

Installation Manual for U-Boot (Linux)





S3C Installation Manual for U-Boot (Linux)

Copyright © 2008 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-Dong, Giheung-Gu,

Yongin-City, Gyeonggi-Do, Korea 449-711

e-mail : mobilesol.cs@samsung.com

Home Page: <http://www.samsungsemi.com>

Revision History

Date	Version	Author	Amendment
2007-02-14	1.0	Jinsung Yang	Master Copy
2007-03-23	1.5	Kim, Yong Hun	Add on SMDK6400
2007-03-27	1.6	Kim, Yong Hun	Arrange common command
2007-04-26	2.0	Jinsung Yang	Update
2008-03-18	2.5	SungJun Bae	Add on SMDK6410
2008-03-20	2.6	JongPill Lee	Add on SMDK2450
2008-07-25	2.7	JonPill Lee	Add on SMDK6430

Contents

1	INTRODUCTION	8
1.1	OVERVIEW.....	8
2	U-BOOT COMPILATION	9
2.1	PREPARATION.....	9
2.2	INSTALLING TOOLCHAIN.....	9
2.3	U-BOOT COMPILATION.....	10
3	CONFIGURATION DETAILS.....	13
3.1	CONFIGURATION, SETUP.....	13
3.2	BOOT SEQUENCE.....	14
3.3	CS8900 SUPPORT.....	14
3.3.1	SMDK2412	14
3.3.2	SMDK2443	15
3.3.3	SMDK2450	15
3.3.4	SMDK6400	16
3.3.5	SMDK6410	16
3.3.6	SMDK6430	16
3.4	NAND FLASH SUPPORT	16
3.5	SMDK2412 BOARD VERSION SELECTION	17
3.6	SYNC/ASYNCR MODE SELECTION ON SMDK6400.....	17
3.7	SYNC/ASYNCR MODE SELECTION ON SMDK6410(OR SMDK6430).....	18
4	WRITE U-BOOT TO TARGET BOARD THROUGH TRACE32 ICD.....	19
4.1	INTRODUCTION	19
4.2	BOARD CONFIGURATION	20
4.2.1	SMDK2412	20
4.2.2	SMDK2443	21
4.2.3	SMDK2450	22
4.2.4	SMDK6400	23
4.2.5	SMDK6410(or SMDK6430).....	24
4.3	BURNING U-BOOT SCRIPT FOR TRACE32 ICD	25
4.3.1	SMDK2412	25
4.3.2	SMDK2443	29

4.3.3	SMDK2450	34
4.3.4	SMDK6400	39
4.3.5	SMDK6410(or SMDK6430).....	44
5	WRITE IMAGES TO SMDK TARGET BOARD	50
5.1	INTRODUCTION	50
5.2	MINICOM.....	50
5.3	TFTP SERVER	54
5.4	SETTING IP ADDRESS.....	56
5.4.1	Setting IP Address on Host PC.....	56
5.4.2	Setting IP Address for SMDK Target Board.....	58
5.5	TRANSFER AND WRITE IMAGES USING TFTP.....	65
5.5.1	Transfer and Write "u-boot.bin" (bootloader)	67
5.5.2	Transfer and Write "zImage" (Kernel Image).....	69
5.5.3	Transfer and Write "Qtopia1.7_demo_image.cramfs" (Root File System)	71
5.5.4	Transfer and write image on extra area for JFFS2 and YAFFS2	73

Figures

FIGURE 2-1 MAKEFILE OF S3C-U-BOOT-1.1.6	11
FIGURE 3-1 BOOT SEQUENCE	14
FIGURE 4-1 JUMPER CONFIGURATION FOR SMDK2412	20
FIGURE 4-2 JUMPER CONFIGURATION FOR SMDK2443	21
FIGURE 5-1 MINICOM SETUP - 1	51
FIGURE 5-2 MINICOM SETUP - 2	51
FIGURE 5-3 MINICOM SETUP - 3	52
FIGURE 5-4 MINICOM SETUP - 4	52
FIGURE 5-5 MINICOM SETUP - 5	53
FIGURE 5-6 MINICOM SETUP - 6	53
FIGURE 5-7 MINICOM SETUP - 7	54
FIGURE 5-8 SETTINGS FOR SYSTEM SERVICES - 1	55
FIGURE 5-9 SETTINGS FOR SYSTEM SERVICES - 2	55
FIGURE 5-10 SETTING IP ADDRESS ON HOST PC	57
FIGURE 5-11 PARAMETERS IN SMDK2412	59
FIGURE 5-12 PARAMETERS IN SMDK2443	60
FIGURE 5-13 PARAMETERS IN SMDK2450	61
FIGURE 5-14 PARAMETERS IN SMDK6400	62
FIGURE 5-15 PARAMETERS IN SMDK6410	63
FIGURE 5-16 SET AND SAVE PARAMETERS	65
FIGURE 5-16 TRANSFER AND WRITE 'U-BOOT.BIN'	68
FIGURE 5-17 TRANSFER AND WRITE ZIMAGE	70
FIGURE 5-18 TRANSFER AND WRITE ROOT FILE SYSTEM IMAGE	72

Tables

TABLE 2-1 PREPARATION.....	9
TABLE 2-2 BOARD CONFIGURATIONS.....	12
TABLE 3-1 CONFIGURATION HEADER FILES.....	13
TABLE 5-1 COMMAND PROMPTS IN U-BOOT.....	58
TABLE 5-2 PARTITION ASSIGNMENT.....	66

1 Introduction

In this Chapter, you will understand the following:

- Section 1.1, "Overview"

1.1 Overview

"The "U-Boot" Universal Bootloader project provides firmware with full source code under GPL. Many CPU architectures are supported: PowerPC(MPC5xx, MPC8xx, MPC82xx, MPC7xx, MPC74xx, 4xx), ARM(ARM7, ARM9, StrongARM, Xscale), MIPS(4Kc,5Kc), x86, ..."

U-boot supports many useful features including tftp downloading, nand flash interface, environmental variable support, so it is useful for embedded system CPUs like S3C2412, S3C2443, S3C2450, S3C6400 and S3C6410(OR S3C6430).

2 U-Boot Compilation

In this Chapter, you will understand the following:

- Section 2.1, "Preparation"
- Section 2.2, "U-Boot Compilation"

2.1 Preparation

The following source file is required for U-Boot installation. Copy all the listed files to the working directory **/home/test**.

File Name	Description
cross-4.2.2-eabi.tar.bz2	Toolchain 4.2.2-eabi
s3c-u-boot-1.1.6.tar.bz2	Boot-loader

Table 2-1 Preparation

Below is the list of files are downloaded on the host PC.

```
[root@localhost test]# ls
cross-4.2.2-eabi.tar.bz2
s3c-u-boot-1.1.6.tar.bz2
```

2.2 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 toolchain in the directory mentioned below:

```
[root@localhost test]# mkdir -p /usr/local/arm  
[root@localhost test]# tar jxvf cross-4.2.2-eabi.tar.bz2
```

The above command will generate the **"4.2.2-eabi"** folder under the **"test/"** directory. Copy this folder under **"/usr/local/arm/"** directory.

```
[root@localhost test]# mv 4.2.2 /usr/local/arm/  
[root@localhost test]# export PATH=$PATH:/usr/local/arm/4.2.2-eabi/usr/bin
```

The toolchain object files such as arm compilers, loaders etc. will be available in the **"/usr/local/arm/4.2.2-eabi/usr/bin"** directory.

2.3 U-Boot Compilation

To start compiling U-boot, the source file is compressed with tarball as **"s3c-u-boot-1.1.6.tar.bz2"** you can extract the file by executing following command.

```
[root@localhost test]# tar jxvf s3c-u-boot-1.1.6.tar.bz2
```

Go to **"s3c-u-boot-1.1.6"** directory created after extracting the tar ball and then execute the **"vi Makefile"** command to edit the **"Makefile"** for U-Boot bootloader as shown.

```
[root@localhost test]# cd s3c-u-boot-1.1.6  
[root@localhost s3c-u-boot-1.1.6]# vim Makefile
```

CROSS_COMPILE = /usr/local/arm/4.2.2-eabi/usr/bin/arm-linux-

```

ifeq ($(OBJTREE)/include/config.mk,$(wildcard $(OBJTREE)/include/config.mk))
# load ARCH, BOARD, and CPU configuration
include $(OBJTREE)/include/config.mk
export ARCH CPU BOARD VENDOR SOC

ifndef CROSS_COMPILE
ifeq ($(HOSTARCH),ppc)
CROSS_COMPILE =
else
ifeq ($(ARCH),ppc)
CROSS_COMPILE = powerpc-linux-
endif
ifeq ($(ARCH),arm)
CROSS_COMPILE = arm-linux-
endif
ifeq ($(ARCH),i386)
ifeq ($(HOSTARCH),i386)
CROSS_COMPILE =
else
CROSS_COMPILE = i386-linux-
endif
endif
ifeq ($(ARCH),mips)
CROSS_COMPILE = mips_4kc-
endif
ifeq ($(ARCH),nios)
CROSS_COMPILE = nios-elf-
endif
ifeq ($(ARCH),nios2)
CROSS_COMPILE = nios2-elf-
endif
ifeq ($(ARCH),m68k)
CROSS_COMPILE = m68k-elf-
endif
ifeq ($(ARCH),microblaze)
CROSS_COMPILE = mb-
endif
ifeq ($(ARCH),blackfin)
CROSS_COMPILE = bfin-elf-
endif
ifeq ($(ARCH),avr32)
CROSS_COMPILE = avr32-
endif
endif
endif

CROSS_COMPILE = /usr/local/arm/4.2.2-eabi/usr/bin/arm-linux-
export CROSS_COMPILE

```

Figure 2-1 Makefile of s3c-u-boot-1.1.6

Now execute the configuration loading command.

SMDK evaluation board	Command
SMDK2412	[root@localhost s3c-u-boot-1.1.6]# make smdk2412_config

SMDK2443	<code>[root@localhost s3c-u-boot-1.1.6]# make smdk2443_config</code>
SMDK2450	<code>[root@localhost s3c-u-boot-1.1.6]# make smdk2450_config</code>
SMDK6400	<code>[root@localhost s3c-u-boot-1.1.6]# make smdk6400_config</code>
SMDK6410	<code>[root@localhost s3c-u-boot-1.1.6]# make smdk6410_config</code>
SMDK6430	<code>[root@localhost s3c-u-boot-1.1.6]# make smdk6430_config</code>

Table 2-2 Board configurations

Finally compile U-Boot by executing “**make**” command.

```
[root@localhost s3c-u-boot-1.1.6]# make
```

If the compilation of U-Boot progresses well, U-Boot binary image file will be created under “**s3c-u-boot-1.1.6**” directory.

To port binary image file to target board, run tftp server service on your computer. U-boot.bin image will be automatically copied manually to /tftpboot directory.

```
[root@localhost s3c-u-boot-1.1.6]# cp u-boot.bin /tftpboot/
```

After U-Boot image is made as “**u-boot.bin**” in the directory of “**/tftpboot**”, then it can be downloaded to board and written to NAND flash memory.

3 Configuration details

3.1 Configuration, setup

Configuration file is in "include/configs/" directory. Here you can set configuration of U-Boot for SMDK evaluation board.

SMDK evaluation board	Configuration header file
SMDK2412	include/configs/smdk2412.h
SMDK2443	include/configs/smdk2443.h
SMDK2450	include/configs/smdk2450.h
SMDK6400	include/configs/smdk6400.h
SMDK6410	include/configs/smdk6410.h
SMDK6430	include/configs/smdk6430.h

Table 3-1 Configuration header files

Boot options(*common*) such as IP address, boot delay, are set as below:

```
#define CONFIG_BOOTDELAY      3
/*#define CONFIG_BOOTARGS     "root=ramfs devfs=mount console=ttySA0,9600" */
#define CONFIG_ETHADDR       00:40:5c:26:0a:5b
#define CONFIG_NETMASK       255.255.255.0
#define CONFIG_IPADDR        192.168.0.20
#define CONFIG_SERVERIP      192.168.0.10
```

3.2 Boot sequence

Main sequence is as below:

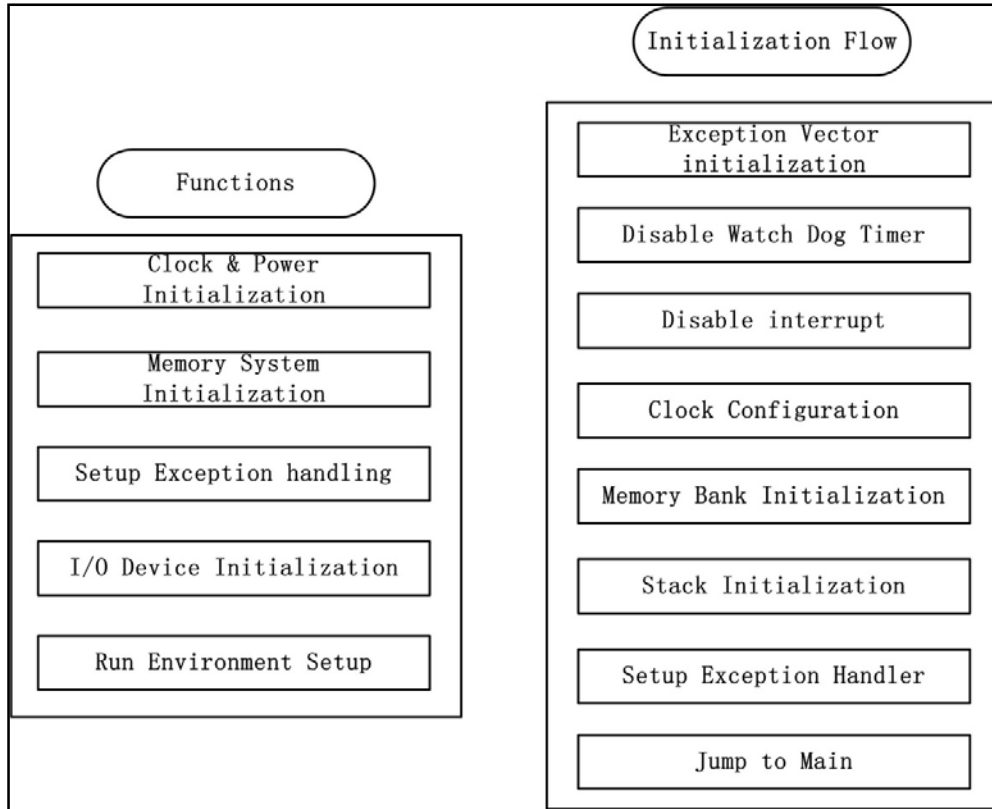


Figure 3-1 Boot Sequence

3.3 CS8900 support

In the configuration file "include/configs/", CS8900 definition is needed.

