第 2-14 讲： PCF8591 芯片之 AD 与 DA 转换

1. 学习目的

1. 了解 ADC 的基本概念：分辨率、精度等。
2. 了解 DAC 数模转换原理。
3. 掌握 IAP15F2K61S2/IAP15W4K61S4 单片机模拟 I2C 的特点以及编程方法。
4. 掌握单片机通过模拟 I2C 控制 PCF8591 芯片实现 AD 和 DA 转换的程序设计。

2. ADC 与 DAC 基本概念

2.1. ADC 简介

实际应用中，我们经常需要将模拟量转换为数字量供 CPU 处理，如电池电压检测、温度检测等等。对于 CPU 来说，他能处理的是数字量，所以，需要通过 A/D 转换(模数转换)将时间连续、幅值也连续的模拟量转换为时间离散、幅值也离散的数字信号，从而实现 CPU 对模拟信号的处理，这种能够实现 A/D 转换功能的电路称之为模数转换器（ADC：Analog-to-digital converter）。

ADC 的结构和实现原理有多种方式，常见的 ADC 的类型有积分型、逐次逼近型、并行比较型/串并行型、Σ-Δ 调制型等。举例最为常见的逐次逼近型 ADC 介绍下，他由一个比较器、数模转换器、寄存器和控制电路组成，在新的转换开始时，采样和保持电路对输入电压进行采样，然后将该采样信号逐次与数模转换器的输出信号进行比较，经 N 次比较而输出数字值。其优点是速度较高、功耗低。

ADC 常用的技术参数有以下几点，这是学习 ADC 必须要掌握的。

1. 分辨率

ADC 分辨率是指输出数字量变化一个最低有效位(LSB)所需的输入模拟电压的变化量。ADC 的分辨率用位数表示，如 10 位的 ADC，分辨率为 2^{10}=1024。如果 ADC 的量程为（0~5）V，那么 ADC 即可“分辨”出（5/1024）V 的电压变化。

2. ADC 精度

ADC 的精度取决于量化误差及系统内其他误差的总和。这里要特别注意 ADC 分辨率和 ADC 精度的区别，“精度”是用来描述物理量的准确程度的，而“分辨率”是用来描述刻度划分的。对于一个给定的器件，他的分辨率是固定的，如 PCF8591，他的分辨率固定为 8 位的，但是他的精度不仅仅受器件本身的影响，还可能会受 PCB 布线、外界环境（温度、湿度、干扰等）的影响而变化。

3. 转换速度

转换速度是指完成一次从模拟转换到数字的 AD 转换所需的时间的倒数。

2.2. DAC 简介

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）的设备或电路。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。

T 型电阻网络方式是一种常见的 DAC 实现方法，由 T 型电阻网络和运算放大器组成，下图是 8 位 DAC 的原理示意图。输入数字量中的每位都按其权值分别转换为模拟量，之后通过运算放大器求和相加。

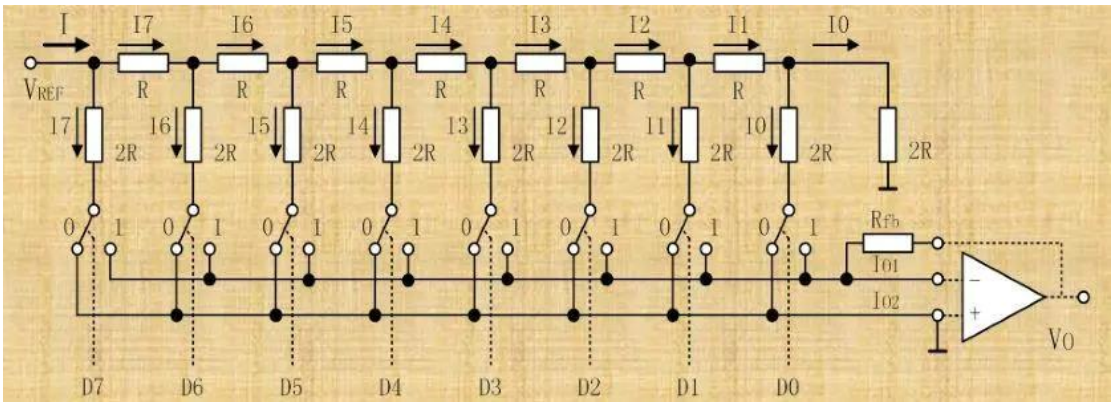

图 1：T 型电阻网络示意图

上图中，由于运算放大器的反相输入端为“虚地”，所以无论模拟开关连接到“0”还是 “1”，从 T 形电阻网络节点对“地”往右看的等效电阻均为 R，由此可计算出基准电流 $I=V_{REF}/R$。再根据电流可计算出流过各个分支的电流从右向左（I0~I7）依次是 I/2、I/4、I/8、I/16、I/32、I/64、I/128 和、I/256。

