.img 镜像中包含 linux kernel 和内存盘 ramdisk，如果内核有修改，就需要重新编译内核，然后在 android 目录下执行下列命令,即可打包生成 boot.img

```
$ . build/envsetup.sh
```

```
$ lunch #选择 wing-xxx 产品
```

```
$ extract-bsp
```

```
$ make bootimage
```

这样就生成了 boot.img，类似的方法就可以重新打包生成 system.img

```
$ . build/envsetup.sh
```

```
$ lunch #选择 wing-xxx 产品
```

```
$ make systemimage-nodex
```

wing-xxx 重新生成的镜像在 android/out/target/product/wing-xxxwing-xxx/ 目录下。

3. 固件烧写

至此，Phoenix 需要的固件已经编译生成完毕，下面就是要将固件下载到主板中，下载的方式有两种。一种是完全下载的方式，另外一个局部下载方式，下面分别说明两种下载方式。

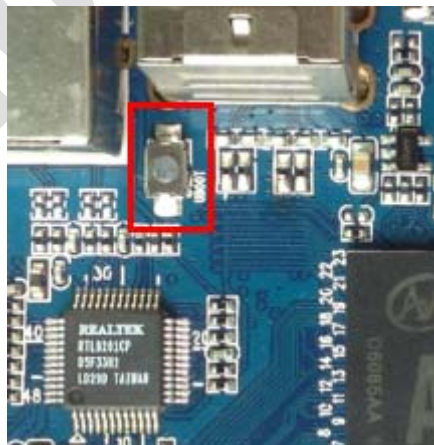
3.1 phoenixsuit 下载

在开发环境一章中，我们已经安装了 phoenixsuit 固件下载工具，可以直接用这个工具来下载工具。具体操作如下：

- 1) 打开 phoenixsuit，选择需要下载的固件。



- 2) 将 USB 线连接到电脑。
- 3) Phoenix 主板断电。
- 4) 按住Phoenix主板中的UBOOT键后,将USB线的另外一端插入Phoenix的升级USB 口。



UBOOT 按键位置

- 5) 插入后主板会自动进入升级模式，并提示是否需要格式化。



- 6) 根据需求选择是否格式化后, 系统将进入升级状态, 这个过程将持续 2 分钟左右。升级完后, 工具将会提示升级成功。



- 7) 升级成功后，系统会自动启动，这个时候是 USB 供电，建议使用 5V 电源供电，如果需要用作移动设备，可以使用移动电源对其进行供电。

3.2 fastboot 方式烧写固件

每次使用 PhoenixSuit 升级时间会比较长，如果用户只是修改了某个 img 文件，而其他分区并没有修改，可以使用 fastboot 的方式进行固件烧写。

fastboot 是一种线刷，就是使用 USB 数据线连接手机的一种刷机模式，在 A20 主控中，可以使用 fastboot 的功能来实现局部系统的更新。

1) 进入 fastboot 模式

启动开发板，在串口界面敲任意按键，可以进入 u-boot；如果进不了 fastboot，将 lichee/tools\pack\chips\sun7i\configs\android\default\env.cfg 中的 bootdelay=0 改成 bootdelay=2 重新打包固件即可（需要安装 google-usb_driver 驱动）。

在串口命令行输入 fastboot 命令，进入 fastboot 模式；

通过 pc 端的 fastboot 工具烧录各个固件包（fastboot 是 windows 下的一个工具（android sdk 中有），上网自己下载一个，解压到本地，然后将 fastboot.exe 添加到 windows 环境变量）

进入在 windows 命令行：cmd 进行命令行模式，在命令行执行 fastboot 指令\

退出 fastboot 模式：ctl+c

2) 使用 fastboot 命令烧写

在 windows 命令行使用 fastboot 命令。

擦除分区命令：

```
$ fastboot erase boot #擦除 boot 分区
$ fastboot erase system #擦除 system 分区
$ fastboot erase data #擦除 data 分区
```

烧写分区命令：

```
$ fastboot flash boot boot.img #把 boot.img 烧写到 boot 分区
$ fastboot flash system system.img #把 system.img 烧写到 system 分区
$ fastboot flash data userdata.img #把 userdata.img 烧写到 data 分区
```

Fastboot 方式更新固件效率比较高，建议使用。

三、 Phoenix调试

1. 串口调试

1.1 串口调试基础环境

Phoenix 直接集成了串口调试口，所以可以直接用串口来进行软件的调试，如果要用串口的方式来调试，首先应该具备下面两个东西：

1) USB 转串口线



一般从网上买的 USB 转串口线，其不同颜色对应的引脚分别为：

红色	电源线
黑色	地线
绿色	RX
白色	TX

这四根线跟板子上调试接口对应的引脚相对应。注意，红色的电源线一般不接。

2) 串口调试工具

串口调试工具比较多，不管是在 windows 还是在 linux 操作系统下，都有很多串口工具可以使用，windows 下常用的是 secureCRT，超级终端，putty 等等，linux 下可以使用 mini COM 等工具。推荐使用 secureCRT 或者 putty。下面演示是用 putty 工具。

