AN10389

Getting started uClinux with LPC22xx

Rev. 01 — 15 February 2007 Application note

Document information

Info	Content
Keywords	uClinux, ARM LPC22xx
Abstract	This application note describes how to use uClinux on NXP's LPC22xx series ARM MCU: setup Linux environment, system configuration, build and load image files as well as simple introduction on uClinux development.

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

© NXP B.V. 2007. All rights reserved.

1. Introduction

1.1 About uClinux

uClinux – Micro-C-Linux -- is a Linux derivative intended for microcontrollers without Memory Management Unit (MMU). It's free and open source software under GNU Public License.

uClinux as an operation system includes Linux kernel releases for 2.0, 2.4 and 2.6 as well as a collection of user applications, libraries and tool chains. The uClinux is much, much smaller than Linux kernel while retaining the main advantages of Linux OS: stability, superior network capability and excellent file system support.

For more about uClinux, refer to its official website via http://www.nxp.com/external/uclinux.

1.2 About LPC22xx

The 16-bit/32-bit LPC2000 family is based on a 1.8 V 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 the LPC22xx, refer to the microcontrollers section of the NXP website http://www.nxp.com/products/microcontrollers.

1.3 uClinux for LPC22xx

Since its release in 1998, uClinux has supported wide range of CPU architectures as M68K, ARM, PowerPC, V850, etc.

LPC22xx is based on supported ARM7 processor. While with up to 64 MB external memory available, LPC22xx can fit for the relative big memory requirement of uClinux (2 MB at least).

Note that LPC21xx series can't run uClinux since its limited internal memory size.

2. Setup environment

2.1 Install Linux on the PC

To begin uClinux based development, a PC Linux environment is necessary.

To setup Linux environment, there are 3 ways:

1. Setup Linux on PC

Buying distribution CD-R or downloading Linux images from website, e.g., Red Hat Linux 9.0 on http://www.nxp.com/external/redhat, and then install Linux on your PC according to its installation guide step by step. By configuring, you can get a dual-boot system: Windows and Linux.

Please be sure there is enough memory space on your PC beforehand.

© NXP B.V. 2007. All rights reserved.

This way is suitable for single users.

2. Telnet to Linux server through intranet

To setup a Linux server is a good way for group users.

The telnet utility “Cuteftp, secureCRT, etc.” can be obtained from internet free.

3. Using Cygwin

Cygwin is a Linux-like environment for Windows, which can be downloaded from Cygwin’s website located at http://www.nxp.com/external/cygwin.

It can be used in small application development or quick verification. Usually Cygwin is not recommended for formal uClinux/Linux development.

2.2 Grab source code packages

To begin your first step on uClinux development, below is the list that you have to grab on your Linux PC.

1) uClinux distribution

The easiest way to get started with uClinux is to download a copy of uClinux-dist from the uClinux.org site via http://www.nxp.com/external/uclinuxdist.

2) uClinux-2.6 Kernel

You could find the latest uClinux/ARM kernel patch at:

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

At the download section, you'll find the link of the original kernel package and the latest linux-2.6.x patch under the kernel directory of the Public Linux Archive:

http://www.nxp.com/external/kernel/pub

3) NXP LPC22xx patch

From NXP’s support engineers or download from maillist of http://www.nxp.com/external/uclinux, you could get the uClinux patch and kernel patch against linux-2.6.x plus the above patch for LPC22xx.

http://www.nxp.com/external/mailman/2005-June

Name: uClinux-Philips-LPC22xx.tar.gz

For updating info, please pay attention to the maillist of uClinux official website.

2.3 Make source code tree

After gathering all files for first compilation (on this guide, the downloaded files are gathered on ~/incoming), we should make up them together.

1) Untar the uClinux distribution

On a directory which has enough free available space (> 2 GB?), untar the uClinux distribution:

```
[root@mylinux /]# tar -zxvf ~/incoming/uClinux-dist-20040408.tar.gz
```

Let's look around what we have on the uClinux-dist directory:

```
[root@mylinux /]# cd uClinux-dist/
[root@mylinux uClinux-dist]# ls -al
total 104
drwxr-xr-x 15 1000 users 4096 Apr 8 09:27 .
```


© NXP B.V. 2007. All rights reserved.

```
drwxr-xr-x 28 root root 4096 Apr 27 20:32 ..
-rw-r--r-- 1 1000 users 18007 Apr 8 09:13 COPYING
drwxr-xr-x 3 1000 users 4096 Apr 8 09:13 Documentation
-rw-r--r-- 1 1000 users 9305 Apr 8 09:13 Makefile
-rw-r--r-- 1 1000 users 4934 Apr 8 09:13 README
-rw-r--r-- 1 1000 users 1743 Apr 8 09:13 SOURCE
drwxr-xr-x 2 1000 users 4096 Apr 15 15:19 bin
drwxr-xr-x 3 1000 users 4096 Apr 8 09:27 config <-- the configuration files for
userland and etc.
drwxr-xr-x 11 1000 users 4096 Apr 8 09:27 freeswan <-- IPsec implementation
drwxr-xr-x 68 1000 users 4096 Apr 8 09:23 glibc <-- Yes, the GNU C library.
drwxr-xr-x 18 1000 users 4096 Apr 8 09:28 lib <-- many libraries ported to uClinux
include uC-libc.
drwxr-xr-x 15 1000 users 4096 Apr 8 09:27 linux-2.0.x <-- uClinux 2.0.x kernel
drwxr-xr-x 16 1000 users 4096 Apr 8 09:27 linux-2.4.x <-- uClinux 2.4.x kernel
drwxr-xr-x 18 1000 users 4096 Apr 8 09:27 linux-2.6.x <-- uClinux 2.6.x kernel
drwxr-xr-x 3 1000 users 4096 Apr 8 09:26 tools <-- utilities for romfs install and
etc.
drwxr-xr-x 17 1000 users 4096 Apr 8 09:27 uClibc <-- the uClibc, from uclibc.org.
Differ to uC-libc.
drwxr-xr-x 174 1000 users 4096 Apr 8 09:27 user <-- the “userland”. The
applications that is ported.
drwxr-xr-x 44 1000 users 4096 Apr 8 09:27 vendors <-- the configuration files for
each vendor/models.
```

2) Make a new linux-2.6.x kernel from the scratch

The kernel version of linux-2.6.x directory in the 20040408 distribution is “linux-2.6.2-uc0”. We need another kernel with newer kernel version for uClinux/ARM 2.6, here.

```
[root@mylinux uClinux-dist]# tar -jxvf ~/incoming/linux-2.6.11.8.tar.bz2
```

Now we have got the whole linux-2.6.11.8 codes, and need to patch the kernel with the Samsung patch.

```
[root@mylinux uClinux-dist]# gzip -dc ~/incoming/linux-2.6.11.8-hsc0.patch.gz |
patch -p0
```

3) Add NXP LPC22xx patch

You should have had two compressed files for LPC22xx.

