IAC-335X-Kit Linux User Manual

Version No. 1.0

2013.06

QIYANG INTELLIGENT TECHNOLOGY CO., LTD
Copyright Reserved

Preface

Welcome to use IAC-335X-Kit from Hangzhou Qiyang Intelligent Technology Co., Ltd.

•This manual introduces the compilation of Linux system, download of the kernel, file system, main board test, etc.

•Hardware interface can refer to *IAC-335X-Kit Hardware Manual*.

•Please read this manual carefully before using.

Company Profile:

Hangzhou Qiyang Technology Co., Ltd. is located at the bank of the beautiful West Lake. It is a high and new technology enterprise which is specializing in R&D, manufacture and sell embedded computer main board with high performance, low power consumption, low cost, small volume, and provides embedded hardware solutions.

We Offer:

◆ Research & develop, manufacture and sell embedded module products which have independent intellectual property rights, and cooperate with TI, ATMEL, Cirrus Logic, Freescale, and other famous processor manufacturers. It has launched a series of hardware products, such as ARM development board, ARM core module, ARM industrial board, sound/video decoding transmission platform, supporting tools and software resources which support user for their next embedded design.

◆ We give full play to the technical accumulation in ARM platform and Windows CE, Linux, Android operating system for many users providing custom service (OEM/ODM), to realize embedded products into the market stably, reliably and quickly.

Tel: +86 571 87858811, +86 571 87858822
Fax: +86 571 87858822
Technology Support E-mail: support@qiyangtech.com
Website: http://www.qiytech.com
Address: 5F, Building 3A, No.8 Xiyuanyi Road, West Lake Science Park, Hangzhou, China
Post code: 310030

Catalogue

Ⅰ. IAC-335X-Kit Resource Introduction 7
1.1 Hardware Resource 7
1.2 Software Resource 7
Ⅱ. Build Embedded Linux Development Environment 7
2.1 Build Virtual Machine Development Environment 7
2.2 Install Cross-Compiler 7
2.2.1 Extract Cross-Compiler Tool Chains 8
2.2.2 Add Environment Variables 8
2.2.3 Make the Environment Variables Effectively 8
2.2.4 Check the Cross-Compiler Version Information. 8
Ⅲ. Compile u-boot and Linux Kernel 9
3.1 Compile u-boot 9
3.2 Compile Linux Kernel 10
3.3 Create ubi file system 10
3.3.1 Create ubifs.img 11
3.3.2 Set up ubinize.cfg 11
3.3.3Create ubi.img 11
Ⅳ. Start and Burn Linux System 12
4.1 Startup Modes 12
4.2. Start from SD Card 13
4.2.1 Create SD Card Boot Disk 13
4.2.2 Start from SD Card 18
4.3 Start from NAND Flash 19
4.3.1 Burning through Serial Network 19
4.3.1.1 Burn u-boot 19
4.3.1.2 Burn the kernel 24
4.3.1.3 Burn the File System 24
4.3.1.4 Restart Mainboard 24
4.3.2 Burn through SD Card 25
4.3.2.1 Preparation 25
4.3.2.2 Burning Procedure 25
4.3.2.3 Start Mainboard 27
4.4 Start Uboot from spi Flash 27
4.4.1 Illustration 27
4.4.2 Preparation 28
4.4.3 Burning Procedure 28
4.4.4 Start mainboard 29
4.4.5 Notice 30
Ⅴ. Develop Application Program 30

5.1 Hello World 30
5.2 Cross Compiling 30
5.3 Running Application Program 31
5.4 Autostart Application Program 32
Ⅵ. Mainboard Test 33
6.1 Preparation 33
6.1.1 Configuration 33
6.1.2 Mainboard Requirement 33
6.1.3 Start Mainboard 33
6.2 Mainboard Test 34
6.2.1 RTC Test 34
6.2.2 Buzzer Test 35
6.2.3 Serial Port Test 35
6.2.4 CAN Bus Test 36
6.2.5 RS485 Test 38
6.2.6 Ethernet Test 38
6.2.7 Login Mainboard by Telnet 40
6.2.8 USB Hard Disk Test 40
6.2.9 SD Card Test 41
6.2.10 PWM Test 42
6.2.11 Display mp3 42
6.2.12 Touch Panel Test 42
6.2.13 Qt Program Test 42
Ⅶ. Migrating QT4.8.2 43
7.1 Preparation 43
7.2 Add the Library of Supporting Touch Panel 43
7.3 Install QT Library 44
7.4 Compile QT Program 47
7.5 Migrate Library to Mainboard 47
7.6 am335x_qt Main Interface Functions 49
7.6.1 Serial Port Communication 50
7.6.2 Audio Test 51
7.6.3 Setting 52
7.6.3.1 Time Setting 53
7.6.3.2 Language Settings 53
7.6.4 Screen Calibration 54
7.6.5 Screen Font 54
Ⅷ. Migrate and use ntp 55
8.1 Illustration 55
8.2 Preparation 55
8.3 Migrating Procedure 55

Sales E-mail:trade@qiyangtech.com sales@qiyangtech.com
Website:http://www.qiytech.com
©2012 Qiyangtech Copyright

8.4 Ntp Clients.. 56
8.5 ntp server... 57

Ⅰ. IAC-335X-Kit Resource Introduction

1.1 Hardware Resource

IAC-335X-Kit mainboard hardware resource, please refer to *IAC-335X-Kit Hardware Manual.*

1.2 Software Resource

Software	version	Features
Bootloader	u-boot-2011.09	DDR2 UBI file system burning TFTP Start from the UART/SD/NAND Flash
Linux kernel	Linux 3.2	Drive:UART,RS485,RTC-ds1338,NANDFlash, Audio, USB HOST,OTG, Touch Panel, LCD,SD Card, Watch Dog etc. Standalone 2-ch CAN Bus drive Dual standalone Ethernet Interface drive TCP/IP protocol UBI file system
file system	None	UBI format file QT 4.8.2 DHCP Shell Automatically mount USB devices and SD card NFS mount network equipment Play mp3 and wav format audio files Hardware interface test

Ⅱ. Build Embedded Linux Development Environment

2.1 Build Virtual Machine Development Environment

IAC-335X-Kit embeds development environment to Linux virtual machine, please refer to: *Ubuntu Installation for Virtual Machine Manual. pdf*

2.2 Install Cross-Compiler

Build [ubuntu] ,install [nfs] and [samba] server according to the manual, compile Linux kernel and cross-compiler, the cross-compiler tool chains are in the CD.

2.2.1 Extract Cross-Compiler Tool Chains

Copy the package of the cross-compiler tool chains to [~/opt]directory:

```
lisl@ubuntu:~$ cp arm-arago-linux-gnueabi.tar.gz ~/opt
lisl@ubuntu:~$ cd ~/opt
lisl@ubuntu:~$ tar -xvzf arm-arago-linux-gnueabi.tar.gz
```

So in [~/opt] directory will generate [arm-arago-linux-gnueabi] folder, this is the cross-complier.

2.2.2 Add Environment Variables

```
lisl@ubuntu:~$ cd ~
lisl@ubuntu:~$ vi .bashrc
```

Edit [.bashrc] file, add path to the file

```
export PATH=$PATH:~/opt/arm-arago-linux-gnueabi/bin
```

2.2.3 Make the Environment Variables Effectively

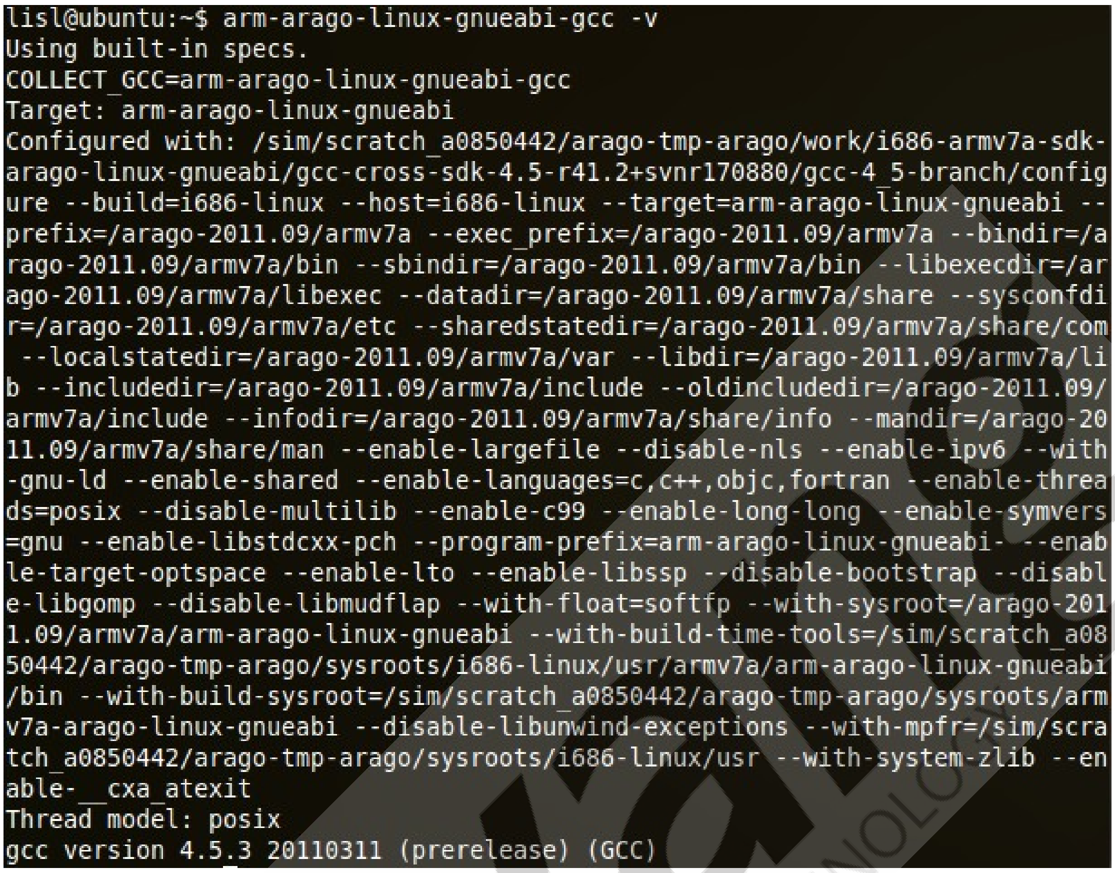
```
lisl@ubuntu:~$ source ~/.bashrc
```

2.2.4 Check the cross-compiler version information.

Check the cross-compiler whether it is installed correctly by typing in the following command in terminal

```
lisl@ubuntu:~$ arm-arago-linux-gnueabi-gcc -v
```

As shown:


```
lisl@ubuntu:~$ arm-arago-linux-gnueabi-gcc -v
Using built-in specs.
COLLECT_GCC=arm-arago-linux-gnueabi-gcc
Target: arm-arago-linux-gnueabi
Configured with: /sim/scratch_a0850442/arago-tmp-arago/work/i686-armv7a-sdk-
arago-linux-gnueabi/gcc-cross-sdk-4.5-r41.2+svnr170880/gcc-4_5-branch/config
ure --build=i686-linux --host=i686-linux --target=arm-arago-linux-gnueabi --
prefix=/arago-2011.09/armv7a --exec_prefix=/arago-2011.09/armv7a --bindir=/a
rago-2011.09/armv7a/bin --sbindir=/arago-2011.09/armv7a/bin --libexecdir=/ar
ago-2011.09/armv7a/libexec --datadir=/arago-2011.09/armv7a/share --sysconfdi
r=/arago-2011.09/armv7a/etc --sharedstatedir=/arago-2011.09/armv7a/share/com
 --localstatedir=/arago-2011.09/armv7a/var --libdir=/arago-2011.09/armv7a/li
b --includedir=/arago-2011.09/armv7a/include --oldincludedir=/arago-2011.09/
armv7a/include --infodir=/arago-2011.09/armv7a/share/info --mandir=/arago-20
11.09/armv7a/share/man --enable-largefile --disable-nls --enable-ipv6 --with
-gnu-ld --enable-shared --enable-languages=c,c++,objc,fortran --enable-threa
ds=posix --disable-multilib --enable-c99 --enable-long-long --enable-symvers
=gnu --enable-libstdcxx-pch --program-prefix=arm-arago-linux-gnueabi- --enab
le-target-optspace --enable-lto --enable-libssp --disable-bootstrap --disabl
e-libgomp --disable-libmudflap --with-float=softfp --with-sysroot=/arago-201
1.09/armv7a/arm-arago-linux-gnueabi --with-build-time-tools=/sim/scratch_a08
50442/arago-tmp-arago/sysroots/i686-linux/usr/armv7a/arm-arago-linux-gnueabi
/bin --with-build-sysroot=/sim/scratch_a0850442/arago-tmp-arago/sysroots/arm
v7a-arago-linux-gnueabi --disable-libunwind-exceptions --with-mpfr=/sim/scra
tch_a0850442/arago-tmp-arago/sysroots/i686-linux/usr --with-system-zlib --en
able-__cxa_atexit
Thread model: posix
gcc version 4.5.3 20110311 (prerelease) (GCC)
```

picture 1

If it shows the above cross-compiler information, it means that the cross-compiler is installed correctly, gcc version is 4.5.3,the next you can use the cross-compiler to compile u-boot, kernel and application program.

Ⅲ. Compile u-boot and Linux Kernel

3.1 Compile u-boot

Copy the configured u-boot source code file package in CD to a directory, user can compile it.

Extract u-boot source code package in CD.

```
$ tar -xvzf   u-boot-2011.09-psp04.06.00.08-IAC335X-20130122.tar.gz
```

Generate a new directory [u-boot-2011.09-psp04.06.00.08] after extracting.

```
$ cd   u-boot-2011.09-psp04.06.00.08
$ ./build_uboot.sh
```

After compiling correctly, generate [am335x] directory in current directory, this three image files (spl/u-boot-spl.bin,MLO,u-boot.img)are required to burnt into mainboard.

3.2 Compile Linux Kernel

Copy the configured u-boot source code file package in CD to a directory, user can configure the kernel.

1. Extract the kernel source code package

```
$ tar -xvzf   linux-3.2.0-04.06.00.08-IAC335X-20130122.tar.gz
```

Generate a new directory [linux-3.2-04.06.00.08] after extracting

2. Configure kernel

```
$ cd linux-3.2-04.06.00.08
$ make menuconfig
```

Popping up a kernel configuration menu, user can adjust the kernel function-option, save & exit.

3. Compile and generate kernel image

```
$ make uImage
```

Observe recompiled kernel image [uImage] in the [arch/arm/boot/] directory after compiling.

3.3 Create ubi file system

After debugging the application program, add it to the file system, then you can burn it and file system into NAND Flash. Next is the introduction about how to add the application program to [ubi] file system image.

Add the compiled application program to the file system source code, then create [ubi] file system image base on the new file system source code.

3.3.1 Create ubifs.img

Enter file system directory, [mkfs.ubifs].

```
$ sudo mkfs.ubifs -r rootfs/ -F -o ubifs.img -m 2048 -e 126976 -c 1580
```

3.3.2 Set up ubinize.cfg

Compile the following content

```
[ubifs]
   mode=ubi
   image=ubifs.img
   vol_id=0
   vol_size=192MiB
   vol_type=dynamic
   vol_name=rootfs
   vol_flags=autoresize
```

3.3.3 Create ubi.img

```
$ sudo ubinize -o ubi.img -m 2048 -p 128KiB -s 512 -O 2048 ubinize.cfg
```

Generate file system image [ubi.img].

Remark:[mkfs.ubifs] and [ubinize] are two commands in [mtd-utils] tool, if without the two commands, need to compile the newer version [mtd-utils],copy the command in CD to [/usr/bin] directory in ubuntu.

