S3C2440 外部中断操作

这两天在调试 2440 外部中断的时候，通过 jlink 来调试，老是进不了中断。因为借鉴了网上很多程序，感觉不应该是程序的问题。后来通过 USB 口，利用 supervivi 的 download&run 功能，把编译产生的 bin 文件下载到内存中。超级终端就提示：Dummy_isr error, interrupt number: 5, INTMSK = 0xffff16ff 已是网上查了这个错误。才知道是什么原因。因为 SDRAM 的起始地址为 0x30000000，把程序烧写进 SDRAM，起始地址不为 0 而不程序中断后,PC 默认指向 0x00-0x1C，内部 SRAM 的地址 0 处都不存在你要调试的程序的中断向量表，所以不能正确将中断引导到你编写的中断服务程序中来，一旦中断发生，程序必然会跑飞。同时 ARM 内核会通过串口打印"Dummy_isr error....."信息。

解决办法：

（1） 直接烧写到 nand 第 0 块（如果代码大于 4K，启动代码中必须要有拷贝到 SDRAM 功能）

（2） 改写自己的程序，在初始化中断之前将中断向量表拷贝到内部 S8/31/2010RAM.

（3） 利用 MMU 的重定向功能

首先，这里先来了解下 S3C2440 中断相关的寄存器。

1.中断分两大类：内部中断和外部中断。

2.外部中断。24 个外部中断占用 GPF0-GPF7（EINT0-EINT7），GPG0-GPG15（EINT8-EINT23）。用这些脚做中断输入，则必须配置引脚为中断，并且不要上拉。具体参考 datesheet 数据手册。

寄存器：EXTINT0-EXTINT2：三个寄存器设定 EINT0-EINT23 的触发方式。

EINTFLT0-EINTFLT3：控制滤波时钟和滤波宽度。

EINTPEND：这个是中断挂起寄存器，清除时要写 1，后面还有几个是写 1 清除。当一个外部中断（EINT4-EINT23）发生后，那么相应的位会被置 1。为什么没有 EINT0-EINT3，呵呵，看看 SRCPND 就知道了，里面没有 EINT4-EINT23 的位子，所以有了 EINTPEND。

