

NET+OS v6.1 GNU

Getting Started Guide



Kueferstrasse 8
☎ +49 7667 908-0
sales@fsforth.de

- Breisach (Germany)
- Fax +49 7667 908-200
- www.fsforth.de

© Copyright 2003:

FS Forth-Systeme GmbH
Postfach 1103, 79200 Breisach, Germany

Release of Document: March 14, 2005
Filename: NET+OS61_GNU_GettingStarted_v1.2.doc
Author: H. Bujanda
Program Version: 6.1-fs1.2_GNU

All rights reserved. No part of this document may be copied or reproduced in any form or by any means without the prior written consent of FS Forth-Systeme GmbH.

Table of Contents

1. Preface.....	6
2. Features.....	7
3. Requirements.....	8
3.1. NET+OS CD	9
3.1.1. doc.....	9
3.1.2. Image_12.....	9
3.1.3. install-cygwin1-5-12	9
3.1.4. tftp.....	9
3.1.5. debuggers/majic.....	9
3.1.6. debuggers/chameleon	9
3.1.7. netos	9
3.1.8. Toolchain	9
3.1.9. uboot.....	9
4. Installation.....	10
4.1. Installing Cygwin.....	10
4.2. Installing the GNU Toolchain	12
4.3. Installing NetOS v6.1	13
4.4. Installing HCC EFFS-STD NAND Full Version Upgrade	13
4.5. Installing tftp Server	13
4.6. Installing debugger software	14
4.6.1. Installing MAJIC debugger software	14
4.6.2. Installing Chameleon debugger software.....	16
5. Getting Started.....	18
5.1. Building a NET+OS Example Application	20
5.2. load and execute a NetOS image with UBOOT Bootloader	22
5.3. Debugging a NetOS image	23
5.3.1. MAJIC Debugger	23
5.3.2. Chameleon Debugger.....	24
5.4. About NAND Flash File System.....	26
5.4.1. General Information	26
5.4.2. HCC EFFS-STD NAND Full Version Upgrade.....	28
5.4.3. File System Size & Position	29
5.4.4. Stress Example.....	29
5.4.5. Remote Example	31
5.4.5.1. ftp server.....	32
5.4.5.2. http server.....	34
5.4.6. Integration of the NAND file system with the C-Library.....	34
5.5. About NVRAM.....	35
5.5.1. Using A9M9750 Serial EEPROM.....	35
5.5.2. NetOS using UBOOT parameters.....	36
5.5.3. Example of using eeprom for user applications	36
5.6. About Real Time Clock	36
5.7. About USB	36
5.7.1. USB Device.....	36
5.7.2. USB Host	38
5.8. About Serial Ports	39
5.8.1. NetOS Release 1.1 with A9Vali_0 Base board.....	40

5.8.2. NetOS Release 1.2 with A9M9750DEV_0 Base board	40
5.8.3. Update UBOOT from A9Vali_0 to A9M9750DEV_0	41
6. Restoring the Factory Default Image	42

Documentation History

Date	Version	Responsible	Description
2005-3-14	1.2	Héctor Bujanda	Released for image1.2_GNU of NET+OS v6.1 External UART support so A9M9750DEV_0 board can be used. Integration of NAND file system with C-Library and http, ftp applications.
2005-1-14	1.1	Héctor Bujanda	Released for image1.1_GNU of NET+OS v6.1 Some debugging issues for Chameleon changed
2004-12-24	1.0	Héctor Bujanda	Released for image1.0_GNU of NET+OS v6.1

1. Preface

This document details how to install NET+OS on the A9M9750 Developer's Kit.

NET+OS is a realtime operating system from NetSilicon, Inc. which uses ThreadX (www.expresslogic.com) as the underlying RTOS kernel. NET+OS is available from NetSilicon in 2 flavors: one which uses the Green Hills MULTI2000 Integrated Development Environment and one which uses the Microcross GNU toolchain. The A9M9750 Developer's Kit for NET+OS is effectively a third variant which uses the GNU toolchain as built by FS Forth-Systeme GmbH. The NET+OS libraries such as tx.a (ThreadX) or tcpip.a are identical in all 3 variants.

2. Features

With NET+OS the user has a platform with the following features:

- ThreadX realtime operating system
- TCP/IP stack with most of Protocols and Services.
- C and C++ supported for application development
- Ethernet, USB Host, USB Device, Serial, SPI, I2C, flash, nand flash, RTC and EEPROM support
- NAND File System integrated into FTP Server, FTP Client, Email client, and the ANSY I/O functions.
- GNU toolchain with the following components:
 - binutils v2.12
 - gcc v3.2
 - newlib v1.11
 - Insight v5.1.1
- Command line and graphical debugging support

3. Requirements

To install and run NET+OS, you need a host PC which runs WindowsNT/2000/XP and meets these hardware requirements:

CPU	400 MHz Pentium processor or faster
RAM	32 MBytes
Disk space	750 MBytes free disk space
Devices	CD-ROM drive
	1024 x 768, 256-color display
	An available serial port (38400Bd)
	An available Ethernet port

3.1. NET+OS CD

The CD contains all the software and documentation needed to install and use NET+OS

3.1.1. doc

This document in PDF format. More documentation coming from NetSilicon like the API, Hardware Reference Manual ...

3.1.2. Image_12

Contains the factory default flash image in case this needs to be restored.

3.1.3. install-cygwin1-5-12

Components from the Cygwin 1.5.12 release.

3.1.4. tftp

Evaluation software of a tftp server to download your netos images (image.uboot) through ethernet to your target using UBOOT bootloader

3.1.5. debuggers/majic

Updates for the majic debugger software and initialization scripts needed for debugging.

3.1.6. debuggers/chameleon

Gdb server necessary to debug using inexpensive chameleon debugger.

3.1.7. netos

The NET+OS software including libraries, board support package (BSP) and application examples.

3.1.8. Toolchain

The GNU cross-development tools for ARM as a tgz archive. In a sub-directory the sources for the tools.

3.1.9. uboot

UBOOT bootloader image in case it needs to be restored.

4. Installation

There are 6 components to the NET+OS package which must be installed in the following order:

1. Cygwin
2. GNU tools
3. NET+OS
4. HCC EFFS-STD NAND Full Version Upgrade
5. tftp server
6. Debugger software

4.1. Installing Cygwin

Cygwin is a UNIX environment for Windows. If you are already using a version of Cygwin on your machine, then you can skip this section and proceed to installing the GNU tools.

With previous versions of the cygwin dll you may have problems with the OcdServer software, necessary to debug using Chameleon debugger.

Note that the 1.5.12 version numbering refers only to the Cygwin DLL. Individual packages like bash, gcc, less, etc. are released independently of the DLL.

The setup.exe utility tracks the versions of all installed components and provides the mechanism for installing or updating everything. Run this program any time you want to install a Cygwin package.

Note also that, by default, setup.exe does not install everything. Only the base Cygwin distribution is installed by default. When running setup.exe, clicking on categories and packages in the package installation screen will provide you with the ability to control what is installed or updated.

You should click on individual categories and install either entire categories or packages from the categories themselves as outlined below. The setup.exe also takes care of dependencies so that clicking on a particular package may lead to a library being included as well, even though you didn't click on the library. This can make the installation a little confusing, but please bear with it. A setup_ref.log file is included on the CD which can be used to verify that you've selected the right packages.

Here are the individual steps:

On the CD, change into the install-cygwin1-5-12 directory.

Click on "setup.exe"

Select "Install from local directory"

Choose your root directory, for example c:/cygwin

Select the "UNIX" and "Just Me" radio buttons.

