Installation Manual for S3C2440 (Linux)

Confidential Property of Samsung Electronics Co., Ltd.

Installation Manual for S3C2440 (Linux)
Copyright © 2004 Samsung Electronics Co, Ltd. All Rights Reserved.

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
2003-10-05	1.0	TS Team	Master Copy
2004-01-27	1.1	Linux Team	Update Document
2004-02-03	1.2	TS Team	Revised Document
2004-02-10	1.3	TS Team	Images Updated

Contents

1 INTRODUCTION 1

1.1 OVERVIEW 1
1.2 BLOCK DIAGRAM 2
1.3 SUPER USER MODE AND DOWNLOADING FILES 3

2 INSTALLING TOOLCHAIN 4

3 COMPILING BOOTLOADER, KERNEL AND ROOT FILE SYSTEM FOR SMDK2440 5

3.1 INTRODUCTION TO BOOTLOADER 5
3.1.1 What is Vivi 5
3.2 COMPILING VIVI 5
3.3 COMPILING KERNEL 8
3.4 CREATING ROOT FILE SYSTEM 10

4 PORTING EMBEDDED LINUX TO SMDK 2440 TARGET BOARD 11

4.1 PORTING LINUX WHILE TARGET BOARD BOOTING 11
4.2 MINICOM 11
4.3 ZTELNET 14
4.3.1 Install ztelnet 14
4.4 EXECUTING MINICOM 15
4.5 SETTING UP AN IP ADDRESS FOR HOST PC AND SMDK 2440 TARGET BOARD 16
4.6 CONFIRMING THE CONNECTION BETWEEN HOST PC AND TARGET BOARD 17
4.7 CONNECTING HOST PC TO TARGET BOARD BY USING ZTELNET 17
4.8 TRANSFERRING IMAGES BY ZTELNET 18
4.9 IMAGEWRITE 19
4.9.1 Creating partitions in SMC 19
4.9.2 Copying the Images to SMC by using imagewrite utility 20

5 PORTING LINUX WHEN TARGET BOARD BOOTING IS DISABLED 21

5.1 UPLOADING 'VIVI' USING JTAG CABLE 21
5.2 UPLOADING IMAGES TO THE TARGET BOARD USING VIVI 25
5.3 SMC PARTITIONING AND WRITING VIVI IMAGE 26
5.4 WRITING KERNEL IMAGE 29
5.5 WRITING ROOT FILE SYSTEM IMAGE 31

6 INSTALLING CAMERA MODULE 34

6.1 COMPILING CAMERA MODULE 34

6.2 PORTING CAMERA MODULE 35

6.3 TRANSFERRING IMAGES BY ZTELNET 35

INDEX 36

Figures

FIGURE 1-1 S3C2440X BLOCK DIAGRAM....2
FIGURE 3-1 VIVI CONFIGURATION....6
FIGURE 3-2 INPUTTING VIVI CONFIGURATION FILE....6
FIGURE 3-3 SAVING NEW KERNEL CONFIGURATION....7
FIGURE 3-4 KERNEL CONFIGURATIONS....8
FIGURE 3-5 INPUTTING KERNEL CONFIGURATION FILE....9
FIGURE 3-6 SAVING NEW KERNEL CONFIGURATION....9
FIGURE 4-1 MINICOM SETUP....11
FIGURE 4-2 SERIAL PORT SETUP I....12
FIGURE 4-3 SERIAL PORT SETUP II....12
FIGURE 4-4 HARDWARE/SOFTWARE FLOW CONTROL SETUP....13
FIGURE 4-5 SAVING MINICOM SETUP....13
FIGURE 4-6 EXITING FROM MINICOM....14
FIGURE 4-7 BOOTING TARGET BOARD....15
FIGURE 4-8 SETTING ARBITRARY IP....16
FIGURE 4-9 IFCONFIG....16
FIGURE 4-10 PING TEST....17
FIGURE 4-11 ZTELNET....17
FIGURE 4-12 COPYING IMAGE FILES TO TARGET BOARD USING ZTELNET....18
FIGURE 4-13 IMAGE FILES ON TARGET BOARD....18
FIGURE 4-14 PARTITIONING SMC....19
FIGURE 4-15 WRITING IMAGES ON SMC....20
FIGURE 5-1 JFLASH-S3C2440 -HELP....21
FIGURE 5-2 SELECTING FUNCTION TO TEST JFLASH-S3C2440....22
FIGURE 5-3 INPUTTING TARGET BLOCK NUMBER....23
FIGURE 5-4 COMPLETION OF VIVI....24
FIGURE 5-5 VIVI SHELL PROMPT....24
FIGURE 5-6 XMODEM_INITIAL_TIMEOUT SETTINGS....25
FIGURE 5-7 PARTITIONING SMC....26
FIGURE 5-8 XMODEM X-FER MODE FOR VIVI....27
FIGURE 5-9 ENTERING FILENAME FOR VIVI....27
FIGURE 5-10 VIVI DOWNLOAD STATUS....28
FIGURE 5-11 VIVI PROMPT....28
FIGURE 5-12 XMODEM X-FER MODE FOR KERNEL IMAGE....29

FIGURE 5-13 ENTERING FILENAME FOR ZIMAGE 29
FIGURE 5-14 ZIMAGE DOWNLOAD STATUS 30
FIGURE 5-15 VIVI PROMPT 30
FIGURE 5-16 XMODEM X-FER MODE FOR ROOT.CRAMFS IMAGE 31
FIGURE 5-17 ENTERING FILENAME FOR ROOT.CRAMFS 31
FIGURE 5-18 ROOT.CRAMFS DOWNLOAD STATUS 32
FIGURE 5-19 VIVI PROMPT 32
FIGURE 5-20 AFTER BOOTING THE TARGET BOARD 33
FIGURE 6-1 EDITING MAKEFILE 34

1 Introduction

1.1 Overview

This manual describes Installing Samsung's S3C2440X BSP on LINUX OS. This product is designed to provide hand-held devices and general applications with cost-effective, low-power, and high-performance micro-controller solution in small die size. To reduce total system cost, the S3C2440X includes the following components separate 16KB Instruction and 16KB Data Cache, MMU to handle virtual memory management, LCD Controller (STN & TFT), NAND Flash Boot Loader, System Manager (chip select logic and SDRAM Controller), 3-ch UART, 4-ch DMA, 4-ch Timers with PWM, I/O Ports, RTC, 8-ch 10-bit ADC and Touch Screen Interface, Camera interface, IIC-BUS Interface, IIS-BUS Interface, USB Host, USB Device, SD Host & Multi-Media Card Interface, 2-ch SPI and PLL for clock generation.

The S3C2440X was developed using an ARM920T core, 0.13um CMOS standard cells and a memory complier. Its low-power, simple, elegant and fully static design is particularly suitable for cost- and power-sensitive applications. It adopts a new bus architecture called Advanced Micro controller Bus Architecture (AMBA).

The S3C2440X offers outstanding features with its CPU core, a 16/32-bit ARM920T RISC processor designed by Advanced RISC Machines, Ltd. The ARM920T implements MMU, AMBA BUS, and Harvard cache architecture with separate 16KB instruction and 16KB data caches, each with an 8-word line length.

By providing a complete set of common system peripherals, the S3C2440X minimizes overall system costs and eliminates the need to configure additional components. The integrated on-chip functions that are described in this document include:

- 1.2V internal, 1.8V/2.5V/3.3V memory, 3.3V external I/O microprocessor with 16KB I-Cache/16KB D-Cache/MMU
- External memory controller (SDRAM Control and Chip Select logic)
- LCD controller (up to 4K color STN and 256K color TFT) with 1-ch LCD-dedicated DMA
- 4-ch DMA with external request pins
- 3-ch UART (IrDA1.0, 64-Byte Tx FIFO, and 64-Byte Rx FIFO) / 2-ch SPI
- 1-ch multi-master IIC-BUS/1-ch IIS-BUS controller
- SD Host interface version 1.0 & Multi-Media Card Protocol version 2.11 compatible
- 2-port USB Host /1- port USB Device (ver 1.1)
- 4-ch PWM timers & 1-ch internal timer
- Watch Dog Timer
- 130-bit general purpose I/O ports / 24-ch external interrupt source
- Power control: Normal, Slow, Idle and Sleep mode
- 8-ch 10-bit ADC and Touch screen interface
- RTC with calendar function
- On-chip clock generator with PLL

1.2 Block Diagram

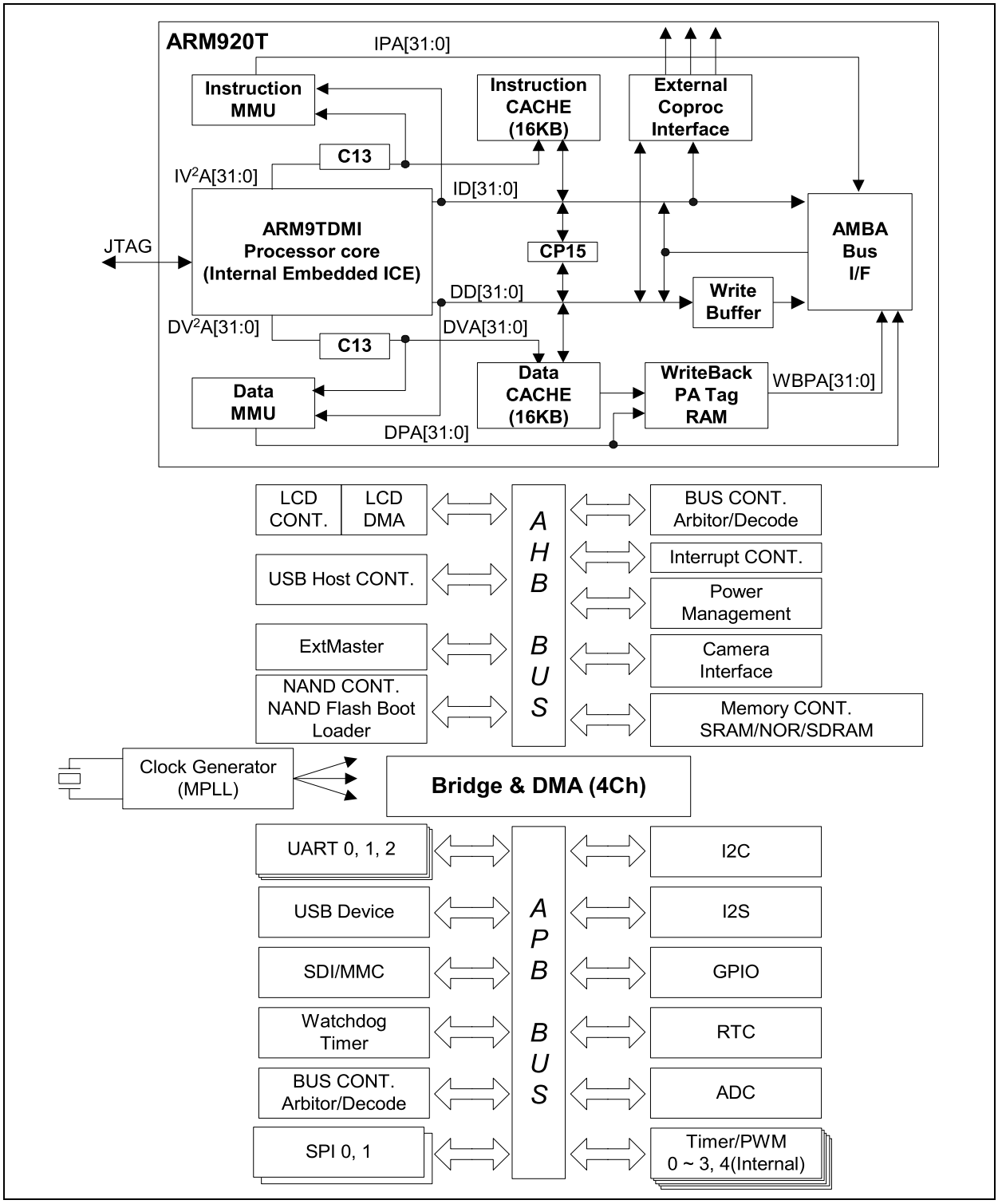

Figure 1-1 S3C2440X Block Diagram

1.3 Super User Mode and Downloading Files

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

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

Please download the following files from www.samsungsemi.com and copy it to the working directory */home/test*.

Filename	Description
s3c2440_vivi_r1.0.tar.bz2	bootloader
s3c2440_kernel2.4.18_r1.2.tar.bz2	kernel
s3c2440_kernel2.4.18_module_camera.tar.bz2	camera
s3c2440_kernel2.4.18_module_sdmmc.tar.bz2	MMC
cross-2.95.3.tar.bz2	Toolchain
root.cramfs	small size root file system (only booting)
root_qtopia_2440.cramfs	Qtopia window Root file System
ztelnet-0.9.1-7mz.i386.rpm	ztelnet Application
jflash-s3c2440	SMC Application

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

```
[root@localhost test]#
[root@localhost test]# ls
s3c2440_vivi_r1.0.tar.bz2
s3c2440_kernel2.4.18_r1.2.tar.bz2
s3c2440_kernel2.4.18_module_camera.tar.bz2
s3c2440_kernel2.4.18_module_sdmmc.tar.bz2
cross-2.95.3.tar.bz2
root.cramfs
root_qtopia_2440.cramfs
jflash-s3c2440
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
[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, Kernel and Root File System for SMDK2440

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 lost all data downloaded if you cut 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 *'s3c2440_vivi_r1.0.tar.bz2'*. Extract it executing following command.