3.1) LPC22xx patch against Linux kernel

The patch is against linux-2.6.11.8 kernel plus the Samsung patch.

```
1    [root@mylinux uClinux-dist]# gzip -dc ~/incoming/linux-2.6.11.8-
     lpc22xx.patch.gz | patch -p0
```

You should see the codes that are patched in some directories that contains “lpc22xx” string. You can use the directory name “linux-2.6.11.8” for your uClinux 2.6 kernel directory without further operation. However, I recommend to use the kernel directory name to “linux-2.6.x” because it is more convenient.

So you don't need the linux-2.6.x directory which is included in the uClinux-dist.

```
2    [root@mylinux uClinux-dist]# rm -rf linux-2.6.x/
```

And we rename the newer patched kernel directory to “linux-2.6.x”.

```
3    [root@mylinux uClinux-dist]# mv linux-2.6.11.8 linux-2.6.x
```

3.2) LPC22xx patch against uClinux distribution

Patch the file uClinux-dist-lpc22xx.patch.gz:

```
[root@mylinux uClinux-dist]# cd..

4    [root@mylinux uClinux]# gzip -dc ~/incoming/uClinux-dist-lpc22xx.patch.gz |
     patch -p0
```


© NXP B.V. 2007. All rights reserved.

The patch is for the new vendor/product item. The result:

```
5    [root@mylinux uClinux-dist]# cd uClinux-dist/vendors
6    [root@mylinux vendors]# cd Philips/LPC22xx
7    [root@mylinux LPC22xx]# ls
8    config.arch                        config.uClibc     inittab    motb
          rc
9    config.linux-2.6.x             config.vendor     Makefile   passwd
```

3.3) Others

Since linux-2.6.10 and later version, the armnommu and arm architecture are combined, user need modify file below:

uClinux-dist/vendosr/config/armnommu/config.arch line 41

from ARCH = armnommu

to ARCH= arm

An overview of the uClinux source code tree is shown in Fig 1.

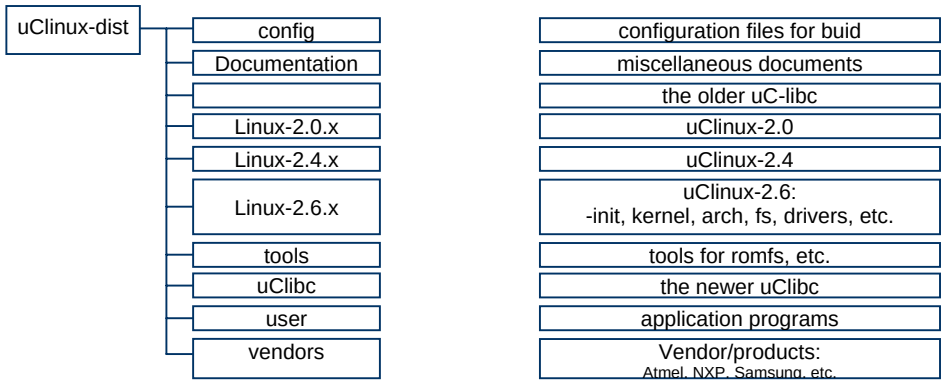

Fig 1. uClinux source code tree

2.4 Get and install cross toolchain

1) Get ARM-ELF toolchain

Since the linux 2.6 kernel code uses some new features of newer binutils, you need a newer ARM-ELF toolchain for kernel compilation than the one on the http://www.nxp.com/external/uclinux website. You could find the latest toolchain at the same place as the kernel and patch download section above, under the Toolchains section.

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

2) Install the toolchain

At the root directory execute the arm-elf-tools-20040427.sh like:

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

2.5 Hardware platform

The hardware tools needed:

© NXP B.V. 2007. All rights reserved.

- LPC22xx development board and power supply for the board
- In-Circuit emulator, e.g. Multi-ICE by ARM company
- A PC with serial port and parallel or USB port for Emulator
- A serial cable

Besides the above necessary equipment, network cable and/or USB cable and others may be needed according to system requirement.

A uClinux development system is shown in Fig 2.

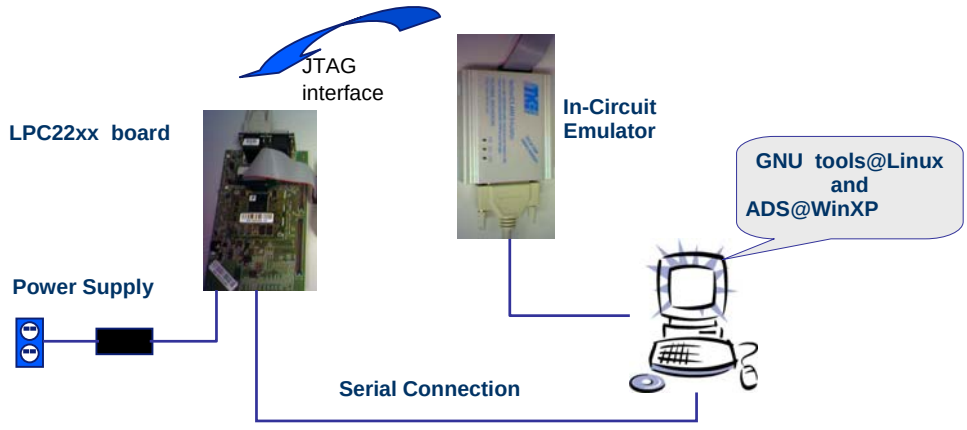

Fig 2. Hardware platform for uClinux development

3. System configuration

System configuration is to configurate the uClinux system options, customize functionality and make the image file size variable meanwhile.

We need setup the configuration for kernel, file systems and user applications.

```
11   [root@mylinux uClinux-dist]# make menuconfig
```

3.1 Distribution configuration

At the first Main Menu, Select the “Vendor/Product Selection” as shown in Fig 3.

© NXP B.V. 2007. All rights reserved.

