RD4730_PMP Development Board

Software Development Guide

Version: 1.1

Data: Jun. 2006

北京君正集成电路有限公司

Ingenic Semiconductor Co. Ltd

Ingenic RD4730_PMP Development Board

Software Development Guide

Release history

Date	Revision	Change
Jun. 2006	1.1	Modified the description of MMC/SD, camera and busybox.
Apr. 2006	1.0	First release

Disclaimer

This documentation is provided for use with Ingenic products. No license to Ingenic property rights is granted. Ingenic assumes no liability, provides no warranty either expressed or implied relating to the usage, or intellectual property right infringement except as provided for by Ingenic Terms and Conditions of Sale.

Ingenic products are not designed for and should not be used in any medical or life sustaining or supporting equipment.

All information in this document should be treated as preliminary. Ingenic may make changes to this document without notice. Anyone relying on this documentation should contact Ingenic for the current documentation and errata.

Ingenic Semiconductor Co. Ltd

22th Floor, Building A, Cyber Tower, No.2, Zhong Guan Cun South Avenue
Haidian District, Beijing 100086, China
Tel: 86-10-82511297
Fax: 86-10-82511589
Http: //www.ingenic.cn

Content

1 Overview

This document describes the procedure in developing Linux based on the RD4730_PMP development board, including how to install cross compiling environment on the host, how to compile U-Boot and Linux kernel, how to download the bootloader to run the Linux kernel and how to make the root file system etc.

During the development stage, a PC server is required to provide the DHCP, TFTP and NFS services. DHCP provides IP address for the target board, TFTP provides a way for RD4730_PMP to download Linux kernel through Ethernet and NFS provides Network File System to be mounted for RD4730_PMP.

RD4730_PMP boots from NOR FLASH，and the bootloader is U-Boot. U-Boot is an open source project suitable for embedded system, and can provide several ways to download programs likes serial port and Ethernet. U-Boot saves the boot configuration on the Flash memory and provides a flexible way to configure your system.

RD4730_PMP runs CE-Linux™ which was distributed by the CELF (Consumer Electronics Linux Forum) organization. CE-Linux was based on the Linux 2.4.20 kernel and was added with many functionalities aiming for embedded system, such as preemptible kernel, O(1) scheduler, POSIX timer, Dynamic Power Management and fast boot sequence etc.

RD4730_PMP uses BusyBox-1.1.3 as the root file system. It supports the shared libraries of GLIBC and other related tools.

RD4730_PMP can run QTOPIA/QTE，and can play MP3/MPEG4，can view JPEG image and play games。It can also support keypad, touchpanel and simple handwriting, and can display the baterry life etc.

2 Install Development Environment

RD4730_PMP provides fully free development tools and source packages as follows:

- mipseltools-jz-linux-20060308.tar.bz2：MIPS little-endian compiler and debugger on Linux host.
- mipseltools-jz-cygwin-20060308.tar.bz2：MIPS little-endian compiler and debugger on Windows host, need to install Cygwin.
- u-boot-1.1.3.tar.bz2：U-Boot source package
- u-boot-1.1.3-jz-yyyymmdd.patch.gz: U-Boot patch for RD4730_PMP.
- celinux-040503.tar.bz2：CELinux source package
- celinux-jz-yyyymmdd.patch.gz：CELinux patch for RD4730_PMP.
- pmp-root.tar.bz2：PMP root file system
- pmp-utils-src.tar.bz2：PMP utilities

Note：please download the latest patches of U-Boot and CELinux from our FTP site (ftp://ftp.ingenic.cn)

2.1 Install MIPS Compiling Tools

To compile U-Boot, Linux and Applications, you need to install the MIPS cross compiling tools. We provide compiling tools for Linux and Windows host. And we recommend you use a Linux as the development host.

