MSP430程序库<十三>硬件乘法器使用

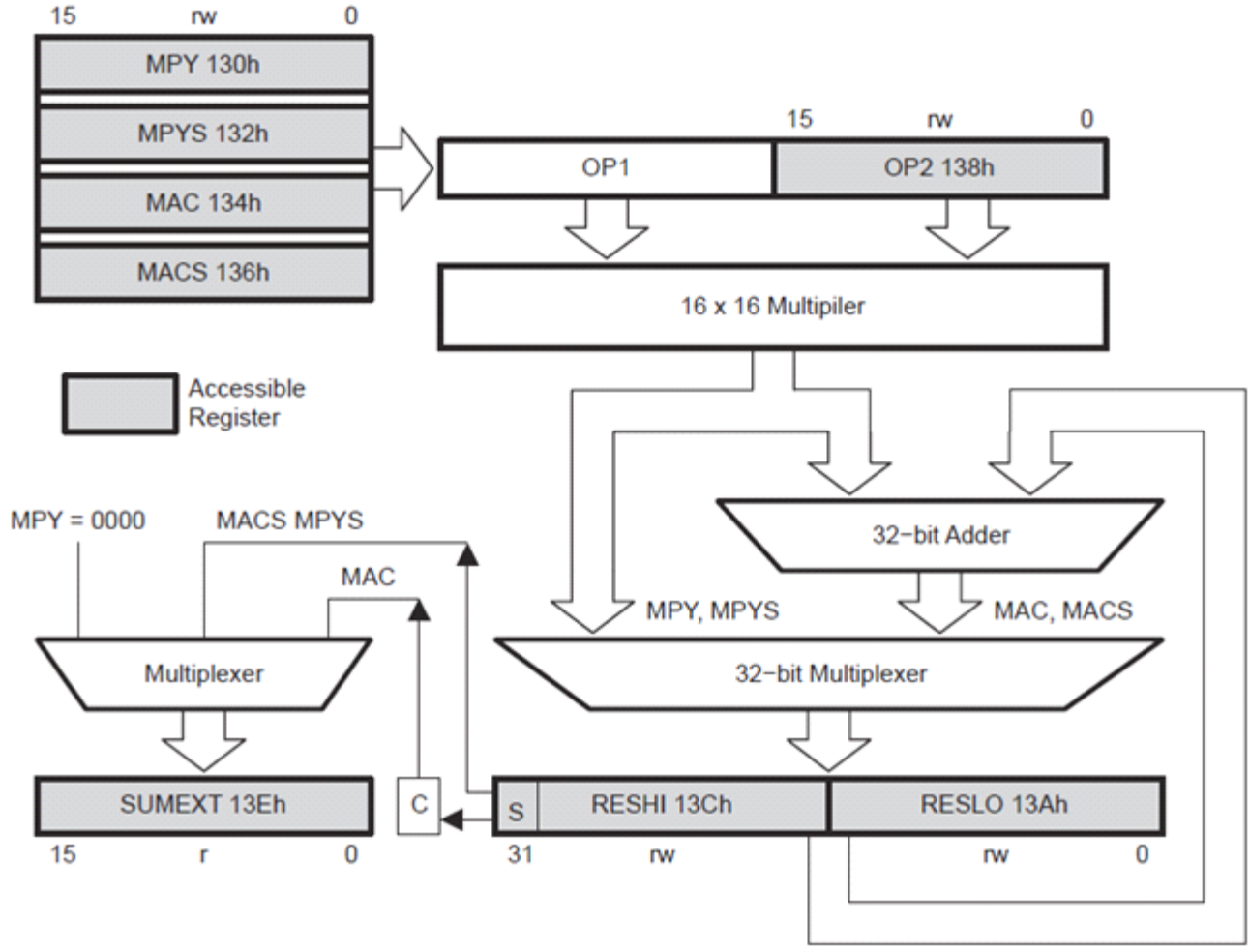
硬件乘法器不占用 CPU 周期，有硬件实现，速度比软件实现的乘法速度快很多。msp430f14x、msp430f16x 中都含有硬件乘法器模块，方便用户需要速度的时候使用。

