



EBR (Electronic Book Reader) Advanced Development Kit Installation Guide

Prime View International Co., Ltd

8F., No. 110 Sec. 1, Nanchang Rd., Zhongzheng District, Taipei City 100. Taiwan

TEL:+886-2-2393-8799 FAX:+886-2-23938809

Document name : EBR Advanced Development kit Users Guide

Document No. : EBR-SDK-20070313

Version : 1.3

Date : March,13, 2007

Status : third release



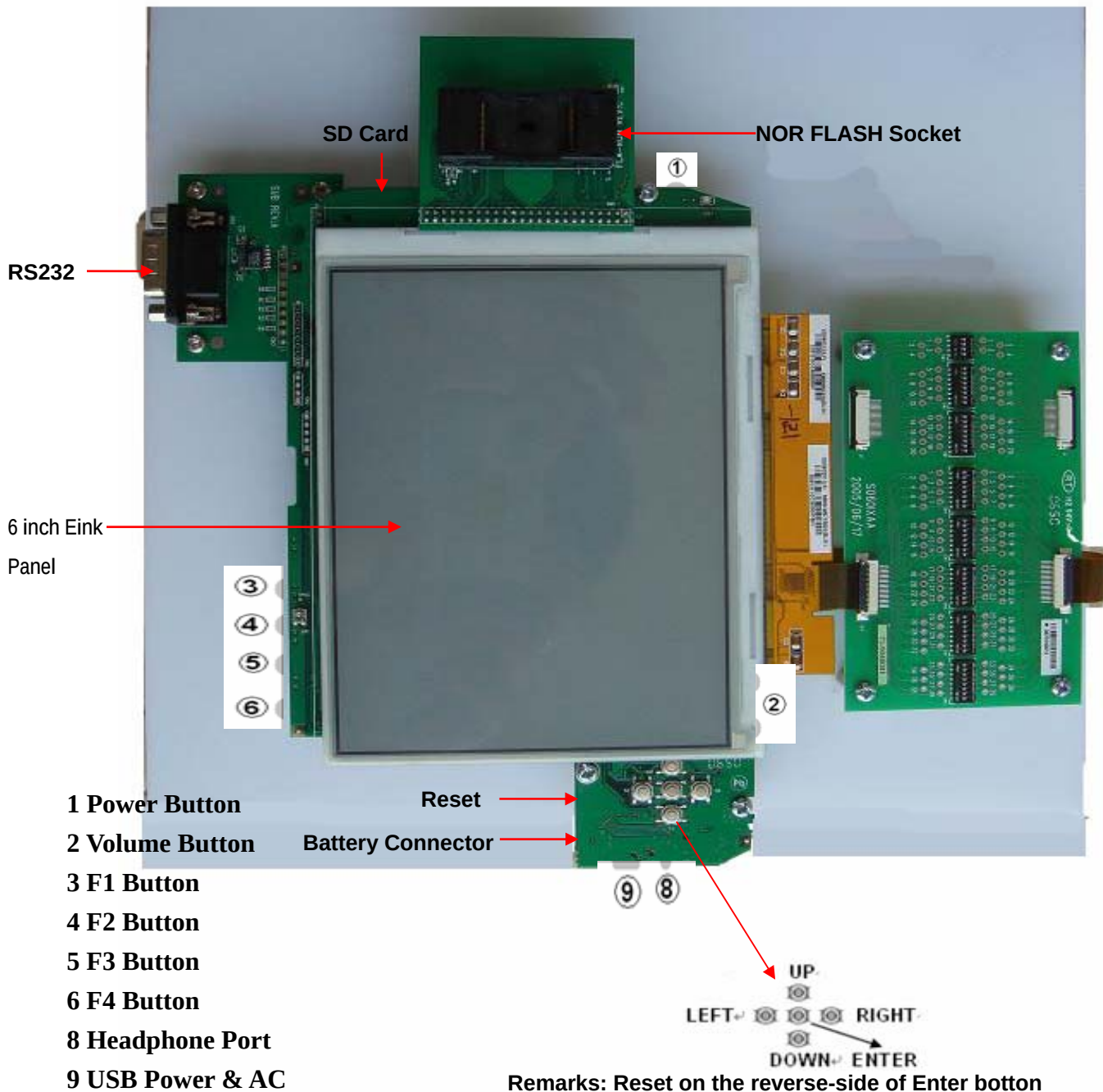
Table of CONTENT

1. VERSION CONTROL	2
2. EBR-DEVELOP BOARD STRUCTURE.....	3
4. EBR-SDK ENVIRONMENT SETUP	6
5. TERMINAL	7
5.1 MINICOM (LINUX)	7
5 · 2 HYPERTRM (WINDOWS).....	11
6. BSP	12
6.1 FILE LIST.....	12
6.2 BSP FILE LIST.....	13
6.3 INSTALL TOOL CHAIN	14
7. BOOTLOADER & KERNEL	15
8. NANO-X(GUI).....	16
9. FILES SYSTEM(FOR APPLICATION).....	17
10. MAKE STARTUP IMAGE.....	19
11. UPDATE	20
11.1 USED SD CARD UPDATE APPLICATION.....	20
11.2 AUTOMATISM STARTUP NEW EBRMAIN WHICH IN SD CARD :	21
11.3 USED SD CARD UPDATE STARTUP IMAGE	23
11.4 USED IC WRITE	24
13. TO DETECT THE SERIAL NUMBER.....	31
15. SAMPLE CODE	33
16. DEMO EXPOSITION.....	34
17. I/O CONTROL	37

1. VERSION CONTROL

Version	Doc. No.	Date	Editor	Description
1.0	EBR-SDK-20060822	2006/08/22	Jemy Huang	First release
1.2	EBR-SDK-20070109	2007/1/19	Jemy Huang	Second release
1.3	EBR-SDK-20070313	2007/3/13	Wanghp	Third release

2. EBR-Develop board Structure





● Accessory of Development Kit



E-book reader SDK board



5V power



USB Cable



Serial Cable



SD Card



Card reader

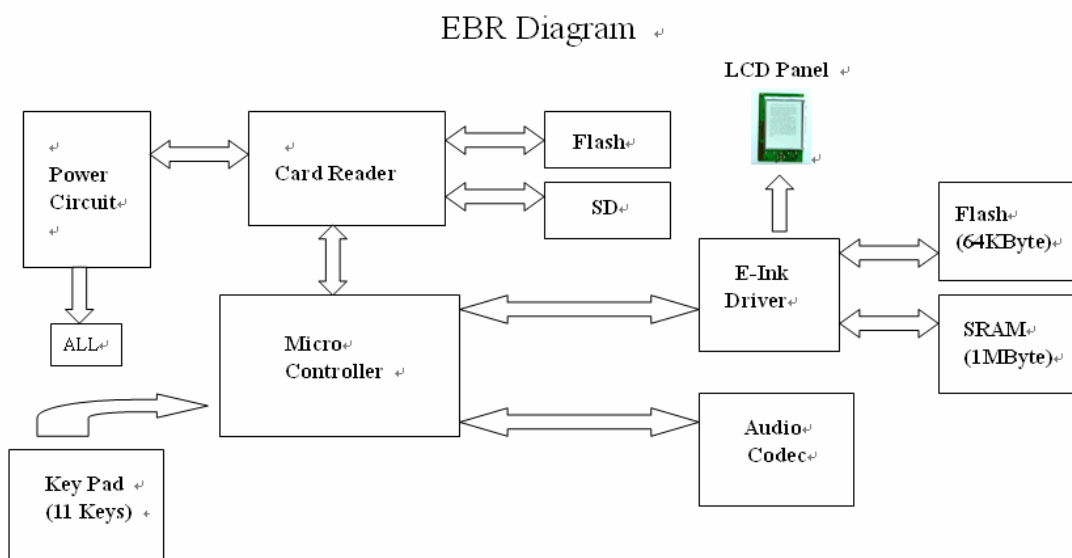


CD-ROM



Battery

3. Block Diagram



4. EBR-SDK Environment Setup

The evaluation environments for the EBR-SDK are shown in Figure 3-1. The serial port (UART0) on the EBR_SDK has to be connected to COM port of the host PC. This can be used as a console for monitoring and debugging the EBR-SDK.

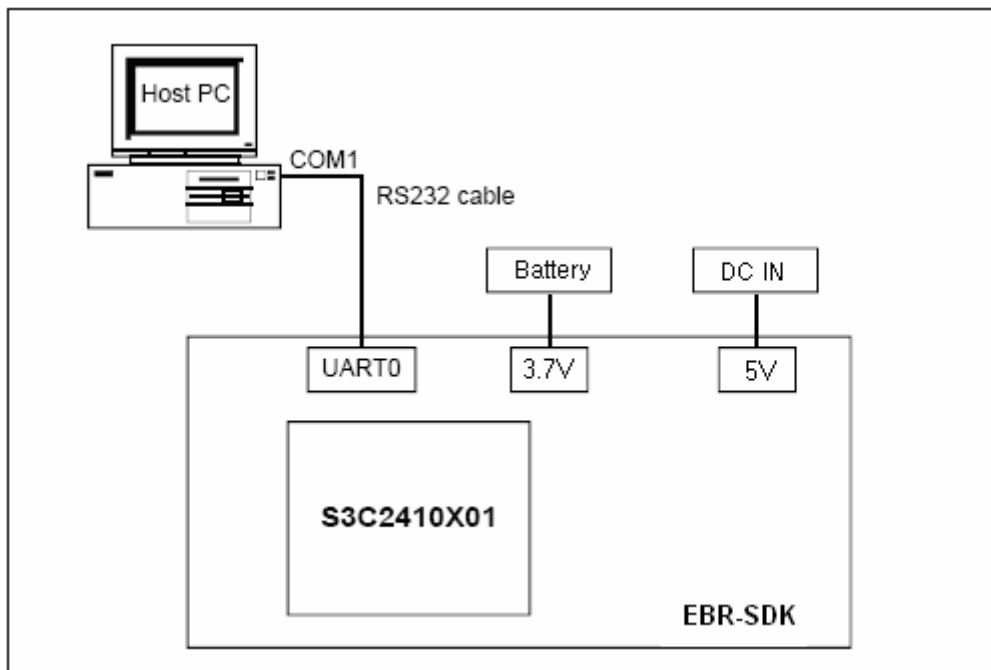


Figure 3-1. Setup Environment for EBR-SDK Board

The serial cable is made as in Figure 3-2. Make sure to check the cable's connections to prevent other pins from being used. The UART0 and PC COM1 or COM2 port has to be connected through this cable connection.