Note: do not install in a directory where the path contains white space. For example, installing in "c:\Program Files\Cygwin" will not work.

Leave the Local Package Directory as default (Where you are executing setup.exe)

Select the packages to install. In the top of this window select "Curr". It is best to view the category fields.

In the following table, "default" means leave the category as is, "all" means select every package in that category.

In Admin leave as default.

In Archive select unzip and zip

In Base leave as default.

In Database leave as default.

In Devel select all

In Doc leave as default.

In Editors select vim

In Gnome leave as default

In Graphics leave as default

In Interpreters leave as default

In Libs leave as default

In Mail leave as default

In Math leave as default

In Net select inetutils and ncftp

In Publishing leave as default

In Shells leave as default

In System leave as default

In Text leave as default

In Utils select all

In Web select wget

Leave rest as default

Click "Next" and the installation starts.

At the end of the toolchain you have the option of installing the icon on your desktop, which we recommend you do.

4.2. Installing the GNU Toolchain

In the toolchain folder on the CD, copy bin-x-arm9-1.2.tgz into a tmp directory under Cygwin, e.g. c:/cygwin/tmp.

Then start a bash shell by clicking on the Cygwin desktop icon. In the bash shell you should have a prompt and be able to enter the following commands:

```
$cd /usr/local  
$tar xvzf /tmp/bin-x-arm9-1.2.tgz
```

Then change to your home directory and using your favorite editor add the toolchain path to your current path.

```
$cd
```

```
$vi .bashrc
```

```
PATH=/usr/local/x-arm9/bin:$PATH
```

Source the file for the changes to take effect.

```
$. ./bashrc
```

Verify that the toolchain path is now present.

```
$echo $PATH
```

The ultimate test is to start the GNU debugger.

```
$arm-elf-gdb
```

The Insight GUI for GDB should now appear. Quit the program.

4.3. Installing NetOS v6.1

In the netos folder on the CD, copy NetOS61_GNU-image-1.2.tar.gz into a tmp directory under Cygwin, e.g. c:/cygwin/tmp.

Then start a bash shell and enter the following commands:

```
$cd /  
$mkdir -p /targets  
$cd /targets  
tar xvzf /tmp/NetOS61_GNU-image-1.2.tar.gz
```

This will install the NET+OS libraries, BSP, examples and utilities.

4.4. Installing HCC EFFS-STD NAND Full Version Upgrade

If you have bought the HCC EFFS-STD NAND Full Version Upgrade, you should have received an upgrade with a file called nand_fs.tar.gz. After installing NetOS (previous step), follow these instructions to complete the upgrade.

In the "HCC EFFS-STD NAND Full Version Upgrade" CD, copy nand_fs.tar.gz into a tmp directory under Cygwin, e.g. c:/cygwin/tmp.

Then in a shell and enter the following commands:

```
$cd /targets/NetOS61_GNU_V102/src  
tar xvzf /tmp/nand_fs.tar.gz
```

This will install sources of the hcc effs-std NAND file system at /src/nand_fs folder.

4.5. Installing tftp Server

You need to setup a tftp server to download your netos images (image.uboot) through Ethernet to your target using UBOOT bootloader

We provide a TFTP server evaluation software from WaluSoft (www.walusoft.co.uk).

This unregistered evaluation version has the following restrictions.

- Max 2 simultaneous clients
- Max 10 transfers before quitting
- No firewall support

- No create path
- Fixed to port 69
- Server stops after one hour and need to be manually restarted

Start the TFTPsvC2001.msi file from TFTP directory of the CD and follow the installation instruction. After installation has been terminated successfully go to Start > Settings > Control Panel. A TFTP Server should be available in the Control Panel.

Open the TFTP Server and click on the Outbound Path tab. Set the path where you want to place the images (image.uboot) to be downloaded. Normally it is a common directory like c:\tftp_Server or the 32b directory of your current application for example src/apps/template/32b

If there is already installed an TFTP server on the PC Host, it is not necessary to follow this instructions.

Any other TFTP server can be used under Windows to download the image.

To manually restart the TFTP server open under Control Panel the Administration Tools. A Window will be opened.

Then open Services. A list of all services will be shown. In the services windows look for the TFTP server service and restart it.

4.6. Installing debugger software

In all documentation we show deeply the use of two branches of debuggers: - The expensive solution is MAJIC debugger from company E.P.I. that connect with your computer through ethernet.

- A cheaper solution is Chameleon POD debugger from company Amontec that connect with your computer through the parallel port.

You only need to install the solution chosen, although both can installed if desired.

4.6.1. Installing MAJIC debugger software

MAJIC hardware debugger will allow you to debug NetOS. GDB commands from your host debugging tool (Insight GNU GDB) sent through ethernet to the MAJIC debugger will be executed in the target processor (NS9750) using JTAG interface.

1. The installation of the EPI MAJIC software is a two-step process.

- a. Install the software from the MAJIC CD (the installer will immediately appear). You will need the jewel case as the installer will ask for the keycode (on the back of the jewel case), and the serial number (on the MAJIC CD itself).
- b. Update the software by extracting file edta20b_sp5ns.zip from debuggers/majic directory of the NetOS61 CD to where you installed MAJIC software of previous step for example C:\program files\EPI Tools\edta20

Overwrite previous files.

2. Select a directory from which to run the MAJIC RDI server. For example, C:\majic\mdi_a9m9750_reference. Extract the majic_mdi_a9m9750.zip file from debuggers/majic directory of the NetOS61 CD into that directory. These are the MAJIC initialization files.
3. Setup a MDI server icon, this is the interface between the gdb and the MAJIC. To do this go to Start->EPI Tools EDTA->MAJIC Setup Wizard (this is the software that was installed as part of step 1).
4. Click Next on the Intro screen.
5. Click on MDI Compliant debugger, in the “Choose debugger” list box
6. Click Go
7. Pick a project name; this will appear on the MDI server icon on your desktop
8. Click Next
9. Pick the Processor type ARM926EJS, in the “Select Your Processor Type” list box
10. Select Big Endian
11. Under “Startup Connection Mode”, select “Intrusive Mode”.
12. Click “Next”
13. Select “I will be using an Ethernet IP address to communicate with my MAJIC
14. Click on “Use my Static IP address”
15. Enter in an IP address for your MAJIC
16. Click Next
17. Click on “Use Existing Startup Files” and select the directory you choose for Step 2
18. Click Next
19. Select a Destination directory. For example C:\majic\mdi_a9m9750
20. Click Next

21. Click on “Perform Actions” button.

22. Lastly, click “Done”

4.6.2. Installing Chameleon debugger software

Here you install OcdRemote application for your host computer. It captures GDB commands from your host debugging tool (Insight GNU GDB) and translates/transmits them to the Chameleon POD through the Parallel port of your computer. Chameleon will execute the commands in the target processor (NS9750) using JTAG interface.

In the debuggers/chameleon folder on the CD, click on self-extracting hwsupport-2.05.exe.

This installs some utilities and DLL's in /usr/local/bin.

Chameleon POD is device based on programmable logic. It can be used for multiple purposes. Here it is used to speed up the process of downloading and debugging an image.