由此，每一位数字量都发挥了有效的位权，流向运算放大器反相输入端的总电流如下：

$$I_{\sum} = I_7 \times D7 + I_6 \times D6 + \cdots + I_0 \times D0$$

该电流经过运算放大器换成模拟电压输出，从而实现由数字信号到模拟信号的转换。这里以 8 位 DAC 示例，输出电压有 256 种变化，当然，这种 T 形电阻网络的转换原理可以推广到 n 位，实现 n 位 DAC。

3. 硬件设计

PK107D 开发板上设计了 I2C 接口的 PCF8591 电路单元，用于我们学习 ADC 和 DAC 的应用。（I2C 接口的学习在 AT24C02 存储器章节有详细介绍，这里不再赘述）

3.1. PCF8591 芯片

PCF8591 是一款单芯片、单电源、低功耗的 8 位 CMOS 数据采集芯片，其具有 4 路模拟输入和 1 个模拟输出接口，允许使用多达 8 个设备连接到 I2C 总线而不需要额外的硬件。该芯片的功能主要包括模拟输入多路复用、片上跟踪和保持功能、8 位模数转换和 8 位数模转换。最大转换速率取决于 I2C 总线的最高速率。

3.1.1. PCF8591 芯片内部结构

PCF8591 芯片有 16 个引脚，引脚功能介绍见下表 1，芯片内部结构图如下。

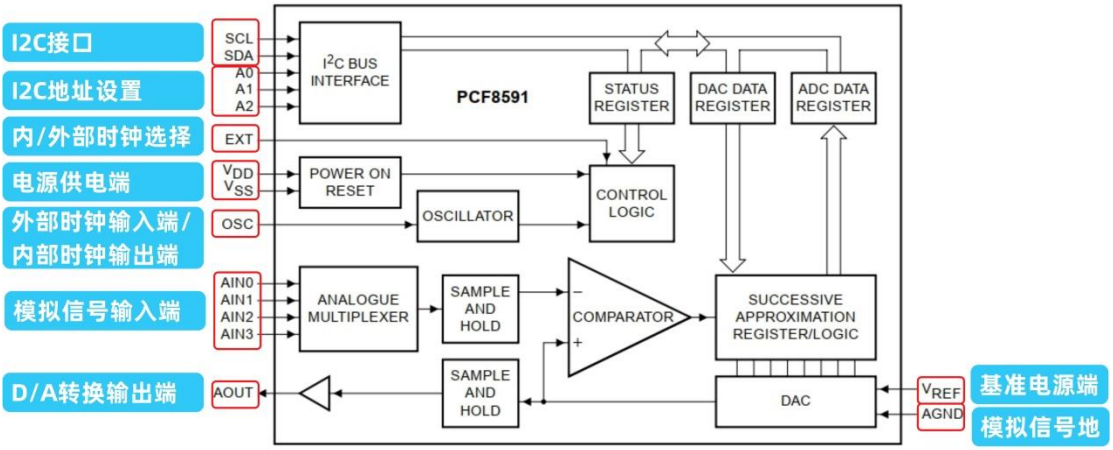

图 2：PCF8591 芯片内部结构示意图

表 1：PCF8591 芯片引脚描述

序号	针脚名称	功能描述
1	AIN0	模拟信号输入端通道 0。
2	AIN1	模拟信号输入端通道 1。
3	AIN2	模拟信号输入端通道 2。
4	AIN3	模拟信号输入端通道 3。
5	A0	引脚地址端 A0。
6	A1	引脚地址端 A1。
7	A2	引脚地址端 A2。
8	VSS	芯片供电地。
9	SDA	I2C 时钟线。
10	SCL	I2C 数据线。
11	OSC	外部时钟输入端，内部时钟输出端。
12	EXT	内部、外部时钟选择线，低电平时选择使用内部时钟。
13	AGND	模拟信号地。
14	VREF	基准电源端。
15	AOUT	D/A 转换输出端。
16	VDD	芯片供电正。

