1 ABOUT SMDK2440 BOARD

SYSTEM OVERVIEW

SMDK2440 (S3C2440 Development Kit) for S3C2440X is a platform that is suitable for code development of SAMSUNG's S3C2440X 16/32-bit RISC microcontroller (ARM920T) for hand-held devices and general applications.

The S3C2440X consists of 16-/32-bit RISC (ARM920T) CPU core, 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, IIC-BUS interface, IIS-BUS interface, USB host, USB device, SD host & multimedia card interface, Camera Interface 2-ch SPI and PLL for clock generation.

The SMDK2440 consists of S3C2440X, boot EEPROM (flash ROM), SDRAM, LCD interface, two serial communication ports, configuration switches, JTAG interface and status LEDs.

SMDK2440 OVERVIEW

The SMDK2440 (S3C2440 Development Kit) shows the basic system-based hardware design which uses the S3C2440X. It can evaluate the basic operations of the S3C2440X and develop codes for it as well.

When the S3C2440X is contained in the SMDK2440, you can use an in-circuit emulator (MULTI-ICE/OPENice32-A900).

This allows you to test and debug a system design at the processor level. In addition, the S3C2440X with MULTI-ICE/OPENice32-A900 capability can be debugged directly using the MULTI-ICE/OPENice32-A900 interface.

Figure 1-1 shows SMDK2440 function blocks.

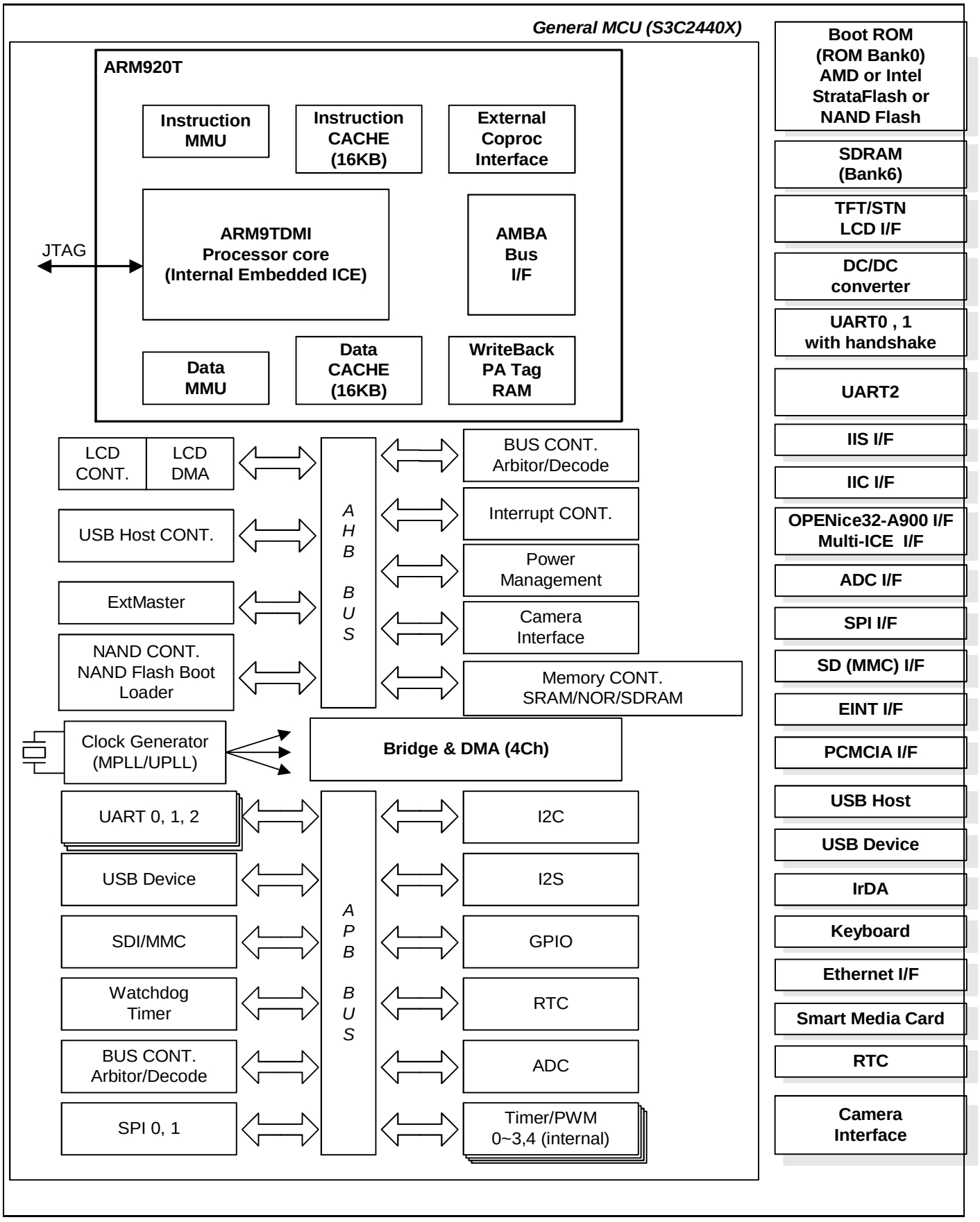


Figure 1-1. SMDK2440 Function Block Diagram

FEATURES

— S3C2440X: 16/32-bit RISC microcontroller

— X-tal operation or oscillator

— Boot ROM: AMD 8M bit 1EA (support half-word size boot ROM)
Intel StrataFlash 16M-byte x 2 (word: 16M-byte x 2 EA): Unload (Option)
SAMSUNG NAND flash 64M-byte 1EA (smart media card),
SAMSUNG NAND flash 64M-byte 1EA (sop type)

— SDRAM: 64M-byte (32M-byte x 2)

— SRAM: 256K x 16 – Unload(Option)

— TFT/STN LCD and touch panel interface

— Three-channel UART (including IrDA)

— One Host Type USB port & Selectable Device and Host Type USH port

— SD host (MMC) interface

— Smart media card

— JTAG port (MULTI-ICE/OPENice32-A900 interface)

— RTC X-tal input logic

— IIC with KS24C080

— ADC interface

— SPI interface

— IIS interface (sound CODEC audio input/output)

— EINT interface

— GPIO Switch Interface

— IrDA interface

— Ethernet interface

— PCMCIA interface

— Extension connector 34P * 3 EA

— LED display (debugging)

— CAMERA Interface

CIRCUIT DESCRIPTION

The SMDK2440 is designed to test S3C2440X and develop software while hardware is being developed. Figure 1-3 shows SMDK2440's block diagram.

POWER SUPPLY

SMDK2440 is operated by 1.2V for ARM core, 2.5V/3.3V for Memory and 3.3V for I/O pad and several peripherals. SMDK2440 is supplied by 9V/2A DC Adaptor Power.

The SMDK2440 has distributed power plane, with power going separately to the MCU and the main power plane. For this reason, power jumpers including J4-C~J15-C on the CPU board, J11-B, J12-B and J601-B on the base board are inserted.

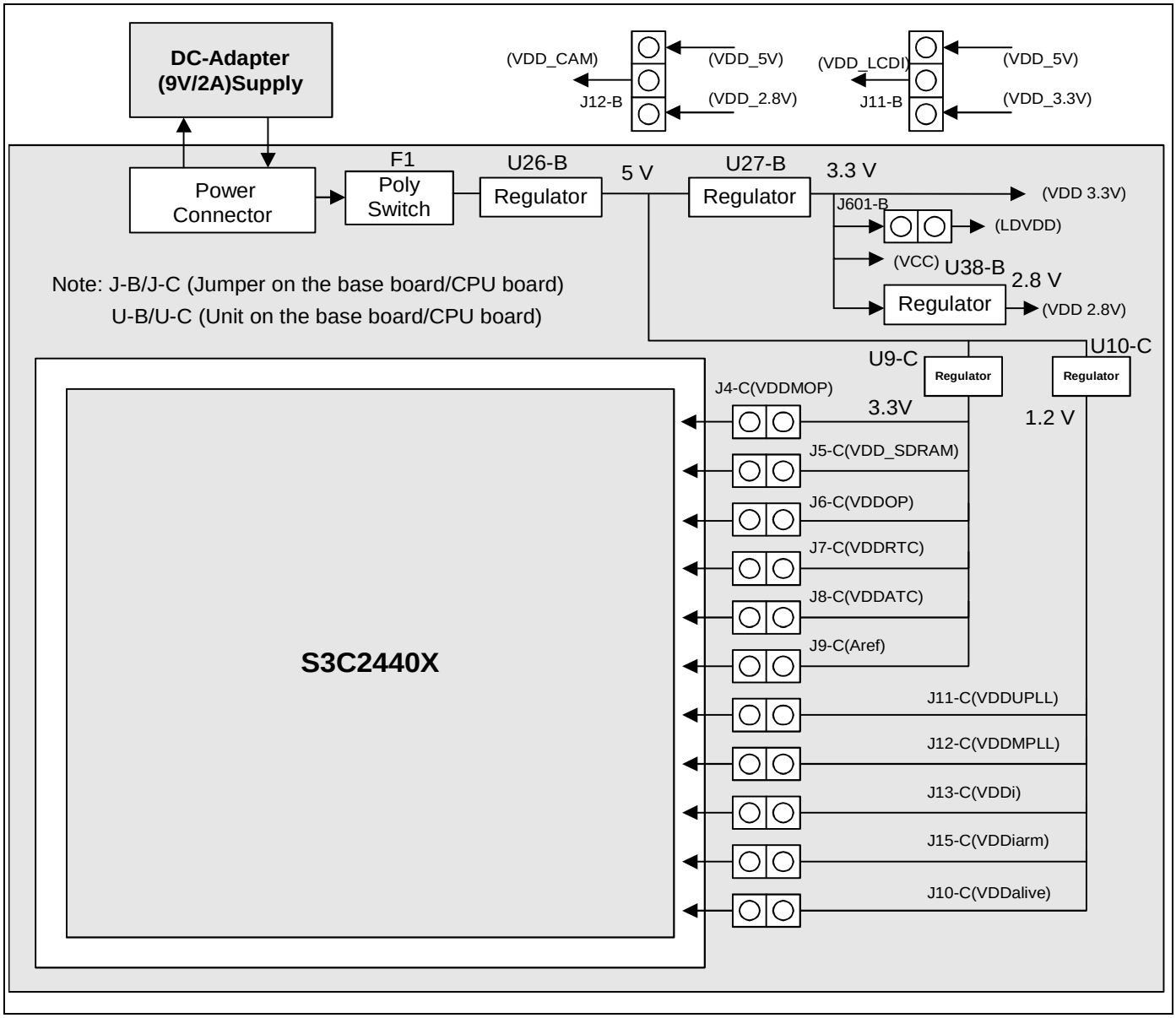


Figure 1-2. SMDK2440 Power Plane

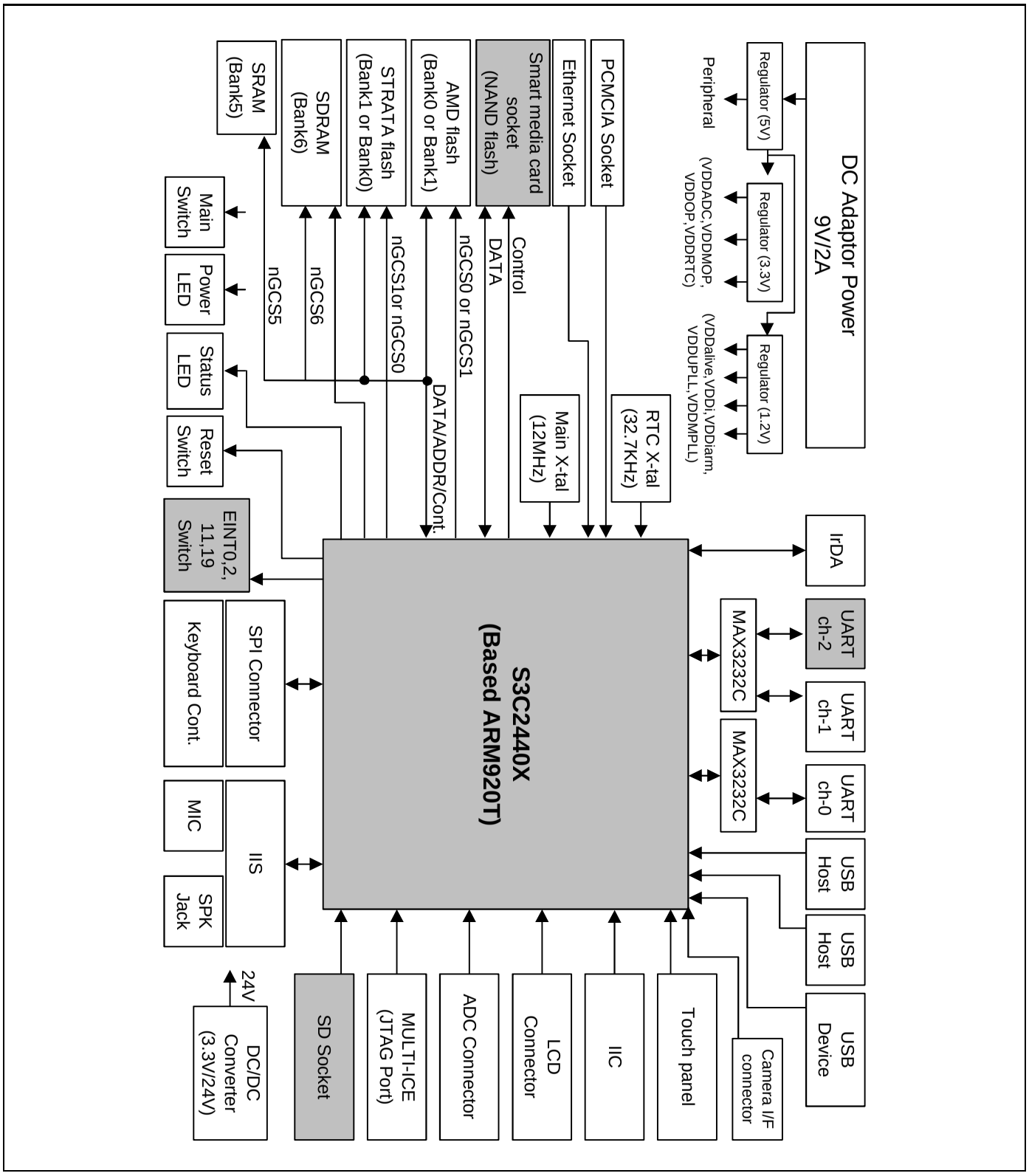


Figure 1-3. Detailed SMDK2440 Board Diagram

SMDK2440 SYSTEM CONFIGURATIONS

CLOCK SOURCE

EXTCLK or X-TAL can be selected for the system clock of S3C2440X and USB by the suitable setting of OM values.

Table 1-1. System Clock (MPLL) & USB Clock (UPLL)

PIN FUNCTIONS	OM[3:2]		DESCRIPTIONS
Clock source selection	0	0	MPLL: XTAL, UPLL: XTAL
	0	1	MPLL: XTAL UPLL: EXTCLK
	1	0	MPLL: EXTCLK, UPLL: XTAL
	1	1	MPLL: EXTCLK, UPLL: EXTCLK

RTC Clock

32.768KHz, X-tal is available in SMDK2440 as the RTC clock source.

NOTES:

1. Although the MPLL starts just after a reset, the MPLL output (Mpll) is not used as the system clock until the software writes valid settings to the MPLLCON register. Before this valid setting, the clock from external crystal or EXTCLK source will be used as the system clock directly. Even if the user wants to maintain the default value of the MPLLCON register, the user should write the same value into the MPLLCON register.
2. OM[3:2] is used to determine test mode when OM[1:0] is 11.

RESET LOGIC

The nRESET (system reset signal) must be held to low level at least 4 CLKs to recognize the reset signal and it takes 128 CLKs between the nRESET and internal nRESET. nRESET and nTRST (JTAG reset signal) are connected through jumper J35-C on the CPU board.

BOOT ROM (BANK0)

The data bus width of BANK0 can be configured in byte, half-word or word in S3C2440X.

In the case of SMDK2440, half-word data bus width (AMD flash memory), half-word or word data bus width (STRATA flash memory), and byte or half-word data bus width (Samsung NAND flash memory) access can be selected by the suitable jumper setting.

AMD flash or STRATA flash memory can be selected by using jumper (J3-B & J4-B) option for boot ROM.
In the SMDK2440, the data bus width of AMD flash memory is fixed by half-word data width (16-bit) and STRATA flash memory can use word (32-bit).

But AMD flash and STRATA flash cannot be selected for BANK0 or BANK1 at the same time.
Data bus width of BANK0 should be set by memory type of BANK0. It is set by OM[1:0](J2-B & J1-B).

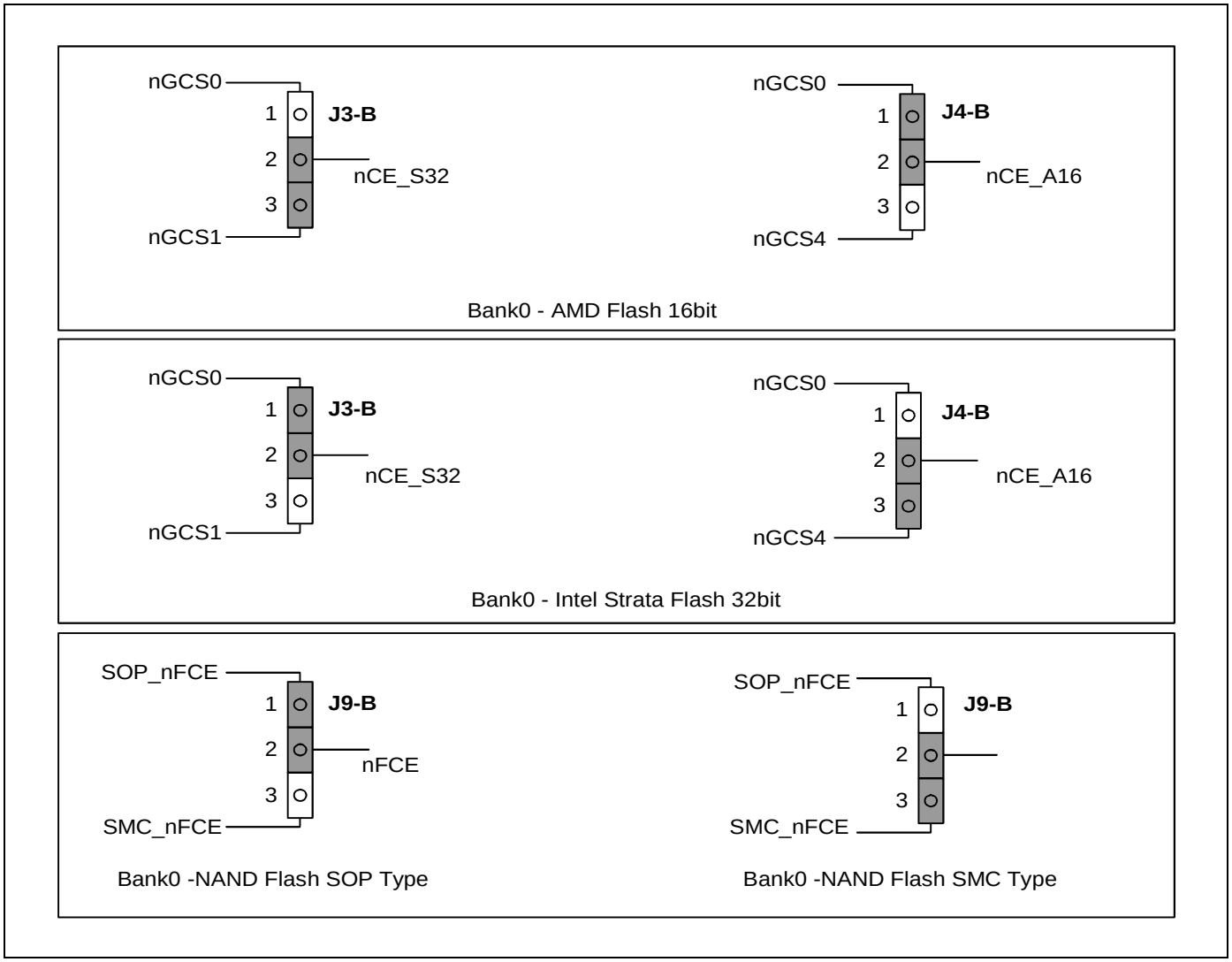


Figure 1-4. SMDK2440 Board Memory Configurations

Table 1-2. Memory Type and Data Bus Width

Pin Functions	J3-B		J4-B		Descriptions
	1-2	2-3	1-2	2-3	
BANK0/1/4 memory type selection and data bus width configuration	Open	Short	Short	Open	AMD flash memory : BANK0 STRATA flash memory : BANK1 Data bus width of BANK0 : Half-word Data bus width of BANK1 : Word
	Short	Open	Open	Short	AMD flash memory : BANK4 STRATA flash memory : BANK0 Data bus width of BANK0 : Word Data bus width of BANK4 : Half-word

Table 1-3. Boot Memory Type and Bus width configuration OM[1:0]

Pin Functions	J1-B [OM0]	J2-B[OM1]	Descriptions
Boot memory type and	2-3(L)	2-3(L)	NAND Boot
bus width configuration	2-3(L)	1-2(H)	Word (32-bit)
	1-2(H)	2-3(L)	Half Word (16-bit)
	1-2(H)	1-2(H)	Test Mode

NAND FLASH CONFIGURATION

Table 1-4. NAND Flash Type Selection

OM[1:0] = NAND Boot Setting (L, L)			
J5-B(NCON 0)		2-3(L)	1-2(H)
		Normal NAND	Advanced NAND
J6-B(PAGE)	2-3(L)	256	1024
	1-2(H)	512	2048
J7-B(ADDR)	2-3(L)	3 Cycle	4 Cycle
	1-2(H)	4 Cycle	5 Cycle
J8-B (WIDTH)	2-3(L)	8-bit Bus Width	
	1-2(H)	16-bit Bus Width	
J9-B (NAND Select)	2-3(L)	Use SMC NAND	
	1-2(H)	Use SOP NAND	

NOTE:

- Jumpers on the base board: J1-B, J2-B, J3-B, …

GENERAL I/O PORTS

The S3C2440X's general I/O ports are used for SMDK2440 key interrupt input, normal input and LED status display. The function of control switch and the status of LED can be defined by user software.

Table 1-5. General I/O Configurations on SMDK2440

General I/O Port Number	I/O Type	Descriptions
GPF[7:4]	Output	LED display
GPF0, GPF2, GPG3 & GPG11	Input	Key input pad (external interrupt input pins). (EINT0, 2, 11 & 19)

U4 (EPM7032) XDMA CHANNEL SELECTION

Table 1-6. U4-C XDMA Channel Selection

Pin Functions	J1-C	J2-C	Descriptions
XDMA channel selection	(1-2)	(1-2)	nXDREQ0, nXDACK0
	(2-3)	(2-3)	nXDREQ1, nXDACK1

NOTE:

- Jumpers on the CPU board: J1-C, J2-C, J3-C, …

LCD INTERFACE

TFT/STN LCD controllers are equipped in the S3C2440X. TFT/STN LCD, touch panel and LCD backlight driver are supported in the SMDK2440.

NOTES:
It is supported 2-type SEC TFT LCD panel(SAMSUNG 3.5” Portrait/256 Color/Reflective a TFT LCD)

LTS350Q1-PD1 Panel with touch panel and front light unit

LTS350Q1-PD2 Panel only

LTS350Q1-PE1 Panel with touch panel and front light unit

LTS350Q1-PE2 Panel only

TOUCH SCREEN

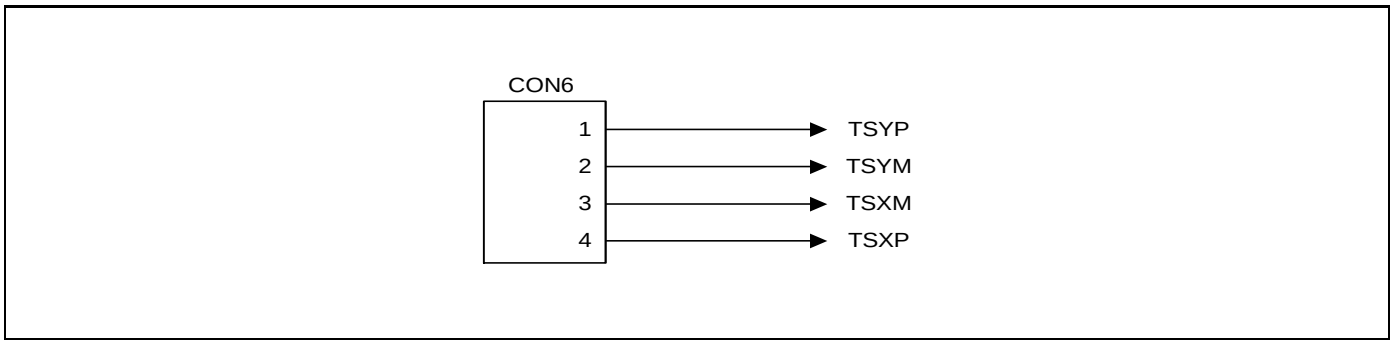


Figure 1-5. Touch Panel Film Connector on SMDK2440

SPI Connector

Figure 1-6 shows the way SMDK2440 provides SPI (CON15) signals.

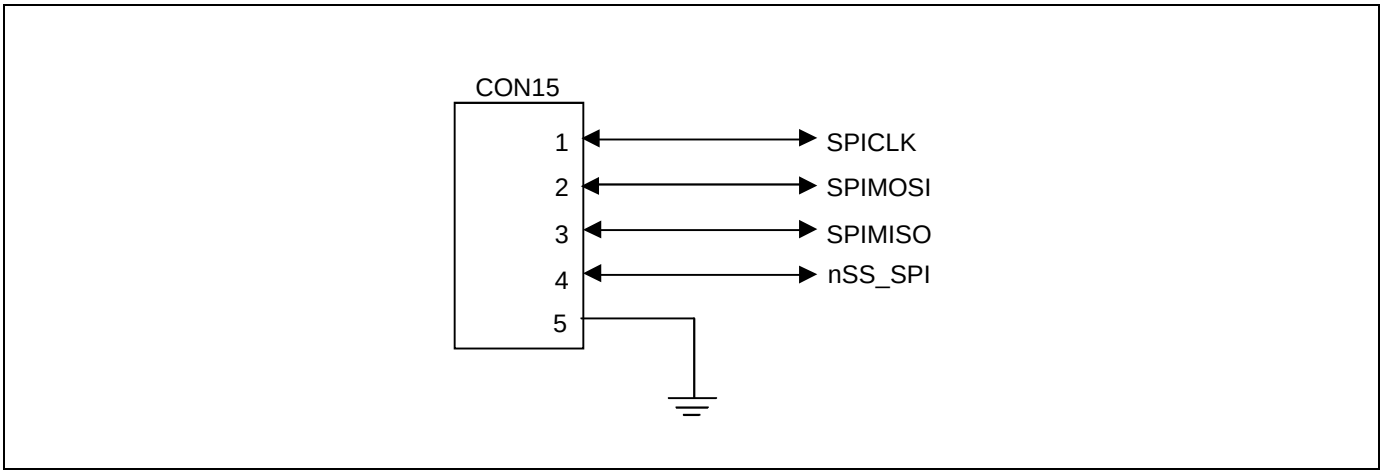


Figure 1-6. SPI Connector on SMDK2440

A/D CONVERTER INTERFACE

The S3C2440X has Analog to Digital Converter (ADC). The ADC has 8-ch analog input signals.
The SMK2440 provides the ADC (CON8) signals as follows:

Table 1-7. ADC Interface on SMDK2440

# of pin	Descriptions	# of pin	Descriptions	# of pin	Descriptions	# of pin	Descriptions
1	AIN0	**4**	AIN3	**7**	TSXM	**10**	GND
2	AIN1	**5**	TSYM	**8**	TSXP		
3	AIN2	**6**	TSYP	**9**	EINT20		

SD HOST (MMC) INTERFACE

SD(MMC) is provided by the S3C2440X and SD card socket (CON13) is supported in the SMDK2440

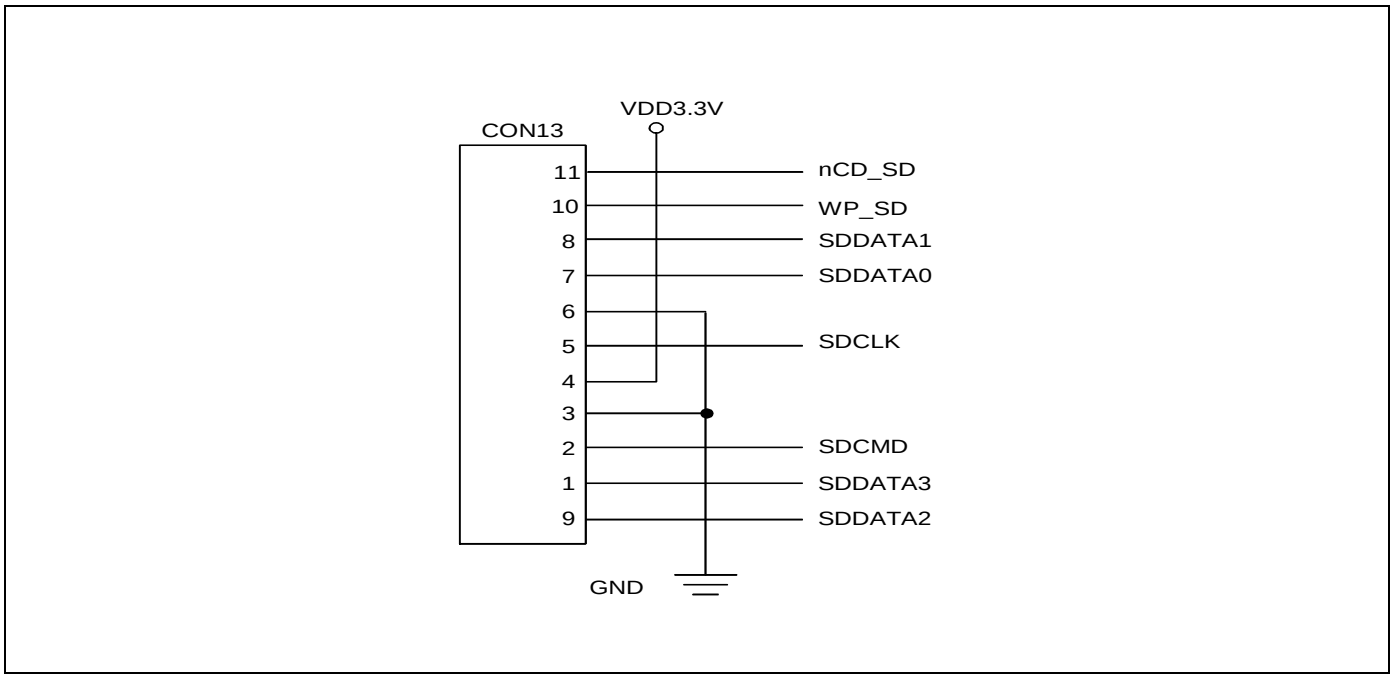


Figure 1-7. SD Card Socket on SMDK2440

IIC INTERFACE

Serial EEPROM S524C80D80 (KS24C080) access function is provided by SMDK2440 and there is also IIC interface between S3C2440X and camera module through U37-B(CBTD3306) buffer.

USB INTERFACE

Dual USB Connector(CON3) for Two USB port A-Type and one USB port B-type(CON5) are supported by the SMDK2440.

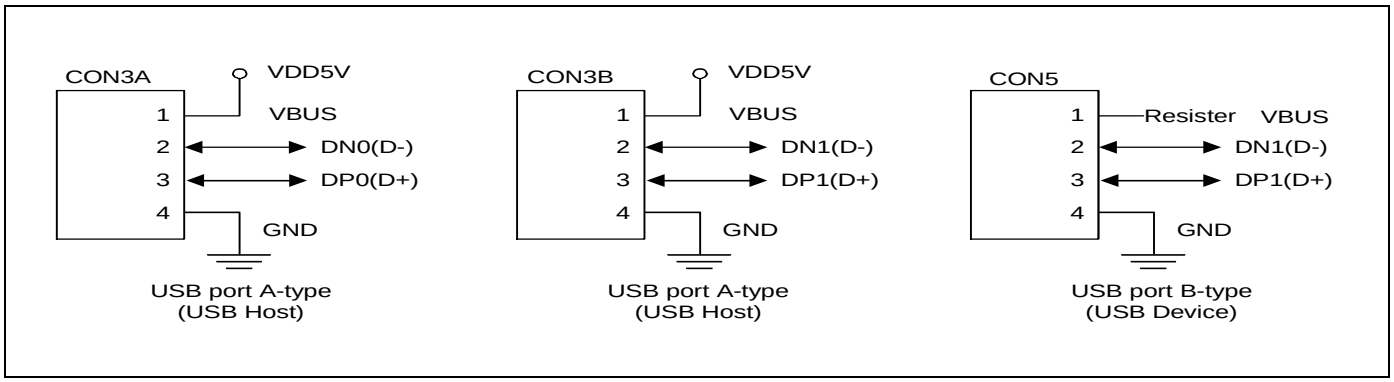


Figure 1-8. USB Ports on SMDK2440

You Can be select the USB port 1 (DN1, DP1) by Jumper (J14-C, J16-C)

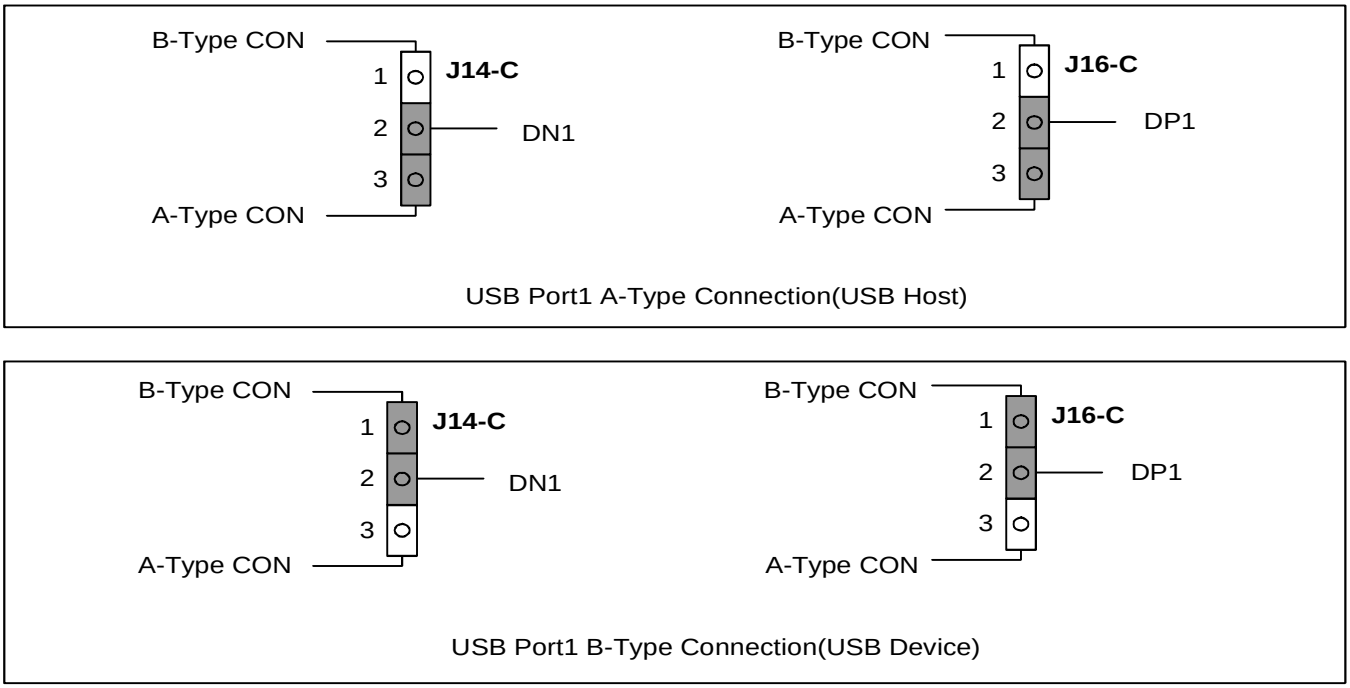


Figure 1-9. USB Port1 Selection

UART INTERFACE

The S3C2440X UART unit provides three independent asynchronous serial I/O (SIO) ports including IrDA. In SMDK2440 board, a user can change the ports connected to connectors by setting related jumpers.

Table 1-8. UART Configurations

Pin Functions	J16-B, J18-B	J17-B, J19-B	Descriptions
UART configurations	(2-3)	(1-2)	CON14: UART0, CON22: UART1
	(1-2)	(2-3)	CON14: UART0, CON22: UART2

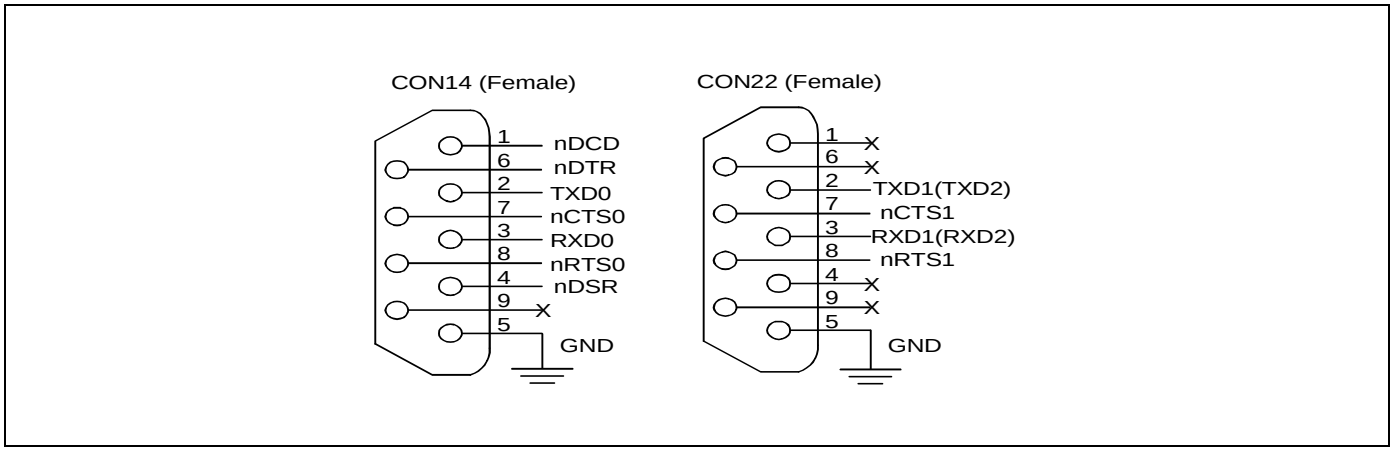


Figure 1-10. UART Ports on SMDK2440

IrDA INTERFACE

IrDA is supported by SMDK2440 and J17-B and J19-B should be set to UART2 (RXD2 and TXD2) for IrDA.

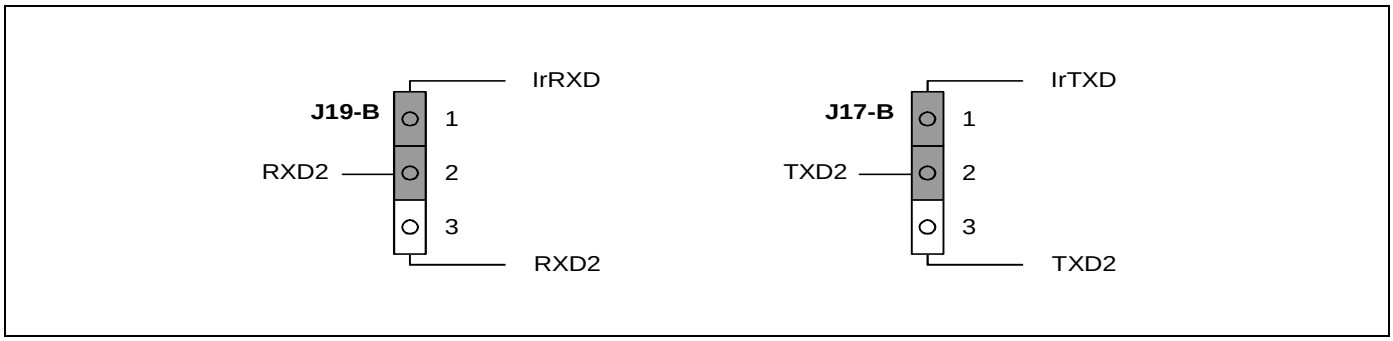


Figure 1-11. SMDK2440 Board IrDA Configurations

Table 1-9. IrDA Configurations

Pin Functions	J17-B, J19-B	Descriptions
UART2	(2-3)	Set UART mode
IrDA	(1-2)	Set IrDA mode

EXTENSION CONNECTOR INTERFACE

Table 1-10. Extension Connector (CON10, CON11 & CON12) on SMDK2440

# of pin	Descriptions	# of pin	Descriptions	# of pin	Descriptions	# of pin	Descriptions
1	GND	**10**	DATA8	**19**	DATA17	**28**	DATA26
2	DATA0	**11**	DATA9	**20**	DATA18	**29**	DATA27
3	DATA1	**12**	DATA10	**21**	DATA19	**30**	DATA28
4	DATA2	**13**	DATA11	**22**	DATA20	**31**	DATA29
5	DATA3	**14**	DATA12	**23**	DATA21	**32**	DATA30
6	DATA4	**15**	DATA13	**24**	DATA22	**33**	DATA31
7	DATA5	**16**	DATA14	**25**	DATA23	**34**	–
8	DATA6	**17**	DATA15	**26**	DATA24	–	–
9	DATA7	**18**	DATA16	**27**	DATA25	–	–

# of pin	Descriptions	# of pin	Descriptions	# of pin	Descriptions	# of pin	Descriptions
1	GND	**10**	A8	**19**	A17	**28**	nWBE0
2	A0	**11**	A9	**20**	A18	**29**	nWBE1
3	A1	**12**	A10	**21**	A19	**30**	nWBE2
4	A2	**13**	A11	**22**	A20	**31**	nWBE3
5	A3	**14**	A12	**23**	A21	**32**	nWE
6	A4	**15**	A13	**24**	A22	**33**	nOE
7	A5	**16**	A14	**25**	A23	**34**	–
8	A6	**17**	A15	**26**	A24	–	–
9	A7	**18**	A16	**27**	nWAIT	–	–

# of pin	Descriptions	# of pin	Descriptions	# of pin	Descriptions	# of pin	Descriptions
1	nGCS2	**10**	GPG7	**19**	nXDACK1	**28**	GND
2	nGCS1	**11**	GPG2	**20**	nXDREQ0	**29**	GND
3	nGCS4	**12**	GPG8	**21**	GPG5	**30**	nRESET
4	nGCS3	**13**	GPG3	**22**	nXDREQ1	**31**	VDD5V
5	nGCS7	**14**	GPG9	**23**	VDD1.8V	**32**	VDD3.3V
6	nGCS5	**15**	GPG4	**24**	GPG12	**33**	CLKOUT1
7	GPG0	**16**	GPG10	**25**	GND	**34**	GND
8	GPG6	**17**	nXDACK0	**26**	CLKOUT0	–	–
9	GPG1	**18**	GPG11	**27**	VDD3.3V	–	–

CAMERA INTERFACE CONNECTOR

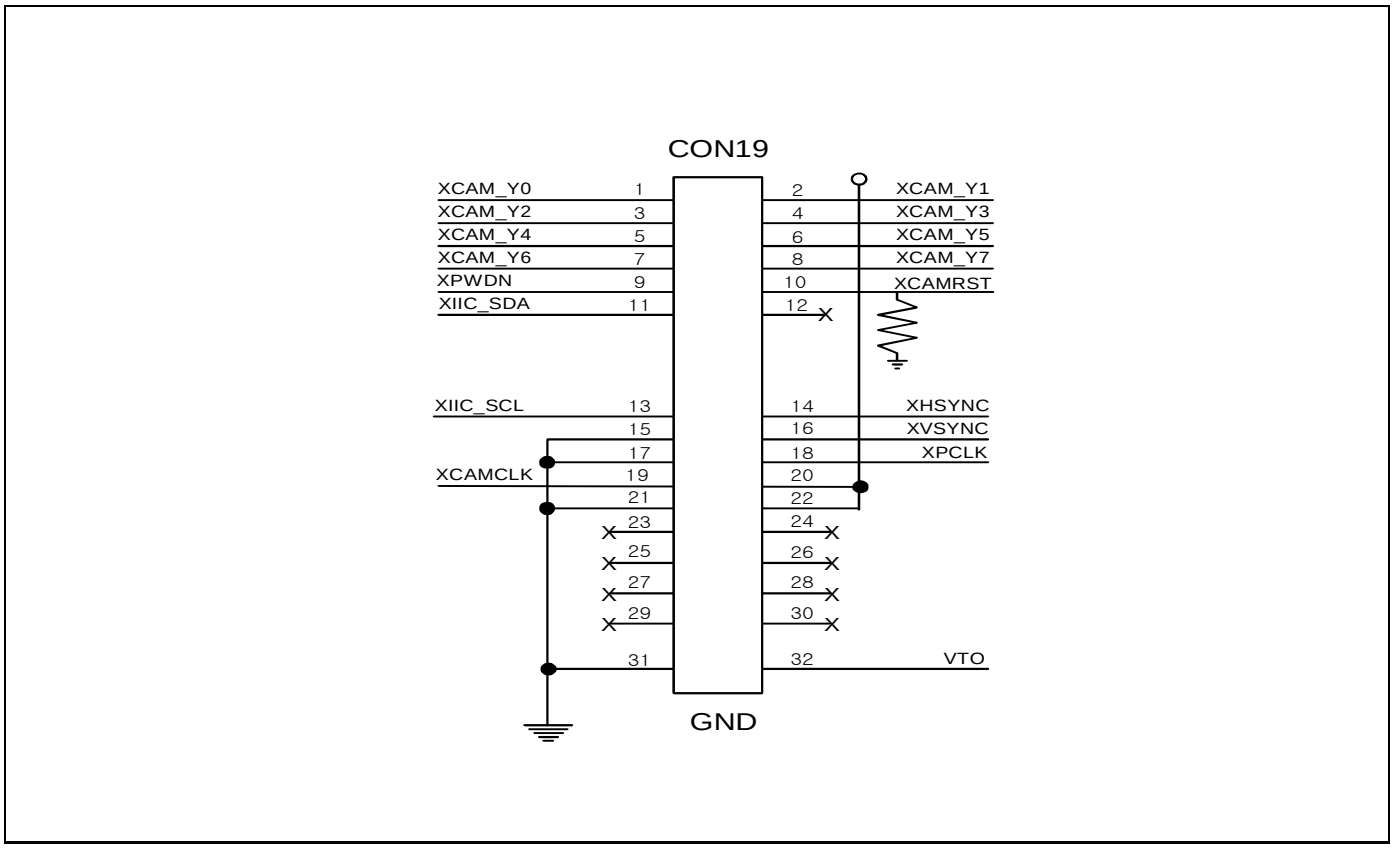


Figure 1-12. CAMERA Interface Connector on SMDK2440

NOTES

2 TOOLKIT AND DEBUGGING

SMDK2440 ENVIRONMENT SETUP

The evaluation environments for the SMDK2440 are shown in Figure 2-1. The serial port (UART1) on the SMDK2440 has to be connected to COM port of the host PC. This can be used as a console for monitoring and debugging the SMDK2440. And the USB device on the SMDK2440 should be connected to the USB host of the host PC for downloading test images.

If you have an emulator such as MULTI-ICE and OPENice32-A900, you can use JTAG port on the SMDK2440 to interface the emulator.

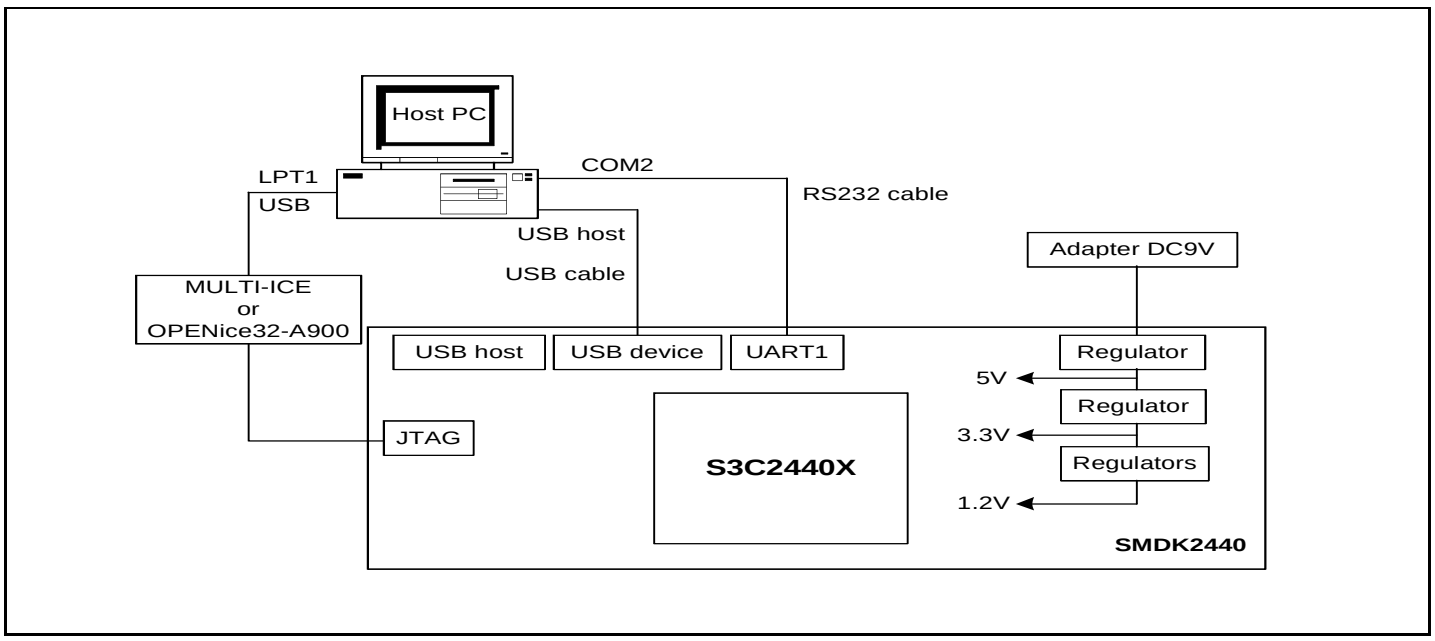


Figure 2-1. Setup Environment for SMDK2440 Board

RS232C CABLE CONNECTION

The serial cable is made as in Figure 2-2. The pins numbered only 2, 3, and 5 are used; make sure to check the cable's connections to prevent other pins from being used.

The UART1(CON22) and PC COM1 or COM2 port has to be connected through this cable connection.

