

LPC32xx Linux FAQ

This FAQ provides common questions and answer on the LPC32xx Linux release.

Q:

I have the RTC driver installed, but can't get it working. The hwclock command can't find the device.

A:

Verify that the device nodes (/dev/rtcX) for the RTC driver has the same major and minor number as the RTC driver. The RTC driver uses major number 254. If the major numbers do not match, delete the device node for the RTC driver and remake them with major number 254 using the mknod command. The sequence below can be used on the target to perform the changes.

If the proc filesystem is enabled, you can use the cat command to see the RTC driver major number.

```
[root@nxp /]# cat /proc/devices
```

Character devices:

```
 1 mem
 2 pty
 3 ttyp
 4 /dev/vc/0
 4 tty
 4 ttyS
 5 /dev/tty
 5 /dev/console
 5 /dev/ptmx
 7 vcs
10 misc
13 input
14 sound
29 fb
89 i2c
116 alsa
128 ptm
136 pts
180 usb
189 usb_device
253 usb_endpoint
254 rtc
```

Block devices:

```
 7 loop
 8 sd
65 sd
66 sd
67 sd
68 sd
69 sd
70 sd
71 sd
128 sd
```

```

129 sd
130 sd
131 sd
132 sd
133 sd
134 sd
135 sd
179 mmc
[root@nxp /]# ls -la /dev/rtc*
crw-r----- 1 root root 10, 135 Jan 8 2009 /dev/rtc
[root@nxp /]# rm /dev/rtc
[root@nxp /]# mknod /dev/rtc c 254 0
[root@nxp /]# ls -la /dev/rtc*
crw-r--r-- 1 root root 254, 0 Jan 8 2009 /dev/rtc
[root@nxp /]#

```

Q:

The LCD backlight doesn't work on one or more of my boards.

A:

There are several different versions of the PHY3250 board. Depending on which version you have, the LCD backlight logic may be active high or active low to enable. The board specific LCD driver functions will automatically generate the correct code to generate the backlight enable state, but only if the board version numbers are correctly selected in the Linux kernel configuration. To verify that the board versions match the kernel configuration, look for the version numbers of the CPU module, carrier board, and LCD board. These will appear as numbers on the board of 1304.x, 1305.y, and 1307.z, respectively. Run the Linux kernel configuration and go to the System Type-→ menu selection. Locate the board version numbers and make sure that they match your board hardware versions.

Q:

I have I2C enabled, but a specific I2C channel isn't working. Or a device that requires I2C isn't working.

A:

Although I2C can be enabled in the driver block, individual I2C channels need to be enabled in the chip specific setup. There are 3 I2C channels on the LPC32xx: 2 standard I2C channels and 1 channel for USB support. Select which channels you want enabled in the System Type -→ area of the Linux kernel configuration menu.

As of this port, the USB OTG I2C channel is needed for USB. I2C0 is needed for the RTC clock and the UDA1380 CODEC.

Q:

USB host isn't working.

A:

Make sure the USB OTG I2C channel is enabled in the System Type-→ area of the Linux kernel configuration menu. Also make sure the USB OHCI host component has been added to the kernel configuration.

Q:

How do I mount and access an SD/MMC card?

A:

Make sure the kernel has support the SD/MMC with the ARM PL180 AMBA SD driver. Install the SD/MMC card into the SD/MMC slot. *VFAT support may need to be added to the kernel.* Use the following sequence to mount and access the card – this sequence assumes the /mnt/src directory exists.

```
[root@nxp /root]# mount /dev/mmcblk0p1 /mnt/src/
[root@nxp /root]# ls /mnt/src/
(kenny~1.mp3  cinema~3.mp3  dixiec~2.mp3  garthb~2.mp3  pictur~1.sre
0000059m.jpg  colinr~1.mp3  dixiec~3.mp3  garthb~3.mp3  pictur~2.sre
00010968.bmp  collin~1.mp3  dixiec~4.mp3  garthb~4.mp3  presen~1
00015603.jpg  config~1      docume~1      garyal~1.mp3  rtc-lp~1.c
0005257m.jpg  cyndit~1.mp3  eboot.nb0    landsl~1.mp3  semwin.sre
16112.bmp     darryl~1.mp3  eeprom.bin   lcdbar~1.sre  semwin2.sre
16260.jpg     demo.mp3      elaine~1.mp3  lesmis~1.mp3  semwin3.sre
38spec~1.mp3  di1606~1.mp3  emerso~1.mp3  lpc32x~1     serenade.mp3
38spec~2.mp3  di26f0~1.mp3  emerso~2.mp3  lpc32x~1.c   software
38spec~3.mp3  di4ce1~1.mp3  faithh~1.mp3  lpc32x~1.tgz  stagel.bin
a1.sre        di50cd~1.mp3  fleetw~1.mp3  misc         tools
as_tim~1.mp3  di65d3~1.mp3  fly.mp3      mstim~1.sre  u-boot.bin
bootex.log    di7b9f~1.mp3  foofig~1.mp3  nkcdc.bin    u-boot~1.bin
brooks~1.mp3  di8442~1.mp3  found.000    nknew.bin    uimage
buenav~1.mp3  dic9b1~1.mp3  franks~1.mp3  nkold.bin    videot~1
ceconfig.h    diecd5~1.mp3  galgcd~1.mp3  nktst.bin
cinema~1.mp3  difa9b~1.mp3  gaed73~1.mp3  p1.sre
cinema~2.mp3  dixiec~1.mp3  garthb~1.mp3  phy325~1.pat
[root@nxp /root]#
```

Q:

How do I use the high speed serial ports?

A:

Enable the LPC32xx high speed serial ports in the kernel driver configuration and select which high speed serial ports of the 3 you want to use in the Linux kernel configuration System Type-→ menu. Add the device nodes to the /dev are using the following command:

```
[root@nxp /root]# mknod /dev/ttyTX0 c 204 196
[root@nxp /root]# mknod /dev/ttyTX1 c 204 197
[root@nxp /root]# mknod /dev/ttyTX2 c 204 198
```

Q:

How do I partition an SD card for EXT2 support?

A:

Caution: If you partition an SD card for EXT2 support, you will most likely erase the original contents on the card. BE VERY CAREFUL WITH THIS SEQUENCE – IF YOU SELECT THE WRONG DEVICE, YOU COULD ERASE YOUR HARD DRIVE! Using a USB SDMMC dongle or other USB memory card device, insert the SD card and plug the SD card into the host machine. Umount the card id it was mounted. Determine the location the card was installed (ie, /dev/sdd). Using the fdisk command, erase any

partitions on the device and replace it with a primary Linux partition. Save the partition data and exit fdisk.

Using the mkfs command, make the EXT2 file on the first partition of the device. (*mkfs -t ext2 /dev/mmcbk0p1*). Try to mount the device and see if it works. Example output of the sequence is shown below.

```
[root@fed_kv proc]# /sbin/fdisk /dev/sdd
```

*The number of cylinders for this disk is set to 167680.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
(e.g., DOS FDISK, OS/2 FDISK)*

Command (m for help): p

```
Disk /dev/sdd: 1030 MB, 1030225920 bytes  
4 heads, 3 sectors/track, 167680 cylinders  
Units = cylinders of 12 * 512 = 6144 bytes  
Disk identifier: 0x00000000
```

<i>Device</i>	<i>Boot</i>	<i>Start</i>	<i>End</i>	<i>Blocks</i>	<i>Id</i>	<i>System</i>
<i>/dev/sdd1</i>	<i>*</i>	<i>1</i>	<i>167680</i>	<i>1006078+</i>	<i>83</i>	<i>Linux</i>

