

AKA Embedded 开放实验室系列普及讲座之一

ARM Evaluator-7T

评估板开发简介

报告人：李明

lmcs00@mails.tsinghua.edu.cn

2003-11-1

Arrangement

- Section 1
ARM Evaluator-7T Installation Guide
- Section 2
ARM Evaluator-7T Board User Guide
- Section 3
Demo of my examples in the manual
- Section 4
Q & A

Section 1

ARM Evaluator-7T Installation Guide

Outline

- Introduction
- Setting-up the Software and Hardware
- Building a Sample Application
- Downloading the Image
- Running the Sample Application
- Debugging an Application
- Further Information
- References

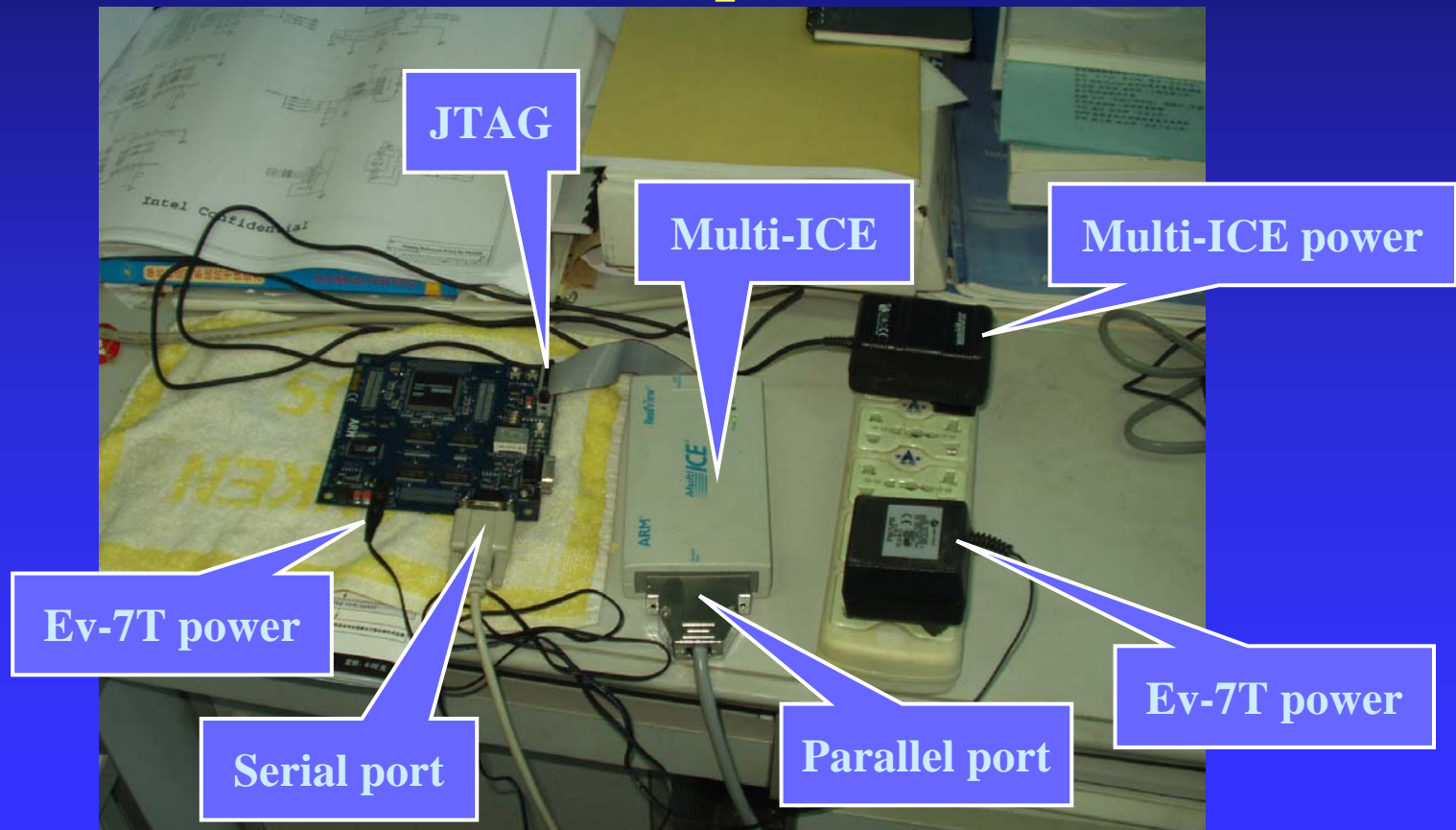
Introduction (1)

■ Evaluator-7T package contains:

- ARM Developer Suite (ADS) Free Evaluation Version v1.0.1, or later, CD-ROM
- ARM Evaluator-7T Board Tools and Documentation CD-ROM
- 9-pin straight through RS-232 serial cable
- Evaluator-7T board
- 9V power adapter.