1.2 串口配置

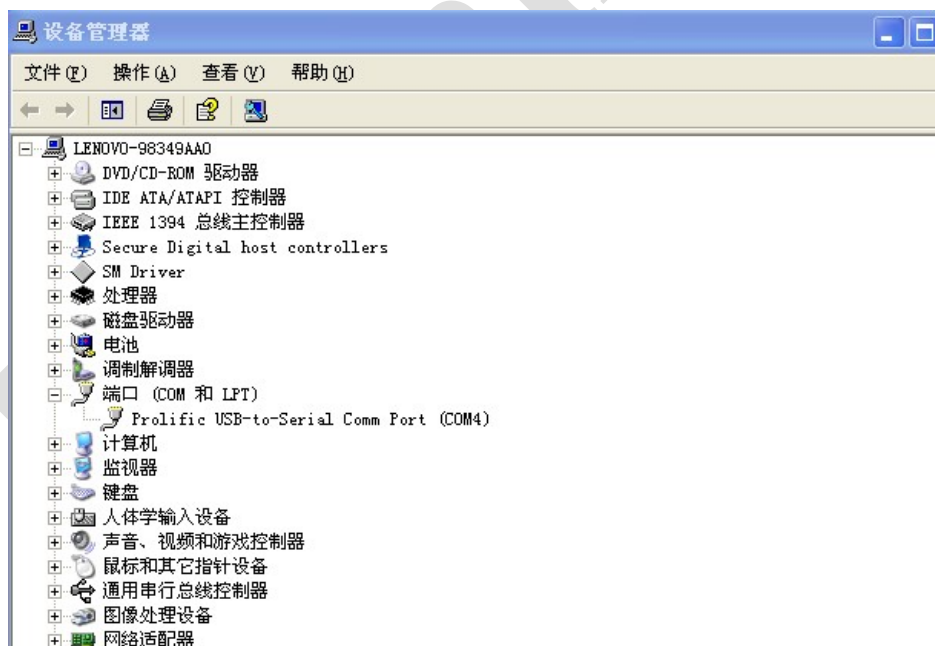
准备好调试线后和调试工具后就可以开始调试了，首先我们进入 window 设备管理器查

看 usb 转串口生成的端口号。

- 1) 右键点击我的电脑->属性，选择硬件

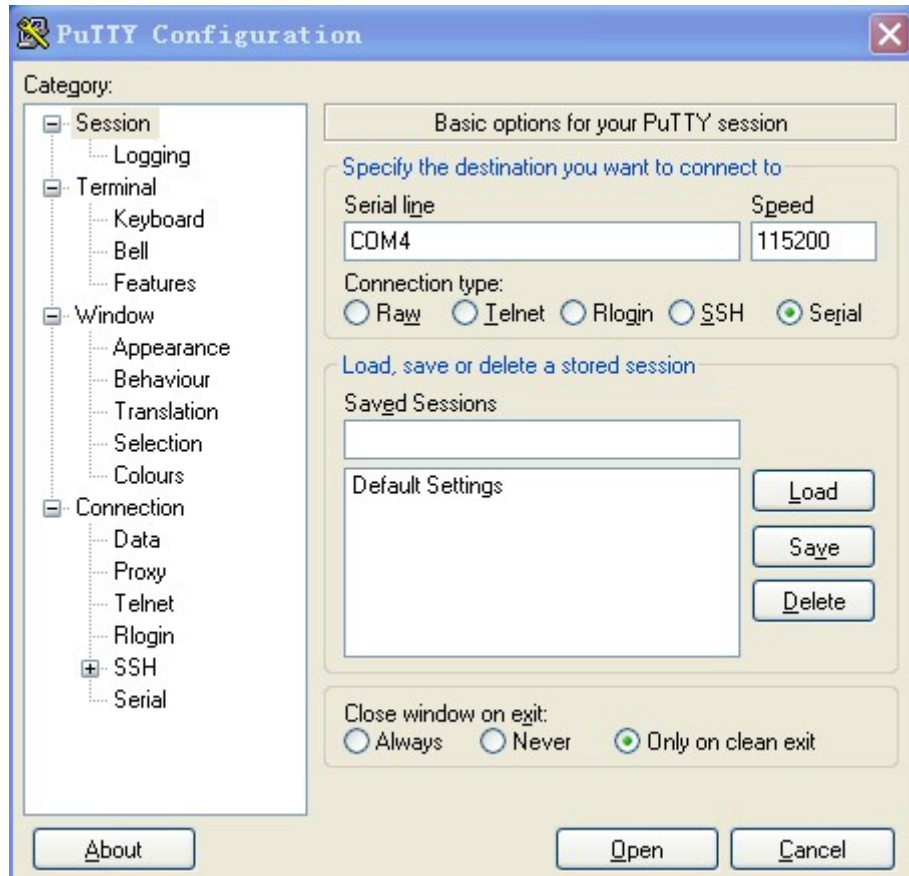


- 2) 选择设备管理器，点击进入，查看端口选项



从图中我们可以看出来，我们现在用的是 COM4.