```
[root@localhost test]#
[root@localhost test]# tar jxvf s3c2440_vivi_r1.0.tar.bz2
```

Go to **'s3c2440_vivi_rel'** directory created after extracting the tarball and then execute the *'make menuconfig'* command.

```
[root@localhost test]# cd s3c2440_vivi_rel
[root@localhost s3c2440_vivi_rel]# make menuconfig
```

Please Select *'Load an Alternate Configuration File'*

Figure 3-1 Vivi configuration

Please enter the path of the configuration file to load *'arch/def-configs/smdk2440'*.

Figure 3-2 Inputting Vivi configuration file

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

Figure 3-3 Saving New Kernel Configuration

Finally save the set points and compile vivi by executing '*make*' command.

```
[root@localhost s3c2440_vivi_rel]# make
```

If the compilation of vivi progresses well, vivi binary image file will be created under '**/s3c2440_vivi_rel**' directory.

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

```
[root@localhost s3c2440_vivi_rel]# mkdir /home/test/image
[root@localhost s3c2440_vivi_rel]# cp vivi /home/test/image
```

```
[root@localhost s3c2440_vivi_rel]#  cd util
[root@localhost util]# arm-linux-gcc -o imagewrite imagewrite.c
[root@localhost s3c2440_vivi_rel]# cp imagewrite /home/test/image
```

After executing above command, imagewrite file will be generated under '*/util*' directory. **Imagewrite** is the utility to download the image files to the SMC (Smart Media Card). Copy Imagewrite utility under '*/home/test/image*' directory.

3.3 Compiling Kernel

Kernel source is compressed by the name of *'s3c2440_kernel2.4.18_r1.2.tar.bz2'*. Extract this bz2 file by executing the following command. After extracting the kernel tarball file 's3c2440_kernel2.4.18_rel' directory will generate.

```
[root@localhost test]# tar jxvf s3c2440_kernel2.4.18_r1.2.tar.bz2
[root@localhost test]# cd s3c2440_kernel2.4.18_rel
```

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 '**s3c2440_kernel2.4.18_rel**' directory.

Please enter the path of the configuration file to load *'arch/arm/def-configs/smdk2440'* file, after selecting *'Load on Alternate Configuration File'* menu.

```
[root@localhost s3c2440_kernel2.4.18_rel]# make menuconfig
```

Please select *'Load an Alternate Configuration File.'*

Figure 3-4 Kernel configurations

Please enter the name of the configuration file you wish to load *'arch/arm/def-configs/smdk2440'*.

Figure 3-5 Inputting Kernel configuration file

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

Figure 3-6 Saving New Kernel Configuration

```
[root@localhost s3c2440_kernel2.4.18_rel ]# make dep
[root@localhost s3c2440_kernel2.4.18_rel ]# make zImage
```

After executing above commands the Kernel image will be created in '***s3c2440_kernel2.4.18_rel***/arch/arm/boot' directory by the name of 'zImage'.

To port the target board easily copy 'zImage' (kernel image) to 'image' directory.

```
# s3c2440_kernel2.4.18_rel/arch/arm/boot/
[root@localhost boot]# cp zImage /home/test/image
```

3.4 Creating 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_2440.cramfs /home/test/image
[root@localhost test]# cp root.cramfs /home/test/image
```

All images (vivi, zImage, root.cramfs, root_qtopia_2440.cramfs, imagewrite) are collected in '/image' directory. In next chapter, we will learn about how to port these images to the target board.

4 Porting Embedded Linux to SMDK 2440 Target Board

4.1 Porting Linux while Target Board Booting

Now in this chapter we will learn how to write *vivi (bootloader)*, *zImage (kernel image)*, root_qtopia_2440.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 the target board, because all works are progressed in target board. Transfer all the images from image directory to the target board by using *ztelnet* utility.

4.2 Minicom

We have to transfer the images using **ztelnet**, before that you should know how to use **Minicom**. In this section we explain how to use **Minicom**.

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.
```

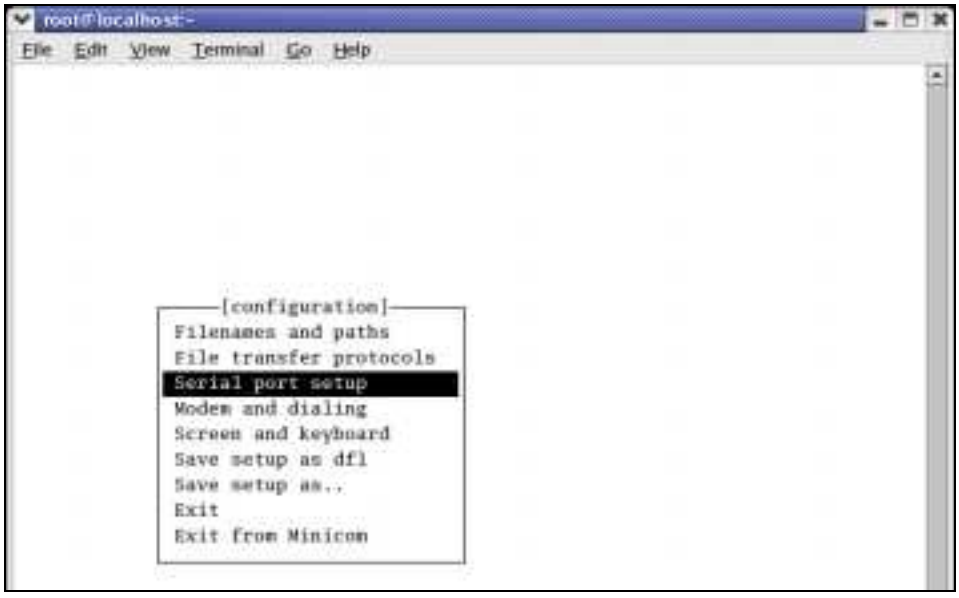

Figure 4-1 Minicom setup

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*.)

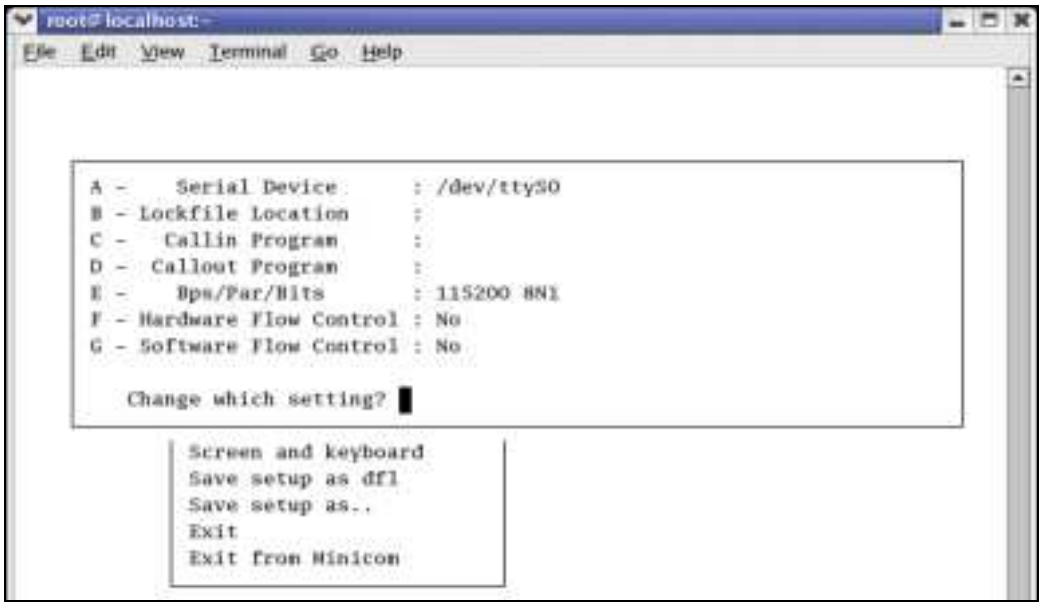

Figure 4-2 Serial Port setup I

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'**.

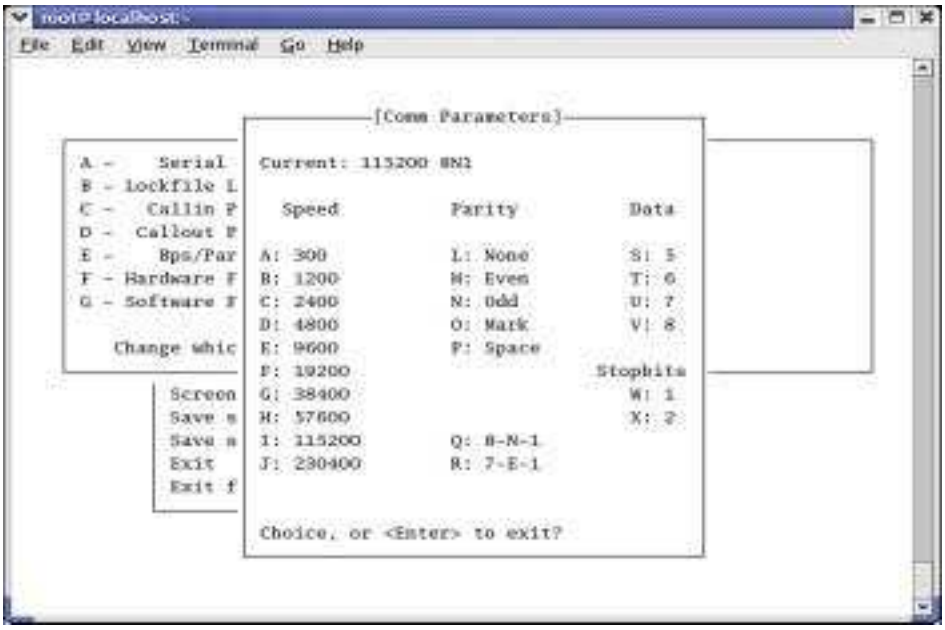

Figure 4-3 Serial Port setup II

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'**.

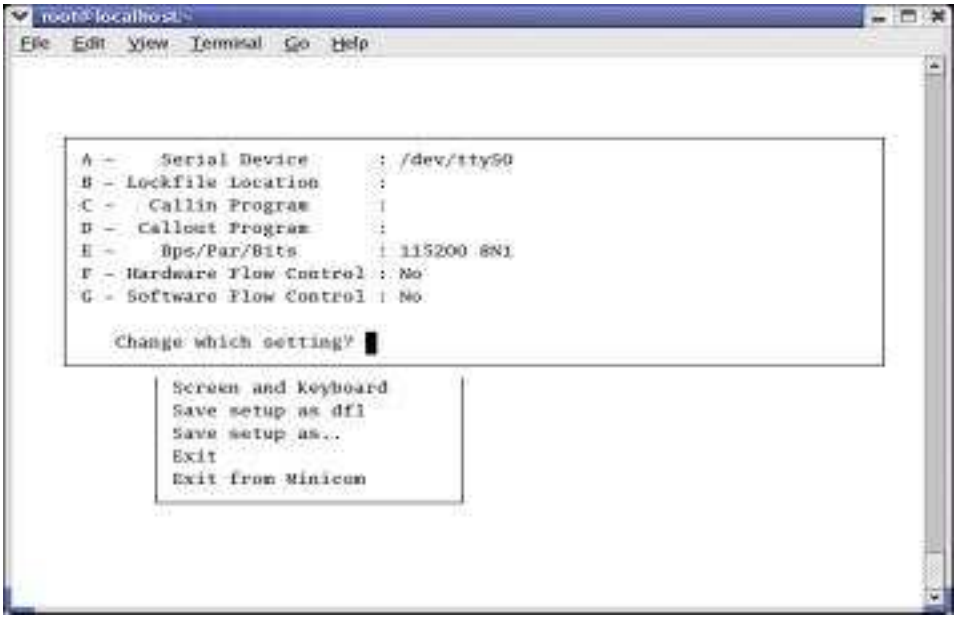

Figure 4-4 Hardware/Software Flow Control Setup

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

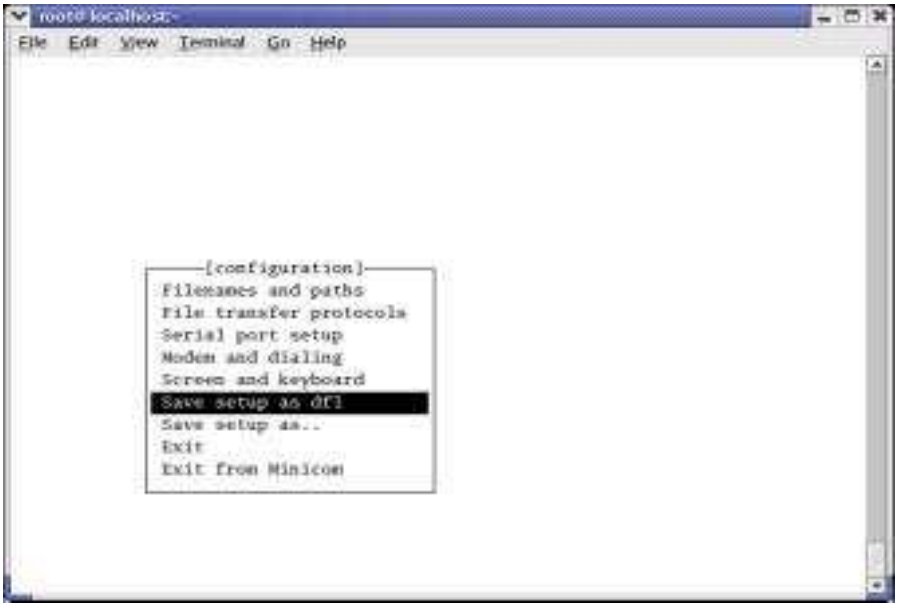

Figure 4-5 saving Minicom Setup

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.

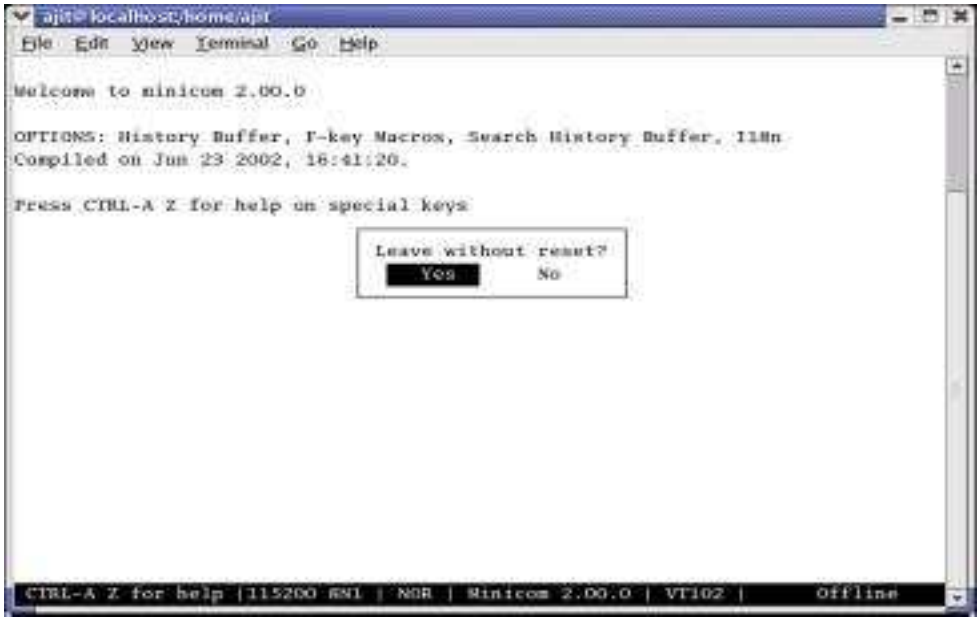

Figure 4-6 Exiting from Minicom

4.3 ztelnet

4.3.1 Install ztelnet

To use ztelnet program you have to install the ztelnet RPM by executing the following command.

