2 HOW TO USE SNDS100 BOARD?

SETUP SNDS100 ENVIRONMENTS

The evaluation environments for SNDS100 are shown in Figure 2-1. Serial port(SIO-0) on SNDS100 have to be connected to COM port of Host PC . This is can be used as console for monitoring and debugging SNDS100.

If you have the emulator like as EmbeddedICE, you can use JTAG port on SNDS100 as the interface for it. Evaluating the 10/100M ethernet function , RJ45 connector on SNDS100 can be connected to Host PC or HUB and Switch. Connected to Host PC, 10/100M ethernet cable has crossed wire connections because of RJ45 connections for SNDS100 were designed to adapter side. In case of HUB/Switch, SNDS100 can be connected to it directly.

DC power adapter which have more then DC6V/800mA output characteristics can be used as input power of the SNDS100 board. This input power regulated to 3.3V and 5.0V for CPU and peripheral device on SNDS100 board.

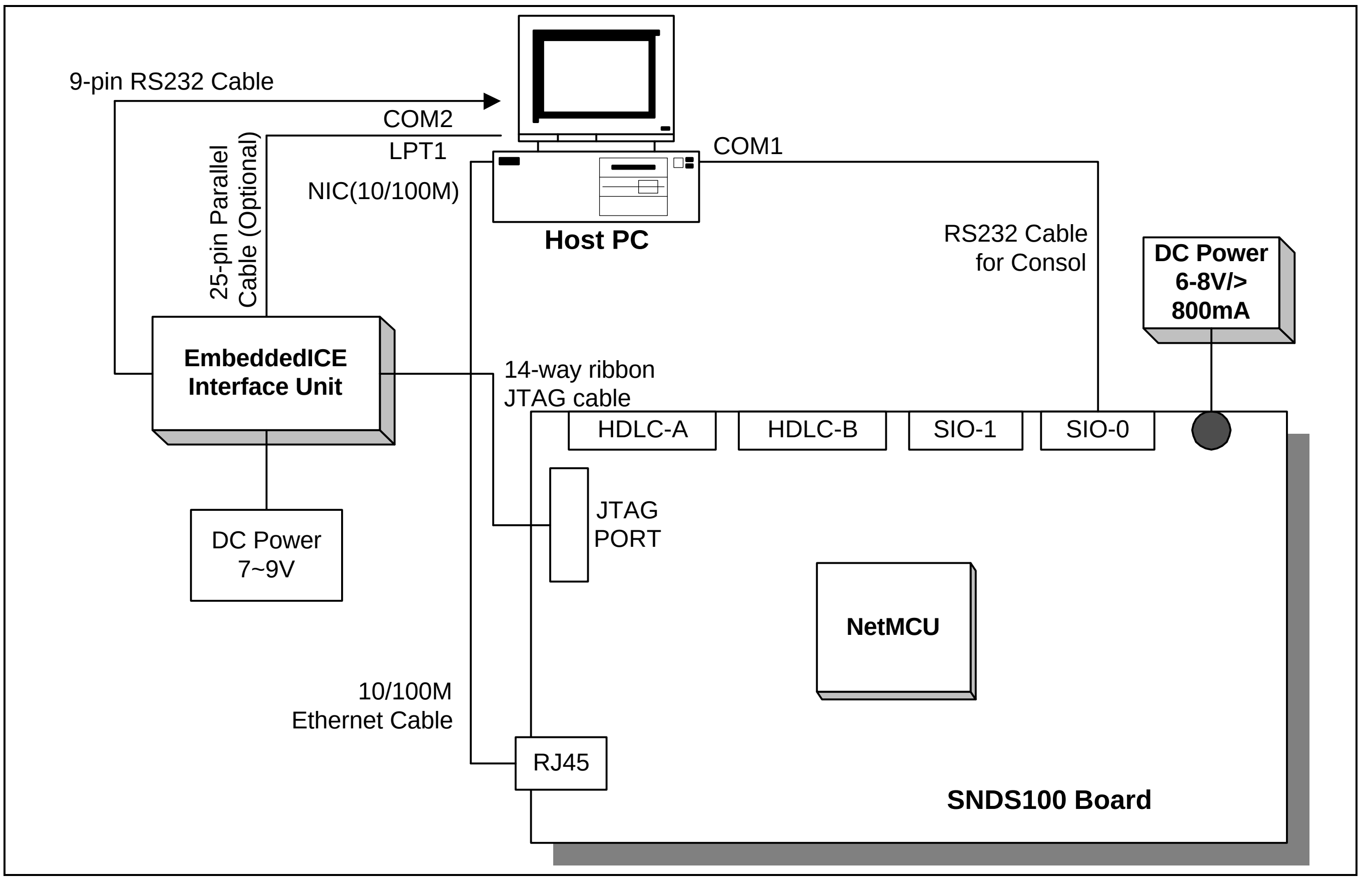
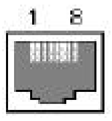


Figure 2-1. Setup Environments for SNDS100 Board

ETHERNET 10/100 BASE-T CONNECTOR

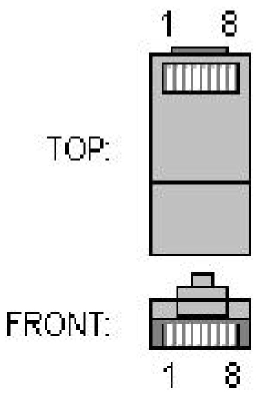
Same connector and pinout for both 10Base-T and 100Base-Tx.



(At the network interface cards/hubs)

(At the cables)

RJ45 FEMALE CONNECTOR at the network interface cards and hubs .
RJ45 MALE CONNECTOR at the cables.



Pin	Name	Descriptions
1	TX+	Transmit Data+
2	TX-	Transmit Data-
3	RX+	Receive Data+
4	N/C	Not Connected
5	N/C	Not Connected
6	RX-	Receive Data-
7	N/C	Not Connected
8	N/C	Not Connected

NOTE: TX & RX are swapped on Hub

CONNECTION METHODS FOR UTP CABLE

RJ45 pins on SNDS100 is defined to Adapter side. So, you can straight connect SNDS100 to HUB through UTP cable. In this case, between the SNDS100 and Hub, the pin numbers are correspond to each others.

Between the SNDS100 board and NIC which is on Host PC, you have to connect each other through UTP cable which is crossover patch cord. See Figure 2-2 (b)

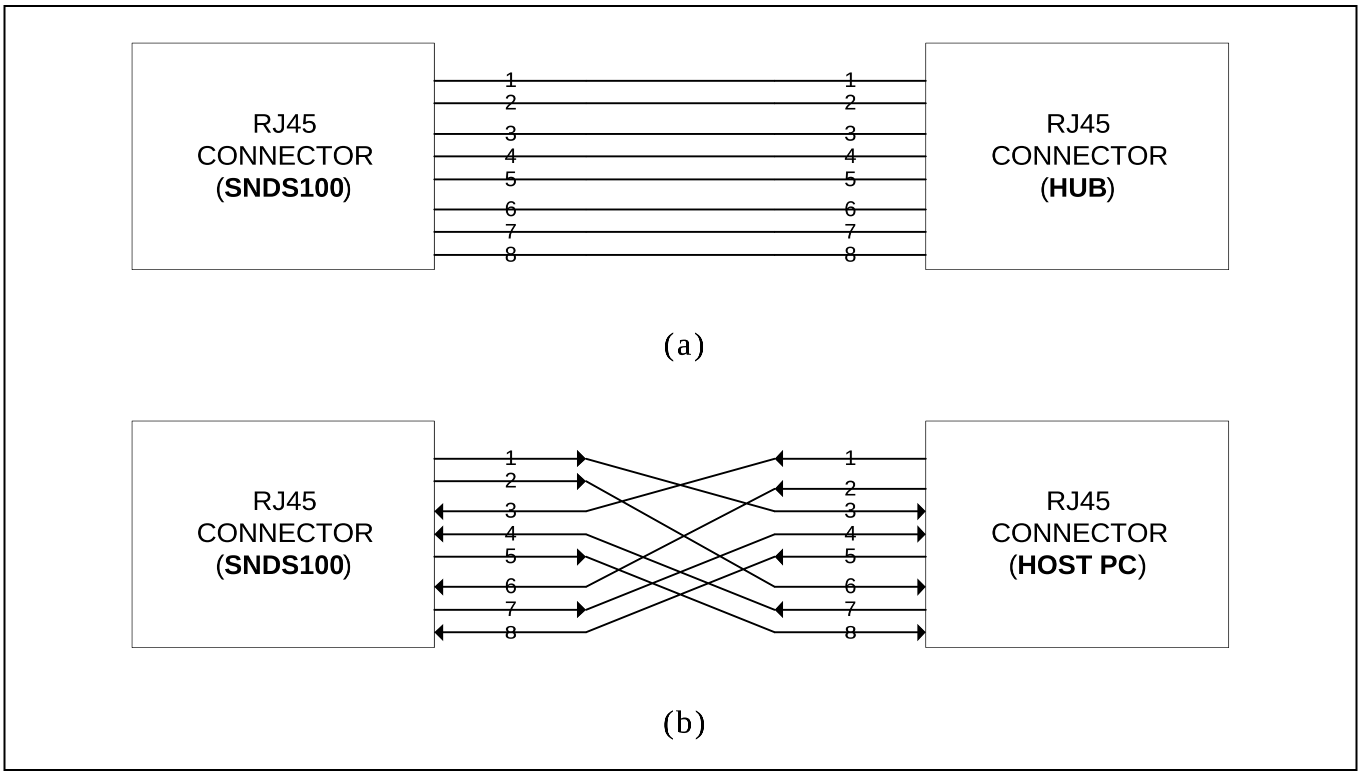


Figure 2-2. UTP Cable Connections

CONNECTION CONFIGURATIONS FOR DEBUG CONSOL

UART channel 0(SIO-0) on SNDS100 is assigned to Debug Console port. Using this port, you can download executable image code to DRAM memory from DOS command windows of Host PC . Also you can monitoring and debugging the device from this port with Hyper terminal which is windows utility program supported by windows 95 and windows NT.

You can communicate through the RS232 cable from UART channel 0 to COM1 or COM2 which is serial port on Host PC. SNDS100 supplies the 9pin D-SUB male connector for communication channel. Detail pin configurations for connecting each others as console port be given as bellows:

Table 2-1. Pin connection configurations for Console port.

SNDS100	**DIR**	**HOST PC**		**DESCRIPTIONS**
9Pin D-SUB MALE		**9Pin D-SUB MALE**	**25Pin D-SUB MALE**	
2. RXD	←	2. RXD	3. RXD	Receive Data
3. TXD	→	3. TXD	2. TXD	Transmit Data
4. DTR	→	4. DTR	20. DTR	Data Terminal Ready
5. DSR	←	5. DSR	6. DSR	Data Set Ready

CONFIGURING THE HYPER TERMINAL (ON WINDOWS 95 OR NT)

To configure the Hyper Terminal which is windows utility program for serial communications be given on windows 95 or NT, please following steps:

1. Running the Hyper Terminal utility program.

   Window 95 or windows NT start tool bar → Program → Accessories → Hyper Terminal Group → Double click Hyperterm.exe → Enter connection name → Select icon → Click OK.

2. Select COM port to communicate with SNDS100 target board.

   Choose COM1 or COM2 as the serial communication port → Click OK

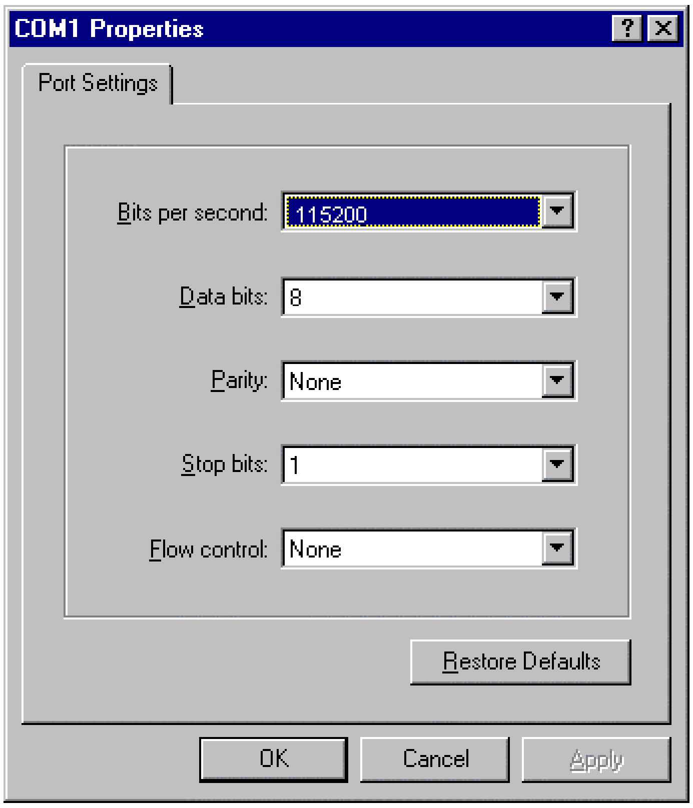
3. Set the serial port properties. (See the Figure 2-3.)

— Bits per second : 115200 bps

— Data bits : 8 bits

— Stop bits : 1

— Flow control : None



Bits per second: 115200

Data bits: 8

Parity: None

Stop bits: 1

Flow control: None

Figure 2-3. Setting Port Properties

CONFIGURING THE HYPER TERMINAL (ON WINDOWS 95 OR NT) (Continued)

4. Select Properties menu

   File → Properties.

