

龙芯 3A、2K 开发与调试手册 v1.3

前 言

中科院计算所从 2001 年开始研制龙芯系列处理器，经过十多年的积累与发展，于 2010 年正式成立龙芯中科技术有限公司，旨在将龙芯处理器的研发成果产业化。

龙芯中科公司致力于龙芯系列 CPU 设计、生产、销售和服务。主要产品包括面向行业应用的专用小 CPU（1 号处理器），面向工控和终端类应用的中 CPU（2 号处理器），以及面向桌面与服务器类应用的大 CPU（3 号处理器）。目前，龙芯系列处理器已经在电力、网安、党政办公等领域广泛应用。

本手册旨在解决用户使用龙芯 2 号和 3 号处理器相关产品时的一些基本问题和调试板卡的基本方法和步骤。本手册共分为 5 部分/章，第一章主要介绍如何获取龙芯嵌入式提供的资源以及龙芯公司的公共资源，调试开发环境及工具；第二章主要介绍龙芯常用 bootloader 即 pmon 的编译，烧写及一些常用调试命令的使用；第三章主要介绍龙芯内核的编译，调试相关的步骤；第四章主要介绍龙芯板卡软硬件调试的基本步骤和内容；第五章主要介绍龙芯公司提供的免费桌面操作系统 loongnix 的安装方法及常见问题的解决方法。

我们希望通过本手册，能够为龙芯用户建立起一个调试开发基本的概念。能够让用户快速上手使用龙芯处理器的相关产品。

龙芯中科技术有限公司

嵌入式事业部

2019 年 1 月

目 录

目 录	1
第一章 资源获取.....	3
1.1 开发环境.....	3
1.2 bios 和 linux 内核	3
1.3 工具链.....	4
1.4 操作系统.....	4
1.5 ejtag.....	4
第二章 Pmon	5
2.1 pmon 相关资源	5
2.2 编译准备.....	5
2.2.1 安装 makedepend.....	5
2.2.2 安装 pmoncfg	5
2.3 pmon 编译	5
2.4 内存相关配置.....	6
2.5 pmon 命令介绍	8
2.5.1 pmon 烧写	8
2.5.2 内核启动相关命令.....	8
2.5.3 跟启动相关的环境变量.....	9
2.5.4 pmon 烧写内核及文件系统到 nand 方法	10
2.5.5 pmon 下查看及修改 dtb.....	11
2.5.6 其他命令简介.....	12
第三章 Kernel	15
3.1 Kernel 获取与编译	15
3.1.1 建立交叉编译环境.....	15
3.1.2 内核源码的获取.....	15
3.2 内核编译所需要的工具.....	16
3.3 内核编译命令	16
3.4 编译 busybox	17
3.5 buildroot 制作及交叉开发 qt 程序	17
3.6 ssd 下内核的替换	22
第四章 板卡调试.....	25
4.1 2K 龙芯派硬件调试.....	25
4.2 3A3000+7a 硬件调试部分	26
4.3 软件部分.....	27
第五章 Loongnix 安装说明	34
5.1 安装盘制作.....	34
5.1.1 下载系统 iso 文件	34
5.1.2 U 盘安装盘制作	34
5.2 安装过程.....	34

5.2.1 启动机器.....	34
5.2.2 开始安装.....	35
5.2.3 硬盘分区设置.....	36
5.2.4 用户设置.....	39
5.2.5 完成安装.....	40
5.3 注意事项.....	40
5.4 常见问题.....	40
附录 A 手工创建新系统分区	43
附录 B mtd 设备的使用	48
附录 C LS2K GPIO 中断配置	51
附录 D 配置 QT（交叉开发 QT 应用）	53

第一章 资源获取

龙芯公司在国产处理器领域深耕近二十年，积累了大量的 IP、系统软件、工具、第三方软件资源等。本章主要针对龙芯公司提供的处理器运行基本软件和 ejtag 调试工具资源的介绍和获取方式。

1.1 开发环境

建议用户使用 ubuntu 长期稳定支持版本作为开发宿主机系统，本文是基于 ubuntu16.04 和 ubuntu 18.04 交叉开发完成。龙芯的 fedora21 系统支持本地编译。

1.2 bios 和 linux 内核

目前，市场上常用的 bios 为 pmon、uboot、昆仑固件以及后续龙芯的 UEFI 等。Pmon 作为龙芯使用较为广泛的 bootloader，龙芯提供开源免费的 pmon。其它商业 bootloader 需对应公司提供，龙芯不提供任何资源。

不同处理器的 Pmon 下载链接如下：

1) 2k pmon

<ftp://ftp.loongnix.org/embedd/ls2k/pmon-loongson3.tar.gz>

2) 3a pmon

<ftp://ftp.loongnix.org/embedd/ls3a/bootloader/pmon-loongson3.tar.gz>

linux 内核，龙芯目前维护特定版本内核,3a 有 2.6.32 和 3.10 两个版本，2k 有 3.10 一个版本内核。3.10 是一个长期支持维护的版本，后续 linux 版本待定。

1) 2k 内核下载链接：

<ftp://ftp.loongnix.org/embedd/ls2k/linux-3.10.tar.gz>

2) 3a 内核下载链接

<ftp://ftp.loongnix.org/embedd/ls3a/kernel/linux-3.10.tar.gz>

1.3 工具链

龙芯官方提供的编译工具为 gcc,2k 和 3a 的工具链为同一套。Pmon 和内核采用不同的编译器。

Pmon 的编译器为:

<ftp://ftp.loongnix.org/embedd/ls3a/toolchain/gcc-4.4-gnu.tar.gz>

linux 内核的编译器为:

<ftp://ftp.loongnix.org/embedd/ls3a/toolchain/gcc-4.9.3-64-gnu.tar.gz>

1.4 操作系统

目前市场上支持龙芯的操作系统有很多,如中标、vxworks、道、锐化等等,这些大多数是商业系统,龙芯不提供系统和技术支持。如用户需要使用这些系统,需从这些商业公司获得。龙芯秉承做操作系统不卖操作系统的原则,提供免费的 loongnix (fedora21)。下载链接为: <http://ftp.loongnix.org/os/loongnix/1.0/liveinst/loongnix-20180930.iso>

loongnix 约三个月更新一次,以上链接会失效。如失效请到如下链接下载:

<http://www.loongnix.org/index.php/%E9%A6%96%E9%A1%B5> 点开此链接打开页面后,可以从页面的 loongnix 字样,进入到下载链接。

1.5 ejtag

Ejtag 作为龙芯处理器一款高效的在线调试工具。支持 1, 2, 3 号的所有处理器;支持 gdb remote 协议。即可以烧写 bios 到 flash 中,也可以通过软硬件配合高效的调试内核和外设。整套 ejtag 由软件部分和硬件部分组成。硬件部分需要付费获得。软件下载链接如下:

<ftp://ftp.loongnix.org/embedd/ls1b> 在此目录下 cygwin 字样的文件名为 windows 下使用软件, .tar.gz 字样的为 linux 下使用软件其中带有 mips64 字样的为龙芯本地使用,在 ejtag 软件 doc 目录下是使用手册,driver 为 windows 下的驱动。具体 ejtag 的使用在第五章中会有示例体现。

第二章 Pmon

2.1 pmon 相关资源

2.1.1 源码及工具链获取

2k :

<http://ftp.loongnix.org/embedd/ls2k/pmon-loongson3.tar.gz>

3a :

<http://ftp.loongnix.org/embedd/ls3a/bootloader/pmon-loongson3.tar.gz>

工具链:

<http://ftp.loongnix.org/embedd/ls3a/toolchain/gcc-4.4-gnu.tar.gz>

2.2 编译准备

2.2.1 安装 makedepand

```
sudo aptitude install xutils-dev
```

```
sudo cp makedepand /usr/bin
```

2.2.2 安装 pmoncfg

```
sudo apt install bison flex
```

```
cd PMONPATH/tools/pmoncfg
```

```
make pmoncfg
```

```
sudo cp pmoncfg /usr/bin
```

2.3 pmon 编译

(1) cd zloader.ls2k

(目录根据实际板卡进行选择, 例如 3a30007a 目录为 zloader.3a3000_7a)

(2) 执行编译脚本里的命令即可, 命令如下:

```
#!/bin/bash
```

```
export PATH=/opt/gcc-4.4-gnu/bin/:$PATH
```

```
make cfg all tgt=rom ARCH=mips CROSS_COMPILE=mipsel-linux- DEBUG=-g
```

(以 2k 为例, 如更改了配置文件 Targets/LS2K/conf/ls2k, 则在编译前要执行 make cfg, 使得更改生效, 如果普通编译没有更改配置, 则每次无需都执行 make cfg 命令)

PS: 执行 make dtb 可以将 dtb 和 gzrom.bin 结合生成 gzrom-dtb.bin, 此命令可以在上面编译完成后执行, dtb 为设备树。在启动内核过程中, 默认使用 Pmon 中的 dtb, 如果找不到 pmon 中的 dtb, 默认使用内核下的 dtb。

2.4 内存相关配置

内存的配置, 包含两部分一部分为 pmon 下使用的配置, 一部分为传参到内核使用的 DTS 内存相关部分。

1) pmon 下使用的相关配置

需要关注的文件及内容(以 2k 为例):

Targets/LS2K/conf/ls2k //ls2k 配置文件, 内有内存相关宏定义主要如下:

AUTO_DDR_CONFIG //自动探测内存的开关, 当板卡使用内存条时需要打开此选项, 颗粒需要关闭

DDR_S1 //关闭 AUTO_DDR_CONFIG 时需要根据具体的内存型号手动配置 s1 的值, 具体配置参考<<内存调试手册.pdf>> 4.2 节 s1 寄存器的设置

DDR_PARAM_018, DDR_PARAM_1e0, DDR_PARAM_1e8 //内存 18, 1e0, 1e8 寄存器设置

CONFIG_DDR_32BIT, CONFIG_DDR_16BIT //32/16 bit 内存配置

DDR_RESET_REVERT //如内存接了反向器需要打开此宏

CORE_FREQ, DDR_FREQ //主频, DDR 频率配置

Targets/LS2K/ls2k/start.S //内有开启内存调试相关宏定义,开启宏定义烧写后可以在 pmon 下动态调整及测试内存

```
#define DEBUG_DDR
```

```
#define DEBUG_DDR_PARAM
```

Targets/LS2K/ls2k/loongson3_clksetting.S //主频及内存频率配置文件

Targets/LS2K/ls2k/loongson_mc2_param.S //内存参数配置文件

其他板卡参考路径类似, 如 3a30007a 路径为 Targets/Bonito3a3000_7a/B 目录下相关文件, 详细内存调试及使用请参考文档<<龙芯内存调试手册.pdf>> 及本文档调试篇.

2) DTS 内存配置参数介绍

以 2k 为例, dts 文件位置:

Targets/LS2K/conf/ls2k.dts //设备树配置文件

内存信息

```
memory {  
    name = "memory";  
    device_type = "memory";  
    reg = <0 0x00200000 0 0x0c800000 // 200MiB at 2MiB  
          1 0x10000000 0 0x50000000>; // 1280MiB at 4352MiB  
};
```

reg 属性中, 地址被分为两段, 低地址段为:

0 0x00200000 0 0x0c800000, 其中 0 0x00200000 为内存起始地址, 0 0x0c800000 为内存大小, 此部分为保留内存, 客户不要修改

高地址段为:

1 0x10000000 1 0x50000000, 其中 1 0x10000000 为内存起始地址, 1 0x50000000 为内存大小(此大小为 64bit addr, 此部分需要根据客户板卡实际内存大小设置, 此大小为实际内存大小减去第一段的 200M 以及 gpu 使用的 512M 剩余的部分).

DTS 其它节点配置请参考 <<FDT 使用说明 V1.0.doc>> 文档

2.5 pmon 命令介绍

2.5.1 pmon 烧写

pmon 下通过网口烧写

ifaddr syn0 10.0.0.2

load -f 0xbfc00000 -r tftp://10.0.0.1/gzrom.bin

pmon 下通过 u 盘烧写

load -rf 0xbfc00000 (usb0,0)/gzrom.bin

2.5.2 内核启动相关命令

网络加载:

load tftp://server-ip/vmlinux

initrd tftp://server-ip/initrd.gz

g console=tty console=ttyS0,115200

u 盘启动:

load (usb0,0)/vmlinux

initrd (usb0,0)/initrd.gz

g console=tty console=ttyS0,115200

PS:以上 ip 及文件名字需要根据实际使用情况做相应的修改,下面对对应命令做出详细解释

命令: load

格式: load <elf-file>

解释: load 命令是 PMON 中一个很重要命令。作用是加载一个 elf 文件到内存中(这里只是存放到内存中,而没有烧写到 flash 中),加载过程是一个自动根据 elf 文件的信息处理 elf 文件重定向的等等操作的总体过程,所以这里不需要指定加载的内存地址,load 命令会自动完成。

如: load tftp://server-ip/vmlinux

如上命令,是从网络 tftp 服务器中加载内核到内存中。

如: `load /dev/mtd0b`

如上命令, 是从 `nandflash` 的第一个分区开始处中加载一个 `elf` 文件到内存中, 这里一般存放着内核, 这也是加载内核的另一种方法。

注: 这里的 `vmlinux` 是一个 `elf` 文件, 如果指定加载的文件不是 `elf` 文件, 将提示错误。

命令: `initrd`

格式: `initrd `

解释: 加载 `initrd image`

命令: `g`

格式: `g [-s][-b bpaddr][-e addr][--args]`

解释: `g` 命令是 `PMON` 中一个很重要的命令, 直接从指定内存地址处开始执行程序

如: `g -e addr`

如上命令, 直接从内存地址 `addr` 处开始执行程序, 正常操作前提是在 `addr` 开始处的内存中已经存放了一个可执行的程序, `addr` 是这个程序的入口地址, 如我们知道内存地址 `0x80010000` 是 `pmon` 代码段存放开始地址, 执行 `g -e 0x80010000` 将会从新执行 `pmon` 代码, 相当于了重启 `PMON` (这时不会从 `flash` 中拷贝执行 `pmon`)

`load tftp://server-ip/vmlinux`

`g console=ttyS0,115200 rdinit=/sbin/init`

如上命令和 `load` 命令配合使用, 在 `load` 完成后, 使用 `g` 命令自动从加载后的内核入口地址开始执行内核, 这是 `g` 后面在参数 “`console=ttyS0,115200 rdinit=/sbin/init`” 是内核启动的数。

2.5.3 跟启动相关的环境变量

跟启动相关的环境变量有 `al1`, `rd`, `append`, 分别对应内核, `initrd` 路径及启动参数, 例如

可以设置为:

`set al1 /dev/mtd0`

`set rd /dev/fs/yaffs2@mtd1/initrd.gz`

`set append console=tty console=ttyS0,115200`

命令: set

格式: set [name] [value]

解释: 查看 pmon 中的环境变量,设置环境变量 name 为 value

如: set (查看显示 pmon 中所有的环境变量);

set wk kkk (添加或者修改环境 wk 的值为 kkk)