```
[root@localhost test]# rpm -i  ztelnet-0.9.17mz.i363.rpm
```

When using **ztelnet**, target board has to be booted. The SMC which is provided with SMDK 2440 Board contains **vivi**, **kernel image**, **root filesystem**, 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)

4.4 Executing Minicom

Terminal 1:	**# cd /image**
Terminal 2:	**# minicom** Switch **ON** 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.

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

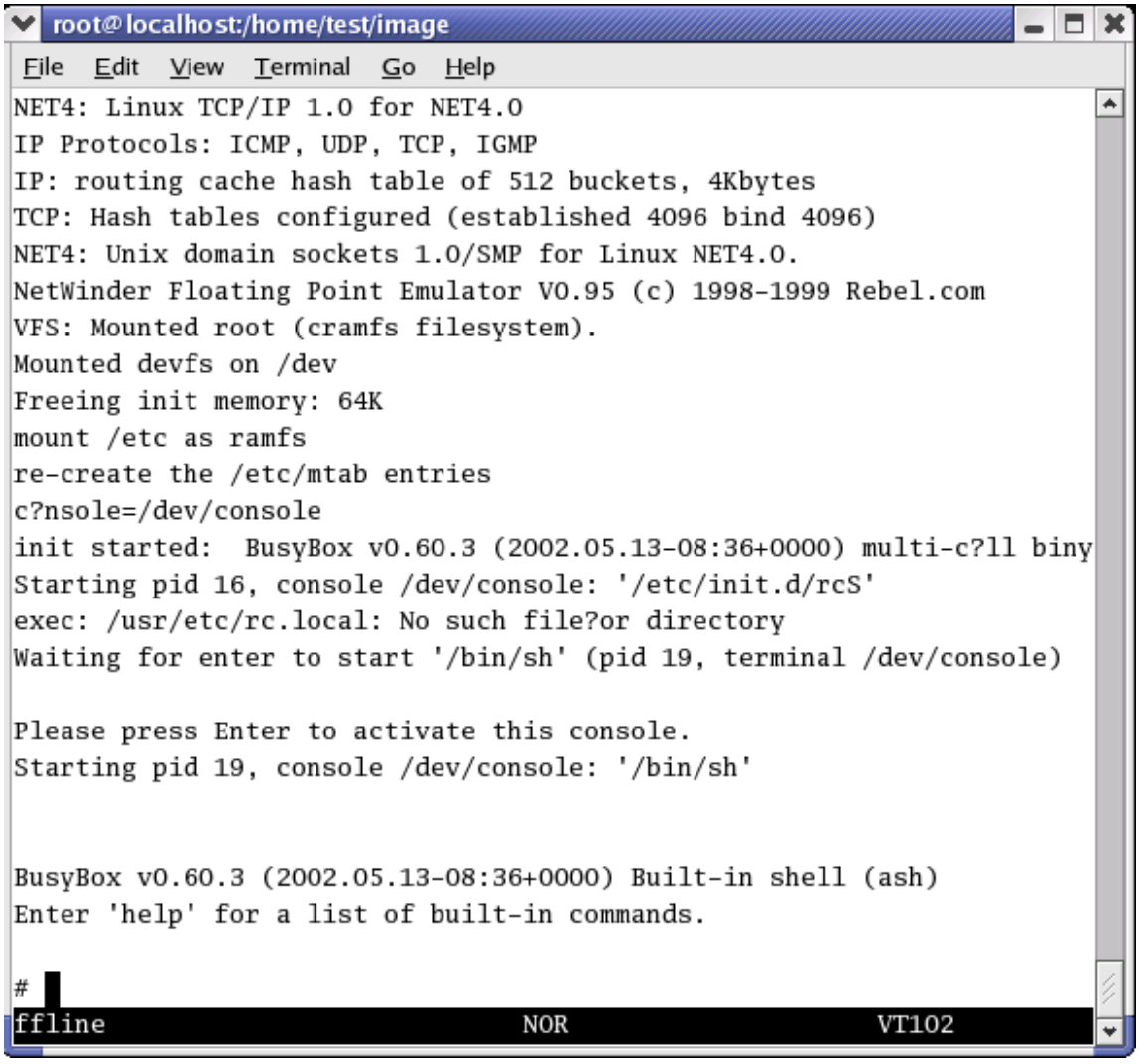

Figure 4-7 Booting Target Board

4.5 Setting up an IP address for Host PC and SMDK 2440 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**

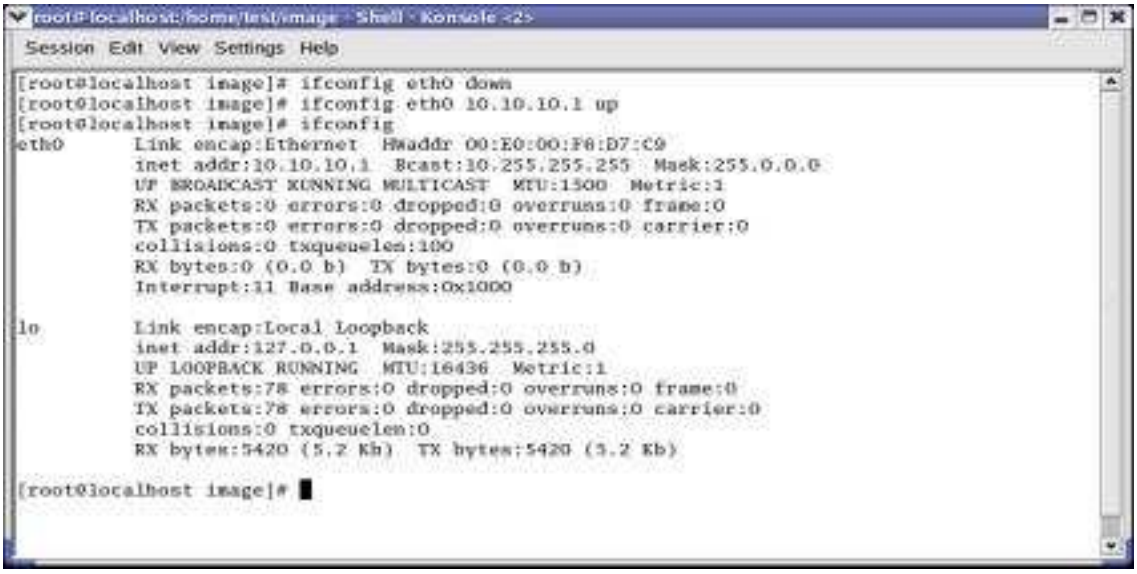

Figure 4-8 Setting arbitrary IP

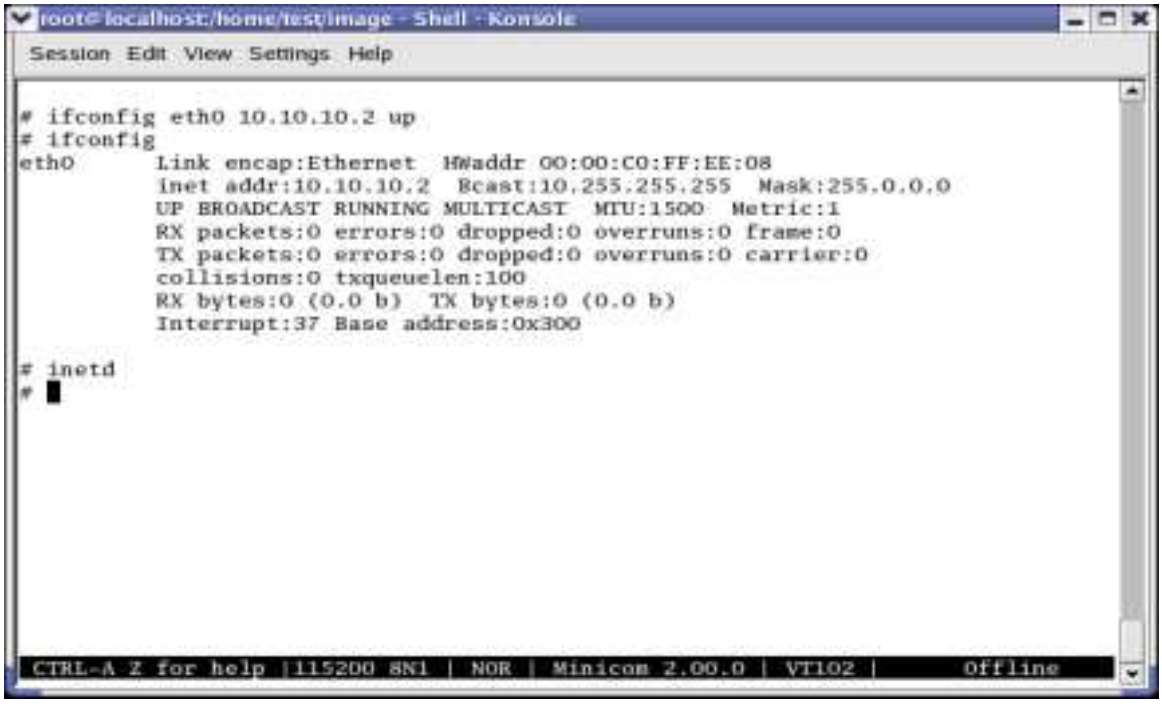

Figure 4-9 ifconfig

4.6 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.

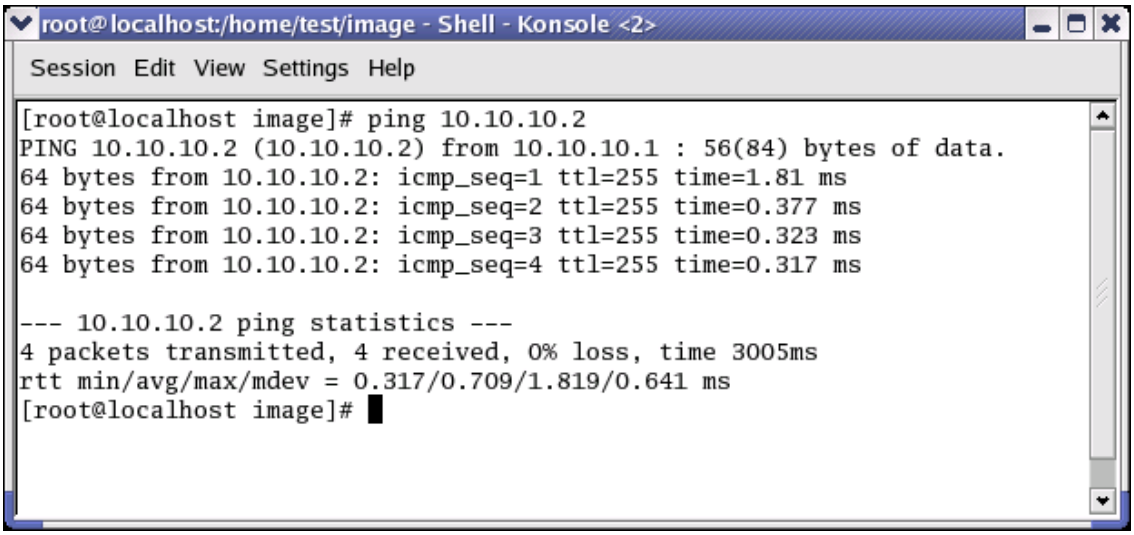

Figure 4-10 Ping Test

4.7 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.

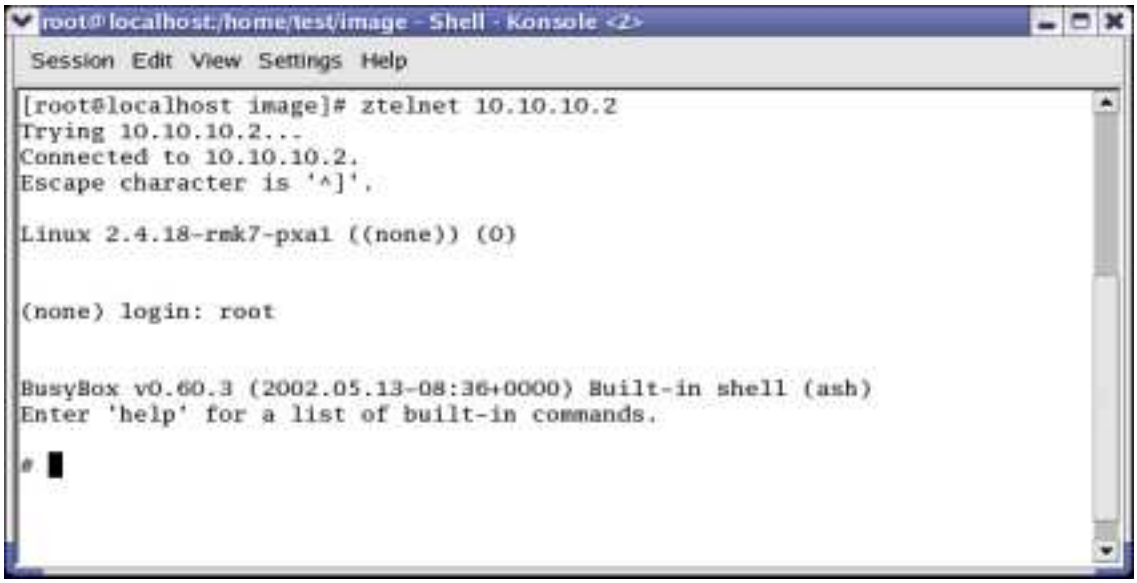

Figure 4-11 ztelnet

4.8 Transferring Images by ztelnet

Terminal 1:	**# cd /tmp** **# rz** Pushing '**Ctrl +]**', '**ztelnet>**' console appears. **ztelnet>** sz vivi zImage `root_qtopia_2440.cramfs` imagewrite
Terminal 2:	**# cd /tmp**

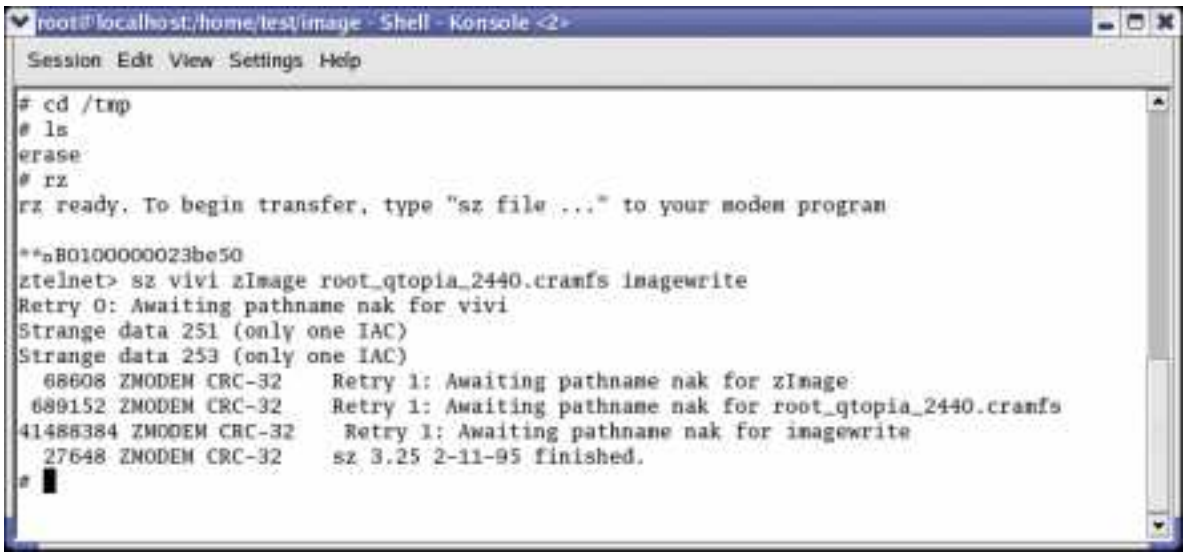

Figure 4-12 Copying Image files to target board using ztelnet

Only **/tmp** directory can be used for both reading and writing, all directories except **/tmp** are read-only file systems. 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.

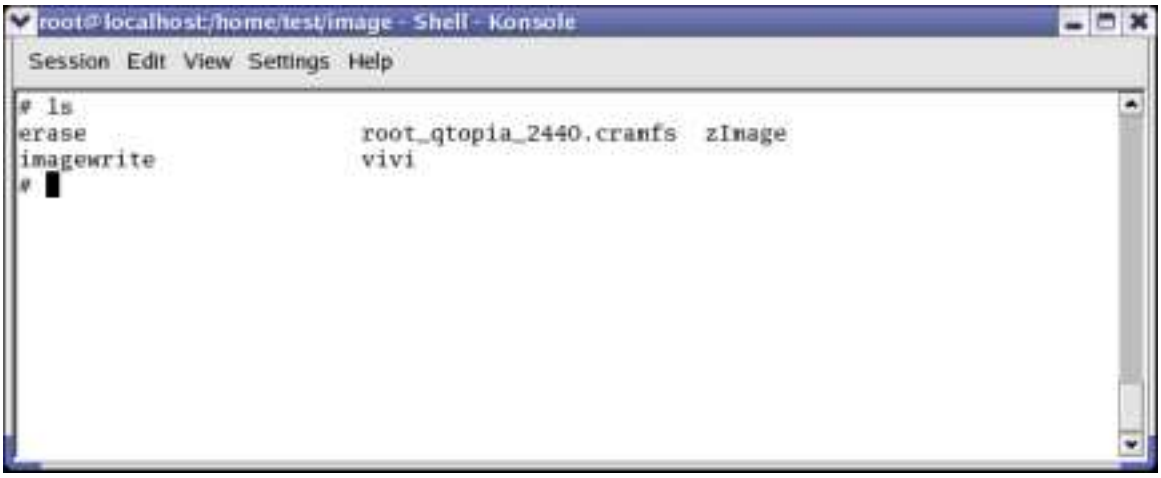

Figure 4-13 Image files on Target Board

4.9 Imagewrite

4.9.1 Creating partitions in SMC

Terminal executing minicom enable host PC user to work inside target board. Now make three partitions.

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

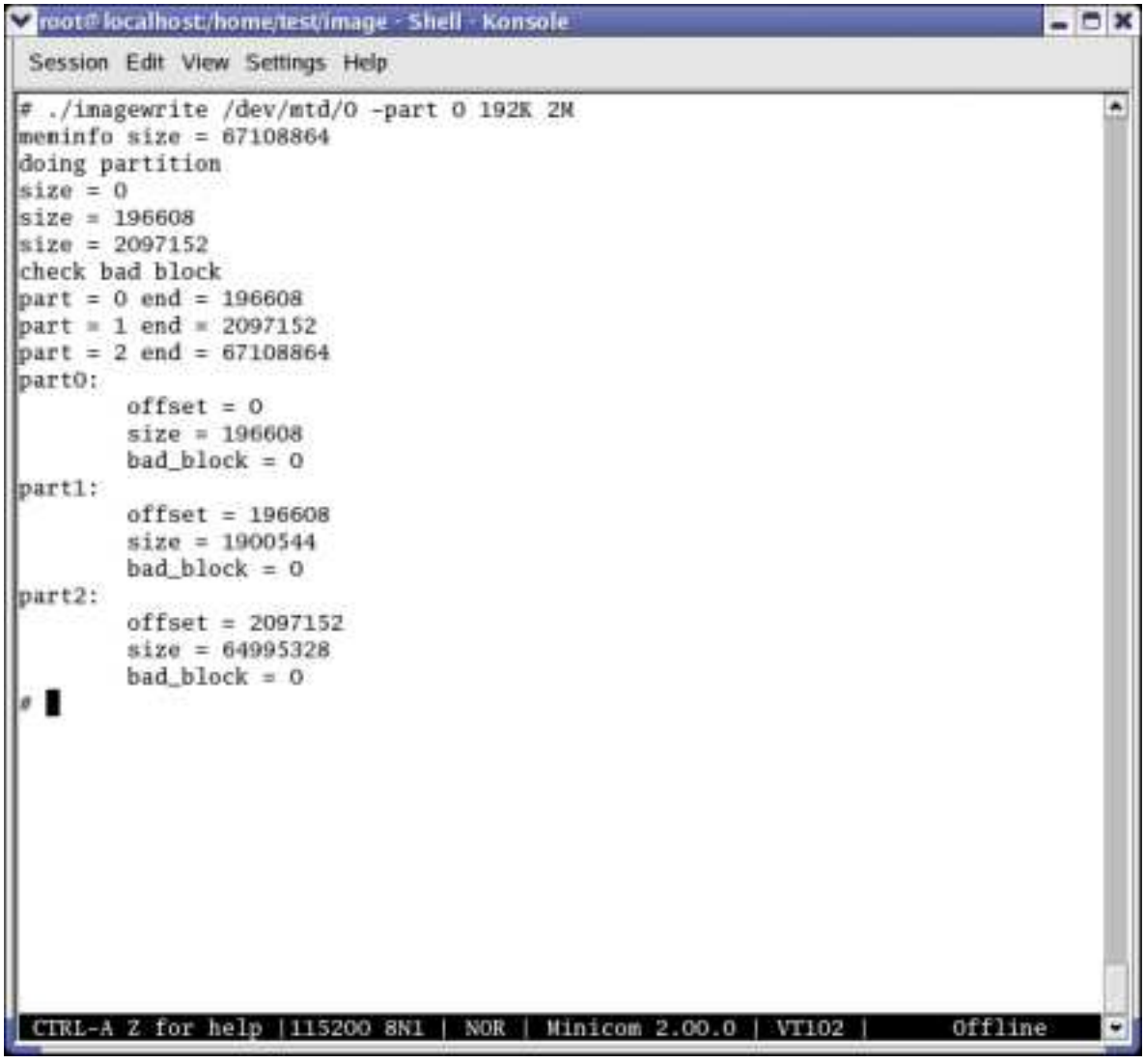

Figure 4-14 Partitioning SMC

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

Vivi Bootloader : 0~192KB
Kernel Image : 192KB~2MB
Root_qtopia_2440.cramfs : 2MB~End-part

write '**vivi**' at '**0~192KB**' partition, '**zImage**' at '**192KB~2MB**' partition, and **root_qtopia_2440.cramfs**' at '**2MB~End-part**' partition.

4.9.2 Copying the Images to SMC by 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_2440.cramfs:2M	: Store root_qtopia_2440.cramfs in SMC.

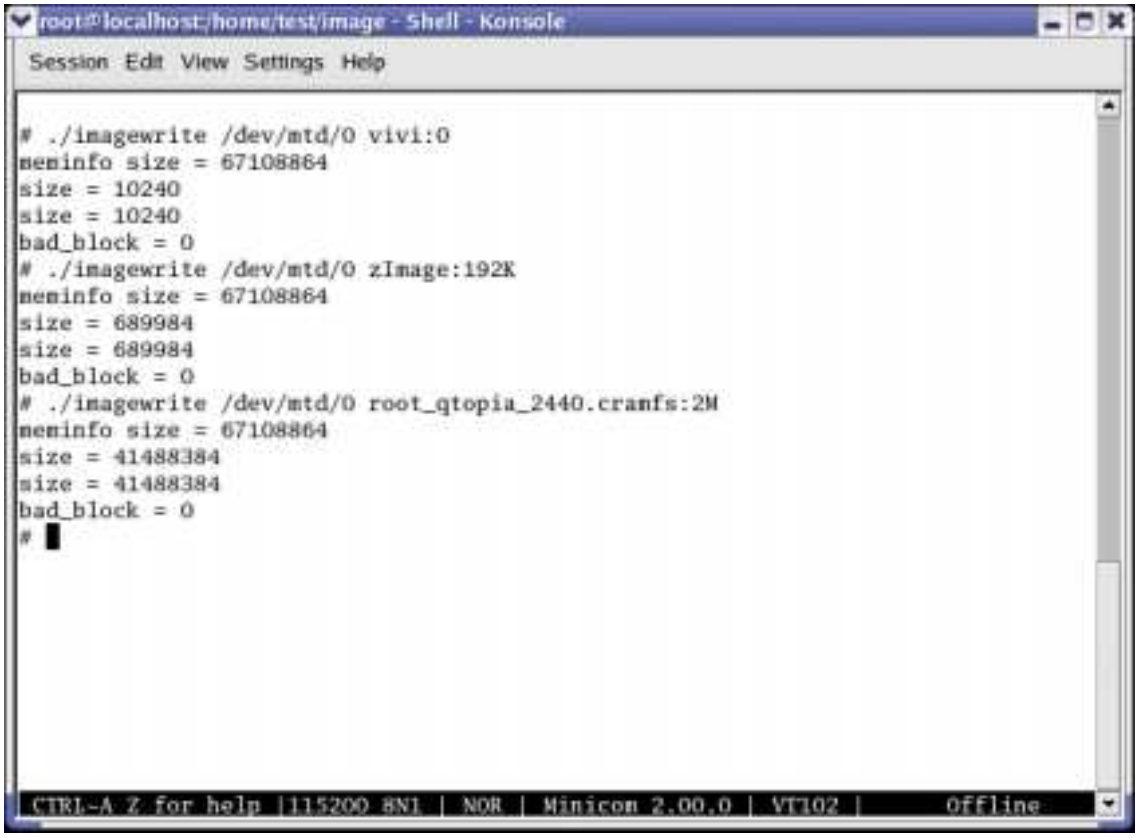

Figure 4-15 Writing Images on SMC

After completing the above procedure please reboot the target board and you can see Linux booting on the target board.

5 Porting Linux when Target Board Booting is disabled

5.1 Uploading 'vivi' using JTAG Cable

JTAG cable and Jflash program are required to port for using this method. Jflash program and HOWTO_USE.txt written about usage of Jflash program are located under **'/Jflash'** directory. Jflash source file is compressed with tarball *'jflash-s3c2440.tar.gz'*. Extract it by executing following command. Go to *'/Jflash'* directory created after extracting the tarball.

