S3C2440A

32-BIT RISC MICROPROCESSOR

APPLICATION NOTES

PRELIMINARY

Revision 0.191

(June 30, 2004)



ABOUT SMDK2440 BOARD

SMDK2440 CPU and Base board revision number.

CPU Board Version	Rev 1.0
Base Board Version	Rev 1.0

SYSTEM OVERVIEW

SMDK2440 (S3C2440 Development Kit) for S3C2440A is a platform that is suitable for code development of SAMSUNG's S3C2440A 16/32-bit RISC microcontroller (ARM920T) for hand-held devices and general applications.

The S3C2440A 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, AC97 interface, USB host, USB device, SD host & multimedia card interface, Camera Interface, 2-ch SPI and PLL for clock generation.

The SMDK2440 consists of S3C2440A, 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 S3C2440A. It can evaluate the basic operations of the S3C2440A and develop codes for it as well.

SMDK2440 is manufactured by MERITECH Co., Ltd and its website is www.mcukorea.com

When the S3C2440A is contained in the SMDK2440, you can use an in-circuit emulator (MULTI-ICE/REALVIEW-ICE (RVI)/OPENice32-A900).

This allows you to test and debug a system design at the processor level. In addition, the S3C2440A with MULTI-ICE/ REALVIEW-ICE (RVI)/OPENice32-A900 capability can be debugged directly using the MULTI-ICE/ REALVIEW-ICE (RVI)/OPENice32-A900 interface.

Figure 1-1 shows SMDK2440 function blocks.



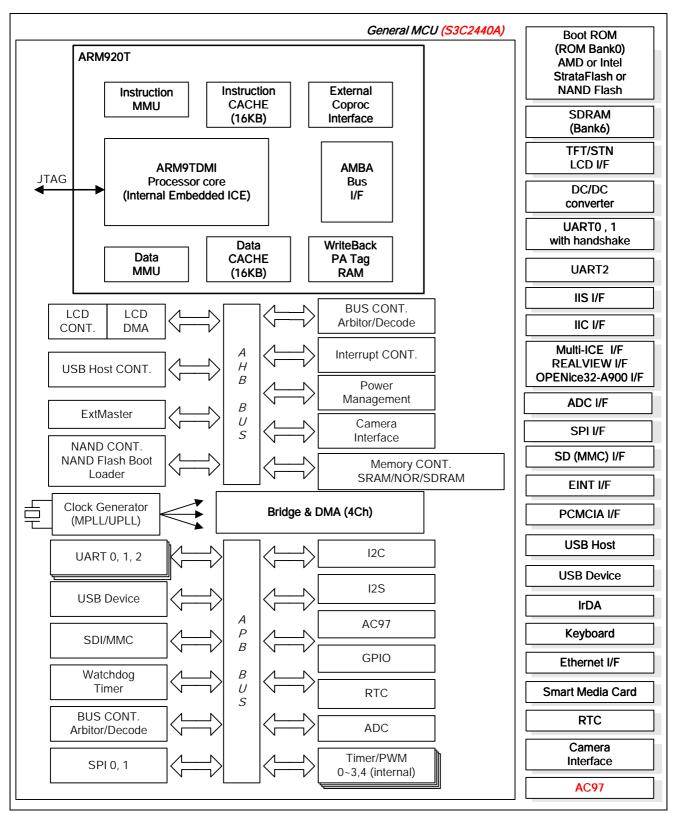


Figure 1-1. SMDK2440 Function Block Diagram



FEATURES

- S3C2440A: 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 USB port
- SD host (MMC) interface
- Smart media card
- JTAG port (MULTI-ICE/REALVIEW-ICE(RVI)/OPENice32-A900 interface)
- RTC X-tal input logic
- IIC with KS24C080
- ADC interface
- SPI interface
- IIS interface (sound CODEC audio input/output)
- AC97 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

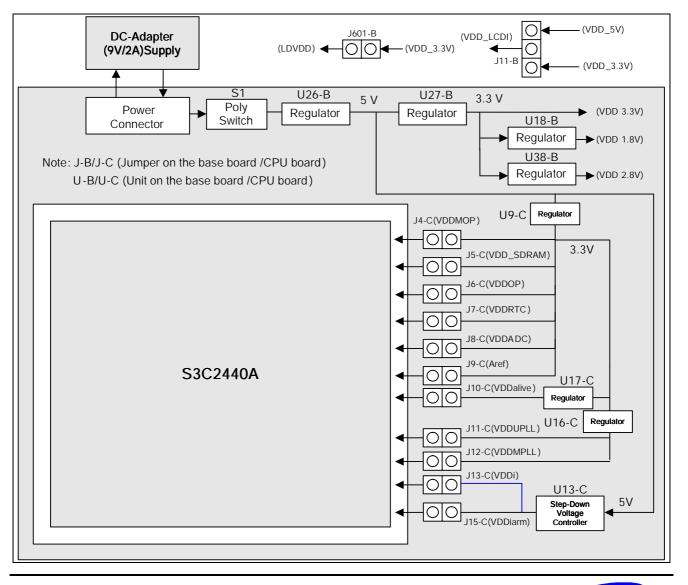
The SMDK2440 is designed to test S3C2440A and develop software while hardware is being developed. Figure 1-3 shows SMDK2440's block diagram.

POWER SUPPLY

SMDK2440 is operated by 1,8V for Extension I/O, 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.

In case of 300Mhz, the Arm core(VDDi and VDDiarm) operates at 1.2V and VDDalive at 1.2V. In 400Mhz, the Arm core operates at 1.3V and VDDalive at 1.3V. But VDDalive can be operates at 1.2V regardless of CPU Frequency.

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~J13-C, J15-C on the CPU board, J11-B and J601-B on the base board are inserted.



S3C2440A

Downloaded from Arrow.com.

1-4

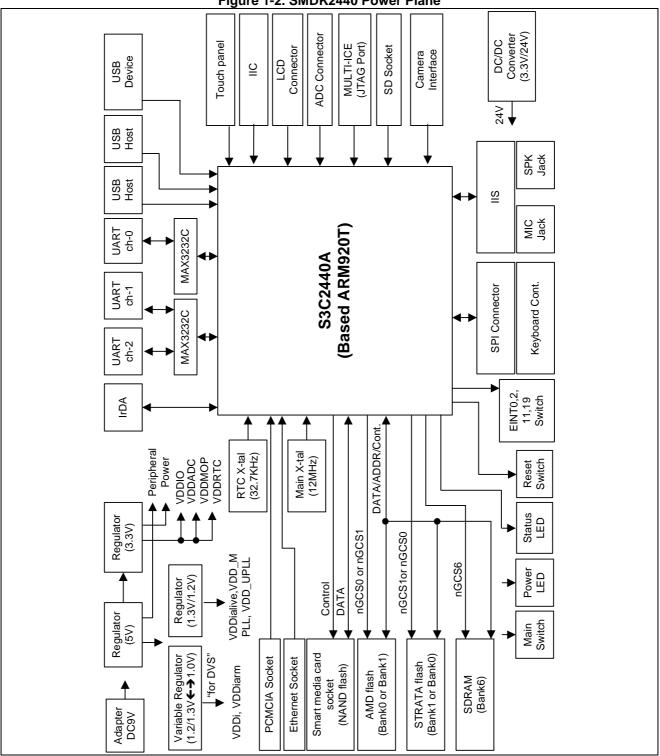


Figure 1-2. SMDK2440 Power Plane

Figure 1-3. Detailed SMDK2440 Board Diagram



SMDK2440 SYSTEM CONFIGURATIONS

CLOCK SOURCE

EXTCLK or X-TAL can be selected for the system clock of S3C2440A and USB by the suitable setting of OM values.

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

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

RTC Clock

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

NOTES:

- 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 J3-C on the CPU board.



BOOT ROM (BANK0)

The data bus width of BANK0 can be configured in byte, half-word or word in S3C2440A.

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).

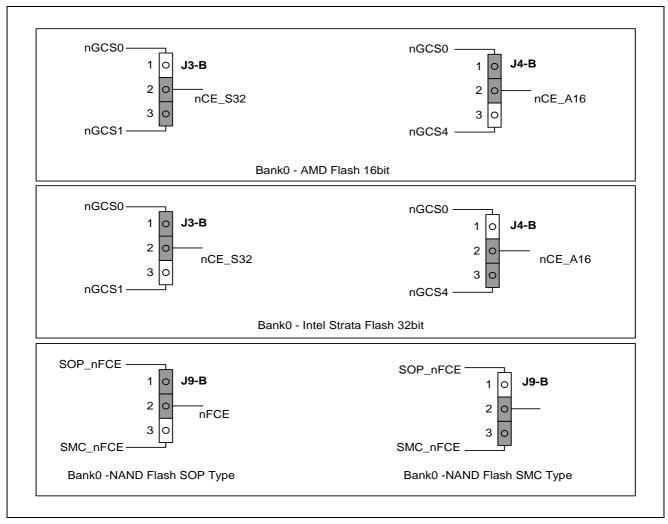


Figure 1-4. SMDK2440 Board Memory Configurations



Pin Functions	J3-В		J4-B		Descriptions
Pin Functions	1-2	2-3	1-2	2-3	Descriptions
BANK0/1/4 memory type	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
selection and data bus width configuration	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 1-2. Memory Type and Data Bus Width

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

NAND FLASH CONFIGURATION

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 S3C2440A'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/О Туре	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, ...



TFT/STN LCD controllers are equipped in the S3C2440A. 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)

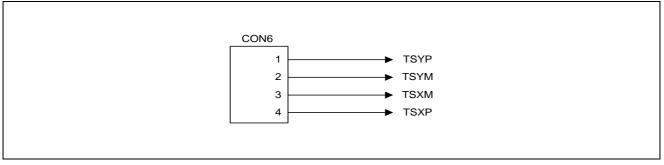
LTS350Q1-PD1 Panel with touch panel and front light unit

LTS350Q1-PD2 Panel only

LTS350Q1-PE1 Panel with touch panel and front light unit

LTS350Q1-PE2 Panel only

TOUCH SCREEN







SPI Connector

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

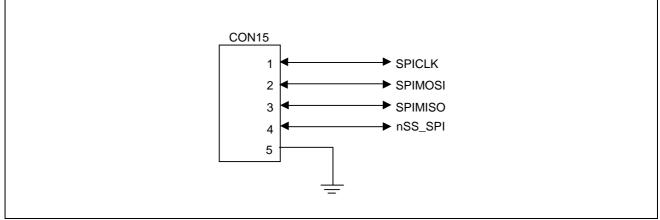


Figure 1-6. SPI Connector on SMDK2440

A/D CONVERTER INTERFACE

The S3C2440A has Analog to Digital Converter (ADC). The ADC has 8-ch analog input signals. The SMDK2440 provides the ADC (CON8) signals as follows:

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

Table 1-7. ADC Interface on SMDK2440



SD HOST (MMC) INTERFACE

SD(MMC) is provided by the S3C2440A and SD card socket (CON13) is supported in the SMDK2440

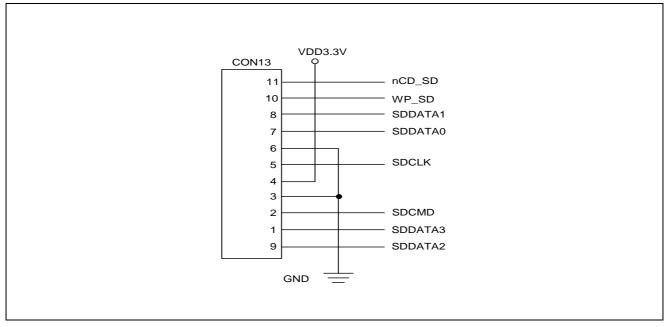


Figure 1-7. SD Card Socket on SMDK2440

IIC INTERFACE

Serial EEPROM S524C80D80 (KS24C080) access function is provided by SMDK2440 and there is also IIC interface between S3C2440A 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.

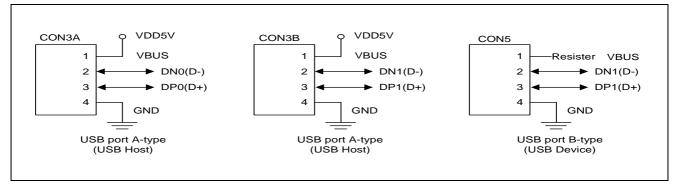
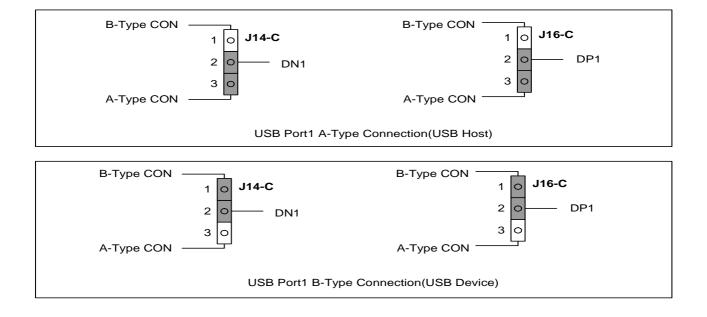


Figure 1-8. USB Ports on SMDK2440

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







UART interface

The S3C2440A 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.

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

 Table 1-8. UART Configurations

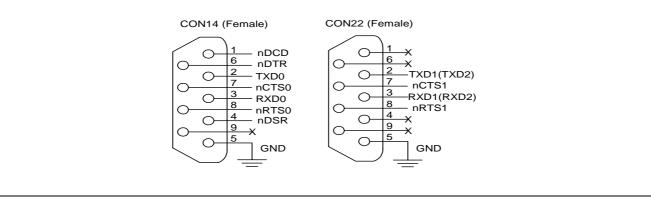


Figure 1-10. UART Ports on SMDK2440



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

# 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	_	_

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

# 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						
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 shows the connector pin assignment for camera interface on SMDK2440

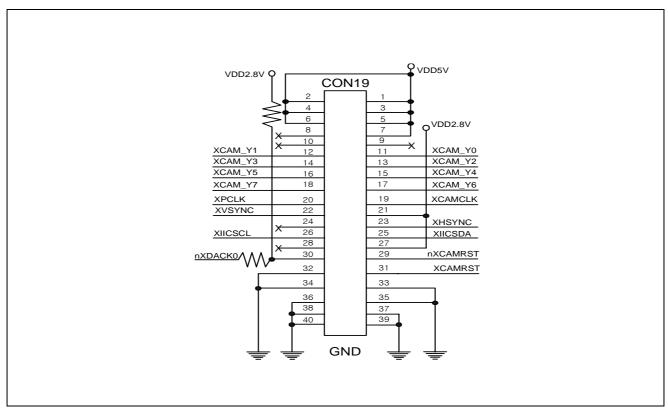


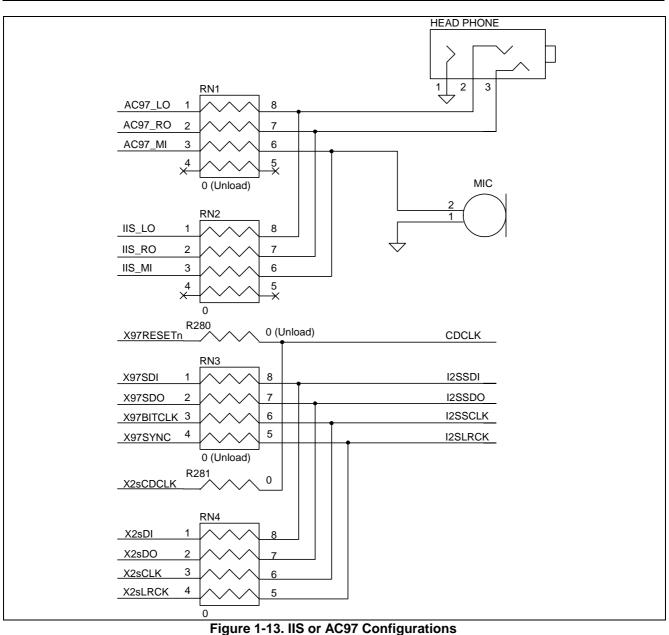
Figure 1-12. CAMERA Interface Connector on SMDK2440



SMDK2440 supports the IIS & AC97 interface. Because AC97 controller and the IIS controller must not be used at the same time, Only one interface of them should be selected by using following resistor configuration.

	AC97	IIS
RN1, RN3, R280	SHORT	OPEN
RN2, RN4, R281	OPEN	SHORT





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**, **RealView ICE(RVI)** and OPENice32-A900, you can use JTAG port on the SMDK2440 to interface the emulator.

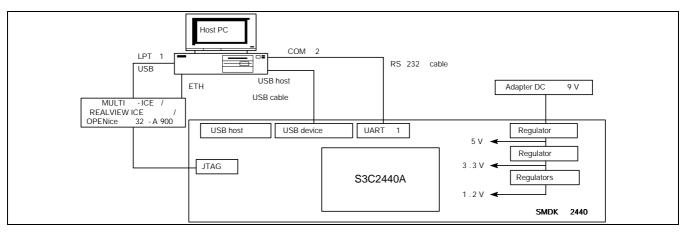


Figure 2-1. Setup Environment for SMDK2440 Board



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.

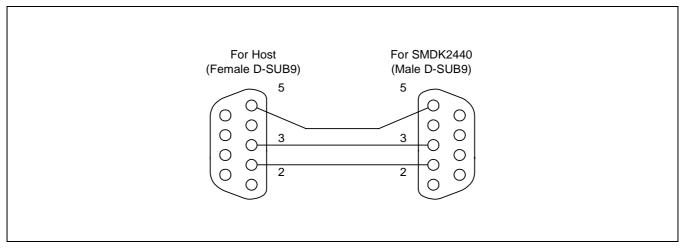


Figure 2-2. Serial Cable Connections for SMDK2440 Board



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.

Add New Hardware	
	This wizard searches for new drivers for:
	SEC S3C2440K Test B/D
	A device driver is a software program that makes a hardware device work.
🎭 🚡	
* *	
	•
	< Back Next > Cancel
	Annu and and and and and a

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': S3C2440A USB downloader firmware.



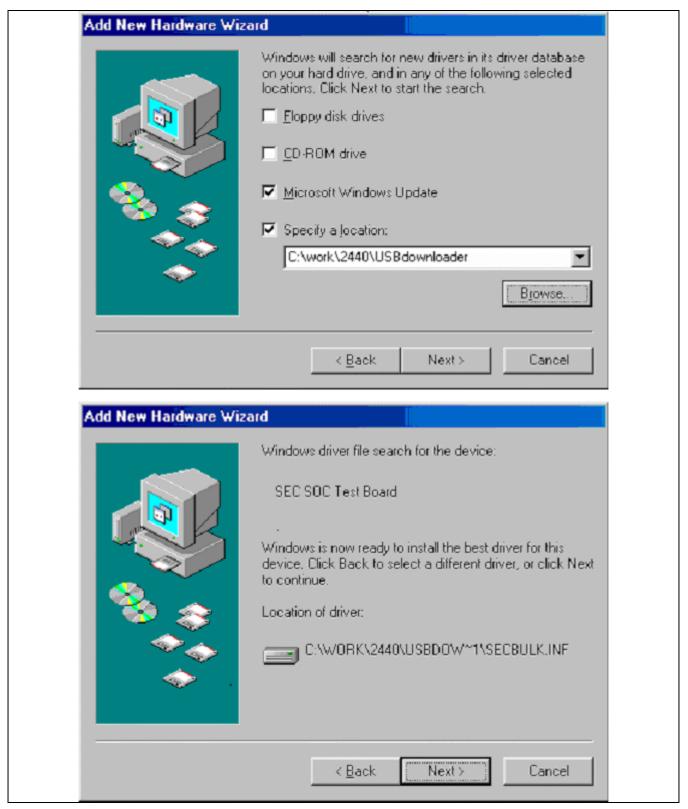


Figure 2-4. USB Device Driver Installation





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 \rightarrow Options
- 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.



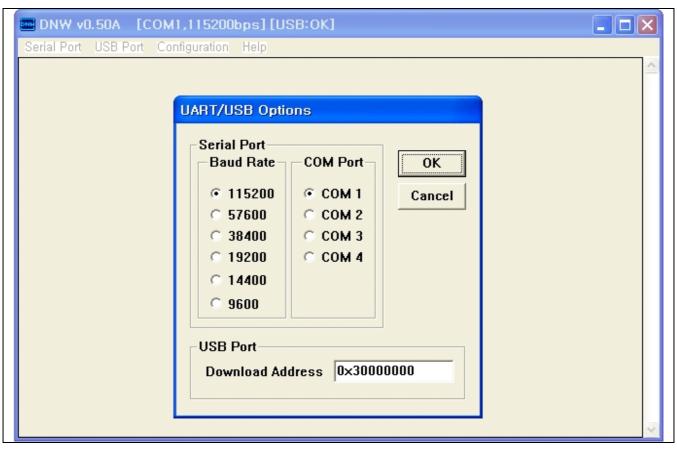


Figure 2-5. Setting UART/USB Options



CONNECT HOST PC AND SMDK2440 WITH DNW

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

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

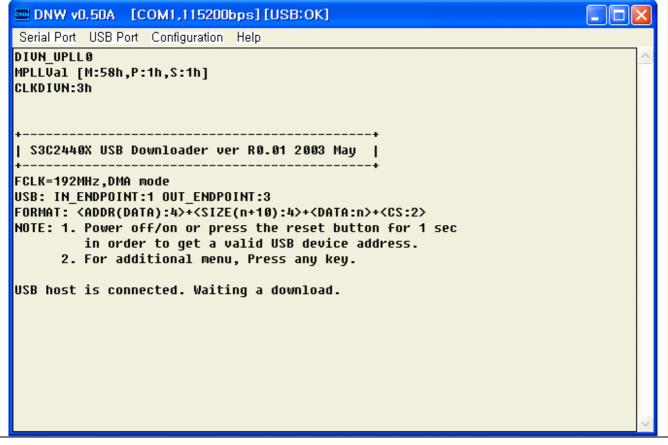


Figure 2-6. Power On Screen



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 S3C2440A 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 S3C2440A 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:

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



DNW v0.50A [COM1,115200bps] [USB:OK] Serial Port USB Port Configuration Help ~ | S3C2440X USB Downloader ver R0.01 2003 May | FCLK=192MHz,DMA mode USB: IN ENDPOINT:1 OUT ENDPOINT:3 FORMAT: <ADDR(DATA):4>+<SIZE(n+10):4>+<DATA:n>+<CS:2> NOTE: 1. Power off/on or press the reset button for 1 sec in order to get a valid USB device address. 2. For additional menu, Press any key. USB host is not connected yet. USB host is connected. Waiting a download. Now, Downloading [ADDRESS:30000000h,TOTAL:390118] RECEIVED FILE SIZE: 390118(1016.1KB/S,0.4S) Now, Checksum calculation Download O.K. [SMDK2440 Board Test Program Ver 0.0] [Fclk:Hclk:Pclk]=[203.2:101.6:50.8]Mhz [Uclk=48.0Mhz] 1:Manual Reg Set2:PCMCIA test3:Stepping stone5:Nand test6:Program Flash7:DMA test9:EINT test10:Cpu speed test11:Power/Clk test13:Camera test14:SPI Test15:IIC Test17:IrDA Test18:SD test19:ADC test 0:User Test 4:nWAIT test 8:FIQ test 12:Lcd test 16:RTC Test 20:ADC TS test 21:Timer test Select the function to test :

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).

If you didn't apply the patch for Multi-ICE v2.2, you must follow the below procedure. But, if you were applied the patch for Multi-ICE v2.2 which can download at <u>http://www.arm.com</u> without payment, you can very easily to use the MULTI-ICE. Because, you can setup the MULTI-ICE server program by "Auto-Configure" menu.

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 \rightarrow 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).



👷 ARM – Multi-ICE Se	rver				
<u>File View Run Control</u>	Connection	<u>S</u> ettings	<u>H</u> elp		
	Ctrl+L				
<u>A</u> uto-Configure	Ctrl+A				
Auto- <u>C</u> onfigure at 20kHz	g	iration			
<u>R</u> eset Target	Ctrl+R				
Lo <u>g</u> <u>S</u> et Log File		_			
<u>1</u> 2440, cfg					
E <u>x</u> it					
-					
l Loodo o configuration filo		lagut	bita 🗖	1	2
Loads a configuration file		Input			<u>د ا</u>

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 S3C2440A 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.



Choose Target	? 🛛			
Target Environments Target RDI File Version ADP 1.5.1 C:\Bin\Bin\Bin\Bin\Bin\Bin\Bin\Bin\Bin\Bin	<u>A</u> dd <u>R</u> emove Re <u>n</u> ame <u>S</u> ave As <u>C</u> onfigure			
Connect the ARM Debugger to a Multi-ICE unit attached to target hardware. Ensure that the unit is powered up and that the server has been configured.				

Figure 2-10. Debugger Configuration: Target Page



ABM Multi-ICE V2.2.3 (Build 1188)	2 🛛	ARM Multi-ICE V2.2.3 (Build 1188)
Connect Processor Settings Advanced Board Translation Location of Multi-ICE Debug using the Multi-ICE connected to: This computer Select a new location (or update):	ce I	Connect Processor Settings Advanced Board Trace (••• Charge the settings for processor: ARNS20T on TAP 0 Cache clean code address The address of a 128 byte region of readable, writable memory that Nutli-(E uses to clean the processor cache, and to restart some cores. Your application must not use this region. The default is 0x50. Address [hexadecima]: 000000000 Reset system on starkup If selected, Multi-(E will assert System Reset with a basekpaint on address D when a debugger connects. This will affect the other processors in a multi-processor system. If elected system on starkup
<u>확인</u> 취소 ·	도응말	확인 취소 도움말

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



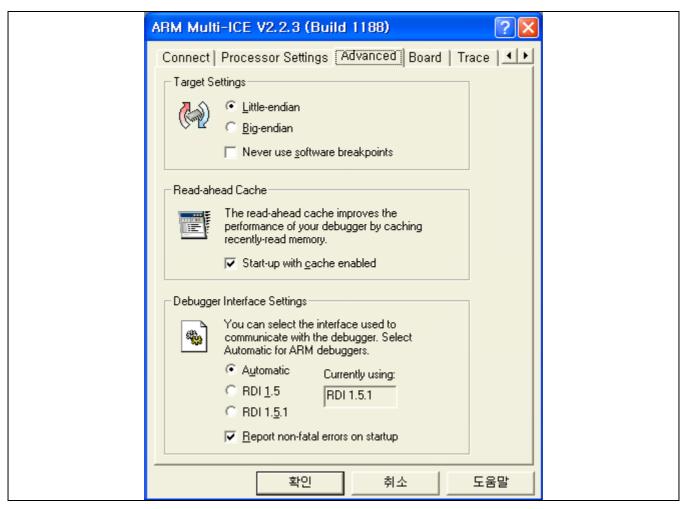


Figure 2-12. Debugger Configuration: Advanced Page



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 0x00swat \$semihosting_enabled 0x00swat psr %IFt_SVCcom 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.