More information, please refer to:

http://processors.wiki.ti.com/index.php/UBIFS_Support#Flashing_UBIFS_image_to_a_NAND_partition

Ⅳ. Start and Burn Linux System

4.1 Startup Modes

IAC-335X-Kit support four startup modes: UART,SD Card ,NAND Flash, SPI Flash. Need to weld or remove resistance R52—R59，R99—R106 to set, specific setting method is as follows:

Weld R99 R100 R101 R102 R103 R104 R105 R106 by 100K resistance.

1. UART Startup

Startup mode: 1000 0000 0000

Means: Remove R52

Weld R53 R54 R55 R56 R57 R58 R59 by 10K resistance.

2. NAND Flash Startup:

Startup Mode: 0100 1000 0000

Means: Remove R53,R56

Weld R52 R54 R55 R57 R58 R59 by 10K resistance

3. MMC0/SD Card Startup:

Startup Mode: 1110 1000 0000

Means: Remove R52 R53 R54 R56

Weld R55 R57 R58 R59 by 10K resistance

4. SPI Flash Startup:

Startup Mode: 0110 1000 0000

Means: Remove R53 R54 R56.

Weld R52 R55 R57 R58 R59 by 10K resistance

4.2. Start from SD Card

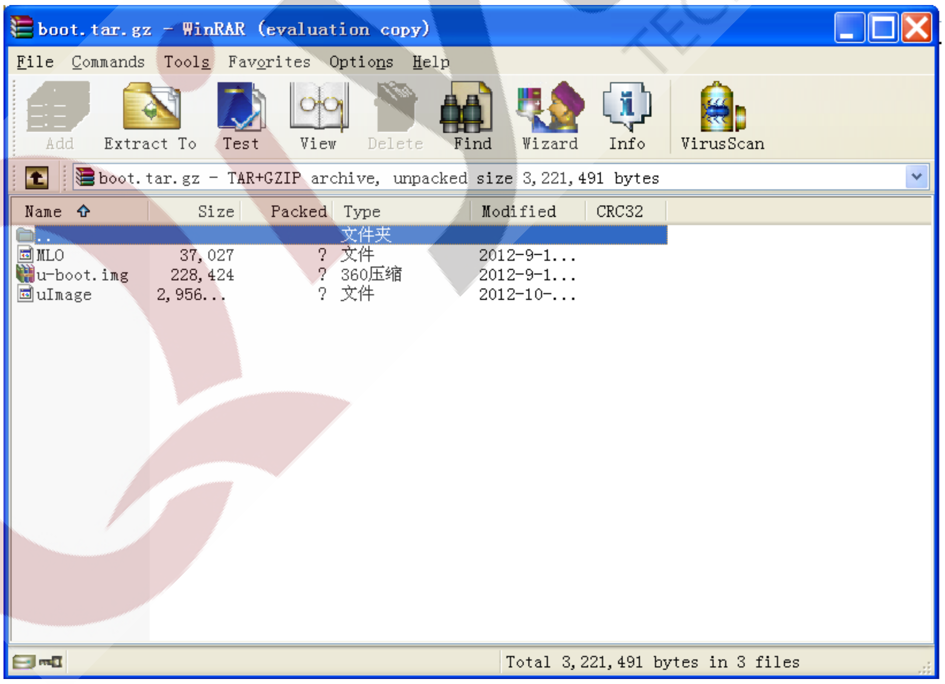
IAC-335X-Kit support SD card startup, prepare a SD card which has been cured system image, how to cure the system image to SD card, as shown:

4.2.1 Create SD Card Boot Disk

1. Start Ubuntu OS in PC.

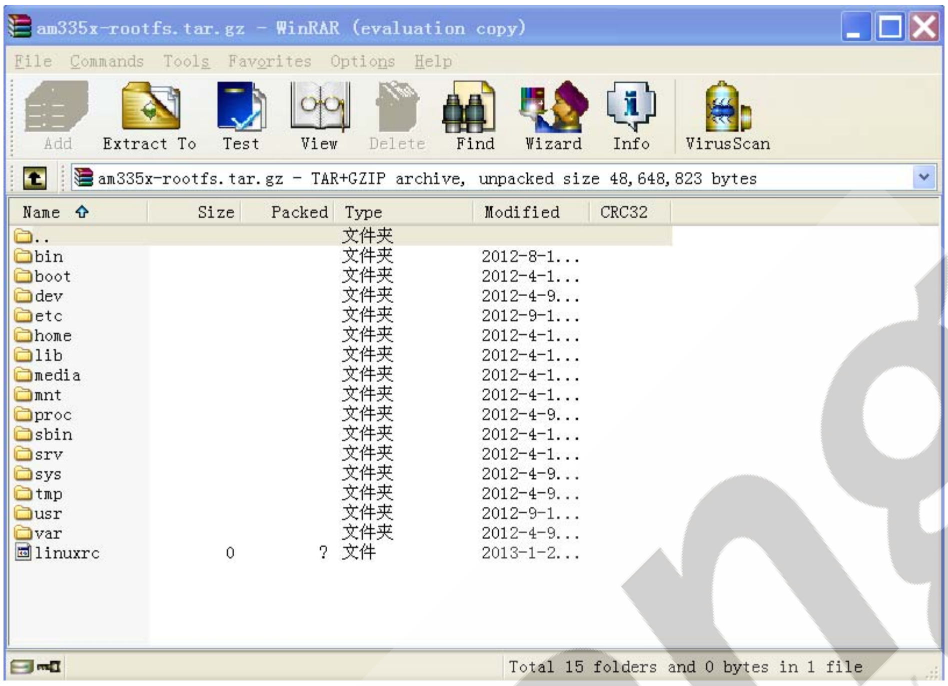
2. Copy [boot.tar.gz],[am335x-rootfs.tar.gz] and [creat-sdcard.sh] to directory in Ubuntu OS, now we put it into [/home/lisl/am335x/sdcard] directory.[boot.tar.gz] includes [MLO],[u-boot.img] and [uImage].

[am335x-rootfs.tar.gz] is the package of file system source code.


boot.tar.gz - WinRAR (evaluation copy)
File Commands Tools Favorites Options Help
Add Extract To Test View Delete Find Wizard Info VirusScan

```
boot.tar.gz - TAR+GZIP archive, unpacked size 3,221,491 bytes
Name        Size    Packed  Type      Modified      CRC32
..                          文件夹
MLO         37,027       ?  文件      2012-9-1...
u-boot.img  228,424      ?  360压缩   2012-9-1...
uImage      2,956...     ?  文件      2012-10-...
Total 3,221,491 bytes in 3 files
```

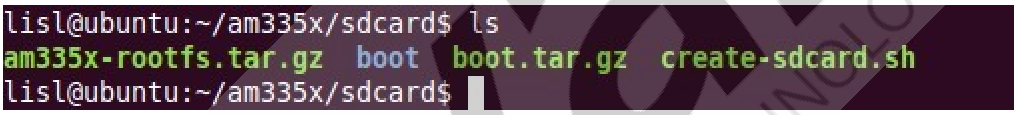
picture 2


am335x-rootfs.tar.gz - WinRAR (evaluation copy)
File Commands Tools Favorites Options Help
Add
Extract To
Test
View
Delete
Find
Wizard
Info
VirusScan
am335x-rootfs.tar.gz - TAR+GZIP archive, unpacked size 48,648,823 bytes

Name	Size	Packed	Type	Modified	CRC32
..			文件夹		
bin			文件夹	2012-8-1...	
boot			文件夹	2012-4-1...	
dev			文件夹	2012-4-9...	
etc			文件夹	2012-9-1...	
home			文件夹	2012-4-1...	
lib			文件夹	2012-4-1...	
media			文件夹	2012-4-1...	
mnt			文件夹	2012-4-1...	
proc			文件夹	2012-4-9...	
sbin			文件夹	2012-4-1...	
srv			文件夹	2012-4-1...	
sys			文件夹	2012-4-9...	
tmp			文件夹	2012-4-9...	
usr			文件夹	2012-9-1...	
var			文件夹	2012-4-9...	
linuxrc	0	?	文件	2013-1-2...	

Total 15 folders and 0 bytes in 1 file

picture 3

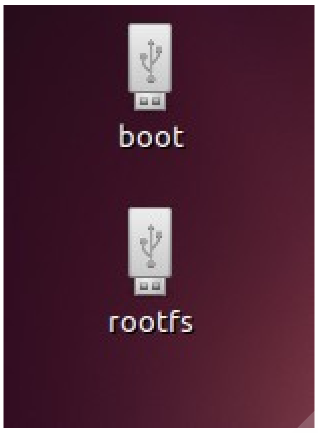


```
lisl@ubuntu:~/am335x/sdcard$ ls
am335x-rootfs.tar.gz  boot  boot.tar.gz  create-sdcard.sh
lisl@ubuntu:~/am335x/sdcard$ 
```

picture 4

3. Prepare a SD card and a card reader, a notebook with a SD card slot use card reader.

4. Connect the SD card to the computer in Ubuntu OS through the card reader, Ubuntu will mount SD card automatically, the two partitions are in the SD card, as shown:



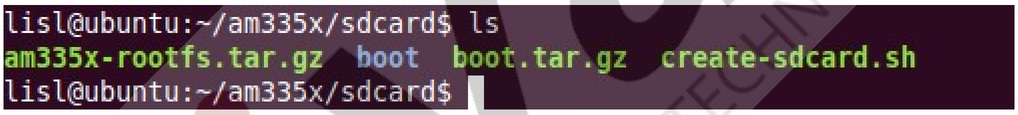
picture 5

5. Umount SD card first.

```
$ umount /media/boot
$ umount /media/rootfs
```

6. Enter the directory.

```
$ cd ~/am335x/sdcard
$ ls
```



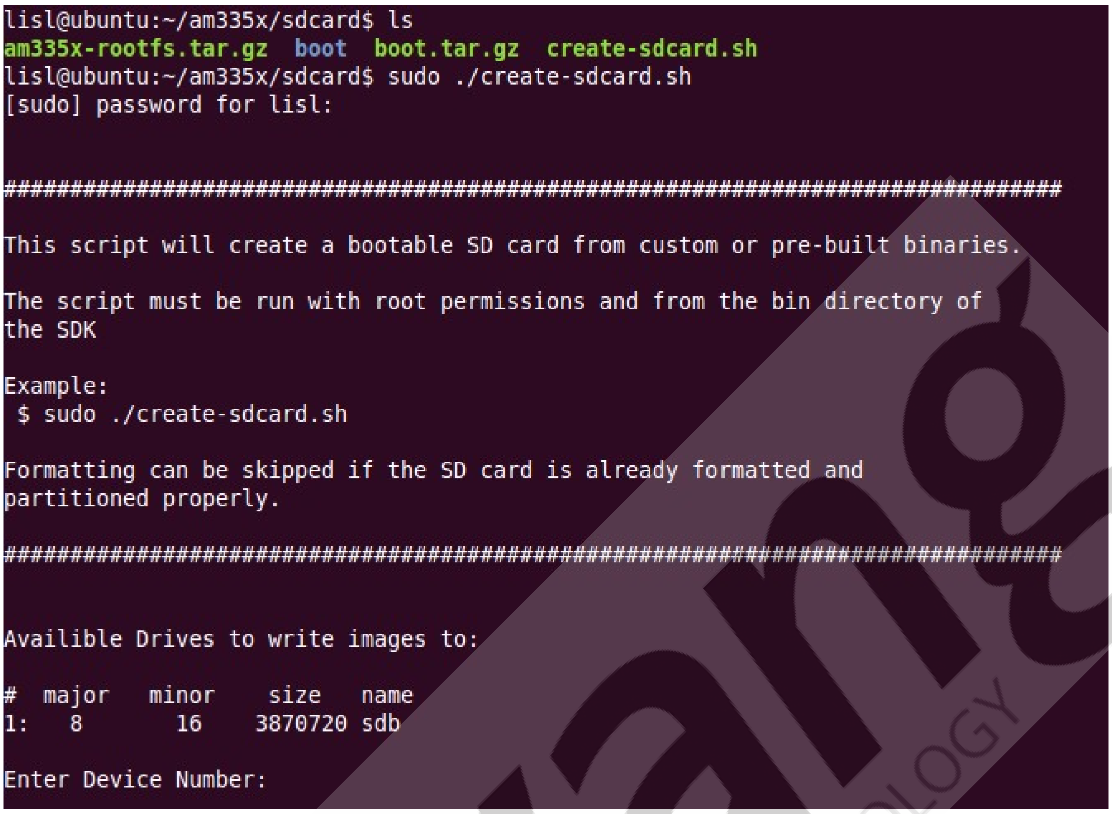
```
lisl@ubuntu:~/am335x/sdcard$ ls
am335x-rootfs.tar.gz  boot  boot.tar.gz  create-sdcard.sh
lisl@ubuntu:~/am335x/sdcard$
```

picture 6

7. Run the following command.

```
$ sudo ./create-sdcard.sh
```

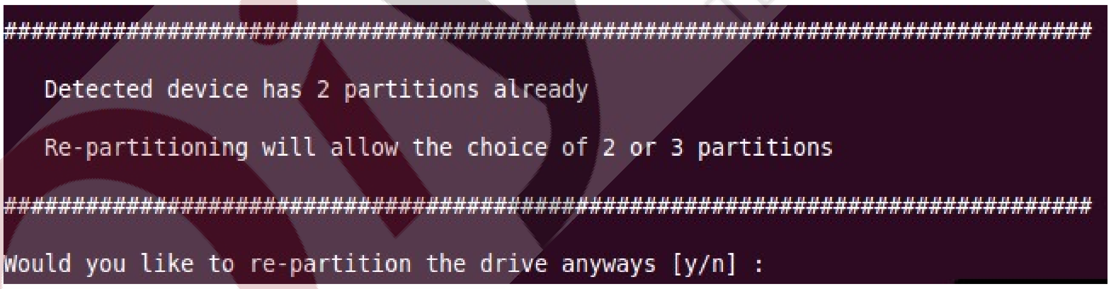
As shown:



```
lisl@ubuntu:~/am335x/sdcard$ ls
am335x-rootfs.tar.gz  boot  boot.tar.gz  create-sdcard.sh
lisl@ubuntu:~/am335x/sdcard$ sudo ./create-sdcard.sh
[sudo] password for lisl:


################################################################################

This script will create a bootable SD card from custom or pre-built binaries.

The script must be run with root permissions and from the bin directory of
the SDK

Example:
 $ sudo ./create-sdcard.sh

Formatting can be skipped if the SD card is already formatted and
partitioned properly.

################################################################################


Availible Drives to write images to:

#  major   minor    size   name
1:   8       16    3870720 sdb

Enter Device Number:
```

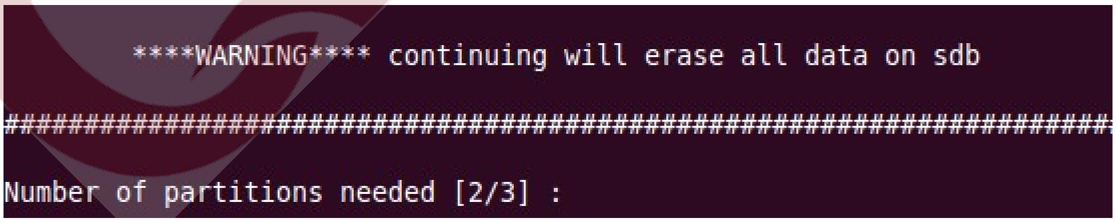
picture 7

Input “1” and press enter



```
################################################################################

   Detected device has 2 partitions already

   Re-partitioning will allow the choice of 2 or 3 partitions

################################################################################

Would you like to re-partition the drive anyways [y/n] :
```