3.3.1 SMDK2412

```

#define CONFIG_DRIVER_CS8900 1 /* we have a CS8900 on-board */

#ifdef CONFIG_SMDK2412_V14
  
```

```
#define CS8900_BASE      (0x29000300)

#define CS8900_BUS16    1 /* the Linux driver does accesses as shorts */

#else

#define CS8900_BASE      (0x2a000600)

#define CS8900_BUS16    1 /* the Linux driver does accesses as shorts */

#endif /* CONFIG_SMDK2412_V14 */
```

CS8900 is connected to the ROM bank 5.

3.3.2 SMDK2443

```
#define CONFIG_DRIVER_CS8900 1 /* we have a CS8900 on-board */

#define CS8900_BASE      (0x09000300)

#define CS8900_BUS16    1 /* the Linux driver does accesses as shorts */
```

CS8900 is connected to the ROM bank 1.

3.3.3 SMDK2450

```
#define CONFIG_DRIVER_CS8900 1 /* we have a CS8900 on-board */

#define CS8900_BASE      (0x09000300)

#define CS8900_BUS16    1 /* the Linux driver does accesses as shorts */
```

CS8900 is connected to the ROM bank 1.

3.3.4 SMDK6400

```
#define CONFIG_DRIVER_CS8900 1 /* we have a CS8900 on-board */
#define CS8900_BASE 0x18800300
#define CS8900_BUS16 1 /* the Linux driver does accesses as shorts */
```

CS8900 is connected to the ROM bank 1.

3.3.5 SMDK6410

```
#define CONFIG_DRIVER_CS8900 1 /* we have a CS8900 on-board */
#define CS8900_BASE 0x18800300
#define CS8900_BUS16 1 /* the Linux driver does accesses as shorts */
```

CS8900 is connected to the ROM bank 1.

3.3.6 SMDK6430

```
#define CONFIG_DRIVER_CS8900 1 /* we have a CS8900 on-board */
#define CS8900_BASE 0x18800300
#define CS8900_BUS16 1 /* the Linux driver does accesses as shorts */
```

CS8900 is connected to the ROM bank 1.

3.4 Nand Flash support

There are commands for Nand erase, read and write, which are **nand** command. The implementation of these utilities is in `common/cmd_nand.c`.

To add these command following setup is needed at configuration file in "include/configs/".

```
#define CONFIG_COMMANDS W
                (CONFIG_CMD_DFL | W
```



```
CFG_CMD_CACHE | W
CFG_CMD_NAND | W
```

3.5 OneNAND boot-up support

For OneNAND boot-up, you shall edit *include/configs/smdk6410.h* as follows :

3.5.1 SMDK6410

```
//#define CONFIG_BOOT_NAND
#define CONFIG_BOOT_ONENAND
//#define CONFIG_NAND
#define CONFIG_ONENAND
```

3.6 SMDK2412 board version selection

If you use upper version 1.4 of SMDK2412, you have to enabling definition for SMDK2412 ver. 1.4 in "include/configs/smdk2412.h"

```
#define CONFIG_SMDK2412_V14 1 /* SMDK2412 Board version above 1.4 */
```

3.7 SYNC/ASYNC mode selection on SMDK6400

S3C6400 has APLL and MPLL, so you can use that MPLL is source for HCLK, PCLK, etc.(ASYNC mode). In default configuration setting is SYNC mode APLL support main clock for FCLK, HCLK, PCLK, etc. If you want to use ASYNC mode, you have to disabling definition for SMDK6400 SYNC mode selection in "include/configs/smdk6400.h"

```
#define CONFIG_SYNC_MODE
```

(SYNC mode is only supported under FCLK 533 Mhz. If you want to use FCLK 667 Mhz, you must use ASYNC mode)

3.8 SYNC/ASYNC mode selection on SMDK6410(or SMDK6430)

S3C6410(OR S3C6430) has APLL and MPLL, so you can use that MPLL is source for HCLK, PCLK, etc.(ASYNC mode). In default configuration setting is SYNC mode APLL support main clock for FCLK, HCLK, PCLK, etc. If you want to use ASYNC mode, you have to disabling definition for SMDK6410 SYNC mode selection in "include/configs/smdk6410.h"

```
#define CONFIG_SYNC_MODE
```

(SYNC mode is only supported under FCLK 533 MHz. If you want to use FCLK 667 MHz, you must use ASYNC mode)

4 Write U-Boot to target board through Trace32 ICD

In this Chapter, you will understand the following:

- Section 4.1, "Introduction"
- Section 4.2, "Installation on linux"
- Section 4.3, "Board configuration"
- Section 4.4, "Burning Script"
- Section 4.5, "Burning U-Boot to target board"

4.1 Introduction

This Installation Guide describes the basic installation and configuration for TRACE32-ICD In-Circuit Debuggers that are implemented using S3C2412, S3C2443, S3C6400 and S3C6410(OR S3C6430) debug interface. Typical application for S3C2412, S3C2443, S3C2450, S3C6400 and S3C6410(OR S3C6430) debug interface is JTAG.

For the installation and configuration of the ROM monitors or for special system configurations (e.g. additional devices, multiprocessor debugging etc.) refer to the **Installation Guide**.

4.2 Board configuration

The configuration for SMDK are depends on which device will the board boot up, and the description for the configuration are as follows:

4.2.1 SMDK2412

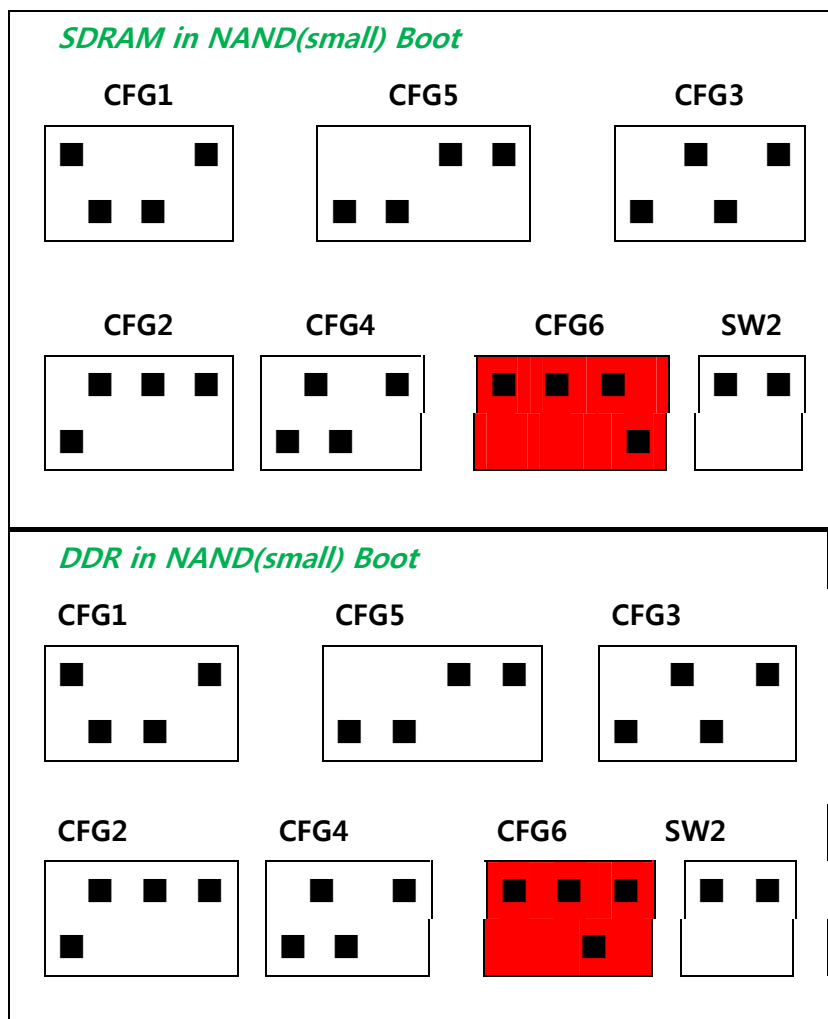


Figure 4-1 Jumper Configuration for SMDK2412

4.2.2 SMDK2443

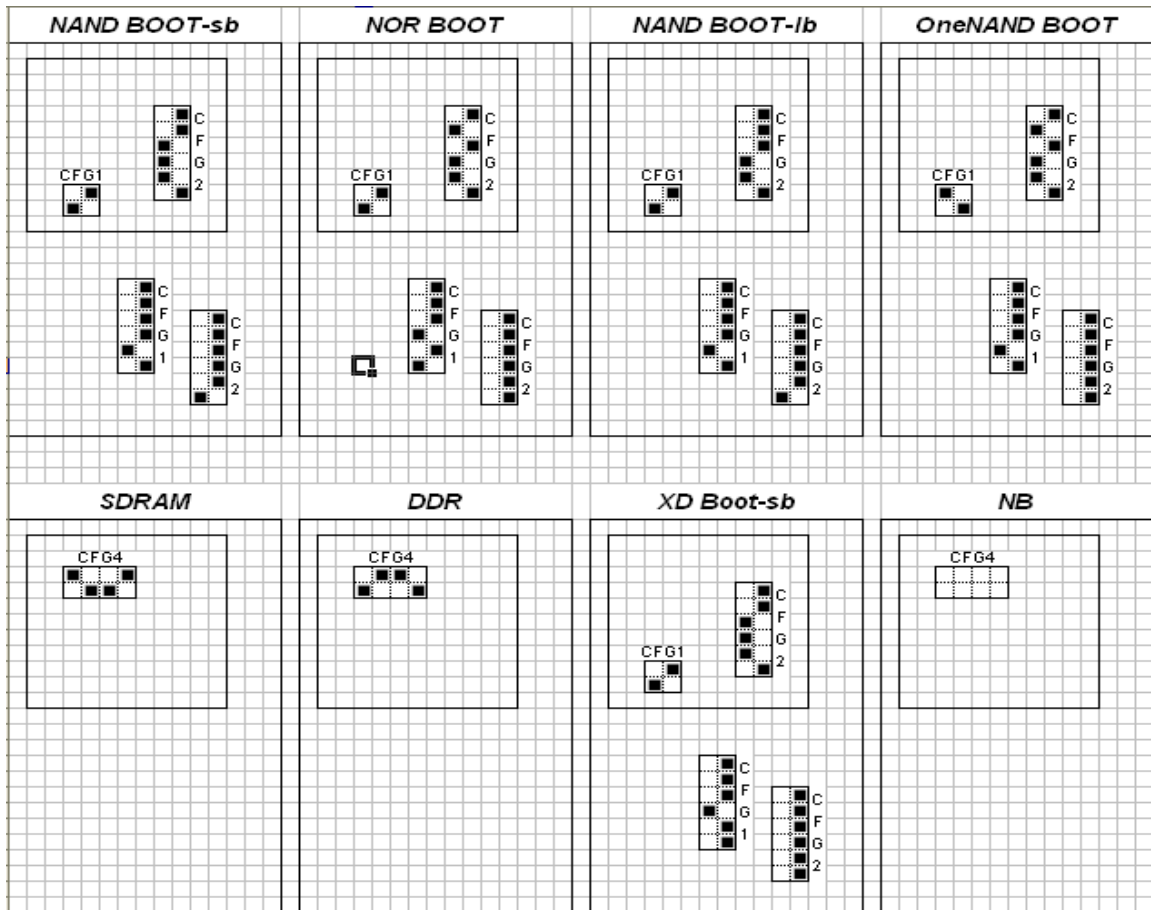


Figure 4-2 Jumper Configuration for SMDK2443

4.2.3 SMDK2450

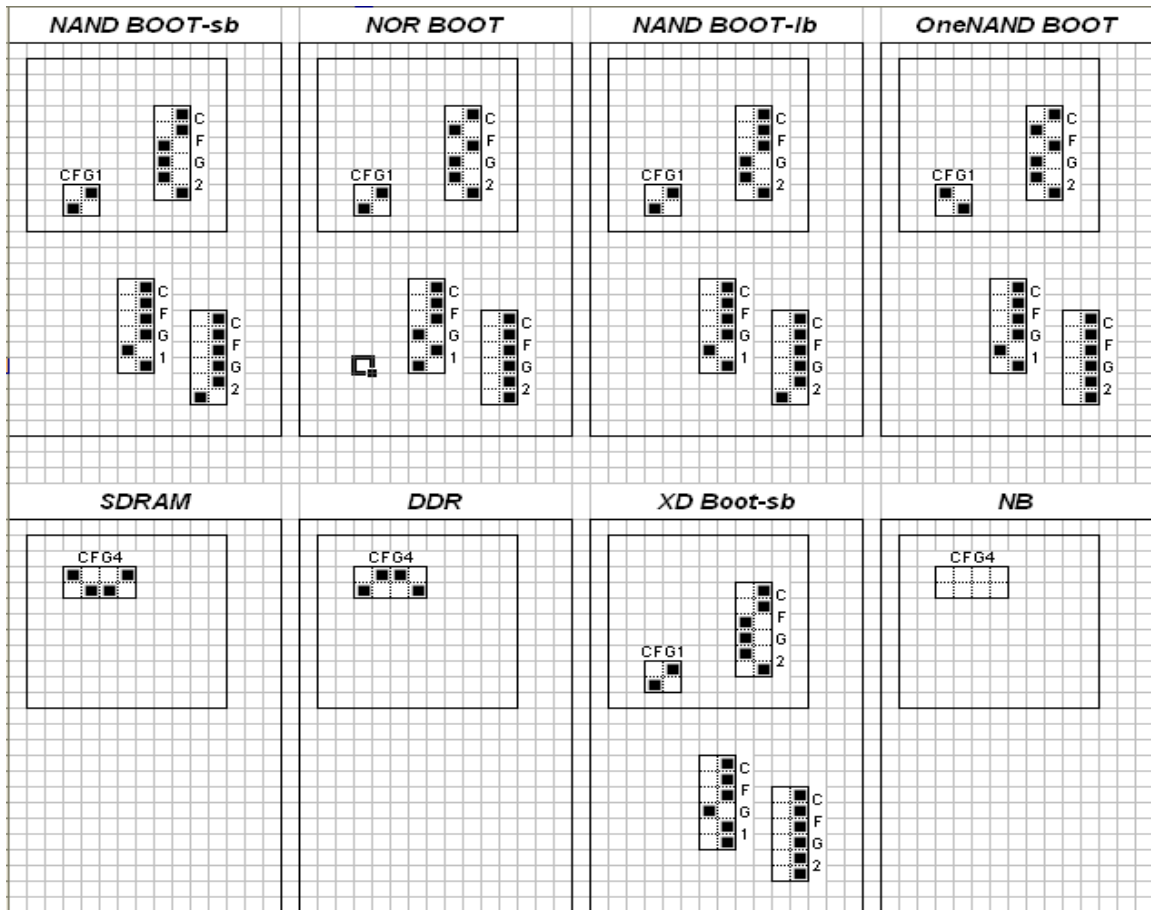


Figure 4-3 Jumper Configuration for SMDK2450

4.2.4 SMDK6400

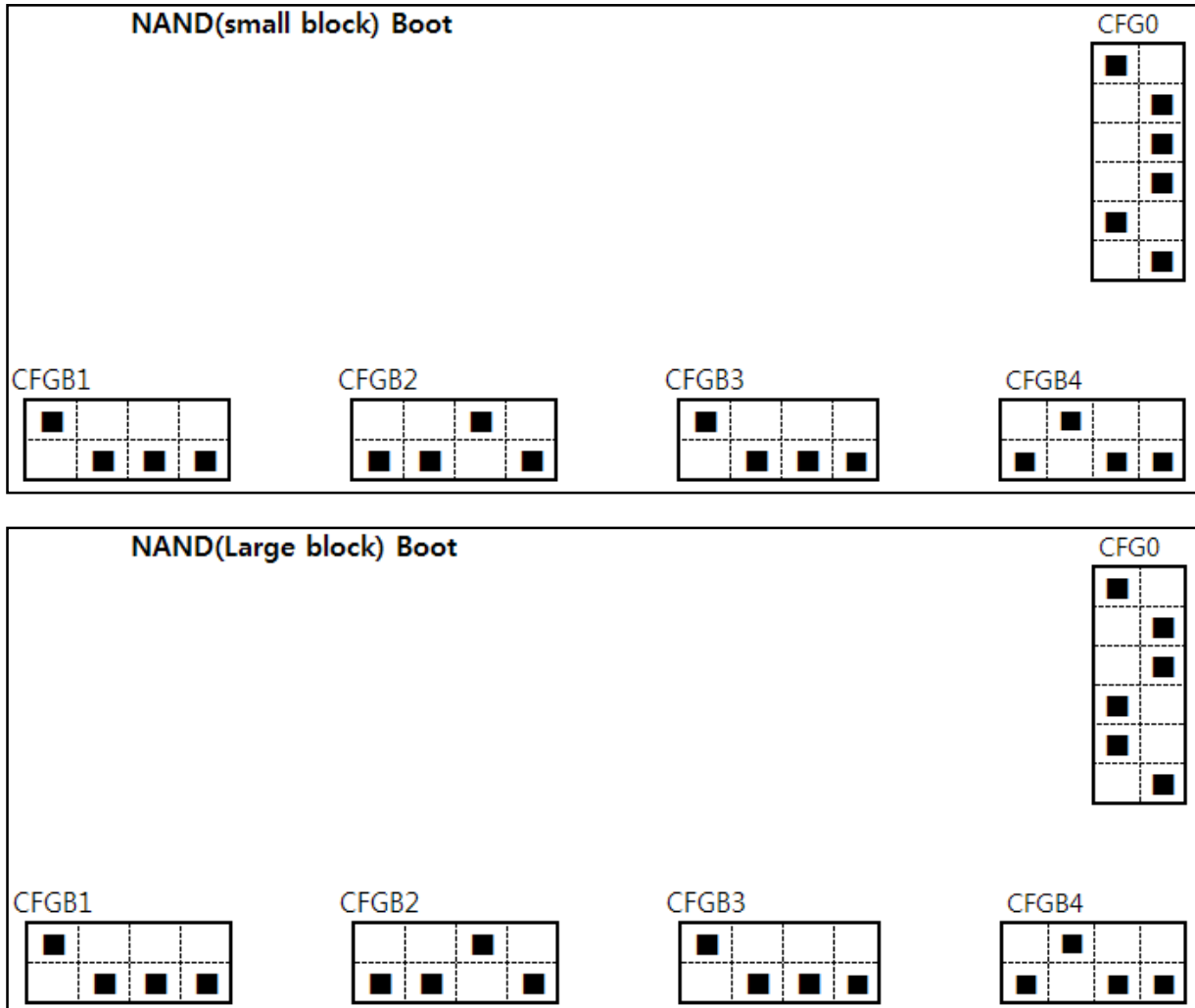


Figure 4-4 Jumper Configuration for SMDK6400

4.2.5 SMDK6410(or SMDK6430)

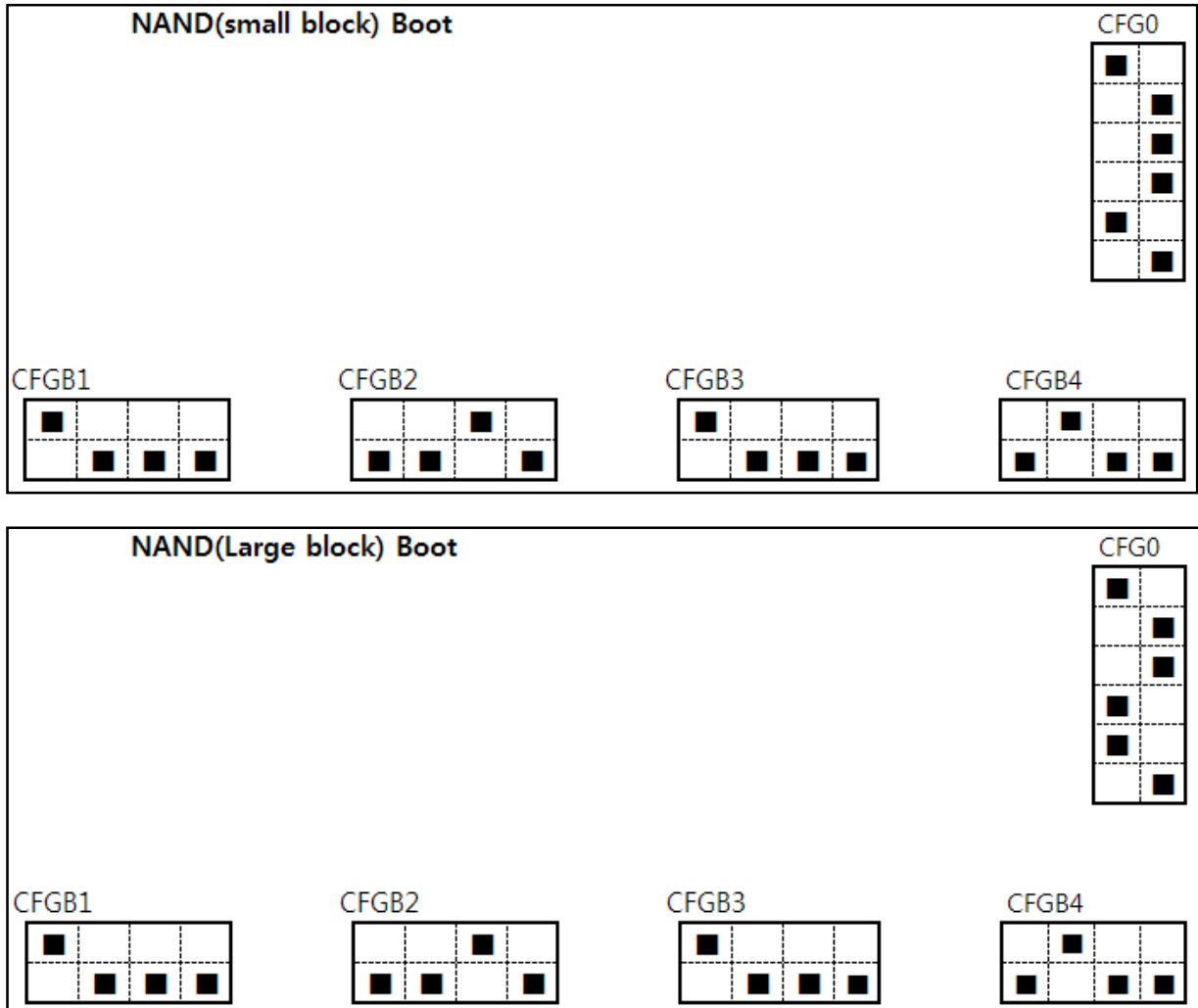


Figure 4-5 Jumper Configuration for SMDK6410

4.3 Burning U-Boot Script for Trace32 ICD

4.3.1 SMDK2412

The burning script is "Burning_uboot_for_s3c2412.cmm" as follows:

```

;~~~~~;
;
; CMM file for burning U-boot to SAMSUNG S3C2412
;
; Copyrighted by SAMSUNG Electronics
;
;~~~~~;

; For path to U-boot
&UBOOT_IMAGE="Y:\tftpboot\w-u-boot.bin"
&UBOOT_LOADED_ADDRESS=0x30008000
&UBOOT_EXECUTED_ADDRESS=0x33E00000

sYmbol.RESet
MMU.RESet
RESet

SYStem.CPU ARM926EJ
SYStem.Option EnReset on
SYStem.Option waitreset on

SYStem.Option mmu OFF
SYStem.Option dacr on
SYStem.Option ResBreak OFF

;TrOnChip.Reset
TrOnChip.Set FIQ OFF

```

```

TrOnChip.Set IRQ OFF
TrOnChip.Set DABORT OFF
TrOnChip.Set PABORT OFF
TrOnChip.Set SWI OFF
TrOnChip.Set UNDEF OFF

Screen.Always

system.JTAGCLOCK RTCK

print "(1/4) system going up....."
SYStem.Up

;~~~~~;
; Initialize S3C2412
;~~~~~;
;~~~~~;
; system configuration
;~~~~~;
;disable watchdog
data.set SD:0x53000000 %LE %LONG 0x0

;ebi
data.set SD:0x48800004 %LE %LONG 0xcc

;r.s cpsr (r(cpsr)&0xfffff00)|0xd3

;set CLOCK
data.set SD:0x4C000014 %LE %LONG 0x65 ;CLKDiv 1:2:4
data.set SD:0x4C000004 %LE %LONG 0x0002a011 ;MPLL - 200MHz, PLL off
data.set SD:0x4C00001C %LE %LONG 0x00000030 ;CLK Source ?

```

```
data.set SD:0x4C000008 %LE %LONG 0x00040070 ;UPLL - 96MHz

;SDRAM Initialization
; 1st : Configuration Register
data.set SD:0x48000000 %LE %LONG 0x48904 ; BANKCFG
data.set SD:0x48000004 %LE %LONG 0x40 ; BANKCON1 - Mobile DRAM
Controller
data.set SD:0x48000008 %LE %LONG 0x57003a ; BANKCON2 - Timing Parameter
data.set SD:0x4800000c %LE %LONG 0x80000030 ; BANKCON3 - EMRS Register

;2nd : make refresh cycle 32clk
data.set 0x48000010 %LONG 0xff

;3rd : wait 120clk
r.s r0 r(r0)

;4th : issue MRS
data.set SD:0x48000004 %LE %LONG 0x42

;5th : set normal operation refresh cycle
data.set SD:0x48000010 %LE %LONG 0x313

;6th : issue EMRS
data.set SD:0x48000004 %LE %LONG 0x43

;7th : issue Normal mode
data.set SD:0x48000004 %LE %LONG 0x40

;SSMC for OneNAND
data.set SD:0x4F000034 %LE %LONG 0x00303012
data.set SD:0x4F000204 %LE %LONG 0x3
```

```
data.set SD:0x4F000024 %LE %LONG 0xe
data.set SD:0x4F000028 %LE %LONG 0xe

;~~~~~;
; Load the IMAGE of U-boot
;~~~~~;
print "(2/4) uboot loading"

Data.LOAD.Binary &UBOOT_IMAGE &UBOOT_EXECUTED_ADDRESS
Data.LOAD.Binary &UBOOT_IMAGE &UBOOT_LOADED_ADDRESS

;1st magic code for burng U-Boot
data.set SD:0x30000000 %LE %LONG 0x24564236

;2nd magic code for burng U-Boot
data.set SD:0x30000004 %LE %LONG 0x20764316

;~~~~~;
; kenrel break & run
;~~~~~;
print "(3/4) wait booting up....."

Register.Set pc &UBOOT_EXECUTED_ADDRESS

go

wait 3.s
break

&value=data.long(a:0x30000000)
```

```

if &value==0x27051956
(
  print "(4/4) U-boot has been burned successfully. The board will reboot ..."
  SYStem.Up
  go
  enddo
)
print "Burning U-boot failed."
enddo

```

4.3.2 SMDK2443

The burning script is "**Burning_uboot_for_s3c2443.cmm**" as follows:

```

;~~~~~;
;
; CMM file for burning U-boot to SAMSUNG S3C2443
;
; Copyrighted by SAMSUNG Electronics
;
;~~~~~;

; For path to U-boot
&UBOOT_IMAGE="Y:\wftpboot\w-u-boot.bin"
&UBOOT_LOADED_ADDRESS=0x30008000
&UBOOT_EXECUTED_ADDRESS=0x33E00000

;-----;
; Project Path ;
;-----;

area.create A000 80. 05.

```

```
area.view A000
area.select A000

;-----;
; Configuring JTAG interface ;
;-----;

print "Reset System"
SYStem.Down
sYmbol.RESet
sys.reset
MMU.RESet

print "Configuring S3C2443"
SYStem.CPU arm920T
SYStem.JtagClock 10M
SYStem.Option EnReset OFF
SYStem.Option WaitReset OFF
SYStem.Option ResBreak OFF
SYStem.Option dacr on

print "system going up"
SYStem.Up

;-----;
; System controller setting ;
;-----;

print "Clock & Timer setting"
Register.Set cpsr (r(cpsr)&0xfffff00)|0xd3

Data.Set SD:0x53000000 %LE %LONG 0x0 ;disable watchdog
Data.Set SD:0x4C000024 %LE %LONG 0x1D ;set clkdiv(1:4:8) - CLKDIV0
```

```

Data.Set SD:0x4C000010 %LE %LONG 0x5C0301      ;Mpll:400MHz(92,3,1) - MPLLCON
Data.Set SD:0x4C000018 %LE %LONG 0x280101      ;Epll:96MHz(40,1,1) - EPLLCON
Data.Set SD:0x4C000020 %LE %LONG 0x10          ;MPLL output

Data.Set SD:0x560000e0 %LE %LONG 0xaaaaaaaa    ;GPKCON to SDATA[31:16],
                                           ;because default value is input
;-----;
; SDRAM Initialization                      ;
;-----;
print "SDRAM Initialization"
print "Select memory type 1) mSDRAM, 2) mDDR SDRAM"
print "select the number : "
enter &choice

if &choice==1 ; mSDR
(
;1st : configuration register
d.s SD:0x48000000 %LE %LONG 0x48904 ; BANKCFG
d.s SD:0x48000004 %LE %LONG 0x40 ; BANKCON1 - mobile dram controller
)
else if &choice==2 ; mDDR
(
;1st : configuration register
d.s SD:0x48000000 %LE %LONG 0x4920D ; BANKCFG
d.s SD:0x48000004 %LE %LONG 0x44000040 ; BANKCON1 - mobile dram controller
)

;2nd
d.s SD:0x48000008 %LE %LONG 0x57003a ; BANKCON2 - timing parameter
d.s SD:0x4800000c %LE %LONG 0x80000030 ; BANKCON3 - EMRS register

