Installation Manual for S3C2410 (Linux)

Installation Manual for S3C2410 (Linux)

Though every care has been taken to ensure the accuracy of this document, Samsung Electronics Co, Ltd. cannot accept responsibility for any errors or omissions or for any loss occasioned to any person, whether legal or natural, from acting, or refraining from action, as a result of the information contained herein. Information in this document is subject to change at any time without obligation to notify any person of such changes.

Samsung Electronics Co, Ltd. may have patents or patent pending applications, trademarks copyrights or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give the recipient or reader any license to these patents, trademarks copyrights or other intellectual property rights.

No part of this document may be communicated, distributed, reproduced or transmitted in any form or by any means, electronic or mechanical or otherwise, for any purpose, without the prior written permission of Samsung Electronics Co, Ltd.

The document is subject to revision without further notice.

All brand names and product names mentioned in this document are trademarks or registered trademarks of their respective owners.

Contact Address

Samsung Electronics Co., Ltd.
San #24 Nongseo-Ri, Giheung-EUP,
Yongin- City, Gyeonggi-Do, Korea
C.P.O Box #37, Suwon 449-900
Home Page: http://www.samsungsemi.com

Revision History

Date	Version	Author	Amendment
2004-04-30	1.0	TS Team	Preliminary Copy
2004-05-07	1.1	TS Team	Document Update

Contents

Figures

1 Downloading BSP

Please log in into Super user mode and add the user.
For egs, to add new user as '**test**', please follow the following commands.

```
[root@localhost root]# adduser test
[root@localhost root]# passwd test
Changing password for user test:
New Password:            --Enter the password for user 'test' as you wish.
```

Please download the source code for S3C2410 from http://www.samsung.com/Products/Semiconductor/SystemLSI/MobileSolutions/MobileASSP/MobileComputing/S3C2410X/S3C2410X.htm and copy it to the working directory ***/home/test***. Following are the necessary files for S3C2410.

Filename	Description
cross-2.95.3.tar.bz2	Toolchain
Ztelnet-0.9.1-7mz.i386.rpm	RPM
s3c2410_vivi_r1.1.tar.bz2	bootloader
s3c2410_kernel2.4.18_r1.1.tar.bz2	kernel
s3c2410_kernel2.4.18_module_mmc.tar.bz2	MMC
root.cramfs	Small Size root file system image (Only for Booting)
root_qtopia.cramfs	Qtopia window Root file System image

Below is the list of downloaded files from the Samsung website.

```
[root@localhost test]#
[root@localhost test]# ls
cross-2.95.3.tar.bz2
root.cramfs
root_qtopia.cramfs
s3c2410_kernel2.4.18_module_mmc.tar.bz2
s3c2410_kernel2.4.18_r1.1.tar.bz2
s3c2410_vivi_r1.1.tar.bz2
Ztelnet-0.9.1-7mz.i386.rpm
```

2 Installing Toolchain

Building the tool chain is not a trivial exercise and for most common situations pre-built tool chains already exists. Unless you need to build your own, or you want to do it anyway to gain a deeper understanding, then simply installing and using a suitable ready-made tool chain is strongly recommended.

Please follow the commands below and Install the tool chain in the directory mentioned below.

```
[root@localhost test]# mkdir –p /usr/local/arm
[root@localhost test]# tar jxvf cross-2.95.3.tar.bz2
```

The above command will generate the **“2.95.3”** folder under the /test/ directory. Copy this folder under **“/usr/local/arm/”** directory.

```
[root@localhost test]# mv 2.95.3 /usr/local/arm/
```

```
[root@localhost test]# export PATH=$PATH:/usr/local/arm/2.95.3/bin
```

The toolchain object files such as arm compilers, loaders etc. will be available in the ***‘/usr/local/arm/2.95.3/bin’*** directory.

3 Compiling Bootloader and Kernel for SMDK2410

3.1 Introduction to Bootloader

In embedded system, general firmware like CMOS does not exist. So to boot embedded system for the first time, we have to make bootloader which can adjust well to target board.

Bootloader plays a very important part in embedded system. The role of bootloader is explained below.

- Copy kernel to RAM from flash memory, and execute kernel.
- Initialize hardware.
- Bootloader have the function that writing data to flash memory. (Downloading kernel or Ram disk by serial port or other network hardware, data is stored in RAM. But RAM loses all data downloaded if you remove the power supply, so to avoid this work you have to store to flash memory.)
- It provides interface to send commands to target board or to inform user's state of target board.

3.1.1 What is Vivi

Vivi is bootloader made to use exclusively at ARM line processor. Because vivi supports only serial interface, to communicate between host PC and embedded system, you have to connect host PC to target board by serial cable and execute Minicom.

3.2 Compiling Vivi

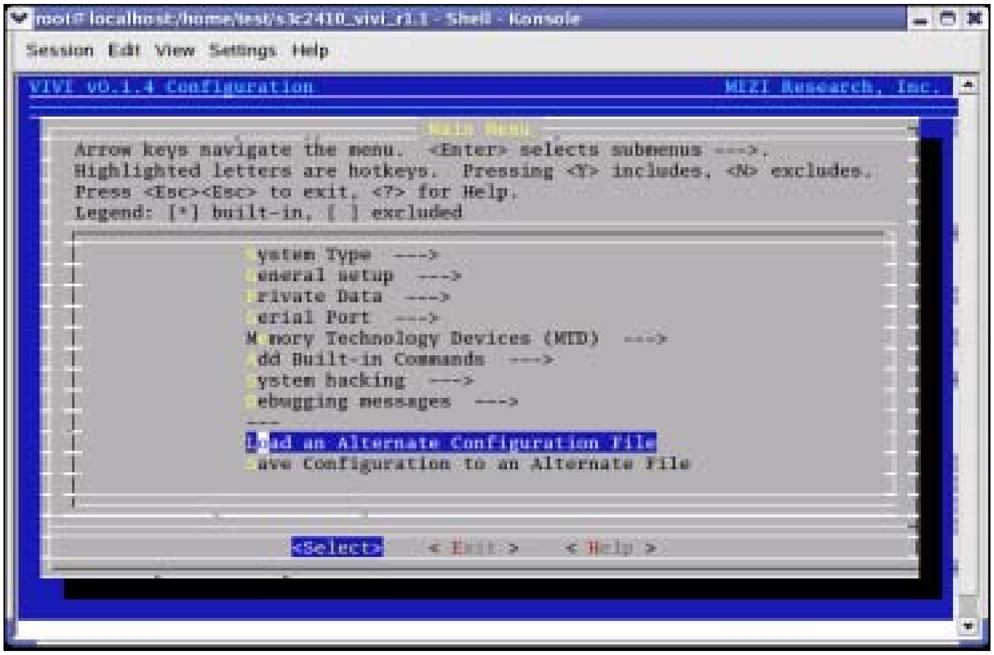
Vivi source file is compressed with tarball, ***'s3c2410_vivi_r1.1.tar.bz2'***. Extract it executing following command.

```
[root@localhost test]#
[root@localhost test]# tar jxvf s3c2410_vivi_r1.1.tar.bz2
```

Go to '**s3c2410_vivi_r1.1'** directory created after extracting the tarball and then execute the following commands.

```
[root@localhost test]# cd s3c2410_vivi_r1.1
[root@localhost s3c2410_vivi_r1.1]# make menuconfig
```

Please Select ***'Load an Alternate Configuration File'*** as shown in figure 3-1.

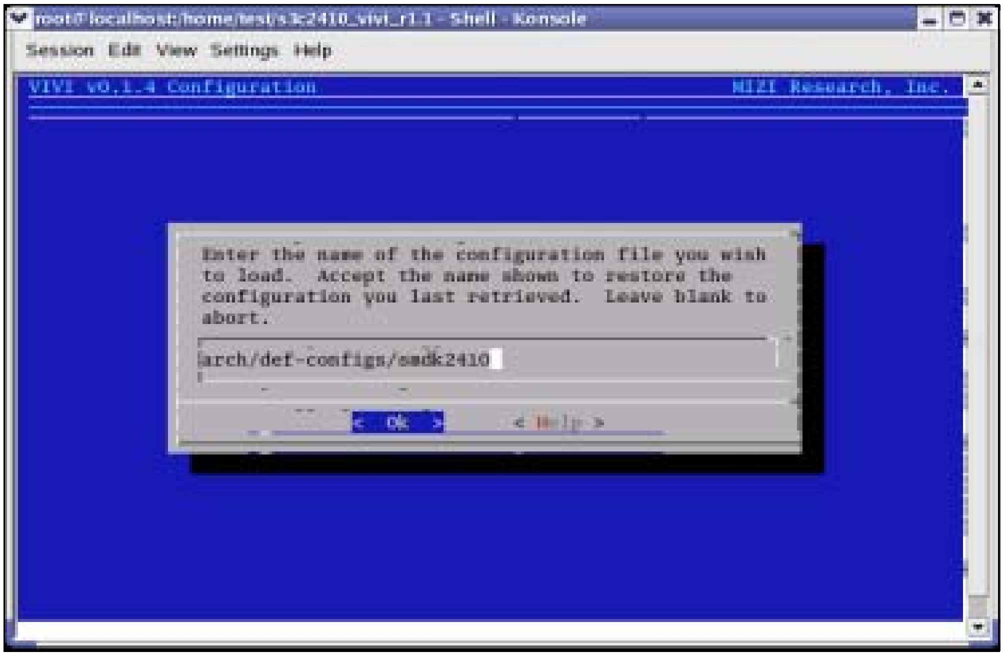


root@localhost:/home/test/s3c2410_vivi_r1.1 - Shell - Konsole
Session Edit View Settings Help

```
VIVI v0.1.4 Configuration                                   MIZI Research, Inc.

                              Main Menu
   Arrow keys navigate the menu.  <Enter> selects submenus --->.
   Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes.
   Press <Esc><Esc> to exit, <?> for Help.
   Legend: [*] built-in, [ ] excluded

                    System Type  --->
                    General setup  --->
                    Private Data  --->
                    Serial Port  --->
                    Memory Technology Devices (MTD)  --->
                    Add Built-in Commands  --->
                    System hacking  --->
                    Debugging messages  --->
                    ---
                    Load an Alternate Configuration File
                    Save Configuration to an Alternate File

                 <Select>    < Exit >    < Help >
```

Figure 3-1 Vivi configuration

Please enter the name of the configuration file you wish to load ***'arch/def-configs/smdk2410'*** as shown in figure 3-2.

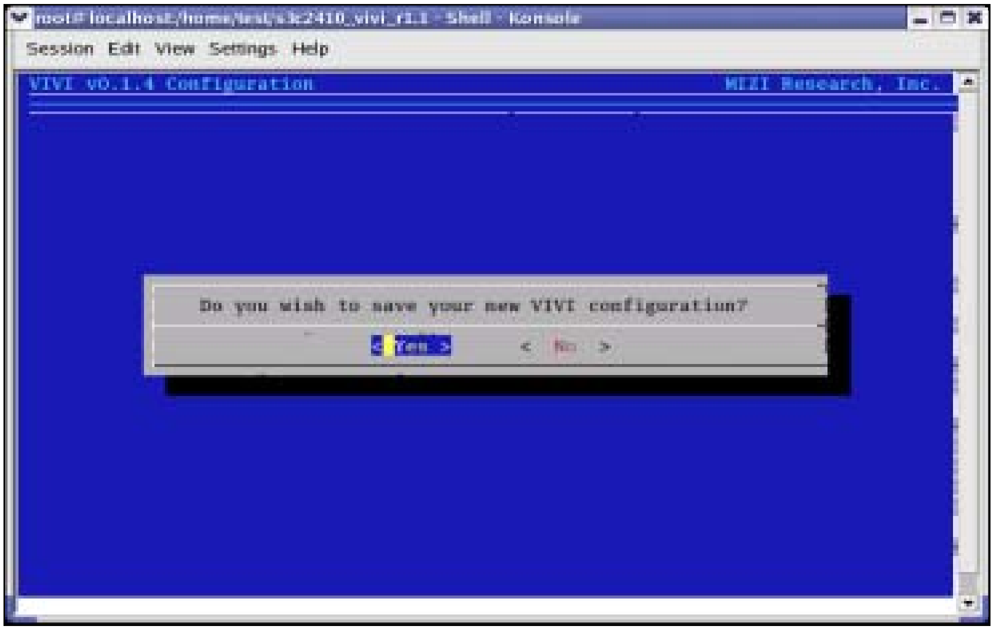


root@localhost:/home/test/s3c2410_vivi_r1.1 - Shell - Konsole
Session Edit View Settings Help

```
VIVI v0.1.4 Configuration                                   MIZI Research, Inc.

      Enter the name of the configuration file you wish
      to load.  Accept the name shown to restore the
      configuration you last retrieved.  Leave blank to
      abort.

      arch/def-configs/smdk2410

                  <  Ok  >        < Help >
```

Figure 3-2 Inputting Vivi configuration file

Select ***'Exit'*** and then ***'Yes'*** to save your new kernel configuration.


root@localhost:/home/test/s3c2410_vivi_r1.1 - Shell - Konsole
Session
Edit
View
Settings
Help
< Yes >
< No >

VIVI v0.1.4 Configuration MIZI Research, Inc.

Do you wish to save your new VIVI configuration?

Figure 3-3 Saving New Kernel Configuration

After saving the New Kernel configuration, please execute the following command to compile Vivi source code.

```
[root@localhost s3c2410_vivi_r1.1]# make
```

If the compilation of vivi progresses well, ***'vivi'*** binary image file will be created under ***/vivi*** directory.

In Next chapter we will learn how to port vivi (bootloader), kernel image, and root file system to target board. To do this work more conveniently, it is good to collect the compiled images to ***/image*** directory.
Create the '***/image'*** directory under '***/home/test'*** and copy the compiled images to '***/image'*** directory.

```
[root@localhost s3c2410_vivi_r1.1]# mkdir /home/test/image
[root@localhost s3c2410_vivi_r1.1]# cp vivi /home/test/image
```

Please execute the following command to create ***imagewrite*** utility, to write the images to the SMC.

```
[root@localhost s3c2410_vivi_r1.1]# cd util
[root@localhost util]# /usr/local/arm/2.95.3/bin/arm-linux-gcc –o imagewrite
imagewrite.c
```

Finally copy the ***'imagewrite'*** utility to ***'/home/test/image'.***

```
 [root@localhost util]# cp imagewrite /home/test/image
```

3.3 Compiling Kernel

Kernel source is compressed by the name of ***'s3c2410_kernel2.4.18_r1.1.tar.bz2'***. Extract this bz2 file by executing the following command. After extracting the kernel tarball file ***'s3c2410_kernel2.4.18_r1.1'*** directory will generate.

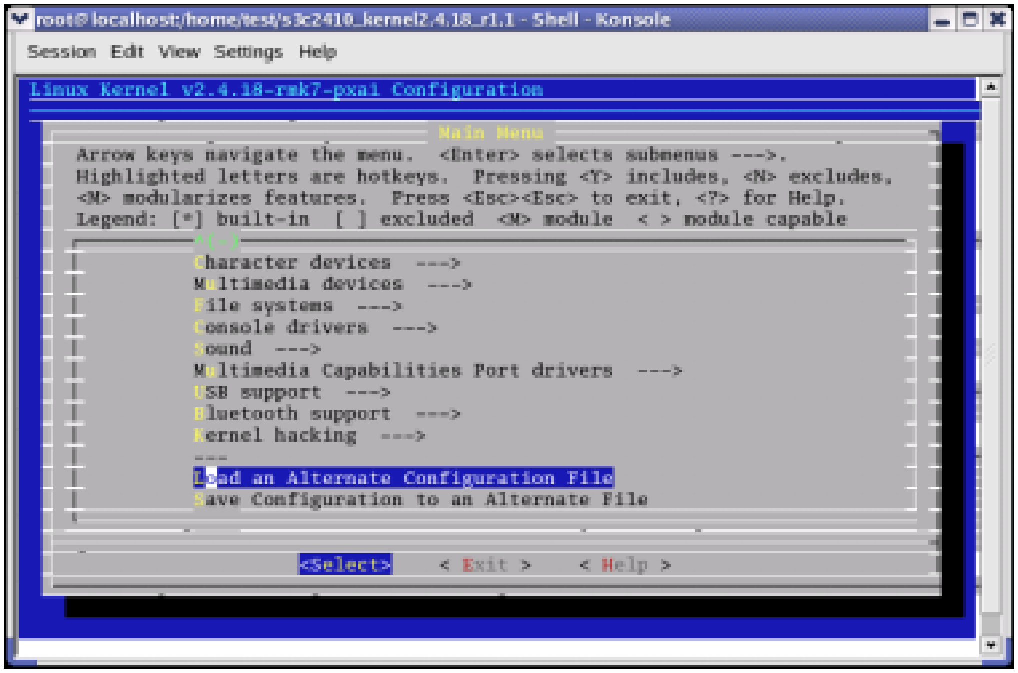
```
[root@localhost test]# tar jxvf s3c2410_kernel2.4.18_r1.1.tar.bz2
[root@localhost test]# cd s3c2410_kernel2.4.18_r1.1
```

Set the values by executing ***'make menuconfig'*** command. You can load default-configuration-file that is composed with values optimized to target board. In the case of kernel, default-configuration-files are located in ***'s3c2410_kernel2.4.18_r1.1'*** directory.