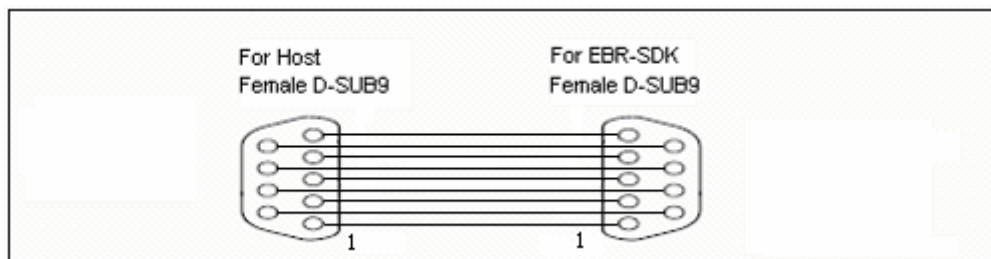


Figure 3-2. Serial Cable Connections for EBR-SDK Board

5. Terminal

5.1 minicom (Linux)

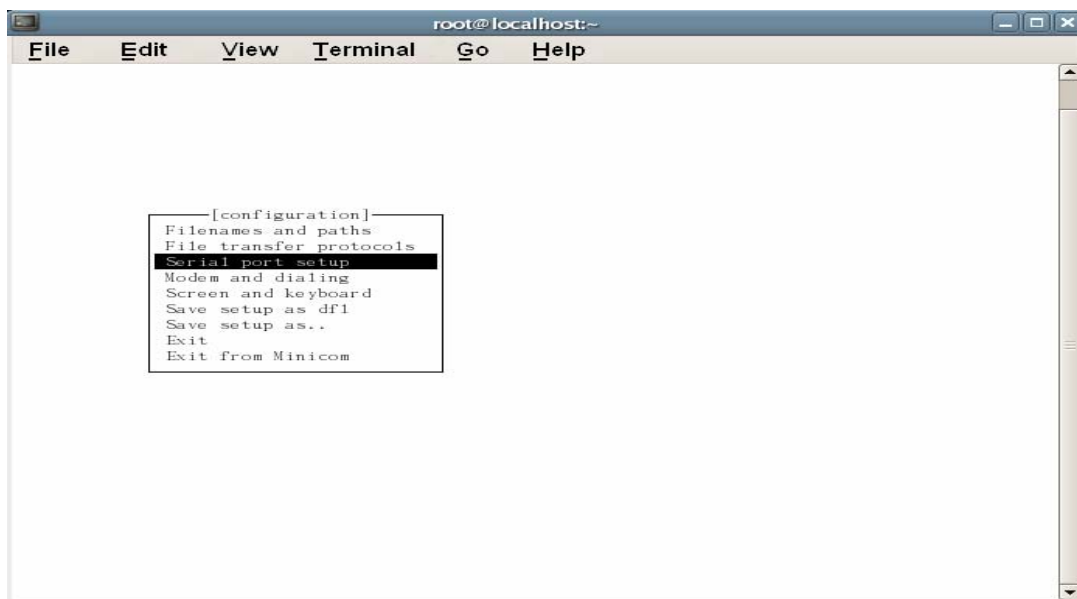
Desktop Linux has **Minicom** program for serial communication. It is used for *command prompt of vivi or shell prompt of embedded Linux*.

Set up the values before using **Minicom** program.

Step 1

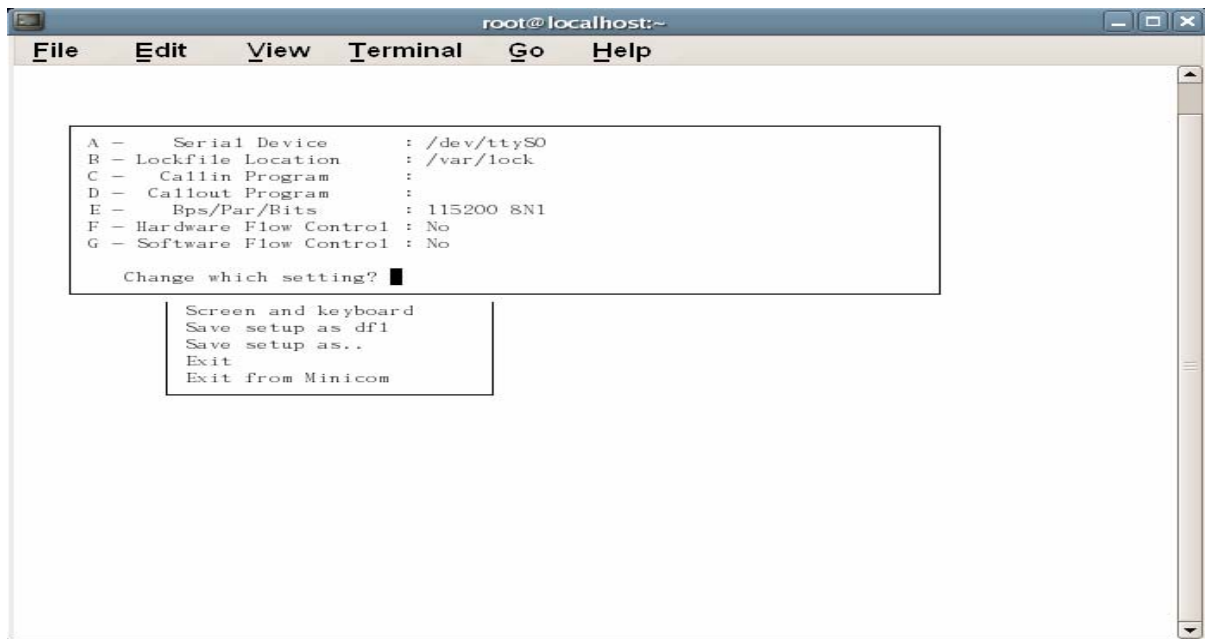
Execute Minicom with setting option.

```
[root@localhost test]# minicom -s
```



Step 2

Please select '**Serial port setup**' Push '**A**' key for setting '**Serial Device,**' then select the identified serial port which is connected to target board. (If you are using **COM1**, write **/dev/ttyS0**, if **COM2**, write **/dev/ttyS1**.)



Step 3

Push '**F**' key for setting up '**Hardware Flow Control**' to '**NO**'.

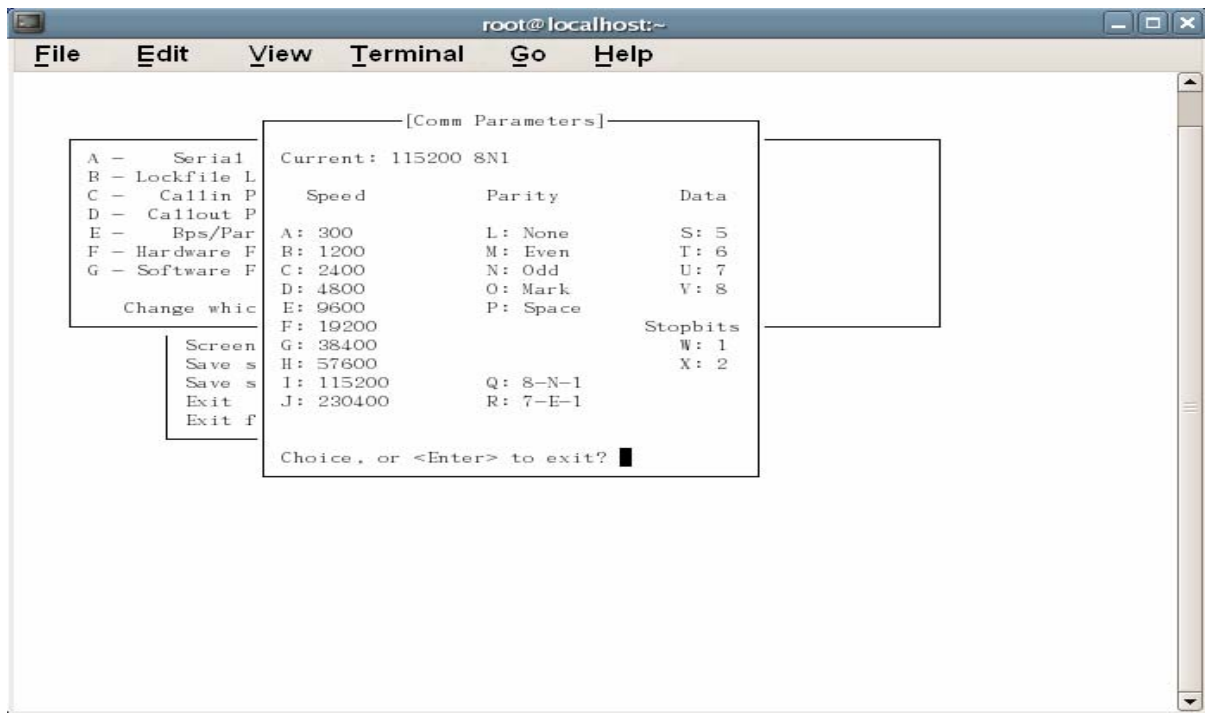
Push '**G**' key for setting up '**Software Flow Control**' to '**NO**'.

