

44b0 到 2410 移植

看本文件请参考《都江堰操作系统与嵌入式系统设计》中地 15 章。

因 44b0 和 2410 同属 ARM 系列，指令集相差很小，移植工作主要集中在启动代码、主频配置、与外设相关的东西：时钟嘀嗒、文件系统的芯片驱动、中断系统、键盘扫描、串口驱动上。

Msk 掉的中断，44b 的 pend 寄存器中还有反映，2410 则没有，但在 srcpnd 上有。

1. 配置文件

config.h 文件中，只需要修改主频相关和中断源数量，如下：

```
#define cn_mclk      (152*M) //主频
#define cn_hclk      (cn_mclk/2)    //高速外设时钟
#define cn_pclk      (cn_mclk/4)    //低速外设时钟
#define cn_timer_clk cn_pclk    //定时器输入时钟
#define cn_int_num   32 //2410 有 32 个中断源.
```

2. 启动代码

启动代码即 initcpu.s 文件，该文件在 44b0 和 2410 中分别完成的工作如下：

44b0	2410
在复位及异常向量处放置一条跳转指令。 _start: b reset_start b except_undef b except_swi b except_pabort b except_dabort b . subs pc,lr,#4 subs pc,lr,#4 最后两行对应的是 IRQ 和 FIQ 向量，因 44b0 版本使用了 cpu 的向量中断功能，按 cpu 说明书要求这样写。	在复位及异常向量处放置一条跳转指令。 _start: b reset_start b except_undef b except_swi b except_pabort b except_dabort b . b start_int b . FIQ 向量对应死循环，是因为 djyos 没有使用 2410 的 FIQ 中断
放置中断向量表，向量地址在 int.s 文件中定义。这 26 个向量的放置地址是由 44b0 的中断控制器要求指定的，必须从 0x20 开始依次放置，空缺的地址用 b . 填充。	直接以一条 b start_int 跳至 IRQ 程序入口。
把 cpu 状态设置为 SVC 模式	同左
禁止看门狗，44b0 cpu 内部包含一个硬件看门狗	同左
外设时钟就是主时钟，无需设置	设置外设时钟与主时钟的分频关系
设置倍频器，打开所有外设时钟。注：在节能产品中，不要这样做，应该只打开使用到的设备时钟，在给设备的 driver 中打开。	同左
配置内存控制器参数，这些参数控制了存储器的读写时序	同左
把 sdram 所在的 0xc000000~0xc7ffff 区间设置为可被 cache 区。打开指令和数据 cache。	设置 mmu 页表，djyos 的 si 模式是平板内存模式，按段模式把 4G 存储空间直接映射到物理地址就可以了。 在 sdram 所在的 0x30000000~0x33ffffff 区间设置为可被 cache 区。

	使能 mmu，并打开指令和数据 cache。
设置各模式栈顶地址，该地址作为符号常量在*.ld 文件中赋值。	同左
死循环（硬件仿真调试用） 跳到 pre_load.c 文件的 load_preload 函数处	同左

