第11章 AT89S51单片机与DAC、ADC的接口

在单片机测控系统中，非电量如温度、压力、流量、速度等，经传感器先转换成连续变化的模拟电信号（电压或电流），然后再将模拟电信号转换成数字量后才能在单片机中进行处理。实现模拟量转换成数字量的器件称为ADC（A/D转换器）。

单片机处理完毕的数字量，有时根据控制要求需要转换为模拟信号输出。数字量转换成模拟量的器件称为DAC（D/A转换器）。本章从应用的角度，介绍典型的ADC、DAC芯片与AT89S51单片机的硬件接口设计以及接口驱动程序设计。

11.1 单片机扩展DAC概述

单片机只能输出数字量，但是对于某些控制场合，常常需要输出模拟量，例如直流电动机的转速控制。下面介绍单片机如何扩展DAC。

1．D/A转换器简介

购买和使用D/A转换器时，要注意有关D/A转换器选择的几个问题。

（1）D/A转换器的输出形式

D/A转换器有两种输出形式：电压输出和电流输出。。

（2）D/A转换器与单片机的接口形式

单片机与D/A转换器的连接，早期多采用8位的并行传输的接口，现在除了并行接口外，带有串行口的D/A转换器品种也不断增多，目前较为流行多采用SPI串行接口。

2．主要技术指标

D/A转换器的指标很多，设计者最关心的几个指标如下。

（1）分辨率

分辨率指单片机输入给D/A转换器的单位数字量的变化，所引起的模拟量输出的变化，通常定义为输出满刻度值与2n之比（n为D/A转换器的二进制位数），习惯上用输入数字量的位数表示。显然，二进制位数越多，分辨率越高

（2）建立时间

建立时间是描述D/A转换器转换速度的参数，表明转换时间长短。其值为从输入数字量到输出达到终值误差± (1/2)LSB（最低有效位）时所需的时间。

（3）转换精度

理想情况下，转换精度与分辨率基本一致，位数越多精度越高。但由于电源电压、基准电压、电阻、制造工艺等各种因素存在误差。严格地讲，转换精度与分辨率并不完全一致。

11.2 单片机扩展并行8位DAC0832的设计

11.2.1 8位并行DAC 0832简介

1. DAC0832的特性

美国国家半导体公司的DAC0832芯片具有两级输入数据寄存器的8位DAC，能直接与AT89S51连接，特性如下。

（1）分辨率为8位。

（2）电流输出，建立时间为1μs。

（3）可双缓冲输入、单缓冲输入或直通输入。

（4）单一电源供电（+5V～+15V），低功耗，20mW。

2. DAC0832的引脚及逻辑结构

引脚见图11-1。

内部结构见图11-2。

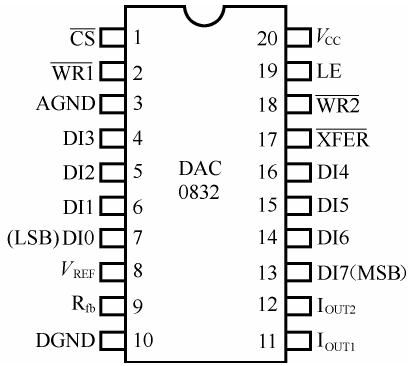

图11-1 DAC0832引脚

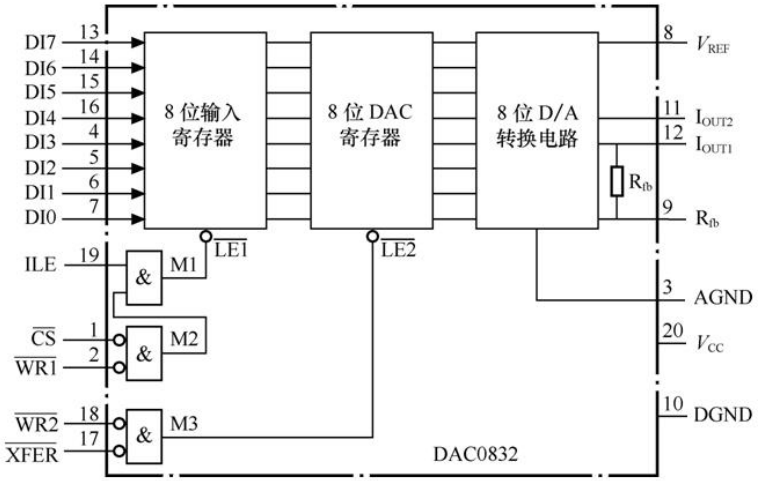

图11-2 DAC0832逻辑结构

由图11-2，片内共两级寄存器，第一级为“8位输入寄存器”，用于存放单片机送来的数字量，使得该数字量得到缓冲和锁存，由LE1*（即M1=1时）加以控制；“8位DAC寄存器”是第二级8位输入寄存器，用于存放待转换的数字量，由LE2*控制（即M3=1时），这两级8位寄存器，构成两级输入数字量缓存。“8位D/A转换电路”受“8位DAC寄存器”输出数字量控制，输出和数字量成正比的模拟电流。如要得到模拟输出电压，需外接I-V转换电路。

各引脚的功能如下。

- **DI7～DI0**：8位数字量输入端，接收发来的数字量。
- **ILE、CS*、WR1*** ：当ILE=1， CS*=0，WR1 *=0时，即M1=1，第一级8位输入寄存器被选中。待转换的数字信号被锁存到第一级8位输入寄存器中。
- **XFER *、WR2 *** ：当 XFER*=0， WR2* =0时，第一级8位输入寄存器中待转换数字进入第二级8位DAC寄存器中。
- I_{OUT1}：D/A转换电流输出1端，输入数字量全为“1”时，I_{OUT1}最大，输入数字量全为“0”时，I_{OUT1}最小。

- I_{OUT2}：D/A转换电流输出2端，$I_{OUT2} + I_{OUT1}$ = 常数。
- R_{fb}： I-V转换时的外部反馈信号输入端，内部已有反馈电阻R_{fb}，根据需要也可外接反馈电阻。
- V_{REF}：参考电压输入端。
- V_{CC}：电源输入端，在+5V～+15V范围内。
- **DGND**：数字地。
- **AGND**：模拟地，最好与基准电压共地

11.2.2案例设计：单片机扩展DAC0832的程控电压源

单片机控制DAC0832可实现数字调压，单片机只要送给DAC0832不同数字量，即可实现不同模拟电压输出。DAC0832输出可用单缓冲方式或双缓冲方式。单缓冲方式是DAC0832片内的两级数据寄存器的有一个处于直通方式，另一处于受AT89S51控制的锁存方式。

单片机控制DAC0832实现数字调压的单缓冲方式接口电路见图11-3。由于 XFER*=0、WR2*=0，所以第二级 “8位DAC寄存器” 处于直通。第一级 “8位输入寄存器”为单片机控制的锁存方式，3个锁存控制端的ILE直接接到有效的高电平，另两个控制端CS*、WR1* 分别由单片机P2.0和P2.1控制。

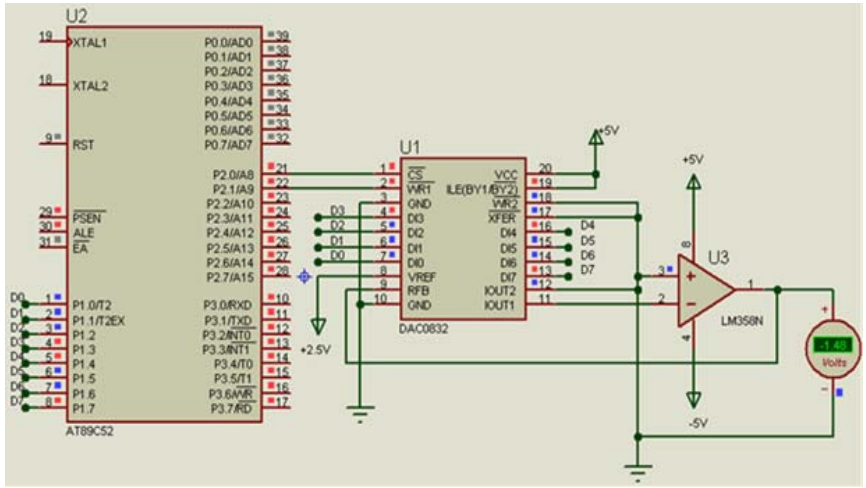

图11-3　单缓冲方式的单片机与DAC0832的接口原理电路

DAC 0832输出电压Vo与输入数字量B关系为：

$$v_0 = -B \bullet \frac{V_{REF}}{256}$$

由上式见，输出模拟电压Vo与输入数字量B以及基准电压V_{REF}成正比，且B为0时，Vo也为0， B为255时，Vo为最大的绝对值输出，且不会大于V_{REF}。下面介绍单缓冲方式下单片机扩展DAC0832程控电压源的设计。

