LPC-E2468 Software Quickstart Guide

By default, the board executes the U-Boot bootloader located in the internal flash. It's menu is displayed on the virtual serial port, emulated by the USB device(you may need to download the USB to serial port drivers from http://www.ftdichip.com). The default settings of the bootloader are:

Com port: 115200 bps, 8N1
LAN settings:
IP address: 192.168.0.157
Netmask: 255.255.255.0
Server IP: 192.168.0.240
MAC: 00:de:ad:b0:05:02

If the boot process is not interrupted, it will load and execute a minimal uClinux system from the on-board NAND flash.

Programming u-boot to the board:

If the u-boot image gets corrupted for some reason, the user can easily restore it by using the free FlashMagic utility (available from http://www.flashmagictool.com). Select the correct processor type and frequency and under "Step 2 - Erase" select "Erase all Flash+Code read protect". Then load the "u-boot-bin.hex" file available in the "Images" section on the CD. Connect the board and close the "ISP_E" and "RST_E" jumpers (located at the lower left corner of the board). Then press the start button at the utility and, after the programming has completed, the u-boot image should be restored to the factory default.

Programming the uClinux and root images:

If you wish to transfer an image to the board and write it to the NAND flash, there are a few available methods - by using an USB flash drive or by using a network TFTP server or through the serial port.

If you wish to use an USB flash drive, copy the file to it, the plug it in the board and issue the following commands

lpc-e2468 # ***usb start***
...
lpc-e2468 # ***fatls usb 0***
...
<size of your file> <filename>
lpc-e2468 # ***fatload usb 0 <load_address> <filename>***
...

For example, if ***vmlinux.bin*** and ***romfs_5.img*** are put on a USB flash drive, then the following u-boot commands will restore the uClinux installation:

lpc-e2468 # **nand erase**
lpc-e2468 # **usb start**
lpc-e2468 # **fatls usb 0**
lpc-e2468 # **fatload usb 0 0xa0008000 vmlinux.bin**
lpc-e2468 # **nand write 0xa0008000 0x0 0x220000**
lpc-e2468 # **fatload usb 0 0xa0800000 romfs_5.img**
lpc-e2468 # **nand write 0xa0800000 0x400000 0x220000**

You can also use a TFTP server on the network to load the images. This saves time during

e.g. development, because the step of transfering the file to an intermediate medium is elliminated. You need to make sure that the "Server IP" option is set correctly for your network (the address can be changed with the "***setenv serverip 192.168.0.240***", "***saveenv***" commands). The command is:

lpc-e2468# ***tftpboot <load_address> <filename>***

Here is an example TFTP configuration:

lpc-e2468 # **setenv ipaddr 192.168.0.131**
lpc-e2468 # **setenv serverip 192.168.0.225**
lpc-e2468 # **setenv ethaddr 00:de:ad:b0:05:08**
lpc-e2468 # **nand erase**
lpc-e2468 # **tftpboot 0xa0008000 vmlinux.bin**
lpc-e2468 # **nand write 0xa0008000 0x0 0x220000**
lpc-e2468 # **tftpboot 0xa0800000 romfs_5.img**
lpc-e2468 # **nand write 0xa0800000 0x400000 0x220000**

If the above methods are unavailable to you for some reason, you can still load a file through the serial line. The syntax is:

lpc-e2468# ***loady <load_address> 115200***

Then send the file through your terminal program using the ymodem protocol.

An example of the above is restoring the default uClinux and ramdisk images to the board using TFTP. The commands are:

lpc-e2468# ***nand erase***
lpc-e2468# ***tftpboot 0xa0008000 vmlinux.bin_lpc2468_2.6.24.2_bigpatch_21***
lpc-e2468# ***nand write 0xa0008000 0x0 0x220000***
lpc-e2468# ***tftpboot 0xa0800000 romfs_5.img***
lpc-e2468# ***nand write 0xa0800000 0x400000 0x220000***

Compiling u-boot

The u-boot image has been built under a cygwin host using the GNUARM-gcc_3.4.3 available under the "Utils" folder. After you have your buld enviroenment set up correctly (basically installing cygwin and GNUARM under windows or installiing an arm gcc cross compiler under linux), extract the sources from the "U-boot" directory and issue:

$ ***make lpc_e2468_config***
$ ***make***

After a while you should have a new u-boot image "***u-boot-bin.hex***".