```
[root@localhost test]# tar zxvf jflash-s3c2440.tar.gz
# cd /root/jflash
```

When you write **vivi** (bootloader) to **SMC** by **'Jflash program'**, you need to copy the **'Jflash program'** i.e. *'Jflash-s3c2440'* which is under *'/Jflash'* directory, to **/image** directory.

```
[root@localhost Jflash]# cp ./Jflash-s3c2440 /home/test/image
```

Connect target board and host PC by JTAG. Insert the 64 MB SMC card in **'SMC card slot'**. Short pin numbers 2 and 3 of jumper **'J1'**, **'J2'**, **'J3'** and **'J4'** for NAND flash setting (SMC Card settings).

```
[root@localhost Jflash]# ./Jflash-s3c2440 -help
```

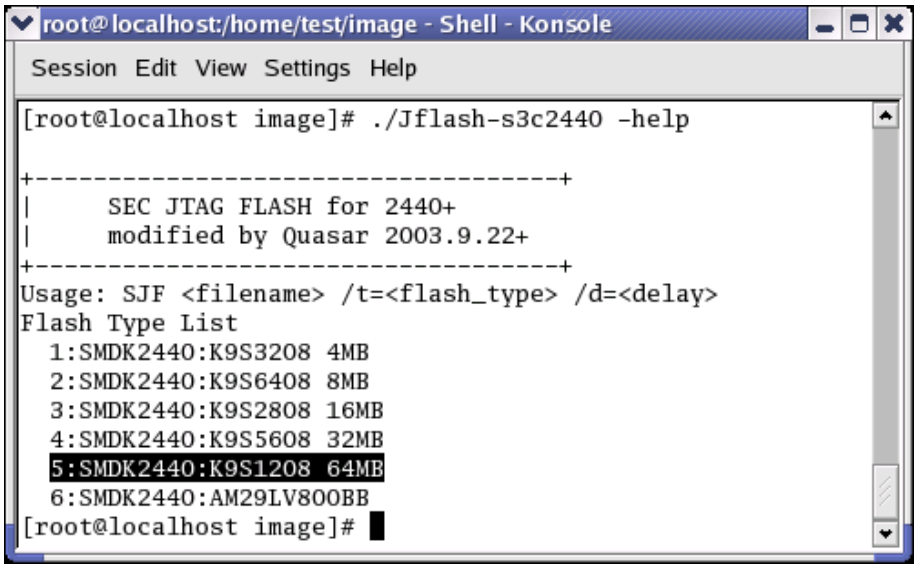

Figure 5-1 Jflash-s3c2440 -help

There are different options after executing Jflash program help, according to the size of flash memory. We use 64MB SMC, so we have to give **'/t=5'** option. Switch ON the target Board and execute Jflash program with **'/t=5'** option. If you use different flash memory, option will be changed.

Go to **/home/test/image** directory and execute the following command.