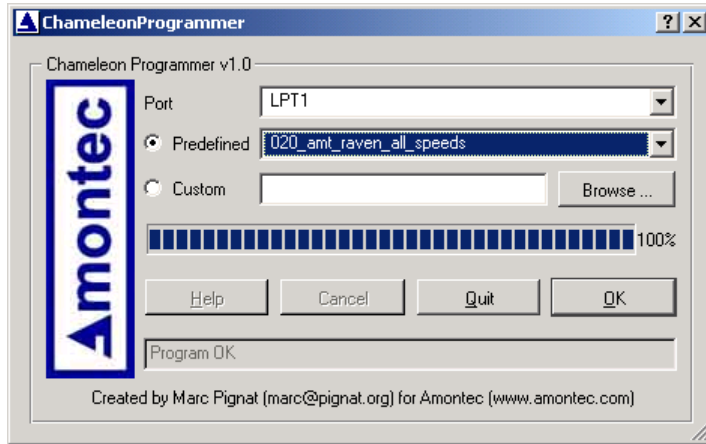
As Chameleon is delivered, it should have a valid programming for our a9m9750. The interface programmed is RAVEN. As explained in chapter "Debugging a NetOS Image", you have to specify RAVEN interface when using OCDRemote command.

The Raven operates as a parallel to serial and serial to parallel converter for OCD signals. The host PC talks to the Raven via EPP mode of the parallel port.

As Chameleon is a re-programmable device, it is possible to change its logic using free software provided by the manufacturer (See http://www.amontec.com/chm_chameleon.shtml).

If you want to update the logic of the Chameleonit, follow following steps:

- Install Parellel port driver from http://www.amontec.com/chm_chameleon.shtml
- Install Chameleon Programmer from the same web page.
- Connect Chameleon to the parallel port of your computer, put its switch in reprogramming mode and power it (yellow led should light).
- Start Chameleon Programmer, Select configuration as shown in following picture and press OK button.
- In some seconds Chameleon will be updated.
- Change Chameleon switch to the original position.



You have now completed the installation and we recommend that you re-boot Your computer.

5. Getting Started

Connect an Ethernet cable to the RJ45 (X17 on A9M9750DEV_0) connector. If you are connecting to a hub, use a Cat-5 patch cable; if you're connecting directly to the Ethernet port on the PC, then use a Cat-5 cross cable.

Connect the serial cable between COMA port of the target (X18 on A9M9750DEV_0)* and COM1 of your PC. Start a terminal program such as Hyperterminal on your PC and set to 38400Bd, 8 data bits, no handshaking.

Connect the Power Supply, delivered with the A9M9750 Developer's Kit, to the Base Board. The 3 Power LEDs should now be lit and UBOOT startup messages should be displayed.

After 3 seconds, if the countdown is not interrupted by pressing a key, the bootcmd command will be executed, running NetosFlash command. This command will load the default NetOS image (template app) from flash to ram and execute it.

*If you were using a A9M9750 module on a A9Vali_0 board, UBOOT serial output is not directly accessible in A9M9750DEV_0. You have to update uboot bootloader. See chapter "xxxxx"

U-Boot 1.1.0 (Jan 7 2005 - 17:52:10)

U-Boot code: 00F80000 -> 00F9E274 BSS: -> 00FA26E8

IRQ Stack: 00f5fd7c

FIQ Stack: 00f5ed7c

RAM Configuration:

Bank #0: 00000000 16 MB

Flash: 0 kB

NAND:32 MB

In: serial

Out: serial

Err: serial

Hit any key to stop autoboot: 3,2,1,0

NAND read: device 0 offset 2097152, size 1048576 ... 1048576 bytes
read: OK

Starting application at 0x00200024 ...

NET+WORKS Version 6.1 Release 1.2 Build on Mar 14 2005

Copyright (c) 2000-2004, NETsilicon, Inc.

PLATFORM: a9m9750

APPLICATION: Sample application for NET+OS6.01

NETWORK INTERFACE PARAMETERS:

The board will obtain IP configuration parameters from the network.

HARDWARE PARAMETERS:

Serial channels will use a baud rate of 38400

This board's serial number is N99999999

This board's MAC Address is 00:04:F3:00:23:88

After board is reset, start-up code will wait 5 seconds

Default duplex setting for Ethernet connection: phy Default

Press any key in 5 seconds to change these settings.

ACE: Have IP address on interface eth0: 192.168.50.50

ACE: Have IP address on interface eth0: 192.168.50.50

Starting NAND Flash file system ...

NAND File System Started OK

Hello World!

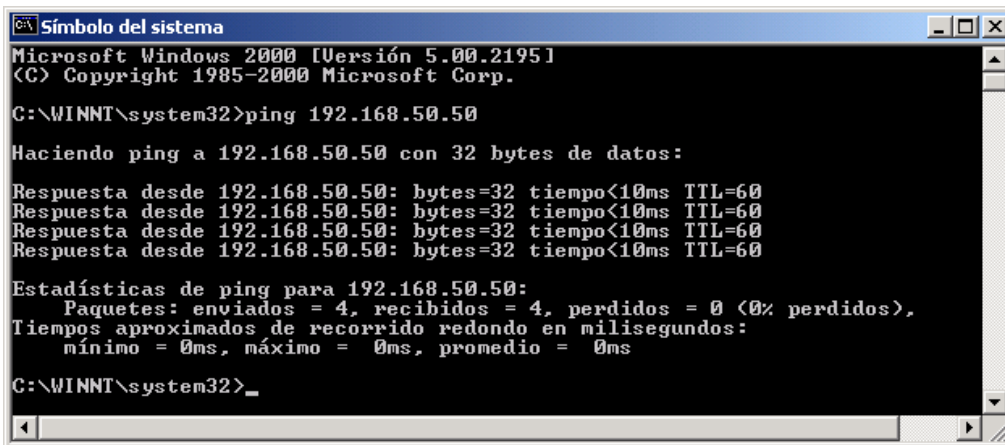
*Note that some of above messages are displayed because BSP was compiled with: make
DEBUG=1

Default NetOS image have DHCP enabled because of following define in /src/apps/template/appconf.h:

```
#define APP_USE_STATIC_IP          FALSE
```

If you don't have a DHCP server in your network your application won't start. You should then reset the board and interrupt the NetOS dialog by pressing any key and modifying them not to obtain the IP settings from the network. (Default root Password is **a9m9750**). Then the Values of ipaddr and netmask programmed in UBOOT will be used (See chapter "About NVRAM").

Now on the PC start a command session and try to ping the IP address reported for your target .



```
Símbolo del sistema
Microsoft Windows 2000 [Versión 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\WINNT\system32>ping 192.168.50.50

Haciendo ping a 192.168.50.50 con 32 bytes de datos:

Respuesta desde 192.168.50.50: bytes=32 tiempo<10ms TTL=60
Respuesta desde 192.168.50.50: bytes=32 tiempo<10ms TTL=60
Respuesta desde 192.168.50.50: bytes=32 tiempo<10ms TTL=60
Respuesta desde 192.168.50.50: bytes=32 tiempo<10ms TTL=60

Estadísticas de ping para 192.168.50.50:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0 (0% perdidos),
    Tiempos aproximados de recorrido redondo en milisegundos:
        mínimo = 0ms, máximo = 0ms, promedio = 0ms

C:\WINNT\system32>
```

5.1. Building a NET+OS Example Application

Before we start with a NET+OS example, you should re-build libflash.a, libnand_fs.a, libposix.a and libbsp.a to make sure they can be built in your environment.

You must first build these libraries if you have installed the package in a different directory to "/targets/NetOS61_GNU_V102".

First build libflash.a. Open a bash shell and change directory to "/targets/NetOS61_GNU_V102". Then enter:

```
make -f flash.mak
```

A new libflash.a library will be created in "/targets/NetOS61_GNU_V102/lib/32b".

Second (**Only** if you have a full version of the NAND file system) build libnand_fs.a. Enter:

```
make -f nand_fs.mak
```

A new libnand_fs.a library will be created in "/targets/NetOS61_GNU_V102/lib/32b".

Third build libposix.a. Enter:

```
make -f posix.mak
```

A new libposix.a library will be created in "/targets/NetOS61_GNU_V102/lib/32b".

