SBC60A5 Linux User Manual

Rev. 1.0

Release: 2014-06-10

Thank you for purchasing our product,
We will do better because of you.

Embest Info&Tech Co., LTD.
Tower B 4/F, Shanshui Building, Nanshan Yungu Innovation Industry Park, Liuxian Ave. No. 1183, Nanshan District, Shenzhen, Guangdong, China (518055)
Tel: +86-755-25635626-863/865/866/867
Ext: 863/865/866/867
Fax: 0755-25635626-666
http://www.embest-tech.com/

Revision Record

Date	Revision Version	CR ID/Defect ID	Section Number	Change Description	Author
2014-06-10	1.0				

Table of contents

Linux System

Chapter 1 Overview of Linux System

The SBC60A5 BSP package is mainly used to customize and generate Linux operating system which runs on the SBC60A5 hardware platform. You can do secondary development based on the development board. Table 1-1 shows what the BSP package provided by the CDROM.

Table 1-1 Specification of the SBC60A5 BSP

Name		Remark
Bootloader	Bootstrap	DATA FLASH
	U-boot	DATA FLASH
		Support SAM-BA/SD/USB/network to download, support to boot kernel and file system
kernel	Linux-3.10.0	Support various kinds of system files such as INITRD, EXT2, EXT3, EXT4, FAT, NFS, JFFS2, UBIFS and so on.
Device Driver	Serial	1 debug port, 4 serial ports
	RTC	RTC module integrated with SAMA5D3x
	NET	10/100/1000M Ethernet card driver
	Flash	NANDFLASH driver , DATAFLASH driver
	LCD	LCD driver, support size: 480x272, 640x480, 800x480, 800x600
	Touch Screen	Integrated Resistive Touch screen controller
	USB Host	3 USB Host Ports Driver, One of them used as OTG
	USB Device	1 USB Device Port (OTG) Driver
	Watchdog	Integrated Watchdog Module Driver
	SD Card	SD Card/MMC Driver
	CAN bus	1 Channel of Integrated CAN bus
	RS485	An extended external RS485 bus
	LED	2 LEDs Driver
	Buzzer	1 Buzzer Driver
	Sound	WM8904 Codec Audio Driver
	Button	2 User Buttons Driver
	GPIO	16 GPIO Driver
	ADC	2 Channels of 12 bit ADC Driver
Root file system	UBIFS	A r/w File System, Support Data Compression
	EXT4	The Next Generation of the EXT2/EXT3 Filesystem
Graphics	Qt	Version 4.8.5

Chapter 2 System Booting Method

Note: If you use the HyperTerminal on the PC when boot SBC60A5, please configure the terminal program for:
Baud rate:115200
Data: 8bit;
Parity: None
Stop: 1bit
Flow control: None

2.1 Booting from DATAFLASH

The development board supports to boot form DATAFLASH. Each development board comes pre-loaded with some code in DATAFLASH; please refer to the chapter 5 to update the system image into DATAFLASH if you want. After updating the image, connect the board to PC with debug serial port, open the hyper terminal and power on, then on the terminal window some system information will be printed.

2.2 Setting the type of LCD Screen

There are four types of LCD screen the development board supports, including 480x272, 640x480, 800x480 and 800x600. Please refer to the table 2-1 to set the right values of JP1 and JP2.

Table 2-1

JP2	JP1	LCDTYPE	SIZE
CLOSE	CLOSE	4.3 inch	480x272
CLOSE	OPEN	5.6 inch	640x480
OPEN	CLOSE	7.0 inch	800x480
OPEN	OPEN	10.4 inch	800x600

Chapter 3 Linux Tests

There is the path related to the test routines:

```
home/app/
|-- adc                    ADC Test Demo
|-- buzz                   Buzzer Test (Ring once)
|-- canbus                 Canbus Test Demo
|     |-- candump
|     `-- cansend
|-- com                    Serial Port and RS485 Test Demo
|--  evtest                 Test Demo for event device(such as button/keypad/touchscreen)
|-- gpio                   GPIO Test Demo
|-- lcd                    LCD Test Demo
|-- lcdswitch              LCD Backlight Test Demo
`-- led                    LED Test Demo (LED Flashing)
```

3.1 Touch Screen Test

For more information about touch screen hardware, please refer to "SBC60A5 Hardware Manual".

1. Enter the following command for the touch screen calibration:

```
root@SAMA5D3x:/# ts_calibrate
```

Follow the on-screen prompts, click the "+" icon five times to complete the calibration.

2. Upon completion of calibration, enter the following command for touchscreen test:

```
root@SAMA5D3x:/# ts_test
```

Follow the on-screen prompts, you can try "draw point" or "draw line" for test.

Note: Press <Ctrl+C> to stop the test.

3.2 LCD Test

For more information about LCD Screen hardware, please refer to "SBC60A5 Hardware Manual".

Please refer to Table 2-1 to set the values of JP1 and JP2 according to the size of your LCD Screen. Embest Logo will be displayed on the LCD screen if you boot the system correctly. Enter the Linux command console and type the following command, RGB and mixed-color image will be displayed.

```
[root@SBC60A5:/]# /home/app/lcd
```

3.3 Backlight test

Type the following command:

```
root@SAMA5D3x:/# /home/app/lcdswitch 5
```

If it runs successful, the backlight brightness will be reduced significantly. The command sets the backlight brightness level 0-9, and value overrange is invalid.

Turn off the backlight

```
root@SAMA5D3x:/# /home/app/lcdswitch off
```

Turn on the backlight

```
root@SAMA5D3x:/# /home/app/lcdswitch on
```

3.4 NET Test

There is a MACB 10/100/1000M Ethernet on the board. Connect the target board to network and type the following commands:

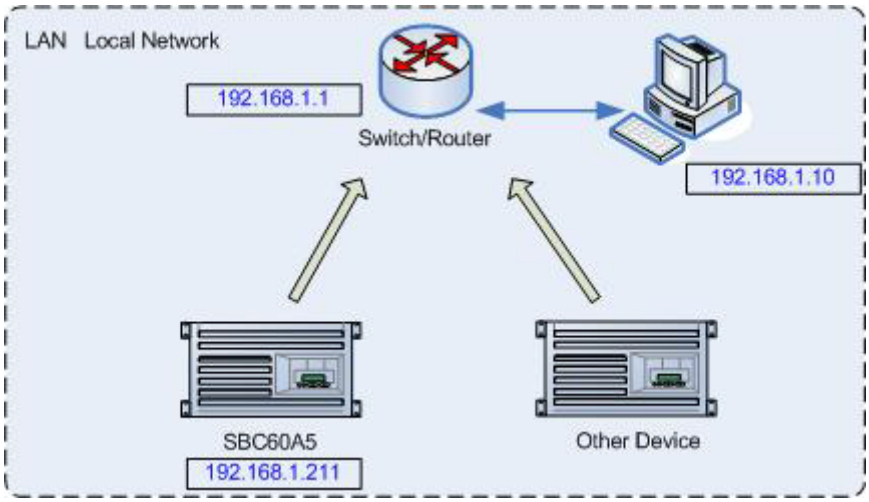


Figure 3.4.1 Network Configuration

```
root@SAMA5D3x:/# ifconfig eth0 192.168.1.211
root@SAMA5D3x:/# ping 192.168.1.10
PING 192.168.1.10 (192.168.1.10): 56 data bytes
64 bytes from 192.168.1.10: icmp_seq=0 ttl=64 time=0.5 ms
64 bytes from 192.168.1.10: icmp_seq=1 ttl=64 time=0.3 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=64 time=0.3 ms
64 bytes from 192.168.1.10: icmp_seq=3 ttl=64 time=0.3 ms
64 bytes from 192.168.1.10: icmp_seq=4 ttl=64 time=0.3 ms
64 bytes from 192.168.1.10: icmp_seq=5 ttl=64 time=0.3 ms
64 bytes from 192.168.1.10: icmp_seq=6 ttl=64 time=0.3 ms

--- 192.168.1.10 ping statistics ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max = 0.3/0.3/0.5 ms
~ $
```

Note: Press the <Ctrl+C> to stop the test.

3.5 UART Test

There are 5 serial ports on the target board, including ttyS0, ttyS1, ttyS2, ttyS3 and ttyS6. While ttyS6 is used for debug, and ttyS3 is used for RS485. For more information, please refer to "SBC60A5 Hardware Manual".

Table 3-1 Serial Port Map

COM 编号	tty 编号	特性
COM0	ttyS0	Support Hardware Flow Control
COM1	ttyS1	3 wired
COM2	ttyS2	Support Hardware Flow Control

COM3	ttyS3	RS485
DEBUG	ttyS6	Debug Port

Type the following command for test:

```
root@SAMA5D3x:/# /home/app/com -f -d /dev/ttyS1 -s 1234567890 -b 115200
```

Indicator:the serial ports send messages as while as receive. The parameters are described as below:

-d --device The device ttyS[0-3] or ttyEXT[0-3]
-s --string Write the device data
-b --speed Set speed bit/s
-f --flow Hardware flow control switch

Note: Press the <Ctrl+C> to stop the test.

3.6 CAN Bus Test

There is an extended CAN Bus device on the target board. For more information about the hardware, please refer to "SBC60A5 Hardware Manual".

1. Set the baud rate and enable the canbus channel:

```
root@SAMA5D3x:/# ifconfig can0 down; ip link set can0 type can bitrate 800000; ip link set can0 type can restart-ms 20; ifconfig can0 up
```

2. Launch the reciever program:

```
root@SAMA5D3x:/# /home/app/canbus/candump can0
```

3. Launch the transmitter program:

```
root@SAMA5D3x:/# /home/app/canbus/cansend can0 "5A1#1122334455667788"
```

When reciever recieves data, it will print information below:

```
can0  5A1  [8]  11 22 33 44 55 66 77 88
```

There is only 1 channel of canbus, so we need another board or certain canbus device to cooperate for the testing.

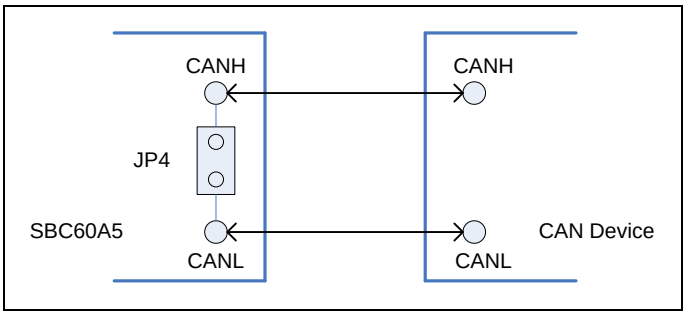


Figure 3.6.1 CAN Connection (close the Jump JP4)

Note: Press the <Ctrl+C> to stop the test.

3.7 RS485 Test

The serial device /dev/ttyS3 is occupied by the RS485 on the target board. The test method is similar to UART test, while there is some difference between them in the hardware connection.

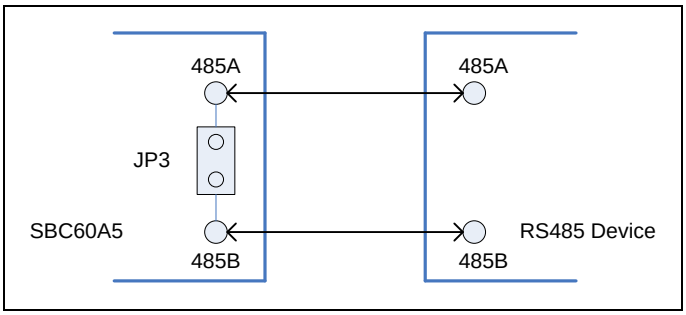


Figure 3.7.1 RS485 Connection (JP3 used to enable termination resistor)

RS485 termination which is to eliminate signal reflections on communication cables is not needed if the data line is shorter than 300m. While at the case of long distance network, it's necessary to enable the termination placed at the ends of the RS485 device.

3.8 USB HOST Test

The development board is equipped with a USB Host connector. The USB device is needed for test (e.g.: a USB disk). Something similar to the following information will be displayed after inserting the USB disk

```
usb 1-1: USB disconnect, address 2
usb 1-1: new full speed USB device using at91_ohci and address 3
usb 1-1: configuration #1 chosen from 1 choice
scsi2 : SCSI emulation for USB Mass Storage devices
scsi 2:0:0:0: Direct-Access     Generic  USB  SD Reader   0.00 PQ: 0 ANSI: 2
sd 2:0:0:0: [sda] 7744512 512-byte hardware sectors (3965 MB)
sd 2:0:0:0: [sda] Write Protect is off
sd 2:0:0:0: [sda] Assuming drive cache: write through
sd 2:0:0:0: [sda] 7744512 512-byte hardware sectors (3965 MB)
sd 2:0:0:0: [sda] Write Protect is off
sd 2:0:0:0: [sda] Assuming drive cache: write through
 sda: sda1
sd 2:0:0:0: [sda] Attached SCSI removable disk
sd 2:0:0:0: Attached scsi generic sg1 type 0
```

Above indicates that the USB disk device is identified as sda1. Follow the commands as below for test.

1. USB disk could be operated after mounted. For example, you can mount **/dev/sda1** to the /mnt directory and specify it as VFAT format.

```
[root@SBC60A5:/]# mount -t vfat /dev/sda1 /mnt/
```

2. Enter the directory and view files

```
[root@SBC60A5:/]# cd /mnt/
[root@SBC60A5:/mnt]# ls
```

3. Uninstall the USB disk

```
[root@SBC60A5:/mnt]# cd /
[root@SBC60A5:/]# umount /mnt/
```

The root file system in CD-ROM mounts the storage device in the /media directory automatically

according to the UDEV rules.

> *Note: U disk is generally recognized as **/dev/sdax, /dev/sdbx** device according to the system prompts.*

3.9 USB Device Test

Let't take serial gadget for example to introduce usb device test.

1. Connect upper usb slot and PC usb slot with A type USB cable.

2. Run command:

```
root@SAMA5D3x:/# modprobe g_serial
```

The rootfs provided in CDROM loads g_serial driver automatically during system booting. Manual operation is needless.

3. Check all loaded modules driver:

```
root@SAMA5D3x:/# lsmod
Module                  Size  Used by
usb_f_acm               3999  1
u_serial                7654  1 usb_f_acm
g_serial                1726  0
libcomposite           26068  2 usb_f_acm,g_serial
configfs               19188  3 usb_f_acm,libcomposite
```

4. On the PC side, a new device alert will pop-up, install the driver.

① Enter the Linux source directroy, and copy ***Documentation/usb/linux-cdc-acm.inf*** to PC windows system.

② Install the driver and point the path to ***linux-cdc-acm.inf*** directory.