3.1.2. PCF8591 芯片读写地址

查询 PCF8591 数据手册可知其地址高 4 位固定为“1001”，如下图所示，紧跟着的 3 个位由芯片的引脚 A2、A1 和 A0 的电平确定，最低位为读写位。开发板上 PCF8591 的硬件电路中将引脚 A2、A1 和 A0 连接到了 GND，因此 A2、A1 和 A0 均为 0，由此可提取出 7 位地址为 0x48，转换为 8 位地址：0x90（写）、0x91（读），如下图所示。

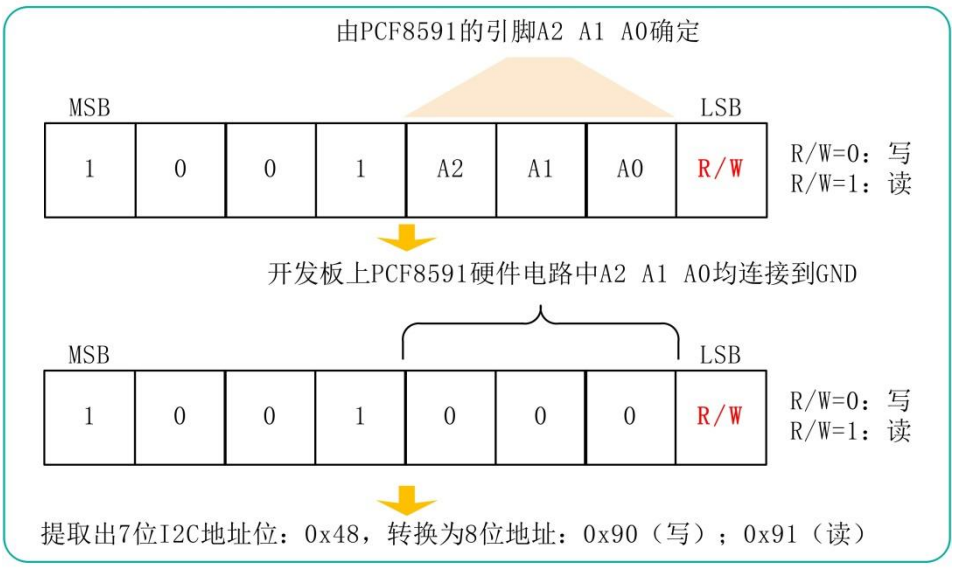

图 3：开发板 PCF8591 芯片读写地址

3.1.3. PCF8591 芯片控制寄存器

PCF8591 芯片控制寄存器实现芯片各功能，如 A/D 转换模拟信号选择哪几个通道输入、控制 D/A 转换的使能位等。关于芯片控制寄存器示意图如下。

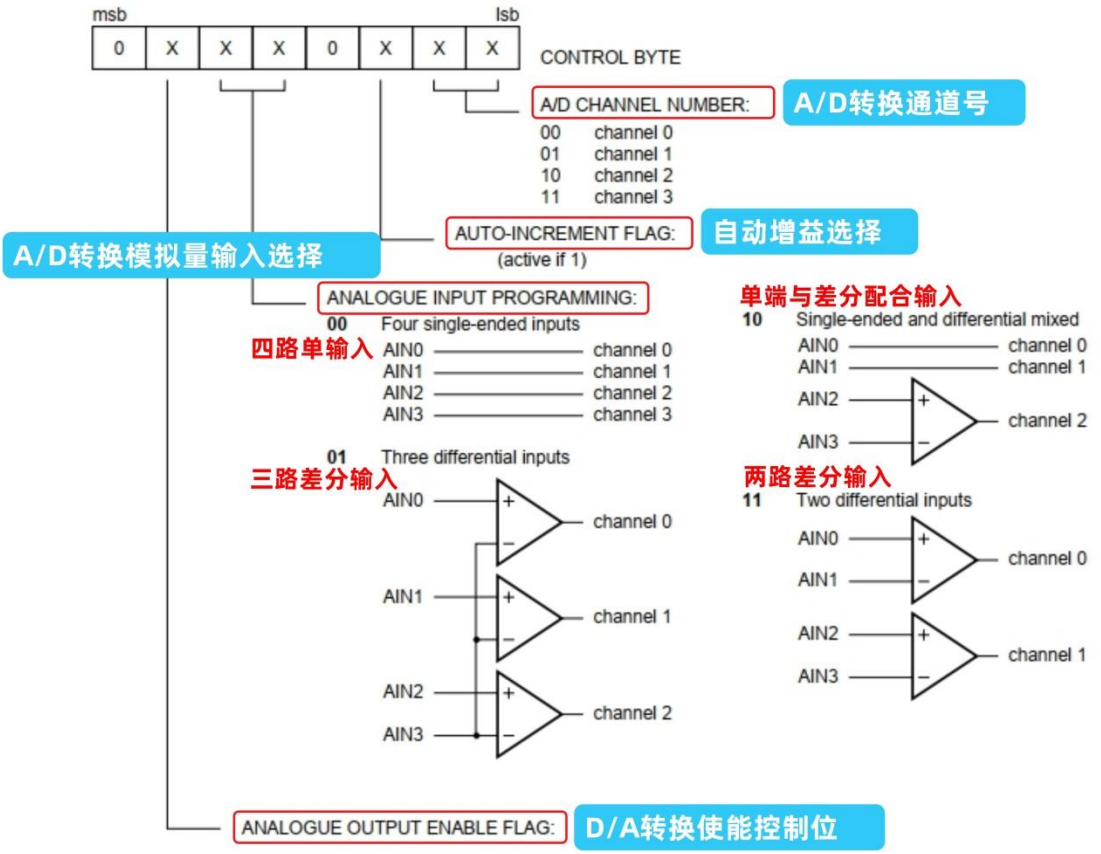

图 4：PCF8591 芯片控制寄存器