Introduction (2)

■ Evaluator-7T development environments



Setting-up Software & Hardware

- **Installing the Evaluator-7T Tools and Documentation**
- **Installing the ADS Evaluation Software**
- **Setting up the Evaluator-7T Hardware**
 1. Connect the serial cable between the COM1 Debug port on the Evaluator-7T and the host PC.
 2. Connect the power adapter to the power connector on the Evaluator-7T
 3. Connect the power adapter to the power socket.
 4. Press the System Reset button on the Evaluator-7T.

Building a Sample Application

■ using CodeWarrior supplied with ADS

1. Start CodeWarrior from the Windows Start->Programs->ARM Developer Suite menu.
2. When CodeWarrior has loaded, select Open from the File menu option.
3. Find the Pascal's Triangle example,
c:\Evaluator7T\source\examples\pascal
4. Select the project file, pascal (filename pascal.mcp)
5. Press the Open button. A project window is displayed with the title pascal.mcp
6. Select the Make option from the Project menu. The Pascal's Triangle is compiled and linked.

Downloading the Image (1)

■ using ARM eXtended Debugger (AXD)

1. Press the System Reset button on the Evaluator-7T board.
2. From CodeWarrior, select Debug from the Project menu to download the Pascal's Triangle example image..
3. In AXD, click on the Add button.
4. Select remote_a.dll from the Open dialog box. This installs the driver that is used to communicate with the Evaluator-7T.

Downloading the Image (2)

■ using ARM eXtended Debugger (AXD)

5. In the Choose Target dialog, ensure that ADP/Remote_A.dll is selected and then click on Configure. The Remote_A Connection dialog is displayed.
6. Click on the Select button. The Available Connection Driver dialog box is displayed.
7. Select the ARM serial driver then click on OK button.
8. The Remote_A Connection dialog is displayed.
9. Now click on the Configure button.

Downloading the Image (3)

■ using ARM eXtended Debugger (AXD)

10. Set the serial port to the COM port you used on the host PC
11. Set the buad rate to 38400
12. Click on the OK button in the Setup serial connection dialog.
13. Click on the OK button in the Remote_A connection dialog.
14. In the Choose Target dialog, ensure that the ADP option is highlighted and click on the OK button. AXD starts to download the image to the Evaluator-7T.

Running the Sample Application

■ Start the example from AXD

1. Select Go from the Execute menu.

The program stops at the main() C function.

2. Select Go again.

The LEDs on the Evaluator-7T flash once and the console window of AXD displays the following:

*** Pascal's Triangle ***

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

Debugging an Application (1)

■ Start the example from AXD

1. Select Go from the Execute menu. The program stops at the main()function.
2. Scroll up to the pascal_triangle function in the Executing pascal.c window.
3. Click on the line containing for (j=1;j<depth;j++){
4. Select Toggle Breakpoint from the Execute menu.
5. The for (j=1;j<depth;j++){ line has a red dot. At this point the Evaluator-7T board lights D1, D2, and D3 LEDs.

Debugging an Application (2)

■ Start the example from AXD

6. Select Go again from the Execute menu. The debugger executes all the lines of code between the current point and the breakpoint. When the breakpoint is encountered, the debugger halts the program.

Only D1 and D2 are active.

7. Select Variables from the Processor Views menu. All the local variables in the pascal_triangle function are displayed.

8. Select Go from the Execute menu. The program continues until completion.

Further Information

■ An email list

you can share information with other Evaluator-7T users.

- To subscribe, send an email to
subscribe-evaluator7t@arm.com
- To query other Evaluator-7T users by sending email to
evaluator7t@arm.com
- To unsubscribe, send an email to
unsubscribe-evaluator7t@arm.com
- For additional information on CodeWarrior IDE and AXD use the online help provided with ADS.

References

- **ARM documentation**

- **Recommended website**

<http://www.arm.com>

- **Recommended books**

- ARM Architecture Reference Manual, edited by Dave Jaggar, ISBN 0-13-736299-4
- ARM System Architecture, by Steve Furber, ISBN 2-201-40352-8.