5. Install successfully, a new COM device will show on Device Manager Window.

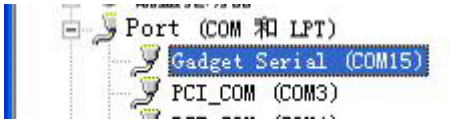
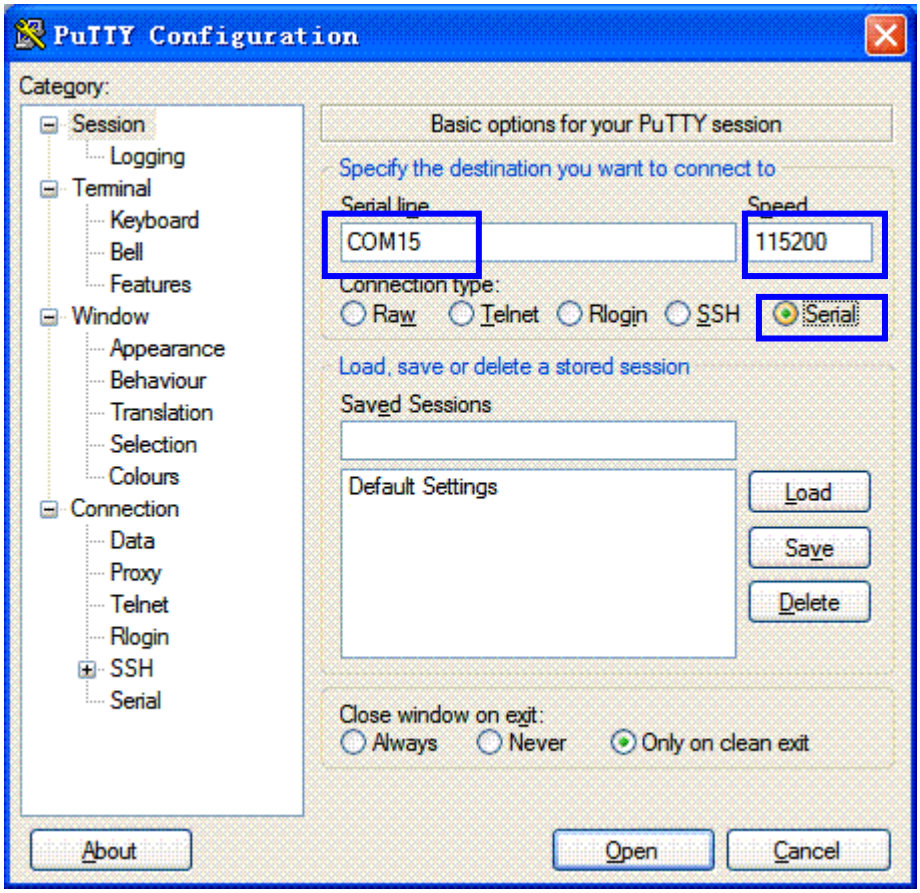
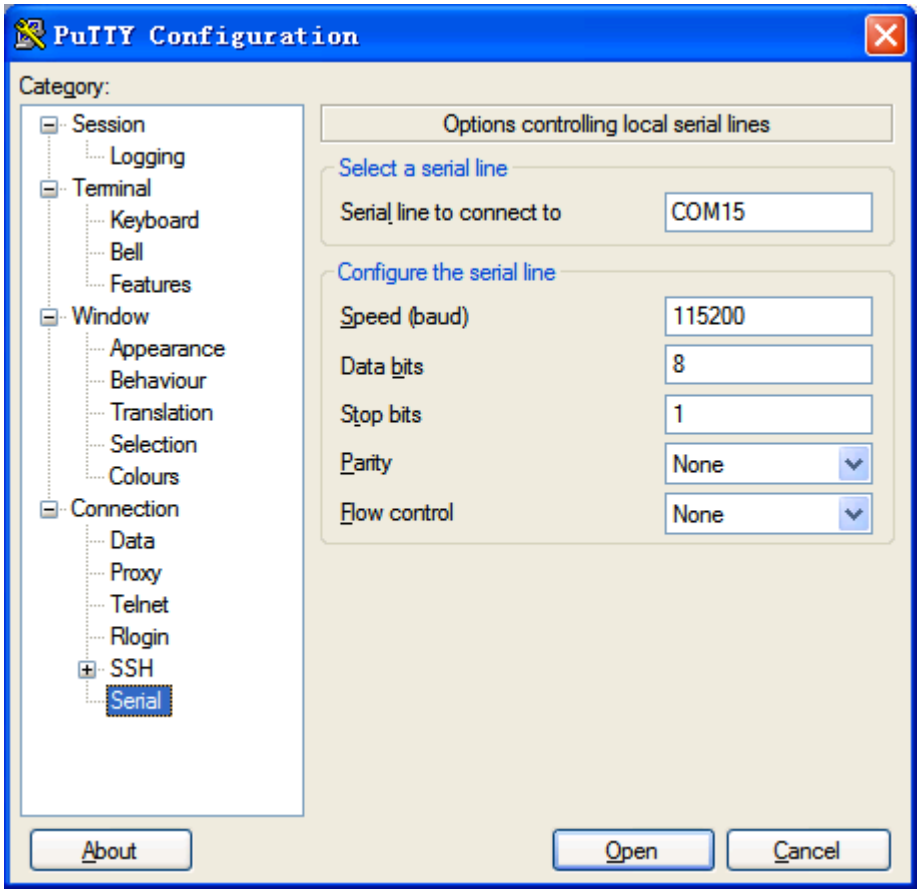


Figure 3.9.1 Serial Gadget Device

6. Configure Putty.


PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Serial
Basic options for your PuTTY session
Specify the destination you want to connect to
Serial line
Speed
COM15
115200
Connection type:
Raw
Telnet
Rlogin
SSH
Serial
Load, save or delete a stored session
Saved Sessions
Default Settings
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit
About
Open
Cancel

(a)



(b)

Figure 3.9.2 Putty Configuration Guide

7. Click ***Open***, view the serial port monitor interface.

Figure 3.9.3 Putty Serial Port Viewer

8. Run serial port test command on SBC60A5:

```
root@SAMA5D3x:/# /home/app/com -d /dev/ttyGS0
SEND: 1234567890
SEND: 1234567890
```

```
SEND: 1234567890
```

Figure 3.9.4 Putty Serial Viewer

3.10 RTC Test

An integrated RTC module, RTC which is used to store and restore system time could be operated as below:

1. View system clock information

```
root@SAMA5D3x:/# date
Tue Jun 10 12:10:36 UTC 2014
```

2. Set the time. For example: 2014-06-10 12:10

```
root@SAMA5D3x:/# date -s "2014-6-10 12:10"
Tue Jun 10 12:10:00 UTC 2014
```

3. Set the Hardware Clock to the System Time

```
root@SAMA5D3x:/# hwclock -w
```

4. Display the current time

```
root@SAMA5D3x:/# hwclock -r
Tue Jun 10 12:10:34 2014    0.000000 seconds
```

5. Set the System Time from Hardware Clock

```
root@SAMA5D3x:/# hwclock -s
root@SAMA5D3x:/# date
Tue Jun 10 12:11:25 UTC 2014
```

3.11 SD Card Test

If you insert the SD card into the card slot, the following information will be printed to the terminal.

```
root@SAMA5D3x:/# mmc1: new SD card at address 0002
mmcblk0: mmc1:0002 N/A     489 MiB
mmcblk0: p1
```

Above indicates that the SD card is identified as mmcblk0p1. Follow the commands as below for test.

1. SD card could be operated after mounted. For example, you can mount mmcblk0p1 to the

/mnt directory and specify it as VFAT type.

```
root@SAMA5D3x:/# mount -t vfat /dev/mmcblk0p1 /mnt/
```

2. Enter the directory and view files

```
root@SAMA5D3x:/# cd /mnt/
root@SAMA5D3x:/mnt# ls
```

3. Uninstall SD card

```
root@SAMA5D3x:/mnt# cd /
root@SAMA5D3x:/# umount /mnt/
```

The root file system in CDROM mounts the storage device under the /media directory automatically according to the UDEV rules.

Note: SD card is generally recognized as mmcblk0p1 device according to the system prompts.

3.12 LED Test

There are 3 LED lamps on the target board for indication. D11 indicates power supply; D12 indicates the operation state of the system; D13 is for user operation. Follow the commands as below to test the rest lights:

1. Application test

```
root@SAMA5D3x:/# /home/app/led
```

D13 is flashing.

2. Manual test

Turn on D13

```
root@SAMA5D3x:/# echo '1' >/sys/class/leds/d13/brightness
```

Turn off D13

```
root@SAMA5D3x:/# echo '0' >/sys/class/leds/d13/brightness
```

3.13 Buzzer test

The development board is equipped with buzzer, follow the command for test:

1. Application test

```
root@SAMA5D3x:/# /home/app/buzz
```

2. Manual test

Buzzer rings

```
root@SAMA5D3x:/# echo '1' >/sys/class/leds/buzz/brightness
```

Buzzer stop

```
root@SAMA5D3x:/# echo '0' >/sys/class/leds/buzz/brightness
```

3.14 GPIO test

The general-purpose port GPIOs are connected to ***U59***:

U59		
PA16		PE8
PA17		PE9
PA18		PE10
PA19		PE11

PA20		PE12
PA21		PE13
PA22		PE14
PA23		PE15

Usage of GPIO demo program:

```
root@SAMA5D3x:/# /home/app/gpio
Usage:
    ./gpio PA16        read PA16 input value
    ./gpio PA16 1      set PA16 output 1
```

Connect PA18 and PE10, set PA18 output and read PE10 input value:

```
root@SAMA5D3x:/# /home/app/gpio PA18 1        #PA18 output high
root@SAMA5D3x:/# /home/app/gpio PE10          #Read PE10 input value
1                                             #High level printed
```

```
root@SAMA5D3x:/# /home/app/gpio PA18 0        #PA18 output low
root@SAMA5D3x:/# /home/app/gpio PE10          #Read PE10 input value
0                                             #Low level printed
```

3.15 ADC Test

The 2 ADC channels on the target board, AD7 and AD11, located at ***U60***.

Then type the following command:

```
root@SAMA5D3x:/# /home/app/adc
ADC7:   0
ADC11: 250
```

Apply different voltage to the adc pin, the data printed will change.

The adc detection range is 0 ~ 3.3V. The voltage input calculation formula as below:

$$V_{in} = \frac{value}{4096} \times 3.3(\mathrm{V})$$

[NOTE] value: data read form adc controller (resolution 12 bits).

3.16 Button Test

Type the following commands to test the KEY1 and KEY2:

```
root@SAMA5D3x:/# /home/app/evtest /dev/event1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "gpio-keys"
Supported events:
  Event type 0 (Sync)
  Event type 1 (Key)
    Event code 256 (Btn0)
    Event code 257 (Btn1)
Testing ... (interrupt to exit)
```

Click KEY1 and KEY2 key, something similar to the following information will be displayed:

```
Event: time 1643862901.338099, type 1 (Key), code 256 (Btn0), value 1
Event: time 1643862901.338111, ---------- Report Sync ------------
Event: time 1643862901.461921, type 1 (Key), code 256 (Btn0), value 0
```

```
Event: time 1643862901.461928, ---------- Report Sync ------------
Event: time 1643862902.239929, type 1 (Key), code 257 (Btn1), value 1
Event: time 1643862902.239935, ---------- Report Sync ------------
Event: time 1643862902.340413, type 1 (Key), code 257 (Btn1), value 0
Event: time 1643862902.340419, ---------- Report Sync ------------
```

Note: Press the <Ctrl+C> to exit.

3.17 Capture Test

Capture command can help to save the current picture displayed on the LCD. It's convenient to display graphics on PC and mainly used to capture the GUI screen.

```
root@SAMA5D3x:/# fbcat /dev/fb0 Figure.jpg
```

3.18 Audio Output Test

Aplay is an open source music player, supports wav format audio files.

```
root@SAMA5D3x:/# aplay /home/mp3/music.wav
```

Put headphones into ears and enjoy the beautiful songs.

Note: Press the <Ctrl+C> to exit. Perform the kill command to kill the process running in the background.

3.19 Audio Record Test

There is a MIC slot on the board located at ***U66***, install a MIC on it then we can record voice around. Enable the audio input channel:

```
root@SAMA5D3x:/# amixer set "Capture" cap
```

Start recording:

```
root@SAMA5D3x:/# arecord -t wav -f cd test.wav
```

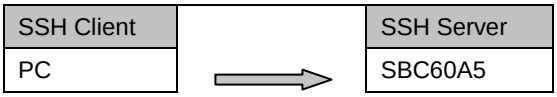
Make sound approatching the MIC head. Press Ctrl+C to terminate recording.

Play back:

```
root@SAMA5D3x:/# aplay test.wav
```

3.20 SSH Login

Telnet is the most commonly used remote login program. But nowadays, the information security is becomming more and more important. The telnet which uses non encrypted connection is replaced by the ssh which uses encrypted one. This section I will show you the specific operation to login SBC60A5 via ssh protocol.



192.168.1.10		192.168.1.211

SSH Server/Client IP Configuration

Client OS: Windows XP3

- Install PuttY (http://www.putty.org).
- Double click putty.exe to start program.
- Configure it:

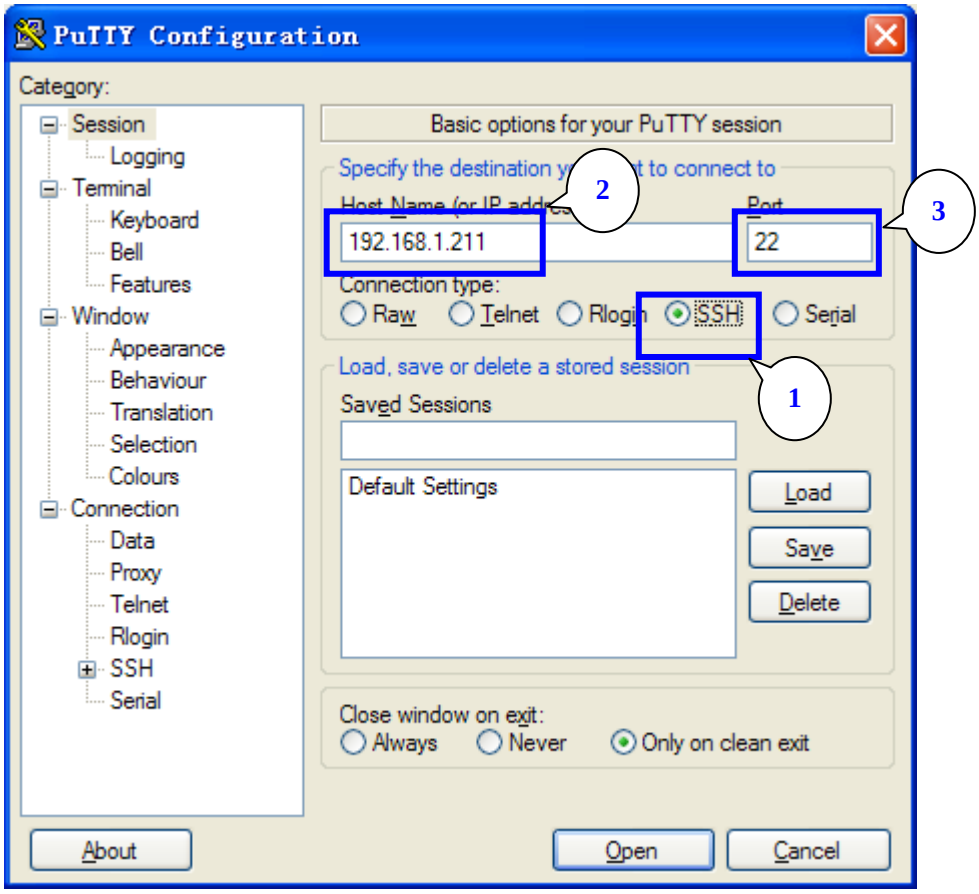


Figure 3.20.1 Putty SSH Configuration

① Choose SSH protocol

② Fill SSH Server IP

③ Fill SSH port number, remain default (22)

- Click ***Open***.
- If connection setup successfully, the dialog may pop-up.

Figure 3.20.2 Putty Security Alert

Click ***Y***, then it won't pop-up at next login.

- Login dialog

Figure 3.20.3 Putty Login dialog

Account : root

Password : None

- If account and password are correct, we can see the SBC60A5 command console.