- 3) 打开 putty，选择 connection type 为 serial，serial line 为 COM4，speed 为 115200

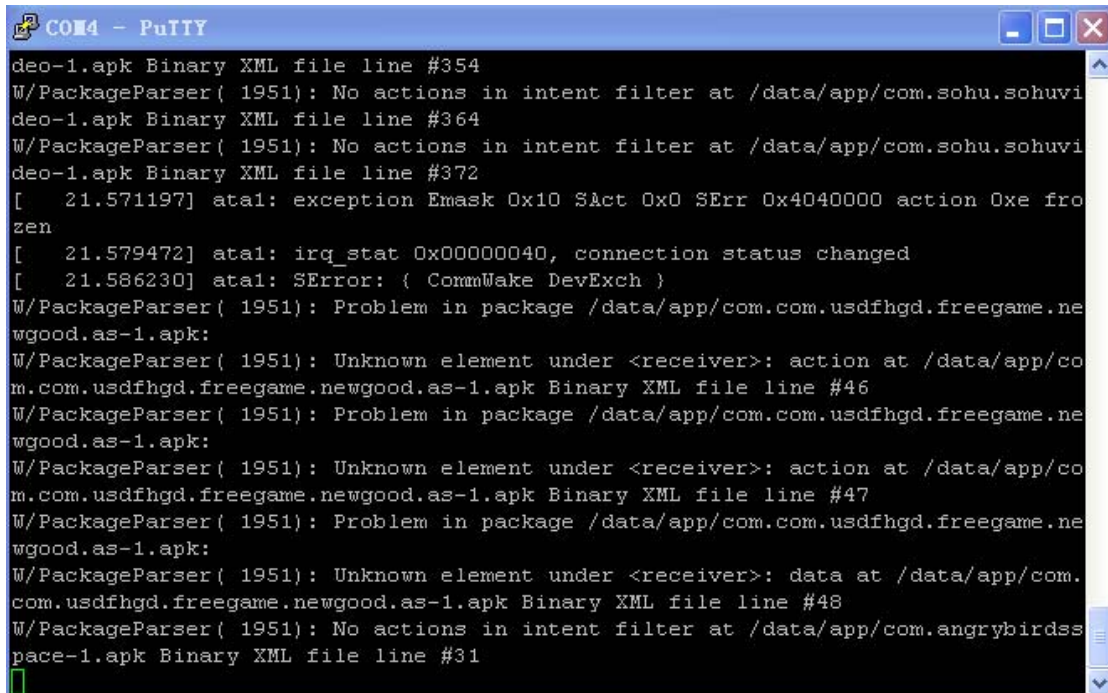


- 4) 点击 open，进入串口界面，Phoenix 上电，可以看到系统启动的打印信息

```

COM4 - PuTTY
Check is correct.
Ready to disable icache.
Succeed in loading Boot1.
Jump to Boot1.
[ 0.149] boot1 version : 2.0.0
[ 0.158] script installed early ok
[ 0.159] pmu type = 3
[ 0.160] bat vol = 0 mv
[ 0.176] axi:ahb:apb=4:2:2
[ 0.176] set dcdc2=1400mv, clock=912M succeeded
[ 0.178] key
[ 0.191] no key found
[ 0.191] flash init start
[ 0.191] NB1 : enter NFB_Init
[ 0.194] NB1 : enter phy init
[ 0.197] [NAND] nand driver(b) version: 0x00000002, 0x00000012, data:
0x20130325
[ 0.207] get the good blk ratio from hwscan : 912
[ 0.210] NB1 : nand phy init ok
[ 1.749] _RepairLogBlkTbl start
[ 1.751] [DUBG], check free block 0x000003f5 ok!
[ 2.253] Log Block Index 0x00000000, LogicBlockNum: 0x00000090, LogBlockTy
pe: 0x00000001
[ 2.256] log0: 0x000002c9, Log1: 0x000003f5, WriteIndex: 0x00000000
  