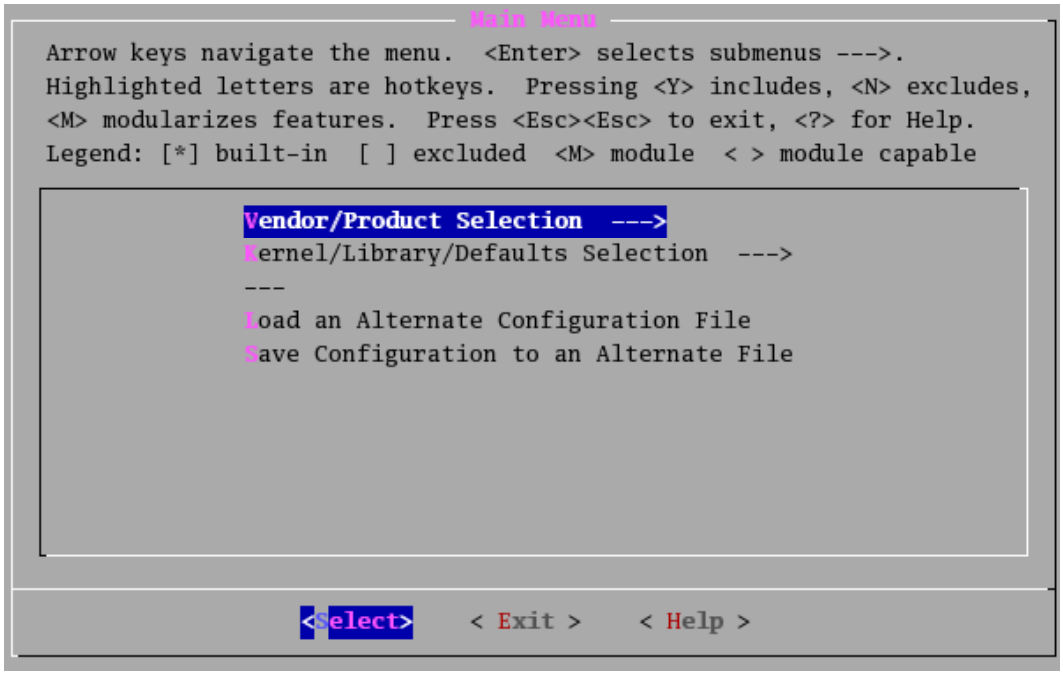

Fig 3. Main menu

Select “Philips” for the “Vendor”, and “LPC22xx” for the “Philips Products” (see Fig 4).

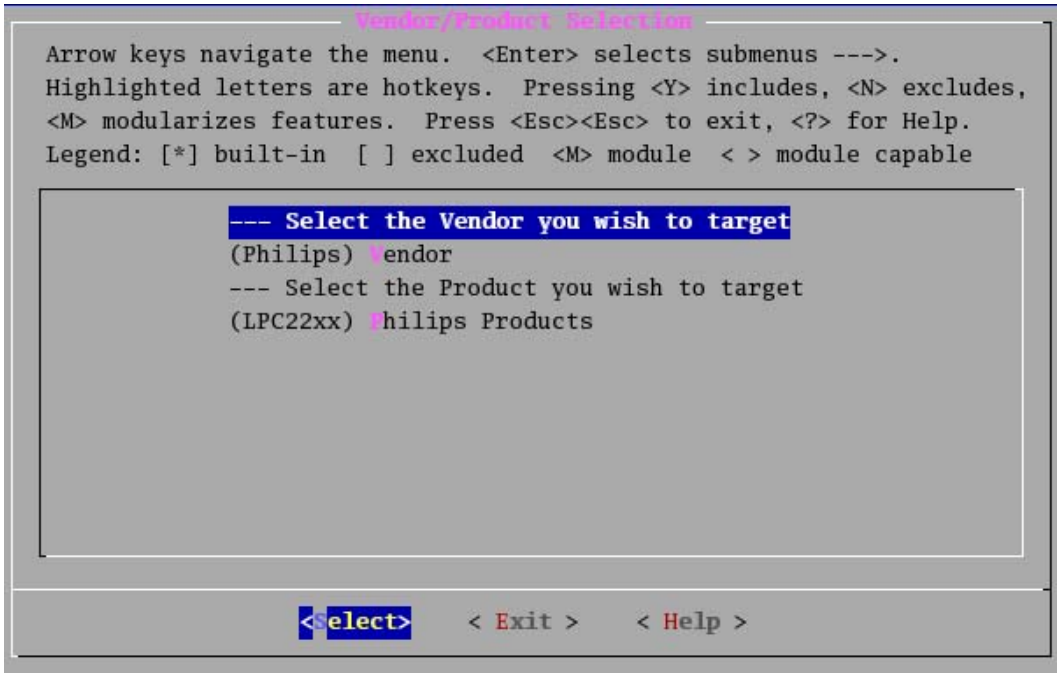

Fig 4. Vendor/product selection menu

© NXP B.V. 2007. All rights reserved.

You can go back to main menu with 'esc' key or “Exit” button.

At the Main Menu, Select the “Kernel/Library/Defaults Selection”. And Select “linux-2.6.x” for the “Kernel Version”, and “uClibc” for the “Libc version”.

And toggle the whole menus shown in Fig 5.

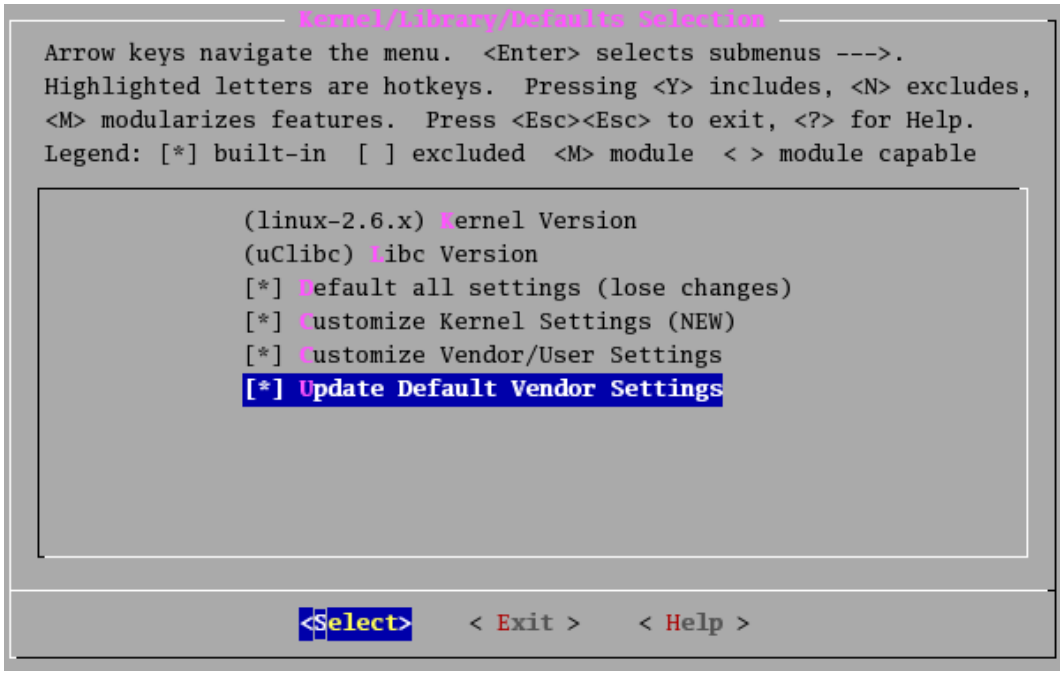

Fig 5. Kernel/library/defaults selection menu

With the first “default all settings”, the configuration files in the vendors directory is loaded.

With the second “Customize kernel settings”, we can edit the kernel configuration.

The third “Customize Vendor/User settings” is for configuration of applications and libraries configuration for making the romfs.img which will be the root file system of the kernel.