3. 中断

中断部分是移植中工作量最大的部分，因为 44b 和 2410 的中断控制器差别很大，主要体现在：

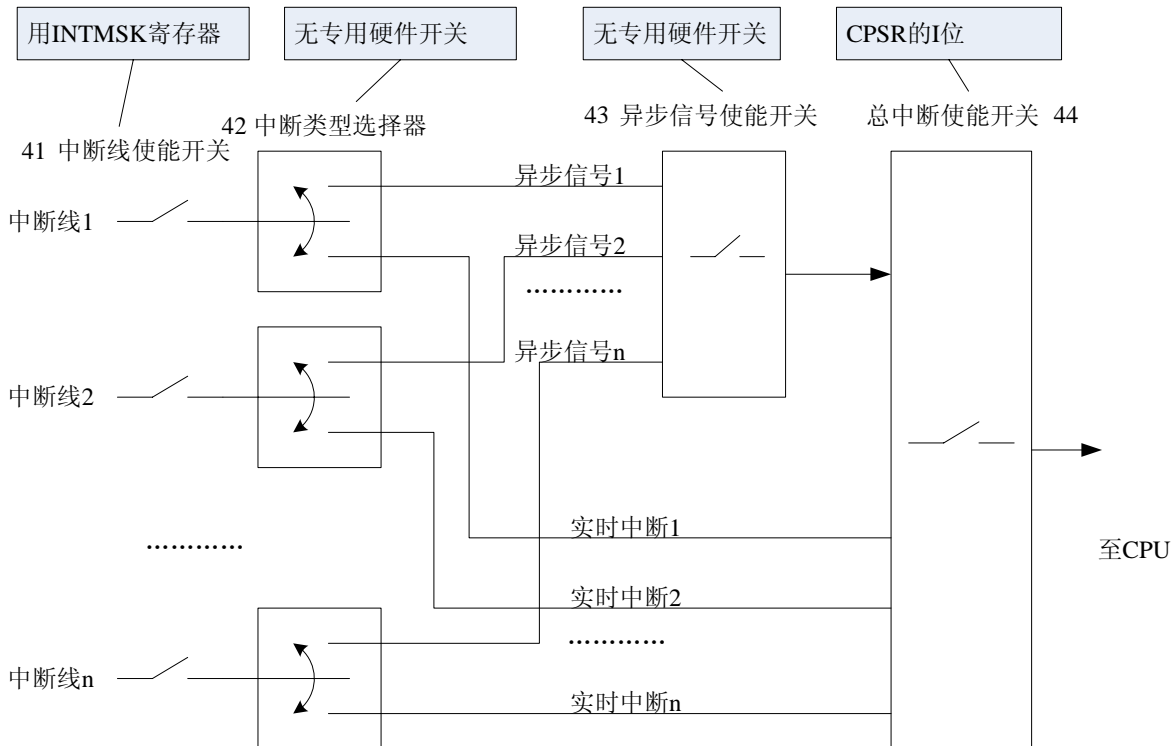
- 1、 44b 提供向量中断模式，cpu 响应中断时直接跳到中断向量地址处。而 2410 则所有 IRQ 共享一个入口地址，但提供一个寄存器 INTOFFSET 表示正在响应的中断的中断号。
- 2、 44b 有一个总开关，2410 则没有。
- 3、 44b 可以通过写入 intpnd 寄存器用软件触发一个中断，2410 则不可以。
- 4、 44b 是通过网 I_ISPC 寄存器的相应位写 1 来清除中断，而 2410 则是网 INTPND 和 SRCPND 寄存器的相应位写 1 清除中断。
- 5、 44b 的和 2410 有许多相同的外设，但其中断线序号和 2410 刚好相反。

44b 和 2410 的中断控制器的共同点：

- 1、 如果中断线被设置成 IRQ，当 cpu 响应 IRQ 时，44b 和 2410 都提供很方便的手段让你获得被响应的中断线序号。如果被设置为 FIQ，则两者都只能通过把 INTPND 寄存器“移位——判零”的方法获取，非常麻烦且慢速。因此，除非你确保只有一个中断线被设置为 FIQ，否则 FIQ 的实际响应速度将远比 IRQ 慢。由于用户可能把不止一个中断线设置为实时中断，所以 djyos 的 44b 版本和 2410 版本均不使用 FIQ 做实时中断。
- 2、 屏蔽中断线的方法相同。

3.1. 中断架构图

2410 和 44b 中断控制器寄存器在操作系统中断架构中的配置是相同的，如下：



3.2. int.s 文件

int.s 文件的责任是：

- 1、保护被中断的线程上下文。
- 2、取得本次中断响应的中断线号。
- 3、把 cpu 模式从 IRQ 切换到 SVC 模式，以支持中断嵌套。
- 4、调用 int.c 文件中用 C 语言实现的中断引擎函数。
- 5、中断引擎函数返回后，恢复 IRQ 状态。
- 6、返回被中断处。

44b 版本和 2410 版本在 step2 处有所不同，其他部分完全一样。44b 使用的是向量方式，中断向量地址中直接获得中断线号，2410 中则从 INTOFFSET 寄存器中取得。

