D/A转换接口实验

1953729 吴浩泽

## 实验目的

了解D/A转换的基本原理；

了解D/A转换芯片AD7528BN的性能及编程方法；

了解实验平台上扩展D/A转换的基本方法。

## 实验设备

硬件：Embest EduKit-IV平台，ULINK2仿真器套件，PC机。

软件：μVision IDE for ARM集成开发环境，Windows 98/2000/NT/XP。

## 实验内容

编写D/A转换程序；

通过转换得到的模拟电压量，控制发光二极管亮度的变化；

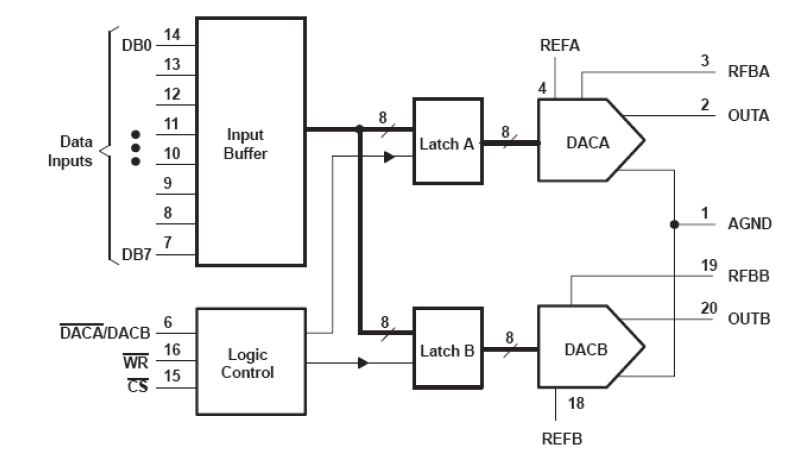
通过示波器，查看不同的波形，熟悉D/A转换的工作原理。

## 实验原理

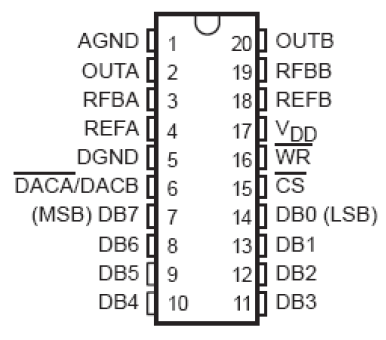
**1. 主要芯片原理功能表**

AD7528BN 是8位双通道的数字—模拟转换器，内部集成数据锁存器，其特点包括紧密的DAC至DAC的一致性。数据通过8位数据线传送至两个DAC数据锁存器，控制输入端的DACA/DACB决定哪一个数据被装载。该器件的访问与随机存储器类似，能方便地与大多数MCU相接。AD7528BN工作电源5~15V，工耗小于15mW。2或4象限乘法功能使这种器件成为许多MCU控制的增益设置和信号控制应用良好选择。他可以工作在电压方式，产生电压输出。

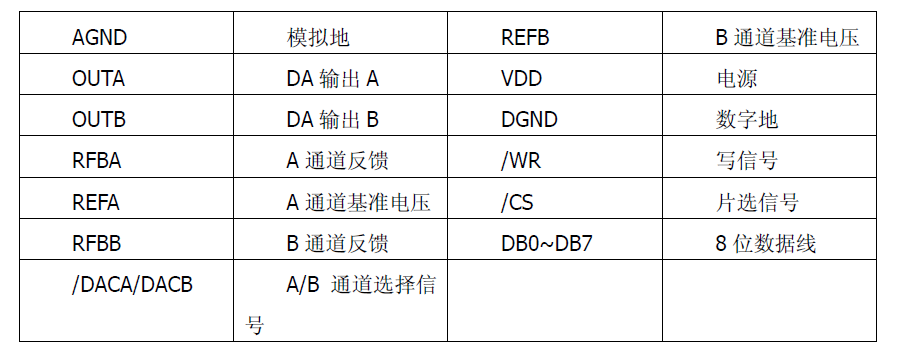
**2. 内部功能模块图**



**3. 引脚排列**



AD7528BN 引脚图



AD7528BN 引脚功能说明

AD7528BN 芯片用来根据一个输入数字量输出对应的模拟量(电压)。其接口的逻辑信息为：

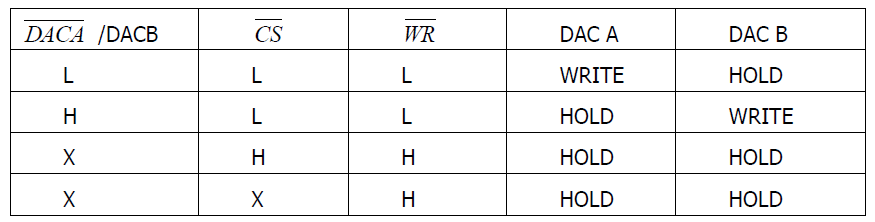
 DAC 选择：两个DAC 锁存共享一个通用8 位输入端口。通过控制输入引脚 *DACA* /DACB