With the final “Update Default Vendor settings”, your changes on the kernel and user application configuration will be saved on the vendors/product directory.

In your developing or debugging stage, you can toggle the Kernel Settings and/or User Settings only.

You can go back to main menu with 'esc' key or “Exit” button, and do again for saving dialog. And save it!

© NXP B.V. 2007. All rights reserved.

3.2 Kernel configuration

If you follow the steps in “confirm the configuration files” above, your kernel should get configured.

Let's have a look at some related configuration items as shown in Fig 6.

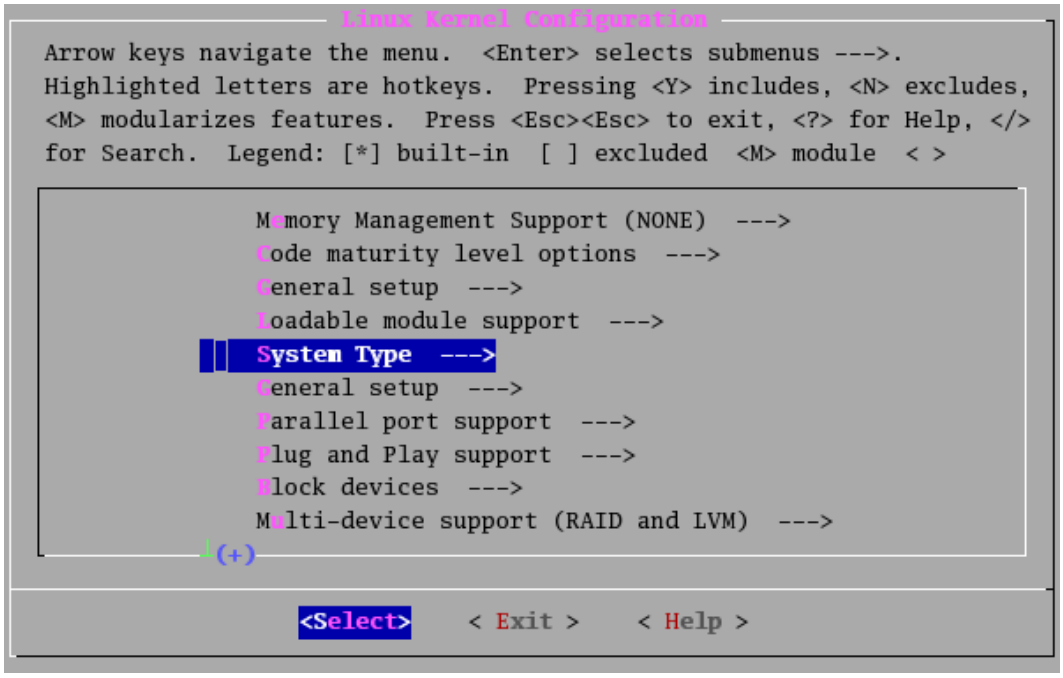

Fig 6. Linux kernel configuration menu

In the System Type option, the SRAM/FLASH base, address and size, the Oscillator Frequency (Fosc) should be consistent with your board. The ARM Core Clock (Fcclk) is your target CPU frequency.

© NXP B.V. 2007. All rights reserved.