3.3. int.c/h 文件

根据硬件中断控制器的差异，这个文件中需要修改的数据和函数：

指针变量 pg_int_reg 的地址

int_query_line

int_echo_line

int_tap_line

__int_echo_all_line

__int_init_hard

具体代码修改处不大，读者可以直接参考源代码。

int.h 文件中需要修改的有（2410 和 44b 毕竟有相似性，如果是全新 cpu，本文件可能完全重写）：

struct hard_reg_int 结构。

各中断线编号，2410 的中断线多一些，排列顺序也正好与 44b0 相反。

cn_int_msk_all_line 常量

3.4. 功能性差异

44b0 版本实现了 int_tap_line，而 2410 由于硬件不支持，无法实现。

4. 定时器和时钟相关

2410 和 44b0 的定时器操作基本相同，其差异性在于：

- 1、44b0 有 6 个定时器，而 2410 只有 5 个。
- 2、44b0 的 0~3 号定时器可选择预分频器的 1/32 分频为输入时钟，2410 所有定时器只能选择最高 1/16 分频为输入时钟。
- 3、44b0 只有 4#定时器可选 tclk 引脚，5#定时器可选 extclk 引脚为输入时钟，而 2410 的 0#、1#时钟可选 EXTCLK0 引脚，2#、3#、4#时钟可选 EXTCLK1 引脚为输入时钟。
- 4、44b0 有 3 个预分频器，0#1#共用 1 个，2#3#共用 1 个，4#5#共用 1 个；2410 只有 2 个预分频器，0#1#共用 1 个，2#3#4#共用 1 个。

根据以上不同 timer_hard.c 中各函数均有所调整，但调整量很小，在此不一一列出。有兴趣的读者可以直接比较代码，代码中有详细的解释。

4.1. 定时器分配

44b0:

timer3, 用于时钟嘀嗒

Timer4, 用于 nandflash 驱动中延时等待操作结束

timer5, 用于测量 for 循环

2410:

Timer3, 用于时钟嘀嗒和测量 for 循环

Timer1, 用于 nandflash 驱动中延时等待操作结束

4.2. 延时常量测量

延时常量测量就是要给这 4 个变量赋值:

```
volatile static uint32_t    u32g_delay_10uS=40;
```

```
uint32_t u32g_ns_of_u32for; //for(i=j;i>0;i--);语句, i 和 j 都是 u32 型, 每循环纳秒数
```

```
uint32_t u32g_ns_of_u16for; //for(i=j;i>0;i--);语句, i 和 j 都是 u16 型, 每循环纳秒数
```

```
uint32_t u32g_ns_of_u8for; //for(i=j;i>0;i--);语句, i 和 j 都是 u8 型, 每循环纳秒数
```

关于这 4 个变量, 请参阅《都江堰操作系统与嵌入式系统设计》一书的第 1.1.3.2 节。

44b0 用 timer5 测量 for 循环时间, 因 2410 没有 timer5, 改为 timer3。由于 2410 的时钟教快, 故预分频数比 44b0 设的高一些, 44b0 是不分频, 而 2410 设为 8 分频, 以备以后可能升级为 2440。

4.3. tick 初始化

这里本来是移植重点, 但因为 44b0 和 2410 定时器操作的相似性, 两者完全一样。

5. gpio 初始化

gpio 要根据所使用的板子的硬件配置适当设置, 略。

6. 串口驱动

44b0 的 uart 外设和 2410 的 uart 外设也是非常相似, 区别只在中断上:

44b0:

2 个串口, 每个串口的收、发各一个独立的中断线, 两个串口的 error 共用一个中断线。

2410:

3 个串口, 每个串口的收、发、error 共用同一个中断线, 中断控制器的 SUBSRCPND 寄存器指出是哪一个中断。

上述差异使 44b0 版本和 2410 版本在串口驱动的中断处理上有很大的不同。44b0 版本中的串口收发分别用不同的 ISR 函数, 而 2410 则使用同一个 ISR 函数, 需要在中断服务函数中自己识别是哪一个中断。

