

目录

1. BIOS 编译.....	1
1.1 pmon 编译及注意事项.....	1
2. 内核编译.....	4
2.1 编译环境及依赖工具下载.....	4
2.2 指定交叉编译工具链.....	4
2.3 设置内核配置文件.....	4
2.4 打开图形化配置界面（可在此环节修改内核相关配置）.....	4
2.5 编译内核.....	5
3. 文件系统制作.....	6
3.1 buildroot 编译.....	6
3.2 LoongOS 编译.....	6
4. 启动与调试.....	7
4.1 启动.....	7
4.2 ejtag 调试.....	10
5. 接口配置与使用.....	12
5.1 GMAC.....	12
5.2 VGA.....	14
5.3 OTG.....	16
5.4 USB2.0.....	18
5.5 USB3.0.....	19
5.6 PCIE.....	20
5.7 PCI.....	22
5.8 SDIO.....	25
5.9 NAND.....	27
5.10 CAN.....	31
5.11 I2C.....	34
5.12 SPI.....	41

5.13 RS485/UART3.....	46
5.14 WATCH DOG.....	48
5.15 PWM.....	50
5.16 GPIO 中断.....	53
附录 A：2K500 复用设置注意事项.....	54
(1) 默认复用设置.....	54
(2) cfg_func_multi 函数设置.....	54
(3) pmon 命令行下调用命令修改复用配置.....	55
(4) 复用冲突处理.....	55

版本记录	
版本号：V0.1	
V0.1	重新加上了编译环境搭建以及依赖工具下载的内容。

1. BIOS 编译

1.1 pmon 编译及注意事项

以编译 2K500 的 pmon 源码为例：

(1) pmon 源码

将源码压缩包拷贝到工作目录，使用下述命令对源码进行解压（注：如果使用的

的工作环境是虚拟机，请不要直接在共享文件夹下进行解压）

```
tar -zxvf pmon-ls2k500.tar.gz
```

(2) 解压交叉编译工具

来到根目录下，解压交叉编译工具，默认会放入 opt 目录下。

```
cd /
```

```
tar -zxvf loongarch_toolchain.tar.gz
```

(3) 编译环境及依赖工具下载

编译环境：ubuntu20.04

```
a. sudo apt install aptitude  
sudo aptitude install xutils-dev
```

```
b. sudo apt install bison flex
```

进入 pmon 源码下进行工具编译：

```
cd PMON 源码/tools/pmoncfg
```

```
make pmoncfg
```

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

c. `sudo apt install acpica-tools`

(4) 根据下述命令编译 pmon

解压过程如果没有“error”信息，则可继续进行下述步骤。

解压完成后，会在当前目录下看到一个 pmon/目录，使用命令 `cd pmon/`

进入源码目录下，使用 `ls` 命令可看到如下目录结构。

```
pdb@pdb:~/work/loongson/source/2k500$  
pdb@pdb:~/work/loongson/source/2k500$ cd pmon/  
pdb@pdb:~/work/loongson/source/2k500/pmon$  
pdb@pdb:~/work/loongson/source/2k500/pmon$  
pdb@pdb:~/work/loongson/source/2k500/pmon$ ls  
conf      doc      fb       lib      Makefile.inc  sys      tools    zloader    zloader.3a5000_7a  
Copyright examples include Makefile pmon      Targets  x86emu    zloader.2k500
```

1) 进入 2K500 对应的编译目录

`cd zloader.2k500/`

2) 设置环境变量，指定当前目录下的编译工具

`export PATH=/opt/loongarch_toolchain/bin:$PATH`

注：编译工具链 loongarch_toolchain.tar.gz 如果解压在/opt 目录下，则使用上述命令进行设置，否则需修改为工作环境下编译工具所在的路径。

3) 使用命令编译 BIOS：

`make cfg all tgt=rom ARCH=loongarch`

`CROSS_COMPILE=loongarch64-linux-gnu- DEBUG=-g`

编译选项解释：

`make cfg` 对 pmon 进行配置；

all 为 Makefile 里的编译项；

tgt=rom，指定 tgt 为 rom，则会生成 gzrom.bin 文件；

ARCH=loongarch，指定架构为 loongarch；

CROSS_COMPILE=loongarch64-linux-gnu-，指定编译工具前缀名；

DEBUG=-g，设置编译的时候携带调试信息。

4) 编译设备树，并生成携带设备树的 BIOS 文件“gzrom-dtb.bin”

```
make dtb ARCH=loongarch CROSS_COMPILE=loongarch64-linux-gnu-
```

编译完成后，会在 zloader.2k500/下，看到最终生成的携带设备树的 gzrom-dtb.bin 文件，以及不带设备树的 gzrom.bin 文件。

注：如更改了配置文件 Targets/LS2K500/conf/ls2k500，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。

(5) pmon 编译脚本文件

可在 pmon 源码目录下，创建下述脚本文件，如需重新编译 pmon，则只需执行该脚本即可。

```
#!/bin/bash
cd zloader.2k500/
export PATH=/opt/loongarch_toolchain/bin:$PATH
make cfg all tgt=rom ARCH=loongarch
CROSS_COMPILE=loongarch64-linux-gnu- DEBUG=-g
make dtb ARCH=loongarch CROSS_COMPILE=loongarch64-linux-gnu-
```

2. 内核编译

2.1 编译环境及依赖工具下载

编译环境：ubuntu20.04

```
sudo apt install libncurses5-dev
```

```
sudo apt install libssl-dev
```

2.2 指定交叉编译工具链

```
export PATH=/opt/loongarch_toolchain/bin:$PATH
```

2.3 设置内核配置文件

2K500 默认配置文件为 arch/loongarch/configs/loongson3_defconfig，需将其拷贝为.config 文件。

```
cp arch/loongarch/configs/loongson3_defconfig .config
```

2.4 打开图形化配置界面(可在此环节修改内核相关配置)

```
make menuconfig ARCH=loongarch
```

注：执行完步骤 2.2 之后，必须需执行步骤 2.3，弹出图形界面后，选择 <Exit>，然后保存退出。（默认配置文件中存在一些依赖配置选项，打开图形化界面并保存后，会自动勾选上这些依赖配置进行编译）

2.5 编译内核

```
make vmlinuz ARCH=loongarch CROSS_COMPILE=loongarch64-linux-gnu-
```

编译完成后，会在当前目录下看到生成的 vmlinuz 文件，与压缩后的内核文件 vmlinuz。

3. 文件系统制作

3.1 buildroot 编译

3.2 LoongOS 编译

1) u 盘在线更新 pmon

注：u 盘需在板卡上电之前插上。

准备一个 u 盘，u 盘最好在 linux 系统下进行格式化，且只有一个分区，分区号为 0，若分区为 fat 格式，则将 gzrom-dtb.bin 文件放入 u 盘，且在板卡上电之前插上 u 盘，输入下述命令进行更新 pmon。

```
load -rf 0x1c000000 (usb0,0)/gzrom-dtb.bin
```

(0x1c000000 是 flash 设备映射的地址，usb0 为 u 盘设备名，0 为 u 盘第 0 个分区)

或者输入：

```
load -rf 0x1c000000 /dev/fs/fat@usb0/gzrom-dtb.bin
```

如果 u 盘为 ext2/ext3/ext4 格式，则需将上述命令中 fat 替换成 ext2 进行加载。

```
PMON> load -rf 0x1c000000 (usb0,0)/gzrom-dtb.bin
-Loading file: (usb0,0)/gzrom-dtb.bin dl_offset 900000000f800000 addr 900000000f800000
(bin)
\
Loaded 1044980 bytes

Programming flash 900000000f800000:ff1f4 into 8000000001c000000
base[0x8000000001c000000], spi[0x80000000018000000],lio[0x8000000001a000000],lpc[0x8000000001df0
flash select: spi
Erase end!
base[0x8000000001c000000], spi[0x80000000018000000],lio[0x8000000001a000000],lpc[0x8000000001df0
flash select: spi
/Programming end!
base[0x8000000001c000000], spi[0x80000000018000000],lio[0x8000000001a000000],lpc[0x8000000001df0
flash select: spi
PMON>
```

2) 网口在线更新 pmon

注：在主机端需搭建并启动 tftp 服务。

```
ifaddr syn0 板卡 ip
```

```
load -rf 0x1c000000 tftp://主机 ip/gzrom-dtb.bin
```

(2) 内核烧写及启动

1) 写入到内存启动，掉电数据丢失

在 pmon 启动过程中，按“c”进入 pmon 命令行后，可通过下述命令加载内核到内存中，并跳转启动内核。

a. 通过 u 盘加载并启动内核

注：u 盘需在板卡上电之前插上。

u 盘要求同 4.1 章节中，<u 盘在线更新 pmon>小节一致。

```
load (usb0,0)/vmlinuz
```

```
initrd (usb0,0)/rootfs.cpio.gz
```

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

b. 通过网口加载并启动内核

注：在主机端需搭建并启动 tftp 服务。

```
ifaddr syn0 板卡 ip
```

```
load tftp://主机 ip/vmlinuz
```

```
initrd tftp://主机 ip/rootfs.cpio.gz
```

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

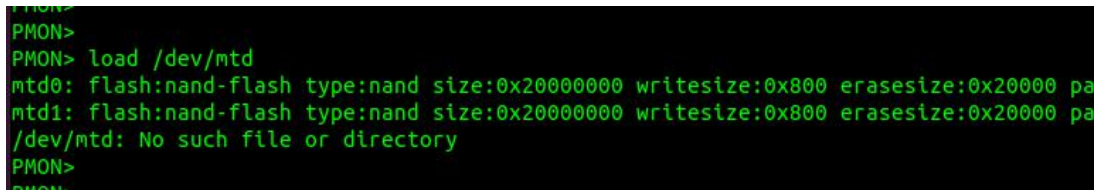
load 命令加载内核到内存中；initrd 命令加载文件系统到内存中；g 命令进行跳转，并指定日志从 ttyS0 输出，内核启动后运行的第一个进程为/sbin/init。如果内核未与文件系统编译在一起，则执行上述步骤即可。如果内核与文件系统编译在了一起，则 initrd 命令需去掉。

2) 写入 nand 启动，掉电不丢失

在 pmon 启动过程中，按“c”进入 pmon 命令行后，可通过下述命令将程序烧写到 nand 中，设置自启动命令自动跳转内核。

查看 pmon 下 nand 分区情况：

`load /dev/mtd`



```
PMON>
PMON> load /dev/mtd
mtd0: flash:nand-flash type:nand size:0x20000000 writesize:0x800 erasesize:0x20000 pa
mtd1: flash:nand-flash type:nand size:0x20000000 writesize:0x800 erasesize:0x20000 pa
/dev/mtd: No such file or directory
PMON>
PMON>
```

如上图，可见 nand 在 pmon 下分成了两个分区，则可在分区 0 中存放内核，分区 1 中存放 yaffs2 文件系统镜像，命令如下：

`mtd_erase /dev/mtd0`

`mtd_erase /dev/mtd1`

`devcp tftp://主机 ip/vmlinuz /dev/mtd0`

`devcp tftp://主机 ip/rootfs-yaffs2.img /dev/mtd1`

`set al /dev/mtd0`

`set append "console=ttyS0,115200 rw root=/dev/mtdblock1`

`rootfstype=yaffs2"`

4.2 ejtag 调试

(1) ejtag 烧写 pmon

在工作目录下解压 ejtag-debug.tar.gz 压缩包，会在当前目录下看到 ejtag-debug 目录。将编译好的 pmon 二进制文件“gzrom-dtb.bin”放入 ejtag-debug/目录下。

进入 ejtag-debug/目录后，在终端执行 `sudo ./la_dbg_tool_usb -t` 会进入 ejtag 命令行。

板卡上电后，输入：`jtagregs d8 1 1`，如果显示 `000000005a5a5a5a`，则表示 ejtag 连接成功。

```
cpu0 -
cpu0 -
cpu0 -jtagregs d8 1 1
00000001: 000000005a5a5a5a          ZZZZ....
cpu0 -
cpu0 -
```