The steps to install MIPS cross compiler are as follows:

First, make a working directory, and copy the tool package to it:

```
$ mkdir –p  /opt/crosstool
$ cd /opt/crosstool
$ tar -xjf mipseltools-jz-linux-20060308.tar.bz2
```

Then, put the compiler path to the PATH environment:

```
$ export PATH=/opt/crosstool/mipseltools/bin:$PATH
```

To install MIPS cross compiler on Windows host, follow next steps:

First of all, you need to install Cygwin on Windows and configure to work properly. You can visit homepage of Cygwin and get install instructions from it. If Cygwin is installer, run its shell program and follow next steps to install the compiler tools:

```
$ mkdir –p /opt/crosstool
```

```
$ cd /opt/crosstool
$ tar xjf mipseltools-jz-cygwin-20060308.tar.bz2
$ export PATH=/opt/crosstool/mipseltools/bin:$PATH
```

2.2 Install and Configure NFS

Network File System provides a way to mount root filesystem on network for RD4730_PMP. You can unpack the root FS package to a directory and then configure your host to export the directory for use, for example:

```
# mkdir -p /nfsroot/pmp-root
# cd /nfsroot/pmp-root
# tar -xjf pmp-root.tar.bz2
```

Open and edit /etc/exports, then restart the NFS service:

```
# cat /etc/exports
/nfsroot/pmp-root    (rw,no_root_squash)
# /etc/rc.d/init.d/nfs restart
```

Now you should be able to mount /nfsroot/pmp-root from another machine on the Ethernet. Suppose your Linux host IP is 192.168.1.20, test it like this:

```
# mount –t nfs 192.168.1.20:/nfsroot/pmp-root /mnt
```

If mounting is successful, it indicates that the NFS server on the host is working properly.

2.3 Start the TFTP Service

U-Boot uses TFTP to download Linux kernel on the host to local memory. So you need to configure and start the TFTP service on the host.

On Linux, suppose you use /tftpboot as TFTP service root, edit /etc/xinetd.d/tftp as follows:

```
service tftp
{
   disable             =   no
   socket_type         =   dgram
   protocol            =   udp
   wait                =   yes
   user                =   root
   server              =   /usr/sbin/in.tftp.d
   server_args         =   -s /tftpboot
   … …
}
```

Then，restart xinet:

```
# /etc/init.d/xinetd    restart
```

2.4 Install U-Boot and CELinux

Now you want to copy and unpack U-Boot and CELinux packages to a working directory, for example:

```
# mkdir –p /home/junzheng
# cd /home/junzheng

# tar –xjf u-boot-1.1.3.tar.bz2
# cd u-boot-1.1.3
# zcat ../u-boot-1.1.3-jz-yyyymmdd.patch.gz | patch –p1
# cd ..
# tar –xjf celinux-040503
# cd celinux-040503
# zcat ../celinux-jz-yyyymmdd.patch.gz | patch –p1
```

2.5 Install PMP Utilities

We provide some PMP utilities such as dynamic power management daemon and camera capturing and displaying tools. Copy and unpack the package to your working directory, and build it as follows:

```
# tar –xjf pmp-utils-src.tar.bz2
# cd pmp-utils
```

```
# cd dpmd
# make clean; make
# cd ..
# cd camera
# make clean; make
```

3 U-Boot and Linux

This section describes how to configure and build U-Boot and Linux, and how to run U-Boot and Linux on RD4730_PMP development board.

3.1 Building U-Boot and Linux

The steps to configure and build U-Boot are:

```
# make pmp_config
# make
```

At the end, you can get the U-Boot binary called u-boot.bin. Then you can copy u-boot.bin to the TFTP directory and use JDI to burn the binary to the NOR Flash of the RD4730_PMP board.

Note: JZ4730 CPU boots from the NOR Flash address 0xbfc00000. So U-Boot should be burned to address 0xbfc00000。

The steps to configure and build Linux are:

```
# make defconfig-pmp
# make xconfig
# make dep
# make uImage
# make zImage
```

“make uImage” will generate a binary ‘uImage’ under arch/mips/uboot/. The binary format is specific for U-Boot. “make zImage” will generate another compressed binary ‘zImage’ under arch/mips/zboot/images/. Now copy uImage to the TFTP directory. Then you can boot the target board and download Linux to execute.