1.硬件介绍：

在 MSP430系列单片机中，硬件乘法器是外围模块，而不是 CPU 内核的一部分；所以它的活动与否与 CPU 的活动与否无关，它的寄存器和其他的外围寄存器一样通过 CPU 指令读写。

硬件乘法器模块支持一下功能：无符号乘法、有符号乘法、无符号乘加、有符号乘加；可以支持16*16 16*8 8*16 8*8bits 的乘法。

硬件乘法器的模块框图如下：



硬件乘法器模块的四种操作类型(无符号乘法、有符号乘法、无符号乘加、有符号乘加)是由写入的第一个操作数的位置决定的。这个模块有两个操作数寄存器：OP1和 OP2、三个结果寄存器 RESLO, RESHI, 和 SUMEXT。RESLO 寄存器存储结果的低字(低16位)；RESHI 寄存器存储结果的高字(高16位)；SUMEXT 寄存器存储结果的有关信息。结果在3个时钟周期后即可完成；写入 OP2后的下一条指令即可读取结果，有一种情况例外：用间接寻址方式访问结果。用间接寻址方式访问结果时，读取结果之前需要有一条 NOP 指令。

操作数 OP1有四个地址(MPY:0130h MPYS:0132h MAC:0134h MACS:0136h)，这四

个寄存器用来选择乘法的操作模式。写入第一个操作数寄存器决定用哪种操作：无符号 用符号等，但是不启动相乘操作；写入第二个操作数寄存器启动相乘的操作。计算完成后结果存入寄存器 RESLO,RESHI, 和 SUMEXT。

操作数1的四个地址对应的操作：

OP1 Address	Register Name	Operation
0130h	MPY	Unsigned multiply（无符号乘法）
0132h	MPYS	Signed multiply（有符号乘法）
0134h	MAC	Unsigned multiply accumulate（无符号乘加）
0136h	MACS	Signed multiply accumulate（有符号乘加）

四种操作模式下高位结果寄存器的内容如下：

Mode	RESHI Contents
MPY	Upper 16-bits of the result
MPYS	The MSB is the sign of the result. The remaining bits are the upper 15-bits of the result. Two's complement notation is usedfor the result.
MAC	Upper 16-bits of the result
MACS	Upper 16-bits of the result. Two's complement notation is used for the result.

四种操作模式 SUMEXT 寄存器的内容：

Mode	SUMEXT
MPY	SUMEXT is always 0000h
MPYS	SUMEXT contains the extended sign of the result 00000h Result was positive or zero 0FFFFh Result was negative
MAC	SUMEXT contains the carry of the result 0000h No carry for result 0001h Result has a carry
MACS	SUMEXT contains the extended sign of the result 00000h Result was positive or zero 0FFFFh Result was negative

连续乘法运算时，如果操作数1不需改变就可以运算，则可以不需要重新写入和以保存内容相同的数；但 OP2必须重新写入以启动乘法运算。

MACS Underflow and Overflow（MACS 时的上溢和下溢）：硬件乘法器不检测有符号乘加时运算结果的上溢出和下溢出。结果的正数范围：0到7FFF FFFFh；负数范围：0FFFF FFFFh 到8000 0000h。下溢出是两个负数的和结果寄存器得到的是正数，上溢出是两个正数的和结果寄存器得到的是负数。SUMEXT 寄存器存储有结果的符号，可以根据它判断是否溢出（0000h 负数和 则上溢 0FFFFh 正数和 则下溢）。使用时 程序必须合适的检测、处理 MACS 的溢出情况。