2.5.4 pmon 烧写内核及文件系统到 nand 方法

烧写之前需要擦除, 命令如下:

mtdd_erase /dev/mtdd0r /0 代表分区, r 后缀代表擦除不跳过坏快

mtdd_erase /dev/mtdd1r

devcp tftp://server-ip/vmlinux /dev/mtdd0

devcp tftp://server-ip/roofs-yaffs2.img /dev/mtdd1y

set al1 /dev/mtdd0

set append console=ttyS0,115200 rdinit=/sbin/init root=/dev/mtddblock1 rw

rootfs=yaffs2

命令: devcp

格式: devcp <src-device> <desc-device>

解释: devcp 命令应该是 PMON 中比较重要的命令。

其实现是:打开 src-device 和 desc-device 两个设备,从 src-device 读取一定数量的字节数据,写到 desc-device 设备中去,完成后关闭这两个设备。

如:devcp tftp://server-ip/vmlinux /dev/mtdd0

如上命令, 从 tftp 服务器中接收读取内核 vmlinux,写到 nandflash 的第一个分区中,这是常用的烧写内核到 flash 中一个常用命令

如:devcp tftp://server-ip/roofs-yaffs2.img /dev/mtdd1y

如上命令, 从 tftp 服务器中接收读取根文件系统的 yaffs2 镜像文件 rootfs-yaffs2.img 烧写到 nandflash 的第二个分区中。这是常用的烧写 yaffs2 文件系统到 nandflash 中的一个常用命令。

命令: mtdparts

格式: mtdparts

解释: 显示 nandflash 的现在分区信息。

命令: mtd_erase

格式: mtd_erase <mtd-device>

解释: 擦除 nandflash 的一个分区

(擦除 nandflash 的第一个分区,跳过已经是坏块的地方,不会尝试擦除已经是坏块的地方);

mtd_erase /dev/mtd0r

(擦除 nandflash 的第一个分区,这时会尝试擦除所有的块,即使是坏块也会尝试执行擦除操作)。

2.5.5 pmon 下查看及修改 dtb

Pmon 命令行支持, 查看、删除、重新烧写 dtb。

Pmon 下查看 dtb 命令:

Print_dtb / 打印出 dtb 所有内容

Print_dtb /soc/dc@0x400c0000 打印出 dc 的所有内容

Pmon 下删除 dtb

rm_dtb_node / 删除整个 dtb

rm_dtb_node /soc/hda@0x400d00000 删除 had 内容

pmon 下重新烧写 dtb

dtb 会把内存大小信息传递给内核, 如果这部分内容传递错误会造成不可预知错误。如果重新烧写内部部分内容, 需要通过 print_dtb 命令把内存部分内容读出, 更新到新的 dts 文件中 (3 号处理器不传递内存信息)。

在 pmon 编译目录下 (如 2.3 节) 执行 make dtb 会生成 LS2K.dtb 文件。此文件可单独烧写。烧写命令和烧写 pmon 一致 (如 2.5.1 节)。

2.5.6 其他命令简介

命令: d1

格式: d1 <addr> <num>

解释: display 简写

按无符号字符型(unsigned char)访问显示 num 个从地址 addr 开始的内容值。

如: d1 0x85000000 0x10

显示从内存地址 0x85000000 地址处的 0x10 个字节(8 位),0x85000000-0x8500000f

命令: d2

格式: d2 <addr> <num>

解释: display 简写

按 (unsigned short) 访问显示 num 个从地址 addr 开始的内容值。

如: d2 0x85000000 0x10

显示从内存地址 0x85000000 地址处的 0x10 个 unsigned short 类型(16 位)的数值,0x85000000-0x8500001e

命令: d4

格式: d4 <addr> <num>

解释: display 简写

按(unsigned long)访问显示 num 个从地址 addr 开始的内容值。

如: d4 0x85000000 0x10

显示从内存地址 0x85000000 地址处的 0x10 个 unsigned long 类型(32 位)的数值,0x85000000-0x8500003c

命令: d8

格式: d8 <addr> <num>

解释: display 简写

按 (unsigned long long) 访问显示 num 个从地址 addr 开始的内容值。

如: d8 0x85000000 0x10

显示从内存地址 0x85000000 地址处的 0x10 个 unsigned longlong(64 位)类型的数值,

0x85000000-0x85000078

命令: m1

格式: m1 <addr> <value>

解释: modify 简写

按无符号字符型(unsigned char)访问修改地址 addr 的内容值。

如: m1 0x85000000 0x10

修改内存地址 0x85000000 地址处的一个 unsigned char 型值相当于:*(unsigned

char*)addr = value;

命令: m2

格式: m2 <addr> <value>

解释: modify 简写

按无符号短整型(unsigned short)访问修改地址 addr 的内容值。

如: m2 0x85000000 0x10

修改内存地址 0x85000000 地址处的一个 unsigned short 型值相当于:*(unsigned

short*)addr = value;

命令: m4

格式: m4 <addr> <value>

解释: modify 简写

按无符号长整型(unsigned long)访问修改地址 addr 的内容值。

如: m4 0x85000000 0x10

修改内存地址 0x85000000 地址处的一个 unsigned long 型值相当于:*(unsigned

long*)addr = value;

命令: m8

格式: m8 <addr> <value>

解释: modify 简写

按无符号字符型(unsigned long long)访问修改地址 addr 的内容值。

如: m8 0x85000000 0x10

修改内存地址 0x85000000 地址处的一个 unsigned long 型值相当于:*(unsigned long long*)addr = value;

命令: devls

格式: devls

解释: 查看 pmon 的部分设备, 这里显示出来的设备不是全部的设备。

一般会有网卡, USB,CF 卡,sd 卡,硬盘等等,主要用查看设备的名字,以供其他操作的使用。

命令: ifaddr

格式: ifaddr <interface> <ipaddr>

解释: 设置网卡的 IP 地址,如:ifaddr syn0 192.168.0.1 (syn0 是网卡的名称,可以使用 devls 命令查看获得,这里不用设置子网掩码,pmon 会自动设置子网掩码为 255.255.255.0)

命令: ping

格式: ping <ipaddr>

解释: 简单检测,如:ifconfig syn0 (syn0 是网卡的名称,可以使用 devls 命令查看获得)

命令: unset

格式: unset <name>

解释: 去掉名字为 name 的环境变量。关于环境变量的讲解参看附录 PMON 环境变量

第三章 Kernel

3.1 Kernel 获取与编译

3.1.1 建立交叉编译环境

1) 交叉编译工具的获取

工具链的获取可以通过访问 <http://ftp.loongnix.org/embed/XXX/toolchain/> 来获取，其中 XXX 可以为 ls1a、ls1b、ls1c、ls2h、ls2k、ls3a 等，需要用户根据需要自行修改访问。

用户可以通过先访问网址 <http://ftp.loongnix.org/embed/>，然后根据需要一步步访问自己需要的目录。

正常情况下编译工具位于对应 cpu 的 toolchain 目录，例如：
<http://ftp.loongnix.org/embedd/ls2k/toolchain/>

2) 内核与交叉编译版本

龙芯目前主推的内核版本有 linux-2.6.32 和 linux-3.10 两个版本，其中 linux-2.6.32 主要用于 1 号、CPU、龙芯 2H 和龙芯 3A，linux-3.10 主要用于龙芯 3A2000 以上的 CPU 版本。

龙芯 1 号系列 CPU 所用交叉编译工具的版本为 gcc-4.3-ls232.tar.gz。

龙芯 2H 和龙芯 3A CPU 所用交叉编译工具的版本为 gcc-4.4-gnu.tar.gz。

龙芯 2K 和龙芯 3A2000 以上版本 CPU 所用交叉编译工具的版本为 gcc-4.9.3-64-gnu.tar.gz。

3) 交叉编译工具的安装

网站下载的交叉编译工具为 tar 格式，一般将其解压到/opt/目录下，

命令：`tar xzvf gcc-4.3-ls232.tar.gz -C /`

为交叉编译工具的路径添加环境变量

命令：`export PATH=/opt/gcc-4.3-ls232/bin:$PATH` （注，此处只是例子）

输入命令：`mipsel-linux-gcc -v` 查看交叉编译工具版本信息，来验证交叉编译工具是否正常安装(提示版本信息即认为交叉编译工具已经正常安装)。

3.1.2 内核源码的获取

1) 用户可以通过先访问网址 <http://ftp.loongnix.org/embed/>，然后根据需要一步步访问自己需要的目录。

正常情况下编译工具位于对应 cpu 的 kernel 目录，例如：

<http://ftp.loongnix.org/embedd/ls3a/kernel/>

特殊情况下，也可能位于对应 CPU 所在的当前目录。例如：

<http://ftp.loongnix.org/embedd/ls2k/>

注：龙芯 2K 及龙芯 3A2000 以上的 CPU 版本，所用内核为 linux-3.10，其他系列 CPU 使用 linux-2.6.32 版本的内核。

3.2 内核编译所需要的工具

1)内核的默认配置

网站下载的交叉编译工具为 tar 格式，需要先对其解压缩之后进入源码所在目录，内核的默认配置位于 arch/mips/configs/，用户需要根据自己的 CPU 来选择相应的配置，如 2H CPU 为 loongson2h_defconfig，2K CPU 为 loongson2k1000_defconfig，3A 系列 CPU 为 loongson3_defconfig。

2) 确认安装图形化配置 “make menuconfig” 依赖的工具 Ncurses (如果安装过跳过此步骤)

命令: apt-get install libncurses5-dev

2)解压缩并进入 linux 源代码树根目录。

确认安装图形化配置 “make menuconfig” 依赖的工具 Ncurses (如果安装过跳过此步骤)

命令: #apt-get install libncurses5-dev

3) 图形化配置(调整终端窗口到合适的大小，不要太小就可以):

make menuconfig ARCH=mips (注意 ARCH 为大写)

这里重点说明下需要我们关心的内核的配置:

1、 Kernel hacking -->

[*] Kernel debugging

[*] Compile the kernel with debug info

此配置，可以编译出带符号表的内核二进制，反汇编以后有助于内核的调试工作。

2、 General setup -->

[*] Initial RAM filesystem and RAM disk (initramfs/initrd) support

(填写 ramdisk 所在路径) Initramfs source file(s)

此配置可以将 ramdisk 编译进内核，如果外部存储设备已经存在内核可以使用的文件系统，此配置可以去掉。

3、 General setup -->

[*] Support initial ramdisks compressed using gzip

[*] Support initial ramdisks compressed using bzip2

[*] Support initial ramdisks compressed using LZMA

此配置在需要编译压缩内核时使用，可根据需要选择压缩格式。

3.3 内核编译命令

如果主机已经按照步骤添加了交叉编译工具的环境变量，并且拷贝了对应 cpu 所使用的 config 文件做.config，然后执行 make menuconfig ARCH=mips 做了相关配置，那编译内核只需要如下操作:

命令: make vmlinux CROSS_COMPILE=mipsel-linux- ARCH=mips

如果要编译出压缩的内核，并已经做了相关配置，执行如下:

命令: make vmlinux CROSS_COMPILE=mipsel-linux- ARCH=mips

注: CROSS_COMPILE 是根据交叉编译工具的命名来填写的,如果交叉编译工具所在目录的 bin 目录下为 mips64el-linux-命名,则 CROSS_COMPILE=mips64el-linux-,当然如果交叉编译工具所在目录没有被添加到环境变量,则 CROSS_COMPILE 需要写出交叉编译工具所在的绝对路径。

3.4 编译 busybox

龙芯给客户提供的 ramdisk 文件本身是由 busybox 编译出来的,如果里面的动态库版本不匹配,客户可以自己编译,方法如下:

1. 下载 busybox

[wget http://busybox.net/downloads/busybox-1.18.5.tar.bz2](http://busybox.net/downloads/busybox-1.18.5.tar.bz2)

tar jxvf busybox-1.18.5.tar.bz2

cd busybox-1.18.5

2. 配置 busybox

make menuconfig

Busybox Settings ---->

Build Options ---->

[*] Build BusyBox as a static binary (no shared libs)

[*] Build with Large File Support (for accessing files > 2 GB)

(mipsel-linux-) Cross Compiler prefix

如果没有将交叉编译工具添加到环境变量 mipsel-linux-前需要添加绝对路径。

3. 编译安装 busbox

make

make install

cd _install

mkdir dev

cp -a /dev/null dev

cp -a /dev/console dev

cp -a /dev/tty* dev

mkdir etc

cp ../examples/inittab etc

3.5 buildroot 制作及交叉开发 qt 程序

一般文件系统都要包含很多第三方软件,比如 busybox, tftp, apache, PHP, DNS, qt 等等,为了避免繁琐的移植工作,buildroot 应运而生,通过 menuconfig 来配置我们需要的功能,不需要的去掉,再执行 make 编译,buildroot 就会自动从指定的服务器上下载源码包,自动编译,自动搭建成我们所需要的嵌入式文件系统。

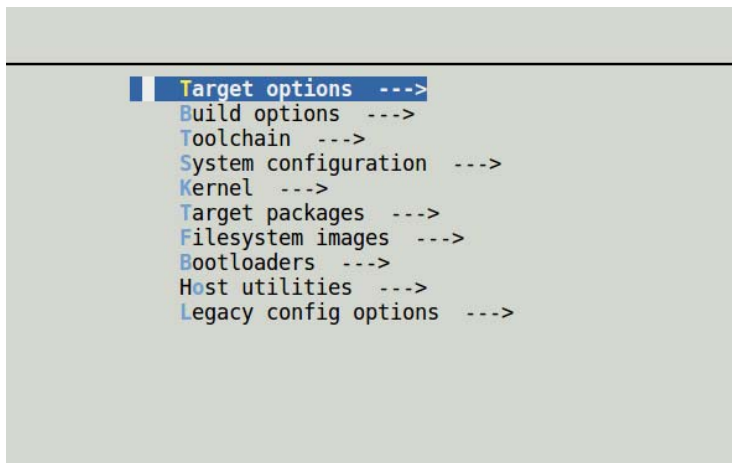
1. 从 <http://ftp.loongnix.org/embedded/ls2h/buildroot/> 下下载 buildroot.tar.gz

2. sudo tar -xvf buildroot.tar.gz 解压

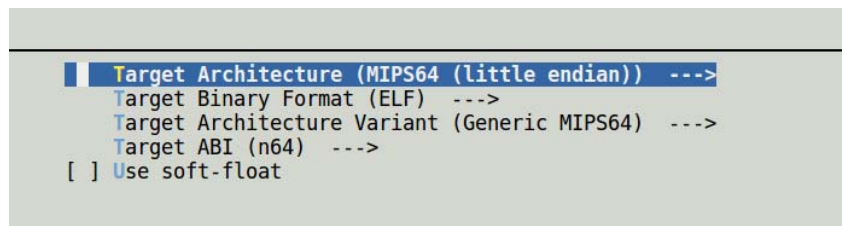
3. 在 buildroot 的顶层目录下写个 cmd.sh 编译的脚本文件。内容如下：

```
1 #!/bin/bash
2
3 export PATH=/opt/gcc-4.9.3-64-gnu/bin:$PATH
4 make arch=mips CROSS_COMPILE=mips64el-linux- -j4 "$@"
```

