MSP430程序库<六>通过 SPI 操作 AD7708

AD7708是16位的Σ-Δ型 AD 转换芯片，在低频应用中，AD7708可以作为单电源供电的完整前端。AD7708内部含有一个 PGA（可编程增益放大器），可以完成对信号的放大；PGA 范围是2^0-2^8八档可编程，当取参考电压2.5v 时可以测量量程20mv 到2.56v 的电压。AD7708和 AD7718引脚功能完全一样，只是位数 AD7718是24位的，若用7718，只需改动少部分的程序，16位的部分改成24位即可。AD7708是通过 SPI 接口通信的；程序使用前一篇实现的 SPI 程序与 AD 芯片通信，可以作为 SPI 详细的使用示例。

1.硬件介绍：

硬件主要是 MSP430的 SPI 接口和 AD7708芯片的说用说明。

msp430的 SPI 接口：支持主机模式和从机模式，且始终的极性和相位可调，在于 AD 转换芯片通信的时候，需要极性一致。有关 msp430的 SPI 的详细介绍，参考：*MSP430程序库<五>SPI 同步串行通信*。

AD7718 的外部引脚有28 个。按性质主要分为模拟、数字两个部分。模拟部分引脚有模拟输入、参考电压输入和模拟电源三类。模拟输入引脚可以配置为8通道或10通道的伪差分输入，他们共同参考 AINCOM 端。

数字部分引脚有 SPI 接口、数据就绪、通用 I/O 口和数字电源四类。SPI 接口的4 根标准信号线分别是片选信号 CS 、串行时钟输入 SCLK、串行数据输入 DIN 和串行数据输出 DOUT。当 AD7718接在 SPI 总线上时是从器件，从引脚 CS 输入低电平信号使能 AD7718。数据就绪 RDY 是一个低电平有效的输出引脚。当所选通道数据寄存器中有有效数据时，输出低电平信号；数据被读出后，输出高电平。AD7718 的通用 I/O 口是2 个一位口 P1 和 P2。它们既可配置成输入也可配置成输出，单片机通过 SPI 口读写 AD7718 片内相关寄存器实现对 P1 和 P2 的操作。它们扩展了单片机的 I/O 接口能力。

AD7718 的模拟电源和数字电源是分别供电的，都既可以采用+3V 供电，也可以采用+5V 供电。但必须一致，要么都用+3V，要么都用+5V。

AD7708和 AD7718是通过一组片内寄存器控制和配置的。这些寄存器的第一个是通信寄存器，它是用来控制转换器的所有操作。这些部件的所有通信必须先写通信寄存器指定要执行的下一个操作。上电或复位后，设备默认等待写通信寄存器。 STATUS 寄存器包含转换器的操作条件的有关信息。 STATUS 寄存器是只读寄存器。模式寄存器用于配置转换模式，校准，斩波（chop）启用/禁用，参考电压选择，通道配置和伪差分 AINCOM 模拟输入操作时的缓冲或无缓冲。模式寄存器是一个读/写寄存器。 ADC 控制寄存器是一个读/写寄存器，用来选择活动的通道和编码输入范围和双极性/单极性操作。I/O 控制寄存器是一个读/写寄存器，用于配置了2个 I/O 端口的操作。滤波寄存器是一个读/写寄存器，用于编码转换器的数据更新率。 ADC 数据寄存器是一个只读寄存器，它包含在所选通道上的一个数据转换的结果。 ADC 的失调寄存器读/写寄存器包含偏移校准数据。有五个偏移寄存器，每个全差分输入通道之一。当配置为伪差分输入模式下的通道共用偏移寄存器。 ADC 增益寄存器是读/写寄存器，包含增益校准数据。有5个 ADC 增益寄存器，每个全差分输入通道之一。当配置为伪差分输入模式通道共享增益寄存器。该 ADC 包含工厂使用的测试寄存器，用户应不改变这些寄存器的操作条件。 ID 寄存器是一个只读寄存器，用于硅识别目的。

我用的硬件连线方式：430的 P3.0接 AD7708的 CS 端，P3.1-P3.2接对应的 AD 芯片的 SPI 口；RDY 信号没有接；所以，程序使用的是查询方式，等待 STATUS 寄存器的 RDY 位指示转换完成。

有关 AD7708的详细信息可以参考它的 datasheet；另外我对数据手册的寄存器部分和程序流程的部分进行了翻译，如果需要，可以在本博客底部的**附件**中下载。

2.程序实现：

首先是对 AD7708的读写寄存器函数，AD7708的每次操作都以写通信寄存器开始，通过这一步，指示下一步将进行什么操作；有关寄存器每一位的意义，参考附件(博客结尾)中的 AD7708-寄存器

写入寄存器：

```
voidAD7708WriteRegister(charaddr,longdat)
{
 SpiWriteData(addr);      //写通信寄存器，通知下个操作：写 addr 寄存器
if(IsLong[addr])          //如果是16位寄存器, 7718则24位若移植要改 if 内语句
   {
        SpiWriteData(dat>>8);
    }
 SpiWriteData(0xFF&dat);         //写入低位数据
}
```

寄存器地址，可以查阅 datasheet 或我翻译的那部分；IsLong 字符数组指示对应的寄存器是8位还是16位的：

```
charIsLong[16] = {0,0,0,0,1,1,1,0,0,0,0,0,1,1,0,0};
```

读取寄存器：

```
longAD7708ReadRegister(charaddr)
{
 charh = 0,l = 0;              //高低字节数据
SpiWriteData(0x40|addr);     //写通信寄存器，通知下个操作：读 addr 寄存器
   if(IsLong[addr])
    {
        h = SpiWriteData(0xFF);
    }
    l = SpiWriteData(0xFF);
    return((unsigned int)h<<8)|l;
}
```

