SEED-DVS8168 用户指南

2012 *DSP Development Systems*

SEED-DVS8168　User's Guide

概述：

文档详细介绍基于 SEED-DVS8168 平台的 Linux 服务器下的开发套件的安装配置与使用，套件下各个部件的使用介绍，以及系统下各个启动方式的配置等操作。

名称：

SEED-DVS8168 Development Software 用户指南

版本：

version 2.0

修改：

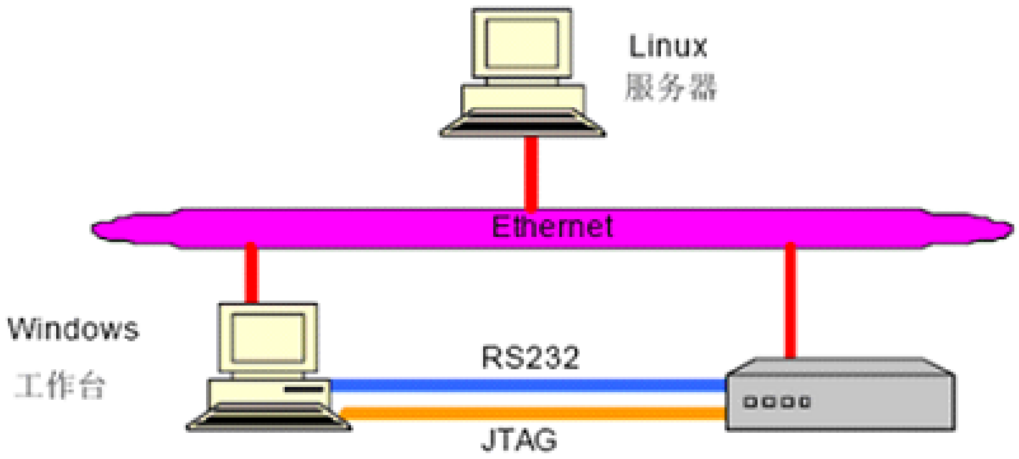
版本	修改时间	修正作者	修正说明
Version0.0	2011/10/8	林勇	文件创建
Version1.0	2011/11/9	陈文华	补充、校正
Version2.0	2012/3/28	陈文华	升级

Note：任何修改操作请在上述表格备注说明。

目　录

第 1 章 SEED-DVS8168 Linux 开发环境

SEED-DVS8168 Linux 开发环境通常包括 Linux 服务器、Windows 工作台及 SEED-DVS8168 平台三者处于同一个网络中，如下图示：



开发工程师在 Linux 服务器上建立交叉编译环境，Windows 工作台通过串口和 JTAG 与 SEED-DVS8168 开发平台连接，开发人员可以在 Windows 工作进行程序开发或者远程登陆到 Linux 服务器进行开发。

Linux 服务器搭建建议选择常用的 Linux 发行版本，便于各种资源的搜集，建议采用以下版本的 Linux 发行版：ubuntu 10.04

Linux 服务器搭建建议选择常用的 Linux 发行版本，便于各种资源的搜集，本文档中所有测试均采用 ubuntu 10.04，ubuntu 10.04 是很多 Linux 开发者采用的比较新的版本的 Linux，使用较为方便。

Linux 系统 PC 机端的安装，在此不做详细介绍，用户可以很方便的从网络上获取丰富的资源。

Note：安装 Linux 系统过程中，请勿选择安装防火墙。

第 2 章　SEED-DVS8168 SDK 安装

SEED-DVS8168 平台随机的开发软件套件为 SEED-DVS8168_SDK，该套件将繁琐的安装、配置、各个目录下程序编译器路径复杂的配置等进行简化，很大程度上减少用户的繁琐操作，降低开发者的开发难度。

Note：以下将 SEED-DVS8168_SDK 简称为 SEED-SDK。

2.1　软件组成

SEED-SDK包含了，ARM端的 ARM 2009q1-203交叉编译器，linux内核，Linux环境的NFS文件系统包及相关软件开发包等。

后面章节将详细叙述基于 LINUX 的软件安装及使用。

2.2　SEED-DVS8168 SDK 安装

SEED-SDK 的安装建议完全按照以下步骤与路径进行配置，以简化后续各种配置的繁琐，安装过程以 root 账号登陆 Linux 服务器，且开发过程也以 root 用户权限进行开发。

约定：

Host #　　表示 Linux 开发机（服务器）控制台提示符

Target #　　表示 SEED-DVS8168 平台的串口控制台提示符

SEED-SDK 安装到 Linux 服务器的安装步骤如下：

- 复制

将SEED-DVS8168 产品光盘中Linux Develop Software目录下的DaVinci开发套件SEED-DVS8168_SDK.tar.gz复制到Linux服务器的/opt目录下；

- 安装

在 Linux 服务器下进入到/opt 目录下，进行解压安装操作，使用命令：

```
Host #cd /opt
Host #tar zxvf *.tar.gz
```

该过程将所需要的软件安装到/opt 根目录下，安装过程需要 5-10 分钟，请等待完成。

SEED-SDK 安装完成后，在/opt/DVRRDK_02.00.00.23 下创建如下目录：


dvr_rdk　　Mcfw（Multi Channel Framework）框架，及对应应用程序例程

 linux_test　linux下部分硬件接口测试程序

 target　　根文件系统

 tftphome　tftp 服务器根目录，用于存放 uImage 等，便于从 tftp 服务器启动系统

 ti_tools　　所有的开发组件，及编译工具链

- SEED-SDK 配置

SEED-SDK 安装完毕仅需对其进行简单的配置即可以使用，进行 demo 等程序的编译等操作。

➢ 配置 ARM 2009q1-203 交叉编译器 PATH

```
Host#: cd /root        //先进入 root 用户根目录
Host#: gedit .bashrc     //打开环境变量文件
```

修改普通用户目录下.bashrc 文件，打开.bashrc 文件，在最后一行添加如下内容

```
PATH="/opt/DVRRDK_02.00.00.23/ti_tools/cgt_a8/arm-2009q1/bin:$PATH"
export PATH
```

保存退出并重启系统。

用户可以通过如下方式测试 ARM 2009q1-203 编译器是否可以使用，在 Linux 服务器控制台输入如下命令：

```
Host # arm-none-linux-gnueabi-gcc
```

显示如下信息时，表示安装正常：

```
seed@seed-desktop:~$ arm-none-linux-gnueabi-gcc
arm-none-linux-gnueabi-gcc: no input files
seed@seed-desktop:~$
```

➢ 安装 NFS

1. 安装 NFS 软件

```
Host# apt-get install portmap nfs-kernel-server
```

2. 修改/etc/default/portmap 成如下配置

```
# Portmap configuration file
#
# Note: if you manually edit this configuration file,
# portmap configuration scripts will avoid modifying it
```

```
# (for example, by Running 'dpkg-reconfigure portmap').


# If you want portmap to listen only to the loopback
# interface, uncomment the following line (it will be
# uncommented automatically if you configure this
# through debconf).
#OPTIONS="-i 127.0.0.1"
```

3. 配置 NFS 文件系统服务

修改/etc/exports 文件，添加如下内容

```
/opt/DVRRDK_02.00.00.23/target/rfs
*(rw,nohide,insecure,no_subtree_check,async,no_root_squash)
```

保存退出即可。

运行以下命令启动 nfs 服务:

```
Host # /etc/init.d/nfs-kernel-server restart    //重启 nfs 服务
Host # /etc/init.d/portmap restart                  //重启端口映射服务
Host #showmount -e                                        //查看添加的文件系统是否暴露
```

- TFTP SERVER 搭建

1. 检查 Linux 服务器是否已经安装 TFTP 服务器，在控制台执行以下命令检查：

```
Host # which tftp↙
```

2. 如果没有安装 tftp server 用户使用如下命令安装：

```
Host # apt-get install tftpd-hpa tftp-hpa xinetd↙
```

3. 配置 tftp server：

```
Host # gedit /etc/default/tftpd-hpa↙
```

对其内容修改如下：

```
TFTP_DIRECTORY="/opt/DVRRDK_02.00.00.23/tftphome"
TFTP_OPTIONS="-l -c -s"
```

4. 重启 tftp server：

```
Host #service tftpd-hpa restart ↙
```

5. 测试 tftp server 是否配置成功：

```
Host # echo ' hello tftp service！ ' > /opt/DVRRDK_02.00.00.23/tftphome/tftp ↙
```

```
Host # tftp ###.###.###.### ↙
tftp> get tftp↙
tftp> quit↙
Host # cat tftp↙
hello tftp service!
```

其中“###.###.###.###”为本机 IP。

至此，SEED-SDK 开发工具安装，配置完毕。

第 3 章　SEED-DVS8168 内核使用

SEED-DVS8168 平台的 Linux 内核，针对硬件平台，实现对平台的音频、视频采集、视频显示、NAND Flash 等各个外设驱动支持。

当用户对 SEED-DVS8168 下的内核驱动源码进行调整或者添加新的设备驱动后，需要对内核进行重新编译配置，编译生成内核镜像后，可以通过 tftp 下载到 SEED-DVS8168 目标板上运行，进行测试，下面详细介绍 Linux 内核的使用。

3.1　SEED-DVS8168 Linux 内核源码

SEED-DVS8168 Linux 的内核源码安装在 Linux 服务器（开发端主机）的 /opt/DVRRDK_02.00.00.23/ti_tools/linux_lsp/linux-psp-dvr-04.00.01.13/src/linux-04.00.01.13 目录下，用户需要修改内核源码时可在此目录下进行操作（注意做好备份），修改完毕在该目录下进行编译就可以生成 uImage 内核二进制镜像。

3.2　SEED-DVS8168 Linux 内核配置

SEED-DVS8168 Linux 配置编译步骤如下：

1、使用下面命令配置 SEED-DVS8168 默认的 Linux 内核（交叉编译器最好指定全路径）：

```
Host# make ARCH=arm CROSS_COMPILE=/opt/DVRRDK_02.00.00.23/ti_tools/cgt_a8/arm-2009q1/bin/arm-none-linux-gnueabi- seed_dvs8168_defconfig
```

2、如果用户需要调整内核选项，可以使用如下命令进入内核配置菜单，对编译选项进行修改：

```
Host # make ARCH=arm CROSS_COMPILE=/opt/DVRRDK_02.00.00.23/ti_tools/cgt_a8/arm-2009q1/bin/arm-none-linux-gnueabi- menuconfig
```

注：若系统中未安装 ncurses-devel 包，则先执行以下命令进行安装，否者无法显示配置界面。

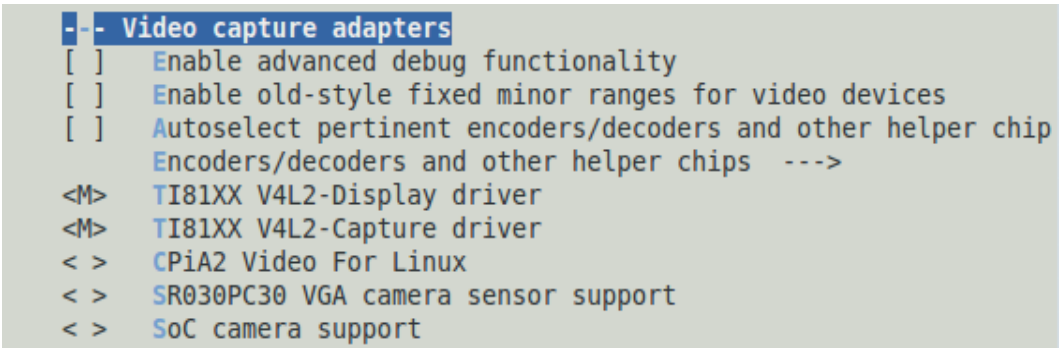
```
Host# apt-get install ncurses-dev
```

