U-boot quick start guide

Getting started U-boot

Rev. 01 — 15 February 2007 User guide

Document information

Info	Content
Keywords	U-boot, LPC2294, LPC2292, LPC2290/01, LPC2220, LPC2214, LPC2212, LPC2210/01
Abstract	This document is a simple user guide for how to use u-boot on LPC2294 MCU: setup u-boot and toolkit; make and program the image and as well as the usage of u-boot

founded by Philips

Revision history

Rev	Date	Description
01	20070215	Initial version

Contact information

For additional information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. Introduction

1.1 About U-boot

The \"U-Boot\" Universal Bootloader project provides firmware with full source code under GPL. Many CPU architectures are supported: PowerPC (MPC5xx, MPC8xx, MPC82xx, MPC7xx, MPC74xx, 4xx), ARM (ARM7, ARM9, StrongARM, Xscale), MIPS (4Kc, 5Kc), x86, etc.

1.2 About Lpc2294

The 16/32-bit LPC2000 family is based on a 1.8V ARM7TDMI-S core operating at up to 60 MHz together with a wide range of peripherals including multiple serial interfaces, 10-bit ADC and external bus options.

LPC22xx series have configurable external memory interface with up to four banks, each up to 16 MB and 8/16/32-bit data width. With their 144 pin package, low power consumption, various 32-bit timers, 8-channel 10-bit ADC, PWM channels and up to 9 external interrupt pins, these microcontrollers are particularly suitable for industrial control, medical systems, access control and point-of-sale. Number of available GPIOs ranges from 76 (with external memory) through 112 pins (single-chip).

For more about LPC22xx, refer to the NXP website:
http://www.nxp.com/products/microcontrollers

1.3 U-boot For LPC2294

Since the u-boot has supported range of CPU architectures including ARM (ARM7, ARM9, StrongARM, Xscale) while LPC2294 is a base-on-supported-arm7 processor, it is possible, as well as useful, to provide the u-boot firmware as the LPC2294 bootloader.

2. Setup U-boot And Toolkits

2.1 Download U-boot-1.1.3

Before installation, you have to download the newest u-boot package. Below is the link where you can find the newest version:

http://www.nxp.com/external/sourceforge/projects/u-boot

Note that the newest version is U-Boot-1.1.3 and we are using U-Boot-1.1.3 as the current version.

2.2 Get and Install The Toolkit

It is strongly recommended that u-boot users install the Cross Development Tools http://www.nxp.com/external/ELDK, below is the link where you can find the manual of ELDK:

http://www.nxp.com/external/DULG/manual

But this time, we use ARM-ELF toolchain. Because there is something wrong compiling with arm-linux-gcc while arm-elf-gcc works perfectly. So it is not necessary to download and install the ELDK. Download the ARM-ELF toolchain 20040427 which can be found at:

http://www.nxp.com/external/opensrcsamsung/download.html

Install the toolchain. Be the root and execute the arm-elf-tools-20040427.sh like:

```
[root@localhost ]# /bin/sh ~/incoming/arm-elf-tools-20040427.sh
```

2.3 Unpack U-boot Package and add the Lpc2294 patch

Unpack the package, and then add the LPC2294 patch(be sure the patch file is in the src directory).

Proceed as follows:

```
[root@localhost src]# bunzip2 < u-boot-1.1.3.tar.bz2 | tar xf –
[root@localhost src]# cd u-boot-1.1.3
[root@localhost u-boot-1.1.3]# patch –p1 < ../u-boot-lpc2294.patch
```

Normally, you can see as follows:

```
patching file board/lpc2294/config.mk
patching file board/lpc2294/flash.c
patching file board/lpc2294/lowlevel_init.S
patching file board/lpc2294/lpc2294.c
patching file board/lpc2294/Makefile
patching file board/lpc2294/u-boot.lds
patching file cpu/arm7tdmi/config.mk
patching file cpu/arm7tdmi/cpu.c
patching file cpu/arm7tdmi/interrupts.c
patching file cpu/arm7tdmi/Makefile
patching file cpu/arm7tdmi/serial.c
patching file cpu/arm7tdmi/start.S
patching file drivers/rtl8019.c
patching file drivers/rtl8019.h
patching file include/asm-arm/arch-arm7tdmi/hardware.h
patching file include/configs/lpc2294.h
patching file Makefile
```