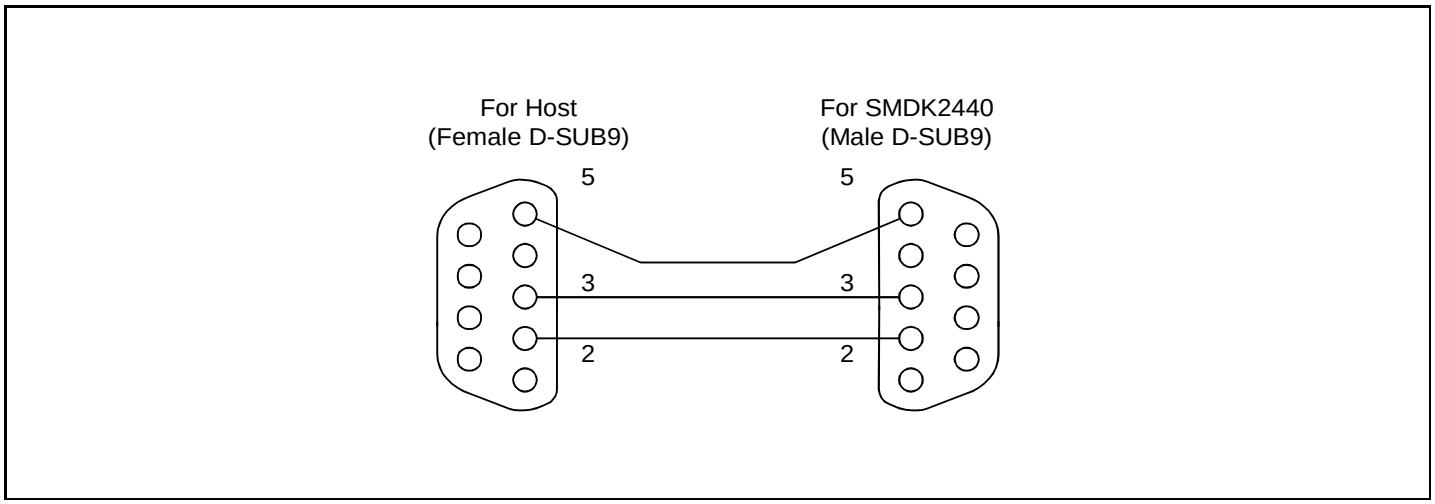


Figure 2-2. Serial Cable Connections for SMDK2440 Board

USB DOWNLOADER INSTALLATION (ON WINDOWS 98, ME, 2000 OR NT)

To install the USB downloader, follow the steps:

1. Program the u2440mon.bin into the flash memory of SMDK2440X board.
2. Configure Boot Jumper Setting (J1-B ~ J9-B).
3. Turn on the SMDK2440.
4. If you installed the device driver for SMDK2400X/SMDK2440X before, overwrite new ‘secbulk.sys’ at C:\WINDOWS\SYSTEM32\DRIVERS. In this case, the step 6 will be skipped.
5. Connect the SMDK2440X board with the PC (See Figure 2-3).
6. When the USB device driver installation window appears, install the USB device driver (secbulk.inf). Note: ‘secbulk.inf’ and ‘secbulk.sys’ should be in the same directory (See Figure 2-4).
7. Run ‘dnw.exe’.
8. Turn the SMDK2440 off and then on.
9. The message ([USB:OK]) in the window title bar indicates that the installation is successfully completed.

Figure 2-3. Add New Hardware Wizard (Window98)

NOTES:
1. If you have installed the device driver before, replace the old 'secbulk.sys' in C:\WINDOWS\SYSTEM32\DRIVERS with the new 'secbulk.sys'.
2. The maximum speed of the 'secbulk.sys' with SMDK2440 will be about 980KB/S.
3. 'dnw.exe': PC USB/serial downloader program.
4. 'secbulk.inf' and 'secbulk.sys': PC USB driver.
5. 'u2440mon.bin': S3C2440X USB downloader firmware.

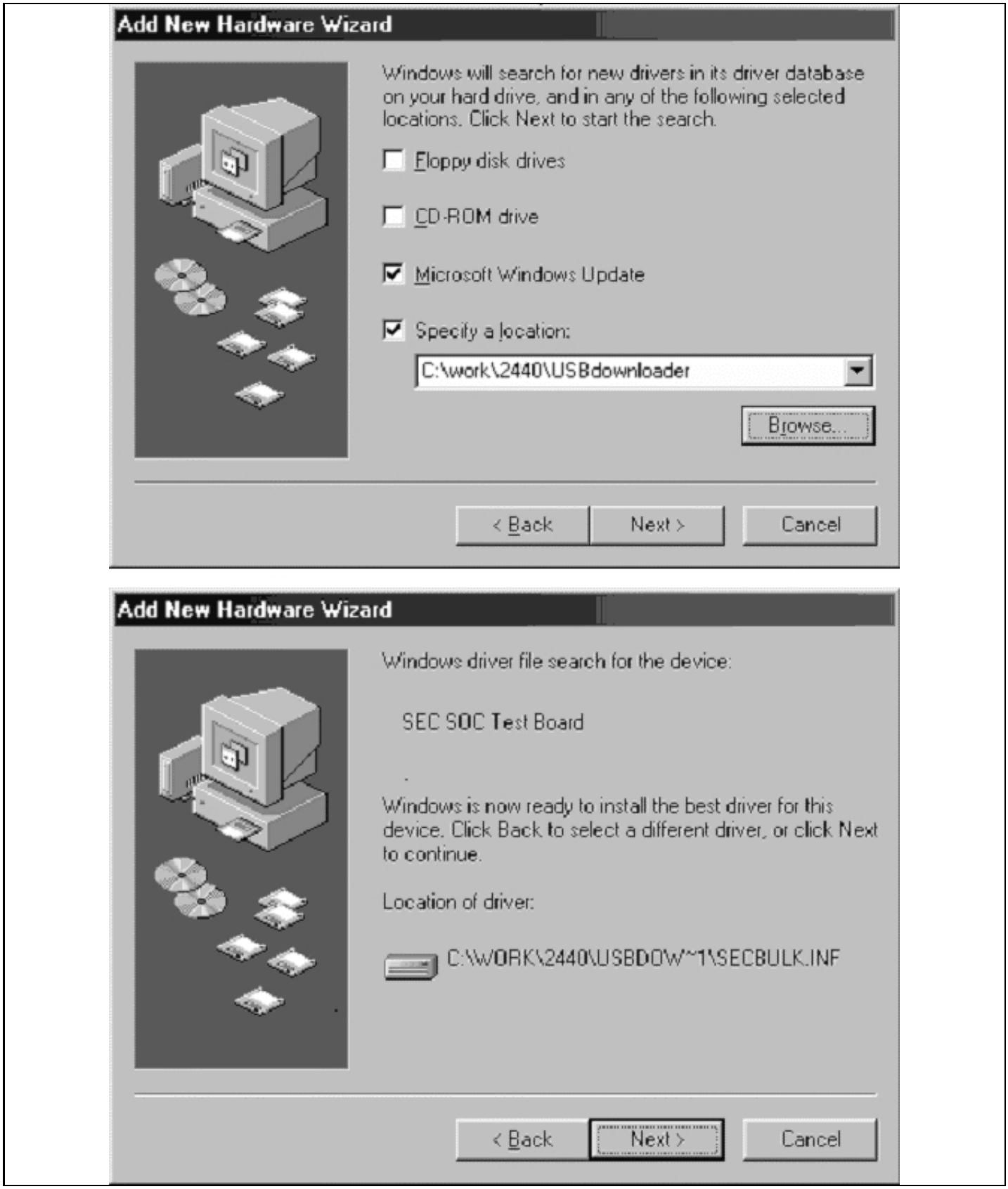


Figure 2-4. USB Device Driver Installation

Figure 2-4. USB Device Driver Installation (Continued)

CONFIGURING DNW

To configure the DNW, which works as USB and serial download utility, follow the steps:

1. Run the DNW.
2. Select Options from the Configuration menu (See Figure 2-5).
   Configuration → Options
3. Select Baud rate for serial communication.
   Serial communication properties of the DNW are as follows:
   Data bits:8-bit / Stop bits:1 / No flow control
4. Select a COM port of the host PC to communicate with the SMDK2440.
5. Set USB download address.
6. Click the OK button.

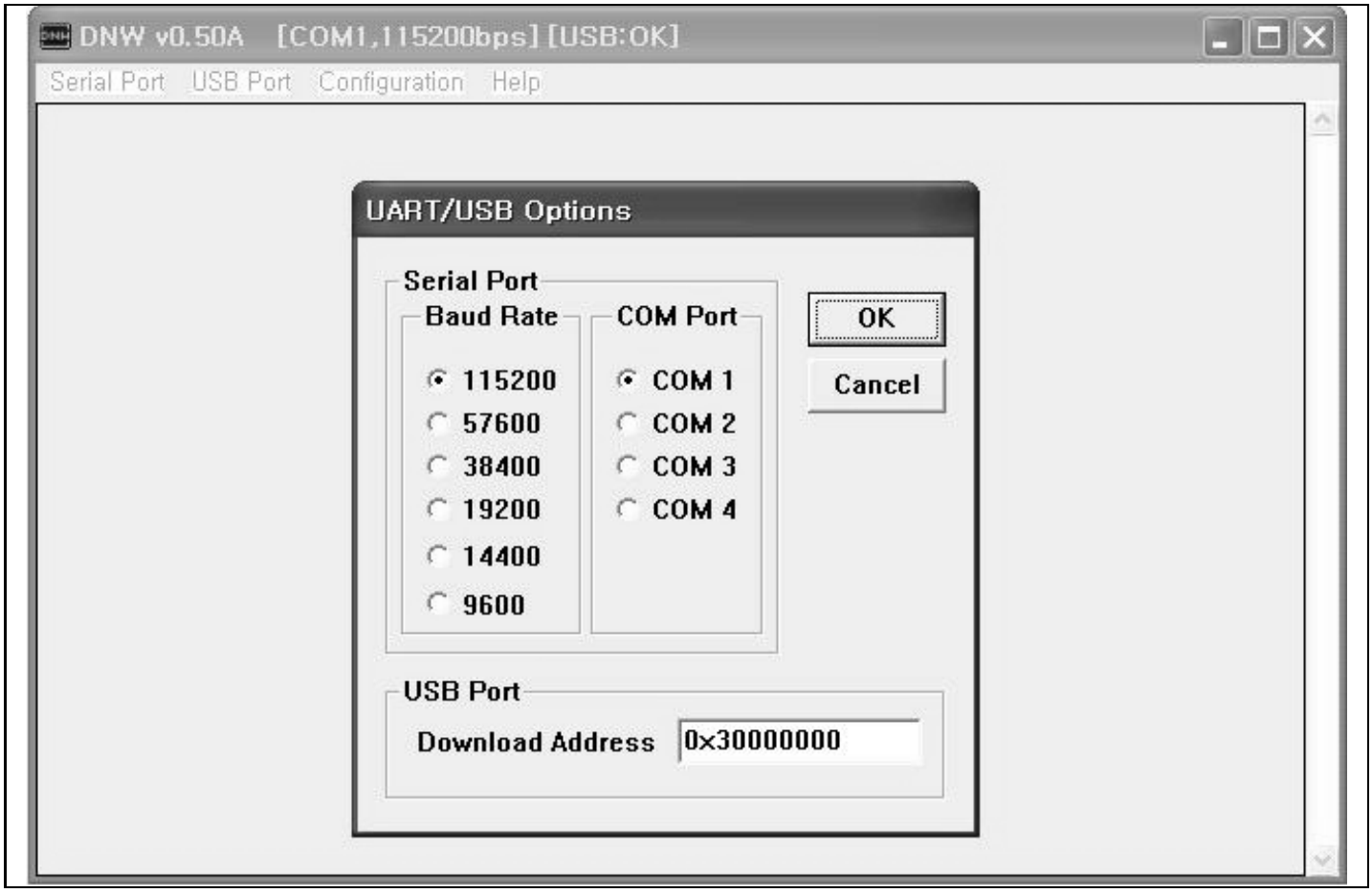


Figure 2-5. Setting UART/USB Options

CONNECT HOST PC AND SMDK2440 WITH DNW

After setting UART/USB options, users can activate UART and USB communication.

1. Select Connect from the Serial Port menu.
   Serial Port → Connect
2. Power on the SMDK2440 (See Figure 2-6).

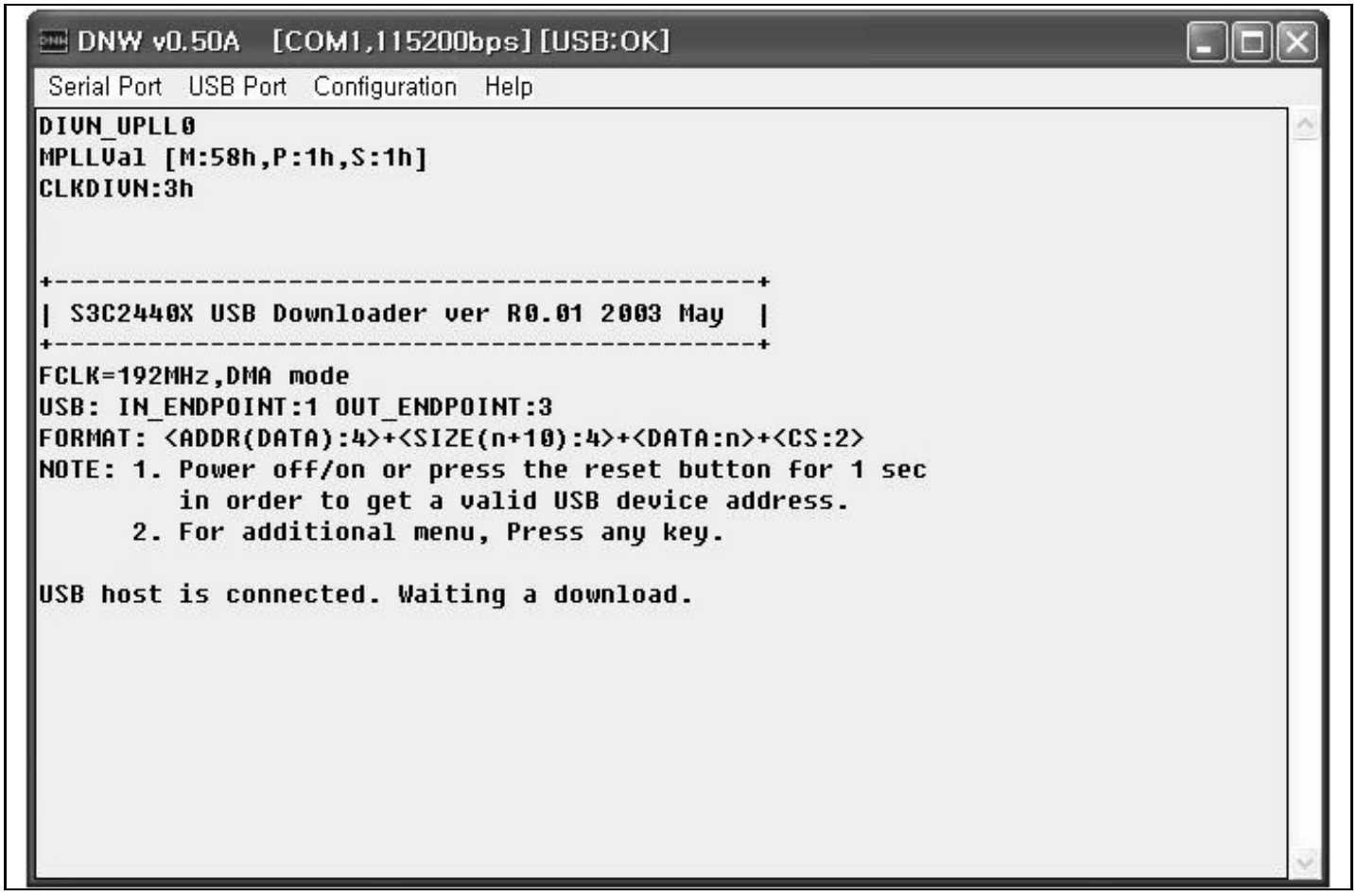


Figure 2-6. Power On Screen

INSTALL ARM TOOLKIT

First of all, install ARM toolkit 2.51, ADS (ARM Developer Suite) 1.0.1 or ADS1.1.

If you installed ARM toolkit 2.11a, then Makefile has to be changed a little. The toolkit 2.11a cannot support fromelf.exe utility. We recommend ARM Developer Suite 1.0.1, which is used in our development environment. We also recommend ADS 1.1.

The DOS environment variable has to be changed as follows after the installation of ARM toolkit 2.51.

SET ARMLIB=C:\ARM251\LIB\embedded
SET ARMINC=C:\ARM251\INCLUDE

HOW TO BUILD EXECUTABLE IMAGE FILE

Executable image file can be built by using the ARM Project manager or makefile. First, you have to build ELF format image (*.ELF or *.AXF). An ELF format image can be used for the ARM debugger directly. The binary file (.bin file) can be extracted from ELF format image.

First of all, you have to download S3C2440X evaluation source code and any other utilities from our web site (www.samsungsemi.com). They are helpful for you to understand the development environments of S3C2440X in an easier way. The distributed evaluation source code consists of following directories.

Directory	Description
BMP	Graphic header file converted from BMP file
obj	Object files
err	Error files

BUILDING 2440TEST.AXF (OR 2440TEST.ELF)

To build the sample source code, 2440TEST, run Makefile using the following commands.

cd 2440Test
armmake –a

or

cd 2440Test
make clean
make

After the procedure, 2440TEST.AXF (or 2440TEST.ELF) and 2440TEST.BIN image files will be seen in 2440TEST directory. The 2440TEST.AXF (or 2440TEST.ELF) file is used for ARM debugger.

The 2440TEST.BIN file is used for downloading through USB.

EXECUTING 2440TEST WITHOUT ARM MULTI-ICE OR OPENICE32-A900

First, U2440MON has to operate on ROM. U2440MON will be ready to receive 2440TEST.BIN. U2440MON will launch 2440TEST.BIN after receiving 2440TEST.BIN.
To download 2440TEST.BIN through USB, after connecting the host PC and the SMDK2440 with the DNW follow the steps below:

1. Select Transmit from the USB Port menu.
   USB Port → Transmit
2. Select 2440TEST.BIN (See Figure 2-7).

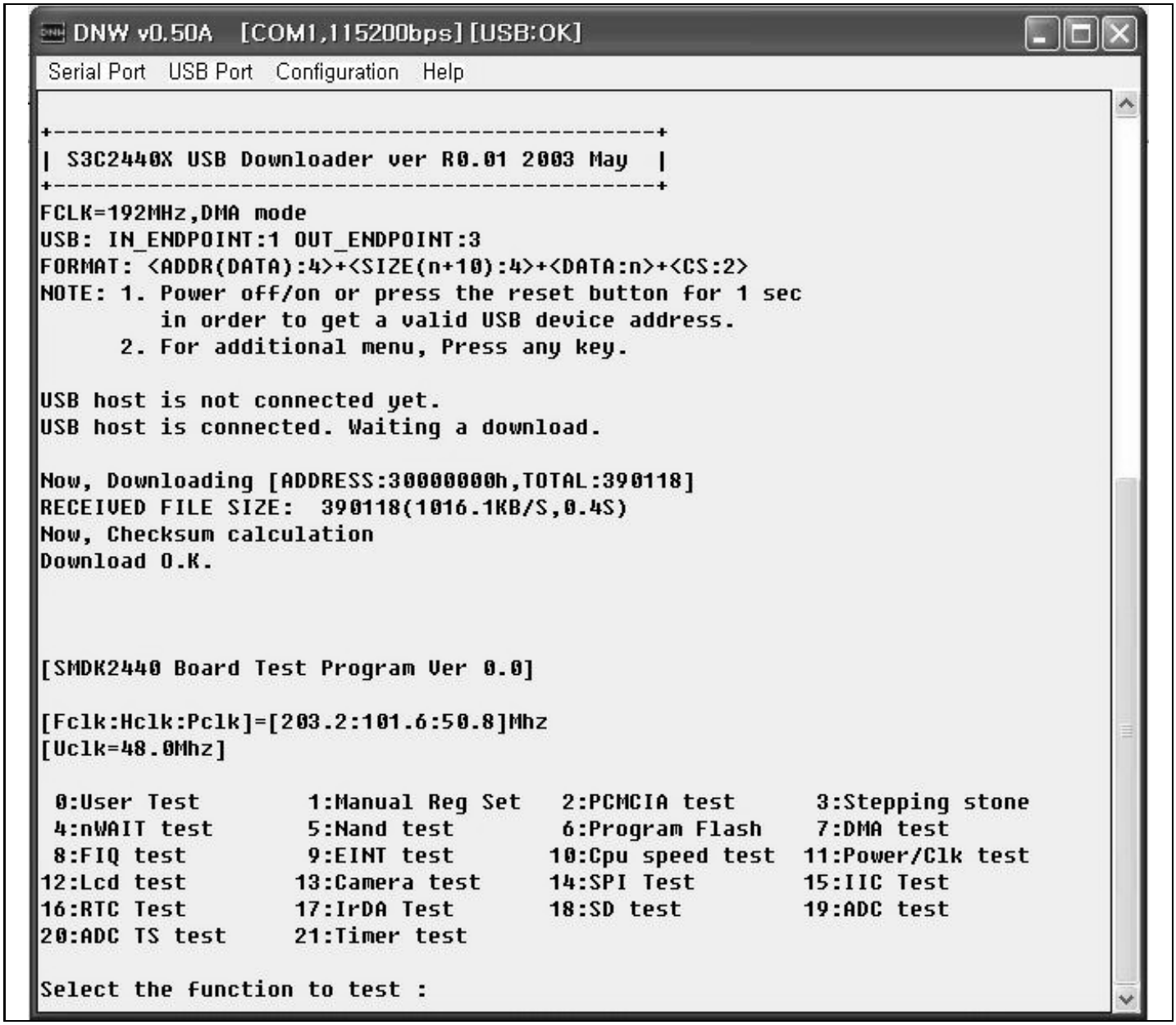


Figure 2-7. 2440TEST Execution After its Downloading through USB

HOW TO USE ARM DEBUGGER WITH ARM MULTI-ICE

If you have built 2440TEST program without any error, you can find 2440TEST.AXF in 2440TEST directory. The generated image file will be downloaded to SDRAM memory on the SMDK2440 by ARM debugger through MDS like a MULTI-ICE. Next, you can start to debug the downloaded image using the ADW (ARM Debugger for Windows).

PREPARING AND CONFIGURING ARM MULTI-ICE

1. MULTI-ICE will be connected through JTAG port on the board. Connect all cables properly following its manual.
2. Start the ARM MULTI-ICE Server (Double click the MULTI-ICE Server icon).
3. Select Load Configuration from File menu and load 2440.CFG (See Figure 2-8).

   File → Load Configuration
4. Contents of 2440.CFG are as follows:

   [TITLE]
   S3C2440/S3C2440 TAP Configuration

   [TAP 0]
   ARM920T

   [Timing]
   Adaptive=OFF
5. Select Start-up Options from Settings menu.

   Settings → Start-up Options
6. Start-up Options dialog box is displayed (See Figure 2-9).

   Now you can select Load Configuration from Start-up Configuration and browse 2440.CFG
7. Start ARM Debugger using ARM debugger icon

   Also, you can start the debugger at DOS command window by typing adw 2440TEST.AXF.
   If you use ARM MULTI-ICE for the first time, you have to add Multi-ICE.DLL to ADW.
8. When ARM Debugger is started, it will load the image code to the Armulator (The Armulator is software emulator for ARM920T CPU).

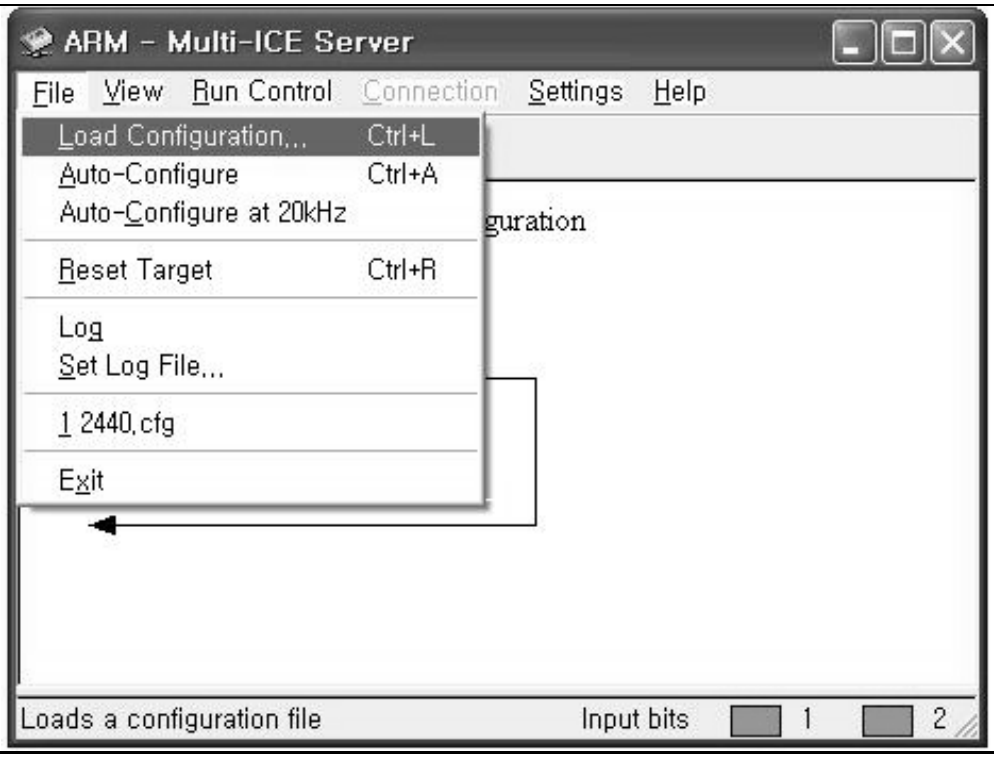


Figure 2-8. Load Configuration

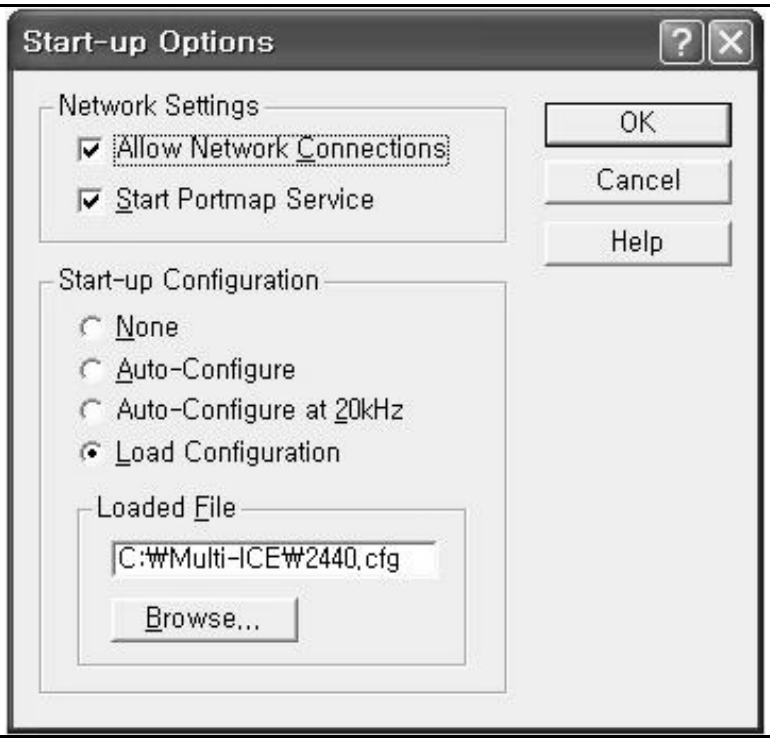


Figure 2-9. Start-up Configuration

CONFIGURING ARM DEBUGGER FOR ARM MULTI-ICE

In order to access a remote target, you should configure ARM Debugger for Windows (ADW) or ARM Extended Debugger(AxD). There are two kinds of ADW: one for Software Development Toolkit (SDT) and the other for ARM Developer Suit (ADS). These two kinds of ADW are basically same except some trivial differences. The below explanation will be described with AXD of ARM Developer Suit (ADS).
The MULTI-ICE interface unit must also be configured for the ARM core in the target system. The ARM920T core is contained in the S3C2440X on the SMDK2440 board.

To configure AXD Debugger using the MULTI-ICE interface, follow the steps:

1. Select Configure Debugger from the Options menu.

   Options -> Configure Target

2. Debugger Configuration dialog box is displayed (See Figure 2-10).
   If there is no Multi-ICE in the target environment, then you have to select Add button and Multi-ICE.DLL.
   — ARMulator: lets you execute the ARM program without any physical ARM hardware by simulating ARM instructions in software.
   — Multi-ICE: connects the AXD Debugger directly to the target board or to a MULTI-ICE unit attached to the target.
3. Select Multi-ICE from Target environment, and click the Configure button.
4. Configure ARM Multi-ICE dialog box (See Figure 2-11).
   — Connect page: select your host and MULTI-ICE communication target configuration.
   — Processor Settings page: set the cache clean code address.
5. Select Advanced from Debugger Configuration dialog box (See Figure 2-12) and configure it.
   — Endian: little (If the big endian is used, Endian: big has to be selected.)
6. If you click the OK button on Debugger Configuration dialog box, the debugger will be restarted. The restarting dialog box is displayed and numbers are rapidly changing, indicating that it is reading and writing to the target. This means that the executable image file is downloaded to the SDRAM code area.

This configuration is initially done and the setting is saved, which relieves the user of repeating another configuration next time.

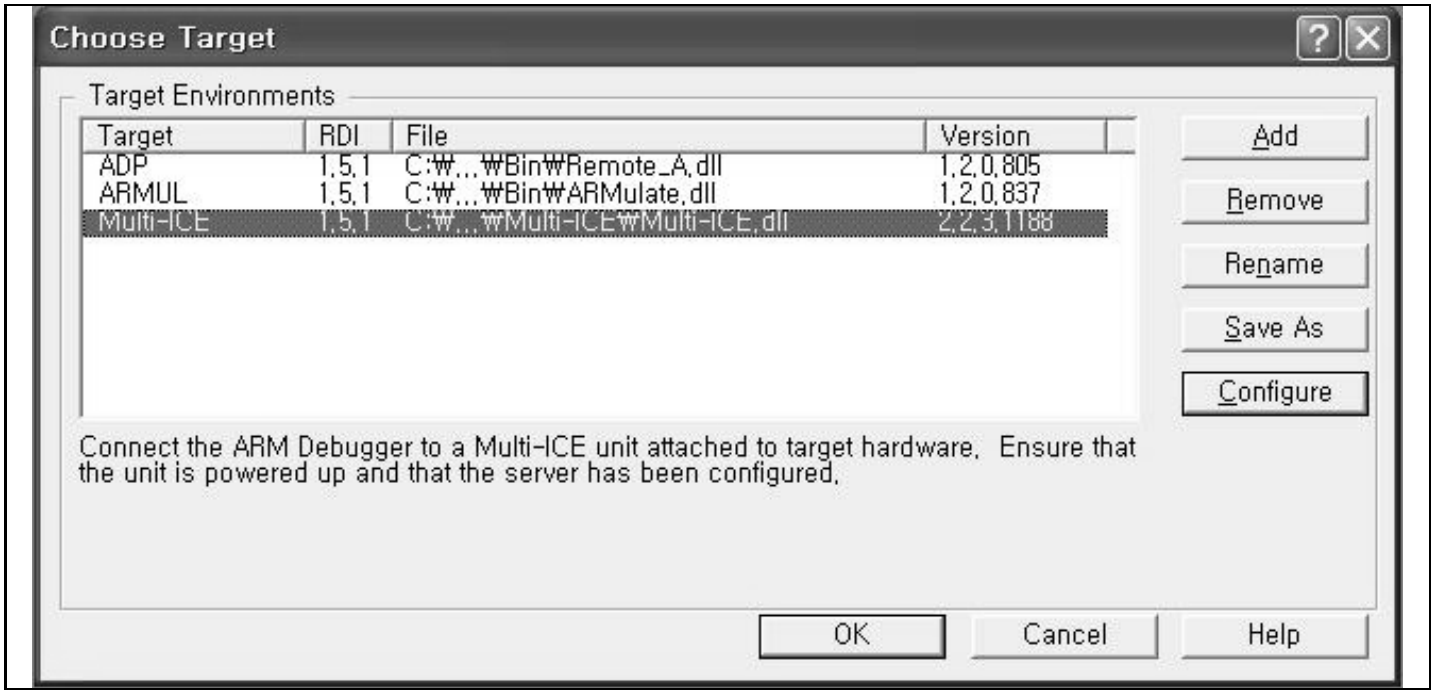


Figure 2-10. Debugger Configuration: Target Page

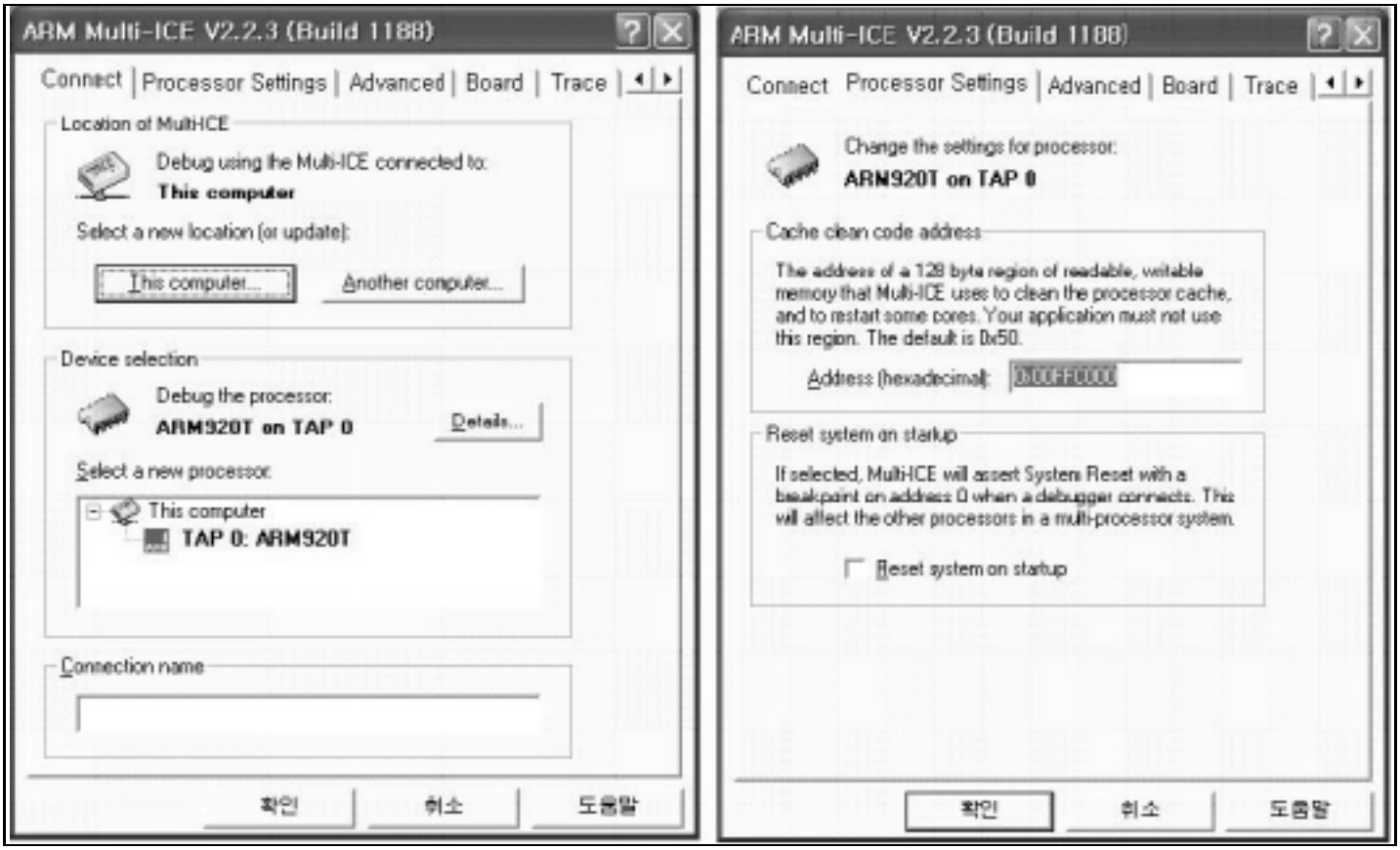


Figure 2-11. ARM Multi-ICE: Connect Page and Processor Settings Page

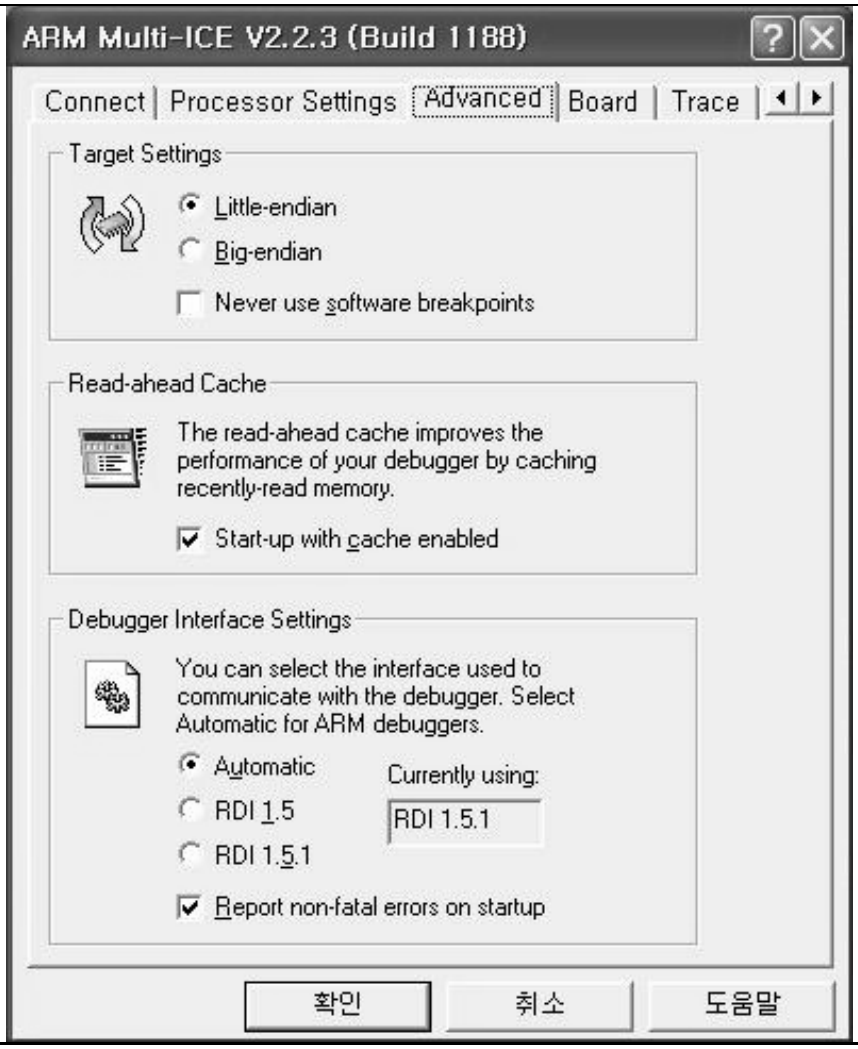


Figure 2-12. Debugger Configuration: Advanced Page

EXECUTING 2440TEST.AXF USING ARM MULTI-ICE

1. Initialize internal variables of the debugger. After a downloading, several windows are displayed, such as Execution window, Console window, and Command window. In Command window, you should initialize the internal variables of the debugger, "$semihosting_enabled" and "$vector_catch", by entering the following command:

```
swat $vector_catch   0x00
swat $semihosting_enabled   0x00    ;To use all H/W break points
swat psr %IFt_SVC                   ;To disable all interrupts
com swat psr %IF_SVC32
```

Or, you can initialize these variables as follows:

First, create a text file named "2440norom.ini", which includes the commands described above. Then, enter the following command in the Command window (See Figure 2-13):

obey C:\WORK\2440\2440norom\2440norom.ini

For more information about these steps, refer to the reference document released by ARM.

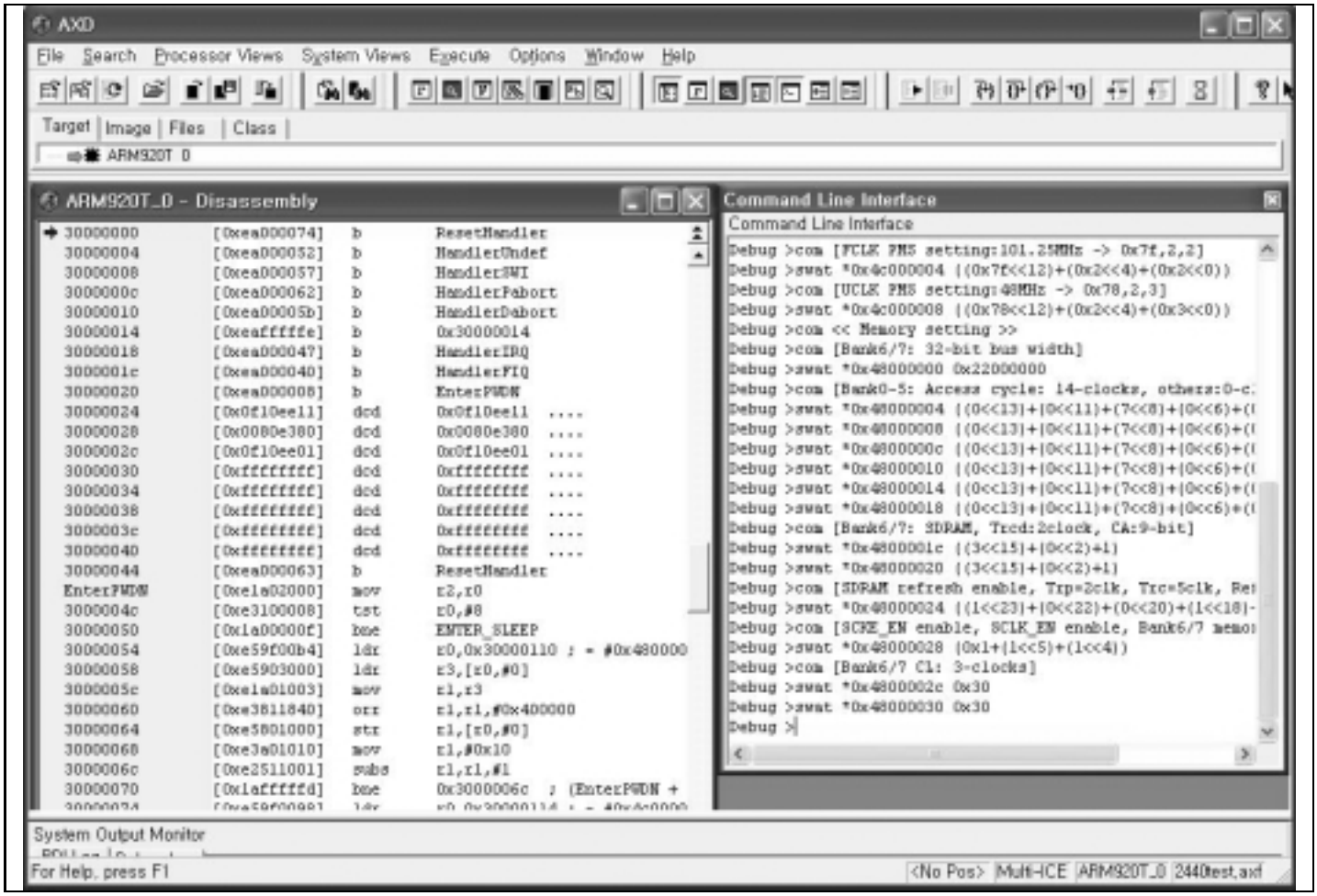


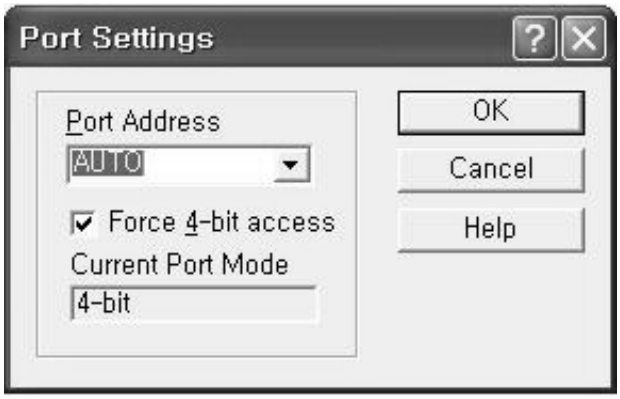
Figure 2-13. ARM Extended Debugger Window (AxD): Command Window

2. Set breakpoint at Main in 2440TEST.c as follows:
   break Main
3. Execute the program by clicking Execute menu→Go. The program execution will stop at Main( ).
4. Now, the downloaded image file will run on SDRAM area. 2440TEST program running status can be monitored on the DNW.

MULTI-ICE CHECKPOINTS

1. Error messages

Refer to Error Messages of the Multi-ICE user's guide. If you cannot solve the problem by using the instructions in the user's guide, then apply the 'Force 4-bit access" option.



2. Multi-ICE current consumption problem

Multi-ICE draws the Multi-ICE operating current from a target board. The current is about 130mA at 3.3V. If the target board cannot supply the 130mA, an external power supply must be used for supplying the current to Multi-ICE.

3. nTRST, TMS, TCK and TDI pin connections

TMS, TCK and TDI pin must be pulled-up with 10K registers. If the Multi-ICE is not used when development is completed, nTRST must be 'L' level at least during the reset.

HOW TO USE ARM DEBUGGER WITH OPENICE32-A900

Followings explain how to download the compiled image to SDRAM memory on the SMDK2440 by ARM debugger through OPENice32-A900, an emulator for ARM processor.

Note:

If you need technical support of OPENice32-A900, please contact AIJI System (http://www.aijisystem.com, openice@aijisystem.com)

CONFIGURING ARM DEBUGGER FOR OPENice32-A900

To debug the target board with OPENice32-A900, you should configure ARM Debugger for Windows (ADW) or ARM Extended Debugger (AxD). As MULTI-ICE, OPENice32-A900 should be connected through JTAG port on the board and switched on.

To configure ARM Debugger for OPENice32-A900 interface, follow the steps:

1. Select Configure Debugger from the Options menu.

   Options -> Configure Target

2. Debugger Configuration dialog box is displayed (See Figure 2-14). If there is no OPENice32-A900 in the target environment box, then you have to click on the Add button and select OPENice32-A900.DLL.
   - ARMulator: lets you execute the ARM program without any physical emulator by simulating ARM instructions in software.
   - OPENice32-A900: connects the ARM debugger to OPENice32-A900 attached to the target board.

3. Select OPENice32-A900 from the Target environment box, and click the Configure button.

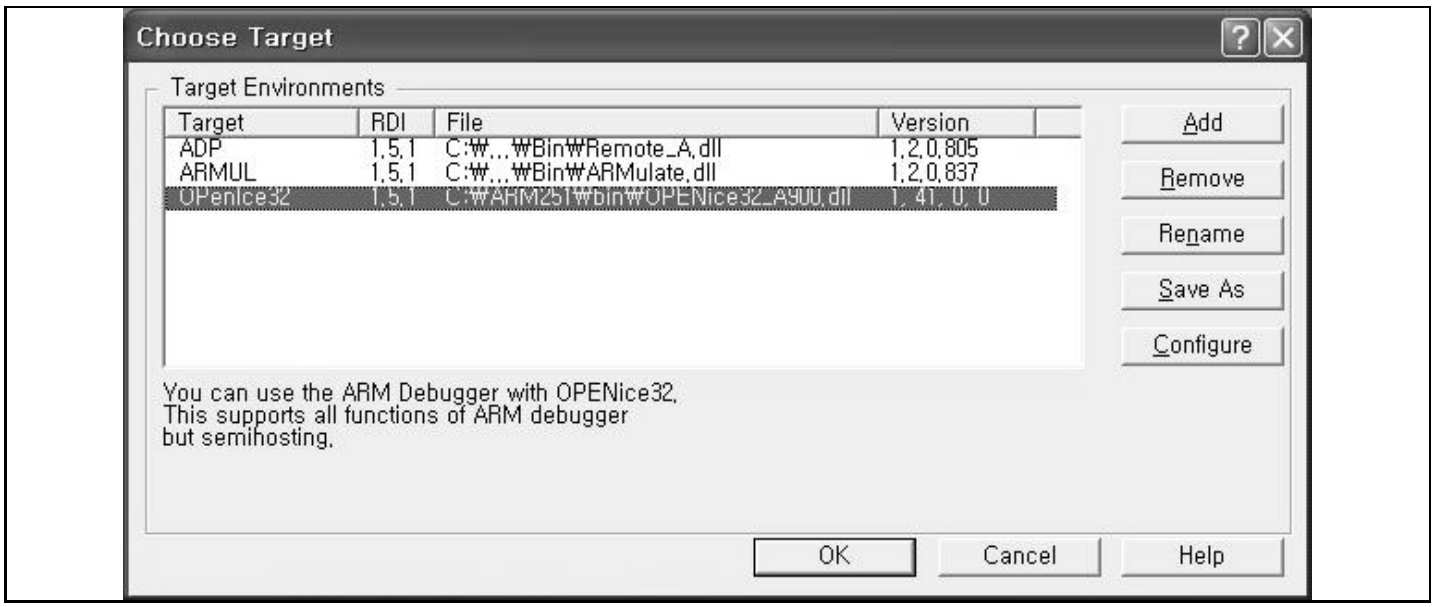


Figure 2-14. Debugger Configuration: Target Page

4. ConfigureOPENice32-A900 dialog box (See Figure 2-15, 2-16).
   — Remote page: select the connection to OPENICE32-A900.