- **Further reading**

- Samsung KS32C50100 Embedded Network Controller User's Manual v0.3

Section 2

ARM Evaluator-7T Board User Guide

Outline

- Introduction
- Hardware Description
- Setting up the Evaluator-7T
- Programmers Reference
- Basic setup with the BSL
- BSL commands
- Demo Application of using BSL
- Practices

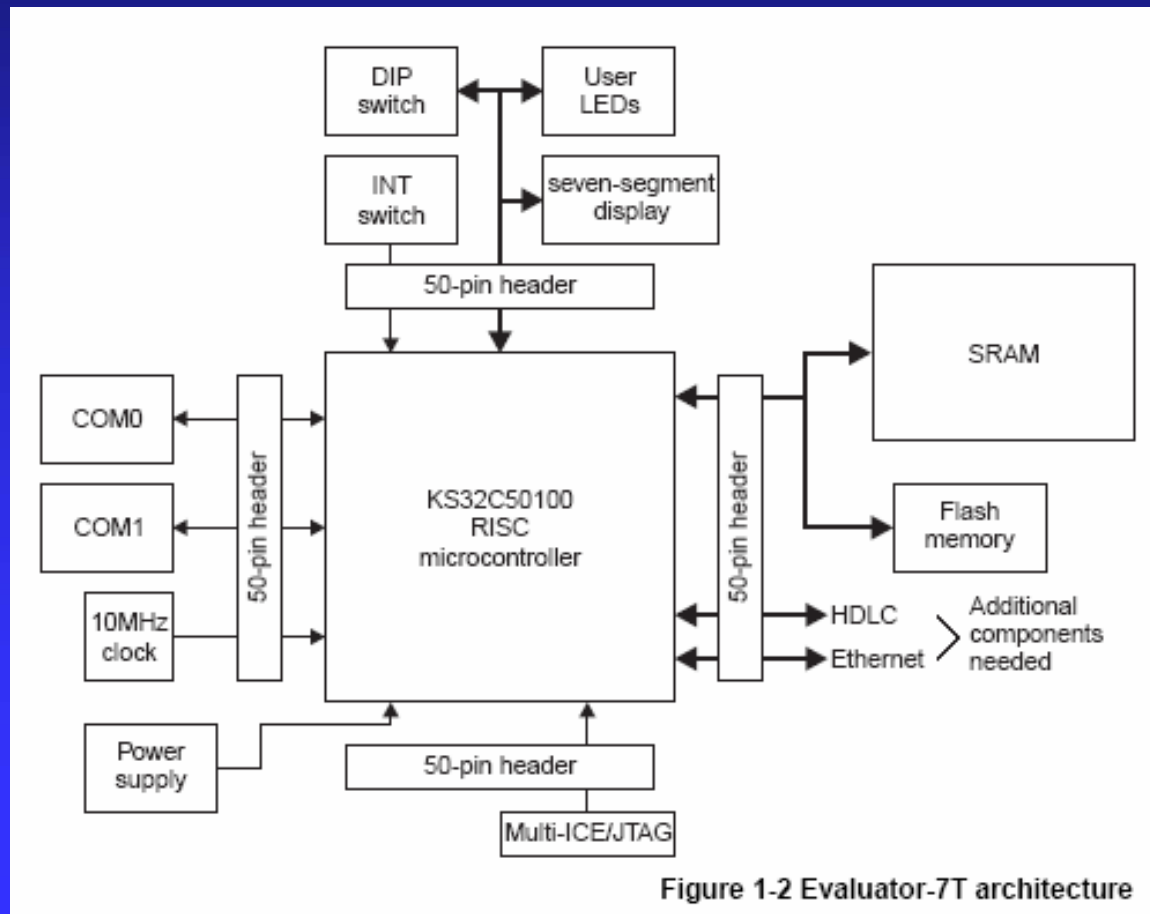
Introduction (1)

■ Evaluator-7T architecture

- Samsung KS32C50100 microcontroller
- 512KB flash EPROM
- 512KB SRAM
- two 9-pin D-type RS232 connectors
- reset and interrupt push buttons
- four user-programmable LEDs and a seven-segment LED display
- 4-way user input DIP switch
- Multi-ICE connector
- 10MHz clock (the processor uses this to generate a 50MHz clock)
- 3.3V voltage regulator.