EINTMASK：这个简单，是屏蔽中断用的，也就是说位为 1 时，此次中断无效。

3.内部中断。内部中断有 8 个寄存器，下面逐一来看。

寄存器：SUBSRCPND：当一个中断发生后，那么相应的位会被置 1，表示一个中断发生了。

INTSUBMSK：与上一个是一伙的，中断屏蔽寄存器，具体屏蔽什么，自己看手册去吧。

INTMOD：中断的方式。一个中断可以是普通中断，也可以是快中断，在这里设置，但只能有一个快中断。

PRIORITY ：优先级寄存器，不说了。

SRCPND ：当一个中断发生后，那么相应的位会被置 1，表示一个或一类中断发生了。

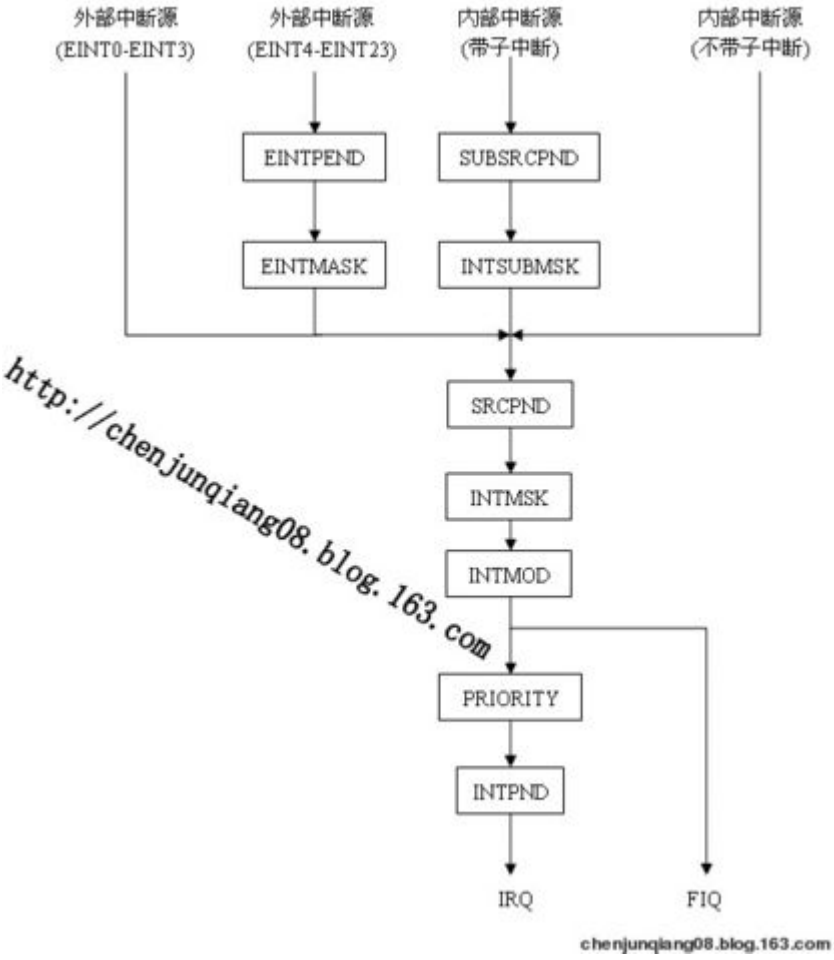
INTMSK ：中断屏蔽寄存器。

INTPND ：中断发生后，SRCPND 中会有位置 1，可能好几个（因为同时可能发生几个中断），这些中断会由优先级仲裁器选出一个最紧迫的，然后

吧把 INTPND 中相应位置 1，所以同一时间只有一位是 1。也就是说前面的寄存器置 1 是表示发生了，只有 INTPND 置 1，CPU 才会处理。

INTOFFSET ：用来表示 INTPND 中哪一位置 1 了，好让你查询，普通中断跳转时查询用。清除 INTPND、SRCPND 时自动清除。

4.各寄存器关系：



下面看图说明：

5.中断过程。

a 如果是不带子中断的内部中断：发生后 SRCPND 相应位置 1，如果没有被 INTMSK 屏蔽，那么等待进一步处理。

b 如果是带子中断的内部中断：发生后 SUBSRCPND 相应位置 1，如果没有被 INTSUBMSK 屏蔽，那么 SRCPND 相应位置 1，等待进一步处理，几个 SUBSRCPND 可能对应同一个 SRCPND，对应表如下：

SRCPND	SUBSRCPND
INT_UART0	INT_RXD0,INT_TXD0,INT_ERR0
INT_UART1	INT_RXD1,INT_TXD1,INT_ERR1
INT_UART2	INT_RXD2,INT_TXD2,INT_ERR2
INT_ADC	INT_ADC_S, INT_TC
INT_CAM	INT_CAM_C, INT_CAM_P
INT_WDT_AC97	INT_WDT, INT_AC97

c 如果是外部中断：EINT0-EINT3 发生后 SRCPND 相应位置 1，如果没有被 INTMSK 屏蔽，那么等待进一步处理。EINT4-EINT23 发生后 EINTPEND 相应位置 1，如果没有被 EINTMASK 屏蔽，那么 SRCPND 相应位 EINT4-7 或 EINT8-23 置 1，如果没有被 INTMSK 屏蔽，等待进一步处理，几个 EINTPEND 对应同一个 SRCPND，对应表如下：

SRCPND	EINTPEND
EINT0	EINT0
EINT1	EINT1
EINT2	EINT2
EINT3	EINT3
EINT4-7	EINT4-EINT4
EINT8-23	EINT8-EINT23

三种中断都等待进一步处理了。接下来从 SRCPND 往下看，看 INTMSK。如果中断被屏蔽了，就不用说了（注意：快中断也能被屏蔽）。如果没有被屏蔽，那么会进一步到 INTMOD。如果是快中断，那么直接出来，进入 FIQ（即 CPU 进入快中断模式处理）。如果是普通中断，那么 SRCPND 可以有多为置 1（FIQ 只能有一个），这时就会经过 PRIORITY 选出一个优先级高的，然后把根据选出的中断把 INTPND 相应位置 1（注意：只能选出一个），进入 IRQ，让 CPU 处理。

6.中断的开启。

a.如果是不带子中断的内部中断，只需设置 INTMSK，让它不屏蔽中断就可以了。

b 如果是带子中断的内部中断，需设置 INTSUBMSK 和 INTMSK，让它门不屏蔽中断就可以了。

c 如果是外部中断，对于 EINT8-23 需要设置 EINTMASK 和 INTMSK。对于 EINT0-EINT3 只需设置 INTMSK。

7.中断的清除。

a.如果是不带子中断的内部中断，只需清除 SRCPND，注意清除需位置 1。

b 如果是带子中断的内部中断，需清除 SRCPND 和 SUBSRCPND，注意先清除 SUBSRCPND，再清除 SRCPND。因为，如果你先清除 SRCPND 的话，然后在清除 SUBSRCPND 的过程中，SRCPND 会以为又有中断发生，又会置 1。也就是说一次中断会响应两次。所以必须先掐断源头。

c 如果是外部中断，对于 EINT8-23 需要清除 EINTPEND 和 SRCPND（同样注意顺序）。对于 EINT0-EINT3 只需清除 SRCPND。

当然这些寄存器，更具体的介绍请看 S3C2440 的 datasheet- -。

介绍了这么多，下面来点实战的吧。又是贴代码，呜呜~其实不然，如果你能把别人的代码变成自己的，那就是学会了，而且借鉴别人的代码来学，这样学会快很多。(纯属个人观点)