The default value is '**NO**'.



Step4:

Push '**E**' key for setting up '**bps/Par/Bits**'.



Push '**I**' to set up '**bps**' to **115200** .

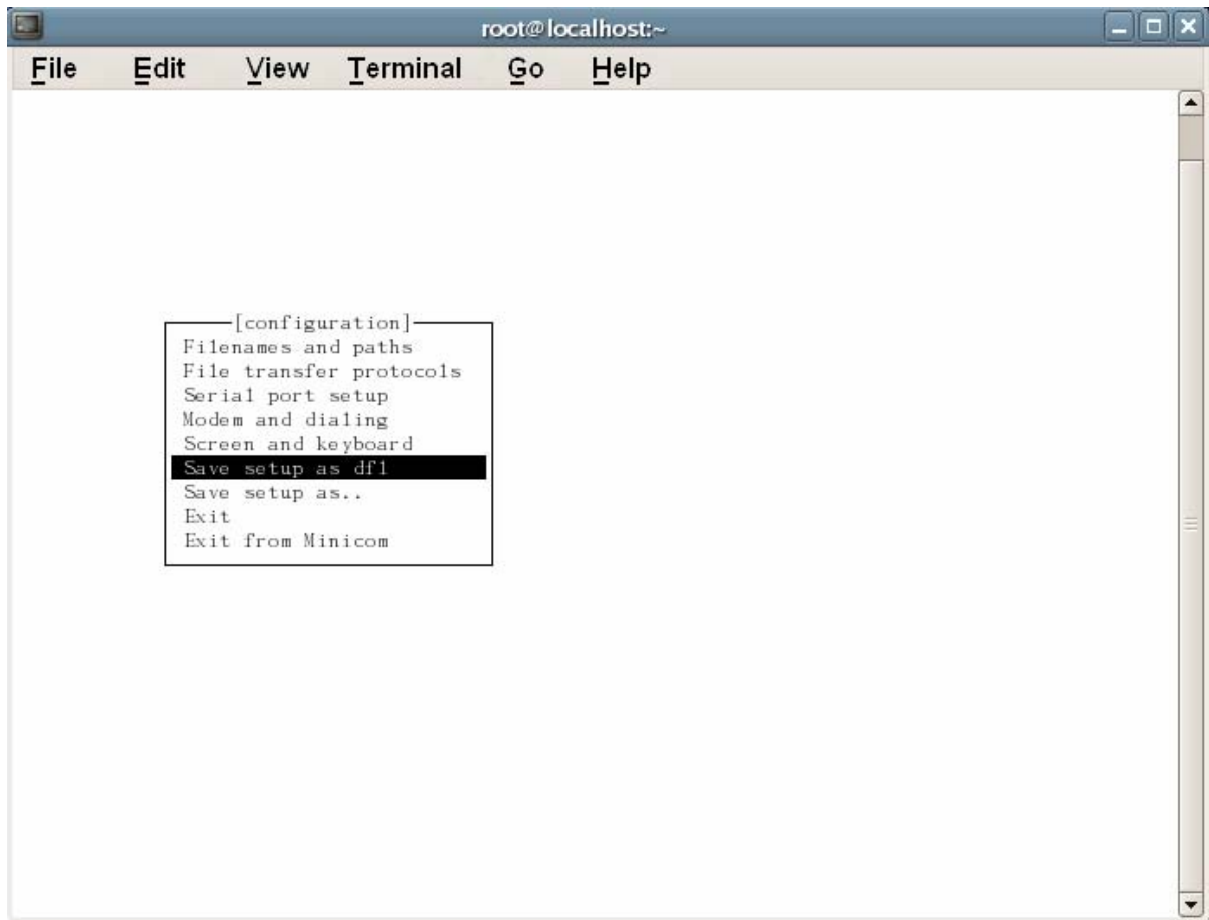
Push '**V**'to set up '**Data bits**' to **8** .

Push '**W**'to set up '**Stop bits**' to **1** .

Push '**L**' to set up '**parity**' to '**NONE**'.

Step 5

Once setting is over, please press '**Enter**' key. And select '**Save setup as df1**' item, then press '**Enter**' for saving the values.



Step 6:

To quit from **Minicom**, please press '**Ctrl + A**' and then '**Z**', at last push '**Q**' key. Then Selecting '**Yes**', **Minicom** is quitted.

5 · 2 hypertrm (windows)

Step 1:

Start -> Programs -> Accessories -> Communications -> HyperTerminal

Step 2:

Type the Terminal name. (such as “ebr”) :

Step 3 :

Select the identified serial port which is connected to target board. (If you are using **COM1**, select COM1, if **COM2**, select COM2.)

Step 4:

Setting the parameter as follows:

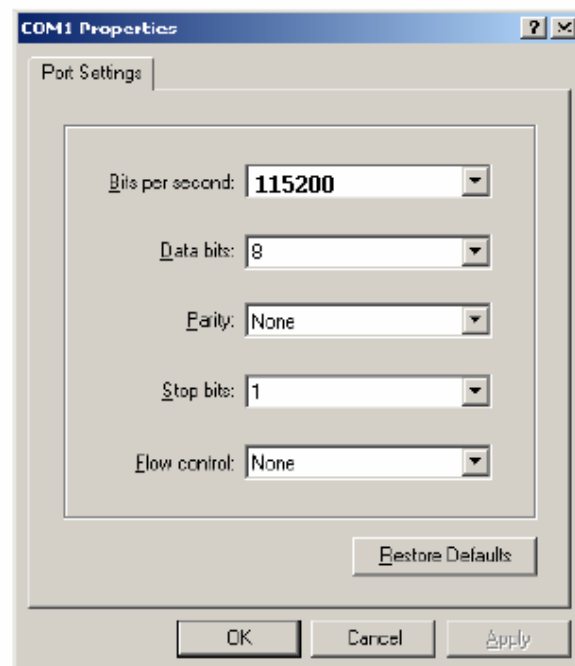
Bits per Second:115200

Data bits : 8

Parity: None

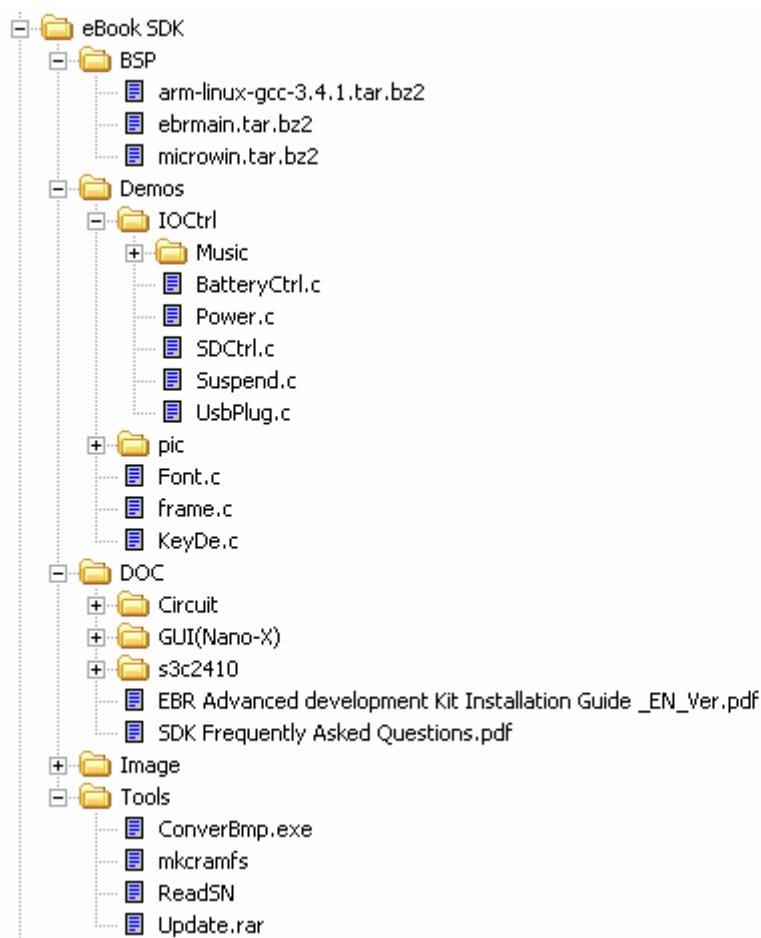
Stop bits : 1

Flow Control : None



6. BSP

6.1 File list





6.2 BSP file list

Filename	Description
arm-linux-gcc-3.4.1.tar.bz2	Toolchain
ebr_microwin.tar.bz2	GUI and ebr sample code
ebr_ebrmain.tar.bz2	Application file system
NewVivi	Bootloader Image file
NewzImage	Kernel Image file
NewRoot.cramfs	Root file system Image file
Newebr.cramfs	Application file system Image file
NewOpen.cmd	Start-Up Image file
Update.rar	Update Application
ConverBmp.exe	Make StartUp Image
mkcramfs	Make cramfs files
ReadSN	Read SN