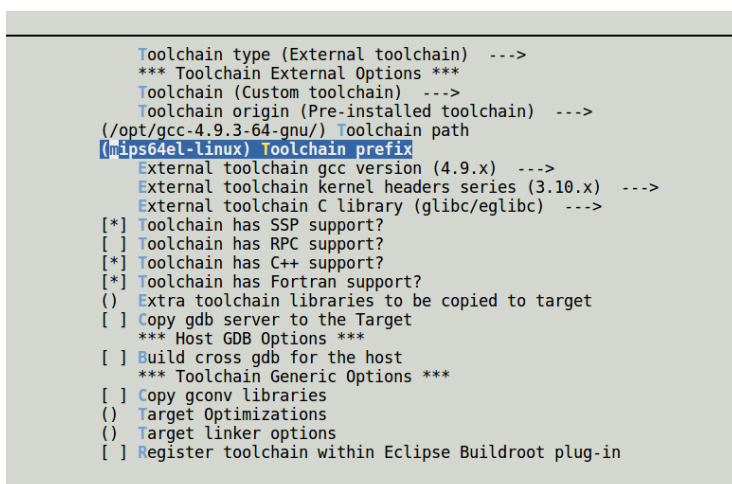
4. chmod 777 cmd.sh
5. 从顶层目录下的 configs 目录下的 ls2k_docker-gcc-4.9.3.config 拷贝到顶层目录下的.config
6. 执行./cmd.sh menuconfig 开始配置 buildroot:



这就是它的配置界面，进入 target options，配置 CPU 参数：



进入 Toolchain，将 Toolchaintype 配置为 External toolchain，然后在 Toolchain 中选择交叉编译工具的版本，如 Custom toolchain，在 Toolchain origin 中选择 Pre-installed toolchain，后面编译时，buildroot 将会根据 Toolchain path 找到我们的交叉编译工具。退回上一界面；



进入 system configuration, system banner 是欢迎语, root password 是登录后的密码为空的没有密码, 默认的用户名是 root,

```

Root FS skeleton (default target skeleton) --->
(buildroot) System hostname
(Welcome to Buildroot) System banner
Passwords encoding (md5) --->
Init system (BusyBox) --->
/dev management (Dynamic using devtmpfs only) --->
(system/device_table.txt) Path to the permission tables
[ ] support extended attributes in device tables
[ ] Use symlinks to /usr for /bin, /sbin and /lib
[*] Enable root login with password
() Root password
/bin/sh (busybox' default shell) --->
[*] Run a getty (login prompt) after boot --->
[*] remount root filesystem read-write during boot
(eth0) Network interface to configure through DHCP
[*] Purge unwanted locales
(C en_US) Locales to keep
() Generate locale data
[ ] Enable Native Language Support (NLS)
[ ] Install timezone info
() Path to the users tables
() Root filesystem overlay directories
() Custom scripts to run before creating filesystem images
() Custom scripts to run inside the fakeroot environment
() Custom scripts to run after creating filesystem images

```

选中 Run a getty(after prompt)after boot,进入, 配置打印串口和波特率:

```

-- Run a getty (login prompt) after boot
(ttyS0) TTY port
Baudrate (115200) --->
(vt100) TERM environment variable
() other options to pass to getty

```

返回顶层的配置界面进入 Filesystem images, 选中, cpio, ext2/3/4 和 tar root filesystem。

```

[ ] axfs root filesystem
[ ] btrfs root filesystem
[ ] cloop root filesystem for the target device
[*] cpio the root filesystem (for use as an initial RAM filesystem)
    Compression method (gzip) --->
[ ] Create U-Boot image of the root filesystem
[ ] cramfs root filesystem
[*] ext2/3/4 root filesystem
    ext2/3/4 variant (ext2 (rev1)) --->
    () filesystem label (NEW)
    (60M) exact size (NEW)
    (0) exact number of inodes (leave at 0 for auto calculation) (NEW)
    (5) reserved blocks percentage (NEW)
    (-O ^64bit) additional mke2fs options (NEW)
    Compression method (no compression) --->
[ ] f2fs root filesystem (NEW)
[ ] initial RAM filesystem linked into linux kernel
[ ] jffs2 root filesystem
[ ] romfs root filesystem
[ ] squashfs root filesystem
[*] tar the root filesystem
    Compression method (no compression) --->
    () other random options to pass to tar (NEW)
[ ] ubi image containing an ubifs root filesystem
[ ] ubifs root filesystem
[ ] yaffs2 root filesystem

```

返回上一级。

最后进入 **target packages** 目录下，有一大堆第三方的开源工具软件，基本上嵌入式上可能会用到得或用不到的，在这里都用，如果你想要把这个工具软件包含进固件里面，很简单，只要在这个软件的名称前面打个*星号，表示要编译这个软件到固件里面。

```

-* BusyBox
(busybox.config) BusyBox configuration file to use?
() Additional BusyBox configuration fragment files
[*] Show packages that are also provided by busybox
[ ] Enable SELinux support
[ ] Individual binaries
[ ] Install the watchdog daemon startup script
Audio and video applications --->
Compressors and decompressors --->
Debugging, profiling and benchmark --->
Development tools --->
Filesystem and flash utilities --->
Fonts, cursors, icons, sounds and themes --->
Games --->
Graphic libraries and applications (graphic/text) --->
Hardware handling --->
Interpreter languages and scripting --->
Libraries --->
Mail --->
Miscellaneous --->
Networking applications --->
Package managers --->
Real-Time ----
Security --->
Shell and utilities --->
System tools --->
Text editors and viewers --->

```

这里按照 **ls2k_docker-gcc-4.9.3.config** 的默认配置不动，只把 **qt** 选择上，进入到 **Graphic libraries and applications**,把 **QT** 选上，保存退出。

```

*** Graphic applications ***
[ ] webcam
[ ] ghostscript
*** glmark2 needs an OpenGL or an OpenGL ES and EGL backend provided by mesa3d ***
[ ] gnuplot
[ ] head
[ ] libva-utils
[ ] mngquant
[ ] rrdtool
[ ] tesseract-ocr ----
*** Graphic libraries ***
[ ] cglut
[ ] directfb
[ ] fbdump (Framebuffer Capture Tool)
[ ] fgrab
[ ] fbset
[ ] fb-test-app
[ ] fbterm
[ ] fbv
[ ] freerdp
[ ] imagemagick
[ ] linux-fusion communication layer for DirectFB multi
[ ] mesa3d ----
[ ] ocra
[ ] splash
[ ] DL
[ ] dl2
*** Other GUIs ***
[*] qt (obsolete) --->
*** QT libraries and helper libraries ***
[ ] extserialport (NEW)
[ ] json (NEW)
[ ] qtui (NEW)
[ ] quazip (NEW)
[ ] qwt (NEW)
*** tekui needs a Lua interpreter and a toolchain w/ threads, dynamic library ***
*** weston needs udev and a toolchain w/ locale, threads, dynamic library, headers >= 3.0 ***

```

7. 最后开始编译 **buildroot**。在 **buildroot** 的顶层目录下执行：**./cmd.sh**（一定要是在超级用户的权限下，并且保证能访问互联网）。

然后就开始了漫长的编译过程。。。

编译的过程中出现了如下的错误：

```

超级块的备份存储于下列块：
8193, 24577, 40961, 57345

正在分配组表： 完成
正在写入inode表： 完成
将文件复制到设备： _populate_fs: 无法为ext2文件系统分配块 写入文件“dockerd”时
mkfs.ext2: 无法为ext2文件系统分配块 于填充文件系统时
*** Maybe you need to increase the filesystem size (BR2_TARGET_ROOTFS_EXT2_SIZE)
fs/ext2/ext2.mk:46: recipe for target '/home/zhangyan/download/buildroot/output/images/rootfs.ext2' failed
make: *** [/home/zhangyan/download/buildroot/output/images/rootfs.ext2] Error 1

```


解决方法：在 menuconfig 中 filesystem images-->ext2/3/4 关掉。

8. 最后在 output/images/下就生成了我们要的文件系统镜像。如下图所示：

```
root@zhangyan-G50-80:/home/zhangyan/download/buildroot/output/images# ls
rootfs.cpio rootfs.cpio.gz rootfs.ext2 rootfs.tar vmlinux
root@zhangyan-G50-80:/home/zhangyan/download/buildroot/output/images# ll rootfs.cpio.gz
```

9.实验一下看能不能跑 qt 程序，把 rootfs.cpio 编译进内核，然后跑一个 qt 的 demo 程序。

1.执行 qt 的 demo 程序，打印如下错误(要带上-qws 参数)：

```
./yocto_for_2k: error while loading shared libraries: libQt5Widgets.so.5: cannot open shared object file: No such file or directory
```

原因：在 menuconfig 中没有把 qt 的 gui 和 qt5widget 选上。按如下图所示选上 gui module 和 widgets module。

```
--- Qt5
    Qt5 version (Latest (5.11)) --->
    *- qt5base
    ( ) Custom configuration options
    ( ) Config file
    [ ] Compile and install examples (with code)
    [ ] concurrent module
    [ ] MySQL Plugin
    [ ] PostgreSQL Plugin
    [ ] SQLite 3 support (No sqlite support) --->
    [*] gui module
    [*] widgets module
    *** OpenGL support needs an OpenGL-capable backend ***
    *- linuxfb support
    *** directfb backend available if directfb is enabled ***
    *** X.org XCB backend available if X.org is enabled ***
    *** eglfs backend available if OpenGL and EGL are enabled ***
    ( ) Default graphical platform
    [*] fontconfig support
    [ ] harfbuzz support
    [ ] GIF support
    [ ] JPEG support
    [ ] PNG support
    [ ]DBus module
```

2. 出现如下问题：

```
# ./qt_demo loongson -qws
QFontDatabase: Cannot find font directory /opt/Qt4.8mips/lib/fonts - is Qt installed correctly?
Aborted
```

解决办法：没有字体库。在 menuconfig 中 target packages

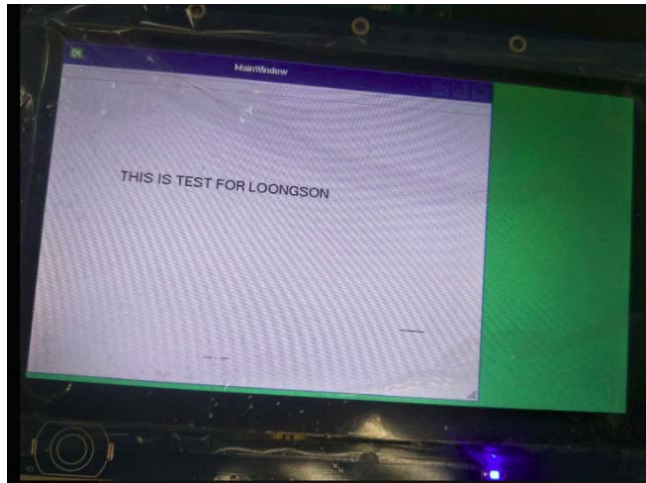
--->Graphic....-->Qt5--->fontconfig support 选上。如下图所示：

```
--- Qt5
    Qt5 version (Latest (5.11)) --->
    *- qt5base
    ( ) Custom configuration options
    ( ) Config file
    [ ] Compile and install examples (with code)
    [ ] concurrent module
    [ ] MySQL Plugin
    [ ] PostgreSQL Plugin
    [ ] SQLite 3 support (No sqlite support) --->
    [*] gui module
    [*] widgets module
    *** OpenGL support needs an OpenGL-capable backend ***
    *- linuxfb support
    *** directfb backend available if directfb is enabled ***
    *** X.org XCB backend available if X.org is enabled ***
    *** eglfs backend available if OpenGL and EGL are enabled ***
    ( ) Default graphical platform
    [*] fontconfig support
    [ ] harfbuzz support
    [ ] GIF support
    [ ] JPEG support
    [ ] PNG support
    [ ]DBus module
    [ ] enable ICU support
    [ ] enable Tslib support
    [ ] qt5charts
```

1.mkdir /opt/Qt4.8mips/lib -p

2.cp /usr/lib/fonts /opt/Qt4.8mips/lib

运行成功如下图所示：



10.遇到的问题

1.将 rootfs.cpio 编译进内核后，在 pmon 下 load 的时候 load 一半直接卡死，原因，rootfs.cpio 太大，把 rootfs.cpio.gz 编译进内核。

2.内核起来后一直打印如下内容：

```
[ 17.758000] libphy: stmmac-0:00 - Link is Up - 1000/Full
udhcpc: sending discover
udhcpc: no lease, failing
FAIL
Starting telnetd: OK
can't open /dev/ttyS0: No such file or directory
can't open /dev/ttyS0: No such file or directory
can't open /dev/ttyS0: No such file or directory
can't open /dev/ttyS0: No such file or directory
can't open /dev/ttyS0: No such file or directory
can't open /dev/ttyS0: No such file or directory
can't open /dev/ttyS0: No such file or directory
```

原因：在 dev 目录下没有生成该设备节点

解决办法：在 menuconfig 中 system configuration-->/dev management 选择(Dynamic using devtmpfs + mdev)

--->path to the permission tables 选择(system/device_table_dev.txt)

最后再重新编译 buildroot。

3.6 ssd 下内核的替换

1.在内核的顶层目录下执行./mymake menuconfig

选择如下选项：


```
Machine selection --->
Endianness selection (Little endian) --->
CPU selection --->
Kernel type --->
General setup --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
Bus options (PCI, PCMCIA, EISA, ISA, TC) --->
Executable file formats --->
Power management options --->
CPU Power Management --->
[*] Networking support --->
Device Drivers --->
Firmware Drivers --->
File systems --->
Kernel hacking --->
Security options --->
*- Cryptographic API --->
↓(+)
```

<Select> < Exit > < Help > < Save > < Load >

然后将 ramdisk 的路径添加到如下的位置:

```
^(~)
[ ] Checkpoint/restore support
[*] Namespaces support --->
[*] Automatic process group scheduling
[*] Enable deprecated sysfs features to support old userspace tools
[ ] Enable deprecated sysfs features by default
[*] Kernel->user space relay support (formerly relayfs)
[*] Initial RAM filesystem and RAM disk (initramfs/initrd) support
/home/zhangyan/ramdisk.cpio) Initramfs source file(s)
(0) User ID to map to 0 (user root)
(0) Group ID to map to 0 (group root)
[*] Support initial ramdisk/ramfs compressed using gzip
[*] Support initial ramdisk/ramfs compressed using bzip2
[*] Support initial ramdisk/ramfs compressed using LZMA
[*] Support initial ramdisk/ramfs compressed using XZ
[*] Support initial ramdisk/ramfs compressed using LZ0
[*] Support initial ramdisk/ramfs compressed using LZ4
[ ] Built-in initramfs compression mode
Compiler optimization level (Optimize for performance) --->
↓(+)
```

<Select> < Exit > < Help > < Save > < Load >

然后保存退出, 执行 ./mymake vmlinuz (开始编译带文件系统的内核)。

```
cp vmlinuz vmlinuz-fs
```

同样把不带 ramdisk 也编译一遍, 将 vmlinuz 拷贝到 U 盘中, 用于替换的内核。

2. 然后按 c 进入到 pmon 命令行下, load 刚编译的内核:

```
load /dev/fs/ext2@usb0/vmlinuz-fs (从 U 盘 load)
```

```
load tftp://192.168.1.249/vmlinuz-fs (从 tftp 服务器 load)
```