A/D 转换模拟量输入选择由寄存器 B4 和 B5 位确定（四路单端输入取 00，三路差分输入取 01，单端与差分输入取 10，两路差分输入取 11）。寄存器 B0 和 B1 位用于确定通道编号位，当对 0 通道的模拟信号进行 A/D 转换时取 00，当对 1 通道的模拟信号进行 A/D 转换时取 01，当对 2 通道的模拟信号进行 A/D 转换时取 10，当对 3 通道的模拟信号进行 A/D 转换时取 11。寄存器 B3 位是自动增益选择位，1 为有效位。寄存器 B6 位为 D/A 转换使能控制位，1 为有效位。寄存器 B3 和 B7 位未定义，固定值是 0。

3.2. PCF8591 电路

开发板上的 PCF8591 硬件电路如下图所示。PCF8591 的器件地址的低 3 位可以通过引脚 A2 A1 A0 配置，本电路中引脚 A2 A1 A0 均连接到 GND，因此，地址的低 3 位均为 0。

PCF8591 芯片通过 I2C 接口和 IAP15F2K61S2/IAP15W4K61S4 引脚 P2.0、P2.1 连接，芯片 AIN0 引脚和 D/A 转换输出引脚均被连接到 J25 单排针上，AIN1 引脚可用于检测光敏电阻变化引起的电压信号变化，AIN2 引脚接到其他电路上，AIN3 引脚可用于检测可调电位器 RV1 抽头电压变化情况。