5. Choose Setting Page (Figure 2-4)

6. Click ASCII Setup button

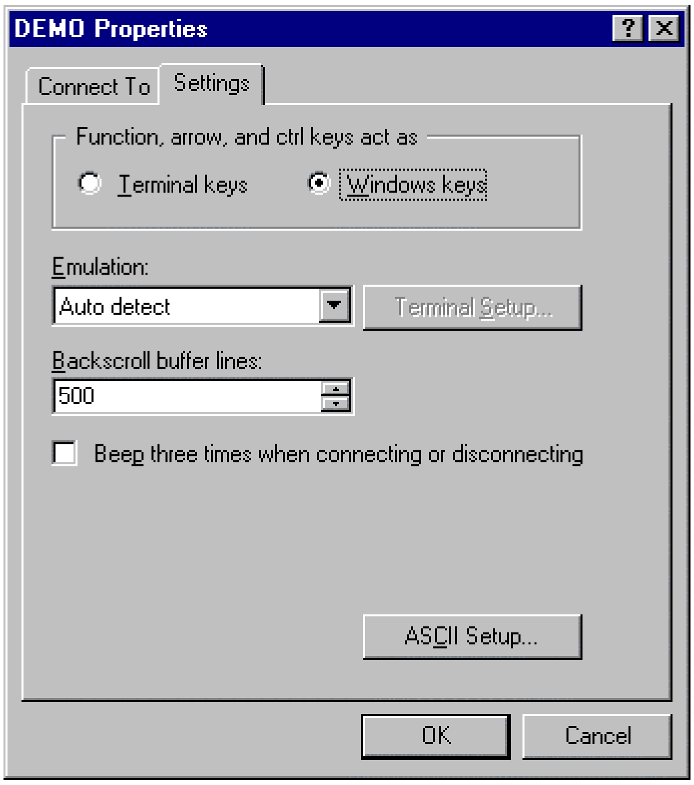
7. Check ASCII Setup menus and set (Figure 2-5)

8. Re-connect Hyper-Terminal to run at new properties

— Disconnect: Call → Disconnect
— Connect : Call → Call

9. Power-On Reset or Push the reset button on SNDS100 board

— Now, The diagnostic menu is showed on the Hyper-Terminal (Refer to Figure 2-6).



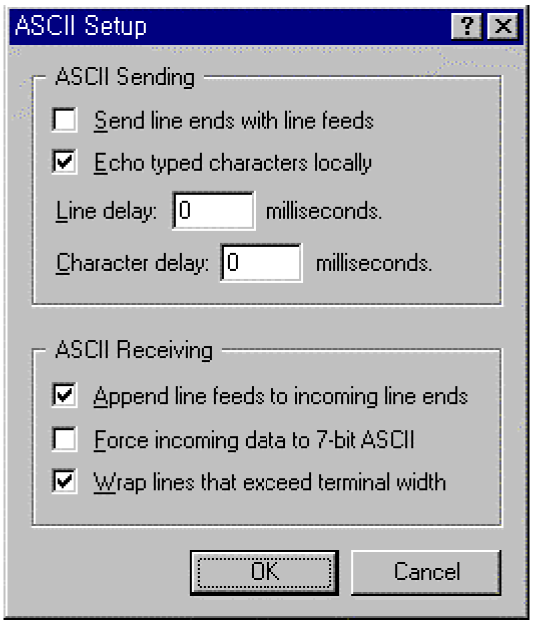
Function, arrow, and ctrl keys act as
( ) Terminal keys [✓ Windows keys]

Emulation: Auto detect

Backscroll buffer lines: 500

☐ Beep three times when connecting or disconnecting

Figure 2-4. Choose Setting Page



ASCII Setup

ASCII Sending

- [ ] Send line ends with line feeds
- [x] Echo typed characters locally

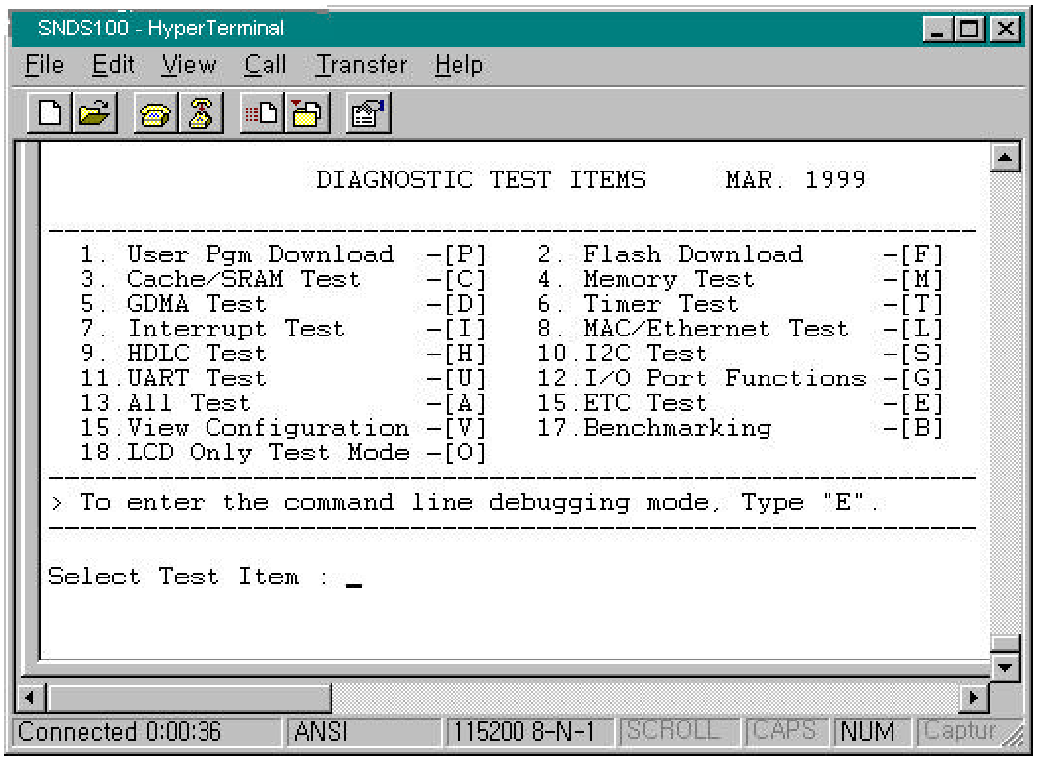
Line delay: 0 milliseconds.

Character delay: 0 milliseconds.

ASCII Receiving

- [x] Append line feeds to incoming line ends
- [ ] Force incoming data to 7-bit ASCII
- [x] Wrap lines that exceed terminal width

Figure 2-5. ASCII Setup Menu



```
                 DIAGNOSTIC TEST ITEMS     MAR. 1999

-----------------------------------------------------------
  1. User Pgm Download    -[P]    2. Flash Download      -[F]
  3. Cache/SRAM Test      -[C]    4. Memory Test         -[M]
  5. GDMA Test            -[D]    6. Timer Test          -[T]
  7. Interrupt Test       -[I]    8. MAC/Ethernet Test   -[L]
  9. HDLC Test            -[H]   10.I2C Test             -[S]
 11.UART Test             -[U]   12.I/O Port Functions   -[G]
 13.All Test              -[A]   15.ETC Test             -[E]
 15.View Configuration    -[V]   17.Benchmarking         -[B]
 18.LCD Only Test Mode    -[O]
-----------------------------------------------------------
> To enter the command line debugging mode, Type "E".
-----------------------------------------------------------

Select Test Item : _
```

Connected 0:00:36 | ANSI | 115200 8-N-1 | SCROLL | CAPS | NUM | Captur

Figure 2-6. SNDS100 Diagnostic Window After Reset

HOW TO BUILD EXECUTABLE DOWNLOAD IMAGE FILE

Execution image file can be built using by ARM Project manager or makefile. If you want to use ARM debugger, you have to build ARM image format(AIF). Any other case, if you want to use only Console debugging SNDS100 board without ARM debugger, then Binary image file have to be prepared to download directly and execute on Target board.

First of all, you must download SNDS100.zip which is evaluation source included boot code from our web site(www.samsungsemi.com) and also any other utilities and follow up the bellows. It will be helpful to more easily understand the development environments of NetMCU series.

USING ARM PROJECT MANAGER

ARM SDT version 2.11a were installed and used for the following procedures.

Creating a New Project

To create a new project, following the steps:

1. Invoke ARM Project Manager

2. Generate a New Project

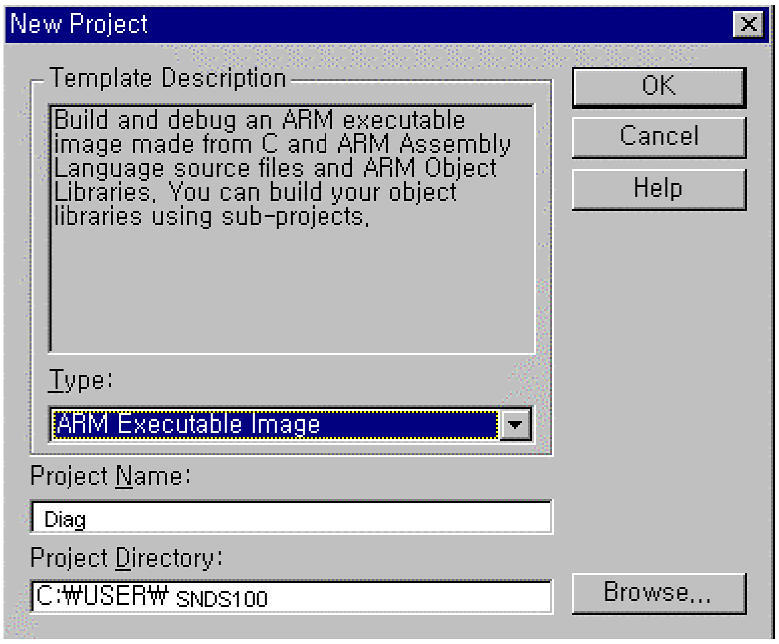
   File menu → New → select Project form the New dialog box

3. Edit New Project dialog box (Figure 2-7)

— Type : ARM Executable image
— Enter Project Name and Directory

4. When you have created a new project, the information in the Project Window is displayed in a hierarchical flow diagram. (Figure 2-8)

— Debug: for creating a target image suitable for debugging, which includes debugging information.
— Release: for creating a target image suitable for release, which includes debugging information.



New Project

Template Description

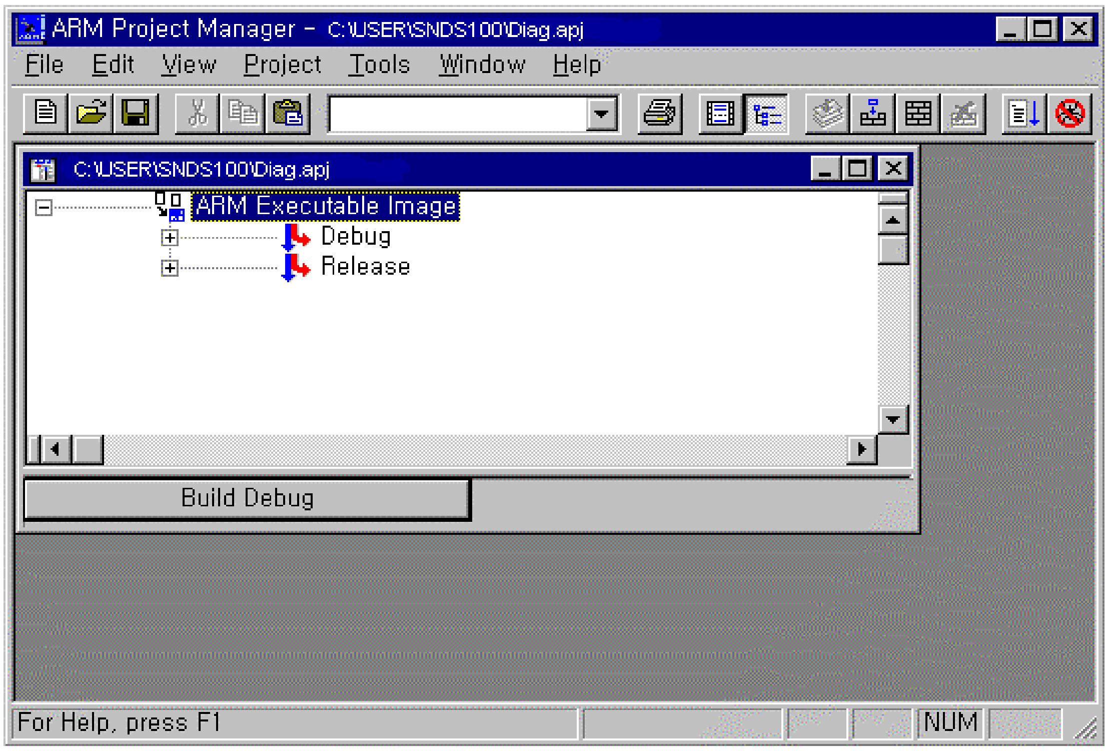
Build and debug an ARM executable image made from C and ARM Assembly Language source files and ARM Object Libraries. You can build your object libraries using sub-projects.

