

# RTEMS on S3C2410 移植备忘录

## 目录

1: RTEMS 开发环境建立 .....	2
1.1 开发环境建立前的准备工作 .....	2
1.2 建立 GNU 工具链 .....	2
1.3 Load Image .....	2
2: 修改 s3c2400==>s3c2410.....	2
2.1 register 肯定不同, 需要修改 .....	2
2.2 然后从链接脚本入手 .....	2
3: 对 linkcmds 的理解 .....	3
3.1: ENTRY(_start)这一句很重要 .....	3
3.2: section 的分配 .....	3
4: 对 start.S 的理解(位于 c/src/lib/libbsp/arm/gp32/start).....	3
6: 多任务怎么开始.....	4
7: 系统初始化及多任务开始流程(对应到函数调用顺序).....	4
8: 系统是怎么实现驱动的 .....	4
9: 从 hello world 入手 .....	5
9.1 需要改动的地方 .....	5
9.2 需要确认的地方: .....	5
9.3 调试方法 .....	6
10: 调试 hello world 时碰到的问题.....	6
11: 调试网络时碰到的问题 .....	7
11.1 MMU .....	7
11.2 照葫芦画瓢 .....	8
11.3 需要添加的东东.....	9
11.4 ping.....	11
11.5 ping 不稳定 .....	11

这是本人的草率之作, 仅作备忘而已, 如有错误或读者发现需要改进之处, 还请各位告之。MSN:

[13872051302@monternet.com](mailto:13872051302@monternet.com) Email: [ximenpiaoxue4016@sina.com](mailto:ximenpiaoxue4016@sina.com) (欢迎交流)

第 1 页

## 1: RTEMS 开发环境建立

### 1.1 开发环境建立前的准备工作

因为 RTEMS 开发环境主要用的是 GNU 的工具链, 所以需要一台装有 Linux 的 HOST.

**1.2 建立 GNU 工具链**的详细步骤可以参考个版本的 doc 中的 start.pdf, 或者参考 [www.rtems.net](http://www.rtems.net) 中 [ray](#) 的 [rtems](#) 连载。

### 1.3 Load Image

烧写 u-boot 到 s3c2410 norFlash 中(也可以是 nandFlash), 详见 s3c2410 开发板套件资料。

烧写成功后, 就可以通过 F T P 下载编译后的 bin 文件到板上运行。(小技巧, 为了使编译后直接生成 bin 文件, 可以在 gp32.cfg 文件中做如下修改:

```
define make-exe
$(LINK.c) $(AM_CFLAGS) $(AM_LDFLAGS) -o $@ \
    $(LINK_OBJS) $(LINK_LIBS)
$(OBJCOPY) -O binary $(basename $@).exe $(basename $@).bin
$(NM) -g -n $(basename $@).exe > $(basename $@).elf
$(SIZE) $(basename $@).exe
Endef)
```

## 2: 修改 s3c2400==>s3c2410

因为用的是 rtems-4.6.99.3 自带的 bsp(gp32), 它针对的 cpu 是 s3c2400, 我们要在 s3c2410 上调试, 有许许多多需要修改的地方。

### 2.1 register 肯定不同, 需要修改

在 s3c2410 开发板套件资料的 armsys\_2410/hardware\_test\_pro 中找到 s3c2410.h, 与 c/src/lib/libcpu/arm/s3c2410/include 中的 s3c2400.h 进行比较, 添加 s3c2400 中没有的 register。

**2.2 然后从链接脚本入手,** (编译器就是通过它来组织 code 和分配 memory 的),

所以找到 c/src/lib/libbsp/arm/gp32/startup/linkcmds, 修改  
sdram : ORIGIN = 0x30000000, LENGTH = 64M  
和  
\_sdram\_base = DEFINED(\_sdram\_base) ? \_sdram\_base : 0x30000000;  
\_sdram\_size = DEFINED(\_sdram\_size) ? \_sdram\_size : 64M;  
三个地方。

这是本人的草率之作, 仅作备忘而已, 如有错误或读者发现需要改进之处, 还请各位告之。MSN:

[13872051302@monternet.com](mailto:13872051302@monternet.com) Email: [ximenpiaoxue4016@sina.com](mailto:ximenpiaoxue4016@sina.com) (欢迎交流)

第 2 页

### 3: 对 linkcmds 的理解

**3.1: ENTRY(\_start)这一句很重要**, 相当与 ads 中的 entry point, 既 go 命令就从这里开始; \_start 是一个标号, 在 c/src/lib/libbsp/arm/gp32/start/start.S 中定义.