连接成功后，在 ejtag 命令行按照下述流程进行烧写：

板卡断电，

`source configs/config.ls2k500`

`loop -1 stop`

板卡上电，在 ejtag 命令行下通过“ctrl + c”打断，再输入下一条命令，

`set`

此时，能看到 pc 值指向 `0x1c000000`，

`program_cachelock ./gzrom-dtb.bin`

```
#stop
^Cbreak!
cpu0 -break!
cpu0 -
cpu0 -
cpu0 -set
#set

zero:0x0          ra:0xdb00000000ffe00  gp:0xffffffffffffff  sp:0xffffffffdb0ffe00
a0:0x550          a1:0x9bce9          a2:0x80000000ff0018a  a3:0x9000000090004b24
a4:0x598138c70560155 a5:0x1          a6:0x1b318019b450189  a7:0xce59b50032094120
t0:0x3330004001110c95 t1:0x900000001c00a0c0 t2:0x900000001c00a600 t3:0xffffffffdb0ffe00
t4:0x6ffc0151      t5:0xbe8874c321815041 t6:0x0          t7:0x800000001ff40800
t8:0x900000001c00a0c0 tp:0x4d2874911d428300 fp:0x43d8451204104032 s0:0x900000000ef74840
s1:0x548          s2:0x548          s3:0x9bce9          s4:0x900000001c00a608
s5:0x548          s6:0xe           s7:0xe           s8:0x900000009003febf
pc:0x1c000000

csr0-crmd:0x8      csr1-prmd:0x0      csr2-cu :0x0
csr4-excfg:0x10000 csr5-exst:0x0      csr6-epc :0x0
csr8-badi:0x0      csrc-ebase:0x1c000000
cpu0 -
cpu0 -
cpu0 -
cpu0 -program_cachelock ./gzrom-dtb.bin
```

5. 接口配置与使用

5.1 GMAC

2K500-pai 上，网口 0 为靠近 USB3.0 侧的网口；网口 1 为靠近 VGA 侧的网口。pmon 命令行下通过 devls 命令可查看到网口设备名，分别为 syn0 与 syn1。pmon 与内核下网口默认已打开。

(1) pmon 下设置网口 ip

可通过 ifaddr 命令或者 ifconfig 命令在 pmon 命令行下设置网口 ip。

ifaddr syn0 板卡 ip

或者 ifconfig syn0 板卡 ip

(2) pmon 设备树中 GMAC 节点

Targets/LS2K500/conf/LS2K500.dts 中打开两个 gmac 网口的节点。

```
331     gmac0: ethernet@0x1f020000 {
332         compatible = "snps,dwmac-3.70a";
333         reg = <0 0x1f020000 0 0x10000>;
334         interrupt-parent = <&icu>;
335         interrupts = <12>;
336         interrupt-names = "macirq";
337         mac-address = [ 64 48 48 48 48 60 ];/* [>mac 64:48:48:48:48:60 <
338         phy-mode = "rgmii";
339         bus_id = <0x0>;
340         phy_addr = <0xffffffff>;
341         dma-mask = <0xffffffff 0xffffffff>;
342     };
```

```

344     gmac1: ethernet@0x1f030000 {
345         compatible = "snps,dwmac-3.70a";
346         reg = <0 0x1f030000 0 0x10000>;
347         interrupt-parent = <&icu>;
348         interrupts = <14>;
349         interrupt-names = "macirq";
350         mac-address = [ 64 48 48 48 48 61 ];/* [>mac 64:48:48:48:48:61 <
351         phy-mode = "rgmii";
352         bus_id = <0x1>;
353         phy_addr = <0xffffffff>;
354         dma-mask = <0xffffffff 0xffffffff>;
355     };
356
357     pci@0x16000000 {
358         compatible = "loongson,ls2k-pci";
Targets/LS2K500/conf/LS2K500.dts

```

(3) 内核 GMAC 驱动配置

CONFIG_STMMAC_ETH、CONFIG_STMMAC_PLATFORM。

(4) 板卡上网口使用的 phy 芯片为 YT8511

pmon 源码中 sys/dev/gmac/synopGMAC_network_interface.c 为网口驱动，驱动中 init_phy 函数有 YT8511 phy 芯片的初始化操作。如果更换了其他 phy 芯片，可在该函数内对应 phy 芯片的读写延时进行初始化。

5.2 VGA

(1) pmon 下配置分辨率

在配置文件 Targets/LS2K500/conf/ls2k500 中打开 1280x1024 分辨率显示配置。

```
124 select      mod_framebuffer
125 select      mod_vesa
126 option      CONFIG_VIDEO_16BPP
127 option      X1280x1024
128 #option     CONFIG_VIDEO_8BPP_INDEX
129 ### config for st7701s lcd
130 #select     st7701s
131 #option     FB_XSIZE=480
132 #option     FB_YSIZE=800
133 #option     CONFIG_VIDEO_32BPP
Targets/LS2K500/conf/ls2k500
```

(2) pmon 设备树中 DC 节点

Targets/LS2K500/conf/LS2K500.dts 中打开 dc 显示节点。

```
259 #if 1
260     dc@0x1f010000 {
261         compatible = "loongson,ls2k0500-dc";
262         #address-cells = <2>;
263         #size-cells = <2>;
264         ranges = <0 0x10000000 0 0x10000000 0 0x10000000
265                 0 0x20000000 0 0x20000000 0 0x20000000>;
266
267         reg = <0 0x1f010000 0 0x10000
268                 0 0x20000000 0 0x20000000>;
269         interrupt-parent = <&extioic>;
270         interrupts = <80>;
271
272         dc_io@0x1fe10000 {
273             compatible = "loongson,ls2k-dc_io";
274             reg = <0 0x1fe10000 0 0x40>;
275         };
276
277         pix0_pll@0x1fe10418 {
278             compatible = "loongson,ls2k-pix0_pll";
279             offset = <0x0418>;
280         };
281
282         pix1_pll@0x1fe10420 {
283             compatible = "loongson,ls2k-pix1_pll";
284             offset = <0x0420>;
285         };
286
287         dc0_i2c: pixi2c@0x1ff4a000 {
288             compatible = "loongson,ls2k-i2c";
289             reg = <0 0x1ff4a000 0 0x0800>;
290             interrupt-parent = <&extioic>;
291             interrupts = <18>;
292         };
293
294         dc1_i2c: pixi2c@0x1ff4a800 {
295             compatible = "loongson,ls2k-i2c";
296             reg = <0 0x1ff4a800 0 0x0800>;
297             interrupt-parent = <&extioic>;
298             interrupts = <19>;
299         };
300     };
301 #endif
Targets/LS2K500/conf/LS2K500.dts
```

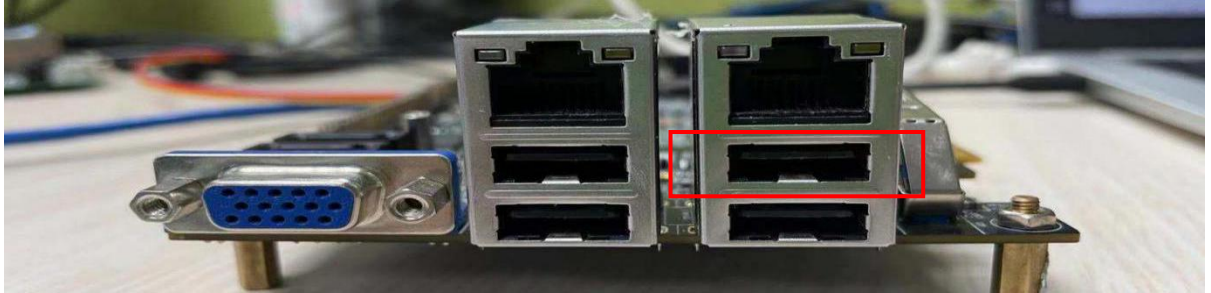
(3) 内核 DC 驱动配置

CONFIG_DRM_LOONGSON_VGA。

5.3 OTG

pmon 下无 otg 驱动，接口不可使用。内核下可设置主/从模式分别使用。

otg 为板卡上红框圈出来的接口。



(1) pmon 设备树中 OTG 节点

设备树 Targets/LS2K500/conf/LS2K500.dts 里需打开 otg 节点。

```
663     otg@0x1f080000 {
664         compatible = "loongson,dwc-otg", "dwc-otg";
665         reg = <0 0x1f080000 0 0x40000>;
666         interrupt-parent = <&extioic>;
667         interrupts = <73>;
668         dma-mask = <0x0 0xffffffff>;
669     };
670
```

(2) 内核下 OTG 主模式配置

`CONFIG_USB_DWC2_HOST`。

(3) 内核下 OTG 从模式配置

(示例中模拟成打印设备，otg 从模式同时只可作为一种从设备)

打开下述配置：

`CONFIG_USB_OTG_WHITELIST`、

`CONFIG_USB_DWC2_PERIPHERAL`、`CONFIG_USB_G_PRINTER`；

关闭下述配置：

`CONFIG_USB_CONFIGFS`。

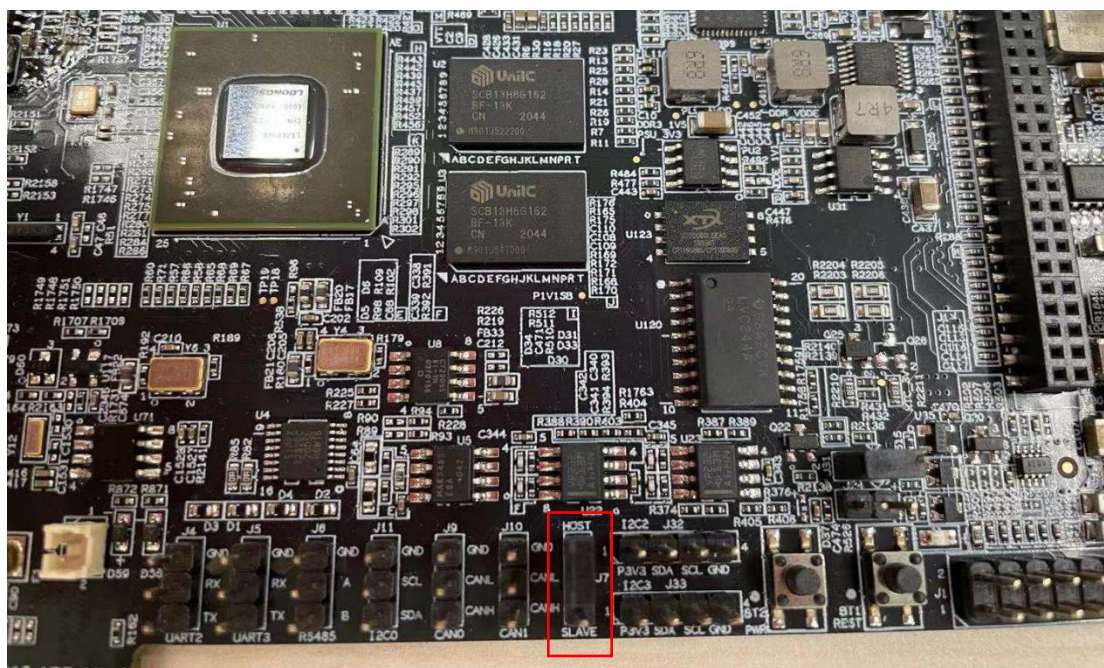
(4) 内核下 OTG 双模式配置

`CONFIG_USB_OTG_WHITELIST`、

`CONFIG_USB_DWC2_DUAL_ROLE`、`CONFIG_USB_G_PRINTER`。

此时，需要硬件跳线设置 ID 脚电平值，来区分主从模式。（ID 脚拉低设置为主模式；ID 脚拉高设置为从模式）

can1 右侧的一排插针 J7，短接上面两个引脚为 HOST 主模式，短接下面两个引脚为 SLAVE 从模式。（下图中短接成了 HOST 模式）



5.4 USB2.0

(1) OHCI (低速/全速设备)

a. pmon 设备树中 OHCI 节点

设备树 Targets/LS2K500/conf/LS2K500.dts 里需打开 ohci 节点。

```
630     ohci@0x1f058000 {  
631         compatible = "loongson,ls2k-ohci", "generic-ohci";  
632         reg = <0 0x1f058000 0 0x8000>;  
633         interrupt-parent = <&extioic>;  
634         interrupts = <72>;  
635         dma-mask = <0x0 0xffffffff>;  
636     };
```

b. 内核下 OHCI 控制器驱动配置

[CONFIG_USB_OHCI_HCD_PLATFORM](#)。

(2) EHCI (高速设备)

a. pmon 设备树中 EHCI 节点

设备树 Targets/LS2K500/conf/LS2K500.dts 里需打开 ehci 节点。

```
638     ehci@0x1f050000 {  
639         compatible = "loongson,ls2k-ehci", "generic-ehci";  
640         reg = <0 0x1f050000 0 0x8000>;  
641         interrupt-parent = <&extioic>;  
642         interrupts = <71>;  
643         dma-mask = <0xffffffff 0xffffffff>;  
644     };
```

b. 内核下 EHCI 控制器驱动配置

[CONFIG_USB_EHCI_HCD_PLATFORM](#)。

5.5 USB3.0

(1) pmon 设备树中 XHCI 节点

设备树 Targets/LS2K500/conf/LS2K500.dts 里需打开 xhci 节点。

```
646     usb2_phy: usb2phy@xhci {
647         compatible = "usb-dummy-phy";
648     };
649
650     usb3_phy: usb3phy@xhci {
651         compatible = "usb-dummy-phy";
652     };
653
654     xhci@0x1f060000 {
655         compatible = "synopsys,dwc3";
656         reg = <0 0x1f060000 0 0x10000>;
657         interrupt-parent = <&extioic>;
658         interrupts = <74>;
659         dma-mask = <0x0 0xffffffff>;
660         usb-phy = <&usb2_phy>, <&usb3_phy>;
661     };
Targets/LS2K500/conf/LS2K500.dts
```