【例11-1】单片机与DAC0832单缓冲方式接口见图11-3，单片机P2.0脚控制DAC0832的CS*脚，P2.1控制WR1*端。当P2.0脚为低时，如果同时WR*有效，单片机就会把数字量通过P1口送入DAC0832 的DI7～DI0端，并转换输出。用虚拟直流电压表测量经运放LM358N的I/V转换后的电压值，并观察输出电压变化。

仿真运行后，可看到虚拟直流电压表测量输出电压在-2.5V～0V（参考电压为2.5V）范围内不断线性变化。如参考电压为5V，则输出电压在-5V～0V范围内变化。如果虚拟直流电压表太小，看不清楚电压显示值，可用鼠标滚轮放大直流电压表。

单片机送给DAC0832不同的数字量，就可得到不同的输出电平，从而使单片机控制DAC0832成为一个程控电压源。

参考程序如下：

```
#include "reg51.h"
#define uchar unsigned char
#define uint unsigned int
#define out P1
sbit DAC_cs=P2^0;
sbit DAC_wr=P2^1;
void main(void)                    //主函数
{
    uchar temp,i=255;
    while(1)
    {
        {
            out=temp;
            DAC_cs=0;   //单片机控制CS 脚为低
            DAC_wr=0;   //单片机控制 WR1 脚为低，向 DAC 写入转换的数字量
            DAC_cs=1;
            DAC_wr=1;
            temp++;
            while(--i);     // i 先减 1，然后再使用 i 的值
            while(--i);
            while(--i);
        }
    }
```


11.2.3 案例设计2：波形发生器的制作

单片机如把不同波形数据发送给DAC0832，就可产生各种不同波形信号。下面介绍单片机控制DAC0832产生各种函数波形案例、

【例11-2】单片机控制DAC0832产生正弦波、方波、三角波、梯形波和三角波。

Proteus 的原理电路见图11-4。单片机P1.0~ P1.4接有5个按键，当按键按下时，分别对应产生正弦波、方波、三角波、梯形波和三角波。

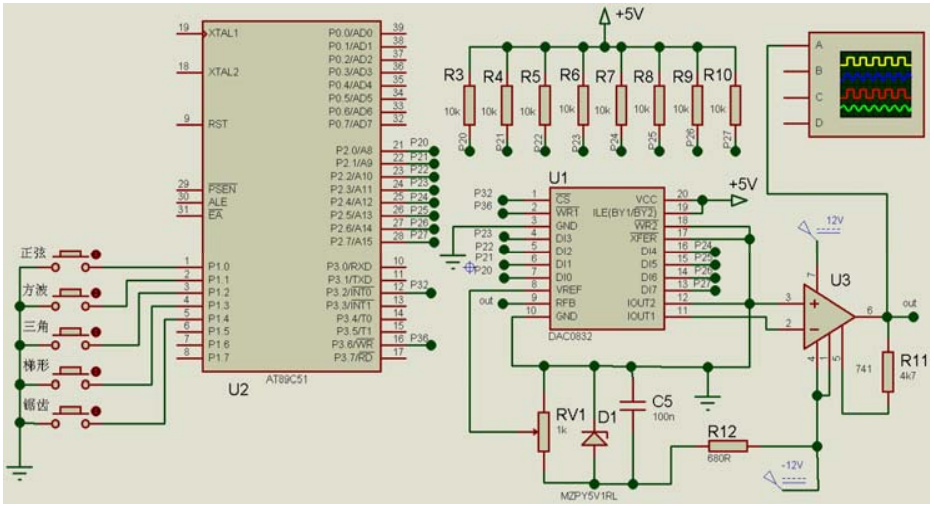

图11-4　控制DAC0832产生各种波形的原理电路

单片机控制DAC0832产生各种波形，实质就是单片机把波形的采样点数据送至DAC0832，经D/A转换后输出模拟信号。改变送出的函数波形采样点后的延时时间，就可改变函数波形的频率。产生各种波形原理如下。