Type: ARM Executable Image

Project Name: Diag

Project Directory: C:₩USER₩ SNDS100

Figure 2-7. New Project Dialog Window



ARM Project Manager – C:\USER\SNDS100\Diag.apj

C:\USER\SNDS100\Diag.apj

Figure 2-8. Project Window

ADDING SOURCE FILES TO THE PROJECT

To add source files to the project, following the steps:

1. Select Add files to Project from the Project menu.
2. Select source files to add (*.c and *.s on working directory)

SETTING THE OPTIONS OF ARM PROJECT MANAGER

You should set the options on Tools menu for setting ARM Project Manager tool.For additional options and descriptions of compiler, assembler, and linker, please refer to the chapter1,2, and 3 of “ARM Software Development Toolkit Reference Guide”.

Setting C Compiler option

1. Select <cc>=armcc option.

   Tools -> Configure -> <cc>=armcc

2. If the modify warning dialog box pops up, read it, and click Yes.

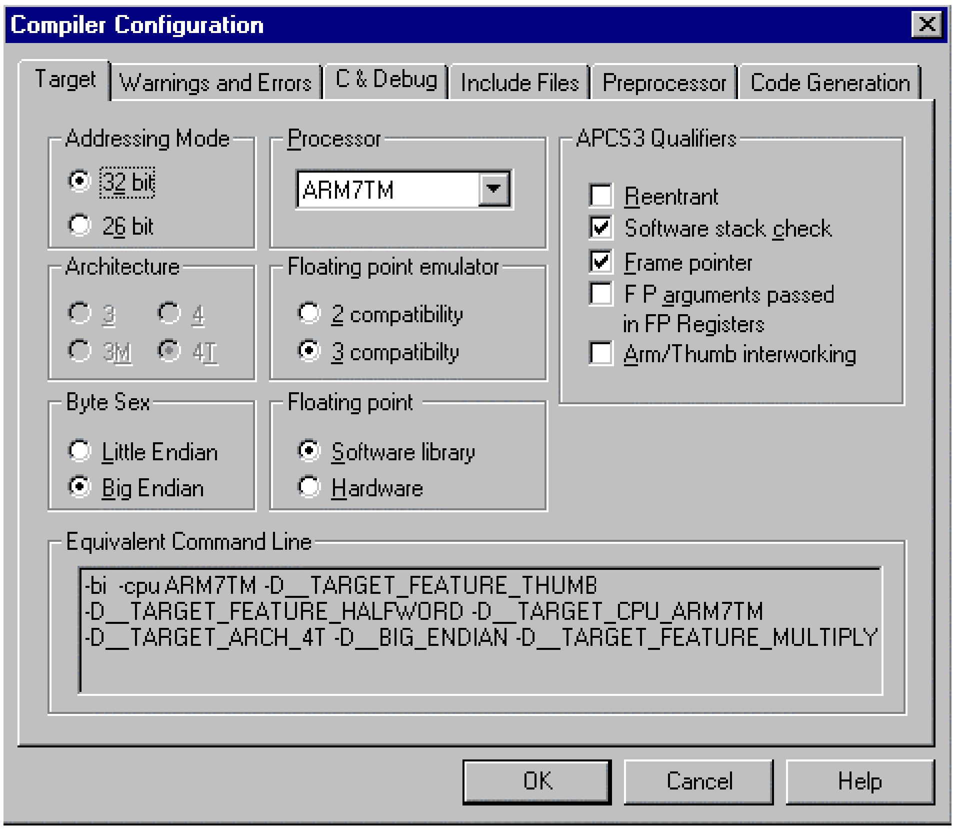
3. Modify Target page on Compiler Configuration dialog box as shown in Figure 2-9.

— Addressing Mode: 32 Bit
— Processor: ARM7TM
— Byte Sex: Big Endian

4. Write option to C&Debug page on Compiler Configuration dialog box as shown in Figure 2-10.

— Extra command line arguments: -fc

5. Click OK.



Addressing Mode: [✓ 32 bit] 26 bit

Processor: ARM7TM

APCS3 Qualifiers:
- [ ] Reentrant
- [✓] Software stack check
- [✓] Frame pointer
- [ ] F P arguments passed in FP Registers
- [ ] Arm/Thumb interworking

Architecture: 3 4 3M [✓ 4T]

Floating point emulator: 2 compatibility [✓ 3 compatibilty]

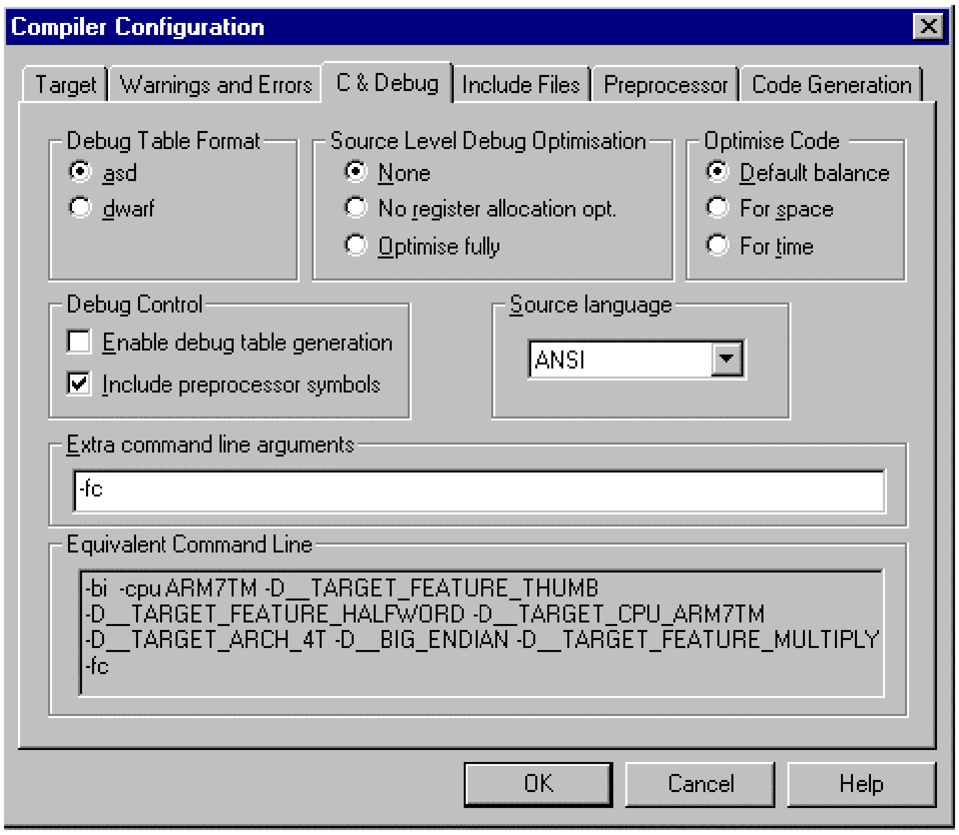
Byte Sex: Little Endian [✓ Big Endian]

Floating point: [✓ Software library] Hardware

Equivalent Command Line

```
-bi -cpu ARM7TM -D__TARGET_FEATURE_THUMB
-D__TARGET_FEATURE_HALFWORD -D__TARGET_CPU_ARM7TM
-D__TARGET_ARCH_4T -D__BIG_ENDIAN -D__TARGET_FEATURE_MULTIPLY
```

Figure 2-9. Compiler Configuration: Target Page


Compiler Configuration

Target | Warnings and Errors | C & Debug | Include Files | Preprocessor | Code Generation

Debug Table Format: [✓ asd] dwarf

Source Level Debug Optimisation: [✓ None] No register allocation opt. Optimise fully

Optimise Code: [✓ Default balance] For space For time

Debug Control:
- ☐ Enable debug table generation
- ☑ Include preprocessor symbols

Source language: ANSI

Extra command line arguments: -fc

Equivalent Command Line:

```
-bi  -cpu ARM7TM -D__TARGET_FEATURE_THUMB
-D__TARGET_FEATURE_HALFWORD -D__TARGET_CPU_ARM7TM
-D__TARGET_ARCH_4T -D__BIG_ENDIAN -D__TARGET_FEATURE_MULTIPLY
-fc
```

Figure 2-10. Compiler Configuration: C & Debug page

Setting Assembler Option

1. Select <asm>=armasm option.

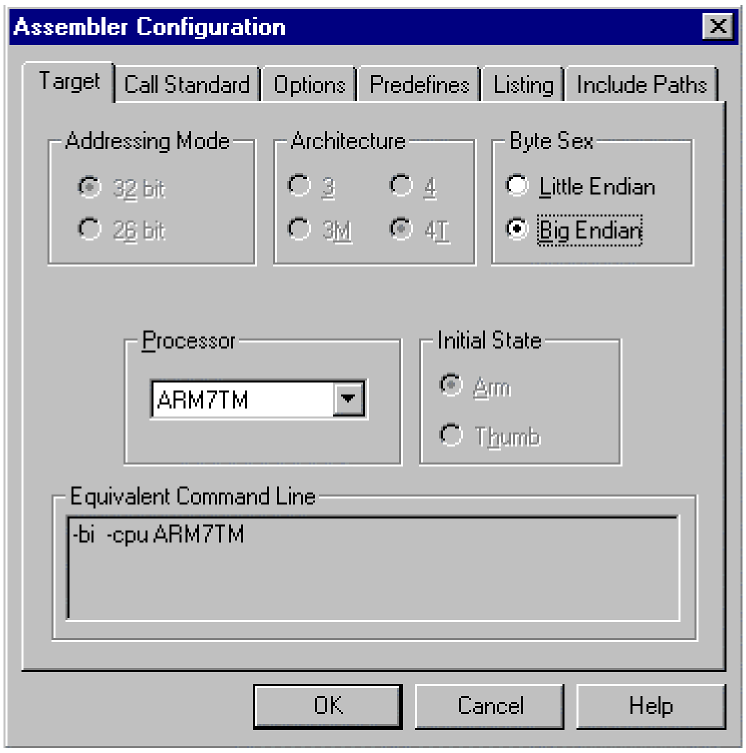
   Tools -> Configure -> <asm>=armasm

2. If the modify warnning dialog box pops up, read it, and click Yes.

3. Modify Target page on Assembler Configuration dialog box (Figure 2-10).

— Processor: ARM7TM
— Byte Sex: Big Endian

4. Click OK.



Addressing Mode: [✓ 32 bit] 26 bit
Architecture: 3 4 3M [✓ 4T]
Byte Sex: Little Endian [✓ Big Endian]
Processor: ARM7TM
Initial State: [✓ Arm] Thumb
Equivalent Command Line: -bi -cpu ARM7TM

Figure 2-11. Assembler Configuration

Setting Link Option

1. Select armlink option.

Tools -> Configure -> armlink

2. If the modify warning dialog box pops up, read it, and click Yes.

3. Modify Output page on Linker Configuration dialog box (Figure 2-12).

   Output Formats: Absolute AIF

4. Modify Entry and Base page on Linker Configuration dialog box (Figure 2-13).

— Read-Only: 0x1000050

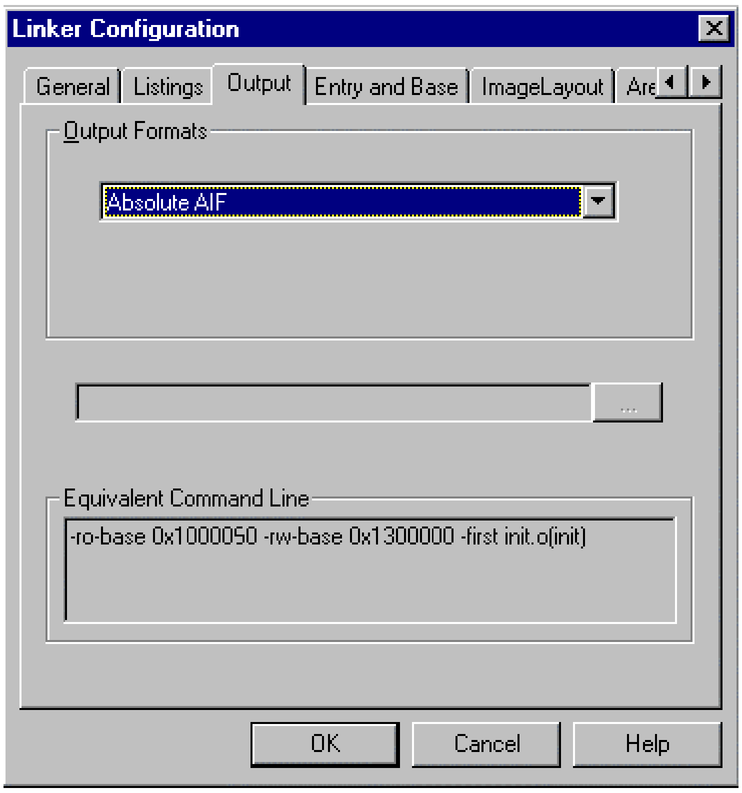
— Read-Write: 0x1300000

5. Modify ImageLayout page on Linker Configuration dialog box (Figure 2-14).

— Object File: init.o

— Area Name: init

4. Click OK.

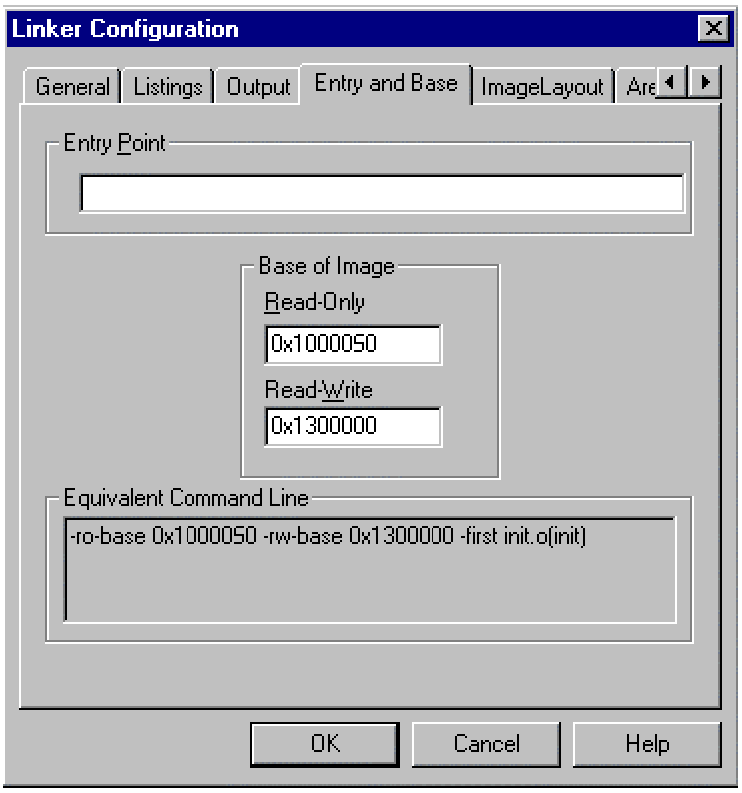


Output Formats: Absolute AIF

Equivalent Command Line

```
-ro-base 0x1000050 -rw-base 0x1300000 -first init.o(init)
```