Command (m for help): d
Selected partition 1

Command (m for help): n
Command action
e extended
p primary partition (1-4)

p
Partition number (1-4): 1
First cylinder (1-167680, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-167680, default 167680):
Using default value 167680

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): 83

Command (m for help): p

```
Disk /dev/sdd: 1030 MB, 1030225920 bytes
```

4 heads, 3 sectors/track, 167680 cylinders
Units = cylinders of 12 * 512 = 6144 bytes
Disk identifier: 0x00000000

Device	Boot	Start	End	Blocks	Id	System
/dev/sdd1		1	167680	1006078+	83	Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
The kernel still uses the old table.
The new table will be used at the next reboot.
Syncing disks.
[root@fed_kv proc]#

Now that the partition has been setup, the partition needs to be formatted with the EXT2 filesystem. Using the mkfs command to do that as follows:

```
[root@fed_kv proc]# /sbin/mkfs -t ext2 /dev/sdd1
mke2fs 1.41.3 (12-Oct-2008)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
62976 inodes, 251519 blocks
12575 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=260046848
8 block groups
32768 blocks per group, 32768 fragments per group
7872 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376
```

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 26 mounts or 180 days, whichever comes first. Use tune2fs -c or -i to override.
[root@fed_kv proc]#

Q:
How do I boot from an SD card?

A:

Prior to being able to boot from the card, the card needs to be formatted in the EXT2 format. *See the other answer in this FAQ for information on how to setup for SD card for EXT2 format.* Using a USB SDMMC dongle or other USB memory card device, insert the SD card and plug the SD card into the host machine. Mount the SD card someplace on your host machine. Copy the contents of the root filesystem to the root directory of the SD card. Umount the card and insert it into the target board. Power up the target board and get to the u-boot prompt. Change the bootargs environment variable to boot from the SD card instead of its current boot source. The changed area of the bootargs variable would appear as “root=/dev/mmcblk0p1”.

For example, to change NFS boot to SD card boot, change the following line:

```
bootargs='console=ttyS0,115200n8 root=/dev/nfs rw  
nfsroot=192.168.1.100:/nfsroot ip=dhcp'
```

to the new line:

```
bootargs='console=ttyS0,115200n8 root=/dev/mmcblk0p1 rw ip=dhcp'
```

Q:

How is MTD NAND partitioned on the Phytex board?

A:

MTD NAND is partitioned using the MTD partition table in the board specific file in the /arch/arm/mach-lpc32xx area. For the Phytex 3250 board, this file is called board-phy3250.c. Partitions can be changed by editing the MTD partition structure (struct mtd_partition) in that file.

For the Phytex 3250 board, the partitions are setup as follows:

Blocks 0 – 90	Stage 1 loader and u-boot
Blocks 90 – 100	u-boot saved environment variables
Blocks 100 – 256	Linux kernel image
Blocks 356 on	Linux root filesystem

The partitioning can be examined at runtime with the cat /proc/mtd command.

Q:

How do I boot from an MTD NAND FLASH?

A:

Booting from kernel from MTD FLASH requires simply saving the kernel image in the MTD kernel partition and setting u-boot to boot from that partition. Based on the Phytex 3250 board partitioning scheme, the Linux kernel image partition is located at blocks 100 to 256 (offset 0x190000 with size 0x270000). The kernel image needs to be downloaded to the board using tftp and then burned into FLASH at the location using the following sequence:

```
uboot> setenv bootfile uImage  
uboot> setenv fileaddr 80100000  
uboot> setenv serverip 192.168.1.100 (Used your host machine address here!!!)  
uboot> dhcp  
HW MAC address: 00:50:C2:8C:ED:B9
```

```
ENET:auto-negotiation complete
ENET:Link status up
ENET:FULL DUPLEX
ENET:100MBase
BOOTP broadcast 1
DHCP client bound to address 192.168.1.62
TFTP from server 192.168.1.100; our IP address is 192.168.1.62
Filename 'uImage'.
Load address: 0x80100000
Loading:
#####
#####
done
Bytes transferred = 1638108 (18fedc hex)
```