If U-Boot has been burned to the NOR Flash of the RD4730_PMP board, then you can connect the serial line, the Ethernet line and the power of the RD4730_PMP board. The serial terminal on the host side should be configure to 115200bps, 8-bit data, 1-bit stop, no odd/even and no flow control. Open power of the RD4730_PMP board by pressing SW1 for a while, then you will receive the booting messages on the serial terminal.

Press any key on the serial terminal to stop the autoboot of U-Boot and enter its command line interface. Now you can configure U-Boot to download and boot Linux kernel. First of all, type ‘help’ to get all the commands supported by U-Boot:

```
PMP # help
PMP # help tftpboot
```

Refer to next steps to configure U-Boot to download Linux through TFTP and Linux mounts root through NFS. This supposes local IP is 192.168.1.10 and local MAC is 00:12:34:56:78:90, the host IP is 192.168.1.20, the name of the Linux kernel is uImage under TFTP root directory, and the NFS root path is /nfsroot/pmp-root.

```
PMP # setenv ipaddr 192.168.1.10
PMP # setenv serverip 192.168.1.20
PMP # setenv bootfile uImage
PMP # askenv bootcmd
bootcmd: tftpboot;bootm
PMP # setenv bootargs mem=64M console=ttyS3,115200 ip=192.168.1.10
ethaddr=00:12:34:56:78:90 nfsroot=192.168.1.20:/nfsroot/pmp-root rw
PMP # saveenv
PMP # boot
```

‘saveenv’ will save all the configurations on the NOR Flash. Run ‘boot’ will boot the system. Here U-Boot will first download Linux kernel from the host through TFTP and then boot the Linux kernel by running ‘bootm’ command. During Linux booting, Linux will mount its root through NFS and run the init program of the root FS.

You can also download Linux kernel to NOR Flash through TFTP and then boot Linux from NOR Flash, refer to next examples:

```
PMP # setenv ipaddr 192.168.1.10
PMP # setenv serverip 192.168.1.20
PMP # erase 0xbfd00000 0xbfffffff
PMP # tftpboot 0xbfd00000 uImage
PMP # setenv bootcmd bootm 0x9fd00000
PMP # saveenv
PMP # boot
```

Both 0xbfd00000 and 0x9fd00000 are mapped to the same physical address 0x1fc00000 on NOR Flash. The difference is that 0xbfc00000 is uncacheable while 0x9fd00000 is cacheable.

3.2 Configure the Linux Device Drivers

The Linux source package we provide includes many device drivers for RD4730_PMP development board. This section will simply describe some important drivers.

3.2.1 Preemptible Kernel

CELinux preemptible kernel enhances the realtime scheduling of the processes. Select CONFIG_PREEMPT=y to configure the kernel:

```
[General setup]
   [•]y  [ ]-  [ ]n     Preemptible Kernel
```

3.2.2 Dynamic Power Management

CELinux supports Dynamic Power Management. DPM provides a way to change the clock of CPU on the fly, to suspend and wakeup CPU to save power of the system.

```
[Power Management]
   [•]y  [ ]-  [ ]n     Power Management support
   [•]y  [ ]-  [ ]n     Enable Dynamic Power Management Support
   [•]y  [ ]-  [ ]n     Advanced Power Management (APM) Support
```

3.2.3 RTC Driver

RD4730_PMP use the PHILIPS's PCF8563 as RTC，configure CELinux to select the driver:

```
[Character devices]
   [•]y  [ ]-  [ ]n     Philips PCF8563 Real Time Clock (I2C Bus)
```

3.2.4 MTD Device Driver

CELinux supports MTD drivers based on both NOR Flash and NAND Flash. Please refer to defconfig-pmp for its configuration.