Figure 2-12. Linker Configuration: Output Page



Entry Point

Base of Image

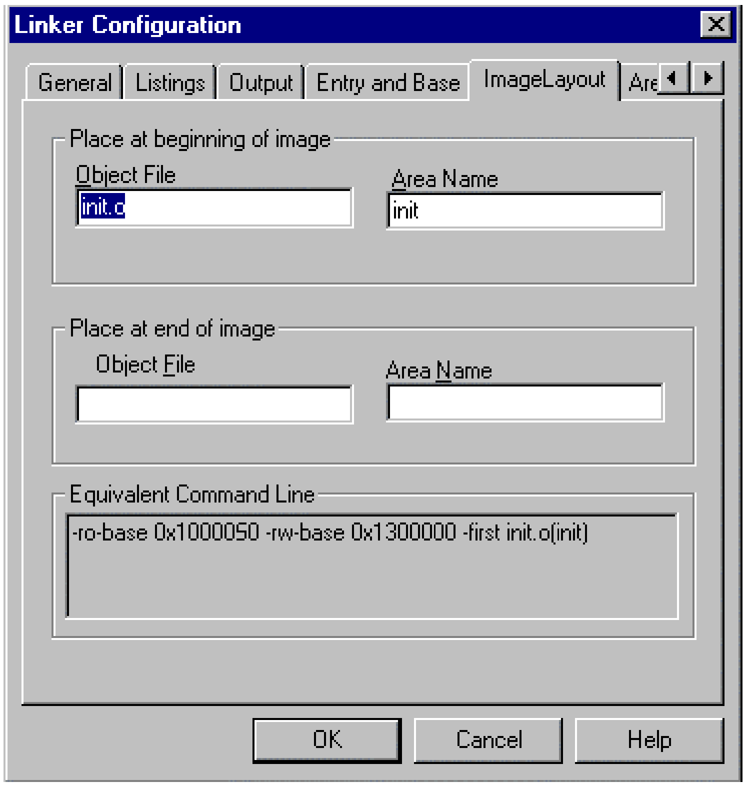
Read-Only: 0x1000050

Read-Write: 0x1300000

Equivalent Command Line

```
-ro-base 0x1000050 -rw-base 0x1300000 -first init.o(init)
```

Figure 2-13. Linker Configuration: Entry and Base Page



Place at beginning of image

Object File: init.o

Area Name: init

Place at end of image

Object File:

Area Name:

Equivalent Command Line

-ro-base 0x1000050 -rw-base 0x1300000 -first init.o(init)

Figure 2-14. Linker Configuration: Image Layout Page

BUILDING DIAG PROJECT

Now, You can start compiling the diag source code in ARM Project manager and Check error message or warning message, and fix problem. If there is no compiling error, executable image file named diag.axf or diag.bin can be generated at working directory.

If the generated image format is ARM image format(ex. Diag.axf), you can execute it by click ARM debug icon on windows or typing "adw diag.axf " at DOS windows. Adw is ARM debugger execution file.

If the generated image format is Binary format, then you can use sftp.exe utility which is available at web site in DOS windows. It`s usage is described at the end of this section.

HOW TO USING ARM DEBUGGER WITH EMBEDDED ICE

If you have built a Diag project without any error, find the executable ARM image output file (diag.axf) on your project sub-directory. Then, execute and debug the project. The generated image file will be downloaded by ARM debugger through the EmbeddedICE(ARM Emulator) interfacing JTAG port to DRAM memory on the SNDS100 target board. Next, you can start to debug the downloaded image using ARM Debugger Window(ADW).

STARTING ARM DEBUGGER

1. Reset the SNDS board and the EmbeddedICE interface unit.

   Reset SNDS (Press reset switch) -> Reset EmbeddedICE interface unit (Press red switch)

2. Start ARM Debugger tool from ARM Project Manager window.

   Project menu -> Debug Diag.apj

   Also, you can start it at DOS windows as you type "adw diag.axf " in there.

3. When ARM Debugger is started, it will load the image code to ARMulator (refer to section 1.5).

   If you ever used ARM Debugger as a remote debugger, Remote Debugger warning message dialog box will be displayed. If the remote debugger option is correct, then select Yes, otherwise, select No.

CONFIGURING ARM DEBUGGER

In order to access a remote target, you should configure ARM Debugger for Windows (ADW) rather than ARMulator. The EmbeddedICE interface unit must also be configured for the ARM core in the target system (SNDS100 board).

The ARM7TDMI core is contained in the KS32C5000 /5000A or KS32C50100 on the SNDS100 board. ARM7TDMI macro cell includes the ARM7 core with Thumb, debug extensions, and the EmbeddedICE macro cell.

To configure ARM Debugger using the EmbeddedICE interface, following the steps:

1. Select Configure Debugger from the Options menu.

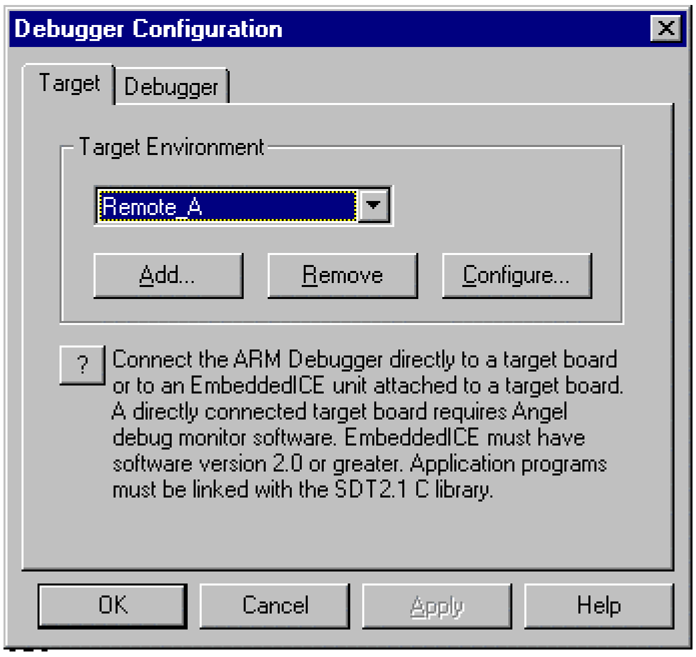
Options -> Configure Debugger

2. Debugger Configuration dialog box is displayed (Figure 2-15). When you select Target page, there are two Target Environment available as follows.

— ARMulator: lets you execute the ARM program without any physical ARM hardware by simulating ARM instructions in software.

— Remote_A: connects the ARM debugger directly to the target board or to an EmbeddedICE unit attached to the target.

3. Select Remote_A from target environments, and click Configure.



Connect the ARM Debugger directly to a target board or to an EmbeddedICE unit attached to a target board. A directly connected target board requires Angel debug monitor software. EmbeddedICE must have software version 2.0 or greater. Application programs must be linked with the SDT2.1 C library.

Figure 2-15. Debugger Configuration: Target Page

CONFIGURING ARM DEBUGGER (Continued)

4. Configure Angel Remote Configuration dialog box(Figure 2-16).

— Remote Connection: select your host and EmbeddedICE communication port configuration.

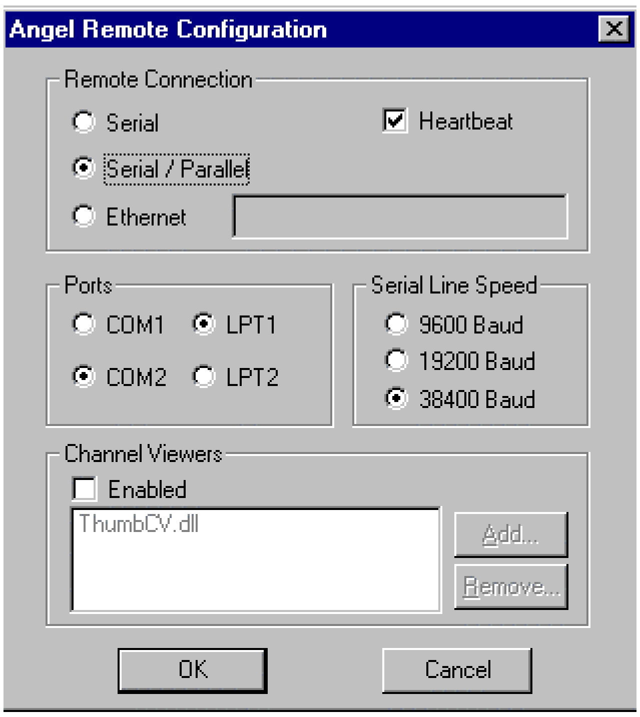
— Select Ports and Serial Line Speed.

5. Select Debugger page from Debugger Configuration dialog box (Figure 2-17) and configure it.

— Endian: big

— Default Memory Map: erase the contents of the default memory map.

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 target. This means that the executable image file is downloaded to the DRAM code area.



Angel Remote Configuration

Remote Connection: Serial [✓ Serial / Parallel] Ethernet; Heartbeat [✓]

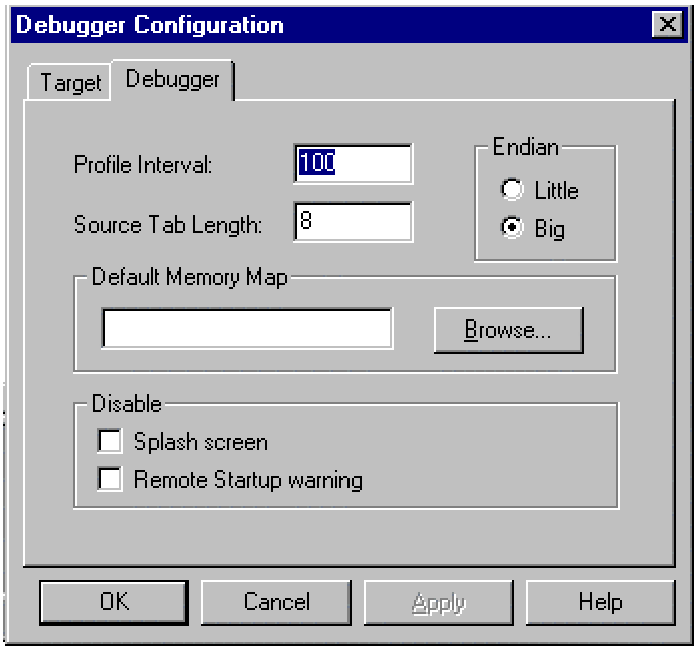
Ports: COM1 [✓ COM2] [✓ LPT1] LPT2

Serial Line Speed: 9600 Baud, 19200 Baud, [✓ 38400 Baud]

Channel Viewers: Enabled [ ]; ThumbCV.dll

Figure 2-16. Angel Remote Configuration

CONFIGURING ARM DEBUGGER (Continued)


Debugger Configuration

Target | Debugger

Profile Interval: 100

Source Tab Length: 8

Endian: Little [✓ Big]

Default Memory Map:

Browse...