来选择那个通道接受输入端口的数据。

 模式选择：输入引脚*CS* 和*WR* 用来控制所选择DAC 通道的操作模式(如表1 所示)。

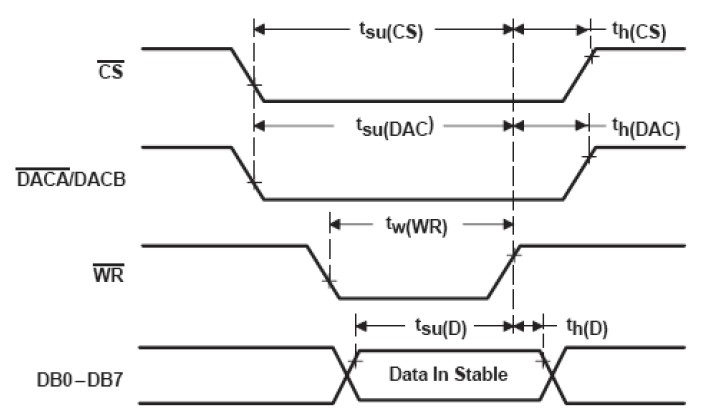
 写模式：当*CS* 和*WR* 都为低电平时，所选择的DAC 通道处于写模式。所选择的DAC 的输入锁存数据输出，输出电压大小与DB0-DB7 引脚信号相关。

 保持模式：所选择的DAC 锁存保持的数据是*CS* 或*WR* 变高之前DB0-DB7 的状态。两个模拟输出保持各自锁存中数据对应的模拟电压值。



模式选择表 L 表示低电平； H 表示高电平； X 表示高或低电平无关

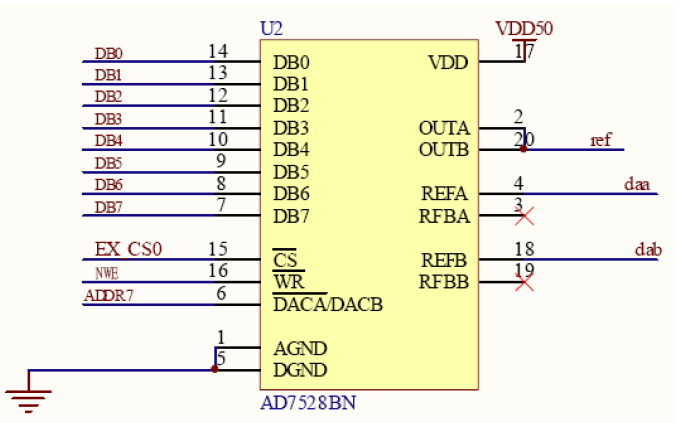
**4. 工作时序**



工作时序图

**5. 硬件连接**

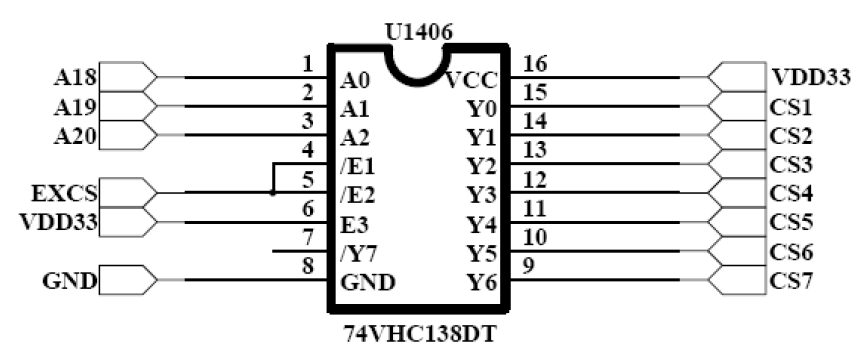
在本系统中AD7528BN 的DB0-DB7 连接到S3C2410 处理器的数据总线的D0-D7；*DACA*/DACB 连到处理器的地址总线的ADDR7（如图8-3-4）; *WR* 连到处理器的NWE 写信号；*CS* 连到一个3/8 译码器的输出输出引脚Y0，3/8 译码器的输出为地址总线的A18-A20，其中EXCS 通过CPLD连到处理器的NGCS4（如图8-3-6）。所以A 通道的地址为0x2100 0000，B 通道的地址为0x21000080。daa 和dab 连到两个LED 灯上用来表示对应通道的电压大小。