6.3 Install Tool chain

Step 1

Create a directory for Toolchain (/usr/local/arm) , and copy file - arm-linux-gcc-3.4.1.tar.bz2 to the director.

```
[root@localhost ~]# mkdir -p /usr/local/arm  
[root@localhost ~]# cp arm-linux-gcc-3.4.1.tar.bz2 /usr/local/arm  
[root@localhost ~]# cd /usr/local/arm  
[root@localhost arm]#
```

Step 2

Decompress the file, Toolchain, and set up paths for compiler environment.
The set-up paths after decompression are /usr/local/arm/3.4.1/bin .

```
[root@localhost arm]# tar -jxvf arm-linux-gcc-3.4.1.tar.bz2  
[root@localhost arm]# export PATH=$PATH:/usr/local/arm/3.4.1/bin
```



7. Bootloader & Kernel

Bootloader (vivi) and Kernel had been built into ebr-sdk Flash ROM and should not be changed. The developer need only to be focus on application layer development skills



8. Nano-X(GUI)

Step 1:

Create a new directory for store Nano-X (microwindows). Copy and decompress the file “**ebr_microwin.tar.bz2**” at the directory. The set-up path after decompression is **/usr/local/src/ebr/microwin**. The path for ebr source code is: **/usr/local/src/ebr/microwin/src/demos/ebrmain**

```
[root@localhost test]# mkdir -p /usr/local/src/ebr
[root@localhost test]# cp ebr_microwin.tar.bz2 /usr/local/src/ebr
[root@localhost test]# cd /usr/local/src/ebr
[root@localhost ebr]# tar -jxvf ebr_microwin.tar.bz2
```

Step2:

Compile microwindows to make an ebr main program executable file and nano-x libraries(lib). The paths are:

ebr main program: **/usr/local/src/ebr/microwin/src/bin/ebrmain**

nano-x Lib : **/usr/local/src/ebr/microwin/src/lib/libnano-X.so** °

```
[root@localhost ebr]# cd microwin/src
[root@localhost src]# make clean
[root@localhost src]# make
```



9. Files system(for Application)

The file system use cramfs (compressed ROM file system) as its format. The cramfs format is read-only and compressed to get smaller size, limited to 256MB.

Step1

Copy and decompress the file “**ebr_ebrmain.tar.bz2**” into the ebr directory.
The root file path after decompression is **/usr/local/src/ebr/ebrmain**.

```
[root@localhost test]# cp ebr_ebrmain.tar.bz2 /usr/local/src/ebr
[root@localhost test]# cd /usr/local/src/ebr
[root@localhost ebr]# tar -jxvf ebr_enrmain.tar.bz2
[root@localhost ebr]# cd ebrmain
[root@localhost ebrmain]#
```

Step 2

Copy the file mkcramfs to the ebr directory. This file is a tool to make a root file system image.

```
[root@localhost tools]# cp mkcramfs /usr/local/src/ebr
[root@localhost tools]# cd /usr/local/src/ebr
[root@localhost ebr]# chmod +x mkcramfs
```



Step 3:

Copy the ebr main program and the nano-x libraries that are mentioned in Chapter 8 into the root directory.

ebr main program: Copy “ebrmain” to the directory **/usr/local/src/ebr/ebrmain**, and **change the file mode as executable**.

nano-x function libraries(lib): Copy “**libnano-X.so**” to the directory **/usr/local/src/ebr/ebrmain/lib**

```
[root@localhost test]# cd /usr/local/src/ebr/ebrmain
[root@localhost test]# cp /usr/local/src/ebr/microwin/src/bin/ebrmain ./
[root@localhost test]# chmod +x ebrmain
[root@localhost test]# cd /usr/local/src/ebr/ebrmain/lib
[root@localhost test]# cp /usr/local/src/ebr/microwin/src/lib/libnano-X.so ./
```

Step 4

Ways of converting a root file system into cramfs format:

```
[root@localhost test]# cd /usr/local/src/ebr
[root@localhost test]# ./mkcramfs ebrmain Newebr.cramfs
```

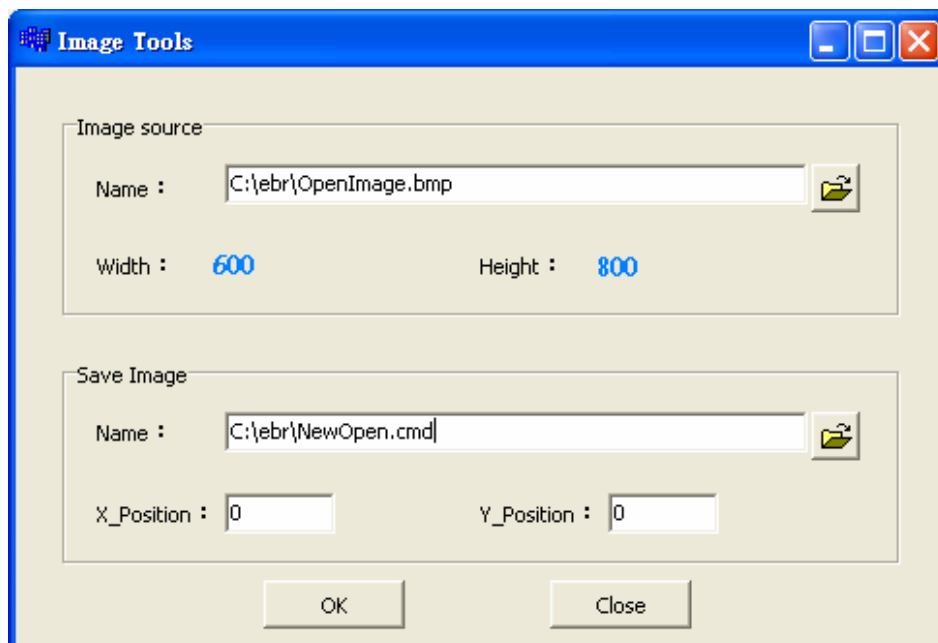
10. Make StartUp Image

Step 1

To create a boot screen image and set the file format as bitmap (with resolution 800 x 600). Save the image with the color setting as 4 gray levels.

Step 2

Click the Image Tools link. The image tools appears, showing ConverBmp.exe that have been created for the system. Select the Image source and save image Category.



Step 3.

Click OK and the boot screen image will be save file name as “NewoOpen.cmd”.

11. Update

11.1 Used SD Card Update Application

Step 1

Decompress Update_ap.rar , copy all of the files into SD Card.

Step 2

Copy Newebr.cramfs(**Please do not change the file name**) to SD Card.

Step 3

Shut down the ebr-sdk board power, and then plug in the SD Card with the files needed for update. And then power on your ebr-sdk board.

Step 4

Please continual press the “Enter” key when there is “ if you want to mount SD Card OR Nand Flash, please press’ Enter’ key” ” in minicom. And this time system will mount SD Card. Then execute Update_up procedure to Update in SD Card.

Step 5

After the update procedure is finished, it will show “Update Successful” on panel, if the update procedure is successful; otherwise, it will show “Update Failed” on panel and end the process.

Step 6

If the update process is successful, don’t remove the SD card and restart the ebr sdk. The ebr sdk will check the updated data. If there is no need of file update, it will switch to execute the ebr application. Otherwise, it will re-start the update procedure automatically (step 4)

11.2 Automatism startup new ebrmain which in SD Card :

Step 1

Please change the directory path of source codes which includes used fonts, pictures and files to the relative path. For example,

(1)/pic/close.bmp => ./pic/close.bmp

(2)/fonts/mfont.fnt => ./fonts/mfont.fnt

Step 2

Change the current directory to /usr/local/src/ebr/microwin/src

```
[root@localhost test]# cd /usr/local/src/ebr/microwin/src  
[root@localhost test]#
```

Step 3

Recompile again

```
[root@localhost test]# make clean  
[root@localhost test]# make
```

Step 4

Please copy /usr/local/src/ebr/microwin/src/bin/ebrmain to the directory /usr/local/src/ebr/ebrmain, and then copy /usr/local/src/ebr/microwin/src/lib/libnano-X.so to the directory /usr/local/src/ebr/ebrmain/lib. After that, please change the current directory to



/usr/local/src/ebr', and set the permission right of ebrmain (Please don not change the name of ebrmain and libnano-X.so)

```
[root@localhost test]# cp bin/ebrmain /usr/local/src/ebr/ebrmain
[root@localhost test]# cp lib/libnano-X.so /usr/local/src/ebr/ebrmain/lib
[root@localhost test]# cd /usr/local/src/ebr/
[root@localhost test]# chmod +x ebrmain/ebrmain
```

Step 5

Please create a text file whose name is 'test' at /usr/local/src/ebr/ebrmain and the context of file as below list.

```
cd /mnt/ext2
./ebrmain
cd /
```

Step 6

Please set the permission right of the file, test.

```
[root@localhost test]#chmod +x test
[root@localhost test]#
```

Step 7

Please copy the all things and directories of /usr/local/src/ebr/ebrmai to SD card

Step 8

After putting the SD card into the EBR SDK Board, you should reboot the EBR SDK Board. Please continual press the "Enter" key when there is " if you want to mount SD Card OR Nand Flash" in minicom. And this time system will mount SD Card.and then the board will execute the new ebrmain of SD Card.

11.3 Used SD Card Update StartUp

Image

Step 1

Decompress Update.rar and copy all of the files into SD Card.