Disable
- [ ] Splash screen
- [ ] Remote Startup warning

Figure 2-17. Debugger Configuration: Debugger Page

7. When download is finished, select Configure EmbeddedICE from the Options menu.

   Options -> Configure EmbeddedICE

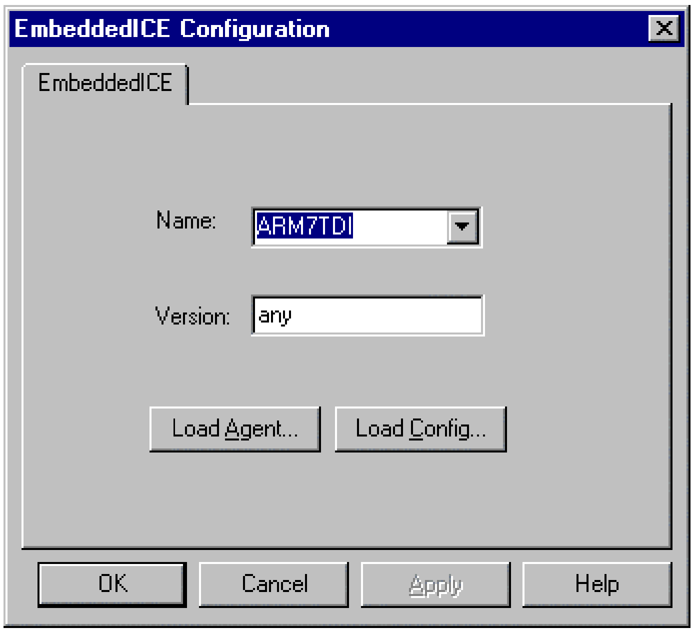
8. EmbeddedICE Configuration dialog box is displayed (Figure 2-18).

— Name: ARM7TDI

9. If you select the OK button in EmbeddedICE Configuration dialog, a new dialog box will be displayed with version number.

10. Click OK.

CONFIGURING ARM DEBUGGER (Continued)



Name: ARM7TDI

Version: any

Figure 2-18. EmbeddedICE Configuration

EXECUTING DIAG PROJECT IMAGE

1. Initialize system variables. After a download, several windows are displayed, such as Execution window, Console window, and Command window. In Command window, you must initialize the system variables, "$semihosting_enabled" and "$vector_catch", by entering the following command:

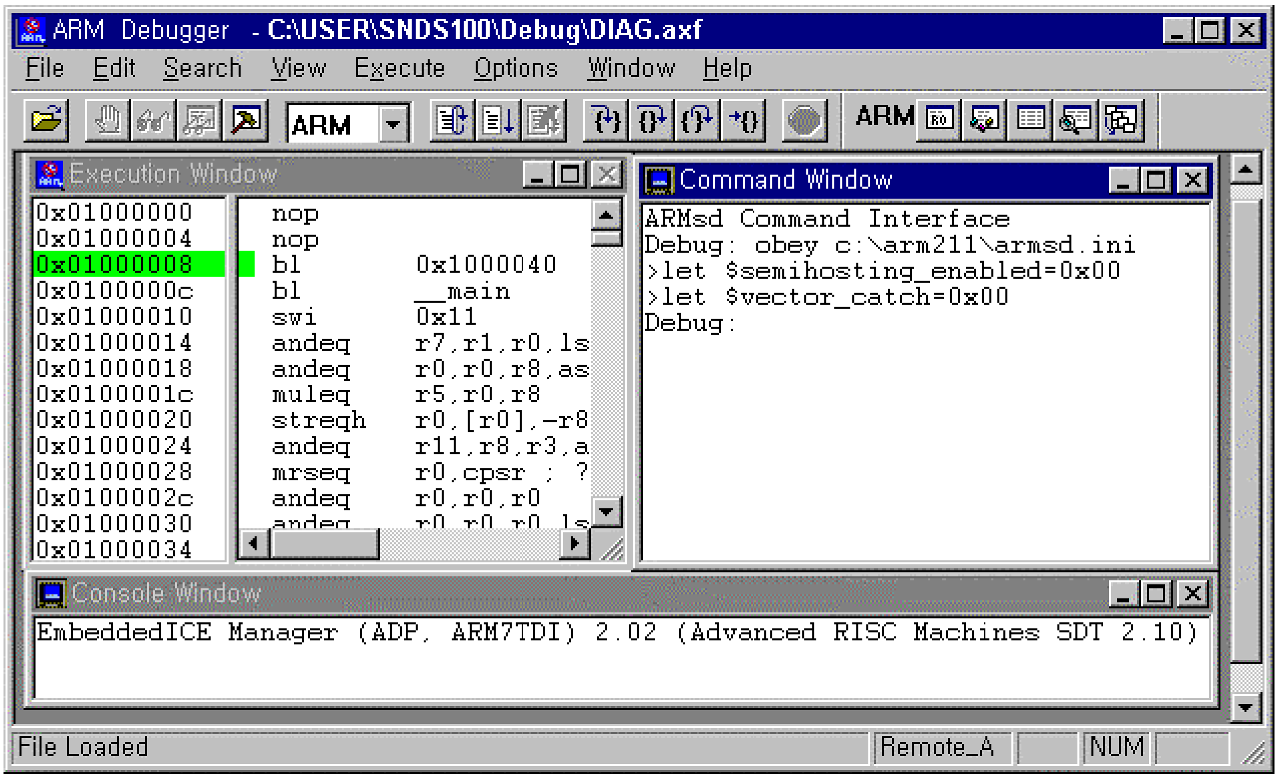
```
let $vector_catch=0x00
let $semihosting_enabled=0x00
```

Or, you can initialize these variables as follows:

First, create a text file named "armsd.ini", which includes the commands described above. Then, enter the following command in the Command window (Figure 2-19):

```
obey c:\arm211\armsd.ini
```

For more information about these steps, please refer to Chapter 6 of "ARM Software Development Toolkit User's Guide".


ARM Debugger - C:\USER\SNDS100\Debug\DIAG.axf
File Edit Search View Execute Options Window Help
ARM
Execution Window
Command Window
Console Window
File Loaded
Remote_A
NUM

```
0x01000000  nop
0x01000004  nop
0x01000008  bl      0x1000040
0x0100000c  bl      __main
0x01000010  swi     0x11
0x01000014  andeq   r7,r1,r0,ls
0x01000018  andeq   r0,r0,r8,as
0x0100001c  muleq   r5,r0,r8
0x01000020  streqh  r0,[r0],-r8
0x01000024  andeq   r11,r8,r3,a
0x01000028  mrseq   r0,cpsr ; ?
0x0100002c  andeq   r0,r0,r0
0x01000030  andeq   r0,r0,r0,ls
0x01000034
```

```
ARMsd Command Interface
Debug: obey c:\arm211\armsd.ini
>let $semihosting_enabled=0x00
>let $vector_catch=0x00
Debug:
```

```
EmbeddedICE Manager (ADP, ARM7TDI) 2.02 (Advanced RISC Machines SDT 2.10)
```

Figure 2-19. ARM Debugger Window(ADW): Command Window

2. Execute the program.

   Execute menu -> Go

3. Now, The downloaded image file(diag.axf) will be run on DRAM area. Diagnostic program running status can be monitored on the current Hyper-Terminal diagnostic window.

DEBUGGING DOWNLOAD IMAGE IN ADW

STEPPING THROUGH THE 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 A 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 the 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 A WATCH POINT

A watch point halts a program when the value of a specified register or a variable changes or is set to a specific number.

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 watchpoint. 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 THE CODE INTERLEAVED WITH THE 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, please refer to Chapter 3 of "ARM Software Development Toolkit User's Guide".

USING MAKEFILE FOR IMAGE BUILD

Now, you know how to build image debug and execute use by ARM SDT on windows environments. And also, you can build the download image file with makefile on DOS environments. Using this makefile, you can generate not only ARM image format but Binary image.

If you have no emulator like as EmbeddedICE, you can simply debug and evaluate the target board with this binary image. At DOS window, this binary image file can be download to target board through serial cable using sftp.exe utility which is available at our web site. More detail refer to the later in this section.

Sometimes, it is more convenience for engineer to build image at DOS windows not using GUI of ARM SDT. The sample makefile for diagnostic sources including boot code is shown bellows:

Makefile

```
###########################################################################
#                                                                         #
#      SNDS100 EVALUATION BOARD ENVIRONMENTS SETUP                        #
#         (KS32C5000/KS32C5000A/KS32C50100)                               #
#                                                                         #
###########################################################################
#                                                                         #
#  !! EDIT THE BELLOW THE OPTION FLAG FOR TO FIT YOUR SYSTEM.             #
#      ( PLEASE,READ "README.TXT" FILES FOR GUIDE )                       #
#                                                                         #
###########################################################################
# ----------------------------------------------------------------- #
#                  ARM TOOLKIT ENVIRONMENTS
# ----------------------------------------------------------------- #
ARMPATH   = C:\ARM211A
ENDIAN    = -BI
ARM_LIB   = ARMLIB_CN.32B
C         = ARMCC
ASSEMBLER = ARMASM
INTERWORK =
IMAGE     = AIF
CPU_OPTS  = -PROCESSOR ARM7TM -ARCH 4T
DEVICE    = KS32C50100

#-----------------------------------------------------------------#
#            ASSEMBLER PREDEFINED FLAGS                           #
#-----------------------------------------------------------------#
#IF(ROM_ADDRESS_AT_ZERO)
#{
#APDFLAG   = TRUE
#MEMORY    = -RO 0X0 -RW 0X1300000
#MEMORY    = -RO 0X0
#}
#ELSE
#{
APDFLAG   = FALSE
MEMORY    = -RO 0X1000050 -RW 0X1300000
#}
```

```
#-----------------------------------------------------------------#
#              C-PREDEFINED OPTION FLAGS                    #
#-----------------------------------------------------------------#
CPDFLAGS     = -D$(DEVICE)  -DEX_UCLK
#CPDFLAGS     = -D$(DEVICE)


######################################################################
#                                                                    #
# !!!DON'T TOUCH THIS MAKEFILE, EXCEPT FOR YOU HAVE TO CHANGE IT.!!  #
#                                                                    #
######################################################################
#                                                                    #
#        SNDS100 EVALUATION BOARD'S MAKEFILE FOR IMAGE BUILD         #
#           (KS32C5000/KS32C5000A/KS32C50100)                        #
#                                                                    #
######################################################################

LINK         = $(ARMPATH)\BIN\ARMLINK
ASM          = $(ARMPATH)\BIN\$(ASSEMBLER)
CC           = $(ARMPATH)\BIN\$(C)
ASM4ARM      = $(ARMPATH)\BIN\ARMASM
CC4ARM              = $(ARMPATH)\BIN\ARMCC
ASM4THUMB    = $(ARMPATH)\BIN\TASM
CC4THUMB     = $(ARMPATH)\BIN\TCC
LIB          = $(ARMPATH)\LIB\$(ARM_LIB)
INC         = $(ARMPATH)\INCLUDE

ASMFLAGS     = $(ENDIAN) -G -APCS 3/32BIT
ASMFLAG_BOOT = $(ENDIAN) -G -APCS 3/32BIT -PD "ROM_AT_ADDRESS_ZERO SETL {$(APDFLAG)}"
CFLAGS       = $(ENDIAN) -G -C -FC -APCS 3/32BIT $(CPDFLAGS) $(CPU_OPTS) -I$(INC)

#***********************************************************************#
#        LINK  OPTION FLAGS FOR BUILD ROM/DRAM/ICE IMAGE            #
#***********************************************************************#
LINK32_5000 = -FIRST INIT.O(INIT) $(MEMORY) -O DIAG -$(IMAGE) -BIN -NOZEROPAD -SYMBOLS
DIAG.SYM $(BSPLIB1) $(BSPLIB2) $(HDLC) $(ARM) $(LIB) $(BSPTEST)

LINK16_5000 = -FIRST TINIT.O(INIT) $(MEMORY) -O TDIAG -$(IMAGE) -DEBUG -NOZEROPAD -SYMBOLS
DIAG.SYM $(BSPLIB1) $(BSPLIB2) $(HDLC) $(THUMB) $(LIB) $(BSPTEST)

LINK32_50100 = -FIRST INIT.O(INIT) $(MEMORY) -O DIAG100 -$(IMAGE) -DEBUG -NOZEROPAD -SYMBOLS
DIAG.SYM $(BSPLIB1) $(BSPLIB2) $(HDLC100) $(ARM) $(LIB) $(BSPTEST)

LINK16_50100 = -FIRST TINIT.O(INIT) $(MEMORY) -O TDIAG100 -$(IMAGE) -DEBUG -NOZEROPAD -
SYMBOLS DIAG.SYM $(BSPLIB1) $(BSPLIB2) $(HDLC100) $(THUMB) $(LIB) $(BSPTEST)

#**************************************************************************#
#            OBJECT CODES FOR BUILD IMAGE                             #
#**************************************************************************#
THUMB         = TINIT.O TSTART.O
ARM          = INIT.O START.O
HDLC          = HDLC.O HDLCINIT.O HDLCLIB.O
```

```
HDLC100      = HDLCMAIN.O HDLC100INIT.O HDLC100LIB.O
BSPLIB1      = DIAG.O DOWN.O FLASH.O MEMORY.O POLLIO.O UART.O ISR.O TIMER.O
BSPLIB2      = IIC.O SYSTEM.O DMA.O KSLIB.O MAC.O MACINIT.O MACLIB.O IOP.O
BSPTEST      = UART_TEST.O IIC_TEST.O TIMER_TEST.O DHRY_1.O DHRY_2.O LCD.O


DIAG: $(BSPLIB1) $(BSPLIB2) $(HDLC) $(ARM) $(BSPTEST)
    $(LINK) $(LINK32_5000)

TDIAG: $(BSPLIB1) $(BSPLIB2) $(HDLC) $(THUMB) $(BSPTEST)
    $(LINK) $(LINK16_5000)

DIAG100: $(BSPLIB1) $(BSPLIB2) $(HDLC100) $(ARM) $(BSPTEST)
    $(LINK) $(LINK32_50100)

TDIAG100: $(BSPLIB1) $(BSPLIB2) $(HDLC100) $(THUMB) $(BSPTEST)
    $(LINK) $(LINK16_50100)

#*********************************************************************#
#                BUILD OPTIONS FOR BOOT CODE                  #
#*********************************************************************#
INIT.O: INIT.S
        $(ASM4ARM) $(ASMFLAG_BOOT) INIT.S -O INIT.O -LIST INIT.LST
START.O: START.S
        $(ASM4ARM) $(ASMFLAGS)  START.S -O START.O
TINIT.O: TINIT.S
        $(ASM4THUBM) $(ASMFLAG_BOOT) TINIT.S -O TINIT.O -LIST TINIT.LST
TSTART.O: TSTART.S
        $(ASM4THUBM) $(ASMFLAGS)  TSTART.S -O TSTART.O

#***********************************************************************#
#              DIAGNOSTIC SOURCE CODES                        #
#***********************************************************************#
DIAG.O: DIAG.C
        $(CC) $(CFLAGS) -ERRORS DIAG.ERR  DIAG.C
DOWN.O: DOWN.C
        $(CC) $(CFLAGS) -ERRORS DOWN.ERR  DOWN.C
FLASH.O: FLASH.C
        $(CC) $(CFLAGS)  -ERRORS FLASH.ERR FLASH.C
MEMORY.O: MEMORY.C
        $(CC) $(CFLAGS)  -ERRORS  MEMORY.ERR MEMORY.C
POLLIO.O: POLLIO.C
        $(CC) $(CFLAGS)  -ERRORS POLLIO.ERR  POLLIO.C
UART.O: UART.C
        $(CC) $(CFLAGS) -ERRORS UART.ERR  UART.C
UART_TEST.O: UART_TEST.C
        $(CC) $(CFLAGS) -ERRORS UART_TEST.ERR  UART_TEST.C
LCD.O: LCD.C
        $(CC) $(CFLAGS) -ERRORS LCD.ERR  LCD.C
ISR.O: ISR.C
        $(CC) $(CFLAGS)  -ERRORS ISR.ERR ISR.C
TIMER.O: TIMER.C
        $(CC) $(CFLAGS)  -ERRORS TIMER.ERR TIMER.C
```

```
TIMER_TEST.O: TIMER_TEST.C
        $(CC) $(CFLAGS)  -ERRORS TIMER_TEST.ERR TIMER_TEST.C
IIC.O: IIC.C
        $(CC) $(CFLAGS)  -ERRORS IIC.ERR IIC.C
IIC_TEST.O: IIC_TEST.C
        $(CC) $(CFLAGS)  -ERRORS IIC_TEST.ERR IIC_TEST.C
SYSTEM.O: SYSTEM.C
        $(CC) $(CFLAGS)  -ERRORS SYSTEM.ERR SYSTEM.C
DMA.O: DMA.C
        $(CC) $(CFLAGS)  -ERRORS DMA.ERR DMA.C
KSLIB.O: KSLIB.C
        $(CC) $(CFLAGS)  -ERRORS KSLIB.ERR KSLIB.C
MAC.O: MAC.C
        $(CC) $(CFLAGS)  -ERRORS MAC.ERR MAC.C
MACINIT.O: MACINIT.C
        $(CC) $(CFLAGS)  -ERRORS MACINIT.ERR MACINIT.C
MACLIB.O: MACLIB.C
        $(CC) $(CFLAGS)  -ERRORS MACLIB.ERR MACLIB.C
HDLC.O: HDLC.C
        $(CC) $(CFLAGS)  -ERRORS HDLC.ERR HDLC.C
HDLCINIT.O: HDLCINIT.C
        $(CC) $(CFLAGS)  -ERRORS HDLCINIT.ERR HDLCINIT.C
HDLCLIB.O: HDLCLIB.C
        $(CC) $(CFLAGS)  -ERRORS HDLCLIB.ERR HDLCLIB.C
HDLCMAIN.O: HDLCMAIN.C
        $(CC) $(CFLAGS)  -ERRORS HDLCMAIN.ERR HDLCMAIN.C
HDLC100INIT.O: HDLC100INIT.C
        $(CC) $(CFLAGS)  -ERRORS HDLC100INIT.ERR HDLC100INIT.C
HDLC100LIB.O: HDLC100LIB.C
        $(CC) $(CFLAGS)  -ERRORS HDLC100LIB.ERR HDLC100LIB.C
IOP.O: IOP.C
        $(CC) $(CFLAGS)  -ERRORS IOP.ERR IOP.C
DHRY_1.O: DHRY_1.C
        $(CC) $(CFLAGS)  -ERRORS DHRY_1.ERR DHRY_1.C
DHRY_2.O: DHRY_2.C
        $(CC) $(CFLAGS)  -ERRORS DHRY_2.ERR DHRY_2.C
```