Please enter the path of the configuration file to load ***'arch/arm/def-configs/smdk24a0'*** file, after selecting ***'Load on Alternate Configuration File'*** menu.

```
[root@localhost s3c2410_kernel2.4.18_r1.1]# make menuconfig
```

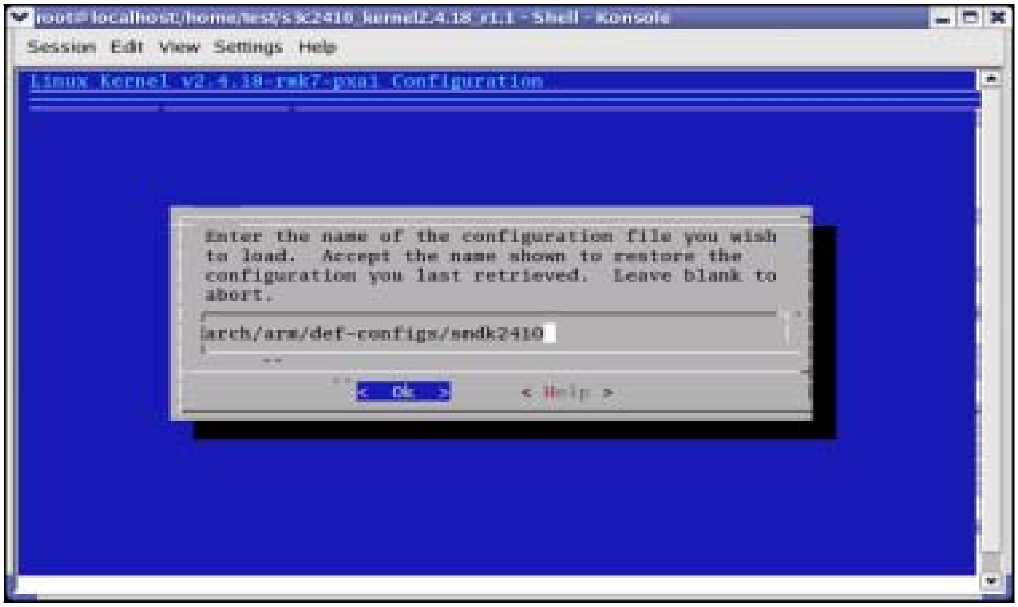
Please select ***'Load an Alternate Configuration File'*** as shown in figure 3-4.


root@localhost:/home/test/s3c2410_kernel2.4.18_r1.1 - Shell - Konsole
Session Edit View Settings Help

```
Linux Kernel v2.4.18-rmk7-pxa1 Configuration

                              Main Menu
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
            ^(-)
            Character devices  --->
            Multimedia devices  --->
            File systems  --->
            Console drivers  --->
            Sound  --->
            Multimedia Capabilities Port drivers  --->
            USB support  --->
            Bluetooth support  --->
            Kernel hacking  --->
            ---
            Load an Alternate Configuration File
            Save Configuration to an Alternate File

                <Select>    < Exit >    < Help >
```

Figure 3-4 Kernel configurations

Please enter the name of the configuration file you wish to load ***'arch/arm/def-configs/smdk2410'*** as shown in figure 3-5.

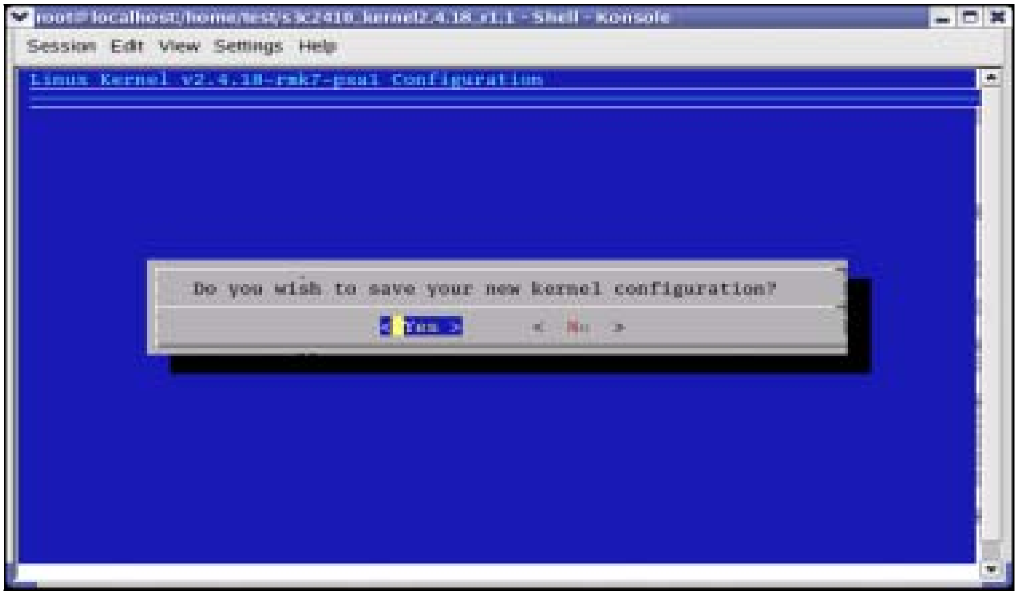


root@localhost:/home/test/s3c2410_kernel2.4.18_r1.1 - Shell - Konsole
Session Edit View Settings Help

```
Linux Kernel v2.4.18-rmk7-pxa1 Configuration

  Enter the name of the configuration file you wish
  to load.  Accept the name shown to restore the
  configuration you last retrieved.  Leave blank to
  abort.

 arch/arm/def-configs/smdk2410

              <  Ok  >          < Help >
```

Figure 3-5 Inputting Kernel configuration file

Select ***'Exit'*** and then ***'Yes'*** to save your new kernel configuration.



root@localhost:/home/test/s3c2410_kernel2.4.18_r1.1 - Shell - Konsole
Session Edit View Settings Help

```
Linux Kernel v2.4.18-rmk7-pxa1 Configuration

  Do you wish to save your new kernel configuration?

              < Yes >      <  No  >
```

Figure 3-6 Saving New Kernel Configuration

```
[root@localhost s3c2410_kernel2.4.18_r1.1]# make dep
[root@localhost s3c2410_kernel2.4.18_r1.1]# make zImage
```

After executing above commands the Kernel image will be created in '***s3c2410_kernel2.4.18_r1.1/arch/arm/boot'*** directory by the name of '***zImage'.***

Copy ***'zImage'***(kernel image) to ***'image'*** directory, to port the kernel image on the target board.

```
# s3c2410_kernel2.4.18_r1.1/arch/arm/boot/
[root@localhost boot]# cp zImage /home/test/image/
```

3.4 Copying Root file System

Root filesystem is composed of *Cramfs (Compressed ROM file system).* Cramfs is designed small and simple. The size is restricted to 256MB, but it doesn't act on a defect in embedded system.

To port the Root File System onto the target board easily, copy the root file system to **'/image/'** directory.

```
[root@localhost test]# cp root_qtopia.cramfs /home/test/image
[root@localhost test]# cp root.cramfs /home/test/image
```

All the images (vivi, zImage, root.cramfs, root_qtopia.cramfs) are collected under '***/image'*** directory. In next chapter, we will learn about how to port these images to the target board.

4 Porting Linux to the Target Board

4.1 Porting Linux to the Target Board

Now in this chapter we will learn how to write *vivi (bootloader), zImage (kernel image),* and *root_qtopia.cramfs* to SMC(Smart Media Card) by using **'imagewrite'** utility. This method can be used after booting target board. So it is used for writing images to new SMC.

Transfer the images and the needed utilities to target board because all works are progressed in target board. Transfer all the images from **/image** directory to target board by using ***ztelnet***.

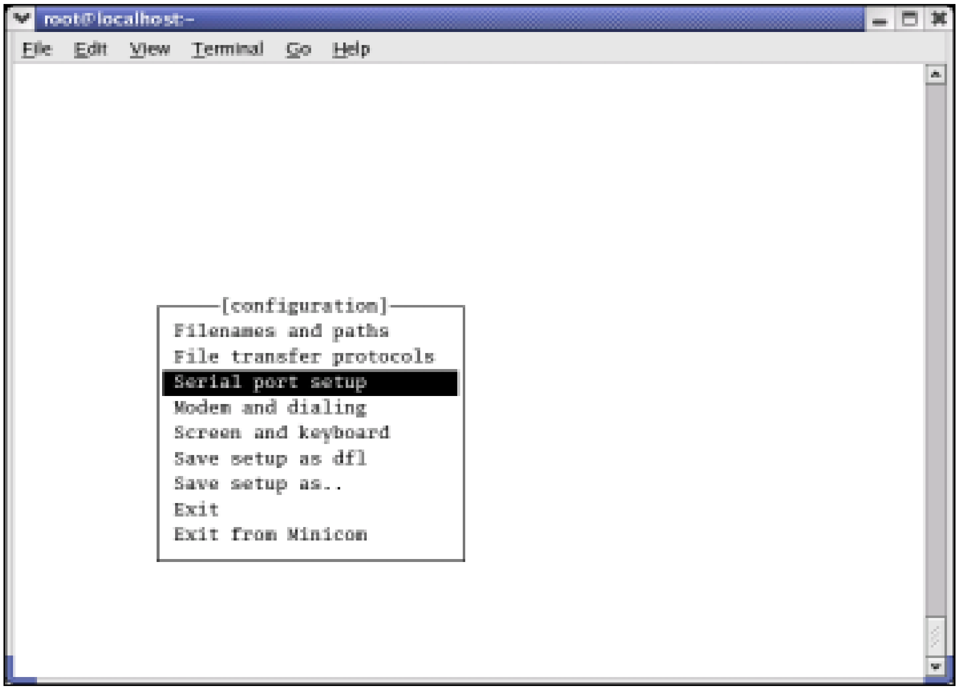
4.2 Minicom

We have to transfer the images using **ztelnet**, before that we should know how to use **Minicom**. In this section we explain how to use **Minicom**. **Ztelnet** is explained later in this chapter.

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.

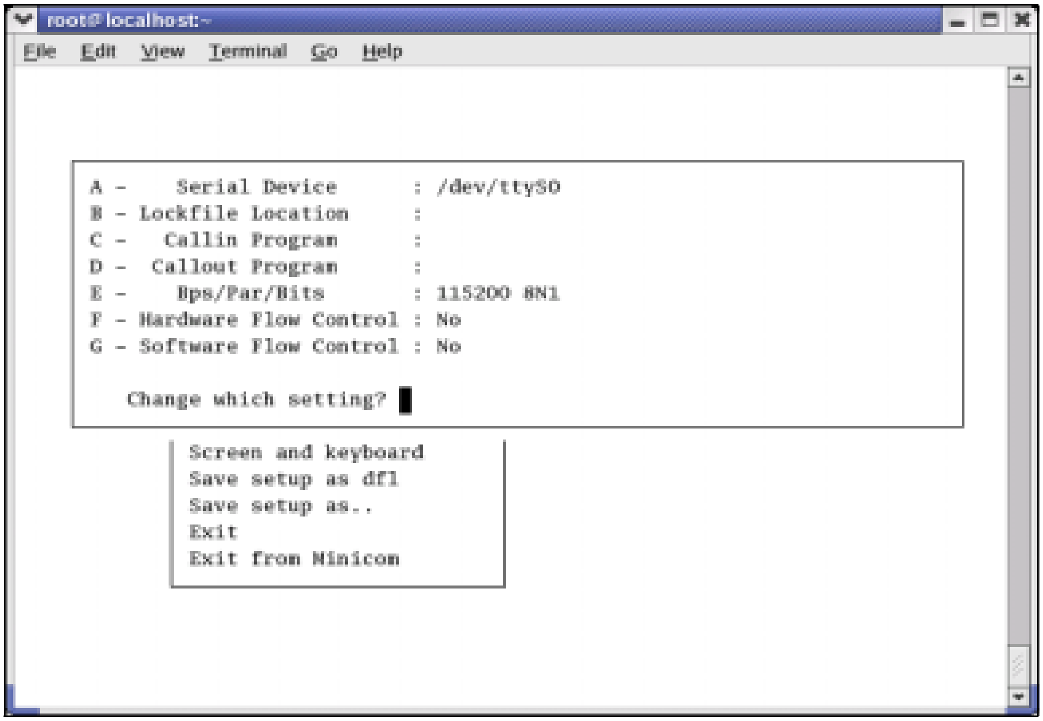
```
[root@localhost root]# minicom -s : Execute Minicom on setting mode.
```



```
 [configuration]
 Filenames and paths
 File transfer protocols
 Serial port setup
 Modem and dialing
 Screen and keyboard
 Save setup as dfl
 Save setup as..
 Exit
 Exit from Minicom
```

Figure 4-1 Minicom setup

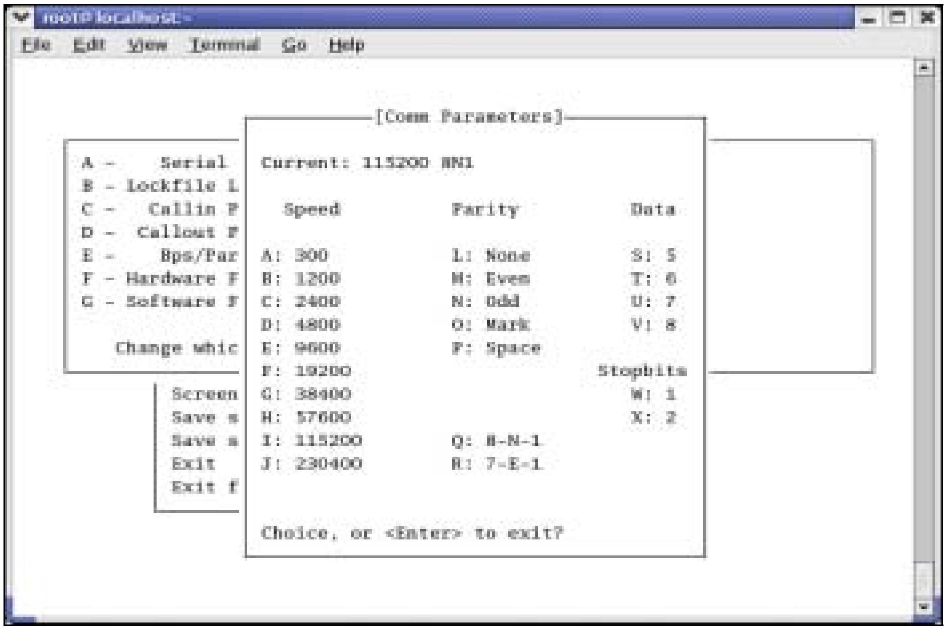
Please select ***'Serial port setup'*** Push ***'A'*** key for setting ***'Serial Device'***, then write serial port which is connected to target board. (If you are using ***COM1***, write ***/dev/ttyS0***, if ***COM2***, write ***/dev/ttyS1***.)



```
A -    Serial Device      : /dev/ttyS0
B - Lockfile Location     :
C -   Callin Program      :
D -  Callout Program      :
E -    Bps/Par/Bits       : 115200 8N1
F - Hardware Flow Control : No
G - Software Flow Control : No

   Change which setting?

     Screen and keyboard
     Save setup as dfl
     Save setup as..
     Exit
     Exit from Minicom
```