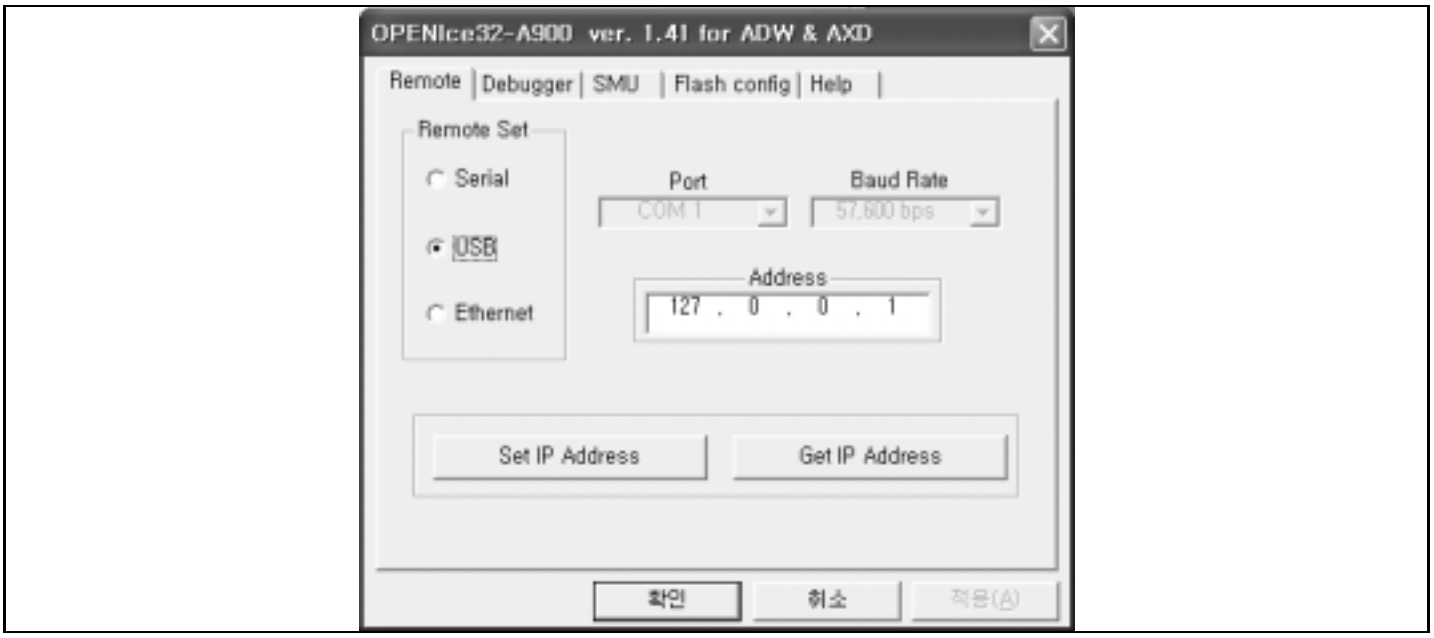


Figure 2-15. OPENice32-A900 Configuration: Communication Setting Page

— Debugger page : set the Endian and decide where initializes S3C2440X without any boot ROM.
Endian : little (If the big endian is used, Endian: big has to be selected.).
It should be matched with the option that you set in the compiler.
To init SMU : If you want to initialize S3C2440X on the board without any boot ROM, check it and set SMU in the SMU page, (Don't check it in this application.)
Flash download : If image file will be downloaded to a flash device, check it and set options in the Flash config page. (Don't check it in this application.)

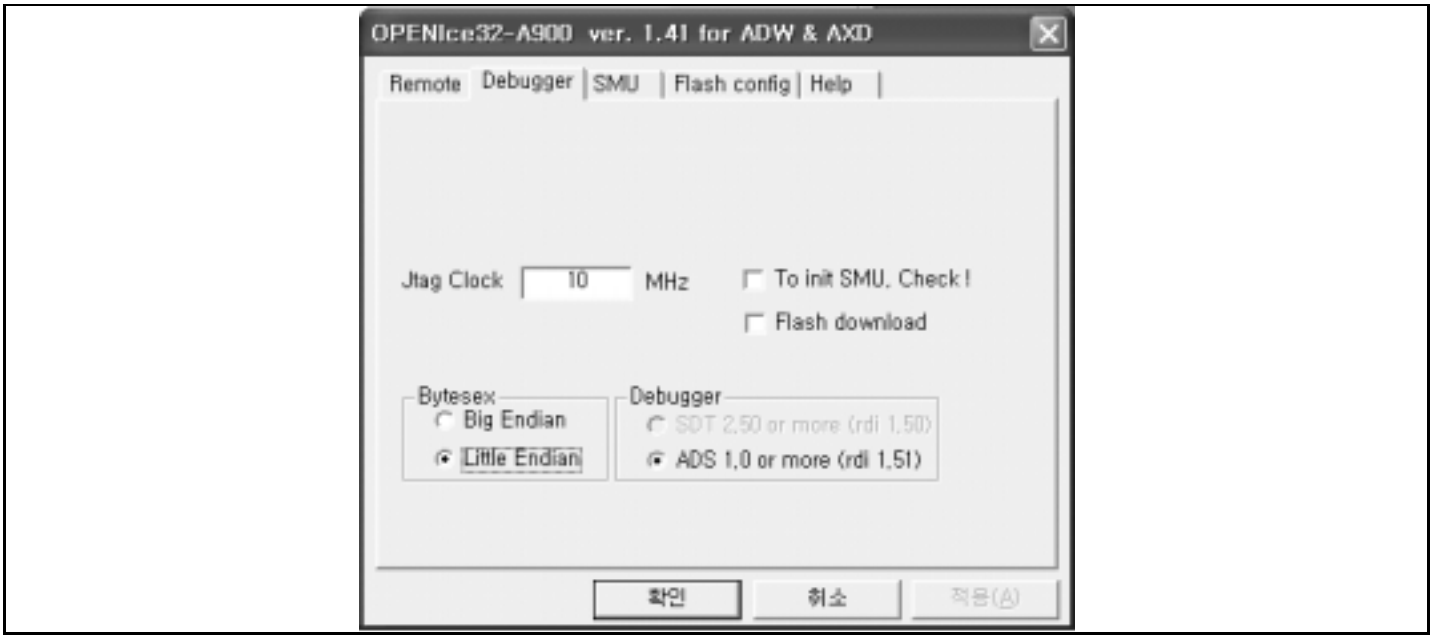


Figure 2-16. OPENice32-A900 Configuration: Endian and SMU Setting Page

— SMU page: set SMU of S3C2440X. Select SMDK2440 or S3C2440X from the device name and modify the values.(No need to set it in this application). SMU of S3C2440 is the same as S3C2410. So, if you cannot find S3C2440 from the device name, then you may select SMU of S3C2410 instead of S3C2440.

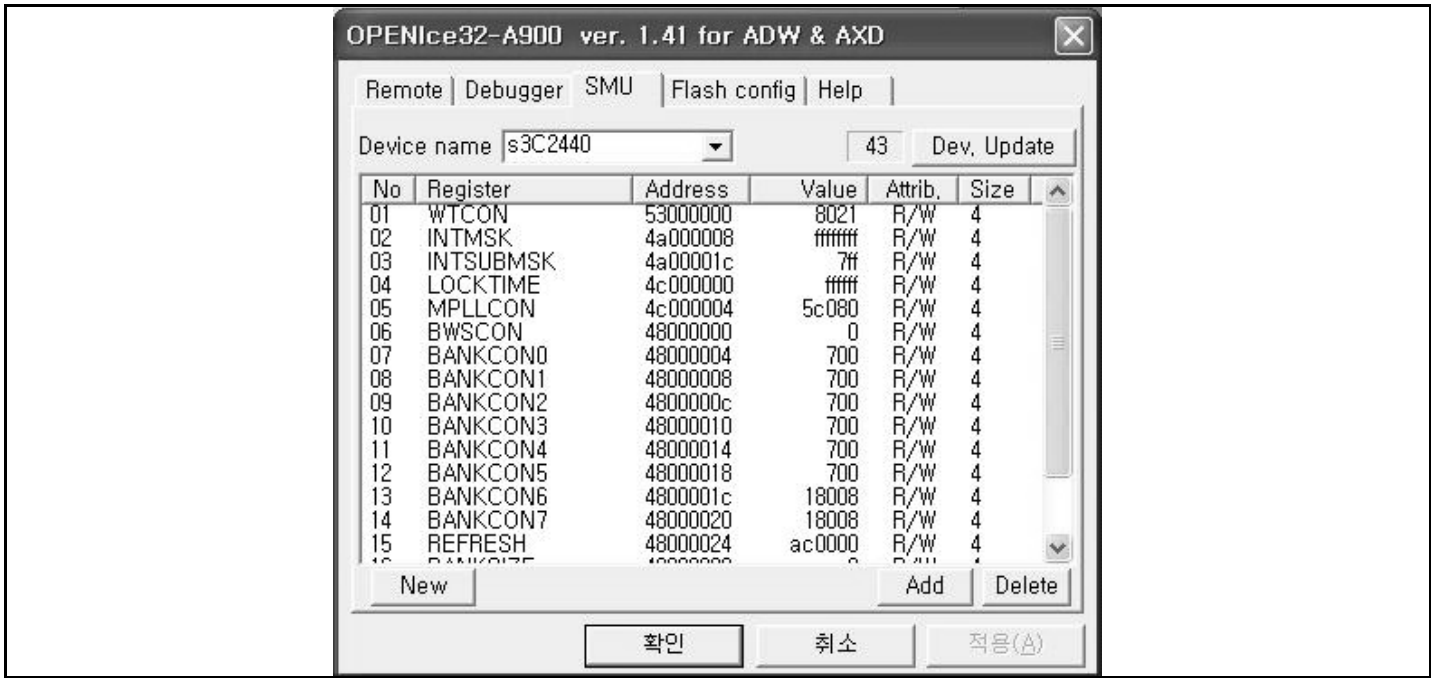


Figure 2-17. OPENice32-A900 Configuration: SMU Setting Page

— Flash Config page: set options for Flash download. (No need to set it in this application).

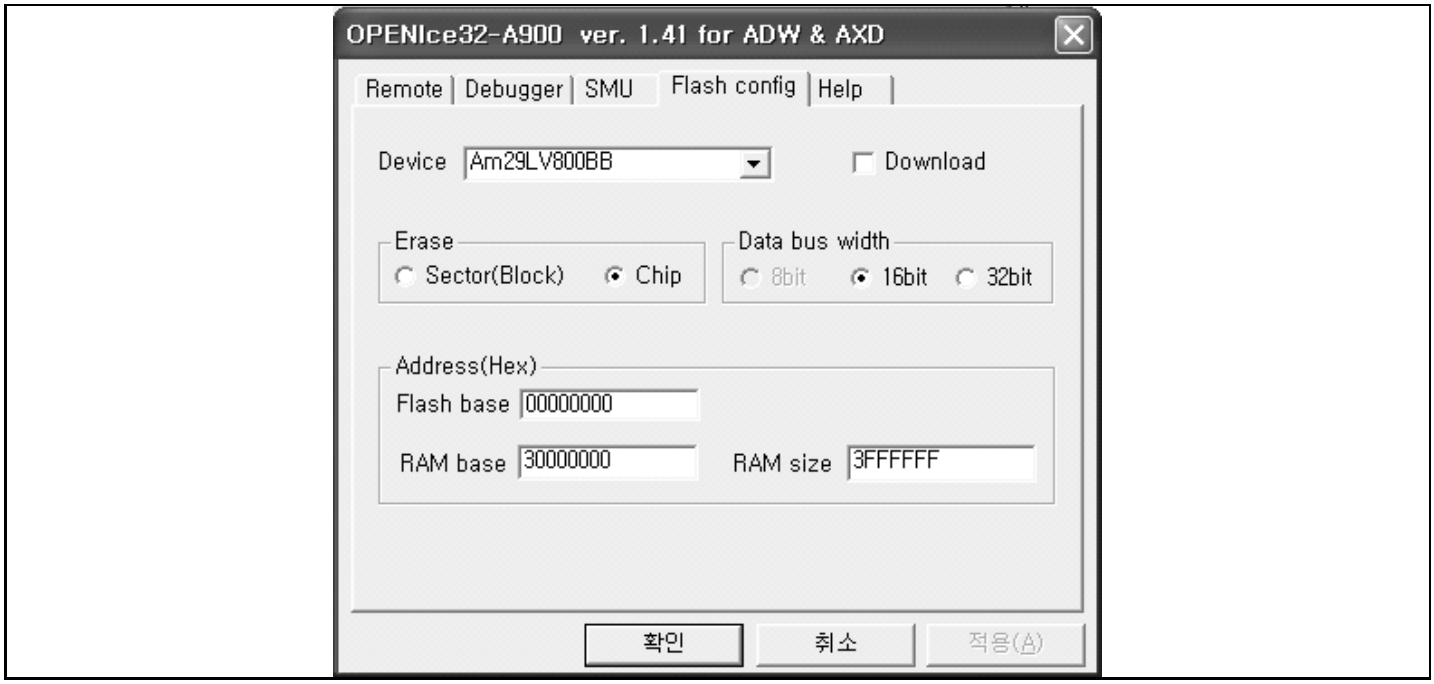


Figure 2-18. OPENice32-A900 Configuration: Flash Configuration Setting Page

5. If you click the OK button on Choose Target dialog box (See Figure 2-19), the debugger will be restarted. The restarting dialog box is displayed and numbers are rapidly changing, indicating that it is reading and writing to the target. This means that the executable image file is downloaded to the SDRAM code area.

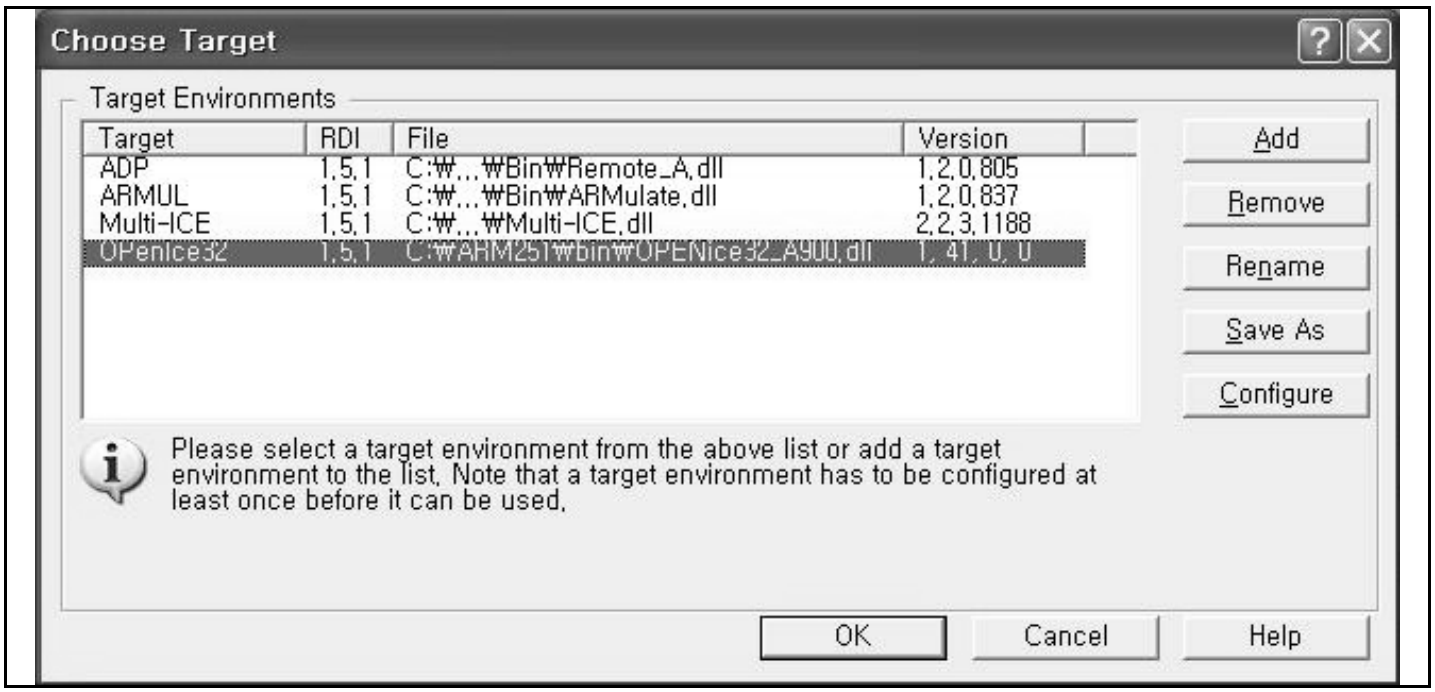


Figure 2-19. Debugger Configuration: Choose Target

This configuration is initially done and the setting is saved, which relieves the user of repeating another configuration next time.

EXECUTING 2440TEST.AXF USING OPENICE32-A900

1. Select Load Image from the File menu and select the compiled image (2440TEST.AXF). Then it will be downloaded to the SDRAM on the board.
2. Execute the program by select Go from the Execute menu.
3. Now, the downloaded image file will run on SDRAM area. 2440TEST program running status can be monitored on the AXD.

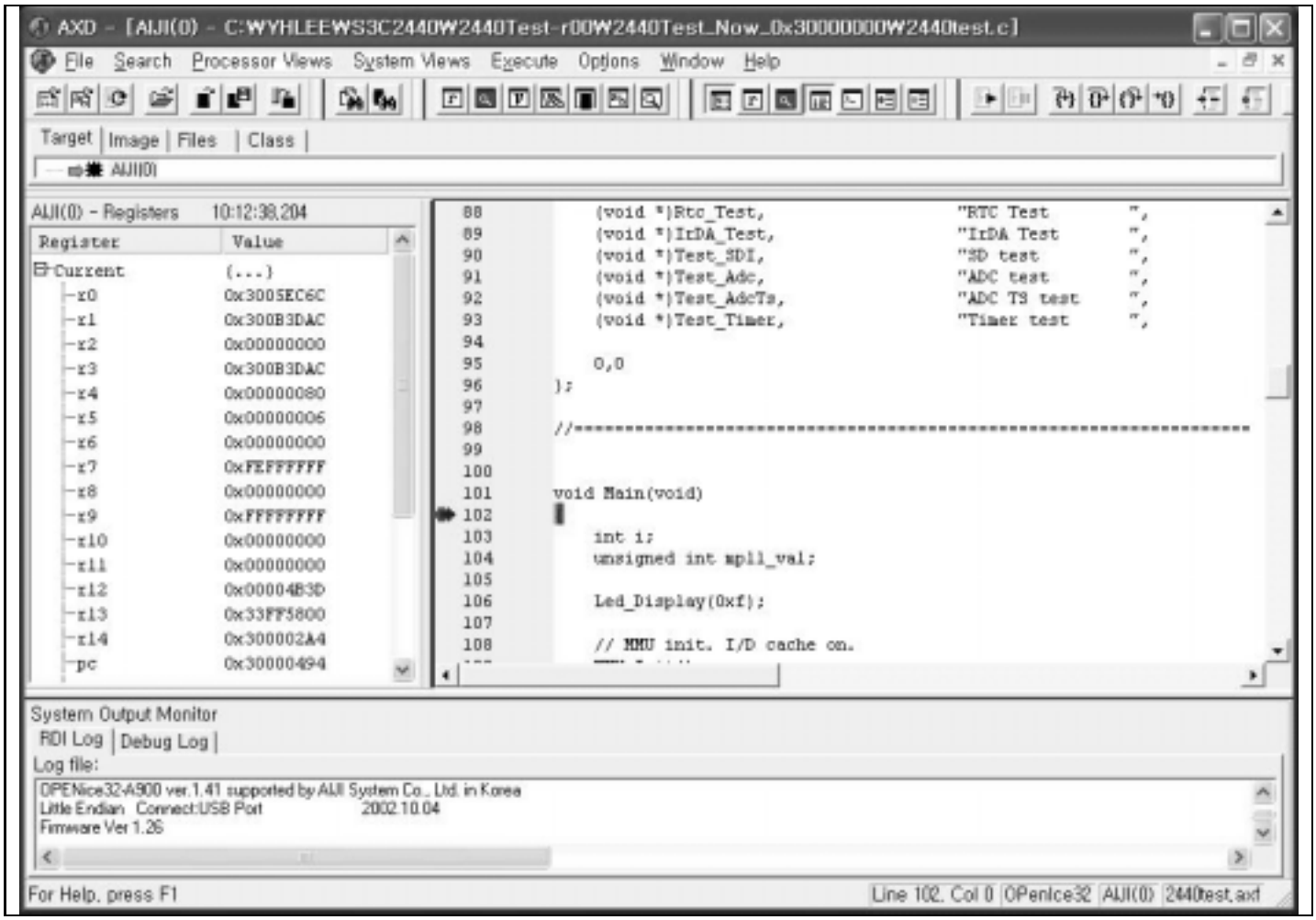


Figure 2-20. ARM Extended Debugger (AxD): After Downloading

DEBUGGING DOWNLOADED IMAGE IN ADW OR AXD

Stepping Through Program

To step through the program execution flow, you can select one of the following three options:

— Step: advances the program to the next line of code that is displayed in the execution window.

— Step Into: advances the program to the next line of code that follows all function calls. If the code is in a called function, the function source is displayed in the Execution window and the current code.

— Step Out: advances the program from the current function to the point from which it was called immediately after the function call. The appropriate line of code is displayed in the Execution window.

Setting Breakpoint

A breakpoint is the point you set in the program code where the ARM debugger will halt the program operation. When you set a breakpoint, it appears as a red marker on the left side of the window.

To set a simple breakpoint on a line of code, follow these steps:

1. Double-click the line where you want to place a break, or choose Toggle Breakpoint from the Execute menu. The Set or Edit Breakpoint dialog box is displayed.
2. Set the count to the required value or expression (The program stops only when this expression is correct).

To set a breakpoint on a line of code within a particular program function:

1. Display a list of function names by selecting Function Names from View menu.
2. Double-click the function name you want to open. A new source window is displayed containing the function source.
3. Double-click the line where the breakpoint is to be placed, or choose Toggle Breakpoint from the Execute menu. The Set or Edit Breakpoint dialog box appears.
4. Set the count to the required value or expression (The program stops only when this expression is correct).

Setting Watch Point

A watch point halts a program when a specified register or a variable, which is set to a specific number, is about to be changed.

To set a watch point, follow these steps:

1. Display a list of registers, variables, and memory locations you want to watch by selecting the Registers, Variables, and Memory options from the View menu.
2. Click the register, variable, or memory area in which you want to set the watch point. Then, choose Set or Edit Watchpoint from the Execute menu.
3. Enter a Target Value in the Set or Edit Watchpoint dialog box. Program operation will stop when the variable reaches the specified target value.

VIEWING VARIABLES, REGISTERS, AND MEMORY

You can view and edit the value of variables, registers, and memory by choosing the related heading from the View menu:

— Variables: for global and local variables.

— Registers: for the current mode and for each of the six register view modes.

— Memory: for the memory area defined by the address you enter.

DISPLAYING CODE INTERLEAVED WITH DISASSEMBLY

If you want to display the source code interleaved with disassembly, choose Toggle Interleaving on the Options menu. This command toggles between Displaying Source Only and Displaying Source Interleaved with Disassembly. When the source code is shown interleaved with disassembly, machine instructions appear in a lighter gray color.

For additional information about ARM Debugger, refer to the reference document released by ARM.

SWITCHING DEVELOPMENT TOOLKIT

USB boot code (U2440mon.c) and test code (2440test.c) can be executed in the SDT or ADS by changing the option in OPTION.H and Makefile. In other words, boot and test code can be translated from ADS to SDT and vice versa by changing option in the following table.

Table 2-1. Toolkit Switching Options

	ADS	SDT
Makefile	fromelf -nodebug -bin -output $(PRJ).bin $(PRJ).elf	fromelf -nodebug -nozeropad $(PRJ).elf -bin $(PRJ).bin
OPTION.H	#define ADS10 TRUE	#define ADS10 FALSE

TRANSLATING CODE FROM ADS INTO SDT

U2440MON and 2440TEST codes were optimized for ADS 1.0. In other words, these codes were compiled and linked by the ADS 1.0. So, these codes should be modified to work on the SDT.
If you want to compile our codes with the SDT, then you have to change the definition of ADS1.0 in OPTION.H from 'TRUE' to 'FALSE' and the option in makefile from 'fromelf -nodebug -bin -output $(PRJ).bin $(PRJ).elf' to 'fromelf -nodebug -nozeropad $(PRJ).elf -bin $(PRJ).bin' option.

TRANSLATING CODE FROM SDT INTO ADS

First function __rt_lib_init(); is applied to the main code. And then old Makefile for SDT is changed to a new one for ADS.
If you have used SDT 2.50, it is recommended that you should read related documents (ADS, Getting Started, and ARM DUI0064A) about the difference between SDT 2.50 and ADS 1.0.

REMOVED OR CHANGED ITEMS FROM MAKEFILE FOR SDT 2.50

1. ARMLINK option
   - first: the path of an object file is not needed.
2. ARMASM option
   - cpu: should be changed as -cpu ARM920T
   - apcs: should be changed to -apcs /noswst
3. Compiler option
   - fc : should be removed.
   - zpz0 : should be removed. This is not needed any more.
   - apcs : should be changed to -apcs /noswst
   - processor : should be removed.
   - arch : should be removed.
   - cpu : should be added as -cpu ARM920T
4. fromelf.exe
   - nozeropad: should be removed. This is not needed any more.
   - output : command line style should be changed using -output option as follows:
     fromelf -nodebug -bin -output $(BIN)\$(PRJ).bin $(BIN)\$(PRJ).AXF

OTHER ITEMS SHOULD BE CHANGED FOR ADS 1.0

1. ammake.exe
   The armmake.exe is not supplied with ADS 1.0. So, you have to use your own Make utility. (nmake.exe, make.exe, pmake.exe, or armmake.exe in SDT 2.50).
2. Embedded library
   There is no separate embedded library in ADS 1.0. All the library in ADS 1.0 is made for embedded applications.
   But, the library must be initialized using __rt_lib_init() function. If you do not use __rt_lib_init(), the C library does not work well.
3. There is no tasm.exe. The tasm.exe is merged into armasm.exe.

EXAMPLE OF MAKEFILE FOR ADS 1.0

This is a sample makefile on ADS 1.0.

```
##### File Definition ####
PRJ = 2440test
INIT= 2440init
AM1 = 2440slib
AM2 = 2440swis
CM1 = 2440lib
CM2 = mmu
CM3 = 2440iis
CM4 = timer
CM5 = 2440RTC
CM6 = 2440IIC
          .
          .
          .
CM38 = spi
CM39 = strata32

#### Destination path Definition ####
OBJ=.\obj
ERR=.\err

#### ARM tool Definition ####
ARMLINK = armlink
ARMASM = armasm
ARMCC   = armcc

#### Option Definition ####
LFLAGS = -ro-base 0x30000000 -elf -map -xref \
          -list list.txt -first $(INIT).o(Init)
AFLAGS = -li -apcs /noswst -cpu ARM920T
CFLAGS = -c -g+ -li -apcs /noswst -cpu ARM920T
#### Object combine Definition ####

OBJS = $(OBJ)\$(INIT).o $(OBJ)\$(AM1).o $(OBJ)\$(AM2).o $(OBJ)\$(PRJ).o \
          $(OBJ)\$(CM1).o $(OBJ)\$(CM2).o $(OBJ)\$(CM3).o $(OBJ)\$(CM4).o \
          $(OBJ)\$(CM5).o $(OBJ)\$(CM6).o $(OBJ)\$(CM7).o $(OBJ)\$(CM8).o \
          $(OBJ)\$(CM9).o $(OBJ)\$(CM10).o $(OBJ)\$(CM11).o $(OBJ)\$(CM12).o \
          $(OBJ)\$(CM13).o $(OBJ)\$(CM14).o $(OBJ)\$(CM15).o $(OBJ)\$(CM16).o \
          $(OBJ)\$(CM17).o $(OBJ)\$(CM18).o $(OBJ)\$(CM19).o $(OBJ)\$(CM20).o \
          $(OBJ)\$(CM21).o $(OBJ)\$(CM22).o $(OBJ)\$(CM23).o $(OBJ)\$(CM24).o \
          $(OBJ)\$(CM25).o $(OBJ)\$(CM26).o $(OBJ)\$(CM27).o $(OBJ)\$(CM28).o \
          $(OBJ)\$(CM29).o $(OBJ)\$(CM30).o $(OBJ)\$(CM31).o $(OBJ)\$(CM32).o \
          $(OBJ)\$(CM33).o $(OBJ)\$(CM34).o $(OBJ)\$(CM35).o $(OBJ)\$(CM36).o \
          $(OBJ)\$(CM37).o $(OBJ)\$(CM38).o $(OBJ)\$(CM39).o
```

```
all: $(PRJ).axf

clean:
        del $(OBJ)\*.o

$(PRJ).axf: $(OBJS)
        del $(PRJ).bin
        del $(PRJ).axf
        $(ARMLINK) $(LFLAGS) -o $(PRJ).axf $(OBJS)
        fromelf -nodebug -bin -output $(PRJ).bin $(PRJ).axf

#For SDT2.5 fromelf -nodebug -nozeropad $(PRJ).elf -bin $(PRJ).bin
#For ADS1.0 fromelf -nodebug -bin -output $(PRJ).bin $(PRJ).elf


$(OBJ)\$(PRJ).o:  $(PRJ).c  2440addr.h  2440lib.h  makefile
        del $(OBJ)\$(PRJ).o
        del $(ERR)\$(PRJ).err
        $(ARMCC) $(CFLAGS)  $(PRJ).c -o $(OBJ)\$(PRJ).o -Errors $(ERR)\$(PRJ).err
        .
        .
        .
        .
        .
$(OBJ)\$(CM27).o:  $(CM27).c  2440addr.h  2440lib.h  makefile
        del $(OBJ)\$(CM27).o
        del $(ERR)\$(CM27).err
        $(ARMCC) $(CFLAGS)  $(CM27).c -o $(OBJ)\$(CM27).o -Errors $(ERR)\$(CM27).err

$(OBJ)\$(CM28).o:  $(CM28).c  2440addr.h  2440lib.h  makefile
        del $(OBJ)\$(CM28).o
        del $(ERR)\$(CM28).err
        $(ARMCC) $(CFLAGS)  $(CM28).c -o $(OBJ)\$(CM28).o -Errors $(ERR)\$(CM28).err

$(OBJ)\$(CM29).o:  $(CM29).c  2440addr.h  2440lib.h  makefile
        del $(OBJ)\$(CM29).o
        del $(ERR)\$(CM29).err
        $(ARMCC) $(CFLAGS)  $(CM29).c -o $(OBJ)\$(CM29).o -Errors $(ERR)\$(CM29).err
        .
        .
        .
        .
        .
```

3 PROGRAMMING FLASH MEMORIES

PROGRAMMING NAND FLASH MEMORY

The SMDK2440 supports NAND flash control interface. There are two methods to write images to NAND flash memory:

— Write image files to NAND flash memory with write-program.

— Write image files to NAND flash memory with JTAG interface.

NAND FLASH WRITE WITH WRITE-PROGRAM

The target image must be downloaded in SDRAM before executing write-program.

To download and write a target image from the host to SDRAM through USB interface, follow the steps:

1. Run the DNW utility program (See Figure 3-1).

Figure 3-1. DNW Window to Download

2. Select Serial Port on the system menu of DNW and click Connect to open the serial port (See Figure 3-2, 3).

Figure 3-2. DNW Window (to Connect Serial Port)

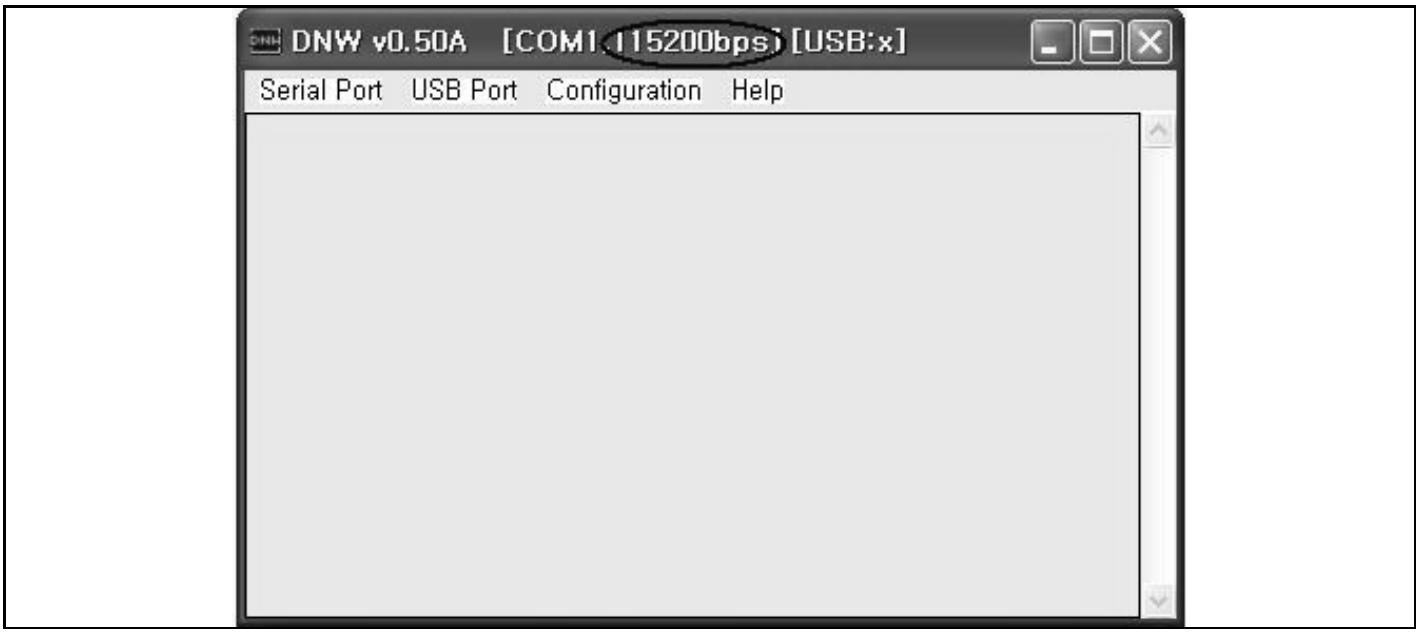


Figure 3-3. DNW Window (after Open Baud-rate is Printed on Title Bar)

3. Connect the serial and USB cable from the host PC to SMDK2440 system and turn on the power of SMDK2440 board (See Figure 3-4).

NOTES:

1. Jumper J1-B, J2-B, J3-B, J4-B must be 'H', 'L', 'L' and 'H' for AMD NOR Booting.
2. SMDK2440 must run monitor program that is provided by SAMSUNG.

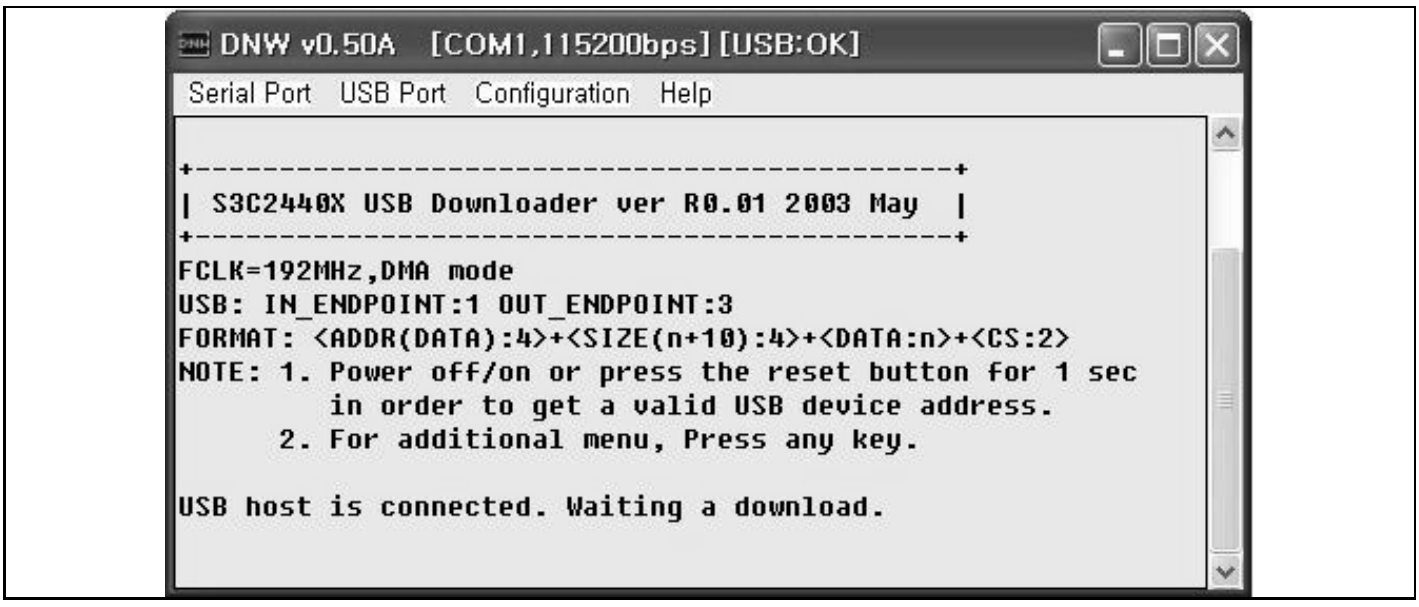


Figure 3-4. DNW Window (after Turning on the SDMK2440)

4. To see the additional menu, press any key on the DNW window (See Figure 3-5).

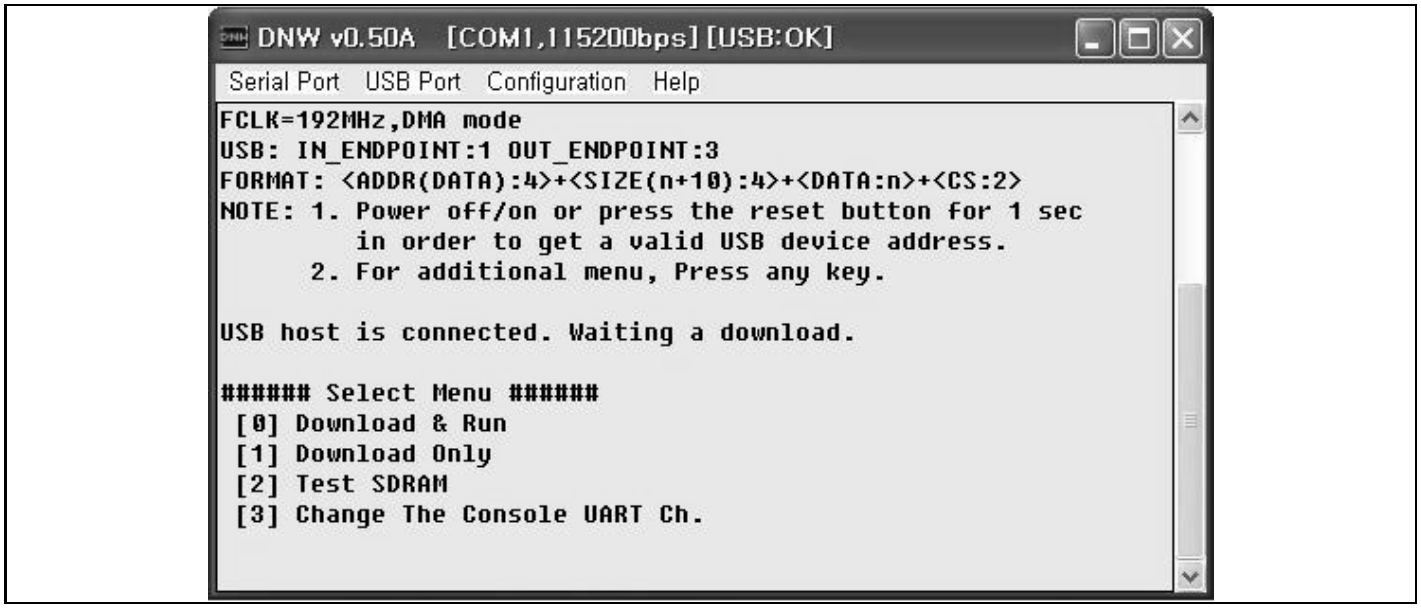


Figure 3-5. DNW Window to Download

5. For downloading a target image, select Download Only item on the DNW window (See Figure 3-6).
6. Write the address to download and press enter key (See Figure 3-6).

NOTE: The target image must be located on 0x30100000 address.

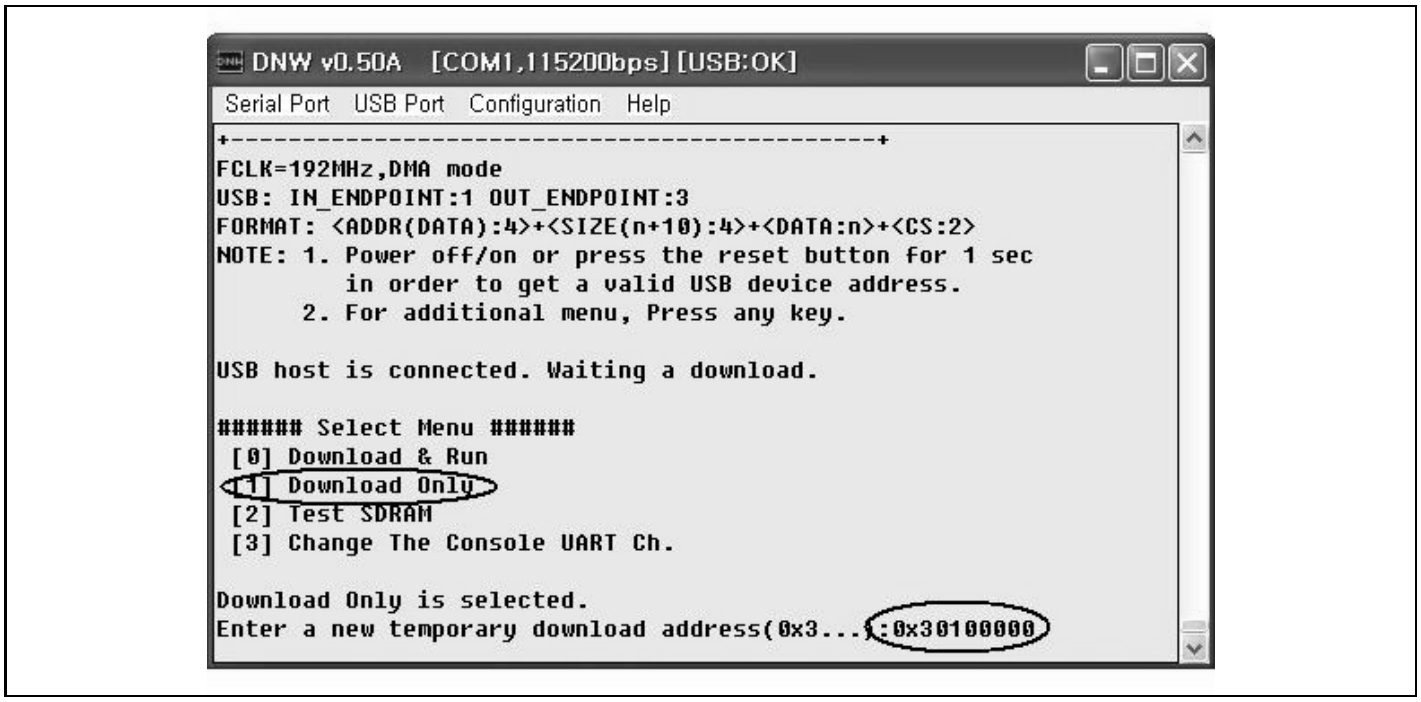


Figure 3-6. DNW Window (to Select Target Image)

7. Select USB Port on the system menu of the DNW and click Transmit to download a target image (See Figure 3-7).

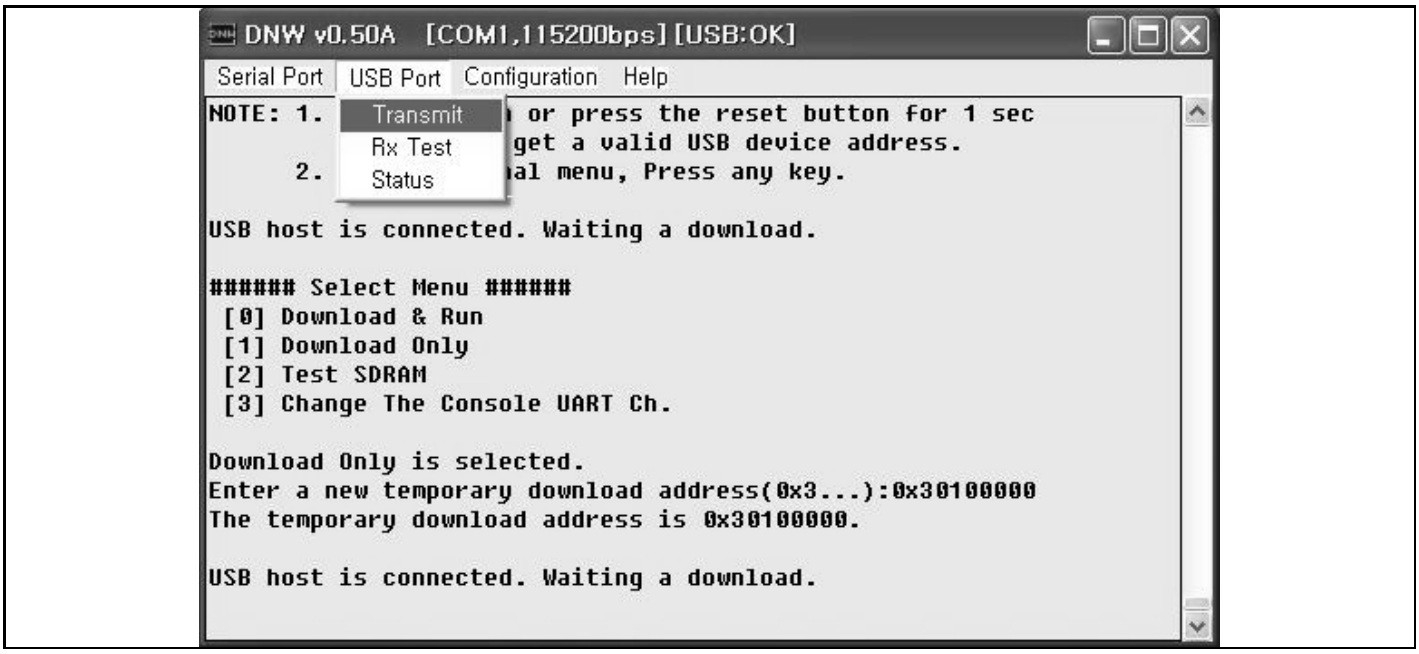


Figure 3-7. DNW Window (for USB Downloading)

8. Select a target image on file open dialog box (See Figure 3-8).

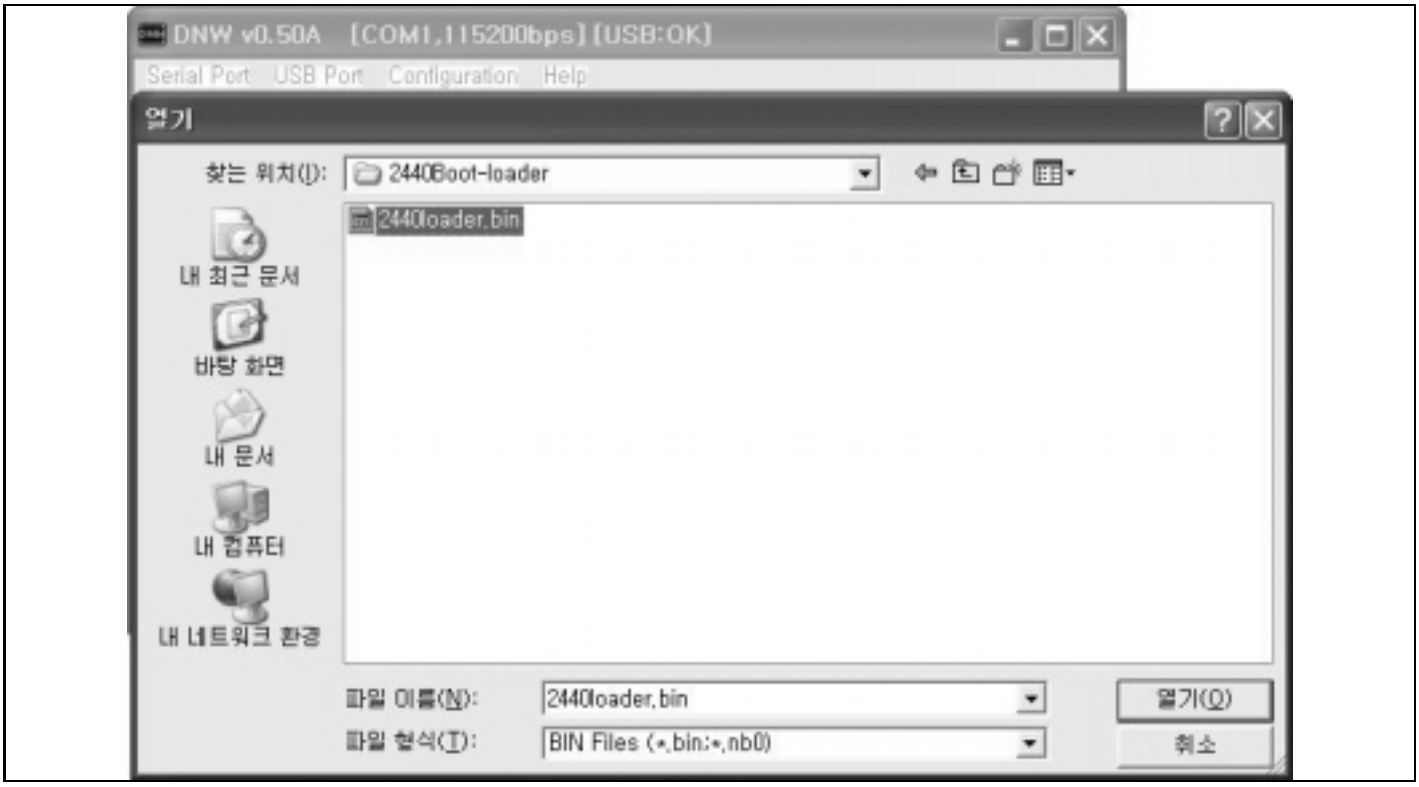


Figure 3-8. DNW Window (File Open Dialog Box)

9. For downloading 2440test program, select Download & Run item on the DNW window and download 2440test program like step 7 (See Figure 3-9).