#### 3.2: section 的分配

linkcmds 把编译后的 bin 文件主要分成 .base, .text, .data, .bss 四个区域 load 到 memory. 即:

.base-->0x30000000~0x30000100, 主要存放 vector table;

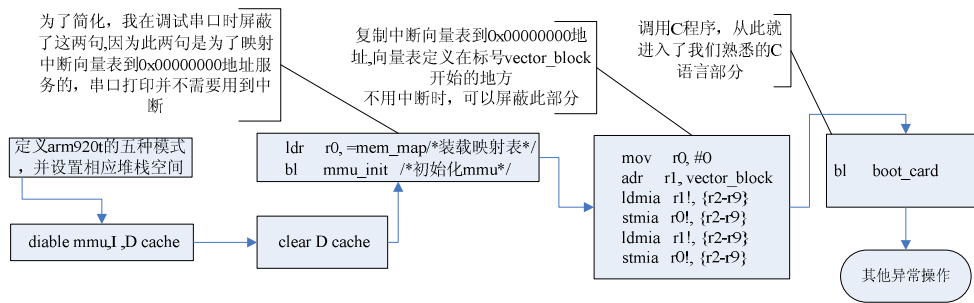
.text-->紧接着 0x30000100 存放, 主要存放 code;

.data-->紧接着 .text, 主要存放已被初始化的全局变量;

.bss-->紧接着 .data, 主要存放没有被初始化的全局变量;(系统自动初始化为 0)

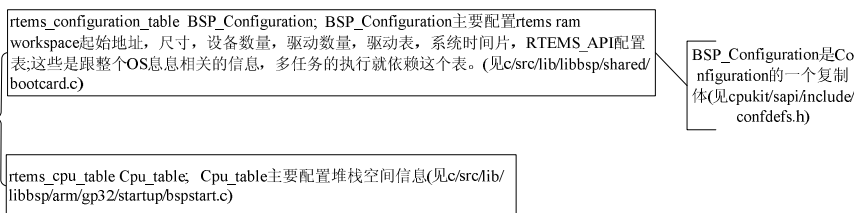
### 4: 对 start.S 的理解(位于 c/src/lib/libbsp/arm/gp32/start)

首先找到 \_start 标号, 程序往下执行, 主要完成以下工作:



### 5: RTEMS 内核是怎么初始化 BSP, CPU 及各部件的

RTEMS 内核的引入主要是根据两个表 (结构体):



这是本人的草率之作, 仅作备忘而已, 如有错误或读者发现需要改进之处, 还请各位告之。MSN:

[13872051302@monternet.com](mailto:13872051302@monternet.com) Email: [ximenpiaoxue4016@sina.com](mailto:ximenpiaoxue4016@sina.com) (欢迎交流)

第 3 页

## 6: 多任务怎么开始

RTEMS 是通过下面两个函数启动多任务的:

`rtems_initialize_executive_early` 初始化 RTEMS 但是不开始多任务;

`rtems_initialize_executive_late` 完成初始化开始多任务;

`rtems_initialize_executive_early` 函数原形为:

```
rtems_interrupt_level rtems_initialize_executive_early(
    rtems_configuration_table *configuration_table,
    rtems_cpu_table            *cpu_table)
```

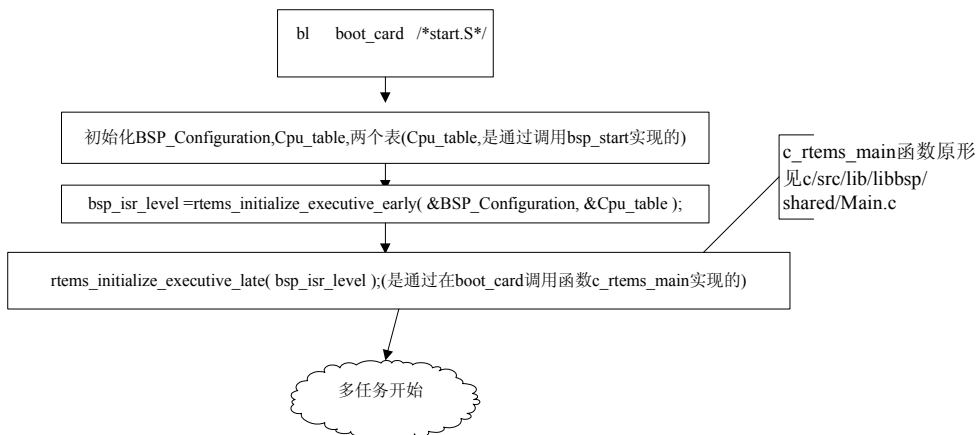
其中的两个参数就为上面提到到了两个配置表. 调用 `rtems_initialize_executive_early` 时将传入 `BSP_Configuration`, `Cpu_table` 这两个表. `rtems_initialize_executive_late` 的函数原形为:

```
void rtems_initialize_executive_late(rtems_interrupt_level bsp_level);( 见
cpukit/sapi/src/Exinit.c)
```