Figure 4-2 Serial Port setup

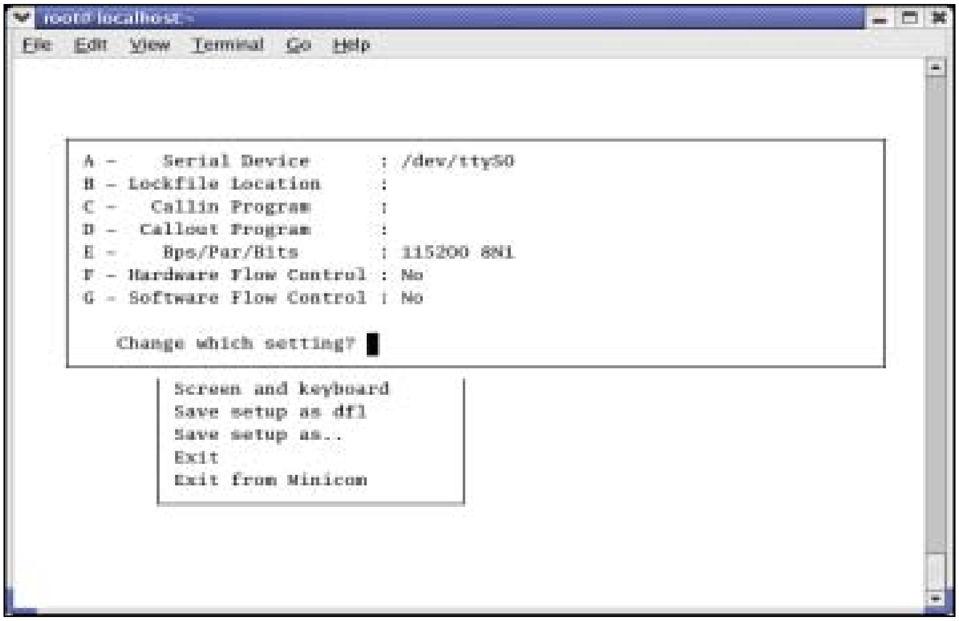
Push **'E'** key for setting up **'bps/Par/Bits'**. Push **'I'** to set up ***'bps'*** to **115200**, Push **'V'** to set up **'Data bits'** to **8**, Push **'W'** to set up **'Stop bits'** to **'1'**, and **'V'** to set up **'parity'** to '**NONE'**.



```
                      [Comm Parameters]

 A -    Serial   Current: 115200 8N1
 B - Lockfile L
 C -   Callin P    Speed          Parity          Data
 D -  Callout P
 E -    Bps/Par  A: 300          L: None         S: 5
 F - Hardware F  B: 1200         M: Even         T: 6
 G - Software F  C: 2400         N: Odd          U: 7
                 D: 4800         O: Mark         V: 8
    Change whic  E: 9600         P: Space
                 F: 19200                      Stopbits
         Screen  G: 38400                        W: 1
         Save s  H: 57600                        X: 2
         Save s  I: 115200       Q: 8-N-1
         Exit    J: 230400       R: 7-E-1
         Exit f

                 Choice, or <Enter> to exit?
```

Figure 4-3 Serial Port setup

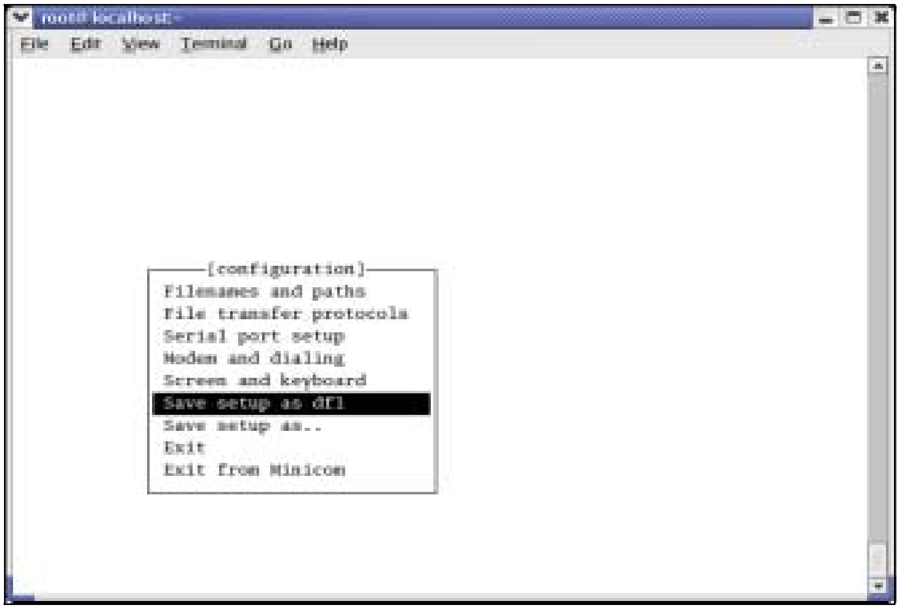
Push **'F'** key for setting up **'Hardware Flow Control'** to **'NO'**.
Push **'G'** key for setting up **'Software Flow Control'** to **'NO'**. The default value is **'NO'**.



```
A -    Serial Device      : /dev/ttyS0
B - Lockfile Location     :
C -   Callin Program      :
D -  Callout Program      :
E -    Bps/Par/Bits       : 115200 8N1
F - Hardware Flow Control : No
G - Software Flow Control : No

   Change which setting?

        Screen and keyboard
        Save setup as dfl
        Save setup as..
        Exit
        Exit from Minicom
```

Figure 4-4 Hardware/Software Flow Control Setup

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

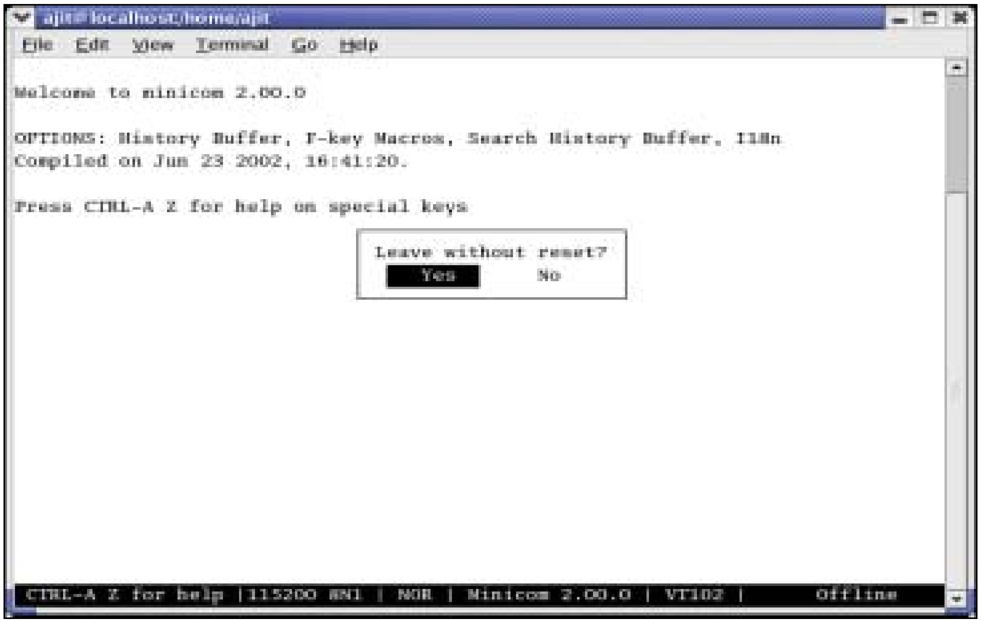


```
-----[configuration]------
 Filenames and paths
 File transfer protocols
 Serial port setup
 Modem and dialing
 Screen and keyboard
 Save setup as dfl
 Save setup as..
 Exit
 Exit from Minicom
```

Figure 4-5 saving Minicom Setup

Push **'Exit'** key, to exit from the setting mode. Currently, the set points are stored to the file **'/etc/minirc.dfl'.**

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



```
Welcome to minicom 2.00.0

OPTIONS: History Buffer, F-key Macros, Search History Buffer, I18n
Compiled on Jun 23 2002, 16:41:20.

Press CTRL-A Z for help on special keys

                    Leave without reset?
                     Yes          No

 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.00.0 | VT102 |      Offline
```

Figure 4-6 Exiting from Minicom

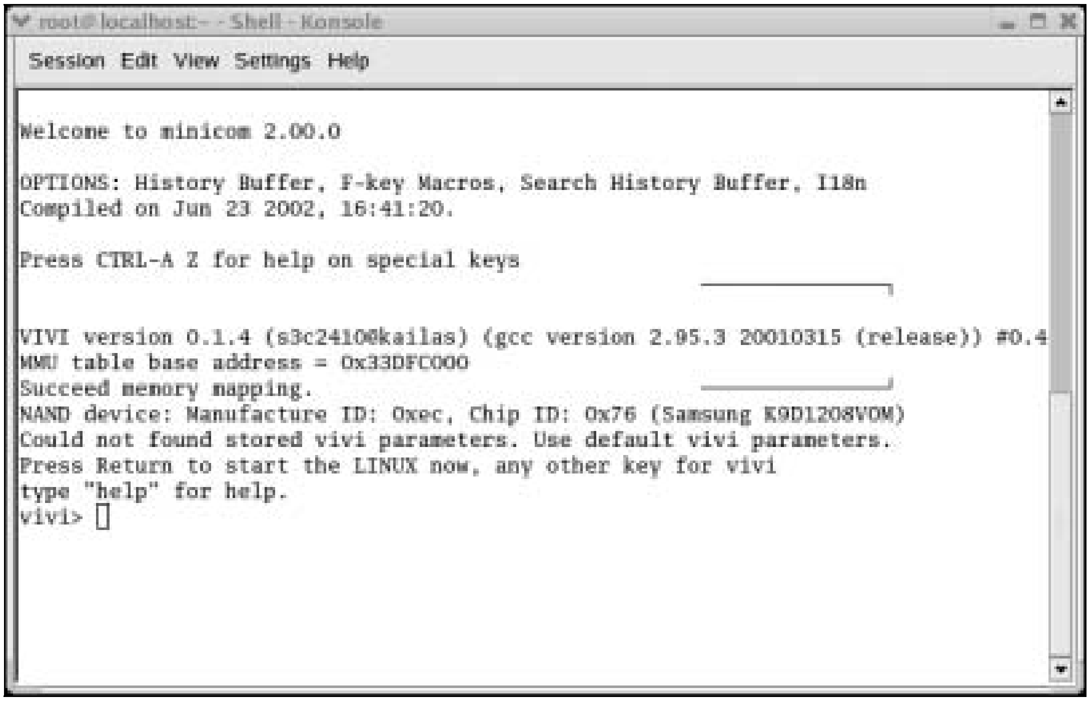
4.3 Uploading ‘vivi’ using JTAG Cable

JTAG cable and Jflash program are required to port the images to the target board. Here we use **Windows** Jflash program, which is compressed with tarball ***‘sjf2410_v4’***. You can download this program from following URL.

http://www.samsung.com/Products/Semiconductor/SystemLSI/MobileSolutions/MobileASSP/MobileComputing/S3C2410X/S3C2410X.htm

Unzip the ***‘sjf2410_v4’*** on Windows PC. This file includes ***‘sjf2410_v4.pdf’*** and source code for Jflash program. Please refer to ***‘sjf2410_v4.pdf’***. With the help of ***‘sjf2410_v4.pdf’*** you can download the **vivi** (bootloader) to your K9S1208 NAND Flash on **Wndows PC** so that you can boot the target board to the vivi prompt, to write the kernel and Qtopia images.

After you complete downloading **vivi** bootloader to the SMC. Please insert the SMC on to the target board. Connect the target board by serial cable and run the **Minicom.** Supply power to target board, in that case target board is waiting inputs during the times defined by developer. If we do not input anything or press ‘**Enter’**, target board begins to boot. Instead, if you input **‘Any’** key, target board enters into **vivi** prompt mode. The delay time is very short, so first keep ‘**Any’** key pressed and switch on the target board, if you want to use target board console.


root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

```
Welcome to minicom 2.00.0

OPTIONS: History Buffer, F-key Macros, Search History Buffer, I18n
Compiled on Jun 23 2002, 16:41:20.

Press CTRL-A Z for help on special keys


VIVI version 0.1.4 (s3c2410@kailas) (gcc version 2.95.3 20010315 (release)) #0.4
MMU table base address = 0x33DFC000
Succeed memory mapping.
NAND device: Manufacture ID: 0xec, Chip ID: 0x76 (Samsung K9D1208V0M)
Could not found stored vivi parameters. Use default vivi parameters.
Press Return to start the LINUX now, any other key for vivi
type "help" for help.
vivi> 
```

Figure 4-7 vivi shell prompt

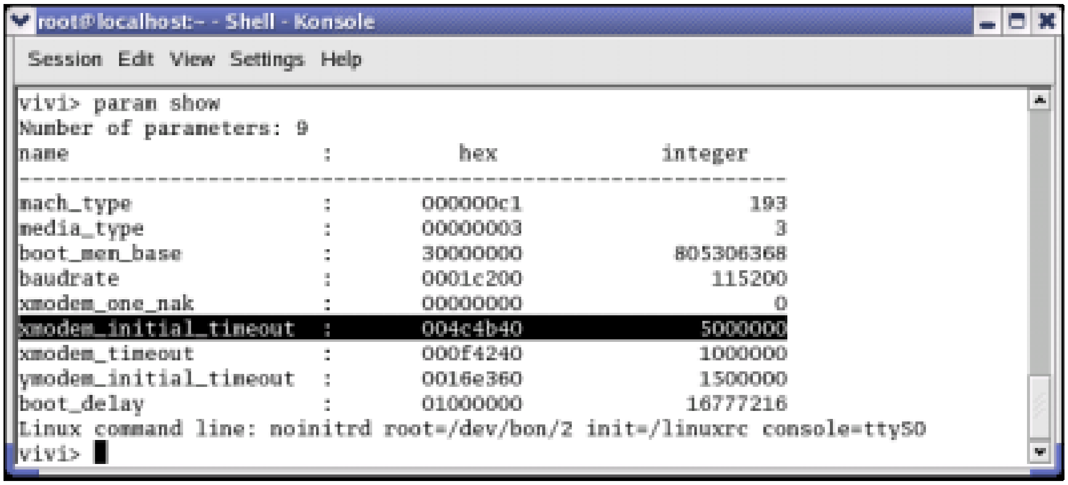
Note: If you can not see the **vivi>** prompt, that means the vivi bootloader has not been downloaded properly. Please try to download vivi bootloader one more time.

4.4 Porting Images using vivi

Once vivi (bootloader) is stored in SMC (NAND flash memory), you can write vivi (bootloader), kernel image, qtopia image etc. to SMC on prompt mode of vivi (bootloader) by **xmodem** of **Minicom**. It can be possible only when bootloader exists in flash memory.