3. 在内核命令行下插入 U 盘, 将要替换的内核拷贝出来:

```
执行命令: mount /dev/sdb /mnt (假如 U 盘识别的是 sdb)
```

```
cp /mnt/vmlinuz /
```

```
umount /mnt
```

4. 然后开始挂载硬盘

```
mount /dev/sda1 /mnt
```

查看/mnt/boot/boot.cfg 将/mnt 或者/mnt/boot/下的 vmlinuz 替换为我们的 vmlinuz(这里最好将之前的 vmlinuz 保存一下)。

```
sync
```

```
umount /mnt
```

重启。

第四章 板卡调试

龙芯众多客户会自行开发板卡，板卡能否用的起来、用的好关于项目的成败。本章内容旨在指导龙芯用户调试自设计板卡。

板卡调试可总结为两个阶段，第一阶段为处理器能够正确取值；第二阶段为处理器正常工作后软件人员介入开始调试。故本章内容分为两部分一硬件工程师调试硬件到处理器取址，第二部分为软硬件工程师配合调试。Ejtag 作为龙芯调试板卡的重要工具，本章会结合调试龙芯派和 3a3000+7a 开发板，列举出一块焊接好板卡的整个调试过程，同时会介绍一些 ejtag 调试板卡过程中的一些常用命令（ejtag 命令的详细列表请参考，ejtag 软件的 doc 目录下的两个 pdf 文档）。

4.1 2K 龙芯派硬件调试

用户在拿到板卡后，上电前应做好如下两个步骤所述。

1、目检

在拿到板卡后，首先进行目检，确保板子上没有明显的锡渣残留，各个器件管脚之间没有连锡，没有不良的焊接等现象。

2、阻抗测量

在目检完成后，上电前需要根据原理图的设计，测量各个电源的对地阻抗。在测量阻抗时应用万用表（或者六位半）的欧姆档进行测量，不要用万用表的二极管档测量。

以 2K 龙芯派板卡为例：该板卡上的电源共有：VDDRSM（1.1V 域电压），3V3_SB（3.3V 域电压），P1V1（1.1V 核电压），P1V35_LS（2K1000 DDR 1.35V 电压），P1V35_DDR（DDR 颗粒电压）。各个电源的对地阻抗如表 1 所示：

表 1 电源对地阻抗

电源	阻抗	备注
P12V	14.63K	
5V_RSM	23.98K	
VDD_RSMIN(1.1V)	2.929K	
P3V3SB	5.46K	
P5V	12.96K	
P1V1(设计为 1.2V)	68.1	
P1V35_LS(设计为 1.353V)	1.444K	
P1V35_DDR(设计为 1.353V)	0.958K	

P3V3	128.4	
VTT	4.91K	

其中，2K1000 的 1.1V 核电压的对地阻抗约为几十欧姆。只要是几块板卡的电源对地阻抗相差不是太大，即可认为阻抗是正确的。

注：测量阻抗时，应该用红表笔接电源的正极，黑表笔接电源的负极。若是反接，测量到的结果与表笔正接的结果会不一样。

阻抗全部正常之后，可对板卡进行上电。上电后的初始测量有以下几部分。

1、电压测量

按照原理图的设计对各个电压源进行测量，所得到的结果应该在设计的允许范围之内。若有不正常的电压，应找出原因后再进行下边的工作。

2、复位的测量

2K1000 的 CPU 对复位也有要求，复位主要测量三部分：

A、系统复位；

B、RTC_RSMRSTn，该信号是复位 Resume 域逻辑，需在 resume 域上电稳定后保持一段时间有效（推荐>5ms）

C、RTCRSTN，RTC 域复位，建议在 RTC 电源稳定 10ms 后再解除复位。

3、时钟测量

电源全部正常后，用示波器测量板卡上的时钟，2K1000 芯片必不可少的时钟为 100MHz 的系统时钟和 32.768KHz 的 RTC 域时钟。任何偏离该时钟太多的时钟信号都被认为是不正常的，可能会造成板卡的启动异常。

4、取址测量

2K1000 的启动方式有 LIO 启动、SPI 启动、SDIO 启动、NAND 启动，用户根据自己的需求选择不同的启动方式。

根据不同的启动方式来测量相应的取址信号，下面仅以 2K 龙芯派为例进行说明。

2K 龙芯派的启动方式为 SPI 启动。需要测量的信号有 SPI 的 CS（片选）、MOSI 和时钟。测量的信号应符合相关的 SPI 协议中的规定。

若没有测量到有取址信号发出，可能的原因有：

- 1、SPI 的信号也连接了其他的设备，信号电平被其他设备钳位住了；
- 2、板卡的上电时序不对；板卡的复位关系不能满足 2K1000 芯片的需求；
- 3、2K1000 芯片虚焊；2K1000 芯片损坏。

上述测量项测试完毕后没有问题的话，可以烧写 PMON 程序了

4.2 3A3000+7a 硬件调试部分

1、电源对低阻抗测量

7A 开发板的电源较为复杂，总共有 P5VSB、P3V3SB、P1V1SB、P2V5SB、P1VSB、P12V、P5V、P3V3、P3V3_3A、P3V3_7A、P1V15_CORE_3A、P1V1_VDD_MEM_3A、P1V8_3A、P1V5_VDDQ_3A、P1V2_HT_3A、P1V1_PLL_3A、P1V5_DUAL、P0V75_VTT_3A、P1V1_CORE_7A、P2V5_7A（P2V5_3A）、P1V8_7A、P1V2_HT_7A、P1V1_PLL_7A、P5V_DUAL、P5V_DUAL_USB、P0V75_VTT_7A、P1V5_VDDQ_7A、P1V1SB_7A、P1V5_VDDQ_7A_CORE、P3V3SB_7A 等 30 路电源。

板卡在上电之前这 30 路电源的对地阻抗都要测量一遍，找到每一路电源的退耦电容测量退耦电容两端阻抗并记录，若出现阻抗为 0 或者阻抗很小（一两欧大小）那么需要注意，这一路电源可能存在短路。注意 3A 核电阻抗本身就很小只有几欧测量时不要发生误解。

2、电源电压测量

阻抗测量完成并没有发现短路的情况下就可以上电了，上电后的第一步先量取这 30 路电源的电压值并记录，对比所记录的电压值，与相对应的标称值误差不要大于±5%。

3、时钟测量

3A+7A 板卡时钟设计同样较为复杂，板卡上有 3A 系统时钟 25MHZ；MEM 时钟 33MHZ；3A LPC 时钟 33MHZ；7A LPC 时钟 33MHZ；3A PCI 时钟 33MHZ；7A 系统时钟 100MHZ；3A HT LVDS 电平差分时钟 200MHZ；7A HT LVDS 电平差分时钟 200MHZ；各路 HCSL 电平的 PCIE 参考时钟 100MHZ。

4、取址信号测量

3A 的启动方式选择由 PCI_CONFIG0 来决定，PCI_CONFIG0 为高从 SPI 启动，为低从 LPC 启动。

从 SPI 启动时，量取 SPI_CS；SPI_CLK；SPI_SDO，SPI_SDI 也需要在烧录 PMON 后确认信号是否有数据读出来且信号质量正常。

从 LPC 启动时，量取 LPC 时钟；LPC_AD[3:0]；LPC_LFRAME#是否存在信号，且信号质量是否正常。

4.3 软件部分

在硬件工程师确认硬件已经工作后，软件工程师需要配合硬件工程师，确认处理器是否已经正常工作后，整个板卡的调试工作软件部分开始介入。

4.3.1 ejtag 的使用

Ejtag 在整个板卡的调试过程，运用的合理会起到事半功倍的效果。

第一步，我们需要检查 ejtag 到处理器链接是否正确（以下 ejtag 使用以 linux 下为例）：

（1）运行 ejtag 软件

在板卡开机前以 root 权限运行 ejtag 软件 `./ejtag_debug_usb -t`

进入 ejtag 命令行模式，ejtag 支持龙芯所有的处理器，如果现在调试的是龙芯派或者其它 2K 处理器板卡，需要加载对应处理器的配置。执行如下命令：

```
source configs/config.ls2k
```

如果是 3a3000 处理器的相关板卡，需要执行如下命令：

```
source configs/config.ls3a3000
```

所有支持的处理器都可以在 configs 目录下找到相应的 config 文件。

(2) 检查 ejtag 到处理器是否连通

Ejtag 通过调试主机 USB 口连接到龙芯主板的示意图 4.1. 在调试开始前，调试人员必须确认主机到主板是否连通。在检查连通性的同时可以确认处理器是否已经正常取址。确认过程分两步，如图 4.1 的 A B 两步。第一步通过 usbver 命令可以检查调试主机是否和 ejtag 连通；第二步可以确认调试器到主板是否连通，同时可以确认龙芯处理器是否正常工作。



图 4.1 ejtag 到主板连接示意图

具体操作如下：

加载完相应处理器的配置后，板卡上电开机，在 ejtag 命令行执行如下命令：

usbver

返回为 20150915 2013 如果不是这个日期则调试机到 ejtag 板，连通性有问题

Jtagregs d8 0 10

可能得结果得到的结果如下：

```

cpu0 -jtagregs d8 0 10
#jtagregs d8 0 10
00000000: 0000000000000000 000000005a5a5a5a .....ZZZZ....
00000002: 0000000000000000 0000000060400001 .....@`....
00000004: 0000000000000000 0000000000000000 .....
00000006: 0000000000000000 0000000000000000 .....
00000008: 0000000000000000 0000000000000000 .....
cpu0 -
  
```

图 4.2 ejtag 寄存器正确值

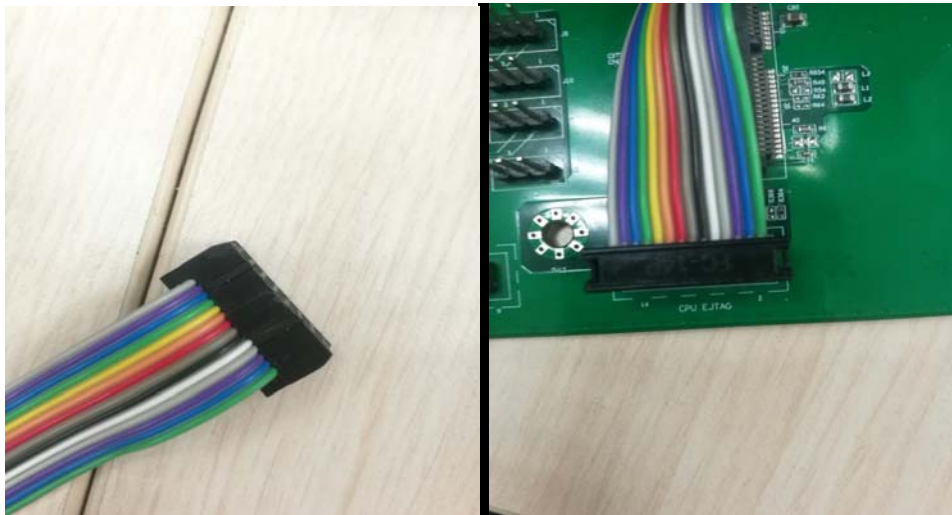
```

cpu0 -jtagregs d8 0 10
#jtagregs d8 0 10
00000000: 00000000ffffffff 00000000ffffffff .....
00000002: 00000000ffffffff 00000000ffffffff .....
00000004: 00000000ffffffff 00000000ffffffff .....
00000006: 00000000ffffffff 00000000ffffffff .....
00000008: 0000ffffffffffff ffffffffffffffff .....
  
```

图 4.3 ejtag 寄存器错误值

在图 4.2 和图 4.3 中,图 4.2 为 ejtag 正确返回值,图 4.3 为错误返回值,还有一种较常见错误为所有寄存器读回的值全为零。出现这情况,有两种情况导致,1, ejtag 到处理器的连接不正常,连接器上三角符号对应的为 1 脚,请参考图 xxj 检查硬件是否连接正确,2, 处理器虽然已经取址但工作状态不对,需要硬件工程师再确认排除问题。

- (1) 硬件确定 EJTAG 信号间无短路
- (2) 由于 2K 的 EJTAG 和 JTAG 复用,则需要确定复用配置是否正确:
EJTAG_SEL (1: JTAG; 0: EJTAG)
- (3) 检查 EJTAG 是否都已经正常上拉至 3.3V
- (4) EJTAG 的 2、4、6、8 均要接地
- (5) 若现已确认以上四项均无误后,EJTAG 还是无法使用的情况下则需要量取 EJTAG 信号,检查 EJTAG 接口信号是否正常通信。



第二步 烧写 pmon 到 flash

如果,生产前启动 flash 中未烧入 bios 文件,完成以上两部操作后处理器回进入异常模式 PC 为 0xffffffffbfc00000。通过 ejtag 命令如下可以查看

cpus

set

- (1) 把第二章编译好的 **gzrom-dtb.bin** 拷贝到/tmp/目录下。
- (2) 关闭板卡电源,在 ejtag 命令行执行 **loop -1 stop**
- (3) 板卡上电开机,这时 ejtag 命令,在循环执行 loop -1 stop 命令,执行 **ctrl ^C** 此时停

在 ejtag 命令行

(4) 执行烧写命令

Ejtag 采用锁 cache（即把 cache 当做内存用）方式烧写 bios。2K 目前应用通常采用 spi 启动；3a3000 支持 spi 和 LPC 启动，两种方式启动的板卡都存在（pmon 的编译请参考第二章）。

2k 处理器烧写 Pmon 命令为 `program_cachelock /tmp/gzrom-dtb.bin`。

如果使用的是 3a 处理器且启动 flash 为 spi 接口执行的烧写命令为 `program_cachlock_spi /tmp/gzrom-dtb.bin`；