其入口参数即调用 `rtems_initialize_executive_early` 函数的返回值 .

批注 [x1]: 见  
cpukit/sapi/src/Exinit.c

## 7: 系统初始化及多任务开始流程(对应到函数调用顺序)



## 8: 系统是怎么实现驱动的

`rtems` 是通过一个初始化任务 `Init` 来初始化 `bsp.cpu,OS` 的.用户在应用程序中定义了 `Init` 任务后,必须得有 `#define CONFIGURE_INIT` 这条语句, `rtems` 就是通过判断用户是否定义了 `#define CONFIGURE_INIT` 来启动多任务并实现驱动的.

在 `cpukit/sapi/include/confdefs.h` 中,系统通过判断用户是否定义了 `CONFIGURE_INIT` 来初始化 `Configuration` 配置表, `Configuration_RTEMS_API` 配置表.

如果用户应用程序中要用到控制台,通常是串口打印信息则必须在应用程序中定义 `#define` 这是本人的草率之作, 仅作参考而已, 如有错误或读者发现需要改进之处, 还请各位告之. MSN:

[13872051302@monternet.com](mailto:13872051302@monternet.com) Email: [ximenpiaoxue4016@sina.com](mailto:ximenpiaoxue4016@sina.com) (欢迎交流)

第 4 页

CONFIGURE\_TEST\_NEEDS\_CONSOLE\_DRIVER,

在 confdefs.h 中, 系统通过判断用户是否定义了 CONFIGURE\_TEST\_NEEDS\_CONSOLE\_DRIVER 来初始化串口。

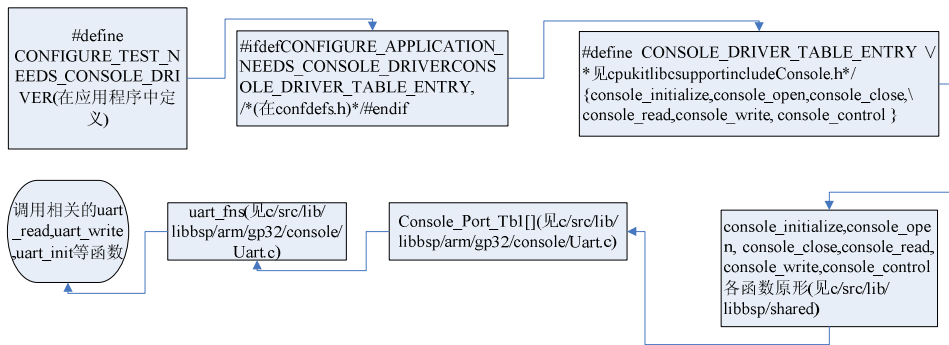
同样, 如果用户应用程序中要用到时钟, 则必须在应用程序中定义 #define CONFIGURE\_APPLICATION\_NEEDS\_CLOCK\_DRIVER, 在 confdefs.h 中系统通过判断用户是否定义了 CONFIGURE\_APPLICATION\_NEEDS\_CLOCK\_DRIVER 来初始化 clock。

所有的驱动放在一个驱动表结构体中, 如 confdefs.h 中的 rtems\_driver\_address\_table Device\_drivers, 而这个表地址在 Configuration 配置表中引入。在 Device\_drivers 中放着各驱动的入口如: CONSOLE\_DRIVER\_TABLE\_ENTRY, 而 CONSOLE\_DRIVER\_TABLE\_ENTRY 定义在 cpukit/libsupport/include/console.h 中, CONSOLE\_DRIVER\_TABLE\_ENTRY 其实是对 console 一系列函数调用。