3. Make the image

3.1 Preparation

Before making, make sure the environment variable CROSS_COMPILE=arm-elf-

since the default CROSS_COMPILE=arm-linux-, you have to make a change. If you have added the LPC2294 patch, you need to do nothing because it has been changed in the Makefile. Or you can change the variable manually:

```
[root@localhost ]# export CROSS_COMPILE=arm-elf-
```

3.2 Make

enter the u-boot-1.1.3 directory

proceed as follows:

```
[root@localhost u-boot-1.1.3]#make distclean
……
[root@localhost u-boot-1.1.3]#make lpc2294_config
……
[root@localhost u-boot-1.1.3]#make
……
```

Normally, u-boot.bin and some other images will be made in the u-boot-1.1.3 directory.

4. Program the image

4.1 Program to the right address

It is based on the configuration of your LPC2294 board. While RESET is low, BOOT1:0 control booting and internal operation:

BOOT1:0=00 selects 8-bit memory on CS0 for boot

BOOT1:0=01 selects 16-bit memory on CS0 for boot

BOOT1:0=10 selects 32-bit memory on CS0 for boot

BOOT1:0=11 selects Internal Flash memory

This time, while resetting, BOOT1:0 selects 32-bit memory on CS0 for boot. So program the u-boot image to the external flash with your favorite flash writer.

4.2 Program using BDI 2000

Since BDI 2000 supports range of flash memories, users can program u-boot.bin to the external flash memory starting at 0x80000000 with BDI2000

4.3 Program using U-boot itself

If BOOT1:0 selects 32-bit memory on CS0 for boot. Users can program u-boot.bin to the external memory starting at 0x80000000 with u-boot itself. For details, see section How to use u-boot.

5. How to use U-boot

5.1 Start up U-boot

After programming u-boot.bin to LPC2294 board correctly, You are ready to run u-boot on your board.

Proceed as follows:

- Connect UART0 to PC com port
- connect a terminal to the board's serial console port with a baudrate of 9600 8n1n
- Power up

Normally, you can see from the serial terminal as follows:

```
U-Boot 1.1.3 (Sep  7 2005 - 18:28:34)

U-Boot code: 81400000 -> 81418CB8  BSS: -> 8141D26C

RAM Configuration:

Bank #0: 81000000  8 MB

Flash:  4 MB

In:    serial

Out:   serial

Err:   serial

Hit any key to stop autoboot:  0

=>
```

5.2 Initial steps

In the default configuration, U-Boot operates in an interactive mode which provides a simple command line, this means that U-Boot shows a prompt (default: `=>`) when it is ready to receive user input. You then type a command, and press enter. U-Boot will try to run the required action(s), and then prompt for another command.

To see a list of the available U-Boot commands, you can type `help` (or simply `?`). This will print a list of all commands that are available in your current configuration.