1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 8 1 6		Egecute Options Window Help	8 7 7 7 7 7 7 7 1 1 1 1 1 1 1 1 1 1 1 1
arget Image Fi	line Chart			
TOSEMRA 🗰	0			
ABM920T_0	- Disassembly			
30000000	[0xes000074]	b	ResetHandler 1	Command Line Interface
30000004	[0xea000052]	b	HandlerUndet	Debug >com [FCLK FHS setting:101.25HHz -> Dx7f,2,2]
30000008	[0xea000057]	b	Handler:SWI	Debug >swat *0x4c000004 ([0x7f<<12]+(0x2<<4]+(0x2<<0)]
3000000c	[0xes000062]	b	HandlerFabort	Debug >com [UCLK FHS setting:48MHz -> 0x78,2,3]
30000010	[0xea00005b]	b	HandlerDabort	Debug >swat *0x4c000008 ((0x78<<12)+(0x2<<4)+(0x3<<0))
30000014	[Oxeafffffe]	b	0x30000014	Debug >com << Remory setting >>
30000018	[0xes000047]	b	HandlerIRQ	Debug >com [Bank6/7: 32-bit bus width]
3000001c	[0xea000040]	b	HandlerFIQ	Debug >swat *0x48000000 0x22000000
30000020	[0xea000008]	b	EnterFMDN	Debug >com [BankD-5: Access cycle: 14-clocks, others:0-c.
30000024	[0x0f10ee11]	ded	0x0f10ee11	Debug >swat *0x48000004 ((0ccl3)+(0ccl1)+(7cc8)+(0cc6)+(1
30000028	[0x0080e380]	dod	0x0080e380	Debug >swat *0x48000008 ((0<<13)+(0<<11)+(7<<8)+(0<<6)+()
3000002c	[0x0f10ee01]	ded	0x0f10ee01	Debug >swat *0x4800000c ((0<<13)+(0<<11)+(7<<8)+(0<<6)+(1
30000030	[0xffffffff]	ded	Oxffffffff	Debug >swat *0x48000010 ((0cc13)+(0cc11)+(7cc8)+(0cc6)+()
30000034	[0xffffffff]	ded	Oxffffffff	Debug >swat *0x48000014 ((0<<13)+(0<<11)+(7<<8)+(0<<6)+(1
30000038	[0xffffffff]	ded	Oxffffffff	Debug Sawat *0x48000018 ([0<<13]+(0<<11)+(7<<8)+(0<<6)+(1
3000003c	[0xffffffff]	ded	OxEEEEEEE	Debug >com [Bank6/7: SDRAM, Trod:2clock, CA:9-bit]
30000040	[0xffffffff]	ded	Oxffffffff	Debug >swat *0x4800001c ((3<<15)+(0<<2)+1)
30000044	[0xea000063]	b	ResetHandler	Debug Sawat *0x48000020 ([3<<15)+(0<<2)+1]
EnterPWDN	[0xela02000]	8.077	x2,x0	Debug >com [SDRAM refresh enable, Trp=2clk, Trc=5clk, Ret
3000004c	[0xe3100008]	tst	x0,#8	Debug >swat *0x48000024 ((1<<23)+(0<<22)+(0<<20)+(1<<18)-
30000050	[0x1w00000f]	bre	ENTER_SLEEP	Debug >com [SCHZ_EN enable, SCLM_EN enable, Bank6/7 memor
30000054	[0xe59£00b4]	1dr	r0,0x30000110 ; = #0x480000	Debug >swat *0x48000028 (0x1+(1<<5)+(1<<4))
30000058	[0xe5903000]	1 dr	x3,[x0,#0]	Debug >com [Bank6/7 CL: 3-clocks]
3000005c	[Oxela01003]	107	r1,r3	Debug >swat *0x4800002c 0x30
30000060	[0xe3811840]	OFE	r1,r1,#0x400000	Debug >swat *0x48000030 0x30
30000064	[0xe5801000]	str	r1,[r0,#0]	Debug >
30000068	[Oxe3a01010]	11.077	r1,#Dx10	< > >
3000006c	[0xe2511001]	5 ab 5	x1,x1,#1	
30000070	[Oxlafffffd]	bne	0x3000006c ; (EnterPWDN +	
30000074	FRee R0400081	1.4+	+0 0+101001114 #0+A-0000	

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.

Port Settings	? 🛛
Port Address	OK
AUTO	Cancel
✓ Force <u>4</u> -bit access Current Port Mode 4-bit	Help

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.



EXECUTING 2440TEST.AXF USING ARM REALVIEW ICE (RVI)

1. Initialize internal variables of the debugger. After a downloading, several pane are displayed, such as Register Pane, Call Stack Pane, Watch Pane, Memory Pane and Output Pane. In cmd tab of Output pane, you should initialize the internal variables of the debugger, "@ SEMIHOST_ENABLED " and " SEMIHOST_VECTOR ", by entering the following command:

setreg @SEMIHOST_VECTOR=0x0 setreg @SEMIHOST_ENABLED= 0x0 setreg @CPSR=0xd3

;To use all H/W break points ;To disable all interrupts

Or, you can initialize these variables as follows:

First, create a text file named "2440norom-rv.inc", which includes the select with window environment. More detail inform, you can get the RealView Debugger Manual by ARM.

1. Select "Include Commands from File..." from the Debug menu.

Debug -> Include Commands from File...

2. Select "2440norom-rv.inc" file from the file open dialog box.

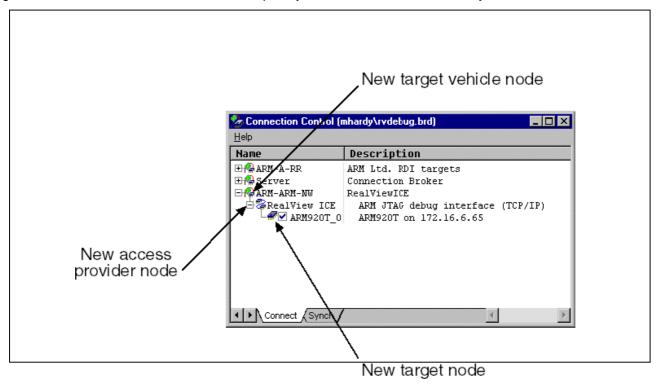
For more information about these steps, refer to the reference document released by ARM.



HOW TO USE RealView Debugger (RVD) WITH RealView ICE(RVI)

To connect to your target hardware using a RealView ICE interface unit, you use the same RealView Debugger features that you use for any other target. You must ensure that you use the RealView ICE target vehicle. To do so, use the nodes in the Connection Control window that are descendents of the ARM-ARM-NW target vehicle node, as shown as following.

When connecting, you might see the error shown as following. This error appears when the software detects that there is already a connection to the target. This might be because someone else is connected to the target, or it might be because a connection has been left open by software that exited incorrectly.



CONFIGURING REALVIEW DEBUGGER FOR REALVIEW ICE

To debug the target board with RealView ICE(RVI), you should configure RealView Debugger(RVD). As RealView ICE should be connected through JTAG port on the board and switched on.

To configure RealView Debugger(RVD) for RealView ICE interface, follow the steps:

1. Turn on the power (Target Board)

Ensure that the target is correctly connected and powered, then turn on the power to RVI. This will take approximately 30 seconds. You can tell this has completed as one of the LEDs beside the JTAG connector will come on (after first flashing, and then switching off for a short time). You must wait for this to complete.

2. Launch RVD and access the Connection Control Window

Select File -> Connection -> Connect to Target to launch the Connection Control windows.



		OEBUG-kealendaro							. 8 X
And Const Lage/Jameis Image: Const Lage/Jameis Stree Se Stree Se Joed Image Constituting Properties Joed Image Constituting Properties Joed Image Constituting Properties Street Street Sease Street S	Pite	New Open	CHI+0	@ * * * * @ 0	8 (B) 8 23 B - [📩 State: Utiknow	9	
Stree Ar. Stree Ar. Street	Not					1	lagikar		* _ B
Consistent in the second secon	<u>C14</u> -	Serve Serve Arc					(No Register C	antext>	
Consistent in the second secon		Warkspace							
Load (wrsps Or SPANO Alsonnest Alsonnest Brind (wrsps Or SPANO Alsonnest Image: Alsonnest			•	Connect to Target	Alt+0				
A I North			P		maria				
				Connection Properties,					
			Ort-Shrbetti	Attach Window to a Connection	T-				
	•		Crosp.		14	7 -	UD (an /	1	E
	÷	Recent Workspaces	*		Talue	÷			-
A 漢書 回;		ELt							
A 漢 👸 🗊 🏹		For see Aroone X st	ance (This / _	E E VARDAT (M	нкого Динкого Динком Ј				-
80aco 	-							▲漢耆 ☑:	
80aco 									
Nazo N klond / Seo / New / News / Secon / Las /									
I I I I I I I I I I I I I I I I I I I	3 10	ao							_
	_		Refind (Second (C	2/			4		P

Click on the cross to the left of the **ARM-ARM-NW** entry to display the **RealView ICE** brach.

🧞 Connection Control (Myunghui_Ry₩rvdebug,brd) 📃 🗖 🗙				
<u>H</u> elp				
Name	Description			
ARM-A-RR ARMulator ARMulator Server ARM-ARM-PP ARM-ARM-PP ARMOAK_MICE ARM-VIA-LP ARM-VIA-LP ARM-VIA-LP ARM-ARM-NW RealView ICE	ARM Ltd. RDI targets ARM instruction set simulator Connection Broker Simulator Broker Multi-ICE direct connect Multi-ICE direct connect (ARM+Oak) Motorola/Macraigor Wiggler emulator Macraigor Wiggler RealViewICE ARM JTAG debug interface (TCP/IP)			





3. Create a new JTAG configuration file

Click on the cross to expand the **RealView ICE** branch.

A List Selection dialogue will appear:

List Selection	×				
Target connection error: Could not load the file C:\Program Files\ARM\RVD\Core\1.7\ 150\win_32-pentium\etc\rvi.rvc See output log for details. You can:					
Retry (you fix the problem first) Edit Board-file to give the specific pathname Configure Device Information					
OK Cancel Help					

Select Configure Device Information... and click OK.

4. Configure the RealView ICE interface unit

The *RVConfig* dialogue should now be visible. Locate the RVI unit on the network by using the *Browse* button.

A. Connecting over a TCP/IP network Click on the Browse button. The TCP/IP Browse dialog appears, and your computer starts scanning the TCP/IP network for available RealView ICE units, adding them to the dialog as it founds them.



2-23

E RVConfig - C:\Program Files\ARM\RV File View Help	D₩Core₩1, 7₩150₩win_32-pentium₩etc₩rvi, rvc 📃 🗖 🗙
RealView ICE (Not Connected)	Connection Type
	O USB
	• TCP/IP
	IP Address / Host Name <u>B</u> rowse
	Connect, Disconnect

C-TCP/IP Brows	se		? ×
		Sca	nning TCP/IP
Found1 ' 192,168,1,22) 22 <rvi></rvi>		
IP Address / Ho	ost Name 🛛 🛙 19	2, 168, 1, 222 <rvi></rvi>	>
ОК	Stop	Cancel	Help

B. When you have chosen the RealView ICE unit that you want to use, click *Connect*. A connection is established to the unit.



Eile View Help	RM₩RVD₩Core₩1,7₩150₩win_32-pentium ×
HealView ICE (Not Connected)	Connection Type
-	C USB
	© TCP/IP 192, 168, 1, 222 < RVI>
	IP Address / Host Name
	<u>B</u> rowse
-	Connect Disconnect

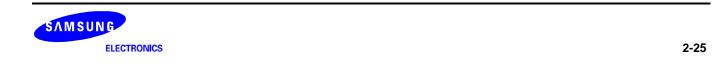
After connection, you can see the following screen.

Elle View Help BealView ICE: (TCP/IP 192, 168, 1, 222) Curvices Move Dave Curvices Move Down JTAG Clock Speed Adaptive Sufficient Curvices Move Curvices Move	E-RVConfig - C:₩Program Files₩ARM₩RVD	₩Core₩1, 7₩150₩win_32-pentium₩etc₩rvi, rvc * 📃 🗖 💌
Scan Chain Configuration 	<u>F</u> ile <u>V</u> iew <u>H</u> elp	
JTAG Clock Speed C Adaptive C 5 MHz C 50 MHz C 20 kHz C 10 MHz C Other MHz Set	E-RealView ICE: (TCP/IP 192,168,1,222) ⊖-Devices ·UNKNOWN	TAP ID Device ID Code IR Length Options Template 0 UNKNOWN 0x0032409D 4
		JTAG Clock Speed
		C 20 kHz C 10 MHz C Other MHz Set

Click the Connect and then click Auto Configure Scan-Chain

Now RVI should connect to your target board: -- Generally Device with ARM cores. Now, the RealView Debugger cannot recognize the Samsung ID code. So, you must add the new device by handcrapt.

Click Remove Device, and Add Device button.



Now you can select the ARM core by handcrapt for your target.

C- Add Device ? 🗙
Device
Registered Devices

C Custom Device
Device Name UNKNOWN
IR Length 4 🚔
<u>OK</u> <u>C</u> ancel <u>H</u> elp



1. Configuring Device

E-RVConfig - C:₩Program Files₩ARM₩RVD♥ File View Help	∀Core₩1, 7₩150₩win_32-pentium₩etc₩rvi, rvc	* X		
⊡-RealView ICE: (TCP/IP 192,168,1,222) ⊖-Devices	Device: ARM920T using the ARM920T template version 1:0:0			
ARM920T Advanced	Item	Value		
Auvanceu	Code Sequence Code Address	0x31FFFF00		
	Code Sequence Code Size	0x000000FF		
	Code Sequence Timeout (ms)	3000		
	Bypass memory protection when in debug	✓ True/False		
	Ignore bad JTAG IDCODE	🗖 True/False		
	Use watchpoint for software breakpoints	✓ True/False		

Enter the value 0x33FFFF00 in the Code Sequence Code Address Item.

Enter the value **0x000000FF** in the Code Sequence Code Size Item.

Select File -> Save before File -> Exit.

Now, save the current configuration and close the configuration dialogue.



Name	Description
⊡ 🖗 ARM-A-RR	ARM Ltd. RDI targets
🗄 🞏 ARMulator	ARM instruction set simulator
🗆 🚷 Server	Connection Broker
🗄 💐 localhost	Simulator Broker
🗆 🚱 ARM-ARM-PP	Multi-ICE direct connect
🗄 🞏 ARMOAK MICE	Multi-ICE direct connect (ARM+Oak)
🗆 🥵 ARM-VIA-LP	Motorola/Macraigor Wiggler emulator
🗄 🞏 MOT_WIGGLER	Macraigor Wiggler
🗆 🥵 arm-arm-nu	RealViewICE
🗄 👺 Real View ICE	ARM JTAG debug interface (TCP/IP)
- <i>-</i> →	ARM920T on 192.168.1.222

2. Connect to the target

Return to the RVD *Connection Control* window and expand the *RealView ICE* branch, then connect to the target:

Click ARM920T_0 item.

<u>H</u> elp		
Name	Description	
🗆 🚱 ARM-A-RR	ARM Ltd. RDI targets	
🗄 🞏 ARMulator	ARM instruction set simulator	
🗆 🗛 Server	Connection Broker	
🗄 🔁 localhost	Simulator Broker	
🗆 🗛 ARM-ARM-PP	Multi-ICE direct connect	
🗄 🞏 ARMOAK_MICE	Multi-ICE direct connect (ARM+Oak)	
🗆 🖗 ARM-VIA-LP	Motorola/Macraigor Wiggler emulator	
🗄 🞏 MOT_WIGGLER	Macraigor Wiggler	
🗆 🖗 ARM-ARM-NW	RealViewICE	
🗄 👺 RealView ICE	ARM JTAG debug interface (TCP/IP)	
	ARM920T on 192.168.1.222	



3. Confirm the connection

Now, your RVD, RVI, Target board connection were completed and you can see the following figure :

ZRVCEBUG (calendar) = @ARM680TL0ARM-ARM-NW	_ 2	×
Bie Edit Find Mew Emiect Inois Debug Help		
□ \$\$\$\$ 2 \$\$\$\$ 1 \$\$ \$\$ \$\$ \$\$ \$\$ \$\$ \$\$ \$\$ \$\$ \$\$ \$\$	State: State:	
Me BNO_SOURCE Prie Unit: Int.		
to source for context: (Unknown)	a Replan	
Click to Load "En/workbook/UND/intre/session1/DebugRel/calendar.act"	P0 342 P1 838795368	4
	R2 4294967295 R3 2205	
	R4 342 R5 147	
	R6 4294967295 R7 030795360	
	R8 838795368 R9 50348168	
	P10 131072 P11 0	
	P12 4278255615 57 838817728 13 6896 2C 6940	
	13. 6096 PC 6940 CFIR 2143480667	
	NECV FIG ING STATE MODE	
	1010 ENA ENA ADE SVC	-
	B TER	
	M IFO	
		•
<u>+ + (less) (m) / / / / / / / / / / / / / / / / / / /</u>		
Take Value D Nake Value D	<bokddrb< td=""><td>•</td></bokddrb<>	•
Chronollipichknown Incartions	(Bakddz)	
(0x00001AF) (0xhnom Location)	<boaddr></boaddr>	
	<bokdur> (Rokdur></bokdur>	
	(Bokddr)	
	(Bok60:)	
	diskdap	
	<bakddr></bakddr>	
E + > Cal Stuck (Locals (Statics (This / T)) + + > Weight (Weight (Weight) T)	(Bakddz)	-1
		-
D > connect, route \$		•
> readboard "C:\Program Files\&FM(SFD\Core\1.7\150\win_32-pentium\home\Hyunghui_Ry\rvdebug.bud" > readboard "C:\Program Files\&FM(SFD\Core\1.7\150\win_32-pentium\home\Hyunghui_Py\rvdebug.bud"		
> connect, conte 5		
> connect 6	▲漢竇 ◎ ⇒	11
ADE FealView ICE		
Base H/W: W1 Rev C-01		
TAEBOTAP REVI 1.00		
Firaware: 1.1.0, Beta, Build 126		
Copyright #FM limited 2002,2003 Attached to stopped device		
Warning: No statk/heap of top of memory defined - setting top of memory to 0x80000.		
Rose Little Endian		
Stap		
V V Cond (2000 / Bulle / Method / SecChi / Log /	7	*
For more information, select Help from Menu I.n 1	I, Cal 1	F



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.

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.

Target Environ	RDI File		Version	Add
ADP ARMUL OPenice32	1.5.1 C:\\Bin\	fRemote_A, dll #ARMulate, dll #DMWOPENice32, A90	1,2,0,805 1,2,0,637 (dil 1, 41, U, U	Remove
or encode				Rename
				Save As
				Configure
You can use the This supports a but semihosting	e ARM Debugger with OF II functions of ARM debu	PENice32. gger		

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.

OPENIce32-A900 ver. 1.41 for ADW & AXD	×
Remote Debugger SMU Flash config Help	
Remote Set	
C Serial Port Baud Rate	
USB Address	
C Ethernet	
	1
Set IP Address Get IP Address	
확인 취소 적용(A	7

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

Debugger page : set the Endian and decide where initializes S3C2440A 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 S3C2440A 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.)

OPENIce32-A900 ver. 1.41 for ADW & AXD
Remote Debugger SMU Flash config Help
Jtag Clock 10 MHz To init SMU, Check I
Flash download
C Big Endian C SDT 2.50 or more (rdi 1,50)
(* Little Endian) (* ADS 1.0 or more (rdi 1.51)
<u>확인</u> 취소 적용(<u>A</u>)

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

SMU page: set SMU of S3C2440A. Select SMDK2440 or S3C2440A 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.

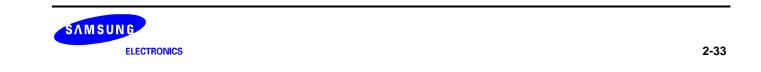
 OPENIce32-A900	ver. 1.41 for AD	DW & AXD	X
Remote Debugger	SMU Flash con	fig Help	
Device name s3C244	• 0	43	Dev, Update
No Register 01 WTCON 02 INTMSK 03 INTSUBMSK. 04 LOCKTIME 05 MPLLCON 06 BWSCON 07 BANKCOND 08 BANKCON2 09 BANKCON2 10 BANKCON4 12 BANKCON5 13 BANKCON7 15 REFRESH	Address 53000000 4a000008 4a00000 4c000000 4c000004 48000004 48000004 48000006 48000006 48000006 48000010 48000010 48000018 48000016 48000016 48000016 48000016 48000016 48000016 48000016 48000016 48000016 48000016 48000016 48000016 48000016 48000016 48000016 48000016 48000016 48000016 48000016 48000016 48000016 480000000 480000000 4800000000000000	8021 R ffiffiff R 7 R 5c080 R 700	trib. Size ▲ /₩ 4 ▲ /₩ 4 ↓ /₩ 4 ↓ /₩ 4 ↓ /₩ 4 ↓ /₩ 4 ↓ /₩ 4 ↓ /₩ 4 ↓ /₩ 4 ↓ /₩ 4 ↓ /₩ 4 ↓
New		_	Add Delete
	확인	취소	적용(<u>A</u>)

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).

OPENIce32-A900 ver. 1.41 for ADW & AXD
Remote Debugger SMU Flash config Help
Device Am29LV800BB
Erase Data bus width
C Sector(Block) C Chip C Bbit C 16bit C 32bit
Address(Hex) Flash base 00000000
RAM base 30000000 RAM size 3FFFFF
확인 취소 적용(<u>A</u>)

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.

Choose Target	? 🛛				
Target Environments Version Target RDI File Version ADP 1,5,1 C:\U00c0,\U00c0Bin\U00c0Remote_A,dll 1,2,0,805 ARMUL 1,5,1 C:\U00c0,\U00c0Bin\U00c0ARMulate,dll 1,2,0,837	Add				
Multi-ICE 1,5,1 C:W.,,WMulti-ICE,dll 2,2,3,1188 OPenIce32 1,5,1 C:WARM251WbinWOPENice32_A900,dll 1,41,0,0	<u>Remove</u> Re <u>n</u> ame				
	Save As				
Please select a target environment from the above list or add a target environment to the list. Note that a target environment has to be configured at least once before it can be used.					
OK Cancel	Help				

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.

rget Image Fi		Se 50	E 🔍 E			0 0 0	6 🖸
a second se	iles Class						
IOIIUA 🗮 🖬							
(0) - Registers	10:12:38,204		00	(void *)Rtc_Test,	"RTC Test	21	
gister	Value	^	89 90	(void *) IrDA_Test,	"IxDA Test "3D test	Ĩ.	
urrent.	()		91	(void *)Test_SDI, (void *)Test_Adc,	"ADC test		
-10	0x3005EC6C		92	(void *)Test_AddTs,	"ADC TS test	-'	
-11	0x30083DAC		93	(void *)Test Timer,	"Timer test	~,	
-r2	0x00000000		94			,	
-13	0x300B3DAC		95	0,0			
-14	0x00000080		96	32			
-15	0x00000006		97				
-16	0x00000000		98	//*************************************			
-E7	0×FEFFFFFF		99				
-18	0x00000000		100	main di Wantan (annu de)			
-19	0×FFFFFFFF		101	void Nain(void)			
			103	int is			
			104				
			10.5				
			106	Led_Display(0xf);			
			107				
			108	// MBU init. I/D cache on.			
po	0x30000494	*	4				
-r10 -r11 -r12 -r13 -r14 -p0	0x00000000 0x00000000 0x0000483D 0x33FF5800 0x300002A4 0x300002A4	×	103 104 105 106 107				

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.



DEBUGGING DOWNLOADED IMAGE IN REALVIEW DEBUGGER(RVD)

Select File Load Image... from the Code window main menu to load an image to a processor for execution. This displays the Load File to Target dialog box shown as following

👰 Load File t	o Target		<u>? ×</u>
Look in: 🔁	RVD	🗈 (* 🖬 •
Core Examples IMP Product Tools			
File name:			Open
Files of type:	Absolute Files [".axi;".out;".eli;".a]	•	Cancel
			Help
Symbols 0	Inly Replace Existing File(s)	PC	
Target Name:		Auto-Se	et PC
Arguments		Set PD	to Entry point

This dialog box contains controls to configure the way the image is loaded for execution:

Symbols Only

By default, any object file loaded from this dialog box also loads the symbols. If you want to load only the symbols then select this check box, for example when you are working with ROM images.

If the program was initially compiled without a symbol table then you must recompile the program before loading only the symbols.

Replace Existing File(s)

By default, loading a new image overwrites any image currently loaded to the target. If you are working with multiple applications, use this check box to carry out separate loads of associated modules such as an RTOS and associated applications.

Target Name:

Use this field to enter the target name, where supported. A name entered here is then used as the argument to a LOAD command (see Specifying the load instruction).



Arguments:

Use this field to enter a space-separated list of arguments to the image.

Entries in this field create an arguments list used with the LOAD command (see Specifying the load instruction).

PC

When you load an image to the debug target you can optionally set the Program Counter (PC):

Auto-Set PC

Selected by default, this control defines the location of the PC when you load an image. RealView Debugger tracks the state of the other check boxes on this dialog box and sets the PC at the normal entry point, if you select the check box Replace Existing File(s).

Unselect the Auto-Set PC check box to have control over the PC when you load an image.

Set PC to Entry point

Where selected, RealView Debugger sets the PC at the start address specified in the object module.

This is the default if you select both:

Auto-Set PC Replace Existing File(s).

Unselect the Set PC to Entry point check box to prevent the load command setting the PC.

If you have started RealView Debugger and are connected to a debug target, you can load an image for execution from the Process Control pane:

1. Select *View -> Pane Views -> Process Control Pane* from the default Code window main menu to display the Process Control pane.

Whilst there is no image loaded, the pane only shows details about the debug target processor and the current location of the PC.

2. Right-click on the top line, the Process entry, to display the Process context menu.

Whilst there is no image loaded, you can also display this menu from the Image entry.

- 3. Select *Load Image...* to display the Load File to Target dialog box.
- 4. Complete the entries in the dialog box, described in Using the Load File to Target dialog box, to load the required image.



EXECUTING 2440TEST.AXF USING REALVIEW ICE(RVI)

- 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 RealView Debugger (RVD)

<mark>∦</mark> RVDEBUG(2440test) = @ARM920T_0:ARM-ARM-N₩	
Ele Edit Find Yew Project Tools Debug Help	
🗅 😂 🖬 🐇 🖻 💼 📄 🔁 🗗 🖓 🗊 🕐 🗥 🕼 🔺 🖊 🗶 🎒 🥵 🕼 🥸 💱 🎊 👘 💭 🔛 🔂 State: [Stopped	
File: 2440init.s Find:	
streg r0,[r0,-r10,ror #1] ;DCD 0x070000ea	▼_ ⊡
] R0 00000000 I R2 00000000 b ResetHandler R4 0027BC3] ResetHandler R4 0027BC3	9 R3 33FF5630 5 R5 00000000
J HandlerUndef phandler for Undefined mode P6 0000127 b HandlerSUI phandler for SUI interrupt R8 0000000 b HandlerFabort phandler for PAbort R10 0000007 b HandlerDabort phandler for DAbort R12 0000007 b . preserved CPSR 0000007 b HandlerIRQ phandler for TRQ interrupt R2CV FIG [RQ) b HandlerFIQ phandler for FIQ interrupt NZCV FIG [RQ)	D P9 0000000 R11 30025020 4 SP 33FF5790 C PC 30000000 3 STATE NODE
;80x20 b EnterPWDN ; Must be 80x20. ⊞ IR0	
ChangeBigEndian ■ FIQ ;80x24 ■ FIQ DCD 0xeell0f10 > 0xeell0f10 DCD 0xe010f10 > 0xeell0f10 DCD 0xe010f10 > 0xeell0f10 DCD 0x0010eell > 0xeell0f10 DCD 0x0000e380 > 0x100f11ee DCD 0x0000e3 > 0x100f11ee DCD 0x0000e3 > 0x100f11ee DCD 0x000f1ee > 0x100f1ee 1 { Max > 1	15 ∠ Cache Operations ∠TLB Operations ∠D ⊀
Charles and a stack (Locals (Statics (This /)))	
<pre>> connect,route 6 > connect 8 ARR RealYiew ICE Base H/W: VI Rev C-01 TurboTAF Rev: 1.00 Firmware: 1.1.0, Fuild 148 Copyright ARM Limited 2002,2003 Attached to stopped device Warning: No stack/heap or top_of_memory defined - setting top_of_memory to 0x80000. Mode: Little Endian > load/pd/r 'C:\work\2440\2440test-r03\2440test.axf' Loading file C:\work\2440\2440test-r03\2440test.axf</pre>	
Stop> Cind J Stallo J Build J FileFind J SrcCtrl J Log J	
Done Loading	

Figure 2-21. RealView Debugger (RVD): After Downloading

.



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.

	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

Table 2-1. Toolkit Switching Options

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

- 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).
- 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 = 2440 swis
CM1 = 2440lib
CM2 = mmu
CM3 = 2440iis
CM4 = timer
CM5 = 2440RTC
CM6 = 2440IIC
CM38 = spi
CM39 = strata32
#### Destination path Definition ####
OBJ=.\obi
ERR=.\err
#### ARM tool Definition ####
ARMLINK = armlink
ARMASM = armasm
ARMCC = armcc
#### Option Definition ####
LFLAGS = -ro-base 0x3000000 -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).0 $(OBJ)\$(CM22).0 $(OBJ)\$(CM23).0 $(OBJ)\$(CM24).0 \
       $(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) -0 $(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 only one methods to write images to NAND flash memory:

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



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).