Now rebuild the BSP. Change directory to "/targets/NetOS61_GNU_V102/src/bsp" and enter¹:

```
make clean
make
*1
```

The BSP will be re-built and a new libbsp.a library created in "/targets/NetOS61_GNU_V102/lib/32b".

Now is the time to build the example.

Change directory to "/targets/NetOS61_GNU_V102/src/apps/template/32b". Enter:

```
make clean
make
```

Application files will be built.

The image.uboot file is the binary image which can be executed with UBOOT bootloader. See next chapter "load and execute a NetOS image with UBOOT Bootloader" to learn how to download and execute it.

This example is the one used to create the image used as the factory default, i.e. the one which is in the flash when the Developer's Kit leaves the factory.

Image.elf is the image in ELF format, containing debug information. See chapter "debugging a NetOS image" to learn how to download and execute it.

¹ If you want some debug info in your BSP type "make DEBUG=1"

5.2. load and execute a NetOS image with UBOOT Bootloader

1. Open a Serial terminal in you host computer at 38400 Bauds, 8 data bits, Non Parity, 1 stop bit and no handshake.
2. Connect your computer to serial port 1 of the A9M9750DEV_0 (Connector X18).
3. Boot the board, you should see some messages and the uboot prompt (A9M9750 #). Press control-C to interrupt autoboot.
4. Configure your board ip and your host computer ip (a computer with a tftp server) by:
 - a. type in the terminal: `setenv ipaddr 192.168.50.53`
 - b. type in the terminal: `setenv serverip 192.168.50.193`
 - c. Store this new values with the command 'saveenv'. This will save environment to eeprom.
5. Make sure your tftp server Outbound Path is pointing to your image.bin, if not copy the image.uboot there or modify the makefile of your example to do it automatically
6. type in the terminal following command to download this image.uboot to your target: `tftp 0x200000 image.uboot`
7. type in the terminal following command to execute it: `go 0x200024`

If you compiled your bsp with option `DEBUG=1` you should see a couple of messages immediately.

After some seconds the Net+OS user interface menu should appear, and then, the application starts.

8. You can create some scripts in UBOOT to make things easier (We have already done the scripts for you and programmed them in the UBOOT eeprom):
 - a. Download and execute an image:
 - type in the terminal:
`setenv Netos tftp 0x200000 image.uboot\;go 0x200024`
 - Now save new environment variable typing: `saveenv`
 - To execute the new script type: `run Netos`
 - b. Execute an image stored in nand flash:
 - first download an image to SDRAM as you done before: `tftp 0x200000 image.uboot`

- then save it to nand flash by typing:

```
nand erase 0x200000 0x100000
```

```
nand write 0x200000 0x200000 0x100000
```
- Create the script: `setenv NetosFlash nand read 0x200000 0x200000 0x100000\;go 0x200024`
- Now save new environment variable typing: `saveenv`
- You can reboot and execute the new script. type: `run NetosFlash`

5.3. Debugging a NetOS image

follow this instructions depending on the debugger being used:

5.3.1. MAJIC Debugger

- a. Check that there is a `gdbinit.majic` file in the `32b` directory of your application and that it has at the end (in the monitor `L` command) the correct **absolute** path of your `image.elf` file (majic does not support a relative path). This is already done for this template example but must be done for other applications. We aware that this file have a piece of code necessary to change the processor to big endian before downloading starts that makes **not necessary** to change the endianes with UBOOT.
- b. Check that your MAJIC is connected to the Ethernet. If you have problems, try pinging your MAJIC to verify that it's not a network problem.
- c. Check that the 20 pin JTAG cable of the MAJIC is connected to X1 connector of the A9M9750DEV_0 board.
- d. Set bits 1 and 3 of switch S2 to ON (up). This enables debugging over JTAG on the NS9750 processor
- e. Power cycle the dev board and the MAJIC, the MAJIC's status light should be green.
- f. Click on the MDI server icon on your desktop (you have created this during installation of the MAJIC debugger). A window that states: "Listening on port 2345" should appear.

If you get an error "cygwin1.dll not found", add "c:\cygwin\bin" to your system path.

- g. From the directory where your `image.elf` and `gdbinit.majic` files are located, type:

```
arm-elf-gdb -se image.elf -x=gdbinit.majic
```

- h. You should see activity in the MDI server window as the download proceeds. You should also look for errors through this window. For example, occasionally your image.elf is deleted. Remove it and restart arm-elf-gdb. The download will take some time while you will see the Ethernet led of the MAJIC debugger blink.

When the download is complete the Insight debugger window appears with the program stopped at the first breakpoint “main”. This breakpoint was defined in the gdbinit.majic script.

You can now either single-step through code or hit continue to allow the program to run till the next breakpoint – defined as “applicationStart”. Before reaching the applicationStart routine, the program will output a dialog menu on the serial port, giving you the option to change parameters such as IP address. After the “applicationStart” breakpoint has been reached, hit ‘continue’ again. The program starts the web server and waits for a request from a browser.

Now on the PC start a command session and try to ping the IP address of the target .

Note: each time you exit from gdb, the mdi-server process also quits. Consequently you must restart mdi-server before you start another session.

You can also start GDB using the command line interface by using the option “-nw” for no window. This method is recommended if you’re having problems with Insight, since this method is more verbose.

```
arm-elf-gdb -nw -se image.elf -x=gdbinit.majic
```

5.3.2. Chameleon Debugger

- a. Check that following files are available at “/targets/NetOS61_GNU_V102” directory:
 - gdbinit.cham: initialization file for a9m9750 for the gdb
 - gdbinitBig: initialization file for the gdb that changes the processor into big endian.
 - cham_dbg.sh: Script that starts the gdbremote application, changes processor into big endian and starts debugging.

It was not possible to integrate the piece of code to change into big endian inside gdbinit.cham so a script has been created to start the all debug process in one go.

- b. Check that the DB25 connector of the Chameleon is connected to the parallel port of your host computer.

- c. Check that the 20 pin JTAG cable of the Chameleon is connected to X13 connector of the A9M9750DEV_0 board.
- d. Set bits 1 and 3 of switch S2 to ON (up). This enables debugging over JTAG on the NS9750 processor
- e. Power cycle the dev board.
- f. Power cycle Chameleon, Red and green leds must light continuously. Yellow led should not light (jumper in low position).
- g. The OCDRemote utility needs to be started first. So open a second bash shell by clicking on the Cygwin desktop icon and change directory to /usr/local/bin. Here you should find the utility "OCDRemote.exe". Start it by entering:

```
$. /OCDRemote -c ARM9 -p 8888 -a 1 -s 1 -d RAVEN
```

It should print in the screen:

```
ocdremote 2.06: RAVEN via LPT 1 at speed : 1
JTAG SDO <-| CPU(1) ARM9 : listening on port 8888 |<- JTAG SDI
```

- h. Go to the window where you built the application From the directory where your image.elf is located, type:

```
/targets/NetOS61_GNU_V102/ cham_dbg.sh
```

This script will launch an instance of ocdremote in background and starts arm-elf-gdb just to change the processor into big endian. When it finish, launch another instance of ocdremote in background and starts normal arm-elf-gdb.

- i. The download will take some time while you will see the red led of the Chameleon debugger blink.

When the download is complete the Insight debugger window appears with the program stopped at the first breakpoint "main". This breakpoint was defined in the gdbinit.cham script.

You can now either single-step through code or hit continue to allow the program to run till the next breakpoint – defined as "applicationStart". Before reaching the applicationStart routine, the program will output a dialog menu on the serial port, giving you the option to change parameters such as IP address. After the "applicationStart" breakpoint has been reached, hit 'continue' again. The program starts the web server and waits for a request from a browser.