Proceed as follows:

```
=>  help

?       - alias for 'help'

autoscr - run script from memory

base    - print or set address offset

bdinfo  - print Board Info structure

boot    - boot default, i.e., run 'bootcmd'

bootd   - boot default, i.e., run 'bootcmd'

bootelf - Boot from an ELF image in memory

bootm   - boot application image from memory

bootp   - boot image via network using BootP/TFTP protocol
```

```
bootvx  - Boot vxWorks from an ELF image
cmp     - memory compare
coninfo - print console devices and information
cp      - memory copy
crc32   - checksum calculation
echo    - echo args to console
erase   - erase FLASH memory
flinfo  - print FLASH memory information
go      - start application at address 'addr'
help    - print online help
iminfo  - print header information for application image
imls    - list all images found in flash
itest   - return true/false on integer compare
loadb   - load binary file over serial line (kermit mode)
loads   - load S-Record file over serial line
loop    - infinite loop on address range
md      - memory display
mm      - memory modify (auto-incrementing)
mtest   - simple RAM test
mw      - memory write (fill)
nfs     - boot image via network using NFS protocol
nm      - memory modify (constant address)
ping    - send ICMP ECHO_REQUEST to network host
printenv- print environment variables
protect - enable or disable FLASH write protection
rarpboot- boot image via network using RARP/TFTP protocol
reset   - Perform RESET of the CPU
run     - run commands in an environment variable
saveenv - save environment variables to persistent storage
setenv  - set environment variables
sleep   - delay execution for some time
tftpboot- boot image via network using TFTP protocol
version - print monitor version
=>
```

With the command `help <command>` you can get additional information about most commands:

```
=>  help setenv printenv
setenv name value ...
    - set environment variable 'name' to 'value ...'
setenv name
    - delete environment variable 'name'
printenv
    - print values of all environment variables
printenv name ...
print value of environment variable 'name'
```

Note these:

- For most commands, you do not need to type in the full command name; instead it is sufficient to type a few characters. For instance, `help` can be abbreviated as `h`.
- The behaviour of some commands depends of the configuration of U-Boot and on the definition of some variables in your U-Boot environment.
- All U-Boot commands expect numbers to be entered in hexadecimal input format.
- Be careful not to use edit keys besides 'Backspace', as hidden characters in things like environment variables can be *very* difficult to find.

5.3 Set environment variables

First you can set and save some environment variables to config u-boot.

For instance:

```
=>  setenv ipaddr 192.168.0.134
=>  setenv ethaddr 00:50:c2:1e:af:fb
=>  setenv serverip 192.168.0.10
=>  saveenv
Saving Environment to Flash...
Un-Protected 1 sectors
Erasing Flash...
 done
Erased 1 sectors
Writing to Flash... done
Protected 1 sectors
=>
```

When resetting, u-boot will load environment variables from the saved flash block

5.4 Flash operation

Here we show the commands which operate the external flash memory.

The command `flinfo` (short: `fli`) can be used to get information about the available flash memory:

```
=>  flinfo
Bank # 1: SST SST39LF/VF160 (16 Mbit, uniform sector size)
  Size: 4 MB in 32 Sectors
  Sector Start Addresses:
80000000     80020000     80040000     80060000     80080000 E
800A0000 E   800C0000 E   800E0000 E   80100000 E   80120000 E
80140000 E   80160000 E   80180000 E   801A0000 E   801C0000 E
801E0000 E   80200000 E   80220000 E   80240000 E   80260000 E
80280000 E   802A0000 E   802C0000 E   802E0000 E   80300000
80320000 E   80340000 E   80360000 E   80380000 E   803A0000 E
803C0000 E   803E0000 E
```

Note that the “E” stands for “Erased”

The command **erase** is used to erase the contents of one or more sectors of the flash memory

```
=>  help erase
erase start end
    - erase FLASH from addr 'start' to addr 'end'
erase start +len
    - erase FLASH from addr 'start' to the end of sect w/addr 'start'+'len'-1
erase N:SF[-SL]
    - erase sectors SF-SL in FLASH bank # N
erase bank N
    - erase FLASH bank # N
erase all
erase all FLASH banks
```

the most frequent usage of this command is to pass the start and end addresses of the area to be erased,

for instance:

```
=>  erase 0x801e0000 0x8025ffff
 done
Erased 4 sectors
```

Note that:

- both the start and end addresses for this command must point exactly at the start resp. end addresses of flash sectors. Otherwise the command will not be executed.

The command **cp** is used for memory copy

The `cp` command "knows" about flash memory areas and will automatically invoke the necessary flash programming algorithm when the target area is in flash memory.