（1）正弦波产生原理

单片机把正弦波的256个采样点的数据送给DAC0832。正弦波采样数据可用软件编程或MATLAB等工具计算。

（2）方波产生原理

单片机采用定时器定时中断，时间常数决定方波高、低电平持续时间。

（3）三角波产生原理

单片机把初始数字量0送DAC0832后，不断增1，增至0xff后，然后再把送给DAC0832的数字量不断减1，减至0后，再重复上述过程。

（4）锯齿波产生原理

单片机把初始数据0送DAC0832后，数据不断增1，增至0xff后，再增1则溢出清“0”，模拟输出又为0，然后再重复上述过程，如此循环，则输出锯齿波。

（5）梯形波产生原理

输入给DAC0832数字量从0开始，逐次加1。当输入数字量为0xff时，延时一段时间，形成梯形波平顶，然后波形数据再逐次减1，如此循环，则输出梯形波。

参考程序如下：

```
#include<reg51.h>
sbit wr=P3^6;
sbit rd=P3^2;
sbit key0=P1^0;              //定义P1.0脚的按键为正弦波键key0
sbit key1=P1^1;              //定义P1.1脚的按键为方波键key1
sbit key2=P1^2;              //定义P1.2脚的按键为三角波键key2
sbit key3=P1^3;              //定义P1.3脚的按键为梯形波键key3
sbit key4=P1^4;              //定义P1.3脚的按键为锯齿波键key4
unsigned char flag;        //flag为1、2、3、4、5时对应正弦波、方波、
                             //三角波、梯形波、锯齿波
```



```
unsigned char  const code  //以下为正弦波采样点数组256个数据
SIN_code[256]={0x80,0x83,0x86,0x89,0x8c,0x8f,0x92,0x95,0x98,0x9c,0x9f,0xa2,
0xa5,0xa8,0xab,0xae,0xb0,0xb3,0xb6,0xb9,0xbc,0xbf,0xc1,0xc4,0xc7,0xc9,0xcc,0x
ce,0xd1,0xd3,0xd5,0xd8,0xda,0xdc,0xde,0xe0,0xe2,0xe4,0xe6,0xe8,0xea,0xec,0xed
,0xef,0xf0,0xf2,0xf3,0xf4,0xf6,0xf7,0xf8,0xf9,0xfa,0xfb,0xfc,0xfc,0xfd,0xfe,0xfe,0
xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xfe,0xfe,0xfd,0xfc,0xfc,0xfb,0x
fa,0xf9,0xf8,0xf7,0xf6,0xf5,0xf3,0xf2,0xf0,0xef,0xed,0xec,0xea,0xe8,0xe6,0xe4,0x
e3,0xe1,0xde,0xdc,0xda,0xd8,0xd6,0xd3,0xd1,0xce,0xcc,0xc9,0xc7,0xc4,0xc1,0xbf
,0xbc,0xb9,0xb6,0xb4,0xb1,0xae,0xab,0xa8,0xa5,0xa2,0x9f,0x9c,0x99,0x96,0x92,0
x8f,0x8c,0x89,0x86,0x83,0x80,0x7d,0x79,0x76,0x73,0x70,0x6d,0x6a,0x67,0x64,0x
61,0x5e,0x5b,0x58,0x55,0x52,0x4f,0x4c,0x49,0x46,0x43,0x41,0x3e,0x3b,0x39,0x3
6,0x33,0x31,0x2e,0x2c,0x2a,0x27,0x25,0x23,0x21,0x1f,0x1d,0x1b,0x19,0x17,0x15
,0x14,0x12,0x10,0xf,0xd,0xc,0xb,0x9,0x8,0x7,0x6,0x5,0x4,0x3,0x3,0x2,0x1,0x1,0
x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x1,0x1,0x2,0x3,0x3,0x4,0x5,0x6,0x
7,0x8,0x9,0xa,0xc,0xd,0xe,0x10,0x12,0x13,0x15,0x17,0x18,0x1a,0x1c,0x1e,0x20,0
x23,0x25,0x27,0x29,0x2c,0x2e,0x30,0x33,0x35,0x38,0x3b,0x3d,0x40,0x43,0x46,0
x48,0x4b,0x4e,0x51,0x54,0x57,0x5a,0x5d,0x60,0x63,0x66,0x69,0x6c,0x6f,0x73,0x
76,0x79,0x7c}
```



```
unsigned char keyscan()                    //键盘扫描函数
{
unsigned char keyscan_num,temp;
 P1=0xff;                                  // P1口输入
  temp=P1;                                 //从P1口读入键值，存入temp中
 if(~(temp&0xff))       //判是否有键按下，即键值不为0xff，则有键按下 {
   if(key0==0)                             //产生正弦波的按键按下，
   P1.0=0
   {
          keyscan_num=1;  //得到的键值为1，表示产生正弦波
    }
  else if(key1==0)              //产生方波的按键按下，
   P1.1=0
    {
          keyscan_num=2;  //得到键值为2，表示产生方波
     }
```



```
else if(key2==0)                    //产生三角波的按键按下， P1.2=0
{
keyscan_num=3;              //得到的键值为3，表示产生三角波
}
else if(key3==0)                    //产生梯形波的按键按下，P1.3=0
{
   keyscan_num=4;                   //得到的键值为4，表示产生梯形波
}
else if(key4==0)                    //产生锯齿波的按键按下， P1.3=0
{
  keyscan_num=5;                    //得到的键值为5，表示产生锯齿波
}
else
{  keyscan_num=0;                          //没有按键按下，键值为0}
 return keyscan_num;                       //得到的键值返回
}
```

```
void init_DA0832()          //DAC0832 初始化函数
{
    rd=0;
    wr=0;
}
void SIN( )                 //正弦波函数
{
    unsigned int i;
    do{
        P2=SIN_code[i];     //由 P2 口输出给 DAC0832 正弦波数据
        i=i+1;              //数组数据指针增 1
        }while(i<256);      //判是否已输出完 256 个波形数据，未完继续输出数据
}
void Square()               //方波函数
{
    EA=1;                   //总中断允许
    ET0=1;                  //允许 T0 中断
    TMOD=1;                 //T0 工作在方式 1
    TH0=0xff;               //给 T0 高 8 位装入时间常数
    TL0=0x83;               //给 T0 低 8 位装入时间常数
    TR0=1;                  //启动 T0
}
```



```
void Triangle ( )           //三角波函数
{
    P2=0x00;                //三角波函数初始值为 0
    do{
        P2=P2+1;            //三角波上升沿
      }while(P2<0xff);      //判是否已经输出为 0xff
        P2=0xff;
    do{
        P2=P2-1;            //三角波下降沿
     }while(P2>0x00);       //判是否已经输出为 0
        P2=0x00;
}
void  Sawtooth ( )          //锯齿波函数
{
    P2=0x00;
    do{
        P2=P2+1;            //产生锯齿波的上升沿
    }while(P2<0xff);        //判上升沿是否已经结束
}
```



```
void Trapezoidal ( )        //梯形波函数
{
    unsigned char i;
    P2=0x00;
    do{
        P2=P2+1;            //产生梯形波的上升沿
    }while(P2<0xff);
    P2=0xff;                //产生梯形波的平顶
    for(i=255;i>0;i--)      //梯形波的平顶延时
    {
        P2=0xff;            //产生梯形波的下降沿
    }
    do{
        P2=P2-1;            //产生梯形波的下降沿
      }while(P2>0x00);      //判梯形波的下降沿是否结束
  P2=0x00;
}
```



```
void main()                                  // 主函数
{
 init_DA0832();              // DA0832的初始化函数
 do
{
 flag=keyscan();             //将键盘扫描函数得到的键值赋给flag
 }while(!flag);
 while(1)
 {
  switch(flag)
      {
        case 1:
        do{
          f lag=keyscan();
           SIN( );
         }while(flag==1);
         break;
```

```
case 2:
 Square ();
 do{
        flag=keyscan();
 }while(flag==2);
TR0=0;
 break;
 case 3:
  do{
        flag=keyscan();
        Triangle ();
        }while(flag==3);
  break;
```



```
case 4:
     do{
        flag=keyscan();
        Trapezoidal ();
        }while(flag==4);
     break;
case 5:
     do{
        flag=keyscan();
        Sawtooth ();
         }while(flag==5);
    break;
default:
 flag=keyscan();
  break;
 }
}
```



```
void timer0(void) interrupt 1                    //定时器T0的中断函数
{
 P2=~P2;                                         //方波的输出电平求反
 TH0=0xff;                                       //重装定时时间常数
 TL0=0x83;
 TR0=1;                                          //启动定时器T0
}
```

本案例在仿真运行时，可看到弹出的虚拟示波器，从虚拟示波器屏幕上可观察到由按键选择的函数波形输出。

如在仿真时关闭该虚拟示波器后，再启用虚拟示波器观察波形，可点击鼠标右键，现下拉菜单，点击“oscilloscope”后，仿真界面又会出现虚拟示波器屏幕。

11.3 AT89S51扩展10位串行DAC-TLC5615

11.3.1 串行DACTLC5615简介

美国TI公司产品，串行接口DAC，电压输出型，最大输出电压是基准电压值两倍。带上电复位功能，即上电时把 DAC 寄存器复位至全零。单片机只需用3 根串行总线就可完成 10 位数据的串行输入，易于和工业标准的微处理器或单片机接口，非常适于电池供电的测试仪表、移动电话，也适用于数字失调与增益调整以及工业控制场合。

TLC5615的引脚见图11-5。

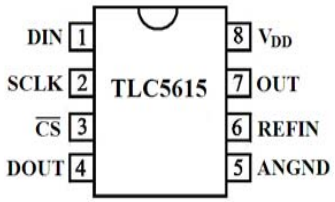

图11-5 TLC5615引脚

8只引脚功能如下：

- DIN：串行数据输入端；
- SCLK：串行时钟输入端；
- CS*：片选端，低电平有效；

- DOUT：用于级联时的串行数据输出端；
- AGND：模拟地；
- REF_{IN}：基准电压输入端，2V～ (VDD - 2)；
- OUT：DAC 模拟电压输出端；
- VDD：正电源端，4.5～5.5V ,通常取 5V。

TLC5615 内部功能框图见图11-6。

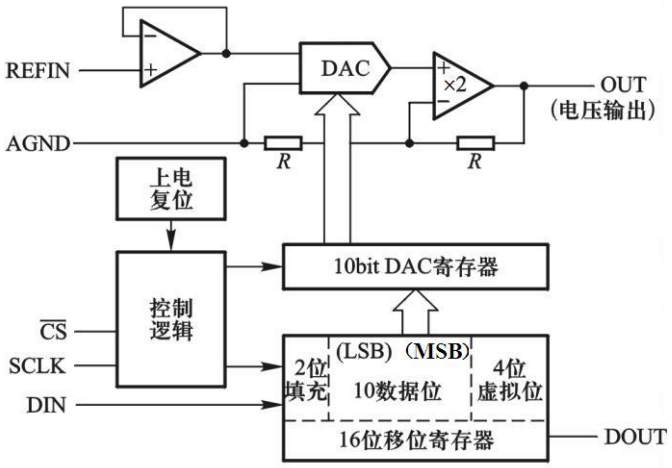

图11-6 TLC5615 内部功能框图

主要由以下几部分组成：

- 10 位 DAC 电路;
- 一个 16 位移位寄存器, 接收串行移入的二进制数，且有一个级联的数据输出端D_{OUT} ;
- 并行输入输出的 10 bit DAC 寄存器, 为 10 位 DAC 电路提供待转换的二进制数据;
- 电压跟随器为参考电压端REF_{IN}提供高输入阻抗，大约10MΩ;
- ×2 电路提供最大值为 2 倍于REF_{IN}输出;
- 上电复位电路和控制逻辑电路。

两种工作方式：

（1）第1种工作方式：12 位数据序列。从图10-8看出，16 位移位寄存器分为高4位的虚拟位、低2位的填充位以及10位有效数据位。在TLC5615 工作时，只需要向 16 位移位寄存器先后输入10位有效位和低2位的任意填充位。

（2）第2种工作方式：级联方式, 即 16 位数据列，可将本片的D_{OUT}接到下一片的 D_{IN}，需向 16 位移位寄存器先后输入高 4 位虚拟位、10 位有效位和低 2 位填充位，由于增加了高 4 位虚拟位, 所以需要 16 个时钟脉冲 。

利用带有串行接口的TLC5615 D/A转换电路，调节可变电阻器，使输出电压可在 0～5V 内调节。调整电位计，用电压表测量DAC转换输出的电压值。

当TLC5615片选脚CS*为低时, 串行输入数据才能被移入16位移位寄存器。当CS*为低时，在每一个 SCLK 时钟的上升沿将 DIN 的一位数据移入 16 位移寄存器。注意, 二进制最高有效位被导前移入。接着， CS* 的上升沿将16位移位寄存器的10位有效数据锁存于 10 位 DAC 寄存器，供DAC电路进行转换；

