AN-759
应用笔记

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106 • Tel: 781/329-4700 • Fax: 781/326-8703 • www.analog.com

扩展ADuC8xx及ADuC702x系列的DAC输出数

作者：Aude Richard

简介

ADI公司推出的MicroConverter®系列产品集成了单路电压输出和多路电压输出的DAC。

在某些应用中，可能需要额外的模拟输出通道。因此，MicroConverter必须外接一个分立的多通道DAC。将MicroConverter与ADI的小封装，兼容SPI的多通道DAC相连可以很容易的实现电压输出通道的扩展。

本应用笔记描述了如何通过一个简单的三线式SPI接口实现ADuC814或ADuC7020与四通道电压输出8-/10-/12-位精度DAC产品AD5304 / AD5314 / AD5324 (AD53x4)的连接。

AD53x4

AD53x4(图1)是一系列带输出电压缓冲器的四通道8-/10-/12-位精度DAC，采用10引脚的微型SOIC封装，单电源供电范围2.5 V至5.5 V，3 V电源供电时消耗500 μA电流。其片内输出放大器能以0.7 V/μs的压摆率实现轨到轨的输出摆幅。采用一个三线式串行接口，时钟速率最高可达30 MHz，且与标准SPI接口兼容。更多详细内容，请参阅AD5304数据手册。

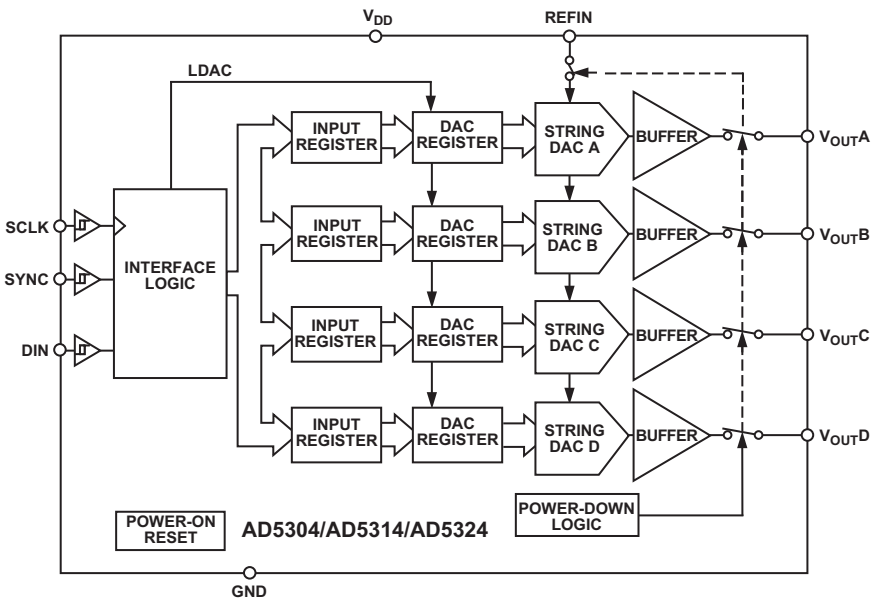

图1. AD53x4功能框图

硬件接口

SPI接口

图2显示的是AD53x4与MicroConverter之间的串行接口。由于AD53x4为从器件，MicroConverter被配置为主器件，即为AD53x4提供SCLK。MOSI(主发从收)引脚连接至DAC的串行数据线(DIN)。AD53x4的SYNC输入，即片选信号，可与ADuC814的一个位可编程引脚相连，比如P3.4。或者与ADuC7020的一个通用目的I/O相连，比如P1.7。

AD53x4输入移位寄存器为16位，当SYNC为低时，该器件要接收来自MicroConverter的两字节数据。

ADuC814上没有从器件选择或片选输出，必须使用一个GPIO。对于ADuC7020，SPI接口提供一个从器件选择或片选输出，但每发完一个字节，都会变为高电平，因而需要用一个GPIO来控制外部DAC的SYNC信号。发送数据时先发送高位。

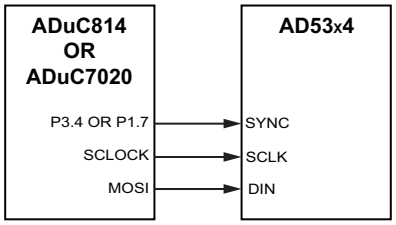

图2 AD53x4与ADuC814/ADuC7020接口

利用MICROCONVERTER的基准电压

如有需要，AD53x4可利用MicroConverter内部的基准电压。

- ADuC814提供一个片上2.5 V精密带隙基准。该2.5 V内部基准电压经工厂校准，绝对精度为2.5V ±2.5%。如果将该内部基准电压用于AD53x4，则应在CREF引脚至AGND处进行缓冲，如图3所示。采用5 V电源供电时，该内部基准电压的典型噪声性能为150 nV/Hz@1 kHz，直流噪声峰峰值为100 mV。
- ADuC7020提供一个2.5 V片上基准电压，经工厂校准，绝对精度为2.5 V ±10 mV。如果将该内部基准电压用于AD53x4，则应在VREF引脚处进行缓冲，并须用一个470 nF电容将该引脚连接至AGND。

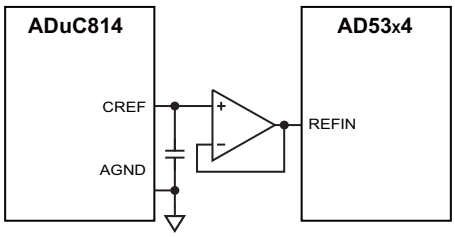

图3 使用ADuC814基准输出

ADuC814软件接口

列表1是从随附代码中选取的与AD53x4接口相关的一段代码。

SPI接口在init814函数内进行初始化，对各种模式参数进行设置，以满足AD53x4的SPI时序要求。

ad53x4out为接口函数，首先，对来自给定参数的16位数据的高位字节进行处理，以载入AD53x4输入移位寄存器的高位字节(有关数据格式内容，请参阅AD5304/AD5314/AD5324数据手册)。