Now on the PC start a command session and try to ping the IP address of the target .

Note: After several debugging sessions, some ocdremote instances may be working in the background. You can check them with the "ps" command and finish them with the "kill" command.

You can also start GDB using the command line interface by using the option "-nw" for no window. This method is recommended if you're having problems with Insight, since this method is more verbose. Modify the file cham_dbg.sh

```
arm-elf-gdb -nw -se image.elf -x=/targets/NetOS61_GNU_V102/gdbinit.cham
```

5.4. About NAND Flash File System

NetOS has an integrated filesystem but, unfortunately, this file system only supports NOR flashes.

Hcc Embedded is a partner company that has provided a NAND Flash system.

We have adapted the NAND Flash driver to our module, integrated it into the Operating System and adapted http and ftp server applications so they can access it.

5.4.1. General Information

A9M9750 has a 32 Mbytes NAND Flash model k9f5608.

To make use of the NAND flash on A9M9750 from your applications we have ported to our module a NAND flash file system from company hcc-embedded.

The library to support this NAND flash is /lib/32b/libnand_fs.a and is only for evaluation, with the following restrictions:

- Total number of directory entries = 5

If you get the license for this product, the library won't have any restriction and you'll also get sources of the driver at /src/nand_fs. Then you could make modifications to the driver and rebuild it as explained in subchapter "HCC EDFS-STD NAND Full Version Upgrade"

Hcc-Embedded NAND file system has the following features:

EFFS-STD is a file system targeted at embedded devices which use NOR or NAND type flash devices that require a high degree of reliability. The system is completely protected against unexpected power-failure or reset.

- 100% Power Fail Safe
- Long Filenames
- Multiple Volumes
- Directory Handling
- Multiple simultaneous files open
- Mix of media types
- Wear-Leveling (Static and Dynamic)
- ECC algorithms
- Bad Block handling
- Standard File API
- Secure File API
- POSIX API
- Reserved Sectors
- ANSI C compliant C source
- Optional Secure API
- Sample Application code
- Sample Drivers for several NOR and NAND devices
- Detailed Implementation Guide

Use example

We provide an example useful to learn how to use this NAND file system from your application. The example is at `/src/a9m9750_examples/naNAND_Filesys`. It will show you how to format the file system, create directories, change to directories, read and write to files....

To build and test this example,

Change directory to `"/targets/NetOS61_GNU_V102/src/a9m9750_examples/naNAND_Filesys/32b"`. Enter:

```
make clean
make
```

Application files will be built. The `image.uboot` file is the binary image which can be executed with UBOOT bootloader. See chapter "load and execute a NetOS image with UBOOT Bootloader" to learn how to download and execute it. Once application starts, the first time the example is executed, you should see in the console:

```
Starting NAND Flash file system ...
NAND Flash NOT formatted. Formatting now ...
NAND Flash formatted OK
NAND File System Started OK
Creating Directory a9m9750
Appending data to file
```

```
Linsting file
Hello
File finished
```

The application starts the file system automatically if not previously started by the BSP (See chapter xxxx). If the file system is not formatted, it format it.

If directory a9m9750 doesn't exist (first time we execute the test), it is created.

Then every time you execute the example text "Hello" will be appended to the file file.bin so next time it is executed, you should see:

```
Starting NAND Flash file system ...
NAND File System Started OK
Appending data to file
Linsting file
HelloHello
File finished
```

And next

```
Starting NAND Flash file system ...
NAND File System Started OK
Appending data to file
Linsting file
HelloHelloHello
File finished
```

5.4.2. HCC EFFS-STD NAND Full Version Upgrade

Library coming in the release for a9m9750 to support the nand flash is /lib/32b/libnand_fs.a and is only for evaluation, with following restrictions:

- Total number of directory entries = 5

After getting the upgrade for this product, you will install the complete sources at /src/nand_fs and after building them, the library won't have any restriction.

- If haven't done yet, install sources by following instructions on chapter "Installing HCC EFFS-STD NAND Full Version Upgrade"

- Change to directory "/targets/NetOS61_GNU_V102". Enter:

```
make -f nand_fs.mak
```

A new libnand_fs.a library will be created in "/targets/NetOS61_GNU_V102/lib/32b".

5.4.3. File System Size & Position

You can change the size and position of the file system in flash by changing defines in file k9f5608x0c_8bit.c and rebuilding libnand_fs.a

Here you can see current values:

```
/*
 * Flash Device specific definitions
 * K9F5608X0C Samsung
 */
***** /

#define BLOCK_SIZE    0x4000 /* 16k */
#define SECTOR_SIZE   0x1000 /* 4k */
#define BLOCK_NUM     2048 /* 2048 block all */
#define PAGE_SIZE     512 /* page size is 512 byte */
// #define BLOCK_START 1 /* 1st block is kept for any purpose */
#define BLOCK_START   1024 /* Second half of the flash for File
                           Sys (16 MBytes) */
#define SEPARATE_DIR 1 /* directory is in separated block from
                       fat */
```

Here we are defining 2048 blocks of 16k each, which is a total of 32 Mbytes, but we are defining block 1024 as the start block so first half of the flash is free (for example to store other NetOS Images or other OS). Second 16 Mbytes of the flash will be used for the NAND flash system.

You can change BLOCK_NUM and BLOCK_START to different values. But take care they follow rules described in hcc-embedded documentation NetOS61_GNU_V102\src\nand_fs\docs\leffs_160.pdf. You should use utility NetOS61_GNU_V102\src\nand_fs\utils\fsmem.exe to test correct values for the file system.

Then rebuild the libnand_fs.a:

Change directory to "/targets/NetOS61_GNU_V102". Then enter:

```
make -f nand_fs.mak
```

If you ever want to completely erase the file system, you can use following UBOOT bootloader:

```
nand erase 0x1000000 0x1000000
```

5.4.4. Stress Example

We provide another example at /src/a9m9750_examples/naNAND_Filesys_Stress used for exercising the system and ensuring that the file system is working correctly. Most

functionality of the file system is exercised with this program including file read/write/append/seek/file content, directories and file manipulation functions.

Att: This example will only work fine if you have a full version of the hcc-embedded NAND file system. Errors will appear if you are using the evaluation version due to the limitation of 5 directories.

To build and test this example,

Change directory to `"/targets/NetOS61_GNU_V102/src/a9m9750_examples/naNAND_Filesys_Stress/32b"`. Enter:

```
make clean
make
```

Application files will be built. The image.uboot file is the binary image which can be executed with UBOOT bootloader. See chapter "load and execute a NetOS image with UBOOT Bootloader" to learn how to download and execute it. Once application starts, you should see in the console the result of tests while they are being executed:

```
NET+WORKS Version 6.1 Release 1.2 Build on Mar 14 2005
Copyright (c) 2000-2004, NETsilicon, Inc.

PLATFORM: a9m9750
APPLICATION: Application for Stress testing hcc-embedded NAND file system on
A9M9750
-----
NETWORK INTERFACE PARAMETERS:
  IP address on LAN is 192.168.50.50
  LAN interface's subnet mask is 255.255.255.0
  IP address of default gateway to other networks is 192.168.50.5
HARDWARE PARAMETERS:
  Serial channels will use a baud rate of 38400
  This board's serial number is N99999999
  This board's MAC Address is 00:04:F3:00:23:88
  After board is reset, start-up code will wait 5 seconds
  Default duplex setting for Ethernet connection: phy Default
-----
Press any key in 5 seconds to change these settings.

ACE:  Have IP address on interface eth0: 192.168.50.50

Starting NAND Flash file system ...
NAND File System Started OK
HCC_EFFS_STD ver:1.60
Start: (HH:MM:SS) 15:44:39
File system test started...
fsm_formatting
passed...
fsm_dirtest
passed...
fsm_findingtest
passed...
fsm_powerfail
passed...
fsm_seeking with 128
passed...
```

```
fsm_seeking with 256
passed...
fsm_seeking with 512
passed...
fsm_seeking with 1024
passed...
fsm_seeking with 2048
passed...
fsm_seeking with 4096
passed...
fsm_seeking with 8192
passed...
fsm_seeking with 16384
passed...
fsm_seeking with 32768
passed...
fsm_opening
passed...
fsm_appending
passed...
fsm_writing
passed...
fsm_dots
passed...
fsm_flushing
passed...
fsm_rit
passed...
End of tests...
End: (HH:MM:SS) 15:50:20
Elapsed: 341 secs
```

5.4.5. Remote Example

Example at `/src/a9m9750_examples/naNAND_Filesys_Remote` provides an http and ftp Servers that have access to the NAND File system.