当片选端 CS*为高时，串行输入数据不能被移入16位移位寄存器。

11.3.2 案例设计：单片机扩展串行DACTLC5615的设计

【例11-3】单片机控制串行DAC-TLC5615进行D/A转换，原理电路及仿真见图11-7。调节可变电位计RV1的值，使TLC5615的输出电压可在0～5V 内调节，从虚拟直流电压表可观察到DAC转换输出的电压值。

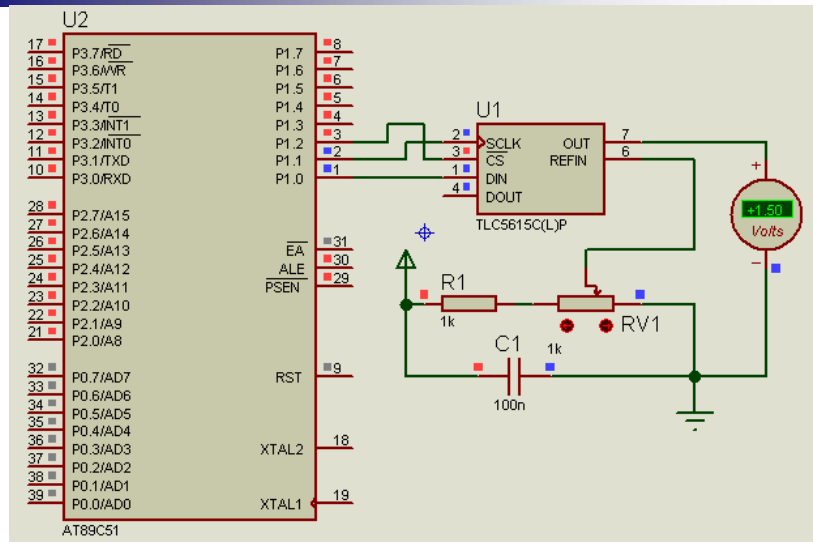

图11-7 单片机与DAC-TLC5615的接口电路

参考程序如下：

```
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
sbit   SCL=P1^1;
sbit  CS=P1^2;
sbit  DIN=P1^0;
uchar bdata dat_in_h;
uchar bdata dat_in_l;
sbit h_7 = dat_in_h^7;
sbit l_7 = dat_in_l^7;
void delayms(uint j)
{
    uchar i=250;
    for(;j>0;j--)
```



```
{
        while(--i);
        i=249;
        while(--i);
        i=250;
}
}
void Write_12Bits(void)                    //一次向TLC5615中写入12bit数据函数
{
            uchar i;
            SCL=0;                                  //置零SCL，为写bit做准备；
            CS=0;                                   //片选端 =0；
            for(i=0;i<2;i++)                        //循环2次，发送高两位；
            {
                        if(h_7)                     //高位先发；
            {
```



```
                        DIN = 1; //将数据送出；
                        SCL = 1; //提升时钟，写操作在时钟上升沿触发
            SCL = 0; //结束该位传送，为下次写作准备；
            }
                        else
            {
                        DIN= 0;
                        SCL = 1;
                        CL = 0;
            }
            dat_in_h <<= 1;
            }
```



```
for(i=0;i<8;i++)//循环八次，发送低八位；
{
            if(l_7)
            {
                        DIN = 1; //将数据送出；
                        SCL = 1;            //提升时钟，写操作在时钟上升沿触发
                        SCL = 0; //结束该位传送，为下次写作准备
            }
            else
            {           DIN= 0;
                        SCL = 1;
                        SCL = 0;}
                        dat_in_l <<= 1;}
                        for(i=0;i<2;i++)                //循环2次，发送两个填充位
```

```
{
  DIN= 0;SCL = 1;SCL = 0;
}  CS = 1;  SCL = 0;}
void TLC5615_Start(uint dat_in)                    //启动DAC转换函数
{
            dat_in %= 1024;
            dat_in_h=dat_in/256;
            dat_in_l=dat_in%256;
            dat_in_h <<= 6;
            Write_12Bits();
}
void main( )                                                    //主函数
{          while(1)
            {TLC5615_Start(0xffff);   delayms(1);}
}
```


11.3 单片机扩展ADC概述

A/D转换器（ADC）把模拟量转换成数字量，单片机才能进行数据处理。随着超大规模集成电路技术的飞速发展，大量结构不同、性能各异的A/D转换芯片应运而生。

1．A/D转换器简介

目前单片ADC芯片较多，对设计者来说，只需合理的选择芯片即可。现在部分的单片机片内也集成了A/D转换器，位数为8位、10位或12位，且转换速度也很快，但是在片内A/D转换器不能满足需要的情况下，还是需要外扩。因此，作为外部扩展A/D转换器的基本方法，读者还是应当掌握。

尽管A/D转换器种类很多，但目前广泛应用在单片机应用系统中的主要有逐次比较型转换器和双积分型转换器，此外Σ-Δ式转换器也逐渐得到重视和应用。

逐次比较型ADC，在精度、速度和价格都适中，是最常用的A/D转换器。双积分型ADC，具有精度高、抗干扰性好、价格低廉等优点，与逐次比较型A/D转换器相比，转换速度较慢，近年来在单片机应用领域中已得到广泛应用。

Σ-Δ式ADC具有积分式与逐次比较型ADC的优点。它对工业现场串模干扰具有较强抑制能力，不亚于双积分ADC，它比双积分ADC有较高转换速度。与逐次比较型ADC相比，有较高信噪比，分辨率高，线性度好。由于上述优点，Σ-Δ式ADC得到了重视，已有多种Σ-Δ式A/D芯片可供用户选用。

A/D转换器按照输出数字量的有效位数分为4位、8位、10位、12位、14位、16位并行输出以及BCD码输出的位、位、位等多种。目前，除了并行的A/D转换器外，带有同步SPI串行接口的A/D转换器的使用也逐渐增多。串行接口的A/D转换器具有占用单片机的端口线少、使用方便、接口简单等优点，已经得到广泛的使用。较为典型的串行A/D转换器为美国TI公司的TLC549（8位）、TLC1549（10位）以及TLC1543（10位）和TLC2543（12位）等。

A/D转换器按照转换速度可大致分为超高速（转换时间≤1ns）、高速（转换时间≤1μs）、中速（转换时间≤1ms）、低速（转换时间≤1s）等几种不同转换速度的芯片。目前许多新型的A/D转换器已将多路转换开关、时钟电路、基准电压源、二/十进制译码器和转换电路集成在一个芯片内，为用户提供了极大方便。

2．A/D转换器主要技术指标

（1）转换时间或转换速率。转换时间是指A/D转换器完成一次转换所需要的时间。转换时间的倒数为转换速率。

（2）分辨率。分辨率是衡量A/D转换器能够分辨出输入模拟量最小变化程度的技术指标。分辨率取决于A/D转换器的位数，习惯上用输出的二进制位数或BCD码位数表示。例如，A/D转换器AD1674的满量程输入电压为5V，可输出12位二进制数，即用212个数进行量化，其分辨率为1LSB，也即5V/212=1.22mV，其分辨率为12位，或能分辨出输入电压1.22mV的变化。

（3）转换精度。定义为一个实际A/D转换器与一理想A/D转换器在量化值上的差值，可用绝对误差或相对误差表示。

11.4 单片机并行扩展8位A/D转换器ADC0809

1. ADC0809引脚及功能

逐次比较型8路模拟输入、8位数字量输出的A/D转换器，引脚见图11-8。

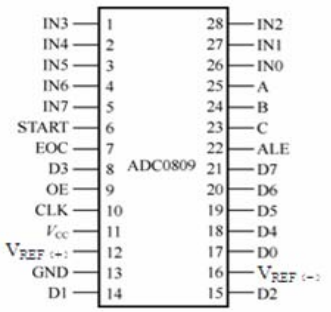

图11-8 ADC0809引脚

共28个引脚，双列直插式封装，引脚功能如下：

- IN0～IN7：8路模拟信号输入。
- D0～D7：转换完毕的8位数字量输出端。
- C、B、A与ALE：C、B、A端控制8路模拟输入通道的切换，分别与单片机的3条地址线相连。C、B、A＝000～111对应IN0～IN7通道地址。各路模拟输入通道之间切换由改变加到C、B、A上的编码来实现。ALE为ADC0809接收C、B、A编码时的锁存控制信号。
- OE、START、CLK：OE为转换结果输出允许端；START为启动信号输入端；CLK为时钟信号输入端，ADC0809的CLK信号须外加。
- EOC：转换结束输出信号。当A/D转换开始时，该引脚为低电平，当A/D转换结束时，该引脚为高电平。
- VR（+）、VR（-）：基准电压输入端。