```

```
;3rd : issue precharge all command
d.s SD:0x48000004 %LE %LONG 0x41

;4th : make refresh cycle 255clk
d.s SD:0x48000010 %LE %LONG 0xff

;5th : wait 2 auto-refresh cycle: 120clk?
r.s r0 r(r0)

;6th : issue MRS
d.s SD:0x48000004 %LE %LONG 0x42

;7th : set normal operation refresh cycle
d.s SD:0x48000010 %LE %LONG 0x313

;8th : issue EMRS
d.s SD:0x48000004 %LE %LONG 0x43

;9th : issue Normal mode
d.s SD:0x48000004 %LE %LONG 0x40

print "DRAM Initialization done..."

;~~~~~;
; Load the IMAGE of U-boot
;~~~~~;
print "(2/4) uboot loading"

Data.LOAD.Binary &UBOOT_IMAGE &UBOOT_EXECUTED_ADDRESS
Data.LOAD.Binary &UBOOT_IMAGE &UBOOT_LOADED_ADDRESS
```



```
;1st magic code for burnng U-Boot
data.set SD:0x30000000 %LE %LONG 0x24564236

;2nd magic code for burnng U-Boot
data.set SD:0x30000004 %LE %LONG 0x20764316

;~~~~~;
; kenrel break & run
;~~~~~;
print "(3/4) wait booting up...."

Register.Set pc &UBOOT_EXECUTED_ADDRESS

go

wait 3.s
break

&value=data.long(a:0x30000000)

if &value==0x27051956
(
    print "(4/4) U-boot of 2443 has been burned successfully. The board will reboot ..."
    system.mode.go
)
else
(
    print "Burning U-boot failed. value = &value"
)
)
```

```
enddo
```

4.3.3 SMDK2450

The burning script is "**Burning_uboot_for_s3c2443.cmm**" as follows:

```

;~~~~~;
;
; CMM file for burning U-boot to SAMSUNG S3C2450
;
; Copyrighted by SAMSUNG Electronics
;
;~~~~~;

; For path to U-boot
&UBOOT_IMAGE="W:\working\Wu-boot\W3c-u-boot-1.1.6\Wu-boot.bin"
&UBOOT_LOADED_ADDRESS=0x30008000
&UBOOT_EXECUTED_ADDRESS=0x33E00000

;-----;
; Project Path ;
;-----;

area.create A000 80. 05.
area.view A000
area.select A000

;-----;
; Configuring JTAG interface ;
;-----;

print "Reset System"
SYStem.Down

```

```
sYmbol.RESet
sys.reset
MMU.RESet

print "Configuring S3C2450"
SYStem.CPU arm926EJ
SYStem.JtagClock RTCK
SYStem.Option EnReset ON
SYStem.Option WaitReset ON
SYStem.Option ResBreak OFF
SYStem.Option dacr on
SYStem.Option AMBA on

TrOnChip.Reset
TrOnChip.Set FIQ OFF
TrOnChip.Set IRQ OFF
TrOnChip.Set DABORT OFF
TrOnChip.Set PABORT OFF
TrOnChip.Set SWI OFF
TrOnChip.Set UNDEF OFF

print "system going up"
SYStem.Up

;-----;
; System controller setting                ;
;-----;

print "Clock & Timer setting"
Register.Set cpsr (r(cpsr)&0xfffff00)|0xd3

Data.Set SD:0x53000000 %LE %LONG 0x0          ;disable watchdog
```

```

Data.Set SD:0x4C000024 %LE %LONG 0x1D          ;set clkdiv(1:4:8) - CLKDIV0
Data.Set SD:0x4C000010 %LE %LONG 0x5C0301      ;Mpll:400MHz(92,3,1) - MPLLCON
Data.Set SD:0x4C000018 %LE %LONG 0x280101      ;Epll:96MHz(40,1,1) - EPLLCON
Data.Set SD:0x4C000020 %LE %LONG 0x10          ;MPLL output
Data.Set SD:0x560000e0 %LE %LONG 0xaaaaaaaa    ;GPKCON to SDATA[31:16],
;-----;
; SDRAM Initialization                          ;
;-----;
print "SDRAM Initialization"
print "Select memory type 1) mSDRAM, 2) mDDR SDRAM"
print "select the number : "
enter &choice

if &choice==1 ; mSDR
(
;1st : configuration register
d.s SD:0x48000000 %LE %LONG 0x48904 ; BANKCFG
d.s SD:0x48000004 %LE %LONG 0x40 ; BANKCON1 - mobile dram controller
)
else if &choice==2 ; mDDR
(
;1st : configuration register
d.s SD:0x48000000 %LE %LONG 0x4920D ; BANKCFG
d.s SD:0x48000004 %LE %LONG 0x44000040 ; BANKCON1 - mobile dram controller
)

;2nd
d.s SD:0x48000008 %LE %LONG 0x57003a ; BANKCON2 - timing parameter
d.s SD:0x4800000c %LE %LONG 0x80000030 ; BANKCON3 - EMRS register

;3rd : issue precharge all command

```

```
d.s SD:0x48000004 %LE %LONG 0x41

;4th : make refresh cycle 255clk
d.s SD:0x48000010 %LE %LONG 0xff

;5th : wait 2 auto-refresh cycle: 120clk?
r.s r0 r(r0)

;6th : issue MRS
d.s SD:0x48000004 %LE %LONG 0x42

;7th : set normal operation refresh cycle
d.s SD:0x48000010 %LE %LONG 0x313

;8th : issue EMRS
d.s SD:0x48000004 %LE %LONG 0x43

;9th : issue Normal mode
d.s SD:0x48000004 %LE %LONG 0x40

print "DRAM Initialization done..."

;~~~~~;
; Load the IMAGE of U-boot
;~~~~~;
print "(2/4) uboot loading"

Data.LOAD.Binary &UBOOT_IMAGE &UBOOT_EXECUTED_ADDRESS
Data.LOAD.Binary &UBOOT_IMAGE &UBOOT_LOADED_ADDRESS
```

```
;1st magic code for burng u-boot
data.set SD:0x30000000 %LE %LONG 0x24564236

;2nd magic code for burng u-boot
data.set SD:0x30000004 %LE %LONG 0x20764316

;~~~~~;
; kenrel break & run
;~~~~~;
print "(3/4) wait booting up...."

Register.Set pc &UBOOT_EXECUTED_ADDRESS

go

wait 3.s
break

&value=data.long(a:0x30000000)
print "value = &value"

if &value==0x27051956
(
    print "(4/4) U-boot of 2443 has been burned successfully. The board will reboot ..."
    system.mode.go
)
else
(
    print "Burning U-boot failed. value = &value"
)
)
```

enddo

4.3.4 SMDK6400

The burning script is "**Burning_uboot_for_s3c6400.cmm**" as follows:

```

;~~~~~
~~~~~;
;
; CMM file for burning U-boot to SAMSUNG S3C6400
;
; Copyrighted by SAMSUNG Electronics
;
;~~~~~
~~~~~;
; For path to U-boot
&UBOOT_ORG="Y:\wftpboot\w-u-boot.bin"
&UBOOT_LOADED_ADDRESS=0x50008000
&UBOOT_EXECUTED_ADDRESS=0x57E00000

;-----;
; Configuring JTAG interface ;
;-----;

print "Resest System"
RESet
SYStem.Down
sYmbol.RESet
SYStem.RESet
MMU.RESet

print "Configuring S3C6400"
SYStem.CPU arm1176jzf

```

```

SYStem.JC RTCK
SYStem.Option mmu off
SYStem.Option dacr on
SYStem.Option enreset off
SYStem.Option trst on
SYStem.MultiCore IRPRE 4 ;IR core C
SYStem.MultiCore DRPRE 1 ;count of Codre
SYStem.MultiCore ETBIRPOST 5 ;setting for ETB
SYStem.MultiCore ETBDRPOST 1

print "System going up"
SYStem.up

;-----;
; System controller setting ;
;-----;

print "Clock & Timer setting"
Register.Set cpsr (r(cpsr)&0xfffff00)|0xd3
PER.S C15:0x42F %LE %LONG 0x70000013 ; Peripheral Port Enable
Data.Set SD:0x7e004000 %LE %LONG 0x0 ; Disable Watchdog

;Data.Set SD:0x7e00f120 %LE %LONG 0x1000 ; CS0:16bit, Mem1:32bit, CS2=SR0MC
Data.Set SD:0x7e00f120 %LE %LONG 0x0003 ; CS0:8 bit, Mem1:32bit, CS2=NAND
;Data.Set SD:0x7e00f120 %LE %LONG 0x1002 ; CS0:16bit, Mem1:32bit, CS2=OND

print "Operating Mode Change to Sync. Mode"
Data.Set SD:0x7e00f900 %LE %LONG 0x805E ; Change SYNCMUX[6] to "1"
wait 1000.us ; Wait for a while...
Data.Set SD:0x7e00f900 %LE %LONG 0x80DE ; Assert SYNCREQ&VICSYNCEN to "1"(rb1004
modify)
wait 1000.us ; while Others[11:8] to 0xF

```



```

Data.Set SD:0x7e00f000 %LE %LONG 0xffff           ; APLL LockTime
Data.Set SD:0x7e00f004 %LE %LONG 0xffff           ; MPLL LockTime
Data.Set SD:0x7e00f020 %LE %LONG 0x1047310         ; ARMCLK:HCLK:PCLK = 1:4:16
Data.Set SD:0x7e00f00c %LE %LONG 0x81900302        ; A:400, P:3, S:2 => 400MHz
Data.Set SD:0x7e00f010 %LE %LONG 0x81900303        ; M:400, P:3, S:3 => 200MHz

Data.Set SD:0x7e00f01c %LE %LONG 0x3              ; APLL,MPLL Clock Select

;-----;
; DRAM Initialization                               ;
;-----;

print "DRAM Initialization"

Data.Set SD:0x7e001004 %LE %LONG 0x4              ; Enter the Config State
Data.Set SD:0x7e001010 %LE %LONG 0x30C           ; Refresh Period register (7800ns), 100MHz
0x30E
;Data.Set SD:0x7e001010 %LE %LONG 0x40E         ; Refresh Period register (7800ns), 133MHz
0x40E

Data.Set SD:0x7e001014 %LE %LONG 0x6             ; CAS Latency = 3
Data.Set SD:0x7e001018 %LE %LONG 0x1            ; T_DQSS
Data.Set SD:0x7e00101c %LE %LONG 0x2            ; T_MRD
Data.Set SD:0x7e001020 %LE %LONG 0x7            ; T_RAS(45ns)
Data.Set SD:0x7e001024 %LE %LONG 0xA            ; T_RC(67.5ns)
Data.Set SD:0x7e001028 %LE %LONG 0xC            ; T_RCD(22.5ns) = 4, Scheduled RCD = 1
Data.Set SD:0x7e00102C %LE %LONG 0x10B          ; T_RFC(80ns) = 11, Scheduled RFC= 8
Data.Set SD:0x7e001030 %LE %LONG 0xC            ; T_RP(22.5ns) = 4, Scheduled RP = 1
Data.Set SD:0x7e001034 %LE %LONG 0x3            ; T_RRD(15ns)=3
Data.Set SD:0x7e001038 %LE %LONG 0x3            ; T_WR(15ns)=3
Data.Set SD:0x7e00103C %LE %LONG 0x2            ; T_WTR
Data.Set SD:0x7e001040 %LE %LONG 0x2            ; T_XP (1tck + tIS(1.5ns))

```

```

Data.Set SD:0x7e001044 %LE %LONG 0x11          ; T_XSR(120ns)
Data.Set SD:0x7e001048 %LE %LONG 0x11          ; T_ESR

print "Memory Configuration Register"
Data.Set SD:0x7e00100C %LE %LONG 0x00010012    ; 1 CKE, 1Chip, 4burst, Always, AP[10],
ROW/Column bit
print "Memory Configuration Register 2"
Data.Set SD:0x7e00104C %LE %LONG 0x0B41        ; Read delay 1 Cycle, mDDR, 32bit, Sync.
print "Chip 0 Configuration"
Data.Set SD:0x7e001200 %LE %LONG 0x150F8 ; Bank-ROW-Column, 0x5000_0000 ~ 0x57ff_ffff
(128MB)

print "Memory Direct Command"
Data.Set SD:0x7e001008 %LE %LONG 0xc0000 ; Chip0 Direct Command :NOP5
Data.Set SD:0x7e001008 %LE %LONG 0x0          ; Chip0 Direct Command :PreCharge all
Data.Set SD:0x7e001008 %LE %LONG 0x40000 ; Chip0 Direct Command :AutoRefresh
Data.Set SD:0x7e001008 %LE %LONG 0x40000 ; Chip0 Direct Command :AutoRefresh
Data.Set SD:0x7e001008 %LE %LONG 0xA0000 ; EMRS, DS:Full, PASR:Full
Data.Set SD:0x7e001008 %LE %LONG 0x80032      ; MRS, CAS3, BL4
Data.Set SD:0x7e001004 %LE %LONG 0x0          ; Enable DMC1

;~~~~~
~~~~~;
; Load the IMAGE of U-boot
;~~~~~
~~~~~;

print "(2/4) uboot loading"
Data.LOAD.Binary      &UBOOT_ORG &UBOOT_EXECUTED_ADDRESS
;Data.LOAD.Binary      &UBOOT_IMAGE      &UBOOT_EXECUTED_ADDRESS
Data.LOAD.Binary      &UBOOT_IMAGE      &UBOOT_LOADED_ADDRESS
Data.LOAD.ELF         &UBOOT_CODE /ABSLIFETIMES /gnu /nocode /STRIPPART 3.