Att: If you don't have a full version of the hcc-embedded NAND file system, errors will appear if you try to create more than 5 entries (files/directories) in the file system.

To build and test this example,

Change directory to `"/targets/NetOS61_GNU_V102/src/a9m9750_examples/naNAND_Filesys_remote/32b"`. Enter:

```
make clean
make
```

Application files will be built. The `image.uboot` file is the binary image which can be executed with UBOOT bootloader. See chapter "load and execute a NetOS image with UBOOT Bootloader" to learn how to download and execute it. Once application starts, you should see how the file system is initialized and the server started.

```
ACE: Have IP address on interface eth0: 192.168.50.50
```

```
Begin the A9M9750 NAHTTP Server over NAND File System application.  
Starting NAND Flash file system ...  
NAND File System Started OK  
File system already started for CLibrary.  
Application running.
```

5.4.5.1. ftp server

Open a dos box and start a ftp sesion to the target IP. Do not enter any login or password.

```
C:\WINNT\system32>ftp 192.168.50.50  
Connected to 192.168.50.50.  
220 NET+ARM FTP Server 1.0 ready.  
User (192.168.50.50:(none)):  
230 User (none) logged in.  
ftp>
```

Now you can process with standard ftp commands: dir, mkdir, rmdir, put, get

Remember that in the trial version of the file system we have a maximun of 5 entries in the file system. In the following screen shot we delete files created by previous examples and copy some images and web pages to the file system:

```
ftp> dir  
200 PORT command Ok.  
150 File Listing Follows in ASCII mode  
  
Current directory: /  
a9m9750      <dir>    2028-01-04 22:55:54  
      0 file(s)      0 bytes  
      1 dir(s)      16691200 bytes free  
  
226 Transfer complete.  
ftp: 111 bytes received in 0.11Segundos 1.01KB/s.  
ftp> cd a9m9750  
250 Directory is changed  
ftp> dir  
200 PORT command Ok.  
150 File Listing Follows in ASCII mode  
  
Current directory: /a9m9750  
..          <dir>    1980-00-00 00:00:00  
file.bin    25      2028-01-04 23:02:40  
      1 file(s)      25 bytes  
      1 dir(s)      16691200 bytes free  
  
226 Transfer complete.  
ftp: 155 bytes received in 0.11Segundos 1.41KB/s.  
ftp> delete file.bin  
250 DELE command successful.  
ftp> cd ..  
250 Directory is changed
```



```
ftp> rmdir a9m9750
250 RMD command successful.
ftp> dir
200 PORT command Ok.
150 File Listing Follows in ASCII mode

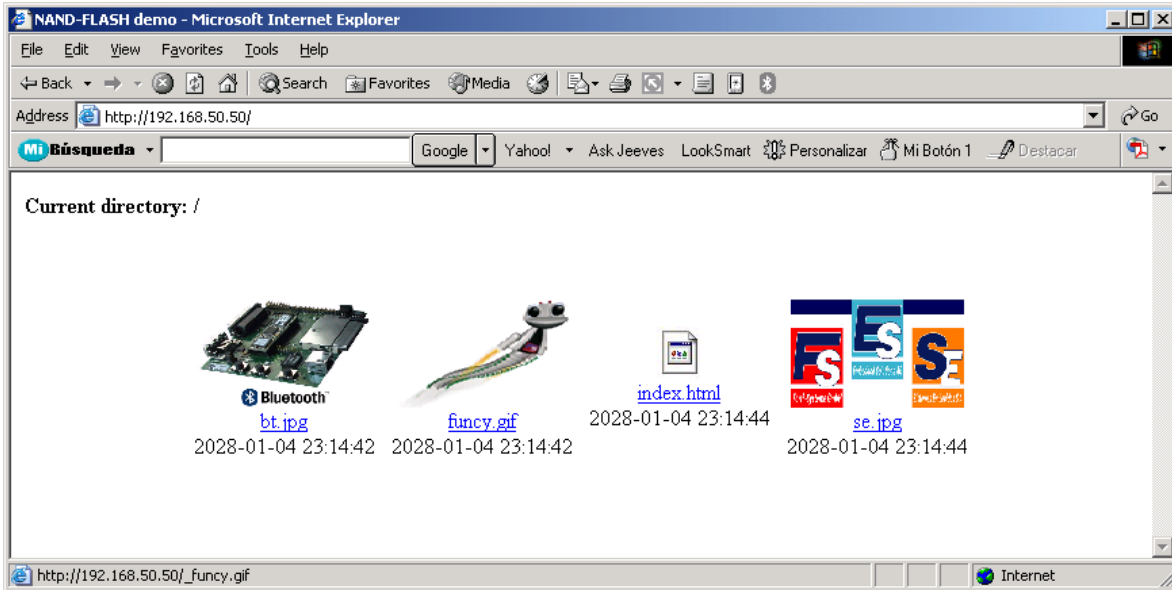
Current directory: /
Card error: 7
226 Transfer complete.
ftp: 36 bytes received in 0.11Segundos 0.33KB/s.
ftp> mput *
mput bt.jpg? y
200 PORT command Ok.
150 About to open data connection.
226 Transfer complete
ftp: 42164 bytes sent in 0.00Segundos 42164000.00KB/s.
mput funcy.gif? y
200 PORT command Ok.
150 About to open data connection.
226 Transfer complete
ftp: 3372 bytes sent in 0.00Segundos 3372000.00KB/s.
mput index.html? y
200 PORT command Ok.
150 About to open data connection.
226 Transfer complete
ftp: 1181 bytes sent in 0.00Segundos 1181000.00KB/s.
mput se.jpg? y
200 PORT command Ok.
150 About to open data connection.
226 Transfer complete
ftp: 43850 bytes sent in 0.02Segundos 2740.63KB/s.
ftp> dir
200 PORT command Ok.
150 File Listing Follows in ASCII mode

Current directory: /
bt.jpg          42164    2028-01-04 23:14:42
funcy.gif       3372    2028-01-04 23:14:42
index.html     1181    2028-01-04 23:14:44
se.jpg         43850   2028-01-04 23:14:44
    4 file(s)          90567 bytes
    0 dir(s)          16596992 bytes free

226 Transfer complete.
ftp: 230 bytes received in 0.09Segundos 2.45KB/s.
ftp>
```

5.4.5.2. http server

Open a browser and type the IP of the target. You should see the content of the current directory.



The http example here implemented is a kind of picture viewer. File with extension *.gif and *.jpg are previewed and if you click on them, the picture is viewed full size.