注意: 上述一切初始化动作是通过在应用程序中预定义的, 所以在预编译的时候就初始化了。所以在调用 rtems\_initialize\_executive\_early 时, 通过 &BSP\_Configuration, &Cpu\_table 这两个参数可以对 BSP, CPU, OS, 各驱动进行初始化。

## 9: 从 hello world 入手

首先, 需要理清 console 驱动的函数调用关系, 其主要流程如下:



### 9.1 需要改动的地方:

- ① c/src/lib/libcpu/arm/s3c2410/include/s3c2400.h 相应 register 的修改。
- ② 因为要使用串口功能, 所以需要在应用程序中添加端口初始化函数, 来初始化各端口, 主要初始化 Port F, H, 端口 F 为 LED 显示功能所用, 端口 H 为串口打印功能所用。  
(可以直接复制 s3c2410 开发板套件资料的 armsys\_2410/hardware\_test\_pro 中的 Port\_Init 函数)
- ③ 在 c/src/lib/libbsp/arm/gp32/startup/Bspstart.c, bsp\_start\_default 函数中添加 bank0-bank7 的相关 register 的设置
- ④ 在 start.S (见 c/src/lib/libbsp/arm/gp32/start) 中添加 LED 显示子程序。

### 9.2 需要确认的地方:

- ① 在 c/src/lib/libbsp/arm/gp32/console/Uart.c 中对照 s3c2410 datasheet 确信各 register

这是本人的草率之作, 仅作备忘而已, 如有错误或读者发现需要改进之处, 还请各位告之。MSN:

[13872051302@monternet.com](mailto:13872051302@monternet.com) Email: [ximenpiaoxue4016@sina.com](mailto:ximenpiaoxue4016@sina.com) (欢迎交流)

第 5 页

地址是否正确.

②确信 baudrate 是否与 PC 机上的串口 baudrate 一致.

③ Fclk,Pclk,Hclk 的值为多少,相关函数参照  
c/src/lib/libcpu/arm/s3c2400/clock/Support.c  
c/src/lib/libbsp/arm/gp32/include/Bsp.h  
c/src/lib/libbsp/arm/gp32/startup/Bspstart.c

知道 baudrate,Fclk,Pclk,Hclk 是怎么计算的,并对照 s3c2410 datasheet 确信  
计算结果是否准确.

### 9.3 调试方法:

LED 跟踪(并借助 ads)

所以需要写一个 LED 显示程序,最好用汇编写,并在 ads 上验证该程序能正常调用.

## 10: 调试 hello world 时碰到的问题

我用 LED 一直跟踪到 boot\_card 函数都很顺利,但到执行 BSP\_RTEMS\_Configuration = \*Configuration.RTEMS\_api\_configuration;这条语句时程序跳飞。显然,这条跳飞的原因大半是指针造成的,Configuration.RTEMS\_api\_configuration 是一个指针,使用指针的大忌是不初始化就使用,因为没有初始化,所指向的地址未知,因此程序跑飞.所以,我定义了一个 rtems\_api\_configuration\_table 结构体,并把其地址赋给 Configuration.RTEMS\_api\_configuration,果然 BSP\_RTEMS\_Configuration = \*Configuration.RTEMS\_api\_configuration;执行过去了.我开始接着往下调,但一个 OS 往往是牵一发而动全身,改动了 Configuration.RTEMS\_api\_configuration,接下来很多地方都要改动,所以要找到根源.

BSP\_RTEMS\_Configuration 是 Configuration 的一个复制体,而 Configuration 是在 confdefs.h 中定义的,是一个已初始化的全局变量,所以我怀疑是不是全局变量初始化不成功?初始化的全局变量是放在.data 区域的,而这一区域是定位在 sdram 中的.那么怎样才能看到这一区域的内容呢?最后想到了 ads,我通过 ftp 把编译好的 bin 文件下载到板子后,再通过 AXD 调试软件查看 sdram 中的内容.这里,可以定义些一眼就能看出来的特征值,比如定义两个全局变量

```
int a=0xaaaaaaaa;  
int b=0xbbbbbbbbb;
```