AD7528BN引脚连接图



AD7528BN与LED连接图



3/8 译码器

## 实验步骤

**1. 准备实验环境**

使用ULINK2 仿真器连接Embest EduKit-IV 实验平台的主板JTAG 接口；使用Embest EduKit-IV 实验平台附带的交叉串口线，连接实验平台主板上的COM2 和PC 机的串口（一般PC 只有一个串口， 如果有多个请自行选择，笔记本没有串口设备的可购买USB 转串口适配器扩充）；将D/A 转换功能模块插入到Extent B 插槽上，并连接好相应的跳线P1 与P2，使用Embest EduKit-IV 实验平台附带的电源适配器，连接实验平台主板上的电源接口。 **2. 串口接收设置**

在PC 机上运行windows 自带的超级终端串口通信程序，或者使用实验平台附带光盘内设置好了的超级终端，设置超级终端：波特率115200、1 位停止位、无校验位、无硬件流控制，或者使用其它串口通信程序。（注：超级终端串口的选择根据用户的PC 串口硬件不同，请自行选择，如果PC 机只有一个串口，一般是COM1）

**3. 打开实验例程**

1）拷贝实验平台附带光盘DISK3\_S3C2410\03-Codes\01-MDK\Mini2410-IV 文件夹到MDK 的安装路径：Keil\ARM\Boards\Embest\（如果本实验之前已经拷贝，可以跳过这一步）。（注：用户也可拷贝工程到任意目录，本实验为了便于教学，故统一实验路径）；

2）运行μVision IDE for ARM 软件，点击菜单栏“Project”，选择“Open Project…”，在弹出的对话框选择实验例程目录8.3\_DAC\_Test 子目录下的DAC\_Test.Uv2 工程。

3）默认打开的工程在源码编辑窗口会显示实验例程的说明文件readme.txt，详细阅读并理解实验内容。

4）工程提供了两种运行方式：一是下载到SDRAM中调试运行，二是固化到Nor Flash中运行。用户可以在工具栏Select Target下拉框中选择在RAM中调试运行还是固化Flash中运行。如下图所示：



选择运行方式

下面实验将介绍下载到SDRAM中调试运行，所以我们在Select Target下拉框中选择DAC\_Test IN RAM。

5）接下来开始编译链接工程，在菜单栏“Projiet”选择“Build target”或者“Rebuild all target files”编译整个工程，用户也可以在工具栏单击“”或者“”进行编译。

6）编译完成后，在输出窗口可以看到编译提示信息，比如“".\SDRAM\DAC\_Test.axf" - 0 Error(s), 1 Warning(s).”，如果显示“0 Error(s)”即表示编译成功。

7）拨动实验平台电源开关，给实验平台上电，单击菜单栏Debug->Start/Stop Debug Session项将编译出来的映像文件下载到SDRAM中，或者单击工具栏“”按钮来下载。

8）下载完成后，单击菜单栏Debug->Run项运行程序，或者单击工具栏“”按钮来全速运行程序。用户也可以使用进行单步调试程序。

9）全速运行后，用户可以在超级终端看到程序运行的信息，出现“Please choice the channel number:”的提示信息，根据提示选择想要的通道，接着出现提示信息“Please choice the fuction:”,根据提示信息选择想要输出的波形，可通过LED的变化或示波器查看。

10）用户可以Stop程序运行，使用μVision IDE for ARM的一些调试窗口跟踪查看程序运行的信息。

注：如果在第4）步用户选择在Flash中运行，则编译链接成功后，单击菜单栏Flash->Download项将程序固化到NorFlash中，或者单击工具栏按钮“”固化程序，从实验平台的主板拔出JTAG线，给实验平台重新上电，程序将自动运行。