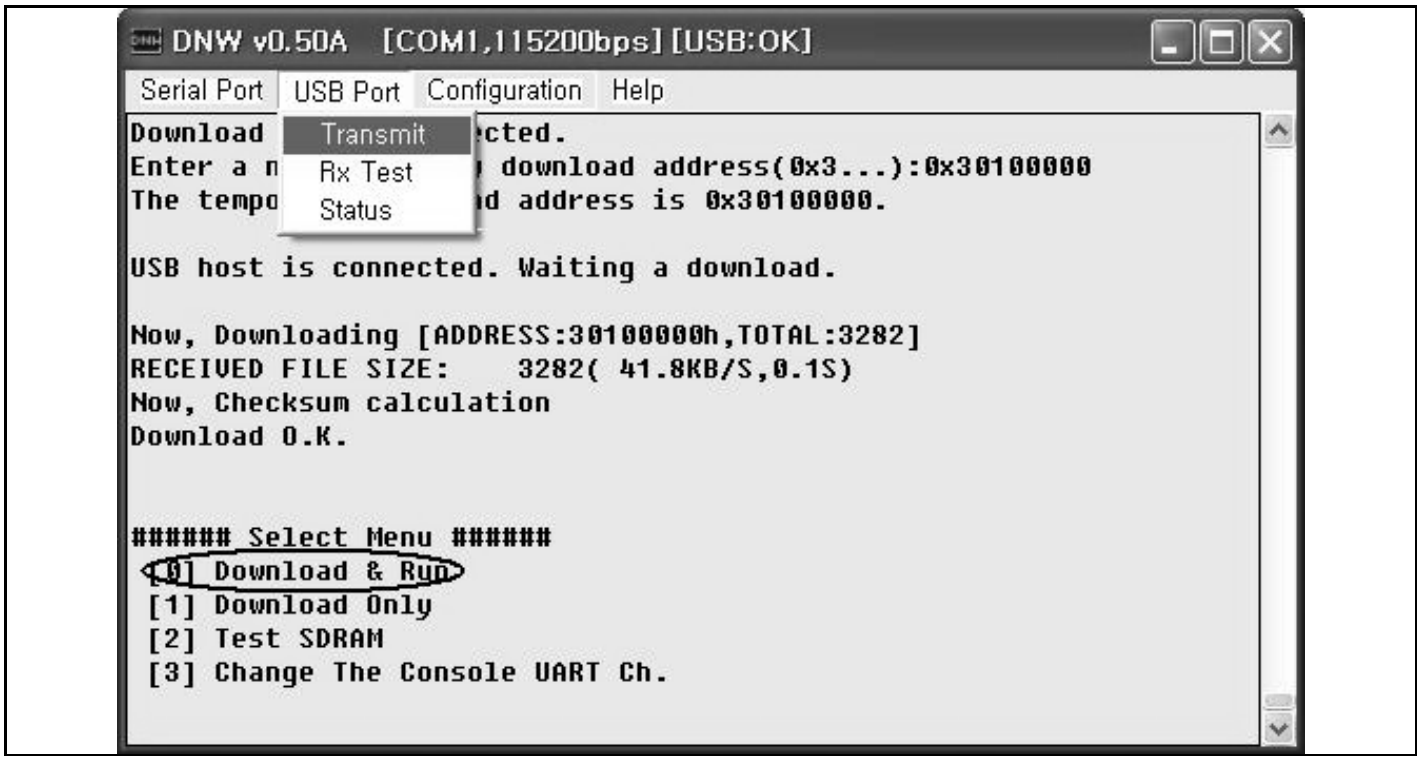


Figure 3-9. DNW Window (File Open Dialog Box)

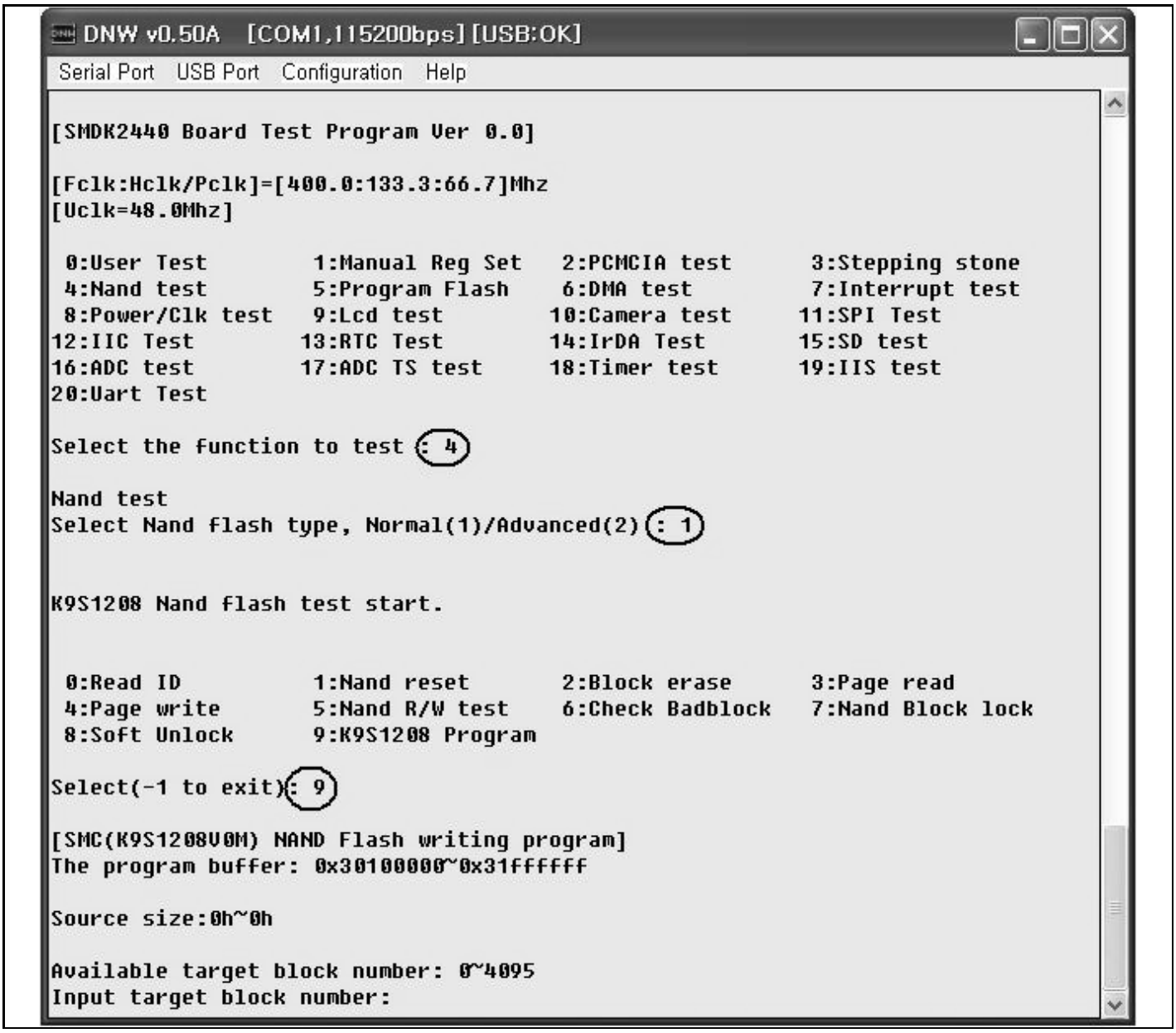


Figure 3-10. DNW Window (2440test Program)

10. Write ’5’ and press enter key on the DNW window (See Figure 3-10).
11. Select Nand flash type (See Figure 3-10).
12. Write ’4’ and press enter key on the DNW window (See Figure 3-10).
13. Write the target block number that is the start block to write and press enter key on the DNW window (See Figure 3-11).
14. Write total byte size of the target image, it should be aligned 0x4000 (one block size) bytes (See Figure 3-11).

NOTE: 2440test program supports normal NAND flash type (K9S1208: SmartMedia card, SAMSUNG) and advanced NAND flash type (K9K2G16: SAMSUNG). To write another type of device, it is required to modify the source codes NAND.C(normal K9S1208) or K9K2G16.C(advanced K9K2G16).

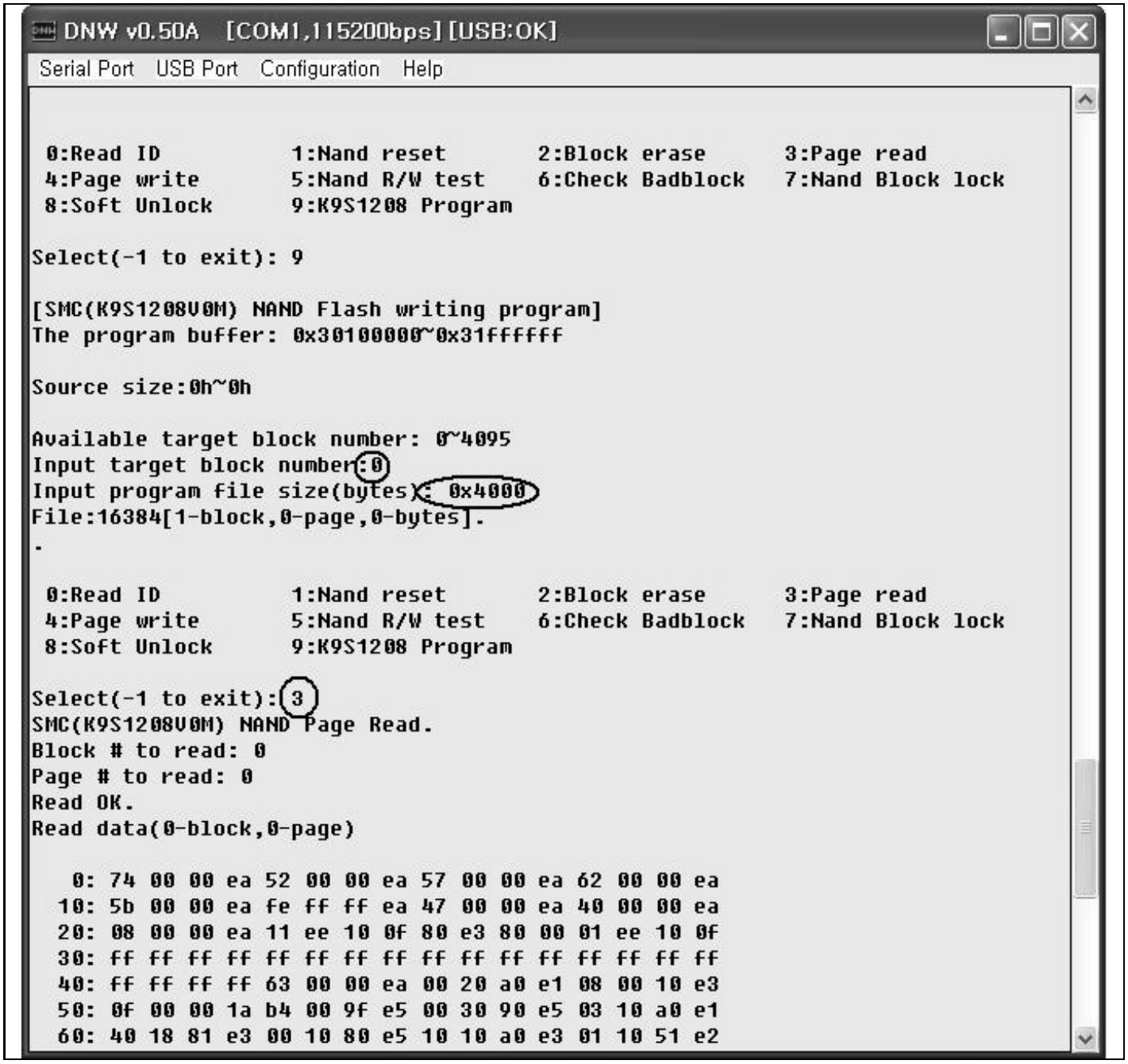


Figure 3-11. DNW Window (NAND Flash Write Program)

15. To check the contents of NAND flash, select Page read function (See Figure 3-11).

AUTO BOOTING THROUGH NAND FLASH

The S3C2440 supports auto booting operation with NAND flash memory.

Before power on the SMDK2440 system, it must have SmartMedia card with boot-loader and OS image.

NOTES:

1. Jumper J1-B, J2-B, J3-B, J4-B must be ‘L’, ‘L’, ‘L’ and ‘L’ for NAND Booting.
2. 2440test program and boot-loader, which is supplied by SMSUNG, support normal NAND flash type (K9S120: SmartMedia card, SAMSUNG) and advanced NAND flash type (K9K2G16: SAMSUNG). To write another type of device, it is required to modify the source codes NAND.C(normal K9S1208) or K9K2G16.C(advanced K9K2G16).

BOOTING NAND FLASH

To make NAND flash memory for auto booting, follow the steps:

1. Program the boot-loader image to the block 0 of NAND flash memory.
2. Program the OS image to the other blocks of NAND flash memory. The OS image must be located block 1 to the rest blocks.

NAND FLASH ECC (ERROR CHECKING AND CORRECTION)

The S3C2440X supports ECC algorithm, which is based on XOR calculation, for error checking and correction.

1. Example of one byte ECC (find error bit)

Old

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	0	0

Old ECC code

P3	NP3	P2	NP2	P1	NP1
1	0	1	0	1	0

New

D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	0	0	0	0	0

New ECC code

P4	NP3	P2	NP2	P1	NP1
0	0	0	0	1	1

Old ECC ^ New ECC

S3	NS3	S2	NS2	S1	NS1
1	0	1	0	0	1

P3 = [7]^[6]^[5]^[4]
P2 = [7]^[6]^[3]^[2]
P1 = [7]^[5]^[3]^[1]

NP3 = [3]^[2]^[1]^[0]
NP2 = [5]^[4]^[1]^[0]
NP1 = [6]^[4]^[2]^[0]

	P3	NP3	P2	NP2	P1	NP1
Old	1	0	1	0	1	0
New	0	0	0	0	1	1
(Old ECC) ^ (New ECC)	1	0	1	0	0	1
Error code	1		1		0	
Result	Bit 6					

NOTES:

1. [n] means bit [n] (or Dn).
2. The ’10b’ value of ‘Old ECC ^ New ECC’ corresponds to ‘1b’ Error code and ‘01b’ corresponds to ‘0b’.

2. Example of n byte ECC

	D7	D6	D5	D4	D3	D2	D1	D0
0th byte	[7]0	[6]0	[5]0	[4]0	[3]0	[2]0	[1]0	[0]0
1st byte	[7]1	[6]1	[5]1	[4]1	[3]1	[2]1	[1]1	[0]1
2nd byte	[7]2	[6]2	[5]2	[4]2	[3]2	[2]2	[1]2	[0]2
3rd byte	[7]3	[6]3	[5]3	[4]3	[3]3	[2]3	[1]3	[0]3
…	…							
509th byte	[7]509	[6]509	[5]509	[4]509	[3]509	[2]509	[1]509	[0]509
510th byte	[7]510	[6]510	[5]510	[4]510	[3]510	[2]510	[1]510	[0]510
511th byte	[7]511	[6]511	[5]511	[4]511	[3]511	[2]511	[1]511	[0]511

— Column parity (CPn)

CP3 =	NCP3 =
[7]0^ [7]1^[7]2^[7]3^ … ^[7]509^[7]510^[7]511	[3]0^ [3]1^[3]2^[3]3^ … ^[3]509^[3]510^[3]511
^[6]0^ [6]1^[6]2^[6]3^ … ^[6]509^[6]510^[6]511	^[2]0^[2]1^[2]2^[2]3^… ^[2]509^[2]510^[2]511
^[5]0^ [5]1^[5]2^[5]3^ … ^[5]509^[5]510^[5]511	^[1]0^[1]1^[1]2^[1]3^ … ^[1]509^[1]510^[1]511
^[4]0^ [4]1^[4]2^[4]3^ … ^[4]509^[4]510^[4]511	^[0]0^[0]1^[0]2^[0]3^ … ^[0]509^[0]510^[0]511

CP2 =	NCP2 =
[7]0^ [7]1^[7]2^[7]3^ … ^[7]509^[7]510^[7]511	[5]0^ [5]1^[5]2^[5]3^ … ^[5]509^[5]510^[5]511
^[6]0^ [6]1^[6]2^[6]3^ … ^[6]509^[6]510^[6]511	^[4]0^ [4]1^[4]2^[4]3^ … ^[4]509^[4]510^[4]511
^[3]0^ [3]1^[3]2^[3]3^ … ^[3]509^[3]510^[3]511	^[1]0^[1]1^[1]2^[1]3^ … ^[1]509^[1]510^[1]511
^[2]0^[2]1^[2]2^[2]3^… ^[2]509^[2]510^[2]511	^[0]0^[0]1^[0]2^[0]3^ … ^[0]509^[0]510^[0]511

CP1 =	NCP1 =
[7]0^ [7]1^[7]2^[7]3^ … ^[7]509^[7]510^[7]511	[6]0^ [6]1^[6]2^[6]3^ … ^[6]509^[6]510^[6]511
^[5]0^ [5]1^[5]2^[5]3^ … ^[5]509^[5]510^[5]511	^ [4]0^ [4]1^[4]2^[4]3^ … ^[4]509^[4]510^[4]511
^[3]0^ [3]1^[3]2^[3]3^ … ^[3]509^[3]510^[3]511	^ [2]0^[2]1^[2]2^[2]3^… ^[2]509^[2]510^[2]511
^[1]0^[1]1^[1]2^[1]3^ … ^[1]509^[1]510^[1]511	^ [0]0^[0]1^[0]2^[0]3^ … ^[0]509^[0]510^[0]511

— Row parity (RPn)

RP9 = [7]511^[6]511^[5]511^[4]511^[3]511^[2]511^[1]511^[0]511 ^… ^[7]384^[6]384^[5]384^[4]384^[3]384^[2]384^[1]384^[0]384 ^[7]383^[6]383^[5]383^[4]383^[3]383^[2]383^[1]383^[0]383 ^… ^[7]256^[6]256^[5]256^[4]256^[3]256^[2]256^[1]256^[0]256	NRP9 = [7]255^[6]255^[5]255^[4]255^[3]255^[2]255^[1]255^[0]255 ^… ^[7]128^[6]128^[5]128^[4]128^[3]128^[2]128^[1]128^[0]128 ^[7]127^[6]127^[5]127^[4]127^[3]127^[2]127^[1]127^[0]127 ^… ^[7]0^[6]0^[5]0^[4]0^[3]0^[2]0^[1]0^[0]0

RP8 = [7]511^[6]511^[5]511^[4]511^[3]511^[2]511^[1]511^[0]511 ^… ^[7]384^[6]384^[5]384^[4]384^[3]384^[2]384^[1]384^[0]384 [7]255^[6]255^[5]255^[4]255^[3]255^[2]255^[1]255^[0]255 ^… ^[7]128^[6]128^[5]128^[4]128^[3]128^[2]128^[1]128^[0]128	NRP8 = [7]383^[6]383^[5]383^[4]383^[3]383^[2]383^[1]383^[0]383 ^… ^[7]256^[6]256^[5]256^[4]256^[3]256^[2]256^[1]256^[0]256 ^[7]127^[6]127^[5]127^[4]127^[3]127^[2]127^[1]127^[0]127 ^… ^[7]0^[6]0^[5]0^[4]0^[3]0^[2]0^[1]0^[0]0

…

RP1 = [7]511^[6]511^[5]511^[4]511^[3]511^[2]511^[1]511^[0]511 ^[7]509^[6]509^[5]509^[4]509^[3]509^[2]509^[1]509^[0]509 ^[7]507^[6]507^[5]507^[4]507^[3]507^[2]507^[1]507^[0]507 ^… ^[7]1^[6]1^[5]1^[4]1^[3]1^[2]1^[1]1^[0]1	NRP1 = ^[7]510^[6]510^[5]510^[4]510^[3]510^[2]510^[1]510^[0]510 ^[7]508^[6]508^[5]508^[4]508^[3]508^[2]508^[1]508^[0]508 ^[7]506^[6]506^[5]506^[4]506^[3]506^[2]506^[1]506^[0]506 ^… ^[7]0^[6]0^[5]0^[4]0^[3]0^[2]0^[1]0^[0]0

3. Example of 4 bytes ECC (find 1bit error in the 4 bytes)

— Old data

	D7	D6	D5	D4	D3	D2	D1	D0
0th byte	0	0	0	0	0	0	0	0
1st byte	0	0	0	0	0	0	**1**	0
2nd byte	0	0	0	0	0	0	0	0
3rd byte	0	0	0	0	0	0	0	0

— New data

	D7	D6	D5	D4	D3	D2	D1	D0
0th byte	0	0	0	0	0	0	0	0
1st byte	0	0	**1**	0	0	0	**1**	0
2nd byte	0	0	0	0	0	0	0	0
3rd byte	0	0	0	0	0	0	0	0

— Column parity

	CP3	NCP3	CP2	NCP2	CP1	NCP1
Old data	0	1	0	1	1	0
New data	1	1	0	1	0	0
(Old ECC) ^ (New ECC)	1	0	0	0	1	0
Error code	1		0		1	
Result	5th bit					

— Row parity

	RP3	NRP3	RP2	NRP2
Old data	0	1	1	0
New data	0	0	0	0
(Old ECC) ^ (New ECC)	0	1	1	0
Error code	0		1	
Result	1st byte			

— Result: The 5th bit in the 1st byte is in error.

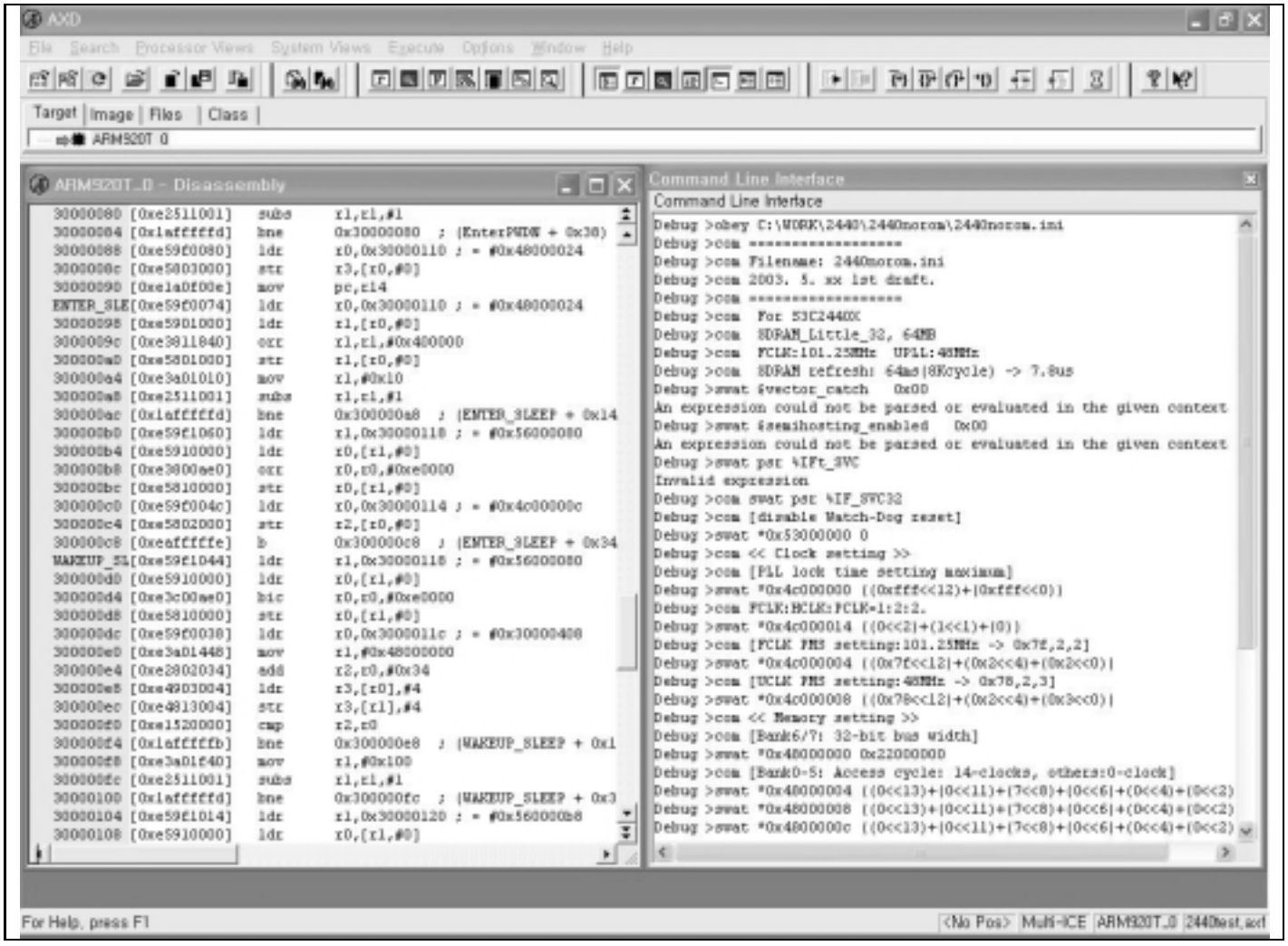
PROGRAMMING NOR FLASH MEMORY

The SMDK2440 supports NOR flash control interface. There are two types of NOR flash memories in SMDK2440: AMD and Intel STRATA flash memory. The actual methods:

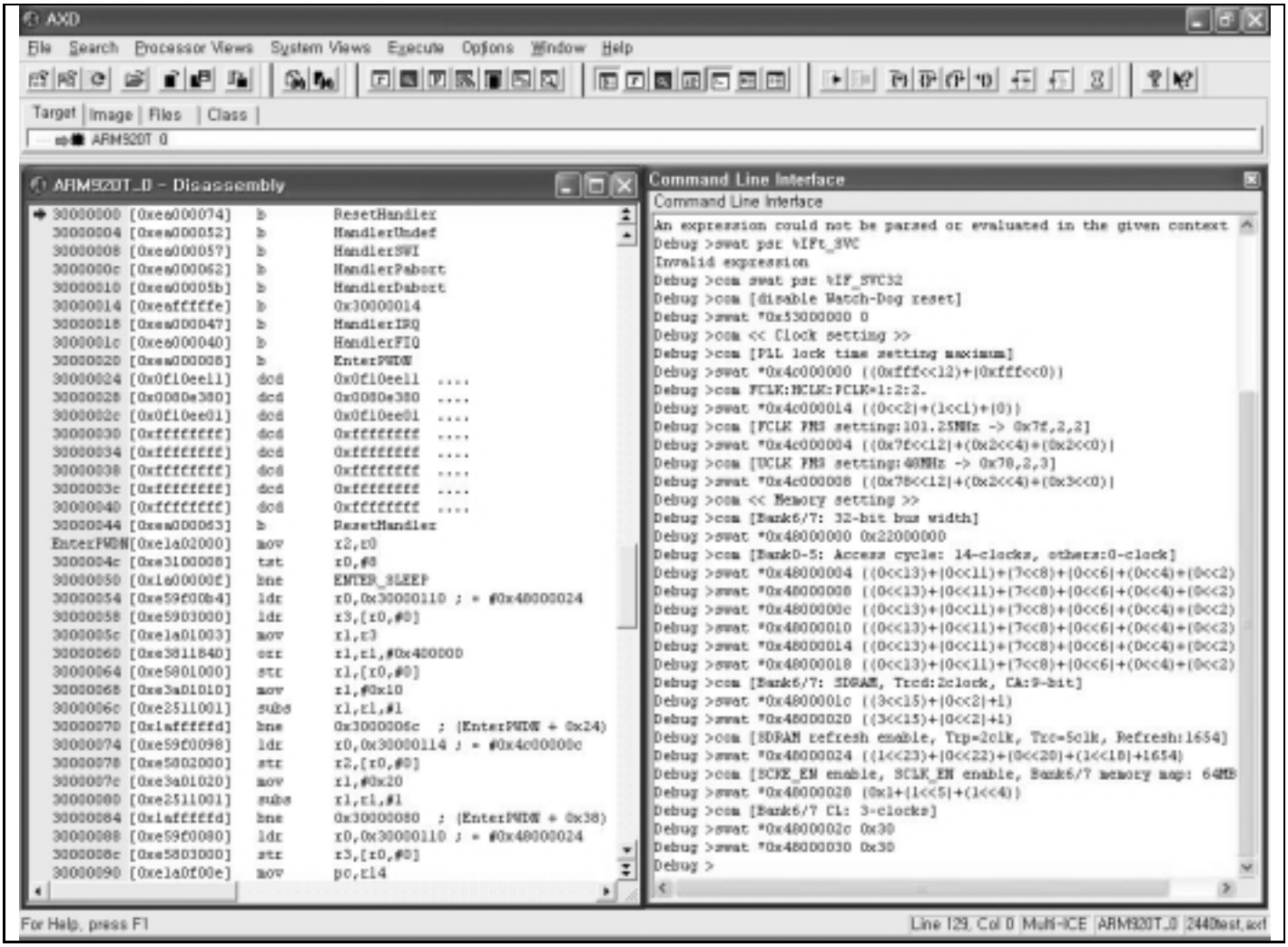
— Write image files to AMD flash memory with UART.
— Write image files to AMD flash memory with Multi-ICE.
— Write image files to AMD flash memory with OPENice32-A900
— Write image files to Intel STRATA flash memory with UART.
— Write image files to Intel STRATA flash memory with Multi-ICE.
— Write image files to Intel STRATA flash memory with OPENice32-A900

WRITING IMAGE FILES TO AMD FLASH MEMORY WITH UART

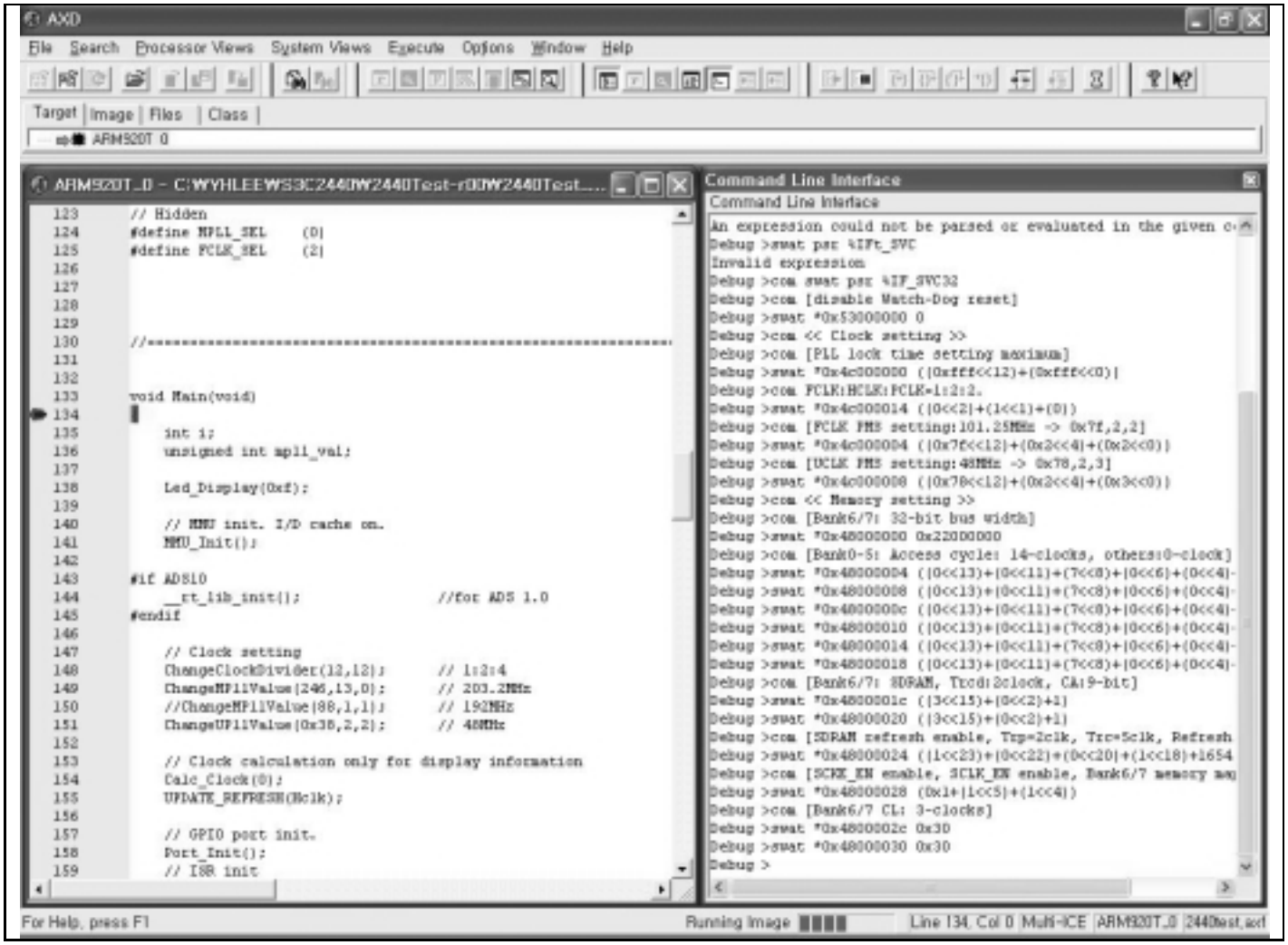
1. Connect MULTI-ICE and execute " 2440norom.ini " file.



2. Load the image file (2440TEST.axf) to execute.

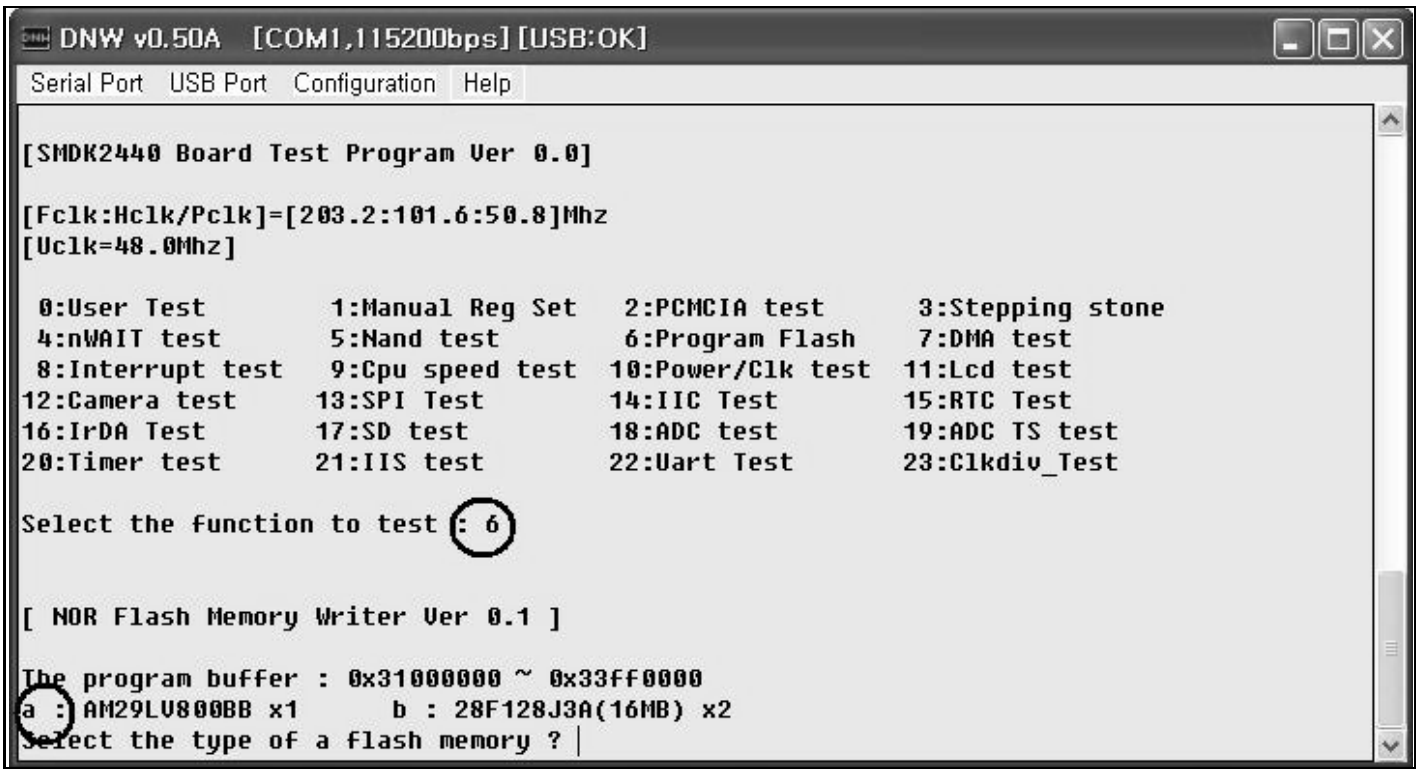


3. Execute 2440TEST code with Go command.



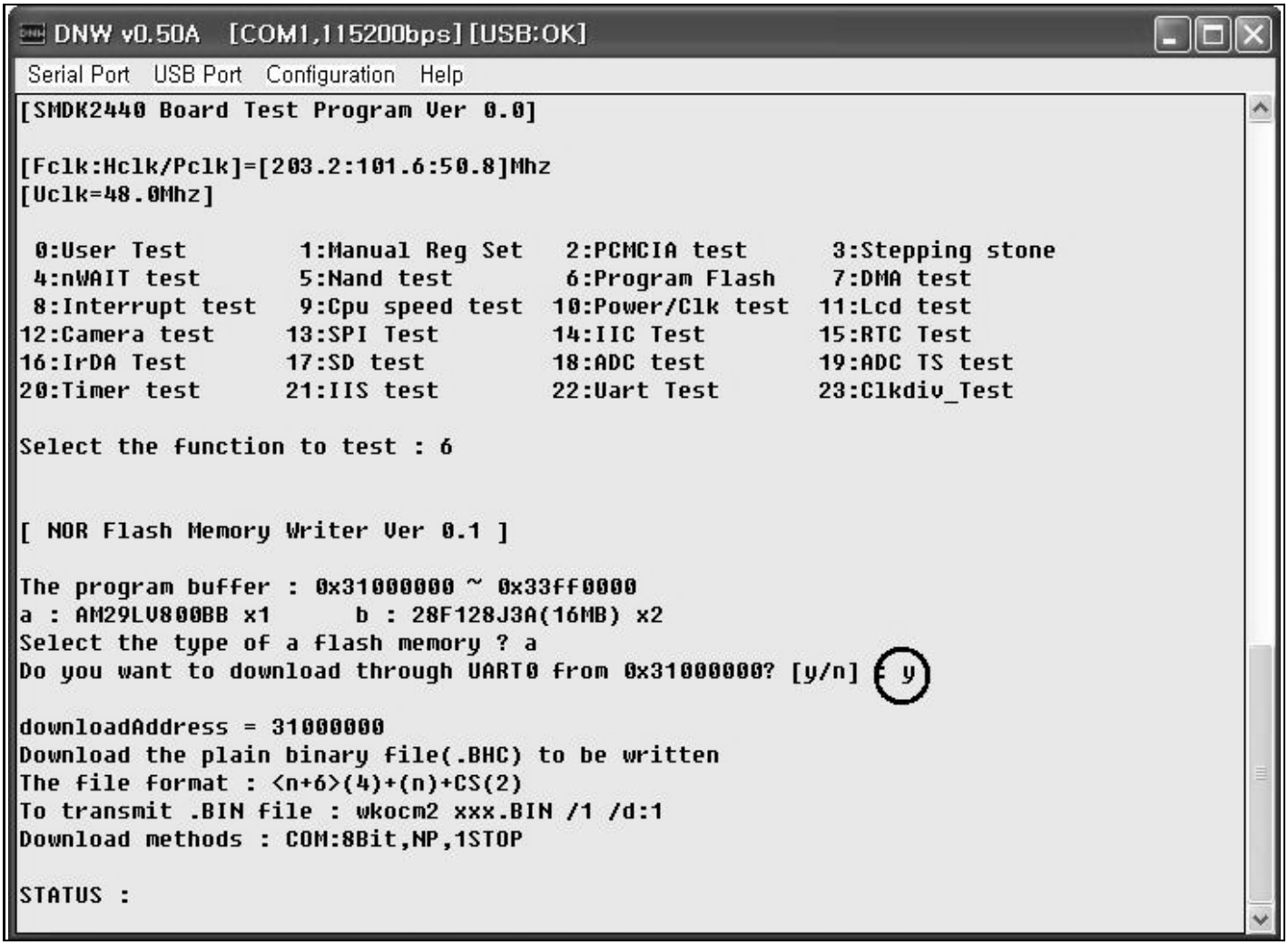
4. Select " 6:Program Flash " on the DNW.

NOTE: If you want to download 2440TEST.bin without MULTI-ICE, then skip the 1, 2 & 3 steps above and download 2440TEST.bin using the DNW (See EXECUTE 2440TEST WITHOUT MULTI-ICE).
After download 2440TEST.bin with the DNW, then you can also see the figure below.



5. Select the type of memory as AM29LV800BB x1 (AMD Flash) by typing ‘a’.
6. Select whether you download through UART or MULTI-ICE.

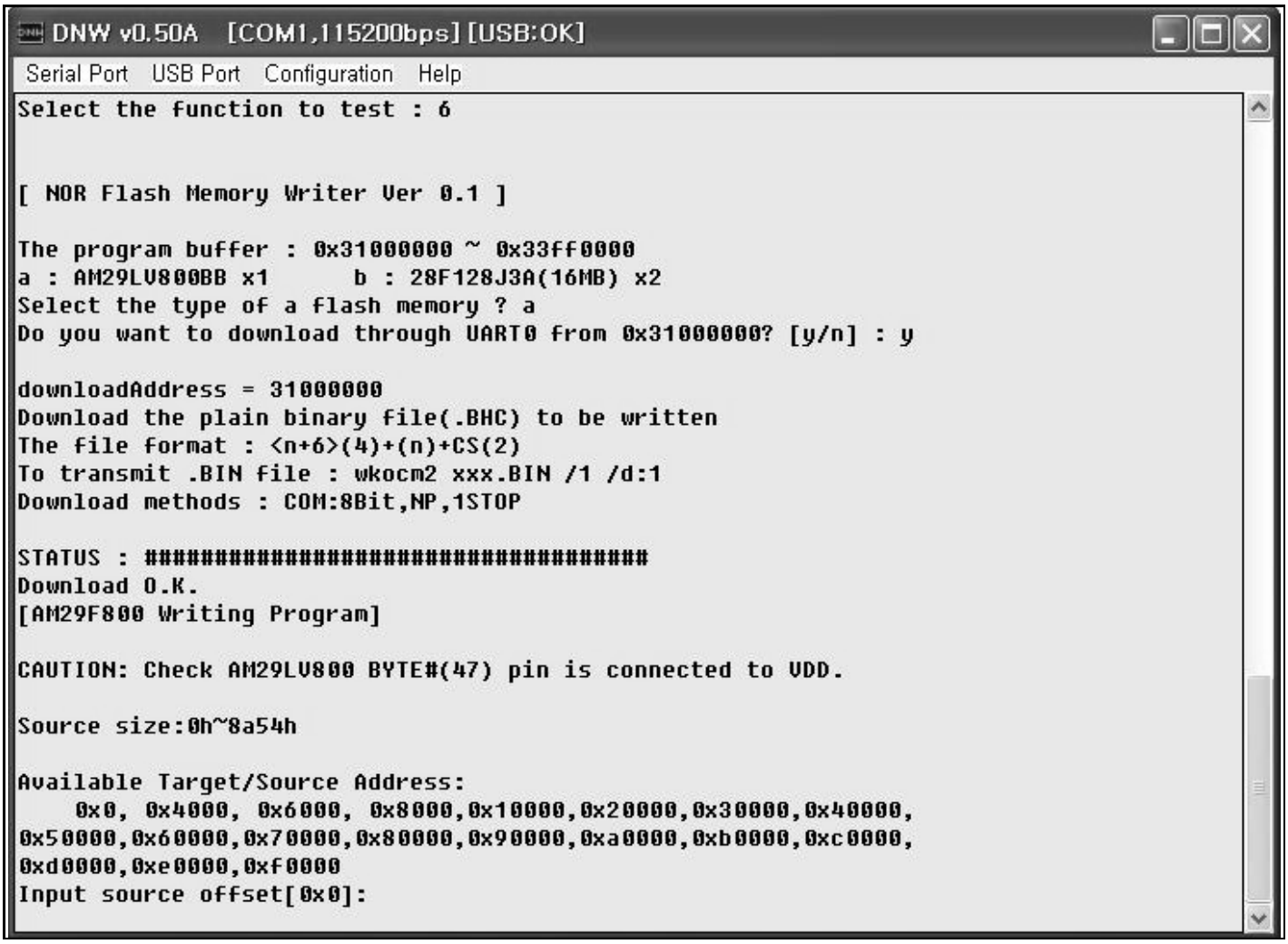
— Type ‘y’ then you can download target files through UART. See the figure below.



7. Download target files with the DNW by selecting Transmit menu from Serial Port.

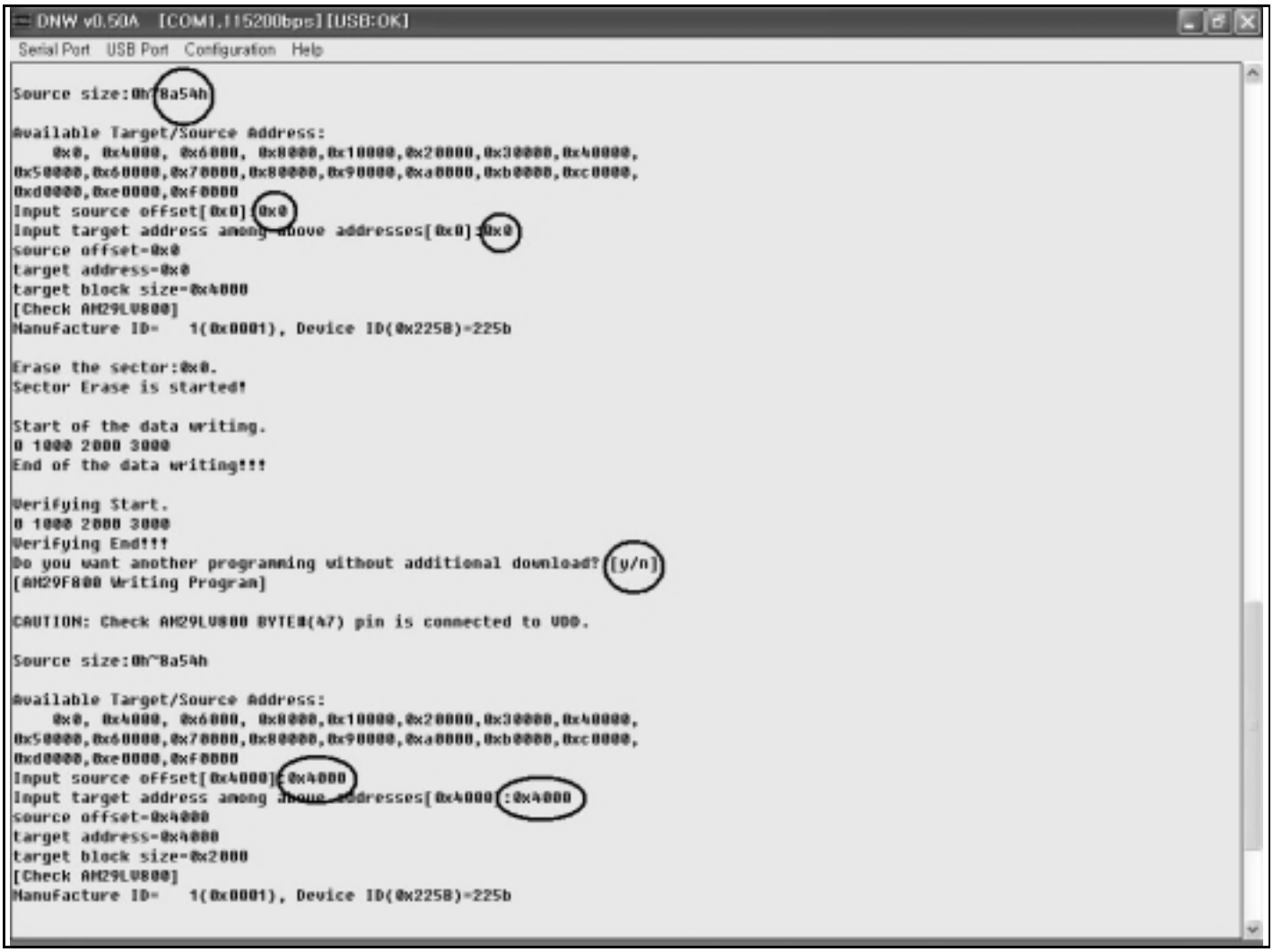
— Serial Port → Transmit

— Select and Download a target file.



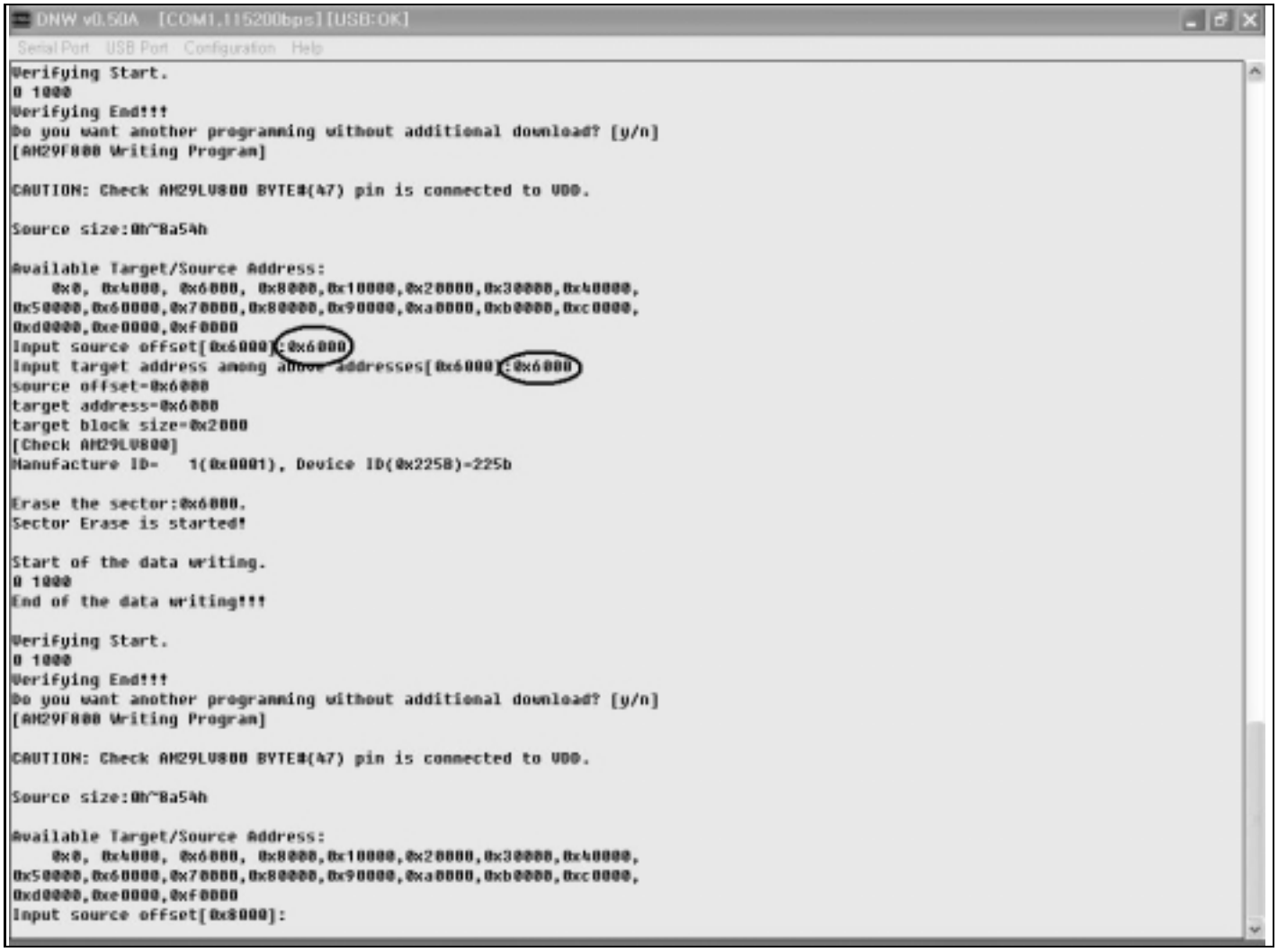
8. Write input source offset and target offset and type ‘y’ repeatedly until the source size is reached. The example below shows how to setting source offset and target offset. The size of target file is 0x8a54 bytes.

— Write source offset ‘ 0x0 ’ and write target address ‘ 0x0 ’, and then type ‘y’.