Introduction (2)

■ Evaluator-7T architecture



Hardware Description

- The Samsung KS32C50100 microcontroller
- Memory (Flash、SRAM)
- Serial ports (DEBUG、USER)
- LEDs
 - four surface-mounted LEDs
 - a seven-segment LED display
- Switches
 - DIP switch
 - User interrupt switch
- JTAG port

Setting up the Evaluator-7T

■ Evaluator-7T can be used in the following ways:

- Using the bootstrap loader

The BootStrap Loader (BSL) is a component of the resident firmware preloaded into the bottom of the flash memory

- Using the Angel debug monitor

The Angel debug monitor is preloaded into the flash as a bootstrap loader module.

- Using Multi-ICE

Multi-ICE requires almost no resources.

Rather than being an application on the board.

Programmers Reference (1)

■ Memory map after remap

Table 3-1 Memory map after remap

Address range	Size	Description
0x00000000 to 0x0003FFFF	256KB	32 bit SRAM bank, using ROMCON1
0x00040000 to 0x0007FFFF	256KB	32 bit SRAM bank, using ROMCON2
0x01800000 to 0x0187FFFF	512KB	16 bit flash bank, using ROMCON0
0x03FE0000 to 0x03FE1FFF	8KB	32 bit internal SRAM
0x03FF0000 to 0x03FFFFFF	64KB	Microcontroller register space

———— Note ————

The BSL does not enable the cache. When the caches are enabled, you cannot use the 32-bit internal SRAM.

Programmers Reference (2)

■ SRAM usage under the BSL

Table 3-2 shows the SRAM usage under BSL.

Table 3-2 SRAM usage under BSL

Address range	Description
0x00000000 to 0x0000003F	Exception vector table and address constants
0x00000040 to 0x00000FFF	Unused
0x00001000 to 0x00007FFF	Read-write data space for BSL
0x00008000 to 0x00077FFF	Available as download area for user code and data
0x00078000 to 0x0007FFFF	System and user stacks

Programmers Reference (3)

■ SRAM usage under Angel

Table 3-3 shows the SRAM usage under Angel.

Table 3-3 SRAM usage under Angel

Address range	Description
0x00000000 to 0x0000003F	Exception vector table and address constants
0x00000040 to 0x000000FF	Unused
0x00000100 to 0x00007FFF	Read-write data and privileged mode stacks
0x00008000 to 0x00073FFF	Available as download area for user code and data
0x00074000 to 0x0007FFFF	Angel code execution region

Programmers Reference (4)

■ Flash memory usage

Table 3-4 shows the flash memory usage.

Table 3-4 Flash memory usage

ADDRESS RANGE	DESCRIPTION
0x01800000 to 0x01806FFF	Bootstrap loader
0x01807000 to 0x01807FFF	Production test
0x01808000 to 0x0180FFFF	Reserved
0x01810000 to 0x0181FFFF	Angel
0x01820000 to 0x0187FFFF	Available for your programs and data

Bootstrap Loader Reference

- **The BSL has the following main functions:**
 - connecting to the host using a standard serial port and terminal application
 - providing facilities to configure the board
 - providing user help
 - managing the images in flash as a set of executable modules
 - allowing you to download applications to SRAM and execute them.

Basic setup with the BSL (1)

■ communicate with the BSL

Figure 4-1 shows the Evaluator-7T setup.