.但是,64M 的 sdram 怎么查呢!用过 ads 的人应该知道,编译后的执行文件(.o)对每个函数,全局变量都已分配好了地址,这些地址在编译完成时在 Section Cross References 中会列出来.那么,在 gcc 中编译后怎么查看其中的函数、变量的地址呢?找相关资料,发现 readelf 就是干这个事的,所以 readelf -a helloworld.exe >map 得到 map 文件,用 UltraEdit -32 打开查看,记下全局变量 a,b 的地址(map 中你还可以看到各函数的具体位置及占用空间的大小.要知道自己添加的文件是否编译进去,这是个很好的方法,我在编译 network 时,就如此).然后,在 axd 中定位到刚才记下的地址,发现 AXD 中 a,b 的地址与刚记下的地址(map)相差 0x100 偏移量.后面,我还做了大量的实验,比如定义结构体.都证明 gcc 编译后的全局变量的地址与 AXD 中查到的地址相差 0x100,AXD 查到的地址应该是 sdram 中的实际反映.因此我马上想到了 linkcmds 文件的.base 段,其定义的就是 sdram 起始的 0x100 个字节,果然,当我把.base 段去掉后原来跑飞的地方通过了.但去掉.base 也不是办法,之后把它移到.bss 段.但后来想来想去还是不妥,既然系统把它定义到前面一定不会错,我把它改动了,还不知有什么后患呢?那还有什么办法呢?后来突然想到,我通过 uboot 下载 bin 文件到 sdram 时,load 地址是 0x30000000,我把 load 地址改为 0x30000100 后,也一样解决了问题.

在调试 hello world 例程时碰到的主要是这个问题.其他地方只要调试时细心应该不会有有什么问题.

这是本人的草率之作,仅作备忘而已,如有错误或读者发现需要改进之处,还请各位告之. MSN:

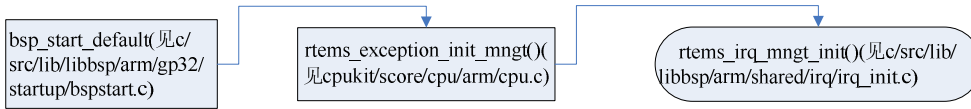
[13872051302@monternet.com](mailto:13872051302@monternet.com) Email: [ximenpiaoxue4016@sina.com](mailto:ximenpiaoxue4016@sina.com) (欢迎交流)

第 6 页

## 11: 调试网络时碰到的问题

在 testsuites/samples 中找到 loopback 例程,发现要用到 clock 驱动. 而专门测试 clock 的例程是 testsuites/samples 下的 ticker.clock 是通过中断来实现的,所以接下来要添加中断.

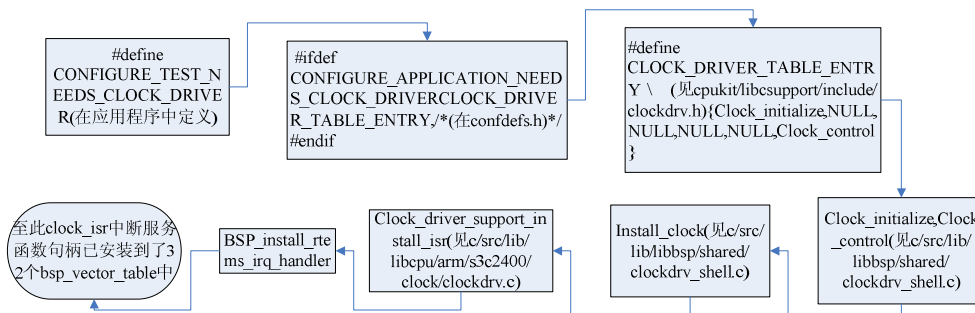
首先,弄清楚 RTEMS 内核,中断的实现机制.rtems-4.6.99.3 关于中断的函数调用流程如下:



rtems\_exception\_init\_mngt 安装六类异常向量地址到 rtems\_vector\_table(在 linkcmds 中定义,实际地址为 0x30000020)开始的地方,rtems\_irq\_mngt\_init 以实际的 irq 句柄初始化异常向量表,并以 default\_int\_handler 地址初始化 32 个 rtems bsp

irq 中断向量地址.当发生 irq 中断时,系统跳到 0x00000018 地址执行,0x00000018 地址处装载的就是 rtems\_irq\_mngt\_init 安装的实际的 irq 句柄,通过这个句柄就可以找到相应的 32 个 rtems bsp irq 中断向量的哪个发生了中断,从而跳到对应 isr 服务程序执行.

系统是怎么找到 clock 中断 isr 服务地址的,rtems-4.6.99.3 关于 clock 中断的函数调用流程如下



到现在虽然中断句柄已经安装好了,但还没有做映射,系统怎么访问 0x00000018 地址呢?