Input “Y” and press enter

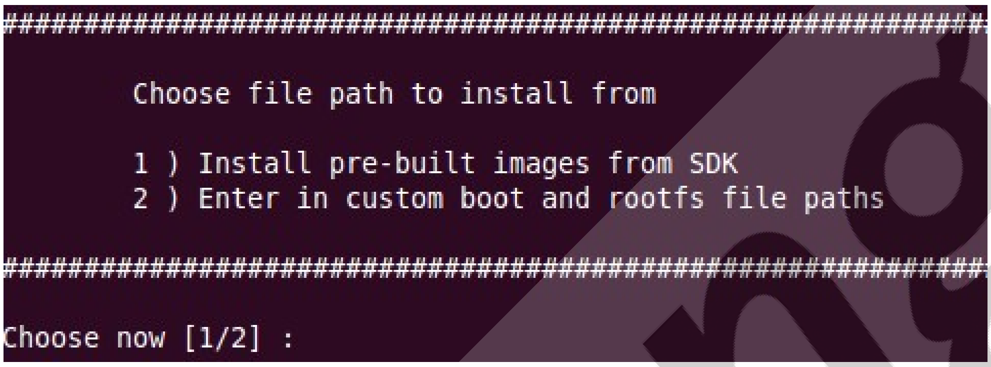


```
        ****WARNING**** continuing will erase all data on sdb

################################################################################

Number of partitions needed [2/3] :
```

Input “2” and press enter, SD card is divided into two partitions: boot and rootfs; Input “3” and press enter, SD card will be divided into three partitions:

Hint: Would you like to continue? [y/n]

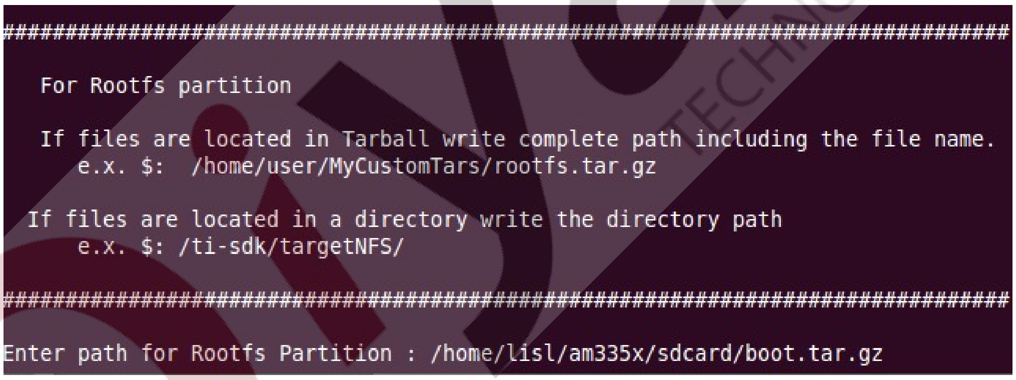
Input “Y” and press enter.



```
################################################################

        Choose file path to install from

        1 ) Install pre-built images from SDK
        2 ) Enter in custom boot and rootfs file paths

################################################################

Choose now [1/2] :
```

picture 8

Input “2” and press enter, then input the path of the boot partition [/home/lisl/am335x/sdcard/boot.tar.gz].

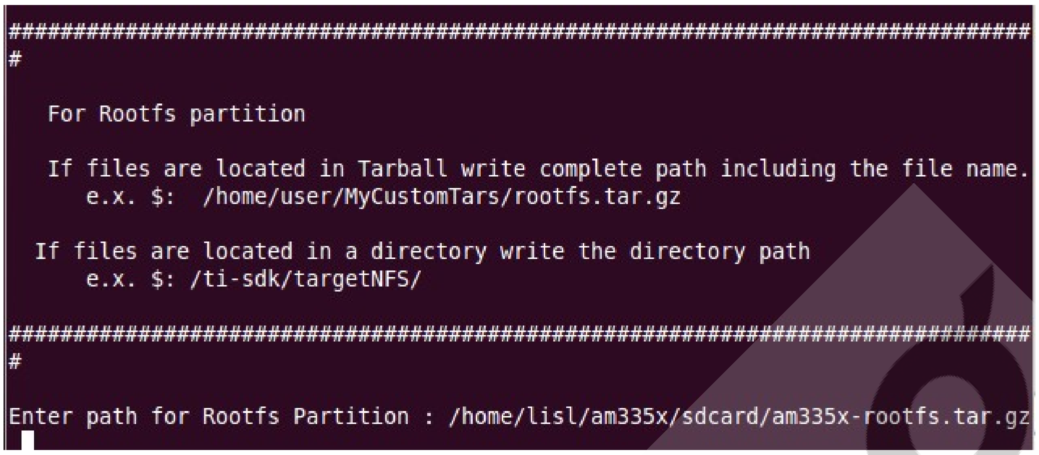


```
################################################################################

  For Rootfs partition

  If files are located in Tarball write complete path including the file name.
     e.x. $:  /home/user/MyCustomTars/rootfs.tar.gz

 If files are located in a directory write the directory path
     e.x. $: /ti-sdk/targetNFS/

################################################################################

Enter path for Rootfs Partition : /home/lisl/am335x/sdcard/boot.tar.gz
```

picture 9

Press Enter, [boot.tar.gz] will be burnt into boot partition in SD card, then input the path of the [rootfs] partition:

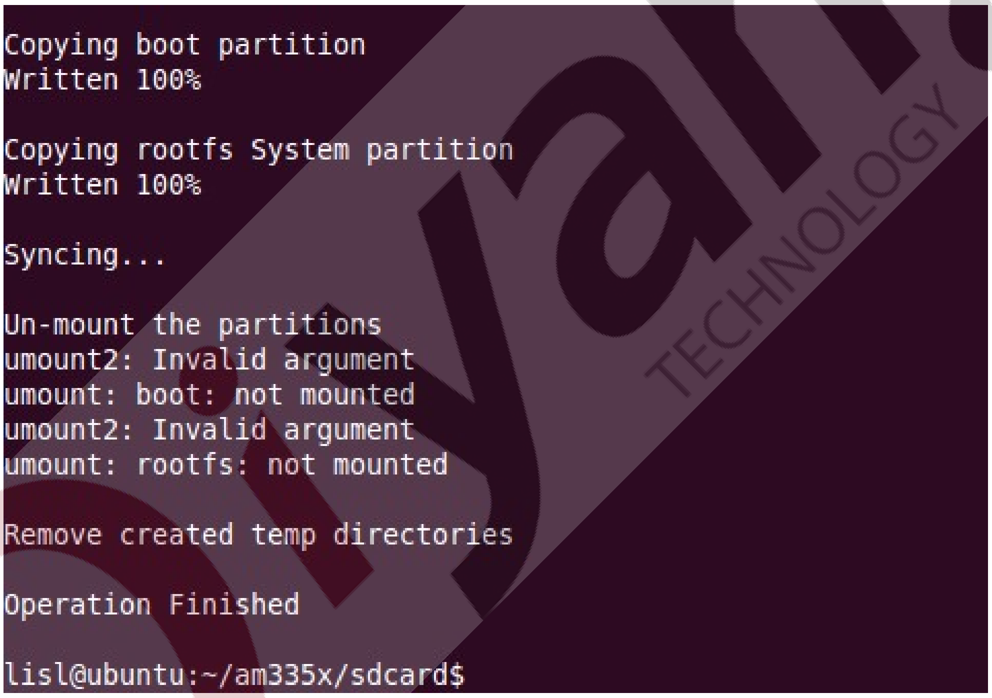
[/home/lisl/am335x/sdcard/amm335x-rootfs.tar.gz].



```
################################################################################
#

   For Rootfs partition

   If files are located in Tarball write complete path including the file name.
      e.x. $:  /home/user/MyCustomTars/rootfs.tar.gz

  If files are located in a directory write the directory path
      e.x. $: /ti-sdk/targetNFS/

################################################################################
#

Enter path for Rootfs Partition : /home/lisl/am335x/sdcard/am335x-rootfs.tar.gz
```

picture 10

Press Enter, file system will be burnt into rootfs partition in SD card.



```
Copying boot partition
Written 100%

Copying rootfs System partition
Written 100%

Syncing...

Un-mount the partitions
umount2: Invalid argument
umount: boot: not mounted
umount2: Invalid argument
umount: rootfs: not mounted

Remove created temp directories

Operation Finished

lisl@ubuntu:~/am335x/sdcard$
```

picture 11

At this time, the SD card boot disk in Linux OS is created.

4.2.2 Start from SD card

1. Set startup dial switch to SD card startup, specific setting method can refer to user manual.

2. Insert SD card into the mainboard.

3. Power on to the mainboard, system will start from SD card.

4.3 Start from NAND Flash

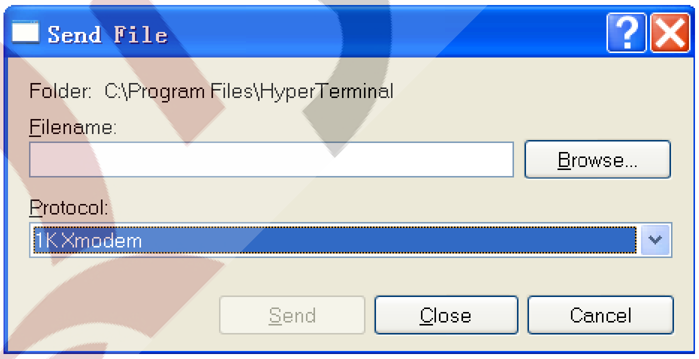
4.3.1 Burning through Serial Network

4.3.1.1 Burn u-boot

1. Set startup resistance in mainboard to UART startup mode, specific setting method, please refer to user manual.

2. Connect serial port with PC, open hyper terminal in Windows, select bund rate [115200], stop bit [1], data bit [8], parity bit [none]and data flow control [none].

3. Power on to the mainboard, when hint “CCCCC” in hyper terminal, then select:[send]→[send file]→[1K Xmodem]， as shown:


Send File

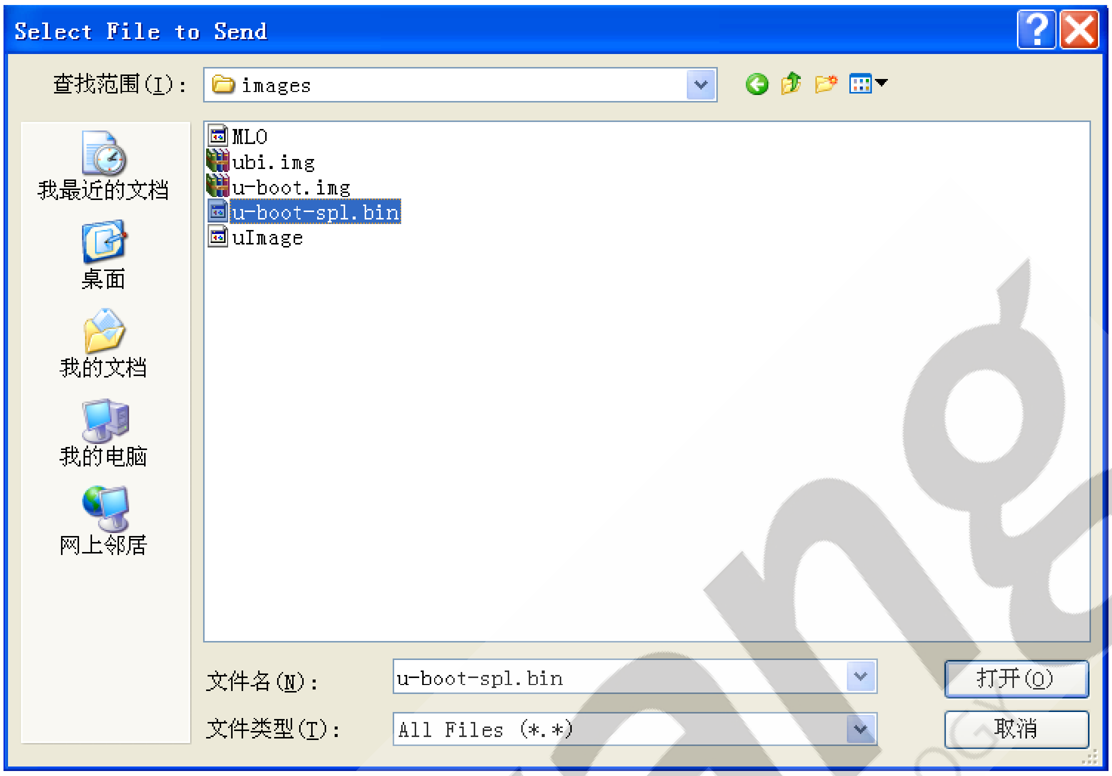
Folder: C:\Program Files\HyperTerminal

Filename:

Protocol: 1K Xmodem

picture 12

4. Click [browse], select [u-boot-spl.bin] file.



picture 13

Then, select [send]; transfer the [u-boot-spl.bin] file.

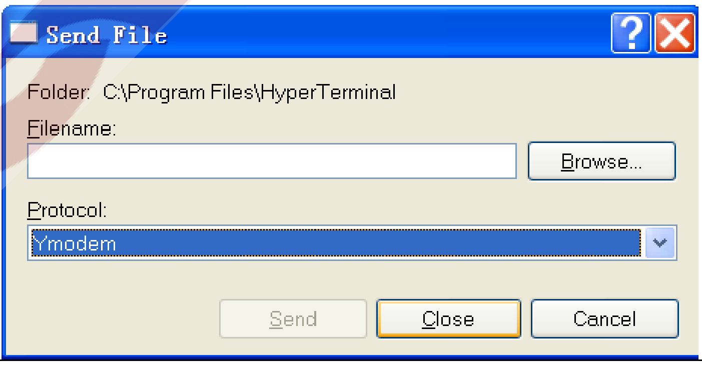
5. Transmission finished, CPU will boot [u-boot.bin] automatically, as shown:

```
U-Boot SPL 2011.09 (Jun 08 2012 - 16:35:14)
Texas Instruments Revision detection unimplemented
CC
```

picture 14

6. Hint “CC“,transfer the [u-boot.img] file.

Select: [send]→[send file]→[Ymodem],as shown:



Folder: C:\Program Files\HyperTerminal

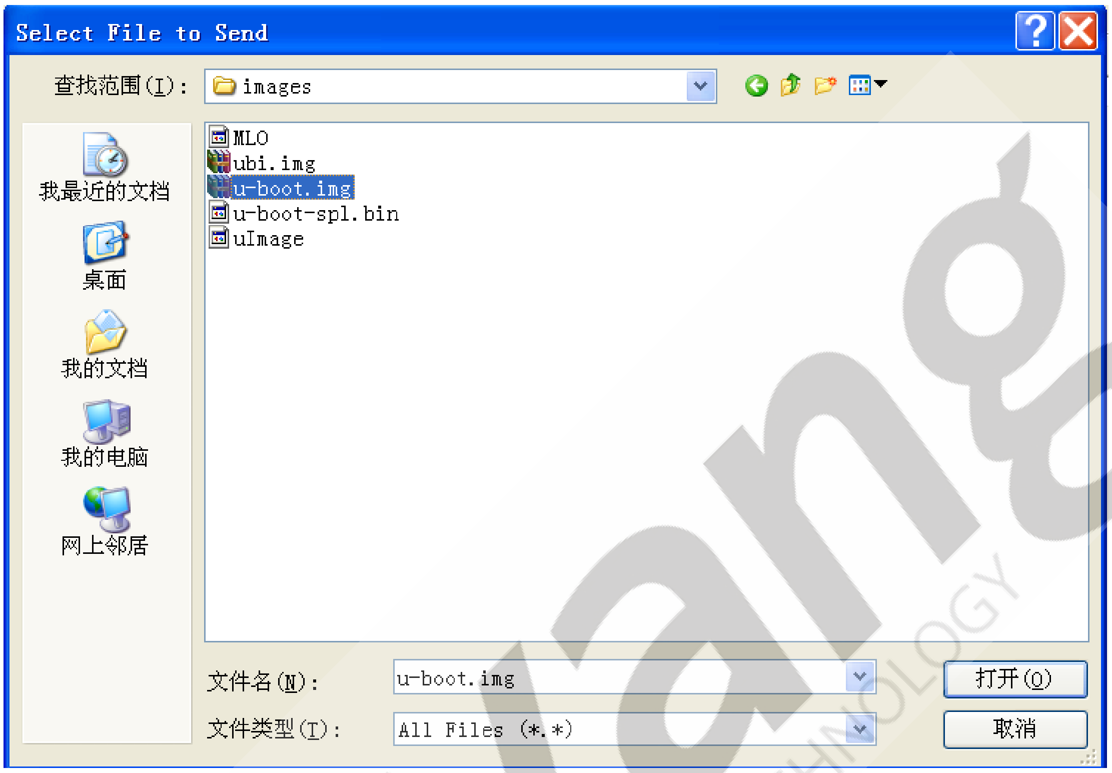
Filename:

Protocol: Ymodem

©2012 Qiyangtech Copyright

picture 15

7. Click [browse], select [u-boot.img] file



picture 16

Then, select [send]; transfer the [u-boot.img] file.