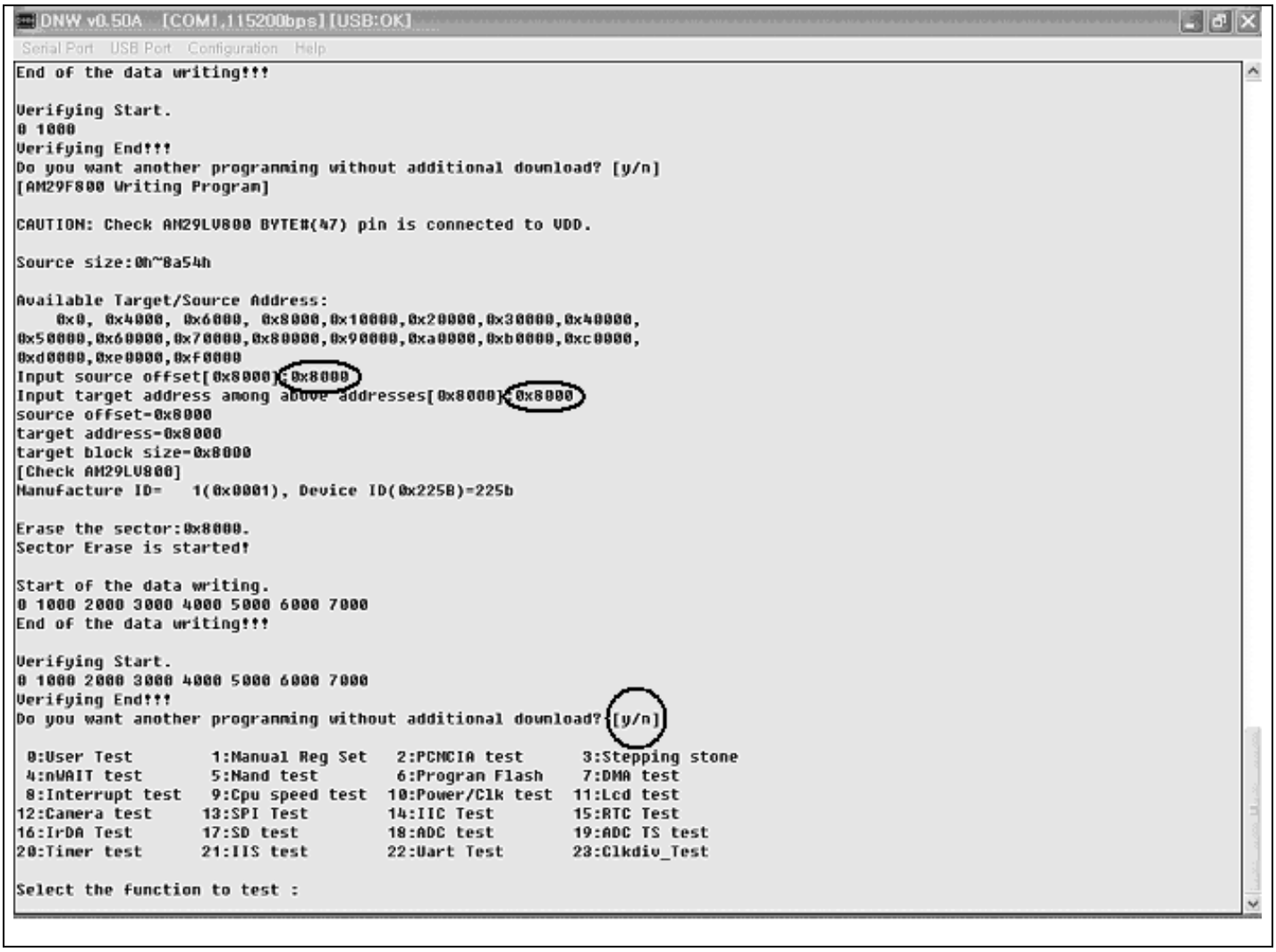


— Write source offset ‘ 0x4000 ’ and write target address ‘ 0x4000 ’, and then type ‘y’.

— Write source offset ‘ 0x6000 ’ and write target address ‘ 0x6000 ’, and then type ‘y’.



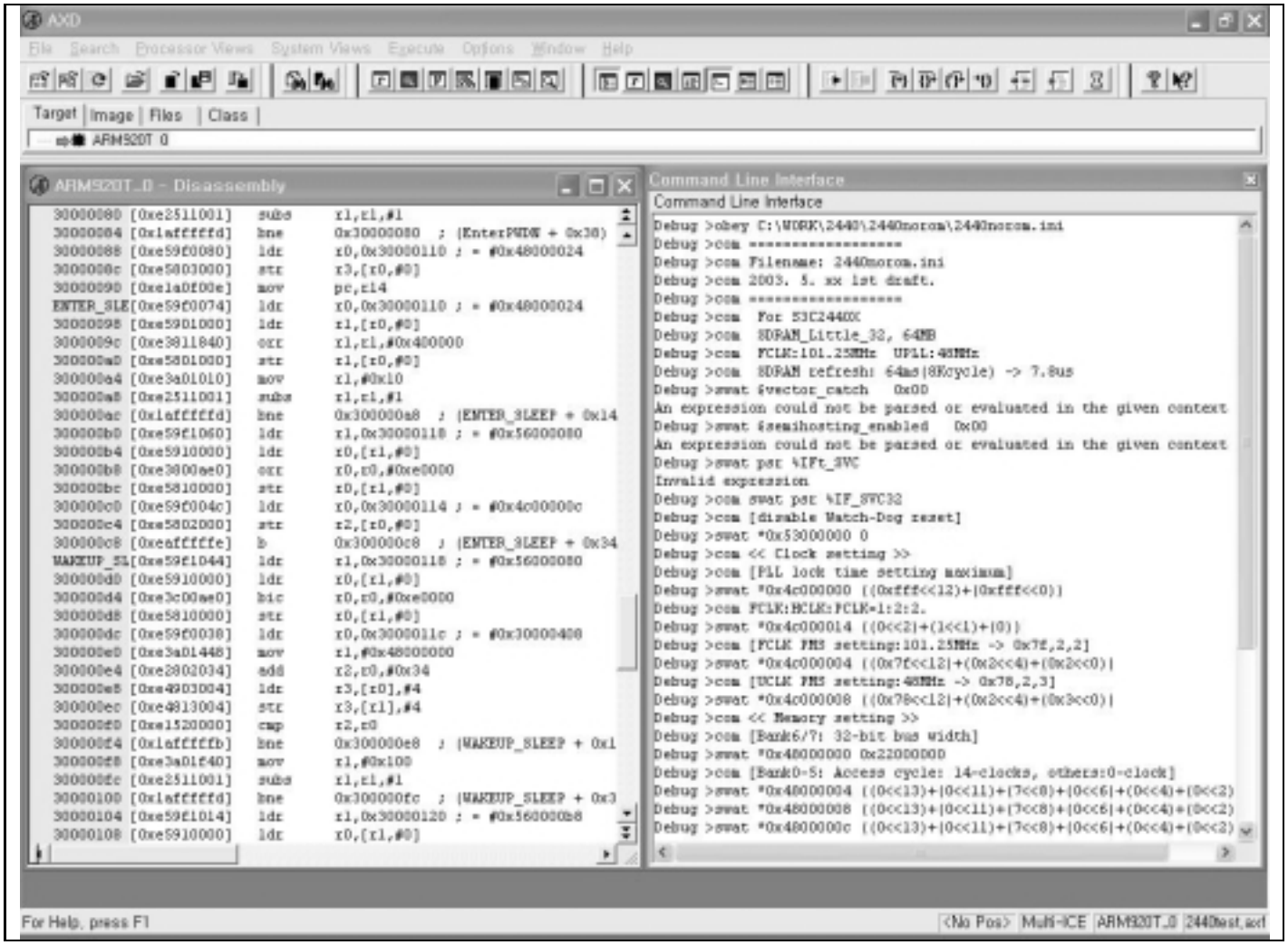
— Write source offset ‘ 0x8000 ’ and write target address ‘ 0x8000 ’, and then type ‘n’.



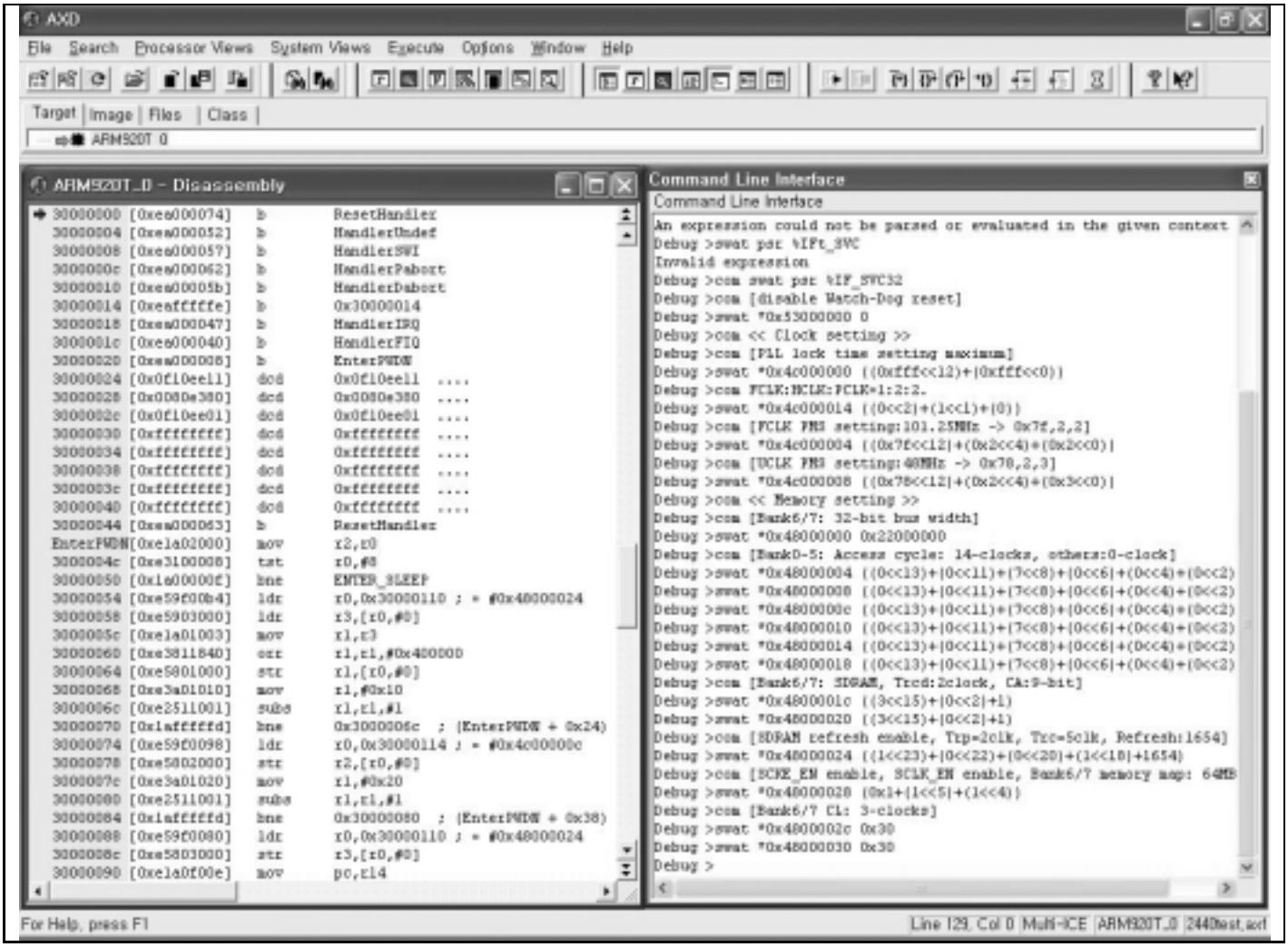
9. Turn the SMDK2440 off and then on.

WRITING IMAGE FILES TO AMD FLASH MEMORY WITH MULTI-ICE

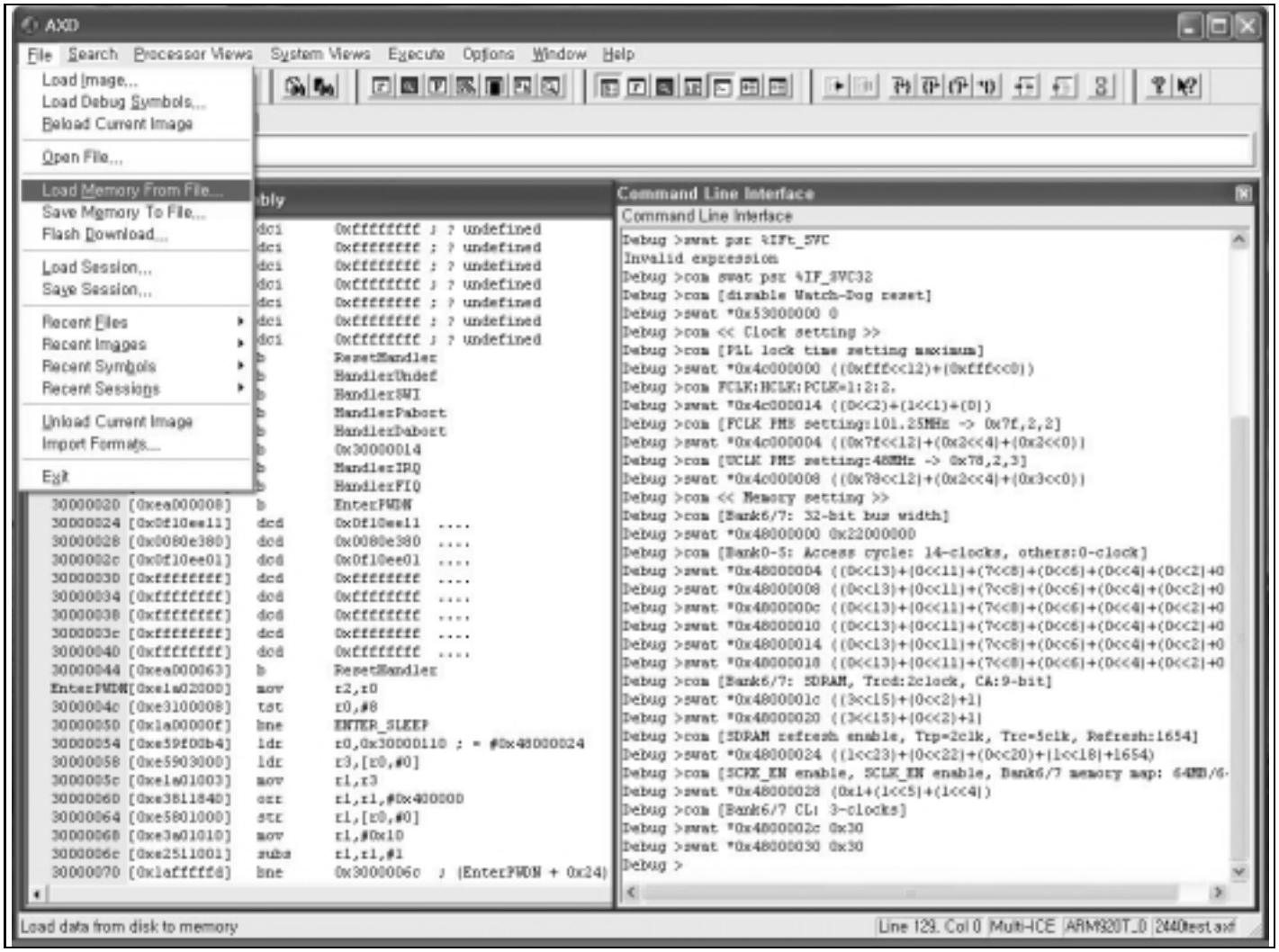
1. Connect MULTI-ICE and execute " 2440norom.ini " file.



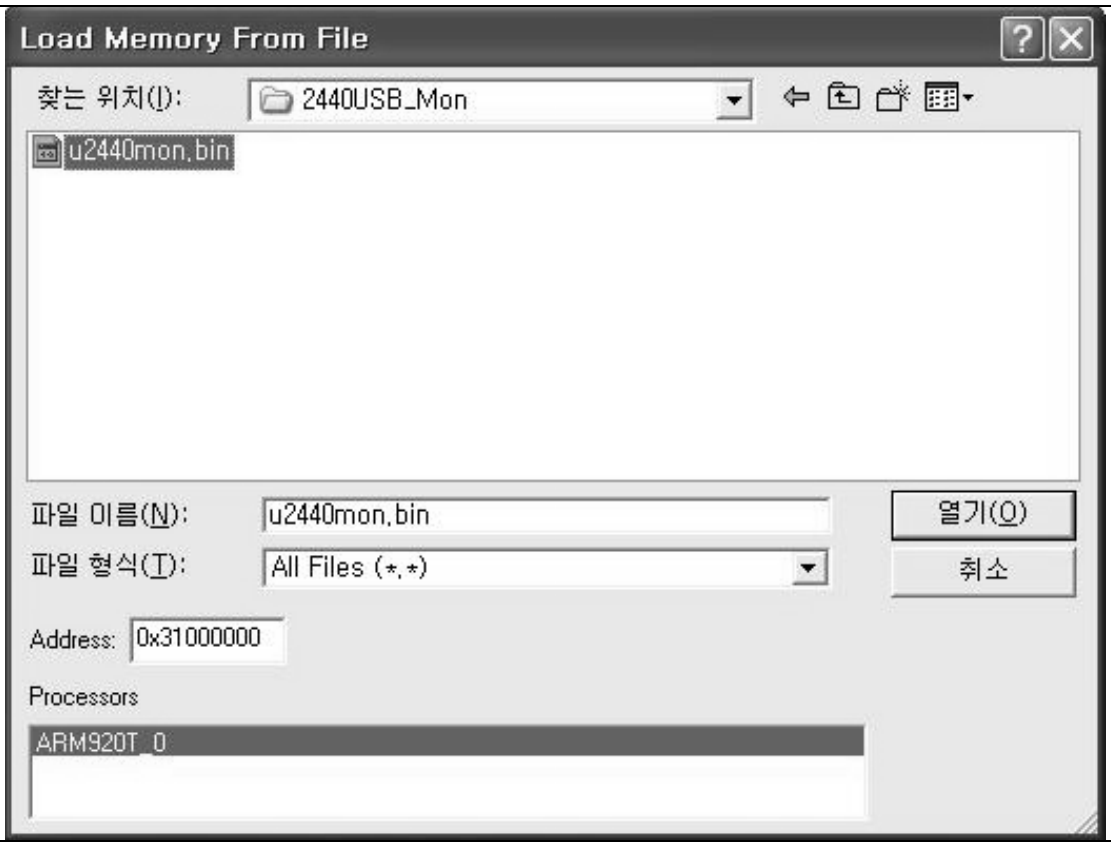
2. Load the image file (2440TEST.axf) to execute.



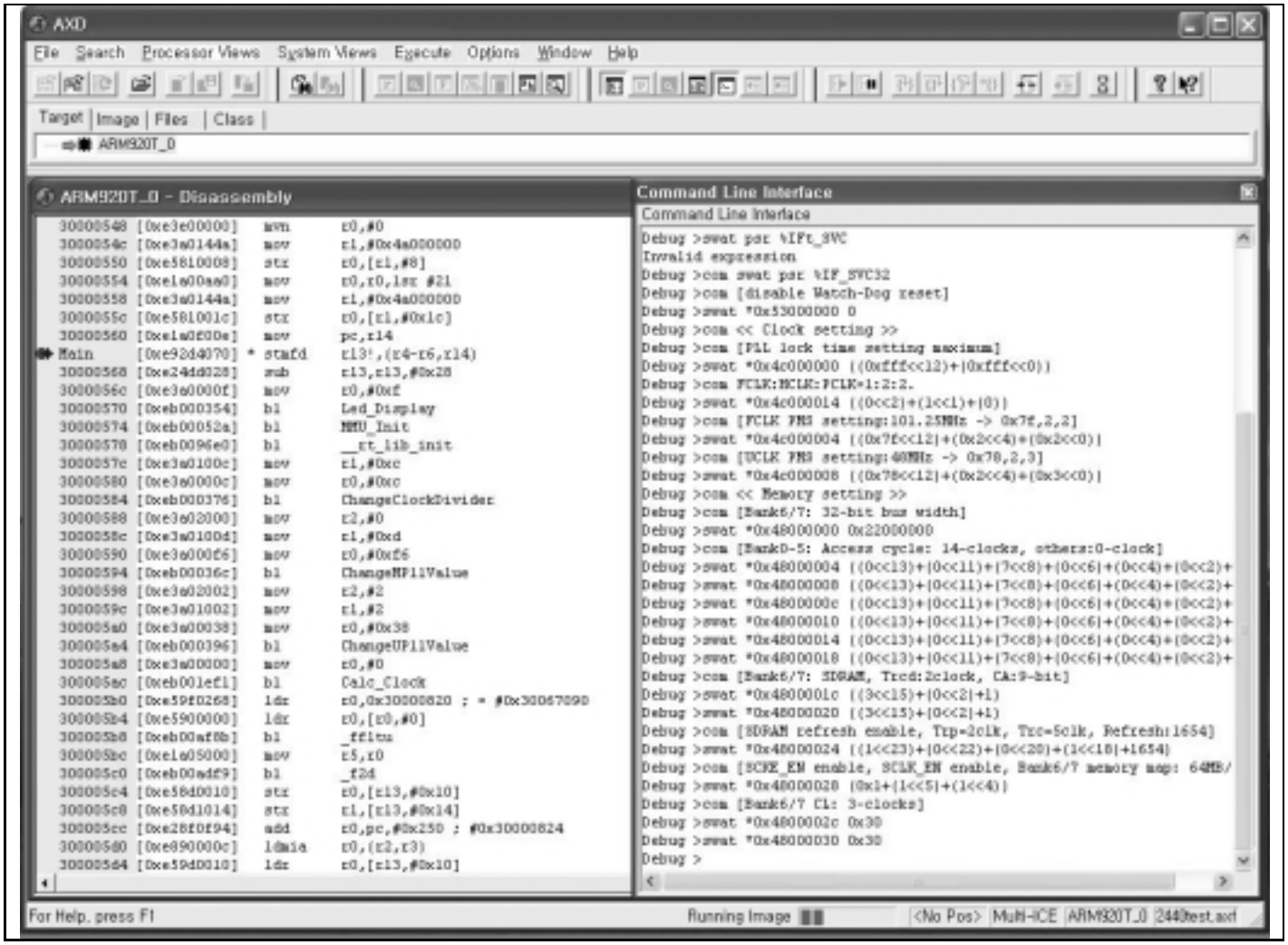
3. Select Load Memory From File… on the file menu of AXD.



— Get a target file to 0x31000000 in SMDK2440 Board.

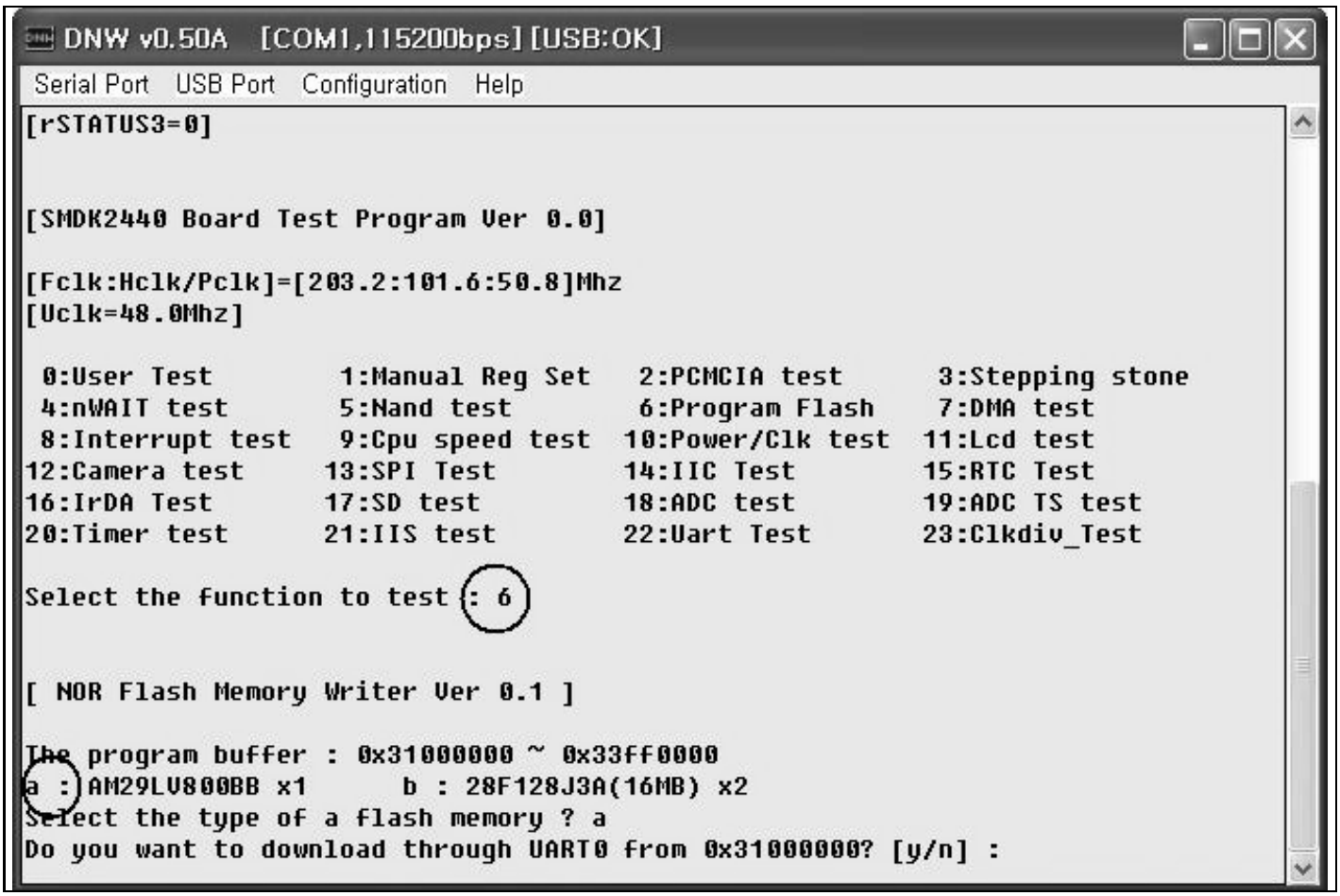


4. Execute 2440TEST.axf file with GO command.



5. Select " 6:Program Flash" on the DNW.

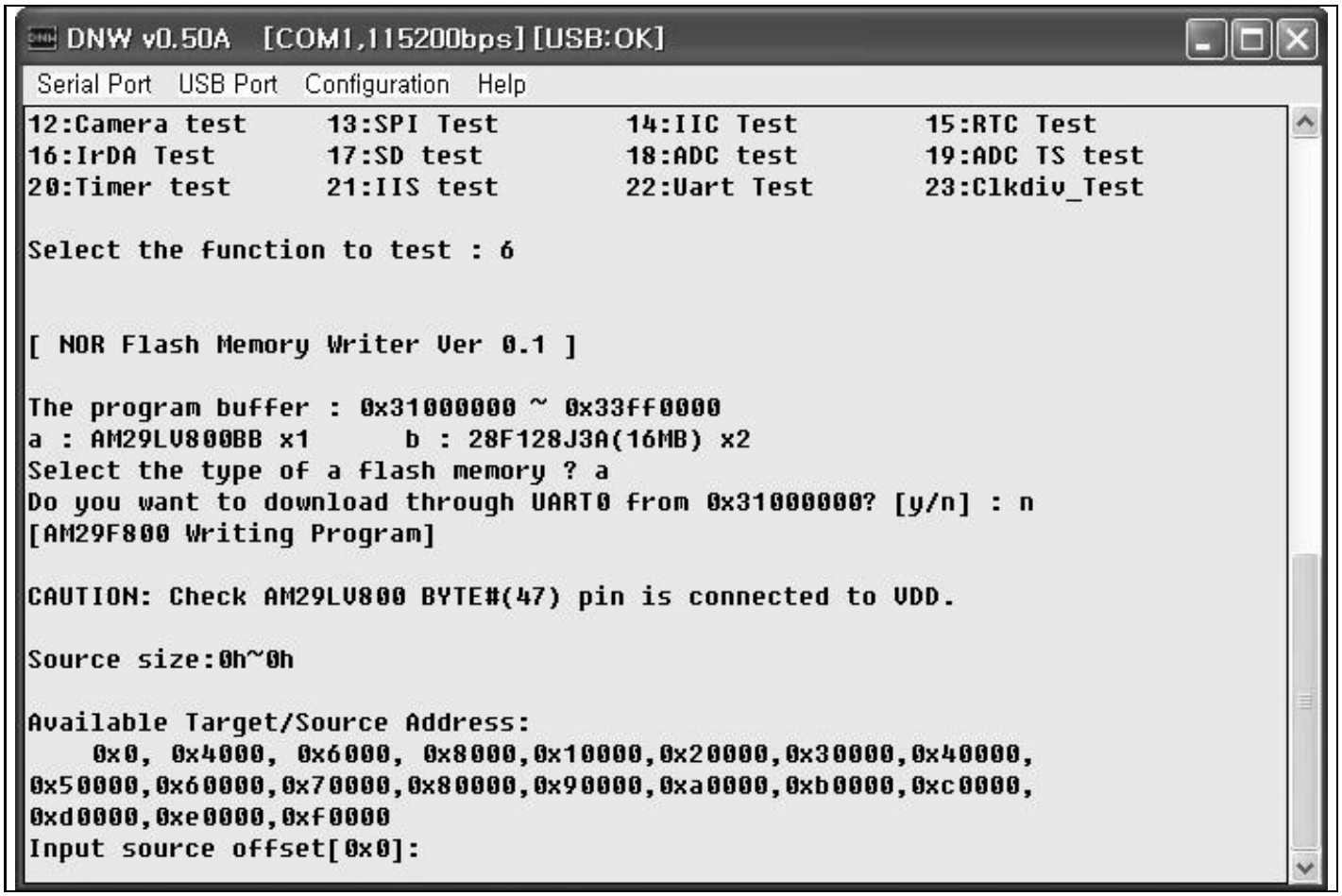
NOTE: If you want to download 2440TEST.bin without MULTI-ICE, then skip the 1, 2 & 3 steps above and download 2440TEST.bin using the DNW (See EXECUTE 2440TEST WITHOUT MULTI-ICE).
After downloading 2440TEST.bin with the DNW, then you can also see the figure below.



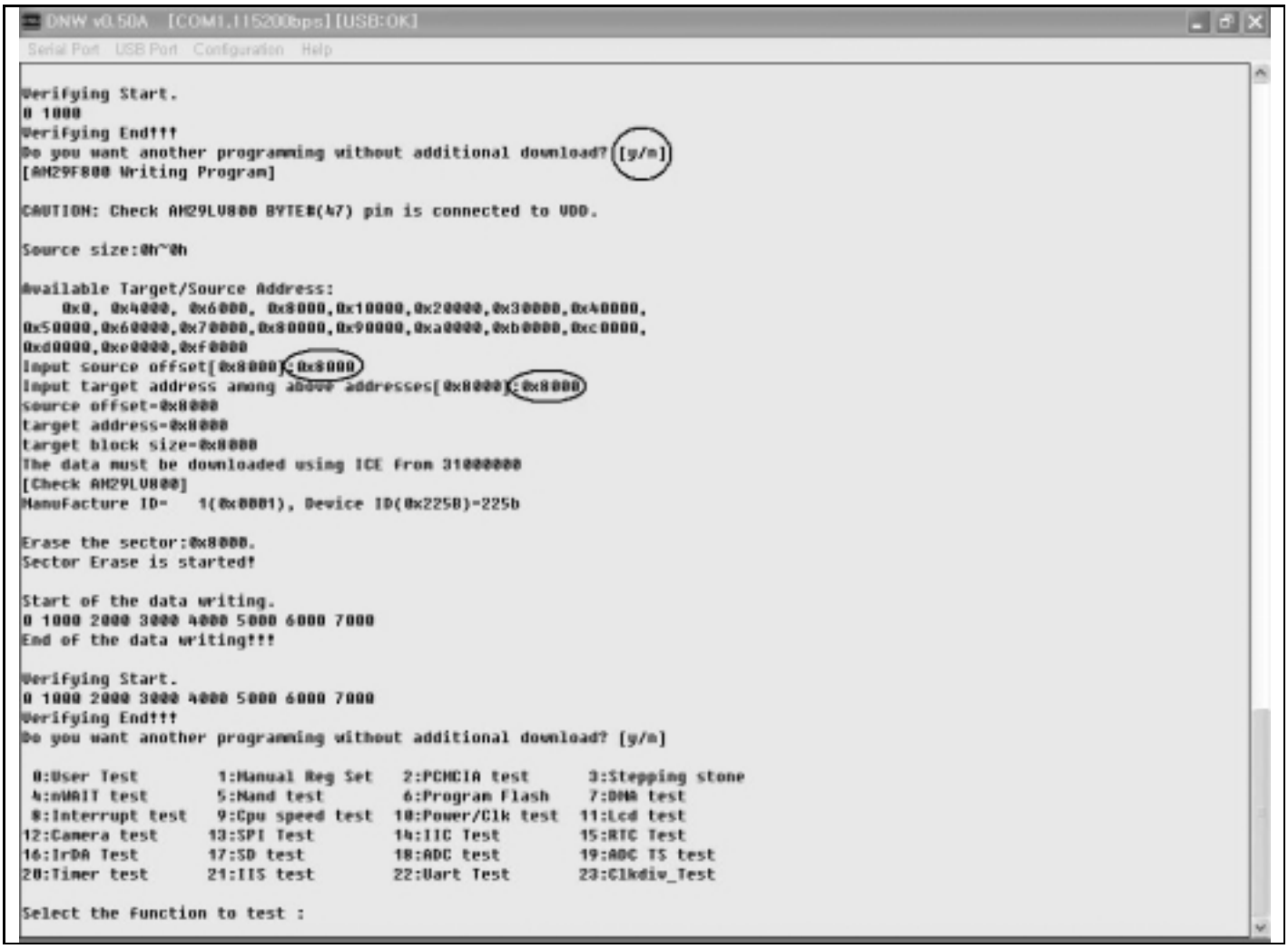
6. Select the type of memory as AM29LV800BB (AMD) by typing ‘a’.

7. Select whether you download through UART0 or MULTI-ICE.

— Type ‘n’ then you can see the figure below in the DNW.



8. Write input source offset and target offset and type ‘y’ repeatedly until the source size is reached. See Writing the image file to AMD Flash memory with UART.

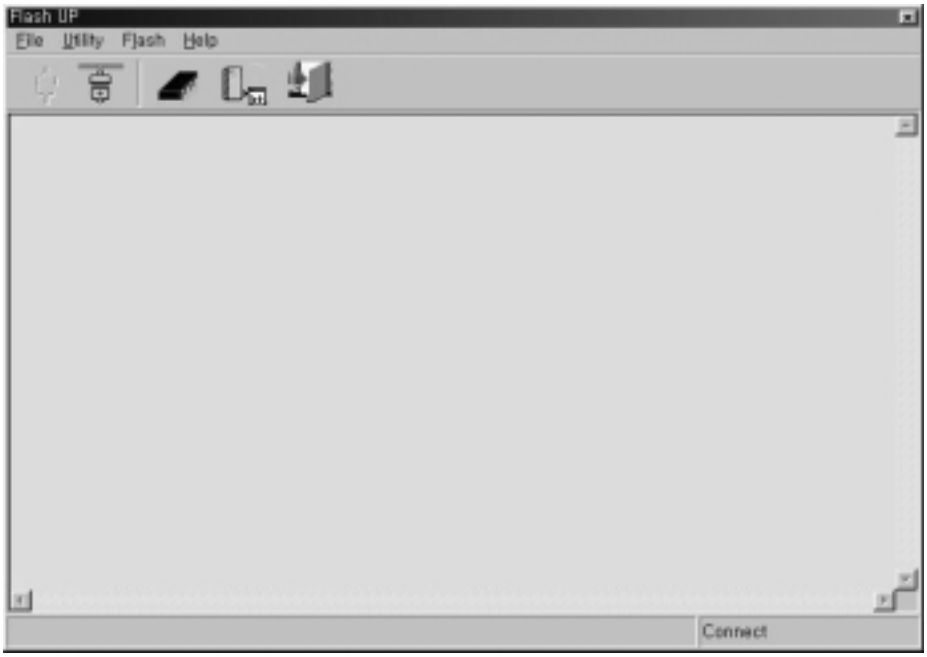


9. Turn off and on the SMDK2440.

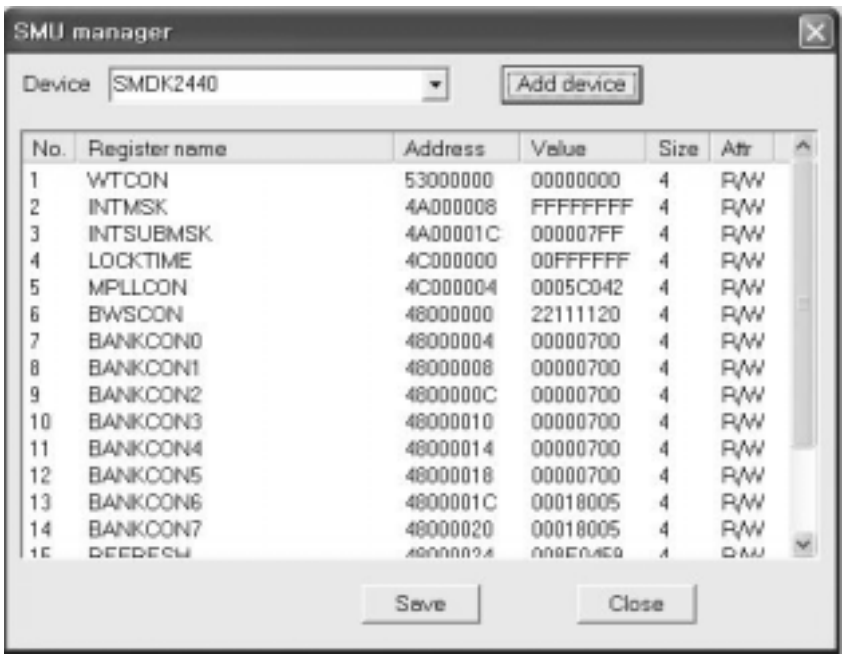
WRITING IMAGE FILES TO AMD FLASH MEMORY WITH OPENICE32-A900

OPENice32-A900 can write image to AMD Flash memory as Multi-ICE. However, OPENice32-A900 provide a Flash Write Program that is easy to use and don't require ARM SDT/ADS debugger nor DNW. For more information on the program, refer to OPENice32-A900 manual or contact AIJI System (www.aijisystem.com).

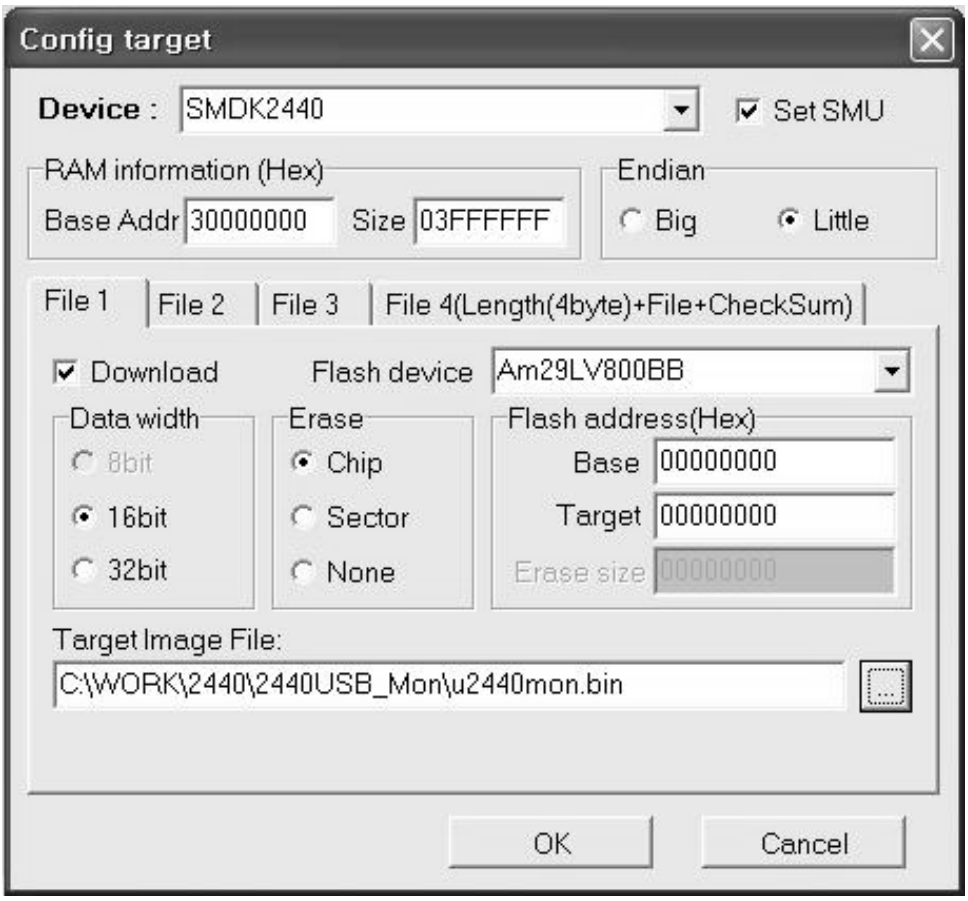
1. Connect OPENice32-A900 to PC through USB and to SMDK2440 board with 20pin Cable.
2. Run the Flash Write program and select Connect MDS from the File menu.



3. Select SMU Manger from the utility menu and choose a device file, SMDK2440. It is used to initialize the system registers in case of there is no boot ROM. If you can't find the file, download the device file SMDK2410 instead of SMDK2440. After that, edit each value if necessary.



4. Select Config.. from the Flash menu and Set the write options as followings



— Device: SMDK2440

— Set SMU: Checked

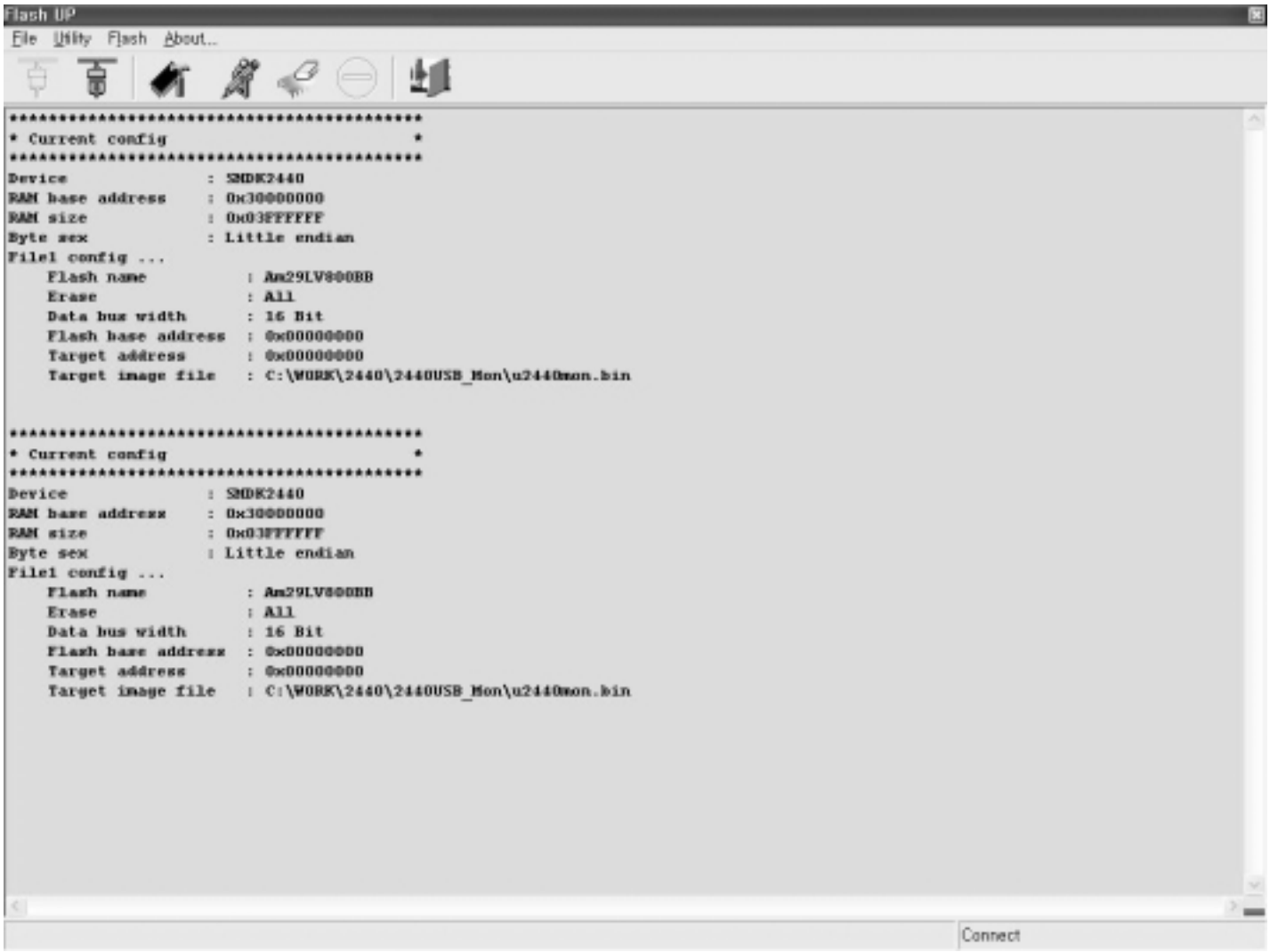
— RAM Information: Base Address:30000000 Size: 3FFFFFF

— Endian: Little

— File 1 page
Download: checked
Flash Device Name:AM29LV800BB
Erase:Chip
Data Bus width: 16bit
Flash Address: Base Address: 0 Target Address:0

— Target Image File: u2440mon.bin

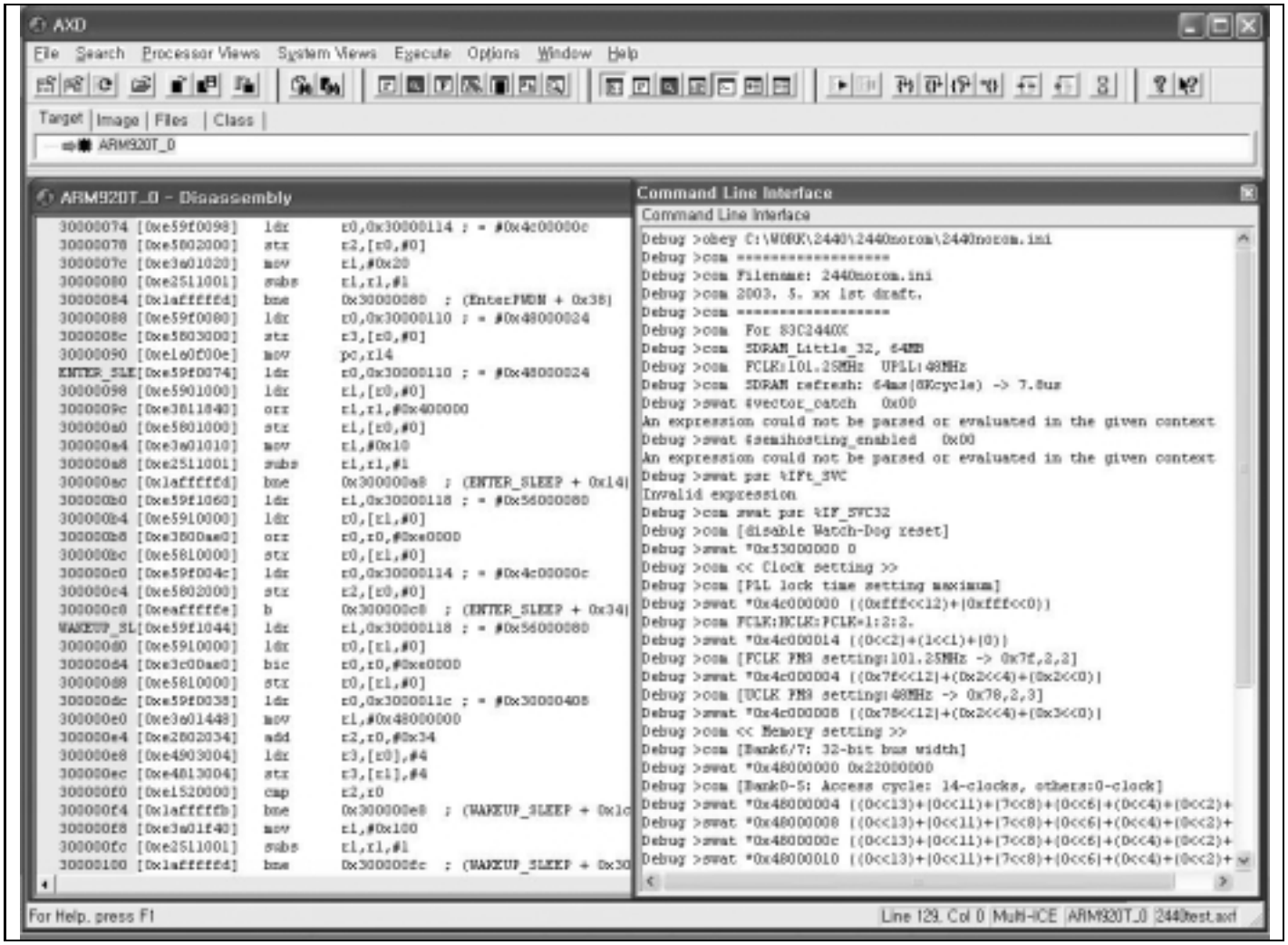
5. Click OK. Then the current configuration is displayed in the window.



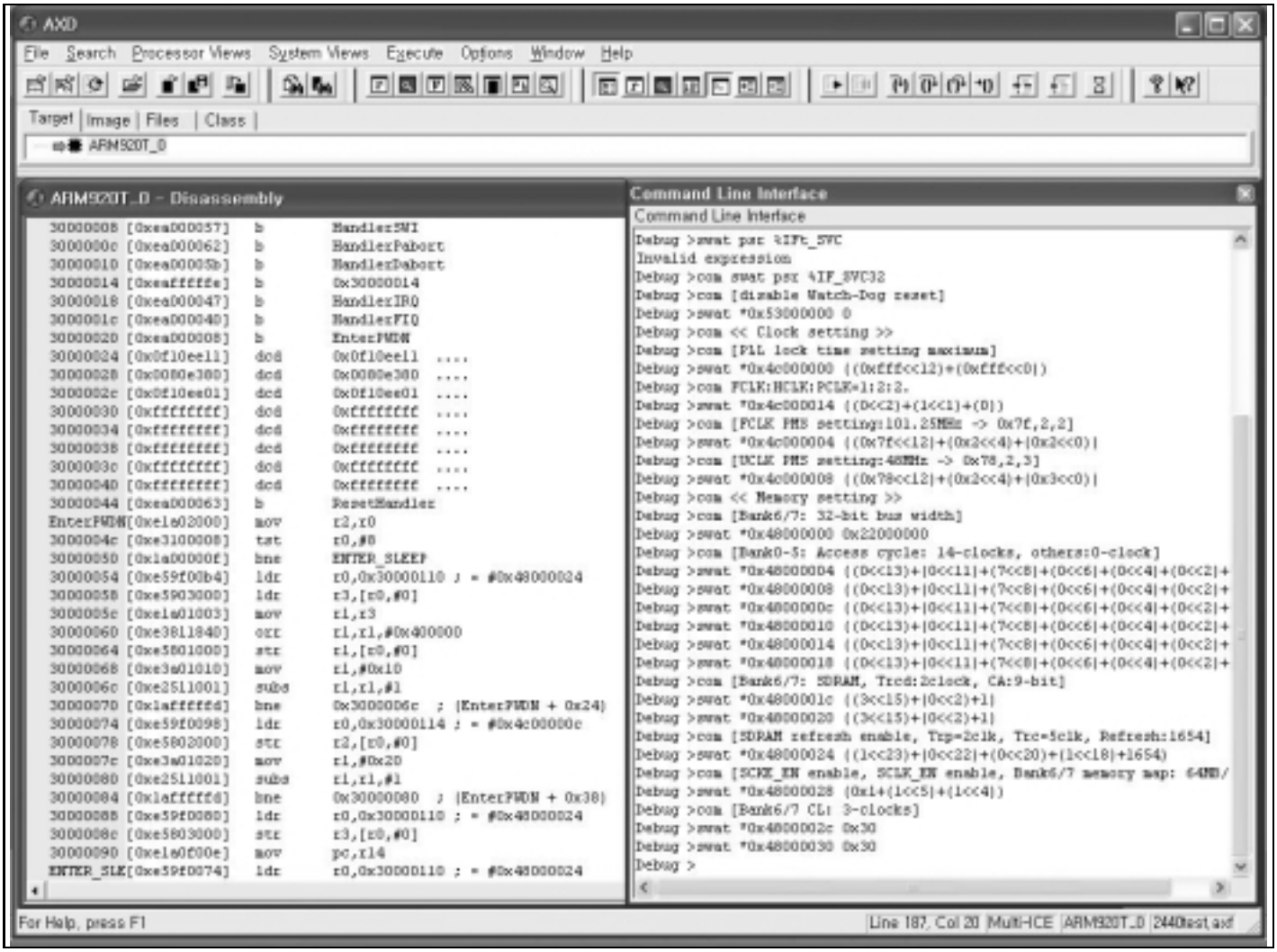
6. Select Write from the Flash Menu. Then it starts to erase the specified area of AMD Flash and write the image to the Flash memory. It takes about 10 second.