用户通过键盘上下左右键进行选择配置，添加删减不要的内核模块，配置完成退出并保

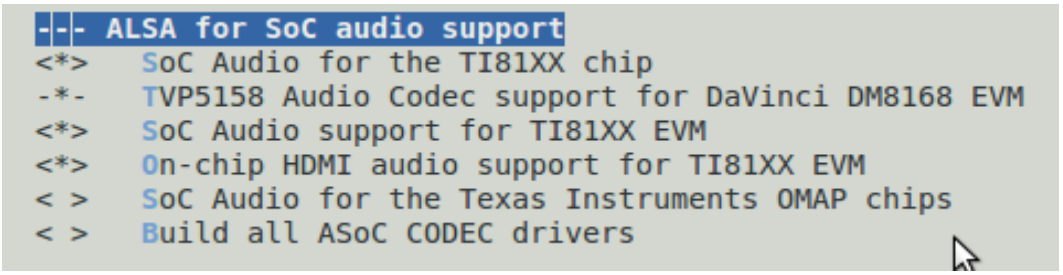
存配置即可。

下面为 SEED-DVS8168 部分 linux 驱动的配置界面。

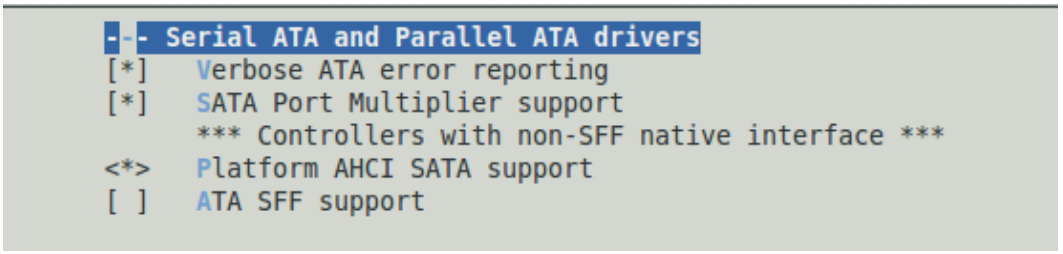
3.2.1 DVS8168 Linux 视频驱动



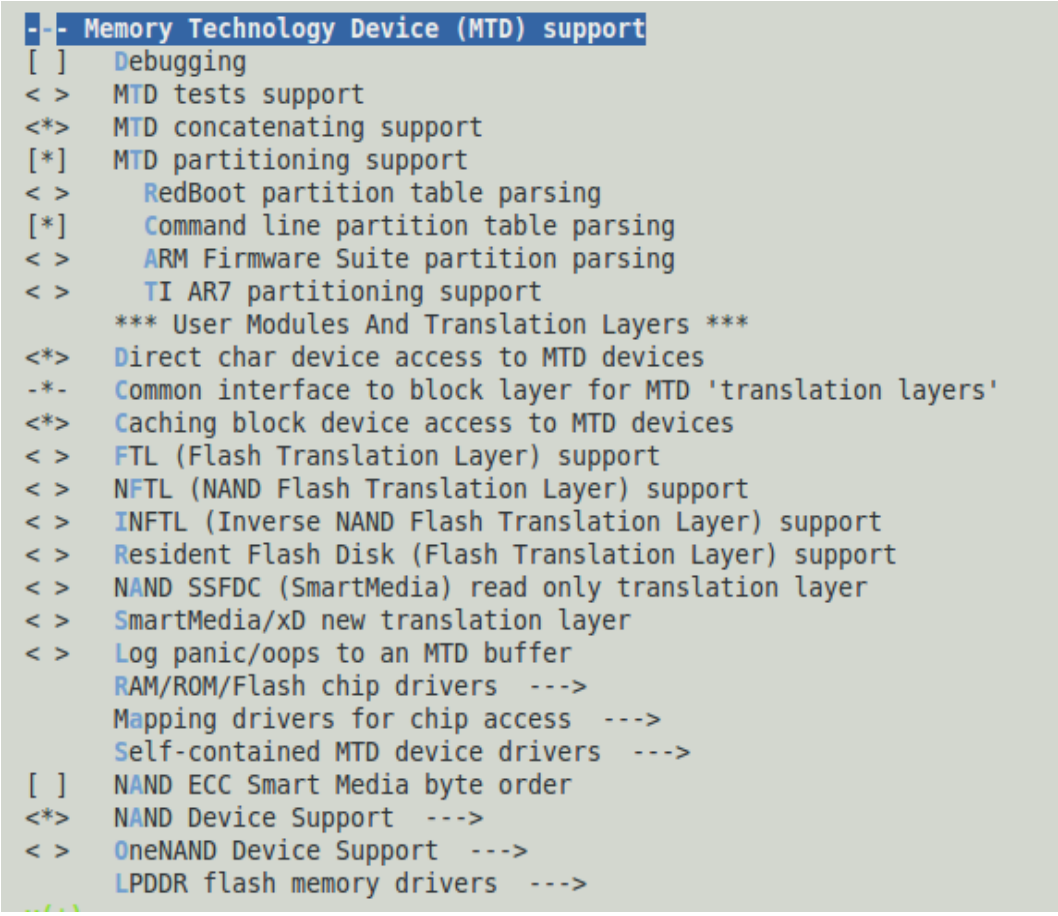
3.2.2 DVS8168 Linux 音频驱动



3.2.3 DVS8168 SATA 硬盘驱动



3.2.4 DVS8168NAND FLASH 驱动



3.3 SEED-DVS8168 Linux 内核编译

SEED-DVS8168 内核配置保存后，运行下面指令进行内核编译：(注：交叉编译器最好指定全路径)

Host # make ARCH=arm CROSS_COMPILE=/opt/DVRRDK_02.00.00.23/ti_tools/cgt_a8/arm-2009q1/bin/arm-none-linux-gnueabi- uImage

编译过程可能需要较长时间，编译完成生成 uImage 文件，该文件在内核源码当前目录下的 arch/arm/boot/目录下。

如果用户想编译模块，可以运行如下命令进行编译内核模块：

Host #make ARCH=arm CROSS_COMPILE=/opt/DVRRDK_02.00.00.23/ti_tools/cgt_a8/arm-2009q1/bin/arm-none-linux-gnueabi- modules

编译过程可能需要较长时间，编译完成生成用户需要的内核模块文件。

第 4 章　U-Boot 的使用

4.1　SEED-DVS8168 U-Boot 源码

源码在 Linux 服务器（开发端主机）/opt/DVRRDK_02.00.00.23/ti_tools/linux_lsp/linux-psp-dvr-04.00.01.13/src/u-boot-04.00.01.13 目录下，用户需要修改源码时可在此目录下进行操作（注意做好备份），修改完毕在该目录下进行编译就可以生成 u-boot.noxip.bin 二进制镜像。

4.2　SEED-DVS8168 U-Boot 配置

进入目录 uboot 所在目录

Host #cd /opt/DVRRDK_02.00.00.23/ti_tools/linux_lsp/linux-psp-dvr-04.00.01.13/src/u-boot-04.00.01.13

进入目录后使用下面命令清理

Host# make distclean

使用如下命令编译 u-boot

Host # make ARCH=arm CROSS_COMPILE=/opt/DVRRDK_02.00.00.23/ti_tools/cgt_a8/arm-2009q1/bin/arm-none-linux-gnueabi- ti8168_evm_config

Host# make ARCH=arm CROSS_COMPILE=/opt/DVRRDK_02.00.00.23/ti_tools/cgt_a8/arm-2009q1/bin/arm-none-linux-gnueabi- u-boot.ti

编译成功，在目录下生成 **u-boot.noxip.bin** 文件。

4.3　U-Boot 常用命令和常用环境变量

4.3.1　U-Boot 常用命令

U-Boot 提供了周详的命令帮助，通过 help 命令还能查看每个命令的参数说明。由于研

发过程的需要，有必要先把 U-Boot 命令的用法弄清晰。接下来，解释一下常用命令的功能和参数。

变量	用法：变量 [command ...]。 说明：列出命令的帮助信息，当不带参数时，列出所有命令及简要说明
help	同 ?
bootm	用法：bootm [addr [arg ...]] 说明：bootm 命令能引导启动存储在内存中的程式映像。这些内存包括 RAM 和能永久保存的 Flash。 第 1 个参数 addr 是程式映像的地址，这个程式映像必须转换成 U-Boot 的格式。 第 2 个参数对于引导 Linux 内核有用，通常作为 U-Boot 格式的 RAMDISK 映像存储地址；也能是传递给 Linux 内核的参数（缺省情况下传递 bootargs 环境变量给内核）。
bootp	用法：bootp [LoadAddress] [bootfilename] 说明：bootp 命令通过 bootp 请求，需求 DHCP 服务器分配 IP 地址，然后通过 TFTP 协议下 载指定的文件到内存。 第 1 个参数是下载文件存放的内存地址。 第 2 个参数是要下载的文件名称，这个文件应该在研发主机上准备好。
cmp	用法：cmp [.b, .w, .l] addr1 addr2 count 说明：cmp 命令能比较 2 块内存中的内容。.b 以字节为单位；.w 以字为单位；.l 以长字为单 位。注意：cmp.b 中间不能保留空格，需要连续敲入命令。 第 1 个参数 addr1 是第一块内存的起始地址。 第 2 个参数 addr2 是第二块内存的起始地址。 第 3 个参数 count 是要比较的数目，单位按照字节、字或长字。
cp	用法：cp [.b, .w, .l] source target count

	说明：cp 命令能在内存中复制数据块，包括对 Flash 的读写操作。 第 1 个参数 source 是要复制的数据块起始地址。 第 2 个参数 target 是数据块要复制到的地址。这个地址如果在 Flash 中，那么会直接调用写 Flash 的函数操作。所以 U-Boot 写 Flash 就使用这个命令，当然需要先把对应 Flash 区域擦干净。 第 3 个参数 count 是要复制的数目，根据 cp.b cp.w cp.l 分别以字节、字、长字为单位。
erase	用法：erase start end erase N:SF[-SL] erase bank N erase all erase 命令能擦 Flash。 参数必须指定 Flash 擦除的范围。 说明：按照起始地址和结束地址，start 必须是擦除块的起始地址；end 必须是擦除末尾块的结束地址。这种方式最常用。举例说明：擦除 0x20000？0x3ffff 区域命令为 erase 20000 3ffff。 按照组和扇区，N 表示 Flash 的组号，SF 表示擦除起始扇区号，SL 表示擦除结束扇区号。另外，还能擦除整个组，擦除组号为 N 的整个 Flash 组。擦除全部 Flash 只要给出一个 all 的参数即可。
printenv	用法：printenv name ... 说明：printenv 命令打印环境变量。 能打印全部环境变量，也能只打印参数中列出的环境变量。
sleep	用法：sleep N 说明：sleep 命令能延迟 N 秒钟执行，N 为十进制数。
setenv	用法：setenv name value ... 说明：setenv 命令能设置环境变量。 第 1 个参数是环境变量的名称。 第 2 个参数是要设置的值，如果没有第 2 个参数，表示删除这个环境变量。
tftpboot	用法：tftpboot [LoadAddress] [bootfilename]

说明：tftpboot 命令能使用 TFTP 协议通过网络下载文件。按照二进制文件格式下载。另外使用这个命令，必须设置好相关的环境变量。例如 serverip 和 ipaddr。 第 1 个参数 LoadAddress 是下载到的内存地址。 第 2 个参数是要下载的文件名称，必须放在 TFTP 服务器相应的目录下。

4.3.2 U-Boot 常用环境变量

U-Boot 中使用 printenv 可以查看各个环境变量，使用 setenv 可以设置各个环境变量，而 saveenv 则用于保存各个环境变量。 常用环境变量如下表示：

环境变量	描述
bootdelay	设置自启动延时时间。单位为秒。只有当bootcmd 变量被设置后，该变量才有效。该变量值范围为大于等于-1 的整数。当设置为-1 时，关闭自启动的功能。在延迟时间内可按任意键切换到命令行模式。 例：setenv bootdelay 4 说明：设置自启动延时4 秒。 例：setenv bootdelay -1 说明：关闭自启动功能。
netmask	定义以太网接口的掩码 例：setenv netmask 255.255.255.0 说明：设置子网掩码为 255.255.255.0
gatewayip	设置网关。 例：setenv gatewayip 192.168.0.1 说明：设置网关 ip 地址为 192.168.0.1。
ethaddr	定义以太网接口的 MAC 地址 例：setenv ethaddr xx：xx：xx：xx：xx：xx 说明：设置以太网MAC地址为xx：xx：xx：xx：xx：xx Note：设置 ethaddr 后，应运行 saveenv，重启才能使 MAC 地址配置有效
serverip	定义 tftp 服务器端的 IP 地址

	例：setenv serverip xxx.xxx.xxx.xxx 说明：设置tftp服务器ip为xxx.xxx.xxx.xxx
ipaddr	定义本地开发板的 IP 地址 例：setenv ipaddr xxx.xxx.xxx.xxx 说明：设置ip地址为xxx.xxx.xxx.xxx
bootcmd	定义自动启动时执行的命令 例：setenv bootcmd bootm 0x23450000 说明：设置启动后自动执行 0x23450000 处的代码。
Stdin	定义标准输入设备，一般是串口
stdout	定义标准输出设备，一般是串口
Stderr	定义标准出错信息输出设备，一般是串口

第 5 章　常用启动方式配置

SEED-DVS8168 平台支持如下启动方式：

- TFTP 下载内核启动挂载网络文件系统方式
- NAND Flash 启动挂载网络文件系统方式
- NAND Flash 启动挂载 jffs2 文件系统方式

SEED-DVS8168 平台默认的启动方式是从 NAND Flash 中启动 U-Boot、uImage，挂载 jffs2 文件系统的方式。

5.1　启动方式硬件连接

- 将串口线的一端连接 SEED-DVS8168 平台的 RS232 串口 J41，另一端连接 Windows PC 机的 COM1（或 COM2 等）串口；
- 将网线一端连接 SEED-DVS8168 平台的网络接口 J7，另一端连接路由器或其他网络接口；
- 将板卡 SW3 开关置成从 NAND 启动。
  SW3[10:1] ---> 00000 10010

5.2　主机端串口控制台搭建

用户通过主机端串口同 SEED-DVS8168 交互，对 SEED-DVS8168 进行控制并运行程序等。主机端串口使用 PC 机 windows 系统自带的超级终端即可。配置如下：

- 点击 PC 机左下角开始-->程序-->附件-->通讯-->超级终端；
- 在“您的区号（或城市号）是什么（C）？”下键入 010 后点击确定；
- 点击确定，在新弹出的对话框中输入你喜欢的名称，如 davinci；
- 在新的对话框中的“连接时使用”下选择你希望使用的串口设备，点击确定；
- 在端口设置选项中配置波特率 115200，数据位 8，奇偶校验无，停止位 1，数据流控制无。

5.3 TFTP 下载内核启动挂载网络文件系统方式

TFTP 启动需要将内核镜像文件 uImage 放置在 tftp 服务器配置路径下，在第二章中 SDK 包的配置中已经将该路径配置为/opt/ DVRRDK_02.00.00.23/tftphome/。

执行以下命令将 uImage 复制到/opt/DVRRDK_02.00.00.23/tftphome/目录下：

Host # cp /opt/DVRRDK_02.00.00.23/ti_tools/linux_lsp/linux-psp-dvr-04.00.01.13/src/linux-04.00.01.13/arch/arm/boot/uImage /opt/DVRRDK_02.00.00.23/tftphome/

TFTP 下载内核启动挂载网络文件系统方式，即通过 tftp 服务器下载内核，文件系统挂载到 Linux 服务器下的已经搭建好的/opt/DVRRDK_02.00.00.23/target/rfs 文件系统目录下。具体配置启动方式操作如下：

- SEED-DVS8168 上电启动，通过 PC 机串口控制台显示启动信息如下：

- 显示 Hit any key to stop autoboot： 3 时按下回车键，中断系统自动系统，进行启动参数配置，此时显示如下提示符： SEED-DVS8168_v1.0#；

- 配置 TFTP 服务器 IP 地址：

```
SEED-DVS8168_v1.0# setenv serverip xxx.xxx.xxx.xxx
```

其中，xxx.xxx.xxx.xxx 为 Linux 主机服务器的 IP 地址；

- 配置 SEED-DVS8168 平台的 IP 地址：

```
SEED-DVS8168_v1.0# setenv ipaddr xxx.xxx.xxx.xxx
```