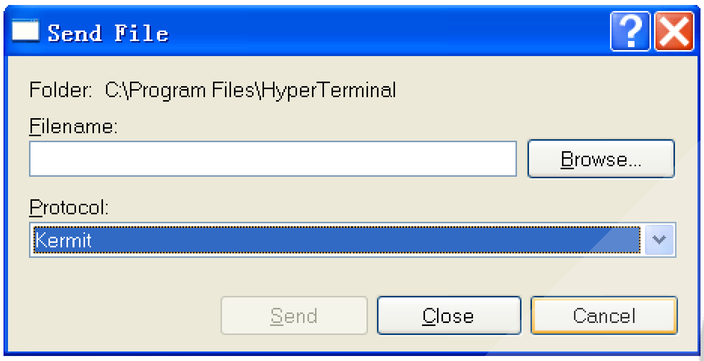
8. Transmission finished, SPL will be boot into [u-boot] in SDRAM automatically.

9. Click enter key in 3 seconds, hint “U-Boot#”, u-boot has been burnt into DDR, but still haven't been burnt into NAND Flash. Now we burn image into NAND Flash by UART.

10. Input data according to the hint of “U-Boot#”:

```
U-Boot# mw.b 0x82000000 0xFF 0x20000
U-Boot# loadb 0x82000000
```

Open the transport protocol, select: Select:[send]→[send file]→[Kermit],as shown:



Send File

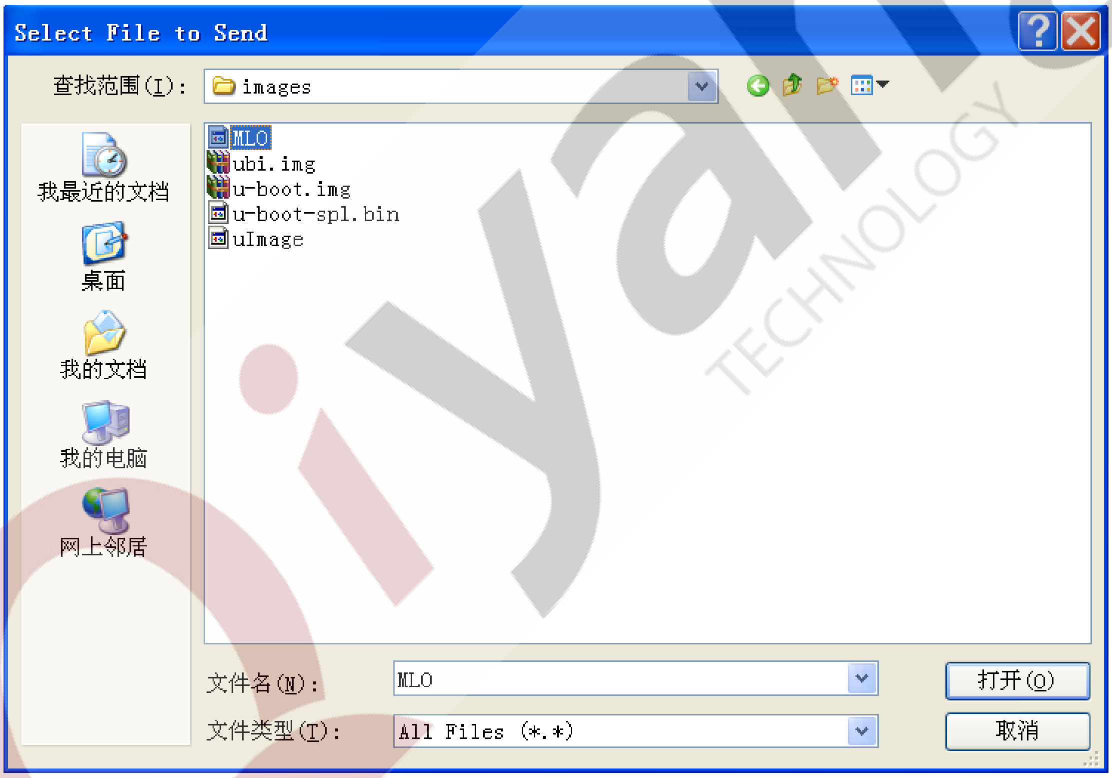
Folder: C:\Program Files\HyperTerminal

Filename:

Protocol: Kermit

picture 17

11. Click [browse], select [MLO] file.



Select File to Send

查找范围(I): images

MLO
ubi.img
u-boot.img
u-boot-spl.bin
uImage

文件名(N): MLO

文件类型(T): All Files (*.*)

picture 18

Then, select [send]; transfer the [MLO] file.

12. Transmission finished ,input data according to the hint of “U-Boot#”:

```
U-Boot# nand erase 0x0 0x20000
U-Boot# nandecc hw 2
```

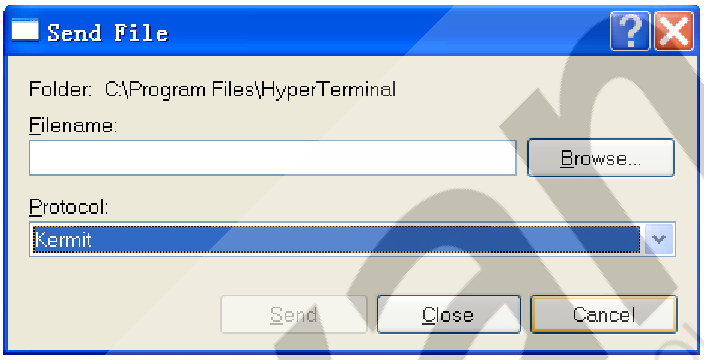
```
U-Boot# nand write 0x82000000 0x0 0x20000
```

Now, MLO has been burnt into NAND Flash, next burn the [u-boot.img] file into NAND Flash.

13. Input data according to the hint of "U-Boot#":

```
U-Boot# mw.b 0x82000000 0xFF 0x40000
U-Boot# loadb 0x82000000
```

Open the transport protocol, Select: [send]→[send file]→[Kermit], as shown:



picture 19

14. Click [browse],select [u-boot.img]file, then select [send],transfer [u-boot.img] file

15. Transmission finished, input data according to the hint of "U-Boot#":

```
U-Boot# nand erase 0x80000 0x40000
U-Boot# nandecc hw 2
U-Boot# nand write 0x82000000 0x80000 0x40000
```

Now the level-3 bootloader [u-boot.img] has been burnt into NAND Flash.

Set startup resistance in mainboard to "NAND Flash startup", power on again, bootloader start from NAND Flash.

Next, Burn kernel and file system through TFTP, please make sure that the network connectivity is ok.

4.3.1.2 Burn the kernel

1. Open the [tftpd32.exe] software in windows OS of CD, as shown:

picture 20

2. Copy the [uImage] kernel image to the [tftpd2] directory.

3. Set the IP address of mainboard and server according to the hint of "U - Boot#";

Set mainboard address:

```
U-Boot# setenv ipaddr 192.168.1.112
```

Set the Windows's IP address to server IP address:

```
U-Boot# setenv serverip 192.168.1.75
```

4. Input data according to the hint of "U - Boot#":

```
U-Boot# run loadkernel
```

Wait a moment; the kernel will be burnt into NAND Flash.

4.3.1.3 Burn the File System

1. Copy the file system image [ubi.img] in [ubi] file to the [tftpd2] directory

2. Input data according to the hint of "U - Boot#":

```
U-Boot# run loadfs
```

Wait a moment; the file system will be burnt into NAND Flash.

4.3.1.4 Restart Mainboard

Input data according to the hint of "U-Boot#":

```
U-Boot# reset
```

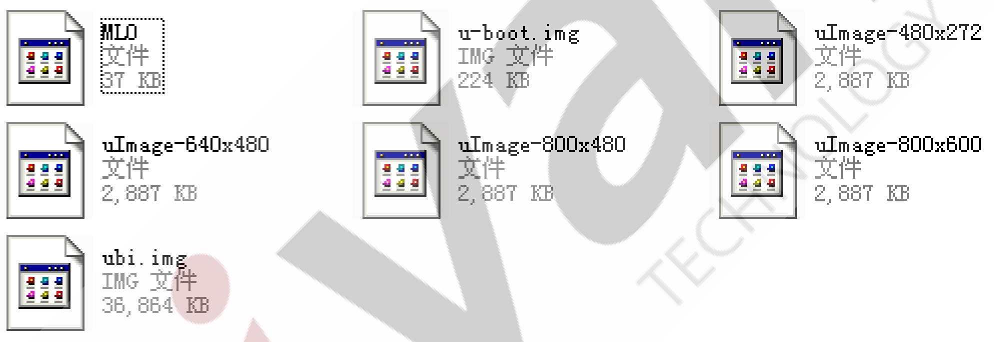
If can enter the system terminal, means the Linux system burnt correctly.

4.3.2 Burn through SD Card

4.3.2.1 Preparation

1. Refer to *section 3.2.1*; create a SD card boot disk

2. After creating the boot disk, use the first partition of SD card, copy four kinds of resolution for kernel and file system to SD card. Include the following files:



picture 21

4.3.2.2 Burning Procedure

1. Confirm that the SD card includes 4 image files(MLO, u-boot.img, uImage, ubi.img),insert SD card to mainboard card slot.

2. Set startup resistance to SD card startup mode 1110 1000 0000,

Weld R99,R100,R101,R102,R103,R104,R105,R106 by 100K resistance.

Then remove R52 R53 R54 R56

©2012 Qiyangtech Copyright

Weld R55 R57 R58 R59 by 10K resistance

3. Power on, press any keys in 3 seconds, appear the hint of [U-Boot#], burn the image into NAND Flash through SD card.

4. Burn MLO

Input data according to the hint of [U-Boot#]:

```
U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 MLO
U-Boot# nandecc hw 2
U-Boot# nand erase 0x0 0x20000
U-Boot# nand write.i 0x82000000 0x0 0x20000
```

If there are no errors, [MLO] has been burnt into NAND Flash.

5. Burn u-boot.img

Input data according to the hint of [U-Boot#]:

```
U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 u-boot.img
U-Boot# nandecc hw 2
U-Boot# nand erase 0x80000 0x40000
U-Boot# nand write.i 0x82000000 0x80000 0x40000
```

If there are no errors, [u-boot.img] has been burnt into NAND Flash.

6. Burn the uImage

Input data according to the hint of [U-Boot#]:

```
U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 uImage-800x600
```

Remark: Change uImage-800*600 to kernel file name which need burning.

```
U-Boot# nandecc hw 2
U-Boot# nand erase 0x280000 0x500000
U-Boot# nand write.i 0x82000000 0x280000 0x500000
```

If there are no errors, [uImage] has been burnt into NAND Flash.

7. Burn file system [ubi.img]

Input following data according to the hint of [U-Boot#]

```
U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 ubi.img
U-Boot# nandecc hw 2
U-Boot# nand erase 0x780000 0xF880000
U-Boot# nand write 0x82000000 0x780000 ${filesize}
```

If there are no errors, [uImage] has been burnt into NAND Flash.

4.3.2.3 Start mainboard

1. Close power supply, take out the SD card.

2. Set startup resistance in mainboard to NAND Flash startup mode 0100 1000 0000：

Remove R53,R56

Weld R52 R54 R55 R57 R58 R59 by 10K resistance.

Power on to the mainboard, again, start bootloader from NAND Flash, if can enter the system terminal, means that the Linux system has been burnt correctly.

4.4 Start uboot from spi flash

4.4.1 Illustration

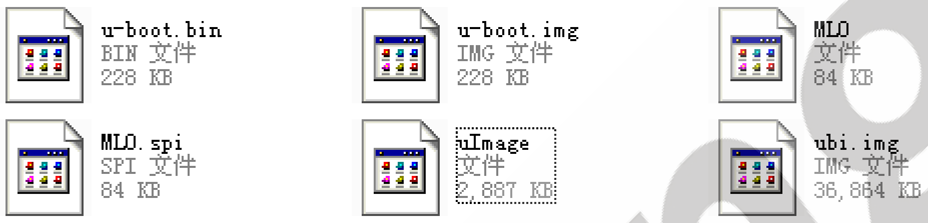
1. The former version of PSP04.06.00.08 (and AMSDK 05.05.00.00) do not support starting U-Boot from SPI flash

2. Need to burn image to SPI flash through SD card.

4.4.2 Preparation

1. Create SD card boot disk,Refer to *section 4.2.1*.

2. After creating SD card boot disk, copy file to the first partition in SD card. Include the following files:



u-boot.bin
BIN 文件
228 KB

u-boot.img
IMG 文件
228 KB

MLO
文件
84 KB

MLO.spi
SPI 文件
84 KB

uImage
文件
2,887 KB

ubi.img
IMG 文件
36,864 KB

picture 22

If NAND Flash has been burnt into the kernel and file system, can delete [uImage] and [ubi.img].

4.4.3 Burning Procedure

1. Confirm that the SD card includes 4 files at least:

MLO, MLO.spi,u-boot.bin,u-boot.img, insert SD card to mainboard slot.

2. Set startup resistance in mainboard to SPI startup mode 0110 1000 0000,

Weld R99,R100,R101,R102,R103,R104,R105,R106 by 100K resistance.

Remove R53 R54 R56

Weld R52 R55 R57 R58 R59 by 10K resistance

3. Power on, press any keys in 3 seconds, appear the hint of [U-Boot#]，burn the U-Boot image into SPI Flash through SD card.

4. Input the following command to test whether it has dateflash chip.

```
U-Boot# sf probe 0
```

Print the following information if normally:

SF: Detected GD25Q16 with page size 4 KiB, total 2 MiB

Print the following information if abnormally.

SF: Unsupported manufacturer 00

Failed to initialize SPI flash at 0:0

5. If it is normally ,then execute erasing ,burning etc, the command is as follows:

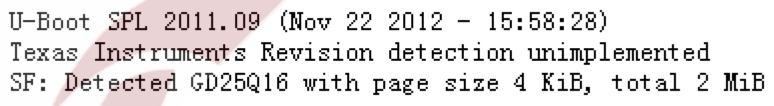
```
U-Boot# sf erase 0 +80000
U-Boot# mmc rescan
U-Boot# fatload mmc 0 ${loadaddr} MLO.spi
U-Boot# sf write ${loadaddr} 0 ${filesize}
U-Boot# fatload mmc 0 ${loadaddr} u-boot.bin
U-Boot# sf write ${loadaddr} 0x20000 ${filesize}
```

If there are no errors, U-Boot has been burnt into SPI Flash, then take out SD card, restart mainboard.