(2) 内核下 DWC3 主模式配置

CONFIG_USB_XHCI_PLATFORM、CONFIG_USB_DWC3_HOST。

(3) 内核下 DWC3 从模式配置

打开下述配置：

CONFIG_USB_XHCI_PLATFORM、CONFIG_USB_DWC3_GADGET、

CONFIG_USB_G_PRINTER、CONFIG_USB_OTG_WHITELIST。

关闭下述配置：

CONFIG_USB_CONFIGFS。

5.6 PCIE

(1) PCIE0 EP 模式使用示例

1) RC 端 pmon 下，查看 PCIE EP 设备信息

```
>> BUS 1 <<
Dev Fun Device description
-----
0 0 vendor/product: 0x0014/0x1a05 (bridge, host, interface: 0x00, revision: 0x01)
0x500000008:0xf0000008 mem @0x50000000, 268435456 bytes
>> BUS 0 <<
```

读取设备内的数据：

```
PMON>
PMON> pcs -1
select normal memory access (64 bit uncached physcal address)
PMON>
PMON> d4 0x50000000 32
9000000050000000: 50000000 0400c02c 0400c42d 141e128c ...P,...-.....
9000000050000010: 03b5a18c 1600000c 0324018c 4c000180 .....$....L
9000000050000020: 77777777 77777777 88888888 88888888 wwwwwwwww.....
9000000050000030: 11111111 11111111 eeeeeeee eeeeeeee .....
9000000050000040: 00000000 ffffffff 00000000 ffffffff .....
9000000050000050: 00000000 ffffffff 00000000 ffffffff .....
9000000050000060: 00000000 ffffffff 00000000 ffffffff .....
9000000050000070: 00000000 ffffffff 00000000 ffffffff .....
PMON>
```

2) EP 端修改内存里的数据值

```
PMON>
PMON> d4 0x00000000 32
8000000000000000 : 50000000 0400c02c 0400c42d 141e128c ...P,...-.....
8000000000000010 : 03b5a18c 1600000c 0324018c 4c000180 .....$....L
8000000000000020 : 77777777 77777777 88888888 88888888 wwwwwwwww.....
8000000000000030 : 11111111 11111111 eeeeeeee eeeeeeee .....
8000000000000040 : 00000000 ffffffff 00000000 ffffffff .....
8000000000000050 : 00000000 ffffffff 00000000 ffffffff .....
8000000000000060 : 00000000 ffffffff 00000000 ffffffff .....
8000000000000070 : 00000000 ffffffff 00000000 ffffffff .....
PMON>
PMON> m4 0x00000000 0x11111111
PMON> m4 0x00000004 0x22222222
PMON> m4 0x00000008 0x33333333
PMON> m4 0x0000000c 0x44444444
PMON> m4 0x00000010 0xaaaaaaaa
PMON> m4 0x00000024 0xbbbbbbbb
PMON> m4 0x00000038 0xcccccccc
PMON> m4 0x0000004c 0xdddddddd
PMON>
PMON>
PMON> d4 0x00000000 32
8000000000000000 : 11111111 22222222 33333333 44444444 .... " " "3333DDDD
8000000000000010 : aaaaaaaaa 1600000c 0324018c 4c000180 .....$....L
8000000000000020 : 77777777 bbbbbbbb 88888888 88888888 wwwwww.....
8000000000000030 : 11111111 11111111 cccccccc eeeeeeee .....
8000000000000040 : 00000000 ffffffff 00000000 dddddddd .....
8000000000000050 : 00000000 ffffffff 00000000 ffffffff .....
```

3) RC 端再次读取 EP 设备寄存器值

```
PMON>
PMON> d4 0x50000000 32
9000000050000000: 11111111 22222222 33333333 44444444 ...."3333DDDD
9000000050000010: aaaaaaaa 1600000c 0324018c 4c000180 .....$.L
9000000050000020: 77777777 bbbbbbbb 88888888 88888888 www.....
9000000050000030: 11111111 11111111 cccccccc eeeeeeee .....
9000000050000040: 00000000 ffffffff 00000000 dddddddd .....
9000000050000050: 00000000 ffffffff 00000000 ffffffff .....
9000000050000060: 00000000 ffffffff 00000000 ffffffff .....
9000000050000070: 00000000 ffffffff 00000000 ffffffff .....
PMON>
```

(2) PCIE RC 模式

1) pmon 设备树中 PCIE 节点

设备树 Targets/LS2K500/conf/LS2K500.dts 里需打开 pcie 控制器节点。

```
292     pci@0x16000000 {
293         compatible = "loongson,ls2k-pci";
294         #interrupt-cells = <1>;
295         bus-range = <0x1 0x6>;
296         #size-cells = <2>;
297         #address-cells = <3>;
298         linux,pci-domain = <1>;
299
300         reg = < 0xfe 0x00000000 0 0x10000000>;
301         ranges = <0x02000000 0 0x40000000 0 0x40000000 0 0x40000000
302                 0x01000000 0 0x00004000 0 0x16404000 0x0 0x4000>;
303
304         pci_bridge@0,0 {
305             compatible = "pciclass060400",
306                         "pciclass0604";
307
308             reg = <0x0000 0x0 0x0 0x0 0x0>;
309             interrupts = <81>;
310             interrupt-parent = <&extioic>;
311
312             #interrupt-cells = <1>;
313             interrupt-map-mask = <0 0 0 0>;
314             interrupt-map = <0 0 0 0 &extioic 81>;
315         };
316         pci_bridge@1,0 {
317             compatible = "pciclass060400",
318                         "pciclass0604";
319
320             reg = <0x0800 0x0 0x0 0x0 0x0>;
321             interrupts = <82>;
322             interrupt-parent = <&extioic>;
323
324             #interrupt-cells = <1>;
325             interrupt-map-mask = <0 0 0 0>;
326             interrupt-map = <0 0 0 0 &extioic 82>;
327         };
328     };
```


5.7 PCI

(1) PCI 复用设置

PCI 与 PRINT、LIO、PWM2 存在复用冲突。复用注意事项可查看附录“2K500 复用设置注意事项”。PCI 复用需要按下述信息进行配置，如果不使用 PCI，则需关闭下述 PCI 选项。

gpio100~148，设置为芯片主功能；

gpio86，设置为 gpio 功能，用作 pci 中断引脚。

在 pmon 源码下的 Targets/LS2K500/ls2k500/pincfgs.c 文件中的 default_pin_cfgs 数组里，配置的是板卡 pmon 默认的 gpio 复用关系，由上，需将 gpio100~148 设置为芯片主功能，gpio86 设置为 gpio 功能。

```
105 {100, 5}, //5:pci_ad[0]      2:ltoa[0]      1:pr_int      (p
106 {101, 5}, //5:pci_ad[1]      2:ltoa[1]      1:pr0_clk
107 {102, 5}, //5:pci_ad[2]      2:ltoa[2]      1:pr0_start
108 {103, 5}, //5:pci_ad[3]      2:ltoa[3]      1:pr0_ready
109 {104, 5}, //5:pci_ad[4]      2:ltoa[4]      1:pr0_enable
110 {105, 5}, //5:pci_ad[5]      2:ltoa[5]      1:pr0_shold
111 {106, 5}, //5:pci_ad[6]      2:ltoa[6]      1:pr0_data
112 {107, 5}, //5:pci_ad[7]      2:ltoa[7]      1:pr0_hsync
113 {108, 5}, //5:pci_ad[8]      2:ltoa[8]      1:pr1_enable
114 {109, 5}, //5:pci_ad[9]      2:ltoa[9]      1:pr1_shold
115 {110, 5}, //5:pci_ad[10]     2:ltoa[10]     1:pr1_data
116 {111, 5}, //5:pci_ad[11]     2:ltoa[11]     1:pr2_clk
117 {112, 5}, //5:pci_ad[12]     2:ltoa[12]     1:pr2_start
118 {113, 5}, //5:pci_ad[13]     2:ltoa[13]     1:pr2_ready
119 {114, 5}, //5:pci_ad[14]     2:ltoa[14]     1:pr2_enable
120 {115, 5}, //5:pci_ad[15]     2:ltoa[15]     1:pr2_shold
121 {116, 5}, //5:pci_ad[16]     2:ltoa_data[0] 1:pr2_data
122 {117, 5}, //5:pci_ad[17]     2:ltoa_data[1] 1:pr2_hsync
123 {118, 5}, //5:pci_ad[18]     2:ltoa_data[2] 1:pr3_enable
124 {119, 5}, //5:pci_ad[19]     2:ltoa_data[3] 1:pr3_shold
125 {120, 5}, //5:pci_ad[20]     2:ltoa_data[4] 1:pr3_data
126 {121, 5}, //5:pci_ad[21]     2:ltoa_data[5] 1:pr4_clk
127 {122, 5}, //5:pci_ad[22]     2:ltoa_data[6] 1:pr4_start
128 {123, 5}, //5:pci_ad[23]     2:ltoa_data[7] 1:pr4_ready
129 {124, 5}, //5:pci_ad[24]     2:ltoa_data[8] 1:pr4_enable
130 {125, 5}, //5:pci_ad[25]     2:ltoa_data[9] 1:pr4_shold
131 {126, 5}, //5:pci_ad[26]     2:ltoa_data[10] 1:pr4_data
132 {127, 5}, //5:pci_ad[27]     2:ltoa_data[11] 1:pr4_hsync
Targets/LS2K500/ls2k500/pincfgs.c [+]
```

```

132 {127, 5}, //5:pci_ad[27] 2:lio_data[11] 1:pr4_hsync
133 {128, 5}, //5:pci_ad[28] 2:lio_data[12] 1:pr5_enable
134 {129, 5}, //5:pci_ad[29] 2:lio_data[13] 1:pr5_shold
135 {130, 5}, //5:pci_ad[30] 2:lio_data[14] 1:pr5_data
136 {131, 5}, //5:pci_ad[31] 2:lio_data[15] 1:pr6_clk
137 {132, 5}, //5:pci_cbe[0] 2:lioa[16] 1:pr6_start
138 {133, 5}, //5:pci_cbe[1] 2:lioa[17] 1:pr6_ready
139 {134, 5}, //5:pci_cbe[2] 2:lioa[18] 1:pr6_enable
140 {135, 5}, //5:pci_cbe[3] 2:lioa[19] 1:pr6_shold
141 {136, 5}, //5:pci_frame 2:lioa[20] 1:pr6_data
142 {137, 5}, //5:pci_irdy 2:lioa[21] 1:pr6_hsync
143 {138, 5}, //5:pci_devsel 2:lioa[22] 1:pr7_enable
144 {139, 5}, //5:pci_trdy 2:liocsn[0] 1:pr7_shold
145 {140, 5}, //5:pci_stop 2:liocsn[1] 1:pr7_data
146 {141, 5}, //5:pci_idsel 2:liowrn
147 {142, 5}, //5:pci_par 2:liordn
148 {143, 5}, //5:pci_perr 4:sdio1_clk (module)
149 {144, 5}, //5:pci_serr 4:sdio1_cmd (module)
150 {145, 5}, //5:pci_req[0] 4:sdio1_d[0](module)
151 {146, 5}, //5:pci_req[1] 4:sdio1_d[1](module)
152 {147, 5}, //5:pci_gnt[0] 4:sdio1_d[2](module)
153 {148, 5}, //5:pci_gnt[1] 4:sdio1_d[3](module)
Targets/LS2K500/ls2k500/pincfgs.c [+]