Compiling uClinux

You need to use a linux host in order to build the uClinux kernel due to some cross-compiling issuse. After you have installed the cross-compiler (for example by extracting the provided arm-linux-tools-20061213.tar.gz to /usr/local/arm and adding it to the PATH), extract the pre-patched sources from the 'uClinux-dist" directory and simply issue a "***make menuconfig***", select vendor NXP and product LPC2468, and then issue "***make***". The resulting images ***vmlinux.bin*** and ***romfs_5.img*** will be placed in the ***images*** subdirectory.

Common Questions

Q: What drivers are included in LPC-E2468 uBoot?
A: The following peripherals have drivers:

- Ethernet
- RS232 (used for console)
- SDRAM
- USB host
- SD/MMC (**see next question!**)

Q: I have problems using an SD/MMC card under U-Boot.
A: There are known problems with the U-Boot version used on our LPC boards. Here is what our customer Fred Rothganger emailed us:

In case you are curious, the problem has very little to do with the driver, and mostly to do with an improperly relocated interrupt vector block. This is a bug in the version of U-Boot that you distributed on CD, and is triggered any time an interrupt occurs in U-Boot. Since the SD/MMC driver is the first module that actually sets up an interrupt, this is where the crash occurs.

Even more specifically, the bug is in board/lpc_2478_stk/lowlevel_init.c. The first thing the lowlevel_init() function does is (try to) copy the interrupt vectors to the beginning of SRAM. It does this using a GCC inline assembly directive. The problem with the assembly code is that it executes conditioned on NE, rather than unconditionally. Changing it to the unconditional form fixes the problem with crashes during an interrupt. There are several other problems with this code, but I'm too tired now to enumerate them clearly. You should have someone check this code over carefully to ensure it makes sense.

Q: What drivers are included in LPC-E2468 uClinux?
A: The following peripherals have drivers:

- Ethernet
- RS232 (used for Linux console)
- SDRAM

The following peripherals **lack** support in uClinux:

- USB host
- SD/MMC

Here is a sample output from the LPC-E2468 uClinux/busybox shell:

```
# cat /proc/devices
Character devices:
  1 mem
  4 /dev/vc/0
  4 tty
  4 ttyS
  5 /dev/tty
  5 /dev/console
  7 vcs
 10 misc
 13 input
 90 mtd
```

```
Block devices:
  1 ramdisk
  7 loop
 31 mtdblock
```

Q. I have problems compiling. What cross compiler should I use?
A. Please check that you are using the following compiler:
http://ftp.snapgear.org/pub/snapgear/tools/arm-linux/arm-linux-tools-20061213.tar.gz
An early batch of CDs contained incorrect compiler. Also triple-check that the snapgear toolchain is the only cross compiler reachable from PATH.

Q: I can't build uClinux. I get strange syntax or link errors.
A: First ensure that you are using the correct cross compiler (the issue is discussed above). You also must have a host compiler, development libraries and tools installed.

Provide a link to the kernel sources:

$ rm -f linux-2.6.x
$ ln -s linux-2.6.24.2-lpc2468-patched linux-2.6.x

Clean your uClinux-dist directory:

$ make mrproper

Get rid of symbolic links left from previous builds:

$ rm -f tools/ucfront-gcc
$ rm -f tools/ucfront-g++
$ rm -f tools/cksum
$ rm -f tools/ucfront-ld
$ rm -f lib/libz.a
$ rm -f lib/uClibc

Ensure that environment variables are set correctly:

$ printenv CROSS_COMPILE
arm-linux-
$ printenv ARCH
arm

Finally configure for “Vendor=NXP, Product=LPC2468” and retry the build:

$ make menuconfig
$ make