```

- 5) 输入 logcat，可以看到 android 系统的打印



```

COM4 - PuTTY
deo-1.apk Binary XML file line #354
W/PackageParser( 1951): No actions in intent filter at /data/app/com.sohu.sohuvi
deo-1.apk Binary XML file line #364
W/PackageParser( 1951): No actions in intent filter at /data/app/com.sohu.sohuvi
deo-1.apk Binary XML file line #372
[ 21.571197] atal: exception Emask 0x10 SAct 0x0 SErr 0x4040000 action 0xe fro
zen
[ 21.579472] atal: irq_stat 0x00000040, connection status changed
[ 21.586230] atal: SError: ( CommWake DevExch )
W/PackageParser( 1951): Problem in package /data/app/com.com.usdfhgd.freegame.ne
wgood.as-1.apk:
W/PackageParser( 1951): Unknown element under <receiver>: action at /data/app/co
m.com.usdfhgd.freegame.newgood.as-1.apk Binary XML file line #46
W/PackageParser( 1951): Problem in package /data/app/com.com.usdfhgd.freegame.ne
wgood.as-1.apk:
W/PackageParser( 1951): Unknown element under <receiver>: action at /data/app/co
m.com.usdfhgd.freegame.newgood.as-1.apk Binary XML file line #47
W/PackageParser( 1951): Problem in package /data/app/com.com.usdfhgd.freegame.ne
wgood.as-1.apk:
W/PackageParser( 1951): Unknown element under <receiver>: data at /data/app/com.
com.usdfhgd.freegame.newgood.as-1.apk Binary XML file line #48
W/PackageParser( 1951): No actions in intent filter at /data/app/com.angrybirdss
pace-1.apk Binary XML file line #31

```

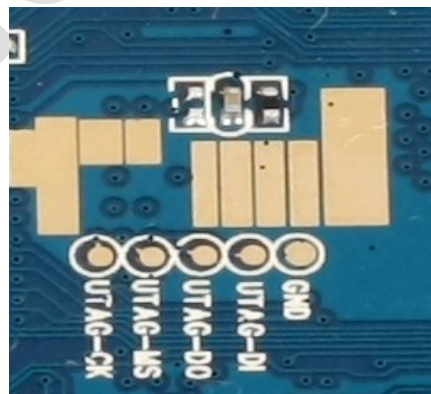
6) 在串口环境下，默认是在 shell 模式，你可以输入 shell 的一些命令进行相应的操作。

2. USB 调试

USB 调试一般用于 android 的 adb 调试，具体操作方式，可以在网上搜索相关文档。

3. JTAG 调试

Phoenix 预留了 JTAG 的测试点，有特殊需求的用户可以通过飞线的方式将 JTAG 调试口引出来进行调试。



JTAG 调试的具体方法，不再详述。

四、 Phoenix软件配置说明

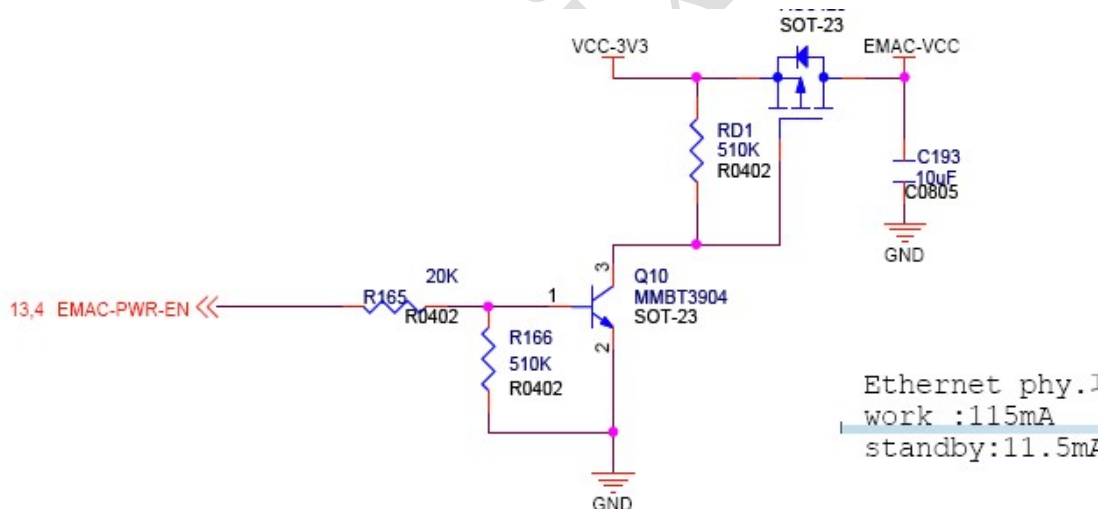
Phoenix 默认带有 android 系统，主要的外围模块的功能都已经打开，用户可以直接使用，考虑到用户会自己定制系统，所以下面将详细说明一下 Phoenix 各个模块的使用，以及软件配置如何使用。

Phoenix 外围模块的很多功能都可以通过修改配置文件来打开，配置文件为 /lichee/tools/pack/chips/sun7i/configs/android/sugar-standard/目录下的 sys_config.fex，另外一个则是 sys_partitions.fex 该文件主要描述系统分区的一些信息，将会在另外一个文档中详细说明。