2.程序示例(用户指南上给出的汇编示例)：

所有乘数模式的例子如下。所有的8x8模式使用的寄存器的绝对地址，因为汇编器将不允许 B 访问到字寄存器时使用标准定义的文件标签。

```
; 16x16 Unsigned Multiply
MOV #01234h,&MPY ; Load first operand
MOV #05678h,&OP2 ; Load second operand
; ... ; Process results
; 8x8 Unsigned Multiply. Absolute addressing.
MOV.B #012h,&0130h ; Load first operand
MOV.B #034h,&0138h ; Load 2nd operand
; ... ; Process results
; 16x16 Signed Multiply
MOV #01234h,&MPYS ; Load first operand
MOV #05678h,&OP2 ; Load 2nd operand
; ... ; Process results
; 8x8 Signed Multiply. Absolute addressing.
MOV.B #012h,&0132h ; Load first operand
SXT &MPYS ; Sign extend first operand
MOV.B #034h,&0138h ; Load 2nd operand
SXT &OP2 ; Sign extend 2nd operand
; (triggers 2nd multiplication)
; ... ; Process results
; 16x16 Unsigned Multiply Accumulate
MOV #01234h,&MAC ; Load first operand
MOV #05678h,&OP2 ; Load 2nd operand
; ... ; Process results
; 8x8 Unsigned Multiply Accumulate. Absolute addressing
MOV.B #012h,&0134h ; Load first operand
MOV.B #034h,&0138h ; Load 2nd operand
; ... ; Process results
; 16x16 Signed Multiply Accumulate
MOV #01234h,&MACS ; Load first operand
MOV #05678h,&OP2 ; Load 2nd operand
; ... ; Process results
; 8x8 Signed Multiply Accumulate. Absolute addressing
MOV.B #012h,&0136h ; Load first operand
SXT &MACS ; Sign extend first operand
MOV.B #034h,R5 ; Temp. location for 2nd operand
SXT R5 ; Sign extend 2nd operand
MOV R5,&OP2 ; Load 2nd operand
; ... ; Process results
```

上面的程序虽然和标准的汇编差异比较大，但是有一定汇编基础的人还是很容易就能够看懂。这里的程序给出了多种方式写入操作数寄存器。

间接寻址结果寄存器时，在写入 OP2操作数启动乘法后，至少需要一个指令的延迟后才能

访问结果寄存器 RESLO 等；直接寻址时可以写入 OP2后，下一条指令即可读取结果。示例程序（汇编）：

```
; Access multiplier results with indirect addressing
MOV #RESLO,R5 ; RESLO address in R5 for indirect
MOV &OPER1,&MPY ; Load 1st operand
MOV &OPER2,&OP2 ; Load 2nd operand
NOP ; Need one cycle 写入两个操作数 乘法运算开始后 需要一个 NOP
MOV @R5+,&xxx ; Move RESLO
MOV @R5,&xxx ; Move RESHI
```

如果在写入 OP1和写入 OP2之间产生了中断，中断响应后，源操作数的计算模式丢失；运算结果不确定。为了避免这种情况的发生，在写入操作数时禁止中断或在中断响应函数中不使用硬件乘法器。如：

```
; Disable interrupts before using the hardware multiplier
DINT ; Disable interrupts
NOP ; Required for DINT
MOV #xxh,&MPY ; Load 1st operand
MOV #xxh,&OP2 ; Load 2nd operand
EINT ; Interrupts may be enable before
; Process results
```