2．ADC0809的内部结构

结构见图11-9。ADC0809采用逐次比较方法完成A/D转换，由单一+5V电源供电。片内带有锁存功能的8路选1模拟开关，由加到C、B、A引脚编码决定所选通道。

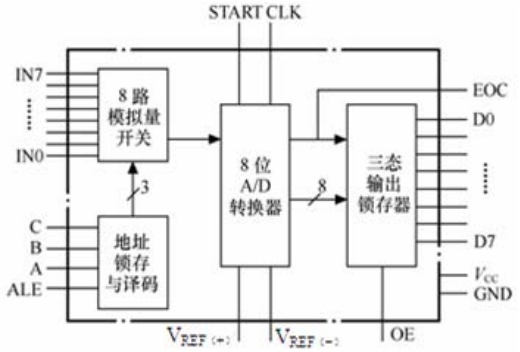

图11-9 ADC0809结构框图

ADC0809完成一次转换需100μs（此时加在CLK引脚的时钟频率为640MHz，即转换时间与加在CLK引脚的时钟频率有关），它具有输出TTL三态锁存缓冲器，可直接连到AT89S51单片机的数据总线上。通过适当的外接电路，ADC0809可对0～5V的模拟信号进行转换。

3．输入模拟电压与输出数字量的关系

ADC0809输入模拟电压与转换输出结果数字量关系如下：

$$V_{IN}=\frac{[V_{REF(+)}-V_{REF(-)}]}{256}\bullet N+V_{REF(-)}$$

其中：V_{IN}处于（VREF(+)-VREF(-)）之间，N为十进制数。通常情况下VREF(+)接+5V，VREF(-)接地，即模拟输入电压范围为0~+5V，对应的数字量输出为0x00~0xff。

4． ADC0809的转换工作原理

讨论接口设计前，先了解单片机如何控制ADC开始转换，如何得知转换结束以及如何读入转换结果的问题。

单片机控制ADC0809进行A/D转换过程如下：首先由加到C、B、A上的编码决定选择ADC0809的某一路模拟输入通道，同时产生高电平加到ADC0809的START引脚，开始对选中通道转换。

当转换结束时，ADC0809发出转换结束EOC（高电平）信号。当单片机读取转换结果时，需控制OE端为高电平，把转换完毕的数字量读入到单片机内。

单片机读取A/D转换结果可采用查询方式和中断方式。

查询方式是检测EOC脚是否变为高电平，如为高电平则说明转换结束，然后单片机读入转换结果。

中断方式是单片机启动ADC转换后，单片机执行其他程序。ADC0809转换结束后EOC变为高电平，EOC通过反相器向单片机发出中断请求，单片机响应中断，进入中断服务程序，在中断服务程序中读入转换完毕的数字量。很明显，中断方式效率高，特适合于转换时间较长的ADC。

11.4.1 案例：单片机控制ADC0809进行A/D转换

【例11-4】采用查询方式控制ADC0809（Proteus元件库中没有ADC0809，可用库中ADC0808替代，与ADC0809性能完全相同，用法一样，只是在非调整误差方面有所不同，ADC0808为±1/2LSB，而ADC0809为±1LSB）进行A/D转换，原理电路见图10-12。输入给ADC0809模拟电压可通过调节电位器RV1来实现，ADC0808将输入的模拟电压转换成二进制数字，并通过P1口输出，控制发光二极管亮与灭，来显示转换结果的二进制数字量。

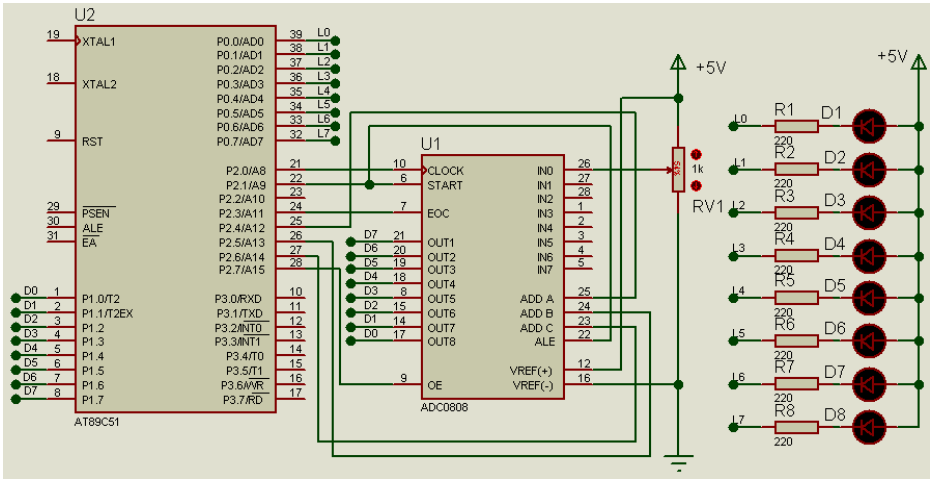

图11-10　单片机控制ADC0809进行转换

ADC0808转换一次约需100μs，采用查询方式，即使用P2.3来查询EOC脚电平，判断A/D转换是否结束。如果EOC脚为高，说明转换结束，单片机从P1口读入转换二进制的结果，然后把结果从P0口输出给8个发光二极管，发光二极管被点亮的位，对应转换结果“0”。

```
#include "reg51.h"
#define uchar unsigned char
#define uint unsigned int
#define LED  P0
#define out  P1
sbit start=P2^1;
sbit OE=P2^7;
sbit EOC=P2^3;
```



```
sbit CLOCK=P2^0;
sbit add_a=P2^4;
sbit add_b=P2^5;
sbit add_c=P2^6;

void main(void)
{
uchar  temp;
add_a=0;add_b=0;add_c=0;                    //选择ADC0809的通道0
while(1)
        {
                start=0;
                start=1;
                start=0;                    //启动转换
while(1)
{
clock=!clock;if(EOC==1)break;}                      //等待转换结束
```



```
OE=1;                               //允许输出
temp=out;                           //暂存转换结果
OE=0;                               //关闭输出
LED=temp;                           //采样结果通过P0口输出到LED
  }
 }
}
```

A/D转换时须加基准电压，单独用高精度稳压电源供给，其电压变化要小于1LSB，这是保证转换精度的基本条件。否则当被转换的输入电压不变，而基准电压的变化大于1LSB，也会引起A/D转换器输出的数字量变化。如用中断方式读取结果。可将EOC引脚与单片机P2.3脚断开，EOC引脚接反相器（例如74LS04）的输入，反相器输出接单片机外部中断请求输入端（INTx 脚），转换结束时，向单片机发出中断请求信号。可将本例接口电路及程序修改，采用中断方式来读取A/D转换结果。

11.4.2 案例：2路输入的数字电压表的设计

【例11-5】设计一个单片机采用查询方式对2路模拟电压（0～5V）交替采集的数字电压表。原理电路与仿真见图11-11。

2路0～5V被测电压加到ADC0809IN0和IN1通道，进行A/D转换，两路输入电压的大小可通过手动调节RV1和RV2来实现。

本例将1.25V和2.5V作为两路输入的报警值，当通道IN0和IN1的电压分别超过1.25V和2.50V时，对应二进制数值分别为0x40和0x80。当A/D转换结果超过该数值时，将驱动发光二极管D2闪烁与蜂鸣器发声，以表示超限。测得的输入电压交替显示在LED数码管上，同时也显示在两个虚拟电压表图标上，通过鼠标滚轮来放大虚拟电压表图标，可清楚地看到输入电压测量结果。

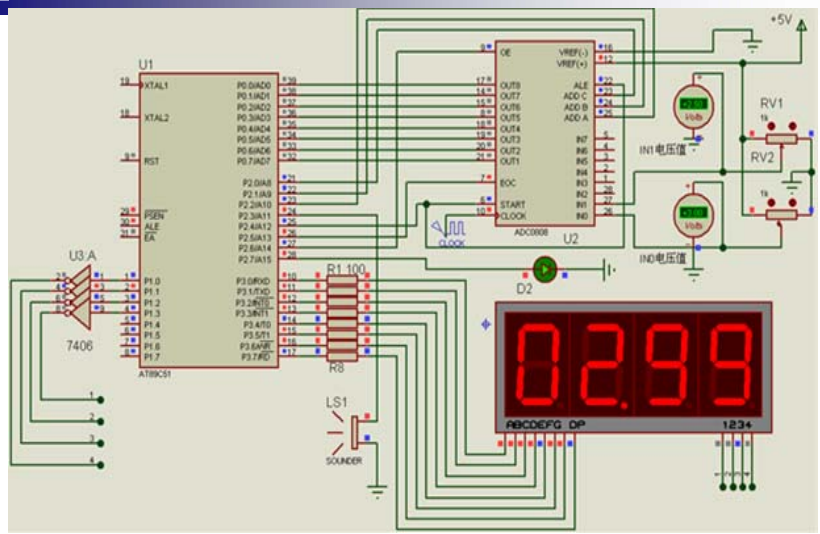

图11-11 查询方式的数字电压表电路原理图与仿真

ADC0809采用的基准电压为+5V。由10.3.2小节，转换所得结果二进制数字addata所代表的的电压的绝对值为(addata÷256)×5V，而若将其显示到小数点后两位，不考虑小数点存在（将其乘以100），其计算数值为：