其中，xxx.xxx.xxx.xxx 为 *DVS8168* 的 IP 地址，此处配置为静态 IP 地址，需要与 serverip 在同一网段即可；

- 配置 SEED-DVS8168 启动环境变量

```
SEED-DVS8168_v1.0# setenv bootargs 'mem=116M console=ttyO0,115200n8
root=/dev/nfs nfsroot=192.168.253.142:/opt/DVRRDK_02.00.00.23/target/rfs
ip=192.168.253.183:192.168.253.142:192.168.253.1:255.255.255.0::eth0:off vram=20M
notifyk.vpssm3_sva=0xbfd00000'
SEED-DVS8168_v1.0# setenv bootcmd 'tftp 0x81000000 uImage;bootm 0x81000000'
SEED-DVS8168_v1.0# save
```

Note：ip 后面参数需要根据网络环境进行修改，其中“192.168.253.183”为板卡 IP，“192.168.253.142”为 sever IP，“192.168.253.1”为网关，“255.255.255.0”为子网掩码，若 IP 使用动态获取，可将 ip 参数配置为“ip=dhcp”。

- 配置完成系统启动，敲如下命令启动系统

```
SEED-DVS8168_v1.0# boot
```

5.4 NAND Flash 内核启动挂载网络文件系统方式

NAND Flash 内核启动挂载网络文件系统方式，即通过读取 NAND Flash 中烧写的内核启动，文件系统挂载到 Linux 服务器下的已经搭建好的/opt/DVRRDK_02.00.00.23/target/rfs 文件系统目录下。具体配置启动方式操作如下：

- SEED-DVS8168 上电启动，通过 PC 机串口控制台显示启动信息如下：

- 显示 Hit any key to stop autoboot： 3 时按下回车键，中断系统自动系统，进行启动参数配置，此时显示如下提示符：SEED-DVS8168_v1.0#；
- 配置启动参数，在提示符下输入以下配置参数

```
SEED-DVS8168_v1.0# setenv bootargs 'mem=116M console=ttyO0,115200n8
root=/dev/nfs nfsroot=192.168.253.142:/opt/DVRRDK_02.00.00.23/target/rfs
ip=192.168.253.183:192.168.253.142:192.168.253.1:255.255.255.0::eth0:off vram=20M
notifyk.vpssm3_sva=0xbfd00000'
SEED-DVS8168_v1.0# setenv bootcmd 'nand read 0x81000000 0x00580000
0x260000;bootm 0x81000000'
SEED-DVS8168_v1.0#save
```

Note：ip 后面参数需要根据网络环境进行修改，其中“192.168.253.183”为板卡 IP，“192.168.253.142”为 sever IP，“192.168.253.1”为网关，“255.255.255.0”为子网掩码，若 IP 使用动态获取，可将 ip 参数配置为“ip=dhcp”。

- 配置完成系统启动，敲如下命令启动系统

```
SEED-DVS8168_v1.0# boot
```

Note：NAND Flash 中内核的烧写操作请参考第六章第二节内容。

5.5 NAND Flash 内核 jffs2 文件系统方式

NAND Flash 内核 jffs2 文件系统方式，即通过读取 NAND Flash 中烧写的内核启动，文件系统挂载到 NAND Flash 中烧写的文件系统目录下，文件系统格式为 jffs2。具体配置启动方式操作如下：

- SEED-DVS8168 上电启动，通过 PC 机串口控制台显示启动信息如下：

- 显示 Hit any key to stop autoboot： 3 时按下回车键，中断系统自动系统，进行启动参数配置，此时显示如下提示符：SEED-DVS8168_v1.0#；

- 配置启动参数，在提示符下输入以下配置参数

```
SEED-DVS8168_v1.0#setenv bootargs ' mem=116M console=ttyO0,115200n8
root=/dev/mtdblock4 rw rootfstype=jffs2
```

```
ip=192.168.253.183:192.168.253.142:192.168.253.1:255.255.255.0::eth0:off vram=20M
notifyk.vpssm3_sva=0xbfd00000'
SEED-DVS8168_v1.0#setenv bootcmd 'nand read 0x81000000 0x00580000
0x260000;bootm 0x81000000'
SEED-DVS8168_v1.0#save
```

Note：ip 后面参数需要根据网络环境进行修改，其中“192.168.253.183”为板卡 IP，“192.168.253.142”为 sever IP，“192.168.253.1”为网关，“255.255.255.0”为子网掩码，若 IP 使用动态获取，可将 ip 参数配置为“ip=dhcp”。

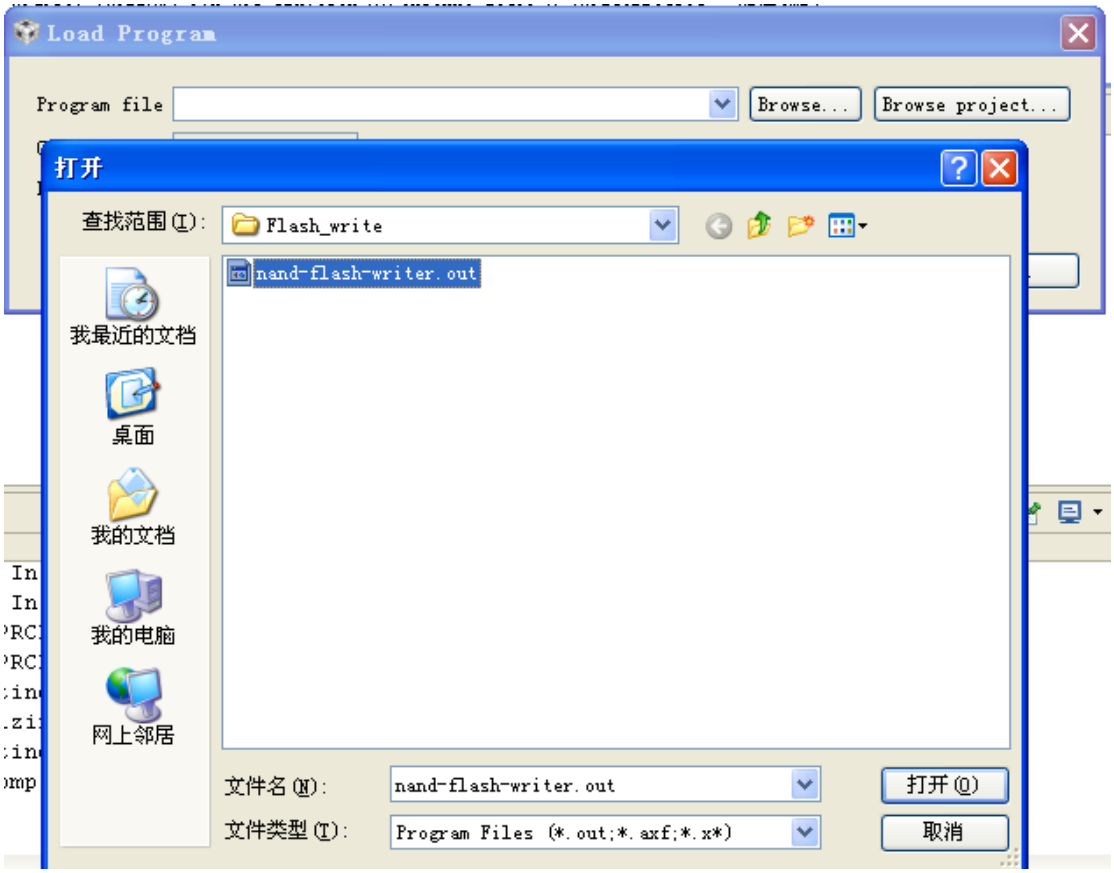
- 启动 NAND Flash 中的内核，在提示符下输入命令：

```
SEED-DVS8168_v1.0#boot
```

第 6 章　Nandflash 烧写

6.1　烧写 U-BOOT 到 NAND FLASH

- 打开 CCStudio v5，单击 CortexA8，从 SEED-DVS8168 光盘根目录 Flash_write 文件夹中加载 gel 文件(evm816x.gel)，并连接 CortexA8，连接 SEED-DVS8168 目标板；
- 从 SEED-DVS8168 光盘根目录 Flash_write 文件夹中加载可执行文件加载 nand-flash-writer.out，CCS 上操作操作路径为“Run->Load->Load Program”



- 点击运行，对话框出现如下内容，选择“1”，按回车键，进行 uboot 烧写

```
[CortexA8] Choose your operation
[CortexA8] Enter 1 ---> To Flash an Image
[CortexA8] Enter 2 ---> To ERASE the whole NAND
[CortexA8] Enter 3 ---> To DDR Simple test
[CortexA8] Enter 4 ---> To EXIT
1
```

- 然后输入要烧写的 uboot 文件路径，并按回车开始烧写 uboot；

```
[CortexA8] Welcome to CCS Nand Flash Utility
[CortexA8]
[CortexA8]
[CortexA8] Choose your operation
[CortexA8] Enter 1 ---> To Flash an Image
[CortexA8] Enter 2 ---> To ERASE the whole NAND
[CortexA8] Enter 3 ---> To DDR Simple test
[CortexA8] Enter 4 ---> To EXIT
1
[CortexA8] Enter image file path
D:\Flash_write\u-boot.noxip.bin
[CortexA8] Starting NETRA NAND writer
```

- 如果烧写程序，将输出如下信息。

```
[CortexA8] Writing image data to Block 1 Page0x12
[CortexA8] Writing image data to Block 1 Page0x13
[CortexA8] Writing image data to Block 1 Page0x14
[CortexA8] Writing image data to Block 1 Page0x15
[CortexA8] Writing image data to Block 1 Page0x16
[CortexA8] Writing image data to Block 1 Page0x17
[CortexA8] Writing image data to Block 1 Page0x18
[CortexA8] Writing image data to Block 1 Page0x19
[CortexA8] Application is successfully flashed
[CortexA8]
[CortexA8]
[CortexA8] NAND boot preparation was successful!
```

6.2 烧写 uImage 到 NAND FLASH

完成 U-boot 烧写后，就可烧写 uImage 文件，采用 TFTP 方式进行烧写。具体步骤如下：

- SEED-DVS8168 上电启动，通过 PC 机串口控制台显示启动信息如下：

- 显示 Hit any key to stop autoboot： 3 时按下回车键，中断系统自动系统，进行启动参数配置，此时显示如下提示符：SEED-DVS8168_v1.0#
- 将烧写镜像文件 uImage 从 SEED-DVS8168 光盘根目录 Flash_write 文件夹中放入 tftp server 的根目录下，如 linux 主机/opt/ DVRRDK_02.00.00.23/tftphome/目录下（默认该目录下已经包含该文件）；
- 设置系统环境变量，如下：

```
SEED-DVS8168_v1.0# setenv serverip ##.##.##.##
SEED-DVS8168_v1.0# setenv ipaddr ##.##.##.##
```

- 下载 uImage 镜像文件到 DDR 空间；

```
SEED-DVS8168_v1.0# tftp 0x81000000 uImage
```

- 擦除 Nand flash 内容；

```
SEED-DVS8168_v1.0# nand erase <start> <length>
```

其中：

start ：为需要擦出 nand flash 的起始地址；

length ：为长度

需要注意 start 和 length 需根据具体情况配置，SEED-DVS8168 中，NAND FLASH kernel 分区的起始地址是 0x00580000，因而该值最好配成 0x00580000；length 取决于 uImage 文件的大小。

NOTE：烧写 SEED-DVS8168 光盘中提供的内核可以使用如下命令

```
SEED-DVS8168_v1.0# nand erase 0x00580000 0x440000
```

- 烧写 uImage 镜像文件

```
SEED-DVS8168_v1.0# nand write 0x81000000 <start> <length>
```

NOTE：烧写 SEED-DVS8168 光盘中提供的内核可以使用如下命令

```
SEED-DVS8168_v1.0# nand write 0x81000000 0x00580000 0x300000
```

- 等待烧写完成。
- 设置启动命令：

```
SEED-DVS8168_v1.0# setenv bootcmd 'nand read 0x81000000 0x00580000
 0x260000;bootm 0x81000000'
SEED-DVS8168_v1.0# save
```

重启板卡，进入文件系统。

6.3 烧写根文件系统到 NAND FLASH

完成 uImage 烧写后，就可以烧写根文件系统，根文件系统通过 NFS 烧写到 NAND FLASH 中，具体步骤如下：

- SEED-DVS8168 上电启动，通过 PC 机串口控制台显示启动信息如下：

- 显示 Hit any key to stop autoboot： 3 时按下回车键，中断系统自动系统，进行根文件系统烧写，此时显示如下提示符：SEED-DVS8168_v1.0#；
- 配置启动参数，在提示符下输入以下配置参数

```
SEED-DVS8168_v1.0# setenv bootargs 'mem=116M console=ttyO0,115200n8
root=/dev/nfs nfsroot=192.168.253.142:/opt/DVRRDK_02.00.00.23/target/rfs
ip=192.168.253.183:192.168.253.142:192.168.253.1:255.255.255.0::eth0:off vram=20M
notifyk.vpssm3_sva=0xbfd00000'
SEED-DVS8168_v1.0# setenv bootcmd 'nand read 0x81000000 0x00580000
0x260000;bootm 0x81000000'
```

- 启动 NAND Flash 中的内核，并挂载 NFS

```
SEED-DVS8168_v1.0#boot
```

- 系统启动后输入如下命令进行根文件系统的烧写

```
seeddvs8168 login: root
root@seeddvs8168:~#cd /mnt
```

```
root@seeddvs8168:~#./restore.sh
```

- 等待提示烧写完成。
- 重启系统系统后，设置启动命令：

```
SEED-DVS8168_v1.0# setenv bootargs ' mem=116M console=ttyO0,115200n8
 root=/dev/mtdblock4 rw rootfstype=jffs2
 ip=192.168.253.183:192.168.253.142:192.168.253.1:255.255.255.0::eth0:off vram=20M
 notifyk.vpssm3_sva=0xbfd00000'
SEED-DVS8168_v1.0# save
```