```

```

88 {83, 5}, //5:pmu0[0] 0:gpio83(module)
89 {84, 5}, //5:pwm[0](pai) 0:gpio84(module)
90 {85, 5}, //5:pwm[1](pai) 0:gpio85(module)
91 {86, 0}, //5:pwm[2](pai) 0:gpio86(module)
92 {87, 5}, //5:pwm[3](pai) 0:gpio87(module)
93 {88, 5}, //5:gmco rx ctl
Targets/LS2K500/ls2k500/pincfgs.c [+]

```

(2) pmon 下 PCI 配置选项

在配置文件 Targets/LS2K500/conf/ls2k500 中打开 PCI 配置项：

```

194 lxhci0 at localbus0 base 0x800000001f060000 # XHCI
195 lxhci0 at lxhci0
196 usb* at lxhci0
197 select mod_usb lxhci
198 option LS2K500 HAVE_PCI
199
200 ##### USB
201 #lxhci* at pci? dev ? function ?
202
203 ##### Networking Devices
Targets/LS2K500/conf/ls2k500 [+]

```

(3) pmon 设备树中 PCI 节点

Targets/LS2K500/conf/LS2K500.dts 中打开 PCI 节点。

```

329 #if 1
330     ls2k500pci@0x17100000 {
331         compatible = "loongson,ls2k500-pci";
332         #interrupt-cells = <1>;
333         bus-range = <0x10 0x14>;
334         #size-cells = <2>;
335         #address-cells = <3>;
336         pci-gpios = <&pioB 22 0>;
337         linux,pci-domain = <2>;
338
339         reg = < 0x0 0x17100000 0 0x10000
340                0x0 0x17110000 0 0x10000
341                0x0 0x1fe11100 0 0x100 >;
342         ranges = <0x02000000 0 0x20000000 0 0x20000000 0 0x10000000
343                0x01000000 0 0x00008000 0 0x17008000 0x0 0x4000>;
344
345 #if 0
346         pci_bridge@0,0 {
347             compatible = "pciclass060400",
348                         "pciclass0604";
349
350             reg = <0x0000 0x0 0x0 0x0 0x0>;
351             interrupts = <81>;
352             interrupt-parent = <&extioic>;
353
354             #interrupt-cells = <1>;
355             interrupt-map-mask = <0 0 0 0>;
356             interrupt-map = <0 0 0 0 &extioic 81>;
357         };
358 #endif
359     };
360 #endif
Targets/LS2K500/conf/LS2K500.dts

```

5.8 SDIO

2K500-pai 上，使用的 SDIO 接口为 SDIO0。

(1) SDIO0 复用设置

复用注意事项可查看附录“2K500 复用设置注意事项”。

gpio149~154，设置为芯片主功能（SDIO0）。

在 pmon 源码下的 Targets/LS2K500/ls2k500/pincfgs.c 文件中的 default_pin_cfgs 数组里，配置的是板卡 pmon 默认的 gpio 复用关系，由上，需将 gpio149~154 设置为芯片主功能。

```
154 {149, 5}, //5:sdio_clk
155 {150, 5}, //5:sdio_cmd
156 {151, 5}, //5:sdio_d[0]
157 {152, 5}, //5:sdio_d[1]
158 {153, 5}, //5:sdio_d[2]
159 {154, 5}, //5:sdio_d[3]
160 };
161
162 /* add all pins that you want to cfg. just like this, then call cfg all
Targets/LS2K500/ls2k500/pincfgs.c [+]
```

(2) pmon 设备树中 SDIO0 节点

Targets/LS2K500/conf/LS2K500.dts 中打开 SDIO0 节点。

```
471 sdio0@0x1ff64000 {
472     compatible = "loongson,ls2k_sdio";
473     reg = <0 0x1ff64000 0 0x1000>;
474     interrupt-parent = <&extioic>;
475     interrupts = <57>;
476     interrupt-names = "ls2k_mci_irq";
477
478     dmas = <&dma3 1>;
479     dma-names = "sdio_rw";
480     dma-mask = <0xffffffff 0xffffffff>;
481
482     cd-gpios = <&pioA 44 0>;
483 };
```

(3) 内核下 SDIO 相关配置项

CONFIG_MMC_BLOCK、CONFIG_MMC_LS2K、
CONFIG_LS_APBDMAC、CONFIG_DMADEVICES。

(4) 内核启动日志

```
Btrfs loaded, crc32c=crc32c-generic
hctosys: unable to open rtc device (rtc0)
usb-storage 1-1:1.0: USB Mass Storage device detected
scsi host2: usb-storage 1-1:1.0
mmc0: new SD card at address 21cf
loongson3_acpi_cpufreq: Boost capabilities not present in the processor
ALSA device list:
  No soundcards found.
mmcblk0: mmc0:21cf XTSDA 990 MiB
Warning: unable to open an initial console.
mmcblk0: p1
Freeing unused kernel memory: 31904K
```

由上图可见，内核驱动检测到了 sdio 设备，并在/dev/目录下创建了设备节点 mmcblk0，且在设备下存在 p1 分区，分区设备名为/dev/目录下的 mmcblk0p0。

5.9 NAND

NAND 在 pmon 及内核下，设置的 ecc 校验模式，设置的分区大小情况，需严格保持一致。

(1) NAND 复用设置

NAND 与 CAN0、CAN1 、I2C0、I2C2、I2C3 以及 SPI0、SPI1 的片选引脚都存在复用冲突。复用注意事项可查看附录“2K500 复用设置注意事项”。

gpio64-75，设置成第 1 复用；

gpio76-83，设置成芯片主功能。

在 pmon 源码下的 Targets/LS2K500/ls2k500/pincfgs.c 文件中的 default_pin_cfgs 数组里，配置的是板卡 pmon 默认的 gpio 复用关系，由上，需将 gpio64~75 设置为第 1 复用，将 gpio76~83 设置为芯片主功能。

```
69 { 64, 1}, //5:scl0 1:nand_rdy[1] 3:spi0_cs[3]
70 { 65, 1}, //5:sda0 1:nand_ce[1] 3:spi0_cs[2]
71 { 66, 1}, //5:can0_rx 1:nand_rdy[2] 2:sda2
72 { 67, 1}, //5:can0_tx 1:nand_ce[2] 2:scl2
73 { 68, 1}, //5:can1_rx 1:nand_rdy[3] 2:sda3
74 { 69, 1}, //5:can1_tx 1:nand_ce[3] 2:scl3
75 { 70, 1}, //5:lpc_ad[0] 1:nand_d[0]
76 { 71, 1}, //5:lpc_ad[1] 1:nand_d[1]
77 { 72, 1}, //5:lpc_ad[2] 1:nand_d[2]
78 { 73, 1}, //5:lpc_ad[3] 1:nand_d[3]
79 { 74, 1}, //5:lpc_frame 1:nand_d[4]
80 { 75, 1}, //5:lpc_serirq 1:nand_d[5]
81 { 76, 5}, //5:nand_cle 4:pwm0(module)
82 { 77, 5}, //5:nand_ale 4:pwm1(module)
83 { 78, 5}, //5:nand_rd 4:pwm2(module)
84 { 79, 5}, //5:nand_wr 4:pwm3(module)
85 { 80, 5}, //5:nand_ce[0] 4:pwm4(module)
86 { 81, 5}, //5:nand_rdy[0] 4:pwm5(module)
87 { 82, 5}, //5:nand_d[6] 4:pwm6(module)
88 { 83, 5}, //5:nand_d[7] 0:gpio83(module)
89 { 84, 5}, //5:pwm[0](pai) 0:gpio84(module)
Targets/LS2K500/ls2k500/pincfgs.c
```

(2) pmon 下 NAND 配置选项

1) nand 配置选项

在配置文件 Targets/LS2K500/conf/ls2k500 中打开下述选项：

`select nand`

`option CONFIG_LS2K500_NAND`

2) nand bch 配置选项

在配置文件 Targets/LS2K500/conf/ls2k500 中打开下述选项：

`select nand_bch`

```
282 select      http
283 select      tcp
284 select      inet
285 select      nand
286 select      nand_bch
287 select cmd_xyzmodem
288 option HPET_RTC
289 option PCIVERBOSE=5
290 option PCI_INT_GPIO=86
291 option      USB3_USE_INTER_CLK
292 option      CONFIG_LS2K500_NAND
293 #select      m25p80
294 #option      DDR_RESET_REVERT
295 #option      CPU_DEBUG_UART0
Targets/LS2K500/conf/ls2k500 [+]
```

pmon 源码下 sys/dev/nand/ls2k500-nand.c 为 NAND 的驱动文件，如果在配置文件中关闭了 nand_bch 配置，则会在驱动中设置校验模式为 ecc 软件校验。

```
645     }
646 #else
647     this->ecc.mode      = NAND_ECC_SOFT;
648     this->ecc.size       = 256;
649     this->ecc.bytes      = 3;
650     this->ecc.hwctl      = ls2k500_nand_ecc_hwctl;
651     this->ecc.calculate  = ls2k500_nand_ecc_calculate;
652     this->ecc.correct    = ls2k500_nand_ecc_correct;
653 #endif
654 }
sys/dev/nand/ls2k500-nand.c
```

3) NAND 分区设置

在源码下 Targets/LS2K500/include/pmon_target.h 中修改 pmon 下 NAND 的默认分区：mtdparts 中默认第一个分区为 50M，剩下的空间分给第二个分区。可在此修改 pmon 下的分区设置。

```
68 #if NNAND
69 #define TGT_DEFENV {"mtdparts","nand-flash:50M@0(kernel)ro,-(rootfs)",0,&mtd_rescan}, \
70 {"bootdelay","3",0,0}
71 #else
72 #define TGT_DEFENV {"bootdelay","3",0,0}
73 #endif
Targets/LS2K500/include/pmon_target.h
```

(3) pmon 设备树中 NAND 节点及分区设置

设备树的 nand 节点里，也会设置 nand 分区，但这个是为提供内核驱动进行 nand 分区设置的，上面则是用于 pmon 下的 nand 分区设置。

注：内核下（即设备树中）nand 分区设置需与 pmon 下保持一致。

由下图可知，如果配置文件 Targets/LS2K500/conf/ls2k500 中打开了 CONFIG_LS2K500_NAND 配置项，那么在 pmon 下的设备树文件 Targets/LS2K500/conf/LS2K500.dts 里，会将 nand 节点打开，通过设置设备树里的 nand 分区信息节点“partition”，来进行内核下的 nand 分区配置。