If **'transfer incomplete'** message is appeared while writing images, the reason is that the timeout of **xmodem_initial** is too short. In this case, you can solve the problem by increasing the timeout of **xmodem_initial.** First check the value of **'xmodem_initial_timeout'** parameter. If it is too short, extend timeout properly.

vivi> param show
vivi> param set xmodem_initial_timeout 5000000 : "5000000" means 5 second because unit is in microsecond.
vivi> param save

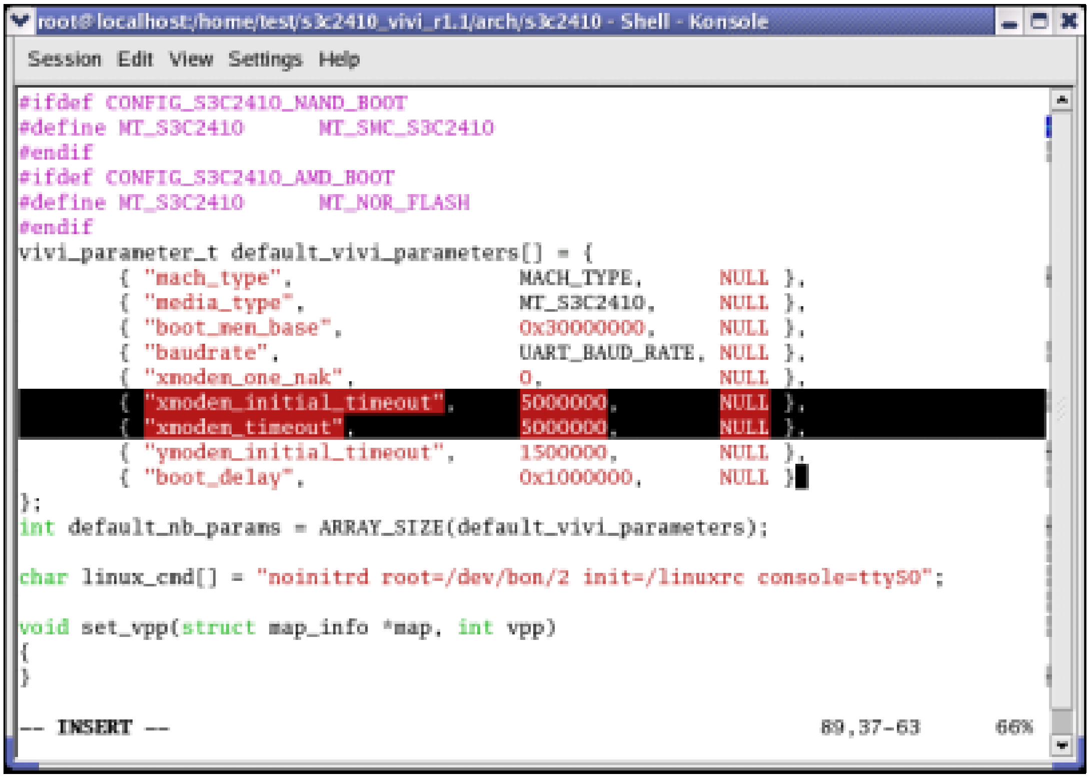

root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

```
vivi> param show
Number of parameters: 9
name                    :         hex             integer
-------------------------------------------------------------
mach_type               :       000000c1              193
media_type              :       00000003                3
boot_mem_base           :       30000000        805306368
baudrate                :       0001c200           115200
xmodem_one_nak          :       00000000                0
xmodem_initial_timeout  :       004c4b40          5000000
xmodem_timeout          :       000f4240          1000000
ymodem_initial_timeout  :       0016e360          1500000
boot_delay              :       01000000         16777216
Linux command line: noinitrd root=/dev/bon/2 init=/linuxrc console=ttyS0
vivi> 
```

Figure 4-8 xmodem_initial_timeout settings on vivi prompt

Once you set the **'xmodem_initial_timeout'** you can write the images.

If it is not possible to change the **‘xmodem_initial_timeout’**, you can edit the **/vivi/arch/s3c2410/smdk.c** file as shown in fig 4-9.


root@localhost:/home/test/s3c2410_vivi_r1.1/arch/s3c2410 - Shell - Konsole
Session Edit View Settings Help

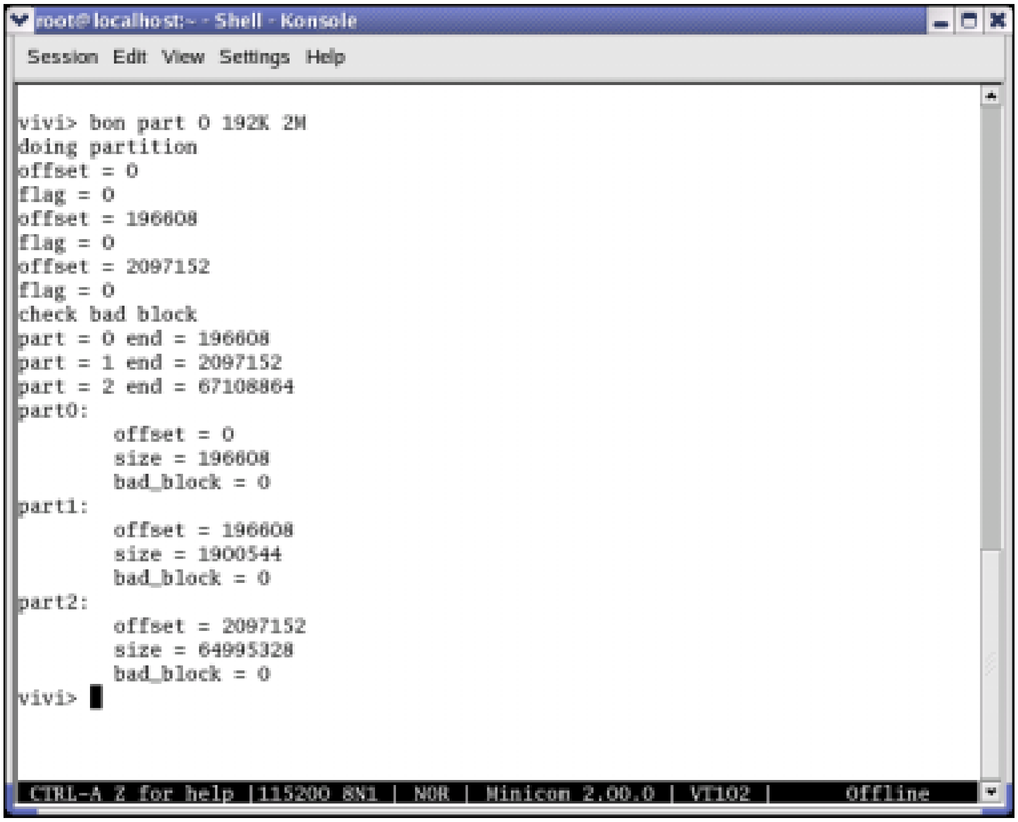
```
#ifdef CONFIG_S3C2410_NAND_BOOT
#define MT_S3C2410      MT_SMC_S3C2410
#endif
#ifdef CONFIG_S3C2410_AMD_BOOT
#define MT_S3C2410      MT_NOR_FLASH
#endif
vivi_parameter_t default_vivi_parameters[] = {
        { "mach_type",                  MACH_TYPE,      NULL },
        { "media_type",                 MT_S3C2410,     NULL },
        { "boot_mem_base",              0x30000000,     NULL },
        { "baudrate",                   UART_BAUD_RATE, NULL },
        { "xmodem_one_nak",             0,              NULL },
        { "xmodem_initial_timeout",     5000000,        NULL },
        { "xmodem_timeout",             5000000,        NULL },
        { "ymodem_initial_timeout",     1500000,        NULL },
        { "boot_delay",                 0x1000000,      NULL }
};
int default_nb_params = ARRAY_SIZE(default_vivi_parameters);

char linux_cmd[] = "noinitrd root=/dev/bon/2 init=/linuxrc console=ttyS0";

void set_vpp(struct map_info *map, int vpp)
{
}

-- INSERT --                                              89,37-63      66%
```

Figure 4-9 xmodem_initial_timeout settings

4.5 SMC partitioning and writing vivi image

Now you can write all images including vivi again through **vivi prompt**. But before writing the images, you have to partition the SMC to assign the memory for each image. SMC is composed of bon file system and vivi supports this. So you can make partitions through vivi prompt with the help of following command.

```
vivi> bon part 0 192k 2M
```


root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

```
vivi> bon part 0 192K 2M
doing partition
offset = 0
flag = 0
offset = 196608
flag = 0
offset = 2097152
flag = 0
check bad block
part = 0 end = 196608
part = 1 end = 2097152
part = 2 end = 67108864
part0:
        offset = 0
        size = 196608
        bad_block = 0
part1:
        offset = 196608
        size = 1900544
        bad_block = 0
part2:
        offset = 2097152
        size = 64995328
        bad_block = 0
vivi> █

 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.00.0 | VT102 |       Offline
```

Figure 4-10 Partitioning SMC

The **bon** command makes 3 partitions of sizes **0~192K**, **192K~2M**, and **2M~End-part**.

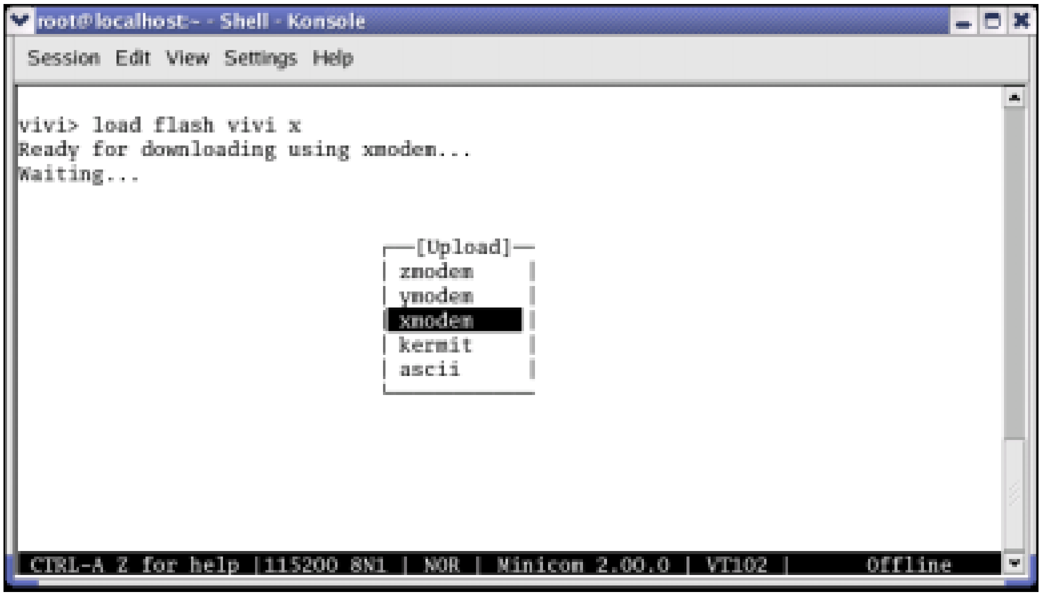
0~192k	: **vivi** will be written here.
192k~2M	: **zImage** (kernel) will be written here.
2M~End-part	: **root.cramfs** (root filesystem) will be written here.

Above command does formatting of SMC as well as partitioning it. So if you do next steps like writing *kernel* and *root filesystem*, you have to write **vivi** again. Write **vivi** by following command.

```
vivi> load flash vivi x
```

To download the **vivi** bootloader press '**ctrl +A' -> 'z'** and then **'S'** to send file.

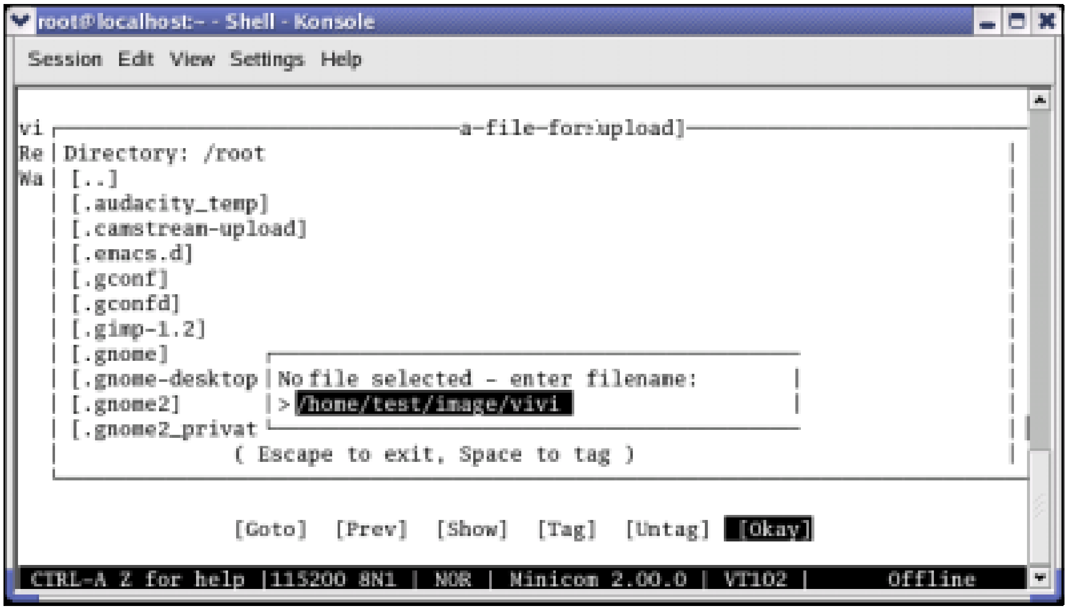
Window questioning about transfer mode will appear. Please select **xmodem** and hit **Enter** Key.


root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

```
vivi> load flash vivi x
Ready for downloading using xmodem...
Waiting...



                              ┌─[Upload]─
                              │ zmodem    │
                              │ ymodem    │
                              │ xmodem    │
                              │ kermit    │
                              │ ascii     │
                              └──────────

 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.00.0 | VT102 |      Offline
```

Figure 4-11 xmodem x-fer mode for Vivi

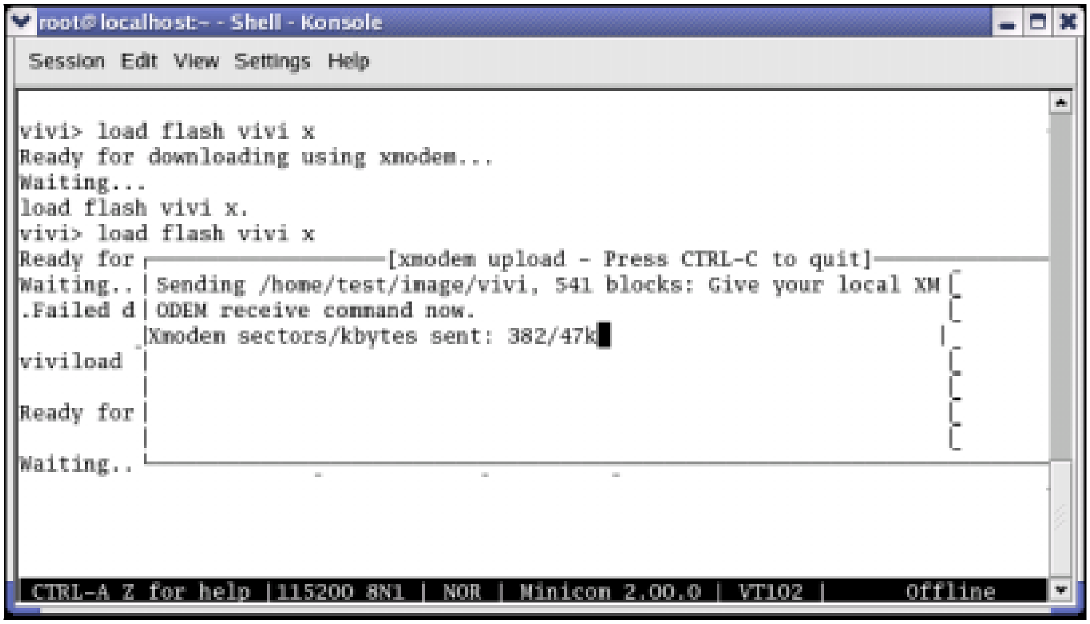
Please give the path of the vivi bootloader file as shown in the following figure.


root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

```
vi ┌────────────────────────────a-file-for:upload]──────────────────────────
Re │ Directory: /root                                                      │
Wa │ [..]                                                                  │
   │ [.audacity_temp]                                                      │
   │ [.camstream-upload]                                                   │
   │ [.emacs.d]                                                            │
   │ [.gconf]                                                              │
   │ [.gconfd]                                                             │
   │ [.gimp-1.2]                                                           │
   │ [.gnome]         ┌──────────────────────────────────────────┐         │
   │ [.gnome-desktop │No file selected - enter filename:        │         │
   │ [.gnome2]       │> /home/test/image/vivi                   │         │
   │ [.gnome2_privat └──────────────────────────────────────────┘         │
   │              ( Escape to exit, Space to tag )                         │
   └───────────────────────────────────────────────────────────────────────

              [Goto]  [Prev]  [Show]  [Tag]  [Untag]  [Okay]

CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.00.0 | VT102 |      Offline
```