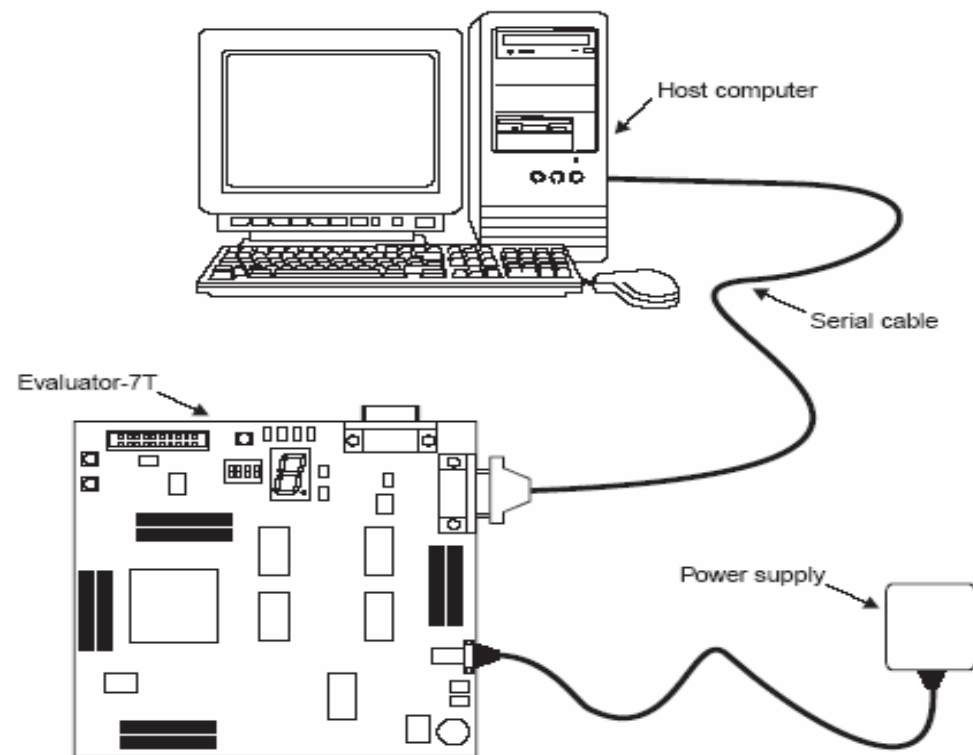


Figure 4-1 Bootstrap loader setup configuration

Basic setup with the BSL (2)

■ Communicating with a PC host

1. start the HyperTerminal program from Windows
Start->Programs->Accessories-> HyperTerminal.
2. Enter a name for this setup in the dialog box.
3. Select the COM port you have connected the
Evaluator-7T to from the Connect using menu and
click on OK.
4. In COMx Properties dialog,
 - select a baud rate at **9600**.
 - select **None** from the Flow Control menu.
5. Click on OK. HyperTerminal is now prepared for
output from the board.

Basic setup with the BSL (3)

■ Resetting the Evaluator-7T

1. Press the SYS RESET button (SW1) on the Evaluator-7T. A banner similar to the following is displayed in the HyperTerm window:

ARM Evaluator7T Boot Monitor Release 1.00

Press ENTER within 2 seconds to stop autoboot

2. Press Enter within 2 seconds to prevent the board from autobooting any other modules that may be stored in flash. The prompt

Boot:

is displayed and the LEDs D3 and D4 are lit.

Basic setup with the BSL (4)

■ Resetting the Evaluator-7T

3. Type boot at the Boot: prompt. The following response is displayed:

Boot: boot

Scanning ROM for modules ...

Found module 'BootStrapLoader' at 018057c8

Found module 'ProductionTest' at 018072c0

Found module 'Angel' at 0181a818

Boot:

BSL commands (1)

■ Basic commands

➤ help

Usage: help <command>

➤ printenv

Usage: printenv

➤ setenv

Usage: setenv <variable-name> <value>

Example: setenv baud 38400
 setenv noautobaud

➤ unsetenv

Usage: unsetenv <variable-name>

Example: unsetenv baud

BSL commands (2)

■ Downloading and executing commands

➤ download

Usage: download [<address>]

Example: download 00008000

➤ go

Usage: go [<program arguments>]

➤ gos

Usage: gos [<program arguments>]

➤ pc

Usage: pc <address>

Example: pc 00008000

BSL commands (3)