You can also navigate through Web pages and directories. Rest of files appear as binaries.

5.4.6. Integration of the NAND file system with the C-Library

If file /src/bsp/platform/a9m9750/bsp.h there is a define that makes the original NetOS file system (only supporting NOR flashes) be integrated with the C-Library.

```
BSP_INCLUDE_FILESYSTEM_FOR_CLIBRARY
```

We have added one that makes hcc embedded NAND file system be included in stead of original file system:

```
#define BSP_INCLUDE_NAND_FILESYSTEM_FOR_CLIBRARY    TRUE
```

After setting this define to TRUE and rebuilding the BSP, you have two advantages:

- The file system is initialized automatically during BSP start up and also formatted if necessary; so it's not necessary you do it in your applications (naNAND_Filesys, naNAND_Filesys_Extress, naNAND_Filesys_remote ...). Anyway if you try to reinitialize it in the applications, won't be a problem.

```
Starting NAND Flash file system ...
NAND File System Started OK
Begin the A9M9750 NAHTTP Server over NAND File System application.
File system already started for CLibrary.
```

- You will be able to access the NAND file system using standard C functions like open, close, read, write ... What allows an easier porting of applications.

Most of the stuff to do this integration is done in file /src/bsp/platform/a9m9750/startfilesystem.c

To test this integration you can use example at src/a9m9750_examples/nafileio, which is just a copy of the /src/examples/nafileio with the makefile updated for a9m9750

5.5. About NVRAM

5.5.1. Using A9M9750 Serial EEPROM

A9M9750 has a 8Kbyte serial eeprom device (24LC64) connected to the I2C port.

To make NetOS use this eeprom as NVRAM we have set following configuration in /src/bsp/platforms/a9m9750/bsp.h:

```
#define BSP_NVRAM_DRIVER    BSP_NVRAM_CUSTOM
```

In this file there are some more defines a9m9750 specific:

```
#define START_OF_DEV_BOARD_PARAMETERS_NETOS    0x700
```

Here we specify the offset in the eeprom where we want to start storing Netos non volatile ram. Take into account that:

- The total size of the eeprom on a9m9750 (8 Kbytes)
- The size needed for NetOS NVRAM is 0x890 bytes (2192 bytes)
- The area of eeprom used by uboot (see below)

5.5.2. NetOS using UBOOT parameters

We have added a feature that allows NetOS use some of the values configured by UBOOT bootloader. If you set them to TRUE they are read from UBOOT eeprom area automatically and you can't change them in the NetOS dialog:

```
#define USE_UBOOT_ETHADDR           TRUE
#define USE_UBOOT_IPADDR           TRUE
#define USE_UBOOT_NETMASK          TRUE
#define USE_UBOOT_BAUDRATE         TRUE
```

To allow this feature we have to indicate NetOS where UBOOT is storing this parameters in the eeprom.

```
#define START_OF_DEV_BOARD_PARAMETERS_UBOOT    0x500
#define LEN_OF_DEV_BOARD_PARAMETERS_UBOOT      0x200
```

*We should review this values if uboot change them.

5.5.3. Example of using eeprom for user applications

Areas of eeprom not used by NetOS and UBOOT can be used by the user to keep non volatile information.

An example on how to do it is at `/src/a9m9750_examples/nai2capp_eeprom`

5.6. About Real Time Clock

A9M9750 has a Real Time Clock chip (DS1337) connected to the i2c port of the processor.

It can be used to keep the system Date&Time.

An example on how to read/update date & time can be found at `/src/a9m9750_examples/nai2capp_rtc`

5.7. About USB

NS9750 has both USB host and device controllers, but only one at the same time can be used. J4 and J5 of A9M9750DEV_0 has to be set appropriately to the chosen configuration.

5.7.1. USB Device

Set jumpers J4 and J5 in position 2-3 to select usb device.

This example will make the A9M9750 report to be kind of printer when connected to a host.

Build the example:

Change `directory` to `"/targets/NetOS61_GNU_V102/src/a9m9750_examples/nausbdevapp/32b"`. Enter:

```
make clean
make
```

Application files will be built. The image.uboot file is the binary image which can be executed with UBOOT bootloader. See chapter "load and execute a NetOS image with UBOOT Bootloader" to learn how to download and execute it.

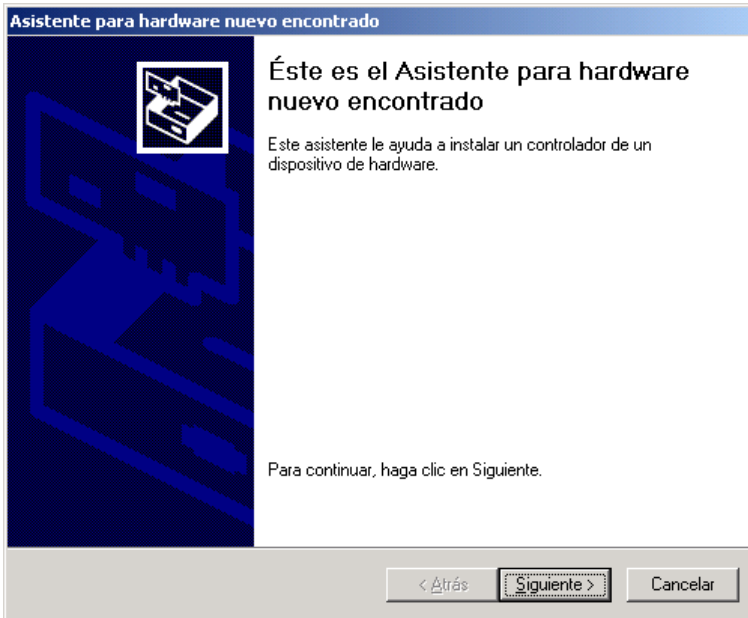
After executing it you should see in the terminal:

```
NET+WORKS Version 6.1 Release 1.2 Build on Mar 14 2005
Copyright (c) 2000-2004, NETsilicon, Inc.

PLATFORM: a9m9750
APPLICATION: Example for time using USB Device
-----
NETWORK INTERFACE PARAMETERS:
  IP address on LAN is 192.168.50.50
  LAN interface's subnet mask is 255.255.255.0
  IP address of default gateway to other networks is 192.168.50.5
HARDWARE PARAMETERS:
  Serial channels will use a baud rate of 38400
  This board's serial number is N00994135
  This board's MAC Address is 00:04:F3:00:23:88
  After board is reset, start-up code will wait 5 seconds
  Default duplex setting for Ethernet connection: phy Default
-----
Press any key in 5 seconds to change these settings.

ACE: Have IP address on interface eth0: 192.168.50.50
ACE: Have IP address on interface eth0: 192.168.50.50
Starting USB Device Test.
size of MC_USB_CONFIGURATION_DESCRIPTOR is 12.
0x90100000=0x823.
0x90600070=0x7.
0x90600018=0x333333bb.
0x90600040=0xffff4e2eb.
```

Now connect a USB cable between USB device connector of the target (X15) and one USB host connectors of your computer. Your computer should detect there is a device and will try to recognize it. A9m9750 reports to be a printer name "CX5000". Your computer will ask you for the proper drivers for the OS you are running. Cancel it.



5.7.2. USB Host

To test usb host you need an additional A9m9750 and A9M9750DEV_0.

In the second A9Vali, set jumpers J4 and J5 in position 1-2 to select usb host.

This example will recognize the device printer used for USB Device example and try to print something through it.

Build the example:

Change `directory` to `"/targets/NetOS61_GNU_V102/src/a9m9750_examples/nausbhost_app/32b"`. Enter:

```
make clean
make
```

Application files will be built. The `image.uboot` file is the binary image which can be executed with UBOOT bootloader. See chapter "load and execute a NetOS image with UBOOT Bootloader" to learn how to download and execute it.