如果使用的是 3a 处理器且启动 flash 为 LPC 接口烧写命令为 `program_cachlock /tmp/gzrom-dtb.bin`。

以上两部操作可以概括为如下图：

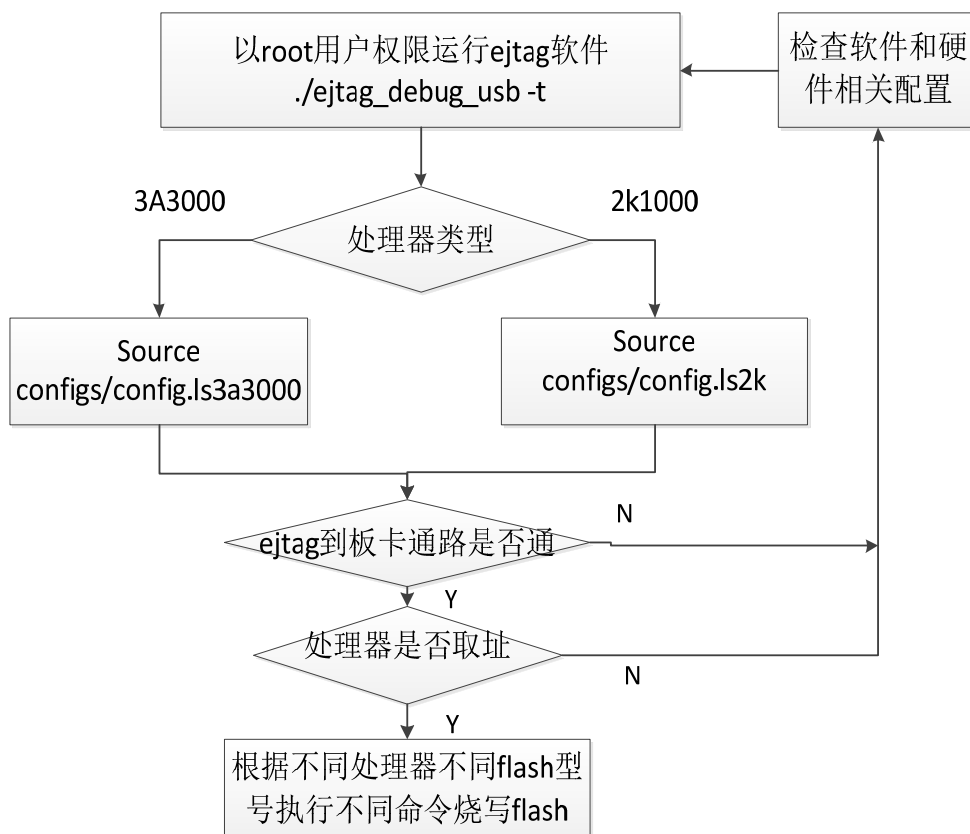


图 4.2 ejtag 烧写 bios 过程

在烧写 bios 过程，还可能遇到问题。比如板卡上有自带看门狗，烧写部分内容就烧不进去了，这需要软硬件协同。硬件可以暂时把看门狗去掉，软件如果对龙芯足够了解，可以

操作相关 IO 把看门狗关掉。

4.3.2 pmon 调试阶段

第二章介绍了 pmon 的编译，以及一些必要文件的位置，上一节中的 ejtag 的使用主要介绍如何烧写 pmon 到 flash 中。板卡的软件参与调试过程可归纳为：软件适应性修改->编译->烧写->问题调试。Pmon 作为 bootloader 的功能，调试过程主要集中在内存的稳定性上。

板卡上使用的内存分为插卡式（带 SPD 信息）和贴片式（不带 SPD 信息）两种类型，以贴片内存为例，不同板卡代码中需要修改的部分如下：

1) 手动配置 S1,在配置文件中（Targets/LS2K/conf/ls2k）关闭 AUTO_DDR_CONFIG，打开 DDR_S1（#option DDR_S1 为关闭选项，option DDR_S1 为打开选项），S1 的定义位于 Targets/LS2K/ls2k/ddr_dir 中，龙芯派的 S1 值为 0xc1a18404。

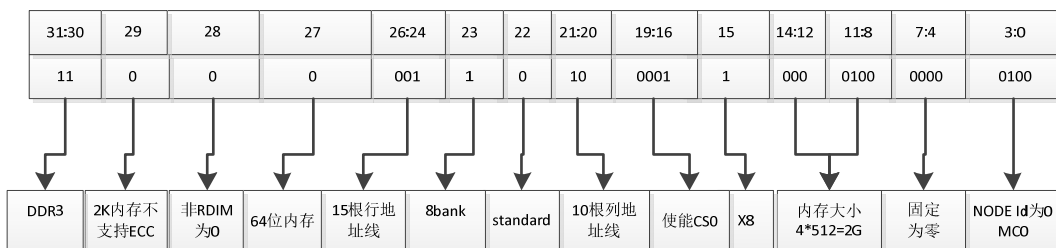


图 4.3 龙芯派 S1 说明

用户需要根据实际板卡使用内存情况，按 S1 的定义准确给出 S1 的值。内存训练程使用值，进行内存训练。

2) 频率修改，处理器和内存的频率同样通过配置文件中（Targets/LS2K/conf/ls2k）的宏进行控制，option DDR_FREQ=400 和option CORE_FREQ=1000 代表DDR运行频率 400M，处理器运行频率 1G。配置时钟的代码位于Targets/LS2K/ls2k/loongson3_clksetting.S中，具体含义参考处理器用户手册第三章时钟结构和第五章PLL相关寄存器说明。

3) start.S中几个和内存相关的宏：（1）#define DISABLE_DDR_A15 需要和硬件及S1 保持一致，如未用A15 地址线打开该宏。（2）#define PRINT_DDR_LEVELING 是否打印leveling信息。（3）#define DEBUG_DDR和#define DEBUG_DDR_PARAM为调试内存比较复杂的程序，需要熟悉DDR3 协议及龙芯内存训练程序。

4) 内存复位信号方向问题，如果内存复位信号硬件是直连到内存的，option DDR_RESET_REVERT注释掉（此宏位于配置文件中），如果硬件做了反向，需要把此宏打开。

5) 关于 0x18 寄存器, 从 2K 板卡调试情况看, 0x18 寄存器是调整内存参数上比较有限的参数。我们已把 DDR_PARAM_018 常用的几组值都放到了配置文件中, 遇到内存不稳定的情况, 可以尝试用不同的值进行的是。

6) 高低温内存不稳定, 内存使用过程中, 经常会遇到在常温下内存稳定。一旦做高低温实验时, 内存会极不稳定。0x1c8 中的 tRFC 和 tREF 可以影响到高低温下的内存稳定性,

刷新频率(tREF)只与温度有关, 而刷新时间(tRFC)只与内存容量有关, 经验上在高低温环境下, 可以适当减少 tREF, 并延长 tRFC (也可以只减少 tREF, 不改变 tRFC, 视调试情况来定)。BIOS 烧写到 flash 中后, 软件调试过程真正的开始。对于 2K 处理器相关板卡, 面对的调试问题主要有: (1) 串口是否出字 (2) 内存是否稳定。对于 3a3000 处理器面对的调试问题主要有: (1) 串口是否出字/乱码 (2) HT 是否能连得上 (3) 内存是否稳定。以下内容处理器来描述。

1, 2K 处理器部分串口:

2K 的串口时钟为内部时钟, 这部分配置软件已固定。只要 bios 正确烧写到了 flash 中, 处理器正确取址了串口会输出正确字符, 如果输出不正确排查问题有如下方法:

ejtag 下通过 cpus 命令观察当前 PC 值(通常 Pmon 阶段我们只关心处理器核 0 的位置), 根据 PC 值得情况有不同的处理器方法。

(1) PC 为 0xbfc00380 考虑 flash 烧写不成功;

(2) PC 为 0xbfc00cxx 值考虑 PCIE 的配置问题, 确认板卡是否使用了 PCIE, 以及相关配置是否正确。

2K1000 的 PCIE0 支持一路 x4 和四路 x1, PCIE0_PRSENTN1/2/3 中任意一个或多个被拉低, PCIE0 即被配置为四路 x1 模式, 否则为 x4 模式;

2K1000 的 PCIE1 支持一路 x4 和两路 x1, PCIE1_PRSENTN1 拉低时, 被配置成两路 x1 模式, 否则为 x4 模式; 两路 x1 模式下, 使用的是 TX0/RX0 和 TX1/RX1 两组;

PCIE 设备上的复位信号, 需要使用芯片的 GPIO, PMON 启动后需要软件控制一下复位, 否则 PCIE 无法正常工作;

龙芯 2K1000 输出的 PCIE CLOCK 理论上不用逻辑转换, 实测直接使用该信号也可以使用, 功能正常。

(3) PC 值大于 0xbfc00cXX, 或者 PC 为 9fcxxxxx、8fxxxxxx 此时处理器已经正常启动,

软件已操作串口。需要排查串口线和串口芯片。

2, 2k 内存调试

内存调试作为 pmon 调试的重要部分, 本篇内容只简单罗列一些内存调试的一些初步方法。如果这些方法和步骤不能解决用户板卡内存的稳定性, 需要根据 DDR 协议以及龙芯提供的文档, 软硬件进一步调试。常规问题应该注意以下几点:

1) 2K 处理器最高可以工作在 1G, 但对处理器核压有要求。如果工作在 1G, 要求核压是 1.2V。如果不满足这个条件部分板卡会表现为内存不稳定。

2) 2K 内存频率最高运行 500MHz, 一般插槽式内存可以跑到 500MHz。我们建议内存频率设置为 400MHz

3) 降频(处理器和内存)升核压(按手册要求的范围)是检验硬件是否有问题的有效手段。

4) 在调试内存前, 硬件工程师除了确认核压以外, 也要确认内存电压, 以及核电和内存电的纹波。纹波要求在该电源的 $\pm 5\%$ 之内。

<<内存调试手册>>中详细的介绍了内存相关的知识、内存参数以及 pmon 中内存训练程序也进行了概要性总结。在确认了以上常规设置后, 内存还存在稳定性问题。用户需要仔细阅读<<内存调试手册>>, 根据<<内存调试手册>>4.4 节中介绍对内存参数进行调整。

4.3.3 3a3000 处理器部分

3a3000 处理器的板卡, 调试流程和 2K 处理器一致。3a3000 中有两路内存控制器, 在配置 S1 时, 按硬件实际使用情况, 配置对应位。3a3000 和 2k 使用的内存控制器是一样的, 所有调试方法和 2K 也保持一致。<<内存调试手册>>适用于 3a2000、3a3000、2k1000 处理器。

4.3.4 内核测试阶段

pmon 中无法产生大压力的内存测试, 当 pmon 测试内存稳定后, 能够启动内核。内存测试转移到内核下进行。常用工具是 stressapptest, 命令格式为 `./stressapptest -M 100 -s 999999 &` 其中 -M 为当前 stressapptest 程序测试内存的大小 100M (大小用户可自行定义), 通常为了给内存产生更大的压力, 会启动多个 stressapptest, 所有程序测试内存大小要小于总内存大小。当大于内存大小时, 进程会被杀死。对应的错误, 请参考<<内存调试手册>>。

安装 loongnix 系统也是检验内存稳定性的一个有效手段, 当安装系统时出现各种奇怪的错误, 安装中出现错误, 通常是内存不稳定引起。

第五章 Loongnix 安装说明

5.1 安装盘制作

5.1.1 下载系统 iso 文件

<http://www.loongnix.org/index.php/Loongnix>

备注：目前最新版本为 loongnix-20180930.iso

5.1.2 U 盘安装盘制作

制作 USB 安装盘, 需要准备:

一个 4GB 以上的 U 盘

一台正常运行的 Linux 主机

linux 主机插入 U 盘（识别盘符为/dev/sdb），在终端上执行命令(需要管理员用户):

```
# dd if=loongnix-20180930.iso of=/dev/sdb bs=8M
```

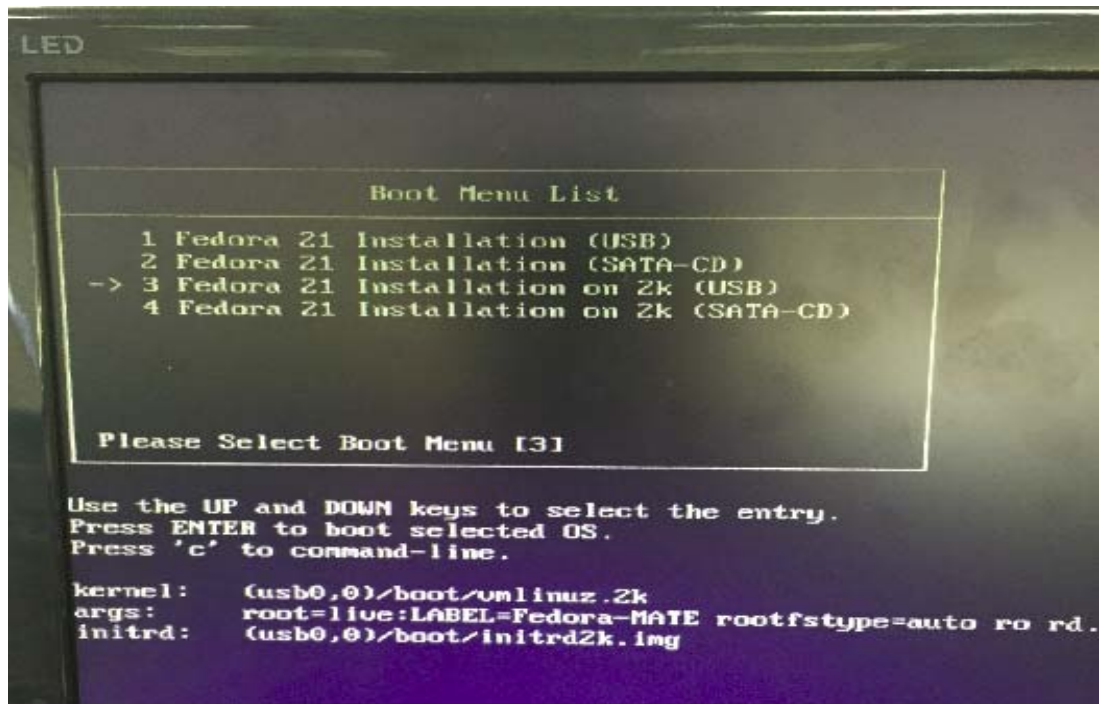
```
# sync
```

完成后拔掉 U 盘

5.2 安装过程

5.2.1 启动机器

龙芯机器上电前插入 U 盘，启动安装盘, 进入选择菜单如下图，使用上下键进行选择，这里选择 2K U 盘安装为例选择第三项。

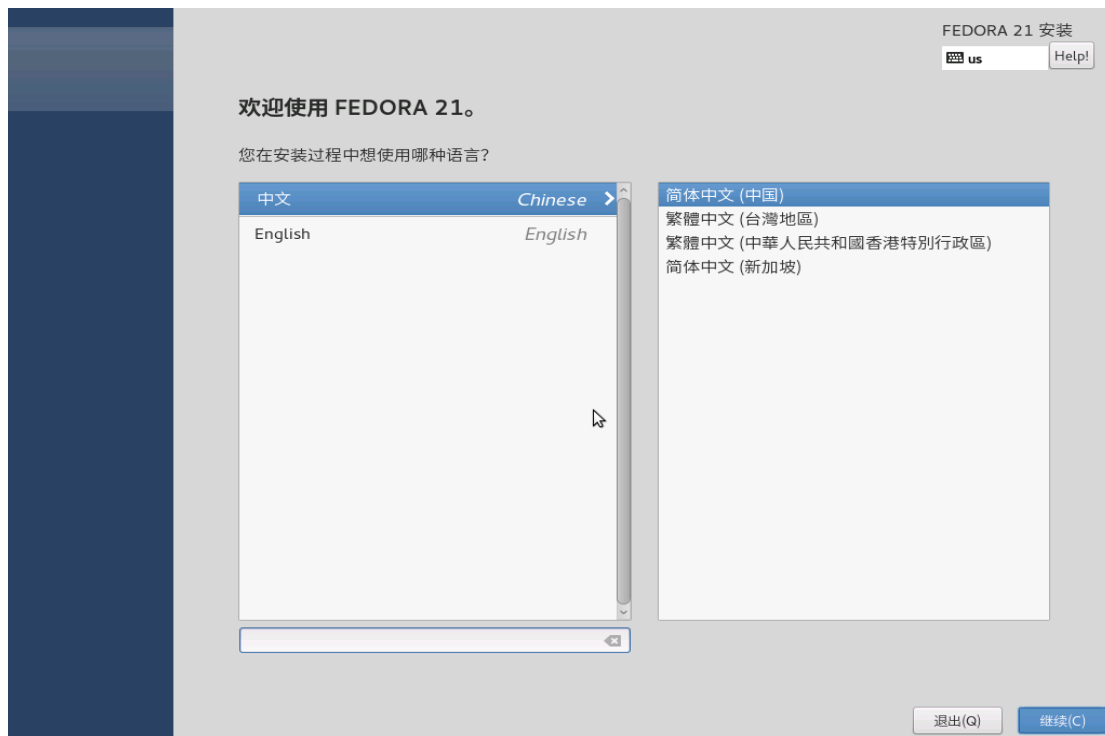


选择完成按回车键，开始加载，等待进入到安装界面如下图



5.2.2 开始安装

在桌面上找到“安装系统”的图标，双击运行



点击右下角“继续”：



该界面中若某个项目出现黄色警告图标，则代表该项目有需要进行设置的内容。

5.2.3 硬盘分区设置

点击“安装位置”，点击磁盘图标，会在上面出现一个“对号”，显示如下



默认为“自动配置分区”（手动分区，操作详见附件 A），点击完成，空间不够会弹出“回收磁盘空间窗口”