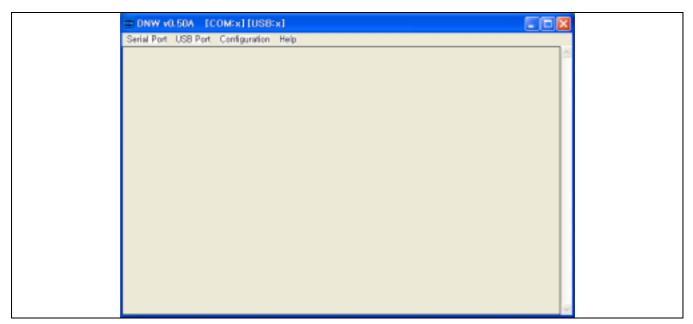


Figure 3-1. DNW Window to Download

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

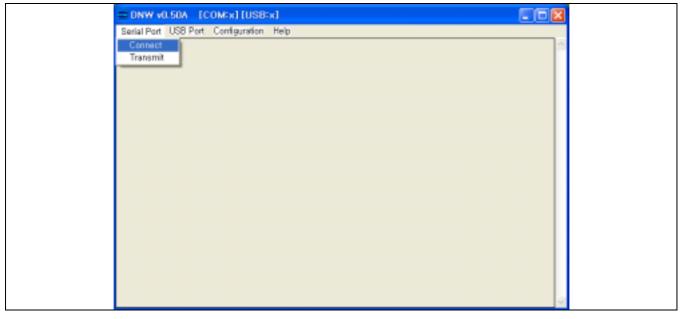


Figure 3-2. DNW Window (to Connect Serial Port)



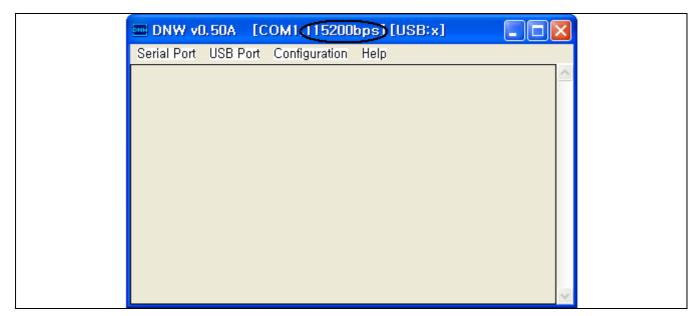


Figure 3-3. DNW Window (after Open Baud-rate is Printed on Title Bar)

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.

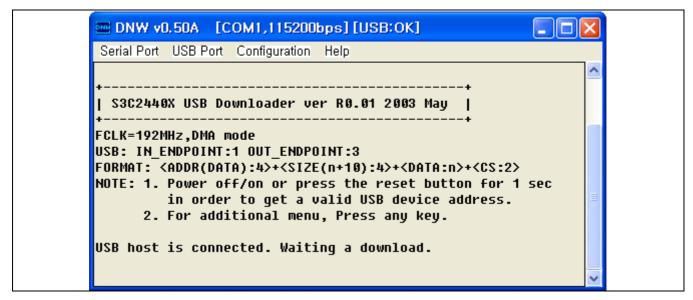


Figure 3-4. DNW Window (after Turning on the SDMK2440)



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

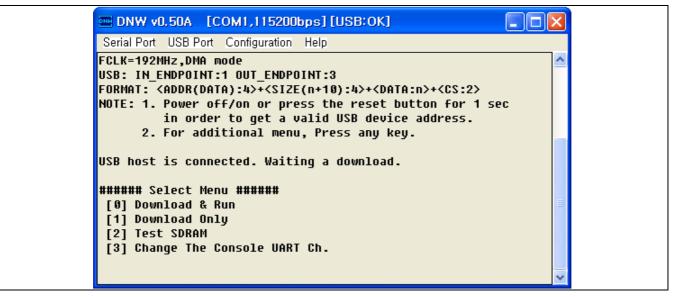


Figure 3-5. DNW Window to Download

- 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.

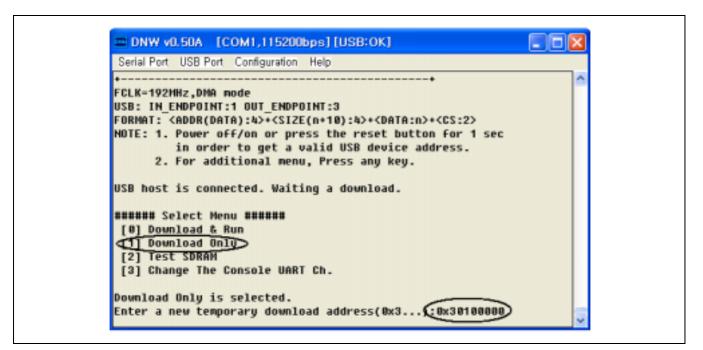


Figure 3-6. DNW Window (to Select Target Image)



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

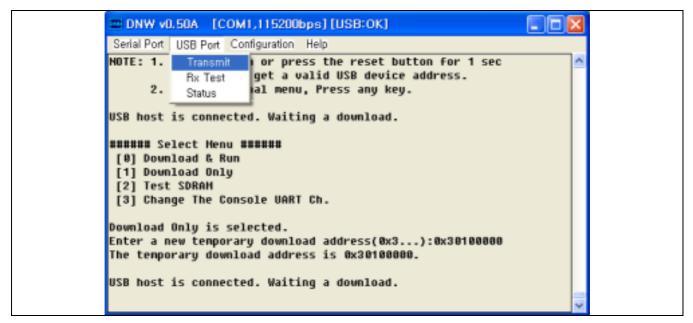


Figure 3-7. DNW Window (for USB Downloading)

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

DNW v0.50A				_ 🗆 🗙	
	ort Configuration	Help			
열기					2 🛛
發는 위치([):	2440Boot-load	er	•	🗢 🗈 💣 🔝 -	
내 최근 문서	⊞ 2440loader,bin				
() 바람 화면					
) 내 문서					
내 컴퓨터					
	TOLOIR ALL	Dura de la			0171/01
	파일 미름(<u>N</u>): 파일 형식(<u>T</u>):	2440loader,bin BIN Files (*,bin:*,nb0)		•	열기(<u>0</u>) 최소
	ware - (D)	pline riles (*, plin, *, libb)		<u> </u>	취소

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).

🔤 DNW v	.50A [COM1,	115200bps][USB:OK]			
Serial Port	USB Port Config	guration Help			
Download Enter a r The tempo	Rx Test	cted. download address(0x3):0x30100000 1 address is 0x30100000.	<u>^</u>		
USB host	is connected.	. Waiting a download.			
RECEIVED Now, Chec	How, Downloading [ADDRESS:30100000h,TOTAL:3282] RECEIVED FILE SIZE: 3282(41.8KB/S,0.1S) How, Checksum calculation Hownload O.K.				
⊈0] Dowr	<u>lect Menu</u> ### load & Rup> load Only	****			

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



DNW v0.50A [COM1,115200bps] [USB:OK]					
Serial Port USB Port C	onfiguration Help				
[SMDK2440 Board Tes [Fclk:Hclk/Pclk]=[4 [Uclk=48.0Mhz]		z		~	
4:Nand test 8:Power/Clk test 12:IIC Test	5:Program Flash 9:Lcd test 13:RTC Test	2:PCMCIA test 6:DMA test 10:Camera test 14:IrDA Test 18:Timer test	7:Interrupt test 11:SPI Test 15:SD test		
Select the functior Nand test Select Nand flash t	Ŭ	anced(2) : 1			
K9S1208 Nand flash	test start.				
4:Page write		2:Block erase 6:Check Badblock			
Select(-1 to exit)	9				
[SMC(K9S1208V0M) NA The program buffer:					
Source size:0h~0h					
Available target bl Input target block				~	

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).



🚥 DNW v0.50A 🛛 [CO	M1,115200bps][USB:	OK]		
Serial Port USB Port C	Configuration Help			
Ø:Read ID	dulland worst	2:Block erase		^
4:Page write	5:Nand R/W test 9:K9S1208 Program	6:Check Badblock		
Select(-1 to exit)	: 9			
	AND Flash writing pr : 0x30100000~0x31fff			
Source size:Oh~Oh				
Available target b Input target block	Lock number: 0~4095			
	size(bytes) 0x4000	D		
	,0-page,0-bycesj.			
0:Read ID	1:Nand reset 5:Nand R/W test	2:Block erase	3:Page read	
4:Page write 8:Soft Unlock		O:CHECK BAUDIOCK	7:Mand Block lock	
Select(-1 to exit)				
SMC(K9S1208V0M) NAM Block # to read: 0	ND Page Read.			
Page # to read: 0				
Read OK.				
Read data(0-block,	0-page)			
0.74 00 00 oc	52 00 00 ea 57 00 0(1 o 2 6 9 9 9 o c -		
	52 00 00 ea 57 00 01 Fe ff ff ea 47 00 01			
	11 ee 10 Of 80 e3 80			
	FF FF FF FF FF FF FF			
	53 00 00 ea 00 20 al			
	b4 00 9f e5 00 30 9(80 40 96 e5 40 40 e			
00:40 18 81 83	00 10 80 e5 10 10 at	0 e3 01 10 51 e2		×

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.
- 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.



Downloaded from Arrow.com.

3-10

NAND FLASH ECC (ERROR CHECKING AND CORRECTION)

The S3C2440A supports ECC algorithm, which is based on XOR calculation, for error checking and correction.

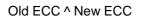
1. Example of one byte ECC (find error bit)

Old

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

New

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



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

Old ECC code					
P3	NP3	P2	NP2	P1	NP1
1	0	1	0	1	0

New ECC code					
P4	NP3	P2	NP2	P1	NP1
0	0	0	0	1	1

S3	NS3	S2	NS2	S1	NS1
1	0	1	0	0	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	()
Result			Bi	t 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]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

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

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

— Column parity (CPn)

	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

2. Example of n byte ECC

S3C2440A



RP1 =	NRP1 =
[7]511^[6]511^[5]511^[4]511^[3]511^[2]511^[1]511^[0]511	^[7]510^[6]510^[5]510^[4]510^[3]510^[2]510^[1]510^[0]510
^[7]509^[6]509^[5]509^[4]509^[3]509^[2]509^[1]509^[0]509	^[7]508^[6]508^[5]508^[4]508^[3]508^[2]508^[1]508^[0]508
^[7]507^[6]507^[5]507^[4]507^[3]507^[2]507^[1]507^[0]507	^[7]506^[6]506^[5]506^[4]506^[3]506^[2]506^[1]506^[0]506
^	^
^[7]1^[6]1^[5]1^[4]1^[3]1^[2]1^[1]1^[0]1	^[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]383^[6]383^[5]383^[4]383^[3]383^[2]383^[1]383^[0]383
^	^
^[7]384^[6]384^[5]384^[4]384^[3]384^[2]384^[1]384^[0]384	^[7]256^[6]256^[5]256^[4]256^[3]256^[2]256^[1]256^[0]256
[7]255^[6]255^[5]255^[4]255^[3]255^[2]255^[1]255^[0]255	^[7]127^[6]127^[5]127^[4]127^[3]127^[2]127^[1]127^[0]127
^	^
^[7]128^[6]128^[5]128^[4]128^[3]128^[2]128^[1]128^[0]128	^[7]0^[6]0^[5]0^[4]0^[3]0^[2]0^[1]0^[0]0

NRP8 =

— Row parity (RPh)	
RP9 =	NRP9 =
[7]511^[6]511^[5]511^[4]511^[3]511^[2]511^[1]511^[0]511	[7]255^[6]255^[5]255^[4]255^[3]255^[2]255^[1]255^[0]255
^	^
^[7]384^[6]384^[5]384^[4]384^[3]384^[2]384^[1]384^[0]384	^[7]128^[6]128^[5]128^[4]128^[3]128^[2]128^[1]128^[0]128
^[7]383^[6]383^[5]383^[4]383^[3]383^[2]383^[1]383^[0]383	^[7]127^[6]127^[5]127^[4]127^[3]127^[2]127^[1]127^[0]127
^	^
^[7]256^[6]256^[5]256^[4]256^[3]256^[2]256^[1]256^[0]256	^[7]0^[6]0^[5]0^[4]0^[3]0^[2]0^[1]0^[0]0

Row parity (RPn)

- 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	()		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	()		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 RealView ICE(RVI).
- 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.

18 0 📽 🖬 🖷			
rget Image Files Cla			
D TOSEMRA #			
ARM920T_0 - Disast		. 🗆 🗙	Command Line Interlace
30000080 [Owe2511001]	500.0	x1,x1,#1	Command Line Interface
30000084 [0x1afffffd]		0x30D00D80 ; (EnterFMDN + 0x38)	Debug >sbey C:\W0800\2440\2440norom\2440norom.ini
30000088 [Dxe5920080]		z0,0x30D00110 ; = #0x45000024	Dabug >rom
30000000c [0x+5003000]		r3,[r0,#0]	Debug >rom Fileneme: 2440norom.ini
30000090 [0xela0000e]		pc,x14	Debug >rom 2003. 5. xx 1st dreft.
ENTER_SLE[Oue5920074]		r0.0x30000110 ; = #0x49000024	Debug >con
30000098 [0xe5901000]		r1,[r0,#0]	Dabug >con For 53024400
3000009c [Dxe3811840]		r1,r1,#0x400000	Debug >rom SIGAM_Little_32, 6400
300000aD [Dx+5001000]		r1,[r0,#0]	Debug >rom FCLE:101.25MHz UPLL:40MHz
300000a4 [0xe3a01010]		x1,#0x10	Debug >rom. SDRAM refresh: 64ms(8Kryole) -> 7.8us
300000a8 [Oue2511001]		x1,x1,#1	Debug >swat @vector_catch 0x00
300000ac [Oxlafffffd]		0x300000a8 : (ENTER_SLEEP + 0x14	An expression could not be parsed or evaluated in the given contest
300000bD [0xe5921060]		z1,0x30000118 ; = #0x56000080	Debug >swat @semihosting_enabled Dx00
300000b4 [0xe5910000]		r0,[r1,#0]	An expression could not be parsed or evaluated in the given context
300000b8 [0xe3800ae0]		x0,x0,#0xe0000	Debug >swat psr %IFt_SWC
300000bc [Owe5810000]		x0,[x1,#0]	Invalid expression
300000c0 [0xe59£004c]		r0,0x30000114 ; = #0x4c00000c	Debug >com swat par %17_5VC32
300000c4 [Dxe5802000]		r2,[r0,#0]	Debug >com [disable Watch-Dog neset]
300000cB [Dxeafffffe]	ь	0x300000c8 ; (ENTER_SLEEP + Dx34	Debug >smat *Dx53000000 D
WAREUP_SL[OxeS921044]	ldr	x1,0x30000110 ; = #0x56000000	Debug >com << Clock setting >>
30000040 [Oxe5910000]	ldg	r0,[r1,#0]	Debug >com [FL1 lock time setting maximum]
30000044 [Dxe3c00ae0]	bic	r0,r0,#0xe0000	Debug >swat *Dx4c000000 (0xfffccl2)+(0xfffcc0))
300000dB [0xe5810000]	att	z0,[z1,#0]	Dabug brom FCLR:HCLR:PCLR=1:2:2.
300000dc [0xe5920030]	1dr	r0,0x3000011c ; = #0x30000408	Debug >swat *0x4c000014 (0cc2)+(1cc1)+(0) Debug >swat (0c10 200 average)0 2500 - 0 020 2 20
300000e0 [Oxe3a01449]	3600	x1,#0x48000000	Debug >rom [FCLK FMS setting:101.25NHz -> 0x7f,2,2]
300000e4 [0xe2802034]	bbo (x2,x0,#0x34	Debug >swat *0x4c000004 ((0x7foc12)+(0x2cc4)+(0x2cc0))
300000e8 [0xe4903004]		r3,[r0],#4	Debug >com [UCLE FES setting:46026 -> 0x78,2,3] Debug >swat *Dx4c000008 [(0x78cc12)+(0x2cc4)+(0x3cc0)]
300000ec [0xe4813004]		z3,[z1],#4	Debug >rom << Henory setting >>
300000fD [0xe1520000]		12,10	Debug >rom [Benit6/7: 32-bit bus width]
300000f4 [0xlafffffb]		0x300000e8 ; (WAREUP_SLEEP + 0x1	Debug >swat *0x48010000 0x22010000
300000f8 [0we3a01f40]		x1,#0x100	Debug >com [Bank0-5: Access cycle: 14-clocks, others:0-clock]
300000fc [0xe2511001]		r1,r1,#1	Debug Semi Tox48000004 (0<13]+(0<11]+(7<0)+(0<6]+(0<4)+(0<2)
3000010D [Dx1afffffd]		0x30D00Dfc ; (WAREUF_SLEEP + 0x3	Debug >mat *0x40000000 [(0cc13]+(0cc11]+(7cc0)+(0cc6]+(0cc4)+(0cc2)
30000104 [Dxe59f1014]		z1,0x30000120 ; = #0x560000b8	Debug >swat *0x4801000c [(0ccl3]+(0ccl1]+(7cc8)+(0cc6]+(0cc4)+(0cc2)
30000108 [Oxe5910000]	1 ldr	x0,[z1,#0]	constitution and an and a state of the second
		•	¢ 3

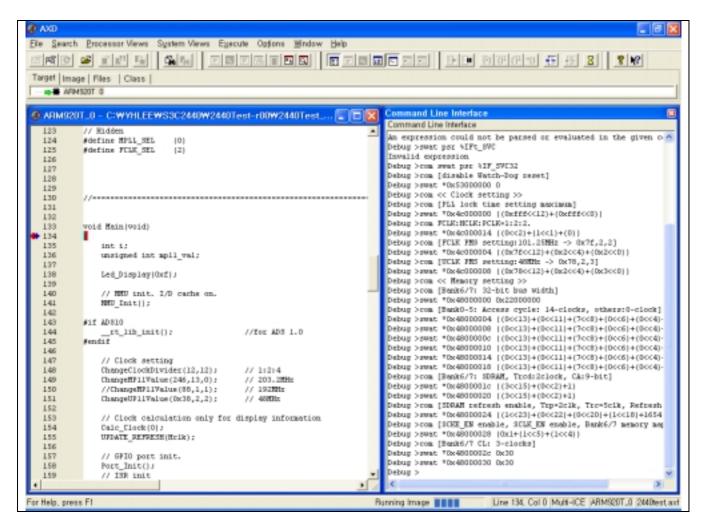


2. Load the image file (2440TEST.axf) to execute.

		n Views Egecute Options Window Help	
S 🔊 🕑 📽 🗊 🖪	6		
arget Image Files Class	1		
APPASSOT 0			
ARM920T_0 - Disasse	mbly		Command Line Interface
			Command Line Interface
	b	BesetHandler 1	An expression could not be parsed or evaluated in the given context
30000004 [0xea000052]	b	HandlerUndef	Dabug brwat pur kIFt SWC
	b	Handler5MI HandlerPahort	Invalid expression
3000000c [Dxea000062]	b		Debug >com swat par 41F SVC32
30000010 [0xea00005b]	b	HandlerDabort	Debug >com [disable Watch-Dog peset]
terrere formered	b	0x30000014	Debug >gwat *Dx53000000 D
30000018 [0xee0000047]	b	HandlerIPQ	Debug >com << Clock setting >>
3000001c [Dxem000040] 30000020 [Dxem0000000]	b	Handler FIQ Enter PHIM	Debug >com [FL1 lock time setting maximum]
	ded		Debug >swat *0x4c000000 ((htfffcc12)+(0xfffcc0))
30000024 [0x0f10ee11]	008 008	0x0f10ee11 0x0080e380	Debug >con FCLK: HCLK: FCLE-1:2:2.
30000028 [0x0080e380]		0x0f10ee01	Dabug >gwat *Dx4c000014 (0002)+(1001)+(0)
3000002c [0x0£10ee01]	ded	Oxffffffff	Dabug >com [FCLE 7ES setting:101.25Mfz -> 0x7f,2,2]
30000030 [Dxfffffff]	ded	Oxffffffff	Dabug >smat *0x4c000004 [(0x7fcc12)+(0x2cc4)+(0x2cc0)]
30000034 [0xfffffff]	des	Oxffffffff	Debug >rom [UCLE FES setting: 498Es -> 0x78,2,3]
30000038 [0xfffffff]	008 008	ONETETEEE	Debug >swat *Dx4c000008 (0x78cc12)+(0x2cc4)+(0x3cc0))
(111111111x0] 0E00000E	ded	Oxffffffff	Dabug Scon ((Memory setting S)
30000040 [0xffffffff] 30000044 [0xen000063]	des b	DepetMandler	Dabug >rom [Bank6/7: 32-bit bus width]
EnterPHDM[0xe1a02000]	D BOV	r2,r0	Debug >smat *0x40000000 0x22000000
3000004c [0xe3100008]	Lat	12,10 x0,#0	Debug >com [BankO-S: Access cycle: 14-clocks, others:0-clock]
30000042 [0x23100008] 30000050 [0x1e0000001]	bue	ENTER SLEEP	Debug >swat *Dx48000004 (0cc13)+(0cc11)+(7cc8)+(0cc6)+(0cc4)+(0cc2)
30000054 [0xe59£00b4]	ldr	r0.0x30000110 ; = #0x48000024	Dabug >zwat *Dx48000008 (0cc13]+(0cc11]+(7cc8)+(0cc6)+(0cc4)+(0cc2)
30000058 [0xe5903000]	1dr	10,003000110 ; = #00000024 13,[10,#0]	Dabug >zmat *Dx480000Dc (0cc13]+(0cc11]+[7cc8)+(0cc6]+(0cc4)+(Dcc2)
3000005c [0xe1a01003]	1df	1,13	Debug >smat *Dx40000010 (0cc13)+(0cc11)+(7cc0)+(0cc6)+(0cc4)+(0cc2)
30000060 [0xe3811840]	OXE	x1,x1,#0x400000	Debug >swat *0x48000014 (0cc13)+(0cc11)+(7cc8)+(0cc6)+(0cc4)+(0cc2)
30000064 [0xe5801000]	STE	x1,[x0,#0]	Debug >swat *0x48000018 (0cc13)+(0cc11)+(7cc8)+(0cc6)+(0cc4)+(0cc2)
30000068 [0xe3a01010]	BOV	11,[E0,#0] 11,#0x10	Dabug >rom [Bank6/7: SDGAM, Tred:2clock, CA:9-bit]
3000006c [0xe2511001]	auba	11,70010 11,11,#1	Dabug >zmat *Dx4800001c (30015 +(0002)+1)
3000007D [0x1afffffd]	bne	0x3000006c ; (EnterFWDW + 0x24)	Debug >smat *0x40000020 (3<<15 +(0<<2)+1)
30000074 [0xe59£0090]	ldr	x0,0x30000114 ; = #0x4000000c	Debug >rom [NDRAM refresh enable, Trp=2clk, Trc=5clk, Refresh(1654]
30000078 [0xe5902000]	STE	r2,[r0,#0]	Debug >swat *0x48000024 (1cc23]+(0cc22]+(0cc20]+(1cc18)+1654]
3000007c [Dxe3a01020]	BOV	11,#Dx2D	Debug >rom [SUNE_EN enable, SULK_EN enable, Bank6/7 memory map: 6403
30000030 [Dxe2511001]	auba	11,70,0010 11,11,#1	Dabug >zmat *Dx48000028 [0x1+(1<<5)+(1<<4)]
300000034 [Dx1afffffd]	bne	0x30000000 ; (EnterFMDN + 0x30)	Debug >com [Bank6/7 CL: 3-clocks]
30000088 [0xe5920080]	ldr	x0,0x30000110 ; = #0x49000024	Debug >swat *0x4800002c 0x30
3000008c [0xe5803000]	STE	x3, [x0, #0]	Debug >swat *0x48000030 0x30
30000090 [Oxela0200e]	BOW	pc,r14	Debug >
[< > > > > > > > > > > > > > > > > > > >



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.

🔤 DNW v0.50A [CO	M1,115200bps][USB:	юк]		
Serial Port USB Port (Configuration Help			
[SMDK2440 Board Te	st Program Ver 0.0]			<u>^</u>
[Fclk:Hclk/Pclk]=[[Uclk=48.0Mhz]	203.2:101.6:50.8]Mh	z		
4:nWAIT test	5:Nand test	2:PCMCIA test 6:Program Flash 10:Power/Clk test	7:DMA test	
		14:IIC Test		
16:IrDA Test	17:SD test	18:ADC test	19:ADC TS test	
20:limer test	21:11S test	22:Uart Test	23:Clkdiv_lest	
Select the functio	n to test 🙃			
[NOR Flash Memory	Writer Ver 0.1]			=
	: 0x31000000 ~ 0x3 b : 28F128J3A			
Select the type of	a flash memory ?			~



- 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.

DNW v0.50A [COM1,115200bps][USB:OK]					
Serial Port USB Port Configuration Help					
[SMDK2440 Board Test Program Ver 0.0]	^				
[Fclk:Hclk/Pclk]=[203.2:101.6:50.8]Mhz					
[Uclk=48.0Mhz]					
0:User Test 1:Manual Reg Set 2:PCMCIA test 3:Stepping stone					
4:nWAIT test 5:Nand test 6:Program Flash 7:DMA test					
8:Interrupt test 9:Cpu speed test 10:Power/Clk test 11:Lcd test					
12:Camera test 13:SPI Test 14:IIC Test 15:RTC Test					
16:IrDA Test 17:SD test 18:ADC test 19:ADC TS test 20:Timer test 21:IIS test 22:Uart Test 23:Clkdiv_Test					
20:Timer test 21:IIS test 22:Uart Test 23:Clkdiv_Test					
Select the function to test : 6					
select the function to test : o					
[NOR Flash Memory Writer Ver 0.1]					
The program buffer : 0x31000000 ~ 0x33ff0000					
a : AM29LV800BB x1 b : 28F128J3A(16MB) x2 Select the type of a flash memory ? a					
Do you want to download through UARTO from 0x31000000? [y/n]					
downloadAddress = 31000000					
Download the plain binary file(.BHC) to be written					
The file format : <n+6>(4)+(n)+CS(2)</n+6>					
To transmit .BIN file : wkocm2 xxx.BIN /1 /d:1					
Download methods : COM:8Bit,NP,1STOP					
STATUS :					
	~				

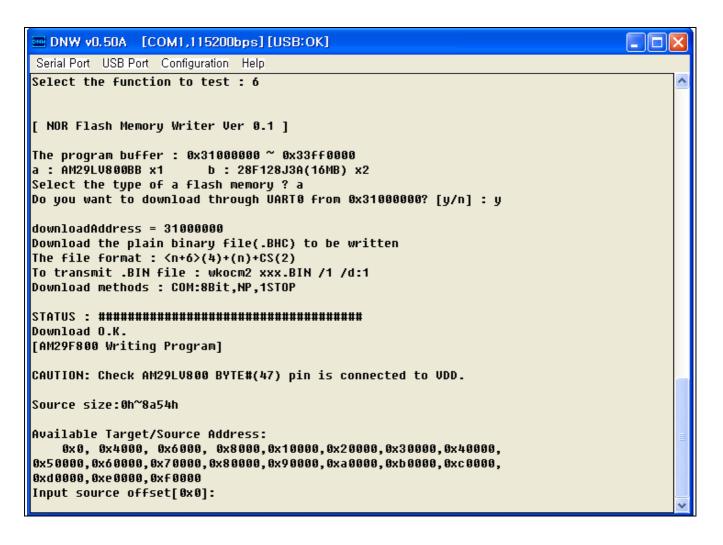


- 7. Download target files with the DNW by selecting Transmit menu from Serial Port.
- Serial Port \rightarrow Transmit

🔤 DNW v	0.50A [COM1,1	15200bps][US	B:OK]		
Serial Port	USB Port Config	uration Help			
[SMDK244	열기				? 🛛
[Fclk:Ho [Uclk=48		2440USB_Mon			
0:User 4:nWAI1	교 내 최근 문서	📾 u2440mon, bin			
8:Inter 12:Camer 16:IrDA	() 바탕 화면				
20:Timer Select (▷ 나 문서				
[NOR F]	9 내 컴퓨터				
The proc a : AM29 Select 1	🧐 내 네트워크 환경				
Do you u download		파일 이름(<u>N</u>):	u2440mon,bin	•	열기(<u>0</u>)
Download		파일 형식(<u>T</u>):	BIN Files (*,bin)*,nb0)	_	취소
To trans	+ormat : <n+6. mit .BIN file methods : COM</n+6. 	: wkocm2 xxx.B	IN /1 /d:1		
STATUS :					~

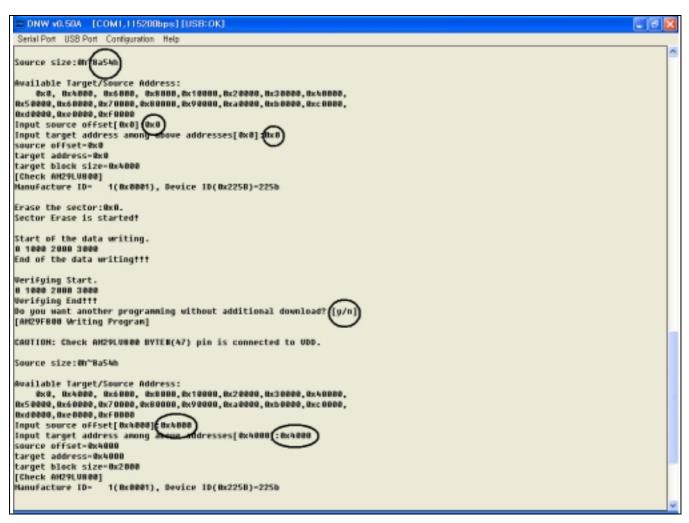
- Select and Download a target file.