```
[root@localhost image]# ./Jflash-s3c2440 vivi /t=5
```

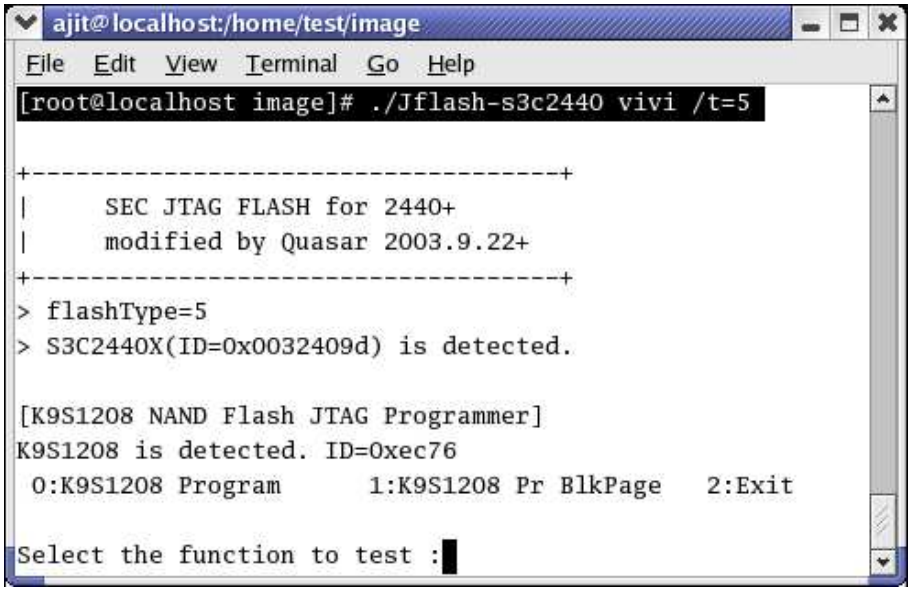

Figure 5-2 selecting function to test Jflash-s3c2440

Please select **'0'** for **'K9S1208 Program'.**

Enter Input target block number as **'0'.**

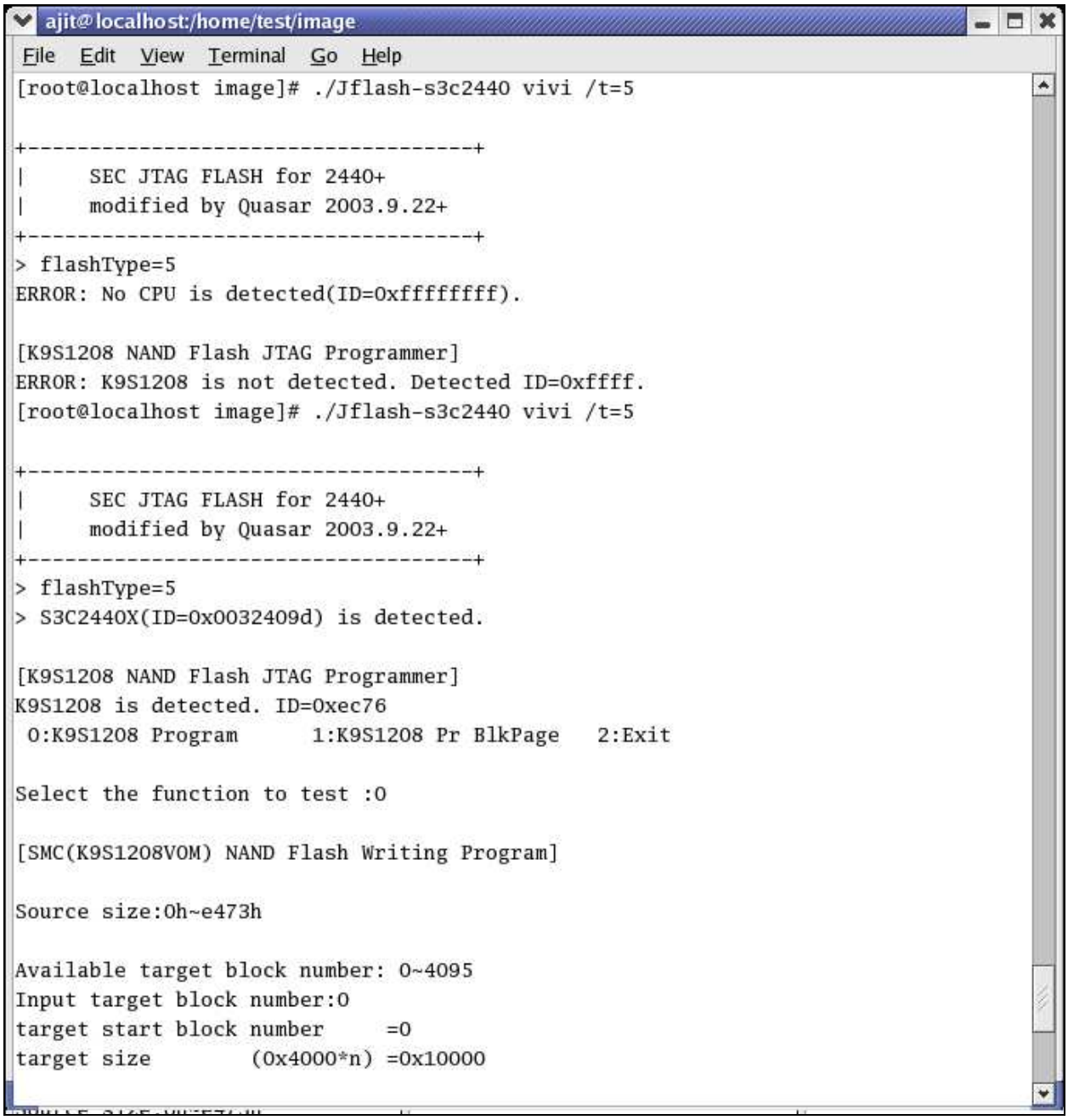

Figure 5-3 inputting target block number

You can see the status of vivi while downloading through JTAG port.

Note: Please wait for some time for the console to appear as shown in figure 4.4. As **vivi** (bootloader) takes long time to write to SMC.

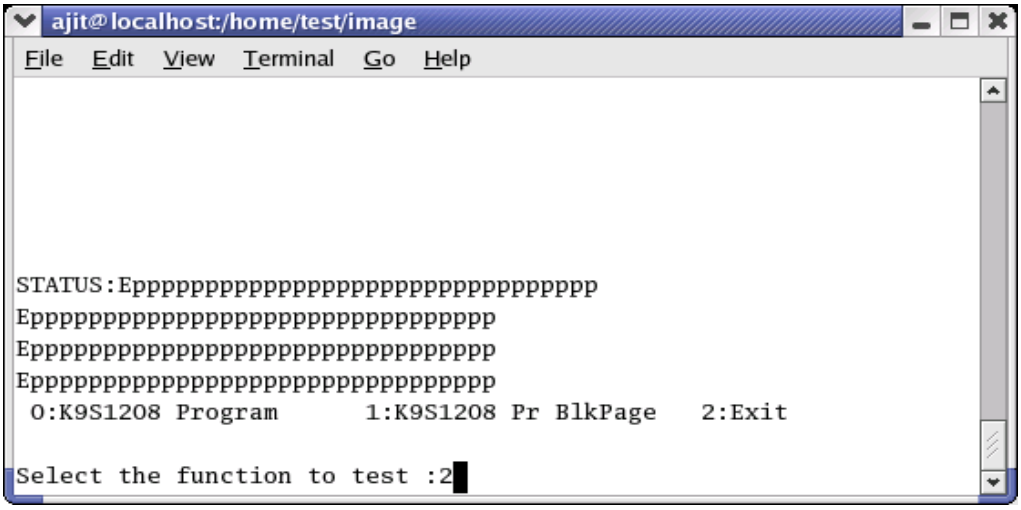

Figure 5-4 Completion of vivi

After **vivi** download completes, please select **'2'** to Exit.

Run the **Minicom** after connecting host to target board by serial cable. 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 **'space-bar'** key, target board enters into **vivi** prompt mode. The waiting time of target board is very short so hit **'space-bar'** key quickly if you want to use target board console.

Figure 5-5 vivi shell prompt

Note 1: To see the **vivi>** prompt, please keep any key pressed continuously before booting the board.

Note 2: 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.

5.2 Uploading Images to the target board using vivi

Once vivi (bootloader) is stored in SMC (NAND flash memory), you can write vivi (bootloader), kernel image, root filesystem 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 1000000 : "1000000" means 1 second because a unit is microsecond.
vivi> param save