重启板卡，板卡将挂载烧写的 jffs2 根文件系统。

6.4 烧写 BMP 格式 logo 文件

烧写 Logo 文件可用于系统启动时在 HMDI 接口显示。Logo 文件的烧写采用 TFTP 方式进行烧写。烧写完 uboot 即可进行该项烧写操作，具体步骤如下：

- SEED-DVS8168 上电启动，通过 PC 机串口控制台显示启动信息如下：

- 显示 Hit any key to stop autoboot： 3 时按下回车键，中断系统自动系统，进行启动

参数配置，此时显示如下提示符：SEED-DVS8168_v1.0#

- 将烧写镜像文件 720p.bmp 从 SEED-DVS8168 光盘根目录 Flash_write 文件夹中放入 tftp server 的根目录下，如 linux 主机/opt/ DVRRDK_02.00.00.23/tftphome/目录下（默认该目录下已经包含该文件）；
- 设置系统环境变量，如下：

```
SEED-DVS8168_v1.0# setenv serverip ##.##.##.##
SEED-DVS8168_v1.0# setenv ipaddr ##.##.##.##
```

若前面步骤已经进行了这些配置并保存，无需重复配置。

- 下载 uImage 镜像文件到 DDR 空间；

```
SEED-DVS8168_v1.0# tftp 0x81000000 720p.bmp
```

- 擦除 Nand flash 内容；

```
SEED-DVS8168_v1.0# nand erase <start> <length>
```

其中：

start ：为需要擦出 nand flash 的起始地址；

length ：为长度

需要注意 start 和 length 需根据具体情况配置，SEED-DVS8168 中，NAND FLASH 存放 logo 的分区起始地址是 0x280000；length 取决于 logo 文件的大小。

NOTE：烧写 SEED-DVS8168 光盘中提供的 logo 文件可以使用如下命令

```
SEED-DVS8168_v1.0# nand erase 0x00280000 0x300000
```

- 烧写 logo 文件

```
SEED-DVS8168_v1.0# nand write 0x81000000 <start> <length>
```

NOTE：烧写 SEED-DVS8168 光盘中提供的 logo 文件可以使用如下命令

```
SEED-DVS8168_v1.0# nand write 0x81000000 0x00280000 0x300000
```

- 等待烧写完成。
- 设置启动命令：

uboot 中启动的 logo 的命令为“logo on”，其命令格式如下：

logo on <logo_read_address> <final_bmp address> <Display time in seconds> <Display fps (less than 60)>

其中：

logo_read_address：logo 数据源地址；

final_bmp address：logo 数据目的地址；

Display time in seconds：logo 显示的时间，单位为秒；

Display fps：logo 显示的帧率，必须小于 60

若从 NAND 中启动内核，并启动 logo，则可以设置 uboot 命令参数如下:

```
SEED-DVS8168_v1.0# setenv bootcmd 'nand read 0x82000000 0x280000
 0x300000;logo on 0x82000000 0xA0000000 100 60;nand read 0x81000000
 0x00580000 0x260000;bootm 0x81000000'
```

若从 tftp 启动内核，并启动 logo，则可以设置 uboot 命令参数如下:

```
SEED-DVS8168_v1.0# setenv bootcmd 'nand read 0x82000000 0x280000
 0x300000;logo on 0x82000000 0xA0000000 40 60; tftp 0x81000000 uImage;bootm
 0x81000000'
```

- 保存启动命令：

```
SEED-DVS8168_v1.0# save
```

重启板卡，即可在系统启动时显示 logo 文件。

第 7 章　CCS 程序测试

SEED-DVS8168 软件开发环境须采用 TI 的 CCSv5 版本。SEED-DVS8168 平台测试程序提供了 DDR、NAND Flash、RTC、GPIO、音频采集、SD 内存卡接口、串口、网络接口、USB 接口、视频接口等测试例程，实现对 SEED-DVS8168 平台各个外设的测试功能。

7.1　SEED-DVS8168 测试准备

SEED-DVS8168 平台硬件测试时，硬件统一连接如下：

- 将仿真器的 JTAG 连接器与 SEED-DVS8168 平台的 JTAG 仿真器插座连接；
- 将 SEED-DVS8168 的 SW3 全部置“0”；
- SEED-DVS8168 平台的 S3 连接 12V 电源，给系统上电；

7.2　硬件测试概述

SEED-DVS8168 的测试程序包括以下几个方面：

- SEED-DVS8168 外设 DDR 的操作示例；
- SEED-DVS8168 外设 NandFlash 的操作示例；
- SEED-DVS8168 外设 AIC3104 的操作示例；
- SEED-DVS8168 外设网口的操作示例；
- SEED-DVS8168 外设 GPIO 的操作示例；
- SEED-DVS8168 I2C 总线上挂载设备的检测示例；
- SEED-DVS8168 外设 RTC 芯片 isl12026 的操作示例；
- SEED-DVS8168 外设电源监测芯片的操作示例；
- SEED-DVS8168 片上 RTC 的操作示例；
- SEED-DVS8168 外设 RS485 串口的操作示例；
- SEED-DVS8168 外设 RS232 串口的操作示例；

如上程序基于 CortexA8 上的程序，在对片上各模块进行测试时，需要添加 GEL 文件，文件路径是 Seed_dvs8168_hardware_test\gel。

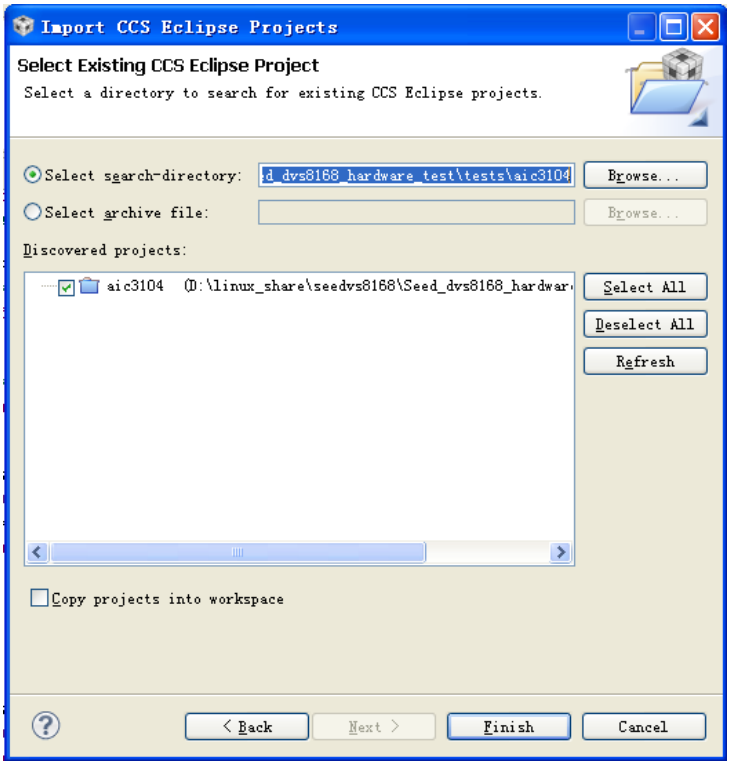
注意：DDR 测试时请加载 gel 文件 DM816x_evm_ddr3.gel，其它测试加载 gel 文件 evm816x.gel

7.3 CCS Studio IDE V5 下硬件测试

7.3.1 aic3104 的测试

1. 测试准备
   - 连接 BNC 线到音频输入口 MIC IN（位于 J2 的下接口），另一端连接音源；
   - 连接 BNC 线到音频输出口 LINE OUT（位于 J2 的上接口），另一端连接耳机、音箱或电视音频口；

   注：连接有源音箱或电视音频口时，将 SEED-DVS8168 的 C9、C10 取下。
2. aic3104 的测试主要测试 tlv320aic3104 芯片采集声音和播放声音正常，步骤如下：
   - 在 CCS 中用 file－>import->general->exsiting projects into workplace…命令，如图所示导入 Seed_dvs8168_hardware_test\tests\aic3104 目录；



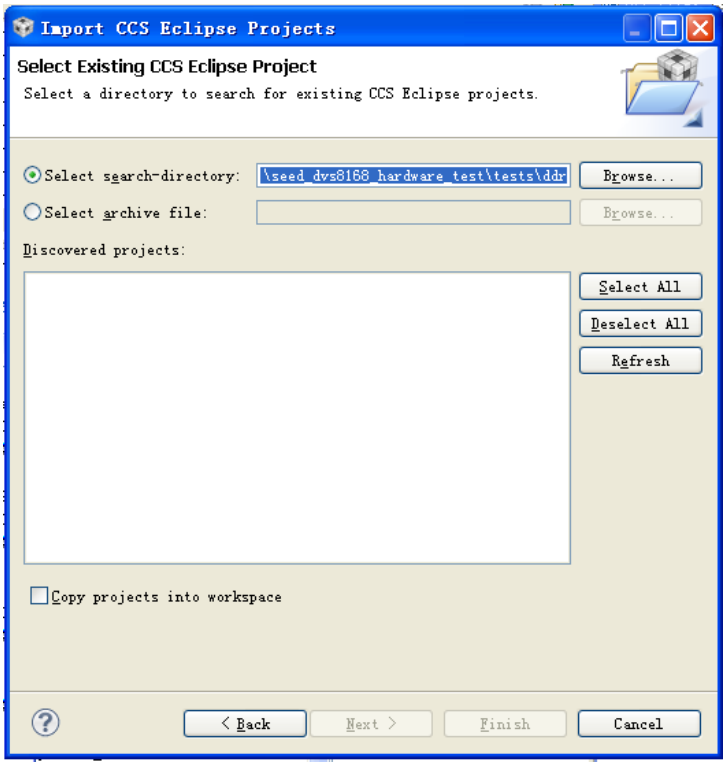
   - 选中当前工程，点击右键，在弹出的对话框中选择“built project”；
   - 在 CCS 中用了 Run->Load－>Load Program…命令，加载 aic3104\Debug 目录下的 aic3104.out 文件；
   - 在 CCS 中用了 Run->resume 运行程序或者按 F8（Run）执行程序，音频输出设备上会听到输入进来的声音。

```
[CortexA8] 01  Testing AIC3104 MCASP...
[CortexA8] <-> Audio Loopback from Line In --> to Lineout
[CortexA8]     PASS
[CortexA8]
[CortexA8] ***ALL Tests Passed***
```

7.3.2 DDR 的测试

DDR 的测试主要测试 dm9168 与 DDR 芯片间读写功能是否正常，步骤如下：

- 在 CCS 中用 file－>import->general->exsiting projects into workplace…命令，如图所示导入 Seed_dvs8168_hardware_test\tests\ddr 目录；

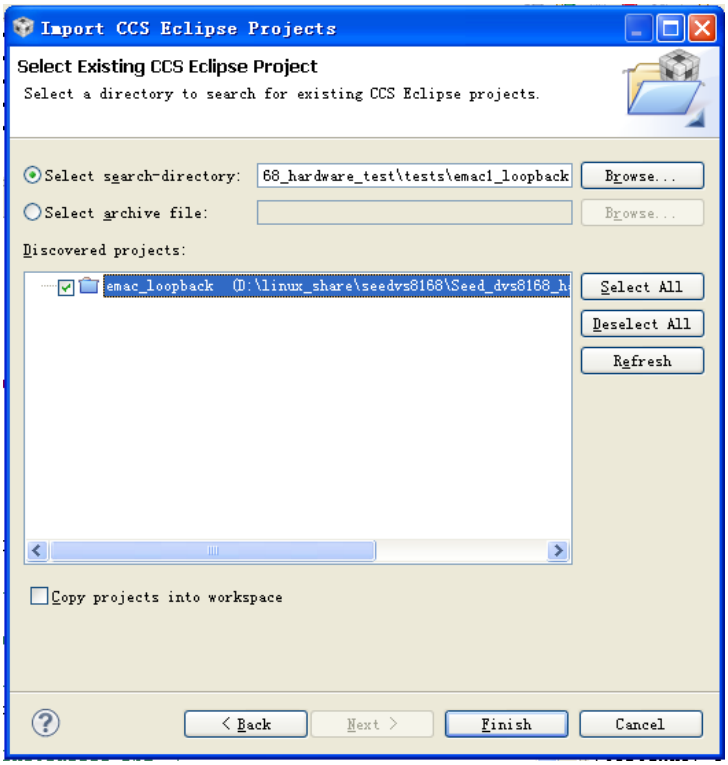


- 选中当前工程，点击右键，在弹出的对话框中选择“built project”；
- 在 CCS 中用 Scripts->DM816x External Memories->DDR3_796MHZ_doall 命令，加载 DDR3 设置；
- 在 CCS 中用了 Run->Load－>Load Program…命令，加载 DDR\Debug 目录下的 ddr.out 文件；
- 在 CCS 中用了 Run->resume 运行程序或者按 F8（Run）执行程序，如果程序执行结束时，在 Console 窗口中输出信息“Pass”，则说明测试成功；否则，测试失败。

注意：本测试程序使用的 gel 文件和其它测试程序使用的不同。

7.3.3 emac_loopback 的测试

1. 测试准备
   - 连接回环网线到 NET1
2. emac1_loopback 的测试主要测试网口 1 功能是否正常，步骤如下：
   - 在 CCS 中用 file－>import->general->exsiting projects into workplace…命令，如图所示导入 Seed_dvs8168_hardware_test\tests\emac1_loopback 目录；



- 选中当前工程，点击右键，在弹出的对话框中选择“built project”；
- 在 CCS 中用了 Run->Load－>Load Program…命令，加载 emac1_loopback\Debug 目录下的 emac_loopback.out 文件；
- 在 CCS 中用了 Run->resume 运行程序或者按 F8（Run）执行程序，若运行结束，显示下图信息，说明测试成功，否则，测试失败。