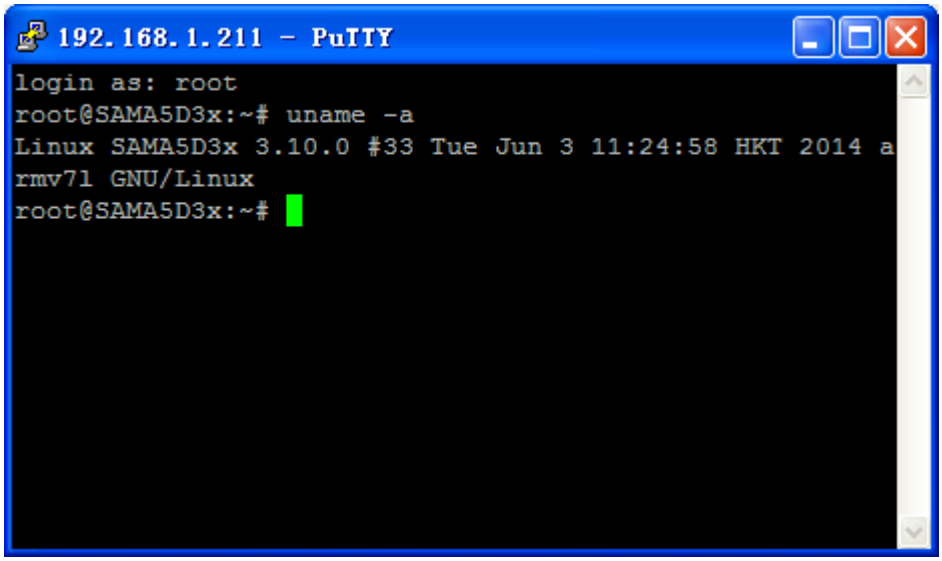


Figure 3.20.4 Putty Shell

Client OS: Linux (Ubuntu 10.04)

- Open terminal console.
- Run command:

root@new-desktop:~# **ssh 192.168.1.211**

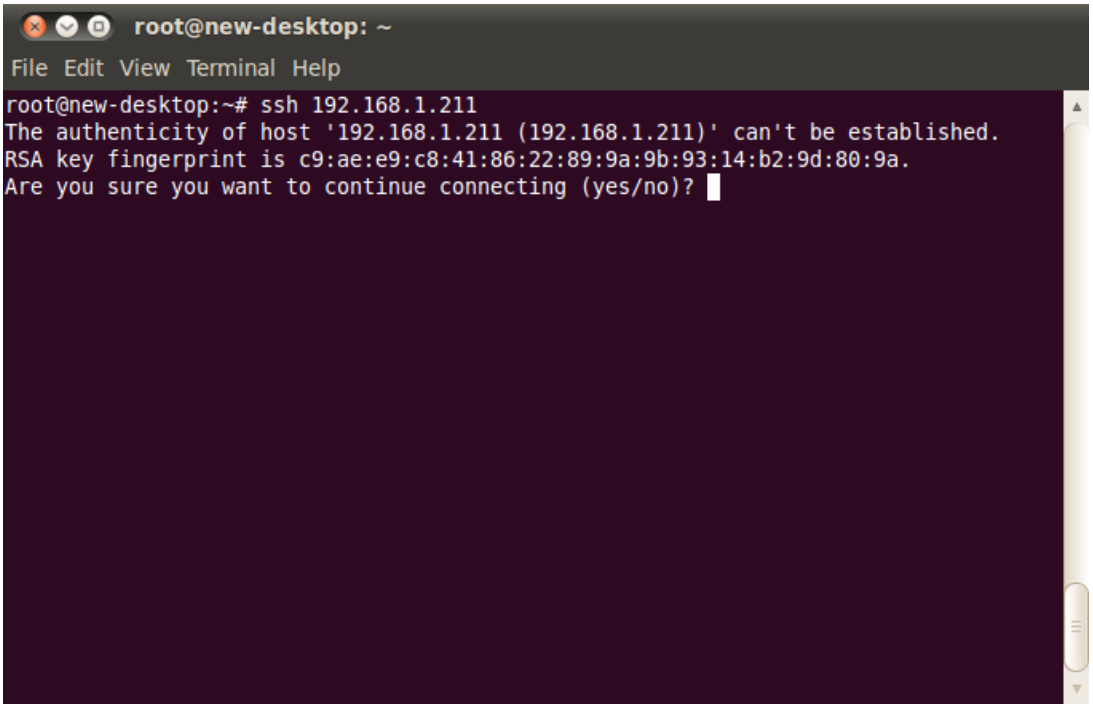


Figure 3.20.5 Ubuntu ssh login

- Input "***yes***", and enter.

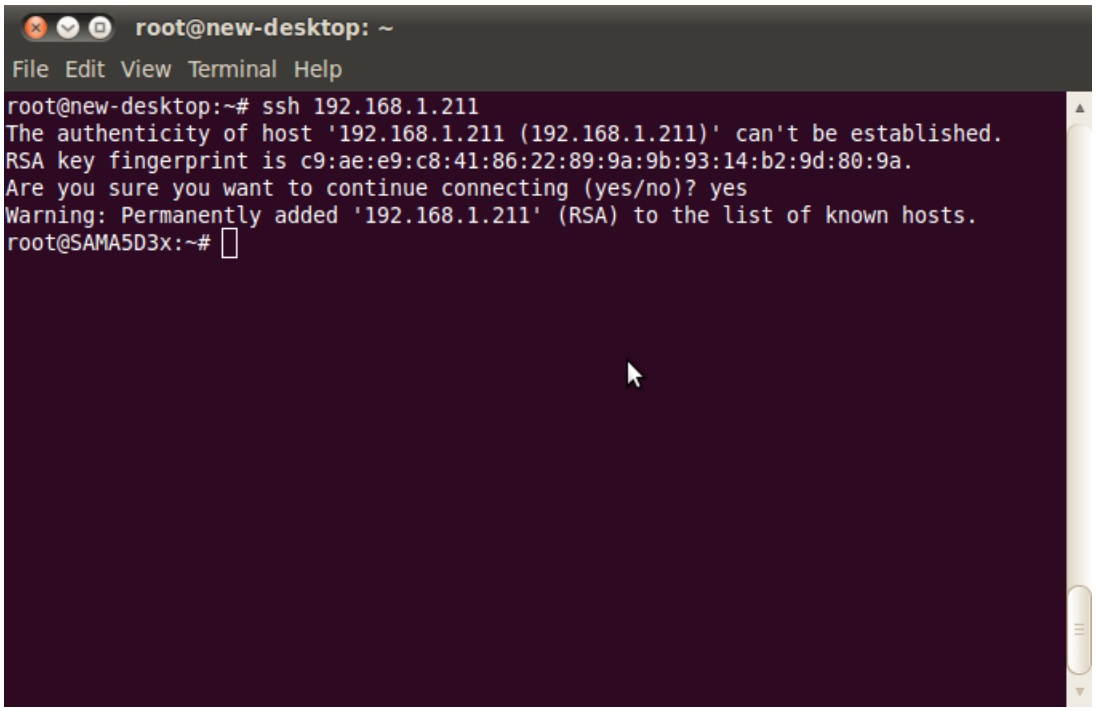


Figure 3.20.6 SSH Login Success

- Run command on target board via SSH.

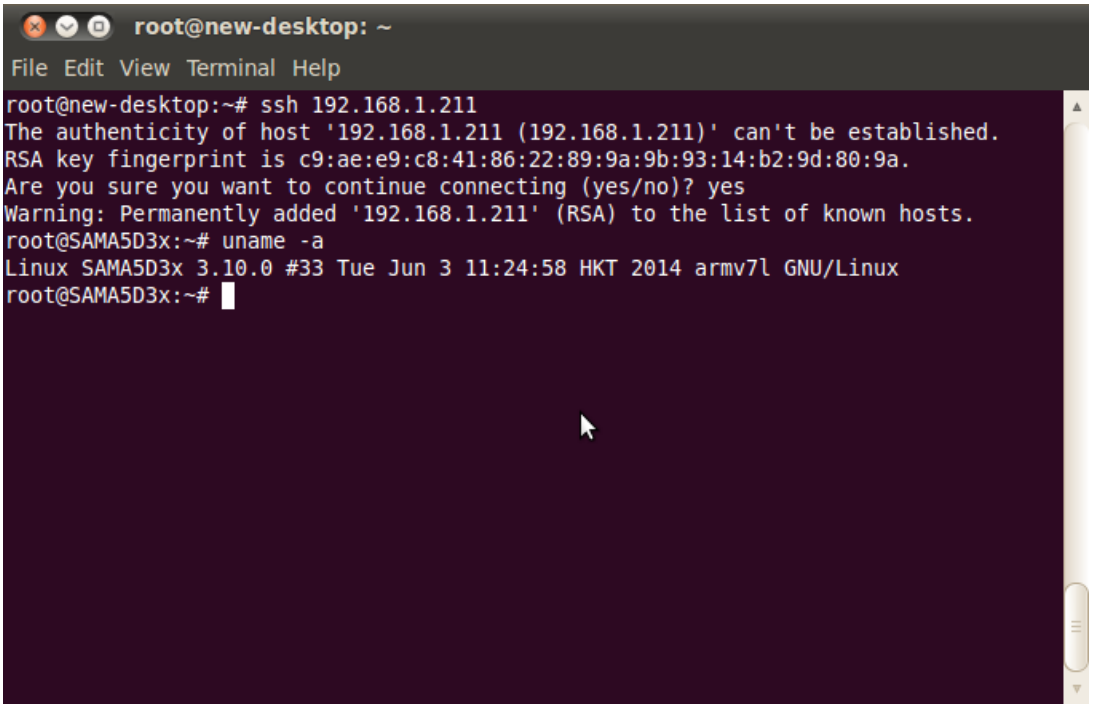


Figure 3.20.7 Run Command via SSH

3.21 Mount Network File System NFS

Access files in remote Linux without uploading and downloading steps, it is a benefit for debugging. Let's setup the NFS sever for Linux system.

- Act as root user

- Edit the file /etc/exports, add the following information at the end of the file

```
/home/nfs *(rw,sync,no_root_squash)
```

/home/nfs Shared directory for sever
* mount directory for clients
no_root_squash mounting the directory, any client will have root privileges

- Save
- Boot NFS and run the command:

```
# /etc/init.d/nfs-kernel-server start
```

- Test NFS on the host PC

```
# mount -o nolock localhost:/home/nfs /tmp
```

The contents of /tmp are exactly the same as that of /nfs without any error message means success.

Test NFS on the ARM board

- Power on and enter the Linux system on the board
- Connect network cable and configure correct IP address
- Type ping and hit <enter> to test, success and go to next, otherwise check the network connection and IP address.
- Type the following command:

```
root@SAMA5D3x:/# busybox mount -o nolock 192.168.1.10:/home/nfs /mnt
```

Mount the /home/nfs on PC to /mnt on the board

- The contents of /mnt are exactly the same as that of /nfs means success, otherwise fail
- Note: Now you have write privilege, any changes will take effect.

3.22 Transfer Files to PC

【Serial Port】Take SecureCRT tool as an example to introduce the serial transmission steps.

The target board receives files from PC

```
root@SAMA5D3x:/# cd /tmp
root@SAMA5D3x:/tmp# rx recvfile
C
```

Make the choice: Transfer → send Xmodem(N)

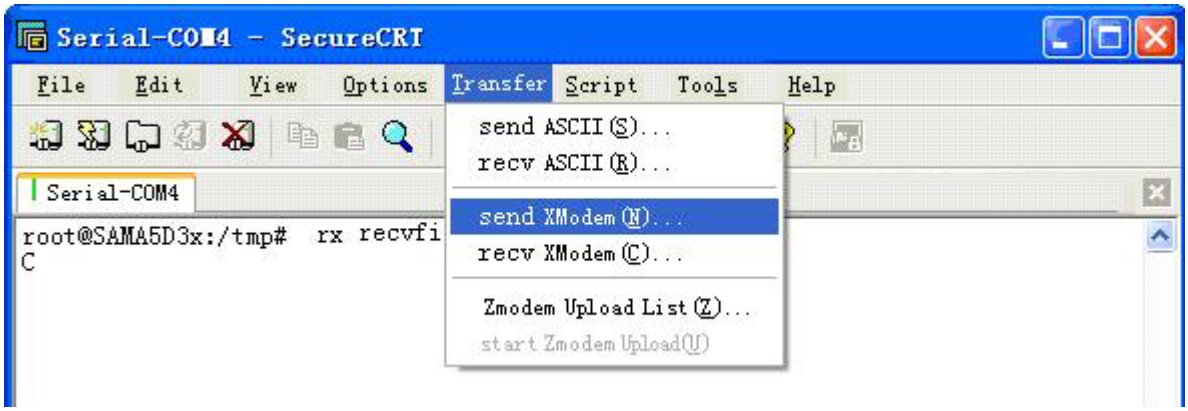


Figure 3.22.1 Choose XModem to Transfer

Dialog:

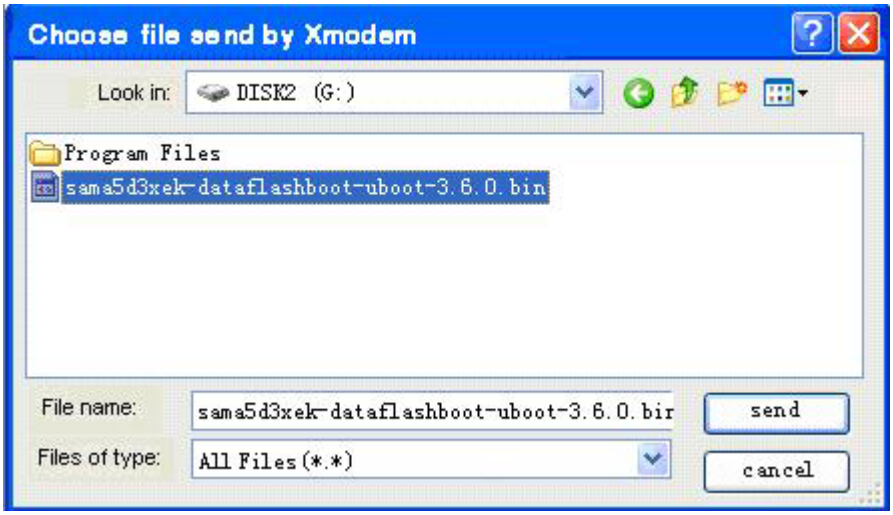


Figure 3.22.2 Select File

Select the file you want to transmit and hit <send>, something similar to the following information will be displayed to the terminal:

```
CC
Starting xmodem sending. Press Ctrl+C to cancel.
Sending sama5d3xek-dataflashboot-uboot-3.6.0.bin...
  100%          4 KB       0 KB/s 00:00:05           0 error(s)
root@SAMA5D3x:/tmp#
```

Finish transmission.

Note: Press the <Ctrl+C> to exit.

Since the transmission speed is slow, only text files and small image files are applicable.

【Network】We can transfer large files via TFTP protocol.

Run the tftp.exe on PC and set the shared directory since the files to be transferred and the received files should be stored in the shared directory. For example, set the G:/ as the shared directory and the file data.bin is for downloading test.

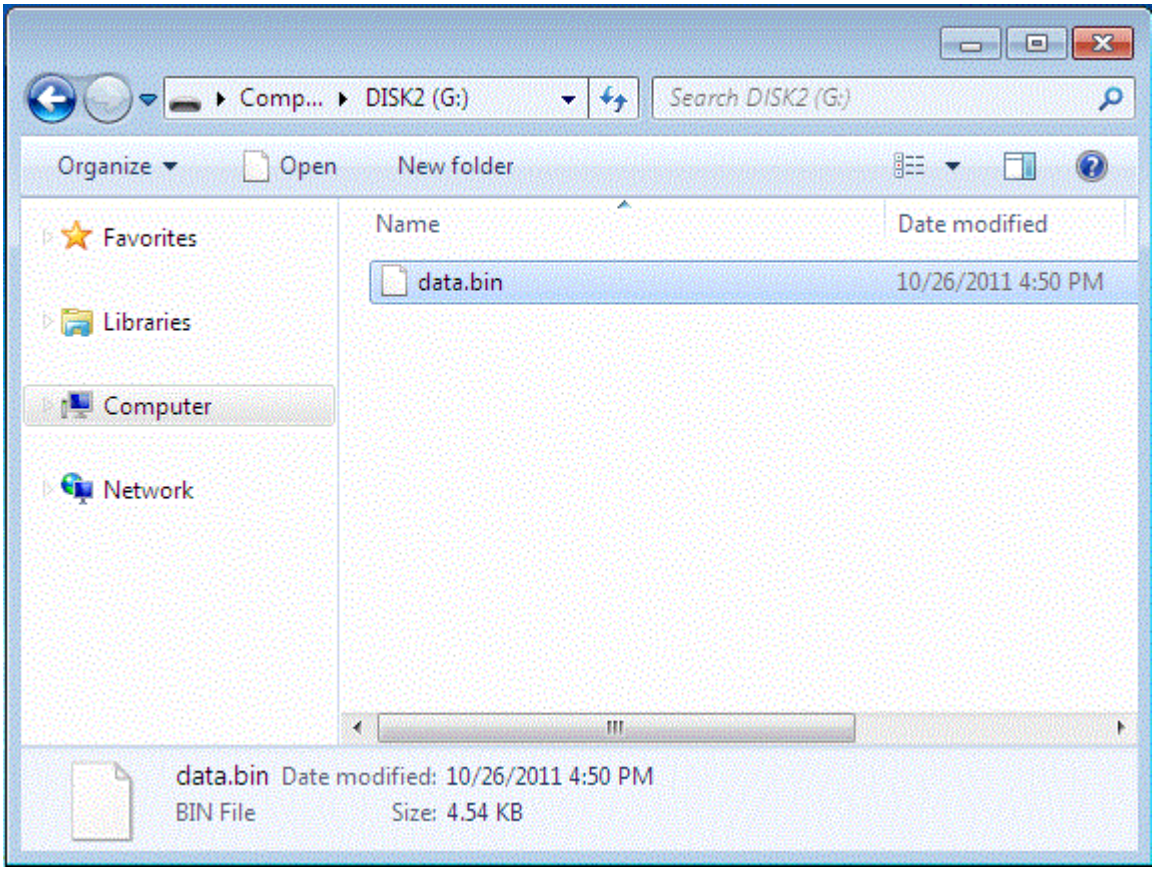


Figure 3.21.3

Set local IP on the target board:

```
root@SAMA5D3x:/tmp# ifconfig eth0 192.168.1.211
```

Type the following commands to download the file data.bin:

```
root@SAMA5D3x:/tmp# tftp -g 192.168.1.10 -r data.bin
root@SAMA5D3x:/tmp# ls -l
-rw-r--r--    1 root     root          4420 Jan  1 00:44 data.bin
```

Download successfully

Rename the file:

```
root@SAMA5D3x:/tmp# mv data.bin data_send.bin
```

Upload file ***data_send.bin***:

```
root@SAMA5D3x:/tmp# tftp -p 192.192.192.71 -l data_send.bin
```

There is the file data_send.bin in the shared directory, means successful operation.