Sys_config.fex 包含了 Phoenix 外围多数功能的配置，下面将详细说明如何修改该文件以实现 Phoenix 主板集成的硬件功能。

1. 以太网功能的使用

Android 系统默认集成了以太网驱动，因此只需要通过简单的配置，即可使用以太网，Phoenix 主板通过控制一个 GPIO 来控制以太网的开关。具体电路如下：



Sys_config.fex 跟以太网相关的配置信息如下：

```

;-----
;Ethernet MAC configuration
;-----
[emac_para]
emac_used          = 1
emac_rxd3          = port:PA00<2><default><default><default>
emac_rxd2          = port:PA01<2><default><default><default>
emac_rxd1          = port:PA02<2><default><default><default>
emac_rxd0          = port:PA03<2><default><default><default>

```

```

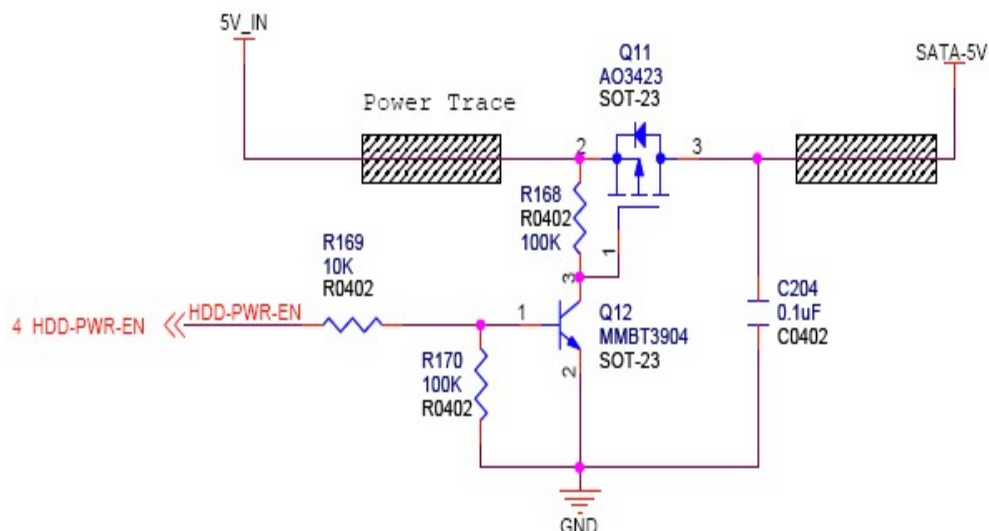
emac_txd3      = port:PA04<2><default><default><default>
emac_txd2      = port:PA05<2><default><default><default>
emac_txd1      = port:PA06<2><default><default><default>
emac_txd0      = port:PA07<2><default><default><default>
emac_rxclk     = port:PA08<2><default><default><default>
emac_rxerr     = port:PA09<2><default><default><default>
emac_rxdV      = port:PA10<2><default><default><default>
emac_mdc       = port:PA11<2><default><default><default>
emac_mdio      = port:PA12<2><default><default><default>
emac_txen      = port:PA13<2><default><default><default>
emac_txclk     = port:PA14<2><default><default><default>
emac_crs       = port:PA15<2><default><default><default>
emac_col       = port:PA16<2><default><default><default>
emac_reset     = port:PA17<1><default><default><default>
emac_power     = port:PH24<1><default><default><default>

```

如果系统需要使用以太网，则必须修改 `emac_use` 配置项为 1，同时 `emac_power` 引脚修改为 PH24（Phoenix 主板上 `emac_power` 对应的引脚是 PH24），修改完后，重新打包固件，即可使用以太网的功能。

2. SATA 硬盘

Phoenix 系统默认支持 sata 驱动，Linux 已经默认将 sata 编入到内核，如果没有，则需要将驱动编译到内核，具体操作方式可以在 `menuconfig` 中修改。Phoenix 通过 GIPO 控制 SATA 电源的开关。



`Sys_config.fex` 跟 sata 相关的配置如下：

```

;-----
;sata configuration
;

```

```

;-----
[sata_para]
sata_used          = 1
sata_power_en      =

```

如果使用主板直接供电，则需要修改 `sata_power_en`，修改方式如下：

```
sata_power_en      = port:PH17<1><default><default><default>
```

如果使用外部供电，则不需要修改，具体可以参考 `sata` 驱动代码。

3. 卡使用

Phoenix 支持 32GB 的 `tf` 卡扩展，跟卡相关的配置如下：

```

[mmc0_para]
sdc_used          = 1
sdc_detmode       = 1
sdc_buswidth      = 4
sdc_clk           = port:PF02<2><1><2><default>
sdc_cmd           = port:PF03<2><1><2><default>
sdc_d0            = port:PF01<2><1><2><default>
sdc_d1            = port:PF00<2><1><2><default>
sdc_d2            = port:PF05<2><1><2><default>
sdc_d3            = port:PF04<2><1><2><default>
sdc_det           = port:PD04<0><1><default><default>
sdc_use_wp        = 0
sdc_wp            =
sdc_isio          = 0
sdc_regulator     = "none"