然后通过spiTx函数发送该字节。随后发送低位字节，从而完成16位数据传输。注意ad53x4cs，就是AD53x4的SYNC信号，在SPI的两个连续字节传输过程中保持不变。

spiTx函数，被ad53x4out调用，通过将字节数据写入SPIDAT寄存器来发送txDAT参数所含字节数据。该函数然后等待ISPI标志位被置位，以示传输结束。注意，在传输繁忙时，硬件不会自动重置ISPI位，因此必须在写入SPIDAT之前通过软件进行设置。另需注意的是，spiTx函数必须在确认传输结束之后返回到调用函数，以确保调用函数在数据传输结束后而非数据传输过程中对SYNC信号取反。

```
sbit ad53x4cs = P3^4;                              // Chip select for ad53x4 (PORT3.4)

void init814(void)                                 // Initialize internal peripherals
 {
  /* Initialize other peripherals here */

  /* Initialize SPI to talk to AD53x4 */
  CFG814 = 0x01;                                   // Serial interface enable for P3.5..P3.7 pins
  SPICON = 0x38;                                   // Enable SPI I/F as master, SCLOCK idle H,
                                                   // advance MSB output, sclock=fcore/2=1.05Mhz
 }

void spiTx(unsigned char txDat)                    // Transmit a byte data over the SPI
 {
  ISPI = 0;                                        // Clear ISPI bit
  SPIDAT = txDat;
  while(!ISPI);                                    // Wait until tx complete
 }

void ad53x4out(unsigned char adrs,                 // A1,A0 bit of the contents
               bit pdN,                            // PD bit of the contents
               bit ldacN,                          // LDAC bit of the contents
               unsigned short dat)                 // 12-bit output data
 {
  unsigned char txDat;

  ad53x4cs = 0;

  txDat = ((unsigned char) (dat>>8)) & 0x0f;
  txDat |= ldacN ? 0x10 : 0x00;
  txDat |= pdN ? 0x20 : 0x00;
  txDat |= (adrs<<6);
  spiTx(txDat);                                    // Tx the upper byte

  txDat = (unsigned char) dat;
  spiTx(txDat);                                    // Tx the lower byte

  ad53x4cs = 1;
 }
```

列表1 摘自随附代码53x4forADuC814.C的相关代码段

ADuC7020软件接口

列表2是从随附代码ADuC7020toAD53x4.c中选取的与AD53x4和ADuC7020接口相关的一段代码。

SPI接口在init702x函数内进行初始化，对各种模式参数进行设置，以满足AD53x4的SPI时序要求。

AD53x4out为接口函数，首先，对来自给定参数的16位数据的高位字节进行处理，以载入AD53x4输入移位寄存器的高位字节(有关数据格式内容，请参阅AD5304/AD5314/AD5324数据手册）。

然后通过spiTx函数发送该字节。随后发送低位字节，从而完成16位数据传输。

注意P1.7，就是AD53x4的SYNC信号，在SPI的两个连续字节传输过程中保持不变。

spiTx函数，被ad53x4out调用，通过将字节数据写入SPITX寄存器来发送txDat参数所含字节数据。该函数然后等待，直到SPISTA寄存器的SPIRX满标志位(位4)被置位，以示传输结束。注意，必须用软件通过读取SPIRX寄存器来清除SPIRX满标志位。

AN05247–0–2/05(0)

```
void init702x(void){
    GP1DAT = 0x80800000;            // Configure P1.7 as output
    GP1SET = 0x00800000;            // and pull /SYNC High
    GP1CON = 0x02020000;            // Configure P1.6 and P1.4 in SPI mode
    SPICON = 0x4B;                  // Configure SPI as Master, clock idles high
    SPIDIV = 0x14;                  // Set the DIV to run at 1.05Mhz
}

void spiTx(unsigned char txDat){
    SPITX = txDat;                  // Send txDat to the AD5304
    while(!(SPISTA & 0x08));        // Wait of end of transfer
    txDat = SPIRX;                  // Read in dummy data from SPIRX to prevent overflow
}

void ad53x4out(unsigned short address, unsigned short pdN, unsigned short ldacN, unsigned short dat){

    txDat1 = 0;                     // Reset txDat1 to 0
    txDat0 = 0;                     // Reset txDat0 to 0

    //configure the high byte to be sent
    txDat1 |= (address << 6);       // Store the address in the upper 2 bits of txDat1 (8 bit)
    txDat1 |= pdN ? 0x20 : 0x00;    // If pdN is set, copy a logic 1 into the next space,
                                    // if not copy a logic 0
    txDat1 |= ldacN ? 0x10 : 0x00;  // If ldacN is set, copy a 1 into the next space, if not copy a 0
    txDat1 |= (dat >> 4);           // Copy the first 4 bits of data into the last 4 bits of txDat1

    //configure the low byte to be sent
    txDat0 |= (dat << 4);           // Copy the last 4 bits of data into the first 4 bits of txDat0
    txDat0 &= 0xf0;                 // Insure the first 4 bits of data in txDat0 are empty

    //send the data
    GP1CLR = 0x800000;              // Pull /SYNC High
    spiTx(txDat1);                  // Send txDat1 to the SPI
    spiTx(txDat0);                  // Send txDat0 to the SPI
    GP1SET = 0x800000;              // Pull /SYNC Low
}
```

列表2 摘自随附代码ADuC7020to53x4.c的相关代码段