如下图，“number-of-parts=<0x2>”，表示设备树一共传递两个分区配置。“partition@0”中的“reg”设置，表示第一个分区偏移为 0，分区大小为 50M；“partition@0x03200000”中的“reg”设置，表示第二个分区偏移为 50M，分区大小为 nand 剩下的空间。


```

714 #ifdef CONFIG_LS2K500_NAND
715     nand@0x1ff58040 {
716         compatible = "loongson,ls-nand";
717         reg = <0 0x1ff58040 0 0x0
718             0 0x1ff58000 0 0x20>;
719         interrupt-parent = <&extioic>;
720         interrupts = <31>;
721         interrupt-names = "nand_irq";
722
723         dmas = <&dma0 1>;
724         dma-names = "nand_rw";
725         dma-mask = <0xffffffff 0xffffffff>;
726
727         number-of-parts = <0x2>;
728         nand-cs = <0x0>;
729
730         partition@0 {
731             label = "kernel_partition";
732             reg = <0 0x00000000 0 0x03200000>;
733         };
734
735         partition@0x03200000 {
736             label = "os_partition";
737             reg = <0 0x03200000 0 0x0>;
738         };
739     };
740 #else
741 #if 0
742     i2c2: i2c@0x1ff49000 {
743         compatible = "loongson,ls2k-i2c";
744         reg = <0 0x1ff49000 0 0x0800>;
745     };
746 #endif
747 #endif

```

Targets/LS2K500/conf/LS2K500.dts

(4) 内核下 NAND 驱动配置

下述配置 NAND 为 bch 校验模式：

注：内核下 nand 校验模式需与 pmon 下保持一致。

CONFIG_MTD_OF_PARTS、CONFIG_MTD_BLOCK、

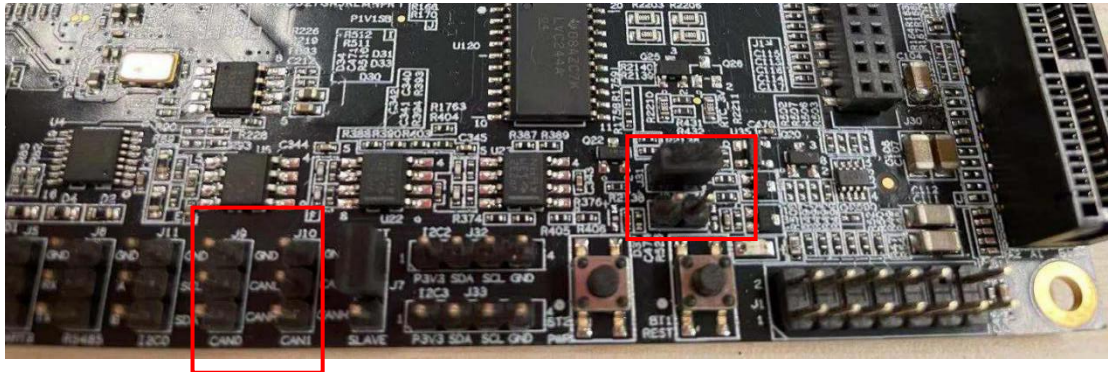
CONFIG_MTD_NAND_LS、CONFIG_MTD_NAND_ECC_BCH。

(5) 内核下 yaffs2 文件系统格式支持

CONFIG_YAFFS_FS。

5.10 CAN

2K500-pai 上有 CAN0 与 CAN1 两个 CAN 接口。当 J31 插上跳线帽，则此时板卡设置为 CAN 功能，可使用 CAN0 与 CAN1；当 J31 取下跳线帽，此时板卡使用 I2C2 与 I2C3。



(1) CAN 复用设置

CAN0、CAN1 与 NAND、I2C2、I2C3 以及 SPI0、SPI1 的片选引脚都存在复用冲突。复用注意事项可查看附录“2K500 复用设置注意事项”。

gpio66~67，设置为芯片主功能（CAN0）；

gpio68~69，设置为芯片主功能（CAN1）。

在 pmon 源码下的 Targets/LS2K500/ls2k500/pincfgs.c 文件中的 default_pin_cfgs 数组里，配置的是板卡 pmon 默认的 gpio 复用关系，由上，需将 gpio66~69 设置为芯片主功能。

```
70 { 65, 1}, //5:sda0 1:nand_ce[1] 3:spi0
71 { 66, 5}, //5:can0_rx 1:nand_rdy[2] 2:sda2
72 { 67, 5}, //5:can0_tx 1:nand_ce[2] 2:scl2
73 { 68, 5}, //5:can1_rx 1:nand_rdy[3] 2:sda3
74 { 69, 5}, //5:can1_tx 1:nand_ce[3] 2:scl3
75 { 70, 1}, //5:lpc_ad[0] 1:nand_d[0]
```

Targets/LS2K500/ls2k500/pincfgs.c [+]

(2) pmon 设备树中 CAN 节点

在 pmon 的配置文件 Targets/LS2K500/conf/ls2k500 中关闭配置项 CONFIG_2K500_NAND。

```
290 option PCI_INT_GPIO=86
291 option      USB3_USE_INTER_CLK
292 #option      CONFIG_LS2K500_NAND
293 #select      m25p80
294 #option      DDR_RESET_REVERT
295 option      HS0636
296 #option      CONFIG_DDR_16BIT
Targets/LS2K500/conf/ls2k500 [+]
```

并在设备树 Targets/LS2K500/conf/LS2K500.dts 里按下图设置“if 0”来打开 can0 与 can1 节点。

```
739 },
740 #else
741 #if 0
742     i2c2: i2c@0x1ff49000 {
743         compatible = "loongson,ls2k-i2c";
744         reg = <0 0x1ff49000 0 0x0800>;
745         interrupt-parent = <&extioiic>;
746         interrupts = <16>;
747         //status = "disabled";
748     };
749
750     i2c3: i2c@0x1ff49800 {
751         compatible = "loongson,ls2k-i2c";
752         reg = <0 0x1ff49800 0 0x0800>;
753         interrupt-parent = <&extioiic>;
754         interrupts = <17>;
755         //status = "disabled";
756     };
757 #else
758     can0: can@1ff44000 {
759         compatible = "nxp,sja1000";
760         reg = <0 0x1ff44000 0 0x1000>;
761         nxp,external-clock-frequency = <100000000>;
762         interrupt-parent = <&extioiic>;
763         interrupts = <10>;
764         //status = "disabled";
765     };
766
767     can1: can@1ff45000 {
768         compatible = "nxp,sja1000";
769         reg = <0 0x1ff45000 0 0x1000>;
770         nxp,external-clock-frequency = <100000000>;
771         interrupt-parent = <&extioiic>;
772         interrupts = <11>;
773         //status = "disabled";
774     };
775 #endif
776 #endif
Targets/LS2K500/conf/LS2K500.dts
```

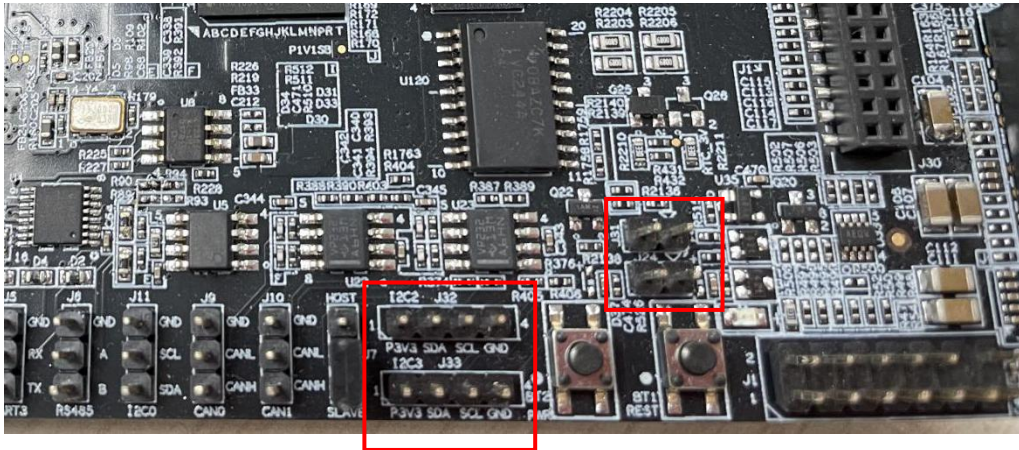
注 :can0 和 can1 节点内 ,status = “disable”;项需按照上图内容注释掉 ,
如果打开 ,内核将不会初始化这一个设备节点。

(3) 内核下 CAN 驱动配置

CONFIG_CAN_SJA1000_PLATFORM。

5.11 I2C

2K500-pai 上共有 I2C0、I2C2、I2C3、I2C4、I2C5 五路 i2c。当 J31 上跳线帽被取下时，此时板卡上 I2C2 与 I2C3 可用。其中 J32 为 I2C2，J33 为 I2C3。



(1) I2C 复用设置

I2C0 与 NAND、SPI0 的片选存在复用冲突；I2C2、I2C3 与 NAND、CAN0、CAN1 以及 SPI0、SPI1 的片选引脚都存在复用冲突。复用注意事项可查看附录“2K500 复用设置注意事项”。

gpio64-65，设置为芯片主功能（I2C0）；

gpio66-67，设置为第 2 复用（I2C2）；

gpio68-69，设置为第 2 复用（I2C3）；

gpio36-37，设置为第 3 复用（I2C4(PIX0)）；

gpio38-39，设置为第 3 复用（I2C5(PIX1)）。

在 pmon 源码下的 Targets/LS2K500/ls2k500/pincfgs.c 文件中的 default_pin_cfgs 数组里，配置的是板卡 pmon 默认的 gpio 复用关系，由上，

需按照下图设置 pmon 下的复用关系。

I2C0、I2C2、I2C3 复用设置如下图：

```
69 { 64, 5}, //5:scl0 1:nand_rdy[1] 3:spi0_cs[
70 { 65, 5}, //5:sda0 1:nand_ce[1] 3:spi0_cs[
71 { 66, 2}, //5:can0_rx 1:nand_rdy[2] 2:sda2
72 { 67, 2}, //5:can0_tx 1:nand_ce[2] 2:scl2
73 { 68, 2}, //5:can1_rx 1:nand_rdy[3] 2:sda3
74 { 69, 2}, //5:can1_tx 1:nand_ce[3] 2:scl3
75 { 70, 1}, //5:lpc_ad[0] 1:nand_d[0]
```

Targets/LS2K500/ls2k500/pincfgs.c [+]

I2C4、I2C5 复用设置如下图：

```
41 { 36, 3}, //5:ac97_datai(module) 3:pix0_scl (pai)
42 { 37, 3}, //5:ac97_datao(module) 3:pix0_sda (pai)
43 { 38, 3}, //5:ac97_sync (module) 3:pix1_scl (pai)
44 { 39, 3}, //5:ac97_reset(module) 3:pix1_sda (pai)
```

(2) pmon 设备树中 I2C 节点

在 pmon 的配置文件 Targets/LS2K500/conf/ls2k500 中关闭配置项 CONFIG_2K500_NAND。

```
290 option PCI_INT_GPIO=86
291 option USB3_USE_INTER_CLK
292 #option CONFIG_LS2K500_NAND
293 #select m25p80
294 #option DDR_RESET_REVERT
295 option HS0636
296 #option CONFIG_DDR_16BIT
```

Targets/LS2K500/conf/ls2k500 [+]

在设备树 Targets/LS2K500/conf/LS2K500.dts 中打开 i2c0、i2c2、i2c3 设备节点。

```
8 aliases {
9     ethernet0 = &gmac0;
10    ethernet1 = &gmac1;
11    serial0 = &cpu_uart2;
12    i2c0 = &i2c0;
13    i2c1 = &i2c1;
14    i2c2 = &i2c2;
15    i2c3 = &i2c3;
16 };
```

如下图，i2c0 节点中，status = “disabled”;项被注释掉。i2c1 节点中，status = “disabled”;项被打开，则内核会根据设备树，初始化 i2c0，不初始化 i2c1。

```
546     i2c0: i2c@0x1ff48000 {
547         compatible = "loongson,ls2k-i2c";
548         reg = <0 0x1ff48000 0 0x0800>;
549         interrupt-parent = <&extioic>;
550         interrupts = <14>;
551         //status = "disabled";
552         eeprom@57 {
553             compatible = "atmel,24c64";
554             reg = <0x57>;
555             pagesize = <32>;
556         };
557         rtc@68 {
558             compatible = "dallas,ds1339";
559             reg = <0x68>;
560         };
561     };
562
563     i2c1: i2c@0x1ff48800 {
564         compatible = "loongson,ls2k-i2c";
565         reg = <0 0x1ff48800 0 0x0800>;
566         interrupt-parent = <&extioic>;
567         interrupts = <15>;
568         status = "disabled";
569         codec@1a {
570             compatible = "codecs,uda1342";
```