- 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'.

DNW v0.50A [COM1,115200bps][US8:0K]	- 8 X
Serial Port USB Port Configuration Help	
Verifying Start.	^
0 1000 Werifying Endttt	
To you want another programming without additional download? [y/n]	
[AM29F808 Writing Program]	
CAUTION: Check AM29LU800 BYTE#(47) pin is connected to VDD.	
endrun, ener marcoube unceren pan as connected to over.	
Source size:0h"Ba54h	
Available Target/Source Address:	
exe, 0x4000, 0x5000, 0x8000,0x10000,0x20000,0x30000,0x400000,	
0x55000,0x60000,0x70000,0x80000,0x90000,0x00000,0x00000,0x00000,	
Bxd8888, Bxe8888, 8xf0888 Input source offset[Bx68888]; Bx6888	
Input target address among above addresses[@x6000] @x6000]	
source offset=@x6000	
target address=8x6000	
target block size-0x2000 [Check AN29LU800]	
Manufacture ID- 1(0x0001), Device ID(0x225B)-225b	
Erase the sector:0x6000.	
Erdse the Sector : 0.00000. Sector Erase is started?	
Start of the data writing.	
0 1000 End of the data writingfff	
che or che oord m'acangiti	
Verifying Start.	
0 1000 Verifuing Endttt	
Bo you want another programming without additional download? [y/n]	
(AN29F800 Writing Program)	
CAUTION: Check AH29LV800 BYTE#(47) pin is connected to VDD.	
and tone and a massive official and the connected to oper	
Source size:8h*BaS4h	
Available Target/Source Address:	
@x0, 0x4000, 0x6000, 0x8000,0x10000,0x20000,0x30000,0x40000,	
0x50200, 0x60000, 0x70000, 0x90000, 0x402000, 0x402000, 0x100000, 0xc00000,	
0xd0000,0xe00000,0xf0000 Input source offset[0x8000]:	
inhar source orrestlingereal.	~



- Write source offset ' 0x8000 ' and write target address ' 0x8000 ', and then type 'n'.

🔤 DNW v0.50A [COM1.115200bps][USB:OK]
Serial Port USB Port Configuration Help
End of the data writing!!!
Verifying Start.
0 1660
Verifying Endfit
Do you want another programming without additional download? [y/n]
[AM29F800 Writing Program]
CAUTION: Check AM29LU800 BYTE#(47) pin is connected to VDD.
Source size:0h~8a54h
Available Target/Source Address:
8x9, 8x4000, 0x6600, 8x8000,0x10000,0x20000,0x30000,0x40000,
0x50000,0x60000,0x70000,0x80000,0x90000,0x00000,0xb0000,0xc0000, 9xd0000,0xe0000,0xf0000
Input source offset[0x8000];0x8000]
Input target address among above addresses0x8000
source offset-0x8000
target address-0x8000
target block size-0x8000 [Check AM29LU800]
Manufacture ID= 1(8x8881), Device ID(8x2258)=225b
Erase the sector:0x8000.
Sector Erase is started!
Start of the data writing.
0 1000 2000 3000 4000 5000 6000 7000
End of the data writing?!?
Verifying Start. 0 1000 2000 3000 4000 5000 6000 7000
Verifying Enditit
Do you want another programming without additional download?{[y/n]
0:User Test 1:Manual Reg Set 2:PCNCIA test 3:Stepping stone 4:nVAIT test 5:Nand test 6:Program Flash 7:DMA test
4:nVAIT test 5:Nand test 6:Program Flash 7:DMA test 8:Interrupt test 9:Cpu speed test 10:Power/Clk test 11:Lcd test
12:Camera test 13:SPI Test 14:IIC Test 15:RTC Test
16:IrDA Test 17:SD test 18:ADC test 19:ADC TS test
20:Timer test 21:IIS test 22:Uart Test 23:Clkdiv_Test
Select the function to test :

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.

12 2 2 1 2			I I I I I I I I I I I I I I I I I I I
vget Image Files Clas	5		
D TOSEMRA #			
ARM920T_0 - Disasse		. 🗆 🖬	Command Line Interlace
30000080 [Oxe2511001]	585	x1.x1.41	Command Line Interface
30000084 [0x1afffffd]	bus	0x30000080 : (EnterFWDW + 0x38)	Debug Sobey C:\W08U(2440)2440norom\2440norom.ini
30000088 [0xe5920080]	1dr	z0,0x30D00110 ; = #0x45000024	Dabug >rom
30000000c [0xe5003000]	str	r3,[r0,#0]	Debug >rom Filename: 2440norom.ini
30000090 [0xela0000e]	BOT	pc,x14	Debug >rom 2003. 5. xx 1st dreft.
ENTER_SLE[Owe5920074]	ldr	r0.0x30000110 ; = #0x49000024	Debug >com
30000098 [0xe5901000]	ldr	r1,[r0,#0]	Dabug >com. For 53024400
3000009c [0xe3811840]	DIE	11,11,#0x400000	Debug >com SDRAM_Little_32, 6400
300000aD [0x+5001000]	str	r1,[r0,#0]	Debug >com FCLN:101.25MHz UFL1:40MHz
300000a4 [0xe3a01010]	BOU	x1,#0x10	Debug >rom SDRAM refresh: 64ms(6Kryole) -> 7.8us
300000a8 [0xe2511001]	548.5	x1,x1,#1	Debug >swat #vector_cetch 0x00
3000D0ac [Ox1afffffd]	bue	0x3010010a8 ; (ENTER_SLEEP + Dx14	An expression could not be parsed or evaluated in the given context
300000bD [0xe59£1060]	ldr	z1,0x30000118 ; = #0x56000080	Debug >rwat @remihosting_emabled Dx00
300000b4 [0xe5910000]	ldr	r0,[r1,#0]	An expression could not be parsed or evaluated in the given context
300000b8 [0xe3900ae0]	OTE	x0,x0,#0xe0000	Debug >swat psr %IFt_SWC
300000bc [0xe5810000]	STE	x0,[x1,#0]	Invalid expression
300000c0 [0xe59£004c]	ldr	r0,0x30000114 ; = #0x4c00000c	Debug >com swat psr %1F_SVC32
300000c4 [Dxe5802000]	str	z2,[z0,#0]	Debug >com [disable Watch-Bog ceset]
300000cB [Dxeafffffe]	b	0x300000c0 ; (ENTER SLEEP + 0x34	Debug >smat *0x53000000 D
MAREUP SL[OxeS9f1044]	ldr	x1,0x30000118 ; = #0x56000080	Debug >rom << Clock setting >>
30000040 [Oxe5910000]	ldr	x0,[x1,#0]	Debug >com [FL1 lock time setting maximum]
30000044 [0xe3c00ae0]	bic	r0,r0,#0xe0000	Debug >swat *Dx4c000000 (0xfffccl2)+(0xfffcc0))
300000d8 [0xe5810000]	ate	r0,[r1,#0]	Debug >com FCLR:HCLR:PCLR=1:2:2.
300000dc [0xe5920030]	1dr	x0,0x3000011c ; = #0x30000408	Debug >swat *0x4c000014 (0cc2)+(1cc1)+(0)
300000e0 [0xe3a01449]	26010	x1,#0x48000000	Debug >com [FCLK FMS setting:101.25MHz -> 0x7f,2,2]
300000e4 [0xe2802034]	add	r2,r0,#0x34	Debug >swat *0x4c000004 (0x7fcc12)+(0x2cc4)+(0x2cc0))
300000e8 [0xe4903004]	1dr	x3,[z0],#4	Debug >com [UCLE FES setting:480En -> Dx76,2,3]
300000ec [Dxe4813004]	att	z3,[z1],#4	Dabug >swat *Dx4c000008 (0x78c(12)+(0x2c(4)+(0x3c(0))) Dabug hown of Memory attring bh
300000fD [0xe1520000]	cmp	r2,r0	Debug >rom << Hemory setting >> Debug >rom (Decked 22, 22-bit how width)
300000f4 [0xlafffffb]	bne	0x300000e8 ; (WAREUP_SLEEP + 0x1	Debug >com (Bank6/7: 32-bit bus width)
300000f8 [0xe3a01f40]	BOW	x1,#0x100	Debug >swat *0x48010000 0x22010000 Debug >com [BankD-5: Access cycle: 14-clocks, others:0-clock]
300000fc [0xe2511001]	242/2	x1,x1,#1	Debug Scon [Barko-S: Access cycle: 14-clocks, Statisto-clock] Debug Samat "Dx48000004 (Scc13 +(Scc11)+(7cc8)+(Dcc5)+(Scc4)+(Dcc2)
30000100 [Dxlafffffd]	bne	0x300000fc ; (WAGEUF_SLEEP + 0x3	Debug Smat *0x40000000 [(0cc13]+(0cc11]+[7cc0)+(0cc6]+(0cc4)+(0cc2) Debug Smat *0x40000000 [(0cc13]+(0cc11]+[7cc0)+(0cc6]+(0cc4)+(0cc2)
30000104 [Dxe59f1014]	ldr	r1,0x30000120 ; = #0x560000b0	Debug >swat *0x48010010 [(0cc13]+(0cc11]+(7cc8)+(0cc6]+(0cc4)+(0cc2)]
30000108 [0xe5910000]	ldr	x0,[z1,#0]	hered stars invasioned (instral_instril_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_instal_inst
		• 4	< >>



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

AND le Search Processor Views	s System	Views Egecute Options Window Help	
sinsi o i e e e e e			
arget Image Files Class			
APMS20T 0			
ARM920T_0 - Disasser	nbly		Command Line Interlace
30000000 [Oxea000074]	b	ResetBandler :	Command Line Interface
30000004 [0xen000052]	b	RepetHandler 1 HundlerUndef 4	An expression could not be parsed or evaluated in the given context
	b	Hardler 5MI	Dabug Sawat par %IFt_SWC
	b	HandlerPahort	Invalid expression
30000010 [0xee000005b]	b	HandlerDabort	Debug >com swet par 41F_89032
	b	0x30000014	Debug >com [disable Watch-Dog reset]
	b	HandlerIPO	Debug >swat TDx53000000 D
L	b	MardlerFIQ	Debug >com << Clock setting >>
	b	EnterPHIM	Debug >com [FL1 lock time setting maximum]
30000024 [0x0f10ee11]	ded	OxOflOgell	Debug >swat *0x4c010000 (0xfffccl2)+(0xfffcc0))
30000028 [0x0030e380]	006	0x0080e380	Debug >com FCLK:HCLK:FCLK-1:2:2.
3000002c [0x0£10ee01]	ded	0x0f10ee01	Debug >gwat *Dx4c000014 (0 <c2)+(1<c1)+(0) < td=""></c2)+(1<c1)+(0) <>
3000003D [Dxfffffff]	ded	Oxfffffffff	Debug >com [FCLE FES setting:101.25MHz -> 0x7f,2,2]
30000034 [Dxfffffff]	ded	Oxffffffff	Debug >swat *0x4c000004 (0x7f<<12)+(0x2<<4)+(0x2<<0)
30000038 [0xfffffff]	ded	Oxfffffffff	Debug >rom [UCLK PHS setting:488Hs -> 0x78,2,3]
30000030 [Oxffffffff]	606	Oxfffffffff	Dabug >swat *0x4c000008 (0x78cc12)+(0x2cc4)+(0x3cc0)
30000040 [0xfffffff]	ded	OxEEEEEEE	Debug >com << Hemory setting >>
30000044 [Dxen000063]	b	RepetHandler	Debug >rom [Bank6/7: 32-bit bus width]
EnterPHIM[Docels02000]	207	12,10	Debug >swat *0x40010010 0x22010010
3000004c [0xe3100008]	tat	x0,#8	Debug >com [BenkO-S: Access cycle: 14-clocks, others:O-clock]
30000050 [0x1e0000001]	bue	ENTER SLEEP	Dabug >swat *0x48000004 (0ccl3)+(0ccl1)+(7cc8)+(0cc6)+(0cc4)+(0cc2)
30000054 [0xe59£00b4]	ldr	r0,0x30000110 ; = #0x48000024	Debug >swat *Dx48000008 (0cc13]+(0cc11]+(7cc8)+(0cc6]+(0cc4)+(0cc2)
30000056 [Dxe5903000]	1dr	r3,[r0,#0]	Dabug >smat "Dx480000Dc (0cc13]+(0cc11]+(7cc8)+(Dcc6]+(0cc4)+(Dcc2)
3000005c [0xe1a01003]	207	11,13	Debug >swat *Dx40000010 (0<<13)+(0<<11)+(7<<0)+(0<<6)+(0<<4)+(0<<2)
30000060 [0xe3811840]	OFF	x1,x1,#0x400000	Debug >swat *0x48010014 (0ccl3)+(0ccl1)+(7cc8)+(0cc6)+(0cc4)+(0cc2)
30000064 [Oxe5801000]	STE	x1,[x0,#0]	Debug >swat *0x48000018 (0cc13)+(0cc11)+(7cc8)+(0cc6)+(0cc4)+(0cc2)
30000068 [Dxe3a01010]	BOW	r1,#0x10	Debug >com [Bank6/7: SDGAM, Trod:Sclock, CA:9-bit]
3000006c [Dxe2511001]	auba	11,11,#1	Debug >smat *Dx4800001c (3<<15]+(0<<2)+1)
3000007D [Dx1afffffd]	bne	0x3000006c ; (EnterFMDN + 0x24)	Debug >smat *0x40010020 (3<<15 +(0<<2)+1)
30000074 [0xe59£0098]	1dr	x0,0x30000114 ; = #0x4c00000c	Debug >rom [3DRAM refresh enable, Trp=201k, Tro=Solk, Refresh(1654]
30000078 [Oxe5802000]	STE	r2,[r0,#0]	Debug >swat *0x48000024 (1cc23)+(0cc22)+(0cc20)+(1cc18)+1654)
3000007c [Dxe3a01020]	BOV	11,#Dx20	Debug >rom [SCHE_EN enable, SCLK_EN enable, Bank6/7 memory map: 64MB
30000080 [0xe2511001]	autor	11,11,#1	Dabug >smat *Dx48000028 [0x1+(1<<5)+(1<<4)]
30000004 [0x1affffd]	bne	0x30000000 ; (EnterFMDW + 0x30)	Debug >rom [Bank6/7 CL: 3-clocks]
30000088 [0xe5920080]	ldr	x0,0x30000110 ; = #0x49000024	Debug >swat *0x48000020 0x30
3000008c [Oxe5803000]	STE	x3,[x0,#0]	Debug >swat *0x48000030 0x30
30000090 [Oxela0200e]	BOW	pc,114 ¥	Debug >
		1	< > >



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

AND .			
		lews E⊻ecute Options ∭indow H	
Load Image Load Debug Symbols	Ge 6		
Beload Current Image			
Open File,			
Load Memory From File	bly		Command Line Interface
Save Mgmory To File,			Command Line Interface
Flash Download		haffffffff ; > undefined	Debug >swat par 4IPt SWC
		hffffffff ; ? undefined	Invalid expression
Load Session		<pre>httffffff : ? undefined htffffffff : ? undefined</pre>	Dabug >com swat pur %DF 5WC32
Saye Session		httfffffff ; / undefined	Debug >com [disable Watch-Dog reset]
Recent Elles		httfffffff 1 7 undefined	Debug >swat *0x53000000 0
Line Canal		Afffffffff : 2 undefined	Debug >com << Elock setting >>
nacent im@ges *		Resetfindler	Debug >com [PLL lock time setting maximum]
Recent Symbols +	b 8	RandlerUndef	Debug >swat *0x4c000000 (0xfffccl2)+(0xfffcc0))
Recent Sessions	ъ в	fandler3VI	Debug >com FCLK:HCLK:FCLK=1:2:2.
Unload Current Image	- · · ·	fandler7abort	Debug Somet "Gr4c000014 [(Doc2]+(1oc1)+[0)] Debug Some [FCLK PMS setting:101.25MHz -> 0x7f,2,2]
Import Formats	- · ·	EandlerDabort	Debug >swat *0x4e000004 ((0x7Ecc12)+(0x2cc4)+(0x2cc0))
import Pormais	P .	hx30000014	Debug >con [UCLK PHS setting: 45HHz -> 0x78,2,3]
Exit		fandlerIRQ	Debug >gmat *0x4c000000 (0x70<<12)+[0x2<<4)+(0x3<<0])
30000020 [0xea000008]		EandlerFIQ EnterFWDN	Debug >com << Memory setting >>
30000024 [0x0f10ee11]		wofloell	Debug >com [Bank6/7: 32-bit bus width]
30000028 [0x0000e300]		1x0000x350	Debug >swat #0x40000000 0x22000000
3000002c [0x0fl0ee01]		h0f10ee01	Debug >com [Bank0-5: Access cycle: 14-clocks, others:0-clock]
30000030 [0xfffffff]		xtttttttt	Debug >swat *0x48000004 (Dcc13)+(Dcc11)+(7cc8)+(Dcc6)+(Dcc4)+(Dcc2)+0
30000034 [0xfffffff]	ded 0	affffffff	Dabug >zwat *0x48000008 (Dcc13)+(Dcc11)+ 7cc8)+(Dcc6)+(Dcc4)+(Dcc2)+0
30000030 [0xffffffff]		INITITELEE	Debug >swat *0x4800000c (0cc13)+(0cc11)+(7cc8)+(0cc6)+(0cc4)+(0cc2)+0
30000030 [0xfffffff]		xttttttt	Debug >swat *0x48000010 (Dcc13)+(Dcc11)+(7cc8)+(Dcc4)+(Dcc4)+(Dcc2)+0 Debug >swat *0x48000014 (Dcc13)+(Dcc11)+(7cc8)+(Dcc4)+(Dcc4)+(Dcc2)+0
30000040 [0xfffffff]		1xfffffff	Debug Same *0x48000014 [(0cc13)+(0cc1)+[0cc3)+(0cc3)+(0cc4)+(0cc2)+0 Debug Samet *0x48000018 [(0cc13)+(0cc1)+[7cc8)+(0cc6)+(0cc4)+(0cc2)+0
second foregroups		kesetHandler	Debug >com [Bank6/7] #D8AH, Trodi2clock, CA19-bit]
		12,10	Debug >pwst *0x4800001c (3o(15)+(0c(2)+1)
3000004c [0xe3100008] 30000050 [0x1m00000f]		tD,#8 INTER SLEEP	Debug >swat #0x40000020 (30015)+(0002)+1)
30000054 [0xe5900064]		(0,0x30000110 ; = #0x48000024	Debug >com [SDRAM refresh enable, Trp=2clk, Trc=5clk, Befresh:1654]
30000058 [0xe5903000]		(3,[±0,#0]	Debug >swat *0x48000024 (1cc23)+(0cc22)+(0cc20)+(1cc18)+1654)
		11,13	Debug >com [SCHE_EN enable, SCLE_EN enable, Bank6/7 memory map: 6405/6-
		1, 11, #0x400000	Debug >gwat *0x48000028 [0x1+[1<<5)+[1<<4)]
30000064 [Oxe5801000]		1.00.401	Debug >com [Bank6/7 CL: 3-clocks]
30000068 [0xe3a01010]	now r		Debug >swat *0x4800002c 0x30 Debug >swat *0x48000030 0x30
3000006c [0xe2511001]		C1, E1, #1	Tachara a
30000070 [Ox1afffff6]	boe 0	hx3000006c ; (EnterPMDN + 0x24)	· · · · · · · · · · · · · · · · · · ·
•			<
Load data from disk to memory			Line 129, Col D Multi-ICE ARM920T_0 2040test auf
and all company			parte text der e prose text p artiste text

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

Load Memory I	From File		? 🗙
찾는 위치(!):	🗀 2440USB_Mon	- 🗢 主	📸 🎟 •
₪ u2440mon, bin			
파일 이름(<u>N</u>): 파일 형식(<u>T</u>):	u2440mon,bin All Files (*,*)	_	열기(<u>0</u>) 취소
Address: 0x310000	00		
Processors			
ARM920T_0			



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

AXD	laure Suctor	n Views Execute Options Window b	
ing 🖻 itra	<u> </u>		
arget Image Files C	ass		
ARM920T D			
ALL			
🕽 ARM920T_0 - Disan	nembly		Command Line Interface
30000548 [0xe3e0000	1 800	£0,#0	Command Line Interface
30000540 (0xe3a0144		r1,#0x4a000000	Debug Sawat par %IFt_SVC
30000550 [0xe581000		z0,[z1,#8]	Invalid expression
30000554 [Oce1a00aa]		r0,r0,1sr #21	Debug >com swat psc %IF_SVC32
30000550 [Oxe3a0144		r1,#0x4m000000	Debug >com [disable Watch-Dog reset]
3000055c [Oxe581001/	1 atx	z0,[z1,#0x1c]	Debug >gwat *0x53000000 0
30000560 (0xela0f00)	B07	p0, £14	Debug >com << Clock setting >>
Eain [0xe92d407] * stafd	£13!,(£4-£6,£14)	Debug >com [PL1 lock time setting maximum]
30000568 [0xe24dd02] and	r13,r13,#0x28	Debug >swat *0x4c000100 ((0xfffccl2)+(0xfffcc0))
3000056c [0xe3a00003		E0,#0xE	Debug >com FULK:HULK:PULK=1:2:2.
30000570 [Oceb00035/		Led_Display	Debug Sawat 70x4c000014 ((DoC2)+(1oC1)+(0))
30000574 (0xeb00053)		MMJ_Init	Debug >com [FCLK FHS setting:101.25HHz -> 0x7f,2,2]
30000578 [0xeb0096e]		rt_lib_init	Debug >swat *0x4c000004 ((0x7fcc12)+(0x2cc4)+(0x2cc0))
3000057c [0xe3a0100		r1,#0xc	Debug >com [UCLK FHS setting(40HHs -> 0x78,2,3] Debug >swat *0x4c000008 ((0x78cc12)+(0x2cc4)+(0x5cc0))
30000580 [0xe3a0000		z0,#0xc	Debug >com << Hemory setting >>
30000584 [Oxeb000376		ChangeClockDivider	Debug >com [Bank6/7: 32-bit hus width]
30000588 (0xe3a0200)		z2,#0	Debug >gwar *0x4000000 0x2200000
3000058c [0xe3a0100		r1,#0xd	Debug >com [BankO-Si Access cycle: 14-clocks, others:0-clock]
30000590 [Oxe3a00020		£0,#0x£6	Debug >swat *0x48000004 ((Doc13)+(Doc11)+(7oc8)+(Doc6)+(Doc4)+(Doc2)+
30000594 [0xeb00036		ChangeM911Value	Debug >prest *0x48000008 ((Doc13)+(Doc11)+(7008)+(Doc6)+(Doc4)+(Doc2)+
30000598 [Oxe3a02003		12,62	Debug >mmt "0x4000000c ((Dcc13)+(Dcc11)+(7cc0)+(Dcc6)+(Dcc4)+(Dcc2)+
30000590 (Oxe3a0100)		r1,#2	Debug >gwat *0x40000010 ((Doc13)+(Doc11)+(7cc0)+(Doc6)+(Doc4)+(Doc2)+
300005a0 [0xe3a0003		£0,#0x38	Debug >swat *0x48000014 ((Doc13)+(Doc11)+(7oc8)+(Doc6)+(Doc4)+(Doc2)+
300005e4 [Oxeb000396		ChangeUP11Value z0,#0	Debug >pwst *0x48000018 ((Doc13)+(Doc11)+(7oc8)+(Doc6)+(Doc4)+(Doc2)+
300005a0 [Oxe3a0000] 300005ac [Oxeb001ef]		Calc Clock	Debug >com [Bank6/7: SDRAM, Trod:2clock, CA:9-bit]
300005b0 (0xe590026)		<pre>care_clock g0,0x30000820 J = #0x30067090</pre>	Debug >swat *0x4000001c [(3cc15)+(0cc2)+1]
300005b4 (0xe590000)		E0,[E0,#0]	Debug >swat *0x40000020 ((3cc15)+(0cc2)+1)
300005b8 [0xeb00af80		filta	Debug >com [8DRAM refresh enable, Trp=201k, Tro=Solk, Refresh:1654]
300005bc [0xe1a0500		15,10	Debug >swat *0x48000024 [(loc23)+(Doc22)+(Doc20)+(loc18)+1654]
300005c0 [Oxeb00adf		£26	Debug >com [SCHE_EN enable, SCHE_EN enable, Bank6/7 memory map: 6405/
300005c4 [0xe58d0010		E0,[E13,#0x10]	Debug >zwat *0x40000028 (0x1+(1<<5)+(1<<4))
300005c8 [0xe58d101		r1,[r13,#0x14]	Debug >com [Bank6/7 CL: 3-clocks]
300005cc [0xe2860694		r0,pc,#0x250 : #0x30000624	Debug >swat *0x4010012c 0x30
300005d0 [0xe090000		r0,(r2,r3)	Debug >pwat *0x4000030 0x30
30000564 [Oxe59d0010		r0,[r13,#0x10]	Debug >
			<
			Running Image Karl (No Pos) Multi-ICE (ARM320T_D (2440est, ad



- 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.

)M1,115200bps][USB	:0К]		
Serial Port USB Port	Configuration Help			
[rstatus3=0]				^
[SMDK2440 Board Te	st Program Ver 0.0]			
[Fclk:Hclk/Pclk]=[[Uclk=48.0Mhz]	203.2:101.6:50.8]Mh	z		
4:nWAIT test	5:Nand test	2:PCMCIA test 6:Program Flash	7:DMA test	
		10:Power/Clk test		
		14:IIC Test		
20:Timer test	21:IIS test	18:ADC test 22:Uart Test	23:Clkdiv Test	
Select the functio	n to test : 6		_	
[NOR Flash Memory	Writer Ver 0.1]			
(a :) AM29LV800BB x1	: 0x31000000 ~ 0x3 b : 28F128J3A	(16MB) x2		
	a flash memory ? a mload through UARTO) from 0x31000000? [y/n] :	~

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.

	OM1,115200bps][US	B:OK]			
Serial Port USB Port	Configuration Help				
		14:IIC Test		^	
			19:ADC TS test		
20:limer test	21:IIS test	22:Uart Test	23:Clkdiv_Test		
Select the function to test : 6 [NOR Flash Memory Writer Ver 0.1] The program buffer : 0x31000000 ~ 0x33ff0000 a : AM29LV800BB x1 b : 28F128J3A(16MB) x2 Select the type of a flash memory ? a Do you want to download through UART0 from 0x31000000? [y/n] : n					
[AM29F800 Writing Program] CAUTION: Check AM29LV800 BYTE#(47) pin is connected to VDD.					
Source size:0h~0h					
	3x6000, 0x8000,0x1 x70000,0x80000,0x9 xf0000	0000,0x20000,0x300 0000,0xa0000,0xb00			
Inpac Source offse				~	



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.