Step 2

Copy NewOpen.cmd(**Please do not change the file name**) to SD Card.

Step 3

Shut down the ebr-sdk board power, and then plug in the SD Card with the files needed for update. And then power on your ebr-sdk board.

Step 4

Please continual press the “Enter” key when there is “ if you want to mount SD Card OR Nand Flash, please press’ Enter’ key” in minicom. And this time system will mount SD Card. Then execute Update_up procedure to Update in SD Card.

Step 5

After the update procedure is finished, it will show “Update Successful” on panel, if the update procedure is successful; otherwise, it will show “Update Failed” on panel and end the process.

Step 6

If the update process is successful, don't remove the SD card and restart the ebr sdk. The ebr sdk will check the updated data. If there is no need of file update, it will switch to execute the ebr application. Otherwise, it will re-start the update procedure automatically (step 4)

11.4 Used IC Write

ROM Map:

File name	Address
NewVivi	0x000000
NewzImage:	0x030000
NewRoot.cramfs	0x130000
Newebr.cramfs	0x2A0000
NewOpen.cmd	0x7E0000

Please use IC Write and follow ROM map address instruction to write the image files into ROM

The case uses LEAPER-48,a type of LEAP electronic, and LP48-TSOP-48S, a type of Flash ROM for explaining

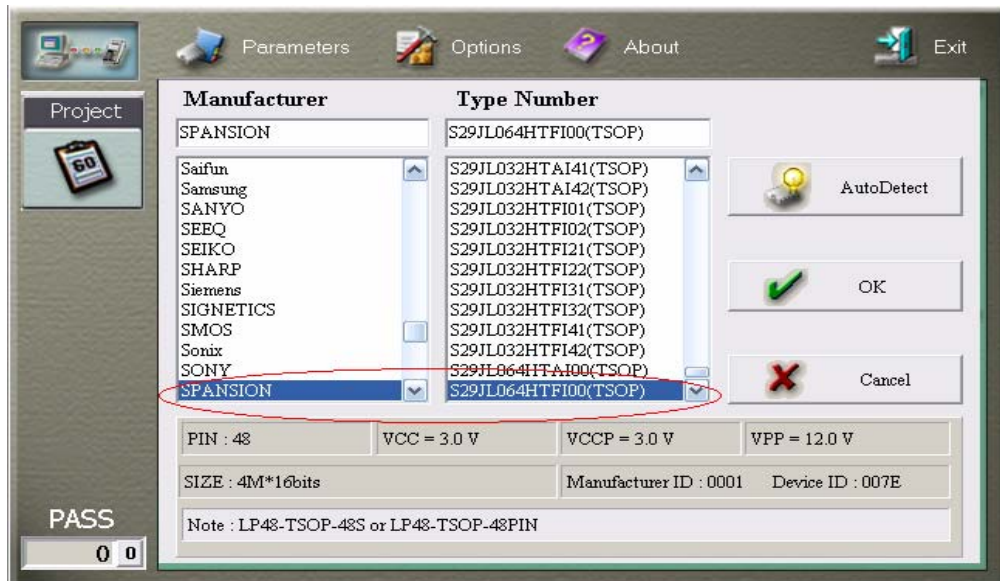
Step 1

When you start the program, you could see a image on the screen as below picture.



Step 2

Please choose the kind of Flash ROM and then click the [OK] button



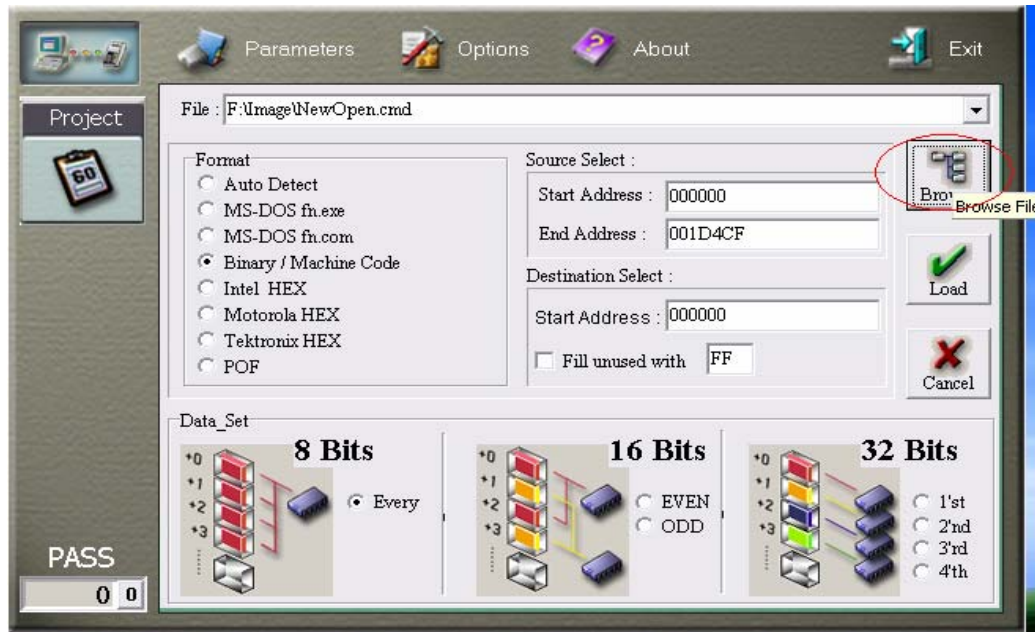
Step 3

Please select [Load] icon.



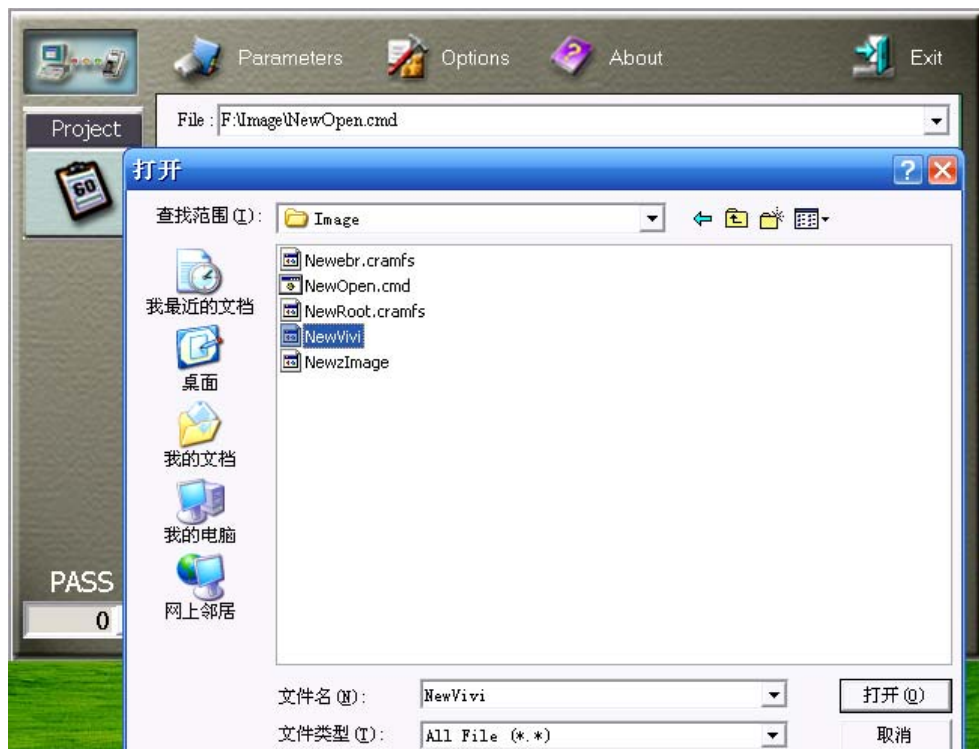
Step 4

Please choose the [Browse] button



Step 5

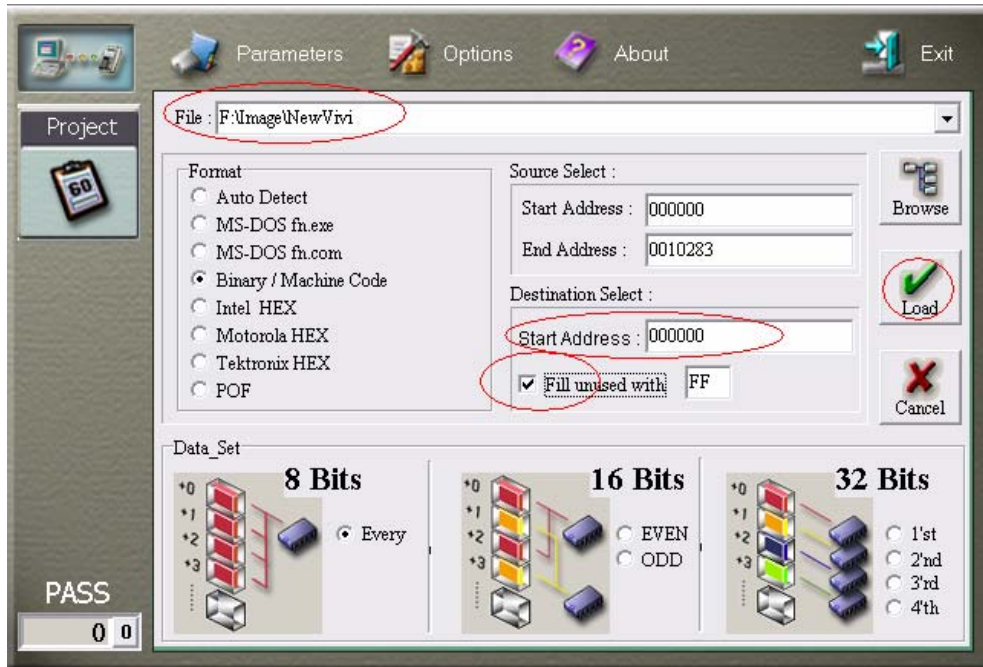
Please select the file, 'NewVivi'