硬件部分就说这么多了，有什么不大明白的可以参考用户指南。

3.使用示例：

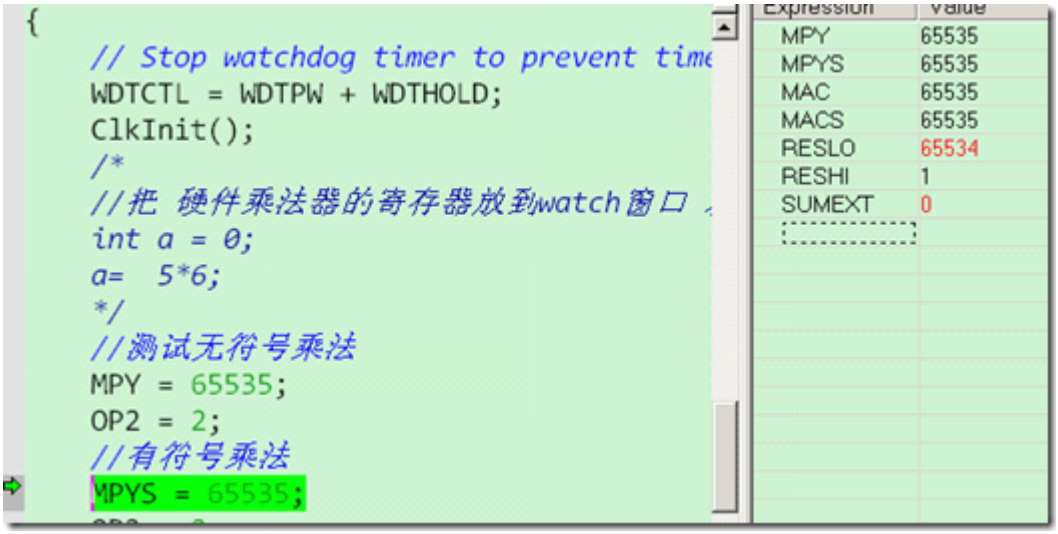
我的程序仅仅是用 C 语言演示硬件乘法器的使用。程序主要内容如下：

```
#include<msp430x16x.h>
/*************************************************************
***************
* 名      称：main 主程序
* 功      能：硬件乘法器程序库使用演示
* 入口参数：无
* 出口参数：无
*************************************************************
**************/
  voidmain( void)
  {
   // Stop watchdog timer to prevent time out reset
     WDTCTL = WDTPW + WDTHOLD;
      ClkInit();

  /*把 硬件乘法器的寄存器放到 watch 窗口 观察是否变化
  int a = 0;
  a=   5*6;
  */
```

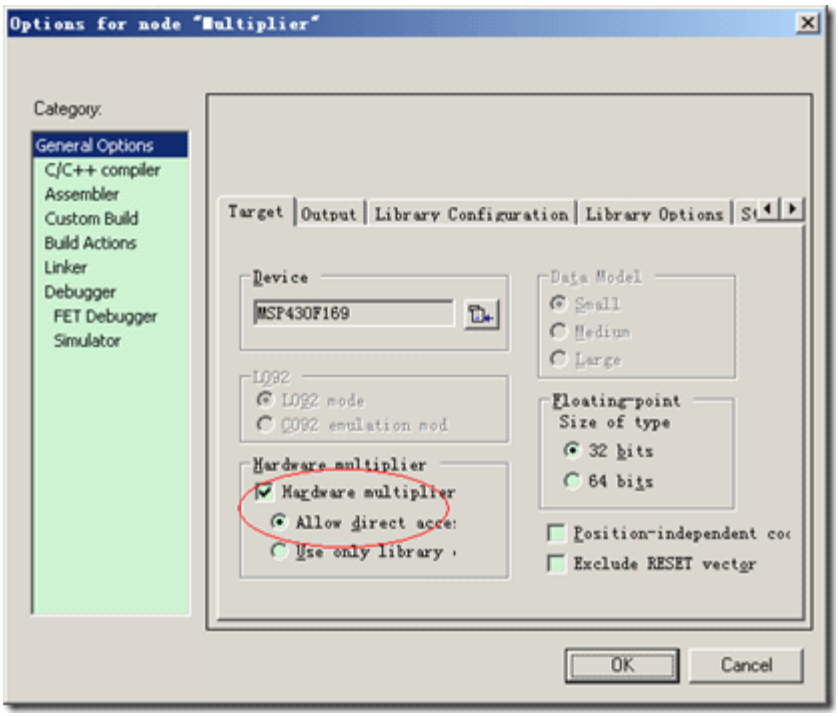
```
    //测试无符号乘法
      MPY = 65535;
       OP2 = 2;
    //有符号乘法
      MPYS = 65535;
       OP2 = 2;
    //无符号乘加
      MAC = 65535;
       OP2 = 2;
    //有符号乘加
      MACS = 65535;
       OP2 = 2;
       LPM0;
}
```