按照上面内容 CONFIG_2K500_NAND 配置项，并且按下图定义“#if 1”，注释掉 status 参数，则会在设备树中打开 i2c2 与 i2c3 节点。

```
738 #else
739 #if 1
740     i2c2: i2c@0x1ff49000 {
741         compatible = "loongson,ls2k-i2c";
742         reg = <0 0x1ff49000 0 0x0800>;
743         interrupt-parent = <&extioic>;
744         interrupts = <16>;
745         //status = "disabled";
746     };
747
748     i2c3: i2c@0x1ff49800 {
749         compatible = "loongson,ls2k-i2c";
750         reg = <0 0x1ff49800 0 0x0800>;
751         interrupt-parent = <&extioic>;
752         interrupts = <17>;
753         //status = "disabled";
754     };
755 #else
756     can0: can@1ff44000 {
757         compatible = "nxp,sja1000";
758         reg = <0 0x1ff44000 0 0x1000>;
759     };
760 #endif
761 #endif
762 #endif
763 #endif
764 #endif
765 #endif
766 #endif
767 #endif
768 #endif
769 #endif
770 #endif
771 #endif
772 #endif
773 #endif
774 #endif
775 #endif
776 #endif
777 #endif
778 #endif
779 #endif
780 #endif
781 #endif
782 #endif
783 #endif
784 #endif
785 #endif
786 #endif
787 #endif
788 #endif
789 #endif
790 #endif
791 #endif
792 #endif
793 #endif
794 #endif
795 #endif
796 #endif
797 #endif
798 #endif
799 #endif
800 #endif
801 #endif
802 #endif
803 #endif
804 #endif
805 #endif
806 #endif
807 #endif
808 #endif
809 #endif
810 #endif
811 #endif
812 #endif
813 #endif
814 #endif
815 #endif
816 #endif
817 #endif
818 #endif
819 #endif
820 #endif
821 #endif
822 #endif
823 #endif
824 #endif
825 #endif
826 #endif
827 #endif
828 #endif
829 #endif
830 #endif
831 #endif
832 #endif
833 #endif
834 #endif
835 #endif
836 #endif
837 #endif
838 #endif
839 #endif
840 #endif
841 #endif
842 #endif
843 #endif
844 #endif
845 #endif
846 #endif
847 #endif
848 #endif
849 #endif
850 #endif
851 #endif
852 #endif
853 #endif
854 #endif
855 #endif
856 #endif
857 #endif
858 #endif
859 #endif
860 #endif
861 #endif
862 #endif
863 #endif
864 #endif
865 #endif
866 #endif
867 #endif
868 #endif
869 #endif
870 #endif
871 #endif
872 #endif
873 #endif
874 #endif
875 #endif
876 #endif
877 #endif
878 #endif
879 #endif
880 #endif
881 #endif
882 #endif
883 #endif
884 #endif
885 #endif
886 #endif
887 #endif
888 #endif
889 #endif
890 #endif
891 #endif
892 #endif
893 #endif
894 #endif
895 #endif
896 #endif
897 #endif
898 #endif
899 #endif
900 #endif
901 #endif
902 #endif
903 #endif
904 #endif
905 #endif
906 #endif
907 #endif
908 #endif
909 #endif
910 #endif
911 #endif
912 #endif
913 #endif
914 #endif
915 #endif
916 #endif
917 #endif
918 #endif
919 #endif
920 #endif
921 #endif
922 #endif
923 #endif
924 #endif
925 #endif
926 #endif
927 #endif
928 #endif
929 #endif
930 #endif
931 #endif
932 #endif
933 #endif
934 #endif
935 #endif
936 #endif
937 #endif
938 #endif
939 #endif
940 #endif
941 #endif
942 #endif
943 #endif
944 #endif
945 #endif
946 #endif
947 #endif
948 #endif
949 #endif
950 #endif
951 #endif
952 #endif
953 #endif
954 #endif
955 #endif
956 #endif
957 #endif
958 #endif
959 #endif
960 #endif
961 #endif
962 #endif
963 #endif
964 #endif
965 #endif
966 #endif
967 #endif
968 #endif
969 #endif
970 #endif
971 #endif
972 #endif
973 #endif
974 #endif
975 #endif
976 #endif
977 #endif
978 #endif
979 #endif
980 #endif
981 #endif
982 #endif
983 #endif
984 #endif
985 #endif
986 #endif
987 #endif
988 #endif
989 #endif
990 #endif
991 #endif
992 #endif
993 #endif
994 #endif
995 #endif
996 #endif
997 #endif
998 #endif
999 #endif
1000 #endif
```

i2c4 与 i2c5 被定义在 dc 节点内。

```
286     dc0_i2c: pixi2c@0x1ff4a000{
287         compatible = "loongson,ls2k-i2c";
288         reg = <0 0x1ff4a000 0 0x0800>;
289         interrupt-parent = <&extioic>;
290         interrupts = <18>;
291     };
292
293     dc1_i2c: pixi2c@0x1ff4a800 {
294         compatible = "loongson,ls2k-i2c";
295         reg = <0 0x1ff4a800 0 0x0800>;
296         interrupt-parent = <&extioic>;
297         interrupts = <19>;
298     };
299 };
300
301 ahci@0x1f040000{
302     compatible = "snps,spear-ahci";
Targets/LS2K500/conf/LS2K500.dts [+]
```

(3) 内核下 I2C 驱动配置

1) 内核下 I2C 驱动配置

[CONFIG_I2C_LS2X](#)。

2) 内核下 I2C 从模式驱动配置

[CONFIG_I2C_SLAVE](#)、[CONFIG_I2C_SLAVE_EEPROM](#)。

(4) I2C 主/从模式测试用例

1) pmon 下 I2C 主模式测试用例

2K500-pai 的 I2C0 的 0x57 地址挂载了一个 64Kb 的 eeprom 设备，地址左移 1 位，在 pmon 下被扫到的地址就变为 0xae。pmon 下的 eeprom 设备驱动为 Targets/LS2K500/dev/eeprom.c，驱动中已将默认设备地址设置为

了 0xae。

```
14 #include "generate_mac_val.c"
15
16 #define EE_SIZE 0x2000 // 8KB(64Kb)
17
18 #define CAT24C64_ADDR 0xae
```

在 pmon 命令行下，可按照下述命令对 eeprom 进行操作：

```
PMON>
PMON> set_dev_pins i2c0
PMON>
PMON> i2c_base 0
PMON>
PMON> i2c_scan
after ls_i2c_init
read dev_addr: 0x00
Eeprom has no ack, Pls check the hardware!
```

```
read dev_addr: 0xac
Eeprom has no ack, Pls check the hardware!
read dev_addr: 0xae
ret : 0xff
read dev_addr: 0xb0
Eeprom has no ack, Pls check the hardware!
read dev_addr: 0xb2
```

```
PMON>
PMON> eepread 0 6
0: 0x11
1: 0x22
2: 0x33
3: 0x44
4: 0x55
5: 0x66
PMON>
PMON>
PMON>
PMON> eepwrite 0 "51 52 53 54 55 56"
0 <= 0x51
1 <= 0x52
2 <= 0x53
3 <= 0x54
4 <= 0x55
5 <= 0x56
PMON>
PMON> eepread 0 6
0: 0x51
1: 0x52
2: 0x53
3: 0x54
4: 0x55
5: 0x56
```

2) 内核下 I2C 主模式测试用例

2K500-pai 的 I2C0 上 0x68 地址接的有一个 RTC 设备 AT8339 , 兼容 DS1338 设备驱动。

a. 内核打开 RTC 驱动配置

[CONFIG_RTC_DRV_DS1307](#)。

b. 设备日志及操作现象

```
[ ... ] Loading completed in 11.000 seconds
[ Btrfs loaded, crc32c=crc32c-generic
[ rtc-ds1307 0-0068: setting system clock to 2000-01-01 00:01:46 UTC (946684906)

root@ls3a5000:~#
root@ls3a5000:~# date -s "2021-08-24 09:45:40"
Tue Aug 24 09:45:40 UTC 2021
root@ls3a5000:~#
root@ls3a5000:~# hwclock -w
```

断电 6 分钟后 , 上电启动内核读取时间 (需连接 rtc 电池):

```
root@ls3a5000:~#
root@ls3a5000:~# hwclock -r
Tue Aug 24 09:51:23 2021  0.000000 seconds
root@ls3a5000:~#
root@ls3a5000:~#
root@ls3a5000:~# date
Tue Aug 24 09:51:26 UTC 2021
root@ls3a5000:~#
```

3) 内核下 I2C 从模式测试用例

根据 5.11 中第三节的内容 , 进行 I2C 从模式的配置后 , 便可进行下述操作。

将从端 I2C2 与主端 I2C2 连接起来 , 从端设置 I2C2 为 0x64 地址。(设置地址时需要或上 0x1000 , 驱动 drivers/i2c/i2c-core-base.c 的

i2c_sysfs_new_device 函数处理 new_device 设备时，会判断地址 flasg 是否为 I2C_ADDR_OFFSET_SLAVE(0x1000))

```
root@ls3a5000:~#
root@ls3a5000:~# echo slave-24c02 0x1064 > /sys/bus/i2c/devices/i2c-2/new_device
[ 249.575046] i2c i2c-2: new_device: Instantiated device slave-24c02 at 0x64
root@ls3a5000:~#
```

主端对 I2C2 的 0x64 设备进行读写操作

```
root@ls3a5000:~#
root@ls3a5000:~# i2cdump -f -y 2 0x64
No size specified (using byte-data access)
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .....
10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .....
20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .....
30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .....
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .....
50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .....
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .....
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .....
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .....
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .....
a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .....
b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .....
c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .....
d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .....
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .....
f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .....
root@ls3a5000:~#
root@ls3a5000:~#
root@ls3a5000:~#
root@ls3a5000:~#
root@ls3a5000:~#
root@ls3a5000:~# i2cset -f -y 2 0x64 0x0 0x11
root@ls3a5000:~# i2cset -f -y 2 0x64 0x1 0x22
root@ls3a5000:~#
root@ls3a5000:~# i2cdump -f -y 2 0x64
No size specified (using byte-data access)
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 11 22 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ?".....
10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .....
20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .....
```

注：将同一底板的不同 I2C 分别设置主、从模式，也可正常通讯；但如果 I2C 下接的有设备，则该路 I2C 不可设置为从模式。

5.12 SPI

2K500-pai 上有 SPI0、SPI1、SPI3 三路 spi。

(1) SPI 复用设置

SPI0、SPI1 的片选 1、片选 2、片选 3 三个引脚信号 ,分别与 NAND、I2C0、I2C2、I2C3、CAN0、CAN1 存在复用冲突。复用注意事项可查看附录“2K500 复用设置注意事项”。

gpio40~43 ,设置为芯片主功能 ,gpio64~66 ,设置为第 3 复用 ;(spi0)

gpio44~47 ,设置为芯片主功能 ,gpio67~69 ,设置为第 3 复用 ;(spi1)

gpio32~35 ,设置为第 3 复用。(spi3)