Figure 4-12 Entering filename for vivi

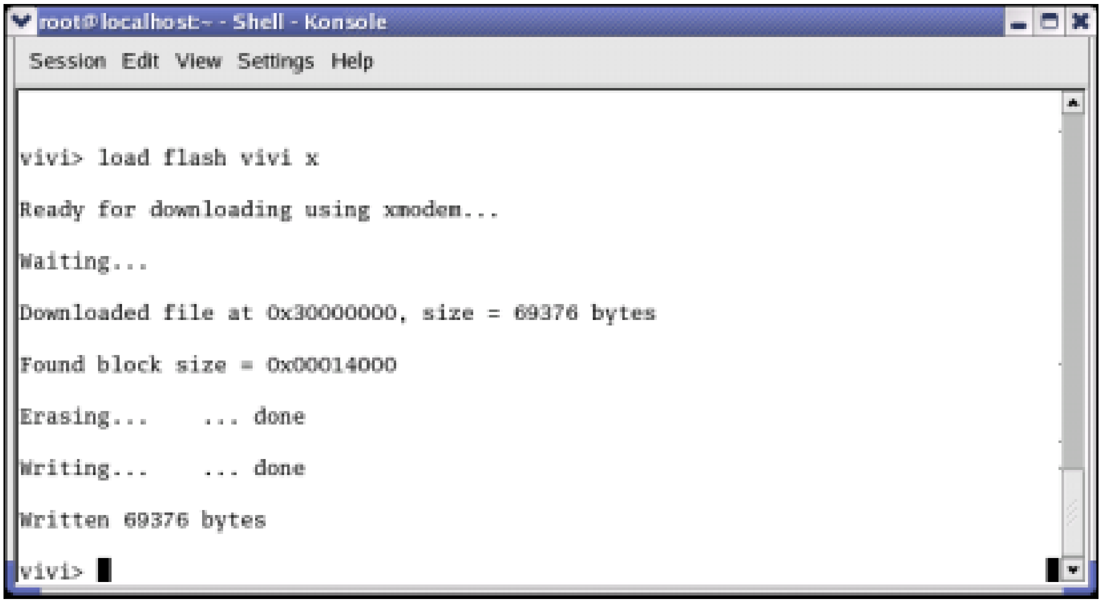
You can see the sending status of **vivi** bootloader as shown in the following figure.


root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

```
vivi> load flash vivi x
Ready for downloading using xmodem...
Waiting...
load flash vivi x.
vivi> load flash vivi x
Ready for ┌────────────────[xmodem upload - Press CTRL-C to quit]──────────
Waiting.. │ Sending /home/test/image/vivi, 541 blocks: Give your local XM
.Failed d │ ODEM receive command now.
          │Xmodem sectors/kbytes sent: 382/47k
viviload  │
          │
Ready for │
          │
Waiting.. └────────────────────────────────────────────────────────────────

CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.00.0 | VT102 |      Offline
```

Figure 4-13 vivi download status

After sending **vivi** bootloader image completes hit **Enter** key to come to **vivi** prompt.


root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

```
vivi> load flash vivi x

Ready for downloading using xmodem...

Waiting...

Downloaded file at 0x30000000, size = 69376 bytes

Found block size = 0x00014000

Erasing...    ... done

Writing...    ... done

Written 69376 bytes

vivi>
```

Figure 4-14 vivi Prompt

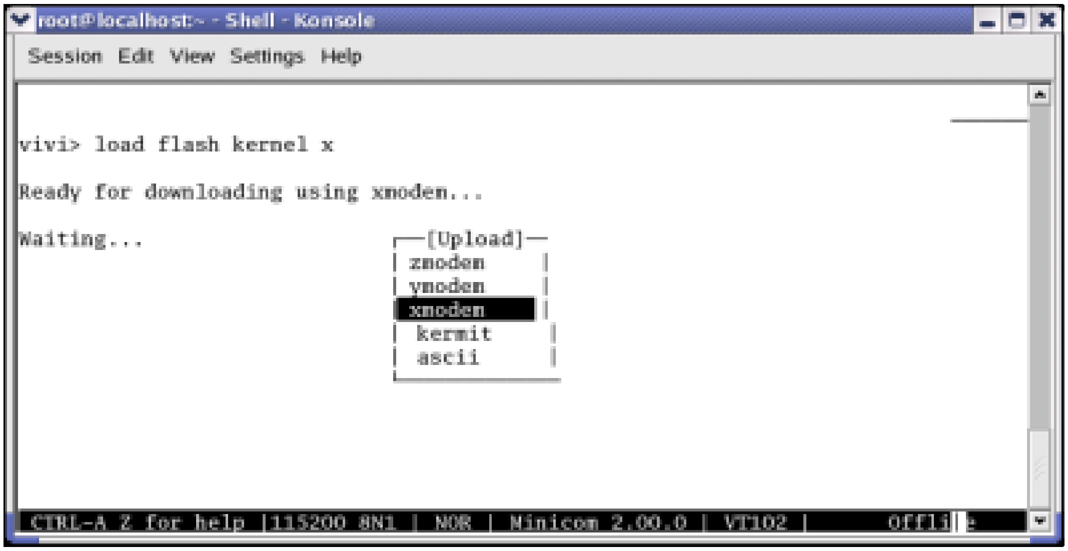
4.6 Writing Kernel Image

To upload kernel Image please execute the following command.

```
vivi> load flash kernel x
```

To download the **vivi** bootloader press '**ctrl +A' -> 'z'** and then **'S'** to send file.

Window questioning about transfer mode will appear. Please select **xmodem** and hit **Enter** Key.

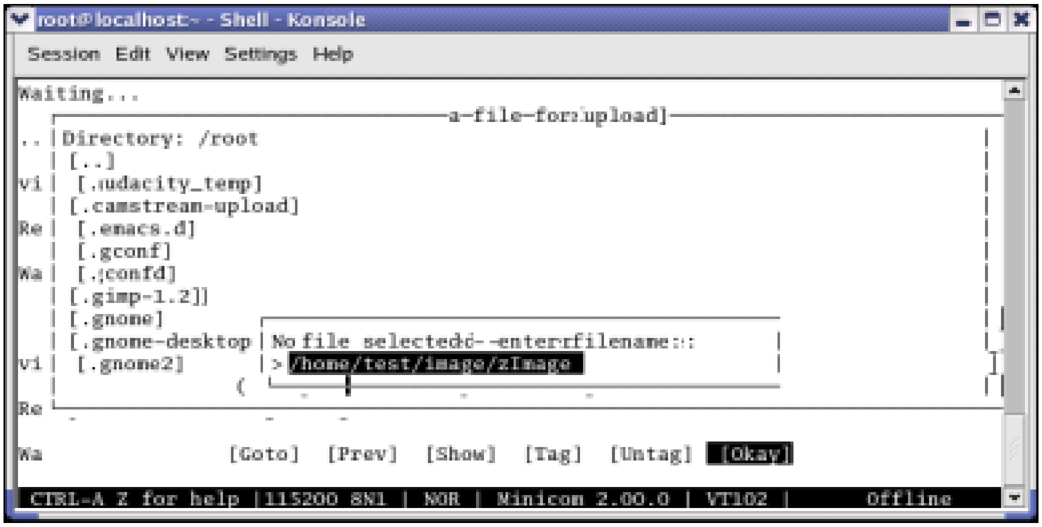


root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

```
vivi> load flash kernel x

Ready for downloading using xmodem...

Waiting...                   ┌─[Upload]─
                             | zmodem    |
                             | ymodem    |
                             | xmodem    |
                             |  kermit    |
                             |  ascii     |
                             └──────────

 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.00.0 | VT102 |       Offline
```

Figure 4-15 xmodem x-fer mode for kernel Image

Please give the path of the kernel image (zImage) file as shown in the following figure.

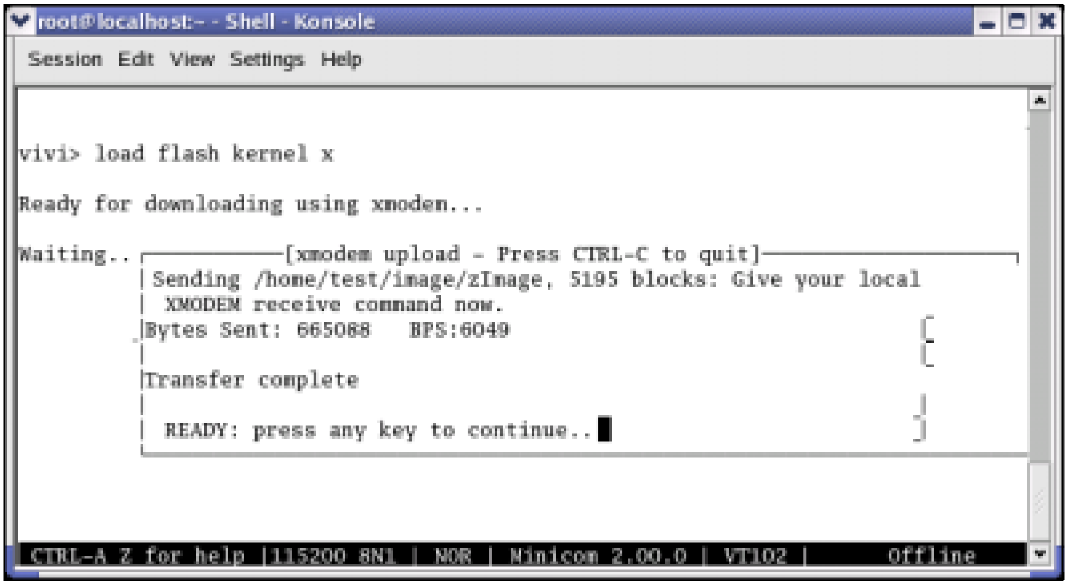


root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

```
Waiting...
   ┌────────────────────────────────a-file-for:upload]──────────────
.. |Directory: /root
   | [..]
vi |  [.audacity_temp]
   | [.camstream-upload]
Re |  [.emacs.d]
   |  [.gconf]
Wa |  [.gconfd]
   | [.gimp-1.2]]
   | [.gnome]
   | [.gnome-desktop | No file selected - enter filename:
vi |  [.gnome2]      | > /home/test/image/zImage
   |
Re └───────────────────────────────────────────────────────────────

Wa                    [Goto]  [Prev]  [Show]  [Tag]  [Untag]  [Okay]

 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.00.0 | VT102 |       Offline
```

Figure 4-16 Entering filename for zImage

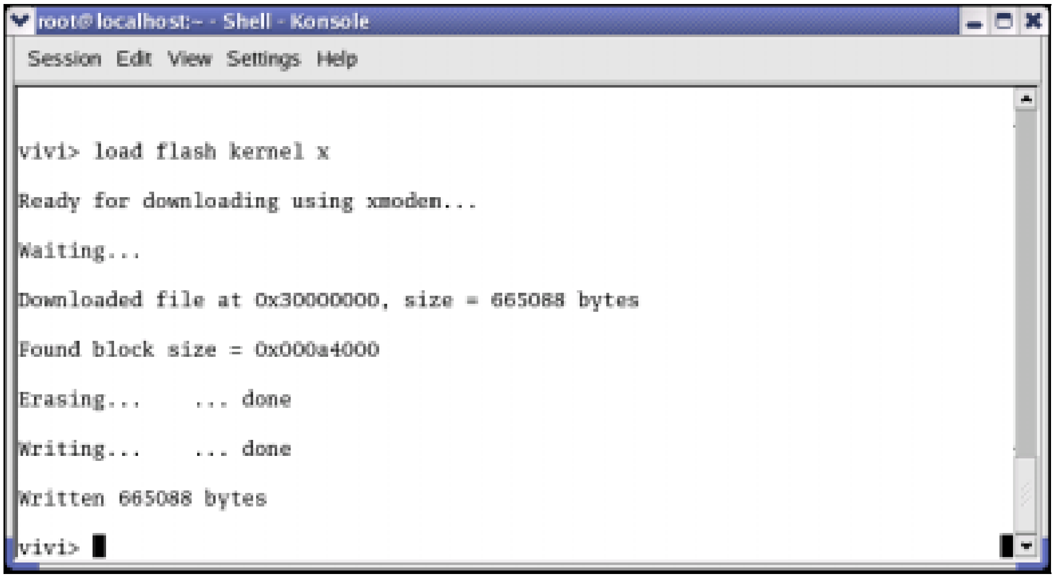
You can see the sending status of **zImage** as shown in the following figure.


root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

```
vivi> load flash kernel x

Ready for downloading using xmodem...

Waiting.. ┌──────────[xmodem upload - Press CTRL-C to quit]──────────────────┐
          │ Sending /home/test/image/zImage, 5195 blocks: Give your local
          │  XMODEM receive command now.
          │Bytes Sent: 665088   BPS:6049
          │
          │Transfer complete
          │
          │  READY: press any key to continue..
          └──────────────────────────────────────────────────────────────────
 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.00.0 | VT102 |      Offline
```

Figure 4-17 zImage download status

After sending **zImage** completes, hit **Enter** key to come to **vivi** prompt.


root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

```
vivi> load flash kernel x

Ready for downloading using xmodem...

Waiting...

Downloaded file at 0x30000000, size = 665088 bytes

Found block size = 0x000a4000

Erasing...     ... done

Writing...     ... done

Written 665088 bytes

vivi>
```

Figure 4-18 vivi Prompt

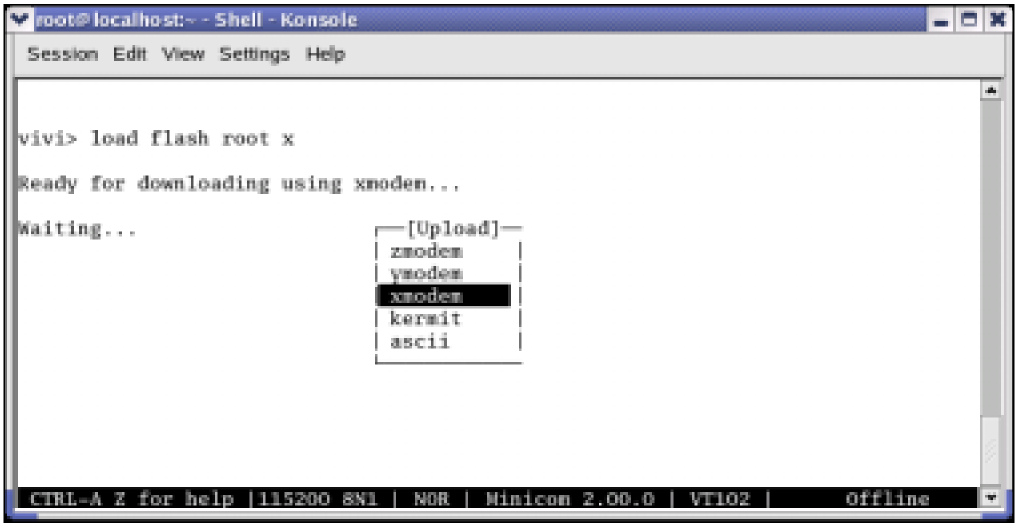
4.7 Writing Root File System Image

To upload *Root File System (root.cramfs)* Image please execute the following command.

```
vivi> load flash root x
```

To download the **root.cramfs** press '**ctrl +A' -> 'z'** and then **'S'** to send file.

