MSP430程序库<十>ADC12模块

msp430内部含有 ADC12模块，可以完成12位的模数转换，当对精度或其他指标要求不高时，可以选用430单片机内部的 ADC12完成模数转换工作。这里主要实现了一个比较通用的 ADC12模块初始化程序，具体的数据存储和处理需要自己在中断处理函数中添加。

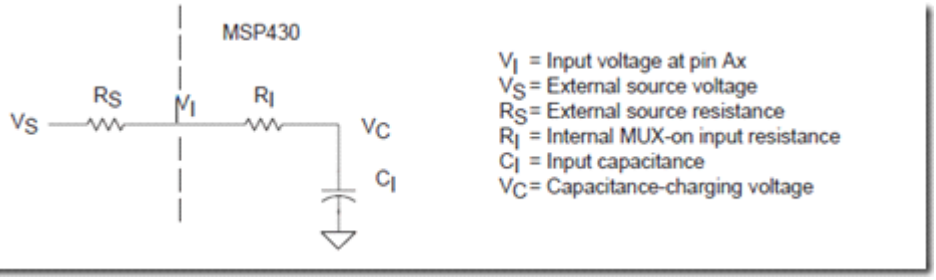
1.硬件介绍：

msp430单片机内的 ADC12模块的特点如下：12位转换精度，1位非线形误差，1位非线形积分误差；多种时钟源给 ADC12模块，切本身自带时钟发生器；内置温度传感器；TimerA/TimerB 硬件触发器；8路外部通道和4路内部通道；内置参考电压源和6种参考电压组合；4种模式的模数转换；16bit 的转换缓存；ADC12关闭支持超低功耗；采用速度快，最高200Kbps；自动扫描和 DMA 使能。430内部的 ADC12功能还是蛮强大的，可以有定时器触发模数转换开始，还可以和内部的 DMA 模块共同使用，完成高速的采样转储等高级功能。

这个 AD 的转化公式如下，可以根据它计算采样的模拟电压值：

$$N_{ADC} = 4095 \times \frac{Vin - V_{R-}}{V_{R+} - V_{R-}}$$

使用 AD 是还要注意采样时间，430单片机的模数 ADC12模块的等效模拟电压输入电路如下：



其中 V_S 是信号源电压，R_S 是信号源内阻，V_I 在 Ax(ADC12模块模拟输入端)上的电压，R_I 单片机内多路开关等效电阻，V_C 是保持电容上的电压(ADC12模块采样的电压)，C_I 是电容的值。需要根据这些值计算采样时间：

