Linux Quick Start Guide for the Olimex LPC-H3131

1. Introduction

This document aims to provide you with basic information about restoring, rebuilding and using Linux for the Olimex LPC-H3131 board.

Actually, the board is very similar to the Embedded Artists EA-OEM-3131 board which has a very interesting and thorough tutorial about how to build an embedded Linux system:

http://ics.nxp.com/support/documents/microcontrollers/pdf/embedded.artists.lpc3131.quickstart.lpc313x.bsp.linux.pdf

All the information in the document is copyright free and can be used as reference as is done in this document. Here you will often be redirected to follow certain sections of the document mentioned as only the differences have been described here.

It is worth mentioning some of the basic hardware differences between the two boards. They have been summarized in the following table:

	Olimex LPC-H3131	EA-OEM-3131
SPI flash memory size	2MB	4MB
SDRAM memory size	32MB	64MB
NAND memory size	512MB	256MB
Network support	no	yes
SD/MMC card detect	no	yes

In order to port Linux for the Olimex LPC-H3131 all the software sources available for the EA-OEM-3131 were used with patches created to support for edits in source code and configuration.

As you will see in the tutorial for the EA-OEM-3131 there are several approaches for building an embedded Linux system. Build process for the LPC-H3131 uses the following:

- Build process follows the LTIB approach;
- The board uses the U-boot bootloader;
- U-boot bootloader is stored in the SPI flash memory and “SPI NOR flash” should be the default jumper boot configuration;
- Lunux kernel image “uImage” and ramdisk image “rootfs.ext2.gz.uboot” are both stored in the NAND flash chip so U-boot boots Linux from there.

2. Restoring factory defaults

If you have accidently corrupted your system you can restore it to factory default state by following these steps:

2.1. Writing U-boot to the internal SPI flash memory

2.1.1. Configure jumpers for UART boot and follow section EA “5.1.1 UART boot” to get U-boot running.

Note: Boards shipped without Linux programmed have SPI flash chips with default page size of 528B instead of the 512B that U-boot can currently handle. In order to store U-boot executable in SPI flash you need to convert the page size to the correct one of 512B. To check your board’s SPI flash page size run “sf probe 0” command. If you see “2112 KiB” as a response read on, otherwise continue to 2.1.2.

This is a run-once procedure and can be accomplished by an attempt to erase a page in SPI flash chip as follows:

```
OLIMEX-LPCH3131 # sf probe 0
2112 KiB AT45DB161D at 0:0 is now current device
OLIMEX-LPCH3131 # sf erase 0 210
OLIMEX-LPCH3131 #
```

Power cycle the board to complete the page size modification procedure. Now repeat the EA “5.1.1 UART boot” section to continue.

2.1.2. Follow section EA “5.1.3.1 Programming SPI-NOR flash using u-boot” to store U-boot image.

```
OLIMEX-LPCH3131 # sf probe 0
2048 KiB AT45DB161D at 0:0 is now current device
OLIMEX-LPCH3131 # loady
##  Ready  for  binary  (ymodem)  download  to  0x30800000  at  115200
bps...
CCxyzModem - CRC mode, 0(SOH)/182(STX)/0(CAN) packets, 3 retries
## Total Size      = 0x0002d2c4 = 185028 Bytes
OLIMEX-LPCH3131 # sf erase 0 2f000
OLIMEX-LPCH3131 # sf write 30800000 0 2f000
OLIMEX-LPCH3131 #
```

2.1.3. Issue the following command to force default environment variables to load at next boot:

```
OLIMEX-LPCH3131 # sf erase 40000 200
```

2.1.4. Set jumpers to “SPI NOR flash” and reboot with the help of the button

2.1.5. Issue the following command to save environment variables:

```
OLIMEX-LPCH3131 # saveenv
Saving Environment to SPI Flash...
Erasing SPI flash...Writing to SPI flash...done
OLIMEX-LPCH3131 #
```

2.2. Restoring Linux kernel image and rottfs

2.2.1. Boot U-boot and interrupt the autoboot sequence

2.2.2. If things have gone really messy run the following command to recheck NAND for bad blocks and restore the bad block tables (it takes several minutes to complete):

```
OLIMEX-LPCH3131 # nand_format hard
It will erase whole nand device.Do you really want to format <y/n>
/
Done Checking.
Found total of 0 bad blocks!.
Erasing Nand...
Writing Flash Params to Nand... done
curr_pos:0, bad_blk_count:0, magic_pos:1
OLIMEX-LPCH3131 #
```

2.2.3. Erase the NAND flash:

```
OLIMEX-LPCH3131 # nand erase
For LPC313X based board, Do you want to erase block 0: <y/n> (type "n")
device 0 offset 0x20000, size 0x1ffe0000
Skipping bad block at  0x00000000

OK
OLIMEX-LPCH3131 #
```