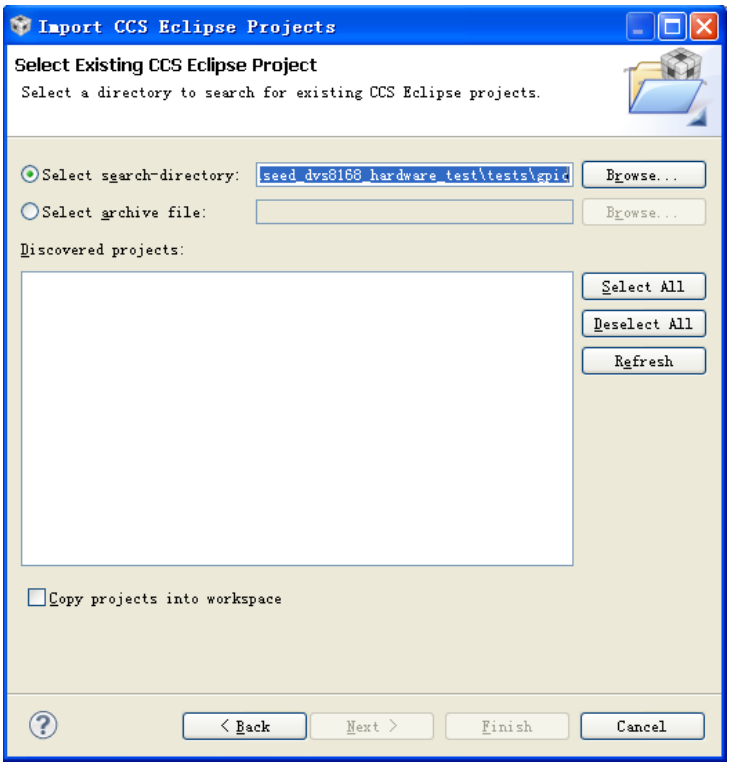
```
[CortexA8] 02  Testing MII loopback...
[CortexA8]     PHY found at address 1
[CortexA8]     In MII mode
[CortexA8]     Waiting for link...
[CortexA8]     Link Detected
[CortexA8]     PASS
[CortexA8]
[CortexA8] ***ALL Tests Passed***
```

注：要测试网口 2 使用 emac2_loopback 中程序即可，测试步骤和上述步骤一致。

7.3.4 gpio 的测试

gpio 的测试主要测试用户可用 gpio 功能是否正常，步骤如下：

- 在 CCS 中用 file－>import->general->exsiting projects into workplace…命令，如图所示导入 Seed_dvs8168_hardware_test\tests\gpio 目录；

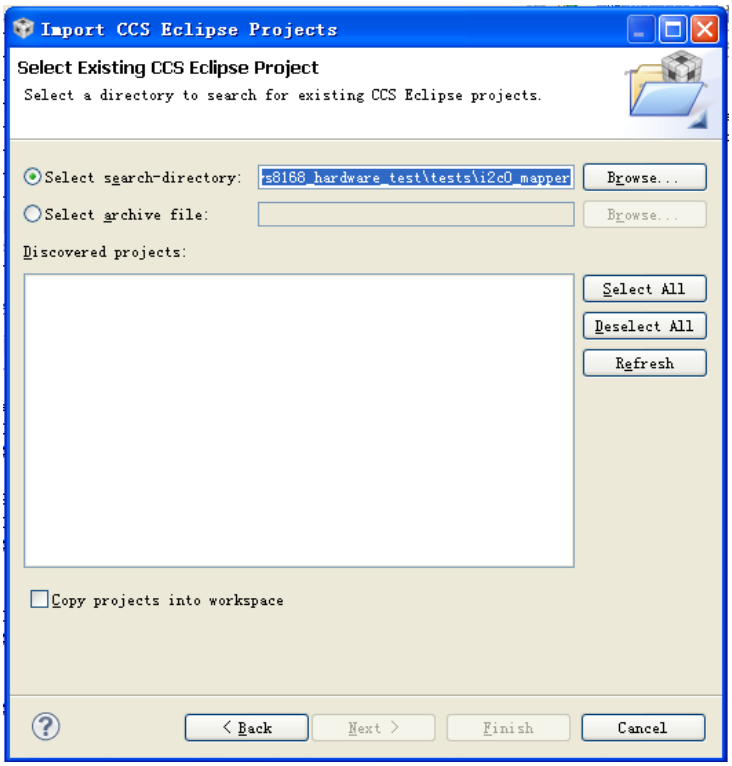


- 选中当前工程，点击右键，在弹出的对话框中选择“built project”；
- 在 CCS 中用了 Run->Load－>Load Program…命令，加载 gpio\Debug 目录下的 gpio.out 文件；
- 在 CCS 中用了 Run->resume 运行程序或者按 F8（Run）执行程序，程序运行时继电器发出会发出响声，否者测试失败。

7.3.5 i2c0_mapper 的测试

i2c0_mapper 的测试主要测试 i2c0 功能是否正常，步骤如下：

- 在 CCS 中用 file－>import->general->exsiting projects into workplace…命令，如图所示导入 Seed_dvs8168_hardware_test\tests\i2c0_mapper 目录；



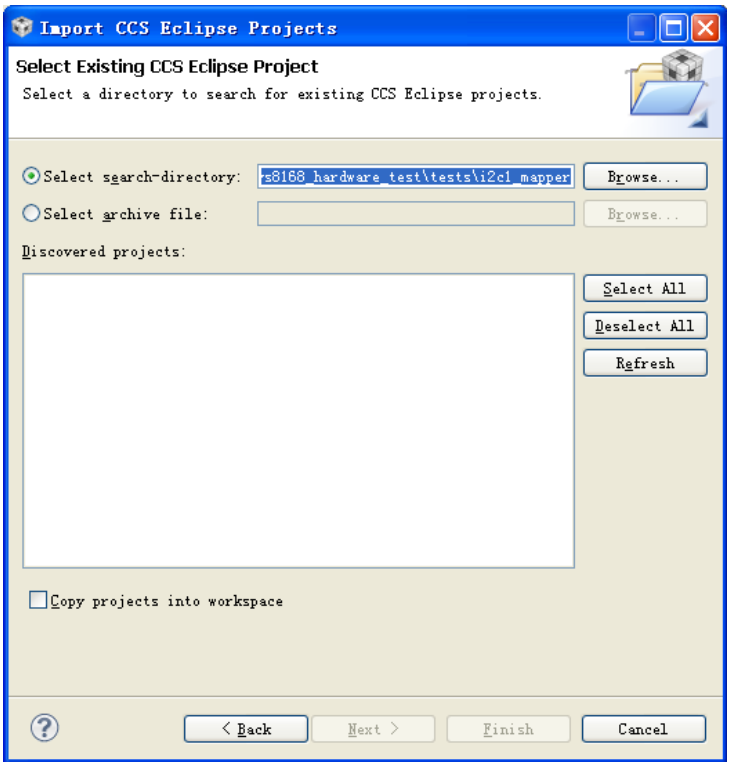
- 选中当前工程，点击右键，在弹出的对话框中选择“built project”；
- 在 CCS 中用了 Run->Load－>Load Program…命令，加载 i2c0_mapper \Debug 目录下的 i2c0_mapper.out 文件；
- 在 CCS 中用了 Run->resume 运行程序或者按 F8（Run）执行程序，若运行结束，显示下图信息，说明测试成功，否则，测试失败。

```
[CortexA8] 01  Testing I2C0 mapper...
[CortexA8] I2C address 18 is valid
[CortexA8] I2C address 50 is valid
[CortexA8] I2C address 57 is valid
[CortexA8] I2C address 6f is valid
[CortexA8]     PASS
[CortexA8]
[CortexA8] ***ALL Tests Passed***
```

7.3.6 i2c1_mapper 的测试

i2c1_mapper 的测试主要测试 i2c1 功能是否正常，步骤如下：

- 在 CCS 中用 file－>import->general->exsiting projects into workplace…命令，如图所示导入 Seed_dvs8168_hardware_test\tests\i2c1_mapper 目录；



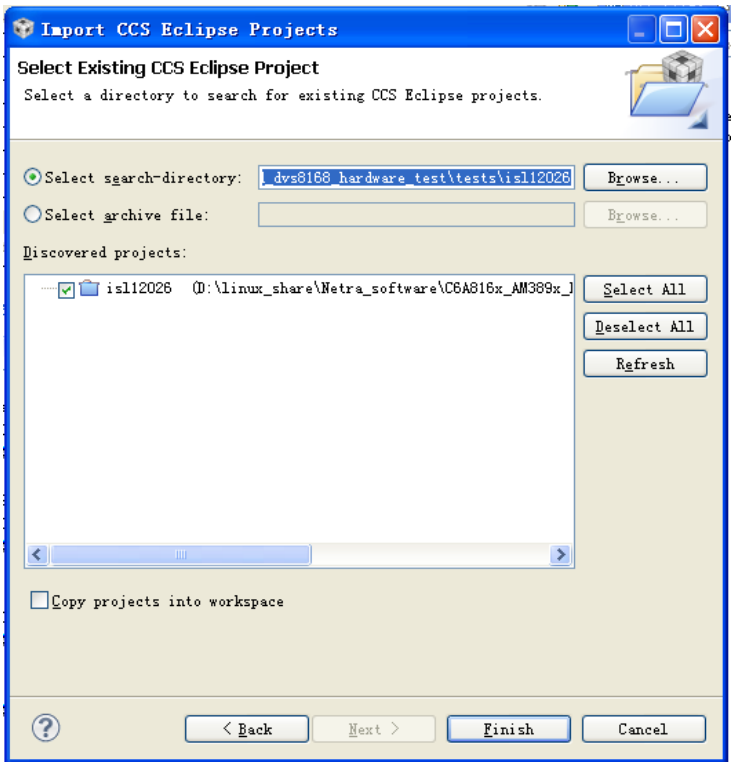
- 选中当前工程，点击右键，在弹出的对话框中选择“built project”；
- 在 CCS 中用了 Run->Load－>Load Program…命令，加载 i2c1_mapper\Debug 目录下的 i2c1_mapper.out 文件；
- 在 CCS 中用了 Run->resume 运行程序或者按 F8（Run）执行程序，若运行结束，显示下图信息，说明测试成功，否则，测试失败。

```
[CortexA8] 01  Testing I2C1 mapper...
[CortexA8] I2C address 39 is valid
[CortexA8] I2C address 3d is valid
[CortexA8] I2C address 40 is valid
[CortexA8] I2C address 58 is valid
[CortexA8] I2C address 59 is valid
[CortexA8] I2C address 5a is valid
[CortexA8] I2C address 5b is valid
[CortexA8]     PASS
[CortexA8]
[CortexA8] ***ALL Tests Passed***
```

7.3.7 isl12026 的测试

isl12026 的测试主要测试 isl12026 功能是否正常，步骤如下：

- 在 CCS 中用 file－>import->general->exsiting projects into workplace…命令，如图所示导入 Seed_dvs8168_hardware_test\tests\ isl12026 目录；



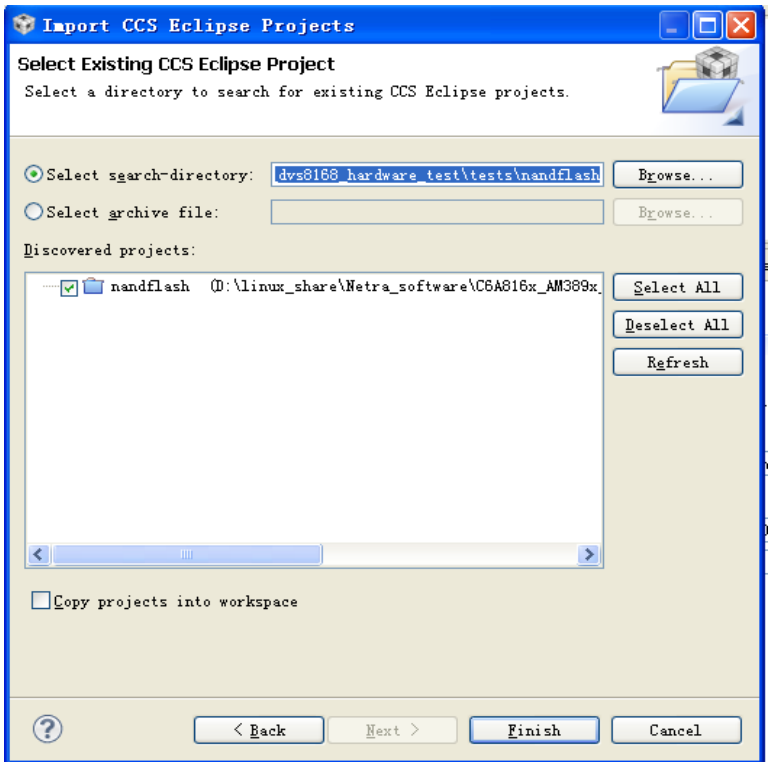
- 选中当前工程，点击右键，在弹出的对话框中选择“built project”；
- 在 CCS 中用了 Run->Load－>Load Program…命令，加载 isl12026\Debug 目录下的 isl12026.out 文件；
- 在 CCS 中用了 Run->resume 运行程序或者按 F8（Run）执行程序，若运行结束，显示下图信息，说明测试成功，否则，测试失败。

```
[CortexA8] 01  Testing RTC...
[CortexA8] The time is:10年12月16日12时5分5秒
[CortexA8] The time is:10年12月16日12时5分6秒
[CortexA8] The time is:10年12月16日12时5分7秒
[CortexA8] The time is:10年12月16日12时5分8秒
[CortexA8] The time is:10年12月16日12时5分9秒
[CortexA8] The time is:10年12月16日12时5分10秒
[CortexA8] The time is:10年12月16日12时5分11秒
[CortexA8] The time is:10年12月16日12时5分12秒
[CortexA8] The time is:10年12月16日12时5分13秒
[CortexA8] The time is:10年12月16日12时5分14秒
[CortexA8] The time is:10年12月16日12时5分15秒
[CortexA8] The time is:10年12月16日12时5分16秒
[CortexA8] The time is:10年12月16日12时5分17秒
[CortexA8] The time is:10年12月16日12时5分18秒
[CortexA8] The time is:10年12月16日12时5分19秒
[CortexA8] The time is:10年12月16日12时5分20秒
[CortexA8] The time is:10年12月16日12时5分21秒
```