另外, 由于 44b0 的中断模块支持软件触发中断, 因此在发送数据时, 只要简单地触发中断, 即可启动发送, 然后由发送中断不断地从串口设备的发送缓冲区中取出数据连续发送。而 2410 的中断控制器没有这个功能, 则发送事件服务函数 `uart_send_service` 就必须从串口设备中取出一定数量 (与 fifo 等长) 的数据写入串口 fifo, 才能启动发送。

7. 键盘驱动

键盘驱动中需要修改的是 `key_hard.c` 文件中的

```
uint8_t key_scan_hard(uint16_t *key); 函数
```

这是一个与键盘电路密切相关的函数, 44b0 版本使用的远峰开发板和 2410 版本使用的 s3eb2410 开发板均有 4 个按键, 且都是用口线直接读入, 故两个函数极为相似, 仅口线不一样。

8. 文件系统

移植文件系统也是与具体板件相关而不是 cpu 相关的部分，即使相同的 cpu，板件的硬件配置不一样时，也需要移植。

File.c/h 是文件系统的公共部分，任何移植都不需要修改的。

Flashfile.c/h 是 flash 文件系统的公共部分，无论目标板提供的是 nandflash 还是 norflash，都不需要修改 Flashfile.c/h。

需要修改的是芯片驱动程序，移植要考虑的因素有：

- 1、板上的 flash 芯片配置。
- 2、Cpu 与 flash 芯片如何接口。

远峰板和英蓓特板的不同点为：

44b0	2410
法拉盛容量，16M	Flash 容量：64M
Cpu 接口：44b0 不含读写 nandflash 的外设，用总线模拟读写 nandflash	2410 内含读写 nandflash 的外设，读写 flash 是通过访问这个外设来进行的。

移植的工作就是，重写 44b0 版本中直接访问 flash 芯片的部分，44b0 版本和 2410 版本的区别如下：

8.1. 写入芯片地址

nandflash 内部存储阵列的地址不是用 cpu 的地址寻址的，而是用一系列写入操作从数据端口写进去的，写入地址的方式，44b0 版本和 2410 版本有很大的不同。

44b0 版本用下列语句序列把地址写入芯片：

```
address_start();           //开始写入地址
*pg_nand_address = (uint8_t)address;      // A0 ~ A7
*pg_nand_address = (uint8_t)(address >> 9); // A9 ~ A16
*pg_nand_address = (uint8_t)(address >> 17); // A17 ~ A22
address_end();           //完成写入地址
```

地址位 A8 包含在命令中，不通过地址位写入，读者可参考 nandflash 芯片手册。

2410 版本用下列语句序列把地址写入芯片：

```
pg_nand_reg->NFADDR = (uint8_t)addr;
pg_nand_reg->NFADDR = (uint8_t)(addr>>9);
pg_nand_reg->NFADDR = (uint8_t)(addr>>17);
pg_nand_reg->NFADDR = (uint8_t)(addr>>25);
```

因 44b0 版本的 flash 容量比较小，故地址用 3 个字节表示，而 2410 用 4 个字节表示，这种区别与板件配置的 nandflash 芯片有关，与 CPU 无关。

8.2. 写入命令字

nandflash 通过写入命令字的方式告诉芯片 cpu 希望执行的操作是什么，两者写入命令字的方式也是不一样的。

44b0 版本用下列语句序列把命令字写入芯片：

```
nand_cle_port |= nand_cle_bit;
*pg_nand_address = val;
nand_cle_port &= ~nand_cle_bit;
```

2410 版本只用 1 条语句把命令字写入芯片：

```
pg_nand_reg->NFCMD = cmd;
```

8.3. 读取和写入数据

两个版本从 nandflash 芯片读取和写入数据的方式也不同，区别在于：

44b0 版本读写数据的方法：

```
data = *pg_nand_address;           //读取数据
```

```
*pg_nand_address = data;          //写入数据
```

2410 版本读写数据的方法：

```
data = pg_nand_reg->NFDATA;       //读取数据
```

```
pg_nand_reg->NFDATA = data;       //写入数据
```