在 pmon 源码下的 Targets/LS2K500/ls2k500/pincfgs.c 文件中的 default_pin_cfgs 数组里 ,配置的是板卡 pmon 默认的 gpio 复用关系 ,由上 ,需按照下图设置 pmon 下的复用关系。

```
37 { 32, 3}, //5:kb_clk (module) 3:spi3_clk (pai)
38 { 33, 3}, //5:kb_dat (module) 3:spi3_miso (pai)
39 { 34, 3}, //5:ms_clk (module) 3:spi3_mosi (pai)
40 { 35, 3}, //5:ms_dat (module) 3:spi3_cs (pai)
41 { 36, 5}, //5:ac97_datai(module) 3:pix0_scl (pai)
42 { 37, 5}, //5:ac97_datao(module) 3:pix0_sda (pai)
43 { 38, 5}, //5:ac97_sync (module) 3:pix1_scl (pai)
44 { 39, 5}, //5:ac97_reset(module) 3:pix1_sda (pai)
45 { 40, 5}, //5:spi0_clk
46 { 41, 5}, //5:spi0_miso
47 { 42, 5}, //5:spi0_mosi
48 { 43, 5}, //5:spi0_cs[0]
49 { 44, 5}, //5:spi1_clk 0:GPIO44
50 { 45, 5}, //5:spi1_miso 0:GPIO45
51 { 46, 5}, //5:spi1_mosi 0:GPIO46
52 { 47, 5}, //5:spi1_cs[0] 0:GPIO47
53 { 48, 1}, //1:qmac1_rx_ctl
Targets/LS2K500/ls2k500/pincfgs.c [+]
```

spi0 与 spi1 的片选 1/2/3 引脚如果没使用到的话 ,下图中关于 spi 片选引

脚复用的设置可以不用修改。

```
69 { 64, 3}, //5:scl0 1:nand_rdy[1] 3:spi0_cs
70 { 65, 3}, //5:sda0 1:nand_ce[1] 3:spi0_cs
71 { 66, 3}, //5:can0_rx 1:nand_rdy[2] 2:sda2
72 { 67, 3}, //5:can0_tx 1:nand_ce[2] 2:scl2
73 { 68, 3}, //5:can1_rx 1:nand_rdy[3] 2:sda3
74 { 69, 3}, //5:can1_tx 1:nand_ce[3] 2:scl3
Targets/LS2K500/ls2k500/pincfgs.c [+]
```

(2) pmon 下 spi norflash 配置项

1) norflash 配置项

在 pmon 的配置文件 Targets/LS2K500/conf/ls2k500 中打开 [select nand](#)、[select m25p80](#)，如果用到了 spi0 或 spi1 的 cs1/cs2/cs3 引脚，则需要按照下图中的配置关闭 option CONFIG_LS2K500_NAND。

```
285 select nand
286 select nand_bch
287 select cmd_xyzmodem
288 option HPET_RTC
289 option PCIVERBOSE=5
290 option PCI_INT_GPIO=86
291 option USB3_USE_INTER_CLK
292 #option CONFIG_LS2K500_NAND
293 select m25p80
294 #option DDR_RESET_REVERT
295 #option CPU_DEBUG_UART0
Targets/LS2K500/conf/ls2k500
```

2) norflash 分区设置

在 Targets/LS2K500/include/pmon_target.h 添加 pmon 下 norflash 的默认分区：

SPI0 的片选 0 上的 norflash，前 1M 空间用于存放 pmon，如果需要初始化 SPI0 上的 norflash，需要按照图片里的内容设置分区，保留前 1M 空间。


```

67 #include "nand.h"
68 #if NNAND
69 #define TGT_DEFENV {"mtdparts", "nand-flash:50M@0(kernel)ro, -(rootfs);m25p800:3M@1M(data)", 0, &mtd_rescan}, \
70 {"bootdelay", "3", 0, 0}
71 #else
72 #define TGT_DEFENV {"bootdelay", "3", 0, 0}
73 #endif
Targets/LS2K500/include/pmon_target.h

```

3) norflash 操作命令

pmon 在 Targets/LS2K500/dev/spi_w.c 驱动中，通过 ls_m25p_probe 函数进行设备初始化，在函数里会根据 spi_nand 结构体对 norflash 进行初始化，如下图中，ls2k500_spi0.base = LS2K500_SPI0_BASE，则是初始化 spi0 上的设备；.chip_select = 0，表示设备在片选 0 上。

```

543 static struct ls2k500_spi {
544     void *base;
545     int hz;
546 } ls2k500_spi0 = {LS2K500_SPI0_BASE};
547
548 struct spi_device spi_nand =
549 {
550     .dev = &ls2k500_spi0,
551     .chip_select = 0,
552     .max_speed_hz = 12500000,
553 };
554
Targets/LS2K500/dev/spi_w.c

```

pmon 命令行下，可通过下述命令对设备进行操作：

- a. 查看设备分区情况：load /dev/mtd

```

PMON>
PMON> load /dev/mtd
mtd0: flash:m25p800 type:nor size:0x4000000 writesize:0x1 erasesize:0x1000 partoffset=0x100000,partsize=0x300000 data
/dev/mtd: No such file or directory
PMON>

```

（可见，此时 mtd0 为 norflash 设备的分区，大小为 3M）

- b. 擦除分区：mtd_erase /dev/mtd0
- c. 从 u 盘中拷贝文件到分区：devcp (usb0,0)/filename /dev/mtd0

(3) pmon 设备树中 SPI 设备节点

在设备树 Targets/LS2K500/conf/LS2K500.dts 中打开 spi 节点。按下图所示，注释掉 spi0、spi1、spi3 节点中的 status = "disabled"项。如果打开，内核会跳过改节点不进行初始化，例如图片中的 spi2 节点。

如下图所示，spi0 下添加了一个 norflash 设备节点。"reg = <0>;"表示设备在片选 0 上。"number-of-parts = <0x1>;"表示设备树只传递 1 个分区信息给内核。"partition@0x00100000"中的"reg"设置，表示第一个分区偏移为 1M，分区大小为 norflash 的剩余空间。

注：内核下 norflash 分区设置需与 pmon 下保持一致。

```
484     spi0: spi@0x1fd00000 {
485         compatible = "loongson,ls-spi";
486         reg = <0 0x1fd00000 0 0x10>;
487         //status = "disabled";
488         spidev@0{
489             compatible = "m25p80";
490             spi-max-frequency = <12500000>;
491             reg = <0>;
492             number-of-parts = <0x1>;
493
494             partition@0x00100000 {
495                 label = "os_partition";
496                 reg = <0 0x00100000 0 0x0>;
497             };
498         };
499     };
500
501     spi1: spi@0x1fd40000 {
502         compatible = "loongson,ls-spi";
503         reg = <0 0x1fd40000 0 0x10>;
504         //status = "disabled";
505     };
506
507     /* SPI2~5 has only one CS, which is set by SPCS */
508     spi2: spi@0x1ff50000 {
509         compatible = "loongson,ls-spi";
510         reg = <0 0x1ff50000 0 0x10>;
511         spi-nocs;
512         status = "disabled";
513     };
514
515     spi3: spi@0x1ff51000 {
516         compatible = "loongson,ls-spi";
517         reg = <0 0x1ff51000 0 0x10>;
518         spi-nocs;
519         //status = "disabled";
520     };
521
Targets/LS2K500/conf/LS2K500.dts [+]
```


(4) 内核下 SPI 及 norflash 驱动配置

CONFIG_SPI_LS、

CONFIG_MTD_OF_PARTS、

CONFIG_MTD_BLOCK、

CONFIG_MTD_SPI_NOR、

CONFIG_MTD_M25P80。

5.13 RS485/UART3

2K500-pai 上，RS485 与 UART3 使用的是同一引脚，板卡上默认使用的功能是 RS485，如需使用 UART3，需要改硬件。

RS485 为半双工工作模式，需要通过 gpio45 控制数据收发，当 gpio45 为高电平时，发送数据；当 gpio45 为低电平时，接收数据。

UART3 为全双工工作模式，不需要 gpio 进行控制。

(1) RS485 / UART3 复用设置

复用注意事项可查看附录“2K500 复用设置注意事项”。

gpio62~63，设置为芯片主功能。

gpio45，设置为 gpio 功能。

在 pmon 源码下的 Targets/LS2K500/ls2k500/pincfgs.c 文件中的 default_pin_cfgs 数组里，配置的是板卡 pmon 默认的 gpio 复用关系，由上，需按照下图设置 pmon 下的复用关系。

```
49 { 44, 0}, //5:spi1_clk      0:GPIO44
50 { 45, 0}, //5:spi1_miso    0:GPIO45
51 { 46, 0}, //5:spi1_mosi    0:GPIO46
Targets/LS2K500/ls2k500/pincfgs.c

67 { 62, 5}, //5:uart3_tx(pai)  1:pix1_scl(module)
68 { 63, 5}, //5:uart3_rx(pai)  1:pix1_sda(module)
69 { 64, 1}, //5:scl0         1:nand_rdy[1]      3:spi0
Targets/LS2K500/ls2k500/pincfgs.c
```

(2) pmon 设备树中 RS485/UART3 节点

在设备树 Targets/LS2K500/conf/LS2K500.dts 中打开 uart3 节点。按下图所示，注释掉 cpu_uart3 节点中的 status = "disabled"项。

```
154     cpu_uart3: serial@0x1ff40c00 {
155         compatible = "ns16550a";
156         reg = <0 0x1ff40c00 0 0x10>;
157         clock-frequency = <100000000>;
158         #if 0
159         interrupt-parent = <&extioic>;
160         interrupts = <3>;
161         #else
162         interrupt-parent = <&icu>;
163         interrupts = <3>;
164         #endif
165         no-loopback-test;
166         //status = "disabled";
167     };
Targets/LS2K500/conf/LS2K500.dts
```

(3) 内核下串口驱动配置

打开 CONFIG_SERIAL_8250、

设置 CONFIG_SERIAL_8250_NR_UARTS 与

CONFIG_SERIAL_8250_RUNTIME_UARTS 为 10

(4) 现象日志

内核下 uart3/rs485 被初始化为 ttyS1 节点

```
pcteport 0000.00.01.0: Signalling FHE with IRQ 07
Serial: 8250/16550 driver, 10 ports, IRQ sharing enabled
console [ttyS0] disabled
1ff40800.serial: ttyS0 at MMIO 0x1ff40800 (irq = 16, base_baud = 6250000) is a 16550A
console [ttyS0] enabled
1ff40c00.serial: ttyS1 at MMIO 0x1ff40c00 (irq = 17, base_baud = 6250000) is a 16550A
```

5.14 WATCH DOG

(1) ACPI 内部看门狗（等待 5s，系统复位）

写入 WD_Timer 寄存器的值为 $5 \times 100000000 = 500000000 = 0x1dcd6500$

```
[2021-08-13 16:39:38] root@ls3a5000:~# devmem2 0xff6c038 w 0x1DCD6500
[2021-08-13 16:39:48] /dev/mem opened.
[2021-08-13 16:39:48] Memory mapped at address 0x7ff65c0000.
[2021-08-13 16:39:48] Read at address 0xff6c038 (0x7ff65c0038): 0xffffffff
[2021-08-13 16:39:48] Write at address 0xff6c038 (0x7ff65c0038): 0x1DCD6500, readback 0x1DCD6500
[2021-08-13 16:39:48] root@ls3a5000:~#
[2021-08-13 16:39:49] root@ls3a5000:~# devmem2 0xff6c030 w 0x2
[2021-08-13 16:39:57] /dev/mem opened.
[2021-08-13 16:39:57] Memory mapped at address 0x7ff5910000.
[2021-08-13 16:39:57] Read at address 0xff6c030 (0x7ff5910030): 0x00000000
[2021-08-13 16:39:57] Write at address 0xff6c030 (0x7ff5910030): 0x00000002, readback 0x00000002
[2021-08-13 16:39:57] root@ls3a5000:~#
[2021-08-13 16:39:57] root@ls3a5000:~#
[2021-08-13 16:39:58] root@ls3a5000:~# devmem2 0xff6c034 w 0x1
[2021-08-13 16:40:02] /dev/mem opened.
[2021-08-13 16:40:02] Memory mapped at address 0x7ff42dc000.
[2021-08-13 16:40:02] Read at address 0xff6c034 (0x7ff42dc034): 0x00000000
[2021-08-13 16:40:02] Write at address 0xff6c034 (0x7ff42dc034): 0x00000001, readback 0x00000001
[2021-08-13 16:40:02] root@ls3a5000:~#
[2021-08-13 16:40:07] initserial good ^.^...
[2021-08-13 16:40:07] Soft CLK SEL adjust begin
[2021-08-13 16:40:07]
[2021-08-13 16:40:07] NODE      :01140489
[2021-08-13 16:40:07] DDR       :010c048f
[2021-08-13 16:40:07] SOC       :0a50048f
[2021-08-13 16:40:07] PIX0      :146d0589
[2021-08-13 16:40:07] PIX1      :146d0589SATA0 enabled
[2021-08-13 16:40:07] SATA1 enabled
[2021-08-13 16:40:07]
[2021-08-13 16:40:07] PMON2000 MIPS Initializing. Standby...
```