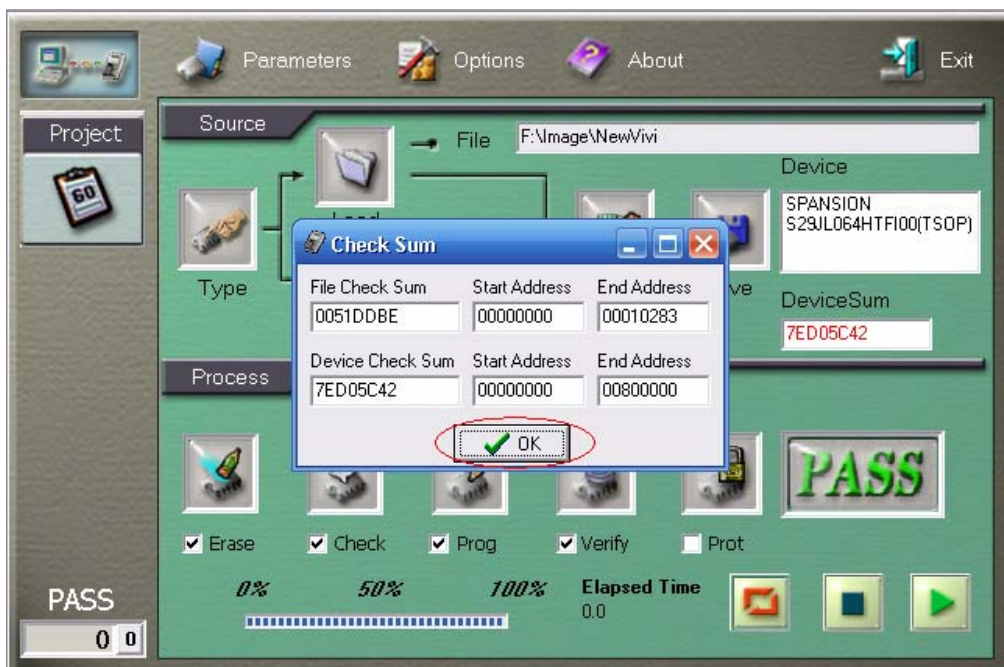
Step 6

Please set the NewVivi address at 0x000000 , and choose the [Fill unused with] , and then click the [Load] button



Step 7

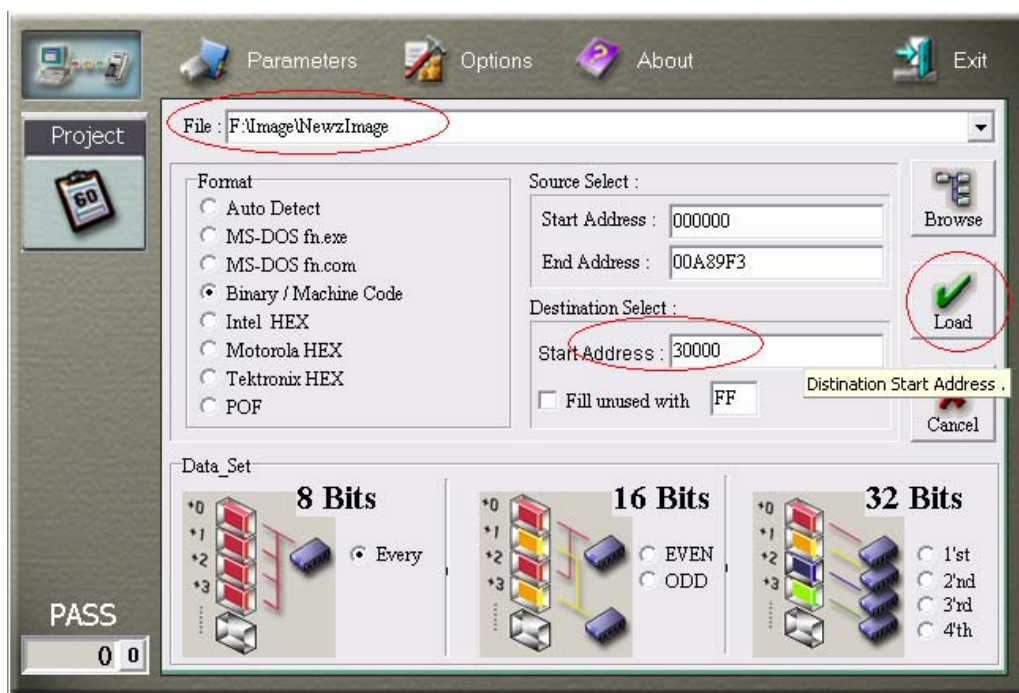
Choose the OK button



Step 8

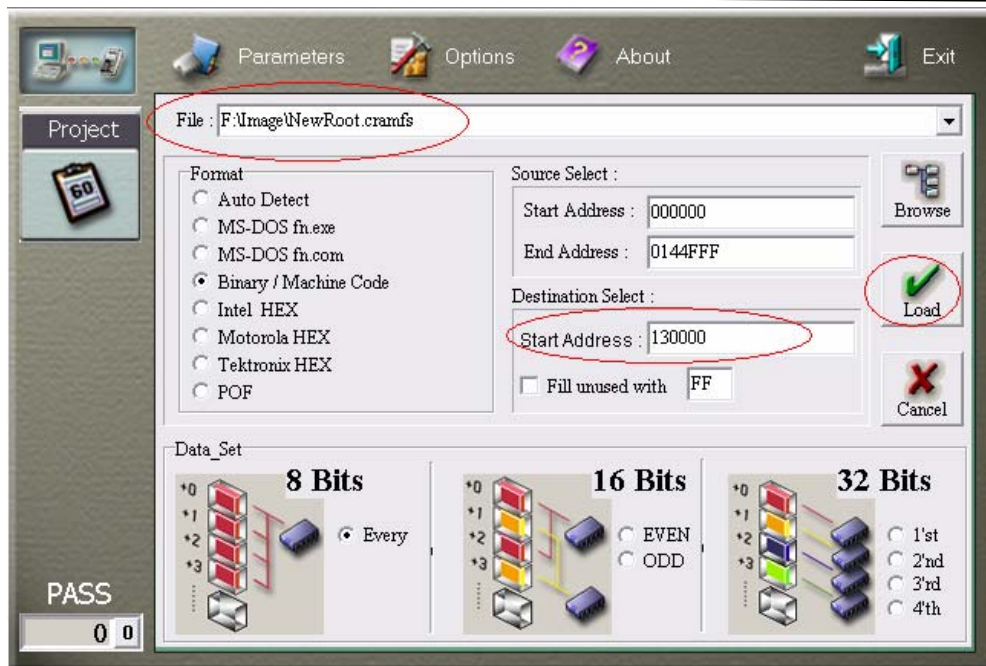


Please Follow the step 3~5 to add on the file, NewzImage, and set the address at 0x30000, and then do not choose [Fill unused with] option.



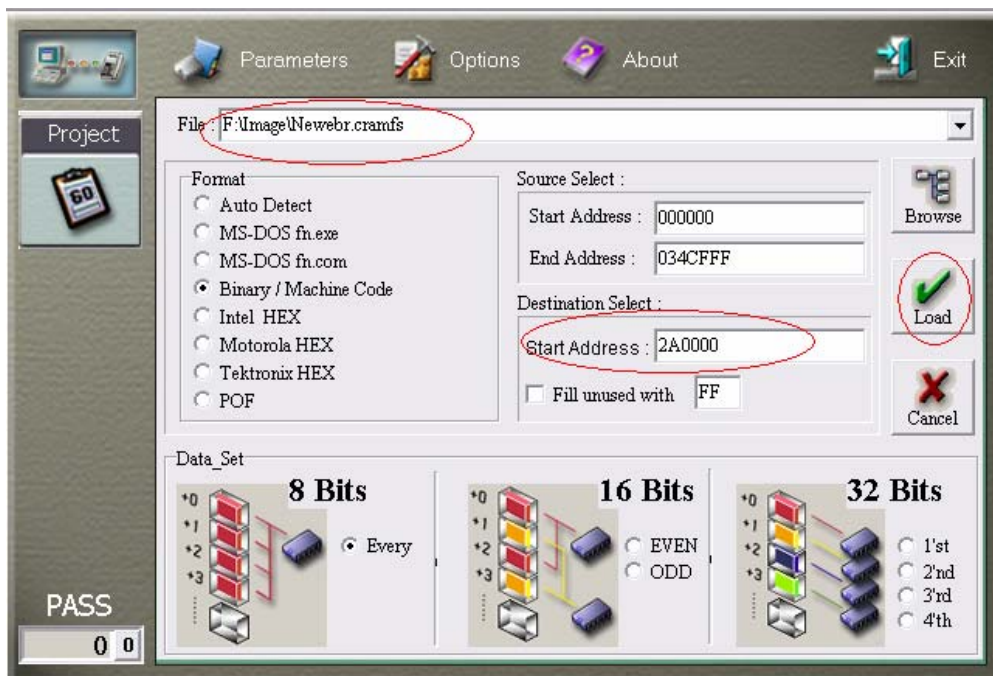
Step 9

Please Follow the step 3~5 to add on the file, NewRoot, and set the address at 0x130000, and then do not choose [Fill unused with] option.