WRITING IMAGE FILES TO INTEL STRATA FLASH MEMORY WITH UART

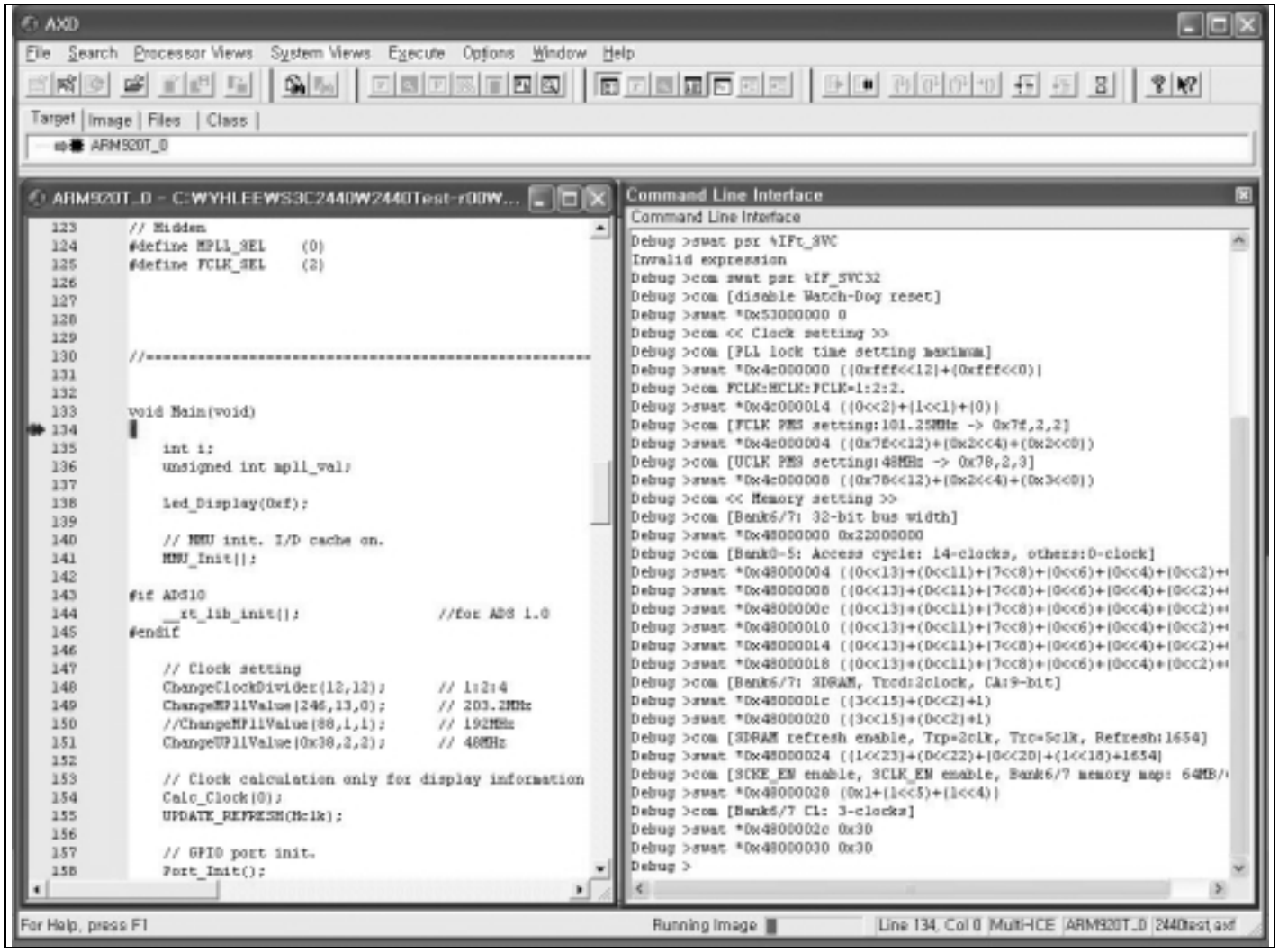
1. Connect MULTI-ICE and execute " 2440norom.ini " file.



2. Load the image file (2440TEST.axf) to execute.

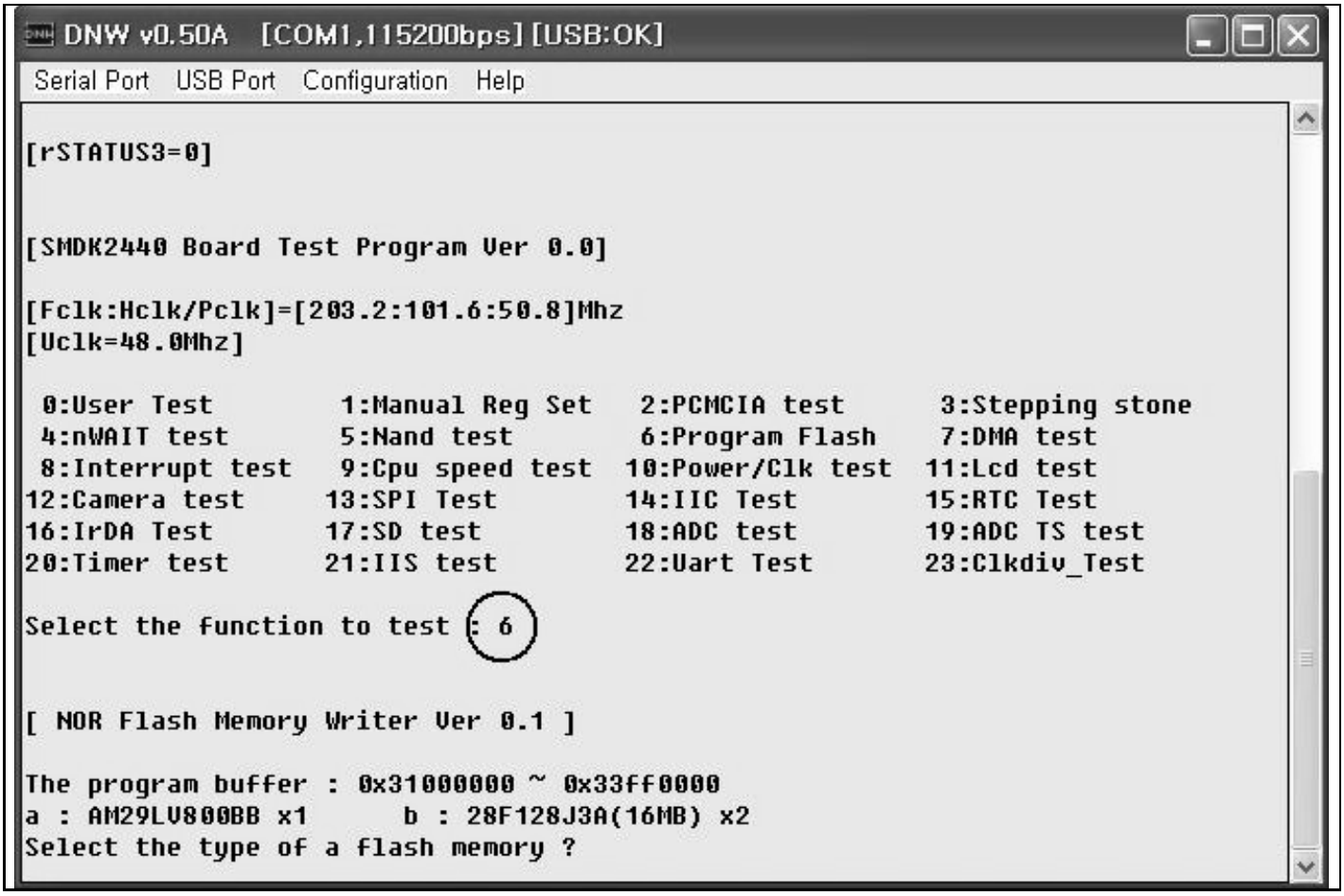


3. Execute 2440TEST code with Go command.



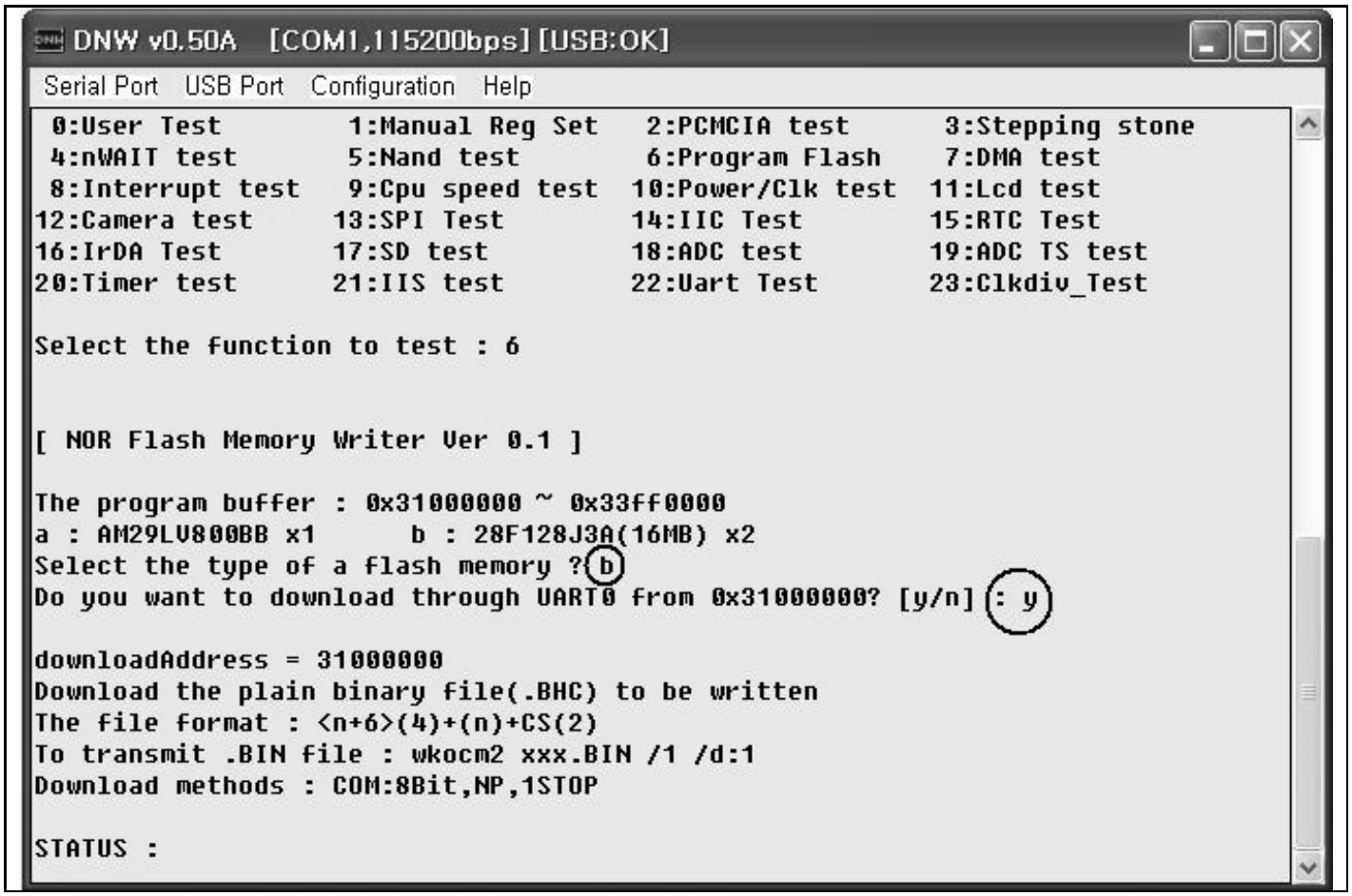
4. Select " 6:Program Flash" on the DNW.

NOTE: If you want to download 2440TEST.bin without MULTI-ICE, then skip the 1, 2 & 3 steps above and download 2440TEST.bin using the DNW (See EXECUTE 2440TEST WITHOUT MULTI-ICE).
After downloading 2440TEST.bin with the DNW, then you can also see the figure below.



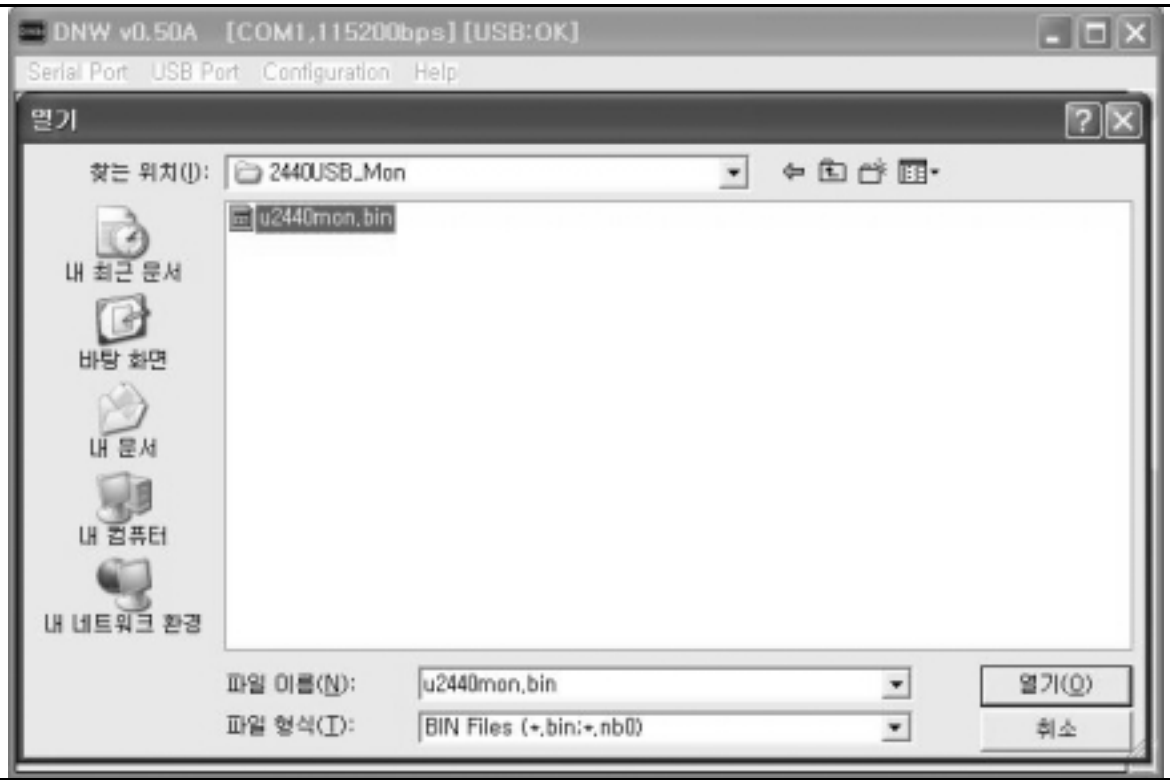
5. Select the type of memory as 28F128J3A (INTEL STRATA flash) by typing 'b'.
6. Select whether you download through UART0 or MULTI-ICE.

— Type 'y' then you can download a target file through UART. See the figure below.

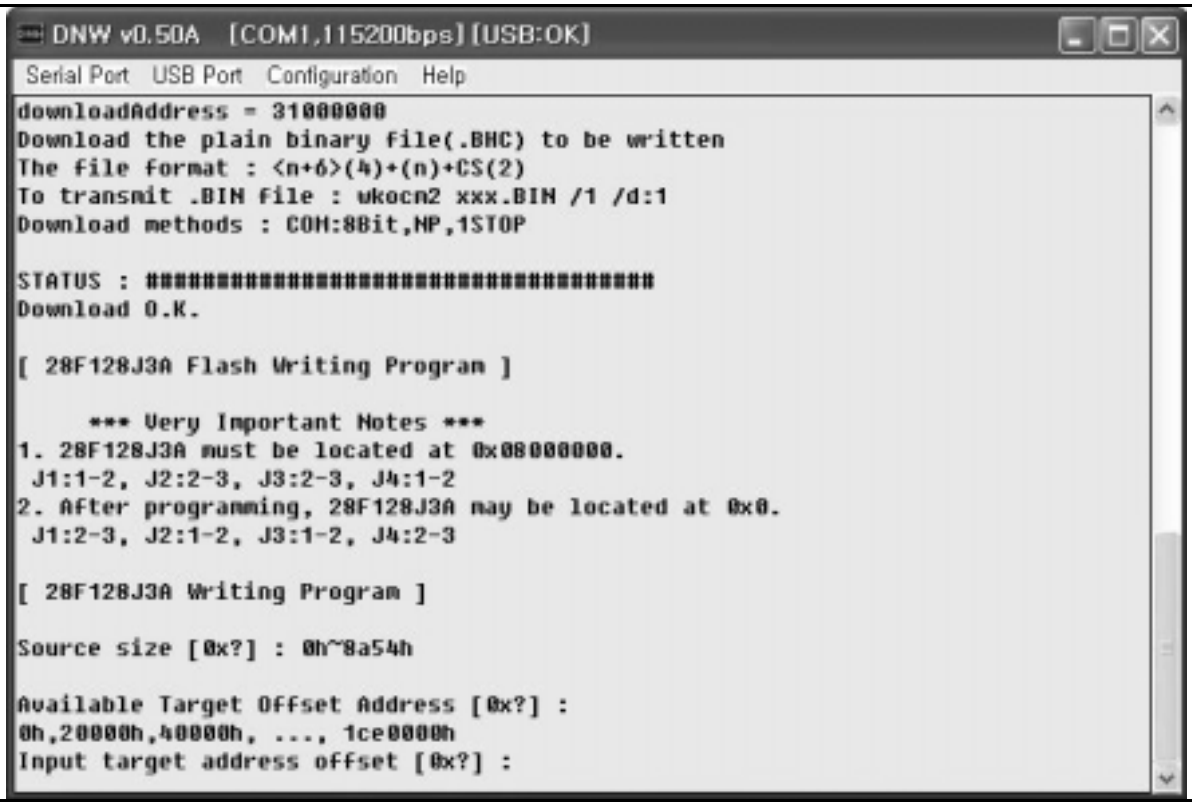


7. Download a target file with the DNW by selecting Transmit menu from Serial Port.

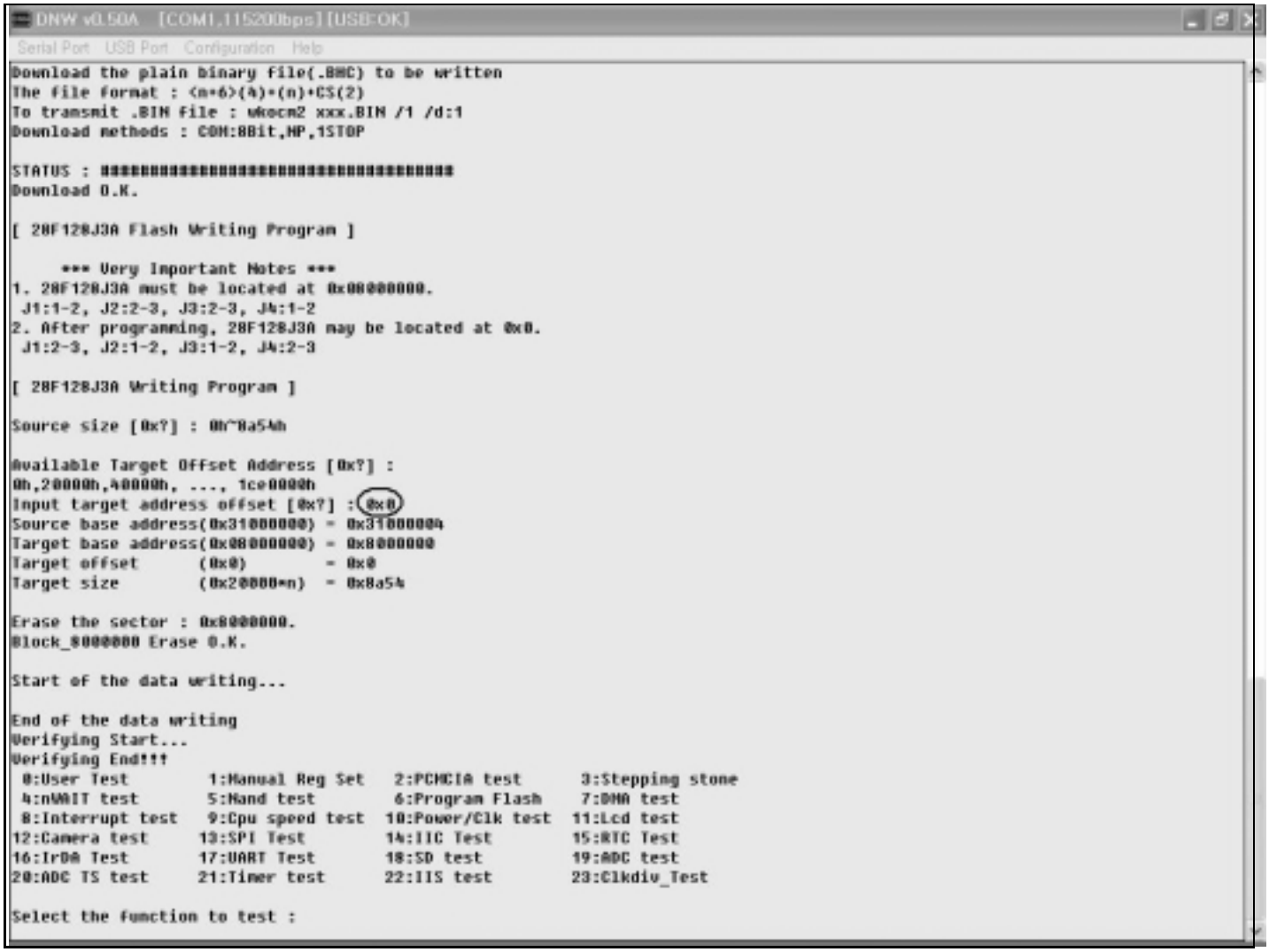
— Serial Port → Transmit



— Select and Download a target file.



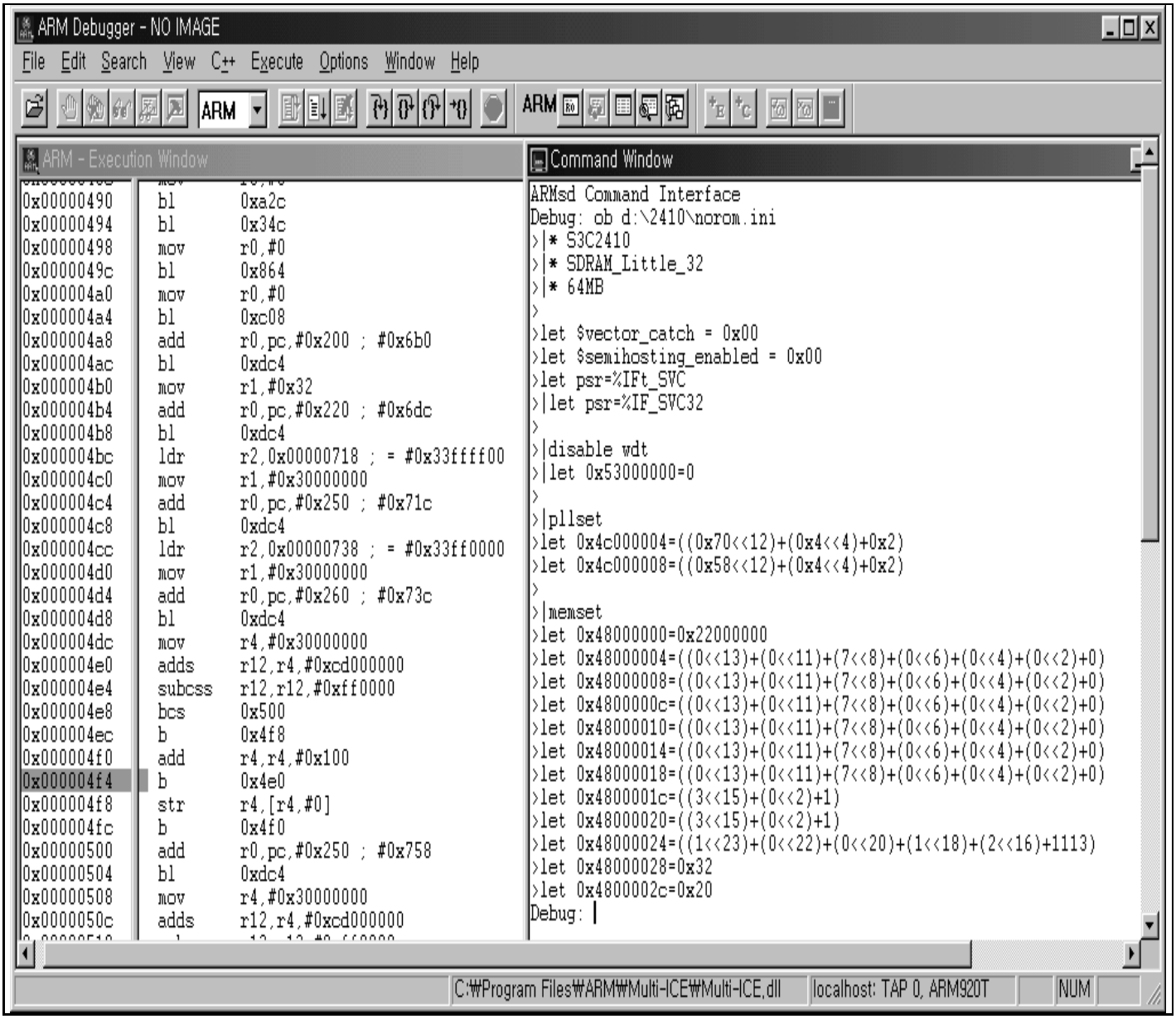
8. Write input target-offset address.



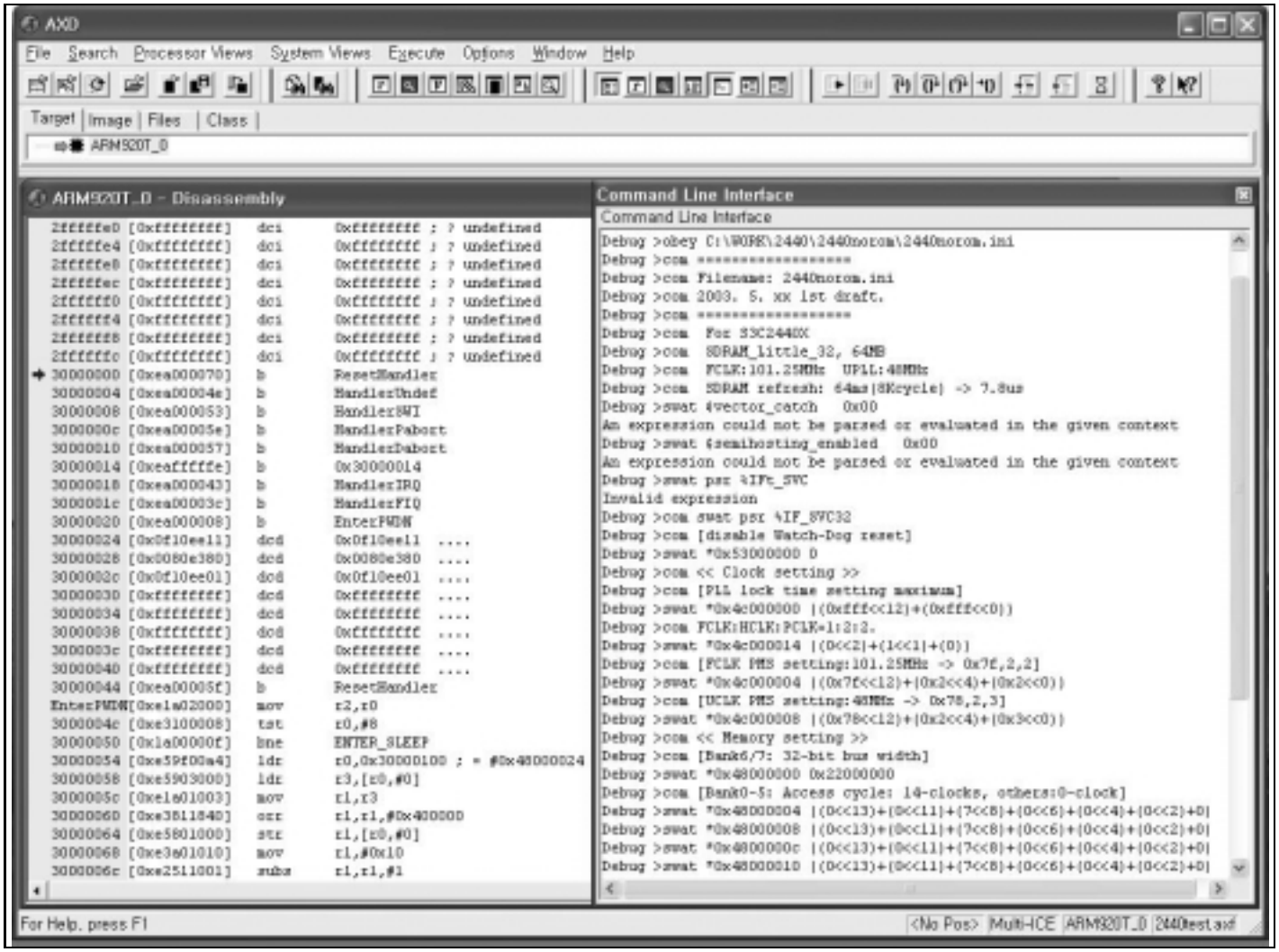
9. Turn the SMDK2440 off and again on.

WRITING IMAGE FILES TO INTEL STRATA FLASH MEMORY WITH MULTI-ICE

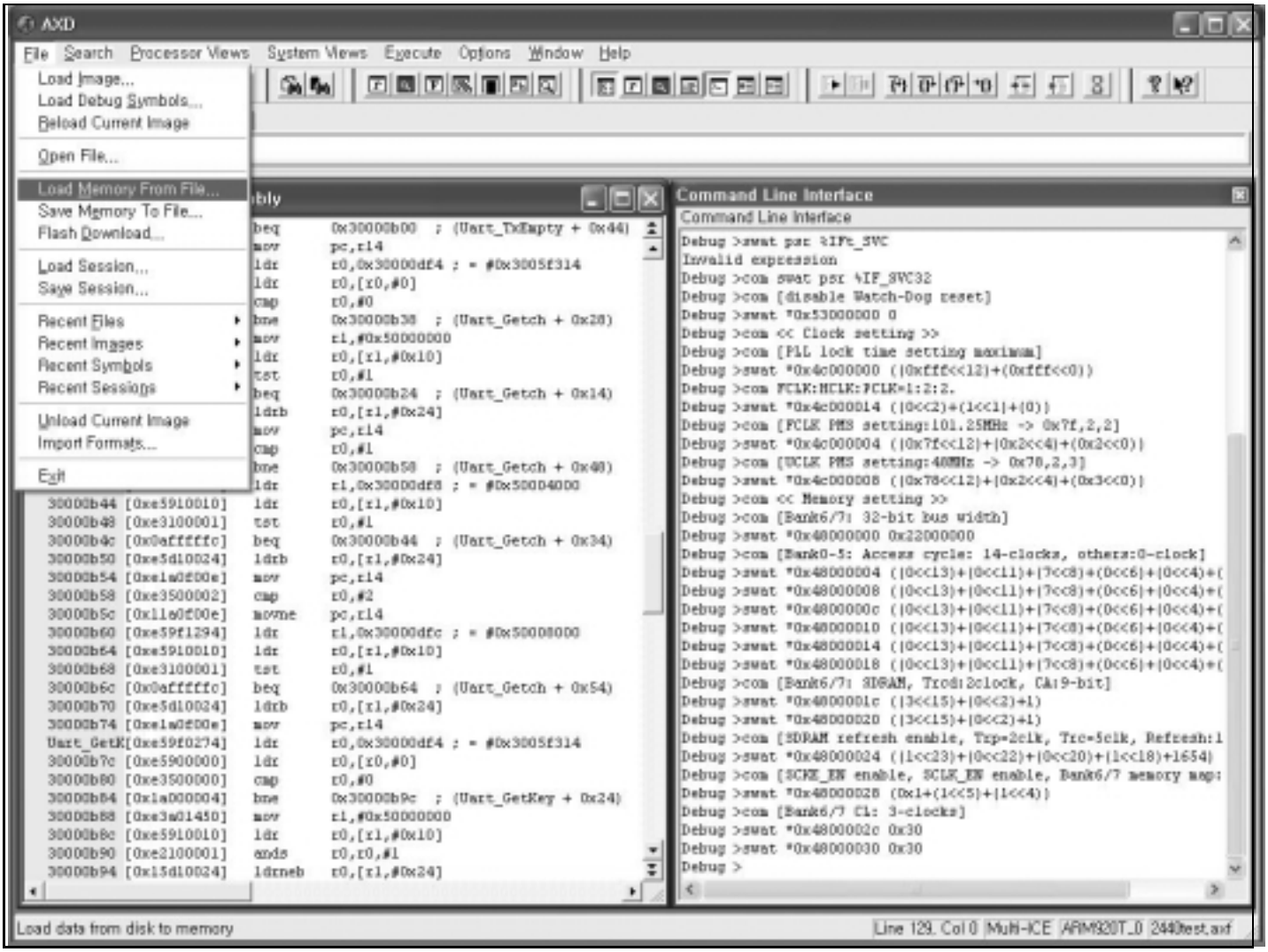
1. Connect MULTI-ICE and execute “ norom.ini “ file.



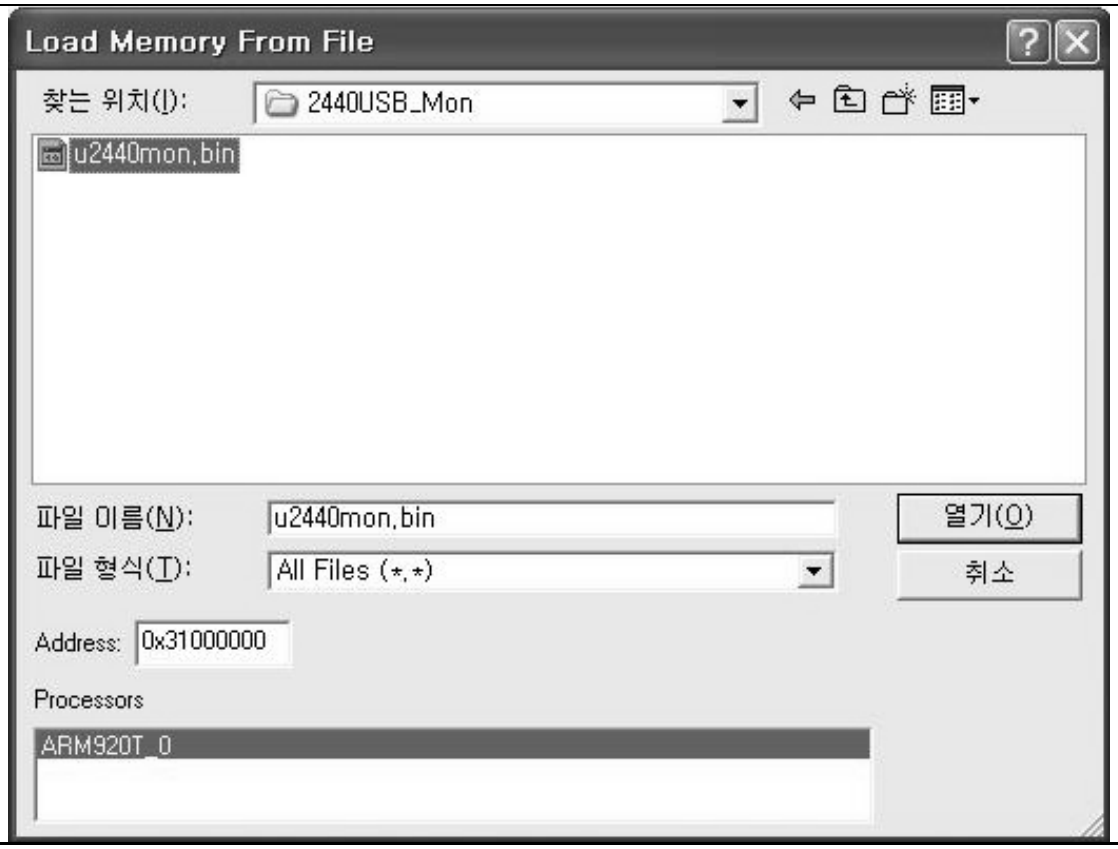
2. Load the image file (2440TEST.axf) to execute.



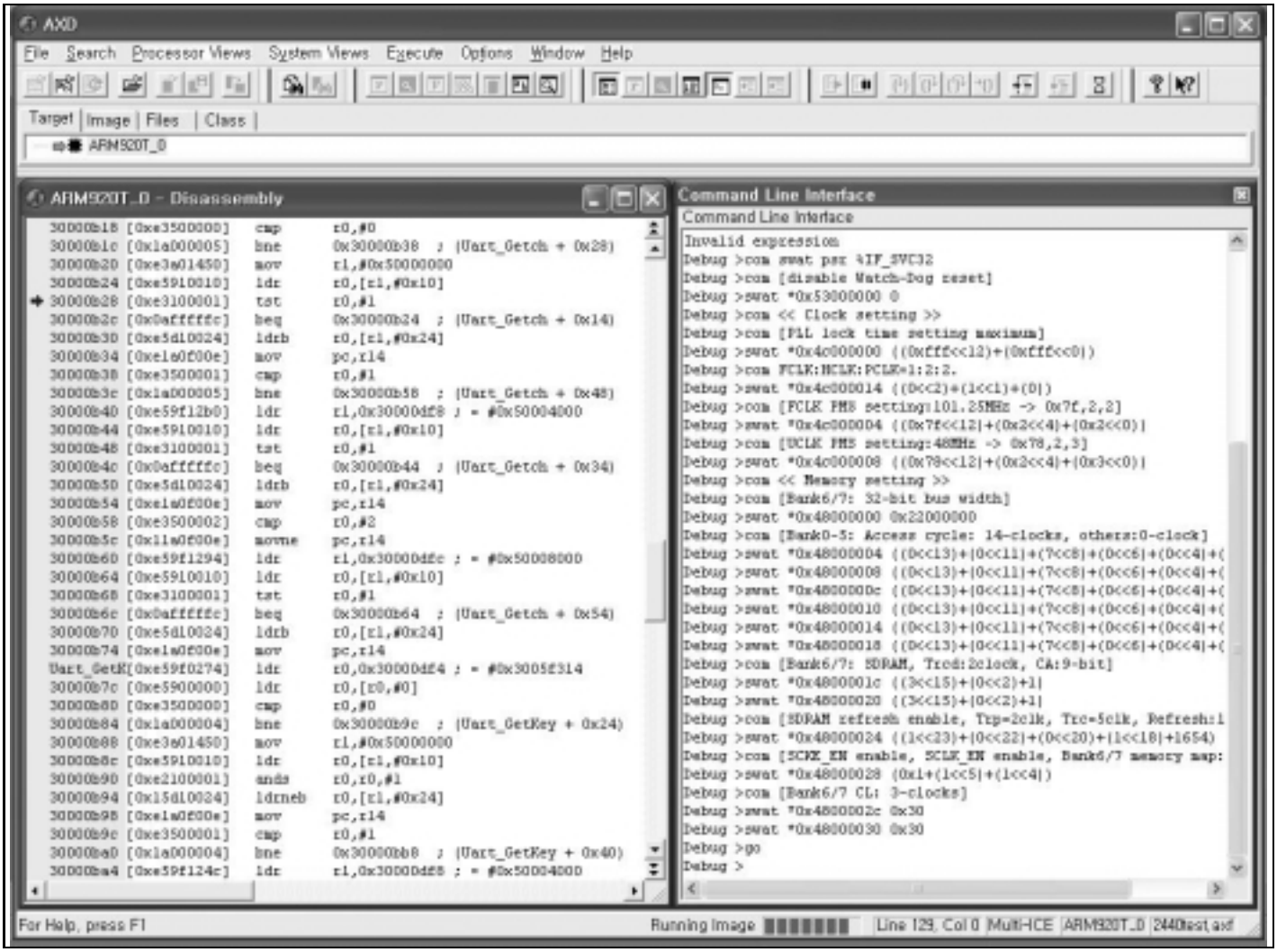
3. Select Load Memory From File… on the file menu of AXD.



— Get the target file to 0x31000000 in SMDK2440 Board.

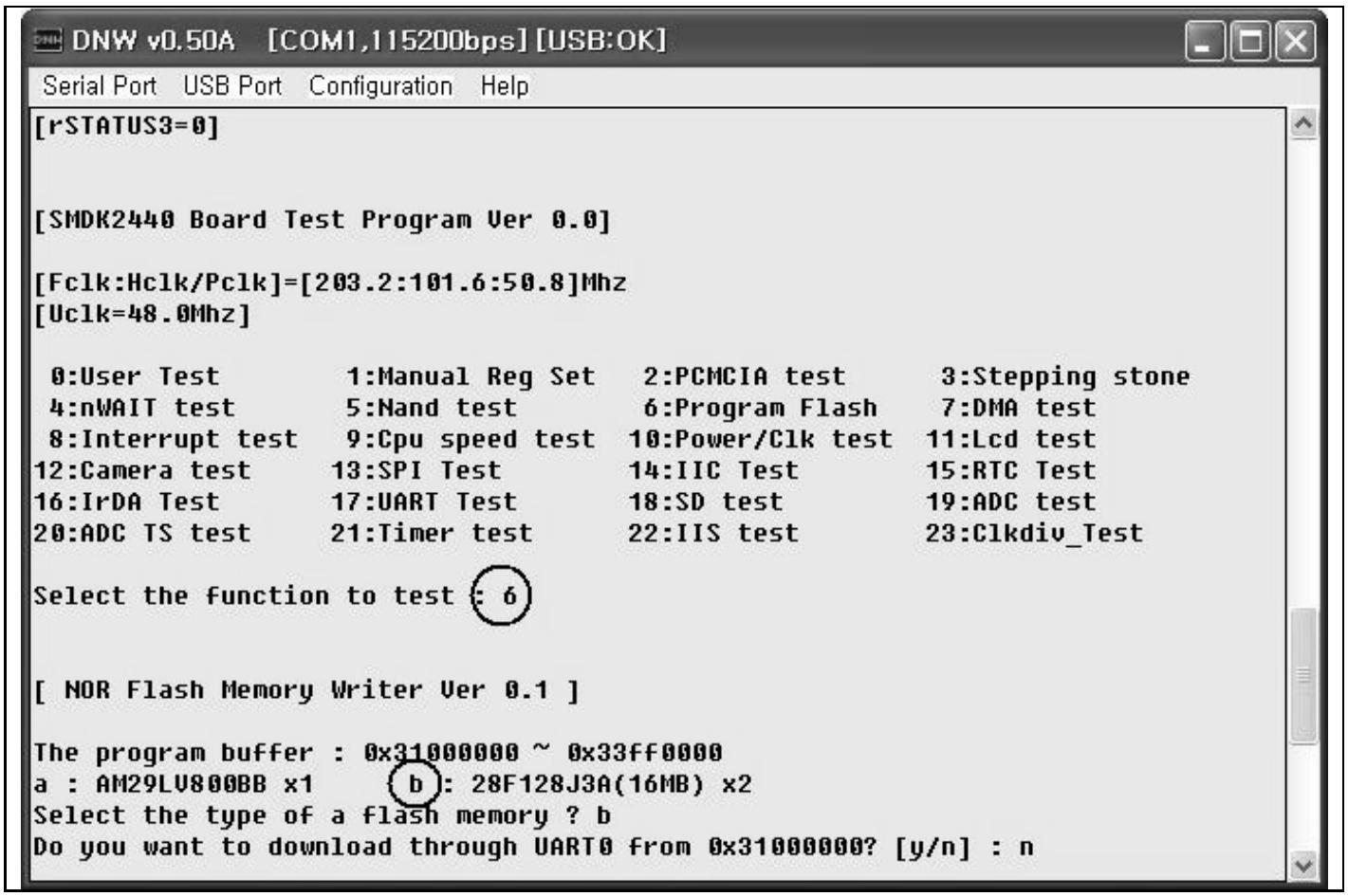


4. Execute 2440TEST.axf file with GO command.



5. Select " 6:Program Flash " on the DNW.

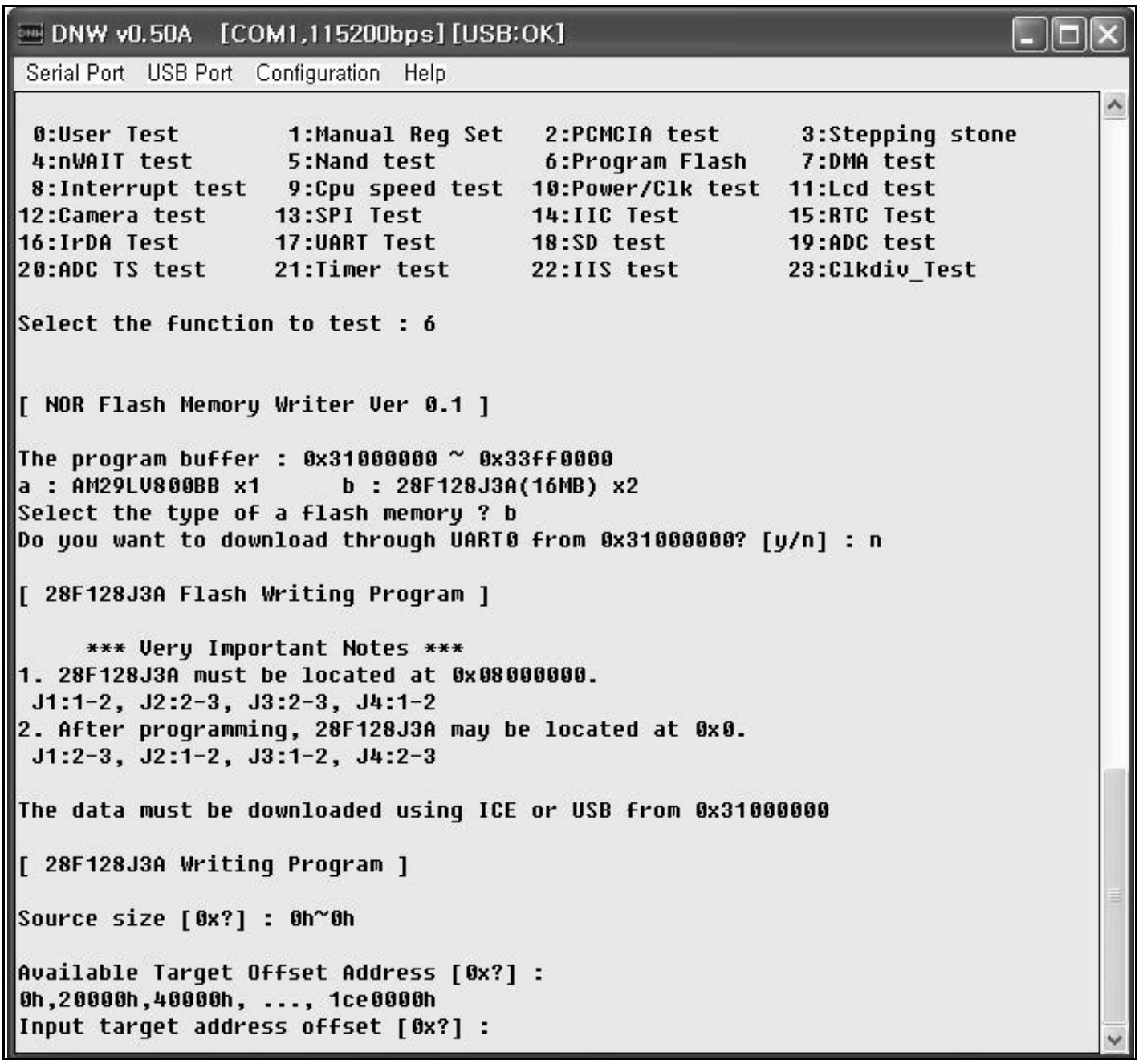
NOTE: If you want to download 2440TEST.bin without MULTI-ICE, then skip the 1, 2 & 3 steps above and download 2440TEST.bin using the DNW (See EXECUTE 2440TEST WITHOUT MULTI-ICE).
After downloading 2440TEST.bin with the DNW, then you can also see the figure below.