**4. 观察实验结果**

在执行到第8）步时，可以看到超级终端上输出如下字符。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\* 英蓓特EduKit系列嵌入式教学系统平台 \*\*  \*\* Embest EduKit Series Embedded Teaching Platform \*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   |  | | --- | | Please choice the channel number:  1:channel 1 2:channel 2 |   选择好通道后，例如选择通道1，在键盘上输入1后，超级终端上将显示   |  | | --- | | You selected 1  Please choice the fuction:  1:Triangle 2:Sina 3:Square | | 选择好波形后，可通过LED亮度的变化或示波器查看。 | | |

## 实验参考程序

|  |  |
| --- | --- |
| |  | | --- | | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* File： dac\_test.c  \* Author: embest  \* Desc： DAC\_Test  \* History:  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*------------------------------------------------------------------------------------------\*/  /\* include files \*/  /\*------------------------------------------------------------------------------------------\*/  #include "2410lib.h"  /\*------------------------------------------------------------------------------------------\*/  /\* constant declare \*/  /\*------------------------------------------------------------------------------------------\*/  #define rCPLDIntControl (\*(volatile unsigned char\*)0x22600000)  #define rCPLDIntStatus (\*(volatile unsigned char\*)0x22200000)  /\*------------------------------------------------------------------------------------------\*/  /\* fuction declare \*/  /\*------------------------------------------------------------------------------------------\*/  void channel\_choice(void);  void \_\_irq int\_int(void);  void int\_init(void);  /\*------------------------------------------------------------------------------------------\*/  /\* global variables \*/  /\*------------------------------------------------------------------------------------------\*/  int flag\_stop; // Wave out stop flag  volatile unsigned char \* DACADDR; // The DAC Address  INT8T Key\_pressed1;  INT8T Key\_pressed2;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* name: triangle\_test  \* func: triangle wave output  \* para: none  \* ret: none  \* modify:  \* comment:  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void triangle\_test()  {  int a;  uart\_printf(" Triangle wave output!\n");  uart\_printf(" Press KEY1 or KEY2 to stop!\n");  while(!flag\_stop)  {  for( a = 255; a > 0; a--)  {  \*DACADDR = a;  delay(2);  }  for( a = 0; a < 255; a++)  {  \*DACADDR = a;  delay(2);  }  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* name: sina\_test  \* func: sina wave output  \* para: none  \* ret: none  \* modify:  \* comment:  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void sina\_test()  {  int i;  float sin[91]= {0.0, 0.0175, 0.0349, 0.0523, 0.0698, 0.0872, 0.1045, 0.1219, 0.1392,  0.1564, 0.1736, 0.1908, 0.2079, 0.2250, 0.2419, 0.2588, 0.2756, 0.2924,  0.3090, 0.3256, 0.3420, 0.3584, 0.3746, 0.3907, 0.4067, 0.4226, 0.4384,  0.4540, 0.4695, 0.4848, 0.5, 0.5150, 0.5299, 0.5446, 0.5592, 0.5736,  0.5878, 0.6018, 0.6157, 0.6293, 0.6428, 0.6561, 0.6691, 0.6820, 0.6947,  0.7071, 0.7193, 0.7314, 0.7431, 0.7547, 0.7660, 0.7771, 0.7880, 0.7986,  0.8090, 0.8192, 0.8290, 0.8387, 0.8480, 0.8572, 0.8660, 0.8746, 0.8829,  0.8910, 0.8988, 0.9063, 0.9135, 0.9205, 0.9272, 0.9336, 0.9397, 0.9455,  0.9511, 0.9563, 0.9613, 0.9659, 0.9703, 0.9744, 0.9781, 0.9816, 0.9848,  0.9877, 0.9903, 0.9925, 0.9945, 0.9962, 0.9976, 0.9986, 0.9994, 0.9998,1.0  };  uart\_printf(" Sina wave output!\n");  uart\_printf(" Press KEY1 or KEY2 to stop!