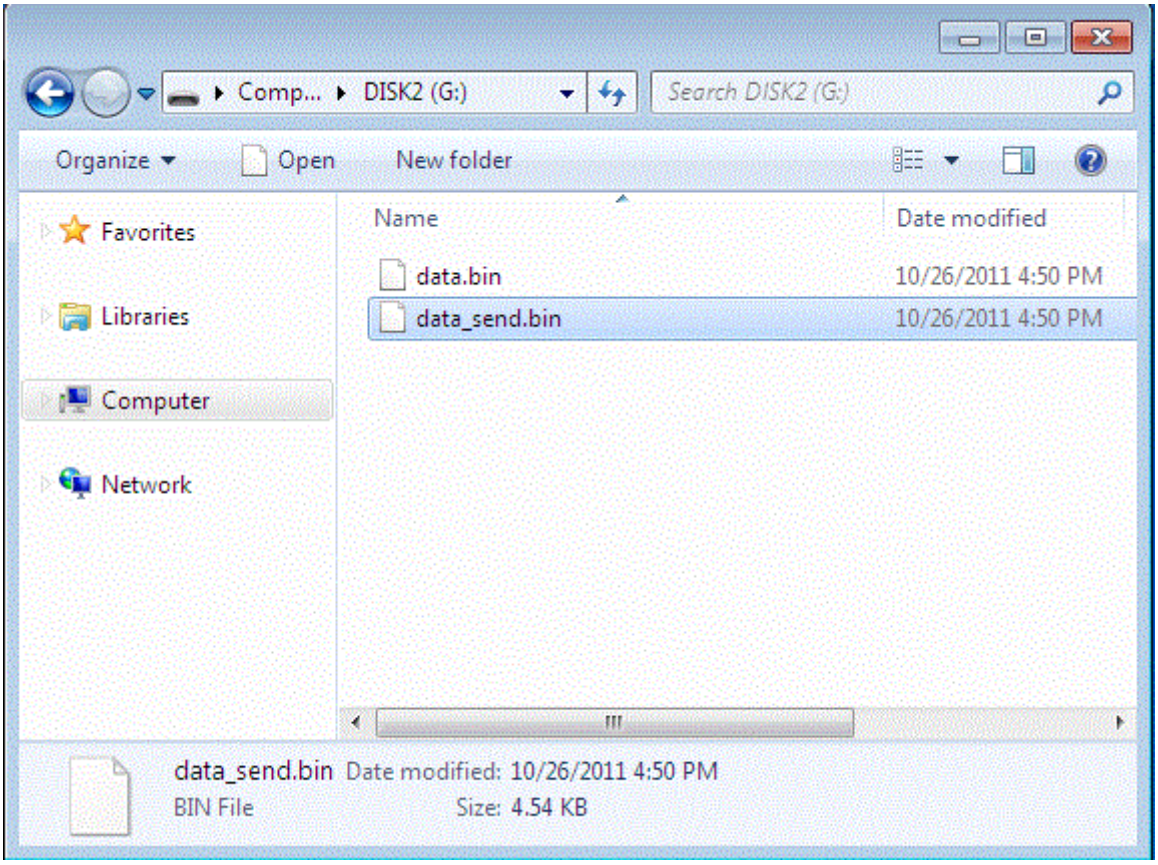


Figure 3.22.4 Upload Successfully

3.23 API Introduction

This is a brief introduction of API, for more detailed information please refer to the source code.

LED-API

LED_API int led_ctrl (char *name, int onoff);	“ledlib.h”

Function : control the state of LED
Parameters : name the name of LED, e.g.”D6”, ”D9” or “D13”
onoff 0:off; 1:on
Return : 0: success, otherwise fail
Example : led_ctrl(“D13”, 1);

Buzzer-API

BUZZ_API int buzz_ctrl (char *name, int onoff);	"buzzlib.h"

Function : control the buzzer
Parameters: name the name of buzzer, there is only one buzzer “buzz”
onoff 0: stop; 1: ring
Return : 0: success otherwise fail
Example : buzz_ctrl ("buzz", ON); buzz_ctrl ("buzz", OFF);

Serial-API

int OpenDev(char *Dev)	“com/main.c”

Function : open the serial device and get its descriptor
Parameters: Dev the string of serial device node, such as “/dev/ttySAC0”
Return : if the value is beyond 0, we get the descriptor of serial port, otherwise failed.

void set_speed(int fd, int speed)	“com/main.c”

Function : set the baud rate for serial
Parameters: fd serial file descriptor
Speed baud rate, such as 115200
Return : none

int set_Parity(int fd,int databits,int stopbits,int parity,int flowctrl)	“com/main.c”

Function : set data, stop, parity and flow control for serial
Parameters: fd serial device descriptor
databits data size (bit)
stopbits stop size (bit)
parity parity type, such as N (none), O (odd), E (even)
flowctrl flow controller switch, 1: enable, 0: disable
Return : 0: success; otherwise fail

size_t read(int fd, const void *buf, size_t nbytes)	<unistd.h>

Function : call system and obtain the data from serial
Parameters: fd serial file descriptor
buf a pointer which points to received data
nbytes the size of data to be written (byte)
Return : if the return value is beyond 0, we get the data size (byte), otherwise failed.

size_t write(int fd, const void *buf, size_t nbytes)	<unistd.h>

Function : call system and send data from serial
Parameters: fd serial file descriptor
buf a pointer which points to the data to be sent
nbytes the size of data that to be sent (byte)
Return : if the value is beyond 0, we will get the data size (byte), otherwise failed.

int close(int fd)	<unistd.h>

Function : call system and close the serial
Parameters: fd serial file descriptor
Return : 0: success; <0: error

GPIO-API

int gpio_init (const char *gpio, enum gpio_direction dir)	"gpiolib.h"

Function : initialize GPIO
Parameters: *gpio GPIO name, such as ”PA16”
dir GPIO direction, such as GPIO_OUTPUT, GPIO_INPUT

Return : 0: success otherwise fail

int gpio_set_direction (const char *gpio, enum gpio_direction dir) "gpiolib.h"

Function : set GPIO direction
Parameters: *gpio GPIO name, such as ”PA16”
dir GPIO direction, such as GPIO_OUTPUT, GPIO_INPUT
Return : 0: success otherwise fail

int gpio_set_output (const char *gpio, int value) "gpiolib.h"

Function : control GPIO to output logic level
Parameters: *gpio GPIO name, such as ”PA16”
value 0: low level; 1: high level
Return : 0: success, otherwise fail

int gpio_get_input (const char *gpio) "gpiolib.h"

Function : obtain the logic level from GPIO
Parameters: *gpio GPIO name, such as ”PA16”
Return : 0/1: the received logic level, otherwise fail

Chapter 4 Setting the Linux Development Environment

The arm-Linux cross development environment should be setup first. The steps to set the development environment in the Ubuntu are as follows, while the steps for other Linux system are similar to those.

Open the terminal console, and login the root account before carrying on the following steps.

4.1 Use The BSP Source Package

Insert the CD, and it is mounted to the directory ***/media/cdrom*** default. Stored in the ***/media/cdrom/02 Linux Kit/02 Tools*** directory, the file named "***tools.tar.bz2***" is the one. Make sure t here is 3GB free disk space at lease.

Type the following commands to install the tool:

```
mkdir -p /home/work/SBC60A5
cd /home/work/SBC60A5
tar -jxvf /media/cdrom/02\ Linux\ Kit/02\ Tools\tools.tar.bz2
```

```
ls /home/work/SBC60A5
build.sh  Image  tools
```

The directory ***Image*** is used to store the target images.

Uncompress bootstrap source:

```
tar -jxvf /media/cdrom/02\ Linux\ Kit/01\ SourceCode/bootloader/at91bootstrap.tar.bz2
```

Uncompress u-boot source:

```
tar -jxvf /media/cdrom/02\ Linux\ Kit/01\ SourceCode/bootloader/u-boot-at91.tar.bz2
```

Uncompress Linux source:

```
tar -jxvf /media/cdrom/02\ Linux\ Kit/01\ SourceCode/kernel/linux-at91.tar.bz2
```

Uncompress rootfs source:

```
mkdir rootfs
tar -jxvf /media/cdrom/02\ Linux\ Kit/01\ SourceCode/rootfs/rootfs.tar.bz2 -C rootfs
```

Copy the demo program source:

```
mkdir app
cp -f /media/cdrom/02\ Linux\ Kit/01\ SourceCode/applicantion/* app/
```

There are contents under the working directory.

```
ls /home/work/SBC60A5
app  at91bootstrap  build.sh  Image  linux-at91  rootfs  tools  u-boot-at91
```

The usage of the build tool script:

```
./build.sh
```

```
usage:
./build.sh all          build all images              #Build all images including cross-compiler
./build.sh compiler     install cross-compiler        #Install cross-compiler
./build.sh boot         build bootsrap                #Build bootstrap
./build.sh u-boot       build u-boot                  #Build u-boot
./build.sh linux        build linux                   #Build linux
./build.sh rootfs       build rootfs                  #Build rootfs
```

4.2 Install the Cross-compiler

Run command below:

```
./build.sh compiler
==========================================
INFO: Install cross-compiler ...
==========================================
...
sama5d3xek/usr/bin/crossscripts/depmodwrapper
sama5d3xek/usr/bin/crossscripts/libtoolize
INFO: cross-compiler DONE!
```

It untar the cross tool ***tools/arm-linux/arm-linux-4.8.1-poky.tar.bz2*** to ***/work/poky/***.

4.3 Set Cross-compiler Environment

Type the following command to define the path of compiler:

```
export PATH=/work/poky/build-atmel/tmp/sysroots/i686-linux/usr/bin/cortexa5hf-vfp-poky-linux-gnueabi:$PATH
```

Type the following command to check:

```
arm-linux-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gcc
…
gcc version 4.8.1 (GCC)
```

If your version number matches the information provided by words in blue, that means success.

Note:You can copy it to ".bashrc" file to add the PATH automatically when OS is booted.

4.4 Compile the System

4.4.1 Build Bootstrap

The board supports booting from DATAFLASH, now we build the first stage bootloader.

Enter to the working directory:

```
cd /home/work/SBC60A5
```

Start to build:

```
./build.sh boot
================================================
INFO: Build Bootstarp ...
================================================
...
Size of sama5d3xek-dataflashboot-uboot-3.6.0.bin is 9800 bytes
[Succeeded] It's OK to fit into SRAM area
INFO: COPY binaries/sama5d3xek-dataflashboot-uboot-3.6.0.bin -> /home/work/SBC60A5/Image/sama5d3xe
k-dataflashboot-uboot-3.6.0.bin : OK
```

If it goes right, the target images ***boot.bin*** and ***sama5d3xek-dataflashboot-uboot-3.6.0.bin*** will be generated under directory ***Image***.

NOTE: boot.bin used for SD booting；sama5d3xek-dataflashboot-uboot-3.6.0.bin used for DATAFLASH booting。

4.4.2 Build U-Boot

U-Boot is the second stage bootloader, build command as below:

```
./build.sh u-boot
================================================
INFO: Build u-boot ...
================================================
...
arm-linux-objcopy -O binary hello_world hello_world.bin 2>/dev/null
arm-linux-objcopy -O srec hello_world hello_world.srec 2>/dev/null
arm-linux-objcopy -O srec atmel_df_pow2 atmel_df_pow2.srec 2>/dev/null
arm-linux-objcopy -O binary atmel_df_pow2 atmel_df_pow2.bin 2>/dev/null
make[1]: Leaving directory `/home/work/SBC60A5/u-boot-at91/examples/standalone'
INFO: COPY u-boot.bin -> /home/work/SBC60A5/Image/u-boot.bin : OK
```

If it goes right, the target images ***uboot.bin*** and ***u-boot.bin*** will be generated under directory ***Image***.

NOTE: uboot.bin used for SD booting；u-boot.bin used for DATAFLASH booting.

4.4.3 Build Linux

Run command to start building:

```
./build.sh linux
================================================
INFO: Build linux ...
================================================
...
  Image arch/arm/boot/uImage is ready
```

```
INFO: COPY arch/arm/boot/uImage -> /home/work/SBC60A5/Image/uImage : OK
INFO: INSTALL dts uImage -> /home/work/SBC60A5/rootfs/boot : OK
```

If it goes right, the target images ***uImage*** and ***sama5d34ek_pda4.dtb*** will be generated under directory ***Image***. At the same time, they are stored to the rootfs directory ***rootfs/boot***.

4.4.4 Build Rootfs

Type the following commands to build rootfs image:

```
./build.sh rootfs
==========================================
INFO: Build Rootfs ...
==========================================
...
INFO: GENERATE ROOTFS /home/work/SBC60A5/Image/rootfs.tar.bz2 : OK
```

The file named ***rootfs.tar.bz2*** will be generated in the directory ***Image***.

4.5 System Customization

In fact, there are many kernel configuration options at Linux kernel. You can modify the kernel features to meet your requirements. This section tells you how to customize the system.

4.5.1 Kernel Customization

The factory default configuration file named ***sama5_defconfig*** is provided in arch/arm/configs/ directory. You can customize your kernel based on it.

```
cd /home/work/SBC60A5/linux-at91
make sama5_defconfig
make menuconfig
```

The description for each device is as follows:

Serial Driver:

1. Select Device drivers
2. Select Character devices
3. Select Serial drivers
4. Select AT91 / AT32 on-chip serial port Support

Button Driver:

1. Select Device drivers
2. Select Input device support
3. Select Keyboards
4. Select GPIO Buttons

GPIO Driver:

1. Select Device drivers
2. Select GPIO Support
3. Select /sys/class/gpio/... (sysfs interface)

LED Driver:

1. Select Device drivers
2. Select LED Support
3. Select LED Class Support
4. Select LED Support for GPIO connected LEDs

SD/MMC Driver:

1. Select Device drivers
2. Select MMC/SD/SDIO card support
3. Select MMC block device driver
4. Select Atmel SD/MMC Driver (Atmel Multimedia Card Interface support)

USB Driver:

1. Select Device drivers
2. Select USB support
3. Select Support for Host-side USB
4. Select USB announce new devices
5. Select EHCI HCD (USB 2.0) support
6. Select Support for Atmel on-chip EHCI USB controller
7. Select OHCI HCD support
8. Select USB Mass Storage support

RTC Driver:

1. Select Device drivers
2. Select Real Time Clock
3. Select AT91RM9200 or some AT91SAM9 RTC

Watchdog Driver

1. Select Device drivers
2. Select Watchdog Timer Support
3. Select WatchDog Timer Driver Core
4. Select AT91SAM9X / AT91CAP9 watchdog

Note: There are two steps should be done to start the Watchdog.