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/s3c2440/smdk.c** file as shown in the following fig

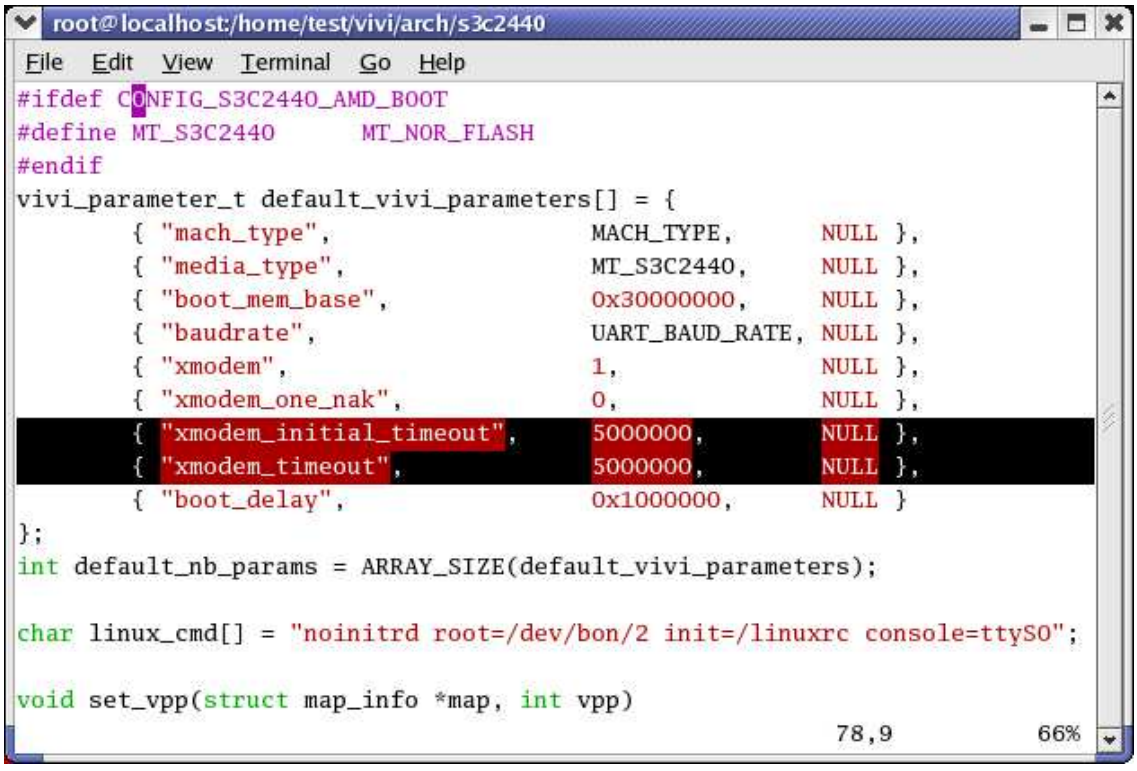

Figure 5-6 xmodem_initial_timeout settings

5.3 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 filesystem and vivi supports this. So you can make partitions through vivi prompt with the help of following command.

vivi> bon part 0 192k 2M

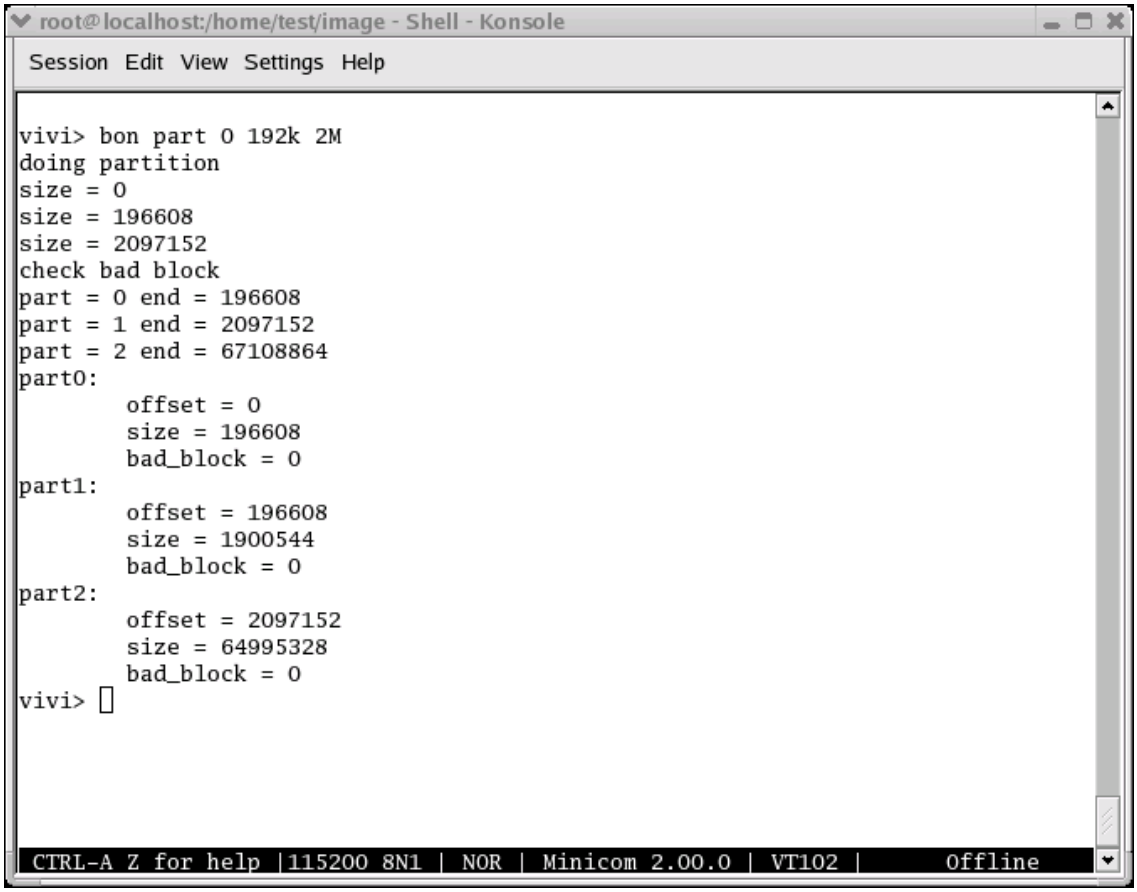

Figure 5-7 Partitioning SMC

The **bon** command makes 3 partitions of sizes **0~192K**, **192K~2M**, and **2M~64M**.

0~192k	**: vivi** (Bootloader) will be written here.
192k~2M	**: zImage** (kernel) will be written here.
2M~64M	**: 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.

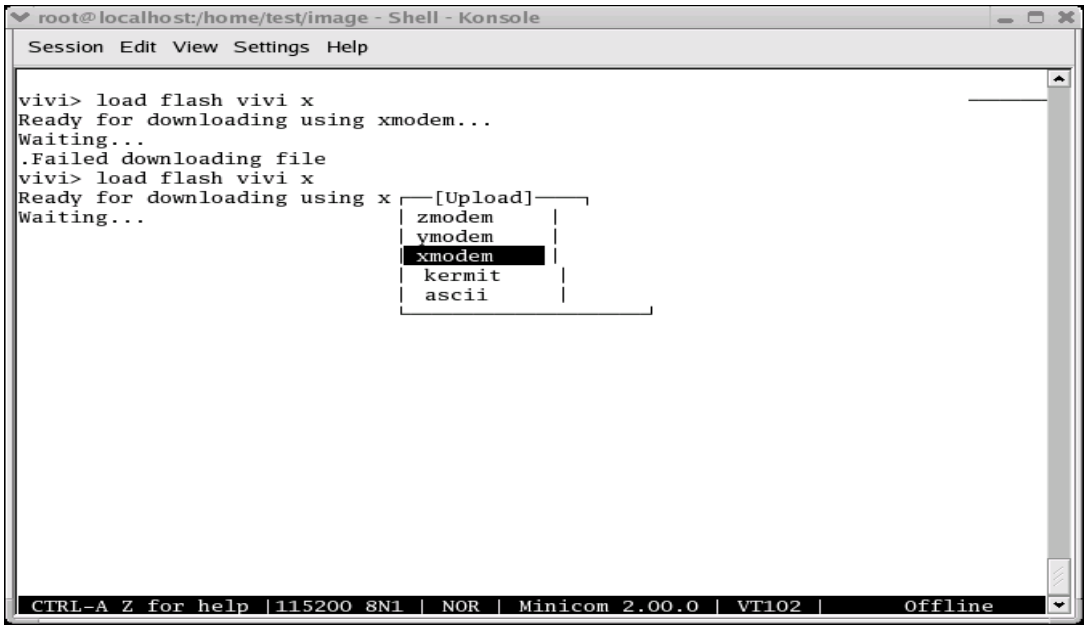

Figure 5-8 xmodem x-fer mode for Vivi

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

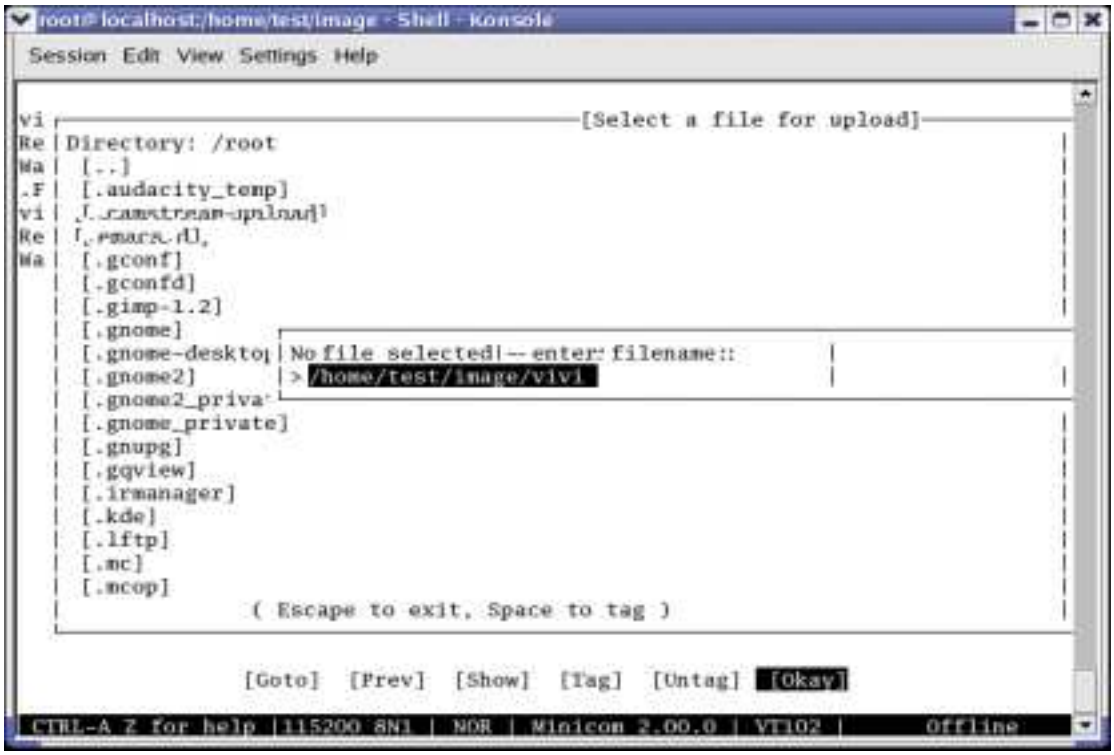

Figure 5-9 Entering filename for vivi

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

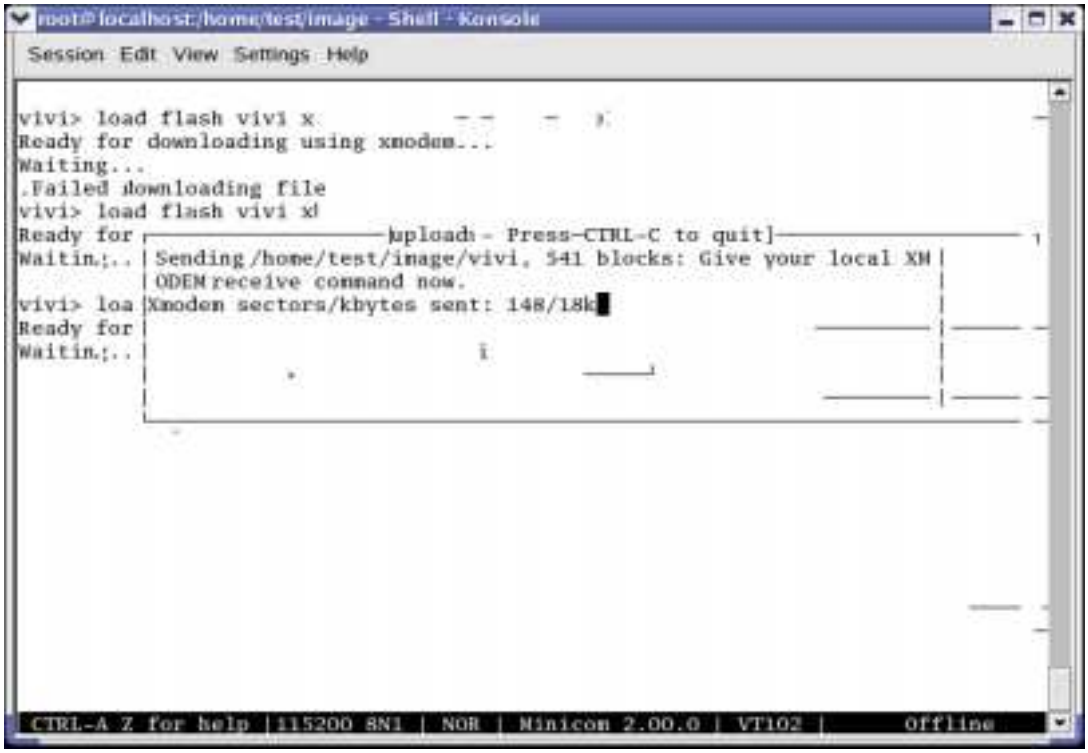

Figure 5-10 vivi download status

After **vivi** bootloader image download completes, hit **Enter** key to come to **vivi** prompt. If you encounter timeout then please try to upload the image again.

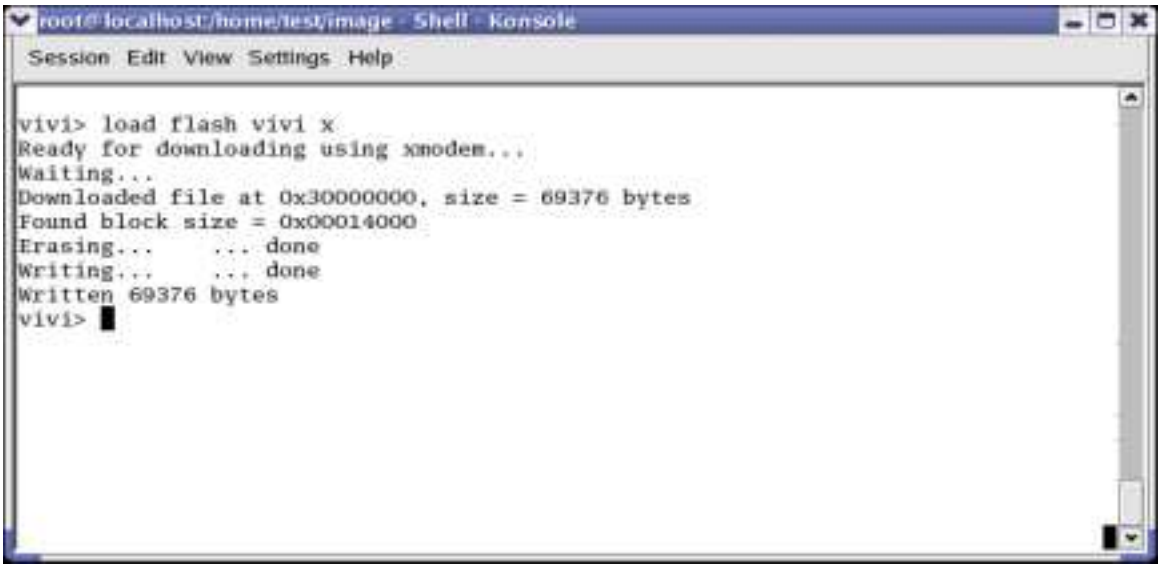

Figure 5-11 vivi Prompt

5.4 Writing Kernel Image

To upload kernel Image please execute the following command.

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

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

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