点击“全部删除”，等待“回收空间”变成可以点击项，返回主界面，开始安装。



点击右下角的“开始安装”按钮，开始安装。



5.2.4 用户设置

在安装过程中，界面上有两个标红的选项“ROOT 密码”和“创建用户”，在安装过程或者安装完成后我们都可以进行这两项的设置。选择“ROOT 密码”，输入相同的密码，点击左上角的“完成”完成设置，（如果密码过于简单，需要点击两次）。



ROOT 密码

完成(D)

FEDORA 21 安装

US Done

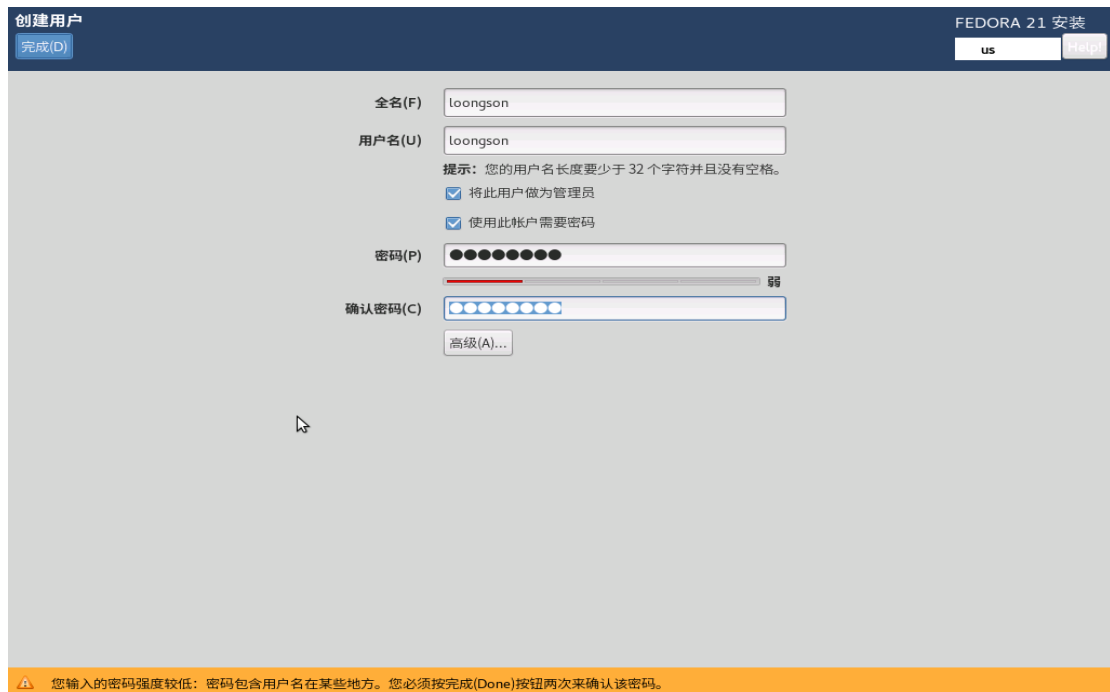
root 帐户用于管理系统。为 root 用户输入密码。

Root 密码:

确认(C):

您输入的密码强度较低。您必须按完成(Done)按钮两次来确认该密码。

选择“创建用户”，创建一个日常工作使用的帐户



创建用户

完成(D)

FEDORA 21 安装

US Done

全名(F)

用户名(U)

提示：您的用户名长度要少于 32 个字符并且没有空格。

☒ 将此用户做为管理员

☒ 使用此帐户需要密码

密码(P)

确认密码(C)

高级(A)...

您输入的密码强度较低：密码包含用户名在某些地方。您必须按完成(Done)按钮两次来确认该密码。

5.2.5 完成安装

经过一段时间，安装结束，右下角会出现“退出”按钮，点击后退回到桌面，在“系统”菜单选择“关机 — 重新启动”。



5.3 注意事项

- 1 loongnix 系统暂时不支持同一个硬盘中安装多个系统。
- 2 pmon USB 不支持热插拔，上电前要连接好键盘。
- 3 U 盘启动盘只能插到处理器自带的 USB 接口，不能使用 PCIE 扩展出来的 USB 接口。
- 4 在制作 U 盘启动盘时，最好在 linux 系统使用 dd 命令来做，如果在 window 下用 U 盘制作工具做的话可能会出现 U 盘无法被识别。

5.4 常见问题

- 1 安装时报错，建议手动加载，将串口打印信息打开，操作如下
 - A、连接上调试串口，重新上电
 - B、进入选择菜单后，按“C”键进入 Pmon 命令行下
 - C、输入 devls 命令可以查看到 usb0
 - D、输入 devcp (usb0,0)/boot/boot.cfg /dev/tty0

```
-select normal memory access (64 bit uncached physical address)
AUTO
Press <Enter> to execute loading image:(wd0,0)/boot/vmlinuz
Press any other key to abort.
01
PMON>
PMON>
PMON> devls
Device name  Type
loopdev0     DISK
syn0         IFNET
syn1         IFNET
usb0         DISK
wd0          DISK
PMON> devcp (usb0,0)/boot/boot.cfg /dev/tty0
\
default 0
showmenu 1

title Fedora 21 Installation (USB)
    kernel (usb0,0)/boot/vmlinuz
    initrd (usb0,0)/boot/initrd.img
    args root=live:LABEL=Fedora-MATE rootfstype=auto ro rd.live.image quiet splash

title Fedora 21 Installation (SATA-CD)
    kernel (cd0,0)/boot/vmlinuz
    initrd (cd0,0)/boot/initrd.img
    args root=live:LABEL=Fedora-MATE rootfstype=auto ro rd.live.image quiet splash

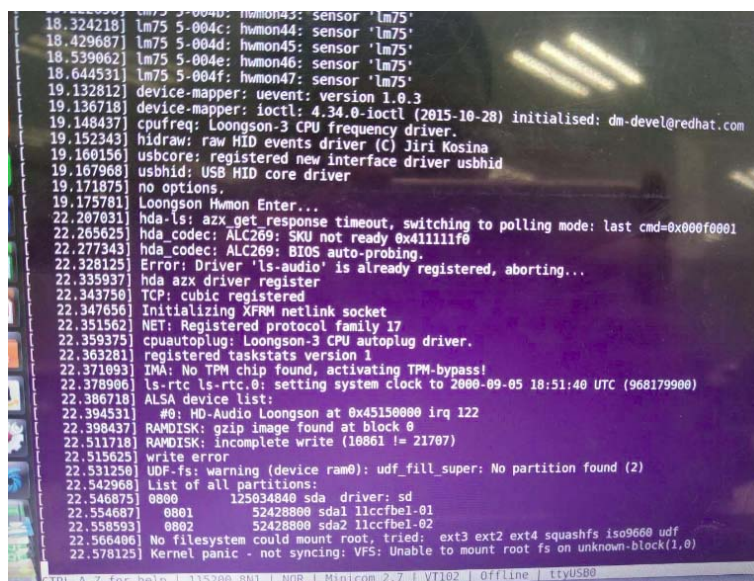
title Fedora 21 Installation on 2k (USB)
    kernel (usb0,0)/boot/vmlinuz.2k
    initrd (usb0,0)/boot/initrd2k.img
    args root=live:LABEL=Fedora-MATE rootfstype=auto ro rd.live.image quiet splash
```

- (3) 输入 load (usb0,0)/boot/vmlinuz.2k
- (4) 输入 initrd (usb0,0)/boot/initrd2k.img
- (5) 输入

g console=ttyS0,115200 root=live:LABEL=Fedora-MATE rootfstype=auto rd.live.image

备注：如更换内核，可在第 E 步加载想替换的内核即可

2. 3a3000+7A 装中标系统和装 loongnix 系统步骤一样，但在装 5.0 和 7.0 的中标系统测试版的时候，会出现如下问题：



```
18.324218] lm75 5-004d: hwmon43: sensor 'lm75'
18.429687] lm75 5-004d: hwmon44: sensor 'lm75'
18.539062] lm75 5-004d: hwmon45: sensor 'lm75'
18.644531] lm75 5-004d: hwmon46: sensor 'lm75'
19.132812] device-mapper: uevent: version 1.0.3
19.136718] device-mapper: ioctl: 4.34.0-ioctl (2015-10-28) initialised: dm-devel@redhat.com
19.148437] cpufreq: Loongson-3 CPU frequency driver.
19.152343] hidraw: raw HID events driver (C) Jiri Kosina
19.160156] usbcore: registered new interface driver usbhid
19.167968] usbhid: USB HID core driver
19.171875] no options.
19.175781] Loongson Hwmon Enter...
22.207031] hda-azx: azx_get response timeout, switching to polling mode: last cmd=0x00f0001
22.205025] hda_codec: ALC269: SKU not ready 0x411111f0
22.277343] hda_codec: ALC269: BIOS auto-probing.
22.328125] Error: Driver 'ls-audio' is already registered, aborting...
22.335937] hda azx driver register
22.343750] TCP: cubic registered
22.347056] Initializing XFRM netlink socket
22.351562] NET: Registered protocol family 17
22.359375] cpufreq: Loongson-3 CPU autoplugin driver.
22.363281] registered taskstats version 1
22.371093] IMA: No TPM chip found, activating TPM-bypass!
22.378906] ls-rtc ls-rtc.0: setting system clock to 2000-09-05 18:51:40 UTC (968179900)
22.386718] ALSA device list:
22.394531] #0: HD-Audio Loongson at 0x45150000 irq 122
22.398437] RAMDISK: gzip image found at block 0
22.511718] RAMDISK: incomplete write (10861 != 21707)
22.515625] write error
22.531250] udf-fs: warning (device ram0): udf_fill_super: No partition found (2)
22.542968] List of all partitions:
22.546875] 0000 125034840 sda driver: sd
22.554687] 0001 52428800 sda1 llccfbel-01
22.558593] 0002 52428800 sda2 llccfbel-02
22.566406] No filesystem could mount root, tried: ext3 ext2 ext4 squashfs iso9660 udf
22.578125] Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(1,0)
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7 | VT102 | Offline | ttyUSB0
```

内核能成功起来，但在加载 initrd 的时候报错：No filesystem could mount root.这时需要用显卡来装，推荐使用 R5230,HD8470,E8860(使用这块显卡装的系统特别流畅，不卡顿)，另外如果用 R5230 显卡和 HD8470 装的系统用 E8860 系统起的话会出现连键的现象。

3.使用 uefi 装 loongnix 系统，要在启动管理里面将 U 盘设置为第一启动，uefi 和 pmon 不一样，不能自动识别到 U 盘的镜像，然后装完系统后要将硬盘重新设置为第一启动。