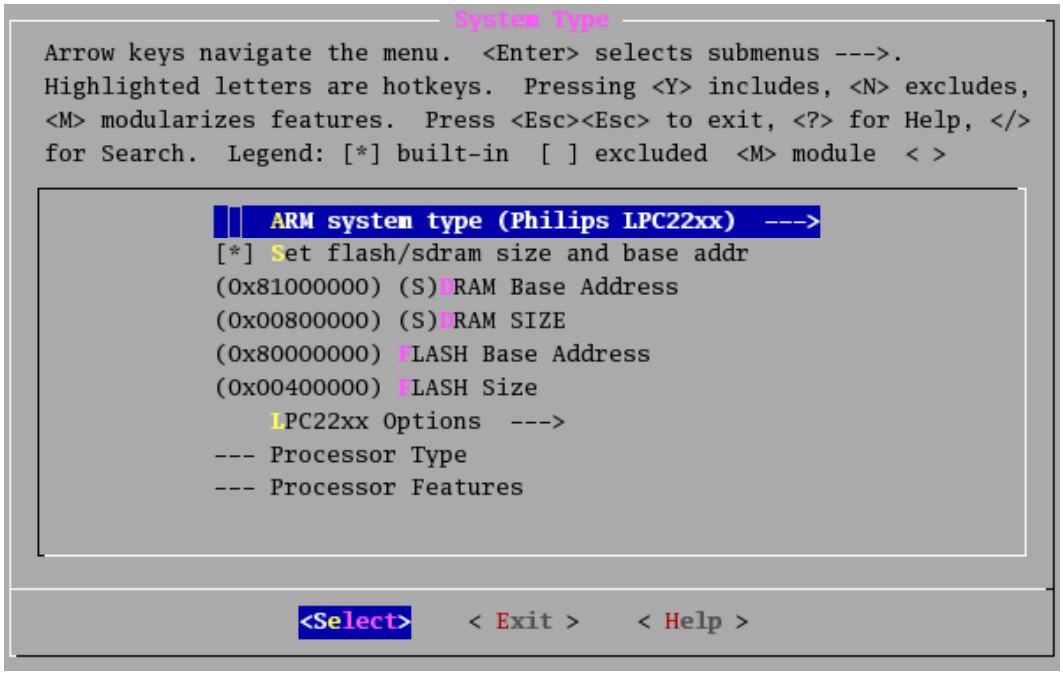

Fig 7. System type menu

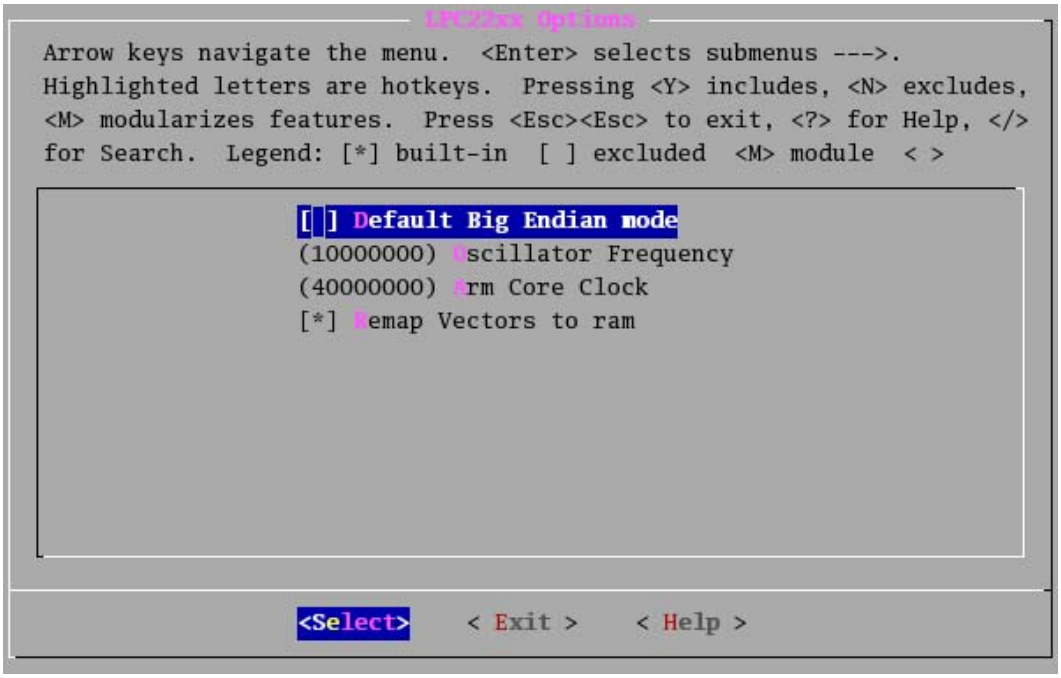

Fig 8. LPC22xx options menu

© NXP B.V. 2007. All rights reserved.

In the following General Setup option, please note the 'kernel command string' (Fig 9). The initial ramdisk base, address and size can be changed according to your romfs.img size.

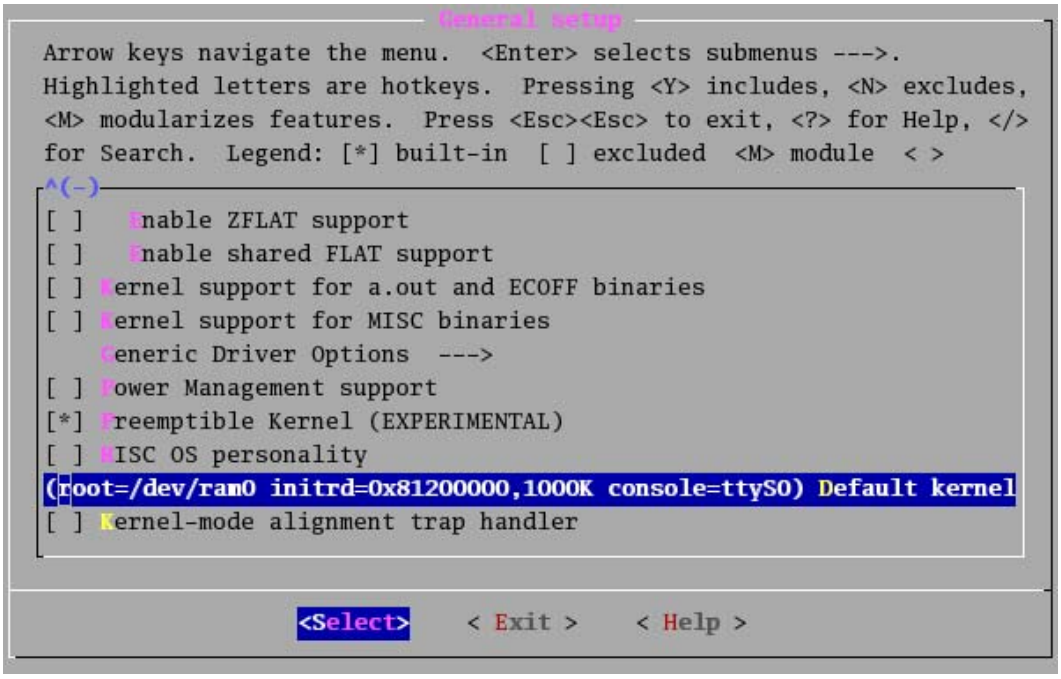

Fig 9. General setup menu

For the serial and console support, please enter into the option Character device→Serial drivers (see Fig 10). The UART of NXP LPC22xx is 16c550 compatible, so toggle two following items for serial and console input/output.

© NXP B.V. 2007. All rights reserved.

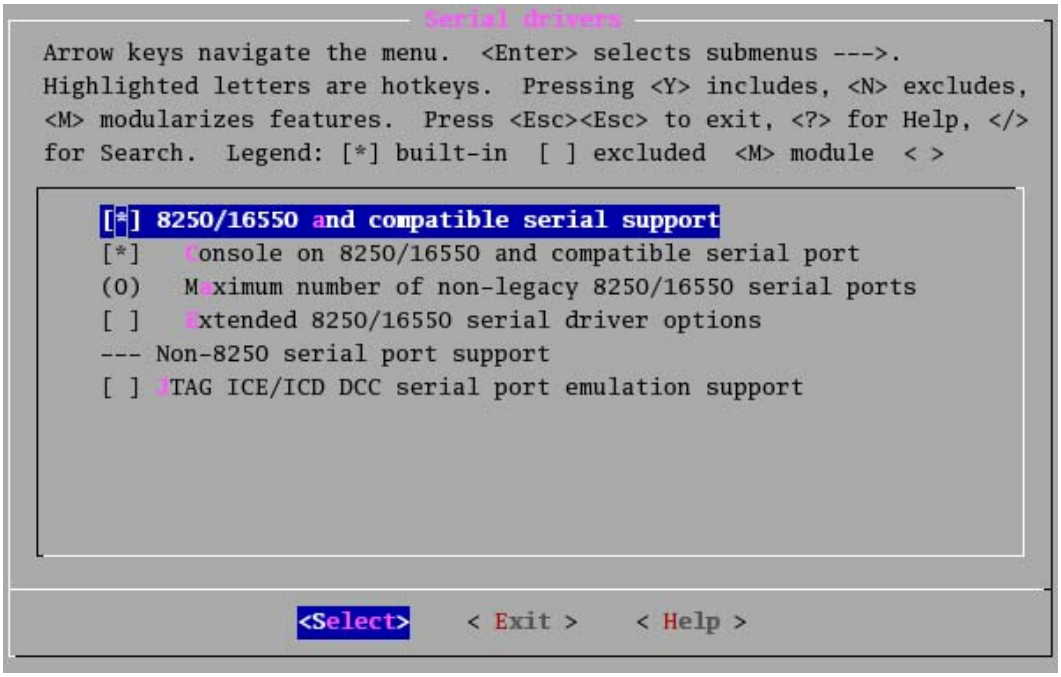

Fig 10. Serial drivers menu

For file systems options, the romfs(read-only) acts as the root file system which is necessary (see Fig 11), whereas the ext2 is read-writable but it consumes significant memory space. You can determine to select ext2 or not or even other file systems according to your memory size and your application.

© NXP B.V. 2007. All rights reserved.