4.4.4 Start mainboard

1. Take out the SD card, power on to the mainboard.

2. After power on to the mainboard, appearing the following information ,it means starting from SPI Flash.



```
U-Boot SPL 2011.09 (Nov 22 2012 - 15:58:28)
Texas Instruments Revision detection unimplemented
SF: Detected GD25Q16 with page size 4 KiB, total 2 MiB
```

picture 23

3. If NAND Flash has burnt kernel and file system, can enter system, If there is no burning ,burn Linux system according to CD.

4.4.5 Notice

If burning is not according to command, or there is a problem in burning file, it leads boot failure of SPI mode. Solution: connect the sixth pin(clock pin) with the fourth pin(ground wire) of U5 date flash on the core board. That will help the date flash stop working, then power on again, enter U-Boot to erase data.

Ⅴ. Develop Application Program

5.1 Hello World

You can develop application program on mainboard. Here is the sample "Hello World".

```
#include <stdio.h>
int main(void)
{
printf("Hello World ! \n");
return 0;
}
```

Save to file [hello.c], cross compiling by the following command.

5.2 Cross Compiling

Compile the application program by the installed cross compiling before running program on the mainboard; We can use the following command to compile:

```
$ arm-arago-linux-gnueabi-gcc –o hello hello.c
$ file hello
```

hello: ELF 32-bit LSB executable, ARM, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.16, not stripped.

Generate executable binary file [hello] on ARM platform in the current directory after compiling.

5.3 Running Application Program

Obtain executable program on the mainboard, introduce how to run application program on the mainboard, two ways: mount NFS server and add to file system.

We can use the configured NFS server, mount mainframe's NFS server to mainboard then can operate mainframe's file on the mainboard, such as copy file, run program. Be easy for debugging, implementation method is as follows:

1. Ensure connecting mainboard with PC by network cable, start NFS service on PC.

2. Set mainboard IP and PC IP, as follows:

PC IP ： 192.168.1.241

Mainboard IP： 192.168.1.110

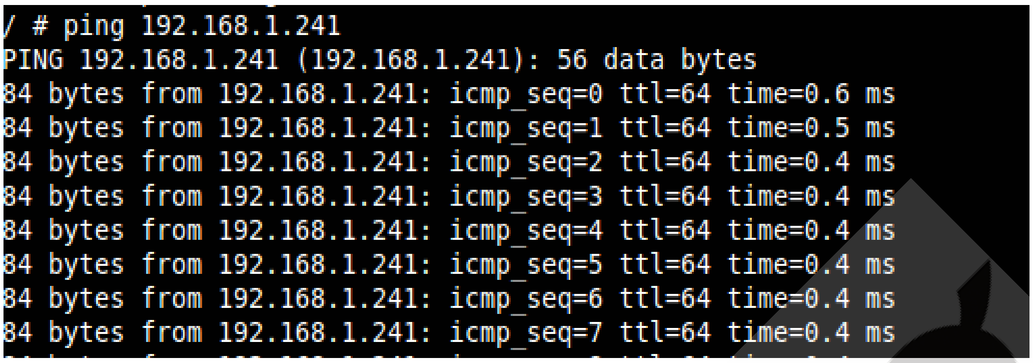
Network Marsk: 255.255.255.0

Broadcast IP: 192.168.1.255

3. Test network connectivity

Run the following command in hyper terminal

```
# ping 192.168.1.241
```



```
/ # ping 192.168.1.241
PING 192.168.1.241 (192.168.1.241): 56 data bytes
84 bytes from 192.168.1.241: icmp_seq=0 ttl=64 time=0.6 ms
84 bytes from 192.168.1.241: icmp_seq=1 ttl=64 time=0.5 ms
84 bytes from 192.168.1.241: icmp_seq=2 ttl=64 time=0.4 ms
84 bytes from 192.168.1.241: icmp_seq=3 ttl=64 time=0.4 ms
84 bytes from 192.168.1.241: icmp_seq=4 ttl=64 time=0.4 ms
84 bytes from 192.168.1.241: icmp_seq=5 ttl=64 time=0.4 ms
84 bytes from 192.168.1.241: icmp_seq=6 ttl=64 time=0.4 ms
84 bytes from 192.168.1.241: icmp_seq=7 ttl=64 time=0.4 ms
```

picture 24

Enter ping to mainboard, if ping runs well in mainframe and mainboard, it means network connectivity is normal.

4. Mount Mainframe NFS Server

Enter the following command in mainboard terminal.

```
# mount –o nolock 192.168.1.241:/    /mnt
# cd /mnt
# ls
# ./hello
```

Mount correctly, found root directory in the [/mnt] directory on the mainboard, then enter the directory to run the application program. By this way, it's easy for debugging program. Until the program is debugged correctly, add application program to file system, burn into the NAND Flash. In this way, avoid burning program to NAND Flash continuously.

5.4 Autostart Application Program

Edit [etc/profile]

```
# vi /etc/profile
```

Add the [./usr/hello &] run command of application program to [profile] file.

Ⅵ. Mainboard Test

6.1 Preparation

6.1.1 Configuration

◆ Build in Linux system(ubuntu or other Linux release version).

◆ Serial port connect: Connect debug UART on mainboard with serial port on PC by serial port line.

◆ Network Connect: Connect Ethernet interface on mainboard with network interface on PC by network cable.

◆ LCD Connect: Connect LCD panel to mainboard by a double row line.

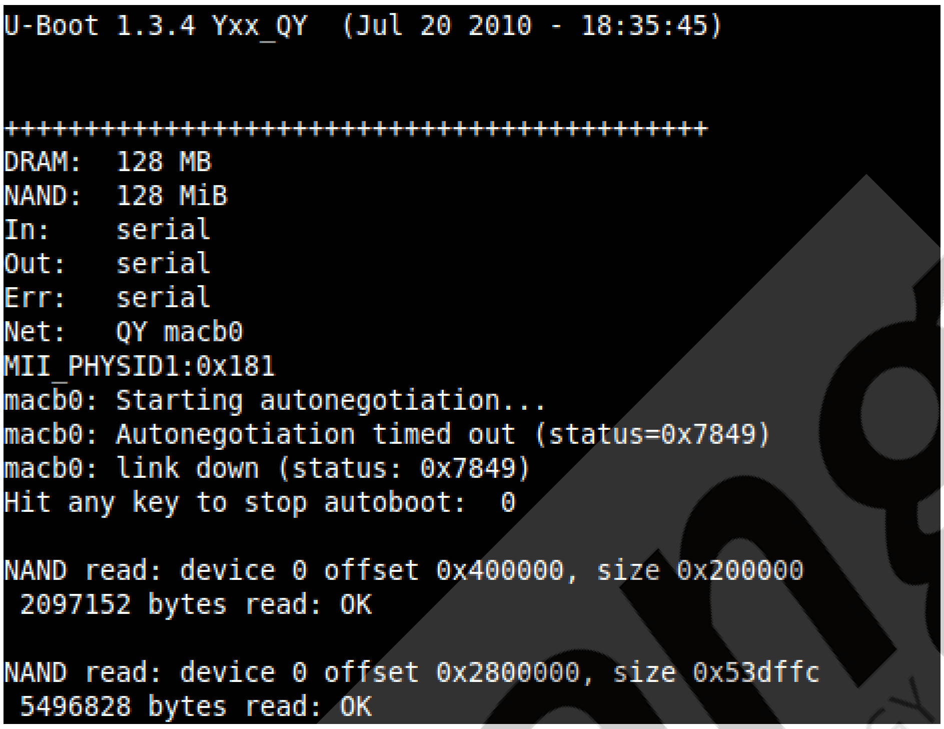
6.1.2 Mainboard Requirement

The following test sample run in Linux environment, please confirm the mainboard has started the Linux system normally. Relevant document, please refer to *IAC-335X-Kit Linux User Manual*.

6.1.3 Start Mainboard

1. Open hyper terminal on PC, set corresponding serial port in PC, select baud rate [115200],stop bit [1], data bit [8], parity bit [none] and data flow control [none].

2. Connect power supply, start mainboard, observe system startup information in terminal, as shown:



```
U-Boot 1.3.4 Yxx_QY  (Jul 20 2010 - 18:35:45)


+++++++++++++++++++++++++++++++++++++++++++++
DRAM:  128 MB
NAND:  128 MiB
In:    serial
Out:   serial
Err:   serial
Net:   QY macb0
MII_PHYSID1:0x181
macb0: Starting autonegotiation...
macb0: Autonegotiation timed out (status=0x7849)
macb0: link down (status: 0x7849)
Hit any key to stop autoboot:  0

NAND read: device 0 offset 0x400000, size 0x200000
 2097152 bytes read: OK

NAND read: device 0 offset 0x2800000, size 0x53dffc
 5496828 bytes read: OK
```

picture 25

3. Enter [/usr/test] directory.

```
# cd /usr/test
# ls
```

4. Enter [test] directory.

```
# cd test
# ls
```

Test program compiled in the test directory; test the relevant module when running every test program.

6.2 Mainboard Test

6.2.1 RTC Test

Kernel includes RTC. Users can test by the following method:

1. Set current time by data command.

```
# date 072210502012          Set the current time to 2010-07-22 10:50:00
```

2. Burn hardware RTC by hwclock.

```
# hwclock –w
```

3. Reboot system after power outage, check the current time by data command.

```
# date
```

By this time, the time has been set. If time has not been burnt into hardware RTC, please check the back plane whether installed the backup battery.

4. Read RTC time by program, refer to [rtc_test.c]

Run [/usr/test/rtc_test]

```
/usr/test # ./rtc_test
```

Test Result: After the program running, read the RTC time in 10 seconds continuously and output print information from the serial port.

·Test Code: rtc_test/rtc_test.c

6.2.2 Buzzer Test

Test program is in [/usr/test] directory, test is as follows:

```
# ./buzzer
```

After buzzing twice, close it.

6.2.3 Serial Port Test

There are 5-ch serial ports:J11(COM2 and COM3),J12(COM4 and COM5) &1-ch debug UART. J13 multiplexs with 2-ch CAN interfaces,J12(COM4 and COM5) multiplexs with RS485.

Serial port test in [/usr/test] directory, test as follows;

```
# cd /usr/test
# ./serial_test
```

Connect serial port on mainboard to PC by a serial port line. Open serial console in PC, select baud rate [115200],stop bit [1], data bit [8], parity bit [none] and data flow control[none]. Then test every serial ports.

Test Result: Appear the following information after running correctly.

（1）Connect COM2:

Display the following information continuously in serial port communication tool: this is a Serial_Port2 test!

Display the data continuously in debug terminal, as: Serial 2:

Len: 9

Pri: 123456789

（2）Connect COM3,COM4,COM5,test result is similar with COM2.

·Test code: serial_test/serial_test.c

6.2.4 CAN Bus Test

IAC-335X-Kit be of 2-ch CAN bus equipments:CAN0 and CAN1,this 2-ch CAN bus equipment are in the software which is packaged as network equipment, named socket-can, specific test procedure is as follows:

1. Correspondingly connect J15(CANH0,CANL0,GND)on board 1 with CAN bus interface(CANH0,CANL0,GND) on board 2.

2. Start system, set bund rate [125000] in serial port.

```
# ip link set can0 type can bitrate 125000
```

If need to start CAN1 equipment, execute the following command.

```
# ip link set can1 type can bitrate 125000
```

Set same bund rate of board 1 and board 2.

（1） Start CAN equipment.

```
# ifconfig can0 up
```

If need to start CAN1 equipment, execute the following command.

```
# ifconfig can1 up
```

（2）Observe a new network equipment can0.

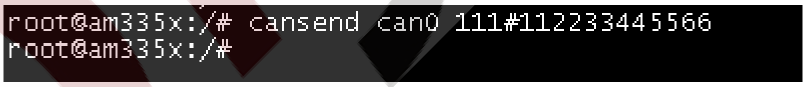
3. Receive and send test.

Start can0 in board1 and board2 respectively. Can do a receive test and send test.

（1）Board1 terminal execute to receive command.

```
# candump can0 &
```

（2）Board2 terminal execute to send command .As shown:



```
root@am335x:/# cansend can0 111#112233445566
root@am335x:/#
```

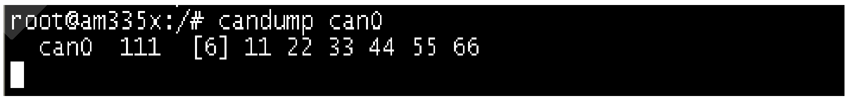
picture 26

If use can1 equipment, execute the following command:

```
# cansend can1 111#112233445566
```

111 is ID, 112233445566 is the sending data frame.

（3）At this time, the received data are printed in terminal on board1 terminal.



```
root@am335x:/# candump can0
  can0  111   [6] 11 22 33 44 55 66
```

If use can1 equipment, execute the following command.

```
# candump can1 &
# cansend can0 111#112233445566
```

6.2.5 RS485 Test

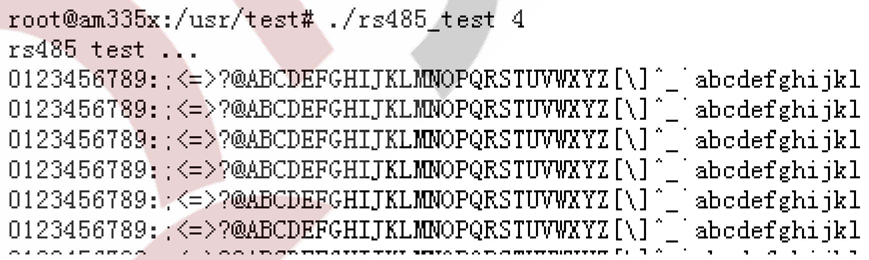
RS485 operation is similar with RS232.Control sending or receiving of 485 by IO port when sending or receiving.

Test RS485 by 485 equipment. If without 485 equipment, connect 2-ch 485 on the mainboard. (Connect pin 1with pin 2 on J14, connect pin3 with pin4 by jumper wire), test RS485 port.

Run program [rs485_test/] in [/usr/test] directory, parameter 0 means ttyO4 sending and ttyO5 receiving, parameter 1 means ttyO5 sending and ttyO4 receiving.

```
#./rs485_test 4
```

Test result: Print on the console after 1-ch 485 sending character strings continuously through another 1-ch 485.



```
root@am335x:/usr/test# ./rs485_test 4
rs485 test ...
0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijkl
0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijkl
0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijkl
0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijkl
0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijkl
0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijkl
```

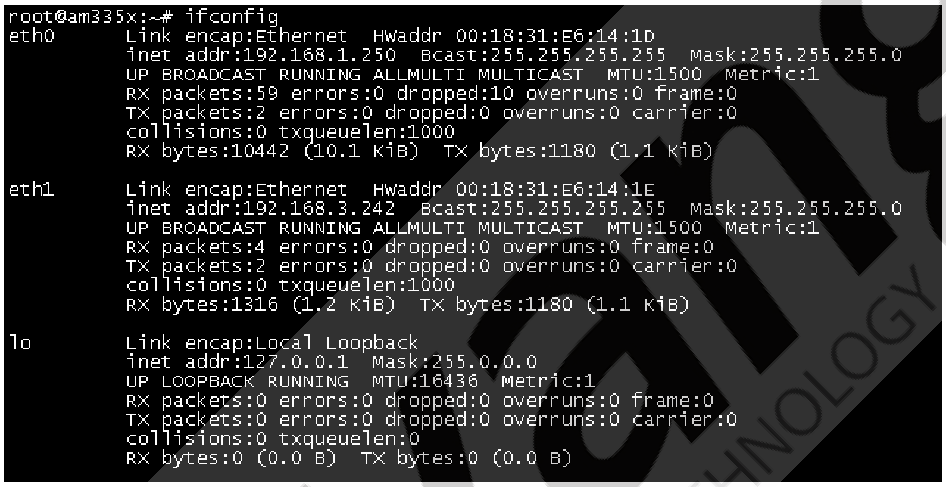
picture 27

·Test Code: Test source code: rs485_test/ rs485_test.c

6.2.6 Ethernet Test

IAC-335X-Kit mainboard has 2-ch Ethernet interfaces, configure dual standalone EMAC mode currently, set 2 network cards in two different network segments, test procedure as follows:

1. Connect network 1(J19) and network 2(J20) to different network segments respectively, examine network card equipment by [ifconfig], two equipments: eth0 and eth1, as shown:



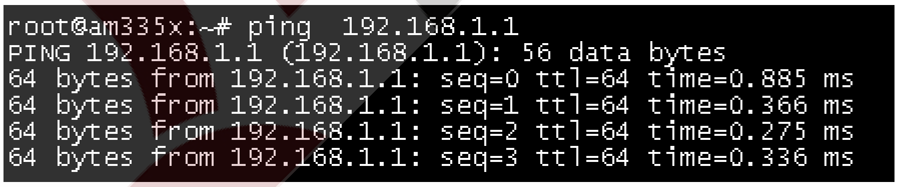
```
root@am335x:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:18:31:E6:14:1D
          inet addr:192.168.1.250  Bcast:255.255.255.255  Mask:255.255.255.0
          UP BROADCAST RUNNING ALLMULTI MULTICAST  MTU:1500  Metric:1
          RX packets:59 errors:0 dropped:10 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10442 (10.1 KiB)  TX bytes:1180 (1.1 KiB)

eth1      Link encap:Ethernet  HWaddr 00:18:31:E6:14:1E
          inet addr:192.168.3.242  Bcast:255.255.255.255  Mask:255.255.255.0
          UP BROADCAST RUNNING ALLMULTI MULTICAST  MTU:1500  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1316 (1.2 KiB)  TX bytes:1180 (1.1 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

picture 28

Network Card 1: IP 192.168.1.250; Network Card 2: IP 192.168.3.242

2. Test connectivity for network equipment eth0.