```

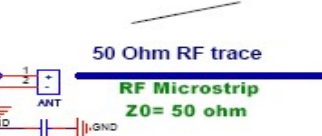
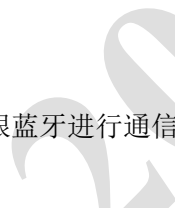
如果需要使能卡的功能，则需要将 `sdc_used` 设置为 1。并且修改 `sdc_det` 引脚。Phoenix 使用 `mmc0` 作为卡控制器，所以只需要修改 `mmc0` 的参数即可。主板上用 `PD04` 作为卡的检测引脚，所以需要将 `sdc_det` 修改为：

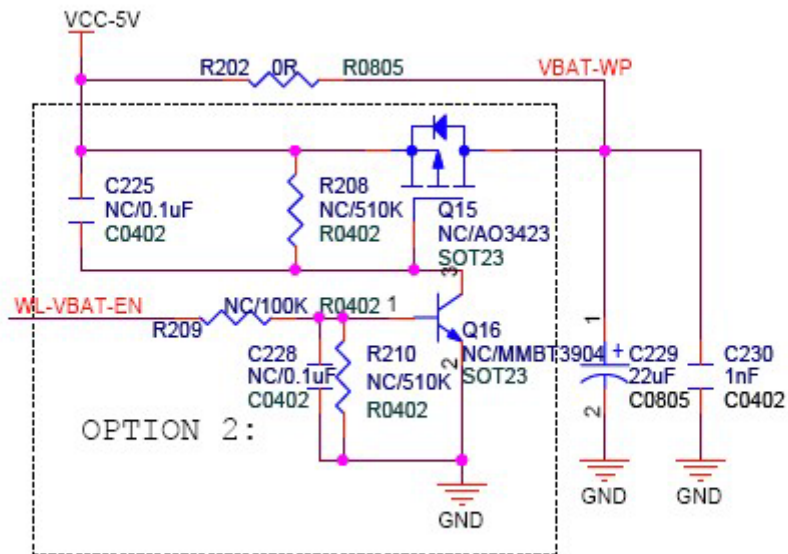
```
sdc_det           = port:PD04<0><1><default><default>
```

4. WIFI 和 BT

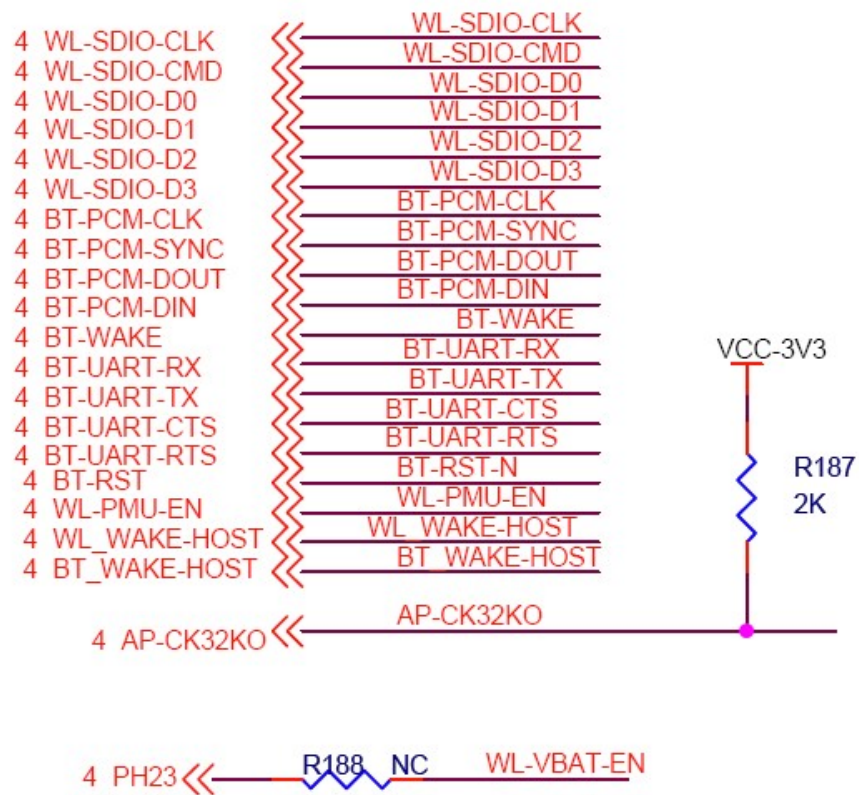
4.1 硬件介绍

Phoenix 采用了美国博通公司的 `wifi` 和蓝牙二合一芯片 `AP6210`，该芯片性能和稳定性远远超过当前其他的 `wifi` 和蓝牙芯片。`AP6210` 支持最新的蓝牙 4.0 协议，具有明显的低功耗优势。