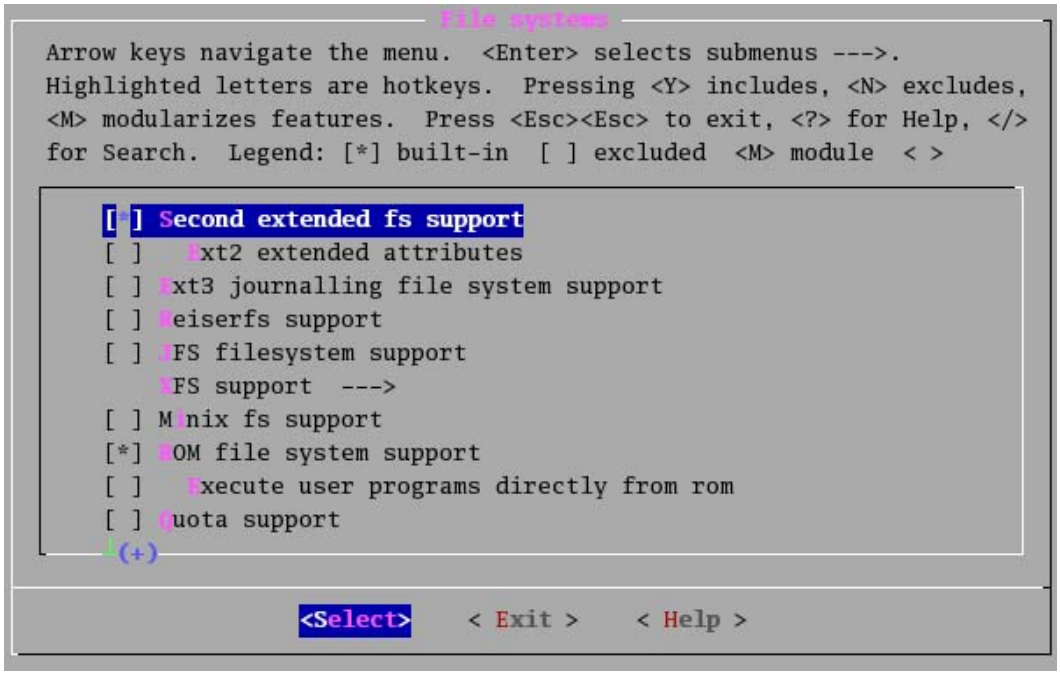

Fig 11. File systems menu

3.3 Application/library configuration

On this menu, you don't need fix anything for now. Look around the configurations and simply “exit” and save.

Only a kindly prompt that you can see an item:

Miscellaneous Applications--->

[*] hello

When you try to develop an application program “hello”, it will appear here. To toggle it will make it to be executable command of your uClinux system.

4. Build image

4.1 Make

We are ready to make the whole bunch. Simply type:

```
[root@mylinux uClinux-dist]# make
```

For linux-2.4.x, you should do “make dep” before “make”. But linux-2.6.x don't need to. It will compile the whole kernel and uClibc, user applications, and make the image files.

4.2 The results

You should get the files in the image directory like:

```
[root@mylinux uClinux-dist]# ls -al images
total 3036
drwxr-xr-x 2 root root 4096 Apr 27 22:13 .
```


© NXP B.V. 2007. All rights reserved.

```
drwxr-xr-x 17 1000 users 4096 Apr 27 22:13 ..
-rw-r--r-- 1 root root 1540272 Apr 27 22:13 image.bin
-rwxr-xr-x 1 root root 45912 Apr 27 22:13 linux.data
-rwxr-xr-x 1 root root 704856 Apr 27 22:13 linux.text
-rw-r--r-- 1 root root 789504 Apr 27 22:13 romfs.img

[root@mylinux uClinux-dist]# ls -al linux-2.6.x/linux*
-rwxr-xr-x 2 root root 499433 Apr 27 22:11 linux-2.6.x/linux
-rwxr-xr-x 2 root root 15499433 Apr 27 22:11 linux-2.6.x/linux.bin
```

If you get the same files, you've the whole kernel and rootfs image.

The linux.bin is the kernel image while the large file "linux" includes the debugging symbols. The "romfs.img" is the file system image.

"linux.bin" and "romfs.img" are the target images that will be programmed and run in the target board.

5. Load image

5.1 Before loading

Before loading Linux kernel and file system images, let’s do some base jobs and know some general knowledge.

5.1.1 Interrupt vector table

For LPC22xx, the interrupt vectors can be mapped to different memory space. The 4 mapping modes are controlled by the MEMMAP register of LPC22xx: boot block, internal Flash, intenal RAM, or external memory. For details about mapping

The Linux system will generate its own interrupt vector table at the beginning address of SRAM automatically. E.g. 0x8100 0000.

So we need to set up a direct link from LPC22xx interrupt vectors to Linux system interrupt vectors.

User should program some code at internal or external flash and make it direct to Linux interrupt vector table.

E.g.

```
    CODE32
    AREA    vectors,CODE,READONLY
    ENTRY
Reset
        LDR     PC, ResetAddr
        LDR     PC, UndefinedAddr
        LDR     PC, SWI_Addr
        LDR     PC, PrefetchAddr
        LDR     PC, DataAbortAddr
        DCD     0xb9205f80
        LDR     PC, IRQ_Addr
        LDR     PC, FIQ_Addr

ResetAddr        DCD     ResetInit
UndefinedAddr    DCD     0x81000004
SWI_Addr         DCD     0x81000008
PrefetchAddr     DCD     0x8100000c
DataAbortAddr    DCD     0x81000010
Nouse            DCD     0
IRQ_Addr         DCD     0x81000018
FIQ_Addr         DCD     0x8100001c
```


© NXP B.V. 2007. All rights reserved.

5.1.2 Bootloader

Bootloader is an independent part from OS kernel, which will do necessary jobs before kernel booting:

- Necessary system initialization
- Load uClinux images to specific position
- Jump to kernel start and run

Bootloader is the very first program launched before uClinux kernel startup.

Besides the above basic functions, a bootloader can be designed with more capabilities as:

- Transfer kernel parameters from bootloader to kernel
- Download image files from PC to board
- Program board flash
- Initialize some peripheral devices for better debugging: UART, Ethernet interface, etc.
- Etc.

A user can design his or her own bootloader to implement basic function or to port some powerful bootloaders. e.g. U-Boot, blob, vivi, lilo, etc.

Here is a popular U-boot website for reference, which is a combination of PPCBoot and ARMBoot and is quite suitable for PowerPC and ARM CPU families.

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

We can also download a uClinux bootloader for LPC22xx by a German developer:

http://www.nxp.com/external/ulrichradig

5.2 Load to RAM

Usually in the debugging stage, the kernel and fs will be loaded to external SRAM.

- Some registers should be initialized before loading, e.g., PINSEL, BCFG, MEMMAP, etc.
- Load uClinux kernel and fs image to specified address
- Jump to kernel start address and run

Here using the ADS environment as example.