```
# ping 192.168.1.1
```

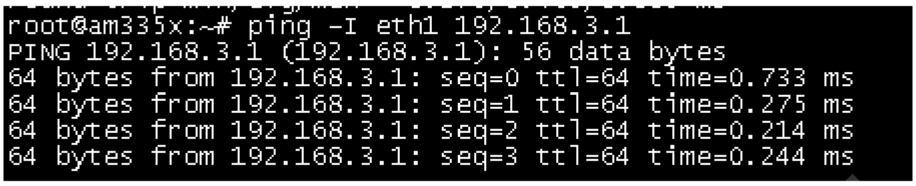


```
root@am335x:~# ping  192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: seq=0 ttl=64 time=0.885 ms
64 bytes from 192.168.1.1: seq=1 ttl=64 time=0.366 ms
64 bytes from 192.168.1.1: seq=2 ttl=64 time=0.275 ms
64 bytes from 192.168.1.1: seq=3 ttl=64 time=0.336 ms
```

picture 29

3. Test connectivity for network equipment eth1.

```
# ping –I eth1 192.168.3.1
```

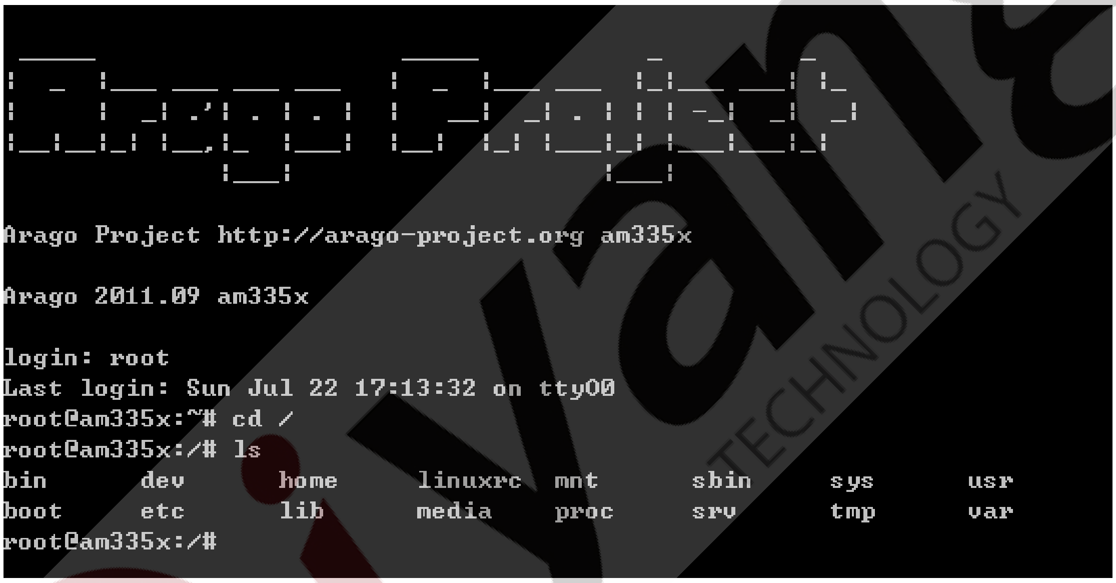


```
root@am335x:~# ping -I eth1 192.168.3.1
PING 192.168.3.1 (192.168.3.1): 56 data bytes
64 bytes from 192.168.3.1: seq=0 ttl=64 time=0.733 ms
64 bytes from 192.168.3.1: seq=1 ttl=64 time=0.275 ms
64 bytes from 192.168.3.1: seq=2 ttl=64 time=0.214 ms
64 bytes from 192.168.3.1: seq=3 ttl=64 time=0.244 ms
```

picture 30

6.2.7 Login Mainboard by Telnet

Input: telnet 192.168.1.250 in Windows command line. Appear a login interface, input root, do not need password. As shown:

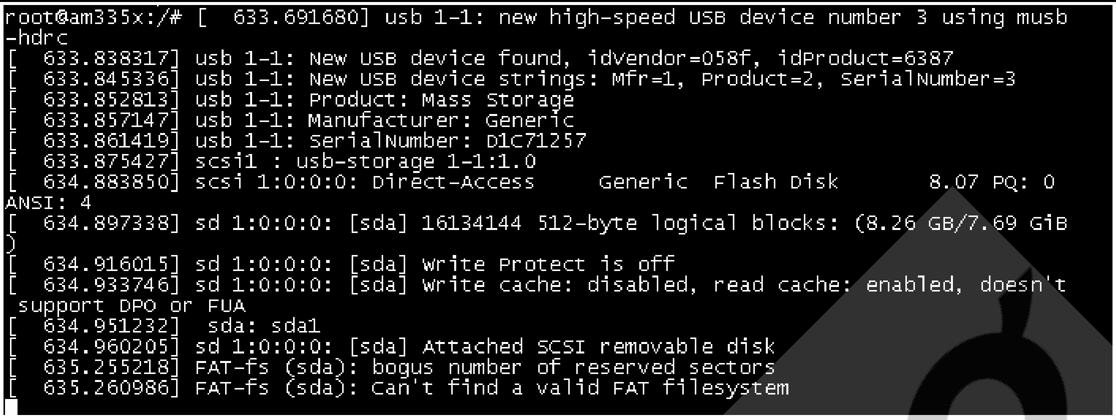


```
 _____                    _____           _         _
|  _  |___ ___ ___ ___   |  _  |___ ___  |_|___ ___| |_
|     |  _| .'| . | . |  |   __|  _| . | | | -_|  _|  _|
|__|__|_| |__,|_  |___|  |__|  |_| |___|_| |___|___|_|
              |___|                    |___|

Arago Project http://arago-project.org am335x

Arago 2011.09 am335x

login: root
Last login: Sun Jul 22 17:13:32 on tty00
root@am335x:~# cd /
root@am335x:/# ls
bin       dev       home      linuxrc   mnt       sbin      sys       usr
boot      etc       lib       media     proc      srv       tmp       var
root@am335x:/#
```

picture 31

6.2.8 USB Hard Disk Test

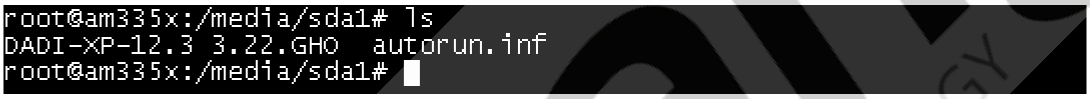
Insert USB hard disk to mainboard, appear the following information in serial console.



```
root@am335x:/# [  633.691680] usb 1-1: new high-speed USB device number 3 using musb
-hdrc
[  633.838317] usb 1-1: New USB device found, idVendor=058f, idProduct=6387
[  633.845336] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[  633.852813] usb 1-1: Product: Mass Storage
[  633.857147] usb 1-1: Manufacturer: Generic
[  633.861419] usb 1-1: SerialNumber: D1C71257
[  633.875427] scsi1 : usb-storage 1-1:1.0
[  634.883850] scsi 1:0:0:0: Direct-Access     Generic  Flash Disk       8.07 PQ: 0
ANSI: 4
[  634.897338] sd 1:0:0:0: [sda] 16134144 512-byte logical blocks: (8.26 GB/7.69 GiB
)
[  634.916015] sd 1:0:0:0: [sda] Write Protect is off
[  634.933746] sd 1:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't
 support DPO or FUA
[  634.951232]  sda: sda1
[  634.960205] sd 1:0:0:0: [sda] Attached SCSI removable disk
[  635.255218] FAT-fs (sda): bogus number of reserved sectors
[  635.260986] FAT-fs (sda): Can't find a valid FAT filesystem
```

picture 32

At this time, system will mount the USB hard disk to [/media/sda1] directory automatically, as shown:

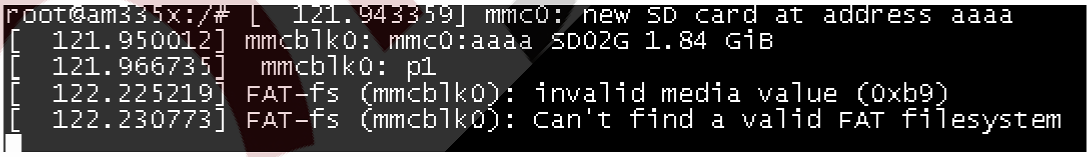


```
root@am335x:/media/sda1# ls
DADI-XP-12.3 3.22.GHO  autorun.inf
root@am335x:/media/sda1#
```

picture 33

6.2.9 SD Card Test

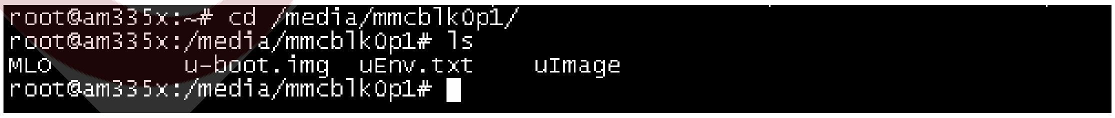
Start system, regard SD card as memory card. Found the following information in serial console when insert a SD card to the SD slot and identify the system.



```
root@am335x:/# [  121.943359] mmc0: new SD card at address aaaa
[  121.950012] mmcblk0: mmc0:aaaa SD02G 1.84 GiB
[  121.966735]  mmcblk0: p1
[  122.225219] FAT-fs (mmcblk0): invalid media value (0xb9)
[  122.230773] FAT-fs (mmcblk0): Can't find a valid FAT filesystem
```

picture 34

Then, system will mount SD card to [/media/mmcblk0p1], as shown:



```
root@am335x:~# cd /media/mmcblk0p1/
root@am335x:/media/mmcblk0p1# ls
MLO         u-boot.img  uEnv.txt    uImage
root@am335x:/media/mmcblk0p1#
```

picture 35

6.2.10 PWM Test

Pwm test, can be used as bright control for the LCD backlight, testing pin is MCA0_MCKR pin on back plane.

Run [pwm_test]program in [/usr/test] directory.

```
#./pwm_test    30
```

30 is duty ratio, input 0~100,can observe that J7(33 pin) output waveform by oscilloscope。

6.2.11 Display mp3

Run program

```
# madplay /usr/test/hotelcalifornia.mp3
```

6.2.12 Touch Panel Test

When connect LCD and touch panel, running calibration program [ts_calibrate], the specific operation is as follows:

```
# ts_calibrate
```

Run the above calibration program, you will see a cross icon on the LCD display, then click it; And you will see a same cross icon, click it again. Finish touch panel calibration after clicking 5 cross icons, then you can run test program.

```
# ts_test
```

Test Result: The debug UART output the current coordinate point if it is running correctly. The cross icon in the LCD will follow.

6.2.13 Qt Program Test

Observe qt demo program interface when running qt test program [am335x_qt] in [/usr/test] directory

```
$ cd /usr/test
$ ./am335x_qt -qws
```

Qt migrating and using, please refer to the following chapter.

Ⅶ. Migrating QT4.8.2

7.1 Preparation

Hardware Resource: IAC-335X-Kit mainboard

Host Machine: ubuntu 12.04

Qt source code: qt-everywhere-opensource-src-4.8.2.tar.gz

Touch Panel Library: tslib1.4.tar.gz

Cross-compiler tool: arm-arago-linux-gnueabi.tar.gz

7.2 Add the Library of Supporting Touch Panel

1. Confirm the cross-compiler tools installed correctly

2. Compile library for supporting touch panel.

（1）Execute the following upgrade commands in [ubuntu] command line:

```
$ sudo apt-get install libtool
$ sudo apt-get install autoconf
$ sudo apt-get install automake
```

（2）Extract [tslib1.4.tar.gz]

```
$ cd   tslib
```

（3）Then execute command

```
./configure   --host=arm-linux    ac_cv_func_malloc_0_nonnull=yes
CC=arm-arago-linux-gnueabi-gcc   CXX=arm-arago-linux-gnueabi-g++
-prefix=/home/liuc/qtenv/tslib
```

Remark: Confirm that the cross-compiler tool is correct, [-prefix] is as installation directory, [tslib] is in the [/home/liuc/qtenv/] directory after compiling.

（4）Compile and Install

```
$ make
$ make install
```

（5）Configure

[tslib] is in the [/home/liuc/qtenv/] directory after compiling. Enter the [tslib] directory, edit [ts.conf'] file.

```
$ cd tslib/etc
$ gedit ts.conf
```

Edit this file, remove # and space in front of 'module_raw input'.

If there is a space in front of 'module_raw input', hint 'Segmentation fault' when running testing software

7.3 Install QT Library

1. Extract the [qt-4.8.2] source code.