1) Bootstrap enables Watchdog
   at91bootstrap/board/sama5d3xek/sama5d3xek.c:
   // at91_disable_wdt();
2) Linux configure the Watchdog

CAN Driver:

1. Select Networking support
2. Select CAN bus subsystem support
3. Select Raw CAN Protocol (raw access with CAN-ID filtering)
4. Select Broadcast Manager CAN Protocol (with content filtering)
5. Select CAN Gateway/Router (with netlink configuration)
6. Select CAN Device Drivers
7. Select Platform CAN drivers with Netlink support
8. Select CAN bit-timing calculation
9. Select Atmel AT91 onchip CAN controller

MACB Driver:

1. Select Device Drivers

2. Select Network device support
3. Select Ethernet driver support
4. Select Cadence devices
5. Select Cadence MACB/GEM support

Graphics support

1. Select Device Drivers
2. Select Graphics support
3. Select Support for frame buffer devices
4. Select AT91 HLCD Controller support

Touch Screen Driver

1. Select Device Drivers
2. Select Industrial I/O support
3. Select Analog to digital converters
4. Select Atmel AT91 ADC

Select *save*, compile the kernel by the following command:

```
make uImage
```

4.5.2 File System Customization

Table 4-1 Customization

Configuration	Path	indication
Driver modules	/lib/modules/3.10.0/	Store driver modules
Configuration for loading driver modules	/etc/modules	
Network address configuration	/etc/network/interfaces	
command prompt configuration	/etc/hostname	
Startup configuration	/etc/autorun.sh	Add to the end of file
Environment variable configuration	/etc/profile /etc/profile.d	
Coordinate files for touchscreen	/etc/pointercal	
Configuration on udev protocol	/etc/udev	
User tests	/home/app	

Network Configuration /etc/network/interfaces

Static IP:

```
auto eth0
iface eth0 inet static
address 192.168.1.211
netmask 255.255.255.0
```

Dynamic IP (DHCP):

```
auto eth0
iface eth0 inet dhcp
```

User program launches when system starts up, 2 methods:

- Create shell script under ***/etc/rc5.d*** to launch user program, make it runnable using chmod command.
- Append launch command for your app program to ***/etc/autorun.sh***.

4.6 Simple Kernel Driver Module

Running in kernel mode, driver program could control the hardware directly and provide a range of interfaces for application to control equipments. There is a simple driver module with overall interfaces.

```
    /* File: device_drv.c */
    #include <linux/kernel.h>
    #include <linux/module.h>
    #include <linux/init.h>
    #include <linux/input.h>
    #include <linux/miscdevice.h>
    #include <asm/io.h>
    #include <asm/uaccess.h>              /*header files, commonly used by drivers */

    #define DEVICE_NAME "demo"
           /* device name, node will be generated on /dev/demo if loading successfully */

    static int result = 0;

    static int device_open(struct inode *inode, struct file *file)
                                           /* achieve open operation */
    {
       result = 0;                         /* initialize result */
       return 0;
    }
    static ssize_t device_read(struct file *filp, char *buffer, size_t count, loff_t *ppos)
                                           /* achieve read operation */
    {
       int ret = copy_to_user (buffer, (char *)&result, sizeof(result));
                                           /* copy the value of result to buffer */
       if (ret < 0) {
          printk (KERN_ERR "%s: copy_to_user error\n", DEVICE_NAME);
          return -1;
       }
       return sizeof(result);
         /* return the effective length of buffer, namely the storage length of result */
    }
    static ssize_t device_write(struct file *filp, const char *buffer, size_t count, loff_t
*ppos)                                     /* write operation */
```

```
    {
        int ret = copy_from_user ((char *)&result, buffer, sizeof(result));
                                        /* copy the data transferred to buffer to result */
        if (ret < 0) {
            printk (KERN_ERR "%s: copy_from_user error\n", DEVICE_NAME);
            return -1;
        }
        return sizeof(result);
    }
    static int device_release(struct inode *inode, struct file *filp)
                                             /* close triggers the function */
    {
        return 0;
    }
    static struct file_operations device_fops =
                                        /* register files operate the interface functions*/
    {
        .owner    = THIS_MODULE,
        .open     = device_open,
        .read     = device_read,
        .write    = device_write,
        .release  = device_release,
    };
    static struct miscdevice device_miscdev =
                                              /* register device information of misc */
    {
        .minor   = MISC_DYNAMIC_MINOR,
        .name    = DEVICE_NAME,
        .fops    = &device_fops,
    };
    static int __init device_init(void)      /* insmod operations trigger the function */
    {
        int ret;

        ret = misc_register(&device_miscdev);   /* register the device */
        if (ret) {
            printk(KERN_ERR    "cannot    register    miscdev    on    minor=%d    (%d)\n",
MISC_DYNAMIC_MINOR, ret);
            goto out;
        }

        printk(KERN_INFO DEVICE_NAME " initialized!\n");
        return 0;
```

```
    out:
       return ret;
    }

    static void __exit device_exit(void)      /* rmmod operations trigger the function */
    {
          misc_deregister(&device_miscdev);
          printk(KERN_INFO DEVICE_NAME " removed!\n");
    }

    module_init(device_init);
    module_exit(device_exit);

    MODULE_LICENSE("GPL");                /* protocol the driver module should support */
    MODULE_DESCRIPTION("Linux Driver Demo");          /* description of driver module  */
```

The driver file can not be used independently; it should be associated with the kernel by the Makefile for successful compiler and loading operation. The contents of the Makefile:

```
    /* File: Makefile */
    ifneq ($(KERNELRELEASE),)
       obj-m := device_drv.o
        /*the driver which is .O file not .C can compile and search for .C file automatically*/
    else
          KERNELDIR ?= /home/work/SBC60A5/linux-at91  /*specify the kernel source path */

          PWD := $(shell pwd)
    all:
          $(MAKE) -C $(KERNELDIR) M=$(PWD) modules
    clean:
          rm -rf *.o *~ core .depend .*.cmd *.ko *.mod.c .tmp_versions Module.symvers
modules.order device_drv.ko
    endif
```

NOTE: Makefile the beginning space must be <tab>, otherwise some compiling errors may happen.

At first, compile the kernel resource code, then you can type make command to compile the driver files. Download the device_drv.ko file generated by successful compiler. Type the following commands:

```
root@SAMA5D3x:/# insmod device_drv.ko
demo initialized!
root@SAMA5D3x:/# ls /dev/demo
/dev/demo
root@SAMA5D3x:/# rmmod device_drv.ko
demo removed!
```

4.7 Application for Absolute Beginners

We learn from the example above how to develop a linux driver. You may find that it is device_init and device_exit two functions that the above operations call. The interface in the device_fops structure is for application layer to call. The introduction of basic components for application is as follows:

```
/* File: demo.c */
#include <stdio.h>
#include <fcntl.h>
#include <string.h>                    /* the required header files */

#define dev  "/dev/demo"               /* file node for demo */

int main (void)
{
   int fd;
   int err = 0;
   int value;

   fd = open (dev, O_RDWR);            /* open the file node, can r/w */
   if (fd < 0) {
      fprintf (stderr, "open fail\n");
      err = 1;
      goto out;
   }

   if (read (fd, &value, sizeof(value)) < 0) {
               /*call the driver read functions and store the value in value directory*/
      fprintf (stderr, "read error\n");
      err = 1;
      goto out;
   }
   printf ("read before write, value=%X\n", value);   /*  print the value */

   int writeValue = 0x5E7F;
   if (write (fd, &writeValue, sizeof(writeValue)) < 0) {
                               /* call function to write 0x5E7F to the driver module*/
      fprintf (stderr, "write error\n");
      err = 1;
      goto out;
   }
   if (read (fd, &value, sizeof(value)) < 0) {
                                        /* read the value again after writing*/
      fprintf (stderr, "read error\n");
      err = 1;
```

```
        goto out;
    }
    printf ("read after write, value=0x%X\n", value);
                                        /*print the value after writing  */

out:
    if (fd > 0) close (fd);
    return err;
}
```

Compile the application:

```
# arm-linux-gcc demo.c -o demo
```

There is an executable file named demo to be generated. Download it to the board and run:

```
root@SAMA5D3x:/# insmod device_drv.ko
demo initialized!
root@SAMA5D3x:/# ./demo
read before write, value=0
read after write, value=0x5E7F
```

The application interacts with the driver successfully.

4.8 Multi-thread Programming for Linux

The threads that we are talking about are created in more than one task. They share the resources of the same process, namely lightweight processes. It is much smaller and quicker in context switching than the process.

Since threads share the resources, some measures should be taken to avoid the competition for accessing resources.

```
/* File: pthread.c */
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>

void read_func(void);
void write_func(void);


int buffer_has_item = 0;                    /* shared resources */

pthread_mutex_t mutex;                      /* mutex */

int main(void)
{
    pthread_t reader, writer;              /* define the thread identifier */
```

```
    pthread_mutex_init(&mutex, NULL);               /* initialize mutex */

    pthread_create(&reader, NULL, (void*)&read_func, NULL);    /* create thread */
    pthread_create(&writer, NULL, (void*)&write_func, NULL);

    pthread_join(reader, NULL);                      /* wait for the end of thread */
    pthread_join(writer, NULL);

    return 0;
}

void write_func(void)
{
    while (1) {
        pthread_mutex_lock(&mutex);             /*lock, block other threads*/
        if (buffer_has_item == 0) {
            printf("create a new item\n");
            buffer_has_item = 1;
        }
        pthread_mutex_unlock(&mutex);             /* unlock, activate other threads */
    }
}

void read_func(void)
{
    while (1) {
        pthread_mutex_lock(&mutex);
        if (buffer_has_item == 1) {
            printf ("destroy item\n");
            buffer_has_item = 0;
        }
        pthread_mutex_unlock(&mutex);
    }
}
```

Compile:

```
# arm-linux-gcc pthread.c -o pthread_demo -lpthread
```

4.9 Network Programming for Linux

Linux network programming could be divided into UDP and TCP generally. Providing simple, transaction-oriented and unreliable transmission service, UDP is a transport protocol without connection. While TCP is a connection-oriented and reliable transport protocol based on bite flows. A simple example based on TCP server and client is as follow:

Server: to monitor the connection from clients, and sent string to clients as soon as the connection is established and found.

Client: connect to the server. Receive and print the information from the server.

```
/* File: server.c */
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/wait.h>

#define MYPORT 3490                /* the port users will be connecting to */
#define BACKLOG 10                 /* how many pending connections queue will hold */

main()
{
    int sockfd, new_fd;            /* listen on sockfd, new connection on new_fd */
    struct sockaddr_in my_addr;    /* local address information */
    struct sockaddr_in their_addr; /* connector's address information */
    int sin_size;

    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        perror (" socket ") ;
        exit(1) ;
    }

    my_addr.sin_family = AF_INET;
    my_addr.sin_port = htons(MYPORT);
    my_addr. sin_addr.s_addr = INADDR_ANY;     /* auto-fill with local IP */

    if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr)) == -1) {
        perror (" bind ") ;
        exit(1) ;
    }

    if (listen(sockfd, BACKLOG) == -1) {
        perror (" listen ") ;
        exit(1) ;
    }

    while(1) {        /* main accept() loop */
```

```
        sin_size = sizeof(struct sockaddr_in);
        if ((new_fd = accept(sockfd, (struct sockaddr *)&their_addr,
        &sin_size)) == -1) {
            perror (" accept ") ;
            continue ;
        }
        printf("server: got connection from %s\n", inet_ntoa(their_addr.sin_addr));
        if (!fork()) { /* this is the child process */
        if (send(new_fd, "Hello, world!\n", 14, 0) == -1)
            perror( " send " ) ;
            close( new_fd ) ;
            exit ( 0 ) ;
        }
        close(new_fd);
        while(waitpid(-1,NULL,WNOHANG) > 0);        /* clean up child processes */
    }
}
```

```
/* File: client.c */
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>

#define PORT 3490                    /* the port client will be connecting to */
#define MAXDATASIZE 100              /* max number of bytes we can get at once */

int main(int argc, char *argv[])
{
    int sockfd, numbytes;
    char buf[MAXDATASIZE] ;
    struct hostent *he;
    struct sockaddr_in their_addr;     /* connector's address information */

    if (argc != 2) {
        fprintf(stderr,"usage: client hostname\n");
        exit (1) ;
    }

    if ((he=gethostbyname(argv[1])) == NULL) {       /* get the host info */
```

```
        herror(" gethostbyname ") ;
        exit (1);
    }

    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        perror( " socket ");
        exit (1);
    }

    their_addr.sin_family = AF_INET;
    their_addr.sin_port = htons(PORT);
    their_addr.sin_addr = *((struct in_addr *)he->h_addr); //inet_addr

    if (connect(sockfd, (struct sockaddr *)&their_addr, sizeof(struct sockaddr)) == -1)
    {
        perror(" connect ");
        exit (1);
    }

    if ((numbytes=recv(sockfd, buf, MAXDATASIZE, 0)) == -1) {
        perror (" recv ");
        exit (1);
    }
    buf[numbytes] = '\0';
    printf("Received: %s",buf);
    close(sockfd) ;
    return 0;
}
```

Compile

```
# arm-linux-gcc client.c -o client
# arm-linux-gcc server.c -o server
```

Generated at the time of successful compiler, the executable client and server files could run on the ARM board. If there are two boards, then one runs server file and the other runs client file to achieve the communication. While if there is only one board, then the board runs client file and the PC runs server file to achieve the communication.

Compile a server program for PC

```
# gcc server.c -o server
```

Run the server program on PC

```
# ./server
```

Download the client program to the board and run it.

```
root@SAMA5D3x:/# chmod 755 client
```

```
root@SAMA5D3x:/# ./client 192.168.1.10          # server IP
Received: Hello, world!
root@SAMA5D3x:/# ./client 192.168.1.10
Received: Hello, world!
root@SAMA5D3x:/# ./client 192.168.1.10
Received: Hello, world!
```

At this point, the following information will be printed to the terminal:

```
server: got connection from 192.168.1.211
server: got connection from 192.168.1.211
server: got connection from 192.168.1.211
```

Chapter 5 Update the System Image

The board is equipped with DATAFLASH and NANDFLASH (not placed as default) storage memory. Currently Linux system supports to boot from DATAFLASH. The introduction of system image partition is shown below.

5.1 System Image Map

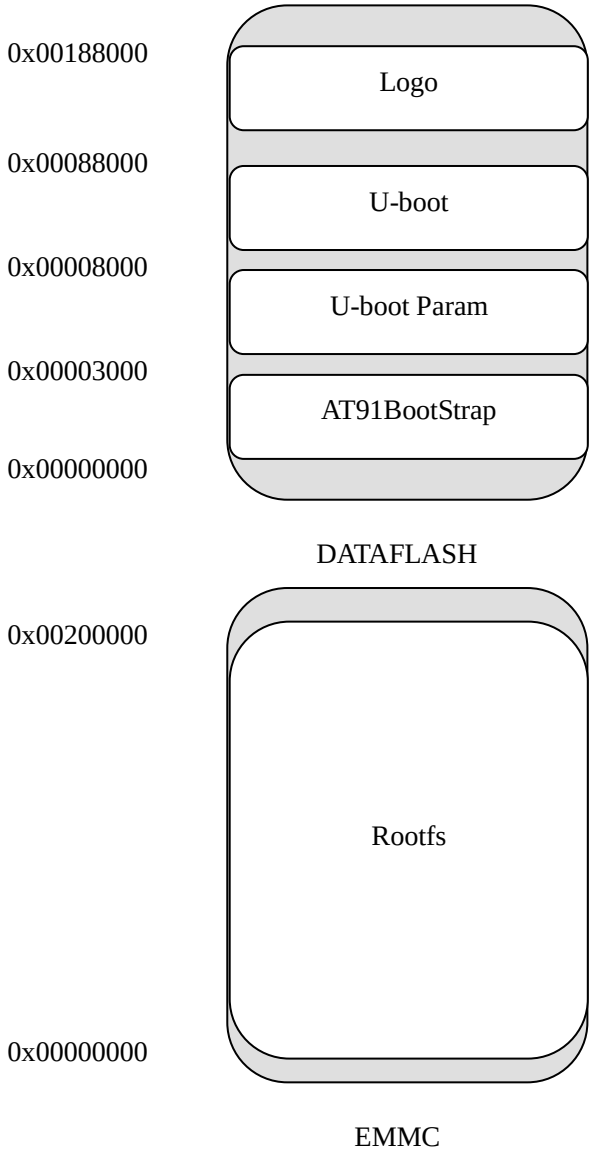


Figure 5.1.1 Storage Partition Map

5.2 Set the Burning Environment for Image

Install the application ***CDROM/02 Linux Kit\02 Tools\SAM-BA*** in your PC. Double click ***sam-ba_2.12.exe*** to install main program, then click ***sam-ba_2.12_patch2a.exe*** to install patches. You will see the icon ***SAM-BA v2.12*** on the PC desktop.