DNW v0.50A [COM1,115200bgm][US8:OK]	X
Serial Port USB Part Configuration Help	
Verifying Start. 8 1800 Verifying End!!! Bo you want another programming without additional download? [y/n]] [AH29F800 Writing Program]	^
CAUTION: Check AM29LV800 BYTE#(47) pin is connected to VDD.	
Source size:0n~0h	
Available Target/Source Address: BxD, dx4000, 0x50000, 0x70000, 0x30000, 0x20000, 0x20000, 0x60000, 0x60000, 0x60000, 0x70000, 0x7000, 0x70000, 0x7000, 0x700, 0x7000, 0x7000, 0x7000, 0x7000, 0x7000, 0x7000, 0x7000, 0x700, 0x700	
Verifying Enditt Do you want another programming without additional download? [u/n]	
8:User Test 1:Hanual Reg Set 2:PCHCIA test 3:Stepping stone A:nHAIT test 5:Hand test 6:Program Flash 7:DHA test 8:Interrupt test 9:Cpu speed test 10:Power/Clk test 11:Lod test 12:Camera test 13:SPI Test 14:IIC Test 15:RTC Test 16:IrbA Test 17:SD test 18:ADC test 10:ADC TS test 20:Timer test 21:IIS test 22:Uart Test 23:Clkdiv_Test Select the function to test :	

9. Turn off and on the SMDK2440.



WRITING IMAGE FILES TO AMD FLASH MEMORY WITH REALVIEW ICE(RVI)

To use RealView Debugger to control Flash memory on your chosen debug target, you must:

- Configure your debug target to describe the Flash memory chip
- Have access to an appropriate Flash MEthod (FME) file.

Depending on your current target, this might mean that you must first define the memory map to specify the Flash memory.

Flash definition files

Files to enable you to use supported Flash devices are included in the root installation and are located in c:\work\2440\Flash directory. Files are collected in subdirectories based on the target Flash device:

Board-specific files

The board_amd_smdk2440.ame file was contain the ASCII format information for an FME file.

These files include Flash memory programming files.

Flash-specific files

These programming files start with flash_amd.ame.

These files contain the algorithm for defining the Flash device and are used to create the FME file for your project.

To see how these files are used:

- 1. Start up RealView Debugger without connecting to a target.
- 2. Select Project -> Open Project... to open the project c:\work\2440\flash\flash_amd_smdk2440.prj
- 3. Select Project -> Project Properties... to display the Project Properties window.
- 4. Left-click on "*ASSEMBLE=arm" and "*COMPILE=arm" in the List of Entries and , the left pane. This group is expanded and the contents are displayed in the Settings Values pane, the right pane.
- 5. Right-click on *Sources and select Explore from the context menu.

This shows the programming file used to create the FME file for the project.

- 6. Left-click on *BUILD in the List of Entries. This group is expanded and the contents are displayed in the Settings Values pane.
- 7. Right-click on *Pre_Post_Link and select Explore from the context menu.

This shows the link commands used to include the Flash definition files for the project.

- 8. Select File -> Close Window to close the Project Properties window without making any changes.
- 9. Select Tools -> Build to create the FME file as defined by the project, that is flash_amd_smdk2440.fme.

Flash Method files

FME files include code to:

- Enable you to write to the Flash on your debug target
- Perform read, write, and erase operations
- Describe the way the Flash is configured on the bus.

Example files are included for all supported Flash devices as part of the root installation.

Defining your target

To configure the Flash target:

- 1. Start up RealView Debugger without connecting to a target.
- 2. Select File ->Connection -> Connect to Target... to display the Connection Control window.
- 3. Right-click on the entry RealView-ICE and select Connection Properties... from the context menu.

This displays the Connection Properties window where you can view configuration settings stored in your board file.

- 4. Click on the entry CONNECTION=RealView ICE, in the left pane, to display the settings values in the right pane.
- 5. Edit Connection Properties

Select the Advanced_Information>Default>Memory_block

And, Create the AMD-Flash, SDRAM, SFR(Special Function Register) memory area for debugging.

- Start
 - 1st Address of Flash (0x0)
- Length
 - Length of Flash Area (0x100000)
- Access
 - Flash
- Flash type
 - FME file path (c:\work\2440\flash\flash_amd_smdk2440.fme)

Connection Properties			
<u>File V</u> iew <u>H</u> elp			
Description: Define an external/ASI	C memory region		
<pre>?rvbroker.brd ?rvbroker.brd ?rvbroker.brd ?connectION=ARMOAK_MICI ?connectION=TKL_MICE ?connectION=RealView I ?connect_with Remote ?Advanced_Information ?Advanced_Information ?Advanced_Information ?connect_with ?connect_w</pre>	Name Attributes Register_Pos_Len KStart KLength Type Access Wait states Flash type Flash type KN Description Notatile	Value 0x0000 0x100000 default Flash 0 "C:\work\2440\Flash\flash_amd_smdk2440.fr "From ASIC/Board"	ne"
ten sFR ten Map rule ▼			
For more information, select Help from Menu	I	ī	



6. Change your top of memory variable for your target. Because, this flash program using this setting value for stack. So,. You must change this value.

Connection Properties			- D ×	
<u>File View H</u> elp				
Description: Configuration for ARM	1 processor use			
<pre>*Memory_block * *Mmory_block * *Amdo-Flash * *SDRAM * * *SDRAM * * *SFR * Map_rule * Register_enum * Register_enum * Concat_Register * Concat_Register * Register_Window * * ARM config * Logic_Analyzer * Cross_trigger * RTOS * *CONNECTION=VPB926EJ-S *</pre>	Name Stack_Heap Vectors Semihosting Armulator Top memory Vector catch	Ox30800000 ♥ True		
For more information, select Help from Men	u	1	NUM ///	

- 6. Select File -> Save and Close to close the Connection Properties window.
- 7. Connect to the target using the Connection Control window.
- 8. Select View -> Pane Views -> Memory Map to display the Map tab in the Process Control window, where you can see the Flash memory on the SMDK2440.
- 9. Starting the RealView Debugger and connect RealView ICE with SMDK2440.

And, execute "2440norom-rv.inc" file by "Include Commands from File..." of "Debug" menu.



KVDEBUS - BARMSZOT_BARM-ARM-ANN Bin Edit Figd Yew Broket Tools Debug Help	1.00
Die Die Lith Daw Golder Tree Band Bich.	
□ \$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$	K State Chroned
	Caller Institute
to source for contexts diminiship Click to load Image to Target	a 255544 *_1 P3 00003860 P1 00038600
	R2 00000000 R3 0000000
	B4 00000101 B5 30024£54
	B6 3006533C B7 30024E30
	R8 30055483 R9 3002482C R10 30055429 R11 30025020
	R12 30065459 37 00000000
	18. 00000000 71 00000060
	CP1R 000001B3
	NDCV FIG 180 STATE BODE
	HUSR
	M IFQ
	M FIQ M SVC
	B AFT
	E UXD
	whether the state of the state
	+ Core (CP15 (Cache Operations (TLB Operatio)
B Hane Value B Hane Value	E (Boldda)
Christenne Location	CableAdD - CableAdD - CableAdD
	<boadda></boadda>
	Cibledite
	CabbleD cabbleD
	Cibledite
	(Boldda) (Boldda) (Boldda) (Boldda)
	(Boldde) (Boldde) (Boldde) (Boldde)
	(Bolda) (Bolda) (Bolda) (Bolda) (Bolda) (Bolda) (Bolda)
	(Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz)
	(Boldfie) (Boldfie) (Boldfie) (Boldfie) (Boldfie) (Boldfie) (Boldfie) (Boldfie) (Boldfie) (Boldfie)
	(Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz)
K Cold Stance & Locales & Statice & This /	(Bolda) (Bolda) (Bolda) (Bolda) (Bolda) (Bolda) (Bolda) (Bolda) (Bolda)
	(Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz)
<pre>M Henory Controller Setupinc's SETMEN /32 0x48000000-bu22000000 inc/ SETMEN /32 0x48000004+([Doc13]+[Doc1]+]7003]+[Doc3]+[Doc4]+(Doc2]+0)</pre>	(Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz)
<pre>Bemory Controller Setupinc> SETMEN /32 0x48000000+bu22000800 inco SETMEN /32 0x48000004+(10cL3)+10cd1+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(0cd)+(</pre>	(Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz)
<pre>B Remory Controller Setupinch SETMEN /32 0x4800000=hu22000800 incb SETMEN /32 0x48000004+([0ccl]+ Dccl]+ Tcc3]+[0cc6]+[0cc4]+(0cc2]+0) incb SETMEN /32 0x48000004+([0ccl]+ Tcc3]+[0cc6]+[0cc4]+(0cc2]+0) incb SETMEN /32 0x48000004+([0ccl]+ Tcc3]+[0cc6]+[0cc4]+(0cc2]+0) incb SETMEN /32 0x48000004+([0ccl]+ Tcc3]+[0cc6]+[0cc4]+(0cc2]+0) incb SETMEN /32 0x48000004+([0cc1]+ Tcc3]+[0cc6]+[0cc4]+(0cc2]+0) incb SETMEN /32 0x48000004+([0cc1]+[0cc1]+[Tcc3]+[0cc6]+[0cc4]+(0cc2]+0) incb SETMEN /32 0x48000004+([0cc1]]+[Tcc3]+[0cc6]+[0cc4]+(0cc2]+0) incb SETMEN /32 0x4800004+([0cc1]]+[Tcc3]+[0cc1]+[Tcc3]+[0cc4]+(0cc2]+0) incb SETMEN /32 0x480004+([0cc1]]+[Tcc3]+[0cc1]+[Tcc3]+[0cc4]+([0cc2]+0) incb SETMEN /32 0x480004+([0cc1]]+[Tcc3]+[0cc1]+[Tcc3]+[0cc4]+([0cc2]+0) incb SETMEN /32 0x480004+([0cc1]]+[Tcc3]+[Tcc3]+[Dcc4]+[[0cc2]+0) incb SETMEN /32 0x480004+([0cc1]]+[Tcc3]+[Tcc3]+[Dcc4]+[[0cc2]+0) incb SETMEN /32 0x480004+([0cc1]]+[Tcc3]+[Tcc3]+[Dcc4]+[[0cc2]+0) incb SETMEN /32 0x480004+([0cc1]]+[Tcc3]+[Tcc3]+[Dcc4]+[[0cc2]+0) incb SETMEN /32 0x480004+([0cc1]]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc3]+[Tcc</pre>	(Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz)
<pre>B Remory Controller Setupinch SETMEM /32 0x48800000+8x22003880 incb SETMEM /32 0x48800004+([0cL1]+[0cL1]+17cd]+[0cc6]+[0cc4]+([0cc2]+0) incb NETMEM /32 0x488800004+([0cL1]+[0cc1]+17cd]+[0cc6]+[0cc4]+([0cc2]+0) incb SETMEM /32 0x488800004+([0cL1]+[0cc1]+17cd]+[0cc6]+[0cc4]+([0cc2]+0) incb SETMEM /32 0x48880010+([0cL1]+[0cc1]+17cd]+[0cc6]+[0cc4]+([0cc2]+0) incb SETMEM /32 0x48880010+([0cL1]+[0cc1]+17cd]+[0cc6]+[0cc4]+([0cc2]+0) incb SETMEM /32 0x48880010+([0cL1]+[0cc1]+17cd]+[0cc6]+[0cc4]+([0cc2]+0)</pre>	(Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz)
Newsy Controller Setupinch SETMEM /32 0x48000000=bx22000000 inch SETMEM /32 0x48000004+([0c(L])+[Dc(L])+[7c0])+[0c0]+[0c0]+[0c2]+0) inch SETMEM /32 0x48000004+([0c(L])+[0c1])+[7c0])+[0c0]+[0c0]+[0c0]+[0c2]+0) inch SETMEM /32 0x48000004+([0c(L])+[7c0])+[0c0]+[0c0]+[0c0]+[0c2]+0) inch SETMEM /32 0x48000004+([0c(L])+[7c0])+[0c0]+[0c0]+[0c0]+[0c2]+0) inch SETMEM /32 0x48000004+([0c(L])+[7c0])+[7c0]+[0c0]+[0c0]+[0c2]+0)	(Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz)
Remory Controller Setupincy SETMEN /32 0x48800000+8x22003800 inco SETMEN /32 0x48800004+([0cL1]+[0cL1]+[7cd]+[0cc6]+[0cc4]+(0cc2]+0) inco SETMEN /32 0x48800004+([0cL1]+[0cc1]+[7cd]+[0cc6]+(0cc4]+(0cc2]+0) inco SETMEN /32 0x48800004+([0cL1]+[7cd]+[0cc6]+[0cc6]+(0cc4]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cL1]+[7cd]+[0cc6]+[0cc6]+(0cc4]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cL1]+[7cd]+[0cc6]+[0cc6]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cc1]+[7cd]+[0cc6]+[0cc6]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cc1]+[7cd]+[0cc6]+(0cc6]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cc1]+[0cc1]+]7cd]+[0cc6]+(0cc4]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cc1]+[0cc1]+[0cc1]+]7cd]+[0cc6]+[0cc4]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cc1]+[0cc1]+[0cc1]+]7cd]+[0cc6]+[0cc4]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cc1]+[0cc1]+[0cc1]+]7cd]+[0cc6]+[0cc4]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cc1]+[0cc1]+[0cc4]+[0cc4]+[0cc4]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cc1]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0c	(Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz)
<pre>B Remory Controller Setupinc> SETMEN /32 0x4800000=bs22003000 inco SETMEN /32 0x48000004=(loc(3)+[boc1]+]7c03]+[boc3]+[boc4]+(boc2]+0) inco SETMEN /32 0x48000008=(loc(3)+[boc1]+]7c03]+[boc6]+[boc4]+(boc2]+0) inco SETMEN /32 0x48000008=(loc(3)+[boc1]+]7c03]+[boc6]+[boc4]+(boc2]+0) inco SETMEN /32 0x48000010=(loc(3)+[boc1]+]7c03]+[boc6]+[boc4]+(boc2]+0) inco SETMEN /32 0x48000104=(loc(3)+[boc1]+]7c03]+[boc6]+[boc4]+(boc2]+0) inco SETMEN /32 0x48000104=(loc(3)+[boc1]+]7c03]+[boc6]+[boc4]+(boc2]+0) inco SETMEN /32 0x48000104=(loc(3)+[boc1]+]7c03]+[boc6]+[boc6]+(boc2]+0) inco SETMEN /32 0x48000104=(loc(3)+[boc1]+]7c03]+[boc6]+[boc6]+(boc2]+0) inco SETMEN /32 0x48000104=(loc(3)+[boc1]+]7c03]+[boc6]+(boc6]+(boc2]+0) inco SETMEN /32 0x48000020=(loc(3)+[boc2]+1) inco SETMEN /32 0x48000020=(loc(3)+[boc2]+1) inco SETMEN /32 0x48000204=(loc(3)+[boc2]+1) inco SETMEN /32 0x48000204=(loc(3)+[boc2]+1) inco SETMEN /32 0x48000204=(loc(3)+[boc2]+1)</pre>	(Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz)
Remory Controller Setupincy SETMEN /32 0x48800000+8x22003800 inco SETMEN /32 0x48800004+([0cL1]+[0cL1]+[7cd]+[0cc6]+[0cc4]+(0cc2]+0) inco SETMEN /32 0x48800004+([0cL1]+[0cc1]+[7cd]+[0cc6]+(0cc4]+(0cc2]+0) inco SETMEN /32 0x48800004+([0cL1]+[7cd]+[0cc6]+[0cc6]+(0cc4]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cL1]+[7cd]+[0cc6]+[0cc6]+(0cc4]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cL1]+[7cd]+[0cc6]+[0cc6]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cc1]+[7cd]+[0cc6]+[0cc6]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cc1]+[7cd]+[0cc6]+(0cc6]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cc1]+[0cc1]+]7cd]+[0cc6]+(0cc4]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cc1]+[0cc1]+[0cc1]+]7cd]+[0cc6]+[0cc4]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cc1]+[0cc1]+[0cc1]+]7cd]+[0cc6]+[0cc4]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cc1]+[0cc1]+[0cc1]+]7cd]+[0cc6]+[0cc4]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cc1]+[0cc1]+[0cc4]+[0cc4]+[0cc4]+(0cc2]+0) inco SETMEN /32 0x48800014+([0cc1]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0cc4]+[0c	(Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz)
<pre> Benery Controller Setupine> SETMEM /32 0x48800000=hs22003800 inco SETMEM /32 0x488000004=(loci3)+[loci1)+[7c63]+[loc6]+[loc6]+[loc4]+(loc2]+0) inco SETMEM /32 0x48800004=(loci3)+[loc11)+[7c6]+[loc6]+[loc4]+(loc2]+0) inco SETMEM /32 0x48800010+(loc13)+[loc11)+[7c6]+[loc6]+[loc4]+(loc2]+0) inco SETMEM /32 0x48800010+(loc13)+[loc11)+[7c6]+[loc6]+[loc4]+(loc2]+0) inco SETMEM /32 0x48800014+(loc13)+[loc11)+[7c6]+[loc6]+[loc4]+(loc2]+0) inco SETMEM /32 0x48800014+(loc13)+[loc11)+[7c6]+[loc6]+[loc6]+(loc2]+0) inco SETMEM /32 0x48800014+(loc13)+[loc11)+[7c6]+[loc6]+[loc6]+(loc2]+0) inco SETMEM /32 0x48800014+(loc13)+[loc11)+[7c6]+[loc6]+(loc6]+(loc2]+0) inco SETMEM /32 0x48800014+(loc13)+[loc1]+] inco SETMEM /32 0x48800014+(loc3)+[loc2]+1) inco SETMEM /32 0x48800024+(loc3)+[loc2]+1) inco SETMEM /32 0x48800024+(loc3)+[loc2]+(loc2]+1) inco SETMEM /32 0x48800024+(loc3)+[loc2]+(loc3)+[loc3]+[loc2]+1] inco SETMEM /32 0x48800024+s20 inco SETMEM /32 0x48800024+s</pre>	(Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz)
<pre>Hencry Controller Setupino' SETMEN /S2 Ox4800000=bs22003000 inco SETMEN /S2 Ox48000004=([DocL3]+[DocL1]+[?oc8]+[Doc6]+[Doc4]+(Doc2]+0) inco SETMEN /S2 Ox48000004=([DocL3]+[DocL1]+[?oc8]+[Doc6]+[Doc4]+(Doc2]+0) inco SETMEN /S2 Ox48000010=([DocL3]+[DocL1]+[?oc8]+[Doc6]+[Doc4]+(Doc2]+0) inco SETMEN /S2 Ox48000014=([DocL3]+[DocL1]+[?oc8]+[Doc6]+[Doc4]+(Doc2]+0) inco SETMEN /S2 Ox48000014=([DocL3]+[DocL1]+[?oc8]+[Doc6]+(Doc4]+(Doc2]+0) inco SETMEN /S2 Ox48000014=([DocL3]+[DocL1]+[?oc8]+(Doc6]+(Doc4]+(Doc2]+0) inco SETMEN /S2 Ox48000014=([DocL3]+[DocL1]+[?oc8]+(Doc6]+(Doc4]+(Doc2]+0) inco SETMEN /S2 Ox48000014=([DocL3]+[DocL1]+[?oc8]+(Doc6]+(Doc4]+(Doc2]+0) inco SETMEN /S2 Ox48000014=([DocL3]+[DocL1]+]?oc8]+(Doc6]+(Doc6]+(Doc2]+0) inco SETMEN /S2 Ox480000014=([DocL3]+[DocL3]+1] inco SETMEN /S2 Ox480000014=([DocL3]+[DocL3]+1] inco SETMEN /S2 Ox480000014=([DocL3]+[DocL3]+(DocL3]+(Doc18)+(Doc6]+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18</pre>	(Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz)
<pre> Remory Controller Setupinch SETMEM /32 0x4800000=bs22000000 inch SETMEM /32 0x48000004+([0cc1]+ Dcc1]+ Tcc3]+ Dcc3]+[0cc4]+([0cc2]+0) inch SETMEM /32 0x48000004+([0cc1]+ Dcc1]+ Tcc3]+ Dcc3]+[0cc4]+([0cc2]+0) inch SETMEM /32 0x48000014+([0cc1]+ Tcc3]+ Dcc3]+[Dcc4]+([0cc2]+0) inch SETMEM /32 0x48000014+([0cc1]+ Tcc3]+ Dcc3]+[Dcc4]+([0cc2]+0) inch SETMEM /32 0x48000014+([0cc1]+ Tcc3]+ Dcc3]+[Dcc4]+([Dcc2]+0) inch SETMEM /32 0x48000014+([Dcc1]+ Tcc3]+[Dcc3]+([Dcc3]+([Dcc4]+([Dcc2]+0) inch SETMEM /32 0x48000014+([Dcc1]+ Tcc3]+([Dcc3]+([Dcc3]+([Dcc4]+([Dcc2]+0) inch SETMEM /32 0x48000020+([Scc15]+([Dcc2]+1) inch SETMEM /32 0x48000020+ls20 inch SETMEM /32 0x48</pre>	(Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldfr) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf) (Boldf
<pre>Hencry Controller Setupino' SETMEN /S2 Ox4800000=bs22003000 inco SETMEN /S2 Ox48000004=([DocL3]+[DocL1]+[?oc8]+[Doc6]+[Doc4]+(Doc2]+0) inco SETMEN /S2 Ox48000004=([DocL3]+[DocL1]+[?oc8]+[Doc6]+[Doc4]+(Doc2]+0) inco SETMEN /S2 Ox48000010=([DocL3]+[DocL1]+[?oc8]+[Doc6]+[Doc4]+(Doc2]+0) inco SETMEN /S2 Ox48000014=([DocL3]+[DocL1]+[?oc8]+[Doc6]+[Doc4]+(Doc2]+0) inco SETMEN /S2 Ox48000014=([DocL3]+[DocL1]+[?oc8]+[Doc6]+(Doc4]+(Doc2]+0) inco SETMEN /S2 Ox48000014=([DocL3]+[DocL1]+[?oc8]+(Doc6]+(Doc4]+(Doc2]+0) inco SETMEN /S2 Ox48000014=([DocL3]+[DocL1]+[?oc8]+(Doc6]+(Doc4]+(Doc2]+0) inco SETMEN /S2 Ox48000014=([DocL3]+[DocL1]+[?oc8]+(Doc6]+(Doc4]+(Doc2]+0) inco SETMEN /S2 Ox48000014=([DocL3]+[DocL1]+]?oc8]+(Doc6]+(Doc6]+(Doc2]+0) inco SETMEN /S2 Ox480000014=([DocL3]+[DocL3]+1] inco SETMEN /S2 Ox480000014=([DocL3]+[DocL3]+1] inco SETMEN /S2 Ox480000014=([DocL3]+[DocL3]+(DocL3]+(Doc18)+(Doc6]+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18)+(Doc18</pre>	(Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz) (Bolditz)



10. Select the "Upload/Download Memory File..." of "Debug -> Memory/Register Operations" menu.

KNDERUG - QARMIZOT_GARM-ARM-N	w	
The Edit Find View Project Tools	Debug Hillp Execution Costrol Single Breatpoints Complex Breatpoints	
Click to Lond Image to Tanget	Memory/Register Operations Induste Conneards From File Set Source Search Path	Set Menary R0 00000000 R1 00000000 Pritch Asserbly R2 00000000 R3 00000000 Set Register R4 00000101 R5 30024C54 Upload(Deenload Penary file R6 300653005 R7 30024E30
	Add/Edit Debugger Macros	Fill Plenory with Pattern Pill 30065403 Pill 3002472C Plash Menary Control Pill 30065429 Pill 30025000 Plash Menary Control Pill 30055459 SP 30000000 Plash Menary Control Pill 30000000 Pill 30000000 Plash Menary Control Pill 30000000 Pill 30000000 Plash Menary Control Pill 30000000 Pill 30000000
		8007 F10 100 STATE H00E 0000 D13 D13 D13 AFM SVC H U32 H U32
		E F10 E 27C E ATT E 000
		x X X X X X X X X X X X X X X X X X X X

You can see the Upload/Download file from/to Memory window.

Now, we must change the options for flash writing.

Upload/Download file from/to Memory
Load File into Memory Save Memory into File
C Verify Memory and File
File: C:\work\2440\u2440mon-rev02-20040126\u2440mon.bir
Type of File:
OBJ_Object File (COFE_ELE_etc)
raw - Raw data one word per word of memory
rawhw - Raw data one halfword per halfword of memory rawb - Raw data one byte per byte of memory
Location: 0x0
(Location is start address for read/verify)
Apply Close Help

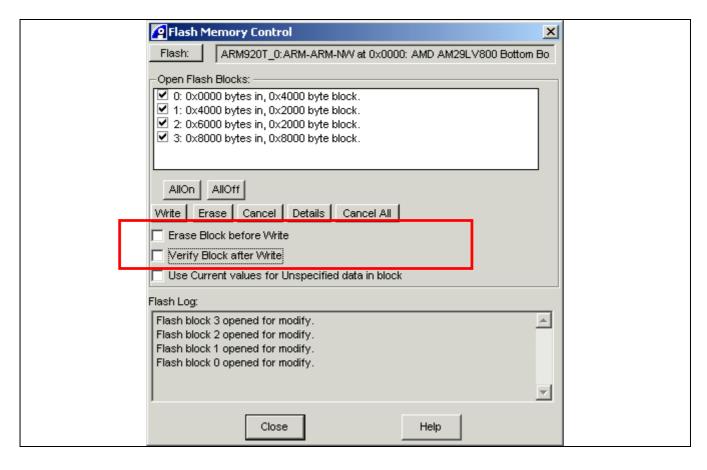
Specify the operation and set up the controls, as follows:

- Select the Load File into Memory radio button. This instructs RealView Debugger to access the specified memory block, write the contents to flash memory.
- In the File text box, enter the full pathname of the file to use to read/write memory values.
- In the Type of File section of the dialog, select the data type to be used in the specified file where:
 - OBJ specifies an object file in the standard executable target format, for example ARM-ELF for ARM-based targets



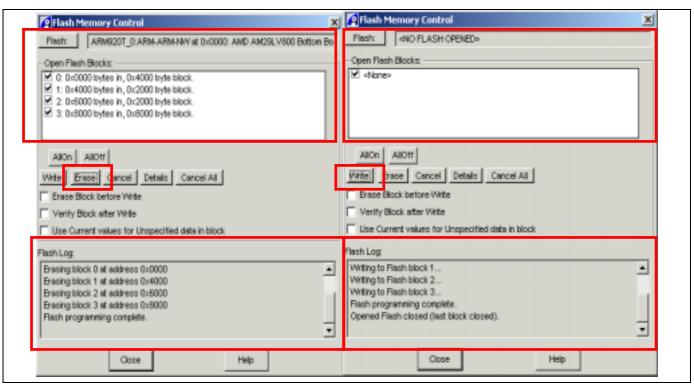
- raw specifies a data file as a stream of 32-bit values
- rawhw specifies a data file as a stream of 16-bit values
- rawb specifies a data file as a stream of 8-bit values
- ascii specifies a space-separated file of hexadecimal values.
- Define the start location of the memory block.
- 11. Click Apply to create and write the specified file.

You can see the following figure after changed option. If you didn't change that you must get some problems during write the image on your AMD flash.



12. Before writing the image on your flash, you must erase the flash.

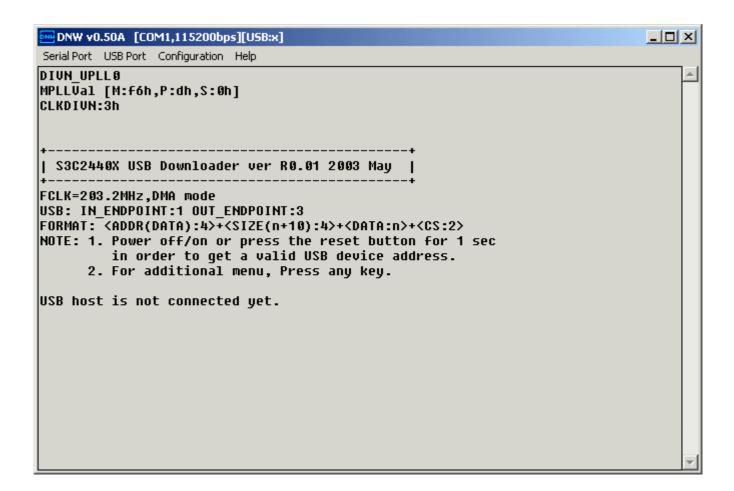




Now, your flash was completed writing. Please, reset the your target.

You can see the following figure after finished the flash writing function.







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.

- 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.

Flash UP		
Ella Utility Flash Help		
🔆 🤤 🚄 🖬	41	
		100
		-
X		1
		Connect

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.