Open System Views →Command Line Interface in the AXD debugger and type “ob d:\ess\config.ini” in the window. Below is the content of config.ini file.

```
setmem 0xE002C000 0x80000005 32
setmem 0xE002C014 0x0f814924 32
setmem 0xFFE00000 0x2000aeef 32
setmem 0xFFE00004 0x20007c67 32
setmem 0xFFE00008 0x1000ffef 32
setmem 0xFFE0000c 0x0000ffef 32
setmem 0xE01FC040 0x01 8
lb d:\ess\linux.bin 0x81008000
lb d:\ess\romfs.img 0x81200000
pc 0x81008000
r
```


© NXP B.V. 2007. All rights reserved.

5.3 Load to flash

After the debugging stage, we need manage to program the bootloader and uClinux images to flash. uClinux kernel and fs images must be programmed to external flash. The bootloader can be written either into internal or external flash.

There are three ways to program LPC22xx internal flash: ISP, IAP and parallel programmer. For details, please refer the LPC22xx user manual.

To program external flash, we can adopt below methods:

- By parallel flash programmer (hardware): for off-board programming
- By Jtag interface and (Wringler or In-Circuit Emulator)
- By Ethernet interface
- By USB interface

For the latter 3 methods, there will have related software running on PC and connect PC with board by corresponding interface. The software would be different according to your flash vendor and part no.

5.4 Memory layout

The generated image files as well as bootloader will be programmed to board flash. Bootloader will lead the system and start up from board flash. Then the uClnux kernel can be started from flash directly or copied to RAM area for better speed performance.

Usually the memory size needed for uClinux system is as:

- Minima

  Flash: 2 MB; RAM: 2 MB

  “Minima” means that only the uClinux kernel and necessary root file system can run.

  If the images are compressed to flash, the minimum flash size can be less to 512 kB.

- Typical

  Flash: 4 MB; RAM: 4 MB

  A “Typical” system can include more than 1 file system, Ethernet driver, LCD driver and some commonly used device drivers.

  For video, audio or some complicated systems, they can take more memory space.

6. Start up uClinux

6.1 Run uClinux on the board

If you have written the bootloader as well as uClinux kernel and file system images into the flash, you are ready to run uClinux on your board.

- Connect the board UART0 with PC COM port by serial cable
- Open a serial terminal on the PC and set its parameters as "9600 8n1n"
- Power on the board

© NXP B.V. 2007. All rights reserved.

Below is the example running process. It may be a little different according to your system.

```
---------------------------------------------------------------------------------------------------------------
Linux version 2.6.11.8-ucLPC (root@localhost.localdomain) (gcc version 2.95.3 2
0010315 (release)
Tue Nov 16 09:19:46 CST 2004
CPU: Philips-LPC22xx [22000000] revision 0 (ARMvundefined/unknown)
Machine: LPC2294, PHILIPS ELECTRONICS Co., Ltd.
On node 0 totalpages: 2048
  DMA zone: 0 pages, LIFO batch:1
  Normal zone: 2048 pages, LIFO batch:1
  HighMem zone: 0 pages, LIFO batch:1
Built 1 zonelists
Kernel command line: root=/dev/ram0 initrd=0x81200000,1000K console=ttyS0
PID hash table entries: 64 (order 6: 512 bytes)
Memory: 8MB = 8MB total
Memory: 5896 KB available (994K code, 133K data, 48K init)
Calibrating delay loop... 3.57 BogoMIPS
Dentry cache hash table entries: 1024 (order: 0, 4096 bytes)
Inode-cache hash table entries: 1024 (order: 0, 4096 bytes)
Mount-cache hash table entries: 512 (order: 0, 4096 bytes)
checking if image is initramfs...it isn't (ungzip failed); looks like an ini
trd
Freeing initrd memory: 1000K
POSIX conformance testing by UNIFIX
Serial: 8250/16550 driver $Revision: 1.90 $ 2 ports, IRQ sharing disabled
ttyS0 at MMIO 0x0 (irq = 6) is a 16550A
ttyS1 at MMIO 0x0 (irq = 7) is a 16550A
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
loop: loaded (max 8 devices)
RAMDISK: romfs filesystem found at block 0
RAMDISK: Loading 948 blocks [1 disk] into ram disk... done.
VFS: Mounted root (romfs filesystem) readonly.
Freeing init memory: 48K
Shell invoked to run file: /etc/rc
Command: hostname Philips-LPC2294
Command: /bin/expand /etc/ramfs.img /dev/ram1
Command: mount -t proc proc /proc
Command: mount -t ext2 /dev/ram1 /var
Command: mkdir /var/tmp
Command: mkdir /var/log
Command: mkdir /var/run
Command: mkdir /var/lock
Command: mkdir /var/empty
Command: cat /etc/motd
Welcome to

          ____ _   _
         /  __| ||_|
  _   _| |  | | _ ____  _   _  _  _
 | | | | |  | || |  _ \| | | |\ \/ /
 | |_| | |__| || | | | | |_| |/    \
 |  ___\____|_||_|_| |_|\____|\_/\_/
 | |
 |_|

For further information check:
http://www.nxp.com/external/uclinux

Command: ifconfig lo 127.0.0.1
Command: route add -net 127.0.0.0 netmask 255.255.255.0 lo
Command: dhcpcd &
[13]
Command: sh

Sash command shell (version 1.1.1)
/> Reading command line: Bad file descriptor
pid 14: failed 256
Execution Finished, Exiting
```


© NXP B.V. 2007. All rights reserved.

```
init: Booting to single user mode

Sash command shell (version 1.1.1)
/>ls -la
drwxr-xr-x  1 0        0              32  Jan  1 00:00 .
drwxr-xr-x  1 0        0              32  Jan  1 00:00 ..
drwxr-xr-x  1 0        0              32  Jan  1 00:00 bin
drwxr-xr-x  1 0        0              32  Jan  1 00:00 dev
drwxr-xr-x  1 0        0              32  Jan  1 00:00 etc
drwxr-xr-x  1 0        0              32  Jan  1 00:00 home
drwxr-xr-x  1 0        0              32  Jan  1 00:00 lib
drwxr-xr-x  1 0        0              32  Jan  1 00:00 mnt
dr-xr-xr-x 18 0        0               0  Jan  1 00:00 proc
lrwxrwxrwx  1 0        0               4  Jan  1 00:00 sbin -> /bin
lrwxrwxrwx  1 0        0               8  Jan  1 00:00 tmp -> /var/tmp
drwxr-xr-x  1 0        0              32  Jan  1 00:00 usr
drwxr-xr-x  7 0        0            1024  Jan  1 00:00 var
/> ps
  PID PORT STAT  SIZE SHARED %CPU COMMAND
    1      S     163K     0K  7.3 /sbin/init
    2      S       0K     0K  0.0 ksoftirqd/0
    3      S       0K     0K  0.0 events/0
    4      S       0K     0K  0.0 kblockd/0
    5      S       0K     0K  0.0 pdflush
    6      S       0K     0K  0.0 pdflush
    8      S       0K     0K  0.0 aio/0
    7      S       0K     0K  0.0 kswapd0
   15      R     102K     0K 11.8 /bin/sh
/> cat /proc/meminfo
MemTotal:         6968 kB
MemFree:          4648 kB
Buffers:          1076 kB
Cached:            400 kB
SwapCached:          0 kB
Active:           1160 kB
Inactive:          300 kB
HighTotal:           0 kB
HighFree:            0 kB
LowTotal:         6968 kB
LowFree:          4648 kB
SwapTotal:           0 kB
SwapFree:            0 kB
Dirty:               0 kB
Writeback:           0 kB
Mapped:              0 kB
Slab:              700 kB
Committed_AS:        0 kB
PageTables:          0 kB
VmallocTotal:  4194303 kB
VmallocUsed:         0 kB
VmallocChunk:  4194303 kB

/>cd var
/var>cd lucy
/var/lucy>hello

/******** Hello Philips! ********/

/******** Hello LPC22xx! ********/

/**** Welcome using uClinux for Philips LPC22xx! ****/
```

Under the uClinux shell, you can type some Linux command to check the effect.