在这里采用了第 1 种方法，而且这里的代码量小于 4k，这里就没有涉及到拷贝功能。关于 MMU 功能和拷贝功能，后面我会补上，因为现在还没有学到那~

/*

文件：main.c

功能：按下 KEY1~KEY4 按键时，对应的 LED 改变状态。按下 KEY5/KEY6 时

LED 分别全亮和全暗

```
*/
/*
文件：main.c
功能：按下 KEY1~KEY4 按键时，对应的 LED 改变状态。按下 KEY5/KEY6 时
LED 分别全亮和全暗
*/
#include "def.h"
#include "option.h"
#include "2440addr.h"
#include "2440lib.h"
#include "2440slib.h"

 /*
------------------------------------------------------------
按键  |   对应的 GPGx   |   对应的 EINTx   |  对应的 EINTPEND 值
------------------------------------------------------------
K1    |      GPG0       |      EINT8       |     0x100
K2    |      GPG3       |      EINT11      |     0x800
K3    |      GPG5       |      EINT13      |     0x2000
K4    |      GPG6       |      EINT14      |     0x4000
K5    |      GPG7       |      EINT15      |     0x8000
K6    |      GPG11      |      EINT19      |     0x80000
------------------------------------------------------------
*/
//=================================
#define LED_ON() (rGPBDAT &= ~(0xf<<5))
#define LED1_ON() (rGPBDAT &= ~(0x1<<5))
#define LED2_ON() (rGPBDAT &= ~(0x1<<6))
#define LED3_ON() (rGPBDAT &= ~(0x1<<7))
#define LED4_ON() (rGPBDAT &= ~(0x1<<8))

#define LED_OFF() (rGPBDAT |= (0xf<<5))
#define LED1_OFF() (rGPBDAT |= (0x1<<5))
#define LED2_OFF() (rGPBDAT |= (0x1<<6))
#define LED3_OFF() (rGPBDAT |= (0x1<<7))
#define LED4_OFF() (rGPBDAT |= (0x1<<8))

//------延迟函数----
void dely(U32 tt)
{
   U32 i;
   for(;tt>0;tt--)
   {
```

```
      for(i=0;i<10000;i++){}
    }
}


//-----IO 口初始化--------
void GPIO_Init(void)
{
 rGPGCON=0x80A882;//          按          键          对          应
//GPG0/GPG3/GPG5/GPG6/GPG7/GPG11
 //rGPGUP = 0x0;
 rGPBCON = 0x155555;//LED 对应 GPB5/GPB6/GPB7/GPB8,设为输出
口
 //rGPBUP = 0x1E0;

}

//-------key 中断服务-------
static void __irq EINT_KEY(void)
{
 //清中断，向对应位写 1 可清 0。
 U32 a,b,c;
 a = rSRCPND;
 rSRCPND = a;
 b = rINTPND;
 rINTPND = b;
 c = rEINTPEND;
 rEINTPEND = c;

 //判断那个按键被按下，控制对应的 LED 状态
 switch(c)
 {
  case 0x100:
         if(rGPBDAT & 0x20)
               LED1_ON();
         else
               LED1_OFF();
         break;
  case 0x800:
         if(rGPBDAT & 0x40)
               LED2_ON();
         else
               LED2_OFF();
         break;
```

```
  case 0x2000:
        if(rGPBDAT & 0x80)
              LED3_ON();
        else
              LED3_OFF();
        break;
  case 0x4000:
        if(rGPBDAT & 0x100)
              LED4_ON();
        else
              LED4_OFF();
        break;
  case 0x8000:
        LED_ON();
        break;
  case 0x80000:
        LED_OFF();
        break;


 }

}

//------------中断初始化----------
void EINT_Init(void)
{
 rSRCPND = 0x00000020;//提前对其置 1(即使其为 0,为 0 即未请求),防止不必要的中断
 rINTMSK = 0xffffffdf;//屏蔽除 EINT8_23 以外的中断源,这里跟 SRCPND 相反,为 0 则中断服务有效
 rEXTINT1 = 0x22202002;//设置 EINT8、11、13、14、15 下降沿触发
 rEXTINT2 = 0x2000;//EINT19 下降沿触发,all Filter Disable
 rEINTMASK &= 0xf716ff;//允许 EINT8、11、13、14、15、19 中断
 rINTMOD = 0x0;//设置为 IRQ 模式
 pISR_EINT8_23 = (unsigned)EINT_KEY;// 中 断 向 量 注册,pISR_EINT8_23 在 240addr.h 里定义了
 //对应启动文件里的中断表,对照着程序数一下：“HandleReset”为 0x00，“HandleUndef”为 0x04，
 //“HandleSWI”为 0x08，“HandlePabort”为 0x0C，“HandleDabort”为 0x10，“HandleReserved”为 0x14，
 //“HandleIRQ”为 0x18，“HandleFIQ”为 0x1C，所以“HandleEINT0”即外部中断 0 为 0x20。
}
```

```
//------主函数-----------
int Main(void)
{
 U8 key;
 U32 mpll_val=0;
 mpll_val = (92<<12)|(1<<4)|(1);

 //init FCLK=400M, so change MPLL first
 ChangeMPllValue((mpll_val>>12)&0xff,       (mpll_val>>4)&0x3f,
mpll_val&3);
 ChangeClockDivider(key, 12);


    GPIO_Init();
    EINT_Init();
    LED_OFF();
    while(1)
    {

    }

   return 0;
}
```