■ Flash operation commands

➤ **flashwrite**

Usage: flashwrite <address><source><length>

Example: flashwrite 01826000 01820000 2d1c

➤ **flashload**

Usage: flashload <address>

Example: flashload 01826000

➤ **flasherase**

Usage: flasherase <address length>

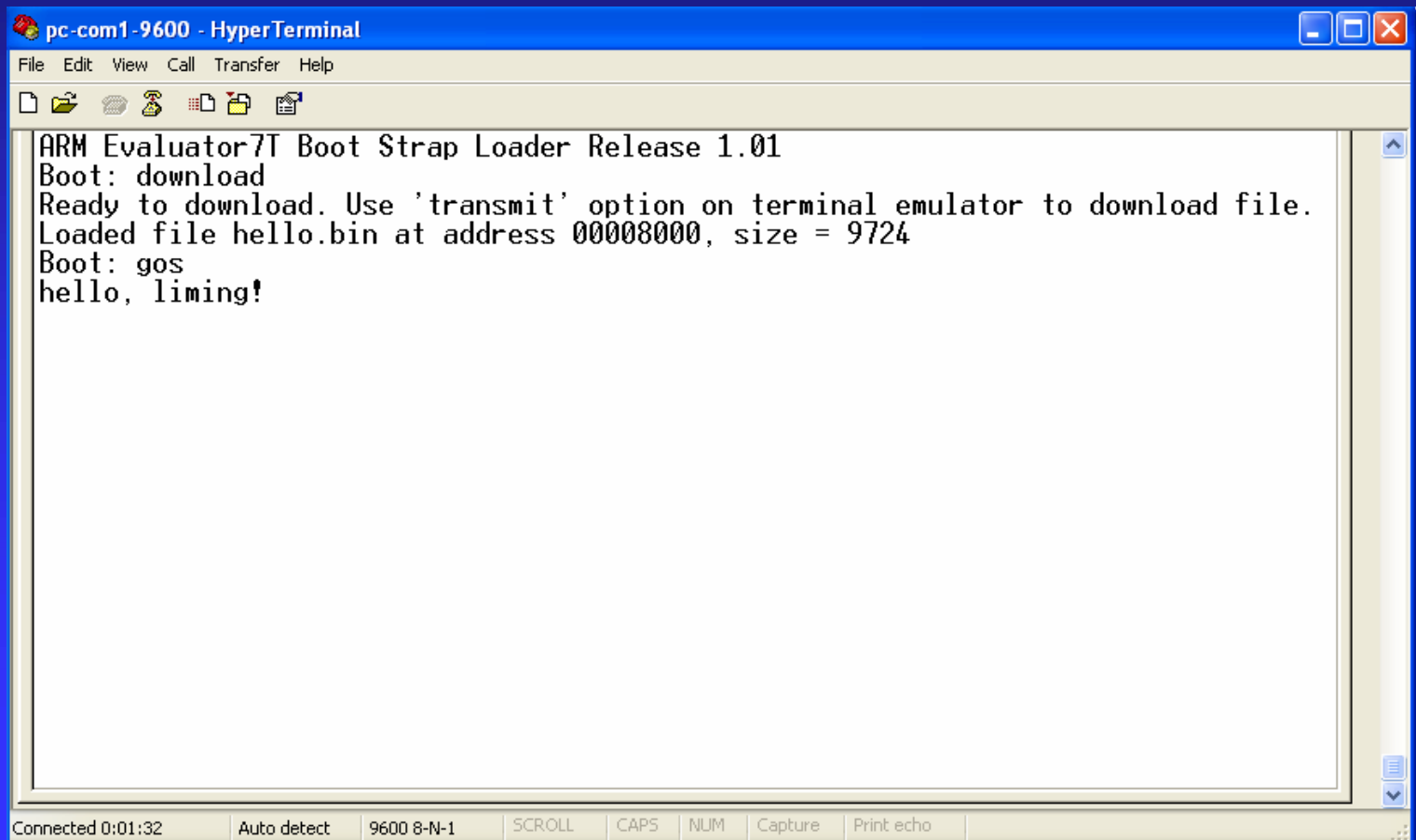
Example: flasherase 01826000 2d1c

Section 3

**Demo of my examples
in the manual**

Demo (1)