引脚对应关系如下:



几个关键引脚对赢的 GPIO 为:

模组引脚	A20 对应引脚
BT-RST	PB05
BT-WAKE	PI20
WL-PMU-EN	PH09
WL-WAKE-HOST	PH10
BT_WAKE-HOST	PI21

4.2 软件配置

要使用 wifi 和蓝牙，首先要修改 sysconfig.fex 中相关的配置，具体如下：

WIFI 配置

```

;-----
;wifi configuration
;wifi_sdc_id    ---  0- SDC0, 1- SDC1, 2- SDC2, 3- SDC3
;wifi_usbc_id   ---  0- USB0, 1- USB1, 2- USB2
;wifi_usbc_type --  1- EHCI(speed 2.0), 2- OHCI(speed 1.0)
;wifi_mod_sel   ---  0- none, 1- bcm40181, 2- bcm40183(wifi+bt),
;                3 - rtl8723as(wifi+bt), 4- rtl8189es(SM89E00),
;                5 - rtl8192cu, 6 - rtl8188eu, 7 - ap6210
;-----
[wifi_para]
wifi_used       = 1
wifi_sdc_id     = 3
wifi_usbc_id    = 2
wifi_usbc_type  = 1
wifi_mod_sel    = 7
wifi_power      = ""

ap6xxx_wl_region = port:PH09<1><default><default><0>
ap6xxx_wl_host_wake = port:PH10<0><default><default><0>
ap6xxx_bt_region  = port:PH23<1><default><default><0>
ap6xxx_bt_wake    = port:PI20<1><default><default><0>
ap6xxx_bt_host_wake = port:PI21<0><default><default><0>

```

蓝牙配置

```

;-----
;blue tooth
;bt_used        ---- blue tooth used (0- no used, 1- used)
;bt_uart_id     ---- uart index
;-----
[bt_para]
bt_used         = 1
bt_uart_id     = 2
bt_wakeup       = port:PI20<1><default><default><default>

```

```
bt_gpio      = port:PI21<1><default><default><default>
bt_rst       = port:PB05<1><default><default><default>
```

5. VGA 和 CVBS

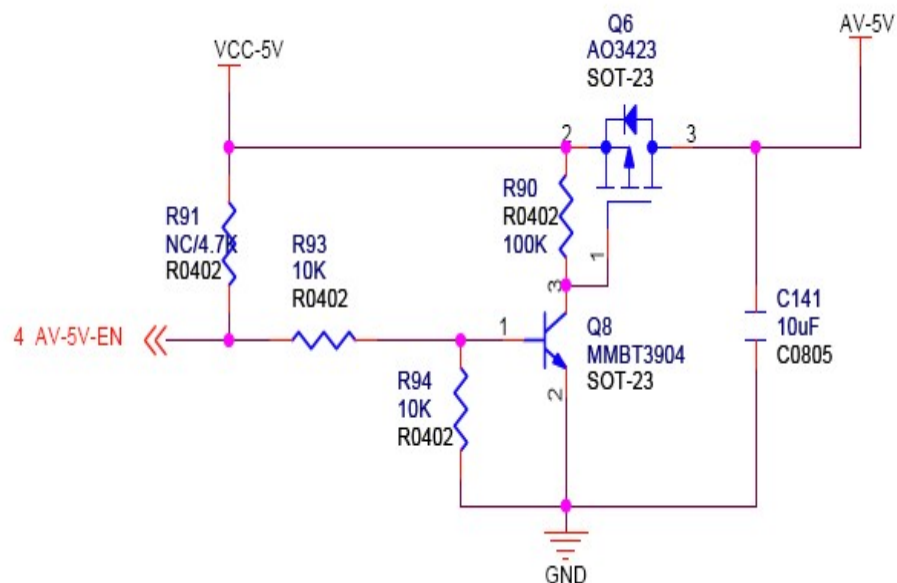
Phoenix 集成 VGA 接口，因此可以直接支持带 VGA 的显示器，跟 VGA 相关的配置如下：

```
-----
;tv out dac configuration
;dacx_src: 0:composite; 1:luma; 2:chroma; 4:Y; 5:Pb; 6: Pr; 7:none
;-----
[tv_out_dac_para]
dac_used      = 1
dac0_src      = 0
dac1_src      = 0
dac2_src      = 0
dac3_src      = 0
```

VGA 接口使用 tvout0-2 分别对应 RGB 输出，因此如果要使用 VGA 功能，需要将配置项改为：

```
dac0_src      = 4
dac1_src      = 5
dac2_src      = 6
```

Phoenix 的默认电路使用了 buffer 电路，所以如果要打开 VGA，需要使能 buffer 电路控制引脚。



具体修改方法可以在代码中直接使能。或者通过配置修改。这个将在 GPIO 使用说明里

会详细描述。

Phoenix 同样支持 CVBS 输出，而 CVBS 的对应了 tvout3,所以如果要使用 cvbs，则需要将配置项改为：

```
dac3_src          = 0
```