Devid	e SMDK2440	-	Add device			
No.	Registername	Address	Value	Size	Attr	^
1	WTCON	53000000	00000000	4	R/W	
2	INTMSK	4A000008	FFFFFFFF	4	RW	
3	INTSUBMSK	4A00001C	000007FF	4	R/W	
4	LOCKTIME	4C000000	00FFFFFF	4	RW	
5	MPLLCON	4C000004	0005C042	4	R/W	
6	BWSCON	48000000	22111120	4	RW	
7	BANKCOND	48000004	00000700	4	R/W	
8	BANKCON1	48000008	00000700	4	R/W	
9	BANKCON2	480000DC	00000700	4	RW	
10	BANKCON3	48000010	00000700	4	R/W	
11	BANKCON4	48000014	00000700	4	RW	-
12	BANKCON5	48000018	00000700	4	R/W	
13	BANKCON6	4800001C	00018005	4	RW	
14	BANKCON7	48000020	00018005	4	R/W	
15	DEEDECH	48000024	00850450	A	DM/	Y
		Save	Clo	se		

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

Config target 🛛 🔀			
Device : SMDK2440 💽 Set SMU			
RAM information (Hex) Endian Base Addr 30000000 Size 03FFFFFF C Big C Little			
File 1 File 2 File 3	File 4(Le	ength(4byte)+File+CheckSum)	
Data widthEraseFlash address(Hex)© 8bit© ChipBase00000000© 16bit© SectorTarget00000000© 32bit© NoneErase size00000000		Flash address(Hex) Base 00000000 Target 00000000	
Target Image File: C:\WORK\2440\2440US	B_Mon\u2	440mon.bin	
OK Cancel			

- Device: SMDK2440
- Set SMU: Checked

—	RAM Information:	Base Address:30000000	Size: 3FFFFFF
—	Endian: Little		
_	File 1 page Download: checked Flash Device Name: A Erase:Chip Data Bus width: 16bit Flash Address:	M29LV800BB Base Address: 0	Target Address:0
	Target Image File: u24	40mon.bin	



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

5 L 10	
Flash UP	8
Ele Utility Fjash About	
🗿 🗟 🖋 🦧 \ominus 🛃	
*****	<u>×</u>
Current config	
Device SIDK2440	
RAW hase address : 0x30000000	
RAM size : 0x03FFFFFF	
Byte sex : Little endian	
Filel config	
Flash name : Am29LV80008 Erase : All	
Data bus width : 16 Bit	
Flash base address : 0x00000000	
Target address : 0x00000000	
Target image file : C:\WORK\2440\2440USB_Mon\u2440mon.bin	
* Current config *	
Device : SMDK2440	
BAN base address : 0x30000000	
RAM size : 0x03FFFFFF Byte sex : Little endian	
File config	
Flash name : Am291V00008	
Eraze : All	
Data bug width : 16 Bit	
Flash base address : 0x00000000 Target address : 0x00000000	
Target address : 0x00000000 Target image file : C:\W09K\2440\2440USB Mon\u2440man.bin	
tarder make tite	
<u>s</u>	2 -
	Connect



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.

Flash UP	8
Ble Utility Fjash About	
हे 🖥 🖋 🦧 \ominus 🔩	
	<u>A</u>
• Current config	
Device SMDR2440 RAM hase address 0x30000000 RAM size : 0x03FFFFF Byte sex : Little endian Filel config Flash name : Am29LVB0088 Erase : All Data bus width : 16 Bit Flash base address : 0x00000000 Target address : 0x00000000 Target image file : C:\W0RK\2440\2440USB_Mon\u2440mon.bin	
FileHame: u2440mom.bin FileSize: 35412	
Chip erase start	
Erase complete!!! u2440mon.bin downloading	
u2440mon.bin programming	
u2440mon.bin program complete !!!	
	2
	Connect



WRITING IMAGE FILES TO INTEL STRATA FLASH MEMORY WITH UART

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

le Search Brocessor View	s System	m Views Egecute Options <u>Window</u> Hel	
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	6		982 8 3 7 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
arget Image Files Class			
ARMSON 0			
ARM920T_0 - Disasse	en de las		Command Line Interface
			Command Line Interface
30000074 [0xe5920098]	Ldx	x0,0x30000114 ; = #0x4c00000c	Debug >obey C:\WORK\2440\2440norom\2440norom.ini
30000078 (0xe5802000)	90E	r2,[r0,#0]	Debug >com +
3000007c [0xe3a01020]	8.097	r1,#0x20	Debug >com Fileneme: 2440norgm.ini
30000080 [0xe2511001]	anpa	r1,r1,#1	Debug >com 2003. 5. xx 1st draft.
300D00D4 [Ox1afffffd]	hne	0x30000000 ; (EnterPMDM + 0x30)	Debug Scon 2003. 5. 38 150 drare.
30000088 [0xe59±0080]	Ldx	x0,0x30000110 ; = #0x48000024	Debug >com For 53C24400
3000008c (0xe5803000)	ace	£3,[£0,#0]	Debug >com SDRAM Little 32, 64MB
30000090 [0xels0f00e]	8.07	pc, E14	Debug >com FCLK(101.25MHz UPLL(48MHz
ENTER_SLE[0xe5920074]	1 dr 1 dr	r0,0x30000110 ; = #0x48000024	Debug >com SDGAM refresh: 64ms(6Krycle) -> 7.8us
30000098 [0xe5901000]		r1,[r0,#0]	Debug Sawat Swector catch 0x00
3000009c [0xe3811840] 300000e0 [0xe5801000]	0EE STE	r1,r1,#0x400000	An expression could not be parsed or evaluated in the given context
300000a0 [0xe3a01000]	802	r1,[r0,#0] r1,#0x10	Debug >swat isemihosting enabled 0x00
300000m8 [0xe2511001]	auba	r1,r1,#1	An expression could not be parsed or evaluated in the given context
300000ac [0x1afffffd]	hne	0x300000a0 ; (ENTER SLEEP + 0x14)	Debug Sowat por VIPt SVC
300000b0 [0xe59f1060]	Ldr	r1,0x30000118 ; = #0x56000080	Invalid expression
300000b4 [0xe5910000]	Leix	E0,[E1,#0]	Debug >com swat psc %IF 57C32
300000b8 [0xe3800ae0]	OFE	E0,E0,#0xe0000	Debug >com [disable Watch-Dog reset]
300000bc [0xe5810000]	str	r0,[r1,#0]	Debug >swat *0x53000100 0
	1 dr	r0,0x30000114 ; = #0x4c00000c	Debug >com << Clock setting >>
300000c4 [0xe5802000]	atx	r2,[r0,#0]	Debug >com [Fil lock time setting maximum]
300000c8 [Oxeafffffe]	b	0x30000008 J (ENTER SLEEP + 0x34)	Debug >swat #0x4c000000 ((Dxfffcc12)+(0xfffcc0))
WAKEUF SL(0xe59f1044]	ldr	r1,0x30000118 : = #0x56000080	Debug >com FCLE:HCLE:PCLE=1:2:2.
300000d0 [0xe5910000]	1 dir	r0,[r1,#0]	Debug >swat *0x40000014 ((Doc2)+(1oc1)+(0))
300000d4 [0xe3c00me0]	hic	r0,r0,#0xe0000	Debug >com [FCLK FMS setting:101.25MHz -> 0x7£,2,2]
30000049 [0xe5810000]	ate	x0,[x1,#0]	Debug >swat *Dx4c000004 ((Dx7fo(12)+(Dx2o(4)+(Dx2o(D)))
30000040 (0xe5920038)	Ldr	r0,0x30000110 J = #0x30000408	Debug >com [UCLK FHS setting: 400Biz -> 0x70,2,3]
300000e0 [0xe3a01448]	10.01	r1,#0x48000000	Debug >gwat *0x4c0000008 ((0x78<<12)+(0x2<<4)+(0x3<<0))
300000e4 [0xe2802034]	add	r2,r0,#0x34	Debug >com << Hemory setting >>
300000e8 [0xe4903004]	1 dir	z3,[z0],#4	Debug >com [Bank6/7: 32-bit bus width]
300000ec [0xe4813004]	ate	z3,[z1],#4	Debug >swat T0x48000000 0x22000000
300000£0 (0xeL520000)	cap	E2,E0	<pre>Debug >com [Bank0-5: Access cycls: 14-clocks, others:0-clock] Debug >swat *0x40000004 ((Doc13)+(Doc11)+(7cc0)+(0cc6)+(0cc4)+(0cc2)+</pre>
300000064 [0x14ffffb]	bne	Ox300000e8 ; (WAREUP_SLEEP + Ox1c	Debug >swat *0x48000104 [(bcc13)+(bcc11)+(bcc1)+(bcc6)+(bcc4)+(bcc2)+ Debug >swat *0x48000108 [(bcc13)+(bcc11)+(bcc8)+(bcc6)+(bcc4)+(bcc2)+
300000f8 [0xe3a01f40]	8.07	r1,#0x100	Debug >pwst *0x40100108 ((Dcc13)+(Dcc11)+(7cc8)+(Dcc4)+(Dcc4)+(Dcc2)+ Debug >pwst *0x40000108 ((Dcc13)+(Dcc11)+(7cc8)+(Dcc6)+(Dcc4)+(Dcc2)+
300000fc [0xe2511001]	arapa	r1,r1,#1	Reduce Served Revenues and an end of the contract of the contr
30000100 [0x1afffff4]	bine	0x300000fc ; (WAREUP_SLEEP + 0x30	seeal sees .maganess [[nots]4[nots]4[nots]4[not]4[not]4]000]4[nots]4
4	_		

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

e Search Processor View 8 168 169 169 169 169		m Views Execute Options Window He	• Feren • PP6• 5 1 2 20
arget Image Files Class			
APMS20E D			
ANNE201_0			
	_		
ARM920T_0 - Disasse	mbly		Command Line Interface
30000008 [0xea000057]	Ъ	Hendler3VI	Command Line Interface
3000000c [0xea000062]	ъ	Handler Pabert	Debug >swat par 4IFt_SWC
30000010 [0xea000005b]	b	HandlerDabort	Invalid expression
30000014 [Oxeafffffe]	b	0x30000014	Debug >com swat psr %IF_SWC32
30000018 [0xea000047]	ъ	HandlerIRQ	Debug >com [disable Watch-Dog reset]
The second secon	ъ	MandlerF10	Debug >swat *0x53000000 0
30000020 [0xea000008]	b	EnterFWDN	Debug >com << Clock setting >>
30000024 [0x0fl0eel1]	608	0x0f10ee11	Debug >com [Pil lock time setting maximum] Debug >swat *0x4c001000 ((0xfffccl2)+(0xfffcc0))
30000028 [0x0080e380]	ded	0x0080e380	Debug >com FCLE:HCLE:FCLE=1:2:2.
3000002c [0x0f10ee01]	ded	GwOflGeeD1	Debug Somat #0x4c000014 ((0<2)+(1<1)+(0))
30000030 [0xffffffff]	808	0xffffffff	Debug >com [FCLK PHS setting:101.25HEz -> 0x7f,2,2]
30000034 [0xfffffff]	ded	Oxffffffff	Debug >swat *0x4c000004 ((0x7foc12)+(0x2oc4)+(0x2oc0))
30000038 [0xfffffff]	ded	Gaffffffff	Debug >com [UCLK PMS setting:40MHz -> 0x78,2,3]
3000003c [0xfffffff]	608 608	Oxfffffff	Debug >smat #0x4c000008 ((0x78<12)+(0x2<<4)+(0x3<<0))
30000040 [0xfffffff] 30000044 [0xem000063]	b	ResetSandler	Debug >com << Hemory setting >>
EnterPHIN[0xela02000]	BOT	x2,x0	Debug >com [Bankd/7: 32-bit bus width]
3000004c [0xe3100008]	ESE	x0,#8	Debug >smat #0x40000000 0x22000000
30000050 [0x1a00000f]	bne	ENTER SLEEP	Debug >com [Bank0-5: Access cycle: 14-clocks, others:0-clock]
30000054 [0xe59200b4]	ldr	r0,0x30000110 ; = #0x48000024	Debug >swat *0x48000004 ((0cc13)+(0cc11)+(7cc8)+(0cc6)+(0cc4)+(0cc2)-
30000058 [0xe5903000]	ldr	x3,[x0,#0]	Debug >smat *0x48000008 ((0cc13)+(Dcc11)+(7cc8)+(Dcc4)+(Dcc2)-
3000005c [0xe1a01003]	BOW	r1,r3	Debug >swat *0x48000000 ((0<<13)+(0<<11)+(7<<8)+(0<<6)+(0<<4)+(0<<2)-
30000060 [0xe3811840]	DEE	r1,r1,#0x400000	Debug >swat #0x48000010 ((0cc13)+(Dcc11)+(7cc8)+(Dcc6)+(Dcc4)+(Dcc2)-
30000064 [0xe5801000]	ste	x1,[x0,#0]	Debug >zwat 70x48000014 ((0<13)+(D<11)+(7<8)+(D<6)+(D<6)+(D<6)+(D<2)-
30000068 [Oxe3m01010]	BOW	x1,#0x10	Debug >swat *0x40000010 ((0cc13)+(0cc11)+(7cc0)+(0cc6)+(0cc4)+(0cc2)-
3000006c [0xe2511001]	anab ar	r1,r1,#1	Debug >com [Bamb6/7: SDRAM, Trod:20lock, CA:9-bit] Debug >swat "0x4800001c ((30015)+(Do02)41)
30000070 [0x1afffffd]	bne	0x3000006c ; (EnterPMDN + 0x24)	Debug Same *0x4000002 ((30015)+(0002)+1)
30000074 [0xe59d0098]	ldr	x0,0x30000114 J = #0x4c00000c	Debug >com [SDRAM refresh enable, Trp=2clk, Trc=5clk, Refresh:1654]
30000076 [0xe5802000]	ptr	x2,[x0,#D]	Debug >swat *0x48000024 ((1cc23)+(0cc22)+(0cc20)+(1cc18)+1654)
3000007c [0xe3a01020]	LOT	r1,#0x20	Debug >com [SCHE EN enable, SCHE EN enable, Bank6/7 memory map: 6408.
30000080 [0xe2511001] 30000084 [0x1afffff6]	SUDS DDE	1, 1, 1, 41 0x30000080 ; (EnterFMDN + 0x38)	Debug >swat *0x49000028 (0x1+(1<<5)+(1<<4))
30000084 [0x14fffff] 30000085 [0xe5920080]	ldr	10,0x30000010 ; (interPMDM + 0x35) 10,0x30000110 ; = #0x48000024	Debug >com [Bank6/7 CL: 3-clocks]
3000008c [0xe5803000]	ate	x3,[x0,#0]	Debug >swat 70x4800002c 0x30
300000000 [0xela0100e]	BOV	pc.c14	Debug >smat #0x40000030 0x30
ENTER_SLE[0xe5960074]	ldr	rD,Dx30000110 : = #0x48000024	Debug >
		setures to the property	<



3. Execute 2440TEST code with Go command.

AXO 🚯		
<u>File Search</u>	Processor Views System Views Egecute Options Window He	
自國陸	SIF 5 96 DECENC	
Target Imag	ge Files Class	
APN		
	1221_0	
-		
ARM920	0T_0 - C:WYHLEEWS3C2440W2440Test-r00W 🔲 🗖 🔀	Command Line Interface
123	// Ridden	Command Line Interface
124	#define NFLL IEL (0)	Debug Sawat par AIFt SVC
125	#define FCLK SEL (2)	Invalid expression
126	Forcast room_one (a)	Debug >oom swat par AIF SVC32
127		Debug >com [disable Watch-Dog reset]
128		Debug >smat *Dx53000000 0
129		Debug >com << Clock setting >>
130	//	Debug >com (FLL lock time setting maximum)
131		Debug Sawat *Dx4c000000 ((0xfffcc12)+(0xfffcc0))
132		Debug >com FCLK:HCLE:PCLE=1:2:2.
133	yoid Hain(void)	Debug >swat *0x4c000014 ((0cc2)+(1cc1)+(0))
134		Debug Scen [FCLK FHS setting:101.25HHz -> 0x7f,2,2]
135	int i;	Debug Sumat *Dx4c0D0004 ((0x7fcc12)+(0x2cc4)+(0x2cc0)) Debug Soma (UCLK FMS setting:48MME -> 0x78,2,3)
136	unsigned int apl1_val;	Debug >dwat *Dx4c0D0008 ((0x78cc12)+(0x2cc4)+(0x3cc0))
137		Debug Scon (C Henory setting SS
135	<pre>led_Display(0xf);</pre>	Debug >com [Bank6/7: 32-bit bug width]
140	// HNU init, I/D cache on,	Debug >#wat *0x48000000 0x22000000
140	MEU_Init();	Debug Scom [Bank0-5: Access cycle: 14-clocks, others:0-clock]
142	Huo_anac();	Debug >swat *Dx40000004 ((0<<13)+(0<<11)+(7<<0)+(0<<6)+(0<<4)+(0<<2)+(
143	#if AD810	Debug >swat *0x48000008 ((0ccl3)+(0ccl1)+(7cc8)+(0cc6)+(0cc4)+(0cc2)+(
144	_rt_lib_init(); //for ADS 1.0	Debug >swat *Dx4800000c ((0cc13)+(0cc11)+(7cc8)+(0cc6)+(0cc4)+(0cc2)+(
145	fendif	Debrug >smmat *Dx480000010 ((0<<13)+(0<<11)+(7<<0)+(0<<6)+(0<<4)+(0<<2)+(
146		Debug >swat *0x48000014 ((0cc13)+(0cc11)+(7cc8)+(0cc6)+(0cc4)+(0cc2)+(
147	// Clock setting	Debug >swat *Dx480D0018 ((0cc13)+(0cc11)+(7cc8)+(0cc6)+(0cc4)+(0cc2)+(
148	ChangeElockDivider(12,12); // 1:2:4	Debug >com [Bank6/7: SDFAM, Trrd:2clork, CA:9-bit]
149	Change#9119alue(246,13,0); // 203.2MHz	Debug >swat *0x4000001c ((3cc15)+(0cc2)+1)
150	//ChangeMP11Value(88,1,1); // 192MHz	Debug >swat *0x48000120 ((3cci5)+(0cc2)+1) Debug >com [SDRAM refresh enable, Trp=2clk, Trc=5clk, Refresh:1654]
151	ChangeU911Value(Dx38,2,2); // 400Hz	Debug Scen [Slean Ferrern enable, trp=2cik, trc=5cik, serrern:1054] Debug Smut *Dx40000024 ((1<2)+(0<22)+(0<20)+(1<10)+1654)
152	At these selecteries are description interview.	Debug >com (3CKE EN enable, SCLK EN enable, Bank6/7 memory map: 6428/
153	// Clock calculation only for display information Calc Clock/Dis	Debug Sowat *Dx48000028 (Dx1+(1c(5)+(1c(4))
154	Calc_Clock(0); UFDATE PEFFESH(Hclk);	Debug Scon [Bank6/7 [L: 3-clocks]
155	STORIG PROPAGE (INCAR) -	Debug >swat *0x40000020 0x30
157	// OPID port init.	Debug >swat *0x48000030 0x30
158	Fort_Init[]:	Debrug >
4		C 2
or Help, pres	is F1	Running Image Line 134, Col D Multi-ICE (ARM320TL0 2440test,auf



- 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.

🚥 DNW v0.50A [CC)M1,115200bps][USB	:ОК]		×
Serial Port USB Port	Configuration Help			
[rSTATUS3=0]				^
[SMDK2440 Board Te	st Program Ver 0.0]			
[Fclk:Hclk/Pclk]=[[Uclk=48.0Mhz]	203.2:101.6:50.8]Mh	z		
4:nWAIT test	5:Nand test	2:PCMCIA test 6:Program Flash	7:DMA test	
8:Interrupt test	9:Cpu speed test	10:Power/Clk test 14:IIC Test	11:LCO test 15:RTC Test	
		18:ADC test		
		22:Uart Test		
Select the functio	n to test : 6			
[NOR Flash Memory	Writer Ver 0.1]			
	: 0x31000000 ~ 0x3			
a : AM29LV800BB x1 Select the type of	b : 28F128J3A	(16MB) x2		
serect the type of	a riash memory :			~

- 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.



DNW v0.50A [COM1,115200bps][USB:OK]							
Serial Port USB Port Configuration Help							
0:User Test1:Manual Reg Set2:PCMCIA test3:Steppin4:nWAIT test5:Nand test6:Program Flash7:DMA test8:Interrupt test9:Cpu speed test10:Power/Clk test11:Lcd test	it 🗌						
12:Camera test 13:SPI Test 14:IIC Test 15:RTC Tes	t						
16:IrDA Test 17:SD test 18:ADC test 19:ADC TS							
20:Timer test 21:IIS test 22:Uart Test 23:Clkdiv_	Test						
Select the function to test : 6							
[NOR Flash Memory Writer Ver 0.1]							
The program buffer : 0x31000000 ~ 0x33ff0000							
a : AM29LV800BB x1 b : 28F128J3A(16MB) x2 Select the type of a flash memory ?(b)							
Do you want to download through UARTO from 0x31000000? [y/n] (: y)							
downloadAddress = 31000000							
Download the plain binary file(.BHC) to be written The file format : <n+6>(4)+(n)+CS(2)</n+6>							
To transmit .BIN file : wkocm2 xxx.BIN /1 /d:1							
Download methods : COM:8Bit,NP,1STOP							
STATUS :	•						



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

DNW v0.50A					- C 🛛
Serial Port USB P	ort Configuration	Help			
월기					? 🔀
찾는 위치([):	2440US8_Mon		• •	🗈 💣 🎫	
내 최근 문서	🖃 u2440mon, bin				
비망 화면					
이 등 세근 () 내 문서					
내 컴퓨터					
내 네트워크 환경					
	파일 이름(<u>N</u>):	u2440mon,bin		•	열기(0)
	파일 형식(T):	BIN Files (+,bin;+,nb0)		(희소

- Select and Download a target file.

DNW VO.	60А [COM1,115200bps][USB:OK]	×
Serial Port L	ISB Port Configuration Help	
Download t The file f To transmi	dress = 31000000 he plain binary file(.BHC) to be written ormat : <n+6>(4)+(n)+CS(2) t .BIN file : wkocm2 xxx.BIN /1 /d:1 ethods : COM:8Bit,NP,1STOP</n+6>	^
Download O	######################################	
1. 28F128J J1:1-2, J 2. After p	ery Important Notes *** 3A must be located at 0x08000000. 2:2-3, J3:2-3, J4:1-2 rogramming, 28F128J3A may be located at 0x0. 2:1-2, J3:1-2, J4:2-3	
[28F128J3	A Writing Program] e [0x?] : 0h~8a54h	1
Øh,20000h,	Target Offset Address [0x?] : 40000h,, 1ce0000h et address offset [0x?] :	*

SAMSUNG ELECTRONICS

8. Write input target-offset address.

DNW v0.50A [COM1.115200bps][USB:OK]	- 🕫 🗙
Serial Port USB Port Configuration Help	
Download the plain binary file(.BHC) to be written	1
The File Format : <m+6>(4)+(m)+CS(2) To transmit .BIM File : wkocm2 xxx.BIM /1 /d:1</m+6>	
lownload nethods : CONINGIAN, ISTOP	
STATUS : MANAMAMAMAMAMAMAMAMAMAMAMAMAMAMAMA Download O.K.	
Boonload U.K.	
[28F128J3A Flash Writing Program]	
*** Very Important Notes *** 1. 28F128J3A must be located at 0x08000000.	
1: 201 22030 MBSC WE ADGRED OF BAGGED OF BAGGED DDDD. J1:1-2, J2:2-3, J3:2-3, J4:1-2	
2. After programming, 20F120J3A may be located at 0x0.	
J1:2-3, J2:1-2, J3:1-2, J4:2-3	
[28F128J3A Writing Program]	
Source size [@x?] : 0h~8a54h	
Available Target OFFset Address [0x7] :	
06,20000h,40000h,, 1ce0000h	
Input target address offset [0x7] :(0x0)	
Source base address(0x310000000) - 0x31000000 Target base address(0x00000000000) - 0x80000000	
Jarget offset (0x0) = 0x0	
Target size (0x20000+n) = 0x8a54	
Erase the sector : 0x8000000.	
Erase the sector , washington.	
Start of the data writing	
End of the data writing	
Verifying Start	
Verifying End!!!	
0:User Test 1:Hanual Reg Set 2:PCHCIA test 3:Stepping stone 4:nWAIT test 5:Nand test 6:Program Flash 7:DHA test	
8:Interrupt test 9:Gpu speed test 10:Power/Clk test 11:Lcd test	
12:Camera test 13:SPI Test 14:IIC Test 15:RTC Test	
16:IrDA Test 17:UART Test 18:SD test 19:ADC test 20:ADC TS test 21:Timer test 22:IIS test 23:Clkdiv Test	
zerou la test zicitaler test zzicita test zaicikulo_rest	
Select the function to test :	

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.

le <u>E</u> dit <u>S</u> earc ≆ ⊕ ⊛ ଜ∕		C <u>++</u> E <u>x</u> ecute Options <u>Wi</u> ndow <u>H</u> elp RM ▼ EFEL EF 79 79 79 40 €	
ARM - Executi	on Window	<u> 10,110</u>	Command Window
x00000490	Ы	0xa2c	ARMsd Command Interface Debug: ob d:\2410\norom.ini
x00000494	Ъl	0x34c	> * S3C2410
00000498	mov	r0,#0	> * SDRAM Little_32
:0000049c	Ъl	0x864	> * 64MB
(000004a0	MOV	r0,#0	
x000004a4	bl	0xc08	>let \$vector_catch = 0x00
x000004a8 x000004ac	add bl	r0,pc,#0x200 ; #0x6b0 0xdc4	>let \$semihosting_enabled = 0x00
000004aC 0000004b0	DI MOV	uxac4 r1 #0x32	>let psr=%IFt_SVC
x000004b0	add	r0,pc,#0x220 ; #0x6dc	> let psr=%IF_SVC32
000004b4	bl	0xdc4	
000004bc	ldr	r2,0x00000718 ; = #0x33ffff0	n > disable wdt
000004c0	nov	r1.#0x30000000) let 0x53000000=0
000004c4	add	r0,pc,#0x250 ; #0x71c	
000004c8	bl	0xdc4	> pllset
000004cc	ldr	r2,0x00000738 ; = #0x33ff000	<pre>0 >let 0x4c000004=((0x70<<12)+(0x4<<4)+0x2)</pre>
000004d0	nov	r1.#0x3000000	<pre>>let 0x4c000008=((0x58<<12)+(0x4<<4)+0x2)</pre>
000004d4	add	r0,pc,#0x260 ; #0x73c	
000004d8	Ъl	0xdc4	> memset
000004dc	mov	r4,#0x3000000	>let 0x48000000=0x22000000
000004e0	adds	r12,r4,#0xcd000000	>let 0x48000004=((0<<13)+(0<<11)+(7<<8)+(0<<6)+(0<<4)+(0<<2)+0)
000004e4	subcss	r12,r12,#0xff0000	>let 0x48000008=((0<<13)+(0<<11)+(7<<8)+(0<<6)+(0<<4)+(0<<2)+0)
:000004e8	bcs	0x500	>let 0x4800000c=((0<<13)+(0<<11)+(7<<8)+(0<<6)+(0<<4)+(0<<2)+0)
:000004ec	Ъ	0x4f8	>let $0x48000010 = ((0<<13)+(0<<11)+(7<<8)+(0<<6)+(0<<4)+(0<<2)+0)$
:000004f0	add	r4,r4,#0x100	<pre>>let 0x48000014=((0<<13)+(0<<11)+(7<<8)+(0<<6)+(0<<4)+(0<<2)+0) >let 0x48000018=((0<<13)+(0<<11)+(7<<8)+(0<<6)+(0<<4)+(0<<2)+0)</pre>
:000004f4	Ъ	0x4e0	>let 0x4800001c=((3<<15)+(0<<2)+1)
:000004f8	str	r4,[r4,#0]	>let 0x48000020=((3<<15)+(0<<2)+1)
000004fc	b	0x4f0	>let 0x48000024=((1<<23)+(0<<22)+(0<<20)+(1<<18)+(2<<16)+1113)
00000500	add bl	r0,pc,#0x250 ; #0x758 0xdc4	>let 0x48000028=0x32
200000504 200000508	DI MOV	uxac4 r4.#0x3000000	>let 0x4800002c=0x20
000000508 00000050c	mov adds	r4,#0x30000000 r12,r4,#0xcd000000	Debug:
00000500		12,14,#0XC100000	



