DAC 数模转换

1. 实验目的

- 了解 DAC 数模转换原理及 RC 积分电路原理。
- 掌握 STC15W4K32S4 系列单片机实现 DAC 功能的硬件和软件设计。

2. 实验内容

- 编写程序，配置 PCA 及 ADC 相关寄存器，实现 DAC 输出不同电压值的模拟信号并通过 ADC 口检测电压值在串口调试助手显示。

3. 硬件设计

3.1. DAC 概念介绍

DAC (全称是 Digital to Analog Convertor)数模转换器是一种将数字信号转换为模拟信号（以电流、电压或电荷的形式）的设备，在很多数字系统中（例如计算机、单片机），信号以数字方式（0 或者 1）存储和传输，而数模转换器 DAC 可以将这样的信号转换为模拟信号，从而使得他们能够被外界（人或其他非数字系统）识别。

数模转换器 DAC 的常见用法是在音乐播放器中将数字形式存储的音频信号输出为模拟的声音。有的电视机的显像也有类似的过程。数模转换器 DAC 有时会降低原有模拟信号的精度，因此转换细节常常需要筛选，使得误差可以忽略。

DAC 有几个常见且重要的性能参数，如分辨率、线性度、建立时间、绝对精度和相对精度等。这里简单说下分辨率和线性度的概念。DAC 的分辨率是输入数字量的最低有效位（LSB）发生变化时，所对应的输出模拟量（电压或电流）的变化量。其反映了输出模拟量的最小变化值。DAC 的线性度（也称非线性误差）是实际转换特性曲线与理想直线特性之间的最大偏差。常以相对于满量程的百分数表示。

现市场上越来越多的 MCU 芯片内部会集成 DAC 外设，也有专用 DAC 芯片，包括一些音频芯片都属于 DAC 芯片，比如 PCM5102A 等。说到单片机 DAC 外设，一定会给出这个 DAC 外设是多少位的，比如单片机片内有 16 位 DAC，就意味着数字信号的范围是 0~65535，假设该 DAC 输出是 0~3.3V 的电压型模拟信号，那 DAC 转换可分辨的最小电压（分辨率）就是 3300mV/65535=0.05mV，这也就意味 DAC 数字信号输出有 1bit 的抖动，那么实际输出的模拟信号就有 0.05mV 的偏差。

DAC 核心部分是由 R-2R 电阻网络(也称倒 T 型电阻网络)、模拟开关和运算放大器所组成。这里之所以说核心部分是因为有些专用 DAC 芯片，比如音频芯片可能还有 I2S 接口相关的外围电路等。下面简单介绍下倒 T 型电阻网络。解析这个原理图时我们必须知道戴维南等效电源定理，然后从右侧向左去等效可分析出 I7 是 I 的 1/2，I6 是 I 的 1/4，I5 是 I 的 1/8，I4 是 I 的 1/16，I3 是 I 的 1/32，I2 是 I 的 1/64，I1 是 I 的 1/128，I0 是 I 的 1/256，如此

每一位都发挥了有效的位权，输出电压有 256 种变化，该 T 型电阻网络等效输出表达式为 D0*Vref/256+D1*Vref/128+D2*Vref/64+D3*Vref/32+D4*Vref/16+D5*Vref/8+D6*Vref/4+D7*Vref/2。