(addata×100÷256)×5V≈addata×1.96 V

控制小数点显示在左边第二位数码管上，即为实际测量电压。

参考程序如下：

```
#include<reg51.h>
unsigned char
a[16]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c,0x39
,0x5e,0x79,0x71,},b[4],c=0x01;
sbit START=P2^4;
sbit OE=P2^6;
sbit EOC=P2^5;
sbit add_a=P2^2;
sbit add_b=P2^1;
sbit add_c=P2^0;
sbit led=P2^7;
sbit buzzer=P2^3;
void Delay1ms(unsigned int count)//延时函数
{  unsigned int i,j;
  for(i=0;i<count;i++)
    for(j=0;j<120;j++);
}
```

```
void show()                                  //显示函数
{
 unsigned int r;
 for(r=0;r<4;r++)
 {
P1=(c<<r);
P3=b[r];
if(r==2)                                     //显示小数点
P3=P3|0x80;
Delay1ms(1);
 }
}
void main(void)
{
 unsigned int addata=0,i;
 while(1)
```



```
{
 add_a=0; add_b=0;add_c=0;//采集第一路信号
START=1;                                     //根据时序启动ADC0808
START=0;
while(EOC==0)
{
 OE=1;
}
 addata=P0;
      if(addata>=0x40) //大于1.25V时，则使用LED和蜂鸣器报警
      {
            for(i=0;i<=100;i++)
            {
                  led=~led;
                  buzzer=~buzzer;
            }
```



```
led=1;          //控制发光二极管VD2闪烁，发出光报警信号
buzzer=1;             //控制蜂鸣器发声，发出声音报警信号
}
else                                         //否则取消报警
{
led=0;                                 //控制发光二极管VD2灭
buzzer=0;                              //控制蜂鸣器不发声
}
      addata=addata*1.96; //将采得的二进制数转换成可读的电压值
        OE=0;
        b[0]=a[addata%10];                        //显示到数码管上
        b[1]=a[addata/10%10];
        b[2]=a[addata/100%10];
        b[3]=a[addata/1000];
```



```
for(i=0;i<=200;i++)
      {
show();
      }

      add_a=1;                                    //采集第二路信号
      add_b=0;
      add_c=0;
START=1;                                    // 启动ADC0808开始转换
START=0;
while(EOC==0)
{
 OE=1;
}
addata=P0;
```

```
if(addata>=0x80)          //当大于2.5V时，则使用LED和蜂鸣器报警
{
      for(i=0;i<=100;i++)
        {
            led=~led;
            buzzer=~buzzer;
        }
        led=1;
        buzzer=1;
}
else                      //否则取消报警
{
        led=0;
       buzzer=0;
}
addata=addata*1.96;       //将采得的二进制数转换成可读的电压
```



```
                OE=0;
                b[0]=a[addata%10];          //显示到数码管上
               b[1]=a[addata/10%10];
               b[2]=a[addata/100%10];
               b[3]=a[addata/1000];
               for(i=0;i<=200;i++)
               {
                 show();
               }
        }
}
```


11.5 单片机扩展串行8位A/D转换器TLC549

串行A/D转换器与单片机连接具有占用I/O口线少的优点，使用较多，大有取代并行A/D转换器的趋势。下面介绍单片机扩展串行8位 A/D转换器TLC549的应用设计。

11.5.1 TLC549的特性及工作原理

TLC549是美国TI公司推出的价廉、高性能的带有SPI接口的8位 A/D转换器，其转换速度小于17μs，最大转换速率为 40kHz，内部系统时钟的典型值为4MHz，电源为3~6V。它能方便与各种单片机通过SPI串行接口连接。

1. TLC549的引脚及功能

引脚见图11-12。

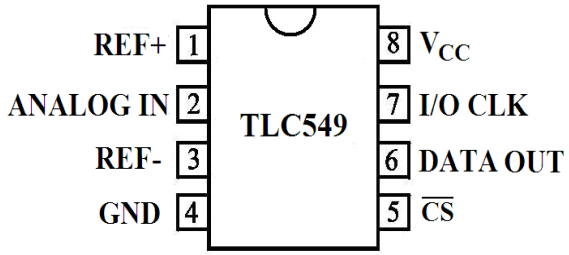

图11-12 TLC549的引脚

引脚功能如下：

- CS*：片选端。
- DATAOUT：转换结果数据TTL电平串行输出端，输出时高位在前。
- ANALOGIN：模拟信号输入端，0≤ANALOGIN≤Vcc，当

  ANALOGIN≥REF+电压时，转换结果为全“1”(0xff)，

  ANALOGIN≤REF- 电压时，转换结果为全“0”(0x00)。
- I/O CLOCK：外接输入/输出时钟输入端，同于同步芯片的输入输出操作，无需与芯片内部系统时钟同步。
- REF+：正基准电压输入 2.5V≤REF+≤Vcc+0.1V。
- REF-：负基准电压输入端，-0.1V≤REF-≤2.5V。且：（REF+）－（REF-）≥1V。

- V_{CC}：电源3V≤Vcc≤6V。
- GND：地。

2. TLC549 工作时序

工作时序见图11-13。从图可知：

（1）串行数据中高位A7先输出，最后输出低位A0；

（2）在每一次I/O COLCK高电平期间DATA OUT 线上数据产生有效输出，每出现一次I/O COLCK，DATA OUT线就输出1位数据。一个周期出现8次I/O COLCK 信号并对应8位数据输出；

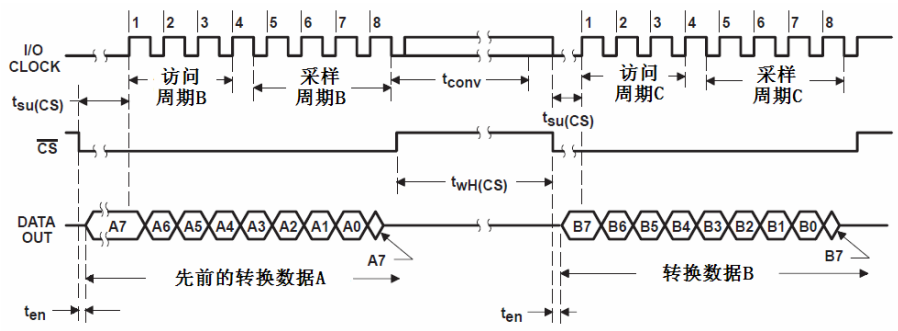

图11-13　TLC549的工作时序

（3）在CS*变为低电平后，最高有效位（A7）自动置于DATA OUT 总线 。其余7位（A6~A0）在前7 个I/O CLOCK 下降沿由时钟同步输出。B7～B0 以同样方式跟在其后；

（4）tsu在片选信号CS*变低后，I/O COLCK 开始正跳变的最小时间间隔1.4μs；

（5）ten是从CS*变低到DATA OUT 线上输出数据最小时间（1.2μs）；

（6）只要I/O COLCK 变高就可读取DATA OUT 线上数据；

（7）只有在CS*端为低时，TLC549 才工作；

（8）TLC549的A/D转换电路没有启动控制端，只要读取前一次数据后马上就可以开始新的A/D 转换。转换完成后就进入保持状态。TLC549每次转换所需时间是17μs， 它开始于 变为低电平后I/OCLOCK 的第8个下降沿，没有转换完成标志信号。

当CS*变为低电平后，TLC549芯片被选中， 同时前次转换结果的最高有效位MSB （A7）自 DATA OUT 端输出，接着要求从I/O CLOCK端输入8个外部时钟信号，前7个 I/O CLOCK信号的作用，是配合TLC549 输出前次转换结果的的A6~A0 位，并为本次转换做准备：

在第4个 I/O CLOCK 信号由高至低的跳变之后，片内采样/保持电路对输入模拟量采样开始，第8个 I/O CLOCK 信号的下降沿使片内采样/保持电路进入保持状态并启动 A/D开始转换。转换时间为 36 个时钟周期，最大为 17μs。