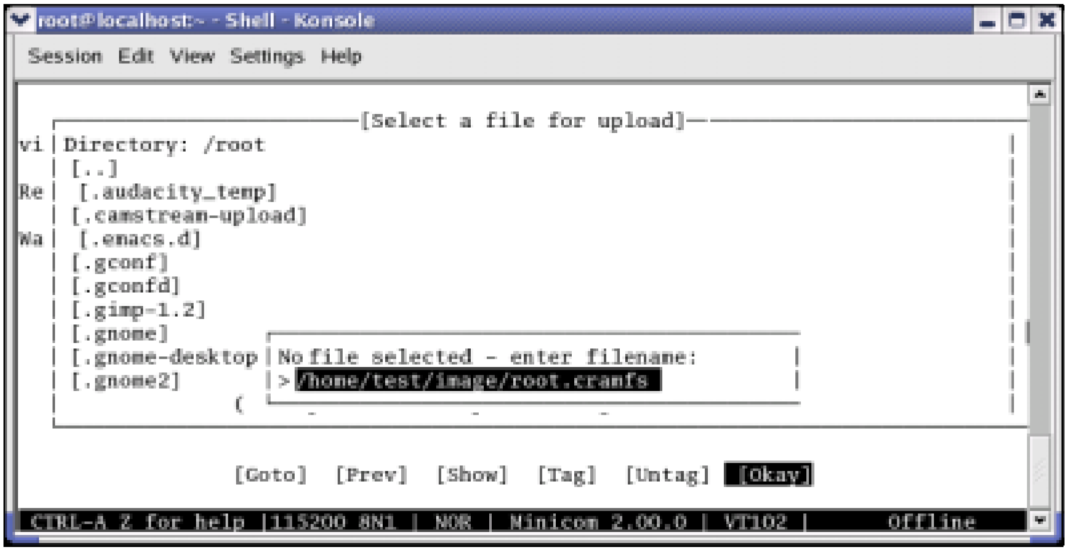
Window questioning about transfer mode will appear. Please select **xmodem** and hit **Enter** Key.


root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

```
vivi> load flash root x

Ready for downloading using xmodem...

Waiting...                  ┌─[Upload]─┐
                            │ zmodem   │
                            │ ymodem   │
                            │ xmodem   │
                            │ kermit   │
                            │ ascii    │
                            └──────────┘

CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.00.0 | VT102 |      Offline
```

Figure 4-19 xmodem x-fer mode for root.cramfs Image

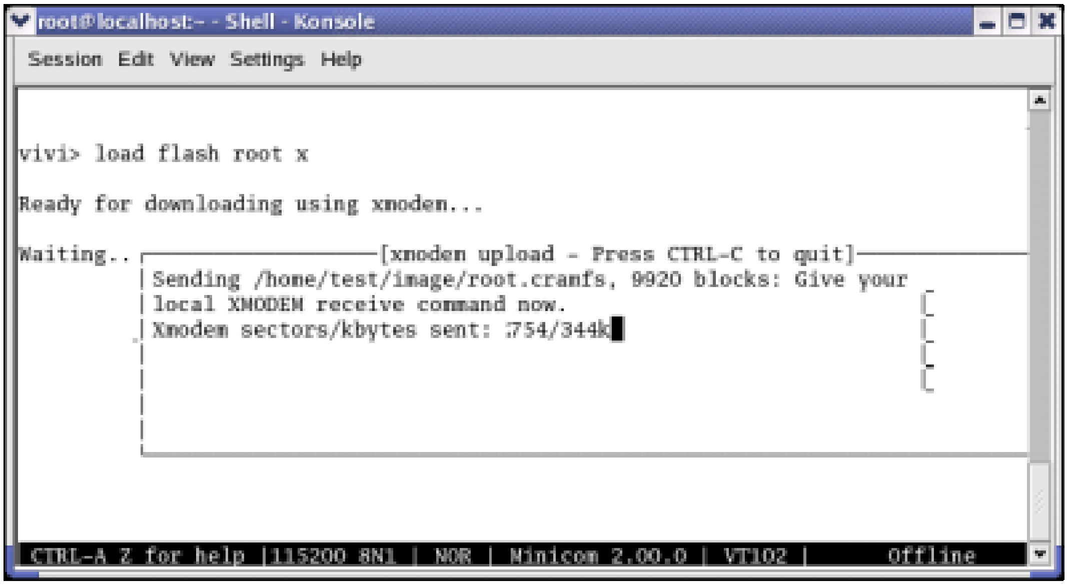
Please give the path of the **root.cramfs** file as shown in the following figure.


root@localhost:~ - Shell - Konsole
Session Edit View Settings Help
[Goto] [Prev] [Show] [Tag] [Untag] [Okay]

```
   ┌──────────────────────[Select a file for upload]──────────────────────
vi │Directory: /root
   │ [..]
Re │  [.audacity_temp]
   │ [.camstream-upload]
Wa │  [.emacs.d]
   │ [.gconf]
   │ [.gconfd]
   │ [.gimp-1.2]
   │ [.gnome]          ┌──────────────────────────────────┐
   │ [.gnome-desktop   │No file selected - enter filename:│
   │ [.gnome2]         │> /home/test/image/root.cramfs    │
   │              (    └──────────────────────────────────┘
   └──────────────────────────────────────────────────────────────────────

CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.00.0 | VT102 |      Offline
```

Figure 4-20 Entering filename for root.cramfs

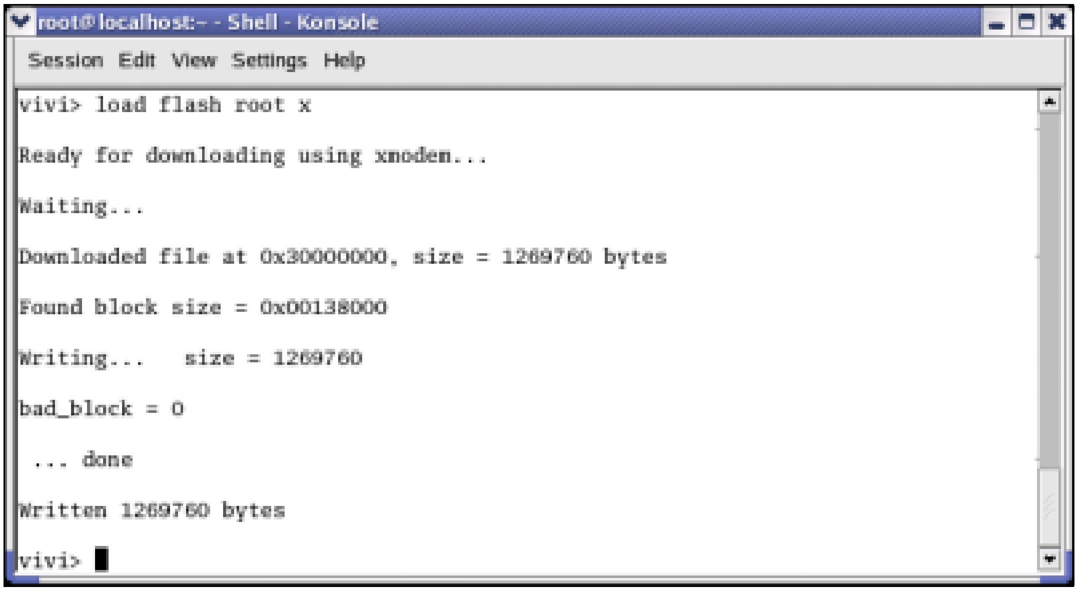
You can see the sending status of **root.cramfs** image as shown in the following figure.


root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

```
vivi> load flash root x

Ready for downloading using xmodem...

Waiting.. ┌──────────────────[xmodem upload - Press CTRL-C to quit]────────────
          | Sending /home/test/image/root.cramfs, 9920 blocks: Give your
          | local XMODEM receive command now.
          | Xmodem sectors/kbytes sent: :754/344k
          |
          |
          |
          └───────────────────────────────────────────────────────────────────

 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.00.0 | VT102 |       Offline
```

Figure 4-21 root.cramfs download status

After sending **root.cramfs** completes, hit **Enter** key to come to **vivi** prompt.


root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

```
vivi> load flash root x

Ready for downloading using xmodem...

Waiting...

Downloaded file at 0x30000000, size = 1269760 bytes

Found block size = 0x00138000

Writing...   size = 1269760

bad_block = 0

 ... done

Written 1269760 bytes

vivi> █
```

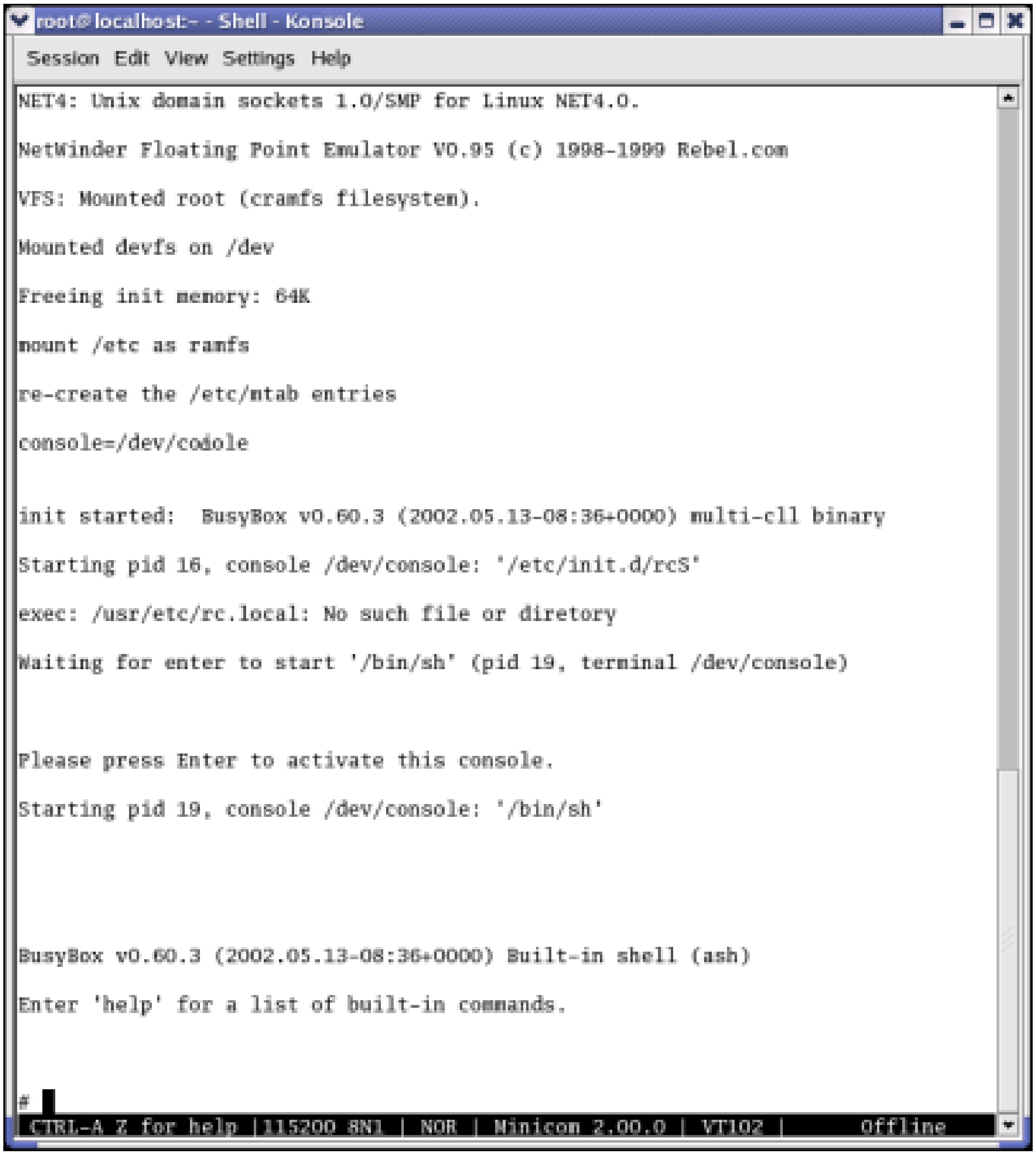
Figure 4-22 vivi Prompt

Now the SMC contains vivi (Bootloader), zImage (kernel), and root.cramfs (Root File System) images.

Please execute **'boot'** command on vivi prompt to boot the target board.

```
vivi> boot
```

Or you can also power **OFF** the board and power **ON** again. In this case target board will wait for some inputs defined by developer. If we do not input anything or press “Enter”, target board begins to boot. After progressed booting of target board, press **‘Ctrl + C’** and **‘Enter’** key, then you can begin to use shell prompt of target board system.


root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

```
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.

NetWinder Floating Point Emulator V0.95 (c) 1998-1999 Rebel.com

VFS: Mounted root (cramfs filesystem).

Mounted devfs on /dev

Freeing init memory: 64K

mount /etc as ramfs

re-create the /etc/mtab entries

console=/dev/codole


init started:  BusyBox v0.60.3 (2002.05.13-08:36+0000) multi-cll binary

Starting pid 16, console /dev/console: '/etc/init.d/rcS'

exec: /usr/etc/rc.local: No such file or diretory

Waiting for enter to start '/bin/sh' (pid 19, terminal /dev/console)


Please press Enter to activate this console.

Starting pid 19, 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.


# 
 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.00.0 | VT102 |       Offline
```

Figure 4-23 after booting the Target Board

Now you can start downloading all the images to target board by using **ztelnet** utility and fuse the NAND flash memory. Ethernet interface is used to transfer files which are more faster than as we did earlier.

4.8 Ztelnet

4.8.1 Installing ztelnet

Please execute the following command to install the **ztelnet** RPM.

```
[root@localhost root]# rpm –i  ztelnet-0.9.1-7mz.i386.rpm
```

While using **ztelnet**, target board has to be booted. The SMC which is provided with SMDK 2410 Board contains **vivi**, **kernel image**, **root file system**, so you can boot target board by using this **SMC**.

Now you can download compiled images to the target board by using **ztelnet**. Before downloading the images, connect host PC and target board by Ethernet cable. The downloading of images can be done by using two terminal windows,

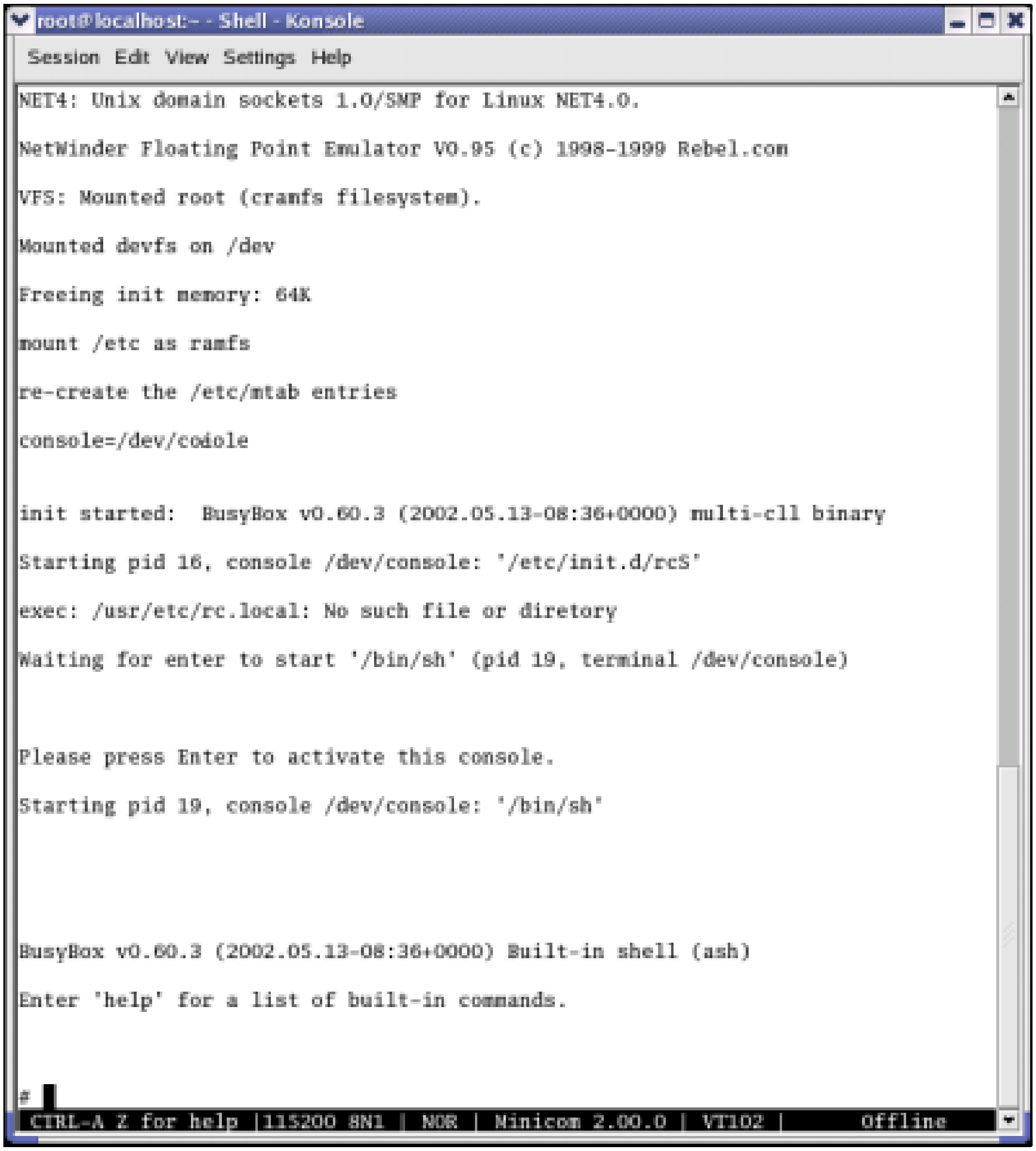
1. The terminal which is used for ztelnet.
2. And the other one which executes Minicom.