### 11.1 MMU

记得,在 start.S 中我屏蔽了

```
ldr    r0, =mem_map/*装载映射表*/
```

```
bl     mmu_init /*初始化 mmu*/
```

现在要打开

程序的调试工作有回到了汇编阶段,所以,我又用 LED 跟踪.

一直跟踪到函数 mmu\_init 中的 mmu\_set\_ctrl 函数

前都没事,但调用这个函数时程序跑飞了.

我开始怀疑是 C 内嵌汇编出了问题,所以我把 mmu\_set\_ctrl 换成了纯汇编程序放到了 start.S 中,同

这是本人的草率之作, 仅作备忘而已, 如有错误或读者发现需要改进之处, 还请各位告之. MSN:

[13872051302@monternet.com](mailto:13872051302@monternet.com) Email: [ximenpiaoxue4016@sina.com](mailto:ximenpiaoxue4016@sina.com) (欢迎交流)

第 7 页

样跟踪到了我转换后的汇编函数中。但,就在执行使能 mmu 这条语句时飞掉了.把这条语句屏蔽了就没事.说明不是内嵌汇编的问题.那么在执行这条语句时,cpu 到底做了些什么呢?要擦看汇编及 pc 指针最好的办法莫过于用 ads 了,用 AXD 调试程序运行到使能 mmu 这条语句时也跑飞了.做了改动的地方只有

mem\_map, gp32 中的映射表为

```
mmu_sect_map_t mem_map[] = {
/* <phys addr> <virt addr> <size> <flags> */
{0x0c000000, 0x00000000, 1, MMU_CACHE_NONE}, /* SDRAM for vectors */
{0x0c000000, 0x0c000000, 7, MMU_CACHE_WTHROUGH}, /* SDRAM W cache */
{0x0c700000, 0x0c700000, 1, MMU_CACHE_NONE}, /* SDRAM W/O cache */
{0x18000000, 0x18000000, 16, MMU_CACHE_NONE}, /* Internals Regs - */
{0x15000000, 0x15000000, 16, MMU_CACHE_NONE}, /* Internal Regs - */
{0x00000000, 0x00000000, 0, 0} /* The end */
};
```

我改动后为

```
mmu_sect_map_t mem_map[] = {
/* <phys addr> <virt addr> <size> <flags> */
{0x30000000, 0x00000000, 1, MMU_CACHE_NONE}, /* SDRAM for vectors */
{0x30000000, 0x30000000, 7, MMU_CACHE_WTHROUGH}, /* SDRAM W cache */
{0x30700000, 0x30700000, 1, MMU_CACHE_NONE}, /* SDRAM W/O cache */
{0x48000000, 0x48000000, 16, MMU_CACHE_NONE}, /* Internals Regs - */
{0x50000000, 0x50000000, 16, MMU_CACHE_NONE}, /* Internal Regs - */
{0x00000000, 0x00000000, 0, 0} /* The end */
};
```

我开始以为映射属性的问题,但怎么换都不行,后来,发现问题出在映射 size 上,64M 的 sdrum 只映射了 9M,暂且不说,因为我程序加各堆栈再怎么样也不会超过 9M,没有映射完 64M 也不会出什么问题.但是,Internals Regs 如果没有映射完就有问题了,我对照 s3c2410.h,发现,从 48000000-4a000000,50000000-5a000000 而我只映射了 16M 也就是只映射了 48000000-49000000,50000000-51000000,这么多 register 没有映射,当要访问没有映射的 register 时肯定会因找不到地址而跑飞.所以,映射所有的 bank0-bank7

```
mmu_sect_map_t mem_map[] = {
/* <phys addr> <virt addr> <size> <flags> */
{0x30000000, 0x00000000, 1, MMU_CACHE_NONE}, /* SDRAM for vectors */
{0x30000000, 0x30000000, 32, MMU_CACHE_WTHROUGH}, /* SDRAM W cache */
{0x32000000, 0x32000000, 32, MMU_CACHE_NONE}, /* SDRAM W/O cache */
{0x48000000, 0x48000000, 256, MMU_CACHE_NONE}, /* Internals Regs - */
{0x50000000, 0x50000000, 256, MMU_CACHE_NONE}, /* Internal Regs - */
{0x00000000, 0x00000000, 0, 0} /* The end */
};
```

这样,运行 ticker 例程时,看到了正确结果.