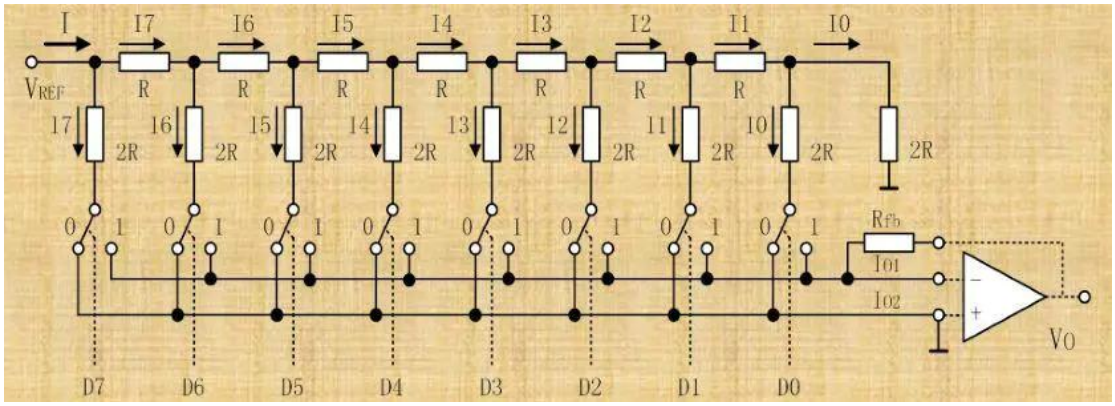

图 1：T 型电阻网络示意图

✧ 注：本 T 型电阻网络是 8 位 DAC 举例，若是 10 位、12 位、16 位依次按此方法增加 R 电阻和 2R 电阻电路部分。

3.2. 开发板 DAC 硬件电路介绍

STC15W4K32S4 系列单片机内部没有集成 DAC 外设，所以进取者 STC15 开发板实现 DAC 转换是基于将高速 PWM 信号通过 RC 电路整合成比较平缓的电压信号作为模拟输出，而改变高速 PWM 信号的占空比可改变输出电平信号的幅值。为了达到比较理想的电压信号输出，P2.5 口输出的 PWM 信号经 2 级 RC 电路整合，如下图。

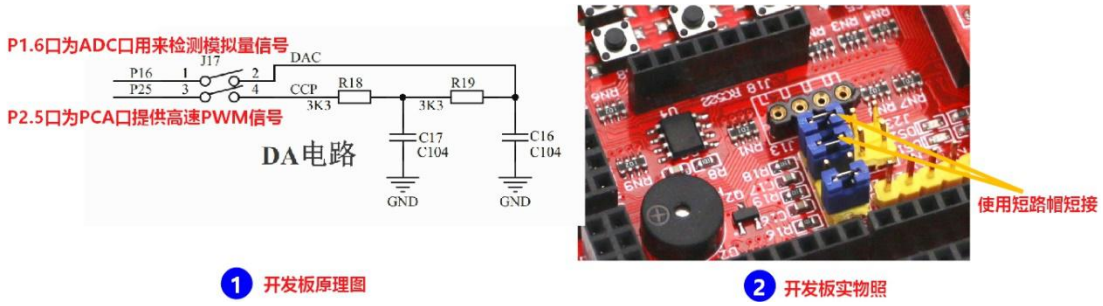

图 2：开发板 DAC 转换电路

✧ 注：开发板出厂 J17 端子短路帽没有短接，做 DAC 实验需短接 J17 端子，另外，开发板出厂会短接 J22 端子的 TOUCH 到 P2.5 口，做 DAC 实验需去掉该短路帽。

■ RC 积分电路简介

由电阻 R 和电容 C 构成的电路称为阻容电路，简称 RC 电路。RC 电路是电子电路中非常常见的一种电路，但 RC 电路的种类和变化很多，首先我们介绍下 RC 积分电路。

在 RC 电路分析中，有时要用到时间常数这一概念。时间常数为电容量与电阻值的乘积。在电容量大小不变时，电阻值决定了时间常数的大小。电阻值不变时，电容量的大小决定了时间常数的大小。输入信号加在电阻 R1 上，输出信号取自电容 C1。输入信号是矩形脉冲，在积分电路中，要求 RC 电路中的时间常数远大于脉冲宽度。