```
$ tar -zxvf qt-everywhere-opensource-src-4.8.2.tar.gz
```

2. Compile

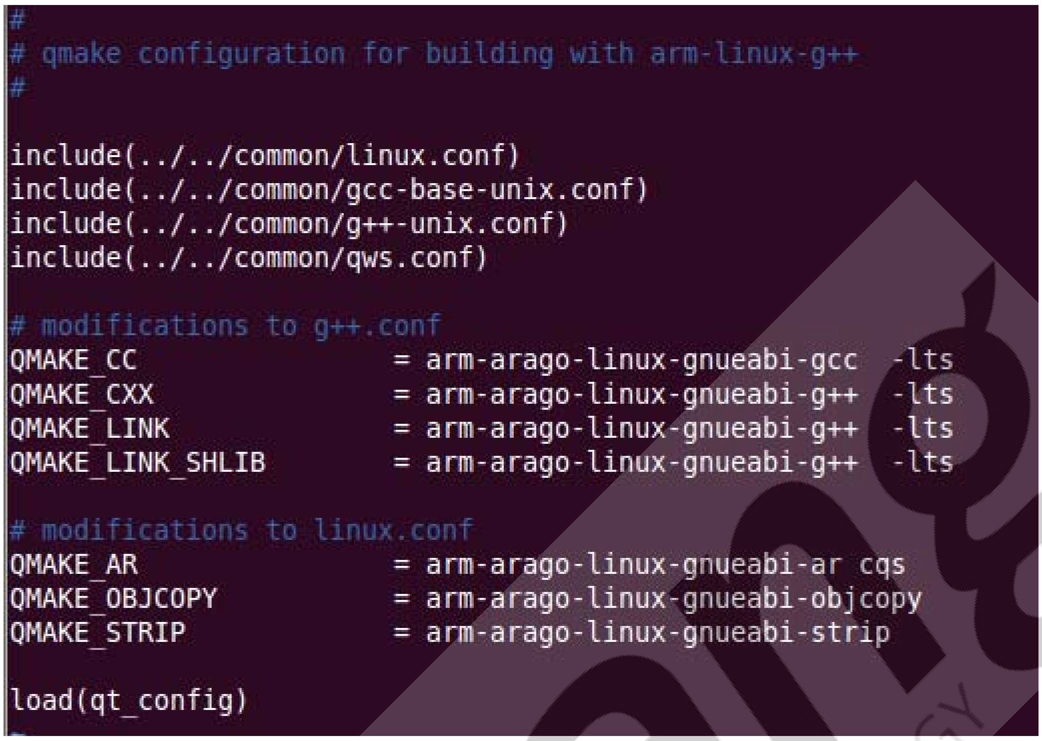
（1）Enter the qt directory

```
$ cd qt-everywhere-opensource-src-4.8.2
```

（2）Specify the cross-compiler

```
$ vi mkspecs/qws/linux-arm-g++/qmake.conf
```

Edit [qmak. conf], change"arm-linux" to "arm-arago-linux-gnueabi" and add parameter "lts" to four options, as shown:



```
#
# qmake configuration for building with arm-linux-g++
#

include(../../common/linux.conf)
include(../../common/gcc-base-unix.conf)
include(../../common/g++-unix.conf)
include(../../common/qws.conf)

# modifications to g++.conf
QMAKE_CC                = arm-arago-linux-gnueabi-gcc  -lts
QMAKE_CXX               = arm-arago-linux-gnueabi-g++  -lts
QMAKE_LINK              = arm-arago-linux-gnueabi-g++  -lts
QMAKE_LINK_SHLIB        = arm-arago-linux-gnueabi-g++  -lts

# modifications to linux.conf
QMAKE_AR                = arm-arago-linux-gnueabi-ar cqs
QMAKE_OBJCOPY           = arm-arago-linux-gnueabi-objcopy
QMAKE_STRIP             = arm-arago-linux-gnueabi-strip

load(qt_config)
```

picture 36

（3）Create [build_qt.sh] script file, and add the following content.

```
# !bin/bash
./configure     -prefix $HOME/qtenv/qt-4.8.2-arm \
  -opensource \
  -confirm-license    \
  -embedded arm \
  -xplatform qws/linux-arm-g++ \
  -platform /qws/linux-x86-g++ \
  -little-endian \
  -host-little-endian \
  -shared \
  -no-qt3support \
  -no-phonon -no-phonon-backend \
  -qt-zlib \
  -no-gif \
  -no-libtiff \
-no-qvfb \
  -qt-libjpeg \
  -no-nis \
-no-opengl \
  -no-cups \
  -no-webkit \
  -no-glib \
  -no-dbus \
  -no-rpath \
  -no-mmx -no-3dnow \
-no-sse -no-sse2 -no-sse3 -no-ssse3 -no-sse4.1 -no-sse4.2 \
  -no-avx -no-neon \
  -no-audio-backend \
  -no-svg \
  -no-javascript-jit \
  -no-script \
  -no-scripttools \
  -no-multimedia \
  -no-openssl \
  -nomake tools \
-qt-mouse-tslib \
  -I/home/liuc/qtenv/tslib/include \
  -L/home/liuc/qtenv/tslib/lib
```

Remark: [-I/home/liuc qtenv/tslib/include] and [-L/home/liuc qtenv/tslib/lib] are the [tslib] installation paths. [-prefix $HOME/qtenv/qt-4.8.2-arm] is corresponding to the QT's installation path in future.

（4）Execute [build_qt.sh] file, and configure qt compiling rules.

```
$ sh build_qt.sh
$ make
$ make install
```

After installation completed, QT libraries and Demo program will be generated in the installation directory.

3. Set system environment variables after finishing compilation and installation, then compile your program.

```
$ vim setARMenv.sh
#!/bin/sh
export   QTEDIR=/home/liuc/qtenv/qt-4.8.2-arm:$QTEDIR
export   PATH=/home/liuc/qtenv/qt-4.8.2-arm/bin:$PATH
export   LD_LIBRARY_PATH=/home/liuc/qtenv/qt-4.8.2-arm/lib: LD_LIBRARY_PATH
```

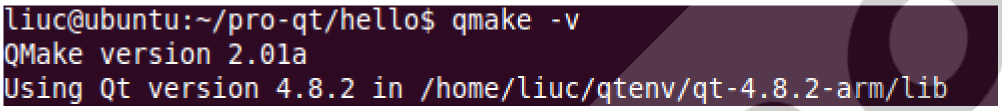
Remark: The above path is to install QT libraries' installation path.

7.4 Compile QT Program

1. Sample:am3359 QT interface routine; finish QT project, copy to ubuntu, modify the environment variables first.

```
$ source setARMenv.sh
```

Then check the [qmake] path, whether it correctly.



```
liuc@ubuntu:~/pro-qt/hello$ qmake -v
QMake version 2.01a
Using Qt version 4.8.2 in /home/liuc/qtenv/qt-4.8.2-arm/lib
```

picture 37

Execute the command

```
$ qmake -project        // Generate project file[.pro]
$ qmake                 // Generate[makefile]
$ make                  // Generate executable file [am335x_qt]
```

2. For the complicating project, suggest using [qtcreator] to compile.

7.5 Migrate Library to Mainboard

1. After compiling successfully, then generate QT library ARM needed in the QT' sinstallation directory, such as:[/home/liuc/qtenv/qt-4.8.2-arm/lib]. In order to migrate easily in future, we suggest packaging the file by the following command in [qt4.8.2-arm] directory.

```
$ tar -zcvf   lib.tar.gz   lib/
```

2. Migrate to the mainboard

Create a same directory as virtual machine [/home/liuc/qtenv/qt-4.8.2-arm] in mainboard.

Extract the compressed library files to this directory; the [lib] directory includes libraries and fonts.

After the library files have been migrated, we need to set the environment variables.

Execute it in the mainboard's root directory:

```
$ vi /etc/profile
```

Then input the following data:

```
export QTDIR=/home/liuc/qtenv/qt-4.8.2-arm:$QTDIR
export LD_LIBRARY_PATH=/home/liuc/qtenv/qt-4.8.2-arm/lib:$LD_LIBRARY_PATH
export QT_QWS_FONTDIR=/home/liuc/qtenv/qt-4.8.2-arm/lib/fonts
```

Save & Eixt!

3. Prepare [tslib] file, enable the touch panel to work normally.

Copy the file in [/home/liuc/qtenv/tslib] directory of virtual machine to [/home/liuc/qtenv/tslib] directory of mainboard. If there is no this file, please create a new file.

Modify the touch panel configuration file

（1）Environment variables

Enable the [tslib] to run correctly, configure the following [tslib] environment variables.

Set the environment variables as follows: (or in shell format)

```
export T_ROOT=/home/liuc/qtenv/tslib/
export LD_LIBRARY_PATH=/home/liuc/qtenv/tslib/lib:$LD_LIBRARY_PATH
export TSLIB_CONSOLEDEVICE=none
export TSLIB_FBDEVICE=/dev/fb0
export TSLIB_TSDEVICE=/dev/input/event0
export TSLIB_PLUGINDIR=$T_ROOT/lib/ts
export TSLIB_CONFFILE=$T_ROOT/etc/ts.conf
export POINTERCAL_FILE=/etc/pointercal
export TSLIB_CALIBFILE=/etc/pointercal
export QWS_MOUSE_PROTO=tslib:/dev/input/event0
```

Also can burn into [/etc/profile].

（2）Support the touch panel.

After calibrating the touch panel, execute the following command to support the touch panel.

```
$ export QWS_MOUSE_PROTO=Tslib:/dev/input/event0
```

（3）Support the mouse.