\n");  while(!flag\_stop)  {  for(i = 0; i <= 90; i++)  {  \*DACADDR = sin[i] \* 127 + 127 ;  delay(2);  }  for(i = 90; i >= 0; i--)  {  \*DACADDR = sin[i] \* 127 + 127 ;  delay(2);  }  for(i = 0; i <= 90; i++)  {  \*DACADDR = 127 - sin[i] \* 127 ;  delay(2);  }  for(i = 90; i >= 0; i--)  {  \*DACADDR = 127 - sin[i] \* 127 ;  delay(2);  }  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* name: square\_test  \* func: square wave output  \* para: none  \* ret: none  \* modify:  \* comment:  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void square\_test()  {  int a = 255, b = 0;  uart\_printf(" Square wave output!\n");  uart\_printf(" Press KEY1 or KEY2 to stop!\n");  while(!flag\_stop)  {  \*DACADDR = a;  delay(500);  \*DACADDR = b;  delay(500);  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* name: int\_init  \* func: Interrupt initialize  \* para: none  \* ret: none  \* modify:  \* comment:  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void int\_init(void)  {  rSRCPND = rSRCPND; // Clear all interrupt  rINTPND = rINTPND; // Clear all interrupt  rCPLDIntControl = 0xFF;  rCPLDIntControl = 0xF9;  pISR\_EINT8\_23 = (UINT32T)int\_int;  rEINTPEND = 0xffffff;  rSRCPND = BIT\_EINT8\_23; // Clear the previous pending states  rINTPND = BIT\_EINT8\_23;  rEXTINT1 &= ~((0x7<<4)|(0x7<<0));  rEXTINT1 |= ((0x2<<4)|(0x2<<0));  rEINTMASK &= ~(3<<8);  rINTMSK &= ~(BIT\_EINT8\_23);  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* name: int\_int  \* func: EXTINT interrupt service routine  \* para: none  \* ret: none  \* modify:  \* comment:  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void \_\_irq int\_int(void)  {  unsigned char Status;  Status = rCPLDIntStatus;  Status = ~(Status & 0x6);  if(Status & 0x2)  {  rCPLDIntControl |= (1<<1);  rCPLDIntControl &= ~(1<<1);  }  else if(Status & 0x4)  {  rCPLDIntControl |= (1<<2);  rCPLDIntControl &= ~(1<<2);  }  rEINTPEND = (1 << 9);  if(flag\_stop == 0)  flag\_stop =1;  ClearPending(BIT\_EINT8\_23);  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* name: dac\_test()  \* func: out put the wave  \* para: none  \* ret: none  \* modify:  \* comment:  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void dac\_test(void)  {  int\_init();  channel\_choice();  uart\_printf(" Please choice the fuction: \n");  uart\_printf(" 1:Triangle 2:Sina 3:Square\n");  flag\_stop = 0;  while(!(rUTRSTAT1 & 0x1)); // if receive data ready on UART0  Key\_pressed2 = RdURXH1(); // or get value from UART0  if( Key\_pressed2 == ' ' ) // press SPACE to return  return ;  if(( Key\_pressed2 < '1' ) || ( Key\_pressed2 > '3' ))  {  uart\_printf(" You have pressed an invalid key! Use deafult value! \n");  Key\_pressed2 = '1';  }  uart\_printf("\n You selected %d\n", Key\_pressed2 - 0x30);  switch(Key\_pressed2)  {  case '1':  triangle\_test();  uart\_printf(" \n");  break;  case '2':  sina\_test();  uart\_printf(" \n");  break;  case '3':  square\_test();  uart\_printf(" \n");  break;  default:  break;  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* name: channel\_choice  \* func: chose the chanel  \* para: none  \* ret: none  \* modify:  \* comment:  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void channel\_choice(void)  {  uart\_printf(" Please choice the channel number:\n");  uart\_printf(" 1:channel 1 2:channel 2\n");  while(!(rUTRSTAT1 & 0x1)); // if receive data ready on UART1  Key\_pressed1 = RdURXH1(); // or get value from UART1  if( Key\_pressed1 == ' ' ) // press SPACE to return  return ;  if(( Key\_pressed1 < '1' ) || ( Key\_pressed1 > '2' ))  {  uart\_printf(" You have pressed an invalid key! Use deafult value! \n");  Key\_pressed1 = '1';  }  uart\_printf("\n You selected %d\n", Key\_pressed1 - 0x30);  switch(Key\_pressed1)  {  case '1':  DACADDR = (volatile unsigned char\*)0x21000000; // DAC chanel 1 Address  break;  case '2':  DACADDR = (volatile unsigned char\*)0x21000080; // DAC chanel 2 Address  break;  default:  break;  }  } | |