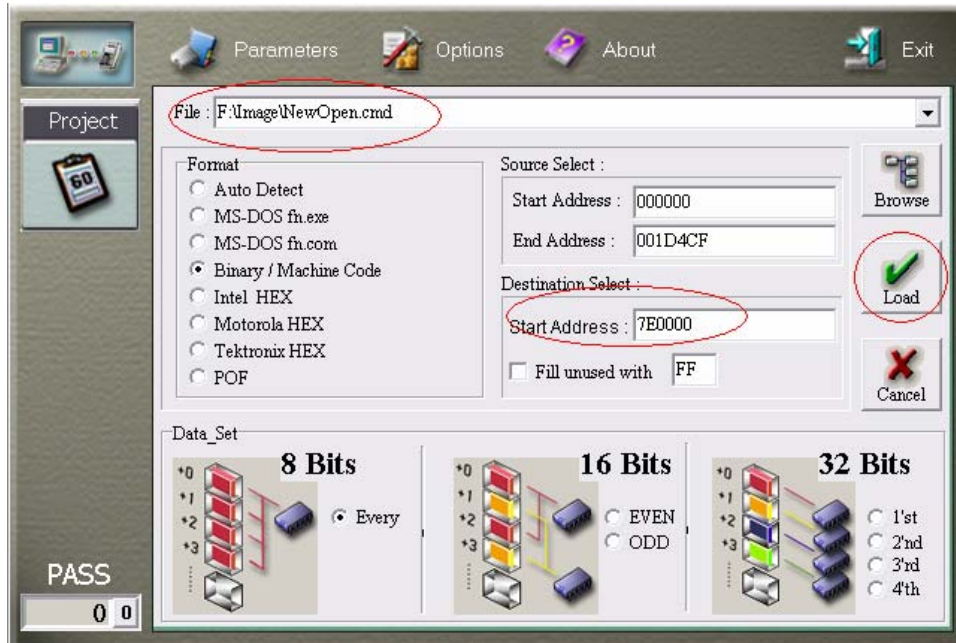
Step 10

Please Follow the step 3~5 to add on the file, Newebr.cramfs, and set the address at 0x2A0000, and then do not choose [Fill unused with] option.



Step 11

Please Follow the step 3~5 to add on the file, NewOpen.cmd, and set the address at 0x7E0000, and then do not choose [Fill unused with] option



Step 12

After all the files are loaded, the Flash ROM will start to write the data. If it's success, the word, 'PASS', will be shoed on screen.



13. To detect the serial number

Every SDK board has a unique serial number. The users could get the serial number to compare with the serial number of Flash ROM label if the numbers are the same. You just need to follow the blow steps:

Step 1

Please copy the document, 'ReadSN', which is under the directory, 'Tools', to SD card.

Step 2

Please put into the SD Card and then to execute minicom and to start EBR. if you environment is in windows, you should execute hyperterm. At this time, you could see the message on the screen as below picture.

```
scsi0 : SCSI emulation for USB Mass Storage devices
  Vendor: STAR      Model: eBOOK STK-101 -0  Rev: 1.00
  Type:   Direct-Access          ANSI SCSI revision: 02
cd: can't cd to /ebrmain/led
/usr/etc/rc.local: ./nled1: No such file or directory
Error -3 while decompressing!
c01617b4(2035972279)->c0cc4000(4096)
./ebrmain: error while loading shared libraries: ./ebrmain: undefined symbol: Gr
Open
Waiting for enter to start '/bin/sh' (pid 46, terminal /dev/console)

Please press Enter to activate this console.  Vendor: STAR      Model: eBOOK ST
K-101 -1  Rev: 1.00
  Type:   Direct-Access          | ANSI SCSI revision: 02
Attached scsi removable disk sda at scsi0, channel 0, id 0, lun 0
Attached scsi removable disk sdb at scsi0, channel 0, id 0, lun 1
SCSI device sda: 1984000 512-byte hdwr sectors (1016 MB)
sda: Write Protect is off
Partition check:
/dev/scsi/host0/bus0/target0/lun0: p1
SCSI device sdb: 128000 512-byte hdwr sectors (66 MB)
sdb: Write Protect is off
/dev/scsi/host0/bus0/target0/lun1: unknown partition table
-
```

Step 3

When the process of program was finished, you need to push "Ctrl+C" and then push "Enter" on the keyboard. The Shell of SDK system will show out.



```
Please press Enter to activate this console.  Vendor: STAR      Model: eBook ST
K-101 -1 Rev: 1.00
Type: Direct-Access                      ANSI SCSI revision: 02
Attached scsi removable disk sda at scsi0, channel 0, id 0, lun 0
Attached scsi removable disk sdb at scsi0, channel 0, id 0, lun 1
SCSI device sda: 1984000 512-byte hdwr sectors (1016 MB)
sda: Write Protect is off
Partition check:
/dev/scsi/host0/bus0/target0/lun0: p1
SCSI device sdb: 128000 512-byte hdwr sectors (66 MB)
sdb: Write Protect is off
/dev/scsi/host0/bus0/target0/lun1: unknown partition table
Process '/bin/sh' (pid 45) exited. Scheduling it for restart.? Waiting for ent
er to start '/bin/sh' (pid 49, terminal /dev/console)

Please press Enter to activate this console.
Starting pid 49, console /dev/console: '/bin/sh'

BusyBox v0.60.3 (2002.05.13-08:36+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

#
```

Step 4

Please key in the command, 'df', to determine if the system already mounted the SD Card. if SD Card is not mounted, the way you could do it by manual mounting is as below list. The "/dev/discs/disc0/part1" is the SD card's device name but the driver name may different in some cases.

```
# mount -t vfat /dev/discs/disc0/part1 /mnt/ext2
```

OR

```
# mount -t vfat /dev/discs/disc0/part4 /mnt/ext2
```

If the action of mounting SD Card is succeed, you could use the command, 'df', to see the result which is like the below picture. By the way, the SD Card should be mounted at /mnt/ext2/ directory.



```
# df
Filesystem            1k-blocks      Used Available Use% Mounted on
/dev/mtdblock/2        2948          2948         0 100% /
tmpfs                  7320           0        7320   0% /dev/shm
/dev/mtdblock/5        6652          6652         0 100% /ebrmain
tmpfs                  7320           0        7320   0% /dev/shm
# mount -t vfat /dev/discs/disc0/part1 /mnt/ext2
modprobe: modprobe: Can't open dependencies file /lib/modules/2.4.18-rmk7-pxa1/modules.dep (No such file or directory)
modprobe: modprobe: Can't open dependencies file /lib/modules/2.4.18-rmk7-pxa1/modules.dep (No such file or directory)
# df
Filesystem            1k-blocks      Used Available Use% Mounted on
/dev/mtdblock/2        2948          2948         0 100% /
tmpfs                  7320           0        7320   0% /dev/shm
/dev/mtdblock/5        6652          6652         0 100% /ebrmain
tmpfs                  7320           0        7320   0% /dev/shm
/dev/discs/disc0/part1 989792       139156     850636   14% /mnt/ext2
# _
```

Step 5

Finally, you need to change the current directory to the SD Card's directory and execute the program, 'ReadSN', for comparing the serial number with the Flash ROM label.

```
[root@localhost test]#cd /mnt/ext2
[root@localhost test]#chmod +x ./ReadSN
[root@localhost test]#./ReadSN
```

15. Sample code

The path of ebr source code is

/usr/local/src/ebr/microwin/src/demos/ebrmain.

Hot keys for operating the ebr sample:

F2: Font Demo

F3: stop slideshow

Left: backward

Right: forward

Enter: slideshow

16. Demo exposition

16.1 Font (**Demos\Font.c**)

In this case, we try to show a font and control the code of changing. The changing of font primarily controls by buttons. If you push the [left] or [bottom] button down, the size of font will decrease. In contract with this, if you put the [Right] or [Top] button down, the size of font will increase in a loop from 24.28.32.36 to 40. You could just follow the below steps to setting the front size.

```
GR_FONT_ID font;          //define font

font = GrCreateFont("mfont.mbf",0, NULL);    //load the font

GrSetFontSize(font,fontsize);    //set the size of font

GrSetGCFont(gc,font);          //use the font
```

Firstly, you could use the function, GrCreateFont(), to load font. Secondly, you may use the function, GrSetFontSize(), to set the size of font. Finally, you can use the function, GrSetGCFont(gc,font), to apply the font just set before to the graphic structure ,which is pointed by gc and the font size is 24. If you want to know more in detail, please see the document at Demos\Font.c.

16.2 Keyboard (**Demos\KeyDe.c**)

This document includes a case of how to control keyboard. The controls of keyboard include left, right, bottom, Enter, F1, F2 ,F3, F4, Volume + and Volume -. If



you try to put any button down, it will be showed on E-ink. In EBR, only the 10 buttons, which have just listed, are defined the specific key value. Otherwise, the other buttons like Power and Reset are not defined and the Reset is not controlled by progress. If you want to know how to control the Power button, please refer the part of controlling I/O of Power in the book.