7.3.8 nandflash 的测试

nandflash 的测试主要测试 nandflash 功能是否正常，步骤如下：

- 在 CCS 中用 file－>import->general->exsiting projects into workplace…命令，如图所示导入 Seed_dvs8168_hardware_test\tests\ nandflash 目录；



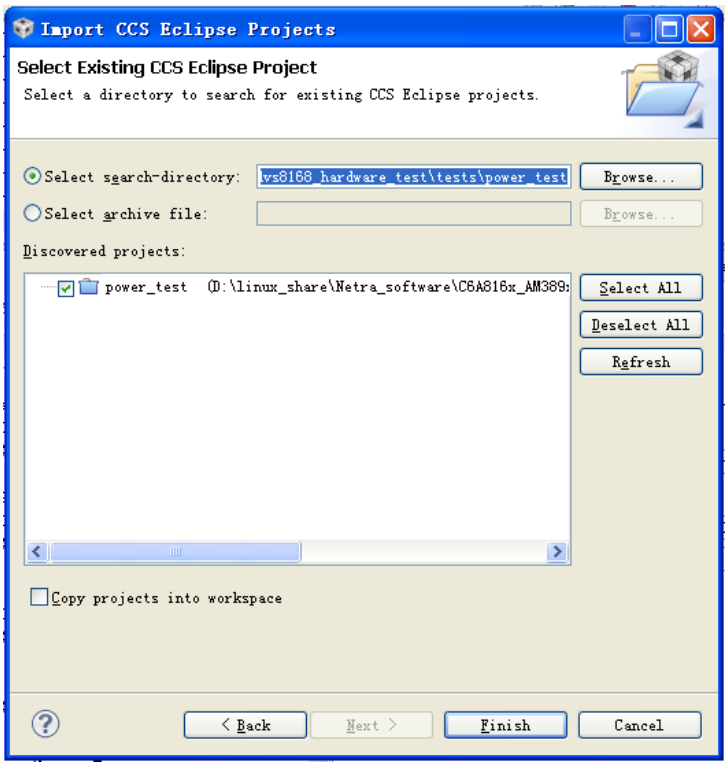
- 选中当前工程，点击右键，在弹出的对话框中选择“built project”；
- 在 CCS 中用了 Run->Load－>Load Program…命令，加载 nandflash\Debug 目录下的 nandflash.out 文件；
- 在 CCS 中用了 Run->resume 运行程序或者按 F8（Run）执行程序，若运行结束，显示下图信息，说明测试成功，否则，测试失败。

```
[CortexA8] 01  Testing NAND Flash...
[CortexA8] NAND MFGID = 2c
[CortexA8] NAND DEVID = ca
[CortexA8] Testing 8 blocks
[CortexA8] Erasing NAND blocks
[CortexA8] --> 0 block erase errors
[CortexA8] Programming NAND pages
[CortexA8] Comparing data
[CortexA8] --> 0 page verify errors
[CortexA8]     PASS
[CortexA8]
[CortexA8] ***ALL Tests Passed***
```

7.3.9 power_test 的测试

power_test 的测试主要测试电源监测芯片是否正常，步骤如下：

- 在 CCS 中用 file－>import->general->exsiting projects into workplace…命令，如图所示导入 Seed_dvs8168_hardware_test\tests\ power_test 目录；



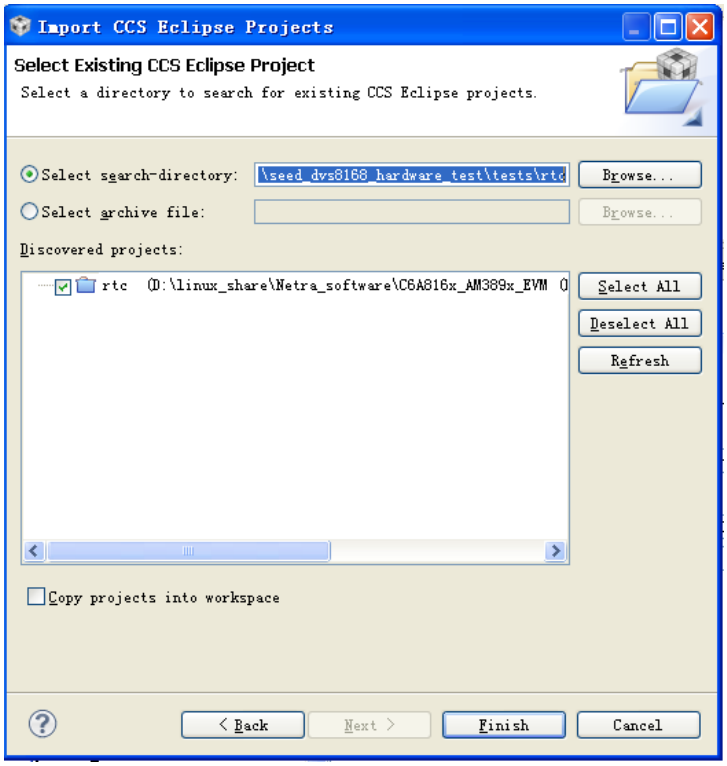
- 选中当前工程，点击右键，在弹出的对话框中选择“built project”；
- 在 CCS 中用了 Run->Load－>Load Program…命令，加载 power_test\Debug 目录下的 power_test.out 文件；
- 在 CCS 中用了 Run->resume 运行程序或者按 F8（Run）执行程序，若运行结束，显示下图信息，否则，测试失败。

```
[CortexA8] 01  Testing POWER...
[CortexA8] Bus voltage    = 1092mV
[CortexA8] Shunt current = 1675.6mA
[CortexA8] Bus voltage    = 1092mV
[CortexA8] Shunt current = 1676.4mA
[CortexA8] Bus voltage    = 1092mV
[CortexA8] Shunt current = 1676.4mA
[CortexA8] Bus voltage    = 1092mV
[CortexA8] Shunt current = 1676.4mA
[CortexA8] Bus voltage    = 1092mV
[CortexA8] Shunt current = 1676.4mA
[CortexA8] Bus voltage    = 1092mV
[CortexA8] Shunt current = 1675.2mA
[CortexA8] Bus voltage    = 1092mV
[CortexA8] Shunt current = 1674.8mA
[CortexA8] Bus voltage    = 1092mV
[CortexA8] Shunt current = 1676.0mA
[CortexA8] Bus voltage    = 1092mV
[CortexA8] Shunt current = 1675.2mA
[CortexA8] Bus voltage    = 1092mV
[CortexA8] Shunt current = 1674.8mA
[CortexA8]      PASS
[CortexA8]
[CortexA8] ***ALL Tests Passed***
```

7.3.10 rtc 的测试

rtc 的测试主要测试片上 rtc 功能是否正常，步骤如下：

- 在 CCS 中用 file－>import->general->exsiting projects into workplace…命令，如图所示导入 Seed_dvs8168_hardware_test\tests\ rtc 目录；

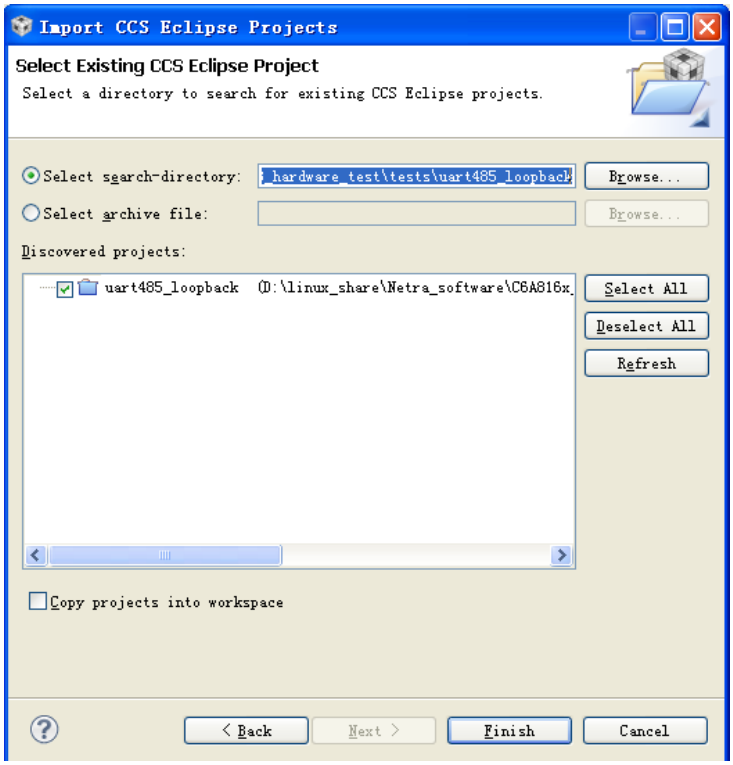


- 选中当前工程，点击右键，在弹出的对话框中选择“built project”；
- 在 CCS 中用了 Run->Load－>Load Program…命令，加载 rtc\Debug 目录下的 rtc.out 文件；
- 在 CCS 中用了 Run->resume 运行程序或者按 F8（Run）执行程序，若运行结束，显示下图信息，否则，测试失败。

```
[CortexA8]
[CortexA8] 01  Testing RTC...
[CortexA8] Starting time is:  07/08/84, 01:30:01, Day of Week = 1
[CortexA8]
[CortexA8] Setting time to:  12/25/08, 10:45:00, Day of Week = 4
[CortexA8]
[CortexA8] Waiting 5 seconds
[CortexA8]
[CortexA8] Final time is:  12/25/08, 10:45:05, Day of Week = 4
[CortexA8]
[CortexA8]     PASS
[CortexA8]
[CortexA8] ***ALL Tests Passed***
```

7.3.11 uart485_loopback 的测试

1. 测试准备工作：
   - 将 RS232/485 串口转换器，一端接 PC 机串口，另一端接板卡 J5 上 31 和 32 口(转换器 T/R+接 J5 上方接口，T/R-接 J5 下方接口)；
   - 在 PC 端打开串口调试器，设置波特率 115200，8 位数据，1 位停止位，无奇偶校验位。
2. uart485_loopback 的测试主要测试 rs485 串口功能是否正常，步骤如下：
   - 在 CCS 中用 file－>import->general->exsiting projects into workplace…命令，如图所示导入 Seed_dvs8168_hardware_test\tests\ uart485_loopback 目录；

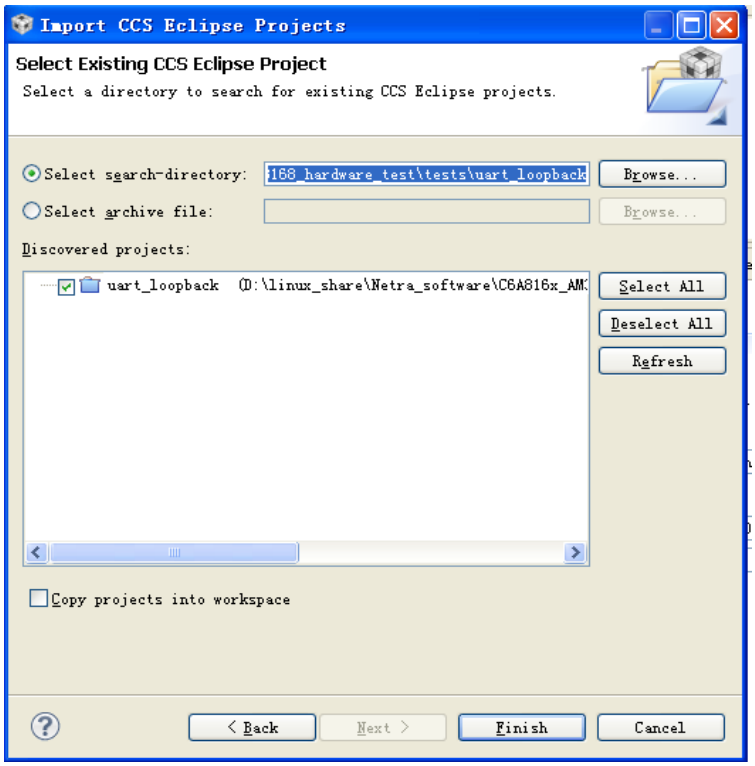


   - 选中当前工程，点击右键，在弹出的对话框中选择“built project”；
   - 在 CCS 中用了 Run->Load－>Load Program…命令，加载 uart485_loopback\Debug 目录下的 uart485_loopback.out 文件；
   - 在 CCS 中用了 Run->resume 运行程序或者按 F8（Run）执行程序，发送下图字符串，在接收区中将接收到相同的字符串。

注：CCS 测试程序，开发板端等待 16 个字符，才把数据发回 pc 端。所以，发送字节要大于等于 16，而板卡只发送 16 字节给 PC 端，所以 PC 端只能收到 16 字节字符。

7.3.12 uart_loopback 的测试

1. 测试准备工作：
   - 使用串口线连接 PC 串口，与板卡 RS232 串口 J41；
   - 在 PC 端打开串口调试器，设置波特率 115200，8 位数据，1 位停止位，无奇偶校验位。
2. 本测试主要测试 r232 串口 uart0 功能是否正常，步骤如下：
   - 在 CCS 中用 file－>import->general->exsiting projects into workplace…命令，如图所示导入 Seed_dvs8168_hardware_test\tests\ uart_loopback 目录；



- 选中当前工程，点击右键，在弹出的对话框中选择“built project”；
- 在 CCS 中用了 Run->Load－>Load Program…命令，加载 uart_loopback \Debug 目录下的 uart_loopback.out 文件；
- 在 CCS 中用了 Run->resume 运行程序或者按 F8（Run）执行程序，发送下图字符串，在接收区中将接收到相同的字符串。