## 读书笔记

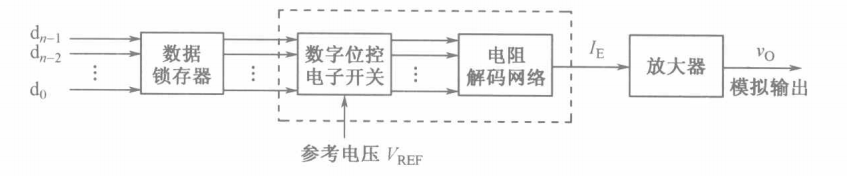
这一篇介绍D/A转换原理以及在TX-1C上的接线方式（实现方法），再用一个例子来加深理解

D/A转换原理及参数指标

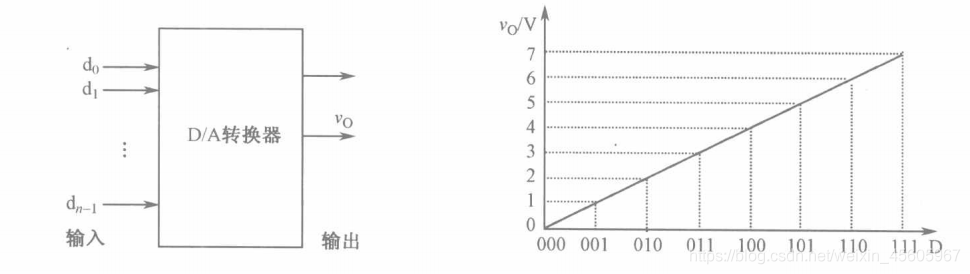
1.基本原理

数字量是二进制代码数位组合而来的，每位都有一定的权重，在D/A转换中，怎么样把这些权重以合适的方法表示出来是转换的关键

为了让数字量转换成模拟量，必须将每一位代码按其权重的大小转换为相应的模拟量，然后再把这些模拟量相加。D/A转换的示意图如下：

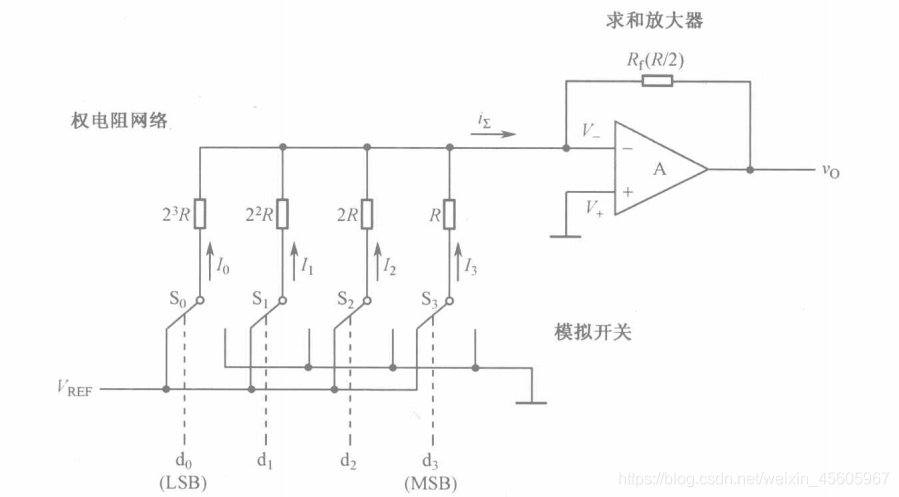


即输入相应的二进制数，得出一个模拟量的输出，而作为一个三位的转换，模拟量与数字量的对应关系可以用下图来表示。在右图中D分别对应了相应的V 0 V\_0V 0