## 11.2 照葫芦画瓢

由于 rtems-4.6.99.3 源码自带的 gp32-bsp 中并没有网络驱动,所幸的是 edp7312 中有 cs8900 的驱动,因这是本人的草率之作,仅作参考而已,如有错误或读者发现需要改进之处,还请各位告之。MSN: [13872051302@monternet.com](mailto:13872051302@monternet.com) Email: [ximenpiaoxue4016@sina.com](mailto:ximenpiaoxue4016@sina.com) (欢迎交流)

此可以 copy edp7312 目录下的整个 network 目录到 c/src/lib/libbsp/arm/gp32 下,并修改相关 makefile 文件,这里我把要注意的地方列出来:

①在 configure rtems-4.6.99.3 时去掉 disable-networking disable-posix;

②对 bank3 bus 设置:

```
/*=====Bank 3 parameter=====*/
#define B3_Tacs      0x0 /*0clk*/
#define B3_Tcos      0x3 /*0clk*/
#define B3_Tacc      0x7 /*14clk*/
#define B3_Tcoh      0x1 /*0clk*/
#define B3_Tah       0x0 /*0clk*/
#define B3_Tacp      0x3
#define B3_PMC       0x0 /*normal*/
rBANKCON3=
```

```
((B3_Tacs<<13)+(B3_Tcos<<11)+(B3_Tacc<<8)+(B3_Tcoh<<6)+(B3_Tah<<4)+(B3_Tacp<<2)+(B3_PMC));
```

不然,连 cs8900 ID 号都读不出来。

要成功编译需要花点时间。但这不是难点!只要细心,应该没有问题。

### 11.3 需要添加的东东

参考 c/src/lib/libbsp/arm/edb7312/include/bsp.h,在 c/src/lib/libbsp/arm/gp32/include/bsp.h 中添加 cs8900 相关的东东:

针对 s3c2410 修改或添加 c/src/lib/libbsp/arm/gp32/network/network.c,有下列内容要添加或修改:

①定义 cs8900 基址,即#define CS8900\_BASE 0x19000300

②中断号,查看原理图,cs8900 用的是 EINT9 号中断,所以 cs8900\_isr\_data 数组要改为

```
rtems_irq_connect_data cs8900_isr_data = {BSP_EINT8_23,
                                           (rtems_irq_hdl)cs8900_isr,
                                           cs8900_isr_on,
                                           cs8900_isr_off,
                                           cs8900_isr_is_on,
                                           3, /* unused for ARM cpus */
                                           0 }; /* unused for ARM cpus */
```

③EINT9 中断初始化,我把 EINT9 的中断初始化放在 cs8900\_isr\_on 中,这样可以在安装 isr\_handler 时

初始化 static void cs8900\_isr\_on(const rtems\_irq\_connect\_data \*unused)

```
{
    rGPGCON = (rGPGCON & 0xfffff3)|(1<<3); /*PG1/9 = EINT9*/
    rGPGUP = 0xffff;
    rEXTINT1 = (rEXTINT1 & ~(7<<4)) | 0x4<<4; /*EINT9=high level triggered*/
    rEINTPEND=(1<<9);
    /*rSRCPND = BIT_EINT8_23; /*to clear the previous pending states*/
    /*rINTPND = BIT_EINT8_23;*/
    rEINTMASK = ~(1<<9|0xf);/*enable EINT9*/
    rINTMSK &= ~BIT_EINT8_23; /*enable EINT8_23*/
```

这是本人的草率之作,仅作备忘而已,如有错误或读者发现需要改进之处,还请各位告之。MSN:

[13872051302@monternet.com](mailto:13872051302@monternet.com) Email: [ximenpiaoxue4016@sina.com](mailto:ximenpiaoxue4016@sina.com) (欢迎交流)

```

    return;
}

```

④**中断复位**,即中断发生后要清中断, 如:

```

rtems_isr cs8900_isr(rtems_vector_number v)
{
    rEINTPEND=(1<<9);
    ClearPending(BIT_EINT8_23);
    cs8900_interrupt(v, g_cs);
}

```

针对 s3c2410 修改或添加 c/src/libchip/network/cs8900.c, 有下列内容要添加或修改:

⑤**声明 cs8900\_device 结构体**, 如 static cs8900\_device cs8900;

⑥**attach cs8900\_device 结构体**, 即在函数 cs8900\_driver\_attach 的 if 语句前添加 config->drv\_ctrl=&cs8900;语句, 这条语句必须添加, 不然出现 Driver eth0 already in use.的错误, 即[printf ("Driver '%s' already in use.\n", config->name);]