直到 A/D转换完成前的这段时间内，TLC549 的控制逻辑要求：或者CS*保持高电平，或者 I/O CLOCK 时钟端保持36个系统时钟周期低电平。

由此可见，在TLC549的 I/O CLOCK 端输入8个外部时钟信号期间需要完成以下工作：读入前次A/D转换结果；对本次转换的输入模拟信号采样并保持；启动本次转换开始。

11.5.2 案例设计：单片机扩展TLC549的设计

【例11-6】单片机控制串行8位A/D转换器TLC549进行A/D转换，原理电路与仿真结果见图11-14。由电位计RV1提供给TLC549模拟量输入，通过调节RV1上的“+”、“–”端，改变输入电压值。编写程序将模拟电压量转换成二进制数字量，本例用P0口输出控制8个发光二极管的亮与灭显示转换结果的二进制码，也可通过LED数码管将转换完毕的数字量以16进制数形式显示出来。

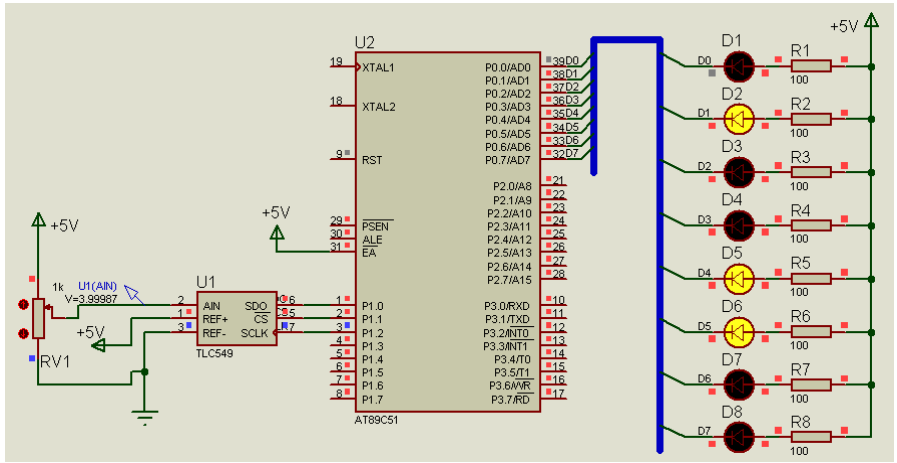

图11-14 单片机与TLC549接口的原理电路

参考程序如下：

```
#include<reg51.h>
#include<intrins.h>                    //包含_nop_()函数头文件
#define uchar unsigned char
#define uint unsigned int
#define  led  P0
sbit sdo=P1^0;      //定义P1.0与TLC549的SDO脚（即5脚DATA OUT）连接
sbit cs=P1^1;       //定义P1.1与TLC549的 脚连接
sbit sclk=P1^2;   //定义P1.2与TLC549的SCLK脚（即7脚I/O CLOCK）连接
void delayms(uint j)            //延时函数
{
        uchar i=250;
        for(;j>0;j--)
        {
while(--i);
        i=249;
        while(--i);
        i=250;
    }
}
void delay18us(void)                    延时约18μs函数
{
_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_
   ();
_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();
   nop_();_nop_();
}
uchar convert(void)
{
   uchar i,temp;
   cs=0;
   delay18us();
   for(i=0;i<8;i++)
      {
          if(sdo==1)temp=temp|0x01;
          if(i<7)temp=temp<<1;
          sclk=1;
          _nop_(); _nop_(); _nop_();_nop_();
          sclk=0;
          _nop_(); _nop_();
      }
      cs=1;
      return(temp);
   }
   void main()
   {
      uchar result;
      led=0;
      cs=1;
          sclk=0;
          sdo=1;
          while(1)
          {
                  result=convert();
                  led=result;             //转换结果从P0口输出驱动LED
                  delayms(1000);
          }
}
```

由于TLC549 转换时间应大于17μs，本例采用延时操作方案，延时时间大约18μs，每次读取转换数据时间大于17μs即可。

11.6 AT89S51扩展12位串行ADC-TLC2543的设计

串行A/D转换器与单片机连接具有占用I/O口线少优点，使用逐渐增多，随着价格降低，大有取代并行A/D转换器趋势。下面首先介绍串行A/D转换器TLC2543基本特性及工作原理。

10.4.1 TLC2543的特性及工作原理

美国TI的12位串行SPI接口的A/D转换器，转换时间为10μs。片内有1个14路模拟开关，用来选择11路模拟输入以及3路内部测试电压中的1路进行采样。为了保证测量结果的准确性，该器件具有3路内置自测试方式，可分别测试“REF+”高基准电压值，“REF-”低基准电压值和“REF+/2”值，

- DATA OUT：A/D转换结果的三态串行输出端。 为高时处于高阻抗状态，为低时处于转换结果输出状态。
- EOC：转换结束端。
- I/O CLOCK：I/O时钟端。
- REF+：正基准电压端。基准电压的正端（通常为Vcc）被加到REF+，最大的输入电压范围为加在本引脚与REF-引脚的电压差。
- REF-：负基准电压端。基准电压低端（通常为地）加此端。
- Vcc：电源。
- GND：地。

该器件的模拟量输入范围为REF+~REF-，一般模拟量的变化范围为0~+5V，所以此时REF+脚接+5V，REF-脚接地。

由于TLC2543与单片机接口简单，且价格适中，分辨率较高，因此在智能仪器仪表中有着较为广泛应用。

1. TLC2543的引脚

TLC2543的引脚见图11-15，各引脚功能如下。

- AIN0～AIN10：11路模拟量输入端。
- CS* ：片选端。
- DATAINPUT：串行数据输入端。由4位的串行地址输入来选择模拟量输入通道。

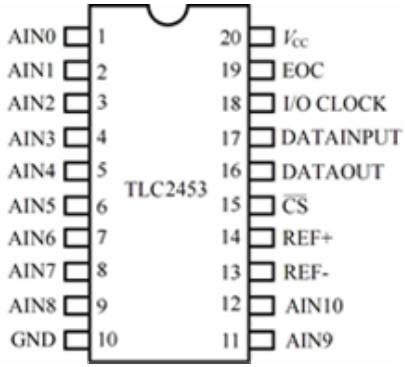

图11-15 TLC2543引脚

2. TLC2543工作过程

工作过程分为两个周期：I/O周期和实际转换周期。

（1）I/O周期

I/O周期由外部提供的I/O CLOCK定义，延续8、12或16个时钟周期，取决于选定的输出数据长度。器件进入I/O周期后同时进行两种操作。

1）TLC2543的工作时序如图10-16所示。在I/OCLOCK的前8个脉冲的上升沿，以MSB前导方式从DATAINPUT端输入8位数据到输入寄存器。其中前4位为模拟通道地址，控制14通道模拟多路器从11个模拟输入和3个内部自测电压中，

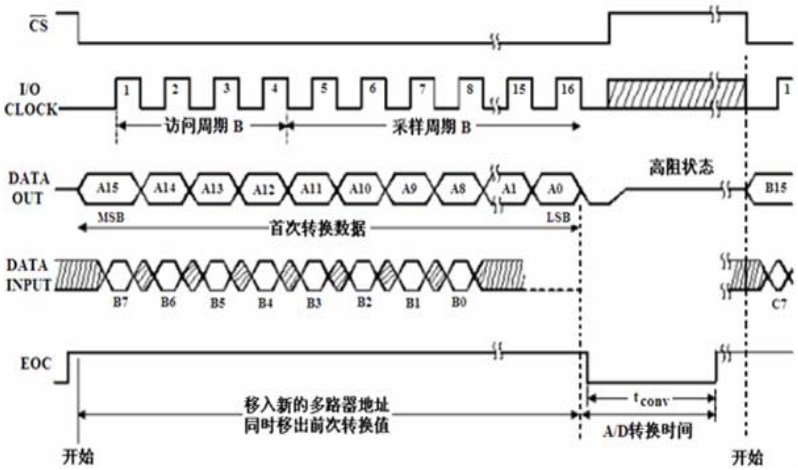

图11-16　TLC2543的工作时序

选通1路到采样保持器，该电路从第4个I/OCLOCK脉冲下降沿开始，对所选的信号进行采样，直到最后一个I/O CLOCK脉冲下降沿。I/O脉冲时钟个数与输出数据长度（位数）有关，输出数据的长度由输入数据的D3、D2可选择为8位、12位或16位。当工作于12位或16位时，在前8个脉冲之后，DATAINPUT无效。

2）在DATA OUT端串行输出8位、12位或16位数据。当 保持为低时，第1个数据出现在EOC的上升沿，若转换由 控制，则第1个输出数据发生在 的下降沿。这个数据是前1次转换的结果，在第1个输出数据位之后的每个后续位均由后续的I/OCLOCK脉冲下降沿输出。