说明：该测试程序，从 pc 发数据，开发板接收到后，再发回给 pc。

第 8 章　Linux 设备驱动测试

8.1　isl12026 RTC 芯片测试说明

8.1.1 测试方法

系统启动后，在板卡的文件系统里，使用命令行测试。
和 RTC 相关的系统命令：
1）date：显示当前的系统时间。
2）date MMDDhhmm[[YY]YY][.ss]：设置当前系统时间。
例如：date 122016382011.30
设置时间为 2011 年 12 月 20 日 16 点 38 分 30 秒
常用的 4 个操作 RTC 芯片的命令：
1） hwclock　　显示硬件时钟；
2） hwclock -r　　显示硬件时钟，　等价于不加参数的 hwclock；
3） hwclock -s　　将“硬件时钟”RTC 的时间写到 Linux“系统”时钟里；
4） hwclock -w　　将“系统”时钟写到“硬件时钟”RTC 里面。

8.1.2 测试

1）查看系统当前时钟：

```
Target # date
Fri Feb  5 03:11:52 UTC 2010 (测试环境不同，可能时间不同)
```

2）读取当前硬件时钟：

```
Target # hwclock
year:2010,mon:2,day:5,hour:3,min:12,sec:28
Fri Feb  5 03:12:28 2010  0.000000 seconds(测试环境不同，可能时间不同)
```

3）改变当前系统时钟到 2012 年 08 月 04 日 16 点 30 分 5 秒：

```
Target # date 080416302012.05
Thu Aug  4 16:30:05 UTC 2012
```

4）将系统时钟写到硬件时钟里：

```
Target # hwclock -w
NEW TIME:year:2012,mon:8,day:4,hour:16,min:30,sec:27
```

5）查看硬件时钟是否写成功：

```
Target # hwclock
year:2012,mon:8,day:4,hour:16,min:30,sec:43
Thu Aug  4 16:30:43 2011  0.000000 seconds
```

6)重新启动，查看硬件时钟，确认 rtc 在工作。

```
Target # hwclock
year:2012,mon:8,day:4,hour:16,min:39,sec:48
Thu Aug  4 16:39:48 2011  0.000000 seconds
```

8.2 GPIO 测试

测试步骤如下：

■ 交叉编译

进入程序包所在目录

```
Host #cd /opt/DVRRDK_02.00.00.23/linux_test/gpio
```

编译读程序

```
Host #arm-none-linux-gnueabi-gcc -o gpio_read api_gpio.c gpio_read.c
```

编译写程序

```
Host #arm-none-linux-gnueabi-gcc -o gpio_write api_gpio.c gpio_write.c
```

■ 拷贝到根文件系统中

```
Host #cp gpio_write gpio_read /opt/DVRRDK_02.00.00.23/target/rfs/opt/test/
```

查看可用 GPIO (原理图 seed-dvs8168v0.0.pdf)

```
可用 gpio 号：6（gp0_6），8（gp0_8）， 18 (gp0_18)， 19(gp0_19)， 28(gp0_28)
        29(gp0_29)， 20 (gp0_20)， 43(gp1_11)， 44(gp1_12)， 48(gp1_16)
        49(gp1_17)， 50(gp1_18)， 51(gp1_19)， 54(gp1_22)， 55(gp1_23)
        56(gp1_24)， 57(gp1_25)， 58(gp1_26).
```

注：29， 28， 20， 43 是输出 gpio，其它为输入 gpio。

■ 运行程序

写 gpio（只指针对输出 GPIO 脚）输入如下命令：

```
Target # ./gpio_write 29

write 1 to GPIO29
write 0 to GPIO29
write 1 to GPIO29
write 0 to GPIO29
write 1 to GPIO29
    …..
```

注：测试程序对 GPIO 29 进行循环写入 0 和 1 状态值。在出现上述显示信息的同时，会听到继电器发出的声音。

读 gpio（只指针对输入 GPIO 脚）输入如下命令：

```
Target # ./gpio_read 6
read 1 from GPIO6
read 1 from GPIO6
read 1 from GPIO6
read 1 from GPIO6
read 1 from GPIO6
```

注：测试程序将循环读入 GPIO 6 的当前状态值。

8.3 uart485 测试

测试步骤如下：

- 交叉编译：

```
Host #cd /opt/DVRRDK_02.00.00.23/linux_test/uart485
Host #arm-none-linux-gnueabi-gcc -o uart_loopback seed_uart_api.c main.c
```

- 拷贝到文件系统

```
Host #cp uart_loopback /opt/DVRRDK_02.00.00.23/target/rfs/opt/test/
```

- 运行程序

运行准备：

- 将 RS232/485 串口转换器，一端接 PC 机串口，另一端接板卡 J5 上 31 和 32 口(转换器 T/R+接 J5 上方接口，T/R-接 J5 下方接口)。
- 在 PC 端打开串口调试器，设置波特率 115200，8 位数据，1 位停止位，无奇偶校验位。

运行开发板端：

```
Target # ./uart_loopback    注：这是该程序堵塞状态，等待 pc 端数据
```

PC 端：打开串口调试助手并按下图发送数据

开发板端程序结果：接收到 pc 端数据，并将接收数据发给 pc，可以从上图观察到串口调试助手接收到同样的数据，证明测试成功。

注：开发板端等待 16 个字符，才把数据发回 pc 端。所以，发送字节要大于等于 16，而板卡只发送 16 字节给 PC 端，所以 PC 端只能收到 16 字节字符。

8.4 tlv320aic3x 音频采集播放测试

测试步骤如下：

- 进入程序包所在目录

Host # cd /opt/DVRRDK_02.00.00.23/linux_test/audio

- alsa 库编译

Host # tar xvfz audio_lib.tar.gz

Host # cd audio/alsa-lib-1.0.9rc4

Host # ./configure --host=arm-none-linux-gnueabi --prefix=/home/seed/Netra/my_lib/alsa_lib --enable-static --enable-share --with-configdir=/home/seed/Netra/my_lib/share

Host # make

Host # make install

注：编译安装结束后，在/home/seed/Netra/my_lib/alsa_lib 路径中生成 alsa 的 lib 库

- 编译测试程序

Host # cd /opt/DVRRDK_02.00.00.23/linux_test/audio

Host # arm-none-linux-gnueabi-gcc -o audio_capture capture.c -I/home/seed/Netra/my_lib/alsa_lib/include -L/home/seed/Netra/my_lib/alsa_lib/lib -lasound

Host # arm-none-linux-gnueabi-gcc -o audio_play play.c -I/home/seed/Netra/my_lib/alsa_lib/include -L/home/seed/Netra/my_lib/alsa_lib/lib -lasound

■ 拷贝到文件系统

```
Host # cp audio_capture audio_play /opt/DVRRDK_02.00.00.23/target/rfs/opt/test/
```

■ 运行程序

运行准备：

- 连接连接音频输入源到板卡“MIC”接口（J2 的下方接口），连接耳机到板卡“line out”（J2 上方接口）接口。

运行如下程序测试，若播放声音正常，则表示音频采集口，输出口正常

```
Target # ./audio_capture > test.pcm          //采集 100 秒保存到 test.pcm 文件中
Target # ./audio_play < test.pcm             //播放文件
```

注：连接有源音箱或电视音频口时，将 SEED-DVS8168 的 C9、C10 取下。

8.5 键盘测试

测试步骤如下：

■ 交叉编译：

```
Host # cd /opt/DVRRDK_02.00.00.23/linux_test/keyboard
Host # arm-none-linux-gnueabi-gcc -o keyboardtest keyboardtest.c serialctrl.c
```

■ 拷贝到文件系统

```
Host #cp keyboardtest /opt/DVRRDK_02.00.00.23/target/rfs/opt/test/
```

■ 运行程序

运行准备：

- 连接键盘到板卡 J43 接口

运行开发板端：

```
Target # cd /opt/test
Target # ./keyboardtest
```

按下键盘的按任意键，在串口终端上将打印出相应的键值，如下图

```
root@seed_dvs8168:/opt/test# ./keyboardtest
keyvalue is 0
keyvalue is 1
keyvalue is 2
keyvalue is 3
keyvalue is 4
keyvalue is 5
keyvalue is 6
```

注：本程序不测试 SEED-DVS8168_PL 的 J20，即面板 6MNO 键。

8.6 U 盘读写测试

测试步骤如下：

■ 系统启动后，插入 U 盘，

■ 等系统识别 U 盘，并执行以下命令

```
Target # cd /opt/test
Target # ./usb_test.sh
```

- 终端中将打印出测试结果。

8.7 SD 卡接口测试

测试步骤如下：

- 系统启动后，插入 SD 卡
- 等系统识别 SD 卡，并执行以下命令

```
Target # cd /opt/test
Target # ./sd_test.sh
```

- 终端中将打印出测试结果。

第 9 章　SEED-SDK 包编译与 demo 演示

SEED-SDK 包集成了 TI 的 RDK DVR 开发包，客户很方便使用该开发包进行 DVR 等产品的开发。

9.1　SEED-SDK 包的编译

SEED-SDK 开发包中已经给用户配置好了各个模块的路径，编译器的路径等，用户编译起来较为方便。

用户可以执行如下命令对整个开发包进行编译：

```
Host # cd /opt/DVRRDK_02.00.00.23/dvr_rdk
Host # make -s sys_all
```

执行上述命令，编译的内容包括

- Uboot
- Linux PSP
- Syslink
- HDVPSS
- DVR RDK

整个包编译的时间较长，若用户需要单独编译某部分程序，可以执行下列命令进行编译

命令	描述
make -s	增量编译 DVR RDK 包
make -s clean	清除 DVR RDK 包编译生成文件
make -s all	清除 DVR RDK 包编译生成文件并重新编译 DVR RDK 包
make -s dvr_rdk_linux	增量编译 DVR RDK 包中 linux 部分代码
make -s dvr_rdk_linux_clean	清除 DVR RDK 包中 linux 部分编译生成的文件
make -s dvr_rdk_linux_all	清除 DVR RDK 包 linux 部分编译生成文件并重新编译 DVR RDK 包 linux 部分
make -s dvr_rdk_bios6	增量编译 DVR RDK 包中 bios 部分代码
make -s dvr_rdk_bios6_clean	清除 DVR RDK 包中 bios 部分编译生成的文件
make -s dvr_rdk_bios6_all	清除 DVR RDK 包 bios 部分编译生成文件并重新编译 DVR RDK 包 bios 部分
make -s sys	增量编译 DVR RDK 包及所有其关联包
make -s sys_clean	清除 DVR RDK 包及所有其关联包编译生成的文件
make -s sys_all	清除 DVR RDK 包及所有其关联包编译生成文件并重新编

	译 DVR RDK 包及所有其关联包
make -s lsp	增量编译 linux 内核包
make -s lsp_clean	清除编译 linux 内核包生成的文件
make -s lsp	清除 linux 内核包编译生成文件并重新编译 linux 内核包
make -s syslink	增量编译 syslink 包（包括 bios 和 linux 部分）
make -s syslink_clean	清除编译 syslink 包生成的文件
make -s syslink_all	清除 syslink 包编译生成文件并重新编译 syslink 包
make -s hdvpss	增量编译 hdvpss 驱动包
make -s hdvpss_clean	清除编译 hdvpss 驱动包生成的文件
make -s hdvpss_all	清除 hdvpss 驱动包编译生成文件并重新 hdvpss 驱动包
make -s uboot	增量编译 uboot 包
make -s uboot_clean	清除编译 uboot 包生成的文件
make -s uboot_all	清除 uboot 包编译生成文件并重新 uboot 包

注：其中 linux 内核包与 uboot 包也可以使用第 3 章和第 4 章中的方法进行编译。

9.2 Demo 程序演示

在 DVR RDK 包中提供了如下 demo：

- McFW(Multi Channel FrameWork) API demo，该 demo 使用 McFW API 接口，关于这些 API 的具体说明可以参考 SDK 包中文档 DVR_RDK_McFW_UserGuide.pdf；
- QT GUI DVR demo，该 demo 界面使用 QT 开发，实现了 16 路 D1 格式 DVR 的基本功能。

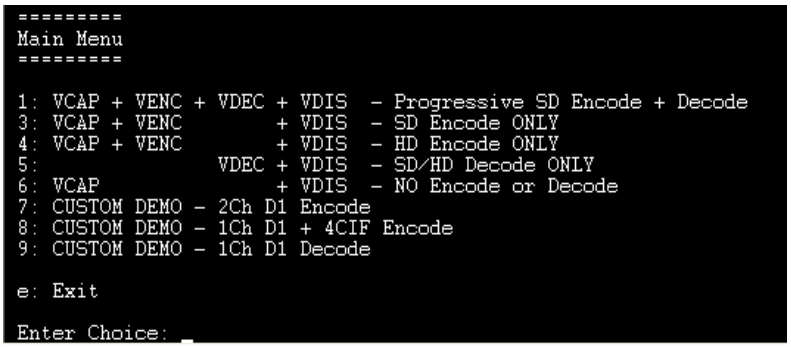
9.2.1 McFW(Multi Channel FrameWork) API demo

该 demo 运行步骤如下：

- 连接 16 路摄像头到 SEED-DVS8168VACON 板上部分 BNC 接口（注意，程序根据第 1 路连接的视频来判断连接的是 PAL 或者 NTSC 摄像头，所以第 1 路摄像头必须连接），将板卡 HDMI 接口 P1、DVI 接口 P2、分量接口 J4 和复合视频接口 J3 连接到显示器；
- 连接音频输入源到 SEED-DVS8168VACON 板下半部分 BNC 接口，连接音箱或者耳麦到 J2 口的上方接口；
- 执行下面命令运行该 demo 程序：

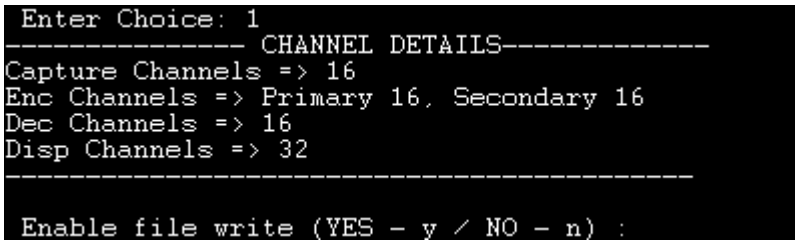
```
Target #cd /opt/dvr_rdk/ti816x
Target #./run_mcfw_demo.sh
```