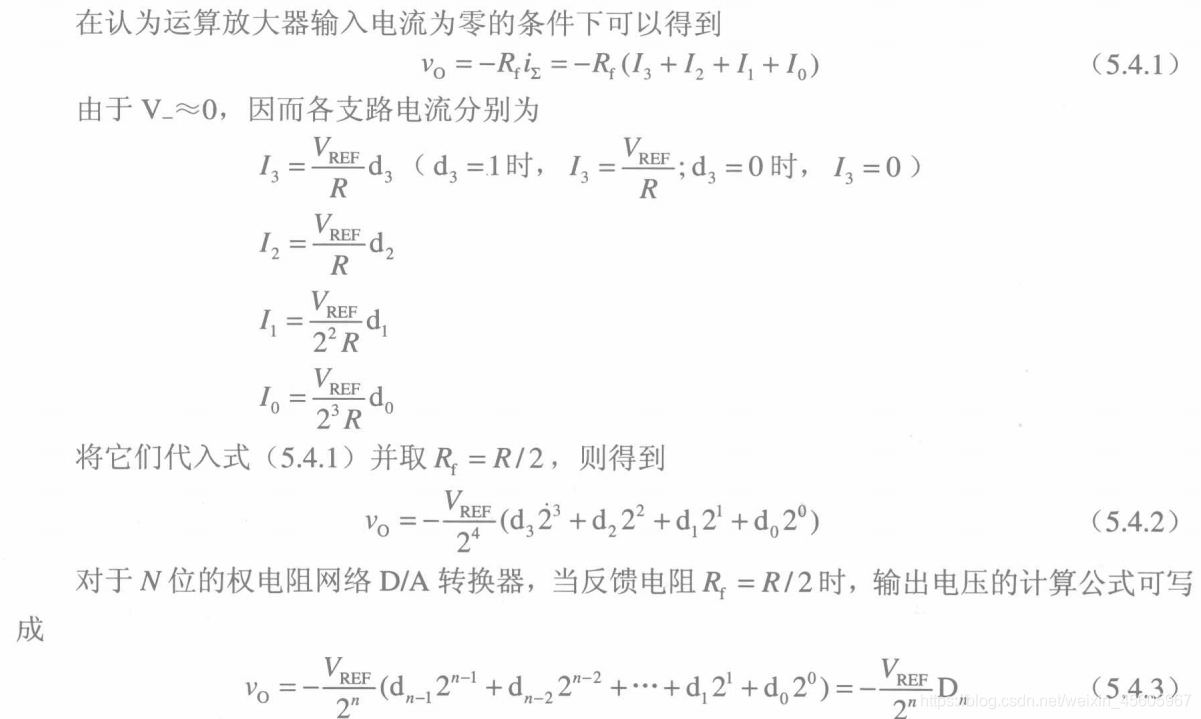
下面介绍集中D/A转换器，他们思路大致相同，常用的分别有这几类：

1.权电阻D/A转换



这其中d i d\_id i 的值可以为0或1，表示开关的打开与闭合，之后通过一个求和放大器输出相应的模拟量

因为是反相放大器，故推导过程如下：



从推导过程可发现当V R E F V\_{REF}V

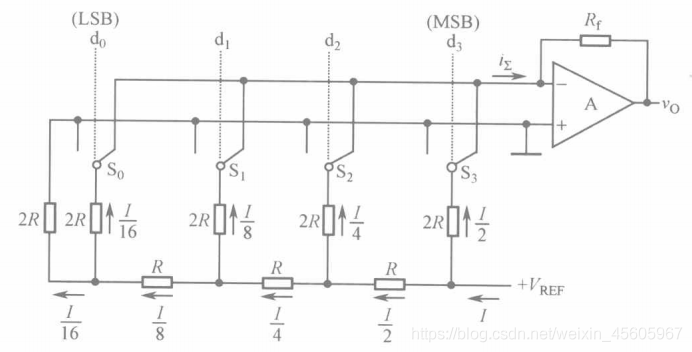
REF 为正电压时最后导出的结果为负值。故在运用中向V R E F V\_{REF}V

REF 导入负值就可。

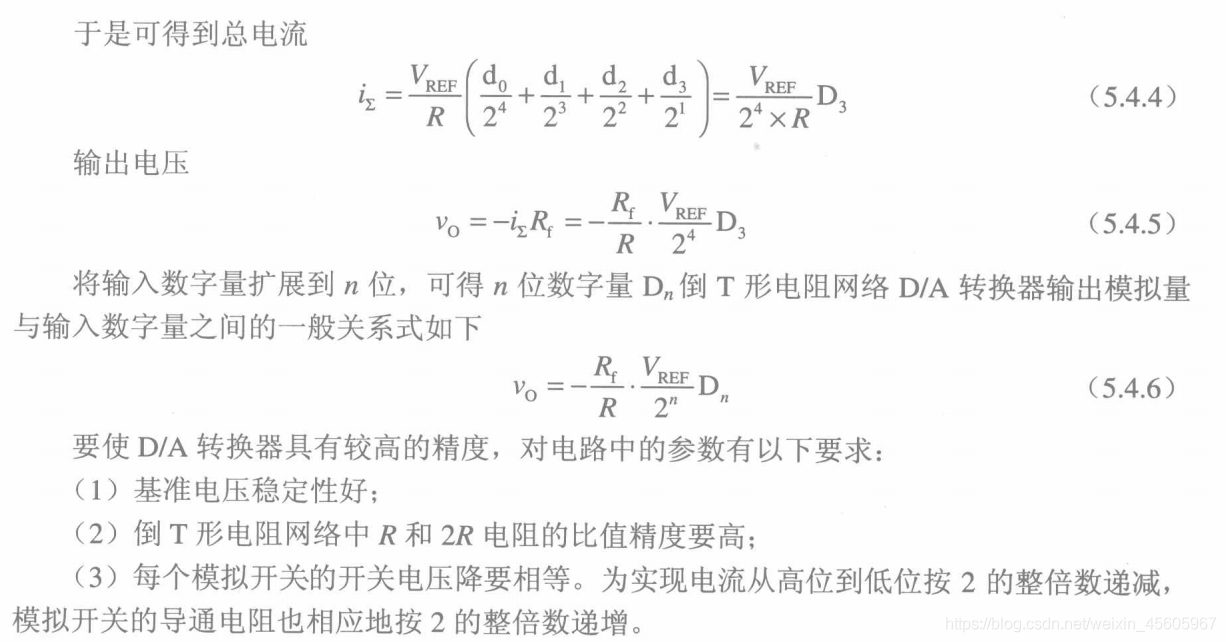
这个电路的问题是电阻的值之间相差可能过大，可能会引起误差，另外开关应该也是有阻值的，也会带来分压的问题。下面是基于此问题的改进版电路