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

AND a Search Processor Views	System Views Execute Options Window	. Help
1 🗳 1 🖉 🥵		
arget Image Files Class		
AFMS20T_0		
		Command Line Interface
ARM920T_0 - Disassembl	y	Command Line Interface
StffffeD [Oxfffffff] do		
2fffffe4 [0xfffffff] de		Debug >sbey C:\WORK\2440\2440norom\2440norom.imi Debug >com
2fffffeb [Oxfffffff] dc		Debug >com Filename: 2440norom.ini
Stffffeo [Oxffffffff] 60		Debug >com 2003, 5, xx 1st draft.
2ffffff0 [Oxfffffff] de		Debug Scon stor. S. AX Int district
Zffffff4 [Dxffffffff] dc		Debug >com For 33C2440%
Sterrers [Oxererers] do		Debug >com HDBAH Little 32, 648B
SIEFFEE [Oxfffffff] do SODODOD [OxemD00070] b	 Oxffffffff J 7 undefined ResetTandler 	Debug Scon FCLK:101.25MHz UPL1:46MHz
L	RandlerUnder	Debug >com SDRAH mefresh: 64ms(0Kcycle) -> 7.8us
30000004 [0xea00004e] b 30000008 [0xea000053] b	Handler3WI	Debug >swat #vector_oatch 0x00
3000000c [0xea00005c] b	HandlerFabort	An expression could not be parsed or evaluated in the given context
30000010 [0xea000057] b	MandlerDabort	Debug >swat Spenihosting enabled 0x00
30000014 [Oxeafffffe] b	0x30000014	An expression could not be parsed or evaluated in the given context
30000018 [0xea000043] b	HandlerTRQ	Debug >swat por hIFt SVC
3000001c [0xem00003c] b	HandlerF10	Invalid expression
30000020 [0xea000008] b	EnterFMDN	Debug >com swat psz %IF_SWC32
30000024 [0x0fl0eel1] do		Debug >com [disable Watch-Dog reset]
30000028 [0x0080e380] dc		Debug >swat *0x53000000 0
3000002c [0x0f10ee01] dc		Debug >com << Clock setting >>
30000030 [0xfffffff] 60		Debug >com [FLL lock time setting maximum]
30000034 [OxEFEFFFF] do		Debug >swat *0x4c000000 ((0xfffcc(12)+(0xfffcc0)))
30000038 [Oxfffffff] do		Debug >com FCLM:HCLM:FCLM=1:2:2.
3000003c [Oxffffffff] do		Debug >swat *0x4c000014 ((0<<2)+(1<<1)+(0))
30000040 [0xfffffff] 60		Debug >com [FCLK FMS setting:101.25MHz -> 0x7f,2,2]
30000044 [OxenD0005f] b	ResetMandler	Debug Sawat *Dx4c000004 ([0x7fc(12)+(Dx2c(4)+(0x2c(0))]
EnterPMDN[0xela02000] no	w z2,z0	Debug >com [UCLK FHS setting:400Hz -> 0x70,2,3]
3000004c [0xe3100008] ts	t x0,#8	Bebug >swat *0x4000008 ([0x78cc12)+(0x2cc4]+(0x3cc0))
30000050 [0x1a00000£] hn		Debug >com << Hemory setting >>
30000054 [0xe59d00a4] 1d		Debug >com [Bank6/7: 32-bit bus width]
30000058 [0xe5903000] 1d		Debug >swat *0x4900000 0x22000000
3000005c [0xe1s01003] mo		Debug >com [Bank0-5: Access cycle: 14-clocks, others:0-clock] Debug >smat *0x40000004 ([0cc13]+(Dcc11)+(7cc0)+(Dcc6)+(Dcc4]+(0cc2)+0)
30000060 [0xe3811840] or		Debug smac *0x40000008 ([0ccl3]+(0ccl1)+(7cc0)+(0cc6]+(0cc4]+(0cc2]+0]
30000064 [0xe5801000] st		Debug >swat *0x48000008 (100c13)+(00c11)+(70c8)+(00c6)+(00c4)+(00c2)+0) Debug >swat *0x48000008 (100c13)+(00c11)+(70c8)+(00c6)+(00c4)+(00c2)+0)
30000068 [0xe3m01010] mo		Debug Smat "0x48000000 ([00013]+(00011)+(7008)+(0008)+(0004]+(0002)+0)
3000006c [0xe2511001] au	bs r1,r1,∉1	same and come and the set of the
1		15
Help, press F1		(No Pos> Multi-ICE ARM820T.0 2040test.ax



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

		Views Egecute Options Window Help	and the second second second second second second
Load [mage	6 9		982 8 7 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Load Debug Symbols			
Beload Current Image			
Open File			
Load Memory From File			Commond I for Interface
Save Memory To File	bly		Command Line Interface
Flash Download	beg	0x30000b00 ; (Uart_TxEmpty + 0x44)	*
riasii Eomioad	2007	pc, 114	Debug >swat psr %IFt_SWC
Load Session	ldr	r0,0x300004f4 ; = #0x3005f314	Invalid expression
Saye Session	ldr	r0,[r0,#0]	Debug >com swat psr %IF_SVC32 Debug >com (diseble Watch-Dog reset)
Recent Files	cup	E0,#0	Debug >swat #0x53000000 0
Recent Elles	bele boy	0x30000b38 ; (Uart_Getch + 0x28) s1,#0x50000000	Debug >com << Clock setting >>
Recent Images	ldr	E0,[E1,#0x10]	Debug >com (Fil lock time setting maximum)
Recent Symbols	tat	z0,#1	Dabug >smat *0x4c000000 [(0xfffc(12)+[0xfffc(0)]
Recent Sessions	beg	0x30000b24 ; (Uart_Getch + 0x14)	Debug >com FCLK:HCLK:FCLK=1:2:2.
Unload Current Image	ldrb	E0,[E1,#0x24]	Debug >swat *0x4c000014 ((Dcc2)+(1cc1)+(0))
Import Formatis	2077	pc,r14	Debug >com [FCLK PHS setting:101.25HHz -> 0x7f,2,2]
import rormajs	cup	20,#1	Debug >swat *0x4c0000014 ((0x7fcc12)+(0x2cc4)+(0x2cc0)) Debug >com [UCLK PMS setting:48MMz -> 0x78,2,3]
Exit	bne	0x30000b58 ; (Dart_Getch + 0x48)	Debug >gwat *0x40000008 ((0x78cc12)+(0x2cc4)+(0x3cc0))
30000644 [0xe5910010]	ldr ldr	<pre>s1,0x300004f8 ; = #0x50004000 s0,[s1,#0x10]</pre>	Debug >com << Hemory setting >>
30000b48 [0xe3100001]	tat	r0,#1	Debug >com [Bank6/7: 32-bit bus width]
30000640 [0x0afffff0]	beg	0x30000b44 J (Uart_Getch + 0x34)	Debug >swat *0x49000000 0x22000000
30000b50 [0xe5d10024]	ldrb	r0,[r1,#0x24]	Debug >com [Sank0-5: Access cycle: 14-clocks, others:0-clock]
30000b54 [0xela0f00e]	26012	pc, x14	Debug >swat *0x48000004 ((0<<13)+(0<<11)+(7<<8)+(0<<6)+(0<<4)+
30000b58 [0xe3500002]	cup	E0,#2	Debug >swat *0x48000008 [(Dcc13)+[0cc11)+(7cc8)+[0cc6)+[0cc4]+
30000b5c [0x11aDfD0e]	novne	pc,r14	Debug Sawat *0x4800000c [(Dec13)+[0cc1])+[7cc8)+[0cc6]+[0cc4]+ Debug Sawat *0x48000010 [(Dec13)+[0cc1])+[7cc8)+[0cc6]+[0cc4]+
30000b60 [0xe59f1294]	ldr	<pre>k1,0x300004fc ; = #0x50008000</pre>	Debug Sawat *0x43000014 ((Dcc13)+(0cc11)+(7cc3)+(0cc4)+(0cc4)+
30000b64 [0xe5910010] 30000b68 [0xe3100001]	1dr tat	z0,[z1,#0x10] z0,#1	Debug >swat *0x40000018 [(Dcc13)+[0cc11)+(7cc8)+[0cc6)+[0cc4]+
30000b6c [0x0afffffc]	beg	0x30000b64 ; (Uart_Getch + 0x54)	Debug >com (Bankd/7: SDRAH, Tred:2clock, CA:9-bit)
30000b70 [0xe5d10024]	ldrb	r0,[r1,#0x24]	Debug >smat *0x4000001c ((3<<15)+(0<<2)+1)
30000b74 [0xela0f00e]	10017	p0,x14	Debug >swat *0x48000020 ((3cc15)+(0cc2)+1)
Uart_GetE[0xe5920274]	ldr	±0,0x30000df4 ; = #0x3005f314	Debug >com [SDRAM refresh enable, Trp=2clk, Trc=5clk, Refresh:)
30000b7c [0xe5900000]	1dr	r0,[r0,#0]	Debug >swat *0x40000024 ((1<<23)+(0<<20)+(1<<10)+1654)
30000b80 [0xe3500000]	cmp	E0,#0	Debug >com (SCRE_EN enable, SCLK_EN enable, Bank6/7 memory map: Dabug >swat *Ox48000028 [Ox1+(1<<5)+(1<<4))
30000b84 [0x1a000004]	bne	0x30000b9c ; (Dart_GetEey + 0x24)	Debug Some (Renk6/7 CL 3-clocks)
30000b88 [0xe3a01450] 30000b8c [0xe5910010]	hov ldr	<pre>x1,#0x5000000 x0,[x1,#0x10]</pre>	Debug Sovat T0x4800002c 0x30
30000b90 [0xe2100001]	ands	r0,r0,#1	Debug >swat *0x40000030 0x30
30000b94 [0x15410024]	ldrneb	E0,[E1,#0x24]	E Debug >



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

Load Memory F	rom File						?
찾는 위치(!):	🗁 2440USB_Mon	•	¢	£	Ċ	∷ ≣ ▼	
🖬 u2440mon, bin							
파일 이름(<u>N</u>):	u2440mon,bin					열기	(<u>0</u>)
파일 형식(<u>T</u>):	All Files (*,*)		•]		취:	소
Address: 0x3100000	0						
Processors							
Processors ARM920T_0							

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

AND le Search Processor View	s System	News Egecute Options Window Help		
				111 11 11 11 11 11 11 11 11 11 11 11 11
arget Image Files Class				
ARMS20T_D				
			-	
ARM920T_0 - Disasse	mbly		\mathbf{X}	Command Line Interface
30000b18 [0xe3500000]	080	x0,#0	1	Command Line Interface
30000blc [0x1a000005]	bne	0x30000b38 ; (Uart_Setch + 0x28)	-	Invalid expression
30000b20 [0xe3a01450]	BOT	r1,#0x50D0000D	-	Debug >com swat psr %IF_SWC32
30000b24 [0xe5910010]	ldr	x0,[x1,#0x10]		Debug >com [disable Watch-Dog reset]
30000b28 [0xe3100001]	6.86	x0,#1		Debug >swat *0x53000000 D
30000b2c [0xDafffffc]	beq	0x30000b24 ; (Uart_Setch + 0x14)		Debug >com << Clock setting >>
30000b30 [0xe5d10024]	ldrb	x0,[x1,#0x24]		Debug >com [PLL lock time setting maximum]
30000b34 [0xela0f00e]	BOV	p0, £14		Debug >swat *0x4c000000 ((0xfffcc12)+(0xfffcc0))
30000b38 [0xe3500001]	cup	z0,#1		Debug >com FUERHUER: FUER-1:2:2.
30000b3c [0x1a000005]	bne	0x30000b50 ; (Uart_Setch + 0x40)		Debug >swat *0x4c000014 ((0<<2)+(1<<1)+(0))
30000b40 [0xe59f12b0]	ldr	x1,0x300004f8 J = #0x50004000		Debug >com [FCLK PMS setting:101.25MHz -> 0x7f,2,2]
30000b44 [0xe5910010]	ldr	r0,[r1,#0x10]		Debug >zwat 70x4c000004 ((0x76oc12)+(0x2oc4)+(0x2oc0))
30000b48 [0xe3100001]	tat	rD,#1		Debug >com [UCLK PHS setting: 400Hz -> 0x70,2,3]
30000b4c [0x0afffffc]	pet	0x30000b44 ; (Tart_Setch + 0x34)		Debug >swat *0x4c000008 ((0x78cc12)+(0x2cc4)+(0x3cc0)) Debug >com << Memory setting >>
30000b50 [0xe5d10024]	ldrb	r0,[r1,#0x24]		Debug >com [Bankd/7: 32-bit bus width]
30000b54 [0xels0f00e]	BOT	pc,r14		Debug >swat *0x40000000 0x22000000
30000b58 [0xe3500002]	cap	x0,#2		Debug >com [Bank0-5: Access cycle: 14-clocks, others:0-clock]
30000b5c [0x11a0f00e]	novne	p0,r14	11	Debug >zwat 70x48000004 ((0cc13)+(0cc11)+(7cc8)+(0cc6)+(Dcc4)+(
30000b60 [0xe59£1294]	ldr	r1,0x30000dfc ; = #0x50008000		Debug >mat #0x40000000 ((0cc13)+(0cc11)+(7cc0)+(0cc6)+(0cc4)+(
30000b64 [0xe5910010]	ldr	r0,[r1,#0x10]		Debug >swat *0x48001000 ((0cc13)+(0cc11)+(7cc8)+(0cc6)+(0cc4)+(
30000b68 [0xe3100001] 30000b6c [0x0afffffc]	tat	10,01		Debug >zwat *0x48000010 ((0cc13)+(Dcc11)+(7cc8)+(Dcc6)+(Dcc4)+(
30000b70 [0xe5d10024]	ldrb	0x30000b64 ; (Uart_Detch + 0x54) r0,[r1,#0x24]	-	Debug >smat #0x480000014 ((0cc13)+(Dcc11)+(7cc0)+(Dcc6)+(Dcc4)+(
30000b74 [0xela0f00e]	ROT	10,[11,9024]		Debug >swat *0x48000018 ((0cc13)+(0cc11)+(7cc8)+(0cc6)+(0cc4)+(
Uart_GetK[0xe59f0274]	ldr	10.0x30000df4 ; = #0x3005f314		Debug >com [Bankd/7: SDUAM, Trod:2clock, CA:9-bit]
30000b7c [0xe5900000]	ldr	10,00000000000000000000000000000000000		Debug >swat *0x4800001c ((3<<15)+(0<<2)+1)
30000b00 [0xe3500000]	Chip	r0,#0		Debug >swat *0x480000020 ((3<<15)+(0<<2)+1)
30000b84 [0x1a000004]	boe	0x30000b9c ; (Uart_SetKey + 0x24)		Debug >com [808AM refresh enable, Trp=201k, Trc=501k, Befresh:1
30000b88 [0xe3a01450]	BOT	r1,#0x50D0000D		Debug >swat *0x48000024 ((10023)+(D0022)+(D0020)+(10018)+1654)
30000b8c [0xe5910010]	ldr	r0,[r1,#0x10]		Debug >com [SCHE_EN enable, SCLE_EN enable, Bank6/7 memory map:
30000b90 [0xe2100001]	ends	x0,x0,#1		Debug >swat *0x48000028 (0x1+(1cc5)+(1cc4))
30000b94 [0x15610024]	ldrneb	r0,[r1,#0x24]		Debug >com [Bankd/7 CL: 3-clocks]
30000b98 [0xela0f00e]	BOT	pc, r14		Debug >smat #0x4800802c 0x30
30000b9c [0xe3500001]	cap	x0,#1		Debug >swat *0x48000030 0x30
30000baD [0x1a000004]	bne	0x30000bb8 ; (Uart_DetKey + 0x40)	-	Debug >go
30000ba4 [0xe59f124c]	1dr	r1,0x30000df8 ; = #0x50004000	Ŧ	Debug >
			1 1	C
Help, press F1	_		_	Inning Image Line 129, Col D Multi-ICE (ARM 920TL0 2440test av



- 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.

```
DNW v0.50A [COM1,115200bps][USB:OK]
                                                                         Serial Port USB Port Configuration Help
[rstatus3=0]
                                                                                ~
[SMDK2440 Board Test Program Ver 0.0]
[Fclk:Hclk/Pclk]=[203.2:101.6:50.8]Mhz
[Uclk=48.0Mhz]
 0:User Test
                   1:Manual Reg Set 2:PCMCIA test
                                                         3:Stepping stone
4:nWAIT test
                   5:Nand test
                                      6:Program Flash
                                                         7:DMA test
8:Interrupt test
                   9:Cpu speed test 10:Power/Clk test 11:Lcd test
12:Camera test
                  13:SPI Test
                                     14:IIC Test
                                                        15:RTC Test
16:IrDA Test
                  17:UART Test
                                     18:SD test
                                                        19:ADC test
20:ADC TS test
                  21:Timer test
                                     22:IIS test
                                                        23:Clkdiv_Test
Select the function to test (
[ NOR Flash Memory Writer Ver 0.1 ]
The program buffer : 0x31000000 ~ 0x33ff0000
a : AM29LV800BB x1
                      (b): 28F128J3A(16MB) x2
Select the type of a flash memory ? b
Do you want to download through UARTO from 0x31000000? [y/n] : n
```

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.

```
DNW v0.50A [COM1,115200bps][USB:OK]
                                                                          Serial Port USB Port Configuration Help
                                                                                ~
 0:User Test
                                      2:PCMCIA test
                   1:Manual Req Set
                                                          3:Stepping stone
4:nWAIT test
                   5:Nand test
                                      6:Program Flash
                                                         7:DMA test
8:Interrupt test
                   9:Cpu speed test 10:Power/Clk test 11:Lcd test
12:Camera test
                  13:SPI Test
                                     14:IIC Test
                                                        15:RTC Test
                                                         19:ADC test
                  17:UART Test
                                      18:SD test
16:IrDA Test
20:ADC TS test
                  21:Timer test
                                     22:IIS test
                                                        23:Clkdiv Test
Select the function to test : 6
[ NOR Flash Memory Writer Ver 0.1 ]
The program buffer : 0x31000000 ~ 0x33ff0000
a : AM29LV800BB x1
                       b : 28F128J3A(16MB) x2
Select the type of a flash memory ? b
Do you want to download through UARTO from 0x31000000? [y/n] : n
[ 28F128J3A Flash Writing Program ]
     *** Very Important Notes ***
1. 28F128J3A must be located at 0x08000000.
J1:1-2, J2:2-3, J3:2-3, J4:1-2
After programming, 28F128J3A may be located at 0x0.
J1:2-3, J2:1-2, J3:1-2, J4:2-3
The data must be downloaded using ICE or USB from 0x31000000
[ 28F128J3A Writing Program ]
Source size [0x?] : Oh~Oh
Available Target Offset Address [0x?] :
0h,20000h,40000h, ..., 1ce0000h
Input target address offset [0x?] :
```

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



DNW v0.50A [COM1.115200best][US8:0K]	. 8 X
Serial Port USB Port Configuration Help	
The program buffer : 0x31000000 ~ 0x33ff0000 a : AM29LU000088 x1 b : 20F120J3A(1600) x2 Select the type of a flash memory ? b Do you want to download through UARTO from 0x31000000? [y/m] : n	^
[28F128J3A Flash Writing Program]	
*** Very Important Hotes *** 1. 28F128J3A must be located at 0x0000000. J1:1-2, J2:2-3, J3:2-3, J4:1-2 2. After programming, 28F128J3A may be located at 0x0. J1:2-3, J2:1-2, J3:1-2, J4:2-3	
The data must be downloaded using ICE or USB From 0x31000000	
[28F128J3A Writing Program]	
Source size [0x7] : 0h~0h	
available Target Offset Address [0x7] : 0h,20000h,40000h,, 1ce0000h Input target address offset [Dx7] : 0x0 Input target size [0x7] <u>0x10000</u> Source base address(0x010000000) - 0x01000000 Target base address(0x010000000) - 0x0000000 Target offset (0x0) = 0x0 Target size (0x20000=n) = 0x10000	
Erase the sector : 0x8000000. Block_8000000 Erase 0.K.	
Start of the data writing [1] End of the data writing Werifying Start Werifying Endfff 8:User Test 1:Manual Reg Set 2:PCHCIA test 3:Stepping stone 4:nWAIT test 5:Mand test 6:Program Flash 7:DMA test 8:Interrupt test 9:Cpu speed test 10:Power/Clk test 11:Led test 12:Camera test 13:SPI Test 14:LET Test 15:RTC Test 16:IrDA Test 17:UART Test 18:SD test 19:ADC test 20:ADC TS test 21:Timer test 22:LIS test 23:Clkdiv_Test	
Select the function to test :	~

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.

- 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-B: 2-3 (short) J2-B: 1-2 (short) J3-B: 1-2 (short) J4-B: 2-3 (short)
- 3. Run the Flash Write program and select Connect MDS from the File menu.

Flash UP	
Elle Utility Flash Help	
🔆 🗟 🚄 📭 📭	
- V 🛱 🛋 🗠 🛄 💶 🖓	
	<u></u>
	-
-	2
	Connect

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.

		· [Add device			
No.	Register name	Address	Value	Size	Attr	^
1	WTCON	53000000	00000000	4	R/W	
2	INTMSK	4A000008	FFFFFFFF	4	RW	
3	INTSUBMSK	4A00001C	000007FF	4	R/W	
4	LOCKTIME	4C000000	00FFFFFF	4	RW	
5	MPLLCON	4C000004	0005C042	4	R/W	
6	BWSCON	48000000	22111120	4	RW	
7	BANKCONB	48000004	00000700	4	R/W	
8	BANKCON1	48000008	00000700	4	RW	
9	BANKCON2	480000DC	00000700	4	RW	
10	BANKCON3	48000010	00000700	4	R/W	
11	BANKCON4	48000014	00000700	4	RW	-
12	BANKCON5	48000018	00000700	4	R/W	
13	BANKCON6	4800001C	00018005	4	RW	
14	BANKCON7	48000020	00018005	4	RW	
	DEEDEQU	49000024	00000400	.4	DAV	~

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

	Config target			
	Device : SMDK2440	▼ Set SMU		
	Base Addr 30000000 Size 03FFFFF	Endian C Big C Little		
File 1 File 2 File 3 File 4(Length(4byte)+File+CheckSum) Image: Download Flash device Image:				
	Target Image File: C:\WORK\2440\2440USB_Mon\u2440mi	on.bin		
	0	K Cancel		
Device: SMDK	ked			
AM Information	on: Base Address:30000000	Size: 3FFFFF		

- Endian: Little

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

- Target Image File: u2440mon.bin

Target Address:0

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

Flash UP	<u> </u>
Ele Litiny Flash About	
इ हि 💣 🔏 🛹 😑 🔩	
* Current config *	
Device : SMDK2440 RAM base address : 0x0000000 RAM size : 0x00FFFFFF Byte sex : Little endian Filei config Flash name : INTEL_20F12&JIA Erase : All Data hos width : 32 Bit Flash base address : 0x00000000 Target address : 0x00000000 Target image file : C1\W00K\2440VSB_Mon\u2440msm.bin	
	Disconnect



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

Flash UP	8
Elle Utilty Fjach About	
頁音 🖌 🦧 🥪 🖯 🛃	
*****	6
* Current config *	
Bevice : SRDK2440	
RAM base address : 0x30000000	
RAN SIZE : ONDUFFTT	
Byte sex : Little endian	
File1 config	
Flash name IFTEL_20F128J3A Erase : All	
Data bur width : 32 Bit	
Flash base address : 0x00000000	
Target address : 0x00000000	
Target image file : C:\WORK\2440\2440USB_Mon\u2440mon.bin	
FileHens: u244Omon.bin	
FileSize: 35412	
Chip erase start	
Erase complete!!! u2440mon.bin downloading	
urasimon.sin downloading	
u2440mon.kin program complete !!!	
c.	2
	Connect
	Comer

4 SYSTEM DESIGN

OVERVIEW

The S3C2440A, SAMSUMG's 16/32-bit RISC microcontroller is cost-effective and high performance microcontroller solution for hand-held devices and general applications. The S3C2440A has the following integrated on-chip functions:

- 1.2V/1.3V int., 1.8V/2.5V/3.3V memory, 3.3V external I/O microprocessor with 16KB I-Cache/16KB D-Cache/MMU
- External memory controller (SDRAM control and Chip select logic)
- LCD controller (up to 4K color STN and 256K color TFT) with 1-ch LCD-dedicated DMA
- 4-ch 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
- AC97 Audio CODEC Interface
- 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 S3C2440A

The S3C2440A, SAMSUMG's 16/32-bit RISC microcontroller offers various functions and high efficiencies. In addition to the high performance, the S3C2440A offers low current consumption, ensuring low costs. The followings are sample applications that can be designed with the S3C2440A:

- 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



4-2

MEMORY INTERFACE DESIGN

BOOT ROM DESIGN

After the system reset, the S3C2440A accesses 0x0000_0000 address and configuring some system variables. Therefore, this special code (boot ROM image) should be located on the address 0x0000_0000. Bus width of boot ROM can be selected by setting OM[1:0] pins.

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

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

NAND BOOT DESIGN

Figure 4-1 shows a design with NAND boot.

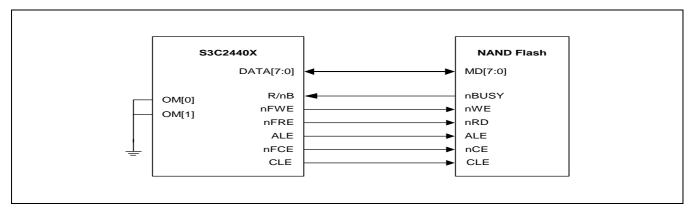


Figure 4-1. NAND Boot Design

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.

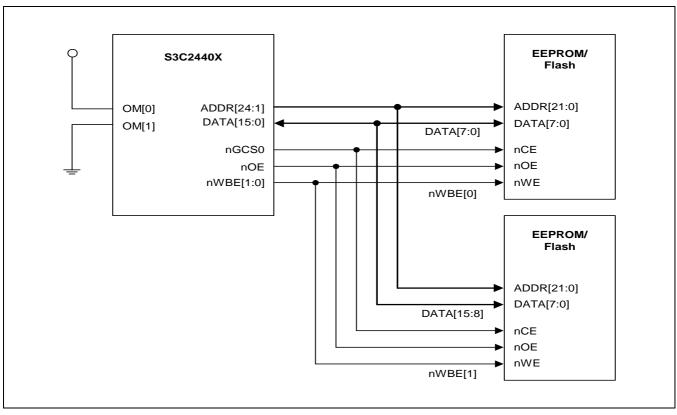


Figure 4-2. Half-word Boot ROM Design with Byte EEPROM/Flash

MAKING HALFWORD ROM IMAGE WITH BYTE EEPROM/FLASH

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

	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.