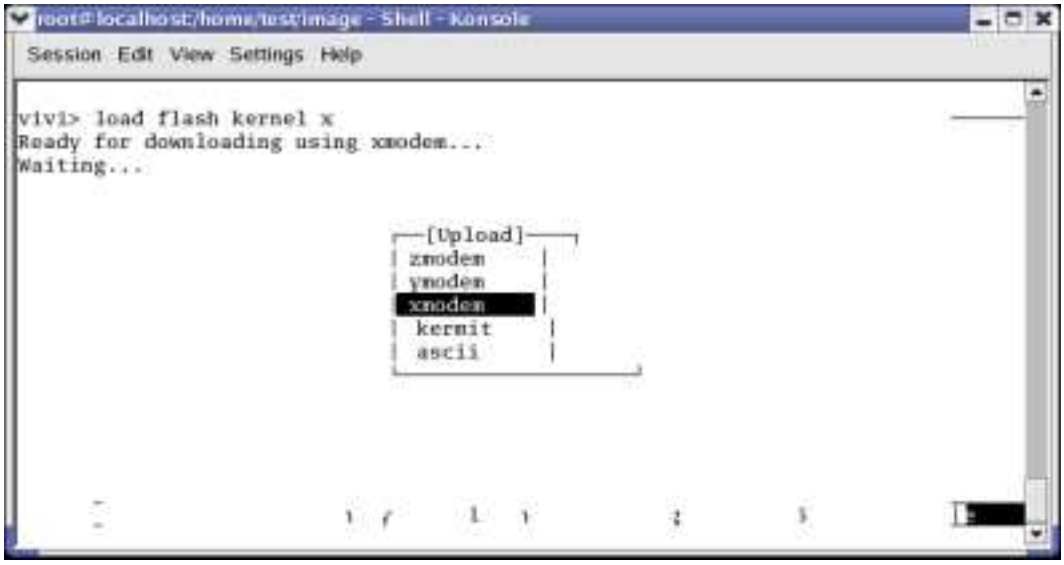

Figure 5-12 xmodem x-fer mode for kernel Image

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

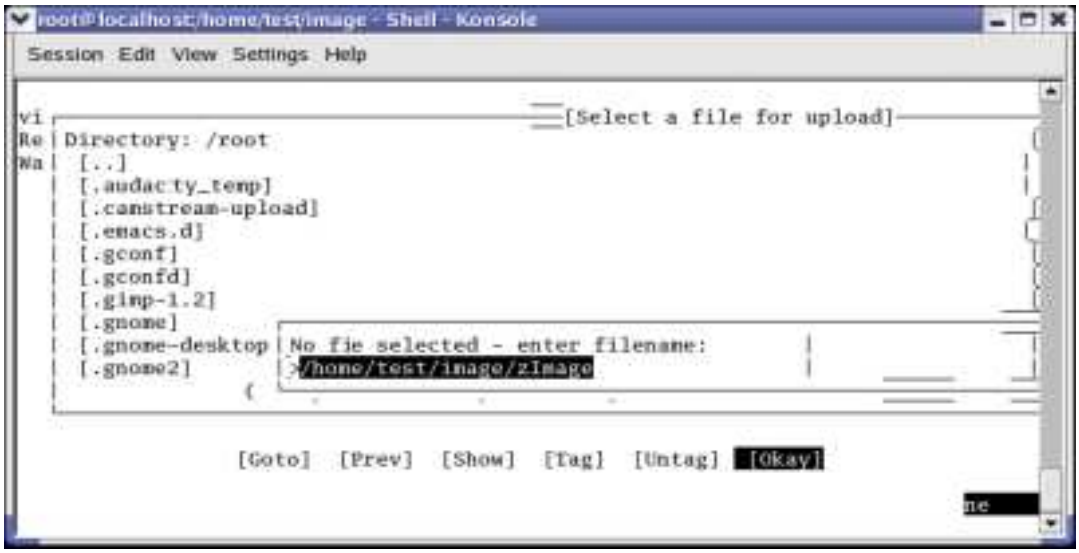

Figure 5-13 Entering filename for zImage

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

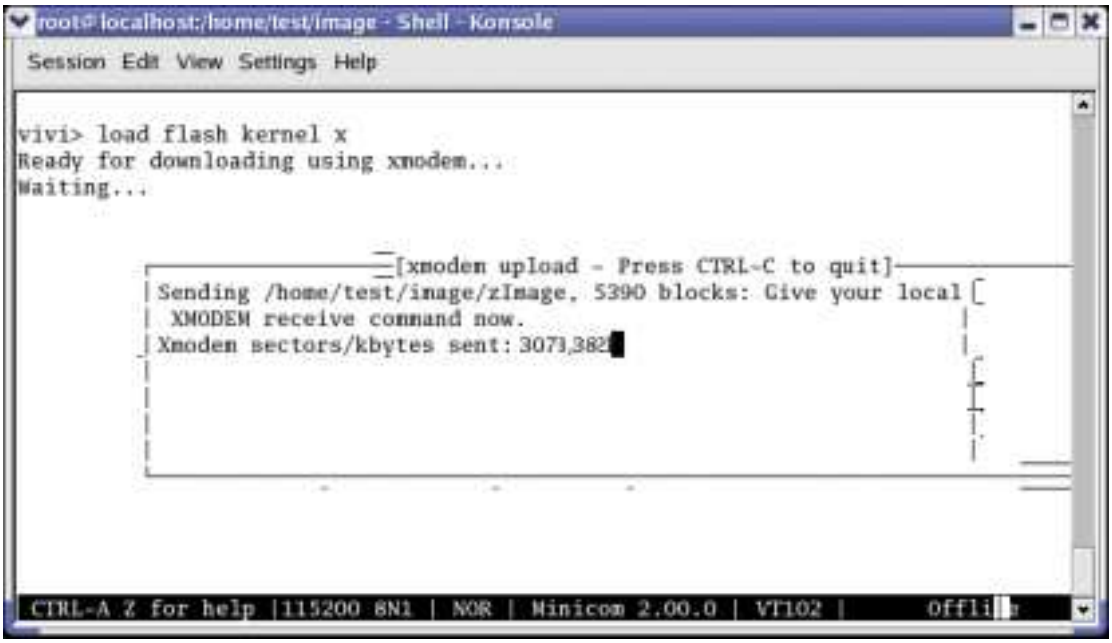

Figure 5-14 zImage download status

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

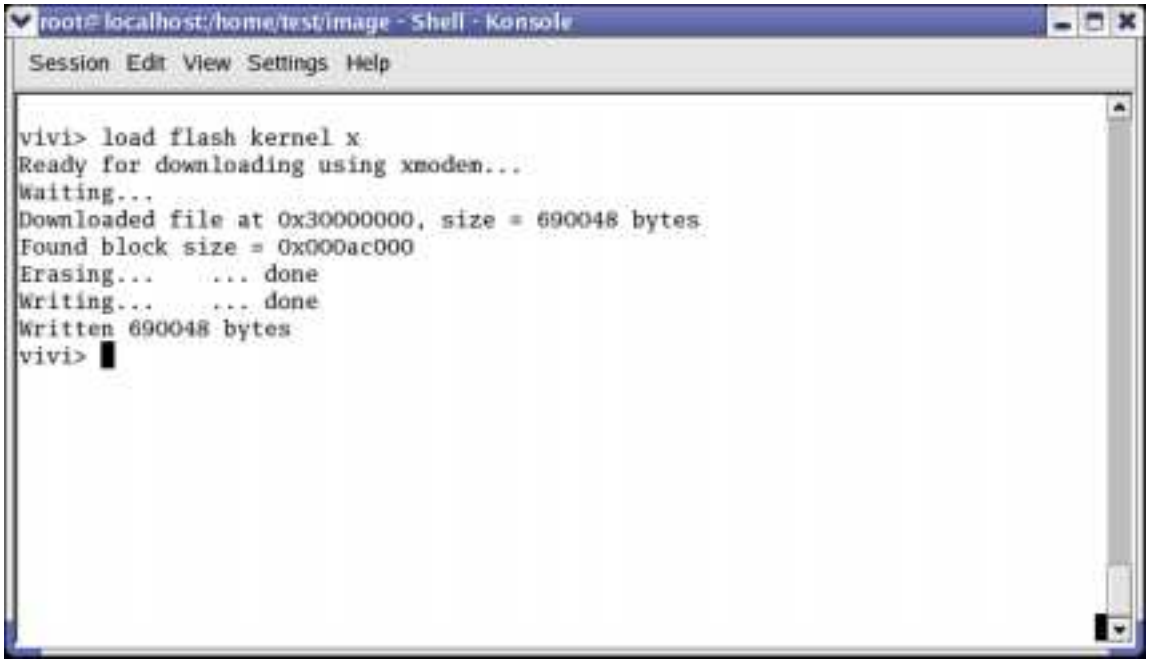

Figure 5-15 vivi Prompt

5.5 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.

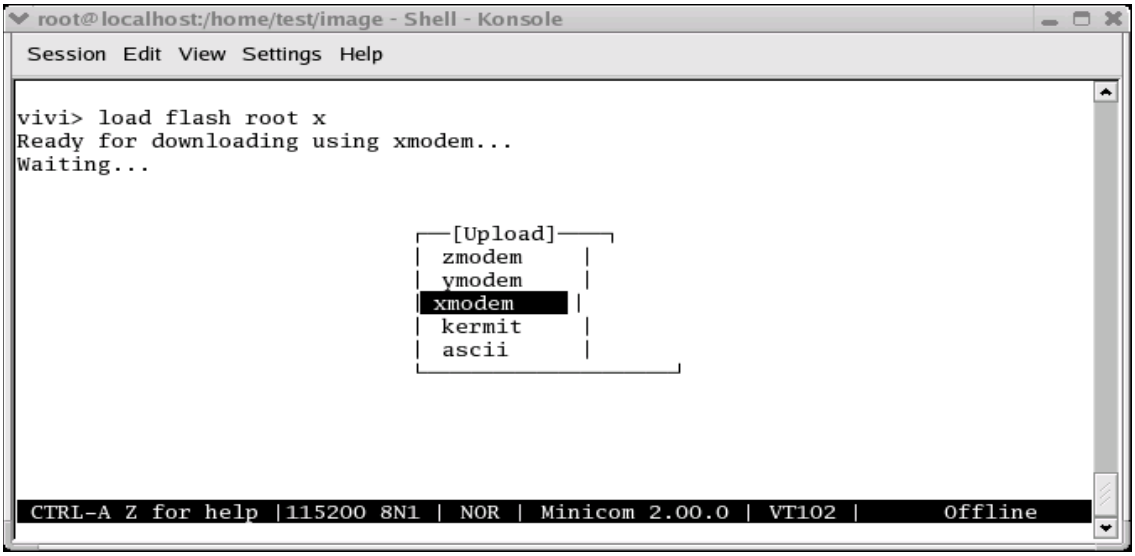

Figure 5-16 xmodem x-fer mode for root.cramfs Image

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

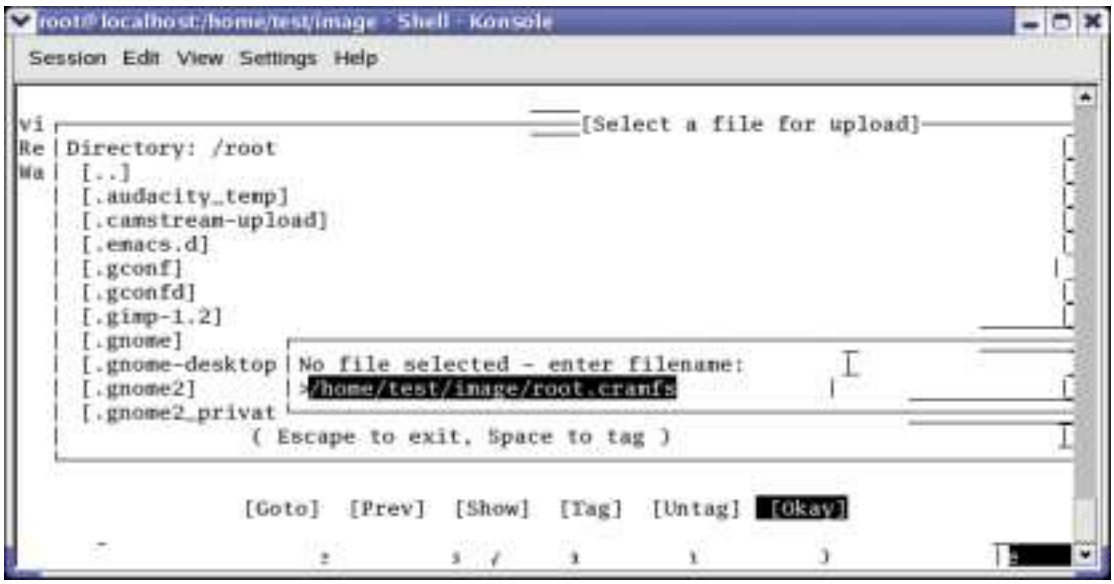

Figure 5-17 Entering filename for root.cramfs

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

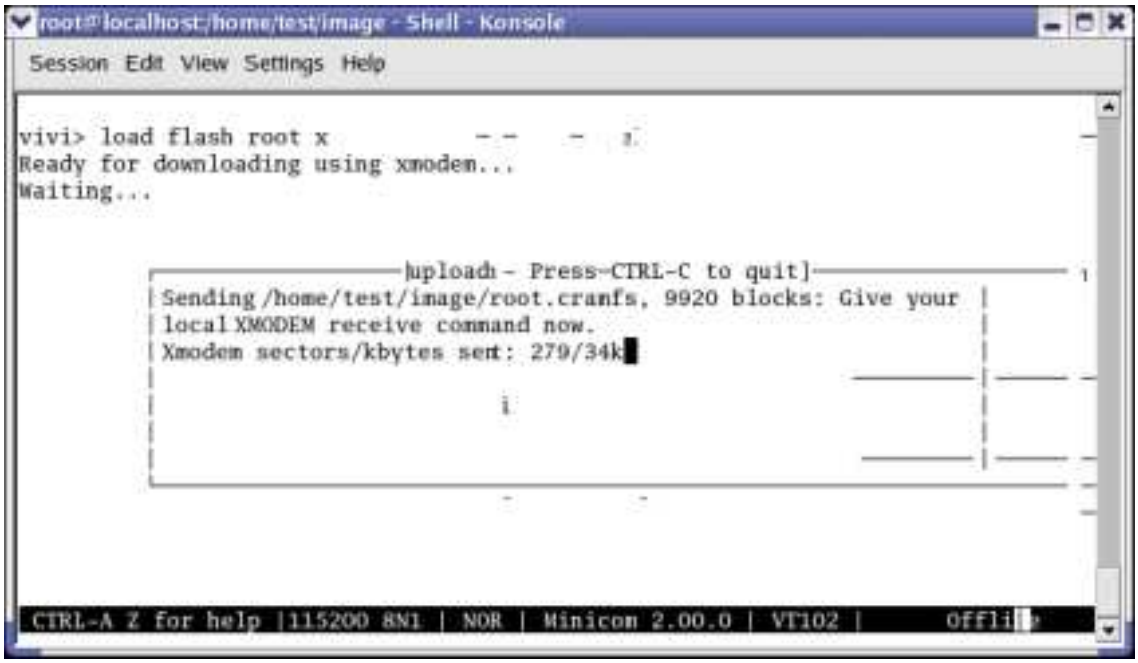

Figure 5-18 root.cramfs download status

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

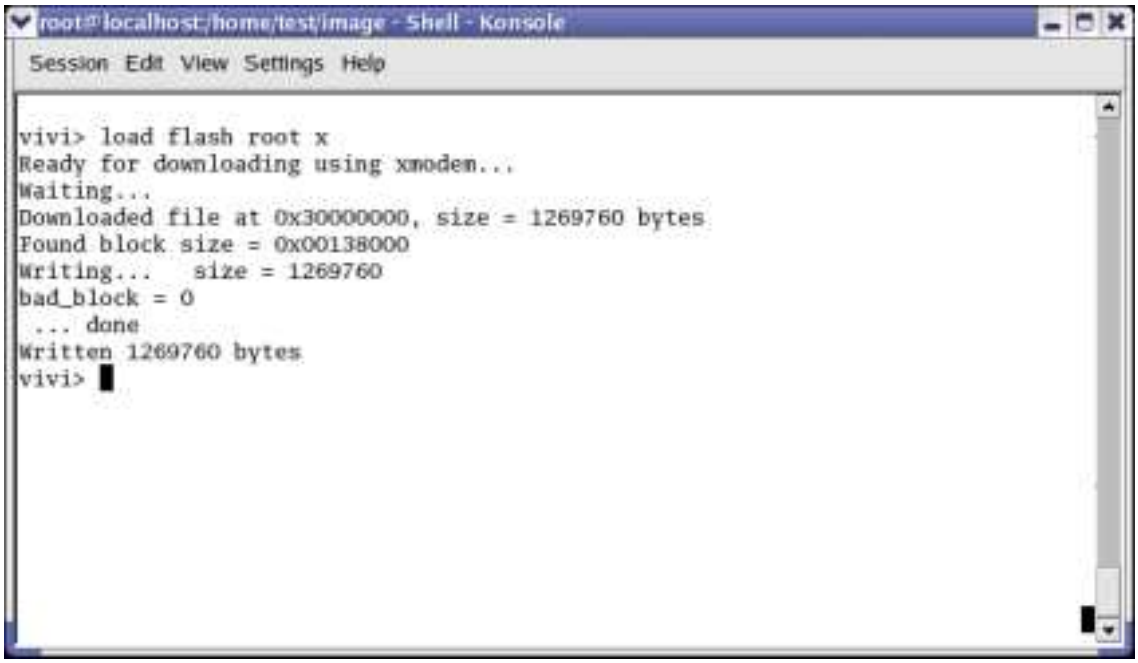

Figure 5-19 vivi Prompt

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.

Figure 5-20 After booting the Target Board

Now you can start downloading all the images to target board through **ztelnet** by referring section **4.3** to session **4.9** and fuse the NAND flash memory.

6 Installing Camera Module

6.1 Compiling Camera Module

Camera Module source file is compressed with tarball *'s3c2440_kernel2.4.18_modules_camera.tar.bz2'*. Extract it by executing following command.