选通1路到采样保持器，该电路从第4个I/OCLOCK脉冲下降沿开始，对所选的信号进行采样，直到最后一个I/O CLOCK脉冲下降沿。I/O脉冲时钟个数与输出数据长度（位数）有关，输出数据的长度由输入数据的D3、D2可选择为8位、12位或16位。当工作于12位或16位时，在前8个脉冲之后，DATAINPUT无效。

2）在DATA OUT端串行输出8位、12位或16位数据。当 保持为低时，第1个数据出现在EOC的上升沿，若转换由 控制，则第1个输出数据发生在 的下降沿。这个数据是前1次转换的结果，在第1个输出数据位之后的每个后续位均由后续的I/OCLOCK脉冲下降沿输出。

（2）转换周期

在I/O周期最后一I/OCLOCK脉冲下降沿后，EOC变低，采样值保持不变，转换周期开始，片内转换器对采样值进行逐次逼近式A/D转换，其工作由与I/OCLOCK同步的内部时钟控制。转换结束后EOC变高，转换结果锁存在输出数据寄存器中，待下一I/O周期输出。I/O周期和转换周期交替进行，从而可减少外部的数字噪声对转换精度影响。

3. TLC2543命令字

每次转换都必须向TLC2543写入命令字，以便确定被转换信号来自哪个通道，转换结果用多少位输出，输出的顺序是高位在前还是低位在前，输出结果是有符号数还是无符号数。命令字写入顺序是高位在前。命令字格式如下：

通道地址选择（D7~D4）	数据的长度（D3~D2）	数据的顺序（D1）	数据的极性（D0）

（1） 通道地址选择位用来选择输入通道。0000~1010分别是11路模拟量AIN0～AIN10的地址；地址1011、1100和1101所选择的自测试电压分别是（（VREF+）-（VREF-））/2、VREF-、VREF+。1110是掉电地址，选掉电后，TLC2543处于休眠状态，此时电流小于20μA。

（2）数据长度（D3~D2）位用来选择转换的结果用多少位输出。D3D2为x 0：12位输出；D3D2为01：8位输出；D3D2为11：16位输出。

（3）数据的顺序位（D1）用来选择数据输出的顺序。D1=0，高位在前；D1=1，低位在前。

（4）数据的极性位（D0）用来选择数据的极性。D0=0，数据是无符号数；D0=1，数据是有符号数。

11.6.2 案例设计：单片机扩展TLC2543的设计

介绍单片机与TLC2543接口设计及软件编程。

【例11-7】单片机与TLC2543接口电路见图11-17，编写程序对AIN2模拟通道进行数据采集，结果在数码管上显示，输入电压调节通过调节RV1来实现。

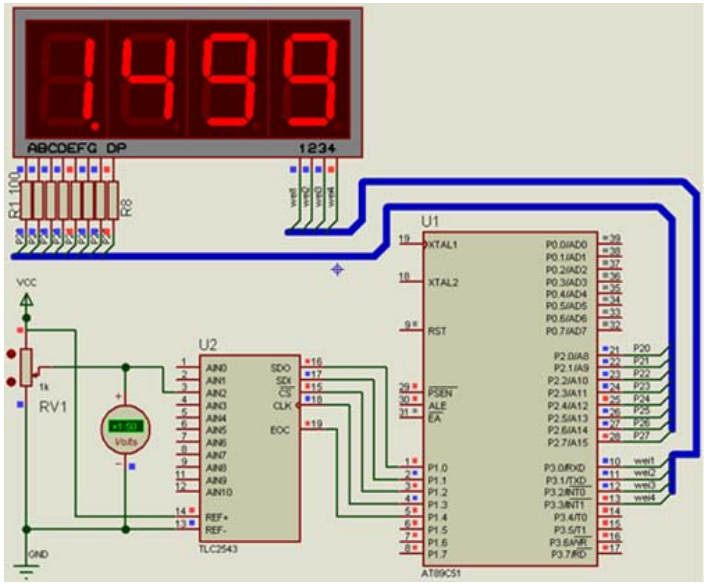

图10-17 AT89S51单片机与TLC2543的接口电路

TLC2543与单片机接口采用SPI串行接口（SPI：串行外设接口，将在第11章介绍），由于AT89S51不带SPI接口，须采用软件与I/O口线相结合，模拟SPI接口时序。TLC2543三个控制输入端分别为I/O CLOCK（18脚，输入/输出时钟）、DATAINPUT（17脚，4位串行地址输入端）以及CS*（15脚，片选），分别由单片机P1.3、P1.1和P1.2控制。转换结果（16脚）由单片机P1.0脚串行接收，AT89S51将命令字通过P1.1引脚串行写入到TLC2543的输入寄存器中。

片内14通道选择开关可选择11个模拟输入中的任一路或3个内部自测电压中的一个并且自动完成采样保持。转换结束

后“EOC”输出变高，转换结果由三态输出端“DATA OUT”输出。

采集的数据为12位无符号数，采用高位在前的输出数据。写入TLC2543的命令字为0xa0。由TLC2543的工作时序，命令字写入和转换结果输出是同时进行的，即在读出转换结果的同时也写入下一次的的命令字，采集11个数据要进行12次转换。第1次写入的命令字是有实际意义的操作，但是第1次读出的转换结果是无意义的操作，应丢弃；而第11次写入的命令字是无意义的操作，而读出的转换结果是有意义的操作

参考程序如下：

```
#include <reg51.h>
#include <intrins.h>                    //包含_nop_()函数的头文件
#define uchar unsigned char
#define unit unsigned int
unsigned char code table[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
unit ADresult[11];                      //11个通道的转换结果单元
sbit  DATOUT=P1^0;                      //定义P1.0与DATA OUT相连
sbit  DATIN=P1^1;                       //定义P1.1与DATA INPUT相连
sbit  CS=P1^2;                          //定义P1.2与 端相连
```

```
sbit  IOCLK=P1^3;                    //定义P1.3与I/O CLOCK相连
sbit  EOC=P1^4;                      //定义P1.4与EOC引脚相连
sbit wei1=P3^0;
sbit wei2=P3^1;
sbit wei3=P3^2;
sbit wei4=P3^3;

void delay_ms(unit i)
{
        int j;
        for(; i>0; i--)
                for(j=0; j<123; j++);
}
```



```
unit getdata(uchar channel)    // getdata()为获取转换结果函数，channel
                                    //为通道号
{
   uchar i,temp;
  unit read_ad_data=0;      // 分别存放采集的数据，先清0
  channel=channel<<4;       // 结果为12位数据格式，高位导前，单极性
                                  // ××××0000
  IOCLK=0;
 CS=0;                              // 下跳沿，并保持低电平
 temp=channel;                      // 输入要转换的通道
 for(i=0;i<12;i++)
 {
  if(DATOUT) read_ad_data=read_ad_data|0x01;//读入转换结果
   DATIN=(bit)(temp&0x80);                          //写入方式/通道命令字
    IOCLK=1;                                        //IOCLK上跳沿
  _nop_();_nop_();_nop_();                          //空操作延时
   IOCLK=0;                                         //IOCLK下跳沿
```



```
_nop_();_nop_();_nop_();
temp=temp<<1;              //左移1位，准备发送方式通道控制字下一位
read_ad_data<<=1;                  //转换结果左移1位
}
CS=1;                              // CS上跳沿
read_ad_data>>=1;                  // 抵消第12次左移，得到12位转换结果
return(read_ad_data);
}
void dispaly(void)             // 显示函数
{
        uchar qian,bai,shi,ge;         // 定义千、百、十、个位
        unit value;
        value=ADresult[2]*1.221;//*5000/4095
        qian=value%10000/1000;
        bai=value%1000/100;
        shi=value%100/10;
        ge=value%10;
```



```
        wei1=1;
          P2=table[qian]-128;
          delay_ms(1);
          wei1=0;
        wei2=1;
          P2=table[bai];
          delay_ms(1);
          wei2=0;
        wei3=1;
          P2=table[shi];
          delay_ms(1);
          wei3=0;
         wei4=1;
          P2=table[ge];
          delay_ms(1);
          wei4=0;
}
```

```
main(void)
{
 ADresult[2]=getdata(2);    //启动2通道转换，第1次转换结果无意义
 while(1)
 {
   _nop_(); _nop_(); _nop_();
   ADresult[2]=getdata(2); //读取本次转换结果，同时启动下次转换
  while(!EOC);                  //判是否转换完毕，未转换完则循环等待
  dispaly();
 }
}
```

由本案例见，AT89S51单片机与TLC2543接口电路十分简单，只需用软件控制4条I/O引脚，按规定时序对TLC2543进行访问即可。