At the fact, the drivers of EBR core already do the most of tasks. (However, it not includes Rest and Power) When a button is pushed, the button key will be kept by the drivers to an event.keystroke.ch (event is a structure of GR_EVENT). The work that AP codes need to make is to get an event structure and then base on the value of event.keystroke.ch to define which button was pushed.

```
GR_EVENT event;           //to define event structure

GrCheckNextEvent(&event);  //to get the event context

switch(event->type){        // to get the type of event

    case GR_EVENT_TYPE_KEY_DOWN://if it is the event of pushing button

        switch(event->keystroke.ch){           //to check the type of keyboard

            case 63490:                          //if it is Top

                .....                            //to handle Top's event

            break;

        }

    }

}
```

If you want to know more in detail, please see the document at Demos\KeyDe.c.



16.3 Display Frame (**Demos\Frame.c**)

This document includes a case of how to control frames. The beginning window, which includes Check A and Check B two options, could be chose the options as you want. When you choose Check B and push the [Enter], you will jump into next window. You can push the [F3] to go back previous window.

The method of changing frame is that you can remove the black frame of original option (it means you can put a while frame to cover the black frame.)and then put a black frame to the option which you want.

The frame's code :

```
GrSetGCBackground(gc,WHITE); //set background color
```

```
GrSetGCForeground(gc,WHITE); //set foreground color
```

```
GrRect(wid,gc,x, y, width,height);
```

```
//at the point(x,y) to paint a rectangle
```

The code of showing pictures :

```
GrDrawImageFromFile ( wid,gc,x,y,width,height,path,0 ) ;
```

```
//At point(x,y) to show a rectangle picture which is pointed by path.
```

The above mentioning units are all pixels. The E-ink range are 600*800 pixels. If you need the more things in detail, please see the Demo\Frame.c

17. I/O control

17.1 、 The control of Power button (Demos\IOCtrl\Power.c)

When the user pushes down the power button, what will the system operate? It depends on the codes. For example, the process is to shutdown in power.c. there are two parts of this issue.

1. How could I get the Power button's status? It's showed by the below codes.

```
#define CM_SW_OFF  110

#define CM_PWR      112

int value = 0;

int fpid = open("/dev/pvi_ioc",O_RDONLY|O_NONBLOCK);

ioctl(fpid, CM_SW_OFF,&value);
```

If the value=0, that means the Power was pushed.

2. What will the system do when the power button is pushed? It's showed by the below codes.

```
ioctl(fpid, CM_PWR,0);    //close the power fpid

close(fpid);              //if you can't shutdown the power, please to use the
system("reboot") .
```

Warning!! Before you shutdown, you should un-mount SD Card and /mnt/ext1. if you want to know more in detail, please see the file, Demos\IOCtrl\Power.c,.

17.2 、 Automatism mount or umount SD Card (SDCtrl.c)



The codes show that after the system start, if you plug in or out the SD Card, the system will mount or un-mount SD Card automatically. The process includes two parts.

1. To check the SD Card if plug in EBR. The below codes show how to get the SD card's status.

```
#define CM_SD_IN    117

int SD_value=0;

int pio_fd = open("/dev/pvi_ioc",O_RDONLY | O_NONBLOCK);

ioctl(pio_fd, CM_SD_IN,&SD_value);

close(pio_fd);
```

SD_value=1 means the SD Card is plugging in EBR. Otherwise, the SD_value=0 means the SD Card is not in EBR.

2. The system will mount or un-mount SD Card when getting the SD Card status in this sample.

In addition to the codes, the SD Card's Device name might be /dev/discs/disc0/part1 or /dev/discs/disc0/part4. General speaking, it would add on /mnt/ext2. if you need the more in detail, you can see the document on Demos\IOCtrl\SDCtrl.c

17.3 、 Suspend mode(Demos\IOCtrl\Suspend.c)

When the system detect when the user does not have any action over a range of time, the system will switch into suspend mode. The codes show the system will switch on suspend mode if the user does not have any action in 30 seconds.



The step of switching suspend mode:

1. At beginning the time counter count time and detect the customer input, if the user don't have any input, the counter add . Otherwise, if the user has action, the counter should be reset.
2. If the time counter keep counting to 30, the system switch on suspend mode. The method how to switch on suspend mode is to use (/lib/modules/suspend.sh). For example, `system("/lib/modules/suspend.sh");`
3. If the system switch on suspend mode, it will not switch off until the buttons in EBR is touched.

If you want to get more in detail, please see the document on
Demos\IOCtrl\Suspend.c

17.4 、 PC mount EBR (Demos\IOCtrl\UsbPlug.c)

The process includes two parts.

1 、 To get if EBR connected with PC

```
#define CM_USB_PC 1

#define CM_USB_EBR 2

#define CM_USB_PLUG_IN 108

int Usb_plug=0;

int fpid = open("/dev/pvi_ioc",O_RDONLY|O_NONBLOCK);

ioctl(fpid, CM_USB_PLUG_IN,&Usb_plug);

close(fpid);
```




Usb_plug = 1 means EBR connected with PC.

- 2、 **How should we do when the device connect with PC? The below codes show a process of a USB device is mounted by PC.**

```
int pio_fd;

umount2("/mnt/ext1",0);    //umount /mnt/ext1

umount2("/mnt/ext2",0);    //umount /mnt/ext2

pio_fd = open("/dev/pvi_ioc",O_RDONLY | O_NONBLOCK);

ioctl(pio_fd, CM_USB_PC,0);

close(pio_fd);
```

Warning!! You have to un-mount SD Card and /mnt/ext1 first before you do the above mentioned actions. PC will see EBR as a USB device. If EBR system does not un-mount the SD Card and /mnt/ext1 which have mounted, it might cause the breaking of the FAT file system. If you want to know the more in detail, please the document on Demos\IOCtrl\UsbPlug.c

17.5、 Battery Level (Demos\IOCtrl\BatteryCtrl.c)

The process includes two parts to get the battery level. :

- 1、 connected with battery
- 2、 **if EBR use Battery, we can get the value of battery.**

```
adc_fd = open("/dev/ebr_adc",O_RDONLY | O_NONBLOCK);

SysPower = ioctl(adc_fd,2,7);
```

SysPower is the remaining value of battery °

the remaining value of battery is not linear stretch, so we offer a method to define the range



of the remaining value of battery.

When playing mp3, you could get the value of SysPower.

SysPower > 619 the battery is full. The power level of battery is 5.

SysPower > 595 & SysPower <= 619 the power level of battery is 4.

SysPower > 579 & SysPower <= 595 the power level of battery is 3.

SysPower > 573 & SysPower <= 579 the power level of battery is 2.

SysPower > 562 & SysPower <= 573 the power level of battery is 1.

SysPower <= 562 the battery out of juice. The power level of battery is 0.

When not playing mp3, you still could get the value of SysPower.

SysPower > 623 the battery is full. The power level of battery is 5.

SysPower > 599 & SysPower <= 623 the power level of battery is 4.

SysPower > 583 & SysPower <= 599 the power level of battery is 3.

SysPower > 577 & SysPower <= 583 the power level of battery is 2.

SysPower > 566 & SysPower <= 577 the power level of battery is 1.

SysPower <= 566 the battery out of juice. The power level of battery is 0.

The range of SysPower' s level just for reference only, you could redefine the value by real situations.

If you want to know more in detail, please see the document on Demos\IOCtrl\BatteryCtrl.c

17.6 、 the control of voice (Demos\IOCtrl\Music\PlayMusic.c)

This is a simple music player code whose the main functions include to play music and to adjust volume. There are two functions, play_mp3(char *mp3file) and VolumeSet (int vol), for implementing this two functions.

mp3(char *mp3file) :The parameter of the function is the music's path

VolumeSet(int vol) :The parameter of the function is the range of volume form 0 to 9



In this case, the way we do is to create a new thread to play music (the music is put at mnt/ext1/ Take_Me_To_Your_Heart.mp3) and to set volume at main progress. When playing music, you also can adjust the volume (by using the V+ and V- buttons). When you do this way, you need put the music file at /mnt/ext1.

If you want to know more in detail, please see the document on
Demos\IOCtrl\Music\PlayMusic.c

17.7、 Auto-off Control (Demos\IOCtrl\autooff.c)

The code will execute auto-off function and auto-off time setting by user. Otherwise auto-off functions is executed by the RTC of s3c2410. Fulfill the steps of functions is as following:

Step 1 : Open RTC

```
int rtc_fd = -1;

rtc_fd = open("/dev/misc/rtc",O_RDONLY | O_NONBLOCK);

if(rtc_fd <0) return 1;
```

Step 2 : Read System Time

```
struct rtc_time wake_up_time;

ioctl(fd_rtc,RTC_RD_TIME,&wake_up_time);
```

Step 3 : Setup System Auto-off time (Turn off time: 1 minutes)

```
wake_up_time.tm_min += 1;
```

Must consider current time and power off time weather time is carry. e.g: the current time is 20:59:30, and hour hand is up If minute hand is up 1 minutes

Step 4: Setup Alarm time, the alarm interrupt is activated



```
ioctl(fd_rtc,RTC_ALM_SET,&wake_up_time);
```

```
ioctl(fd_rtc,RTC_AIE_ON,0);
```

Step 5 : the system RCT will be interrupt by setup alarm as the setup time is time out. And read setup time and off-interrupt.

```
ioctl(fd_rtc,RTC_AIE_OFF,0);
```

```
ioctl(fd_rtc,RTC_RD_TIME,&now_time);
```

```
close(fd_rtc);
```

Step 6 : Compare Current Time and Setup Time. Power off action if it is same.

```
int fpid = open("/dev/pvi_ioc",O_RDONLY|O_NONBLOCK);
```

```
ioctl(fpid,CM_PWR,0);      //CM_PWR= 112
```

```
close(fpid);
```

Details: Please ref: Demos\IOCtrl\autooff.c °