2.2.4. Load the kernel image file "uImage" through the terminal (refer to EA " 5.2.4

2.2.5. UART using Y-modem protocol"):

```
OLIMEX-LPCH3131 # loady
## Ready for binary (ymodem) download to 0x30800000 at 115200
bps...
CCxyzModem - CRC mode, 0(SOH)/1194(STX)/0(CAN) packets, 4 retries
## Total Size      = 0x0012a1b4 = 1221044 Bytes
OLIMEX-LPCH3131 #
```

Note: *there is a quicker way to store kernel image and rootfs if using SD card. Please check out EA tutorial for details.*

2.2.6. Store to NAND:

```
LIMEX-LPCH3131 # nand write $(loadaddr) 0x80000 0x200000
NAND write: device 0 offset 0x80000, size 0x200000
 2097152 bytes written: OK
OLIMEX-LPCH3131 #
```

2.2.7. Load the kernel image file "rootfs.ext2.gz.uboot" through the terminal (yes, it is slow):

```
OLIMEX-LPCH3131 # loady
## Ready for binary (ymodem) download to 0x30800000 at 115200
bps...
CCCCCCCCxyzModem - CRC mode, 0(SOH)/4626(STX)/0(CAN) packets, 10
retries
## Total Size      = 0x00483c56 = 4734038 Bytes
OLIMEX-LPCH3131 #
```

2.2.8. Store to NAND:

```
LIMEX-LPCH3131 # nand write $(loadaddr) 0x2a0000 0x4a0000

NAND write: device 0 offset 0x2a0000, size 0x4a0000
 4849664 bytes written: OK
OLIMEX-LPCH3131 #
```