■ Page 5 – Using BSL to Download Image



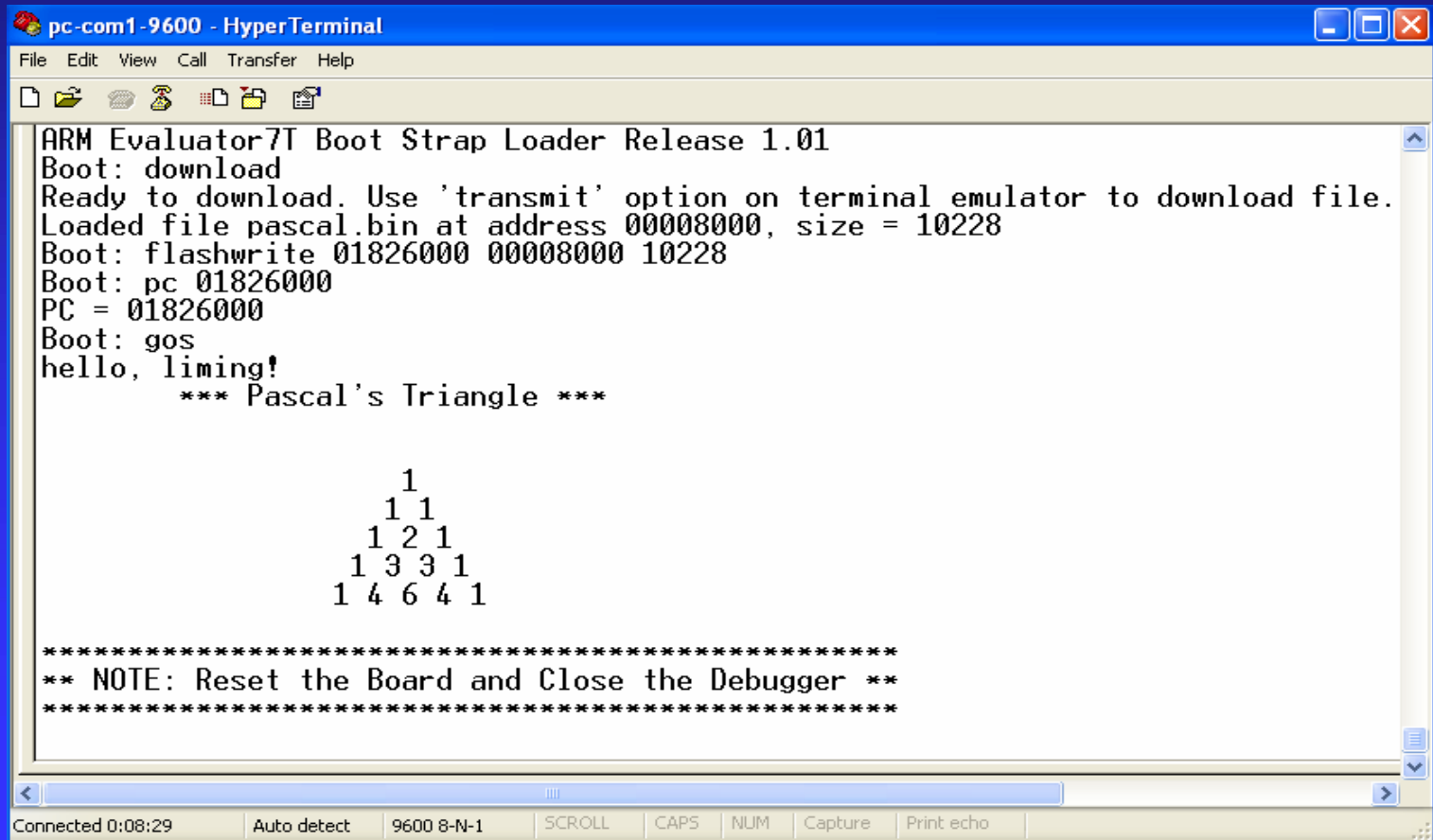
The screenshot shows a HyperTerminal window titled "pc-com1-9600 - HyperTerminal". The window has a menu bar with "File", "Edit", "View", "Call", "Transfer", and "Help". Below the menu bar is a toolbar with icons for file operations. The main text area displays the following output:

```
ARM Evaluator7T Boot Strap Loader Release 1.01
Boot: download
Ready to download. Use 'transmit' option on terminal emulator to download file.
Loaded file hello.bin at address 00008000, size = 9724
Boot: gos
hello, liming!
```

At the bottom of the window, a status bar shows the connection status: "Connected 0:01:32", "Auto detect", "9600 8-N-1", "SCROLL", "CAPS", "NUM", "Capture", and "Print echo".

Demo (2)