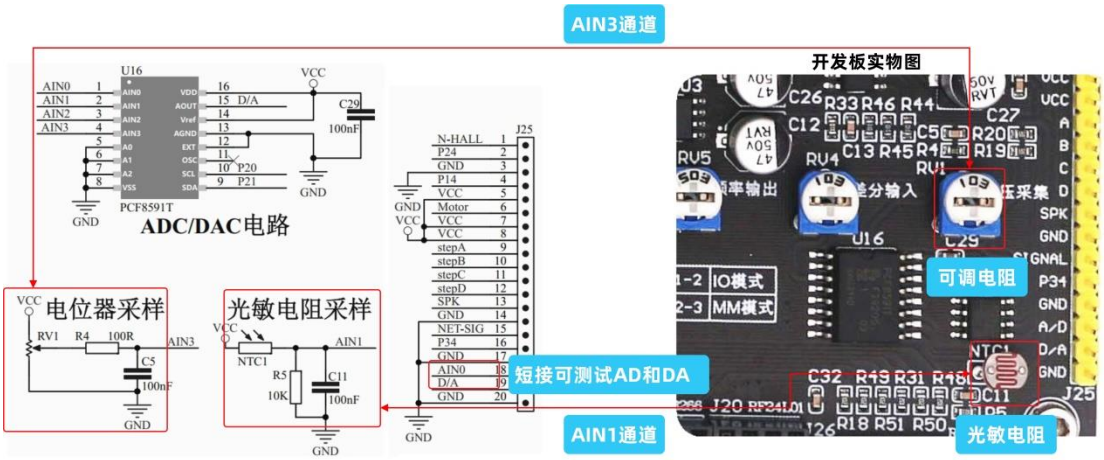

图 5：PCF8591 电路

PCF8591 电路占用的 IAP15F2K61S2/IAP15W4K61S4 的引脚如下表：

表 2：I2C 连接 PCF8591 引脚分配

名称	引脚	说明
SCL	P2.0	非独立 IO 口
SDA	P2.1	非独立 IO 口

✧ 注：关于 PCF8591 芯片的 A/D 和 D/A 功能的编程，后续在软件设计部分讲解。

4. 软件设计

4.1. PCF8591 模数转换 A/D 实验（数码管显示）

✧ 注：本节的实验是在“实验 2-13-1：模拟 I2C 读写 AT24C02 存储器”的基础上修改，本节对应的实验源码是：“实验 2-14-1：PCF8591 芯片 - ADC 测试 - 数码管显示”。

4.1.1. 实验内容

配置 IAP15F2K61S2/IAP15W4K61S4 单片机的 I2C 引脚，模拟 I2C 时序写函数，通过模拟 I2C 总线访问 PCF8591，完成对指定通道的模拟量采集。

4.1.2. 代码编写

1. 新建一个名称为“i2c_hw.c”的文件及其头文件“i2c_hw.h”保存到工程的“Source”文件夹，并将“i2c_hw.c”加入到 Keil 工程中的“SOURCE”组。该文件用于存放 I2C 硬件操作相关的函数。
2. 新建一个名称为“pcf8591.c”的文件及其头文件“pcf8591.h”保存到工程的“Source”文件夹，并将“pcf8591.c”加入到 Keil 工程中的“SOURCE”组。该文件用于存放 PCF8591 芯片操作相关的函数。
3. 引用头文件

因为在“main.c”文件中使用了“pcf8591.c”文件中的函数，所以需要引用下面的头文件“pcf8591.h”。

代码清单：引用头文件

```
1.  //引用 pcf8591 的头文件
2.  #include    " pcf8591.h"
```

4. PCF8591 地址的确定

开发板上 PCF8591 的硬件电路中将引脚 A2、A1 和 A0 连接到了 GND，此前已分析 PCF8591 读写地址：0xA0（写地址）、0xA1（读地址），如下图所示。

代码中，PCF8591 地址定义如下：

代码清单：定义 PCF8591 地址

```
1.  #define PCF8591_ADDR_W  0x90            //I2C 从机写地址
2.  #define PCF8591_ADDR_R  0x91            //I2C 从机读地址
```

5. PCF8591 从指定通道读取 A/D 转换数据

代码清单：从 PCF8591 指定通道读取 AD 采集数据

```
1. /*******************************************************************************
2.  * 描   述 : 从 PCF8591 指定通道读取 AD 采集数据
3.  * 参   数 : Channel：0x00~0x03  0x00:J25 端子 A/D 信号        0x01:光敏电阻 NTC1 分压值
4.  *                              0x02:LM324 第 7 引脚输出信号  0x03:电位器 RV1 抽头电压值
5.  * 返回值 : 读取的数据
6.  *******************************************************************************/
7. u8 PCF8591_read_byte(u8 Channel)
8. {
9.     u8 temp=0;
10.    I2C_Start();                    //发送起始命令，产生起始条件
11.    I2C_SendData(PCF8591_ADDR_W);   //发送器件地址（写）
12.    I2C_WaitACK();                  //接收 ACK
13.    I2C_SendData(Channel);          //发送读取数据的通道信息
14.    I2C_WaitACK();                  //接收 ACK
```

```
    I2C_Stop();                         //发送停止命令，产生停止条件
    delay_us(100);                      //读写操作过程中一些必要的延时

    I2C_Start();                        //发送起始命令，产生起始条件
    I2C_SendData(PCF8591_ADDR_R);       //发送器件地址（读）
    I2C_WaitACK();                      //接收 ACK
    temp=I2C_RecvData();                //读一个字节数据
    I2C_SendACK(1);                     //发送 ACK 命令
    I2C_Stop();                         //发送停止命令，产生停止条件
    return temp;                        //返回读取的数据
}
```