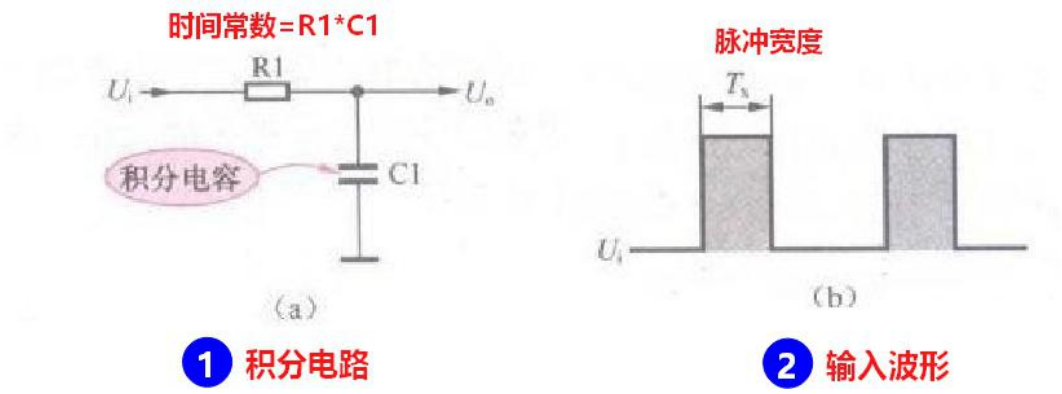

图 3：RC 积分电路分析示意图

✧ 注：RC 积分电路的作用是：消减变化量，突出不变量。故 RC 积分电路可将矩形脉冲波转换为锯齿波或三角波，还可将锯齿波转换为抛物波。

RC 积分电路的条件是时间常数远大于输入信号的脉冲宽度，进取者 STC15 的 RC 电路中，R1 取值 3.3K、C1 取值 100nF，可计算时间常数为 R1*C1=3.3KΩ*100*10^{-9}F=3.3*10^{-4}。如果时间常数远大于输入信号的脉冲周期，则也一定满足 RC 积分电路的条件。换句话说如果时间常数的倒数远小于输入信号的频率，则也一定满足 RC 积分电路的条件。时间常数的倒数可计算出为 3.3*10^{-4} 的倒数约为 3030，所以输入信号的频率只要远大于 3.03KHZ 即可满足积分电路的条件（一般按照 10 倍来作为基本条件）。

进取者 STC15 控制 P2.5 口输出 PWM 信号的频率约是 43.2KHZ（具体不详述），该频率满足 RC 积分电路的条件，所以可将 P2.5 口输出的 PWM 信号整合成比较平缓的电压信号（可使用示波器辅助测量之）。又因为单片机控制 P2.5 口输出的 PWM 信号频率不变、占空比变化，所以最后整合出的电压信号的幅值会不同，从而实现输出不同模拟量的目的。

■ RC 微分电路简介

RC 微分电路和 RC 积分电路在电路形式上相近，RC 微分电路输出电压取自电阻，而且 RC 时间常数与积分电路不同。RC 微分电路中，要求 RC 时间常数远小于输入信号的脉冲宽度。如下图所示。

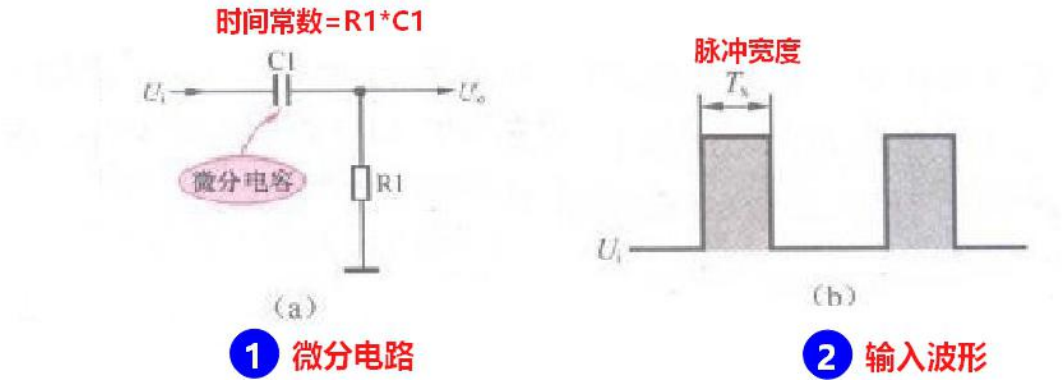

图 4：RC 微分电路分析示意图

✧ 注：RC 微分电路的作用是：消减不变量，突出变化量。故 RC 微分电路可把矩形波转

换为尖脉冲波，电路的输出波形只反映输入波形的突变，即只有输入波形发生突变的瞬间才有输出。而对恒定部分则没有输出。

除了上面描述的 RC 微分电路和 RC 积分电路，RC 电路还有很多不同应用，比如 RC 耦合电路、RC 脉冲分压器以及 RC 滤波电路等。

4. 软件设计

4.1. DAC 检测 - 串口调试助手实验

✧ 注：本节的实验源码是在“实验 2-12-2：ADC 光敏电阻检测（电压值）”的基础上修改。本节对应的实验源码是：“实验 2-13：DAC 检测 - 串口调试助手”。