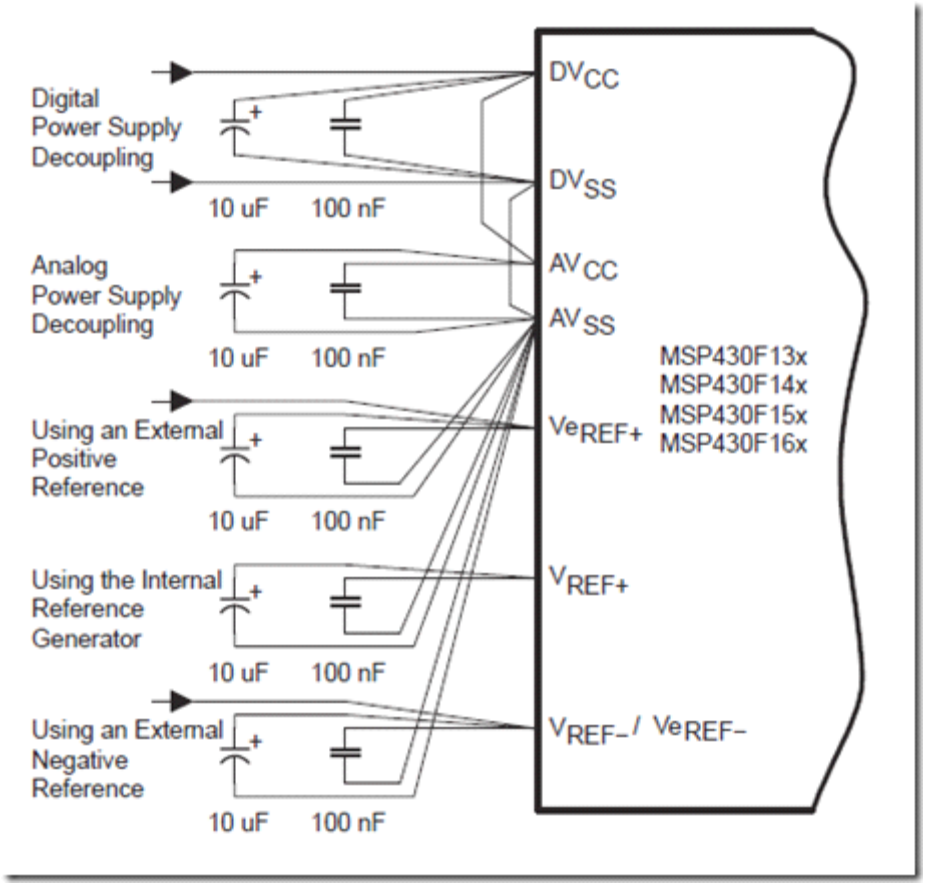
$$t_{sample} > (R_S + R_I) \times \ln(2^{13}) \times C_I + 800ns$$

代入单片机上的参数后公式如下：

$$t_{sample} > (R_S + 2k\Omega) \times 9.011 \times 40pF + 800ns$$

我的程序中采样时间设的是4us，可以算出如果用我的程序(不更改采样时间)的话，最大信号源内阻可以是6.8k，当信号源内阻更大时，可以自己按要求设采样时间(在程序的初始化函数内的寄存器设置部分)。

还有，ADC 模数转换时要求参考电压等很稳定，为了达到这个要求，德州仪器要求这部分的电路如下：



即：所有参考源和电源均并联一组 0.1uF 和10uF 的电容。

硬件部分就说这么多了；如果需要更详细的说明，参考用户指南。

2.程序实现：

程序主要实现的是一个比较通用的初始化程序，内容如下：

```
charADC12Init(charn,charchannels[],charrep)
{
    if(n>15)
        return0;
 //SHT0_0
  ADC12CTL0 = ADC12ON + MSC + SHT0_0 + REFON + REF2_5V;// 开启 ad,
参考电压2.5v
  ADC12CTL1 = SHP + ADC12SSEL_3;                    //Use sampling timer,
SMCLK

   for(inti = 0;i < n;i++)
    {
        if(channels[i] >= 0x80)
            return0;
     *(char*)(ADC12MCTL0_ + i) = channels[i];      //每个 MCTL 设置
   }
  *(char*)(ADC12MCTL0_ + n - 1) |= EOS;             //序列结束
```

```
  if(rep != 0)                                    //多次转换
    {
        ADC12CTL1 |= CONSEQ_3;
     }
   else
    {
        ADC12CTL1 |= CONSEQ_1;
     }

   ADC12IE = 1<<(n-1);                                  // Enable
ADC12IFG.n-1
    return1;
  }
```

程序先判断 n 通道总数是否超过了可用的个数，超过则返回零然后设置 ADC12CTL0和 ADC12CTL1中不需要特殊设置的部分，然后在设置通道模式(根据 rep 参数的值)；for 循环设置的是每个存储寄存器的设置 ADC12MCTLx ；*(char*)(ADC12MCTL0_ + n - 1) |= EOS; //序列结束这句加入序列结束标志；最后设置中断寄存器并返回成功设置标志。其中比较特殊的是 ADC12MCTL0_，这个是430提供的头文件中定义的 ADC12MCTL0的地址值，以其为指针首址操作 ADCMCTLx 寄存器，从而利用循环设置寄存器的内容，大量减少了代码行数。