README.TXT

```
;/************************************************************************************************/
;/*                                                                                              */
;/* FILE NAME                              VERSION                                              */
;/*                                                                                              */
;/*     readme.txt                                  SNDS100 Board version 1.0                   */
;/*                                                                                              */
;/* COMPONENT                                                                                   */
;/*                                                                                              */
;/*                                                                                              */
;/* DESCRIPTION                                                                                 */
;/*                                                                                              */
;/*      It'll be helpful to build Boot ROM or download DRAM image or                           */
;/*      ARM debugger image format for KS32C5000A/KS32C50100.                                   */
;/*                                                                                                  */
;/* AUTHOR                                                                                           */
;/*                                                                                                  */
;/*                                                                                                  */
;/* HISTORY                                                                                          */
;/*                                                                                                  */
;/*       NAME          DATE                                           REMARKS                      */
;/*                                                                                                  */
;/*      in4maker     03-10-1999                           Created initial version 1.0              */
;/*                                                                                                  */
;/*********************************************************************************************/
```

1. HOW TO BUILD DIAGNOSTIC IMAGE FILES FOR KS32C5000A/KS32C50100 ?

```
    *********************************************************************
    1)  First of all, you have to change the option flags in makefile.
        Option flags are as follows:
    *********************************************************************

       (1) ARM SDT ENVIRONMENT SETUP & SELECT DEVICE

          Assign ARM SDT Toolkits installed directory path to ARMPATH.
          And also select used device as like KS32C5000A or KS32C50100
          +-----------------------------------------------------------------------------------+
          ARMPATH =   e:\arm211a
          DEVICE  =   KS32C50100
          +---------------------------------------------------------+

       (2) C-Predefined option flags.
          +-------------------------------------------------------------------------------------------+
          +              C-Predefined option flags                     +
          +-------------------------------------------------------------------------------------------+
          + [LITTLE]     This flag have to be defined for the device to          +
          +              operate in little endian mode.                     +
          +              HDLC,MAC block will be affected by this flag.          +
          + [EX_UCLK]   Using for external UART clock  [usage: -DEX_UCLK]   +
          + [PHY_100M]  Configure PHY to 100Mbps                       +
          + [SNDS1_2]    Have to be defined using SNDS ver1.2 board with      +
          +              KS32C5000A without revise the current diagnostic      +
          +              source code.                                +
          +-------------------------------------------------------------------------------------------+
          CPDFLAGs     = -D$(DEVICE) -DEX_UCLK
          +-------------------------------------------------------------------------------------------+


       (3) DEFINE IMAGE FORMAT AS LIKE ARM/THUBM AND BIG/LITTLE
          +-------------------------------------------------------------------------------------------+
                    THUMB(BIG/LITTLE)        ARM(BIG/LITTLE)
          +-------------------------------------------------------------------------------------------+
          ARMLIB      =  armlib.16b/armlib.16l  armlib.32b/armlib.32l
          C           =          tcc                     armcc
          ASSEMBLER  =        tasm                    armasm
          INTERWORK  =      /interwork
    CPU_OPTs   =                    -processor ARM7TM -arch 4T
    +--------------------------------------------------------------+

  (4) TO BUILD BOOT ROM IMAGE
    +--------------------------------------------------------------+
    IMAGE      =  bin
    APDFLAG    =  TRUE
    MEMORY     =  -RO 0x0 -RW 0x1000050
    +--------------------------------------------------------------+


  (5) TO BUILD DOWNLOAD DRAM IMAGE
    +--------------------------------------------------------------+
    IMAGE      =  bin
    APDFLAG    =  FLASE
    MEMORY     =  -RO 0x1000050 -RW 0x13000000
    +--------------------------------------------------------------+

  (6) TO BUILD ARM DEBUGGER IMAGE
    +--------------------------------------------------------------+
    IMAGE      =  aif
    APDFLAG    =  FLASE
    MEMORY     =  -RO 0x1000050 -RW 0x13000000
    +--------------------------------------------------------------+

***************************************************************************
 2)  sysconf.h

    If you want to change the device's mode like as UART baud rate
    ,IIC serial clock frequency, Internal system clock(fMCLK) etc,
    please update this file.
***************************************************************************
 3)  snds.a
    SNDS100 Board memory configurations and memory MAP can be changed
    by this file.

    Where, the fMCLK also have to be updated to same value as fMCLK
    defined at sysconf.h file.

********  End of readme.txt  **************************************************************
```

Now, you have created makefile to build image for your target systems. And then, you have to compile and link using this file by armmake in DOS windows. *First of all, you have to configure the ARM Project Manager environments(refer to USING ARM PROJECT MANAGER TO BUILD IMAGE) because its default option values are effects on when run the armmake with makefile* .

Its ready to build image in DOS windows . There are four cases as follows:

1. To build 32bit ARM image for KS32C5000/5000A,

   Armmake diag

2. To build 16bit Thumb image for KS32C5000/5000A.

   Armmake tdiag

3. To build 32bit ARM image for KS32C50100,

   Armmake diag100

4. To build 16bit Thumb image for KS32C50100.

   Armmake tdiag100

After this is done with no compile and link error, the following files will be generated at your working directory.

Diag /diag100/tdiag/tdiag100	:	Executable image file which will be ARM or binary image format.
Diag.sym	:	symbol file
<file_name>.o	:	Object file
<file_name>.err	:	Error and warning list files.

If you have emulator (Ex : EmbeddedICE) or Angel potted ROM, you can use ARM debugger window(ADW) .If not, you can download executable binary image file to targets by sftp.exe which is ours program tips. Please refer to the next paragraph.

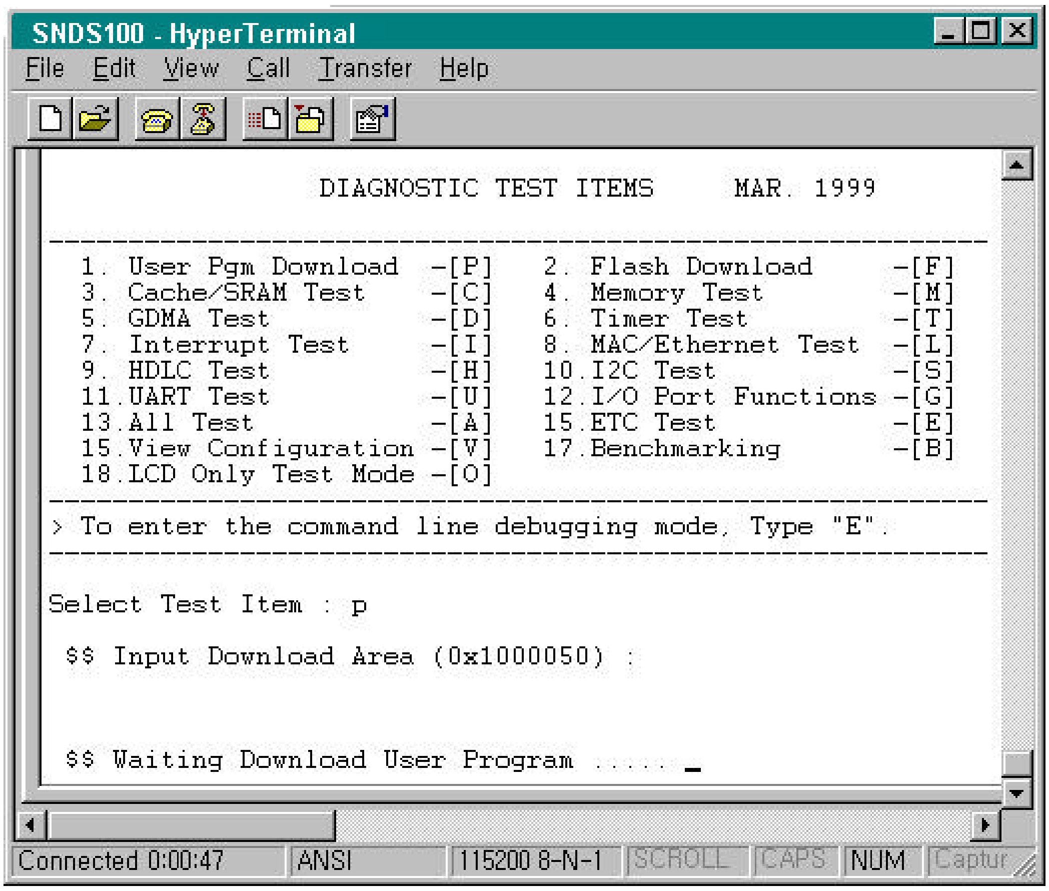
DOWNLOADING EXECUTABLE BINARY IMAGE FILE WITHOUT ADW

Without a Emulator(ex, EmbeddedICE), you can download a binary image file through the serial cable to target. First of all, you ought to get download utility, sftp.exe, from our web sites .