■ Page 9 – Using BSL to Write Flash



```
pc-com1-9600 - HyperTerminal
File Edit View Call Transfer Help

ARM Evaluator7T Boot Strap Loader Release 1.01
Boot: download
Ready to download. Use 'transmit' option on terminal emulator to download file.
Loaded file pascal.bin at address 00008000, size = 10228
Boot: flashwrite 01826000 00008000 10228
Boot: pc 01826000
PC = 01826000
Boot: gos
hello, liming!
*** Pascal's Triangle ***

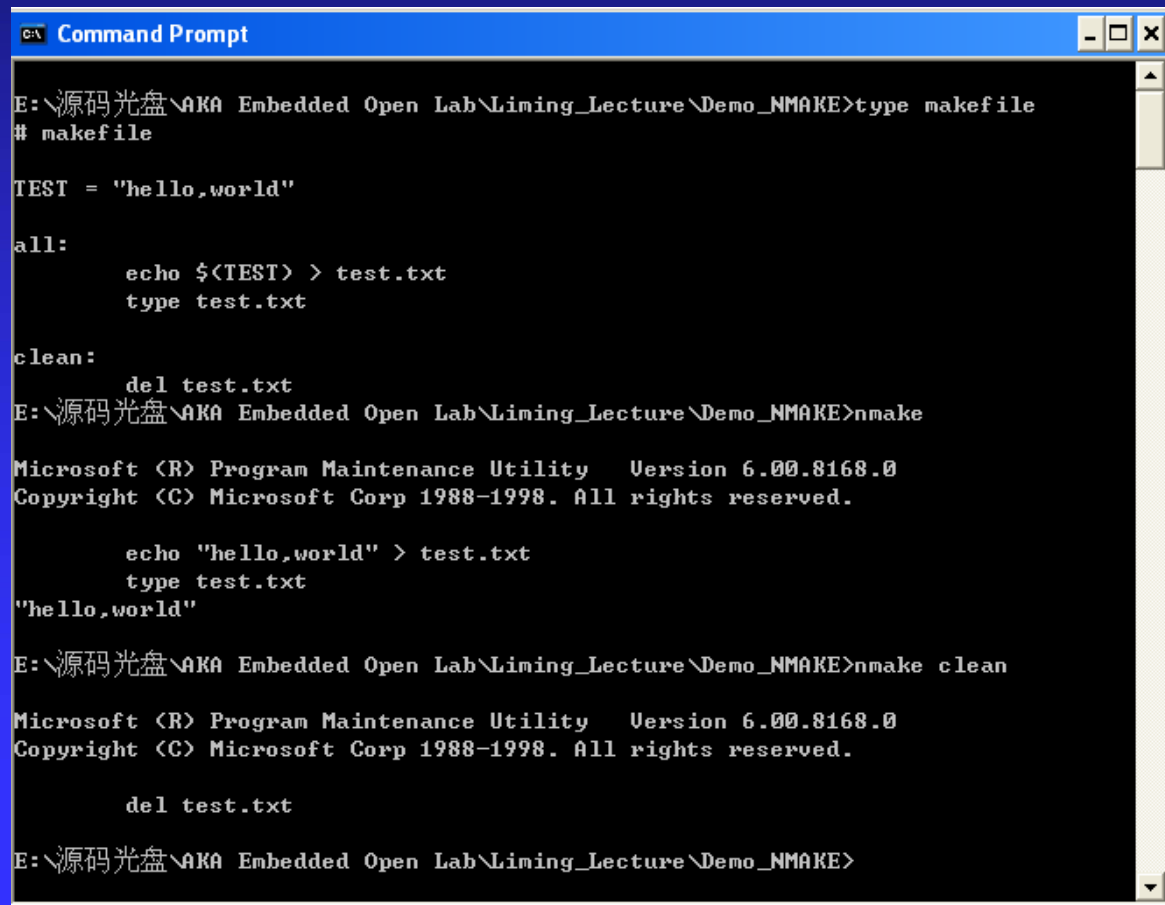
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1

*****
** NOTE: Reset the Board and Close the Debugger **
*****

Connected 0:08:29  Auto detect  9600 8-N-1  SCROLL  CAPS  NUM  Capture  Print echo
```

Demo (3)

■ Page 16 – Using Console to Build Project



```
C:\ Command Prompt

E:\源码光盘\AKA Embedded Open Lab\Lining_Lecture\Demo_NMAKE>type makefile
# makefile

TEST = "hello,world"

all:
    echo $(TEST) > test.txt
    type test.txt

clean:
    del test.txt
E:\源码光盘\AKA Embedded Open Lab\Lining_Lecture\Demo_NMAKE>nmake

Microsoft (R) Program Maintenance Utility   Version 6.00.8168.0
Copyright (C) Microsoft Corp 1988-1998. All rights reserved.

    echo "hello,world" > test.txt
    type test.txt
"hello,world"

E:\源码光盘\AKA Embedded Open Lab\Lining_Lecture\Demo_NMAKE>nmake clean

Microsoft (R) Program Maintenance Utility   Version 6.00.8168.0
Copyright (C) Microsoft Corp 1988-1998. All rights reserved.

    del test.txt

E:\源码光盘\AKA Embedded Open Lab\Lining_Lecture\Demo_NMAKE>
```

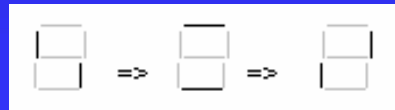
Demo (4)

- Page 30 – LED program (led.uue)
- Page 32 – 7segment program (7segment.uue)
- Page 33 – DIPs program (dips.uue)
- Page 34 – switch program (switch.uue)
- Page 38 – traffic program (traffic.uue)
- ...
- Page 41 – uC/OS-II program (ucos.uue)

Practices

■ Exercises you can do this week ...

- Displaying the Value of the DIP Switches to the Surface-mounted LEDs Continuously
- Displaying the Value of the DIP Switches to the HyperTerminal
- Counting User Interrupt Switches to 7-Segment Display
- Counting DIP Switch State Changes
- Some Light Flickering using timer



Section 4

Q & A

???

Thanks !