uefi 支持的 loongnix 系统要是 2018-09-30 及以上的。

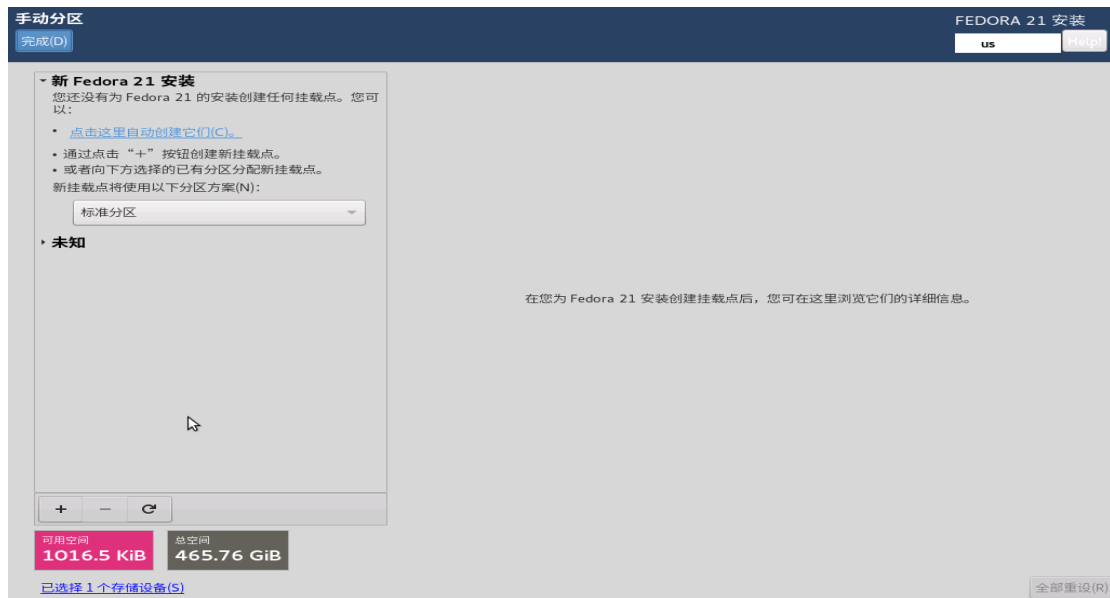
附录 A 手工创建新系统分区

如果你的磁盘已经安装过其他系统或者觉得自动设置的分区方案不合适，想自己设置硬盘分区情况，你可以采用手工创建新系统分区的方式

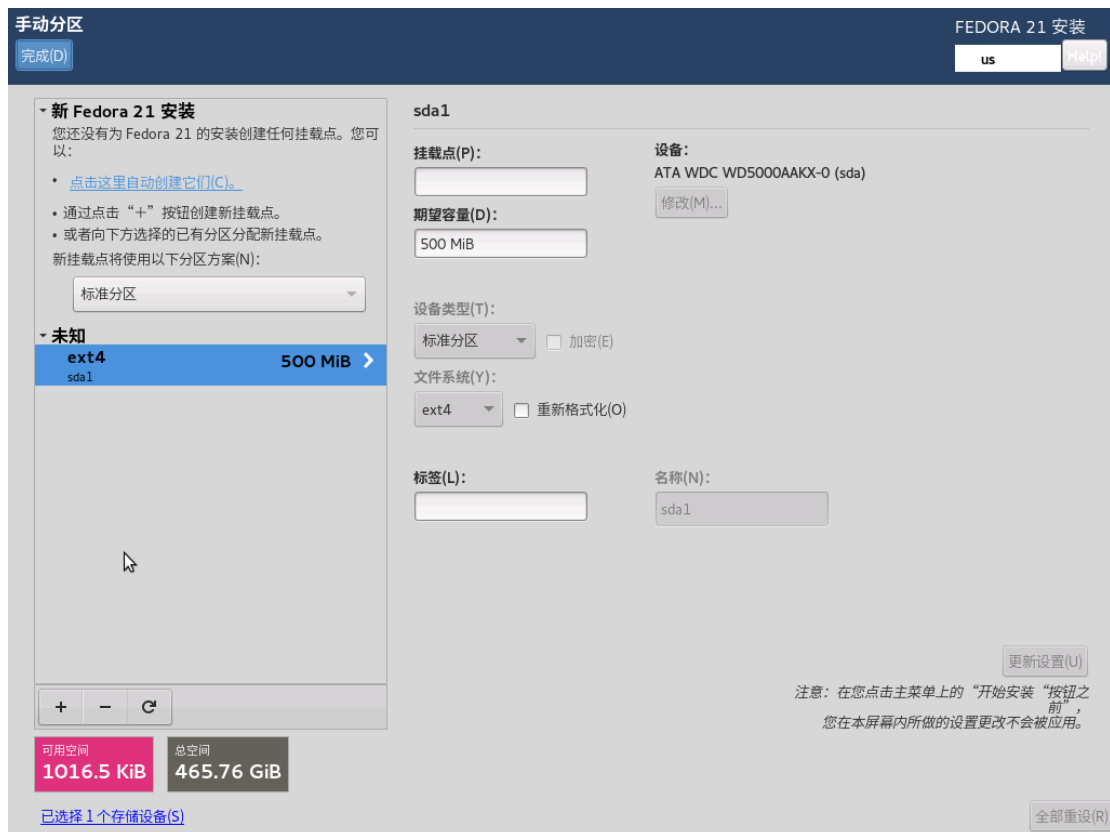
在磁盘安装设置界面，选择“我要配置分区”



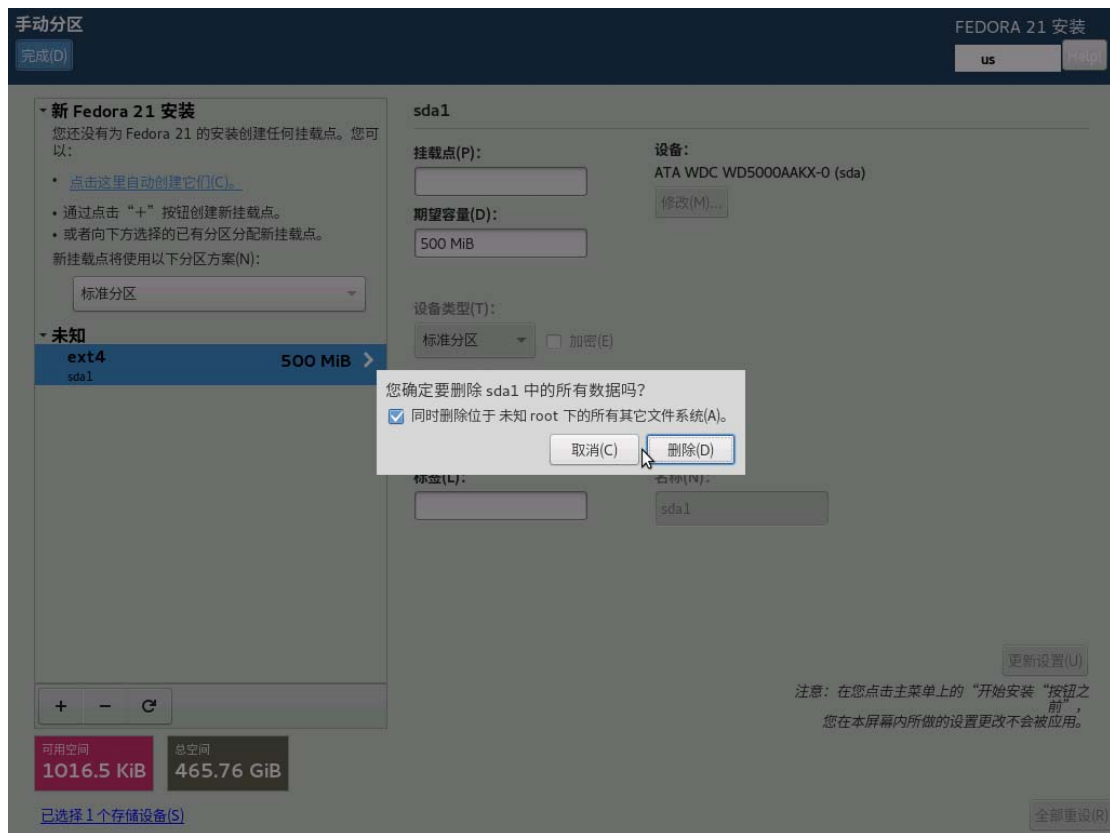
点击“完成”



磁盘可用空间有 1016.5KiB，所以要删除以前分配的空间，点击“未知”选中分区



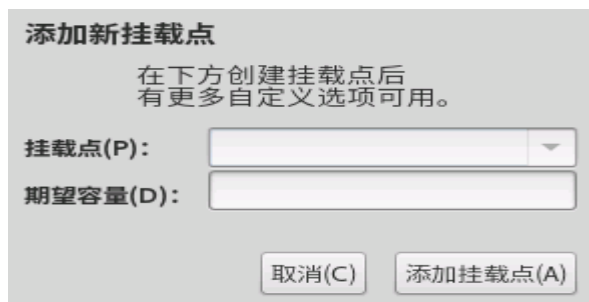
点击“-”号，弹出如下窗口



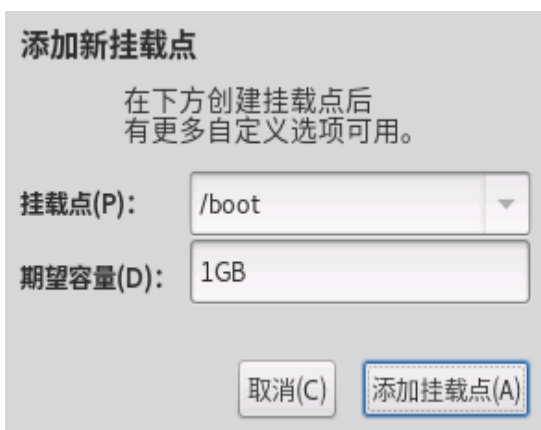
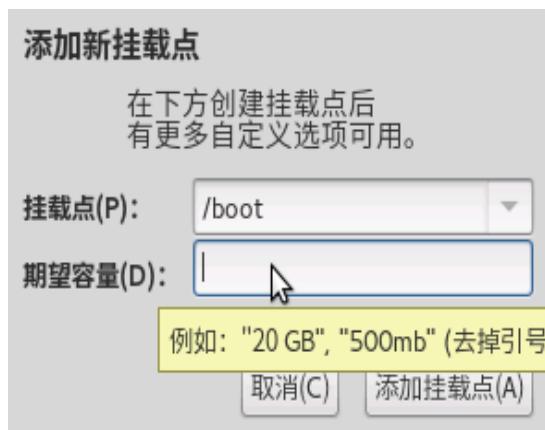
勾选“同时删除...”，点击“删除”



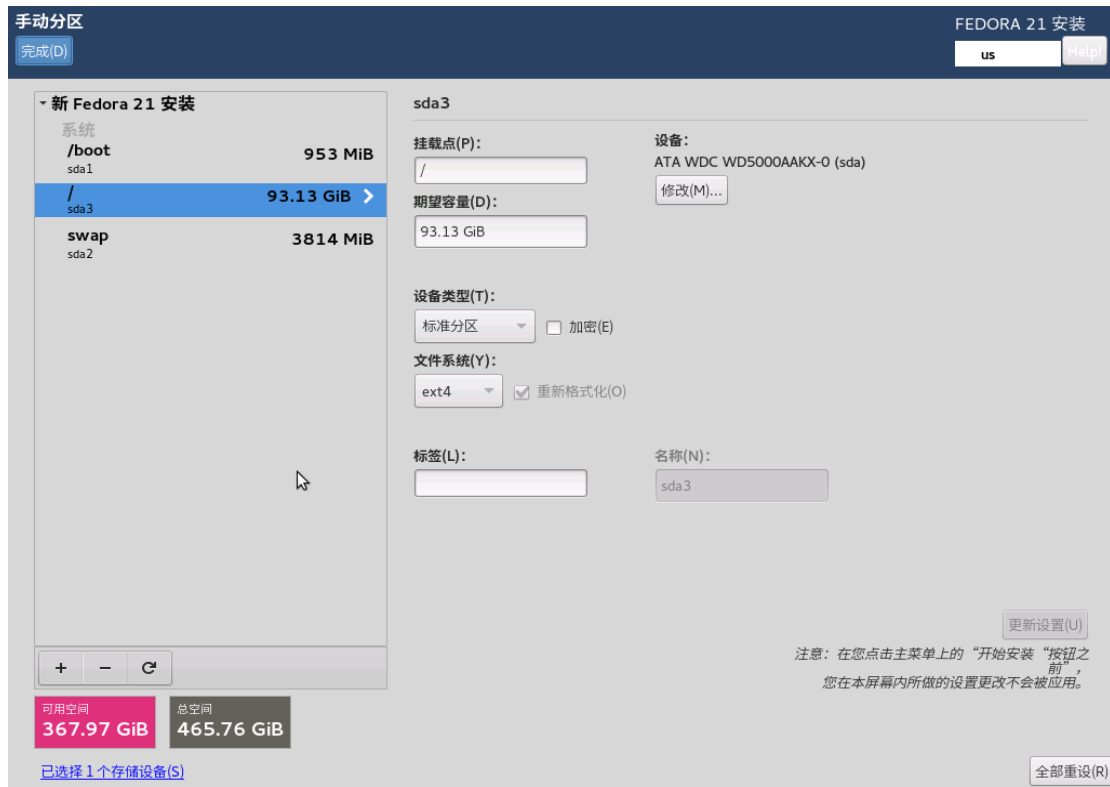
此时可用空间为 465.76GiB。开始手动分区，点击“+”号，弹出如下窗口



本例子里：制作 3 个分区，根分区挂载点(挂载位置为 “/”)、/boot 分区以及 swap 分区，根分区挂载点分 100G；/boot 分区提供 500M~1G 的空间即可；swap 的分区大小建议根据内存大小进行调整，一般是内存容量的 1~2 倍。



按需要创建好挂载点及容量后，界面如下图：



创建完分区后，点击左上角“完成”，弹出如下窗口



点击“接受更改”，手动分区完成，可以开始安装。



附录 B mtd 设备的使用

mtd 设备是嵌入式应用中比较常见的存储设备。目前，龙芯在 1、2 号处理器上对 mtd 相关设备有了比较完善的软件支持。常见的 mtd 设备包括 nand 接口 nanflash；spi 接口 spi norflash 和 spi nandflash。本节介绍 mtd 设备在 pmon 和内核下的使用例程。

1， 相关内容的编译

Pmon 下 mtd 内核内容的编译，配置文件（Targets/LS2K/conf/ls2k）中的选项，如果使用的是 nand 接口 nand 选项为：

```
Select nand
```

```
Option CONFIG_LS2K_NAND
```

如使用的是 spi 接口 nand 选项为：

```
Select nand
```

```
Select spinand_mt29f 注：使用镁光的 spi nand
```

如使用的是 spi 接口的 nor flash 选项为：

```
Select m25p80
```

2， Pmon 下 mtd 设备的使用

请参考 2.5.4 节内容

3， Kernel 下 mtd 设备的使用

在内核下提供了 char 和 block 两种接口访问设备。作为 block 设备访问，设备节点为/dev/mtdblock*（*代表数字 0,1,2.....），常用文件系统为 yaffs2，龙芯提供的内核已经集成了 yaffs2 文件系统。在执行 mount 命令时，自动格式化分区为 yaffs2 格式。Mount 命令如下：

```
mount -t yaffs2 /dev/mtdblok1 /mnt
```

4， Pmon 和 kernel 的传参

Pmon 和 kernel 通过 mtdparts 环境变量传参。在 Pmon 命令行下执行 set 命令可看到默认的 mtdparts 值。

5， 纠错算法

目前 Pmon 和内核中支持 ecc 和 bch 纠错算法。

如果使用 ECC 算法在驱动中（pmon 下驱动为 sys/dev/nand/ls2k-nand.c 内核下驱动为：drivers/mtd/nand/ls-nand.c）设置

```
this->ecc.mode =NAND_ECC_SOFT
```

如果使用 bch 算法，pmon 配置文件下需要打开 select nand_bch 同时 pmon 和内核下设置

```
this->ecc.mode =NAND_ECC_SOFT_BCH
```

6. 内核与文件系统放到 nand 不同分区里

假设 nand 分两个分区，内核放到 mtd0 分区；文件系统放到 mtd1 分区

A、第一种操作方式

ftp 上获取 yaffs2 文件系统

<ftp://ftp.loongnix.org/embedd/ls2k/rootfs.yaffs2>

启动到 pmon 命令行下

a、mtd_erase /dev/mtd0r

b、mtd_erase /dev/mtd1r

c、devcp tftp://ip/vmlinuz /dev/mtd0

d、devcp tftp://ip/rootfs.yaffs2 /dev/mtd1y

e、set al1 /dev/mtd0

f、set rd /dev/fs/yaffs2@mtd1/rootfs.cpio.gz

g、set append "console=ttyS0,115200"

h、reboot

B、第二种操作方式

用 mkyaffs2 工具制作 yaffs2 文件系统

制作 mkyaffs2 工具：

a、获取 yaffs2utils 源码

<http://sources.buildroot.net/yaffs2utils/0.2.9.tar.gz>

b、解压 0.2.9.tar.gz 编译

进 0.2.9 目录下执行 make 编译，执行完生成 mkyaffs2、unyaffs2 工具

```
sudo chmod 777 mkyaffs2
```

```
sudo chmod 777 unyaffs2
```

```
sudo cp mkyaffs2 unyaffs2 /usr/bin/
```

c、制作 yaffs2 文件系统

```
mkyaffs2 -p 4096 -s 128 --yaffs-ecclayout rootfs/ rootfs.img
```

备注：制作 yaffs2 文件系统时根据实际 nand 情况输入参数

启动到 pmon 命令行下

- a、mtd_erase /dev/mtd0r
- b、mtd_erase /dev/mtd1r
- c、devcp tftp://ip/vmlinuz /dev/mtd0
- d、devcp tftp://ip/rootfs.img /dev/mtd1y
- e、set al1 /dev/mtd0
- f、set append "console=ttyS0,115200 init=/linuxrc rw root=/dev/mtdblock1
rootfstype=yaffs2"
- g、reboot