⑦**应用程序需要添加的东西**,我开始调试的是 rtems-4.6.99.3/testsuites/samples/loopback, 但后来想到要从 PC 端 PING 2410, 就直接换成了调试 netlink 例程 (rtems-4.6.99.3 没有 network 例程, 需要下载其他版本的例程), 但调试了几天也没调试成功, 后来看 rtems-4.6.99.3 下面的 doc 文档 networking.pdf 介绍了 netdemo 例程, 又改成调 netdemo。例程 netdemo 中需要添加 networkconfig.h 文件, 并修改其中的 rtems\_bsdnet\_ifconfig, rtems\_bsdnet\_config 两个结构体.修改后的内容如下:

```

static char ethernet_address[6] = { 0x33, 0x44, 0x55, 0x66, 0x77, 0x88 };
static struct rtems_bsdnet_ifconfig netdriver_config = {
    RTEMS_BSP_NETWORK_DRIVER_NAME,          /* name */
    RTEMS_BSP_NETWORK_DRIVER_ATTACH,       /* attach function */
    NULL,                                    /* link to next interface */
    "192.168.13.88",                          /* IP address */
    "255.255.255.0",                          /* IP net mask */
    ethernet_address,                        /* Ethernet hardware address */
    0                                         /* Use default driver parameters */
};

struct rtems_bsdnet_config rtems_bsdnet_config = {
    &netdriver_config,
    NULL,
    0,                                        /* Default network task priority */
    0,                                        /* Default mbuf capacity */
    0,                                        /* Default mbuf cluster capacity */
    "rtems_host",                            /* Host name */
    "nodomain.com",                          /* Domain name */
    "192.168.13.1",                          /* Gateway */
    "192.168.13.11",                         /* Log host */
    {"192.168.13.88"},                       /* Name server(s) */
};

```

这是本人的草率之作, 仅作备忘而已, 如有错误或读者发现需要改进之处, 还请各位告之。MSN:

[13872051302@monternet.com](mailto:13872051302@monternet.com) Email: [ximenpiaoxue4016@sina.com](mailto:ximenpiaoxue4016@sina.com) (欢迎交流)

## 11.4 ping

开始 ping 板子, 什么反应都没有, 所以找了个抓包程序, 发现 PC 端收到的包格式有错误, 收到内容如下: ff ff ff ff ff 33 44 55 66 77 88 08 06 00 00 00 01 08 00 06 04 00 01 33 44 55 66 77 88 c0 a8 0d 58 00 00 00 00 00 c0 a8 0d 0b 0b 0b 0b 0b 0b...分析后发现在 08 06 后面多了两个字节 00 00, 分析原代码(花了很长时间)发现 sizeof (struct ether\_header)=16(应用到此结构的代码在 cpukit/libnetworking/net/if\_ethersubr.c), ether\_header 定义如下:

```
struct ether_header {
    u_char    ether_dhost[ETHER_ADDR_LEN];/* ETHER_ADDR_LEN =6*/
    u_char    ether_shost[ETHER_ADDR_LEN];/**/
    u_short   ether_type;
};/*见 cpukit/libnetworking/net/Ethernet.h*/
```

按理所 sizeof (struct ether\_header)应为 14 但是我 printf 后的结构却为 16, 显然多出的两个字节在此, 后来问同事说是字节对齐问题, 果然在 ether\_header 结构体后加 \_\_packed \_\_attribute\_\_((packed)) 属性后

```
struct ether_header {
    u_char    ether_dhost[ETHER_ADDR_LEN];/* ETHER_ADDR_LEN =6*/
    u_char    ether_shost[ETHER_ADDR_LEN];/**/
    u_short   ether_type;
}__packed __attribute__((packed));
```

果然 ping 通了。

## 11.5 ping 不稳定

好景不长, 在测试过程中发现 ping 一定数量(不等 100-1200)后, 中断, 任务都不发生了。开始以为是 buffer 不够的问题, 但把所有的 stack heap 加大了 4 倍, 问题还是如此。计数接收字节数, 发现有时候接收 4K 字节就死掉了, 有时候接受 500 多 K 才死掉, 数量也没什么规律, 但有规律的是每次测试都会死掉, 后来我不 ping, 只是让它接受其他广播包, 也是会死掉。还有个特点是, 我把网线拔掉, 只让 CS8900 中的 SWint (bufevent) 发生, 发现 SWint interrupt 和 task 能正常工作, 从这一点似乎能肯定问题是接收中断造成的。

问题还在查找中