程序演示了4中乘法模式：使用时单步调试，观察硬件乘法器的有关寄存器。如：

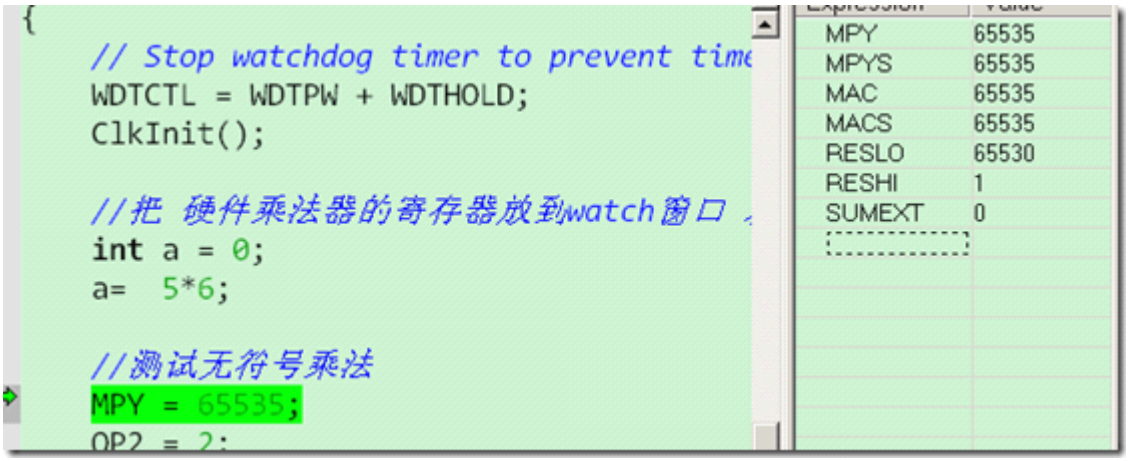


硬件乘法器运算速度很快，只需3个时钟周期；这里 IAR 单步调试时，OP2赋值结束，在 watch 窗口马上就可以看到运算结果。其他三种模式类似。

注释掉的这部分是我用来检测 IAR 编译程序是否使用硬件乘法器进行测试。默认情况下，乘法应该是用硬件乘法器运算的。默认的设置如下：



硬件乘法器是选中的，这时应该是使用硬件乘法器的，但是我的调试结果显示它没有使用硬件乘法器，截图下：



运行后乘法器相关位没有对应变化，如果使用的话，应该变化。

硬件乘法器不选中时，寄存器也没有相应变化，从这看，IAR 没有使用硬件乘法器；也许程序没有优化太多或是 debug 版本不使用硬件乘法器。

如果需要直接使用硬件乘法器，有必要时把设置的硬件乘法器去掉，以防冲突。

下面是直接使用硬件乘法器的一个实例：

```
#include"msp430x16x.h"
    unsigned intResult[7];
    unsigned charData1[7];
    unsigned charData2[7];
    voidmain(void)
    {
        unsigned chari;
```

```
  WDTCTL = WDTPW + WDTHOLD; // 关看门狗
    for(i=0; i<7; i++)
     {
      Data1[i] = 10 * i; // 对两数组赋值
        Data2[i] = 25 * i;
     }
     for(i=0; i<7; i++)
     {
         MPY = Data1[i];
         OP2 = Data2[i];
      _NOP(); // 延迟
        _NOP();
         _NOP();
      Result[i] = RESLO; // 保存结果，由于是8×8型，所以未用到 RESHI；
    }
}
```

这个程序用无符号乘法运算，结果存入结果数组中。值的注意的是程序中的3个 NOP，这里 NOP 不需要，根据头文件推测，IAR 编译器应该使用的是直接寻址方式，可以不要。如果不太放心，一个 NOP 即可，即便用的是间接寻址，一个 NOP 的延迟已经足够。

硬件乘法器一般不会像上面的程序那么使用，如果这样就太浪费了；还不如直接用 *操作符来的简便；硬件乘法器主要用来对时间要求苛刻的情况。如：用430进行数字滤波，快速傅里叶变换等。ti 有一篇应用笔记介绍的就是用 msp430f169实现数字滤波方案。

硬件乘法器就到这里了。

相关文章及附件下载：http://www.ideyi.org/bbs/article_1077_374375.html