Terminal 1: Terminal which location is **/image** directory
Terminal 2: Terminal which executes **Minicom** (console of target board)

4.9 Executing Minicom

Terminal 1:	**# cd /image**
Terminal 2:	**# minicom** Switch **ON** the target Board, after progressed booting of target board, press **'Enter'** key, then you can begin to use shell of target board system.

```
[root@localhost root]# cd /home/test/image
[root@localhost image]#
```

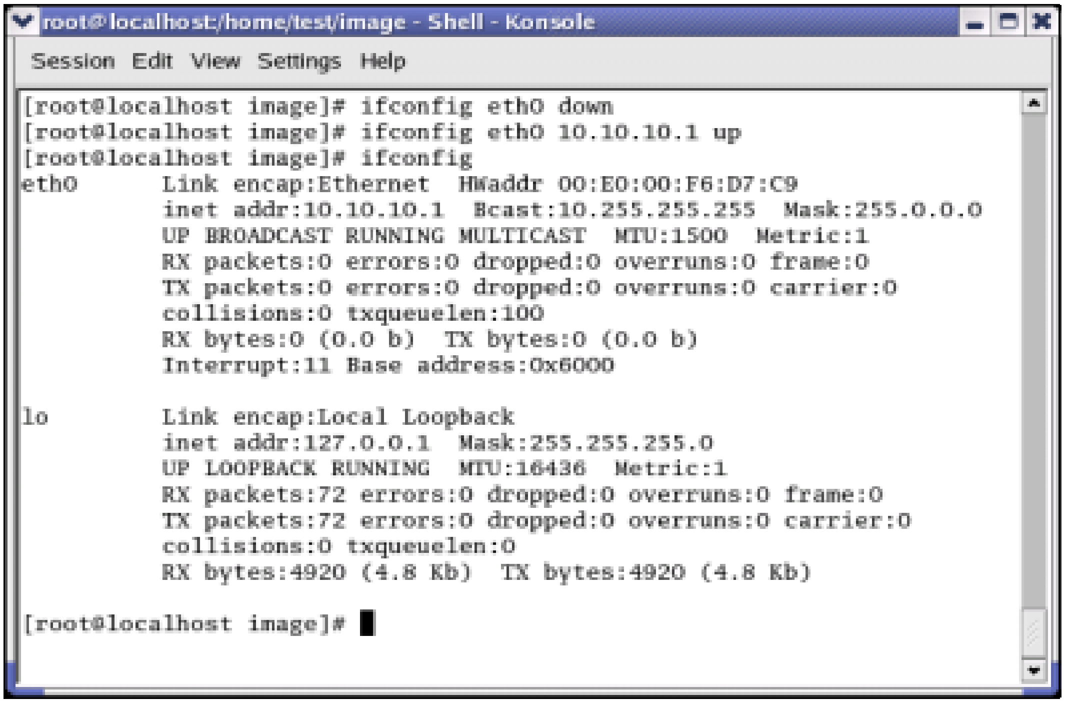

root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

```
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.

NetWinder Floating Point Emulator V0.95 (c) 1998-1999 Rebel.com

VFS: Mounted root (cramfs filesystem).

Mounted devfs on /dev

Freeing init memory: 64K

mount /etc as ramfs

re-create the /etc/mtab entries

console=/dev/codole


init started:  BusyBox v0.60.3 (2002.05.13-08:36+0000) multi-cll binary

Starting pid 16, console /dev/console: '/etc/init.d/rcS'

exec: /usr/etc/rc.local: No such file or diretory

Waiting for enter to start '/bin/sh' (pid 19, terminal /dev/console)


Please press Enter to activate this console.

Starting pid 19, 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.


#
 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.00.0 | VT102 |      Offline
```

Figure 4-24 Booting Target Board

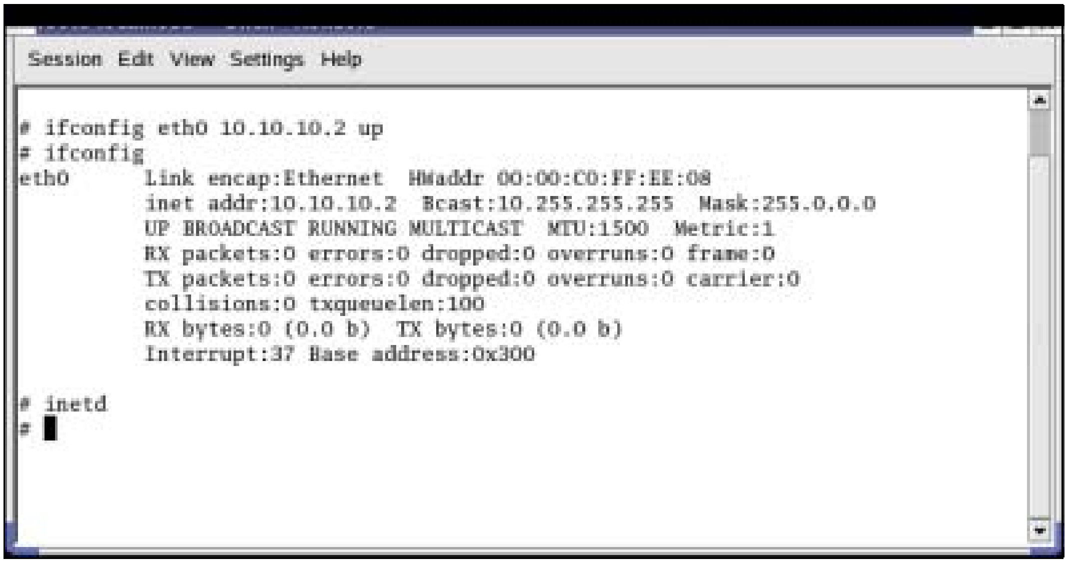
4.10 Setting up an IP address for Host PC and SMDK 2410 Target Board

Terminal 1:	**# ifconfig eth0 down** **# ifconfig eth0 10.10.10.1 up :** Set up an arbitrary IP.
Terminal 2:	**# ifconfig eth0 10.10.10.2 :** Set up IP that can make a pair with that of host PC. **# inetd**


root@localhost:/home/test/image - Shell - Konsole
Session Edit View Settings Help

```
[root@localhost image]# ifconfig eth0 down
[root@localhost image]# ifconfig eth0 10.10.10.1 up
[root@localhost image]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:E0:00:F6:D7:C9
          inet addr:10.10.10.1  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:11 Base address:0x6000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.255.255.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:72 errors:0 dropped:0 overruns:0 frame:0
          TX packets:72 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4920 (4.8 Kb)  TX bytes:4920 (4.8 Kb)

[root@localhost image]# █
```

Figure 4-25 Setting arbitrary IP

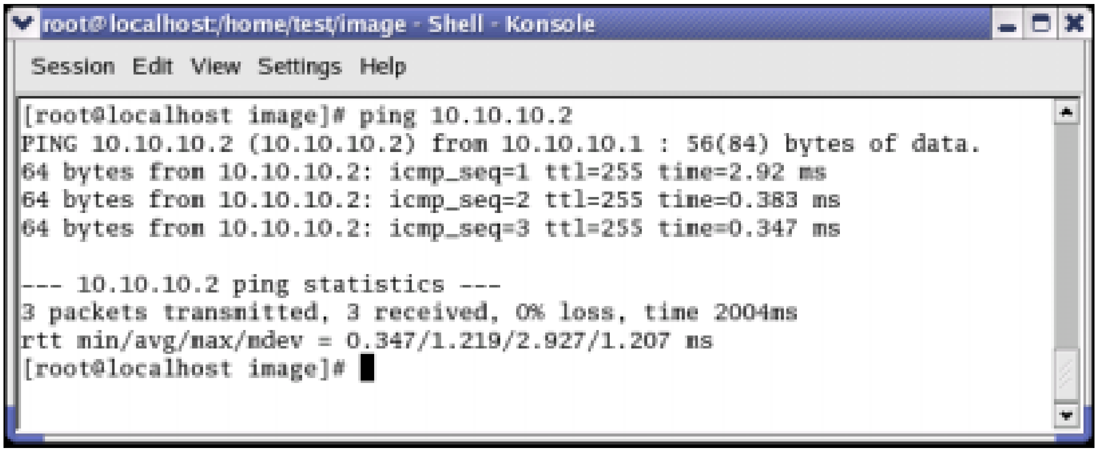

[redacted]

```
# ifconfig eth0 10.10.10.2 up
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:00:C0:FF:EE:08
          inet addr:10.10.10.2  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:37 Base address:0x300

# inetd
# █
```

Figure 4-26 ifconfig

4.11 Confirming the connection between Host PC and Target Board

Terminal 1:	**# ping 10.10.10.2** **:** We can confirm that the Host PC and Target Board can communicate.

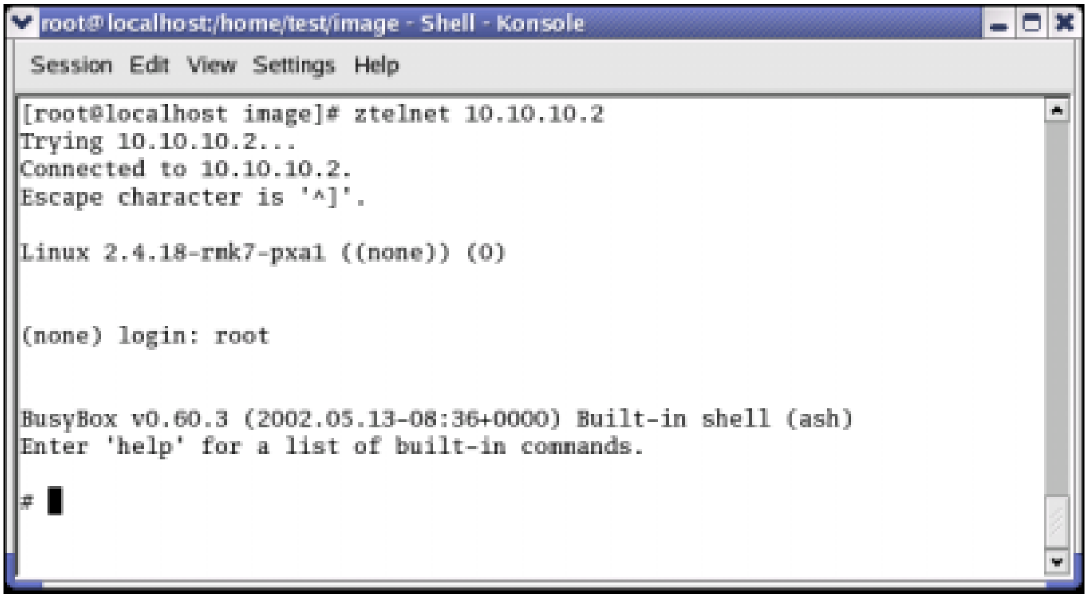

root@localhost:/home/test/image - Shell - Konsole
Session Edit View Settings Help

```
[root@localhost image]# ping 10.10.10.2
PING 10.10.10.2 (10.10.10.2) from 10.10.10.1 : 56(84) bytes of data.
64 bytes from 10.10.10.2: icmp_seq=1 ttl=255 time=2.92 ms
64 bytes from 10.10.10.2: icmp_seq=2 ttl=255 time=0.383 ms
64 bytes from 10.10.10.2: icmp_seq=3 ttl=255 time=0.347 ms

--- 10.10.10.2 ping statistics ---
3 packets transmitted, 3 received, 0% loss, time 2004ms
rtt min/avg/max/mdev = 0.347/1.219/2.927/1.207 ms
[root@localhost image]# █
```

Figure 4-27 Ping Test

4.12 Connecting Host PC to Target Board by using ztelnet

Terminal 1 :	**# ztelnet 10.10.10.2** Login by root account, so that you won't need to input password, and then press **'Enter'** key.

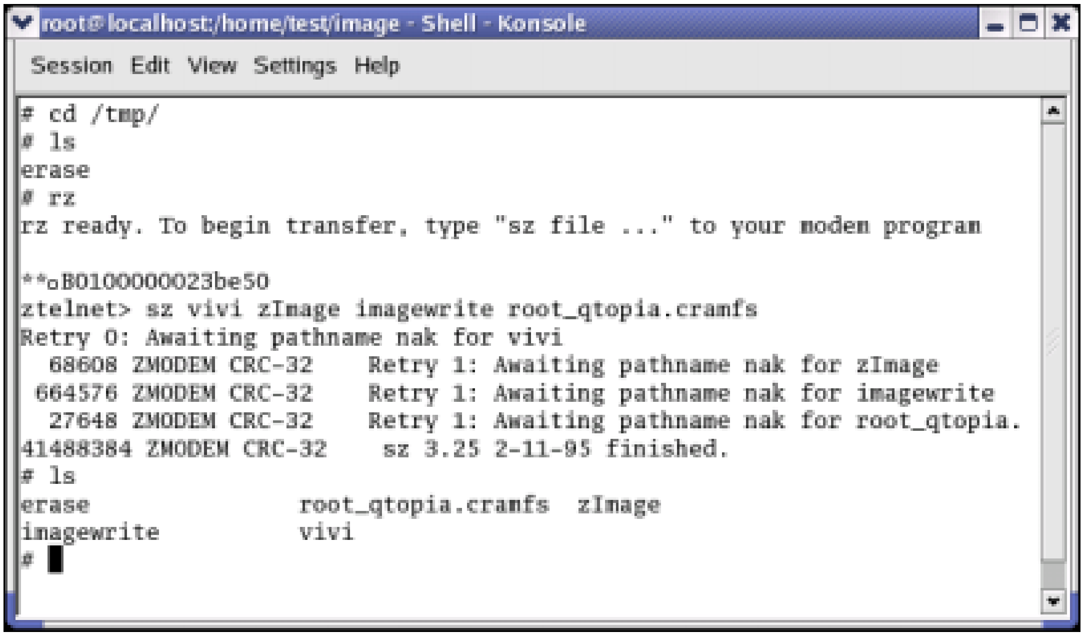

root@localhost:/home/test/image - Shell - Konsole
Session Edit View Settings Help

```
[root@localhost image]# ztelnet 10.10.10.2
Trying 10.10.10.2...
Connected to 10.10.10.2.
Escape character is '^]'.

Linux 2.4.18-rmk7-pxa1 ((none)) (0)


(none) login: root


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

#
```