附录 C LS2K GPIO 中断配置

一、ls2k gpio 简述

龙芯 2K1000 有 60 个 GPIO 引脚，GPIO 引脚与中断引脚的对应关系如下：

表 1-1 LS2K1000 GPIO 中断

GPIO 引脚	中断引脚	中断号	说明
GPIO0	Gpio_int0	68	专用 GPIO 引脚, 与中断引脚一一对应
GPIO0	Gpio_int1	69	
GPIO0	Gpio_int2	70	
GPIO0	Gpio_int3	71	
GPIO[31:04]	Gpio_int_lo	66	GPIO4~31 复用中断引脚 Gpio_int_lo
GPIO[63:32]	Gpio_int_hi	67	GPIO32~63 复用中断引脚 Gpio_int_hi

表 1-2 GPIO 中断相关寄存器

寄存器	地址	描述
Intpol_1	0x1fe11470	中断极性控制:1 代表低电平, 0 代表高电平
Intedge_1	0x1fe11474	触发方式寄存器 (1:脉冲触发;0:电平触发)
Intenset_1	0x1fe11468	设置中断使能寄存器
GPIO0_INTEN	0x1fe10530	63:0 中断使能位, 每一位对应一 GPIO 引脚

寄存器 Intpol_1、Intedge_1、Intenset_1 第 26~31 位对应 GPIO 中断引脚：

第 26、27 位分别对应：Gpio_int_lo、Gpio_int_hi

第 28~31 位分别对应：Gpio_int0、Gpio_int1、Gpio_int2、Gpio_int3

二、Gpio 中断软件配置操作

以设置 GPIO0 中断为例：

第一种模式：下降沿触发中断

1、gpio0 设置为输入模式

2、配置 gpio0 触发类型为下降沿触发

寄存器 Intpol_1 第 28 位置 1

寄存器 Intedge_1 第 28 位置 1

3、使能 GPIO0 中断

寄存器 GPIO0_INTEN 第 0 位置 1

第二种模式：上升沿触发中断

1、gpio0 设置为输入模式

2、配置 gpio0 触发类型为上升沿触发

寄存器 Intpol_1 第 28 位置 0

寄存器 Intedge_1 第 28 位置 1

3、使能 GPIO0 中断

寄存器 GPIO0_INTEN 第 0 位置 1

第三种模式：低电平触发中断

1、gpio0 设置为输入模式

2、配置 gpio0 触发类型为低电平触发

寄存器 Intpol_1 第 28 位置 1

寄存器 Intedge_1 第 28 位置 0

3、使能 GPIO0 中断

寄存器 GPIO0_INTEN 第 0 位置 1

第四种模式：高电平触发中断

1、gpio0 设置为输入模式

2、配置 gpio0 触发类型为高电平触发

寄存器 Intpol_1 第 28 位置 0

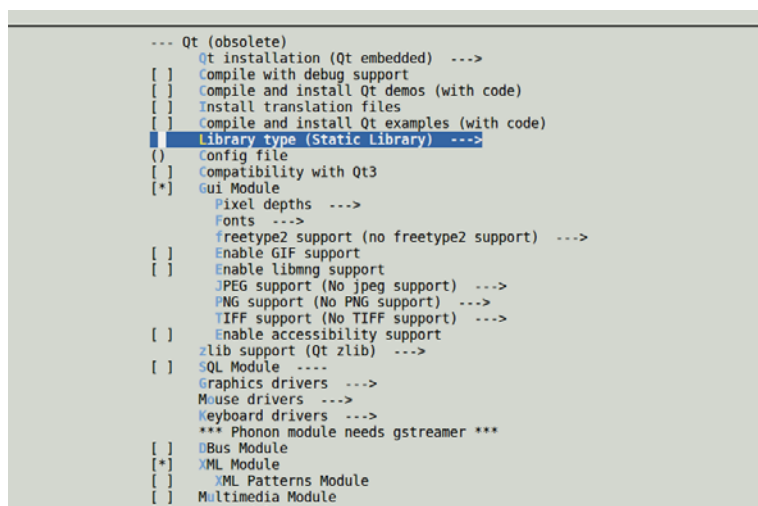
寄存器 Intedge_1 第 28 位置 0

3、使能 GPIO0 中断

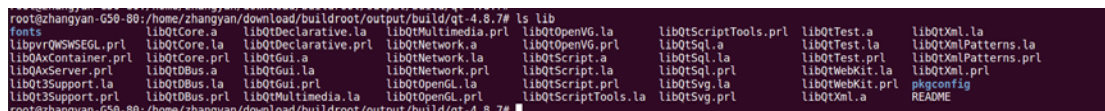
寄存器 GPIO0_INTEN 第 0 位置 1

附录 D 配置 QT（交叉开发 QT 应用）

1. 首先 buildroot 的 qt 源码编译一定要是静态编译，因为动态编译执行程序时 platform plugins 找不到。如果是按照我上面的步骤编译的话，是动态编译 qt 源码的，所以首先将 ./output/build/qt-4.8.7 这个文件夹删除，然后在 menuconfig 中 target packages --> Graphic....-->Qt-->Library type(static Library)选择静态编译，如下图所示：

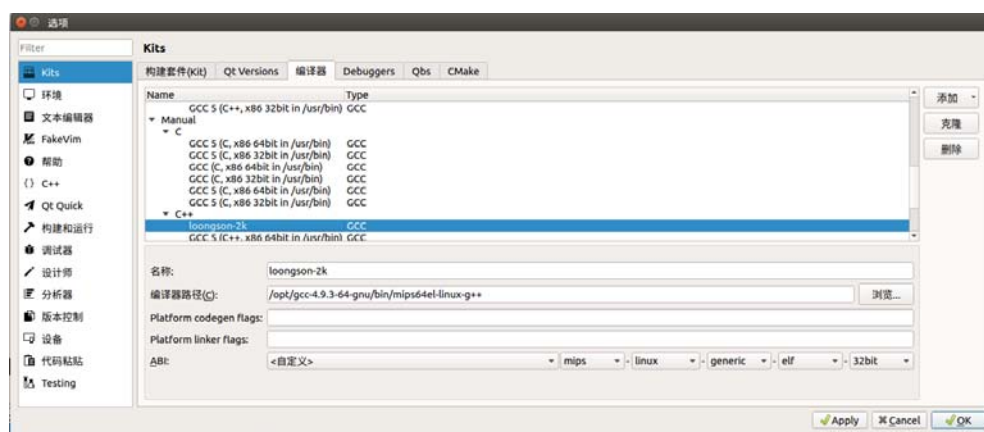


2. 然后执行 ./cmd.sh 重新编译会生成新的 qt-4.8.7 安装目录，看一下 lib 下面的库是不是静态库，如下图所示表示正确：

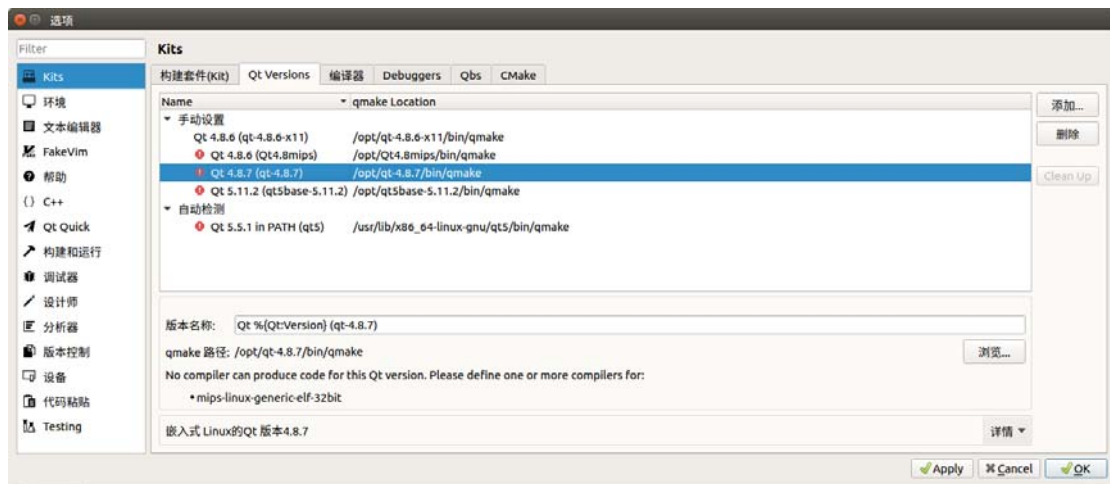


3. 将 qt-4.8.7 这个文件夹拷贝到 /opt 目录下(也可以不拷贝，这里只是问了后面方便找和方便管理)。

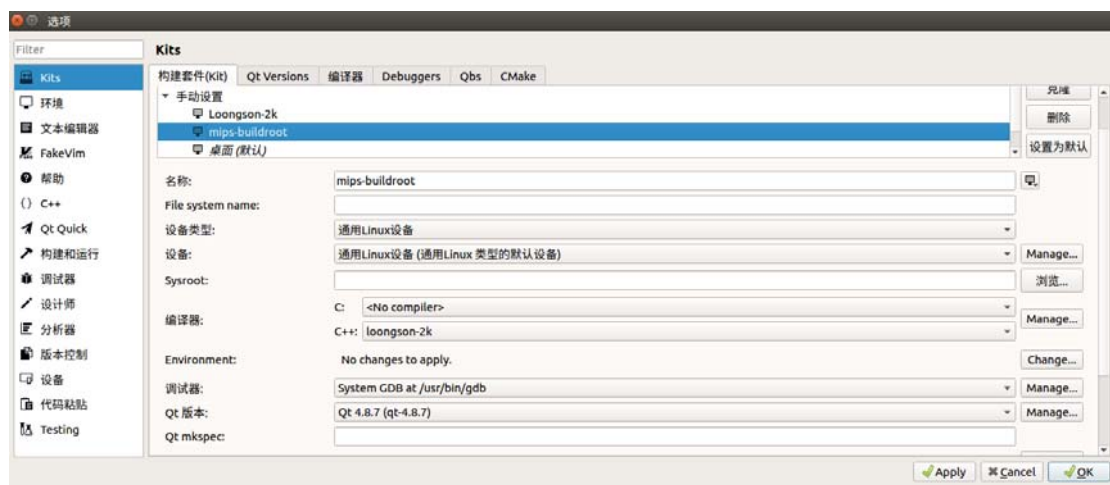
4. 打开 qt creator，打开工具---->选项--->kits.配置交叉编译的工具链，如下图所示：



5. 然后添加 Qt versions，如下图所示(就是第三步拷贝到 /opt 下的 qt-4.8.7 目录下的 bin/qmake)这里有个红色的感叹号，不用管它，不影响使用：



6. 最后添加一个新的套件(kit)，按下图所示操作：

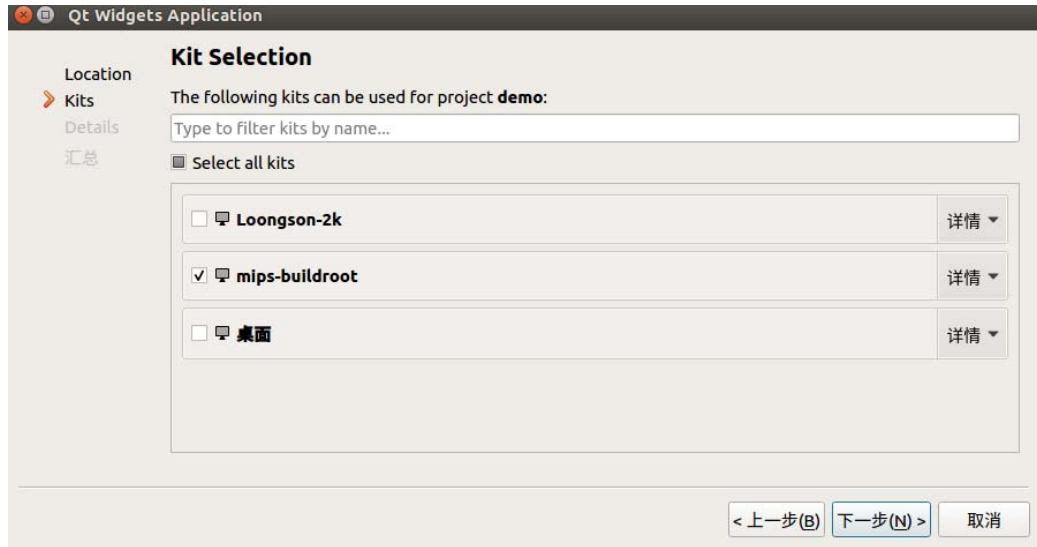


7. 到此环境已经配置完成。

8. 然后新建工程，下图所示的创建路径最终编译后生成的可执行文件就在该目录下：



Kit 套件一定要选择刚自己构建的套件，如下图所示(mips-buildroot 是我的套件)：

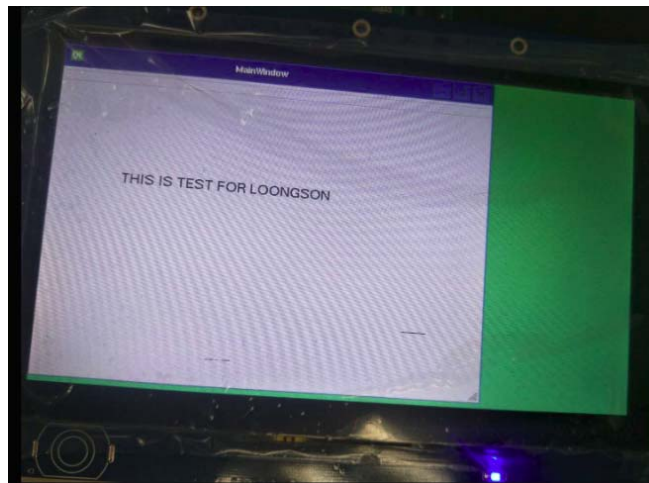


9. 编译会出现错误 :uic 和 moc 这两个命令找不到，解决办法：将 output/build/qt-4.8.7/bin/moc 和 uic 拷贝到 output/host/mips64el-buildroot-linux-gnu/sysroot/usr/bin。然后交叉编译，编译成功(快捷键 ctrl+b 进行快速编译)。

9. 将编译后的可执行文件拷贝到板子里面，直接执行，在拷贝之前可以 file 看一下这个可执行文件，如下图所示 MIPS64 既为正确。

```
zhangyan@zhangyan-G50-88:~/qt_project/build-77-mips_buildroot-Debug$ file build_test
build_test: ELF 64-bit LSB executable, MIPS, MIPS64 rel2 version 1 (SYSV), dynamically linked, interpreter /lib64/l, for GNU/Linux 2.6.32, not stripped
zhangyan@zhangyan-G50-88:~/qt_project/build-77-mips_buildroot-Debug$
```

10. 在板子的内核命令行下执行： ./build_test -qws。 如下图所示：



附录 E 内存问题汇总
(建设中....)