For instance:

```
=>  cp 0x81400000 0x801e0000 0x18cbc

Copy to Flash... done

=>
```

Note that:

- Before writing your data to one or more sectors of the flash memory, make sure that the very area has been erased.

The command **protect** is used to set certain parts of the flash memory to read-only mode or to make them writable again. Flash memory that is "protected" (= read-only) cannot be written (with the `cp` command) or erased (with the `erase` command). Protected areas are marked as `(RO)` (for "read-only") in the output of the `flinfo` command. For instance:

```
=>  help protect

protect on  start end

   - protect FLASH from addr 'start' to addr 'end'

protect on start +len

   - protect FLASH from addr 'start' to end of sect w/addr 'start'+'len'-1

protect on  N:SF[-SL]

   - protect sectors SF-SL in FLASH bank # N

protect on  bank N

   - protect FLASH bank # N

protect on  all

   - protect all FLA

protect off start end

   - make FLASH from addr 'start' to addr 'end' writable

protect off start +len

   - make FLASH from addr 'start' to end of sect w/addr 'start'+'len'-1 wrtable

protect off N:SF[-SL]

   - make sectors SF-SL writable in FLASH bank # N

protect off bank N

   - make FLASH bank # N writable

protect off all

   - make all FLASH banks writable

=>  protect on 0x80300000 0x8031ffff

Protected 1 sectors
```

```
=>  flinfo
Bank # 1: SST SST39LF/VF160 (16 Mbit, uniform sector size)
  Size: 4 MB in 32 Sectors
  Sector Start Addresses:
80000000      80020000      80040000      80060000      80080000 E
800A0000 E    800C0000 E    800E0000 E    80100000 E    80120000 E
80140000 E    80160000 E    80180000 E    801A0000 E    801C0000 E
801E0000      80200000      80220000      80240000      80260000 E
80280000 E    802A0000 E    802C0000 E    802E0000 E    80300000 RO
80320000 E    80340000 E    80360000 E    80380000 E    803A0000 E
803C0000 E    803E0000 E
=>
```

Note that:

- In most cases U-Boot provides just a simple software-protection, i. e. it prevents you from erasing or overwriting important stuff by accident (like the U-Boot code itself or U-Boot's environment variables), but it cannot prevent you from circumventing these restrictions - a nasty user who is loading and running his own flash driver code cannot and will not be stopped by this mechanism. Also, in most cases this protection is only effective while running U-Boot

5.5 Download image

Here we show how to download image via network using TFTP protocol

Before you type the tftp command, make sure that the tftp server has been started and the root directory which contains the image you want to download has been defined.

After setting up the tftp server, you are ready to download image.

First, use the ping command to check out the network:

```
=>  ping 192.168.0.1
host 192.168.0.1 is alive
=>
```

Then, download image using tftpboot command:

```
=>  tftp 0x81008000 linux.bin
TFTP from server 192.168.0.10; our IP address is 192.168.0.134
Filename 'linux.bin'.
Load address: 0x81008000
Loading:
########################################################################
########################################################################
########################################################################
```

```
#####################
done
Bytes transferred = 1104332 (10d9cc hex)
=>  tftp 0x81200000 romfs.img
TFTP from server 192.168.0.10; our IP address is 192.168.0.134
Filename 'romfs.img'.
Load address: 0x81200000
Loading:
#################################################################
#################################################################
#################################################################
##################################
done
Bytes transferred = 1167360 (11d000 hex)
```

After downloading the image ,you can jump to kernel using go command, or write it to the flash memory using cp command:

```
=>  go 0x81008000
Linux version 2.6.5-ucLPC (root@localhost.localdomain)
……………………………………………………………………..
```

6. Legal information

6.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

6.2 Disclaimers

General — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is for the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

6.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

7. Contents

Please be aware that important notices concerning this document and the product(s) described herein, have been included in the section 'Legal information'.

For more information, please visit: http://www.nxp.com
For sales office addresses, email to: salesaddresses@nxp.com

Date of release: 15 February 2007

Document identifier: uboot