After executing it you should see in the terminal:

```
NET+WORKS Version 6.1 Release 1.0 Build on Mar 14 2005
```

```
Copyright (c) 2000-2004, NETsilicon, Inc.
```

```
PLATFORM: a9m9750
```

```
APPLICATION: USB Host Sample Application
```

```
-----  
NETWORK INTERFACE PARAMETERS:
```

```
IP address on LAN is 192.168.50.50
```

```
LAN interface's subnet mask is 255.255.255.0
```

```
IP address of default gateway to other networks is 192.168.50.5
```

```
HARDWARE PARAMETERS:
```

```
Serial channels will use a baud rate of 38400
```

```
This board's serial number is N99999999
```

```
This board's MAC Address is 00:04:F3:00:23:88
```

```
After board is reset, start-up code will wait 5 seconds
```

```
Default duplex setting for Ethernet connection: phy Default
```

```
-----  
Press any key in 5 seconds to change these settings.
```

```
ACE: Have IP address on interface eth0: 192.168.50.50
```

```
ACE: Have IP address on interface eth0: 192.168.50.50
```

```
USB HID boot protocol mouse registered.
```

Now connect a USB cable between **USB device connector of the first target (X15)** and the **USB host connector of the second target (X14)**. Your second target should detect the first one and will report in the console of the second target:

```
printerProbe: USB printer: Bulk out=1(64) in=82(64)  
usbPrinterConfigure: 0 language id 0x409  
                  [CX5000]  
usbPrinterConfigure: set configuration 1  
usbPrinterConfigure: set interface 0, alternate setting 0
```

Then if you disconnect the cable you should see in the console of the second target:

```
printerDisconnect: dev->devnum = 1 usbPrinter=0x27a360  
printerDisconnect: /dev/usb/printer0 removed
```

5.8. About Serial Ports

NS9750 processor has 4 integrated serial ports that can be used in UART mode or SPI mode. These 4 serial ports (Labeled A,B,C,D) use processor GPIOs that can be used for other functionality.

To make one of these ports work as an UART, we must configure two files in our BSP:

- Configure `bsp/platforms/a9m9750/gpio.h` so the desired port pins are routed outside with serial port functionality.

- Configure bsp/platforms/a9m9750/bsp.h to apply correct driver (UART, HDCL or SPI) to that serial port.

5.8.1. NetOS Release 1.1 with A9Vali_0 Base board

In the A9Vali_0 board, PORTA was connected through a TTL-RS232 conversor to a male DB-9 (X9).

This was the BSP configuration used:

- NetOS61_GNU_V101/src/bsp/platforms/a9m9750/gpio.h:

```
#define BSP_GPIO_MUX_SERIAL_A    BSP_GPIO_MUX_SERIAL_8_WIRE_UART
#define BSP_GPIO_MUX_SERIAL_B    BSP_GPIO_MUX_SERIAL_8_WIRE_UART
#define BSP_GPIO_MUX_SERIAL_C    BSP_GPIO_MUX_INTERNAL_USE_ONLY
#define BSP_GPIO_MUX_SERIAL_D    BSP_GPIO_MUX_INTERNAL_USE_ONLY
```

- NetOS61_GNU_V101/src/bsp/platforms/a9m9750/bsp.h:

```
#define BSP_SERIAL_PORT_1        BSP_SERIAL_UART_DRIVER
#define BSP_SERIAL_PORT_2        BSP_SERIAL_UART_DRIVER
#define BSP_SERIAL_PORT_3        BSP_SERIAL_NO_DRIVER
#define BSP_SERIAL_PORT_4        BSP_SERIAL_NO_DRIVER
```

These configuration made two serial ports available: COM0 and COM1.

Setting APP_DIALOG_PORT and APP_STDIO_PORT to "/com/0" in the appconf.h of your application made the console appear through X9.

5.8.2. NetOS Release 1.2 with A9M9750DEV_0 Base board

In the A9M9750DEV_0 board, None of the processor internal serial ports (PortA-PortD) is available on a DB-9 connector. They are available in the expansion connectors in TTL level (would need an external TTL-RS232 converter to connect to a PC).

An external UART ST16C654C (16C550 compatible) has been placed on A9M9750DEV_0 and connected through a TTL-RS232 converter to a male DB-9 (X18).

This has been done to free some processor pins now needed to control some of the peripherals like the touch screen controller, sound chip

A new driver called BSP_SERIAL_EXTERNAL_UART has been provided to make a com port work over the external uart.

This is the BSP configuration used:

- NetOS61_GNU_V102/src/bsp/platforms/a9m9750/gpio.h:

```
#define BSP_GPIO_MUX_SERIAL_A    BSP_GPIO_MUX_SERIAL_2_WIRE_UART
#define BSP_GPIO_MUX_SERIAL_B    BSP_GPIO_MUX_SERIAL_2_WIRE_UART
#define BSP_GPIO_MUX_SERIAL_C    BSP_GPIO_MUX_INTERNAL_USE_ONLY
```



```
#define BSP_GPIO_MUX_SERIAL_D BSP_GPIO_MUX_INTERNAL_USE_ONLY
```

- NetOS61_GNU_V102/src/bsp/platforms/a9m9750/bsp.h:

```
#define BSP_SERIAL_PORT_1 BSP_SERIAL_UART_DRIVER  
#define BSP_SERIAL_PORT_2 BSP_SERIAL_UART_DRIVER  
#define BSP_SERIAL_PORT_3 BSP_SERIAL_EXTERNAL_DRIVER  
#define BSP_SERIAL_PORT_4 BSP_SERIAL_NO_DRIVER
```

These configuration made three serial ports available: COM0, COM1 and COM2.

COM0 and COM1 are available in the expansion connectors in TTL level.

COM2 is available on mail DB-9 connector X18

Setting APP_DIALOG_PORT and APP_STDIO_PORT to "/com/2" in the appconf.h of your application made the console appear through X18.

5.8.3. Update UBOOT from A9Vali_0 to A9M9750DEV_0

If you we using an A9M9750 module on a A9Vali_0 board, it's quite sure you were using an uboot bootloader with input-output through internal serial portA (X9). It should report following compilation date: Aug 13 2004

Now, if you want to use the module on a A9M9750DEV_0 board, you should replace uboot bootloader so it uses input-output through the external uart serial port (X18).

You can do this update process by:

1) Using uboot itself:

- Copy A9M9750.bin image from the uboot/A9M9750DEV_0 directory of the CD to your tftp server Outbound Path.
- Power on the module on the A9Vali_0 (X9 connected to your host PC). Interrupt the auto boot sequence.
- Type following commands

```
nand erase 0x0 0x20000  
tftp 0x10000 A9M9750.bin  
nand write 0x10000 0 0x20000
```

2) Using jtag booster:

- Open a dos box on the directory where you installed the jtag booster software.
- Type: JTAG9750 /pnand A9M9750.bin /LPT-BASE=378

Now if you put the module on the A9M9750DEV_0 (X18 connected to your host PC) you should see uboot console reporting following compilation date: Jan 7 2005

6. Restoring the Factory Default Image

Should you wish to restore the contents of flash to the original condition as delivered from the factory, then you need to burn the image image_12.uboot, which can be found on the CD, into flash using UBOOT software.

Follow instructions on point 8b in chapter "load and execute a NetOS image with UBOOT Bootloader" using image_12.uboot in stead of image.uboot.

The image_12.bin image contains the template example and some debug information.

HAVE FUN DEVELOPING WITH THE A9M9750!