2.2.9. Reset the board and behold Linux running.

```
U-Boot 2009.11 (Dec 18 2011 - 20:29:17)

U-Boot code: 31600000 -> 3162F844  BSS: -> 31678820
RAM Configuration:
Bank #0: 30000000 32 MB
SF: Got idcode 1f 26 00 00 00
SF: AT45 status register: ad
SF: Detected AT45DB161D with page size 512, total 2097152 bytes
In:    serial
Out:   serial
Err:   serial
### main_loop entered: bootdelay=5

### main_loop: bootcmd="run nandram_boot"
Hit any key to stop autoboot:  0

NAND read: device 0 offset 0x80000, size 0x200000
 2097152 bytes read: OK

NAND read: device 0 offset 0x2a0000, size 0x4a0000
 4849664 bytes read: OK
*  kernel: cmdline image address = 0x30800000
## Booting kernel from Legacy Image at 30800000 ...
   Image Name:   Linux-2.6.33
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    1220980 Bytes =  1.2 MB
   Load Address: 30008000
   Entry Point:  30008000
   Verifying Checksum ... OK
   kernel data at 0x30800040, len = 0x0012a174 (1220980)
*  ramdisk: cmdline image address = 0x31000000
## Loading init Ramdisk from Legacy Image at 31000000 ...
   Image Name:   uboot ext2 ramdisk rootfs
   Image Type:   ARM Linux RAMDisk Image (gzip compressed)
   Data Size:    4734548 Bytes =  4.5 MB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
   ramdisk start = 0x31000040, ramdisk end = 0x31483e94
   Loading Kernel Image ... OK
OK
   kernel loaded at 0x30008000, end = 0x30132174
## Transferring control to Linux (at address 30008000) ...

Starting kernel ...
```

```
Uncompressing Linux... done, booting the kernel.
Linux version 2.6.33 (shondll@ubuntu) (gcc version 4.1.2) #9 Sun
Dec 25 16:47:18 EET 2011
CPU: ARM926EJ-S [41069265] revision 5 (ARMv5TEJ), cr=00053177
CPU: VIVT data cache, VIVT instruction cache
Machine: NXP EA313X
Memory policy: ECC disabled, Data cache writeback
Built 1 zonelists in Zone order, mobility grouping on.  Total
pages: 8128
Kernel command line: console=ttyS0,115200n8 root=/dev/ram0 rw
loglevel=7 ramdisk_size=15281
PID hash table entries: 128 (order: -3, 512 bytes)
Dentry cache hash table entries: 4096 (order: 2, 16384 bytes)
Inode-cache hash table entries: 2048 (order: 1, 8192 bytes)
Memory: 32MB = 32MB total
Memory: 25324KB available (2136K code, 167K data, 92K init, 0K
highmem)
Hierarchical RCU implementation.
NR_IRQS:37
irq=30 Event=0x67 bank:3 bit:7 type:3
irq=31 Event=0x7a bank:3 bit:26 type:2
irq=32 Event=0x77 bank:3 bit:23 type:2
irq=33 Event=0x7b bank:3 bit:27 type:0
irq=34 Event=0x18 bank:0 bit:24 type:1
irq=35 Event=0x50 bank:2 bit:16 type:0
irq=36 Event=0x55 bank:2 bit:21 type:0
Console: colour dummy device 80x30
console [ttyS0] enabled
Calibrating delay loop... 89.70 BogoMIPS (lpj=448512)
Mount-cache hash table entries: 512
CPU: Testing write buffer coherency: ok
Registering USB gadget 0x00001120 0x0ec00004 (3)
LPC31: Power Management init.
bio: create slab <bio-0> at 0
SCSI subsystem initialized
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
Advanced Linux Sound Architecture Driver Version 1.0.21.
Trying to unpack rootfs image as initramfs...
rootfs image is not initramfs (no cpio magic); looks like an initrd
Freeing initrd memory: 4620K
pca9532 0-0060: setting platform data
NetWinder Floating Point Emulator V0.97 (double precision)
JFFS2 version 2.2. (NAND) © 2001-2006 Red Hat, Inc.
ROMFS MTD (C) 2007 Red Hat, Inc.
msgmni has been set to 58
io scheduler noop registered (default)
Serial: 8250/16550 driver, 1 ports, IRQ sharing disabled
serial8250: ttyS0 at MMIO 0x15001000 (irq = 10) is a NXP16750
serial8250.0: ttyS0 at MMIO 0x15001000 (irq = 10) is a NXP16750
brd: module loaded
loop: module loaded
NAND device: Manufacturer ID: 0xec, Chip ID: 0xdc (Samsung NAND
512MiB 3,3V 8-bit)
Creating 1 MTD partitions on "lpc313x_nand":
0x0000014c0000-0x000020000000 : "lpc313x-rootfs"
```

```
mtd_dataflash spi0.0: at45db161d (2048 KBytes) pagesize 512 bytes
(OTP)
spi_lpc313x spi_lpc313x.0: chipselect 0 already in use
spi_lpc313x spi_lpc313x.0: LPC313x SPI driver
ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
mice: PS/2 mouse device common for all mice
i2c /dev entries driver
lpc313x-wdt lpc313x-wdt: Watchdog device driver initialized.
cpuidle: using governor ladder
lpc313x_mmc lpc313x_mmc.0: LPC313x MMC controller at irq 26
No device for DAI lpc313x-i2s
LPC313x ASOC main clock : 48000 (36864000)
asoc: UDA1380 <-> lpc313x-i2s mapping ok
ALSA device list:
  #0: LPC313X_I2S_UDA1380 (UDA1380)
RAMDISK: gzip image found at block 0
mmc0: new SD card at address 87c1
mmcblk0: mmc0:87c1 SD02G 1.87 GiB
 mmcblk0: p1
VFS: Mounted root (ext2 filesystem) on device 1:0.
Freeing init memory: 92K
init started: BusyBox v1.11.2 ()
starting pid 317, tty '': '/etc/rc.d/rcS'
Mounting /proc and /sys
Setting the hostname to nxp
Mounting filesystems
mount: mounting usbfs on /proc/bus/usb failed: No such file or
directory
Starting syslogd and klogd
Starting inetd:
/etc/rc.d/rc.local: line 50: sed: not found
starting pid 353, tty '': '-/bin/sh'
[root@nxp /]#
```

3. Rebuilding the Linux system from scratch

As mentioned earlier building the Linux system used the LTIB approach. The complete build process has been tested and works OK on clean installation of Ubuntu 10.04 LTS. You may want to use a dedicated virtual machine instance for this guide only.

3.1. Go to http://ltib.org/resources-download and get LTIB using one of the options described. As described in the EA tutorial it is actually not that easy to make the ltib script run. You may need a lot of packages to install, so be patient and resourceful.

3.2. Copy the supplied “config_olimex_LPC-H3131” to the directory where the ltib script is.

3.3. Run the following command to download the U-Boot sources:

```
$ ./ltib -p u-boot -m prep --preconfig config_olimex_LPC-H3131
```

A configuration window shall appear. Navigate to Exit and wait for completion. When asked if you want to save the configuration answer with YES.

3.4. Run the following command to download the Linux kernel:

```
$ ./ltib -p kernel -m prep
```

A configuration window shall appear. Navigate to Exit and wait for completion.