(2) 外部看门狗 MAX6369（1-3s，系统复位）

外部看门狗通过令 gpio33 输出高电平，使能看门狗定时器；gpio35 进行喂狗操作，保持一种电平，表示不喂狗；不停进行高低电平切换，则表示喂狗操作。

```

[2021-08-26 15:43:26] root@ls3a5000:~#
[2021-08-26 15:43:26] root@ls3a5000:~# devmem2 0x1fe104a0 w 0x55550505
[2021-08-26 15:43:40] /dev/mem opened.
[2021-08-26 15:43:40] Memory mapped at address 0x7ff7748000.
[2021-08-26 15:43:40] Read at address 0x1FE104A0 (0x7ff77484a0): 0x55555555
[2021-08-26 15:43:40] Write at address 0x1FE104A0 (0x7ff77484a0): 0x55550505, readback 0x55550505
[2021-08-26 15:43:40] root@ls3a5000:~#
[2021-08-26 15:43:47] root@ls3a5000:~# devmem2 0x1fe10434 b 0xf5
[2021-08-26 15:43:58] /dev/mem opened.
[2021-08-26 15:43:58] Memory mapped at address 0x7ff5108000.
[2021-08-26 15:43:58] Read at address 0x1FE10434 (0x7ff5108434): 0xFF
[2021-08-26 15:43:58] Write at address 0x1FE10434 (0x7ff5108434): 0xF5, readback 0xF5
[2021-08-26 15:43:58] root@ls3a5000:~#
[2021-08-26 15:43:58] root@ls3a5000:~# devmem2 0x1fe10444 b 0x2
[2021-08-26 15:44:07] /dev/mem opened.
[2021-08-26 15:44:07] Memory mapped at address 0x7ff61e4000.
[2021-08-26 15:44:07] Read at address 0x1FE10444 (0x7ff61e4444): 0x00
[2021-08-26 15:44:07] Write at address 0x1FE10444 (0x7ff61e4444): 0x02, readback 0x02
[2021-08-26 15:44:07] root@ls3a5000:~#
[2021-08-26 15:44:08] initserial good ^_^.
[2021-08-26 15:44:08] Soft CLK SEL adjust begin
[2021-08-26 15:44:08]
[2021-08-26 15:44:08] NODE      :01140489
[2021-08-26 15:44:08] DDR      :0110048f

```

5.15 PWM

2K500-pai 上有 PWM0~3 四路 PWM ,其中 PWM0 设置为了 gpio 功能 ,默认做了 LCD 的背光信号脚 ,如需使用 ,需要硬件改电阻 ;PWM3 用作了板卡上电配置 ,只可做输出功能。

(1) PWM0~3 复用设置

复用注意事项可查看附录“2K500 复用设置注意事项”。

gpio84~87 , 设置为芯片主功能。(PWM0~3)

在 pmon 源码下的 Targets/LS2K500/ls2k500/pincfgs.c 文件中的 default_pin_cfgs 数组里 ,配置的是板卡 pmon 默认的 gpio 复用关系 ,由上 ,需按照下图设置 pmon 下的复用关系。

```
88 { 83, 5}, //5:gpio83(pai) 0:gpio83(module)
89 { 84, 5}, //5:pwm[0](pai) 0:gpio84(module)
90 { 85, 5}, //5:pwm[1](pai) 0:gpio85(module)
91 { 86, 5}, //5:pwm[2](pai) 0:gpio86(module)
92 { 87, 5}, //5:pwm[3](pai) 0:gpio87(module)
93 { 88, 5}, //5:gmac0 rx ctl
Targets/LS2K500/ls2k500/pincfgs.c
```

(2) pmon 设备树中 PWM 设备节点

在设备树 Targets/LS2K500/conf/LS2K500.dts 中打开 PWM 节点。按下图所示 ,注释掉 PWM0~PWM3 节点中的 status = “disabled”项 ,内核则会根据节点信息去初始化 PWM 控制器。如果 status 项不注释调 ,内核则不会初始化这些节点 ,例如 PWM4~PWM7。


```

794     pwm0: pwm@1ff5c000{
795         compatible = "loongson,ls-pwm";
796         reg = <0 0x1ff5c000 0 0x10>;
797         clock-frequency = <100000000>;
798         interrupt-parent = <&extioic>;
799         interrupts = <40>;
800         //status = "disabled";
801     };
802
803     pwm1: pwm@1ff5c010{
804         compatible = "loongson,ls-pwm";
805         reg = <0 0x1ff5c010 0 0x10>;
806         clock-frequency = <100000000>;
807         interrupt-parent = <&extioic>;
808         interrupts = <41>;
809         //status = "disabled";
810     };
811
812     pwm2: pwm@1ff5c020{
813         compatible = "loongson,ls-pwm";
814         reg = <0 0x1ff5c020 0 0x10>;
815         clock-frequency = <100000000>;
816         interrupt-parent = <&extioic>;
817         interrupts = <42>;
818         //status = "disabled";
819     };
820
821     pwm3: pwm@1ff5c030{
822         compatible = "loongson,ls-pwm";
823         reg = <0 0x1ff5c030 0 0x10>;
824         clock-frequency = <100000000>;
825         interrupt-parent = <&extioic>;
826         interrupts = <43>;
827         //status = "disabled";
828     };
829
830     pwm4: pwm@1ff5c040{
831         compatible = "loongson,ls-pwm";
832         reg = <0 0x1ff5c040 0 0x10>;
833         clock-frequency = <100000000>;
834         interrupt-parent = <&extioic>;
835         interrupts = <44>;
836         status = "disabled";
837     };

```

Targets/LS2K500/conf/LS2K500.dts [+]

(3) 内核下 PWM 驱动

CONFIG_PWM_LS。

(4) PWM 输出及捕获配置

1) PWM1 输出方波

```
root@ls3a5000:~# devmem2 0x1ff5c018 w 0x1000000
/dev/mem opened.
Memory mapped at address 0x7ff79b8000.
Read at address 0x1FF5C018 (0x7ff79b8018): 0x00000000
Write at address 0x1FF5C018 (0x7ff79b8018): 0x01000000, readback 0x01000000
root@ls3a5000:~#
root@ls3a5000:~# devmem2 0x1ff5c014 w 0x7fffff
/dev/mem opened.
Memory mapped at address 0x7ff59cc000.
Read at address 0x1FF5C014 (0x7ff59cc014): 0x00000000
Write at address 0x1FF5C014 (0x7ff59cc014): 0x007FFFFF, readback 0x007FFFFF
root@ls3a5000:~#
root@ls3a5000:~#
root@ls3a5000:~#
root@ls3a5000:~# devmem2 0x1ff5c01c w 0x1
/dev/mem opened.
Memory mapped at address 0x7ff6698000.
Read at address 0x1FF5C01C (0x7ff669801c): 0x00000000
Write at address 0x1FF5C01C (0x7ff669801c): 0x00000001, readback 0x00000001
root@ls3a5000:~#
```

2) PWM2 捕获高低电平

```
root@ls3a5000:~# devmem2 0x1ff5c02c w 0x101
/dev/mem opened.
Memory mapped at address 0x7ff51d4000.
Read at address 0x1FF5C02C (0x7ff51d402c): 0x00000000
Write at address 0x1FF5C02C (0x7ff51d402c): 0x00000101, readback 0x00000101
root@ls3a5000:~#
root@ls3a5000:~#
root@ls3a5000:~#
root@ls3a5000:~# devmem2 0x1ff5c028
/dev/mem opened.
Memory mapped at address 0x7ff622c000.
Read at address 0x1FF5C028 (0x7ff622c028): 0x01000000
root@ls3a5000:~# devmem2 0x1ff5c024
/dev/mem opened.
Memory mapped at address 0x7ff513c000.
Read at address 0x1FF5C024 (0x7ff513c024): 0x007FFFFF
root@ls3a5000:~#
```

5.16 GPIO 中断

(1) 2K500 含有 155 个 gpio，其中每 4 个 gpio 使用同一个扩展中断，且每组中的第四个 gpio 不可做 gpio 中断，即 $\text{gpio}(4 * N + 3)$ 的 gpio 号，不可做 gpio 中断。

(2) gpio123~gpio154 不可做 gpio 中断。

(3) gpio 如果要做中断的话，需要令其外部拉低。

(4) 使用相同中断号的一组 gpio，只能注册其中一个 gpio 的中断。

附录 A： 2K500 复用设置注意事项

(1) 默认复用设置

在 pmon 源码 Targets/LS2K500/ls2k500/pincfgs.c 中，
default_pin_cfgs 数组中设置的为引脚的默认复用值。数组中第一个参数为
gpio 号；第二个参数为引脚复用值，0 表示 gpio 功能，1 表示第 1 复用，2
表示第 2 复用，3 表示第 3 复用，4 表示第 4 复用，5 表示芯片主功能。

module 为嵌入式模块板上，对应的接口功能；pai 为 2K500-pai 上对应的
的接口功能，如果设计板卡上使用的接口复用不同，则可在数组中对应的进行
修改。

```
3 pin_cfgs_t default_pin_cfgs[] = {
4     //0:GPIO , 5:main
5     { 0, 0},    //0:GPIO0
6     { 1, 0},    //0:GPIO1
7     { 2, 5},    //5:vga_hsync
8     { 3, 5},    //5:vga_vsync
9     { 4, 5},    //5:lcd_clk          1:can2_rx   (module)
10    { 5, 5},    //5:lcd_vsync        1:can2_tx   (module)
11    { 6, 5},    //5:lcd_hsync        1:can3_rx   (module)
```

(2) cfg_func_multi 函数设置

在 pmon 源码 Targets/LS2K500/ls2k500/pincfgs.c 中 ,cfg_func_multi
函数可以设置接口的功能复用。在 pmon 下的设备驱动中，会通过该函数在初
始化时，对设备复用进行设置，如果设计板卡上使用的接口复用不同，可在函
数中对应修改。

例如函数中 gmac1，便是令 gpio48~59 作第 1 复用功能。

```

254 int cfg_func_multi(const char * name, int skip)
255 {
256     int ret = 0;
257     if (strstr(name, "can0")) {
258         ret = loop_set_pin(66, 67, 5, skip);
259     } else if (strstr(name, "can1")) {
260         ret = loop_set_pin(68, 69, 5, skip);
261     } else if (strstr(name, "can2")) {
262         ret = loop_set_pin(4, 5, 1, skip);
263     } else if (strstr(name, "can3")) {
264         ret = loop_set_pin(6, 7, 1, skip);
265     } else if (strstr(name, "gmac0")) {
266         ret = loop_set_pin(88, 99, 5, skip);
267     } else if (strstr(name, "gmac1")) {
268         ret = loop_set_pin(48, 59, 1, skip);
269     } else if (strstr(name, "ac97")) {
270         ret = loop_set_pin(36, 39, 5, skip);
271         readl(LS2K500_GENERAL_CFG0) &= ~(1 << 9); //disable hda
272     } else if (strstr(name, "hda")) {

```

(3) pmon 命令行下调用命令修改复用配置

板卡调试阶段，如果不想多次烧写 pmon 去修改接口复用，则可在 pmon 命令行下通过 set_dev_pins 命令，对设备复用进行设置。命令会调用(2)中的 cfg_func_multi 函数，通过函数中设置的引脚复用值去设置接口复用。

例：`set_dev_pins i2c0`

(4) 复用冲突处理

注：接口的复用引脚只能设置 1 组，其余同组功能的引脚复用需关闭。

例如 I2C0，gpio48~49 的第 3 复用，以及 gpio64~65 的芯片主功能，这些都可设置为 I2C0 功能，但这些 gpio 引脚，同时必须只能有一组作为 I2C0 功能。

即以 I2C0 为例，如果 gpio64~65 做了芯片主功能，则 gpio48~49 便不可设置为第 3 复用，否则将存在接口功能冲突问题，导致 i2c0 无法被正常使用。