```

```
SYMBOL.SOURCEPATH.SETBASEDIR Y:₩.
```

```
;1st magic code for burng U-Boot
```

```
Data.Set SD:0x50000000 %LE %LONG 0x24564236
```

```
;2nd magic code for burng U-Boot
```

```
Data.Set SD:0x50000004 %LE %LONG 0x20764316
```

```
;~~~~~;
```

```
Break & Run
```

```
;~~~~~;
```

```
print "(3/4) wait booting up...."
```

```
Register.Set pc &UBOOT_EXECUTED_ADDRESS
```

```
go
```

```
wait 3.s
```

```
break
```

```
&value=data.long(a:0x50000000)
```

```
if &value==0x27051956
```

```
(
```

```
    print "(4/4) U-boot of S3C6400 has been burned successfully. The board will reboot ..."
```

```
    system.mode.go
```

```
)
```

```
else
```

```
(
```

```
    print "Burning U-boot failed. value = &value"
```

```
)
```

enddo

4.3.5 SMDK6410(or SMDK6430)

The burning script is "Burning_uboot_for_S3C6410(or S3C6430).cmm" as follows:

```

;~~~~~
~~~~~;
;
; CMM file for burning U-boot to SAMSUNG S3C6410(OR S3C6430)
;
; Copyrighted by SAMSUNG Electronics
;
;~~~~~
~~~~~;
; For path to U-boot
&UBOOT_ORG="Y:\wftpboot\w-u-boot.bin"
&UBOOT_LOADED_ADDRESS=0x50008000
&UBOOT_EXECUTED_ADDRESS=0x57E00000

;-----;
; Configuring JTAG interface ;
;-----;

print "Resest System"
RESet
SYStem.Down
sYmbol.RESet
SYStem.RESet
MMU.RESet

print "Configuring S3C6410(OR S3C6430)"

```

```

SYStem.CPU arm1176jzf
SYStem.JC RTCK
SYStem.Option mmu off
SYStem.Option dacr on
SYStem.Option enreset off
SYStem.Option trst on
SYStem.MultiCore IRPRE 4 ;IR core C
SYStem.MultiCore DRPRE 1 ;count of Codre
SYStem.MultiCore ETBIRPOST 5 ;setting for ETB
SYStem.MultiCore ETBDRPOST 1

print "System going up"
SYStem.up

;-----;
; System controller setting ;
;-----;

print "Clock & Timer setting"
Register.Set cpsr (r(cpsr)&0xfffff00)|0xd3
PER.S C15:0x42F %LE %LONG 0x70000013 ; Peripheral Port Enable
Data.Set SD:0x7e004000 %LE %LONG 0x0 ; Disable Watchdog

;Data.Set SD:0x7e00f120 %LE %LONG 0x1000 ; CS0:16bit, Mem1:32bit, CS2=SROMC
Data.Set SD:0x7e00f120 %LE %LONG 0x0003 ; CS0:8 bit, Mem1:32bit, CS2=NAND
;Data.Set SD:0x7e00f120 %LE %LONG 0x1002 ; CS0:16bit, Mem1:32bit, CS2=OND

print "Operating Mode Change to Sync. Mode"
Data.Set SD:0x7e00f900 %LE %LONG 0x805E ; Change SYNCMUX[6] to "1"
wait 1000.us ; Wait for a while...
Data.Set SD:0x7e00f900 %LE %LONG 0x80DE ; Assert SYNCREQ&VICSYNCEN to "1"(rb1004
modify)

```

```

wait 1000.us                ; while Others[11:8] to 0xF

Data.Set SD:0x7e00f000 %LE %LONG 0xffff        ; APLL LockTime
Data.Set SD:0x7e00f004 %LE %LONG 0xffff        ; MPLL LockTime
Data.Set SD:0x7e00f020 %LE %LONG 0x1047310     ; ARMCLK:HCLK:PCLK = 1:4:16
Data.Set SD:0x7e00f00c %LE %LONG 0x81900302    ; A:400, P:3, S:2 => 400MHz
Data.Set SD:0x7e00f010 %LE %LONG 0x81900303    ; M:400, P:3, S:3 => 200MHz

Data.Set SD:0x7e00f01c %LE %LONG 0x3          ; APLL,MPLL Clock Select

;-----;
; DRAM Initialization          ;
;-----;

print "DRAM Initialization"
Data.Set SD:0x7e001004 %LE %LONG 0x4          ; Enter the Config State
Data.Set SD:0x7e001010 %LE %LONG 0x30C      ; Refresh Period register (7800ns), 100MHz
0x30E
;Data.Set SD:0x7e001010 %LE %LONG 0x40E     ; Refresh Period register (7800ns), 133MHz
0x40E
Data.Set SD:0x7e001014 %LE %LONG 0x6        ; CAS Latency = 3
Data.Set SD:0x7e001018 %LE %LONG 0x1       ; T_DQSS
Data.Set SD:0x7e00101c %LE %LONG 0x2       ; T_MRD
Data.Set SD:0x7e001020 %LE %LONG 0x7       ; T_RAS(45ns)
Data.Set SD:0x7e001024 %LE %LONG 0xA       ; T_RC(67.5ns)
Data.Set SD:0x7e001028 %LE %LONG 0xC       ; T_RCD(22.5ns) = 4, Scheduled RCD = 1
Data.Set SD:0x7e00102C %LE %LONG 0x10B     ; T_RFC(80ns) = 11, Scheduled RFC= 8
Data.Set SD:0x7e001030 %LE %LONG 0xC       ; T_RP(22.5ns) = 4, Scheduled RP = 1
Data.Set SD:0x7e001034 %LE %LONG 0x3       ; T_RRD(15ns)=3
Data.Set SD:0x7e001038 %LE %LONG 0x3       ; T_WR(15ns)=3
Data.Set SD:0x7e00103C %LE %LONG 0x2       ; T_WTR

```

```

Data.Set SD:0x7e001040 %LE %LONG 0x2          ; T_XP (1tck + tIS(1.5ns))
Data.Set SD:0x7e001044 %LE %LONG 0x11        ; T_XSR(120ns)
Data.Set SD:0x7e001048 %LE %LONG 0x11        ; T_ESR

print "Memory Configuration Register"
Data.Set SD:0x7e00100C %LE %LONG 0x00010012  ; 1 CKE, 1Chip, 4burst, Always, AP[10],
ROW/Column bit
print "Memory Configuration Register 2"
Data.Set SD:0x7e00104C %LE %LONG 0x0B41      ; Read delay 1 Cycle, mDDR, 32bit, Sync.
Print "Chip 0 Configuration"
Data.Set SD:0x7e001200 %LE %LONG 0x150F8 ; Bank-ROW-Column, 0x5000_0000 ~ 0x57ff_ffff
(128MB)

print "Memory Direct Command"
Data.Set SD:0x7e001008 %LE %LONG 0xc0000 ; Chip0 Direct Command :NOP5
Data.Set SD:0x7e001008 %LE %LONG 0x0          ; Chip0 Direct Command :PreCharge all
Data.Set SD:0x7e001008 %LE %LONG 0x40000 ; Chip0 Direct Command :AutoRefresh
Data.Set SD:0x7e001008 %LE %LONG 0x40000 ; Chip0 Direct Command :AutoRefresh
Data.Set SD:0x7e001008 %LE %LONG 0xA0000 ; EMRS, DS:Full, PASR:Full
Data.Set SD:0x7e001008 %LE %LONG 0x80032     ; MRS, CAS3, BL4
Data.Set SD:0x7e001004 %LE %LONG 0x0         ; Enable DMC1

;~~~~~
~~~~~;
; Load the IMAGE of U-boot
;~~~~~
~~~~~;

print "(2/4) uboot loading"
Data.LOAD.Binary      &UBOOT_ORG &UBOOT_EXECUTED_ADDRESS
;Data.LOAD.Binary     &UBOOT_IMAGE      &UBOOT_EXECUTED_ADDRESS
Data.LOAD.Binary      &UBOOT_IMAGE      &UBOOT_LOADED_ADDRESS

```

```
Data.LOAD.ELF      &UBOOT_CODE /ABSIFETIMES /gnu /nocode /STRIPPART 3.

SYMBOL.SOURCEPATH.SETBASEDIR Y:₩.

;1st magic code for burng U-Boot
Data.Set SD:0x50000000 %LE %LONG 0x24564236

;2nd magic code for burng U-Boot
Data.Set SD:0x50000004 %LE %LONG 0x20764316

;~~~~~;
Break & Run
;~~~~~;
print "(3/4) wait booting up...."
Register.Set pc &UBOOT_EXECUTED_ADDRESS

go

wait 3.s
break

&value=data.long(a:0x50000000)

if &value==0x27051956
(
    print "(4/4) U-boot of S3C6410(OR S3C6430) has been burned successfully. The board will
reboot ..."
    system.mode.go
)
else
(
```



```
print "Burning U-boot failed. Value = &value"  
)  
enddo
```

5 Write images to SMDK Target Board

In this Chapter, you will understand the following:

- Section 5.1, "Introduction"
- Section 5.2, "Minicom"
- Section 5.3, "TFTP Server"
- Section 5.4, "Setting IP Address"
- Section 5.5, "Transferring and Writing Images by TFTP"

5.1 Introduction

In this chapter, you will understand the procedure for porting embedded linux to SMDK target board and also how to write **u-boot.bin** (bootloader), **zImage** (kernel image) and "**Qt_samsung.cramfs**" to NAND Flash memory by using "**tftp**" utility. This method can be used after booting the target board because it is used for writing images to new NAND Flash.

Transfer the images and the needed utilities to the target board, because all works are progressed in target board. Transfer all the images from "**/tftpboot**" directory to the target board by using tftp utility.

5.2 Minicom

Before transferring images using tftp, you should know how to use Minicom. In this section, you will learn how to setup Minicom. Desktop Linux has Minicom program for serial communication. It is used for command prompt of uboot.bin or shell prompt of embedded Linux. Set up the values before using Minicom program. Execute the command below:

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

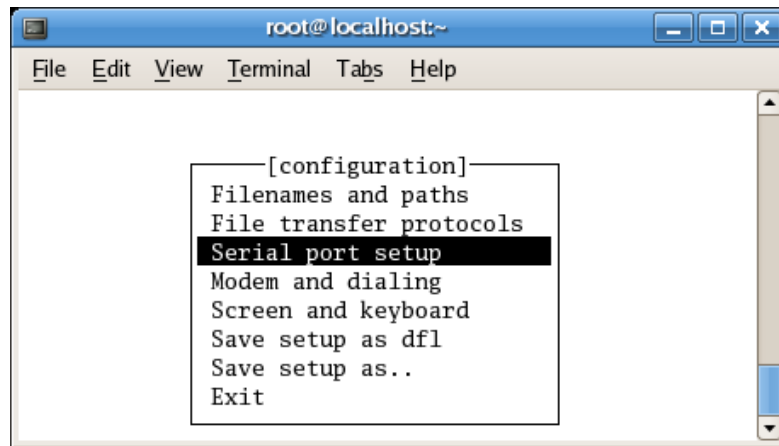


Figure 5-1 Minicom Setup – 1

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

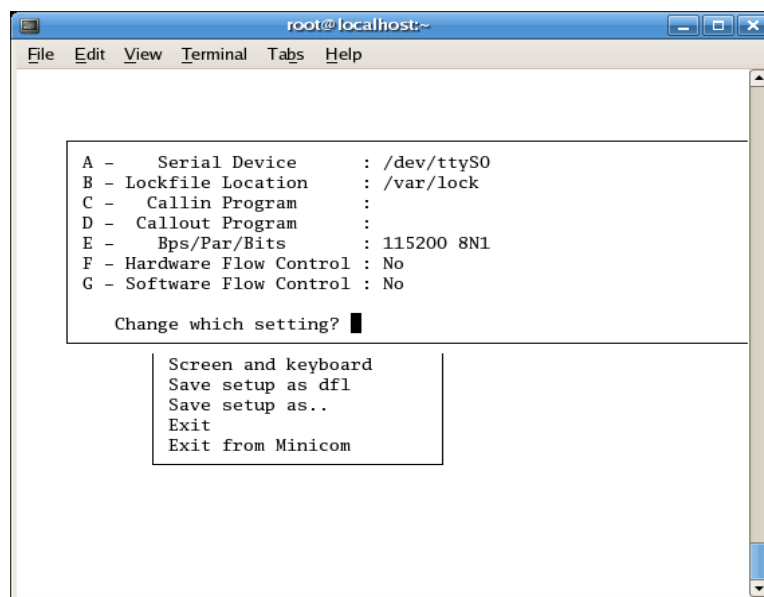


Figure 5-2 Minicom Setup – 2

Select **‘E’** key for setting up **“bps/Par/Bits”**. Select **‘I’** to set up **“bps”** to 115200, select **‘V’** to set up **“Data bits”** to 8, select **‘W’** to set up **“Stop bits”** to **‘1’**, and **‘V’** to set up **“parity”** to **“NONE”**.

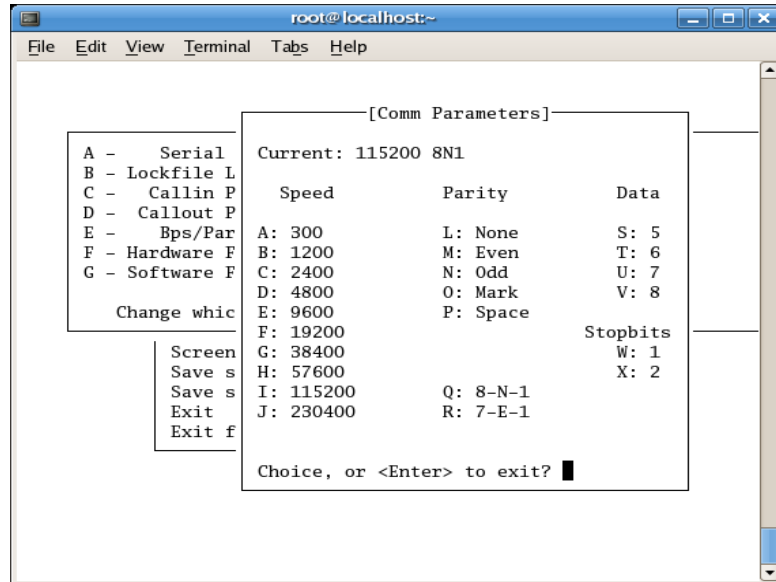


Figure 5-3 Minicom Setup – 3

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

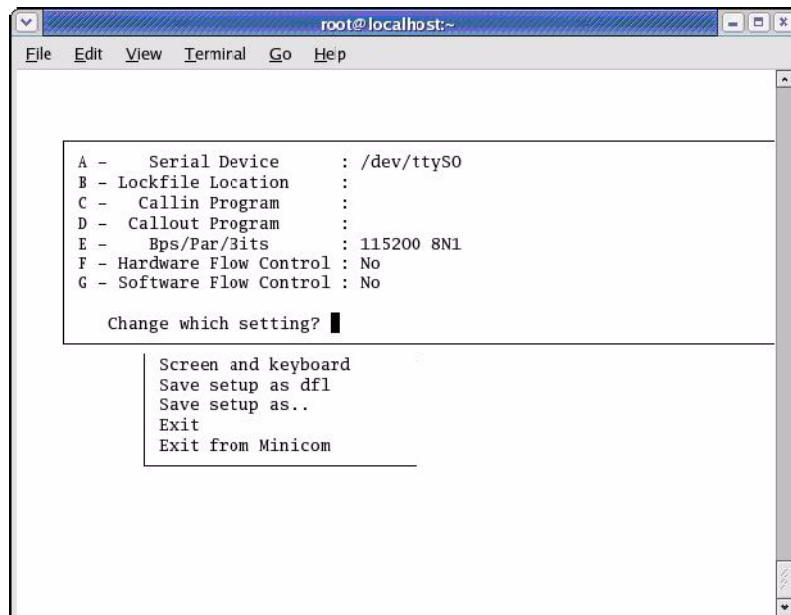


Figure 5-4 Minicom Setup – 4

Once setting is over, please press “Enter” key and select “Save setup as dfl” item, then press “Enter” for saving the values.