3.5. Apply the supplied patch to the original U-boot sources

3.5.1. Copy the supplied “u-boot-2009.11_olimex.patch” to rpm/BUILD/u-boot-2009.11/ relative to main ltib directory

3.5.2. Run the following command to patch the sources (within the rpm/BUILD/u-boot-2009.11/):

```
$ patch -p1 < u-boot-2009.11_olimex.patch
patching file board/ea31xx/config.mk
patching file board/ea31xx/ea31xx.c
patching file cpu/lpc313x/init.c
patching file cpu/lpc313x/nand_format.c
patching file cpu/lpc313x/nand_params.c
patching file cpu/lpc313x/start.S
patching file drivers/mtd/nand/nand_base.c
patching file drivers/mtd/spi/atmel.c
patching file include/config_cmd_default.h
patching file include/configs/ea31xx.h
```

3.6. Apply the supplied patch to the original linux-2.6.33 sources

3.6.1. Copy the supplied “linux-2.6.33_olimex.patch” to rpm/BUILD/linux-2.6.33/ relative to the main ltib directory

3.6.2. Run the following command to patch the sources (within the rpm/BUILD/linux-2.6.33/):

```
$ patch -p1 < linux-2.6.33_olimex.patch
patching file arch/arm/mach-lpc313x/ea313x.c
patching file arch/arm/mach-lpc313x/leds.c
patching file arch/arm/mach-lpc313x/leds-pca9532.c
```

3.7. Now prepare yourself a cup of coffee and run the main script “./ltib”, because if everything is OK you will have to wait a lot of time. After completion of the built process you will have the output images relative to the main ltib directory:

3.7.1. The U-boot initial image file (used only for the UART load) located here:

```
./rootfs/boot/init-u-boot.bin
```

3.7.2. The main U-boot image located here:

```
./rootfs/boot/u-boot.bin
```

3.7.3. The linux kernel image located here:

```
./rootfs/boot/uImage
```

3.7.4. The gzipped rootfs located here:

```
./rootfs.ext2.gz.uboot
```

4. FAQ

Q: *How to play an MP3 file under Linux?*

A: Prepare a FAT formatted SD/MMC card that contains at least one MP3 file. For the board SD card socket doesn't support hardware card detect mechanism the card needs to be inserted prior to booting Linux. If inserted at startup the card should be detected and its parameters displayed at the boot log console output. Here is an example of a card that has been detected:

```
mmc0: new SD card at address 87c1
mmcblk0: mmc0:87c1 SD02G 1.87 GiB
 mmcblk0: p1
```

After Linux has finished booting the SD/MMC card should be mounted to a directory in the file system. To do that, enter the following command:

```
[root@nxp /]# mount /dev/mmcblk0p1 /mnt/floppy/
```

"/mnt/floppy/" is a nice choice because obviously a floppy drive is not present to the system and the directory exists at boot time.

To list the contents of the newly mounted card enter:

```
[root@nxp /]# ls /mnt/floppy/
Metallica SM-No leaf clover.mp3
Sonique - Alive.mp3
[root@nxp /]#
```

Finally run the "mp3play" utility with the file of your liking (present on the card, of course):

```
[root@nxp /]# mp3play /mnt/floppy/Metallica\ SM-No\ leaf\
clover.mp3
/mnt/floppy/Metallica SM-No leaf clover.mp3: MPEG1-III (304352 ms)
Desired clock rate : 44100
Channels           : 2
Data format        : 12288
LPC313x ASOC main clock : 44100 (33868800)
Desired clock rate : 44100
Channels           : 2
Data format        : 12288
LPC313x ASOC main clock : 44100 (33868800)
Desired clock rate : 44100
Channels           : 2
Data format        : 12288
LPC313x ASOC main clock : 44100 (33868800)
```

Don't forget to plug the headphones in the "HEADPHONES" socket to actually hear something!

Q: Want debug print in U-boot?

A: To enable more verbose output on the console for U-boot uncomment the following symbol in the "config.mk" file in the main directory of U-boot and recompile:

```
DBGFLAGS= -g # -DDEBUG
```

to

```
DBGFLAGS= -g –DDEBUG
```

Q: Want to play around with your own application?

A: The LTIB package includes a sample "Hello world" application that can be edited and compiled. Just as you have done with the U-boot and kernel packages you can only download the "helloworld" package and edit it and then build it with the main ltib script:

```
$ ./ltib -p kernel -m prep
```

Edit the sources. Then compile with:

```
$ ./ltib
```

Now your rootfs will incorporate the new and modified "helloworld" application. To test it just run it from your board's embedded Linux console:

```
[root@nxp /]# hello
hello world
[root@nxp /]#
```

Note: A common startup problem is the size of the file system image. To work around you may have to edit the bootargs environment variable in U-boot to a correct size:

```
"ramdisk_size=15281"
```

to something like:

```
"ramdisk_size=15334"
```

for example.

____

Last edit: 28 December 2011 by shondll

THE END :)