Figure 4-28 ztelnet

4.13 Transferring Images by ztelnet

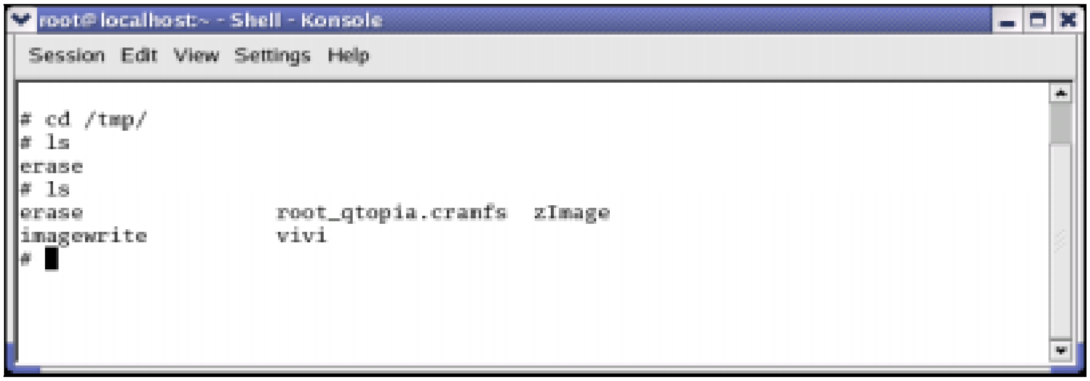
Terminal 1:	**# cd /tmp** **# rz** Pushing '**Ctrl +]', 'ztelnet>'** console appears. **ztelnet>** sz vivi zImage root_qtopia.cramfs imagewrite
Terminal 2:	**# cd /tmp**


root@localhost:/home/test/image - Shell - Konsole
Session Edit View Settings Help

```
# cd /tmp/
# ls
erase
# rz
rz ready. To begin transfer, type "sz file ..." to your modem program

**oB0100000023be50
ztelnet> sz vivi zImage imagewrite root_qtopia.cramfs
Retry 0: Awaiting pathname nak for vivi
  68608 ZMODEM CRC-32    Retry 1: Awaiting pathname nak for zImage
 664576 ZMODEM CRC-32    Retry 1: Awaiting pathname nak for imagewrite
  27648 ZMODEM CRC-32    Retry 1: Awaiting pathname nak for root_qtopia.
41488384 ZMODEM CRC-32    sz 3.25 2-11-95 finished.
# ls
erase                    root_qtopia.cramfs  zImage
imagewrite               vivi
# 
```

Figure 4-29 Copying Image files to target board using ztelnet

Only **/tmp** directory can be used for both reading and writing, all directories except **/tmp** are read-only. But **/tmp** directory is ramfs, so if power supply is cut, all images downloaded are deleted.
If you want to store the images, you have to write those to flash memory by using a special utility.
After downloading all images, check the downloaded items by executing **'ls'** command in both the consoles (Terminal 1 and Terminal 2) as shown in above and below figure.


root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

```
# cd /tmp/
# ls
erase
# ls
erase                    root_qtopia.cramfs  zImage
imagewrite               vivi
# 
```

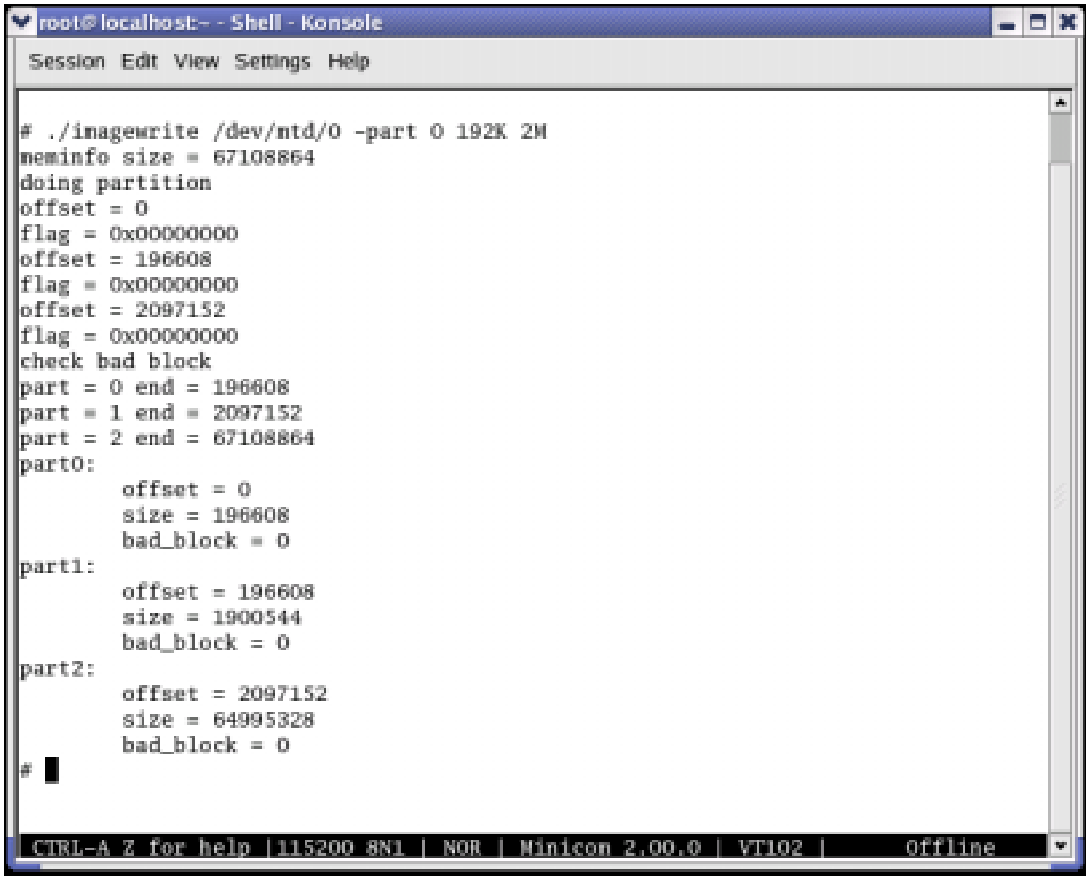
Figure 4-30 Image files on Target Board

4.14 Imagewrite

4.14.1 Creating partitions in SMC

Terminal executing minicom enable host PC user to work inside target board. Now create three partitions in SMC inside Minicom terminal.

```
# ./imagewrite /dev/mtd/0 -part 0 192K 2M
```


root@ localhost:~ - Shell - Konsole
Session Edit View Settings Help

```
# ./imagewrite /dev/mtd/0 -part 0 192K 2M
meminfo size = 67108864
doing partition
offset = 0
flag = 0x00000000
offset = 196608
flag = 0x00000000
offset = 2097152
flag = 0x00000000
check bad block
part = 0 end = 196608
part = 1 end = 2097152
part = 2 end = 67108864
part0:
        offset = 0
        size = 196608
        bad_block = 0
part1:
        offset = 196608
        size = 1900544
        bad_block = 0
part2:
        offset = 2097152
        size = 64995328
        bad_block = 0
# 
 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.00.0 | VT102 |      Offline
```

Figure 4-31 Partitioning SMC

Divide SMC to three partitions, and the size of each partition is as follows:

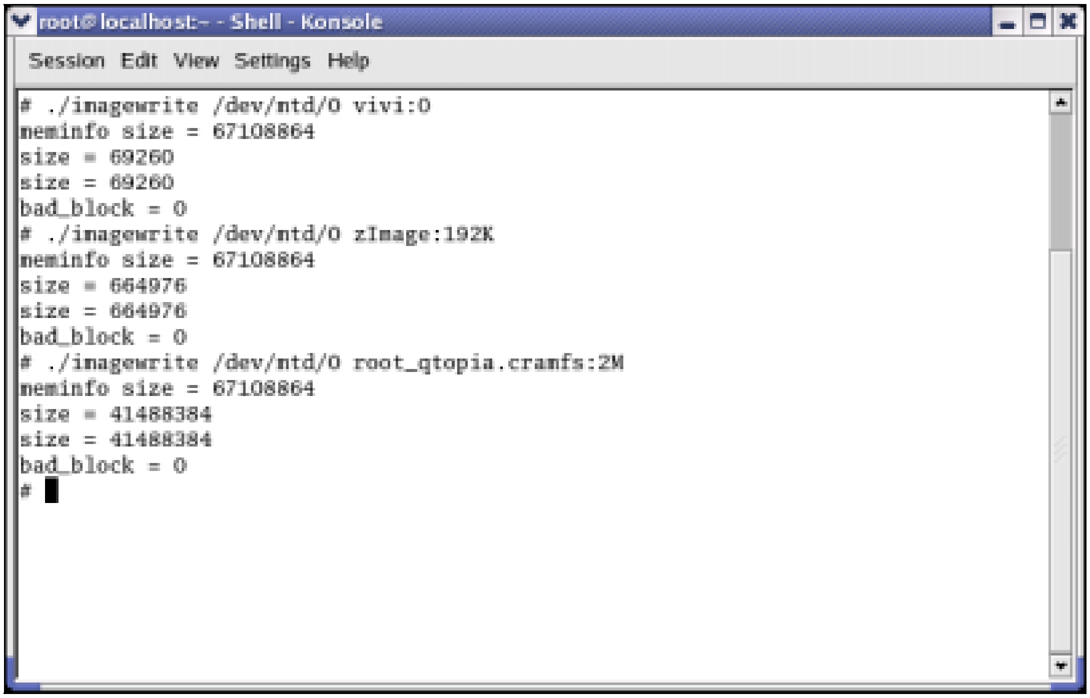
Vivi Bootloader	**:** 0~192KB
Kernel Image	**:** 192KB~2MB
root_qtopia.cramfs	**:** 2MB~End-part

write '**vivi'** at '**0~192KB'** partition, '**zImage'** at '**192KB~2MB'** partition, and '**root_qtopia.cramfs'** at '**2MB~End-part'** partition.

4.14.2 Copying the Images to SMC using imagewrite utility

Usage : # ./imagewrite <mtd_dev> <file:offset>

# ./imagewrite /dev/mtd/0 vivi:0	: Store vivi in SMC.
# ./imagewrite /dev/mtd/0 zImage:192K	: Store zImage in SMC.
# ./imagewrite /dev/mtd/0 root_qtopia.cramfs:2M	: Store root_qtopia.cramfs in SMC.


root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

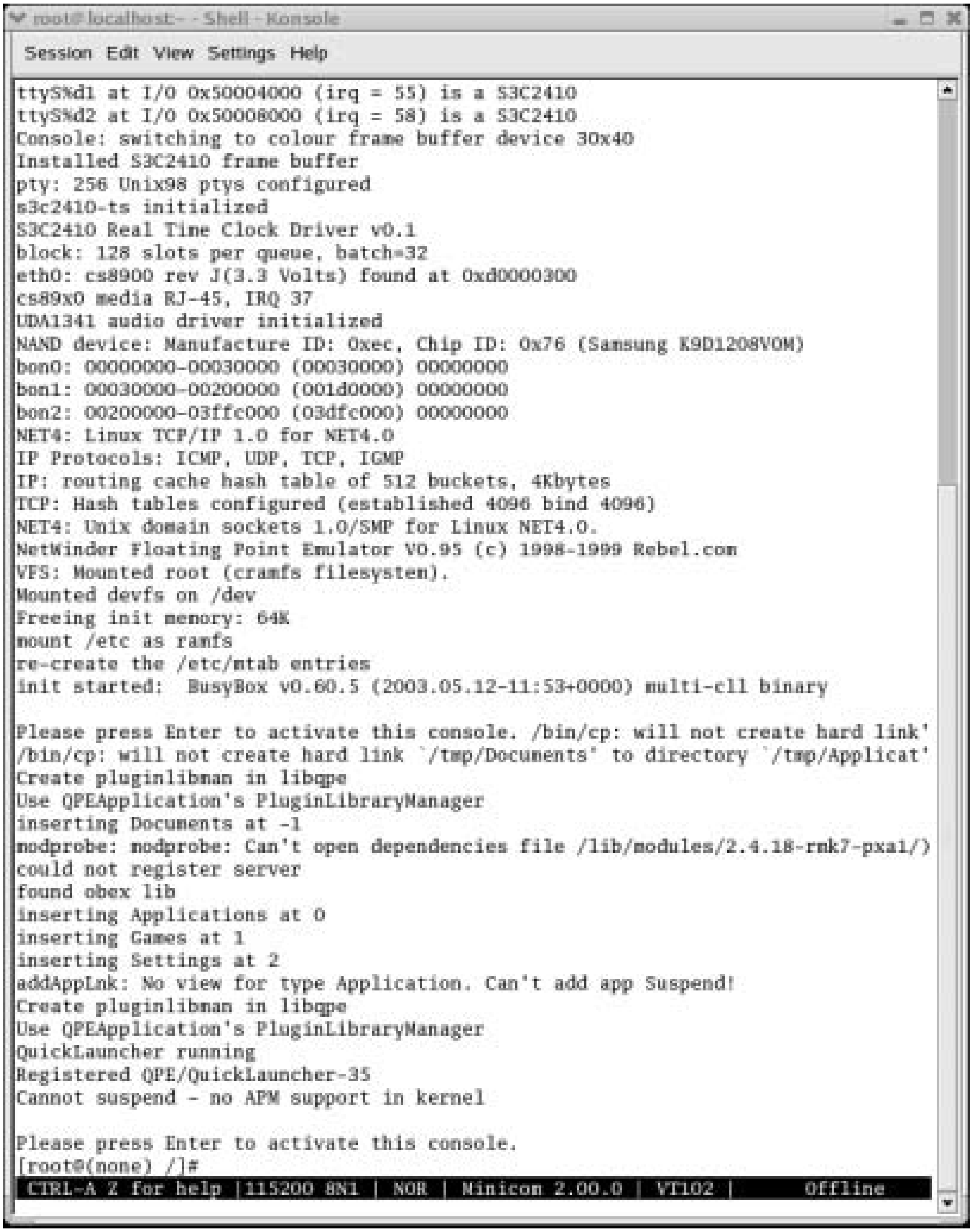
```
# ./imagewrite /dev/mtd/0 vivi:0
meminfo size = 67108864
size = 69260
size = 69260
bad_block = 0
# ./imagewrite /dev/mtd/0 zImage:192K
meminfo size = 67108864
size = 664976
size = 664976
bad_block = 0
# ./imagewrite /dev/mtd/0 root_qtopia.cramfs:2M
meminfo size = 67108864
size = 41488384
size = 41488384
bad_block = 0
# 
```

Figure 4-32 Writing Images on SMC

After completing the above procedure please reboot the target board.

After progressed booting of target board, press **'Ctrl + C'** and **'Enter'** key, then you can begin to use shell of target board system.

The console display will look as shown in the next figure. The root file system is **Qtopia.** The LCD screen on the target board will show different applications and related icons.


root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

```
ttyS%d1 at I/O 0x50004000 (irq = 55) is a S3C2410
ttyS%d2 at I/O 0x50008000 (irq = 58) is a S3C2410
Console: switching to colour frame buffer device 30x40
Installed S3C2410 frame buffer
pty: 256 Unix98 ptys configured
s3c2410-ts initialized
S3C2410 Real Time Clock Driver v0.1
block: 128 slots per queue, batch=32
eth0: cs8900 rev J(3.3 Volts) found at 0xd0000300
cs89x0 media RJ-45, IRQ 37
UDA1341 audio driver initialized
NAND device: Manufacture ID: 0xec, Chip ID: 0x76 (Samsung K9D1208V0M)
bon0: 00000000-00030000 (00030000) 00000000
bon1: 00030000-00200000 (001d0000) 00000000
bon2: 00200000-03ffc000 (03dfc000) 00000000
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP, IGMP
IP: routing cache hash table of 512 buckets, 4Kbytes
TCP: Hash tables configured (established 4096 bind 4096)
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
NetWinder Floating Point Emulator V0.95 (c) 1998-1999 Rebel.com
VFS: Mounted root (cramfs filesystem).
Mounted devfs on /dev
Freeing init memory: 64K
mount /etc as ramfs
re-create the /etc/mtab entries
init started:  BusyBox v0.60.5 (2003.05.12-11:53+0000) multi-cll binary

Please press Enter to activate this console. /bin/cp: will not create hard link'
/bin/cp: will not create hard link `/tmp/Documents' to directory `/tmp/Applicat'
Create pluginlibman in libqpe
Use QPEApplication's PluginLibraryManager
inserting Documents at -1
modprobe: modprobe: Can't open dependencies file /lib/modules/2.4.18-rmk7-pxa1/)
could not register server
found obex lib
inserting Applications at 0
inserting Games at 1
inserting Settings at 2
addAppLnk: No view for type Application. Can't add app Suspend!
Create pluginlibman in libqpe
Use QPEApplication's PluginLibraryManager
QuickLauncher running
Registered QPE/QuickLauncher-35
Cannot suspend - no APM support in kernel

Please press Enter to activate this console.
[root@(none) /]#
 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.00.0 | VT102 |      Offline
```

Figure 4-33 Writing Images on SMC

Index