SPI 解释：430是 SPI 主机模块，当发送的时候，同时，另外一个时钟沿采样接收，所以，每次发送完成后的半个周期，均可得到读出的数据；所以 SpiWriteData 函数写入的同时返回同时收到的字符。发送0xFF 即是为提供读取即将到来的数据提供时钟，详细可以参考上一篇的注意事项部分（刚更新的）。

读取结果数据：

```
longAD7708ReadResultData()
{
 while((AD7708ReadRegister(0x00)&0x80)==0); //等待转换完成
   returnAD7708ReadRegister(0x04);
}
```

等待 STATUS 的 RDY 位变高(AD 数据转换更新完成)，读取 data 寄存器的内容。

校准：校准的过程在 datasheet 中有详细的流程图；可以参考 datasheet 或者附件中的 AD7708-寄存器，这个子函数只完成一个通道的校准，通道地址有参数输入，方便调用：

```
voidAD7708Cal(charchannel)
{
    adccon = (adccon&0x0f)|(channel<<4);
 mode = (mode&0xf8)|0x04;                    //内部0校准
AD7708WriteRegister(0x02,adccon);          //ADC 控制寄存器，channel 通道
AD7708WriteRegister(0x01,mode);             //模式寄存器
while((AD7708ReadRegister(0x01)&0x07)!=0x01);    //等待校准完成

mode = (mode&0xf8)|0x05;                    //内部 满标度校准
AD7708WriteRegister(0x01,mode);             //模式寄存器
while((AD7708ReadRegister(0x01)&0x07)!=0x01);    //等待校准完成
}
```

adccon 是程序记录的前一次输入的 ADCCON 寄存器的内容，mode 是程序记录的上一次输入的 MODE 寄存器的内容，因为串口读取需要时间，为了获取更快的速度，程序记录了这两个变量，以供使用。通道地址参考 datasheet，或附件中的文档。

初始化：

```
voidAD7708Init(charchop)
{
P3DIR|=BIT0;
P3OUT&=~BIT0;                               //CS 选中
 //主机模式，115200,8位数据位，三线模式，时钟模式1（具体见 spi.c）
SpiMasterInit(115200,8,3,1);          //时钟不是准确的115200（具体见 spi.c）
_EINT();                                   //开中断，spi 读写程序要需要中断

   charfilter;
    adccon = 0x0f;
    if(chop == 0)
    {
     filter = 0x03;                    //滤波寄存器设为最小值，可以更改
    mode = 0x91;                        //斩波禁止，10通道，无缓冲，空闲模式
   }
 else
   {
     filter = 0x0D;                    //滤波寄存器设为最小值，可以更改
    mode = 0x11;                        //斩波启用，10通道，无缓冲，空闲模式
   }

 AD7708WriteRegister(0x07,0x00);       //IO 寄存器，不用==
AD7708WriteRegister(0x03,filter);    //滤波寄存器
AD7708WriteRegister(0x02,0x0F);       //ADC 控制寄存器，0通道，单极性
```

```
AD7708WriteRegister(0x01,mode);       //模式寄存器
   if(chop == 0)
        for(inti = 0; i<5;i++)
        {
         //校准，因只有5个失调寄存器，多的就会覆盖之前的，只校准5个即可
           AD7708Cal(5);
        }

    _DINT();
}
```

初始化制引入了斩波这一个参数，其他的均使用固定的参数：10通道伪差分、单极性、无缓冲、滤波寄存器设为斩波或禁止斩波时候的最快速度，需要的话可以自行修改。SPI 初始化之后开中断，目的是向 AD 写内容以初始化 AD。初始化完成后关中断，为了让程序库的初始化后一致，但调用这个函数后，需要开中断，才能正常使用 AD 采样的其它函数。

采样启动：本程序只支持了单词采样的开始，若需要连续模式的，可以自行实现（比较容易实现：只需更改寄存器的值即可）：

```
voidAD7708Start(charchannel)
{
    adccon = (adccon&0x0f)|(channel<<4);
    mode = (mode&0xf8)|0x02;
    AD7708WriteRegister(0x02,adccon);
    AD7708WriteRegister(0x01,mode);
}
```

根据之前一次的控制寄存器和模式寄存器的设置，更改现在需要的值，写入相应寄存器即可。

到此，程序部分完成，需要扩展，可以自行添加。

3.使用说明：

使用时，只需加入 AD7708.c，文件包含 AD7708.h，然后就可以正常使用本程序提供的函数；具体可以参考示例工程和其中的 main.c 文件。

```
longa;
voidmain()
{
 // Stop watchdog timer to prevent time out reset
   WDTCTL = WDTPW + WDTHOLD;
    ClkInit();
 AD7708Init(0);  //禁止斩波  1时启用斩波

_EINT();          //开中断，程序需要用到 SPI 的中断；冲突，可以更改 SPI 函数

while(1)                        //串口测试
   {
        AD7708Start(0);
     a = AD7708ReadResultData();          //读取 AD 采样后的结果
```

```
        //电压计算方法：a*1.024*2.5(参考电压)/65535
       a = AD7708ReadRegister(0);            //去状态值，此处函数不需要用

    }
}
```

AD7708的程序库（简化，其他需求可以自行添加：有读写寄存器的函数之后，添加其他功能比较简单）已经完成。

相关文章及附件下载：http://www.ideyi.org/bbs/article_1077_370084.html