参数 channels[]是每个存储寄存器的设置(除 EOS 位之外的)，含义如下：

```
channels[]:对应通道设置，高四位，参考源选择；
低四位，通道选择。具体如下：
SREFx Bits
6-4
Select reference
000 VR+ = AVCC and VR. = AVSS
001 VR+ = VREF+ and VR. = AVSS
010 VR+ = VeREF+ and VR. = AVSS
011 VR+ = VeREF+ and VR. = AVSS
100 VR+ = AVCC and VR. = VREF./ VeREF.
101 VR+ = VREF+ and VR. = VREF./ VeREF.
110 VR+ = VeREF+ and VR. = VREF./ VeREF.
111 VR+ = VeREF+ and VR. = VREF./ VeREF.
INCHx Bits
3-0
Input channel select
0000 A0
0001 A1
0010 A2
0011 A3
0100 A4
0101 A5
```

```
0110 A6
0111 A7
1000 VeREF+
1001 VREF./VeREF.
1010 Temperature sensor
1011 (AVCC – AVSS) / 2
1100 (AVCC – AVSS) / 2
1101 (AVCC – AVSS) / 2
1110 (AVCC – AVSS) / 2
1111 (AVCC – AVSS) / 2
```

这是从用户指南里复制来的，每一位和 ADC12MCTLx 的意义相同(去掉 EOS 位)，所以可用宏定义来制定这个参数，如：

```
charchannels[3];
channels[0] = SREF_1+INCH_0;
channels[1] = SREF_1+INCH_1;
channels[2] = SREF_1+INCH_2;
ADC12Init(3,channels,1);
```

这是3个通道 A0-A2采样，多次采样。

启动转换函数：

```
voidADC12Start()
{
    ADC12CTL0 |= ENC;
    ADC12CTL0 |= ADC12SC;
}
```

ADC 初始化完成后，调用此函数开始 AD 转换，转换完成后(一个序列通道，如：刚才的0-2)，程序自动进入 AD 中断，用户需要在这里为自己的函数添加处理逻辑；这里只存储了转化的结果：

```
#pragmavector=ADC_VECTOR
__interrupt voidADC12ISR (void)
{
    static inti;
  results[0][i] = ADC12MEM0;                  // Move results, IFG is cleared
 results[1][i] = ADC12MEM1;                  // Move results, IFG is cleared
 results[2][i] = ADC12MEM2;                  // Move results, IFG is cleared
    i++;
  if(i>31)                                  //多次转换时  转换次数
    {
      //多次重复采样时，在这里方处理函数
     ADC12CTL0 &=~ ENC;                          //停止转换
        i=0;
     }
```

```
}
```

该程序实现的是多次 A0-A2 32次转换，把结果存入 results 数组。单次时，仅仅采样一次(A0-A2)可用自己更改处理函数。

程序部分就完成了，调用时注意要自己实现处理逻辑或存储逻辑。

3.使用示例：

本程序使用方式还是加入 C 文件，包含 H 文件；不过和之前的程序不同的是要自己实现中断处理逻辑。

使用示例参见程序库中的 ADC12.

```
#include<msp430x16x.h>
#include"ADC12.h"
  voidmain( void)
  {
   // Stop watchdog timer to prevent time out reset
     WDTCTL = WDTPW + WDTHOLD;
      ClkInit();
      charchannels[3];
      channels[0] = SREF_1+INCH_0;
      channels[1] = SREF_1+INCH_1;
      channels[2] = SREF_1+INCH_2;
      ADC12Init(3,channels,1);
      _EINT();
      ADC12Start();
      LPM0;
  }
```

这里实现的是3通道多次转换，参考电压都是内部参考电压。自己实现的处理逻辑参见前面的程序实现的最后一部分。

ADC12模块部分就到这里了。

相关文章及附件下载：http://www.ideyi.org/bbs/article_1077_372221.html