4.1.1. 工程需要用到的 c 文件

本例需要用到的 c 文件如下表所示，工程需要添加下表中的 c 文件。

表 3：实验需要用到的 c 文件

序号	文件名	后缀	功能描述
1	uart	.c	包含与用户 uart 有关的用户自定义函数。
2	adc	.c	ADC 有关的用户自定义函数。
3	pca	.c	PCA 有关的用户自定义函数。
4	delay	.c	包含用户自定义延时函数。

4.1.2. 头文件引用和路径设置

■ 需要引用的头文件

```
1. #include "delay.h"
2. #include "uart.h"
3. #include "adc.h"
4. #include "pca.h"
```

■ 需要包含的头文件路径

本例需要包含的头文件路径如下表：

表 4：头文件包含路径

序号	路径	描述
1	..\ Source	uart.h、adc.h、pca.h 和 delay.h 头文件在该路径，所以要包含。
2	..\User	15W4KxxS4.h 头文件在该路径，所以要包含。

MDK 中点击魔术棒，打开工程配置窗口，按照下图所示添加头文件包含路径。

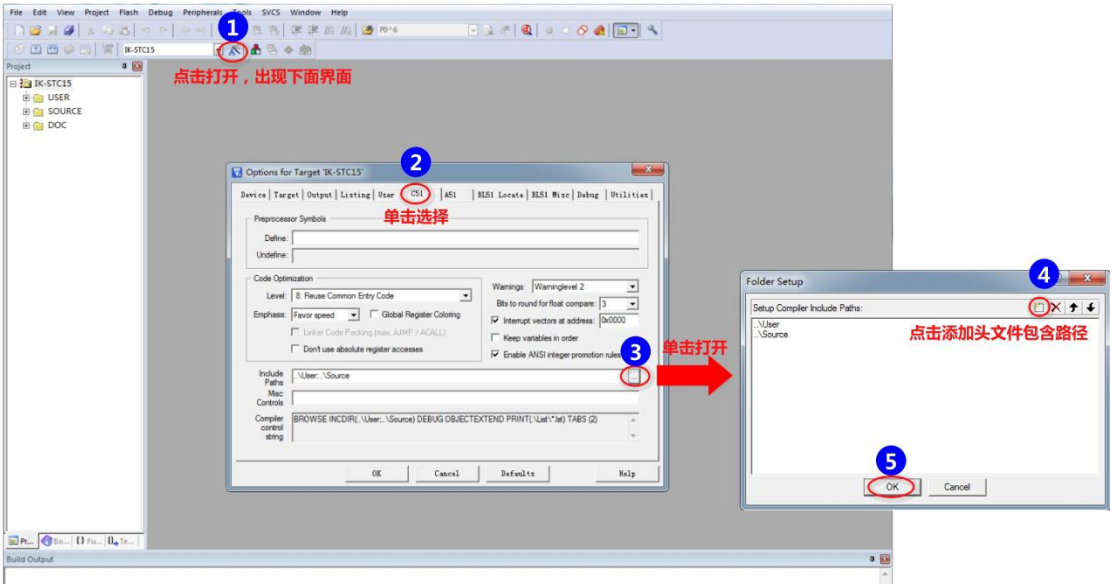

图 5：添加头文件包含路径

4.1.3. 编写代码

首先，在 adc.c 文件中编写操作 ADC 外设会用到的函数，如下表所示。

表 5：ADC 操作函数汇集

序号	函数名	功能描述
1	ADC_config	ADC 口的初始化操作。
2	Get_ADC10bitResult	读取 ADC 转换原始值。
3	HandleADC	处理 ADC 原始值。

关于上面 2 个 ADC 基本操作函数，下面详细给出代码。

程序清单：ADC 口初始化函数

```
1.  /*************************************************************************
2.  功能描述：ADC 口初始化
3.  入口参数：无
4.  返回值：无
5.  *************************************************************************/
6.  void    ADC_config(void)
7.  {
8.      ADC_CONTR|=0x80;          //开 AD 转换电源
9.      delay_ms(10);             //适当延时等待 AD 转换供电稳定
10.     P1ASF|=0x40;              //选择 P1.6 作为模拟功能 AD 使用
11.     ADC_CONTR|=0x06;          //选择 P1.6 作为 AD 转换通道输入使用
12.    ADC_CONTR&=0xFE;          //选择 P1.6 作为 AD 转换通道输入使用
13.     ADC_CONTR|=0x60;          //AD 转换速度为 90 个时钟周期转换一次
14.     ADC_CONTR&=0xEF;          //清 AD 转换完成标志
15.     EADC=0;                   //禁止 ADC 转换中断
```

```
    CLK_DIV|=0x20;          //ADC 转换结果 ADC_RES 存高 2 位，ADC_RESL 存低 8 位
    ADC_CONTR|=0x08;        //启动 AD 转换，ADC_START=1
}
```