5.3 Burn System Image

5.3.1 Burn Bootstrap in booting DATAFLASH mode

- Connect debug serial port. For more details about hardware connection please refer to "SBC60A5 Hardware Manual" chapter 2.
- Connect the upper USB slot on the board to your PC by A type USB cable.
- Open the Hyper Terminal on the PC, configure the terminal program for (BAUD RATE-115200, DATA-8 bit, STOP-1 bit, PARITY-none, FLOW CONTROL-none).
- Make sure there is no code in DATAFLASH or disable the DATAFLASH (disconnect jumper JP16), then power on.
- Double click the ***SAM-BA v2.12*** icon on the PC desktop.

Figure 5.3.1 SAM-BA v2.12 Icon

- As shown in figure 5.3.2, select "***at91sama5d3x-ek***" and "***\USBserial\COM7***", and click "***Connect***". While if there is no "***\USBserial\COM<x>***" option, please press key ***PB5*** to reset the board and launch SAM-BA to try again.

Figure 5.3.2 SAM-BA startup interface

- After clicking "***Connect***", the interface as shown in figure 5.3.3.

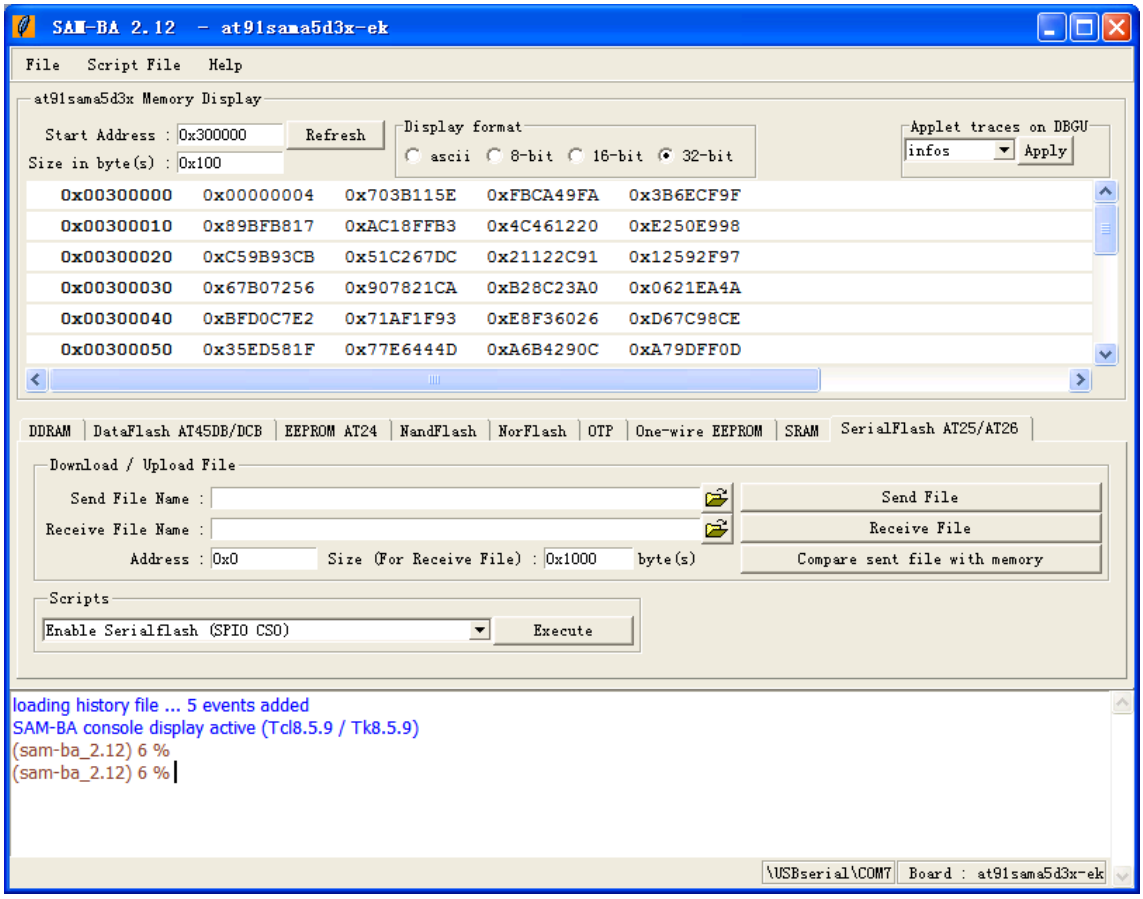


Figure 5.3.3 Burning Interface

- Connect the ***JP6***, select "***SerialFlash AT25/AT26***". And select the option "***Enable Serialflash (SPI0 CS0)***" on "***Scripts***" Pull-down menu, then click "***Execute***". As shown in figure 5.3.4:

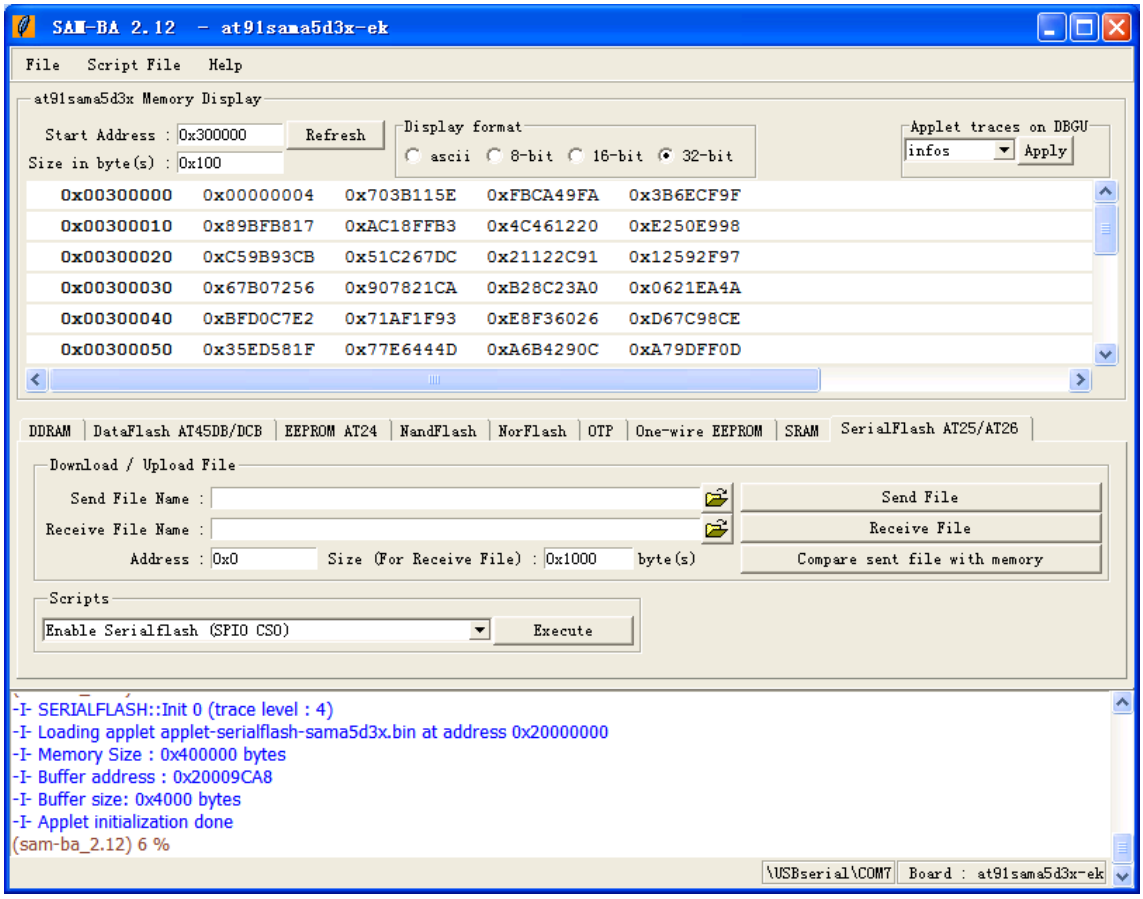


Figure 5.3.4 Enable the DATAFLASH

- On "***Scripts***" Pull-down menu, select "***Erase All***", and click "***Execute***", the DATAFLASH will be erased.

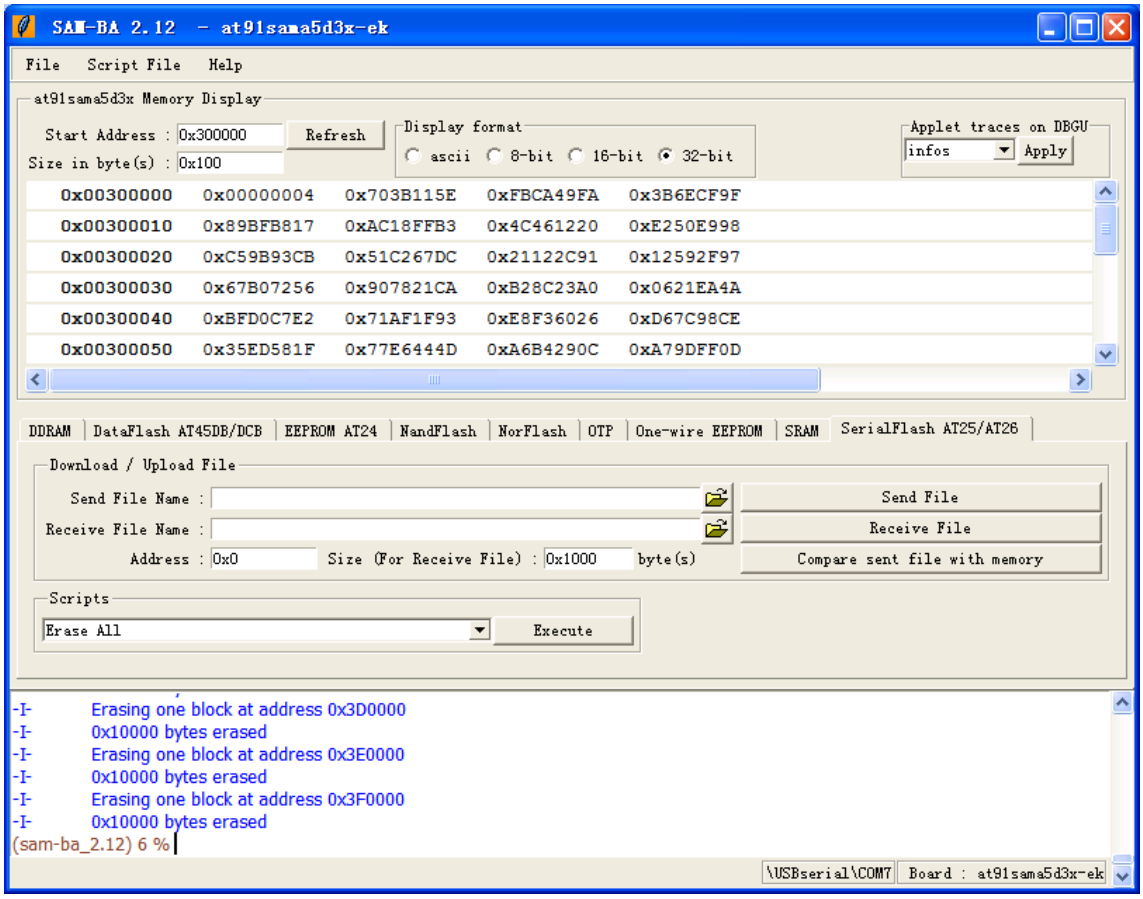


Figure 5.3.5 Erase DATAFLASH Interface

➢ On "***Scripts***" Pull-down menu, select "***Send Boot File***", and click "***Execute***".

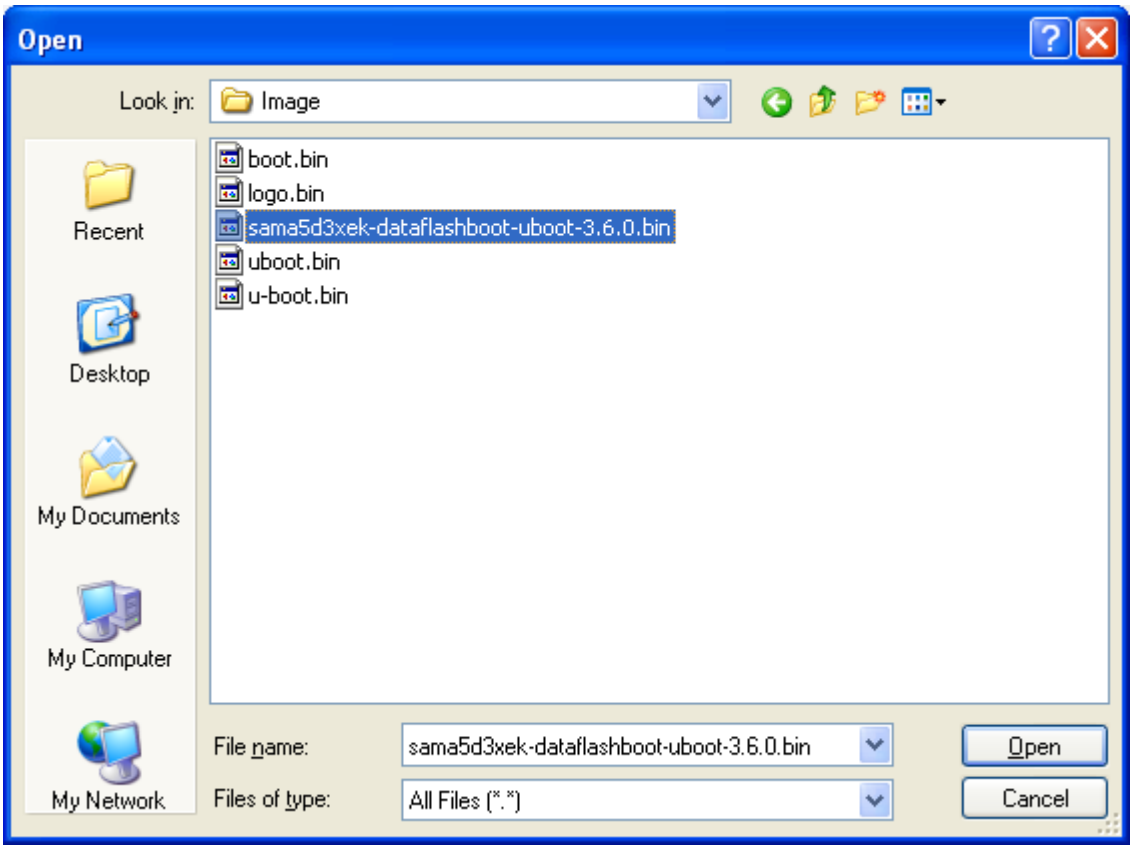


Figure 5.3.6 burn sama5d3xek-dataflashboot-uboot-3.6.0.bin

- If you click "***Execute***", there will be a dialog box, select "***sama5d3xek-dataflashboot-uboot-3.6.0.bin***" and click "***open***". Then you can download this file to target board.