2.倒T形电阻D/A

这里只用了两种电阻R/2R，但通过并联分流同样实现了电流的加和



且此电路中的2R电阻均接地或虚地，这样做的的好处是流经2R电阻的电流与开关位置无关，是一个确定的值。通过进一步推导可得：

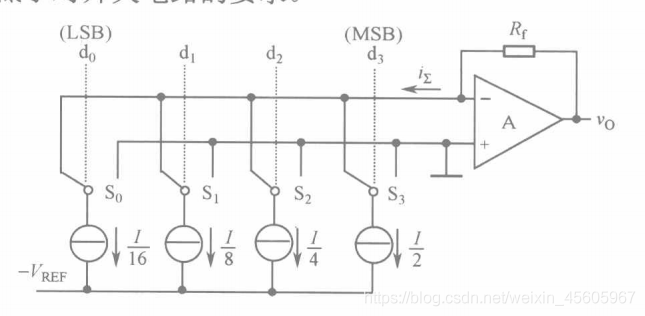


注意也要对模拟开关导通电阻的阻值做限制，不然会出现电流不准确的情况

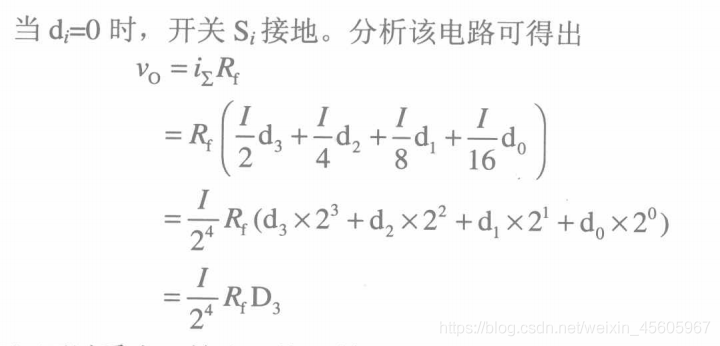
3.权电流D/A（利用电流源）

倒T型D/A中一旦开关或电阻的阻值没有保持二倍关系，电流关系就会随之变化，不太稳定。采取进一步的改进方案

采用恒流源就可以避免电阻阻值对电路的影响

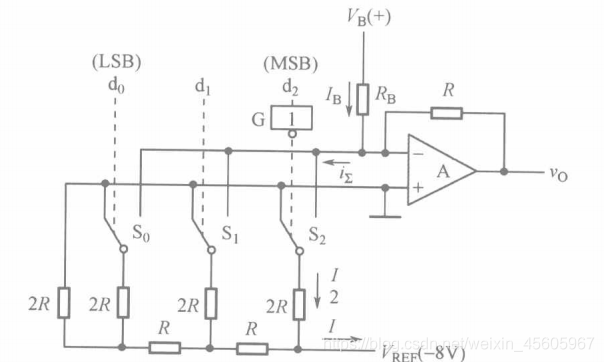


相应分析如下，可发现输出量是正比于输入的数字量的



4.双极性输出D/A

考虑到在实际情况中常把二进制数转换为补码的形式，则若把原码输入后对应的数再转换为转化为补码时偏移的输出就更完善了。对应关系如下



且由于之前的电路只能单极输出，不满足现在的需求，可通过在运放负极加一个带有V B ( + ) V\_B(+)V

B (+)和R的偏置电路解决。下面引出的V R E F ( − ) V\_{REF}(-)V

REF (−) 当然是负的，这样就能实现原码转化为补码并双极输出

电路图如下：