6. USB 控制

7. 串口配置

Phoenix 预留了 4 路串口，其中串口 0 用作 debug 接口。其他几路可以用于跟其他串口设备连接。跟串口相关的配置如下：

```

;-----
;uart configuration
;uart_type --- 2 (2 wire), 4 (4 wire), 8 (8 wire, full function)
;-----
[uart_para0]
uart_used          = 1
uart_port          = 0
uart_type          = 2
uart_tx            = port:PB22<2><1><default><default>
uart_rx            = port:PB23<2><1><default><default>

[uart_para1]
uart_used          = 0
uart_port          = 1
uart_type          = 8
uart_tx            = port:PA10<4><1><default><default>
uart_rx            = port:PA11<4><1><default><default>
uart_rts           = port:PA12<4><1><default><default>
uart_cts           = port:PA13<4><1><default><default>
uart_dtr           = port:PA14<4><1><default><default>
uart_dsr           = port:PA15<4><1><default><default>
uart_dcd           = port:PA16<4><1><default><default>
uart_ring          = port:PA17<4><1><default><default>

[uart_para2]
uart_used          = 1
uart_port          = 2
uart_type          = 4
uart_tx            = port:PI18<3><1><default><default>
uart_rx            = port:PI19<3><1><default><default>

```

```

uart_rts          = port:PI16<3><1><default><default>
uart_cts          = port:PI17<3><1><default><default>

[uart_para3]
uart_used         = 0
uart_port         = 3
uart_type         = 4
uart_tx           = port:PH00<4><1><default><default>
uart_rx           = port:PH01<4><1><default><default>
uart_rts          = port:PH02<4><1><default><default>
uart_cts          = port:PH03<4><1><default><default>

[uart_para4]
uart_used         = 0
uart_port         = 4
uart_type         = 2
uart_tx           = port:PH04<4><1><default><default>
uart_rx           = port:PH05<4><1><default><default>

[uart_para5]
uart_used         = 0
uart_port         = 5
uart_type         = 2
uart_tx           = port:PH06<4><1><default><default>
uart_rx           = port:PH07<4><1><default><default>

[uart_para6]
uart_used         = 0
uart_port         = 6
uart_type         = 2
uart_tx           = port:PA12<3><1><default><default>
uart_rx           = port:PA13<3><1><default><default>

[uart_para7]
uart_used         = 0
uart_port         = 7
uart_type         = 2
uart_tx           = port:PA14<3><1><default><default>
uart_rx           = port:PA15<3><1><default><default>

```

用户只需要使能相应的串口设备，即可使用串口来跟其他设备通信，注意串口 `uart_type` 配置项，目前 phoenix 主板都是使用 2 线的串口，因此该值应该配置为

```
uart_type         = 2
```

8. IIC 配置

Phoenix 预留了两路 IIC 接口，可以用来连接 IIC 设备，也可以用来做触摸屏的控制（触摸屏控制器就是一个 IIC 从设备）。IIC 配置比较简单，具体如下：

```

;-----
;i2c configuration
;-----
[twi0_para]
twi0_used          = 1
twi0_scl           = port:PB0<2><default><default><default>
twi0_sda           = port:PB1<2><default><default><default>

[twi1_para]
twi1_used          = 1
twi1_scl           = port:PB18<2><default><default><default>
twi1_sda           = port:PB19<2><default><default><default>

[twi2_para]
twi2_used          = 1
twi2_scl           = port:PB20<2><default><default><default>
twi2_sda           = port:PB21<2><default><default><default>

[twi3_para]
twi3_used          = 1
twi3_scl           = port:PI0<3><default><default><default>
twi3_sda           = port:PI1<3><default><default><default>

[twi4_para]
twi4_used          = 1
twi4_scl           = port:PI2<3><default><default><default>
twi4_sda           = port:PI3<3><default><default><default>

```

需要做的只是将对应的 IIC 设备使能即可。GPIO 因为是一一对应的，所以不需要修改。系统集成了 IIC 驱动，因此使能后用户可以直接使用 IIC。

9. 红外配置

Phoenix 并没有将红外接收头集成到主板中，而是以扩展接口的方式给出，具体电路可以参考硬件介绍部分。Sys_config.fex 跟红外有关的配置项为：

```

;-----
;ir --- infra remote configuration
;-----
[ir_para]
ir_used            = 1

```

```
ir_rx = port:PB04<2><default><default><default>
```

显然要使用红外，需要将 `ir_used` 设置为 1，然后将 `ir_rx` 配置成相应的引脚，Phoenix 主板对应的引脚为 PH4,所以需要将其配置为

```
ir_rx = port:PB04<2><default><default><default>
```

未完待续

PhoenixA20