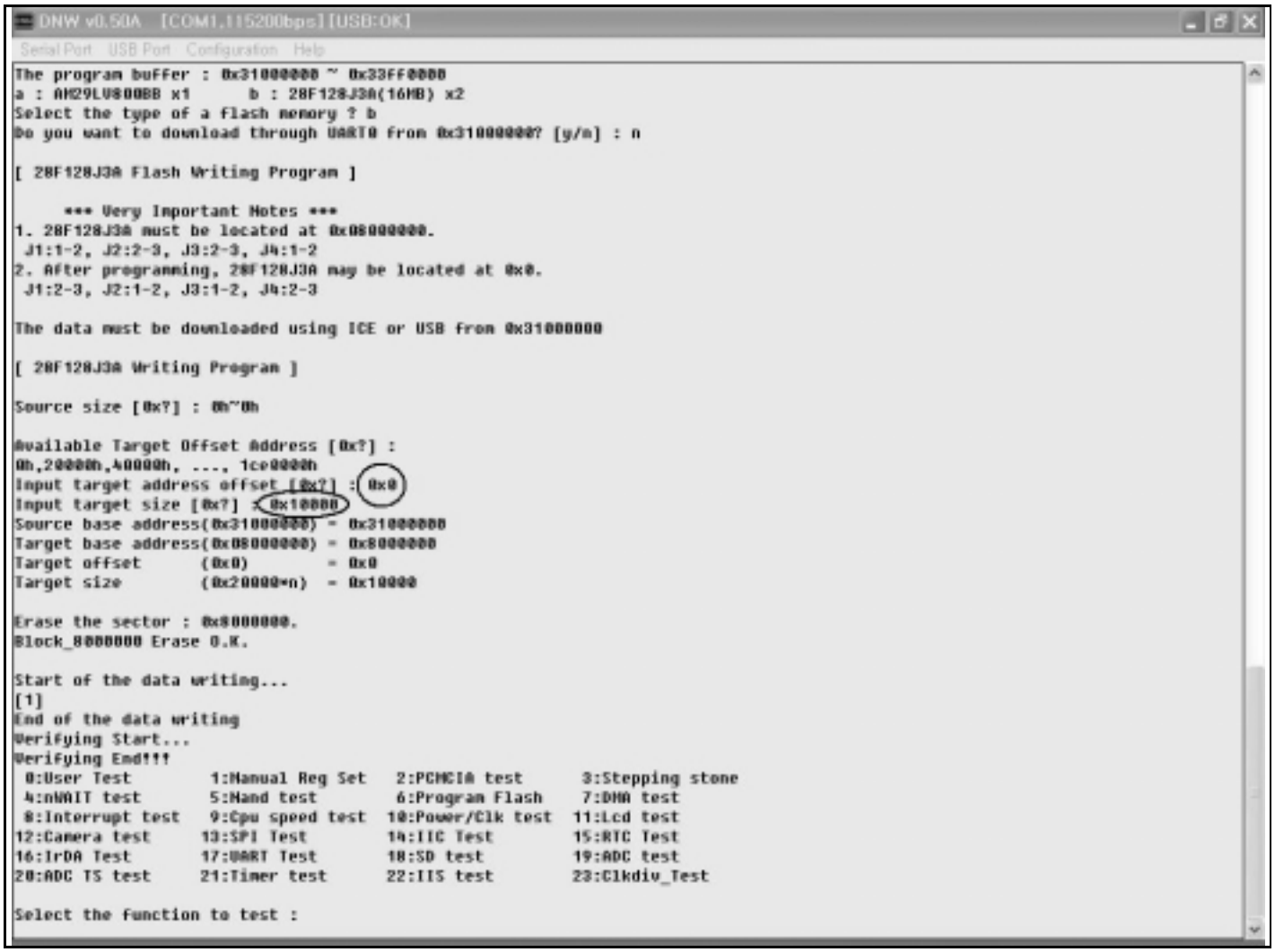
6. Select the type of memory as 28F128J3A (INTEL STRATA flash) by typing ‘b’.

7. Select whether you download through UART0 or MULTI-ICE.

— Type ‘n’ then you can see the figure below in the DNW.



8. Write input target address offset and size of the target file in hexadecimal.



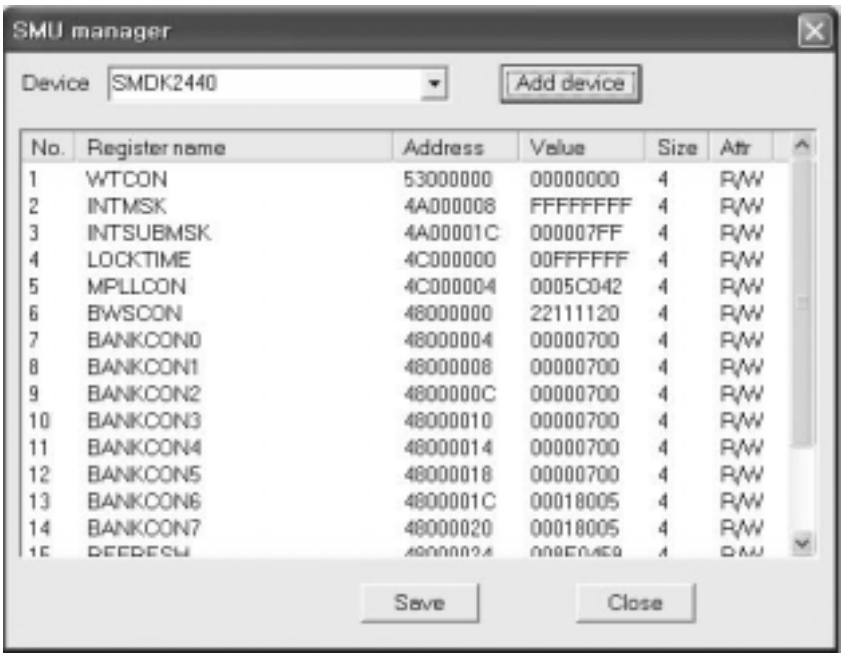
9. Turn the SMDK2440 off and again on.

WRITING IMAGE FILES TO INTEL STRATA FLASH MEMORY WITH OPENICE32-A900

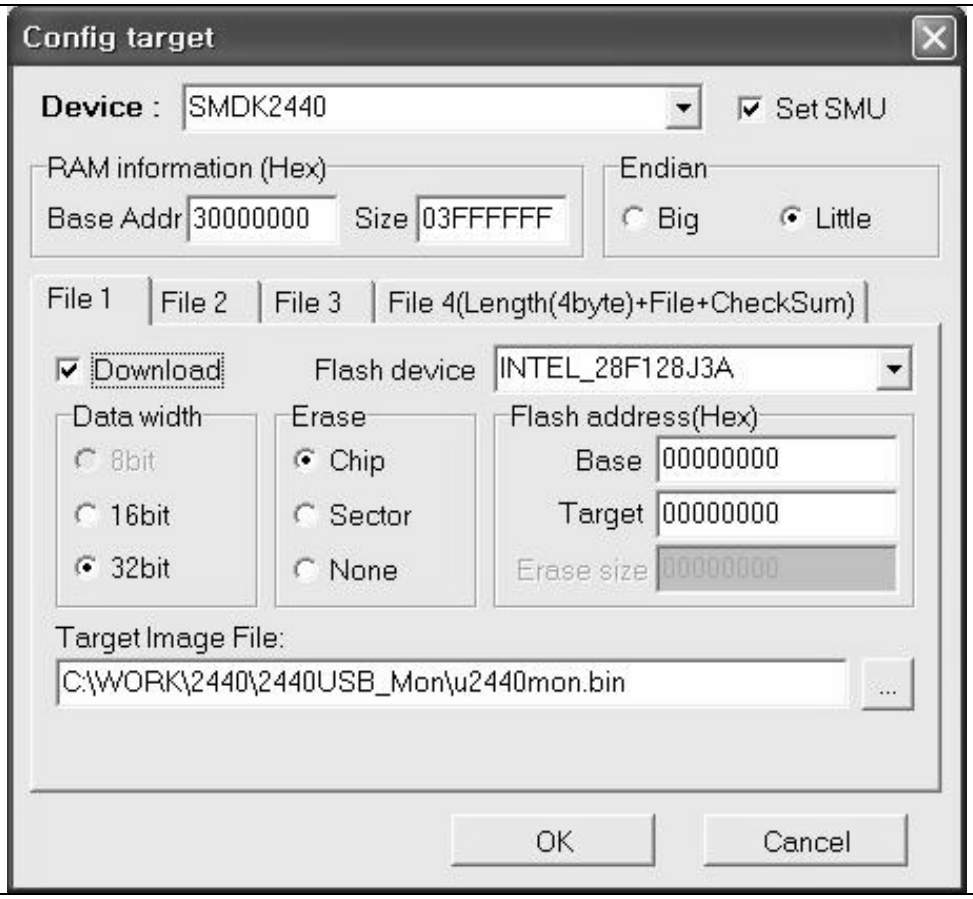
OPENice32-A900 can write image to Intel Strata Flash memory as Multi-ICE. However, OPENice32-A900 provide a Flash Write Program that is easy to use and don't require ARM SDT/ADS debugger nor DNW. For more information on the program, refer to OPENice32-A900 manual or contact AIJI System (www.aijisystem.com).

1. Connect OPENice32-A900 to PC through USB and to SMDK2440 board with 20pin Cable.
2. Set the Jumper J1, J2, J3 and J4 as followings and switch on the board
   J1: 2-3 (short) J2: 1-2 (short) J3: 1-2 (short) J4: 2-3 (short)
3. Run the Flash Write program and select Connect MDS from the File menu.

4. Select SMU Manger from the utility menu and choose a device file, SMDK2440. It is used to initialize the system registers in case of there is no boot ROM. If you can't find the file, download the device file SMDK2410 instead of SMDK2440. After that, edit each value if necessary.



5. Select Config.. from the Flash menu and Set the write options as followings



— Device: SMDK2440

— Set SMU: Checked

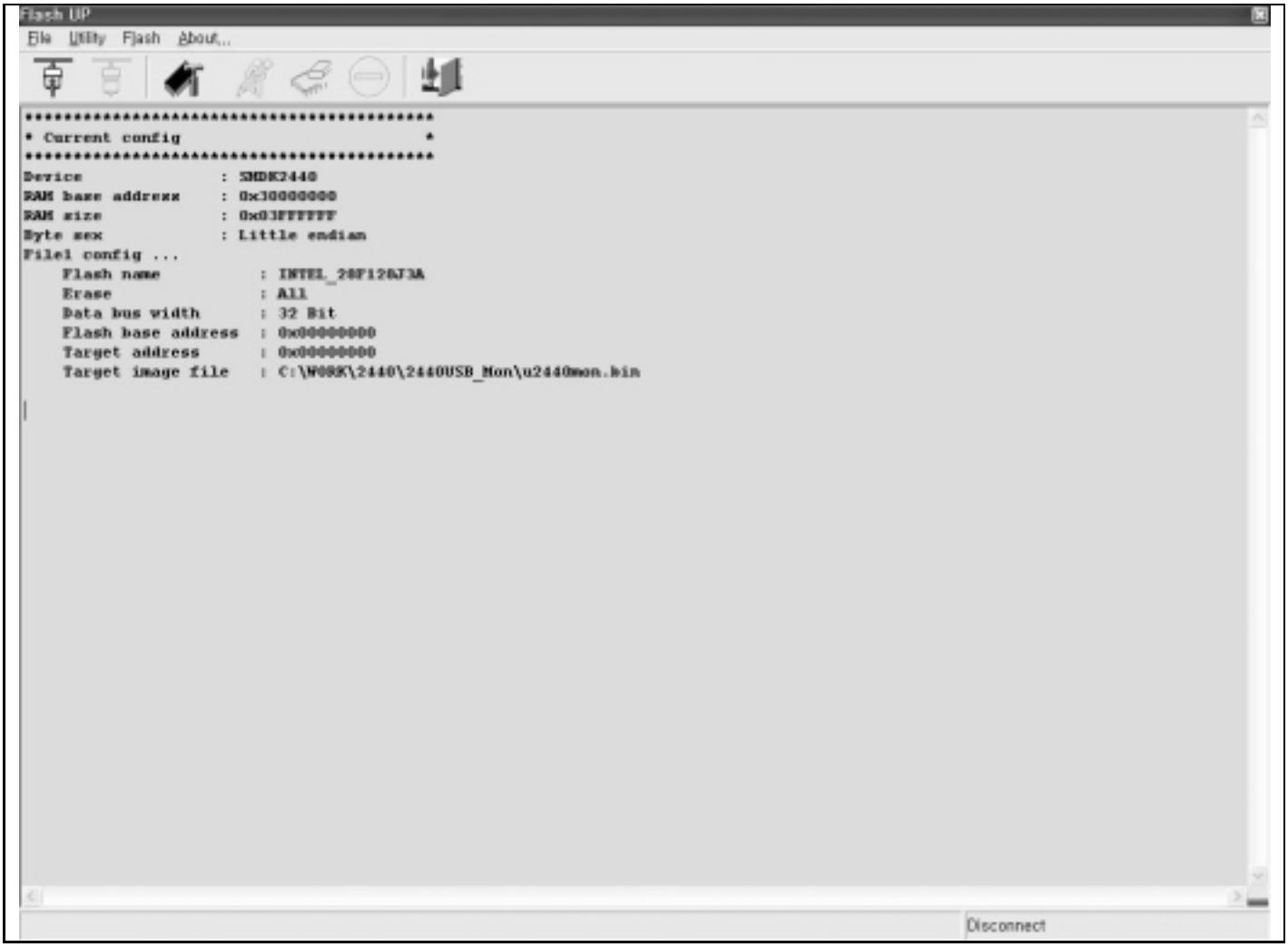
— RAM Information: Base Address:30000000 Size: 3FFFFFF

— Endian: Little

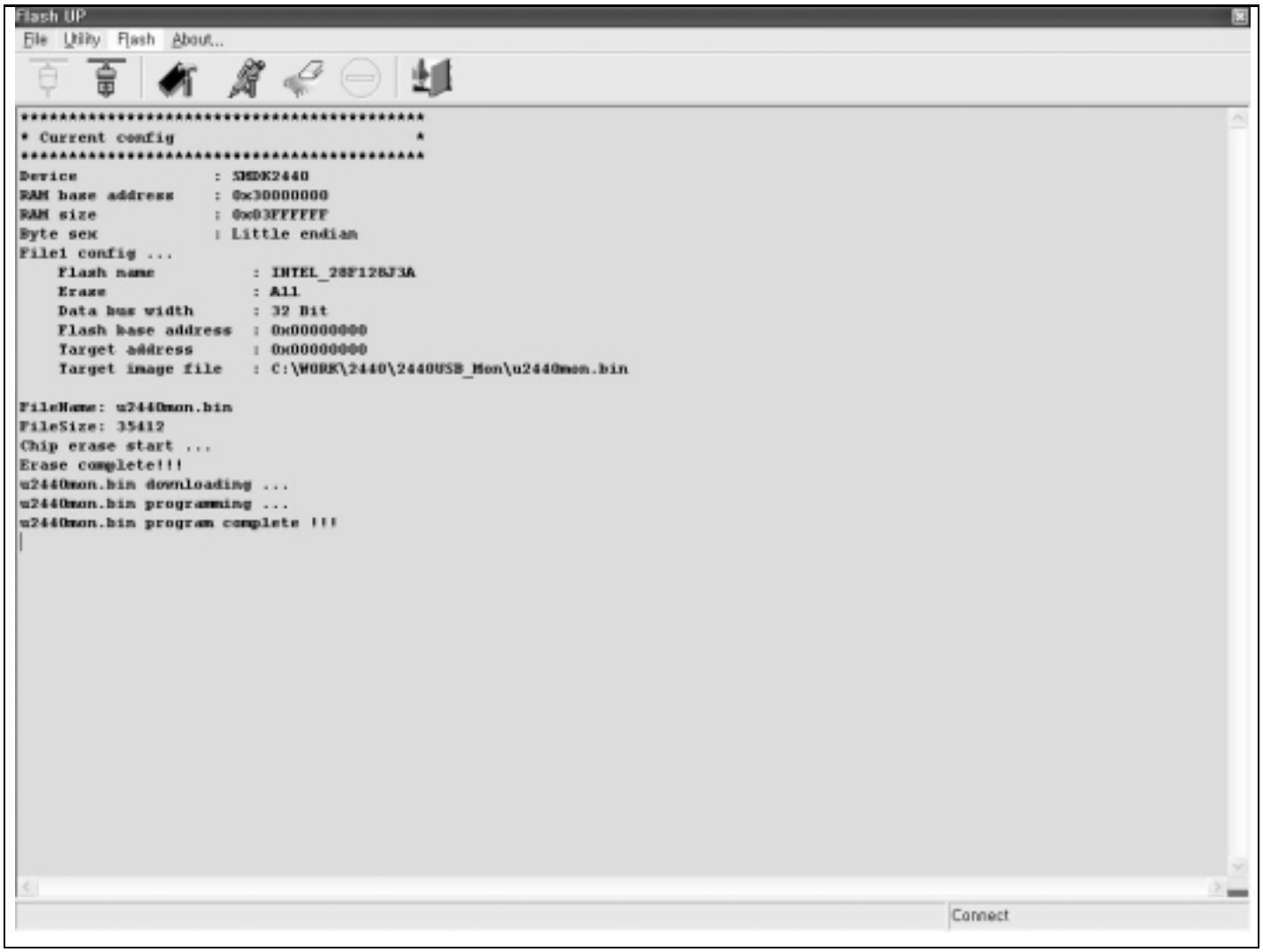
— File 1 page
Download: checked
Flash Device Name: INTEL_28F128J3A
Erase:Chip
Data Bus width: 32bit
Flash Address: Base Address: 0 Target Address:0

— Target Image File: u2440mon.bin

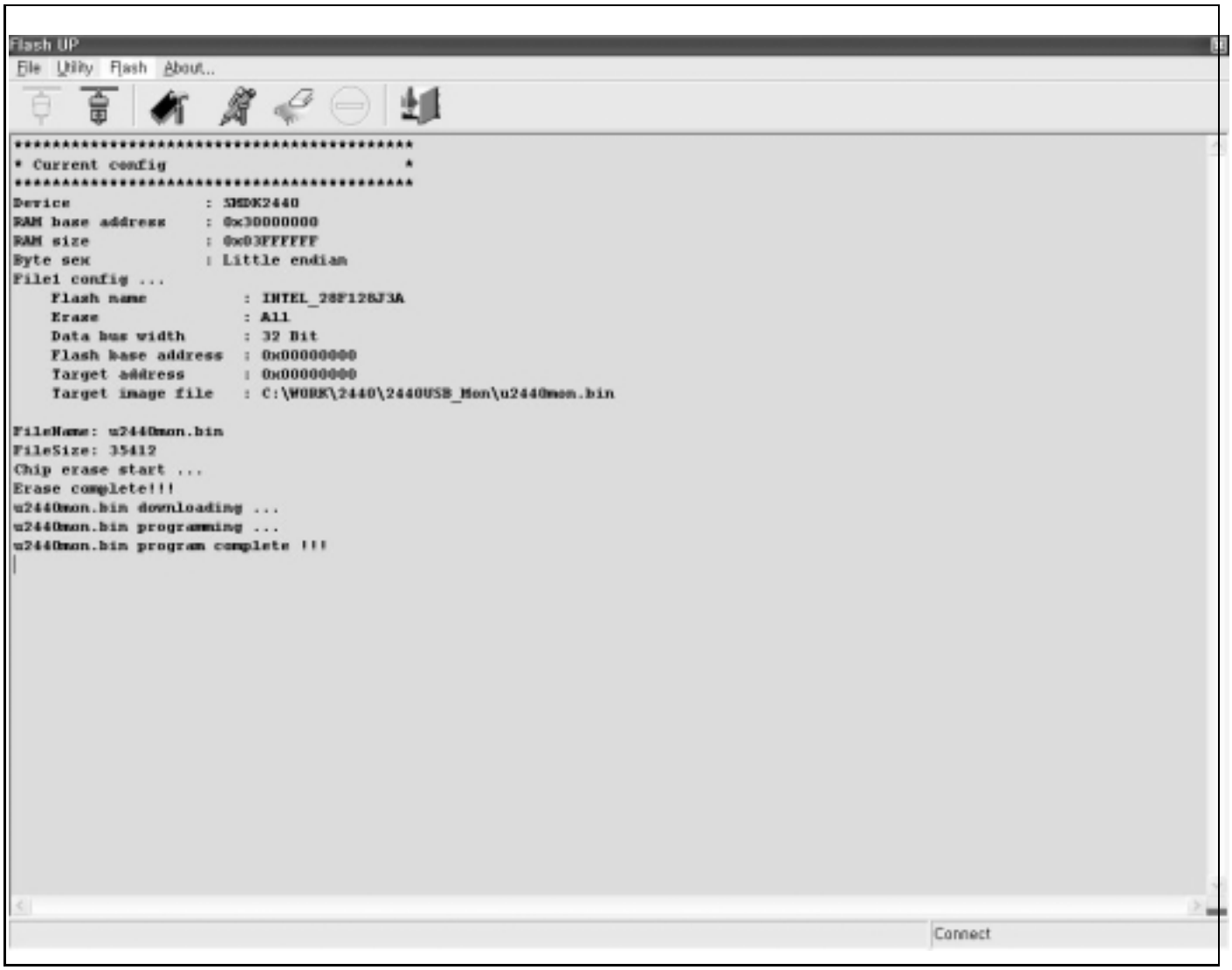
6. Click OK. Then the current configuration is displayed in the window.



7. Select Write from the Flash Menu. Then it starts to erase the specified area of Intel Strata Flash and write the image to the Flash memory. It takes about 10 second.



7. Select Write from the Flash Menu. Then it starts to erase the specified area of Intel Strata Flash and write the image to the Flash memory. It takes about 10 second.



4 SYSTEM DESIGN

OVERVIEW

The S3C2440X, SAMSUMG's 16/32-bit RISC microcontroller is cost-effective and high performance microcontroller solution for hand-held devices and general applications. The S3C2440X has the following integrated on-chip functions:

- 1.2V int., 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 DMAs with external request pins
- 3-ch UART (with 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 general purpose I/O ports / 58 interrupt sources
- Power control: Normal, Slow, Idle and Power-off mode
- 8-ch 10-bit ADC and Touch screen interface
- RTC with calendar function
- On-chip clock generator with PLL

APPLICABLE SYSTEM WITH S3C2440X

The S3C2440X, SAMSUMG's 16/32-bit RISC microcontroller offers various functions and high efficiencies. In addition to the high performance, the S3C2440X offers low current consumption, ensuring low costs.
The followings are sample applications that can be designed with the S3C2440X:

- GPS
- Personal Data Assistance (PDA)
- Fish Finder
- Portable Game Machine
- Fingerprint Identification System
- Car Navigation System
- Smart Phone
- Mobile Information Terminal (MIT)
- Web Screen Phone
- Web Pad

MEMORY INTERFACE DESIGN

BOOT ROM DESIGN

After the system reset, the S3C2440X accesses 0x00000000 address, configuring some system variables. Therefore, this special code (boot ROM image) should be located on the address 0x00000000. Bus width of boot ROM can be selected by setting OM[1:0] pins.

Table 4-1. Data Bus Width for ROM Bank 0

OM[1:0]	Data Bus Width
00	NAND boot
01	16-bit (half-word)
10	32-bit (word)
11	Test mode

NAND BOOT DESIGN

Figure 4-1 shows a design with NAND boot.

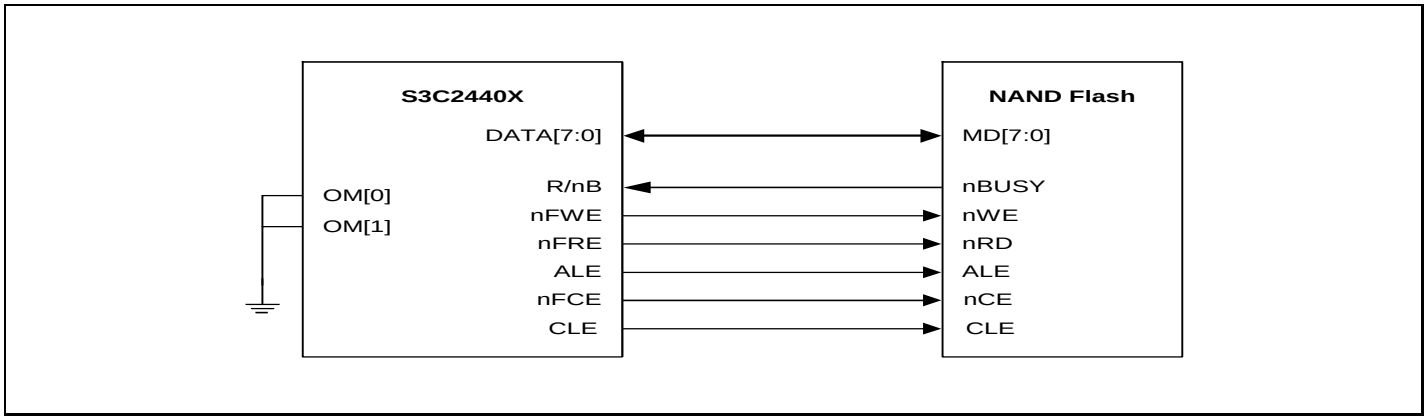


Figure 4-1. NAND Boot Design

MAKING NAND BOOT IMAGE

When making a NAND boot loader image, you can use the binary file that is made from compiling and linking.

HALFWORD BOOT ROM DESIGN WITH BYTE EEPROM/FLASH

Figure 4-2 shows a design with half-word boot ROM with byte EEPROM/Flash.

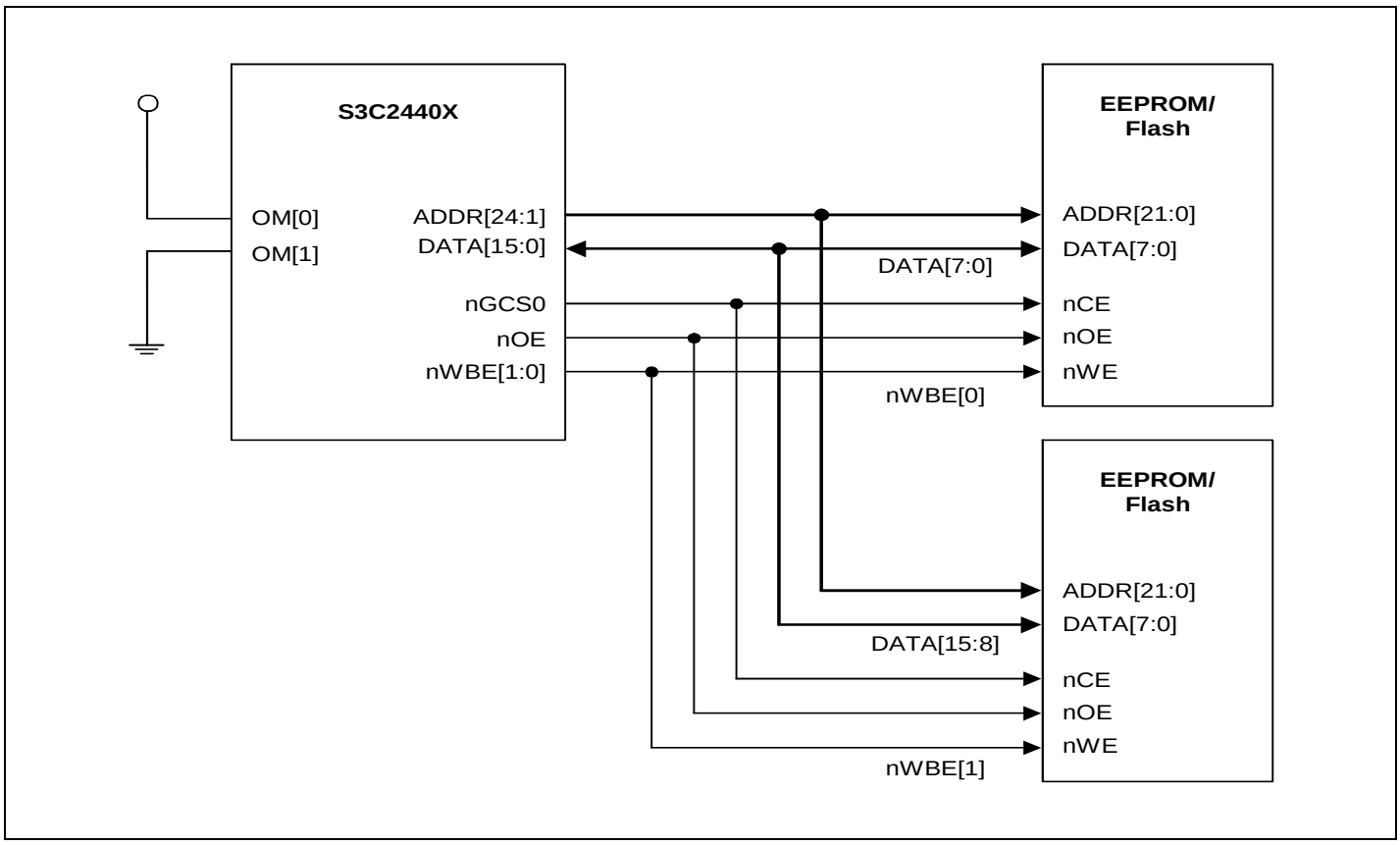


Figure 4-2. Half-word Boot ROM Design with Byte EEPROM/Flash

MAKING HALFWORD ROM IMAGE WITH BYTE EEPROM/FLASH

When make half-word ROM image, you can split two image files, EVEN and ODD.

Table 4-2. Relationship ROM Image and Endian

	Big Endian	Little Endian
DATA[7:0]	Odd	Even
DATA[15:8]	Even	Odd

HALFWORD BOOT ROM DESIGN WITH HALFWORD EEPROM/FLASH

Figure 4-3 shows a design with half-word boot ROM with byte EEPROM/Flash.

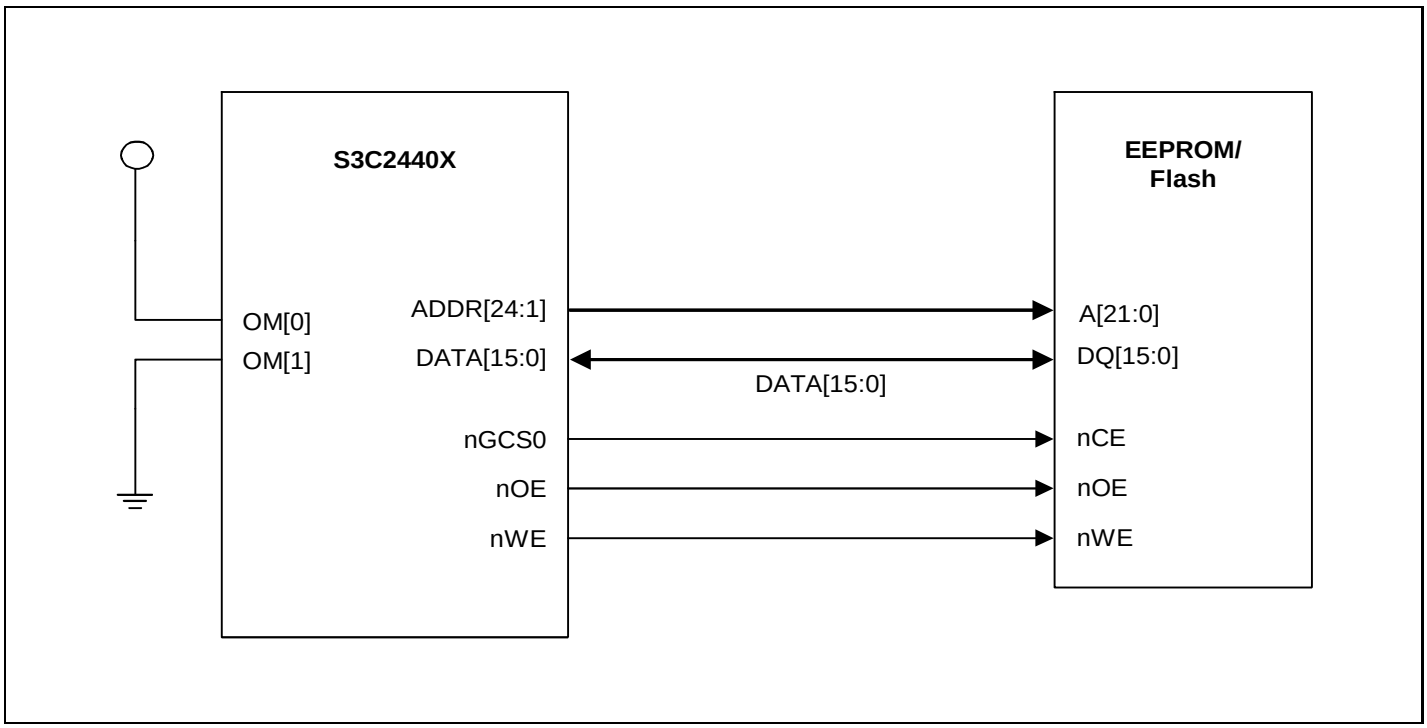


Figure 4-3. The Halfword Boot ROM Design with Halfword EEPROM/Flash

WORD BOOT ROM DESIGN WITH BYTE EEPROM/FLASH

Figure 4-4 shows a design with word boot ROM with byte EEPROM/Flash.

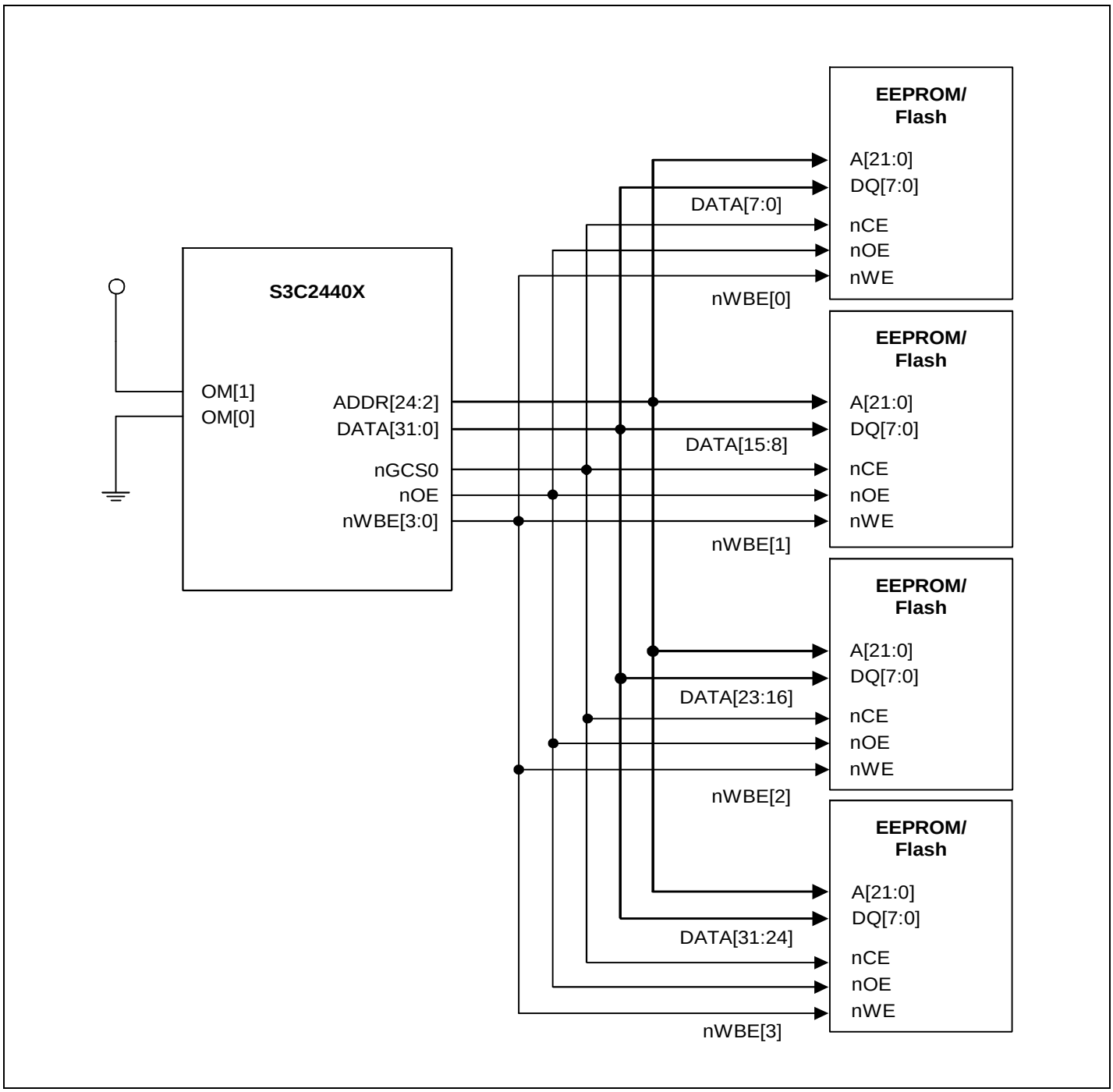
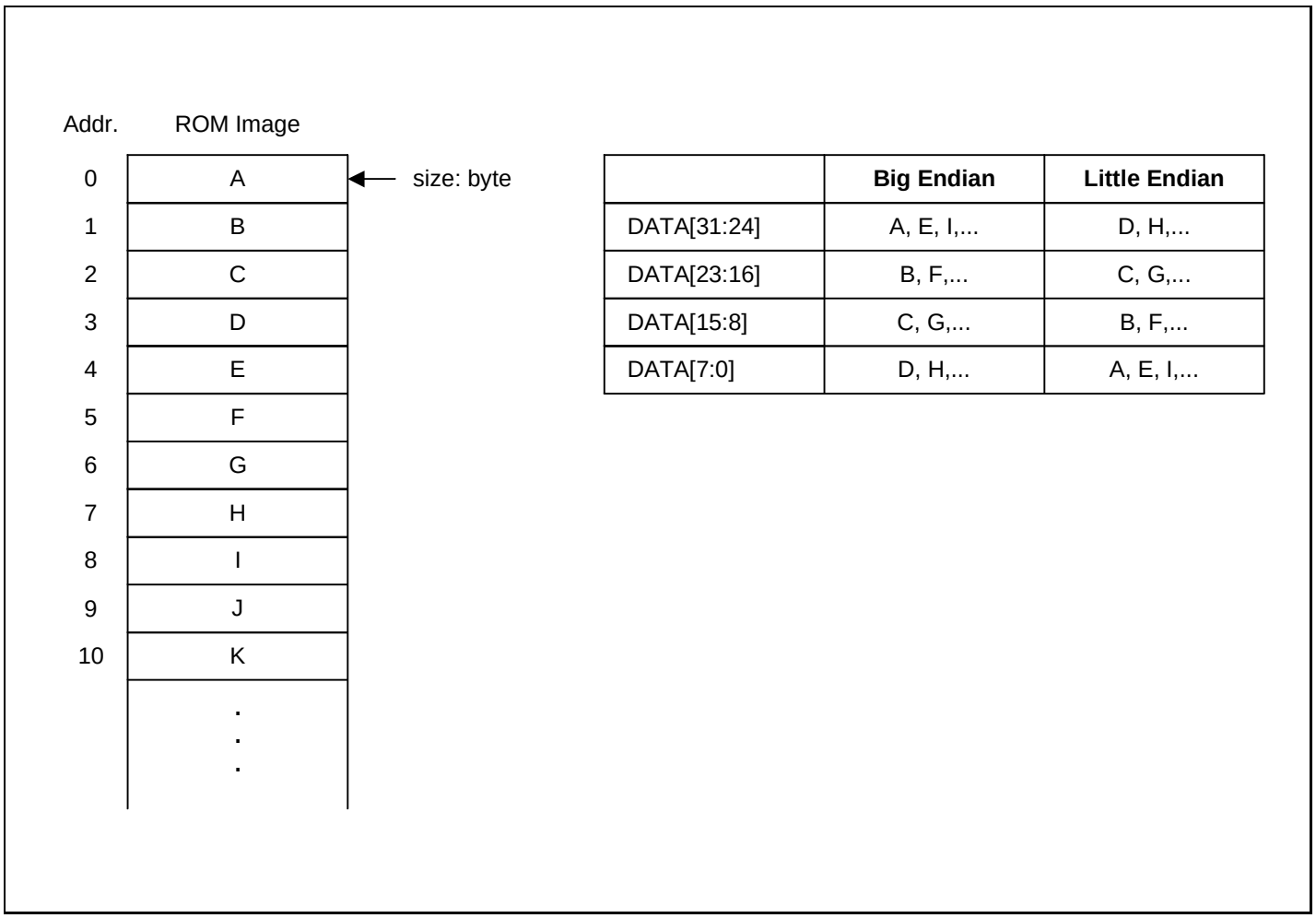


Figure 4-4. The Word Boot ROM Design with Byte EEPROM/Flash

MAKING WORD ROM IMAGE WITH BYTE EEPROM/FLASH

When you make a word ROM image, you can split it into four image files.



	Big Endian	Little Endian
DATA[31:24]	A, E, I,...	D, H,...
DATA[23:16]	B, F,...	C, G,...
DATA[15:8]	C, G,...	B, F,...
DATA[7:0]	D, H,...	A, E, I,...

Figure 4-5. Relationship of ROM Image and Endian

MEMORY BANK DESIGN AND CONTROL

The S3C2440X has six ROM/SRAM banks (including BANK0 for boot ROM) and two ROM/SRAM/SDRAM banks. The system manager on the S3C2440X can control access time, data bus width for each bank by S/W. The access time of ROM/SRAM banks and SDRAM banks is controlled by BANKCON0~5 and BANKCON6~7 control register on the system manager. The data bus width for each ROM/SRAM banks is controlled by BWSCON control register.

The ROM bank0 is used for boot ROM bank, therefore data bus width of bank0 is controlled by H/W. OM[1:0] is used for this purpose.

The control of BWSCON, BANKCON0-7, REFRESH, BANKSIZE, and MRSRB6/7 is performed during the system reset. A sample code for special register configuration is described below.

Sample code for special register configuration

```
        ;Set memory control registers
        LDR    r0,=SMRDATA
        LDR    r1,=BWSCON      ;BWSCON Address
        ADD    r2, r0, #52     ;End address of SMRDATA
0
        LDR    r3, [r0], #4
        STR    r3, [r1], #4
        CMP    r2, r0
        BNE    %B0
         .
         .
         .
         .
         .
         .
SMRDATA
         DCD 0x22111120        ;BWSCON
         DCD 0x00000700        ;GCS0
         DCD 0x00000700        ;GCS1
         DCD 0x00000700        ;GCS2
         DCD 0x00000700        ;GCS3
         DCD 0x00000700        ;GCS4
         DCD 0x00000700        ;GCS5
         DCD 0x00018005        ;GCS6 SDRAM(Trcd=3,SCAN=9)
         DCD 0x00018005        ;GCS7 SDRAM(Trcd=3,SCAN=9)
         DCD 0x008e0000+1113   ;Refresh(REFEN=1,TREFMD=0,Trp=2 clk,
                               ;       Trc=7 clk, Tchr=3 clk,Ref CNT)
         DCD 0x32              ;Bank size, 128MB/128MB
         DCD 0x30              ;MRSR 6(CL=3 clk)
         DCD 0x30              ;MRSR 7(CL=3 clk)
```

ROM/SRAM BANK DESIGN

The ROM/SRAM banks 1-7 can have a variety of width of data bus, and the bus width is controlled by S/W. A sample design for ROM/SRAM bank 1-7 is shown in Figure 4-6, Figure 4-7, Figure 4-8 and Figure 4-9.

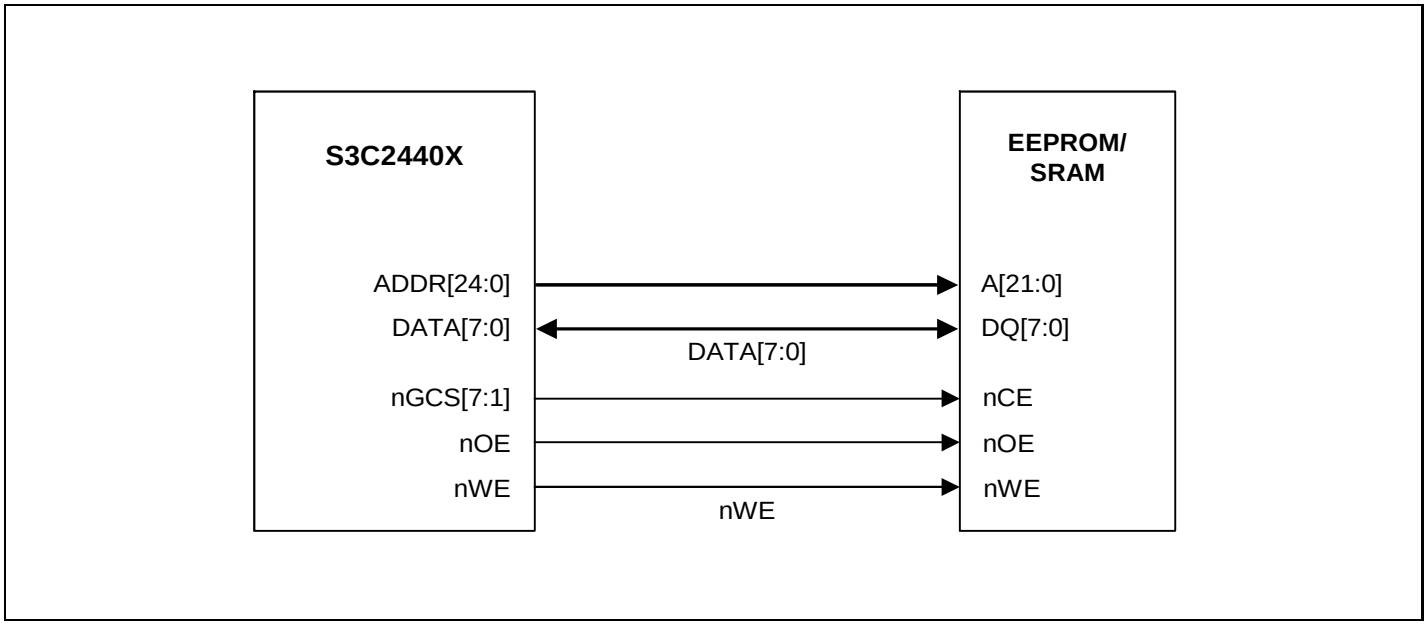


Figure 4-6. One-byte EEPROM/SRAM Bank Design

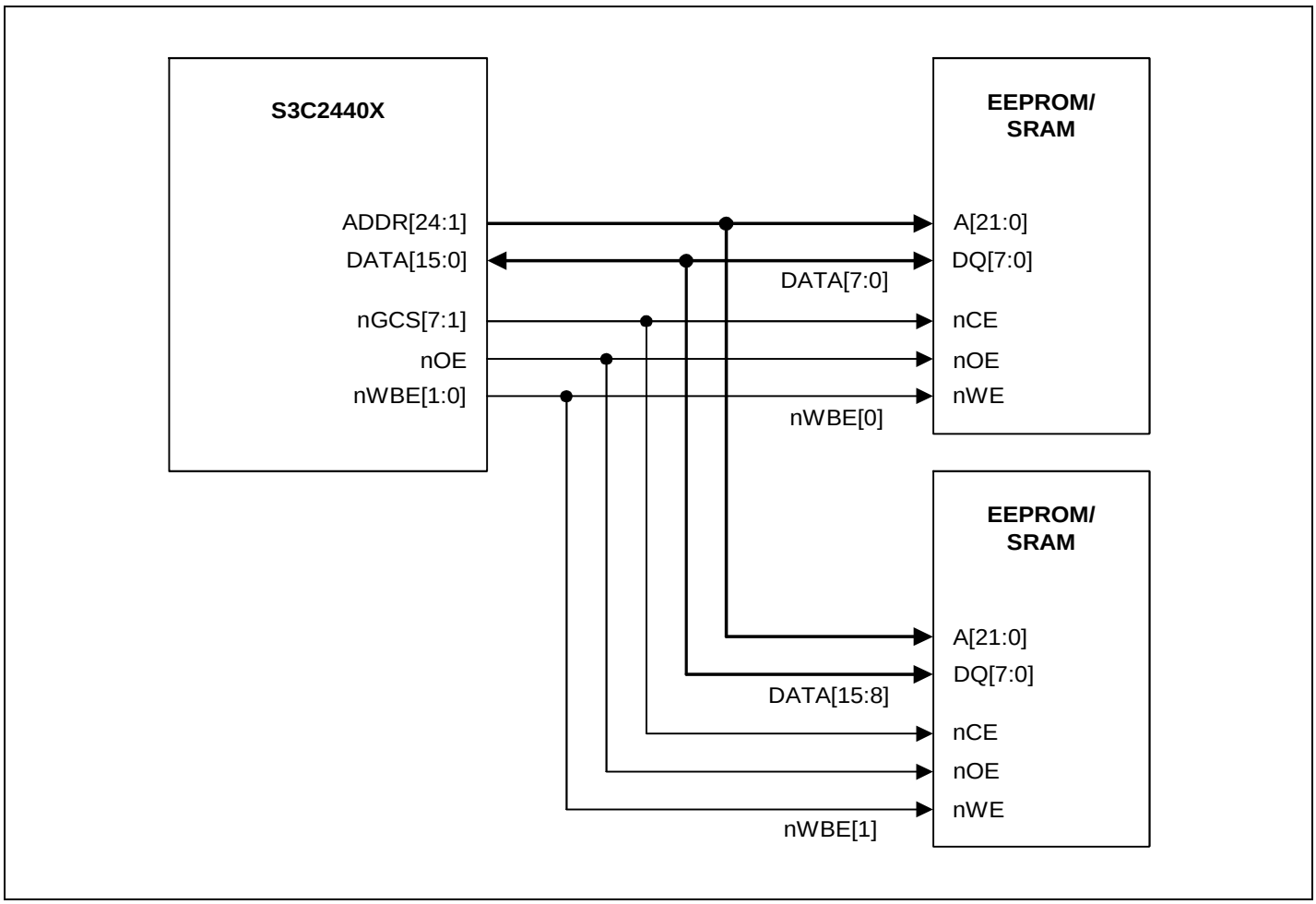


Figure 4-7. Halfword EEPROM/SRAM Bank Design

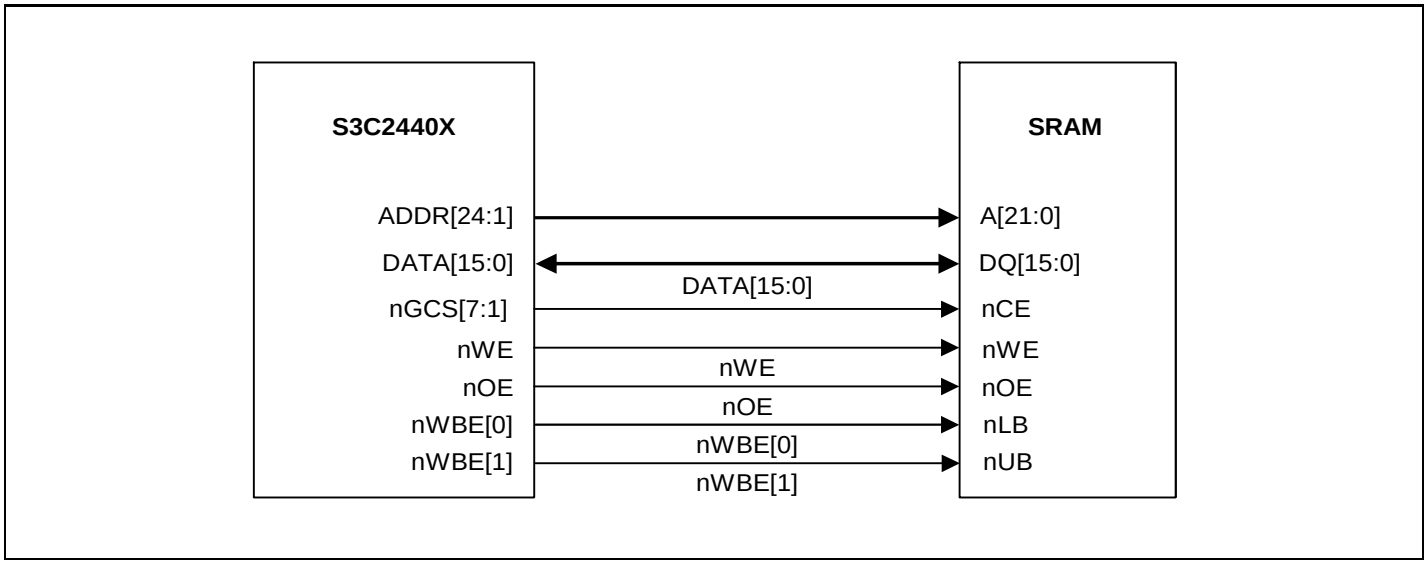


Figure 4-8. Halfword SRAM Bank Design with Halfword SRAM

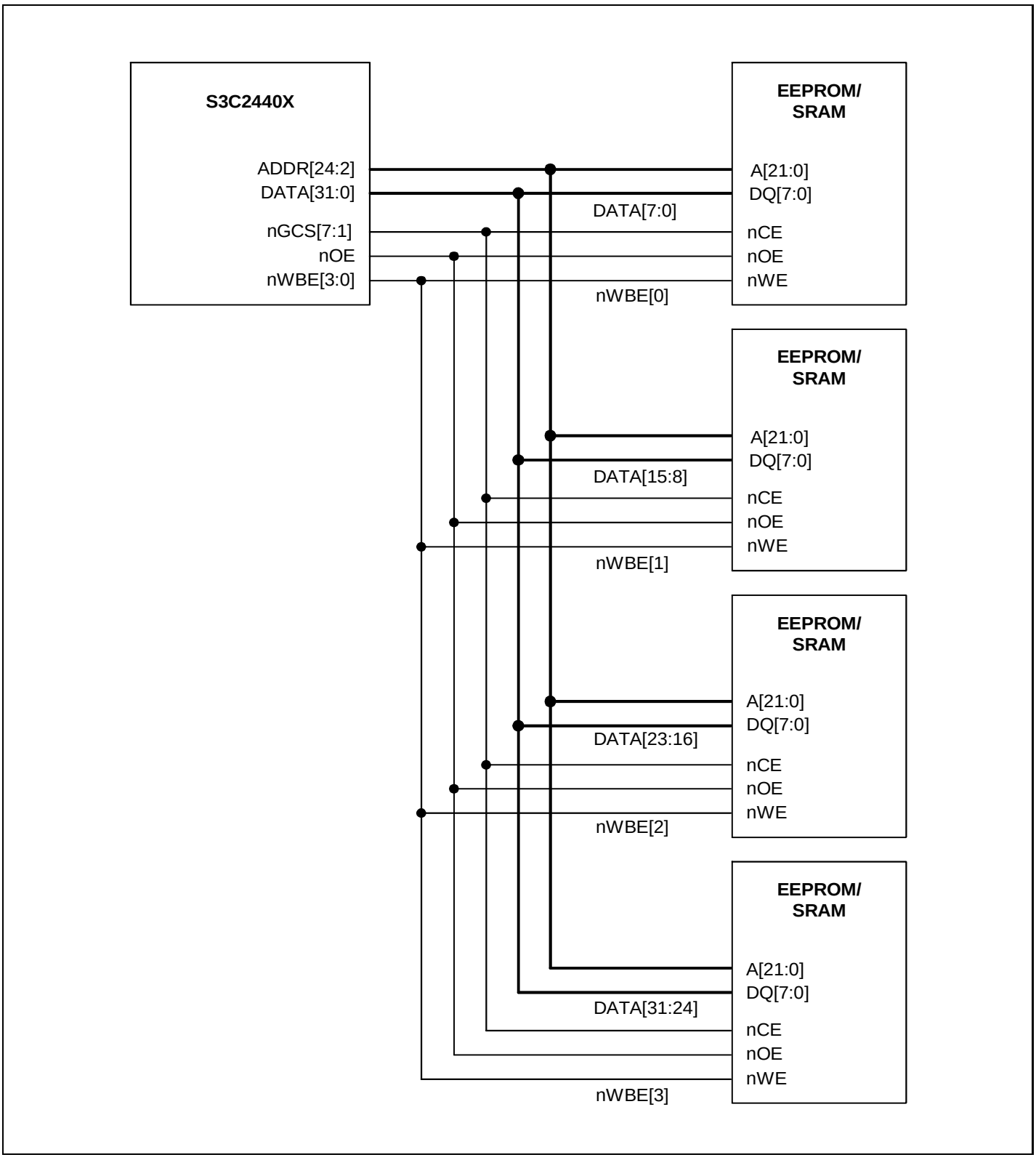


Figure 4-9. Word EEPROM/SRAM Bank Design

SDRAM BANK DESIGN FOR S3C2440X

Table 4-3. SDRAM Bank Address configuration

Bank Size	Bus Width	Base Component	Memory Configuration	Bank Address
2MByte	x8	16Mbit	(1M x 8 x 2Bank) x 1	A20
	x16		(512K x 16 x 2B) x 1	
4MB	x8	16Mb	(2M x 4 x 2B) x 2	A21
	x16		(1M x 8 x 2B) x 2	
	x32		(512K x 16 x 2B) x 2	
8MB	x16	16Mb	(2M x 4 x 2B) x 4	A22
	x32		(1M x 8x 2B) x 4	
	x8	64Mb	(4M x 8 x 2B) x 1	
	x8		(2M x 8 x 4B) x 1	A[22:21]
	x16		(2M x 16 x 2B) x 1	A22
	x16		(1M x 16 x 4B) x 1	A[22:21]
	x32		(512K x 32 x 4B) x 1	
16MB	x32	16Mb	(2M x 4 x 2B) x 8	A23
	x8	64Mb	(8M x 4 x 2B) x 2	
	x8		(4M x 4 x 4B) x 2	A[23:22]
	x16		(4M x 8 x 2B) x 2	A23
	x16		(2M x 8 x 4B) x 2	A[23:22]
	x32		(2M x 16 x 2B) x 2	A23
	x32		(1M x 16 x 4B) x 2	A[23:22]
	x8	128Mb	(4M x 8 x 4B) x 1	
	x16		(2M x 16 x 4B) x 1	
32MB	x16	64Mb	(8M x 4 x 2B) x 4	A24
	x16		(4M x 4 x 4B) x 4	A[24:23]
	x32		(4M x 8 x 2B) x 4	A24
	x32		(2M x 8 x 4B) x 4	A[24:23]
	x16	128Mb	(4M x 8 x 4B) x 2	
	x32		(2M x 16 x 4B) x 2	
	x8	256Mb	(8M x 8 x 4B) x 1	
	x16		(4M x 16 x 4B) x 1	

Table 4-3. SDRAM Bank Address configuration (Continued)

Bank Size	Bus Width	Base Component	Memory Configuration	Bank Address
64MB	x32	128Mb	(4M x 8 x 4B) x 4	A[25:24]
	x16	256Mb	(8M x 8 x 4B) x 2	
	x32		(4M x 16 x 4B) x 2	
	x8	512Mb	(16M x 8 x 4B) x 1	
128MB	x32	256Mbit	(8M x 8 x 4Bank) x 4	A[26:25]
	x8	512Mb	(32M x 4 x 4B) x 2	
	x16		(16M x 8 x 4B) x 2	

The required SDRAM interface pin is CKE, SCLK, nSCS[1:0], nSCAS, nSRAS, DQM[3:0] and ADDR[12]/AP. The sample design with SDRAM is shown in Figure 4-10 and Figure 4-11.