Notice the bytes transferred (1638108) and saved memory address (0x80100000). To save the kernel image in FLASH, the kernel image size needs to be rounded up to the next NAND block size. Each NAND block is 0x4000 bytes, so based on the kernel image size of 1638108 bytes, the kernel currently uses $1638108/16384$ blocks = 99.98 blocks. We need to round this up to 100 and recomputed the kernel image size to save into FLASH. The actual size saved into FLASH will be the new rounded block count times the block size = $100 * 0x400 = 1638400$ bytes = 0x190000 bytes. The MTD NAND commands are then used to unlock, erase, write the number of kernel image bytes, and then relock the NAND data. The sequence is shown below:

```
uboot> nand device 0
Device 0: NAND 64MiB 1,8V 8-bit... is now current device
uboot> nand unlock 190000 270000
device 0 offset 0x190000, size 0x270000
nand_unlock: start: 00190000, length: 2555904!
NAND flash successfully unlocked
uboot> nand erase 190000 270000
```

```
NAND erase: device 0 offset 0x190000, size 0x270000
Erasing at 0x3fc000 -- 100% complete.
OK
uboot> nand write 0x80100000 0x190000 0x190000
```

```
NAND write: device 0 offset 0x190000, size 0x190000
1638400 bytes written: OK
uboot> nand lock 190000 270000
NAND flash successfully locked
```

The last thing to do is tell u-boot to boot using the image from NAND by changing the bootcmd environment variable as shown below:

```
uboot> setenv bootcmd 'nboot 80100000 0 190000; bootm'
uboot> saveenv
```

*Saving Environment to NAND...
Erasing Nand...Writing to Nand... done
uboot>*

Q:
What is LTIB?

A:
LTIB is the Linux Target Image Builder. It not only handles building the bootloader and kernel image, but also the root filesystem and deployment options. LTIB also handles getting and installing the correct packages for the target, toolchain installation, and other tasks that could be complex for the novice Linux developer.

Q:
What is the GPP?

A:
The GPP (Global Package Pool) is the public storage location for all the sources/patches referenced by LTIB. It is located at www.bitshrine.org The 32XX kernel and u-boot patches are here, as well as a prebuilt LPC32xx VFP toolchain.

Q:
Are any floating point benchmarks between soft-float and the VFP in the LPC32XX available? How do I build code to support the VFP?

A:
Using a simple benchmark that recursively computes a floating point number over a period of time, the VFP performs about 5x to 6x faster in single precision operations than just software emulated floating point. Using GCC 4 and a toolchain that has been built that supports VFP, you can use the “`imfloat-abi=softfp -mfpv=vfp`” options to enable VFP support in your application. *Note that this benchmark is ‘loose’ and can vary based on selected kernel and system options, network load, and all types of system factors. EABI versions of the compiler will have differing gains.*

On Linux 2.6.27.8 with GCC 4.3.2 and GLIBC 2.7 on a lightly loaded system, the test application (using single precision floats) logged the following times:

Software emulation only	: 31.44s
VFP supported float	: 5.61s

That’s about a performance gain of 5.6x for single precision float in typical Linux applications.

On the same system using double precision float, the times are as follows:

Software emulation only	: 93.17s
VFP supported float	: 8.17s

That’s about a performance gain of 11.4x for double precision float in typical Linux applications.

Q:

Why am I getting the warning “Warning - bad CRC or NAND: in u-boot?”

A:

This warning occurs when the u-boot parameters stored in NAND FLASH are not initialized or have been erased. u-boot will update FLASH with default values and the message shouldn't appear the next time u-boot is started. If it is consistently appearing, something is erasing or altering NAND FLASH where the u-boot stores it's parameters\

.