STM32 中 BIT\_BAND（位段/位带）和别名区使用入门

(2012-03-20 10:39:34)

[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载▼](javascript:;)

|  |  |
| --- | --- |
|  |  |

**一、 什么是位段和别名区**

    是这样的，记得MCS51吗？ MCS51就是有位操作，以一位（BIT）为数据对象的操作，MCS51可以简单的将P1口的第2位独立操作： P1.2=0;P1.2=1 ；这样就把P1口的第三个脚（BIT2）置0置1。而现在STM32的位段、位带别名区就为了实现这样的功能。

    它的对象可以是SRAM、I/O和外设空间。要实现对这些地方的某一位的操作。它是这样做的：在寻址空间（32位对应的地址空间为 4GB ）的另一地方，取个别名区空间，从这个地址开始处，每一个字（32BIT）对应SRAM或I/O的一位。  
       
    这样，1MB SRAM 就可以有 32MB 的对应别名区空间，就是1位膨胀到32位（1 BIT 变为1个字节）。我们对这个别名区空间内的某一字操作（置0或置1），就等于它映射的 SRAM 或 I/O 相应的某地址的某一位的操作。

**二、使用位段的好处**       
    简单来说，可以把代码缩小， 速度更快，效率更高，更安全。 一般操作要6条指令，而使用位带别名区只要4条指令。一般操作是 读－改－写的方式， 而位带别名区是 写 操作。防止中断对 读－改－写 的方式的影响。

**三、应用说明**

    支持了位带操作（bit\_band），有两个区中实现了位带。其中一个是SRAM 区的最低1MB 范围，第二个则是片内外设区的最低1MB 范围。这两个区中的地址除了可以像普通的RAM 一样使用外，它们还都有自己的“位带别名区”，位带别名区把每个比特膨胀成一个 32 位的字。 每个比特膨胀成一个32 位的字，就是把 1M 扩展为 32M 。

    于是，位于 RAM 地址 0X200000000 的一个字节扩展为8个32 位的字，扩展后每位相对应的的地址是：0X220000000，0X220000004，0X220000008，0X22000000C,0X220000010,0X220000014, 0X220000018,0X22000001C

    支持位带操作的两个内存区的范围是：

0x2000 0000‐0x200F FFFF（SRAM 区中的最低1MB）  
0x4000 0000‐0x400F FFFF（片上外设区中的最低1MB）

    对 SRAM 位带区的某个比特，记该比特所在字节的地址为A，位序号为 n (0<=n<=7)，则它在别名区的地址为：

AliasAddr ＝ 0x22000000 + ((A‐0x20000000)\*8+n)\*4 =0x22000000 + (A‐0x20000000)\*32 + n\*4

对于片上外设位带区的某个比特，记该比特所在字节的地址为A，位序号为 n (0<=n<=7)，则该比特在别名区的地址为：

AliasAddr ＝ 0x42000000 + ((A‐0x40000000)\*8+n)\*4 = 0x42000000 + (A‐0x40000000)\*32 + n\*4

上式中，“\*4”表示一个字为 4 个字节，“\*8”表示一个字节中有 8 个比特。

    把“位带地址＋位序号”转换别名地址宏为：

#define BITBAND(addr, bitnum) ((addr & 0xF0000000)+0x2000000 + ((addr &0xFF FFF)<<5) + (bitnum<<2))

    把该地址转换成一个指针：

#define MEM\_ADDR(addr) \*((volatile unsigned long \*)(addr))  
MEM\_ADDR(BITBAND( (u32)&CRCValue,1)) = 0x1;

例如点亮LED

使用STM32库：  
   GPIO\_ResetBits(GPIOC, GPIO\_Pin\_4); //关LED5  
   GPIO\_SetBits(GPIOC, GPIO\_Pin\_7);   //开LED2

一般读操作：  
    STM32\_Gpioc\_Regs->bsrr.bit.BR4 =1;// 1：清除对应的ODRy位为0  
    STM32\_Gpioc\_Regs->bsrr.bit.BS7 =1;// 1：设置对应的ODRy位为1

如果使用位带别名区操作：  
STM32\_BB\_Gpioc\_Regs->BSRR.BR[4] =1;// 1：清除对应的ODRy位为0  
STM32\_BB\_Gpioc\_Regs->BSRR.BS[7] =1;// 1：设置对应的ODRy位为1

代码比STM32库高效十倍 ！

对内存变量的位操作：

SRAM 变量：long CRCValue;

把“位带地址＋位序号”转换别名地址宏：

#define BITBAND(addr, bitnum) ((addr & 0xF0000000)+0x2000000+((addr &0xFFFFF)<<5)+(bitnum<<2))

把该地址转换成一个指针：

#define MEM\_ADDR(addr) \*((volatile unsigned long \*)(addr))

对32位变量 的BIT1 置 1 ：

MEM\_ADDR(BITBAND( (u32)&CRCValue,1)) = 0x1;

对任意一位( 第23位 ) 判断：

if(MEM\_ADDR(BITBAND( (u32)&CRCValue,23))==1)  
{

}

**四、Cortex-M3中关于位段的定义**

    Cortex-M3 存储器映像包括两个位段(bit-band)区。这两个位段区将别名存储器区中的每个字映射到位段存储器区的一个位，在别名存储区写入一个字具有对位段区的目标位执行读-改-写操 作的相同效果。

    所有STM32F10x外设寄存器都被映射到一个位段(bit-band)区。这个特性在各个函数中对单个比特进行置1/置0操作时被大量使用，用以减小和优化代码尺寸。

映射公式映射

公式给出别名区中的每个字是如何对应位带区的相应位的，公式如下：

bit\_word\_offset = (byte\_offset x 32) + (bit\_number × 4)  
bit\_word\_addr = bit\_band\_base + bit\_word\_offset

其中：  
bit\_word\_offset是目标位在存取器位段区中的位置  
bit\_word\_addr 是别名存储器区中字的地址，它映射到某个目标位。  
bit\_band\_base 是别名区的起始地址。  
byte\_offset 是包含目标位的字节在位段中的序号  
bit\_number 是目标位所在位置（0-31）