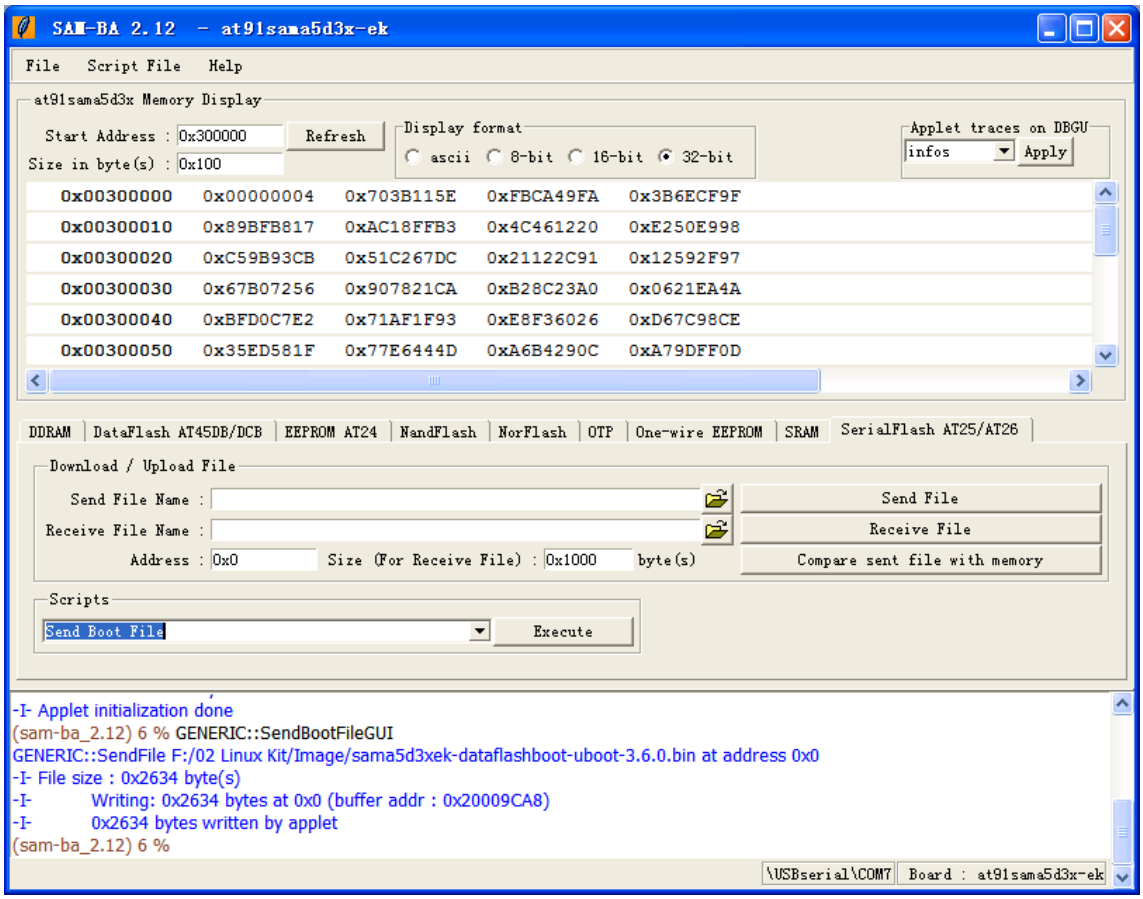


Figure 5.3.7 SAM-BA burn sama5d3xek-dataflashboot-uboot-3.6.0.bin

5.3.2 Burn the U-Boot file.

Burn the u-boot.bin file to dataflash through SAM-BA.

- Type "***0x8000***" on the "***address***" field.

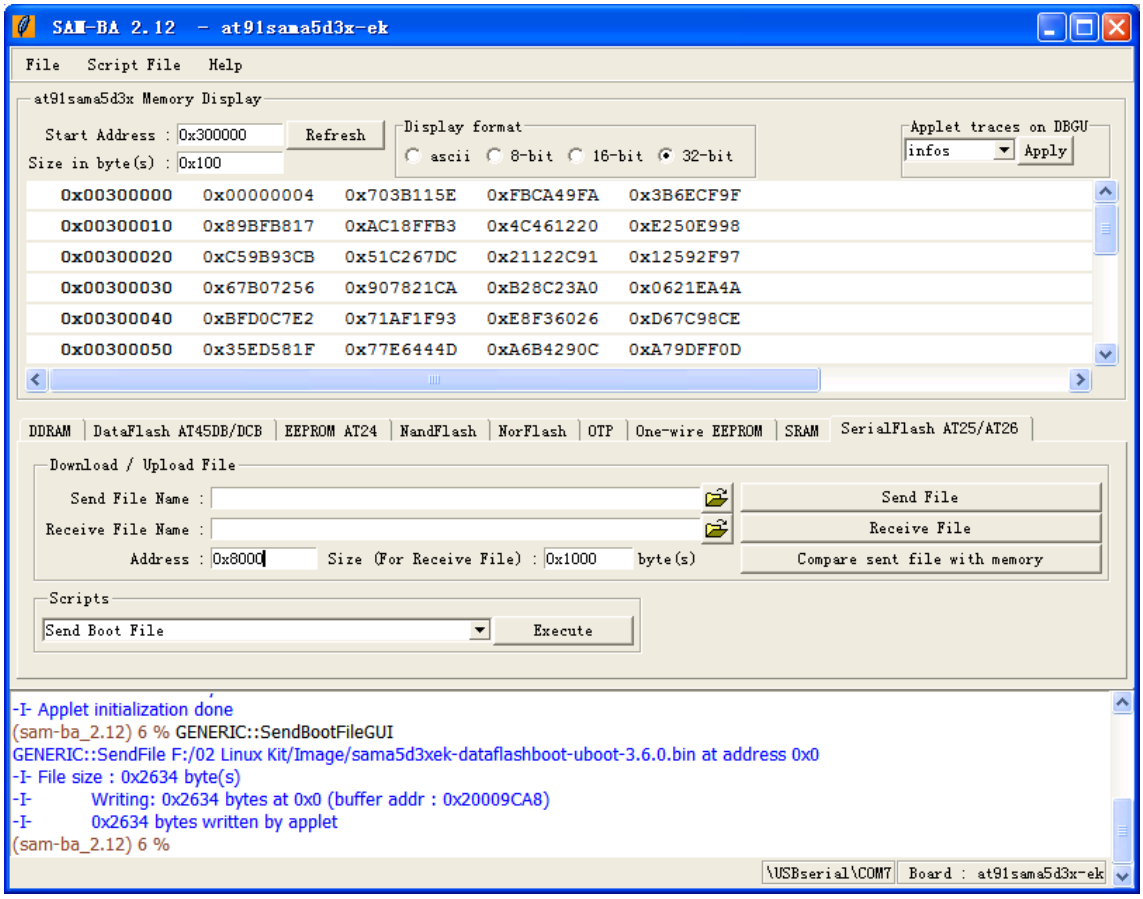


Figure 5.3.8 Burn u-boot.bin

- Click "***open folder***" to open the "***Send File Name***" browse window.

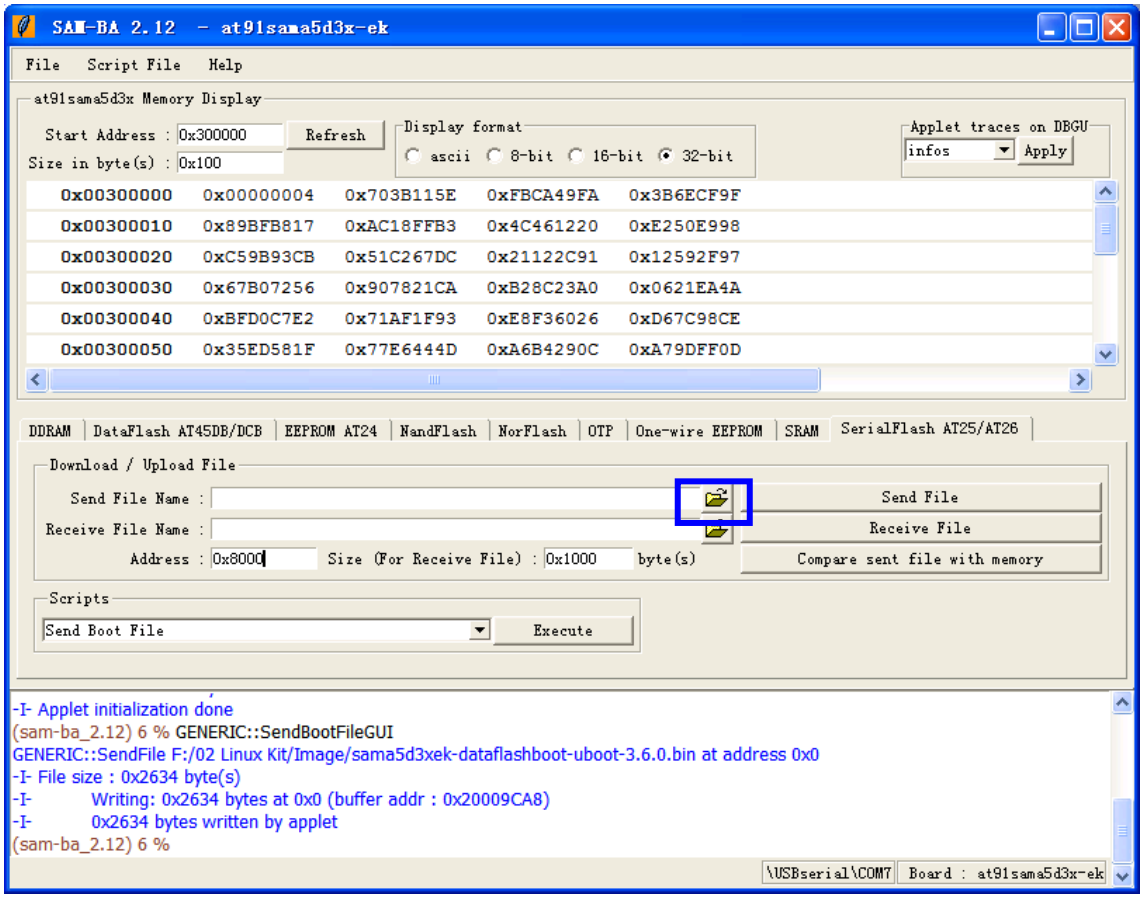


Figure 5.3.9 Burn u-boot.bin

➢ You will open a dialog box, select "***u-boot.bin***" and click "***open***".

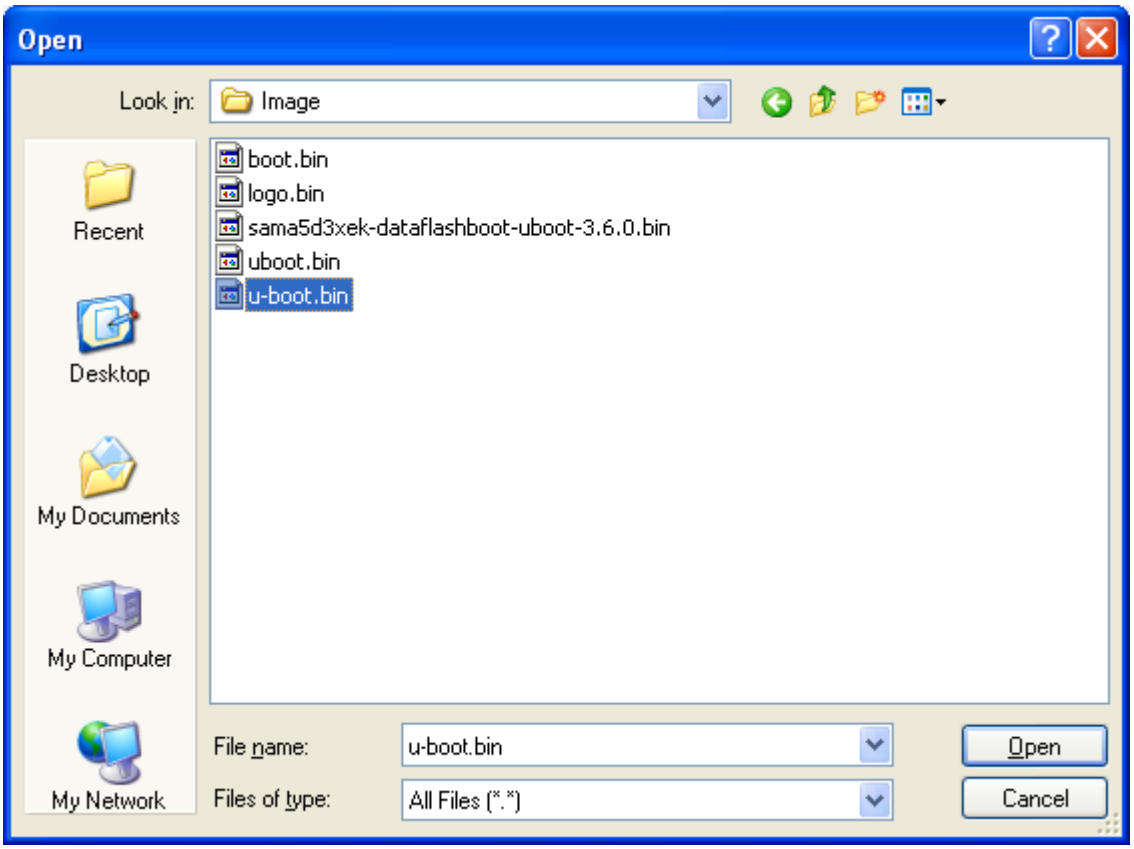


Figure 5.3.10 Burn u-boot.bin

➢ Click "***Send File***" to burn "u-boot.bin" file to DATAFLASH.

5.3.3 Burn the Logo

Burn "logo.bin" file to DATAFLASH through SAM-BA.

➢ Type "***0x88000***" on the "***address***" field

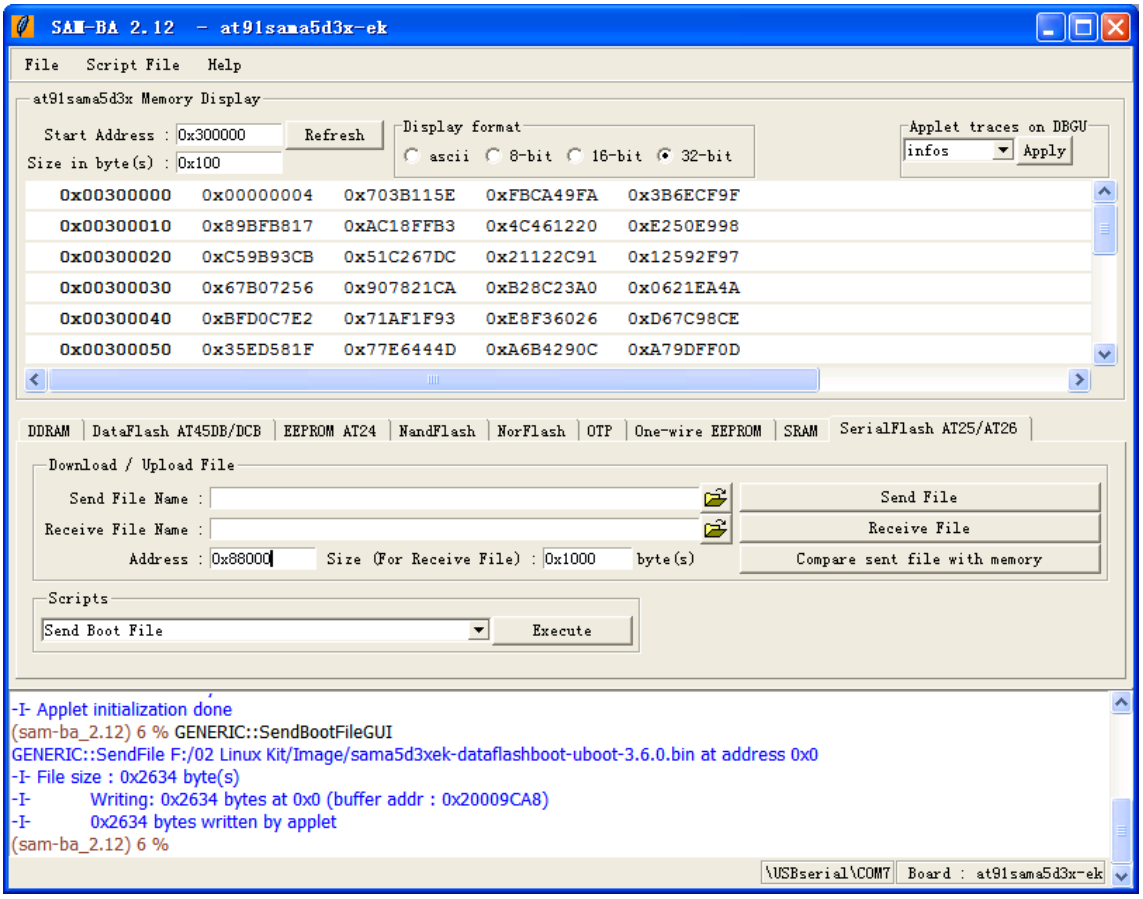


Figure 5.3.11 Burn logo.bin to dataflash

➢ Click "***open folder***" to open the "***Send File Name***" browse window.