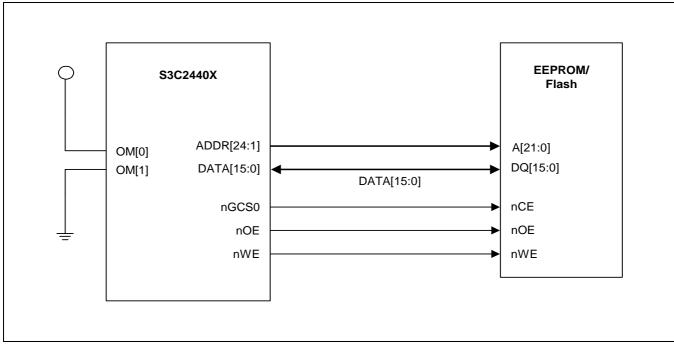


Figure 4-3. The Halfword Boot ROM Design with Halfword EEPROM/Flash



WORD BOOT ROM DESIGN WITH BYTE EEPROM/FLASH

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

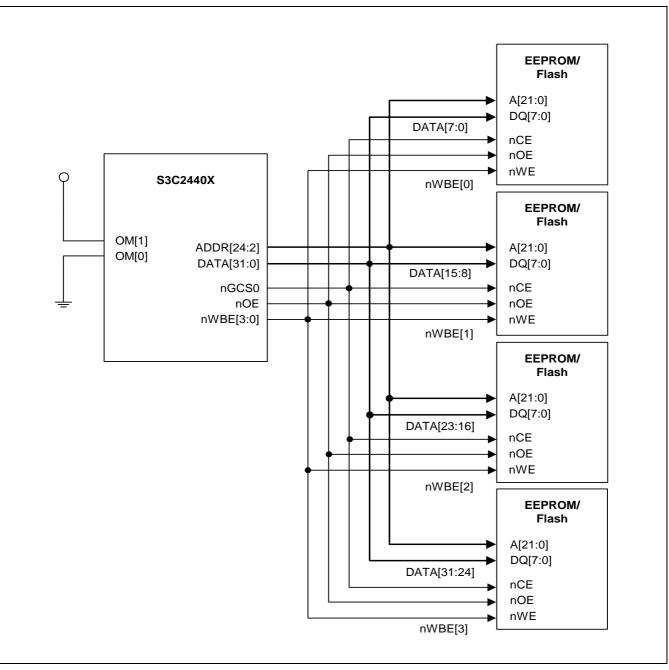


Figure 4-4. The Word Boot ROM Design with Byte EEPROM/Flash



MAKING WORD ROM IMAGE WITH BYTE EEPROM/FLASH

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

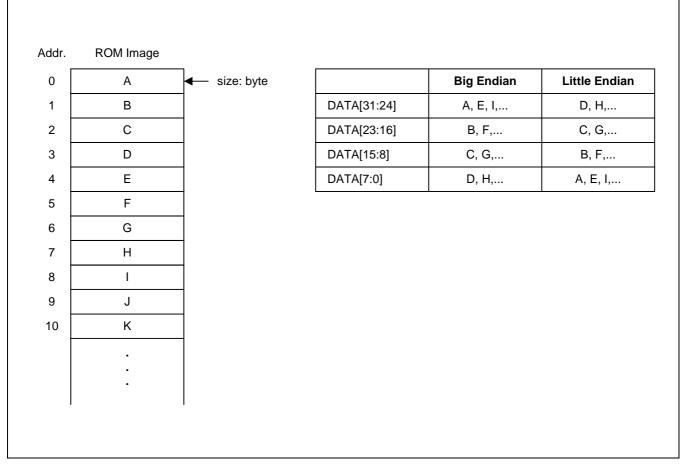


Figure 4-5. Relationship of ROM Image and Endian



MEMORY BANK DESIGN AND CONTROL

The S3C2440A has six ROM/SRAM banks (including BANK0 for boot ROM) and two ROM/SRAM/SDRAM banks. The system manager on the S3C2440A 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				
0		,	;End address of SMRDATA		
U		r3, [r1], #4 r2, r0			
	-				
	-				
	•				
SMR	ΔΑΤΑ				
	DCD	0x22111120			
	DCD	0x00000700	;GCS0		
	-	0x00000700	;GCS1		
	-	0x00000700	;GCS2		
	-	0x00000700	;GCS3		
		0x00000700	;GCS4		
		0x00000700	GCS5		
		0x00018005 0x00018005	;GCS6 SDRAM(Trcd=3,SCAN=9) ;GCS7 SDRAM(Trcd=3,SCAN=9)		
			;GCS7 SDRAM(TTCd=3,SCAN=9) 3;Refresh(REFEN=1,TREFMD=0,Trp=2 clk,		
			; Trc=7 clk, Tchr=3 clk,Ref CNT)		
	DCD	0x32	;Bank size, 128MB/128MB		
	DCD		;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.

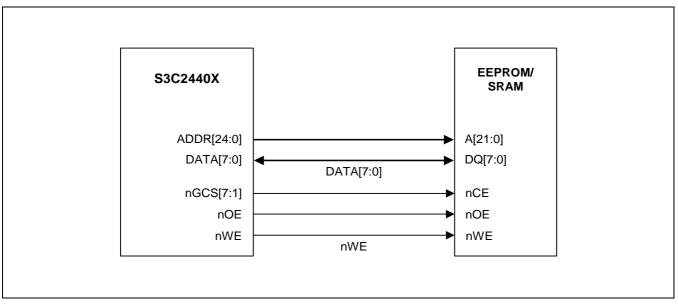
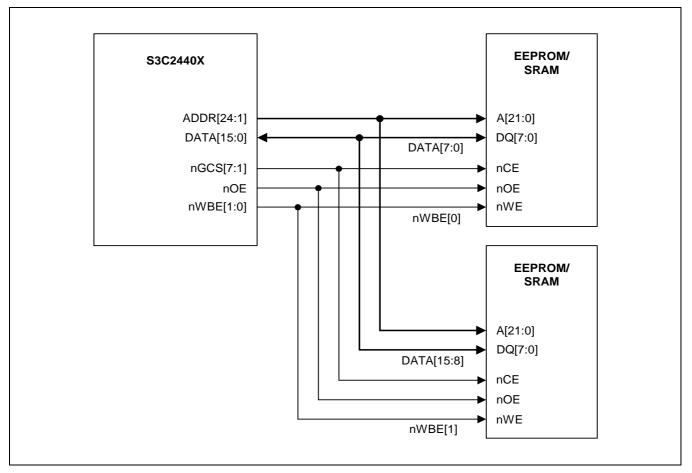


Figure 4-6. One-byte EEPROM/SRAM Bank Design







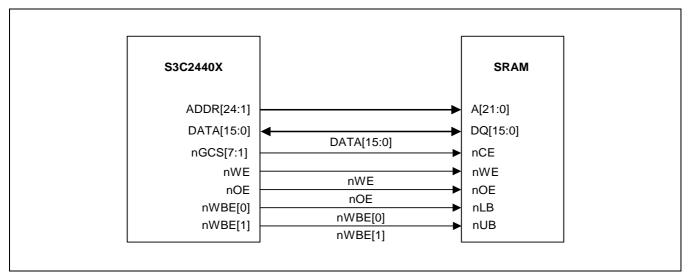


Figure 4-8. Halfword SRAM Bank Design with Halfword SRAM



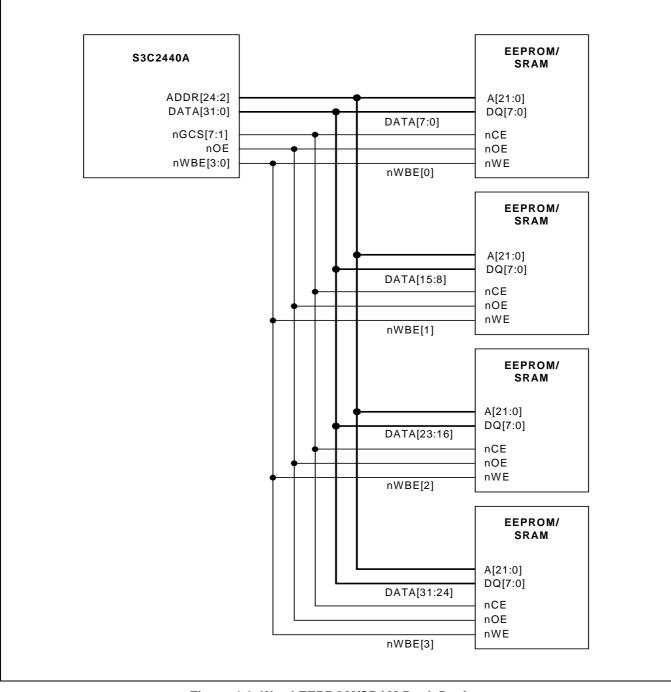


Figure 4-9. Word EEPROM/SRAM Bank Design



SDRAM BANK DESIGN FOR S3C2440A

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	η Γ	(4M x 8 x 2B) x 4	A24
	x32	η Γ	(2M x 8 x 4B) x 4	A[24:23]
	x16	128Mb	(4M x 8 x 4B) x 2	
	x32	η Γ	(2M x 16 x 4B) x 2	
	x8	256Mb	(8M x 8 x 4B) x 1	
	x16	η Γ	(4M x 16 x 4B) x 1	

Table 4-3. SDRAM Bank Address configuration



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	

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

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.

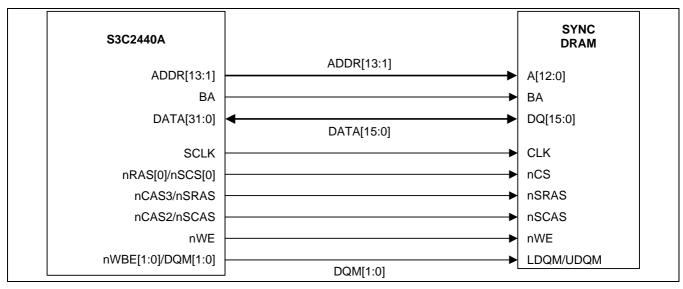


Figure 4-10. Halfword SDRAM Design with Halfword Component



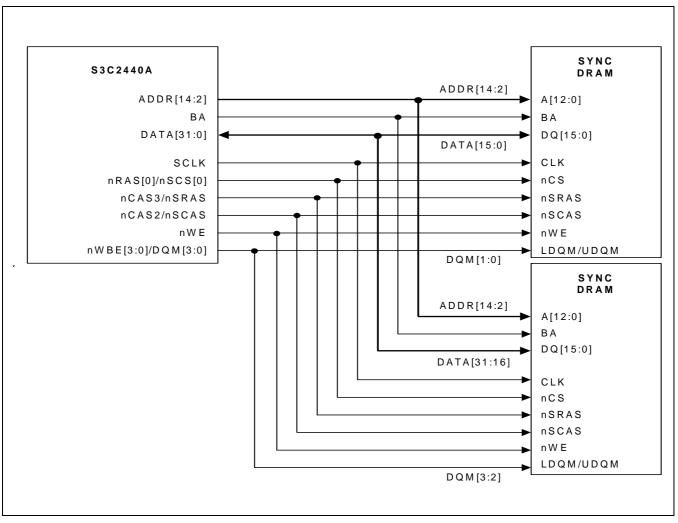


Figure 4-11. Word SDRAM Design with Half-word Component



PC CARD (PCMCIA) INTERFACE APPLICATION USING CL-PD6710 (CIRRUS LOGIC)

The PC card (PCMCIA card) can be interfaced with S3C2440A using following components:

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

We tested the PC 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 PC Card program

•••• DNW v0,49 [COM1,115200bps][USB:x]	
Serial Port USB Port Configuration Help	
[PD6710 test for reading pc_card CIS]	_
Insert PC card!!!	
PC card interrupt is occurred.	
PC card interrupt is occurred.	
Card is inserted. 3.3V card is detected.	
PC card interrupt is occurred.	
[Card Information Structure]	
cisEnd=0~a6	
1, 4,df,4a, 1,ff,1c, 4, 2,d9, 1,ff,18, 2,df, 1,//J	
20, 4, 7,c0, 0, 0,15,20, 4, 1,53,41,4d,53,55,4e,//SAMSUN	
47,20,20,20,20,20,0,53,43,46,43,2d,56,45,52,//G .SCFC-UER	
31,2e,30,20,20, 0, 0, ff,21, 2, 4, 1,22, 2, 1, 1,//1.0!	
22, 3, 2, c, f,1a, 5, 1, 3, 0, 2, f,1b, 8,c0,c0,//" a1, 1,55, 8, 0,20,1b, 6, 0, 1,21,b5,1e,4d,1b, a,//U!M	
c1,41,99, 1,55,64,f0,ff,ff,20,1b, 6, 1, 1,21,b5,//.AUd	
1e,4d,1b, f,c2,41,99, 1,55,ea,61,f0, 1, 7,f6, 3,//.MAU.a	
1,ee,20,1b, 6, 2, 1,21,b5,1e,4d,1b, f,c3,41,99,//!MA.	
1,55,ea,61,70, 1, 7,76, 3, 1,ee,20,1b, 6, 3, 1,//.U.apv	
21,b5,1e,4d,14, 0,ff,	-

Figure 4-12. PC Card CIS Access Example on S3C2440A



10BASE-T ETHERNET CONTROLLER (CS8900A) INTERFACE

The 10BASE-T Ethernet can be supported on S3C2440A using following components:

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

IIS AUDIO CODEC (UDA1341TS) CONNECTION WITH S3C2440A

The S3C2440A 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.

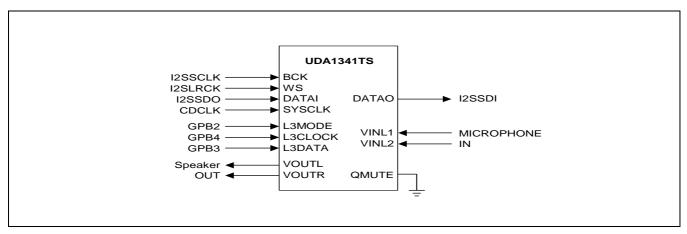


Figure 4-13. UDA1341TS Connection with S3C2440A

AC97 AUDIO CODEC (STAC9767) CONNECTION WITH S3C2440A

The S3C2440A AC97 interface example circuit is as follows:

- STAC9767 from SIGMATEL.
- The AC97 interface (AC_SYNC, AC_BIT_CLK, AC_nRESET, AC_SDATA_IN and AC_SDATA_OUT) is realized between S3C2440A and STAC9767.
- Refer to the sample code of audio application which plays GPCM file.



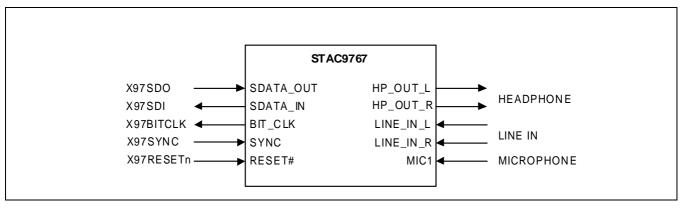


Figure 4-14. STAC9767 Connection with S3C2440A

LCD CONNECTION WITH S3C2440A

The S3C2440A 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.



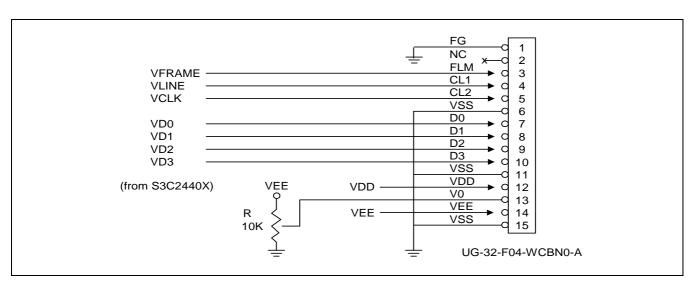


Figure 4-15. UG-32F04 Connection with S3C2440A (320x240 Mono STN LCD)

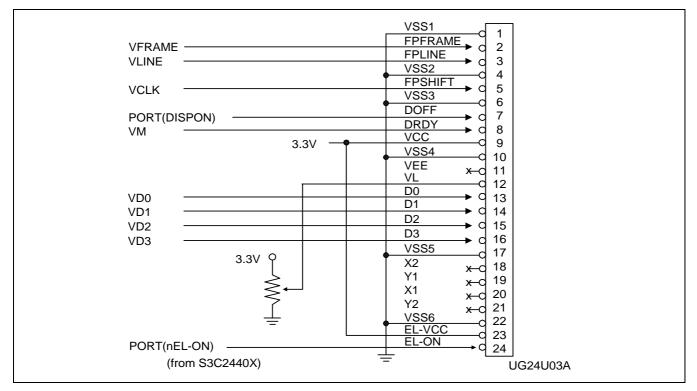


Figure 4-16. UG24U03A Connection with S3C2440A (320x240 Mono STN LCD)



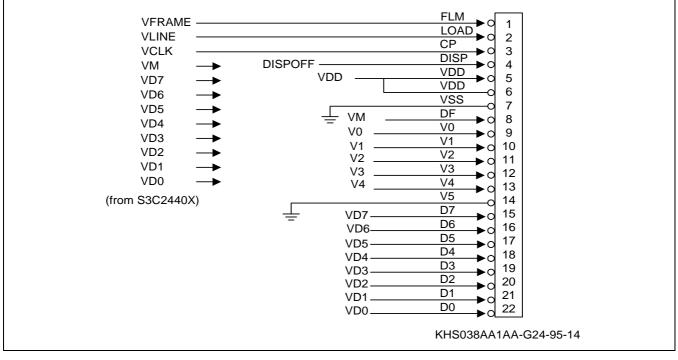


Figure 4-17. KHS038AA1AA-G24 Connection with S3C2440A (256 Color STN LCD)



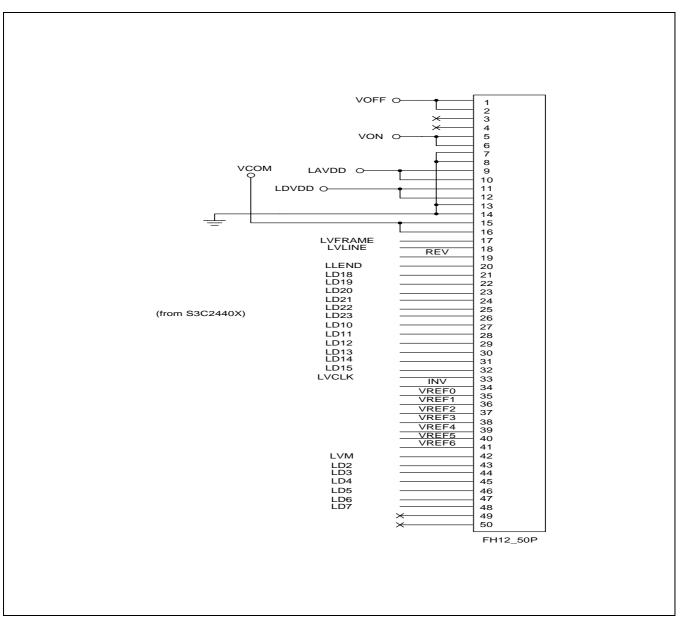


Figure 4-18. LTS350Q1-PE1 Connection with S3C2440A (Samsung 3.5" Transflective TFT LCD)



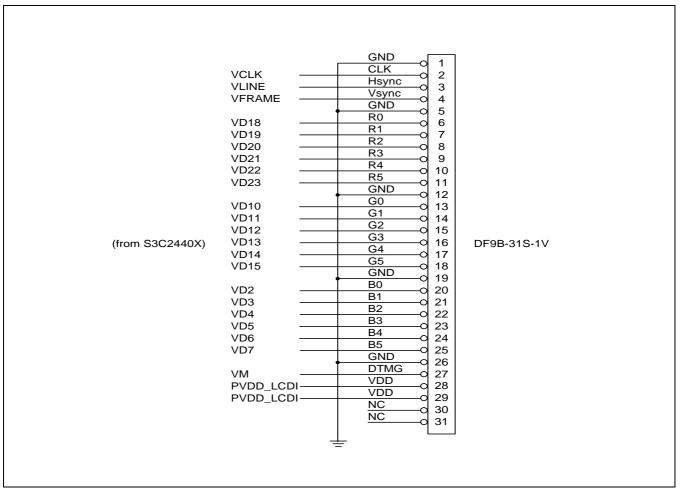


Figure 4-19. LP104V2-W Connection with S3C2440A (LG Philips 10.4" TFT LCD)



	GND 0 1 CLK 2 2
VCLK	
HSYNC	
VSYNC	GND
VD11	
VD12	R2 0 8 R3 0 8
VD13	
VD14	R5 0 11
VD13	GND 0 12
VD5	$\begin{array}{c} G0 \\ G1 \end{array} \xrightarrow{12} 0 13 \end{array}$
VD6	→ 0 14
VD7	
VD8	G4 J 17
VD9 VD10	G5 18
	GND 19
VD0	P G = ·
VD1 VD2	B3 5 22
VD3	
VD4	
VDEN	DENB D 20
VDD_LCDI(5V)	
	U/D 0 31
(from S3C2440X)	<u> </u>

Figure 4-20. V16C6448AB Connection with S3C2440A (TFT LCD)



SYSTEM DESIGN WITH DEBUGGER SUPPORT

MULTI-ICE

The S3C2440A has an Embedded ICE logic that provides debug solution from ARM. MULTI-ICE enables you to debug software running on the S3C2440A. Embedded ICE logic is accessed through the Test Access Port (TAP) controller on the S3C2440A using the JTAG interface.

JTAG port for Embedded ICE Interface

When you build a system with the S3C2440A 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.

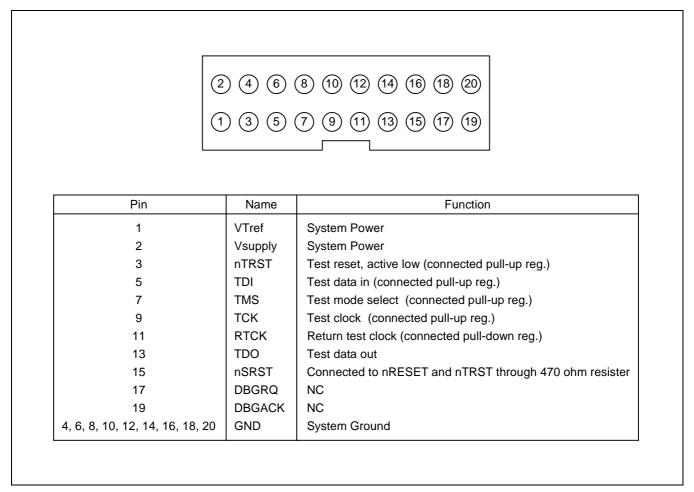


Figure 4-21. MULTI-ICE Interface of JTAG Connector



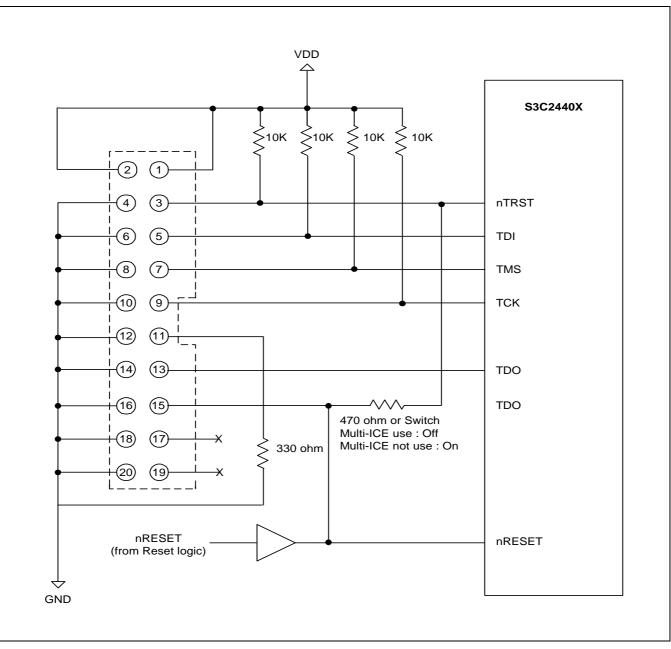


Figure 4-22. MULTI-ICE Interface Design Example



CHECK ITEMS FOR SYSTEM DESIGN WITH S3C2440A

When you design a system with the S3C2440A, 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, XTIpII has to be connected to VDD. If XTIpII 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, S3C2440A may not operate.



NOTES



4-26

5 DVS(DYNAMIC VOLTAGE SCALING)

OVERVIEW

DVS(Dynamic Voltage Scaling) is useful to reduce power consumption in Idle mode.

The basic concept of DVS is to drop the Core and Internal voltage when those blocks don't need to operate heavily and reduce the power consumption.

There are two methods to reduce power consumption; one is drop the voltage while the internal blocks are not working or operating slowly though the system is running. The other is lengthening the system clock speed to reduce power consumption.

DVS uses the two methods, voltage scaling and change clocking.

When DVS is used, the Core power consumption can be reduced maximum by 50% of the core current.

While WMA is playing, Idle state rate is more than 80% and actual CPU operating rate is about 20%. If DVS is applied to this application, power can be saved during 80% Idle period.



POWER SCHEME FOR DVS

Applicable DVS power supply pins are VDDi(Internal block power) and VDDiarm(ARM920T power). To use DVS, the system power has to be supplied two variable voltages. One for normal operation, the other for lower level voltage (for DVS).

The DVS High and Low voltage is as follows.

DVS Pins	Voltage spec.	Normal operating voltage	DVS low voltage
VDDiarm	300Mhz: 1.2V(1.15V ~ 1.25V) 400Mhz: 1.3V(1.25V ~ 1.35V)	300Mhz: 1.2V 400Mhz: 1.3V	Min 1.0V
VDDi	300Mhz: 1.2V(1.15V ~ 1.25V) 400Mhz: 1.3V(1.25V ~ 1.35V)	300Mhz: 1.2V 400Mhz: 1.3V	Min 1.0V

Table 5-1. DVS voltage level

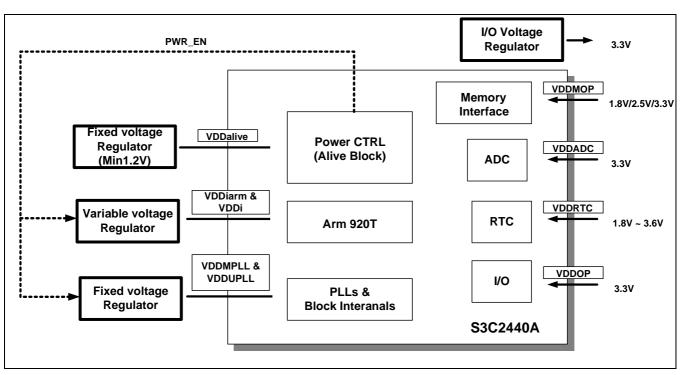


Figure 5-1. Power Scheme for DVS

The DVS scheme can be applied only for VDDiarm, but we strongly recommend to use both VDDiarm and VDDi voltage. Refer the following Schematic diagram for DVS.



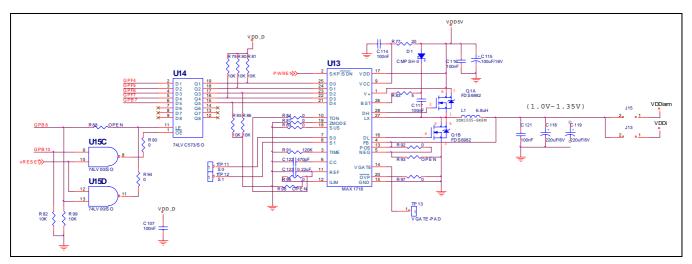


Figure 5-2. Power regulator example diagram for DVS



DVS OPERATING DETAILS

DVS OPERATIG FLOW CHART

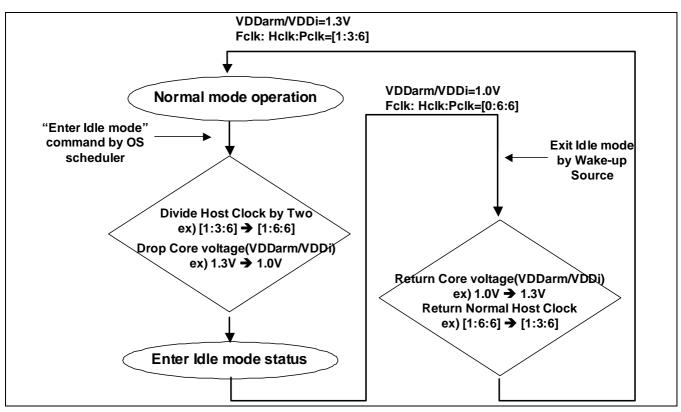


Figure 5-3. DVS Flow chart example(400Mhz)

Figure 5-3 shows the DVS scheme for 400Mhz, in 300Mhz only the High voltage will be 1.2V.



DVS SHCEME DESCRIPTION

Please contact the Samsung CS team for Detailed DVS scheme.



POWER CONSUMPTION OF DVS

Table 5-2 shows how much the power consumption will be reduced when using DVS.

FCLK [Mhz]	DVS Scheme application (VDDiarm/VDDi)	Core Power [mW]		Difference Without → with
		With DVS	Without DVS	
300	Running mode	51	74	+23mW(45%)
	Idle mode	32	64	+32mW{100%}
400	Running mode	68	139	+71mW(104%)
	Idle mode	46	113	+67mW(146%)

Table 5-2. Core current consumption

(Note) Test condition

- Core current = I_VDDi + I_VDDiarm at 1.2V @300Mhz, 1.3V @400Mhz. (Current of VDDalive/VDDUPLL/VDDMPLL are not included).

- For DVS the Core voltage will be down to 1.0V

(1) Execute Batlife.wma file on PPC2003.

(2) No threads ready to run on PPC2003

(3) FCLK:HCLK:PCLK = (300:50:50) and (0:50:50) Mhz for 300Mhz, (400:67:67) and (0:67:67) Mhz for 400Mhz.

(4) FCLK:HCLK:PCLK = (300:100:50), (400:133:67) MHz

Sample # : KYC13AA

OS timer scheduler: 1msec~10msec.

As the upper table, Using DVS, the core power consumption will be reduced quite much

The followings are the 2440 total power consumption comparison between DVS is on and off.