程序清单：读取 ADC 转换原始值函数

```
/*************************************************************************
功能描述：ADC 口检测 AD 转换值函数
入口参数：无
返回值：ADC 10 位数据
 *************************************************************************/
uint16  Get_ADC10bitResult(void)
{
    uint16  AD_Dat=0;
    ADC_CONTR&=0xE7;            // 将 ADC_FLAG 清 0
    //10 位 AD 结果的高 2 位放 ADC_RES 的低 2 位，低 8 位在 ADC_RESL
    AD_Dat = ADC_RES;               //将 ADC_RES 低 2 位移到应在的第 9 位和第 10 位
    AD_Dat <<= 8;
    AD_Dat|= ADC_RESL;          //将 ADC_RESL 的 8 位移到应在的低 8 位

    ADC_CONTR|=0x08;            //重新启动 AD 转换，ADC_START=1。
    return  AD_Dat;
}
```

程序清单： ADC 转换原始值处理函数

```
/*************************************************************************
功能描述：将采集的原始值转换为电压值
入口参数：无
返回值：实测电压值
 *************************************************************************/
float HandleADC(void)
{
    uint16 Temp_signal;
    float g_voltage;

    //读取采集的原始值
    Temp_signal=Get_ADC10bitResult();

    //如果开发板 J3 选择 5V 工作电源，计算电压值公式：V=Temp_signal/1024*5.0。
    if((Temp_signal<TEMPMAX)&&(Temp_signal>TEMPMIN))
    {
    g_voltage=(5.0*Temp_signal)/1024;
    }
    //如果开发板 J3 选择 3.3V 工作电源，计算电压值公式：V=Temp_signal/1024*3.3。
```

```
20. //  if((Temp_signal<TEMPMAX)&&(Temp_signal>TEMPMIN))
21. //  {
22. //    g_voltage=(3.3*Temp_signal)/1024;
23. //  }
24.
25.   //返回实测电压值
26.     return g_voltage;
27. }
```

然后，在 pca.c 文件中编写 PCA 的初始化函数 PCAInit，代码如下。

程序清单：PCA 初始化函数

```
1.   /*************************************************************************
2.  功能描述：PCA 初始化
3.  入口参数：无
4.  返回值：无
5.   *************************************************************************/
6.  void PCAInit(void)
7.  {
8.      AUXR1 &= 0xEF;          //选择 PCA 模块 0 为引脚 P2.5，选择 PCA 模块 1 为引脚 P2.6
9.      AUXR1 |= 0x20;          //选择 PCA 模块 0 为引脚 P2.5，选择 PCA 模块 1 为引脚 P2.6
10.
11.     CCON = 0x00;                    //CF、CR、CCF1、CCF0 位均清零
12.     CMOD &= 0x7F;                   //CIDL 位置 0，空闲模式下 PCA 计数器仍然工作
13.     CMOD &= 0xF1;             //CP2、CP1、CP0 设置为 100，PCA 时钟源选择为系统时钟
14.     CMOD |= 0x08;           //CP2、CP1、CP0 设置为 100，PCA 时钟源选择为系统时钟
15.     CMOD &= 0xFE;           //ECF 位置 0，禁止寄存器 CCON 中 CF 位中断（禁止 PCA 计时中断）
16.     CL = 0x00;                      //PCA 计数器赋初值
17.     CH = 0x00;                      //PCA 计数器赋初值
18.     //PCA 模块 0 初始化部分
19.     CCAPM0 |= 0x40;                 //ECOM0 位置 1，允许比较器功能
20.     CCAPM0 &= 0xDF;                 //CAPP0 位置 0，禁止上升沿捕获
21.     CCAPM0 &= 0xEF;                 //CAPN0 位置 0，禁止下降沿捕获
22.     CCAPM0 &= 0xF7;                 //MAT0 位置 0，禁止匹配控制位
23.     CCAPM0 &= 0xFB;                 //TOG0 位置 0，禁止翻转控制位
24.     CCAPM0 |= 0x02;                 //PWM0 位置 1，开启 PWM 模式
25.     CCAPM0 &= 0xFE;                 //ECCF0 位置 0，禁止 CCF0 中断
26.     PCA_PWM0 &= 0x3F;               //PCA 模块 0 工作于 8 位 PWM 功能
27.     PCA_PWM0 &= 0xFC;                       //EPC0H 位和 EPC0L 位置 0
28.     CCAP0L = 0x00;                  //PCA 比较值寄存器赋初值
29.
30.     CR = 1;                         //启动 PCA 计数器阵列计数
31. }
```