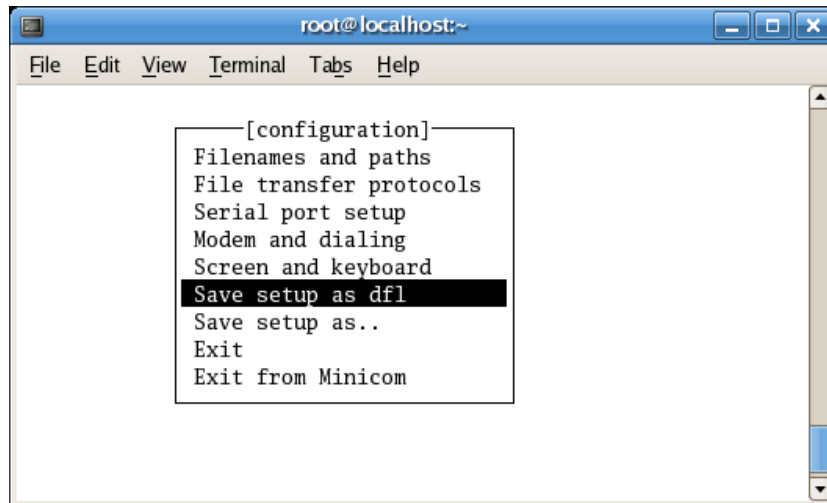


Figure 5-5 Minicom Setup – 5

Select “Exit” key, to exit from the setting mode. Currently, the set points are stored to the file “/etc/minirc.dfl”.

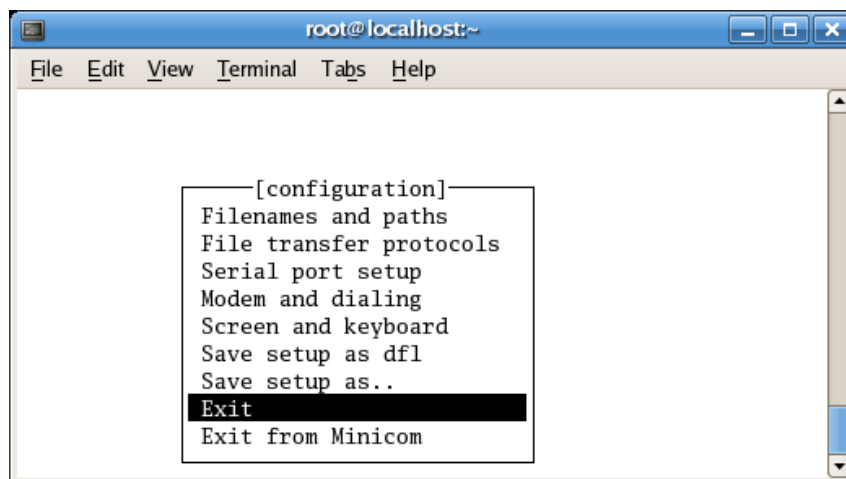


Figure 5-6 Minicom Setup – 6

To quit from Minicom, please press “**Ctrl + A**” and then ‘**Z**’, at last enter ‘**Q**’ key and then Selecting “**Yes**”, Minicom is quitted.

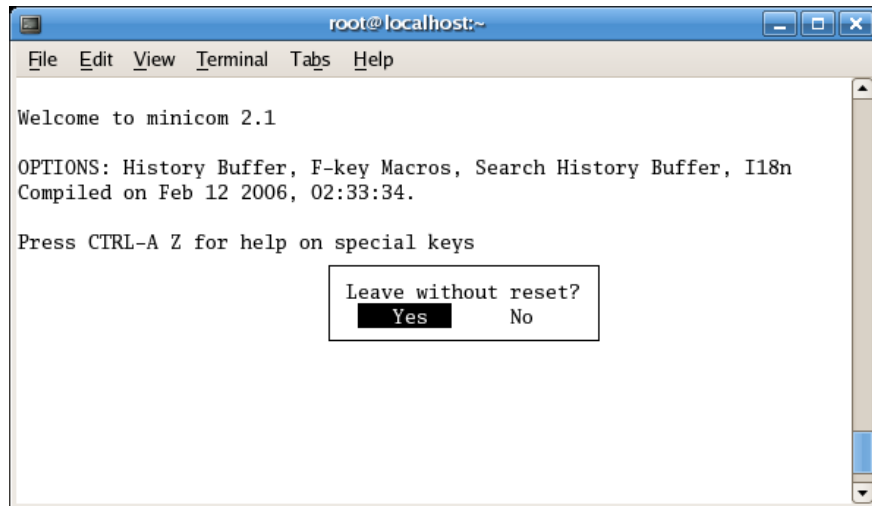


Figure 5-7 Minicom Setup – 7

5.3 tftp Server

To start using tftp server program you have to setup your computer by executing the following command.

```
[root@localhost root]# setup
```

Once the command is executed , you can see the “**Text Mode Setup Utility**”. Please select “**System services**”.

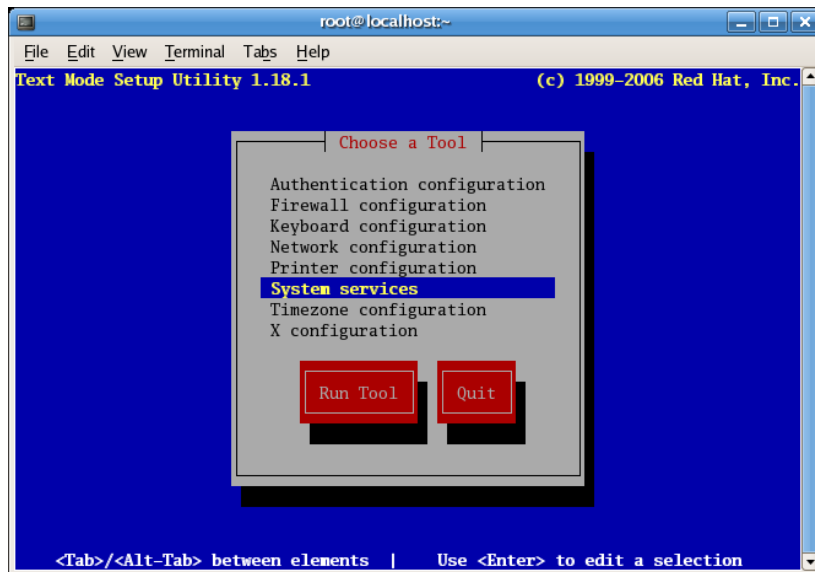


Figure 5-8 Settings for System services – 1

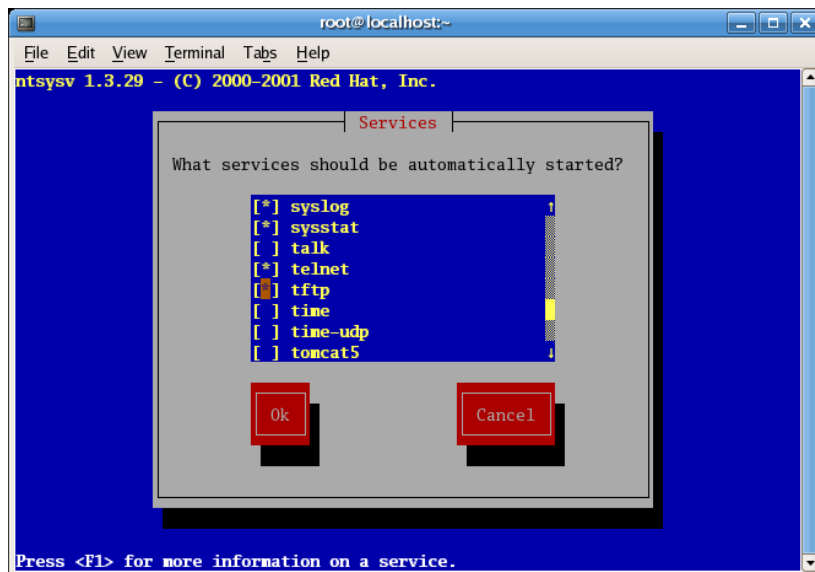


Figure 5-9 Settings for System services – 2

Please select “**tftp**” service and click “**OK**”. Finally “**quit**” setup utility and execute the following command.

```
[root@localhost root]# service xinetd restart
```

Now you can download compiled images to the target board by using tftp. Before downloading the images, connect host PC and target board by Ethernet cable.

5.4 Setting IP Address

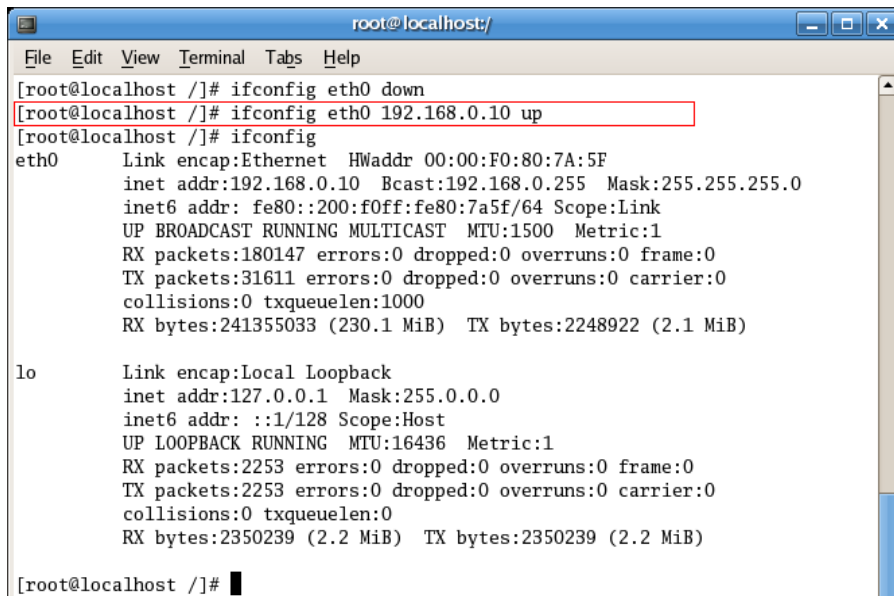
In this section, you will learn how to set IP address. Setting up an IP address helps in downloading the compiled images to the target board.

Please connect host PC and target board by Ethernet cable. If you are connecting PC LAN port to the Target Board, please use the Ethernet cross cable.

5.4.1 Setting IP Address on Host PC

On Your Linux Host PC, run the terminal and execute following commands to set up IP address.

```
[root@localhost tftpboot]# ifconfig eth0 down
[root@localhost tftpboot]# ifconfig eth0 192.168.0.10 netmask 255.255.255.0 up
[root@localhost tftpboot]# ifconfig
```

```
root@localhost:/
File Edit View Terminal Tabs Help
[root@localhost /]# ifconfig eth0 down
[root@localhost /]# ifconfig eth0 192.168.0.10 up
[root@localhost /]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:00:F0:80:7A:5F
          inet addr:192.168.0.10  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::200:f0ff:fe80:7a5f/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:180147  errors:0  dropped:0  overruns:0  frame:0
          TX packets:31611  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:1000
          RX bytes:241355033 (230.1 MiB)  TX bytes:2248922 (2.1 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128  Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:2253  errors:0  dropped:0  overruns:0  frame:0
          TX packets:2253  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:0
          RX bytes:2350239 (2.2 MiB)  TX bytes:2350239 (2.2 MiB)

[root@localhost /]#
```

Figure 5-10 Setting IP Address on Host PC

5.4.2 Setting IP Address for SMDK Target Board

Before starting this section, U-Boot image has to already be written in the NAND flash. To set the IP address for SMDK target boardroom the Minicom and “**Switch ON**” the target board. Please press any key to get each board command prompt.

SMDK evaluation board	Command prompt
SMDK2412	SMDK2412(1.4) #
SMDK2412(under version 1.4)	SMDK2412 #
SMDK2443	SMDK2443 #
SMDK2450	SMDK2450 #
SMDK6400	SMDK6400 #
SMDK6410	SMDK6410 #
SMDK6430	SMDK6430 #

Table 5-1 Command prompts in U-Boot

Execute the command “**printenv**” as shown next.

```
In: serial
Out: serial
Err: serial
Net: Found CS8900@0x29000300
Hit any key to stop autoboot: 0
SMDK2412(1.4) # pri
bootcmd=nand read c0008000 40000 1c0000;bootm c0008000
bootdelay=3
baudrate=115200
ethaddr=00:40:5c:26:0a:5b
ipaddr=192.168.0.20
serverip=192.168.0.10
netmask=255.255.255.0
stdin=serial
stdout=serial
stderr=serial

Environment size: 214/16380 bytes
SMDK2412(1.4) #
```

CTRL-A Z for help | 115200 8N1 | MOR | Minicom 2.1 | VT102 | Offline

Figure 5-11 Parameters in SMDK2412

```

root@yongkal:~
File Edit View Terminal Tabs Help
U-Boot 1.1.6 (Mar 15 2007 - 16:46:04) for SMDK2443

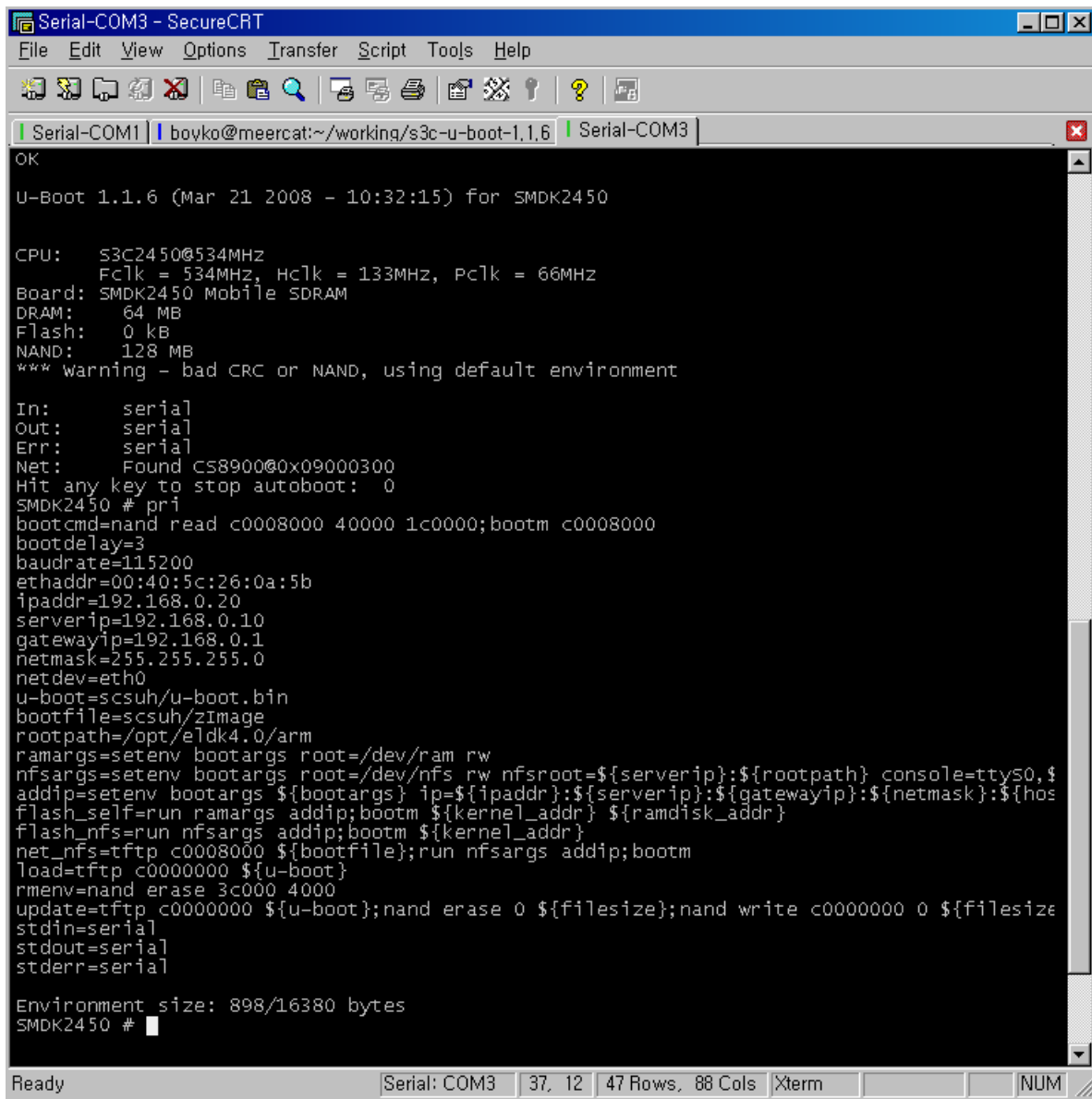
CPU:   S3C2443@534MHz
       Fclk = 534MHz, Hclk = 133MHz, Pclk = 66MHz
Board: SMDK2443 Mobile SDRAM
DRAM:  64 MB
Flash:  0 kB
NAND:   128 MB
ONENAND:0 MB
*** Warning - bad CRC or NAND, using default environment

In:     serial
Out:    serial
Err:    serial
Net:    Found CS8900@0x09000300
Hit any key to stop autoboot:  0
SMDK2443 # pri
bootcmd=nand read c0008000 40000 1c0000;bootm c0008000
bootdelay=3
baudrate=115200
ethaddr=00:40:5c:26:0a:5b
ipaddr=192.168.0.20
serverip=192.168.0.10
gatewayip=192.168.0.1
netmask=255.255.255.0
netdev=eth0
u-boot=scsuh/u-boot.bin
bootfile=scsuh/zImage
rootpath=/opt/eldk4.0/arm
ramargs=setenv bootargs root=/dev/ram rw
nfsargs=setenv bootargs root=/dev/nfs rw nfsroot=${serverip}:${rootpath} consol}
addip=setenv bootargs ${bootargs} ip=${ipaddr}:${serverip}:${gatewayip}:${netma}
flash_self=run ramargs addip;bootm ${kernel_addr} ${ramdisk_addr}
flash_nfs=run nfsargs addip;bootm ${kernel_addr}
net_nfs=tftp c0008000 ${bootfile};run nfsargs addip;bootm
load=tftp c0000000 ${u-boot}
rmenv=nand erase 3c000 4000
update=tftp c0000000 ${u-boot};nand erase 0 ${filesize};nand write c0000000 0 $;
stdin=serial
stdout=serial
stderr=serial

Environment size: 898/16380 bytes
SMDK2443 #
CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.1 | VT102 | Offline