3.2.5 LCD Framebuffer

JZ4730 processor has a LCD controller. It can connect TFT/STN panels. We implement a framebuffer driver for the LCD.

```
[Console drivers]
  [Frame-buffer support]
   [•]y  [ ]-  [ ]n     Support for frame buffer devices
   [•]y  [ ]-  [ ]n     JzSOC OnChip LCD controller support
   [•]y  [ ]-  [ ]n     SAMSUNG LTP400WQF01 TFT panel (480x272)
```

3.2.6 Audio

JZ4730 processor has AC97 and I2S audio controller. RD4730_PMP use a AC97 codec.

```
    [Sound]
        [•]y  [ ]m  [ ]n     Sound card support
        [•]y  [ ]m  [ ]n     JzSOC AC97 sound
```

3.2.7 MMC/SD

JZ4730 processor has a MMC/SD controller. It can support MMC and SD cards.

```
    [MMC/SD Card support]
        [•]y  [ ]m  [ ]n     MMC support
        [•]y  [ ]m  [ ]n     JZ4730 MMC/SD
```

3.2.8 JFFS2 and YAFFS

Configure CELinux to support JFFS2 and YAFFS filesystem, like this:

```
    [File systems]
        [•]y  [ ]m  [ ]n  Yet Another Flash Filing System (YAFFS) support
        [•]y  [ ]m  [ ]n  Journaling Flash File System v2 (JFFS2) support
```

3.2.9 USB Host and Gadget

JZ4730 has a USB 1.1 compliant HOST and DEVICE controller. CELinux implements the USB gadget driver.

```
    [USB support]
        [•]y  [ ]m  [ ]n  Support for USB
        [•]y  [ ]m  [ ]n  OHCI(Compaq…) support
        [•]y  [ ]m  [ ]n  JzSOC OnChip OHCI-compatible host interface
support
        [Support for USB gadgets]
        [ ]y  [•]m  [ ]n  Support for USB Gadgets
        [JzSOC-UDC]       USB Peripheral Controller Driver
        [ ]y  [•]m  [ ]n  Gadget Zero (DEVELOPMENT)
        [ ]y  [•]m  [ ]n  File-backed Storage gadget (DEVELOPMENT)
```

USB Gadget should be built as modules and use insmod to load it.

3.2.10 Camera Device Driver

JZ4730 processor has a camera interface to the camera sensor. It has a 8-bit data interface. CELinux implement a camera driver for capturing frame data from the camera sensor. The camera sensor is initialized through I2C interface in another driver.

```
[Character devices]
    [•]y  [ ]m  [ ]n  JzSOC char devices support
    [•]y  [ ]m  [ ]n  JzSOC Camera Interface Module(CIM) support
    [•]y  [ ]m  [ ]n  JzSOC Common CMOS Camera Sensor driver
```

3.2.11 Touch Panel Driver

```
[Character devices]
    [•]y  [ ]m  [ ]n  JzSOC char devices support
        [•]y  [ ]m  [ ]n  JzSOC touchpanel driver support
        [UCB1400]         Touchpanel codec type
```

3.2.12 Touch Panel Driver

```
[Character devices]
    [•]y  [ ]m  [ ]n  JzSOC char devices support
        [•]y  [ ]m  [ ]n  PMP board poweroff support
        [•]y  [ ]m  [ ]n  PMP board key support
```

4 Root FS

Linux requires a root filesystem to execute init shells and user applications. This section describes how to make a root filesystem using Busybox.

4.1 BusyBox

We can use BusyBox to make a very small filesystem. This is very suitable for embedded system where flash and memory are critical. BusyBox implements a simple shell including all the necessary commands into one single binary.

4.1.1 Configure and Compile BusyBox

Visit the homepage of BusyBox (http://www.busybox.net) and download the source package. Here we use busybox-1.1.3 to illustrate the steps to compile busybox.

Compiling BusyBox is similar to Linux:

```
# make defconfig
# make menuconfig
# make
# make install
```

First，you need to configure BusyBox:

```
# make menuconfig
————————————————— BusyBox Configuration —————————————————
    Busybox Settings  --->
--- Applets
    Archive Utilities  --->
    Coreutils  --->
    Console Utilities  --->
    Debian Utilities  --->
    Editors  --->
    Finding Utilities  --->
    Init Utilities  --->
    Login/Passwd Management Utilities  --->
    Linux EXT2 FS Progs  --->
    Linux Module Utilities  --->
    Linux System Utilities  --->
```

```
    Miscellaneous Utilities  ---->
    Networking Utilities  ---->
    Process Utilities  ---->
    Shells  ---->
    System Logging Utilities  ---->
---
Load an Alternate Configuration File
Save Configuration to an Alternate File

Busybox Settings  ---->
    General Configuration  ---->
    Build Options  ---->
        [ ] Build BusyBox as a static binary (no shared libs)
        [*] Build with Large File Support (for accessing files > 2 GB)
        [*] Do you want to build BusyBox with a Cross Compiler?
        (mipsel-linux-) Cross Compiler prefix
        ( ) Any extra CFLAGS options for the compiler?
        [ ] Compile all sources at once
```