运行程序后出现如下菜单：

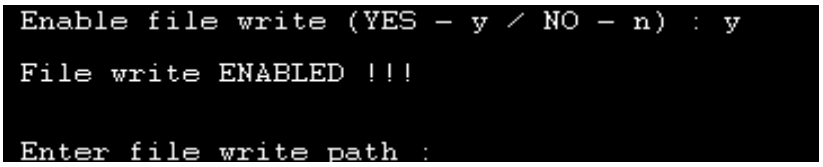


主菜单中列出了可选择运行的 8 个 demo 程序，8 个 demo 程序的流程可以参考 SDK 包中 dvr_rdk/mcfw/src_linux/mcfw_api/usecases/目录下的代码程序，代码开头注释出了流程框图，本文档中不做具体介绍。下面以第 1 个 demo 为例说明程序运行过程。

- 选择“1”，并回车，将出现如下菜单，提示您是否保存编码后的码流文件；

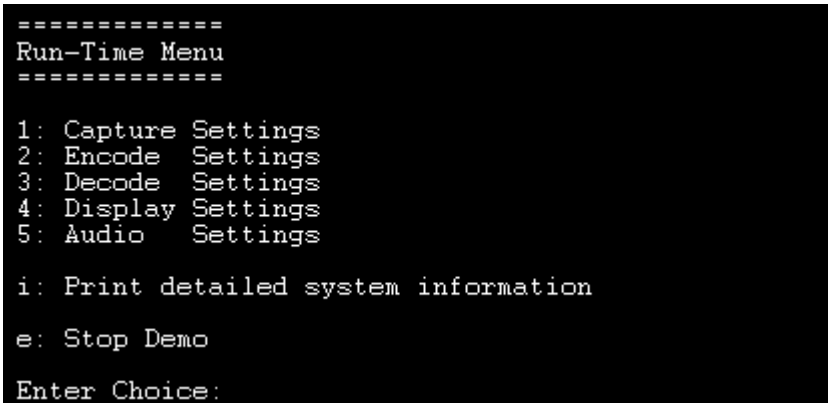


- 选择“y”则出现下面菜单，提示你选择要保存的路径；

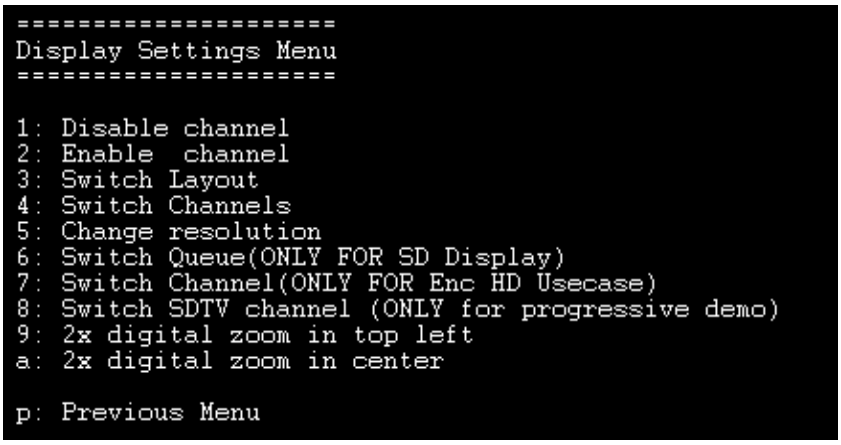


本例程中默认保存第 1 路的码流，若要保存多路码流，可以修改文件 demos/mcfw_api_demos/mcfw_demo/demo_vcap_venc_vdec_vdis.h 中的宏定义 MCFW_IPC_BITS_FWRITE_ENABLE_BITMASK_DEFAULT

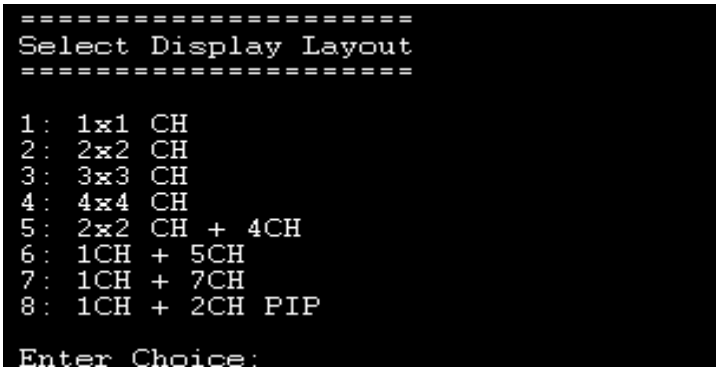
- 输入要保存的目录路径（可以在 NFS 或者硬盘上保存），如“/data”，几秒钟后程序将启动完成，启动后出现如下菜单（如显示实时信息，请按回车），用户可以选择对应的选项，动态的对采集、编码、解码、显示和音频进行操作，以下以显示设置为例说明设置过程；



- 选择“4”可以显示模式配置菜单；



- 选择菜单中对应的选项，可以进行显示通道的使能与关闭，显示风格的切换，显示分辨率的切换等操作，如选择“3”，则可进入显示风格配置界面，如下图：



- 选择各个选项，显示界面上将呈现不同风格的显示，如选择“4”，则选择显示风格为16 分格，显示器上可以同时显示 16 路图像；
- 运行程序后，在 DVI 接口将显示经过 h264 算法编解码处理后图像，HDMI 接口、分量接口和复合视频接口将显示未经编解码处理过的图像。

注：关于本程序的详细运行步骤，可以参考 SDK 包中文档 DM81xx_DVR_RDK_DemoGuide.pdf

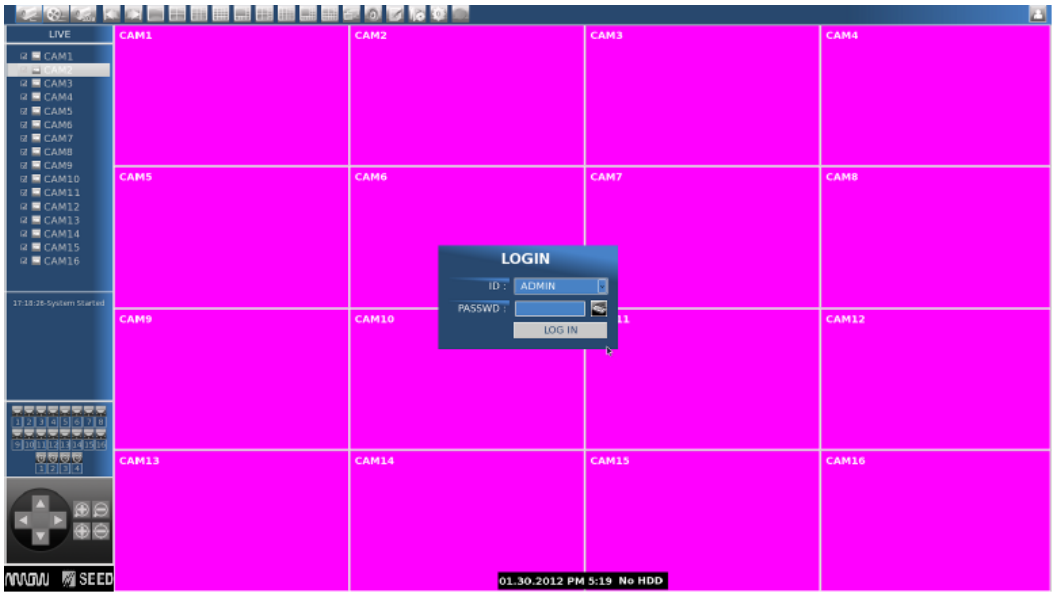
9.2.2 QT GUI DVR demo

该 demo 运行步骤如下：

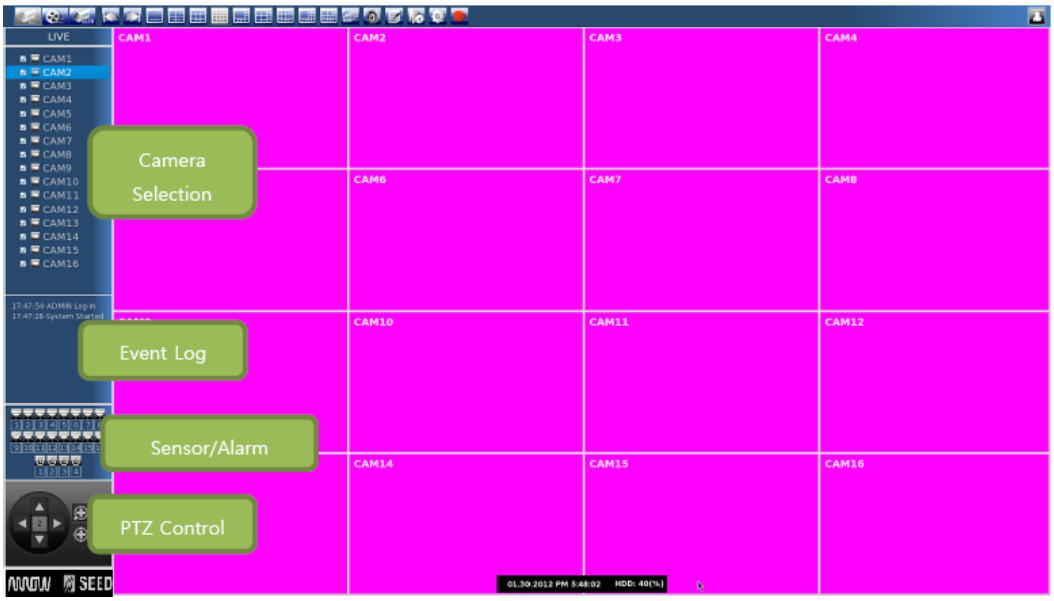
- 连接 16 路摄像头到 SEED-DVS8168VACON 板上部分 BNC 接口（注意，程序根据第 1 路连接的视频来判断连接的是 PAL 或者 NTSC 摄像头，所以第 1 路摄像头必须连接），将板卡 HDMI 接口 P1、DVI 接口 P2、分量接口 J4 和复合视频接口 J3 连接到显示器；
- 将鼠标连接到 USB 口 J6；
- 连接网口 J7 到局域网，让板卡与控制 PC 机处于同一网段；
- 执行下面命令运行该 demo 程序：

```
Target #cd /opt/dvr_rdk/ti816x/
Target #./run_gui.sh
```

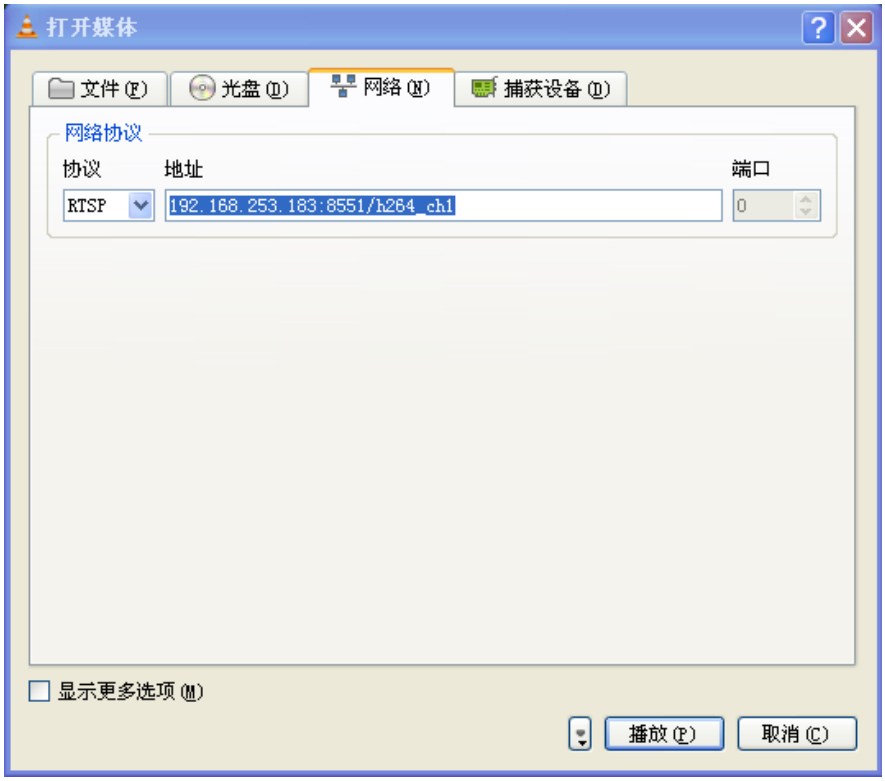
运行程序后在 HDMI 和分量视频接口上将显示如下界面，复合视频输出口输出采集的 1~16 路视频中的 1 路；



- 用鼠标点击登陆对话框中的“LOG IN”按钮，登入控制界面；



- 点击屏幕左侧控制栏按钮，可以对摄像头进行选择，并对云台进行控制等，点击屏幕上方按钮，可以调节图像显示方式，启动/停止录像等。若板卡已经连接了摄像头，屏幕上将显示采集的图像。另外，在“playback”模式下，若播放硬盘中存储的码流，播放的视频图像将从 DVI 接口输出，播放的音频将从 HDMI 接口输出；
  注：要启动录像必须先点击屏幕上方的“setup”按钮，在弹出的对话框中选择“STORAGE”，然后格式化一下硬盘，本界面格式化工具所支持的硬盘必须未经分区过
- 在 PC 上打开 vlc 进行如下操作



注：上述对话框中 192.168.253.183 上为 SEED-DVS8168 板卡的 IP 地址，该地址需根据板卡实际使用的 IP 地址做相应修改，8551 为第 1 通道 rtsp 服务器对应端口号，h264_ch1 为第 1 通道 rtsp 服务器对应名称，本程序中支持 1~8 通道 rtsp 码流传输，对应的端口号分别为 8551~8558，对应名称分别为 h264_ch1~h264_ch8。

- 点击播放，就会看到编码后通过网络发送过来的 CIF 格式子码流，并听到对应的通道采集进来的声音。

注：本界面的详细操作步骤，可以参考 SDK 包中文档 DM8168_DVR_RDK_GUI_Guide.pdf