最后，在主函数中对串口 1 进行初始化，主循环中每 100ms 通过串口 1 发送读取的端口 P16 的电压值。

代码清单：主函数

```
1.  int main()
2.  {
3.      uint16  TempData=0;
4.  ////////////////////////////////////////////////
5.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
6.  //       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
7.  //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
8.  //         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
9.  ////////////////////////////////////////////////
10.     P1M1 &= 0xBF;   P1M0 &= 0xBF;                   //设置 P1.6 为准双向口
11.     P2M1 &= 0xDF;   P2M0 &= 0xDF;                   //设置 P2.5 为准双向口
12.     P3M1 &= 0xFC;   P3M0 &= 0xFC;                   //设置 P3.0~P3.1 为准双向口
13.
14.     PCAInit();                                      //PCA 初始化
15.     ADC_config();                                   //ADC 初始化
16.     Uart1_Init();                                   //串口 1 初始化
17.     EA = 1;                                         //使能总中断
18.   delay_ms(10);                                    //初始化后延时
19.     while (1)
20.     {
21.         TempData++;
22.         CCAP0H = (uint8)(256 - TempData);           //P2.5 引脚输出频率不变但占空比
    不断变化的脉冲信号
23.         if(TempData>255)                        //占空比达到很大时重新设定占空比
24.             TempData=1;
25.         delay_ms(20);
26.         printf("\r\n ADC_P16 端口电压值:  %.1fV\r\n",HandleADC());      //串口打印上
    传的采集的电压值
27.         delay_ms(100);
28.     }
29. }
```

4.1.4. 硬件连接

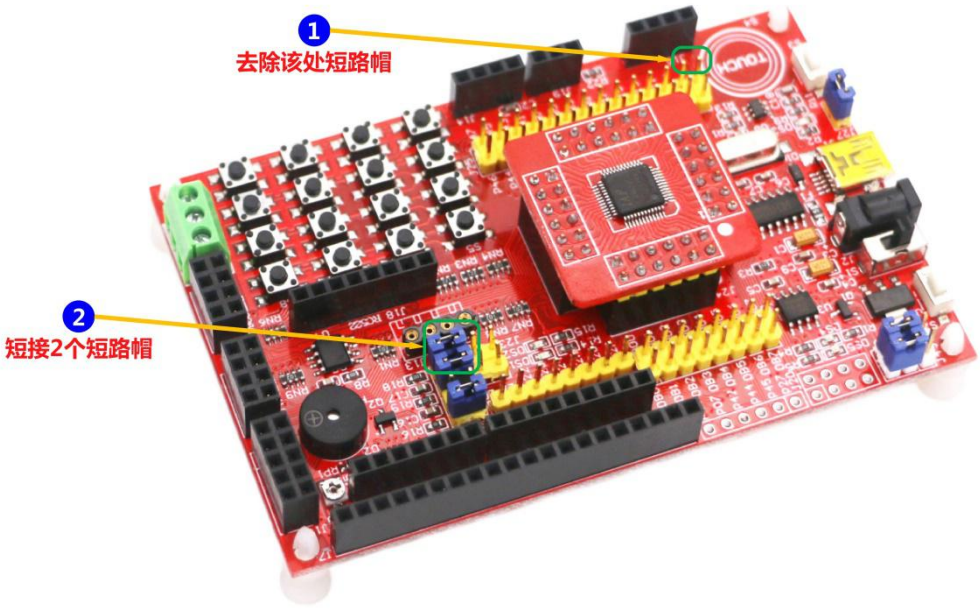

图 6：开发板连接图

4.1.5. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-13：DAC 检测 - 串口调试助手”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\DAC\project”目录下的工程“DAC.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“DAC.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，打开串口调试助手，选择好串口号，设置波特率为 9600，可以观察到串口调试助手实时显示“ADC_P16 端口电压值： xxxxV”内容。（xxxx 是实际的数字）