```

Figure 5-12 Parameters in SMDK2443



```

Serial-COM3 - SecureCRT
File Edit View Options Transfer Script Tools Help
Serial-COM1 | boyko@meercat:~/working/s3c-u-boot-1,1,6 | Serial-COM3
OK
U-Boot 1.1.6 (Mar 21 2008 - 10:32:15) for SMDK2450

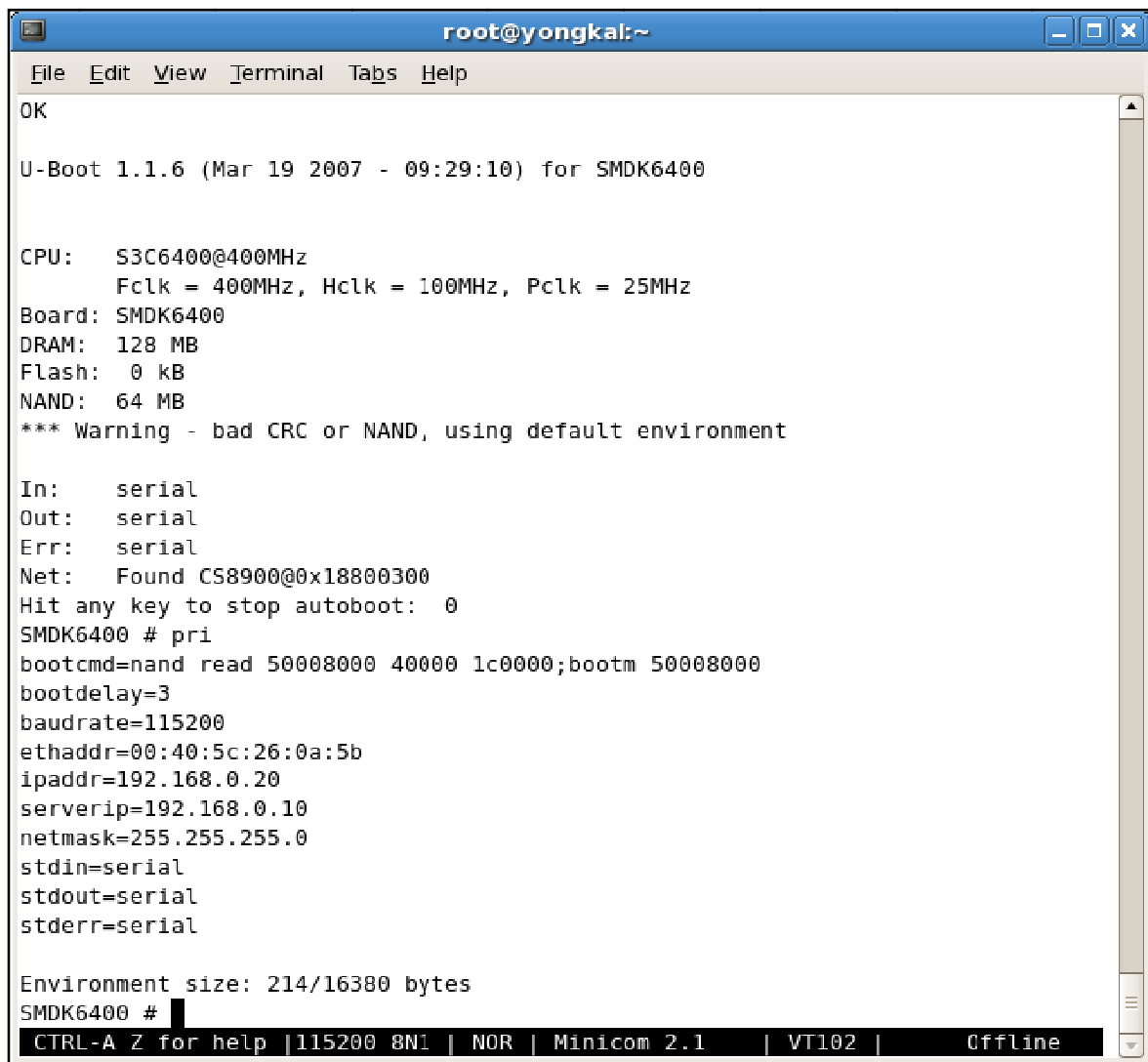
CPU:   S3C2450@534MHZ
       Fclk = 534MHZ, Hclk = 133MHZ, Pclk = 66MHZ
Board: SMDK2450 Mobile SDRAM
DRAM:  64 MB
Flash:  0 kB
NAND:  128 MB
*** Warning - bad CRC or NAND, using default environment

In:     serial
Out:    serial
Err:    serial
Net:    Found CS8900@0x09000300
Hit any key to stop autoboot:  0
SMDK2450 # pri
bootcmd=nand read c0008000 40000 1c0000;bootm c0008000
bootdelay=3
baudrate=115200
ethaddr=00:40:5c:26:0a:5b
ipaddr=192.168.0.20
serverip=192.168.0.10
gatewayip=192.168.0.1
netmask=255.255.255.0
netdev=eth0
u-boot=scsuh/u-boot.bin
bootfile=scsuh/zImage
rootpath=/opt/eldk4.0/arm
ramargs=setenv bootargs root=/dev/ram rw
nfsargs=setenv bootargs root=/dev/nfs rw nfsroot=${serverip}:${rootpath} console=ttyS0,$
addip=setenv bootargs ${bootargs} ip=${ipaddr}:${serverip}:${gatewayip}:${netmask}:${hos
flash_self=run ramargs addip;bootm ${kernel_addr} ${ramdisk_addr}
flash_nfs=run nfsargs addip;bootm ${kernel_addr}
net_nfs=tftp c0008000 ${bootfile};run nfsargs addip;bootm
load=tftp c0000000 ${u-boot}
rmenv=nand erase 3c000 4000
update=tftp c0000000 ${u-boot};nand erase 0 ${filesize};nand write c0000000 0 ${filesize}
stdin=serial
stdout=serial
stderr=serial

Environment size: 898/16380 bytes
SMDK2450 #
Ready          Serial: COM3   37, 12   47 Rows, 88 Cols   Xterm   NUM

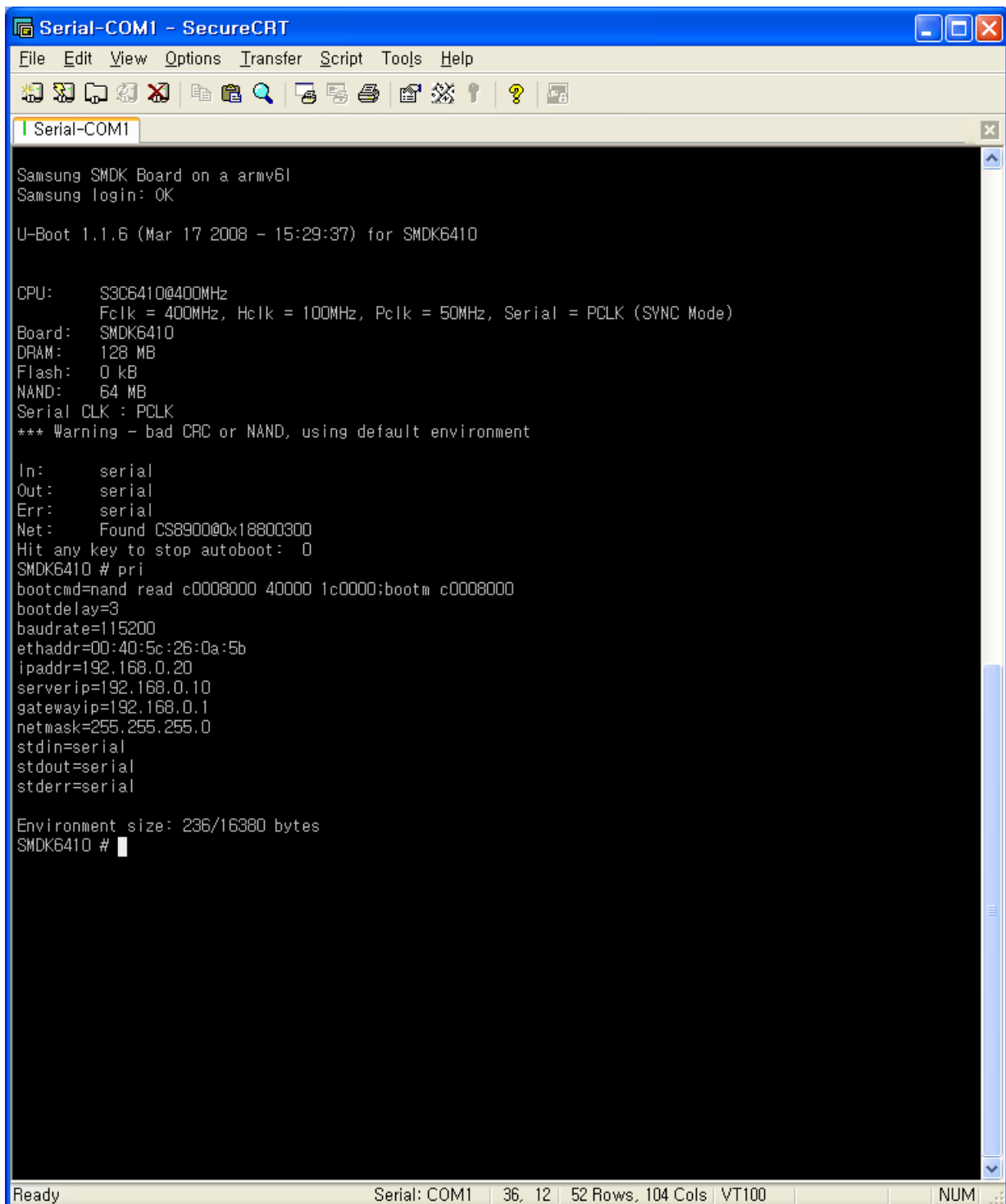
```

Figure 5-13 Parameters in SMDK2450



```
root@yongkal:~  
File Edit View Terminal Tabs Help  
OK  
U-Boot 1.1.6 (Mar 19 2007 - 09:29:10) for SMDK6400  
  
CPU:   S3C6400@400MHz  
       Fclk = 400MHz, Hclk = 100MHz, Pclk = 25MHz  
Board: SMDK6400  
DRAM:  128 MB  
Flash:  0 kB  
NAND:  64 MB  
*** Warning - bad CRC or NAND, using default environment  
  
In:     serial  
Out:    serial  
Err:    serial  
Net:    Found CS8900@0x18800300  
Hit any key to stop autoboot:  0  
SMDK6400 # pri  
bootcmd=nand read 50008000 40000 1c0000;bootm 50008000  
bootdelay=3  
baudrate=115200  
ethaddr=00:40:5c:26:0a:5b  
ipaddr=192.168.0.20  
serverip=192.168.0.10  
netmask=255.255.255.0  
stdin=serial  
stdout=serial  
stderr=serial  
  
Environment size: 214/16380 bytes  
SMDK6400 #  
CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.1 | VT102 | Offline
```

Figure 5-14 Parameters in SMDK6400



```

Serial-COM1 - SecureCRT
File Edit View Options Transfer Script Tools Help
Serial-COM1
Samsung SMDK Board on a armv6l
Samsung login: OK

U-Boot 1.1.6 (Mar 17 2008 - 15:29:37) for SMDK6410

CPU:      S3C6410@400MHz
          Fclk = 400MHz, Hclk = 100MHz, Pclk = 50MHz, Serial = PCLK (SYNC Mode)
Board:    SMDK6410
DRAM:     128 MB
Flash:    0 kB
NAND:     64 MB
Serial CLK : PCLK
*** Warning - bad CRC or NAND, using default environment

In:       serial
Out:      serial
Err:      serial
Net:      Found CS8900@0x18800300
Hit any key to stop autoboot:  0
SMDK6410 # pri
bootcmd=nand read c0008000 40000 1c0000;bootm c0008000
bootdelay=3
baudrate=115200
ethaddr=00:40:5c:26:0a:5b
ipaddr=192.168.0.20
serverip=192.168.0.10
gatewayip=192.168.0.1
netmask=255.255.255.0
stdin=serial
stdout=serial
stderr=serial

Environment size: 236/16380 bytes
SMDK6410 # █
Ready          Serial: COM1  36, 12  52 Rows, 104 Cols  VT100          NUM
  