Here you should pay attention to two options: use shared libs, set cross compiler prefix as mipsel-linux-.

```
    Install Options  ---->
        [*] Don't use /usr
        (./_install) BusyBox installation prefix

    Linux Module Utilities  ---->
        [*] insmod
        [ ] In kernel memory optimization (uClinux only)
        [*] rmmod
        [*] lsmod
        [*] modprobe
        [*] Support version 2.2.x to 2.4.x Linux kernels
        …
```

Note: do not select [In kernel memory optimization] here.

```
    Shells  ---->
        Choose your default shell (ash)  ---->
```

Here select ash as the default shell.

```
    Networking Utilities  ---->
        [ ] Enable IPv6 support
```

```
    [*] telnetd
    [ ]     Support call from inet only
```

This will enable BusyBoxto support telnet server.

Other options are required according to the system.

Then you can make and install busybox.

4.1.2 Root FS Content

After make and make install, BusyBox will generate three directories: bin, sbin, lib and a file linuxrc. But these are not enough to build a full FS. You need to add other directories and files to it. The most important is dev, etc and shared libraries.

```
# mkdir dev etc lib mnt opt proc root tmp usr var
# ls
bin dev etc lib linuxrc mnt opt proc root sbin tmp usr var
```

The content of each directory can be copied from the ROOT FS package provided by us.

4.2 Using Root FS

This section describes how to put a root on NOR or NAND Flash. Generally there are two ways to make a FS for Linux: one is to make an initial ramdisk, then Linux will boot and mount root with the initial ramdisk. The other way is to make a file system on NOR or NAND Flash, then Linux will mount root on the Flash. Here we will simply describe these two ways.

The steps to make initrd are as follows:

```
# dd if=/dev/zero of=initrd bs=8192k count=1
# mke2fs -F -m0 -b 1024 initrd
# mount -t ext2 -o loop initrd /mnt
# cp -a /nfsroot/mips/pmproot/* /mnt/
# umount /mnt
# gzip -9 initrd
```

Then copy initrd.gz to Linux and rename to ramdisk.gz:

```
# cp initrd.gz /CELinux/arch/mips/ramdisk/ramdisk.gz
```

You need to configure Linux to select initrd:

```
[Block devices]
    [•]y  [ ]m  [ ]n  RAM disk support
    [8192]            Default RAM disk size
    [•]y  [ ]m  [ ]n  initial RAM disk (initrd) support
    [•]y  [ ]m  [ ]n  Embed root filesystem ramdisk into the kernel
    [ramdisk.gz]      Filename of gziped ramdisk image
```

Recompile Linux to generate an image with initrd. During booting Linux, you need to add an option “root=/dev/ram0” to the Linux command line.

The steps to make JFFS2 FS are as follows:

```
# mkfs.jffs2 –p –e 0x10000 –d /nfsroot/mips/pmproot –o pmproot.jffs2
```

where:

-d /nfsroot/mips/pmproot: specify the root directory
-e 0x10000: specify the eraseblock size of the FLASH
-p: specify the padding value of one eraseblock
-o pmproot.jffs2: is the output filename

If configuring Linux to support MTD, copy the JFFS2 image to MTD partition:

```
# cat /proc/mtd
# flash_eraseall –j /dev/mtd0
# cp pmproot.jffs2 /dev/mtd0
```

Add “root=/dev/mtdblock0 rw” to the Linux command line and boot Linux to test.

Another way is to use JFFS2 utilities to make the JFFS2 image, like this:

```
# flash_eraseall –j /dev/mtd0
# mount –t jffs2 /dev/mtdblock0 /mnt
# cp –a bin dev etc linux sbin proc … /mnt
# umount
```