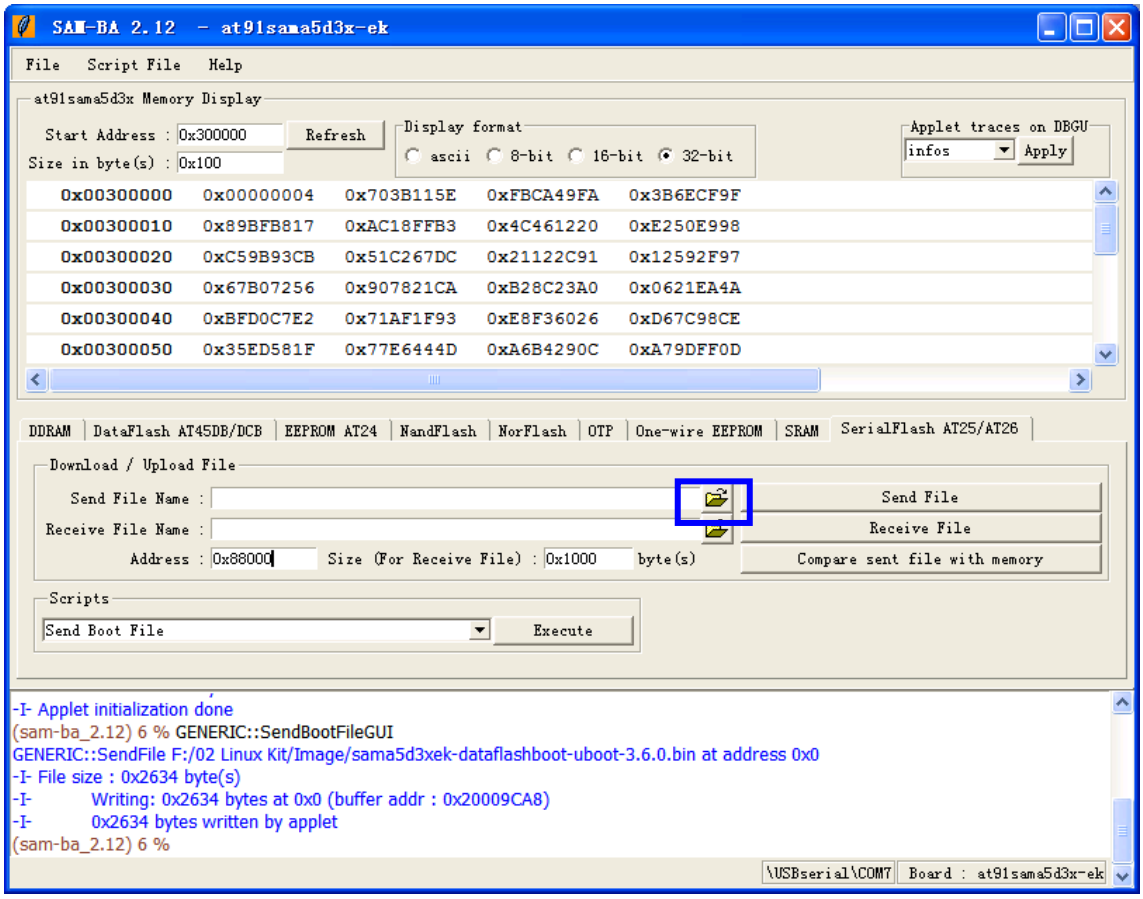


Figure 5.3.12 Burn logo.bin to dataflash

- Open the "***Send File Name***" browse window, select "***logo.bin***", as shown in the figure 5.3.13.
- Click "***Send File***" button to burn "***logo.bin***" to DATAFLASH.

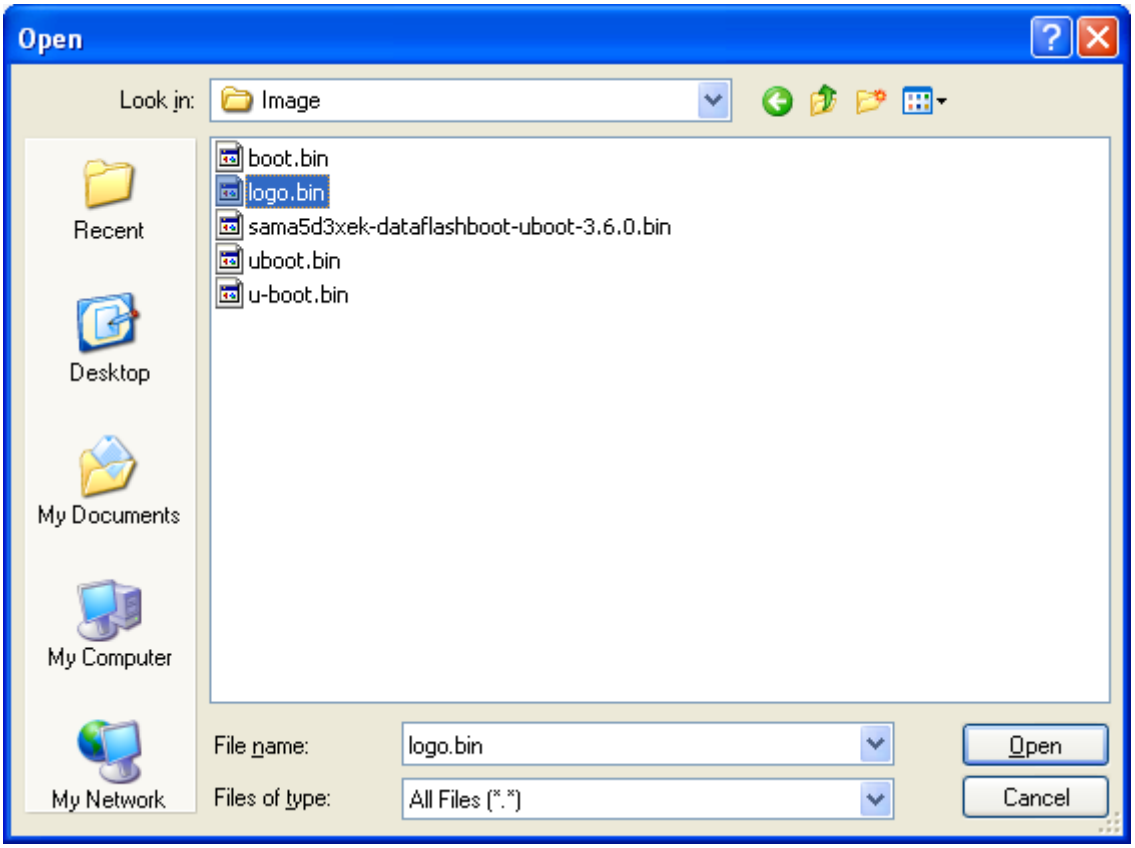


Figure 5.3.13 Burn logo.bin to dataflash

Please remember, the uImage and dtb are integrated with the rootfs, so it is just neccessary to write the rootfs image into EMMC.

- Copy all files under ***CDROM/02 Linux Kit\00 Image*** into SD card.

  ① SD Card Capacity: 2GB or below.

  ② Format SD Card with tool SDFormatter (***CDROM/02 Linux Kit\02 Tools/SDFormatter.7z***)

  ③ Files under SD root directory:

  |-- boot.bin

  |-- logo.bin

  |-- ramdisk.img

  |-- rootfs.tar.bz2

  |-- sama5d34ek_pda4.dtb

  |-- sama5d3xek-dataflashboot-uboot-3.6.0.bin

  |-- u-boot.bin

  |-- uImage

  `-- uboot.bin
- Insert SD card into SD slot on the board, disconnect jumper ***JP6*** to disable DATAFLASH.
- Reboot the board, the buzzer must beep once when SD boot successfuly. Otherwise format the card with SDFormatter and retry.
- if you connect jumper ***JP6*** to enable DATAFLASH, it can update images such as bootstrap/u-boot/logo into DATAFLASH automatically.
- Automatically boot into ramdisk, format EMMC and write rootfs into it.

- When update procedure is going on, the ***LED12*** will flash rapidly.
- When successfully complete, the ***LED12*** restores into heart-beat mode, and the buzzer beeps 3 times.
- Remove the SD card, connect jumper ***JP6*** if it's disconnected. Reboot the board to boot new system images.

If reading carefully enough, you can find out that the SDCard boot program not only updates rootfs, but also updates bootstrap/u-boot/logo. Yes, we design the program to update all images on the board.

5.4 Burn the System Images Automatically

Conditions: 1) a 2GB (or below) SD card
2) boot.bin, uboot.bin and the image files

Table 5-1 Image Files

Image files		Make
Tool images	boot.bin	Tools
	uboot.bin	Tools
	ramdisk.img	Tools
LINUX	sama5d3xek-dataflashboot-uboot-3.6.0.bin	System images
	u-boot.bin	System images, u-boot.bin
	logo.bin	System images
	sama5d34ek_pda4.dtb	System images, dtb
	uImage	System images, uImage
	rootfs.bin	System images, rootfs

The SDCard boot does not always take effects, because the core cpu may detect valid code in DATAFLASH. Now I can tell you that some method to enable SDCard boot.

① No valid code in DATAFLASH (Empty chip).
② Jumper ***JP6*** is disconnected when startup.
③ Using RECOVERY key.

Method ①: mainly used for new board with empty storage memory.
Method ②: No need to update DATAFLASH, just write to EMMC.
Method ③: If you don't want to handle the jumper JP6, or don't want to run u-boot command to erase DATAFLASH, this method is an easy way for you.

- Press down ***BP4*** (***KEY2***), don't release;
- Click ***RESET*** key to reboot the board;
- Release ***BP4***, and then reboot the board, it can detect files in SDCard and boot from it.

5.5 Update Image Online

When Linux system is running on the board, we want to update some system images such as dtb and uImage, but we can not insert SD card into it, nor boot into u-boot command line. So we need to update online. It's quite easy.

- Prepare your dtb and uImage, and load them into Linux file system on the board.
- Run command to update dtb:

```
root@SAMA5D3x:/# cp -f <YOUR_PATH>/sama5d34ek_pda4.dtb /boot/
```

- Run command to update uImage

```
root@SAMA5D3x:/# cp -f <YOUR_PATH>/uImage /boot/
```

- Force to write data back into the storage medium, and reboot.

```
root@SAMA5D3x:/# sync; reboot
```

Chapter 6 QT Demo

Update rootfs image with QtDemo program, it will automatically launch demo program when booting up. Because of different lcd type you use, please calibrate the touch pannel.

1. Enter the following command for touch screen calibration:

```
root@SAMA5D3x:/# ts_calibrate; ts_test
```

Follow the on-screen prompts and click the “+” icon five times to complete the calibration.
Press Ctrl+C to terminate the test.

2. Reboot the board.

```
root@SAMA5D3x:/# sync; reboot
```

Then the following picture will be displayed.

Figure 6.1 Demo Interface

Figure 6.2 Light Controller

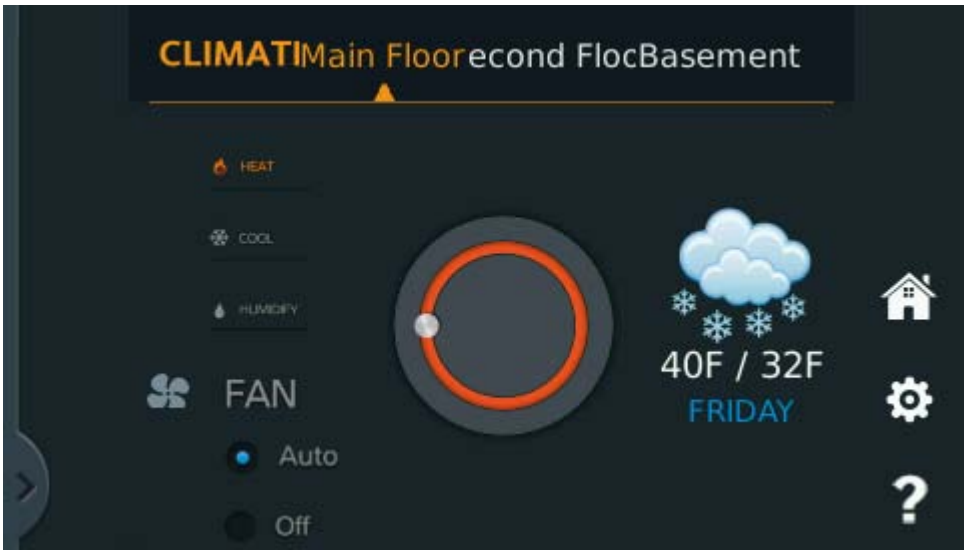


Figure 6.3 Weather Report

Technical Support and Warranty

Technical Support

Embest Technology provides its product with one-year free technical support including:

- Providing software and hardware resources related to the embedded products of Embest Technology;
- Helping customers properly compile and run the source code provided by Embest Technology;
- Providing technical support service if the embedded hardware products do not function properly under the circumstances that customers operate according to the instructions in the documents provided by Embest Technology;
- Helping customers troubleshoot the products.

The following conditions will not be covered by our technical support service. We will take appropriate measures accordingly:

- Customers encounter issues related to software or hardware during their development process;
- Customers encounter issues caused by any unauthorized alter to the embedded operating system;
- Customers encounter issues related to their own applications;
- Customers encounter issues caused by any unauthorized alter to the source code provided by Embest Technology;

Warranty Conditions

1） 12-month free warranty on the PCB under normal conditions of use since the sales of the product;

2） The following conditions are not covered by free services; Embest Technology will charge accordingly:

- Customers fail to provide valid purchase vouchers or the product identification tag is damaged, unreadable, altered or inconsistent with the products.
- Products are damaged caused by operations inconsistent with the user manual;
- Products are damaged in appearance or function caused by natural disasters (flood, fire, earthquake, lightning strike or typhoon) or natural aging of components or other force majeure;
- Products are damaged in appearance or function caused by power failure, external forces, water, animals or foreign materials;
- Products malfunction caused by disassembly or alter of components by customers or, products disassembled or repaired by persons or organizations unauthorized by Embest Technology, or altered in factory specifications, or configured or expanded with the components that are not provided or recognized by Embest Technology and the resulted damage in appearance or function;
- Product failures caused by the software or system installed by customers or inappropriate settings of software or computer viruses;
- Products purchased from unauthorized sales;
- Warranty (including verbal and written) that is not made by Embest Technology and not included in the scope of our warranty should be fulfilled by the party who committed. Embest Technology has no any responsibility;

3） Within the period of warranty, the freight for sending products from customers to Embest Technology should be paid by customers; the freight from Embest to customers should be paid by us. The freight in any direction occurs after warranty period should be paid by customers.

4） Please contact technical support if there is any repair request.

Note:

🕮 Embest Technology will not take any responsibility on the products returned by anyone without the permission of the company.

Contact Information

Technical Support

Telephone Number: +86-755-25635626-872/875/897

Email Address: support@embest-tech.com

Sales Information

Telephone Number: +86-755-25635626-860/861/862

Fax Number: +86-755-25616057

Email Address: globalsales@embest-tech.com

Company Information

Company Website: http://www.embest-tech.com or http://www.timll.com

Company Address: Tower B 4/F, Shanshui Building, Nanshan Yungu Innovation Industry Park, Liuxian Ave. No. 1183, Nanshan District, Shenzhen, Guangdong, China (518055)