```

Figure 5-15 Parameters in SMDK6410

You can set and add the environment parameter using “**setenv**”, “**saveenv**”. Please execute the following commands.

```
SMDK○○○○ # setenv ipaddr XXX.XXX.XXX.XXX
SMDK○○○○ # setenv serverip XXX.XXX.XXX.XXX
SMDK○○○○ # saveenv
```

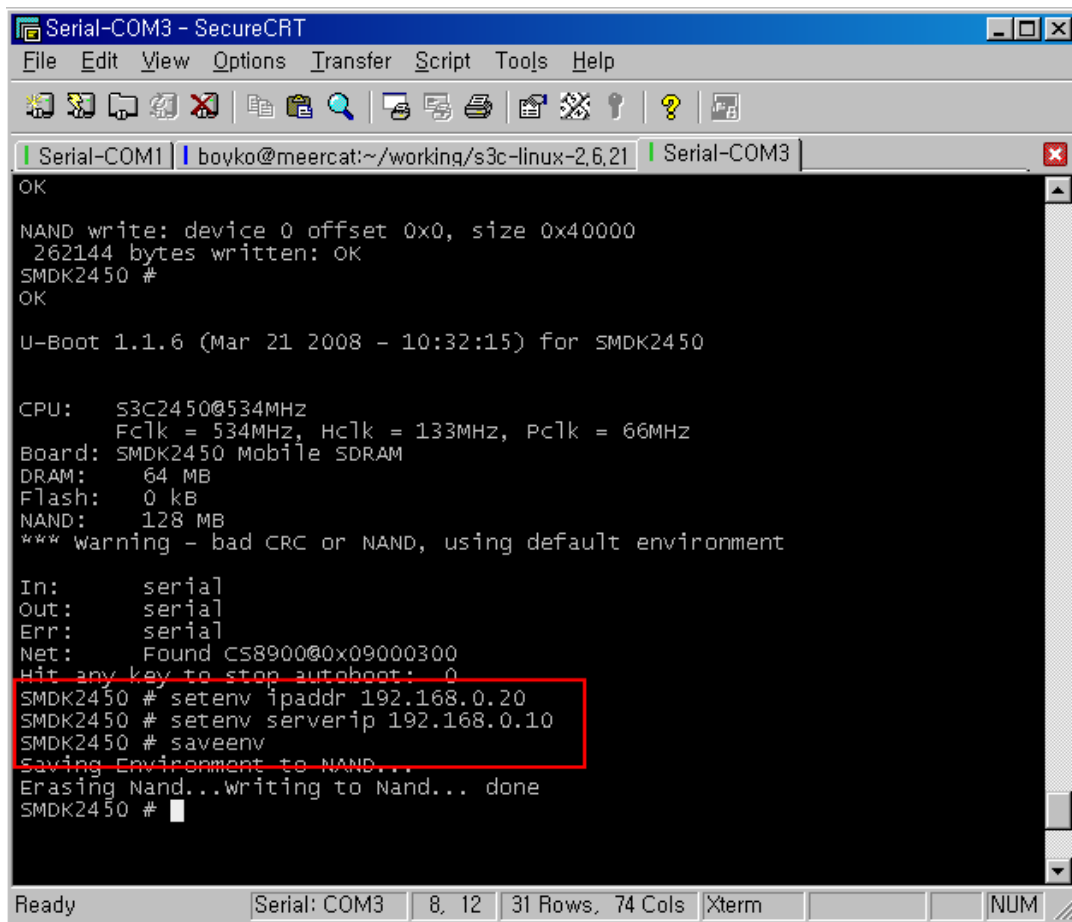
An example is explained in figure.

Example:

```
ipaddr 192.168.0.20
serverip 192.168.0.10
```

Note1: Please make sure the serverip for target board and host PC IP address are same.

Note2: Please disable firewall services on your host machine.



```

Serial-COM3 - SecureCRT
File Edit View Options Transfer Script Tools Help
Serial-COM1 | boyko@meercat:~/working/s3c-linux-2,6,21 | Serial-COM3
OK
NAND write: device 0 offset 0x0, size 0x40000
262144 bytes written: OK
SMDK2450 #
OK
U-Boot 1.1.6 (Mar 21 2008 - 10:32:15) for SMDK2450

CPU:   S3C2450@534MHz
       Fclk = 534MHz, Hclk = 133MHz, Pclk = 66MHz
Board: SMDK2450 Mobile SDRAM
DRAM:  64 MB
Flash: 0 kB
NAND:  128 MB
*** warning - bad CRC or NAND, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   Found CS8900@0x09000300
Hit any key to stop autoboot: 0
SMDK2450 # setenv ipaddr 192.168.0.20
SMDK2450 # setenv serverip 192.168.0.10
SMDK2450 # saveenv
Saving Environment to NAND...
Erasing Nand...writing to Nand... done
SMDK2450 #

```

Figure 5-16 Set and Save Parameters

5.5 Transfer and Write Images using tftp

In this section, you will learn how to transfer and write images using tftp. Images to be transferred to target board are “u-boot.bin”(U-Boot Bootloader), “zImage”(Linux Kernel Image) and “Qtopia1.7_demo_image.cramfs”(Root file system). **Assume that U-Boot is already written NAND in this section.**

Generally NAND is used by 4 part(partition) in SMDK board like as below

NAND block number	NAND block offset	Size	Contents
0x0 (0x0~0xF)	0x0 (0x0~0x3FFFF)	256KB	U-Boot Bootloader
0x10 (0x10~0x7F)	0x40000 (0x40000~0x1FFFFFF)	1.75MB	Linux Kernel
0x80 (0x10~0xC7F)	0x200000 (0x200000~0x31FFFFFF)	48MB	Root file system
0xC80 (0xC80~end)	0x3200000 (0x3200000~end)	(NAND 50MB)	– Extra area

Table 5-2 Partition Assignment

If you are using this NAND part(partition) system, at least NAND block 0x0~0xC7F area should not have bad block. And you use different file system for RFS(ie. YAFFS2, JFFS2), you can use non bad block NAND in block 0x0~0x7F area.

Please follow the commands below to transfer file.

```
SMDK○○○○ # tftp <temporary address> <image name>
SMDK○○○○ # nand erase [clean] <start block offset> <image size>
```

Now, have to execute the following command for erasing NAND before writing image. 'clean' is an optional parameter for jffs2 file system.

Now, execute the following command for writing the file to NAND.

```
SMDK○○○○ # nand write[.jffs2 | .yaffs] <temporary address> <start block offset> <image size>
```

'jffs2' is an optional parameter for jffs2 file system. Also, 'yaffs' is for yaffs2 file system.

5.5.1 Transfer and Write "u-boot.bin" (bootloader)

```
SMDK○○○○ # tftp c0000000 u-boot.bin
SMDK○○○○ # nand erase 0 40000
SMDK○○○○ # nand write c0000000 0 40000
```

Temporary address is the base address of SDRAM, i.e. 0xC0000000(virtual address). Start block offset of bootloader is 0, it is the NAND flash's first block offset. Image size of bootloader will be below 256K(0x40000), hence bootloader will occupy space within 0x10 blocks (block 0~0xF, size 0x40000).

```

root@yongkal:~
File Edit View Terminal Tabs Help
OK
U-Boot 1.1.6 (Mar 19 2007 - 10:08:45) for SMDK2443

CPU:   S3C2443@534MHz
       Fclk = 534MHz, Hclk = 133MHz, Pclk = 66MHz
Board: SMDK2443 Mobile SDRAM
DRAM:  64 MB
Flash: 0 kB
NAND:  128 MB
ONENAND:0 MB
*** Warning - bad CRC or NAND, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   Found CS8900@0x09000300
Hit any key to stop autoboot: 0
SMDK2443 # tftp c0000000 u-boot.bin
TFTP from server 192.168.0.10; our IP address is 192.168.0.20
Filename 'u-boot.bin'.
Load address: 0xc0000000
Loading: #####
done
Bytes transferred = 245760 (3c000 hex)
SMDK2443 # nand erase 0 40000
NAND erase: device 0 offset 0x0, size 0x40000
Erasing at 0x3c000 -- 100% complete.
OK
SMDK2443 # nand write c0000000 0 3c000
NAND write: device 0 offset 0x0, size 0x3c000
245760 bytes written: OK
SMDK2443 # █

CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.1 | VT102 | Offline

```

Figure 5-17 Transfer and Write 'u-boot.bin'

5.5.2 Transfer and Write "zImage" (Kernel Image)

```
SMDK○○○○ # tftp c0000000 zImage  
SMDK○○○○ # nand erase 40000 1c0000  
SMDK○○○○ # nand write c0000000 40000 1c0000
```

Temporary address is the base address of SDRAM, 0xC0000000(virtual address). Start block offset of kernel is 0x40000. Now, the written kernel image size is less than 1.75MB(0x1c0000), it varies depending on the menuconfig options and module selection. Kernel image can occupy up to 1.75MB (0x10~0x7F blocks), so image size will be below 1.75MB.

```

root@yongkal:~
File Edit View Terminal Tabs Help
Err: serial
Net: Found CS8900@0x09000300
Hit any key to stop autoboot: 0
SMDK2443 # tftp c0000000 u-boot.bin
TFTP from server 192.168.0.10; our IP address is 192.168.0.20
Filename 'u-boot.bin'.
Load address: 0xc0000000
Loading: #####
done
Bytes transferred = 245760 (3c000 hex)
SMDK2443 # nand erase 0 40000

NAND erase: device 0 offset 0x0, size 0x40000
Erasing at 0x3c000 -- 100% complete.
OK
SMDK2443 # nand write c0000000 0 3c000

NAND write: device 0 offset 0x0, size 0x3c000
245760 bytes written: OK
SMDK2443 # tftp c0000000 zImage
TFTP from server 192.168.0.10; our IP address is 192.168.0.20
Filename 'zImage'.
Load address: 0xc0000000
Loading: #####
#####
#####
#####
done
Bytes transferred = 1104492 (10da6c hex)
SMDK2443 # nand erase 40000 1c0000

NAND erase: device 0 offset 0x40000, size 0x1c0000
Erasing at 0x1fc000 -- 100% complete.
OK
SMDK2443 # nand write c0000000 40000 110000

NAND write: device 0 offset 0x40000, size 0x110000
1114112 bytes written: OK
SMDK2443 #
CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.1 | VT102 | Offline

```

Figure 5-18 Transfer and Write zImage

5.5.3 Transfer and Write "Qtopia1.7_demo_image.cramfs" (Root File System)

```
SMDK○○○○ # tftp c0000000 Qtopia1.7_demo_image.cramfs
SMDK○○○○ # nand erase 200000 3000000
SMDK○○○○ # nand write c0000000 200000 3000000
```

If you want to write jffs2 file system image on root file system area, use 'nand erase clean 200000 3000000' and 'nand write.jffs2 c0000000 200000 <size> instead of above commands. <size> must be actual size which had downloaded by tftp. After downloading complete via tftp, you can see actual size of image like following figure. Likewise this, to write yaffs2 image, you have to use 'nand write.yaffs 200000 <actual size>' (warning: not 'nand write.yaffs2' !!)

```
done
Bytes transferred = 1104492 (10da6c hex)
```

Temporary address is the base address of SDRAM, 0xC0000000(virtual address). Start block offset of kernel is 0x200000. The Image size of root file system is less than 48Mbyte(0x3000000).

```

root@yongkal:~
File Edit View Terminal Tabs Help
Filename 'zImage'.
Load address: 0xc0000000
Loading: #####
#####
#####
#####
done
Bytes transferred = 1104492 (10da6c hex)
SMDK2443 # nand erase 40000 1c0000

NAND erase: device 0 offset 0x40000, size 0x1c0000
Erasing at 0x1fc000 -- 100% complete.
OK
SMDK2443 # nand write c0000000 40000 110000

NAND write: device 0 offset 0x40000, size 0x110000
1114112 bytes written: OK
SMDK2443 # tftp c0000000 Qtopial.7_demo_image.cramfs
TFTP from server 192.168.0.10; our IP address is 192.168.0.20
Filename 'Qtopial.7_demo_image.cramfs'.
Load address: 0xc0000000
Loading: #####
#####
#####
#####
#####
done
Bytes transferred = 43020288 (2907000 hex)
SMDK2443 # nand erase 200000 3000000

NAND erase: device 0 offset 0x200000, size 0x3000000
Erasing at 0x31fc000 -- 100% complete.
OK
SMDK2443 # nand write c0000000 200000 3000000

NAND write: device 0 offset 0x200000, size 0x3000000
50331648 bytes written: OK
SMDK2443 #
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.1 | VT102 | Offline

```

Figure 5-19 Transfer and Write Root File System Image

Once download all images to the NAND is completed, switch OFF the target board and switch it ON again. You can see “Linux Qtopia” booting on the target board, and the icons will be visible on the LCD screen of the target board.

5.5.4 Transfer and write image on extra area for JFFS2 and YAFFS2

From offset 0x3200000, we assigned extra area. You can use this area for jffs2, yaffs2 or other file systems. In this section, we explain how to write jffs2 and yaffs2 image on extra area.

5.5.4.1 JFFS2

```
SMDK○○○○ # tftp c0000000 hello.jffs2
SMDK○○○○ # nand erase clean 3200000
SMDK○○○○ # nand write.jffs2 c0000000 3200000 <actual size>
```

The length parameter calculated as remained size to end point automatically while using nand command. For instance, 'nand erase 3200000' is same with 'nand erase 3200000 <end - 3200000>'.

After booting the kernel, you can see the contents of extra area by mount. Use 'mount -t jffs2 /dev/mtdblock3 /tmp/test' after mkdir /tmp/test. Our extra area is assigned to /dev/mtdblock3 device node. Of course, you have to insert jffs2 module before kernel compilation.

5.5.4.2 YAFFS2

```
SMDK○○○○ # tftp c0000000 hello.yaffs2
SMDK○○○○ # nand erase 3200000
SMDK○○○○ # nand write.yaffs c0000000 3200000 <actual size>
```

In contrast with jffs2, you have to use only 'nand erase' to format extra area.

After booting the kernel, you can see the contents of extra area by mount. Use 'mount -t yaffs2 /dev/mtdblock3 /tmp/test' after mkdir /tmp/test. Our extra area is assigned to /dev/mtdblock3 device node. Of course, you have to insert yaffs2 module before kernel compilation.

If you are using 512 bytes small page size nand(for eg, Samsung K9F1208U0M, Olympus xD picture card 64M/128M), can't use yaffs2 file system. Maybe, will be selected to yaffs1 automatically.

5.6 oneNAND lock/unlock command

5.6.1 SMDK6410

```
SMDK6410 # onenand lock [on|off] <offset> <size>
```

For examples, if you want to lock up as many as 0x40000 at offset 3200000,

```
SMDK6410# onenand lock on 3200000 40000
```

On the other hand, to unlock the area above,

```
SMDK6410# onenand lock off 3200000 40000
```