```
$ export QWS_MOUSE_PROTO=MouseMan:/dev/input/mice
```

（4）Support the touch panel and mouse at the same time.

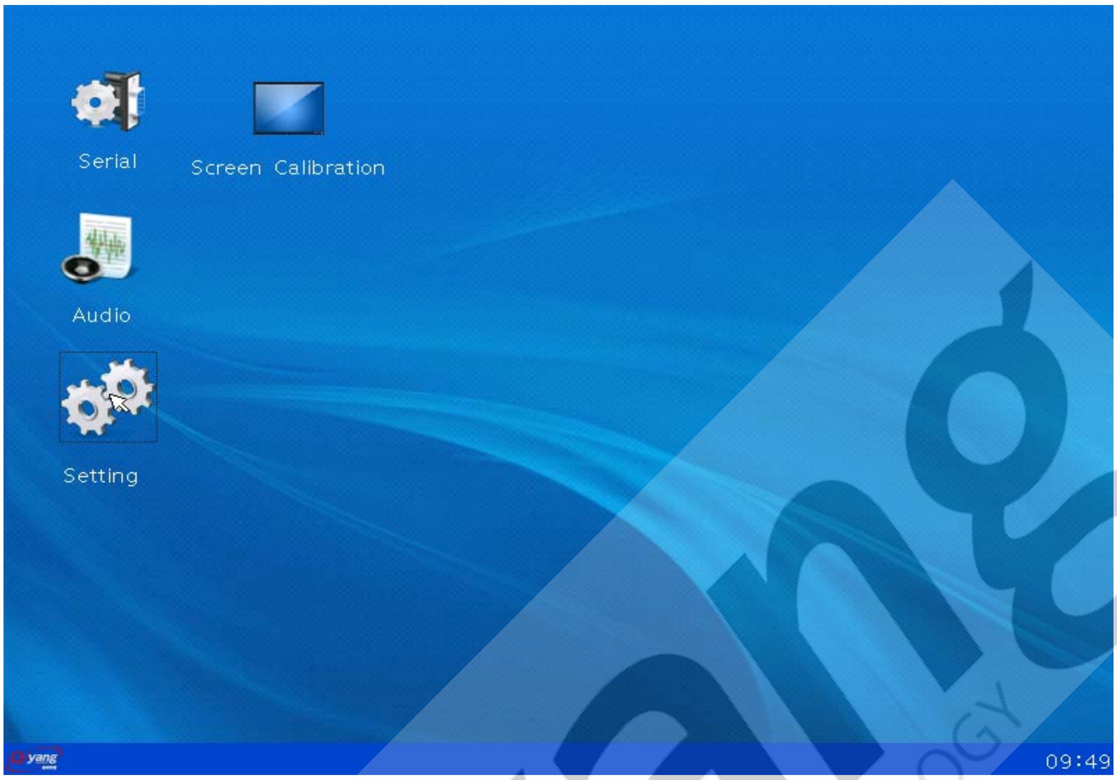
```
$ export set QWS_MOUSE_PROTO="TSLIB:/dev/input/event0
Intellimouse:/dev/input/mice"
```

Then execute test program [./am335x_qt-qws]

The QT interface will appear.

7.6 am335x_qt Main Interface Functions

Run Qt program in auto-start mode, enter Qt main interface after starting development board, the interface includes Serial, Audio, Settings and Screen Calibration, and the Qiyang's logo on the left bottom and the time on the right bottom in status bar,as shown:



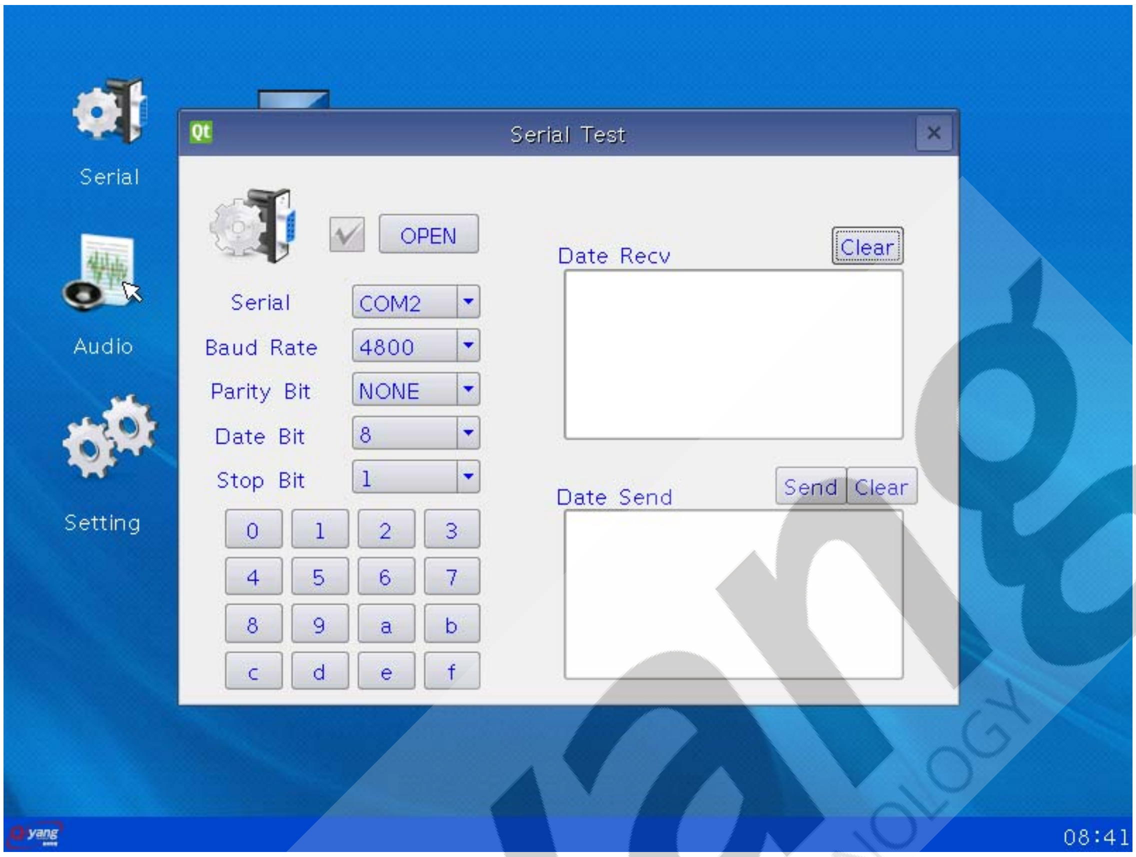
picture 38

7.6.1 Serial Port Communication

Set serial port, baud rate, parity, data bit and stop bit. Then click [open], and the button is changed to be [close]. Do not set baud rate again; if you need to set, close the serial port first then set again.

Can send and receive data after opening successfully.

©2012 Qiyangtech Copyright

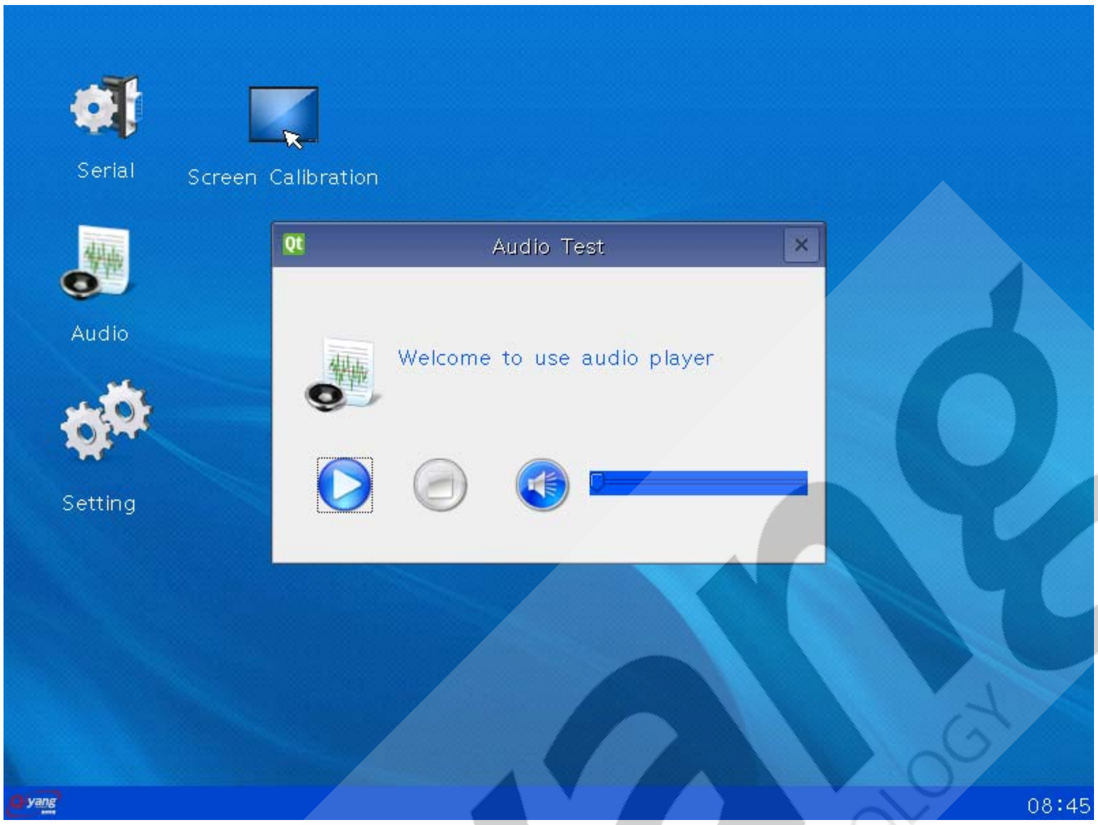


Serial: COM2
Baud Rate: 4800
Parity Bit: NONE
Date Bit: 8
Stop Bit: 1

picture 39

7.6.2 Audio Test

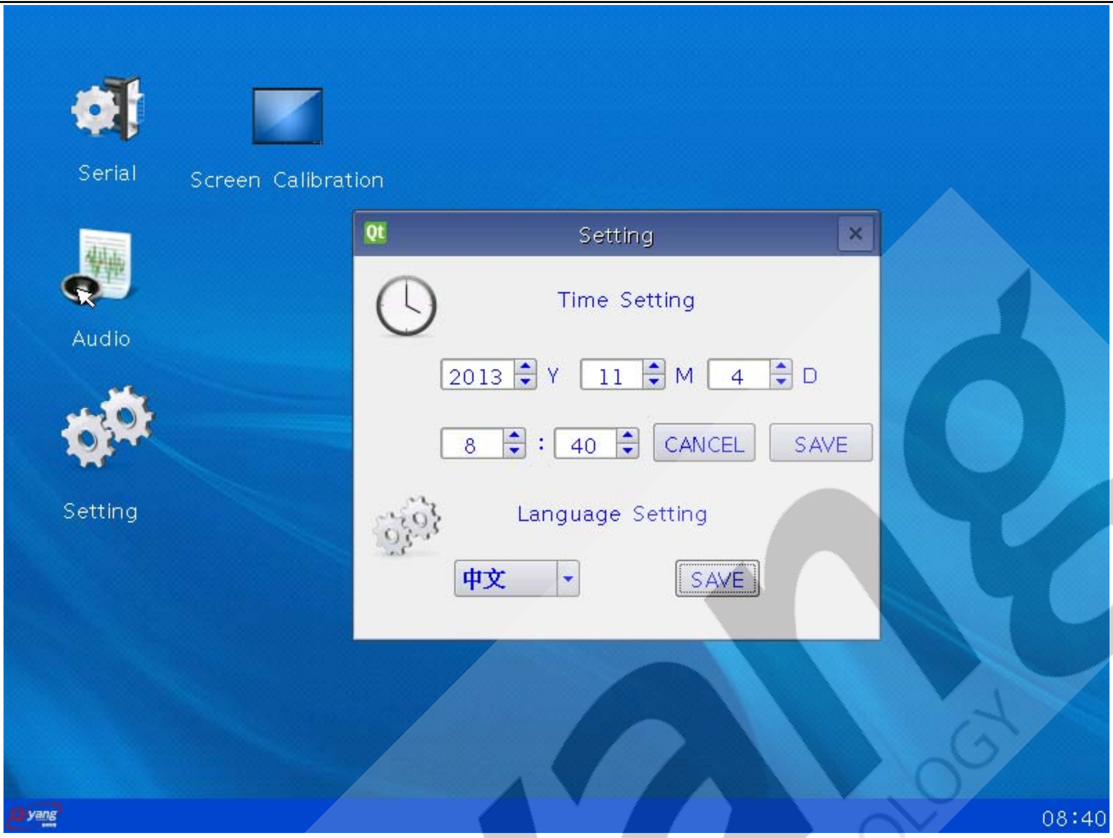
Audio test function includes play, stop, volume control, hint picture.



picture 40

7.6.3 Setting

Setting includes time setting and language setting function, as shown:



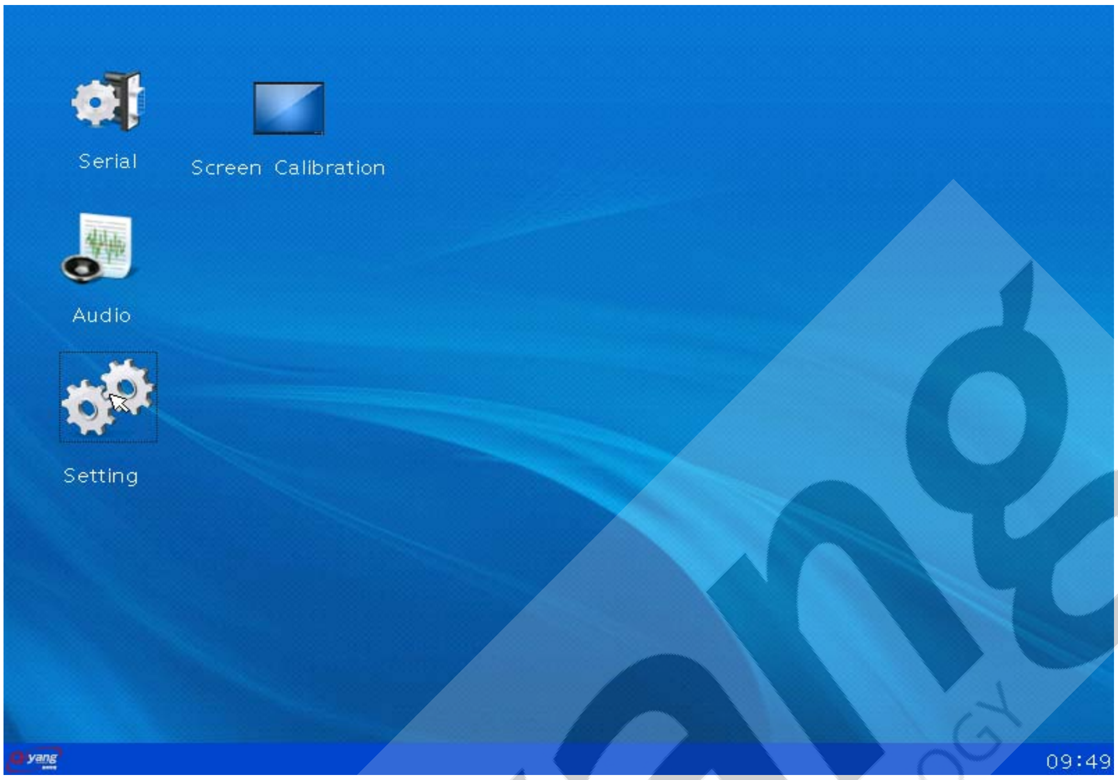
picture 41

7.6.3.1 Time Setting

Setting screen operation is very easy, setup completed, if you don't save, click [cancel]; after click [save], the system time will change to the time you set.

7.6.3.2 Language Settings

Select the language, click [save] ,English interface is as shown:



picture 42

7.6.4 Screen Calibration

Click screen calibration, enter the calibration interface directly, appear a dialog after calibrating, display whether to recalibrate, calibration takes effect after restarting. If calibration is not accurate, you can calibrate again, then restart or cancel.

7.6.5 Screen Font

The default font is [wenquanyi_160_75.qpf], user can also download fond font, correct time automatically and put it into the [/home/liuc/qtenv/qt-4.8.2-arm/lib/fonts] directory.

©2012 Qiyangtech Copyright

Ⅷ. Migrate and use ntp

8.1 Illustration

Network Time Protocol(NTP) keep the clock accurately. If it can visit internet , just install ntp client software to ntp public server in internet, can correct time automatically.

8.2 Preparation

1. ntp source code: ntp-4.2.4p7.tar.gz

2. Cross-compiler tool: arm-arago-linux-gnueabi-gcc

8.3 Migrating Procedure

1. Extract ntp source package to specified directory

```
[liuc@QY-SVN ntp-4.2.4p7]$ tar  zxvf  ntp-4.2.4p7.tar.gz
```

2. Enter extracted directory, and configure

```
[liuc@QY-SVN ntp-4.2.4p7]$ cd ntp-4.2.4p7
```

```
[liuc@QY-SVN ntp-4.2.4p7]$ ./configure --prefix=/home/liuc/am335x/ntp
--exec-prefix=/home/liuc/am335x/ntp
--host=arm-arago-linux-gnueabiCC=arm-arago-linux-gnueabi-gcc
```

Remark: Modify [/home/liuc/am335x/ntp] to the directory you want to install.

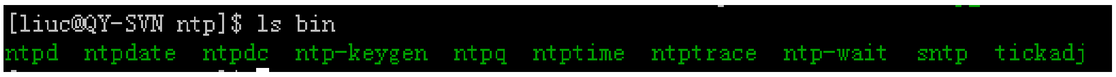
3. Compile and install

```
[liuc@QY-SVN ntp-4.2.4p7]$ make
[liuc@QY-SVN ntp-4.2.4p7]$ make  install
```

4. Generate 3 files in [/home/liuc/am335x/ntp] directory after installation.

bin lib(empty) man.

Command file in [bin]directory is as follows:



```
[liuc@QY-SVN ntp]$ ls bin
ntpd  ntpdate  ntpdc  ntp-keygen  ntpq  ntptime  ntptrace  ntp-wait  sntp  tickadj
```

picture 43

5. Copy command file [bin] directory to [bin] directory in mainboard, or copy all command files to it. The command which ntp clients needs is ntpdate ,The command which ntp server needs is ntpd.

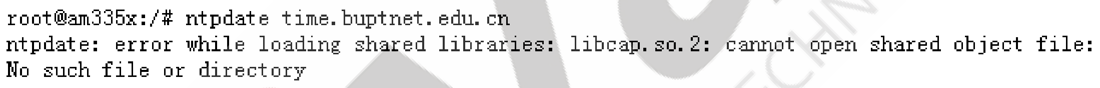
8.4 Ntp Clients

1. Confirm network connectivity normally.

2. Copy [ntddate] command to mainboard, then execute the following command.

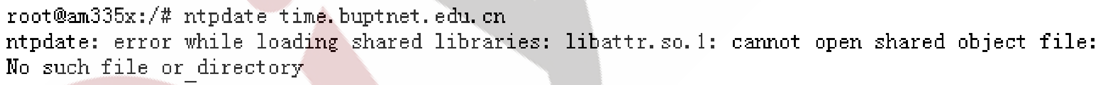
```
root@am335x:/# ntpdate    time.buptnet.edu.cn (or ntpdate 202.112.10.60)
```

Remark: time.buptnet.edu.cn is NTP server （IP: 202.112.10.60） in Beijing time, appear the following question:



```
root@am335x:/# ntpdate time.buptnet.edu.cn
ntpdate: error while loading shared libraries: libcap.so.2: cannot open shared object file:
No such file or directory
```

picture 44

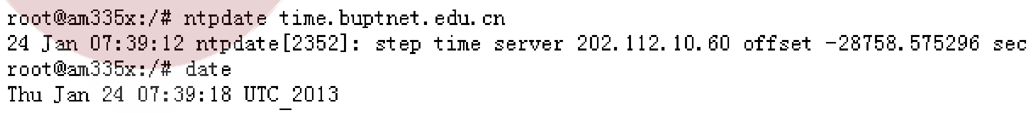


```
root@am335x:/# ntpdate time.buptnet.edu.cn
ntpdate: error while loading shared libraries: libattr.so.1: cannot open shared object file:
No such file or directory
```

picture 45

Solution: Copy [libcap.so.2] and [libattr.so.1] in [arm-arago-linux-gnueabi/usr/lib] directory which installed the cross-compiler tool to [/lib] directory.

3. Finish time synchronization after copying the above two files.



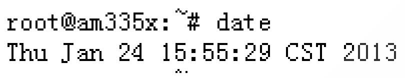
```
root@am335x:/# ntpdate time.buptnet.edu.cn
24 Jan 07:39:12 ntpdate[2352]: step time server 202.112.10.60 offset -28758.575296 sec
root@am335x:/# date
Thu Jan 24 07:39:18 UTC 2013
```

picture 46

Remark: Display UTC time after check by [date] command, CST=UTC+8,so need set time zone.

4. Set time zone to CST time

Copy [Shanghai] file in [/usr/share/zoneinfo/Asia] system directory of [redhat] or [ubuntu] to [/etc] directory in mainboard and change its name to localtime.



```
root@am335x:~# date
Thu Jan 24 15:55:29 CST 2013
```

picture 47

5. Use the command if want to display UTC time.

```
date   –u
```

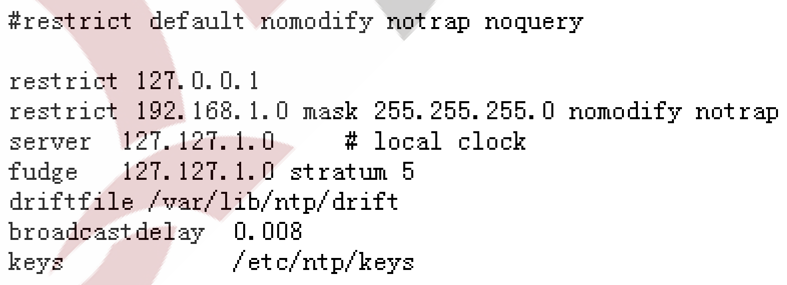
8.5 ntp server

1. Confirm the ntp sever in mainboard and system in ntp client connect network normally.

2. Edit [/etc/ntp.conf] file in mainboard which need to develop server.

Execute the command:

```
vi   /etc/ntp.conf      Execute command: vi   /etc/ntp.conf
```



```
#restrict default nomodify notrap noquery

restrict 127.0.0.1
restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
server  127.127.1.0     # local clock
fudge   127.127.1.0 stratum 5
driftfile /var/lib/ntp/drift
broadcastdelay  0.008
keys            /etc/ntp/keys
```

picture 48

3. Then start ntp server

```
ntpd   -c /etc/ntp.conf
```

4. Execute the following command in client system after the ntp server has started 5-10 minutes , or the time synchronization will fail.

```
    ntpdate    192.168.1.48    (Modify 192.168.1.48    to the server’s IP address)
```

Hangzhou Qiyang Intelligent Technology Co., Ltd

Tel: 86 -571-87858811 / 87858822

Fax: 86-571-89935912

Technology Support: 86-571-89935913

E-MAIL: supports@qiyangtech.com

Website: http://www.qiytech.com

Address: 5F, Building 3A, NO.8 Xiyuanyi Road, West Lake Science Park, Hangzhou, China

Post Code: 310030