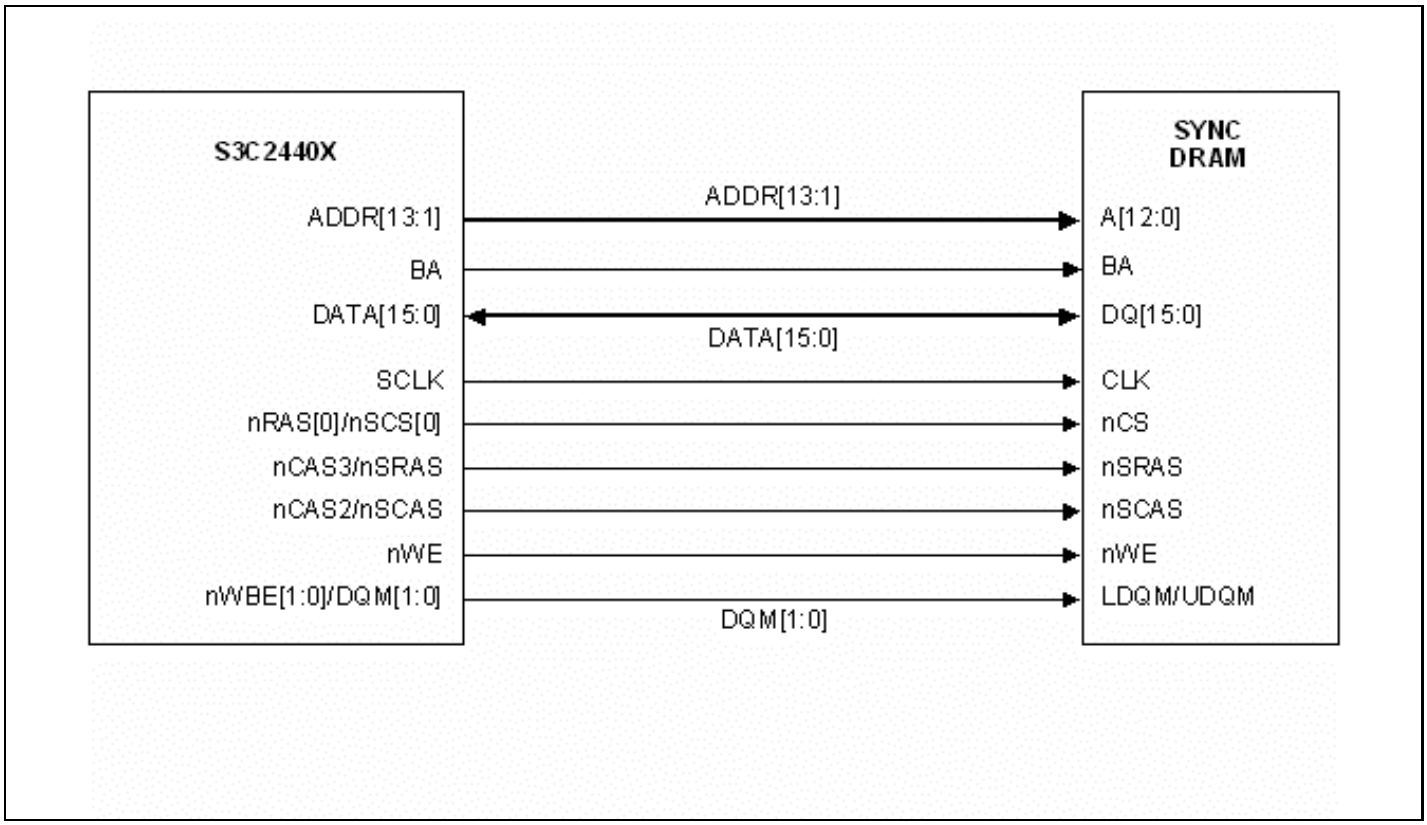


Figure 4-10. Halfword SDRAM Design with Halfword Component

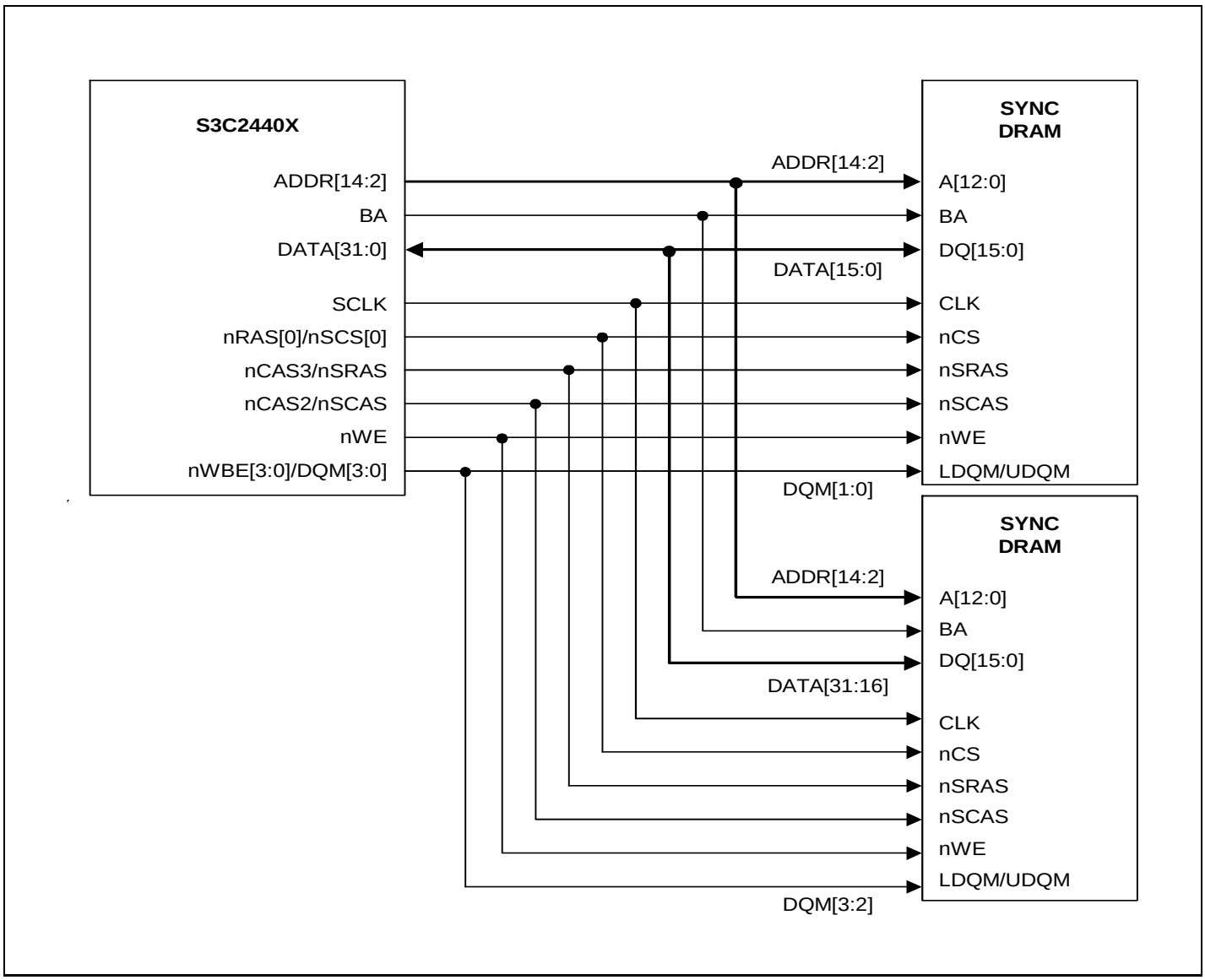


Figure 4-11. Word SDRAM Design with Half-word Component

GPC CARD (GPCMCIA) INTERFACE APPLICATION USING CL-PD6710 (CIRRUS LOGIC)

The GPC card (GPCMCIA card) can be interfaced with S3C2440X using following components:

- CL-PD6710 from Cirrus logic
- TPS2211 from Texas Instruments

We tested the GPC card interface by accessing the card information structure (CIS) in the modem card as Figure 4-12, using following test code.

File Name	File Descriptions
pd6710.h	CL-PD6710 register definitions
pd6710.c	CL-PD6710 GPC Card program

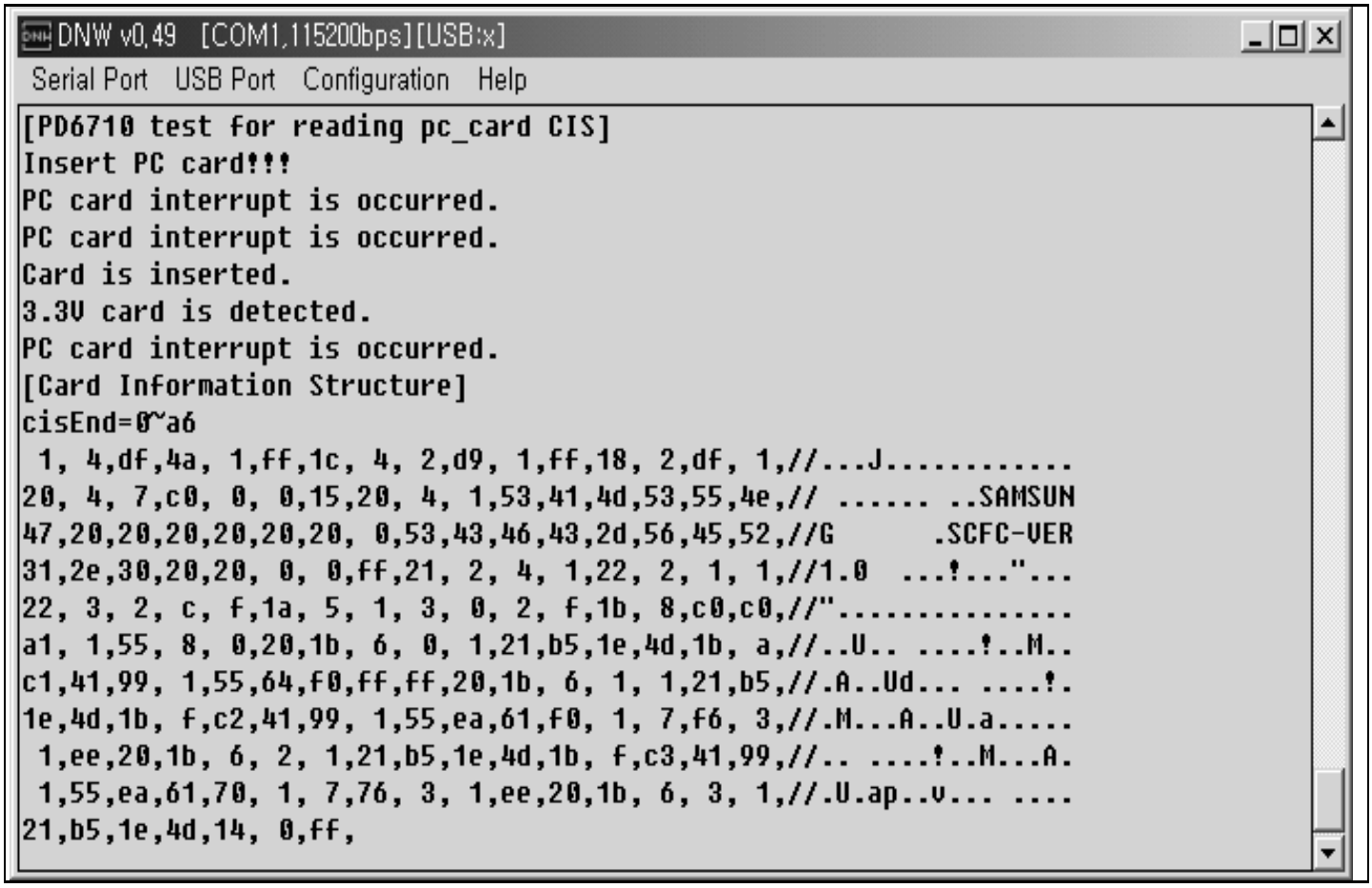


Figure 4-12. GPC Card CIS Access Example on S3C2440X

10BASE-T ETHERNET CONTROLLER (CS8900A) INTERFACE

The 10BASE-T Ethernet can be supported on S3C2440X using following components:

- CS-8900A from Cirrus logic
- XFMRS XF10B11A-COMB1-2S is Ethernet RJ45 with transformer.

AUDIO CODEC (UDA1341TS) CONNECTION WITH S3C2440X

The S3C2440X IIS interface example circuit is as follows:

- UDA1341TS from Philips Semiconductors.
- The L3 interface of Philips (L3MOD, L3CLOCK and L3DATA) is realized by general I/O port.
- Refer to the sample code of audio application which plays GPCM file.

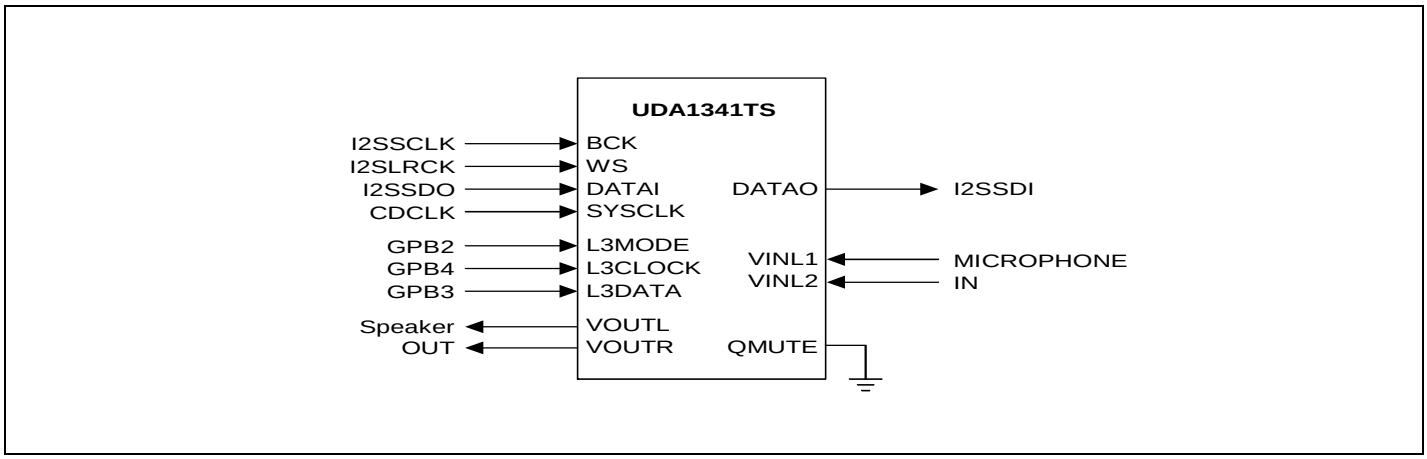


Figure 4-13. UDA1341TS Connection with S3C2440X

LCD CONNECTION WITH S3C2440X

The S3C2440X LCD interface example circuit is as follows:

- UG-32F04 (320x240 mono STN LCD) from SAMSUNG DISPLAY DEVICES CO., LTD. (refer to Figure 4-14)
  - TL497CAN can be used to make VEE (-25V).
- UG-24U03A (320x240 mono STN LCD) from SAMSUNG DISPLAY DEVICES CO., LTD. (refer to Figure 4-15)
  - VEE is generated by the circuit on LCD module.
  - VL is 2.4V typically.
  - DISPON H: display on, L: display off
  - nEL_ON H: EL off L: EL on
- KHS038AA1AA-G24 (256 color STN LCD) from KYOCERA Co. (refer to Figure 4-16)
  - DISP signal can be made using I/O port, or power control circuit or nRESET circuit.
  - V1-V5 can be made using the power circuit recommended by the LCD specification.
- LTS350Q1-PE1 (256K color TFT LCD) from SAMSUNG ELECTRONICS CO., LTD. (refer to Figure 4-17)
  - VDD_LCDI is typically 3.3V.
- LP104V2-W (262,144 color TFT LCD, 10.4") from LG Philips (refer to Figure 4-18)
  - VDD_LCDI is typically 3.3V.
- V16C6448AB (640x480 TFT LCD) from PRIMEVIEW (refer to Figure 4-19)
  - VDD_LCDI, VD and control signal are typically 5.V.

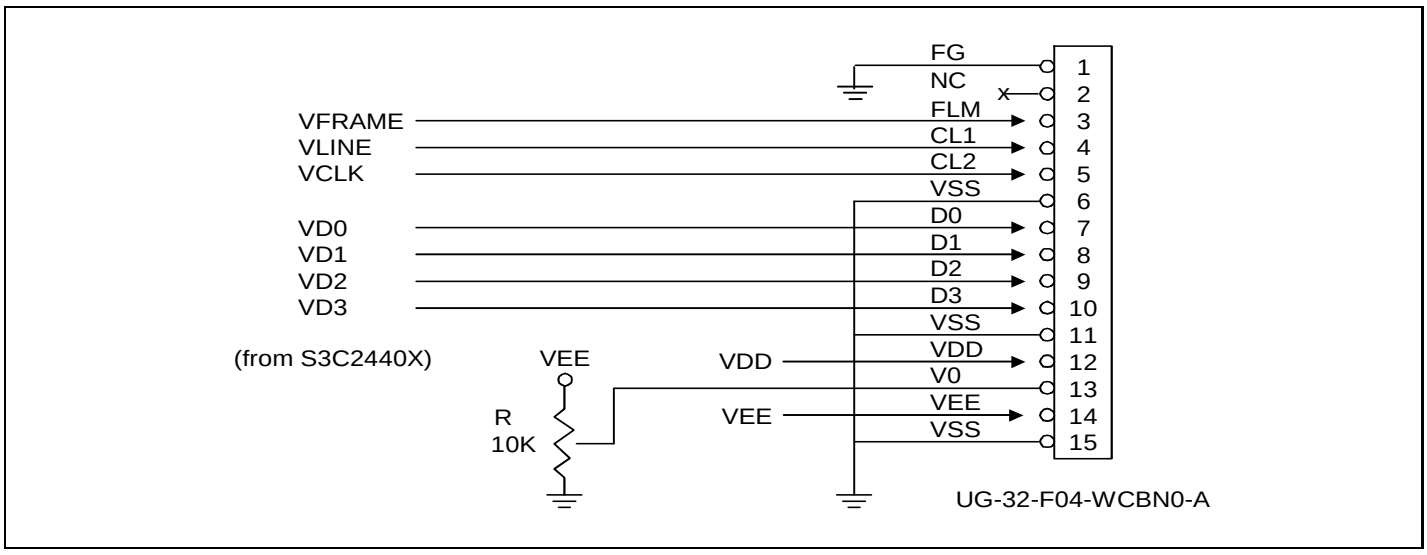


Figure 4-14. UG-32F04 Connection with S3C2440X (320x240 Mono STN LCD)

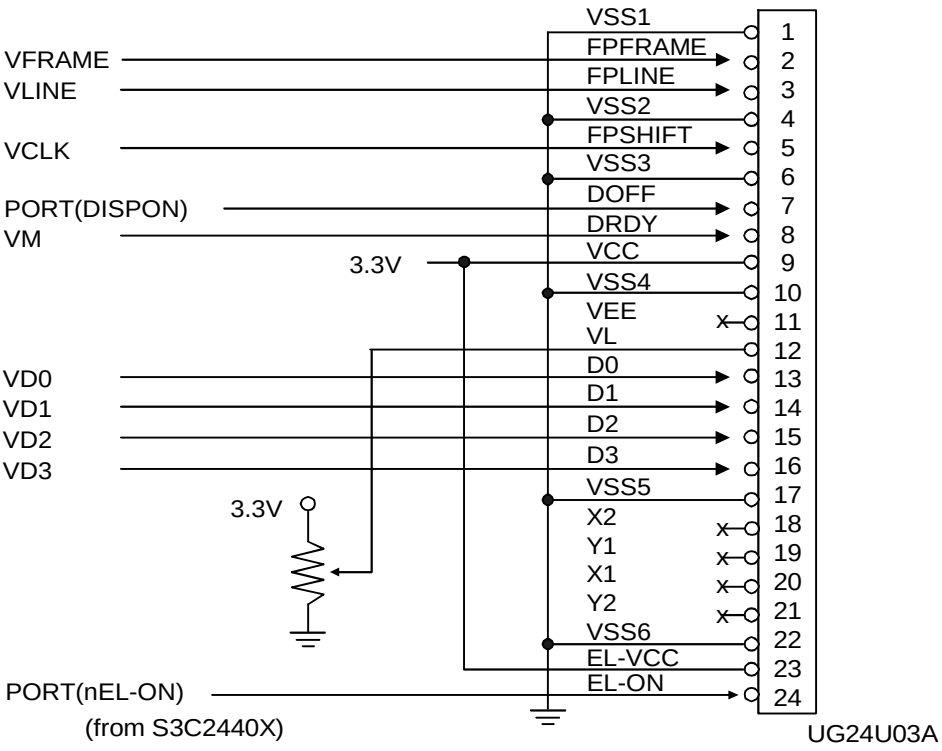


Figure 4-15. UG24U03A Connection with S3C2440X (320x240 Mono STN LCD)

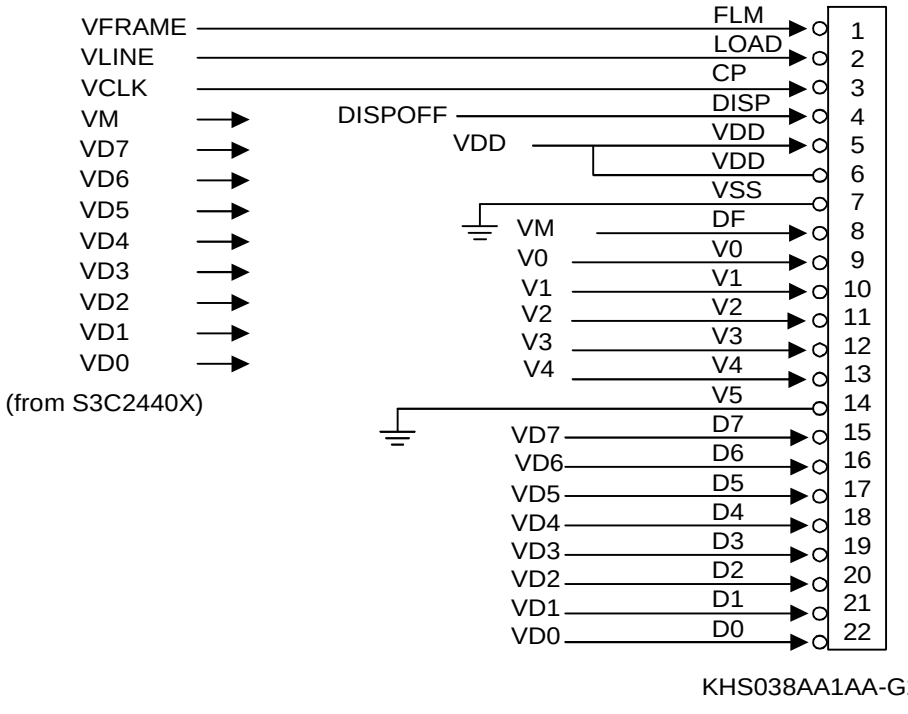


Figure 4-16. KHS038AA1AA-G24 Connection with S3C2440X (256 Color STN LCD)

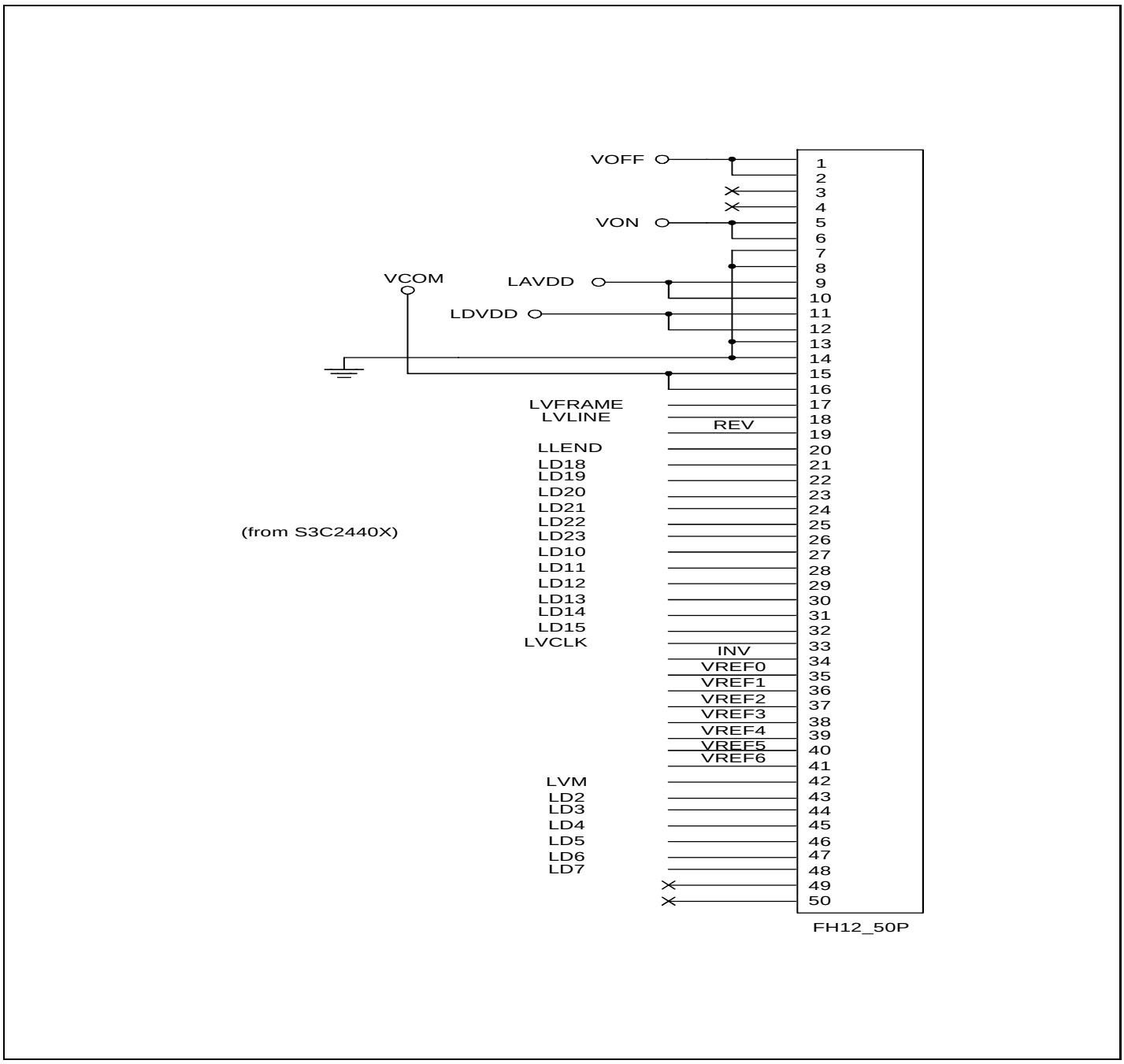


Figure 4-17. LTS350Q1-PE1 Connection with S3C2440X (Samsung 3.5” Transflective TFT LCD)

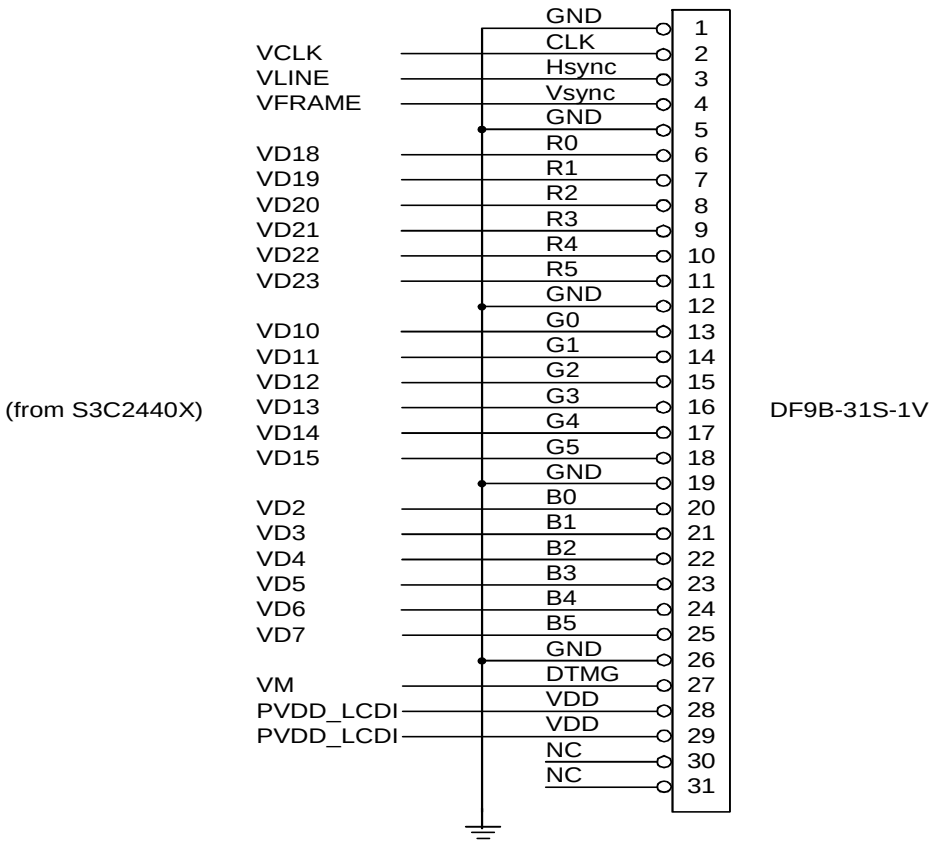


Figure 4-18. LP104V2-W Connection with S3C2440X (LG Philips 10.4” TFT LCD)

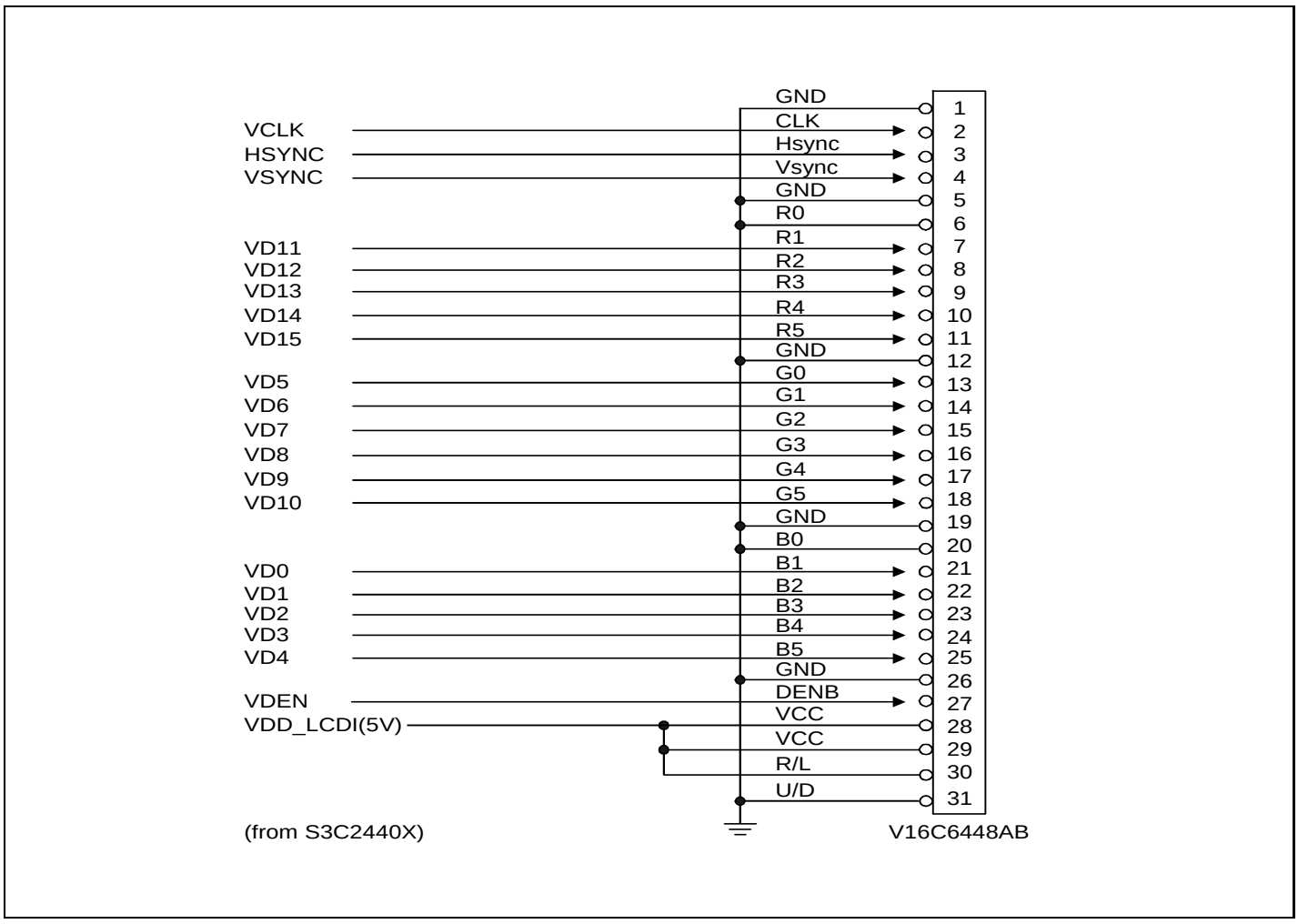


Figure 4-19. V16C6448AB Connection with S3C2440X (TFT LCD)

TOUCH SCREEN PANEL (TSP) INTERFACE CIRCUIT WSSITH S3C2440X

Typically, the TSP consists of two plane resisters (x-axis plane resister and y-axis plane resister). In this reason, TSP has four terminals. To read X coordinate, Q1 and Q2 are turned on while Q3 and Q4 is turned off. Therefore, X coordinate value can be read out from AIN7 by ADC. To read Y coordinates, Q3 and Q4 is turned on while Q1 and Q2 is turned off. So, ADC can read out Y coordinate value from AIN5. The INT_TC can be used to check whether the TSP is touched.

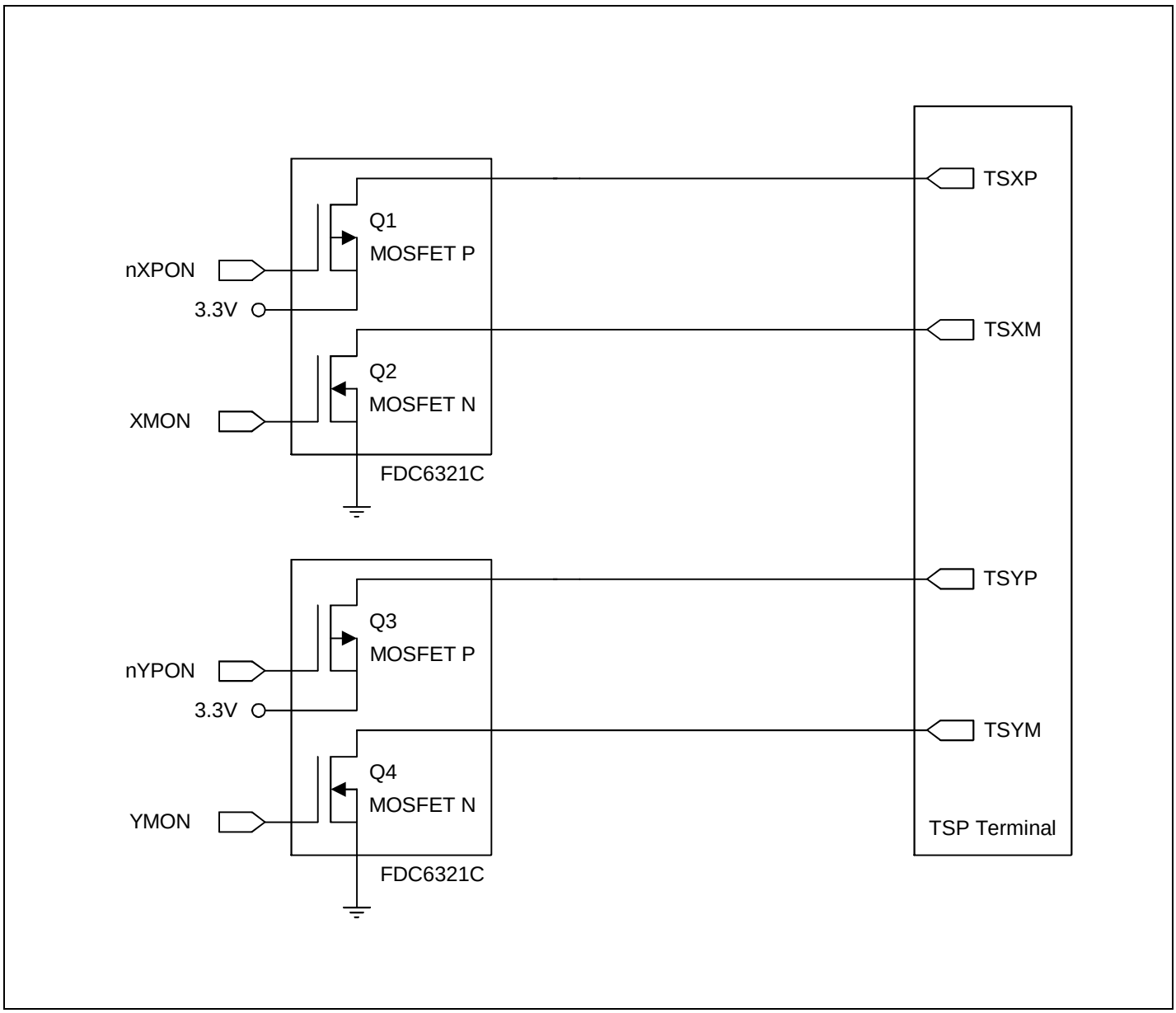


Figure 4-20. TSP Interface Circuit with S3C2440X

SYSTEM DESIGN WITH DEBUGGER SUPPORT

MULTI-ICE

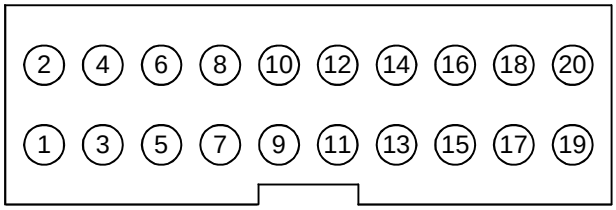
The S3C2440X has an Embedded ICE logic that provides debug solution from ARM. MULTI-ICE enables you to debug software running on the S3C2440X. Embedded ICE logic is accessed through the Test Access Port (TAP) controller on the S3C2440X using the JTAG interface.

JTAG port for Embedded ICE Interface

When you build a system with the S3C2440X Embedded ICE interface, you should design a JTAG port for MULTI-ICE interface. Usually, the interface connector is a 20-way box header, and this plug is connected to the Embedded ICE logic interface module using 20-way IDC socket.

The JTAG port signals, nTRST, TDI, TMS and TCK have to be connected to pulled-up register (10K ohm) externally.

The pin configuration and a sample design are described in Figure 4-21 and Figure 4-22, respectively.



Pin	Name	Function
1	VTref	System Power
2	Vsupply	System Power
3	nTRST	Test reset, active low (connected pull-up reg.)
5	TDI	Test data in (connected pull-up reg.)
7	TMS	Test mode select (connected pull-up reg.)
9	TCK	Test clock (connected pull-up reg.)
11	RTCK	Return test clock (connected pull-down reg.)
13	TDO	Test data out
15	nSRST	Connected to nRESET and nTRST through 470 ohm resister
17	DBGRQ	NC
19	DBGACK	NC
4, 6, 8, 10, 12, 14, 16, 18, 20	GND	System Ground

Figure 4-21. MULTI-ICE Interface of JTAG Connector

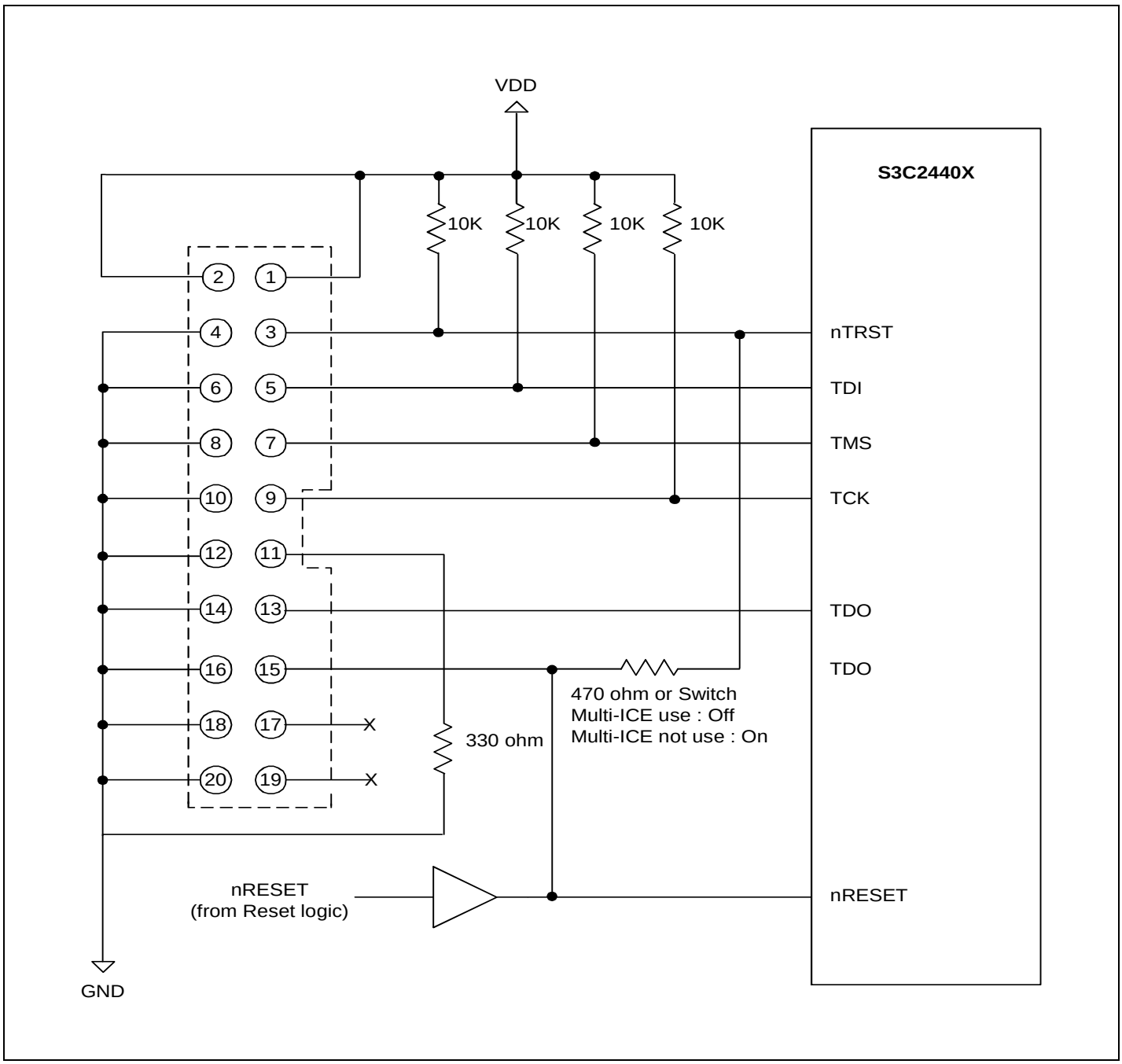


Figure 4-22. MULTI-ICE Interface Design Example

CHECK ITEMS FOR SYSTEM DESIGN WITH S3C2440X

When you design a system with the S3C2440X, you should check a number of items to build a good system. The check items are described below.

- The OM[3:0] pin has to be configured.
- If EXTCLK pin is used for MPLL and UPLL, XTIpll has to be connected to VDD. If XTIpll pin is used for MPLL and UPLL, EXTCLK has to be connected to VDD.
- If an input pin is unused, connect the pin to VDD or GND. If the pin is floated, S3C2440X may not operate.

NOTES

Find price and stock options from leading distributors for s3c2440x on Findchips.com:

https://findchips.com/search/s3c2440x