E.g. “ls, ps, cat, cd, etc.”

And you can run your first “Hello, the World” program in the shell, too.

© NXP B.V. 2007. All rights reserved.

6.2 uClinux startup process

Fig 12 can give you a rough idea about uClinux internal startup process, for your reference.

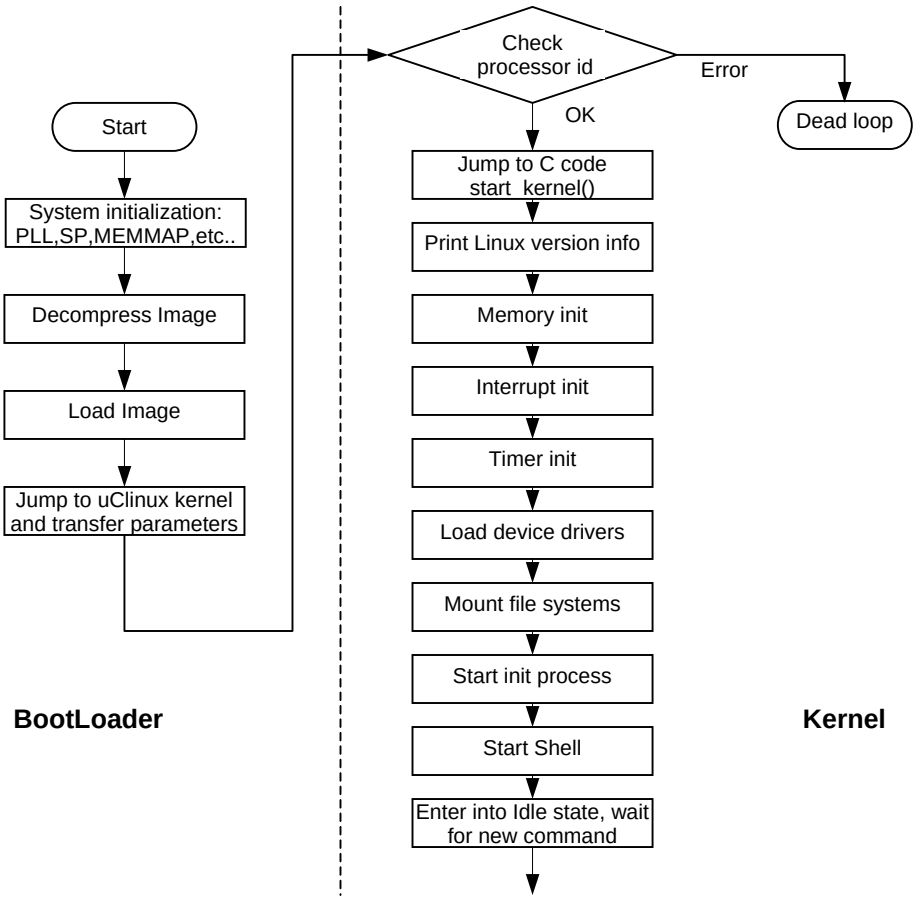

Fig 12. uClinux start-up process

7. uClinux development

This section gives a brief introduction on how to develop device driver, how to develop user applications and how to do debugging based on the established uClinux development platform.

7.1 Develop device drivers

To develop user specific device drivers, here are some references:

- Documents: uClinux-dist\linux-2.6.x\Documentation\driver-model\
- Classical book: *Linux Device Driver: the 2nd edition,* by O'REILLY

Actually the more handy way is to find some similar or related device drivers under the source code directory: uClinux-dist\linux-2.6.x\drivers. Especially when there exist compatible device drivers, the development would become quite laborsaving. It becomes porting instead of creating.

© NXP B.V. 2007. All rights reserved.

To develop the Ethernet driver of RTL8019AS, which is NE2000 compatible, we can find the related source files here \uClinux-dist\linux-2.6.x\drivers\net: ne.c, 8390.c, 8390.h. By providing some basic information 'the SFR startup address, interrupt number, etc.', we can get the new RTL8019AS Ethernet driver for our own board.

7.2 Develop user applications

On how to develop user application program, there is a pretty good user guide under the \Documentation directory of uClinux source code tree.

uClinux-dist\Documentation\Adding-User-Apps-HOWTO

By following the document step by step, we can create and run our first 'Hello, the world!' program in 1 hour.

7.3 Debugging

7.3.1 Debug OS kernel and device driver

There are several ways to debug uClinux kernel and device driver program. E.g.

1) AXD + MultiICE + MultiICE-Server

AXD: ADS Debugger by ARM company, running under Windows environment.

MultiICE: ARM Emulator by ARM company, parallel port interface to PC and JTAG interface to the board.

MultiICE-Sever: driver program for MultiICE, running under Windows environment

2) arm-elf-gdb(insight) + MultiICE-gdb-server + MultiICE + MultiICE-Server

arm-elf-gdb(insight): GNU cross debugger for ARM architecture, running under Linux environment. arm-elf-insight is the GUI version of arm-elf-gdb.

MultiICE-gdb-server: a server for ADS tools to debug GNU program, free provided by ARM company and running under Windows environment.

MultiICE: same as above

MultiICE-Server: same as above

But until now, it seems there is no perfect method for source code awareness. Usually it should be assisted by 'printk' – the original debugging method for Linux kernel debugging. For using 'printk', the serial/UART interface should be initialized in the bootloader previously.

7.3.2 Debug application program

Debugging uClinux application program is relatively easier than kernel debugging. By using arm-elf-gdb and arm-elf-gdb server as a stuck on the user board, the user program can be debugged under PC Linux environment.

Actually user can debug application program under PC Linux environment first and then download to the board for further verification.

© NXP B.V. 2007. All rights reserved.

8. Legal information

8.1 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.

8.2 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.

8.3 Trademarks

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

© NXP B.V. 2007. All rights reserved.

9. Contents

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

© NXP B.V. 2007. All rights reserved.

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: AN10389_1

founded by
PHILIPS