To download a executable binary file, following the steps:

1. Select User Pgm to download item on the SNDS100 console(Hyper-Terminal) menu .

   It is referred to Figure 2-22.



```
                 DIAGNOSTIC TEST ITEMS     MAR. 1999

------------------------------------------------------------
  1. User Pgm Download    -[P]   2. Flash Download     -[F]
  3. Cache/SRAM Test      -[C]   4. Memory Test        -[M]
  5. GDMA Test            -[D]   6. Timer Test         -[T]
  7. Interrupt Test       -[I]   8. MAC/Ethernet Test  -[L]
  9. HDLC Test            -[H]   10.I2C Test           -[S]
  11.UART Test            -[U]   12.I/O Port Functions -[G]
  13.All Test             -[A]   15.ETC Test           -[E]
  15.View Configuration   -[V]   17.Benchmarking       -[B]
  18.LCD Only Test Mode   -[O]
------------------------------------------------------------
> To enter the command line debugging mode, Type "E".
------------------------------------------------------------

Select Test Item : p

 $$ Input Download Area (0x1000050) :


 $$ Waiting Download User Program ..... _
```

Figure 2-22. Download Executable Binary Image on SNDS100 Board

ELECTRONICS

DOWNLOADING EXECUTABLE BINARY IMAGE FILE WITHOUT ADW (Continued)

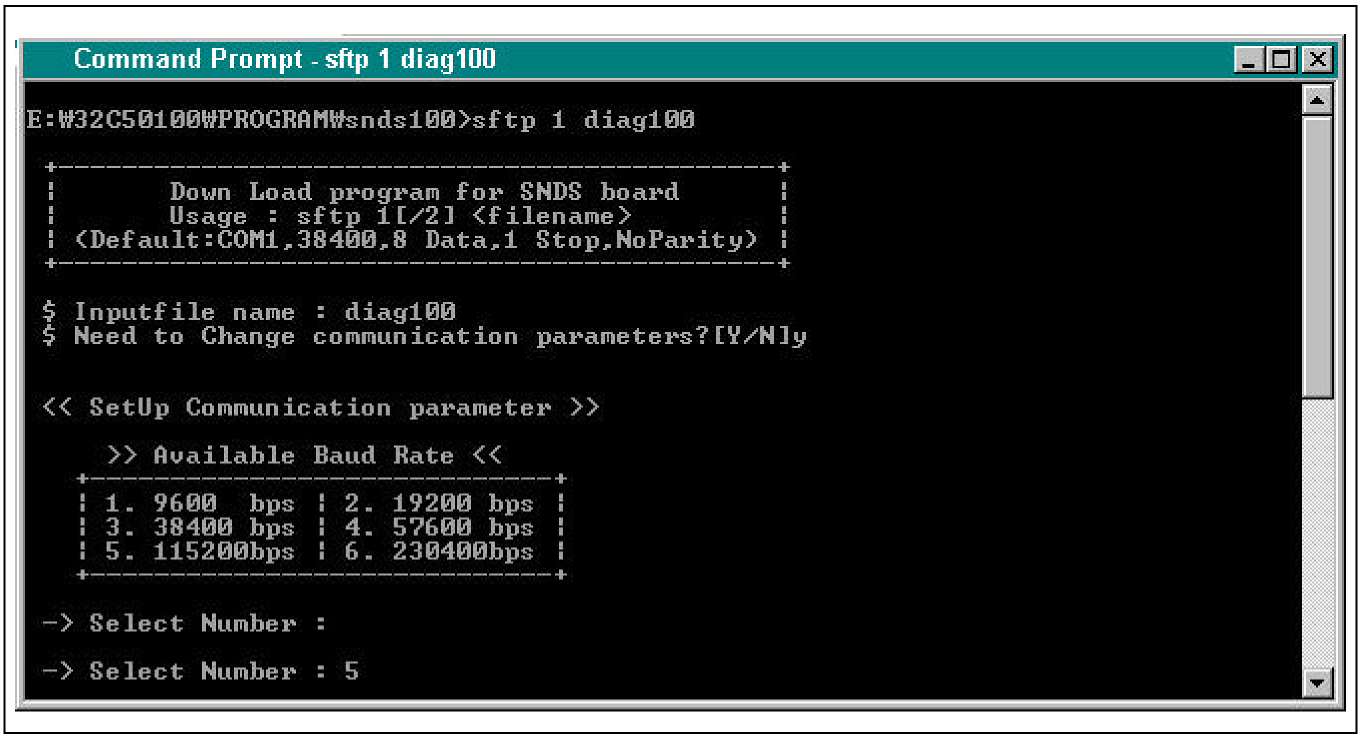
2. Start download the binary image file on host computer.(refer to Figure 2-21)

   `sftp.exe` is used to download binary image file to target. This execution file is compiled at Window95, and the compiler is `Visual C++` If you are in a different environment, you should recompile the serial download utility source code in your environments. More detail information about `sftp.exe` is described later this chapter.

   If you already have the utility tips, Now, you can send the image file from COM1 or COM2 port on Host PC through the serial cable to Console port(Default, UART channel 0, SIO-0) on SNDS100. For example, if the UART baud rate is 125200 on SNDS100, The serial cable is connected from COM1 to SIO-0, then you have to type `*DOS> sftp 1 diag100*` *on DOS window*`s command Prompt.

   USAGE : SFTP <COM PORT> <FILE NAME>

   Example : dos> sftp 1 diag100



```
E:W32C50100WPROGRAMWsnds100>sftp 1 diag100

 +-------------------------------------------+
 |       Down Load program for SNDS board    |
 |       Usage : sftp 1[/2] <filename>       |
 | <Default:COM1,38400,8 Data,1 Stop,NoParity> |
 +-------------------------------------------+

 $ Inputfile name : diag100
 $ Need to Change communication parameters?[Y/N]y


 << SetUp Communication parameter >>

     >> Available Baud Rate <<
   +------------------------------+
   | 1. 9600  bps | 2. 19200 bps |
   | 3. 38400 bps | 4. 57600 bps |
   | 5. 115200bps | 6. 230400bps |
   +------------------------------+

 -> Select Number :

 -> Select Number : 5
```

Figure 2-21. Start Download Binary Image File to Target on Host Computer

2. When the download is finished without any error, press any key to start the download user binary image file on SNDS100 target board.

Now, you can simply evaluating and verifying the downloaded applications on SNDS100.

FUSING BOOT ROM IMAGE TO FLASH(EEPROM)

You can update the boot ROM on SNDS100 into your new boot image include applications using flash download program which is included in diagnostic code. (please, visit our web site to download this source codes).

SST flash device used as Boot EEPROM on SNDS100 board. So, the flash program in diagnostic code is for the SST device. If you want to use any other vendor' s flash device, you have to programming for that to fuse the application to Flash memory without ROM writer.

The above case is just only for updating boot ROM on SNDS100 board. If you have an Emulator(EmbeddedICE), you can fuse the boot & applications through the JTAG port to flash device on board without boot ROM. In this case, the SNDS100 memory map of the CPU(KS32C5000/5000A/50100) have to be configured by ADW. More detail will be described in this section.

UPDATE BOOT ROM ON SNDS100 BOARD

First of all, you have to prepare the DRAM image file which will be used to fuse new boot ROM image or applications to flash memory on SNDS100 board. To build ROM or DRAM image, please refer to "USING MAKEFILE FOR IMAGE BUILD" section of this chapter.

On SNDS100 board, the bus width of Boot ROM can be extended to 16bit. Actually, the bus width of ROM Bank0(Boot ROM bank) can be configured by B0SIZE0(pin 73) and B0SIZE1(pin 74), but you have to edit the flash.h file to modify the bus width of ROM Bank0 as same value as it.

Detail flow charts for fusing EEPROM are given here. And also the procedure according to that is following:

1. Open flash.h file for to update the bus width of ROM Bank0(Boot ROM).

   For example, if you want to use 16bit Flash device, then modify the define statement to "#define *B0SIZE B0SIZE*_SHORT" in flash.h file.

2. Build DRAM image file.

   Please refer to "USING MAKEFILE FOR IMAGE BUILD" section for more detail DRAM image build.

3. Download DRAM image.

   Firstly, you have to *enter a character `P`* to select user pgm download item on Console terminal (Hyper Terminal). And then you can send DRAM image file to target board on DOS prompt.

   DOS> sftp 1 diag100

4. Execute download image.

   After the DRAM image is downloaded to DRAM successfully, press any key to restart the target board. Now, the downloaded DRAM program is running on DRAM area. Next, *you have to enter a character `F`* to download the new Boot ROM image or Applications to DRAM user area.

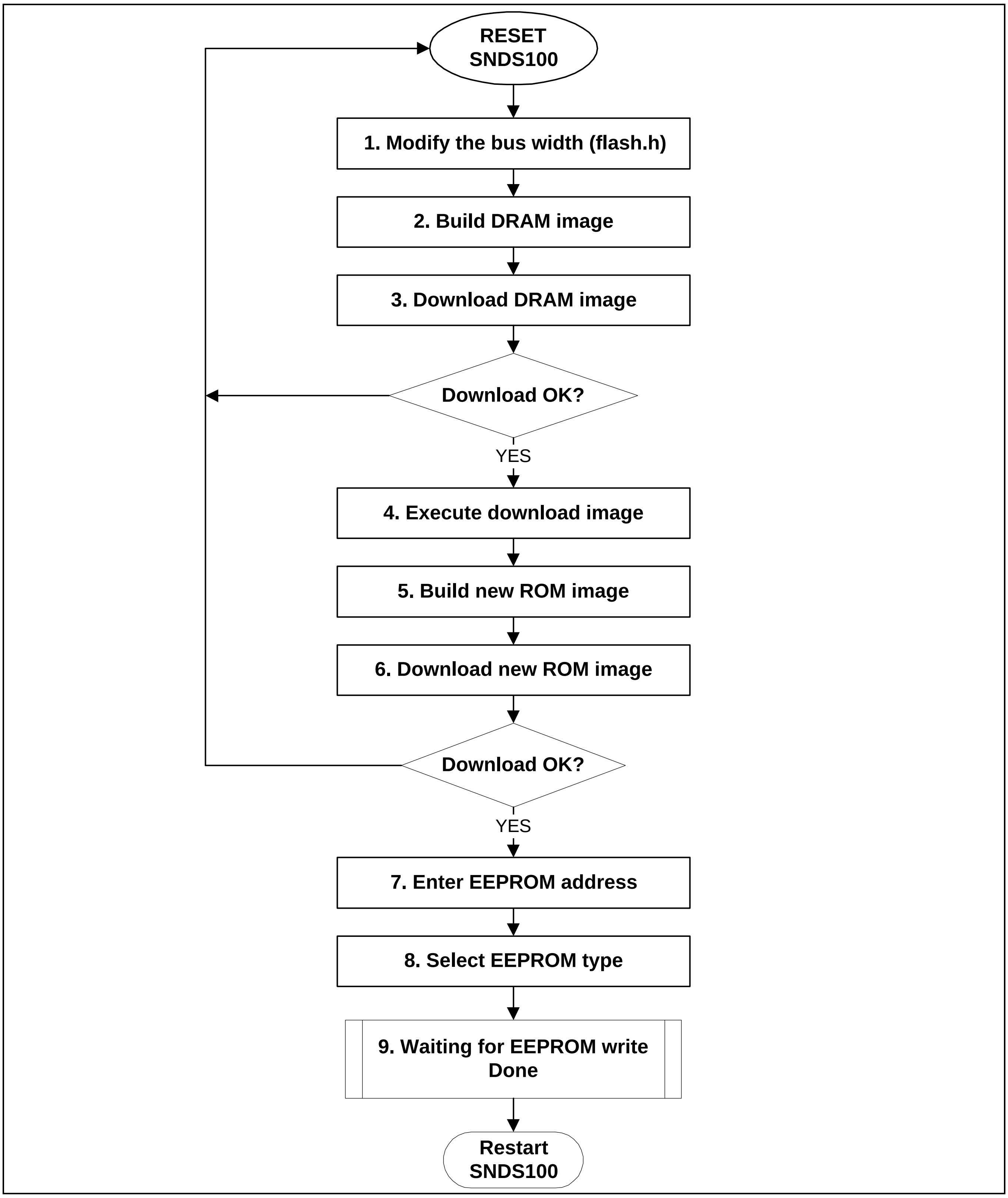


Figure 2-22. Fusing Flow to Update Boot ROM

UPDATE BOOT ROM ON SNDS100 BOARD (Continued)

5. Build New ROM image.

   This image is for the new boot code or applications. It also refer to `USING MAKEFILE FOR IMAGE BUILD` section.

6. Download New ROM image.

   This new ROM image or Applications will be downloaded to DRAM user area and also fused to flash device by flash write program running on DRAM.