```
[root@localhost test]# tar jxvf s3c2440_kernel2.4.18_modules_camera.tar.bz2
[root@localhost test]# cd s3c2440_kernel2.4.18_modules_camera

[root@localhost s3c2440_kernel2.4.18_modules_camera]# ls
cam2fb.c Makefile README s3c2440_ov7620.c smdk2440_ov7620.h

[root@localhost s3c2440_kernel2.4.18_modules_camera]# arm4l-redhat-linux-gcc
-o cam2fb cam2fb.c

[root@localhost s3c2440_kernel2.4.18_modules_camera]# ls
cam2fb cam2fb.c Makefile README s3c2440_ov7620.c smdk2440_ov7620.h
```

It is recommended to read the *'README'* file. Edit the Kernel directory path in *'Makefile'* under *'s3c2440_kernel2.4.18_modules_camera'* directory as shown below.

Edit **KERNLE_DIR :=/home/test/s3c2440_kernel2.4.18_rel** (it depends on your environment and directory structure)

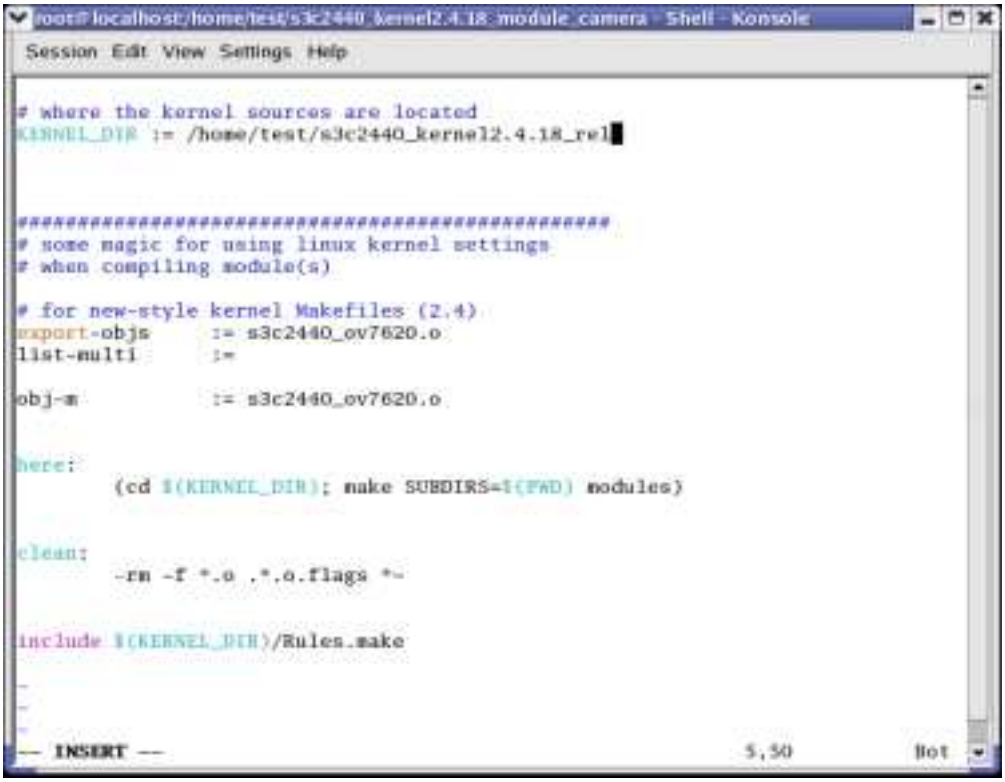

Figure 6-1 Editing Makefile

Execute the *'make'* command as shown below.

```
[root@localhost s3c2440_kernel2.4.18_modules_camera]# make
```

6.2 Porting Camera Module

To port the Camera module please refer to the steps from section 4.1 to section 4.7.

6.3 Transferring Images by ztelnet

Terminal 1:	**# cd /tmp** **# rz** Pushing **'Ctrl +]', 'ztelnet>'** console appears. **ztelnet>** sz cam2fb s3c2440_ov7620.o
Terminal 2:	**# cd /tmp**

```
# ls
# cam2fb erase s3c2440_ov7620.o
# insmod s3c2440_ov7620.o
# ./cam2fb 176 144 16 0 0
```

Index

A

ADC, 1

B

Booting, 12, 16, 22
bootloader, 3, 5, 7, 12, 22, 24, 25, 26, 28, 29
Bootloader, 5, 21, 34

C

camera, 3, 36, 37
Camera Module, 36, 37
Cramfs, 11

D

DMA, 1

E

Embedded, 12

I

IIC-BUS, 1
imagewrite, 7, 12, 19, 20, 21
Imagewrite, 20
IP address, 17

J

Jflash program, 22
Jflash-s3c2440, 22, 23
JTAG, 22, 24

K

kernel, 3, 5, 7, 8, 9, 10, 12, 15, 26, 27, 30, 34
Kernel, 5, 7, 8, 9, 10, 21, 30

L

Linux, 12, 22

M

Minicom, 5, 12, 14, 15, 16, 25, 26
MMC, 3

P

Ping Test, 18
PWM timers, 1

Q

Qtopia, 3

R

RAM, 5
root file system, 3, 7, 11
Root File System, 5, 32, 34
Root filesystem, 11
root_qtopia_2440.cramfs, 3, 11, 12, 19, 21
RTC, 1

S

SDRAM, 1
SMC, 3, 12, 15, 20, 21, 22, 24, 26, 27, 34
SMC partitioning, 27
SMDK 2440, 12, 17

T

Target Board, 12, 16, 18, 19, 22, 35
Toolchain, 3, 4

U

UART, 1
USB, 1

V

Vivi, 5, 6, 21, 28

W

Watch Dog Timer, 1

Z

zImage, 10, 11, 12, 19, 21, 27, 30, 31, 34
ztelnet, 3, 12, 15, 18, 19, 35, 37