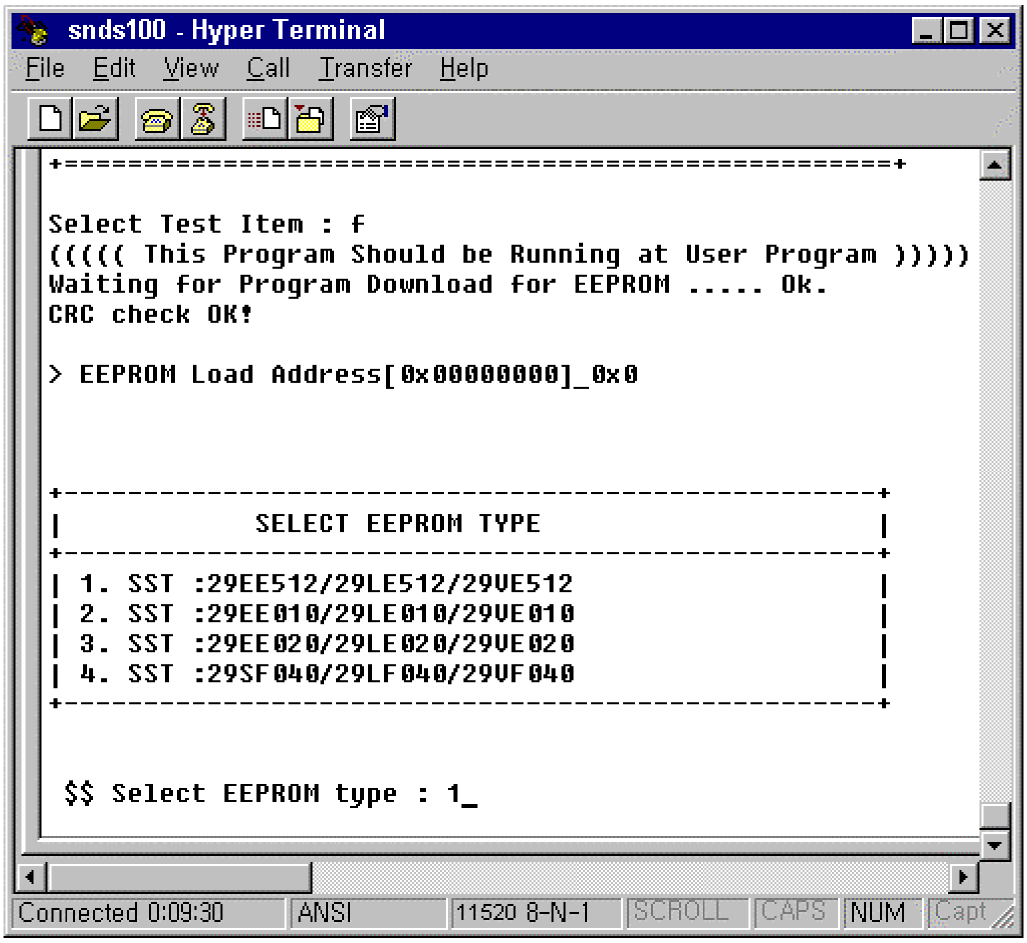
7. Enter EEPROM address.(Figure 2-23)

   If there is no error for downloading, the console (Hyper Terminal) request to enter the EEPROM load address which is default zero. After this, SELECT EEPROM TYPE menu is displayed on Console window.

8. Select EEPROM type.(Figure 2-23)

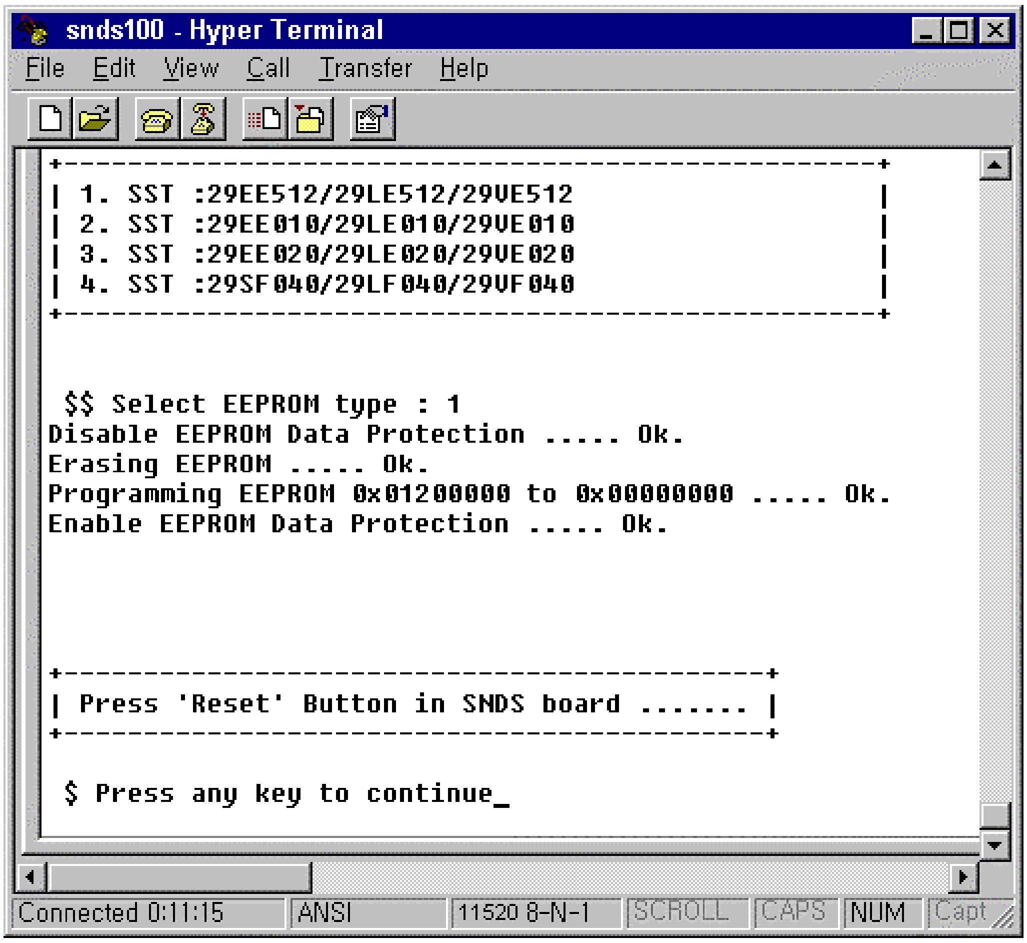
9. Waiting for EEPROM write is DONE.

10. Now, press RESET button on SNDS100 target board to restart it.


snds100 - Hyper Terminal
File Edit View Call Transfer Help
Connected 0:09:30 ANSI 11520 8-N-1 SCROLL CAPS NUM Capt

```
+=========================================================+

Select Test Item : f
((((( This Program Should be Running at User Program )))))
Waiting for Program Download for EEPROM ..... Ok.
CRC check OK!

> EEPROM Load Address[0x00000000]_0x0



+-----------------------------------------------------+
|             SELECT EEPROM TYPE                      |
+-----------------------------------------------------+
| 1. SST :29EE512/29LE512/29VE512                     |
| 2. SST :29EE010/29LE010/29VE010                     |
| 3. SST :29EE020/29LE020/29VE020                     |
| 4. SST :29SF040/29LF040/29VF040                     |
+-----------------------------------------------------+


 $$ Select EEPROM type : 1_
```

Figure 2-23. Start Flash Download Program



```
+-----------------------------------------------------+
| 1. SST :29EE512/29LE512/29VE512                     |
| 2. SST :29EE010/29LE010/29VE010                     |
| 3. SST :29EE020/29LE020/29VE020                     |
| 4. SST :29SF040/29LF040/29VF040                     |
+-----------------------------------------------------+


 $$ Select EEPROM type : 1
Disable EEPROM Data Protection ..... Ok.
Erasing EEPROM ..... Ok.
Programming EEPROM 0x01200000 to 0x00000000 ..... Ok.
Enable EEPROM Data Protection ..... Ok.




+----------------------------------------------+
| Press 'Reset' Button in SNDS board ....... |
+----------------------------------------------+

 $ Press any key to continue_
```

Figure 2-24. Finished Flash Download Program

SERIAL DOWNLOAD PROGRAM ON HOST COMPUTER

We provide the download utility for DOS. This program tips which are compressed into SERIAL.zip file are available on our web site, (www.samsungsemi.com). In this zip file, the "Serial.c" which is main source code is for to manipulate binary data , and the "Trans.c" for to manipulate the serial communications of host computer.

We compiled this source code by "Visual C++" and generated execution file, sftp.exe. Using this download utility tips, you can download executable binary image for ROM, DRAM to SNDS100 target board.

This utility tips is just only for DOS. So, if you want to send binary image file to target, you have to run this tips on DOS Prompt. It' s usage is " DOS> sftp <COM1/COM2> <Binary Image file>".

BINARY DATA TRANSFER FORMAT

To transfer a file to target, the serial transfer utility program for host computer and the flash down program in diagnostic code for SNDS100 use any transfer format which is shown in Figure 2-25.

Length (8 bytes)	DATA (variable)	CRC (8 bytes)

Figure 2-25. Binary Data Transfer Format

Where the Length field is only data(download file) size. Length and CRC field is calculated by download utility program.

CRC CHECK FOR ERROR CHECK

We use CRC check method for error check, The CRC check use CCITT polynomial, and we check all 8-bit data for each character.

Code for CRC Calculation

```
for( j = 0; j < 8; j++ )
{
  CheckSum <<= 1;
  if( CharData & 0x0080 ) CheckSum ^= CCITT_POLYNOM ;
  CharData <<= 1;
 }
```

DATA TRANSFER FLOW

File transfer flows are shown in Figure 2-26. Where the data(File) is an executable binary ROM or DRAM image or general text file.

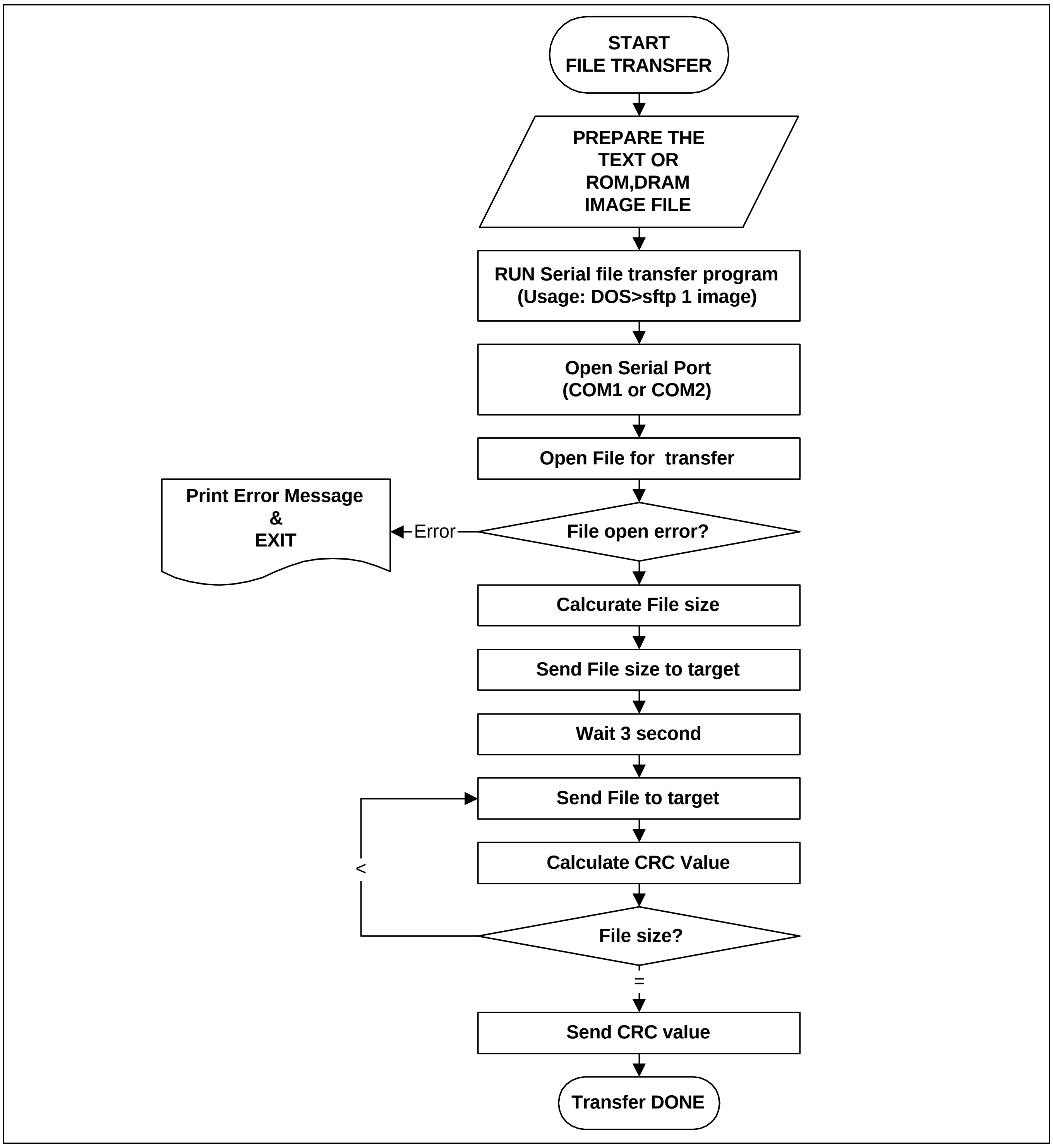


Figure 2-26. File Transfer flow