编写好 A/D 采集数据函数后，我们就可以通过该函数对 PCF8591 指定通道进行数据读取，具体功能如下。

- 按下 S4 按键：读取 PCF8591 芯片 0 通道的模拟量采集数据并在数码管显示。
- 按下 S5 按键：读取 PCF8591 芯片 1 通道的模拟量采集数据并在数码管显示。
- 按下 S8 按键：读取 PCF8591 芯片 2 通道的模拟量采集数据并在数码管显示。
- 按下 S9 按键：读取 PCF8591 芯片 3 通道的模拟量采集数据并在数码管显示。

代码清单：主函数

```
/*******************************************************************************
功能描述：主函数
入口参数：无
返 回 值：int 类型
*******************************************************************************/
int main(void)
{
    u8 btn_val;
    u16 temp=0;
    u16 adc_value=0;
    static u8 adc_channel=0;
    static u8 adcbuff[5];

    P2M1 &= 0x1F;   P2M0 |= 0xE0;     //设置 P2.5、P2.6、P2.7 为推挽输出
    P0M1 &= 0x00;   P0M0 |= 0xFF;     //设置 P0.0 ~ P0.7 为推挽输出
    P3M1 &= 0xF3;   P3M0 &= 0xF3;     //设置 P3.2 ~ P3.3 为准双向口
    P4M1 &= 0xEB;   P4M0 &= 0xEB;     //设置 P4.2、P4.4 为准双向口

    SEG_off();                  //控制 8 位数码管/点阵不显示
    leds_off();                 //熄灭 D1~D8 指示灯
    ULN2003_off();              //控制 ULN2003 输出高电平，关闭蜂鸣器、继电器等
    delay_ms(10);               //延时

```

```
24.     I2C_init();                     //IIC 初始化
25.     timer2_init();                  //timer2 初始化
26.     timer2_start();                 //启动 timer2
27.     EA = 1;                         //使能总中断
28.     delay_ms(10);                   //初始化后延时
29.
30.     while(1)
31.     {
32.         btn_val=keyboard_scan();            //获取开发板矩阵按键检测值
33.         //按下 S4：开启 ADC 采集通道 0
34.         if(btn_val == BUTTON1_PRESSED)
35.         {
36.             adc_channel = 1;                        //通道开启参数赋值为 1
37.         }
38.         //按下 S5：开启 ADC 采集通道 1
39.         else if(btn_val == BUTTON2_PRESSED)
40.         {
41.             adc_channel = 2;                        //通道开启参数赋值为 2
42.         }
43.         //按下 S8：开启 ADC 采集通道 2
44.         else if(btn_val == BUTTON5_PRESSED)
45.         {
46.             adc_channel = 3;                        //通道开启参数赋值为 3
47.         }
48.         //按下 S9：开启 ADC 采集通道 3
49.         else if(btn_val == BUTTON6_PRESSED)
50.         {
51.             adc_channel = 4;                        //通道开启参数赋值为 4
52.         }
53.        //数码管显示读取的计数值
54.         if(adc_channel)      //实测频率值 4 位数
55.         {
56.             temp =(u16)PCF8591_read_byte(adc_channel-1);
57.             temp =(u16)PCF8591_read_byte(adc_channel-1);  //软件滤波取第 2 次读取的值
58.             //PCF8591 是 8 位 ADC，参考电压是 5V，故采样电压=5V*采集值/(256-1)=5000mV*采集值
    /255 ≈ 19.61mV*采集值
59.             adc_value = 19.61*temp;  //将 ADC 采样值转换为电压(单位 mV)
60.             adcbuff[0]=adc_value%10000/1000;
61.             adcbuff[1]=adc_value%10000%1000/100;
62.             adcbuff[2]=adc_value%10000%1000%100/10;
63.             adcbuff[3]=adc_value%10000%1000%100%10%10;
64.
```

```
65.             LEDseg_DispUpdata(LEDSEG_5,adcbuff[0],LEDSEG_DP_ON);//更新第 5 个数码管显示内容，带小数点，
    电压显示单位为 V
66.             LEDseg_DispUpdata(LEDSEG_6,adcbuff[1],LEDSEG_DP_OFF);//更新第 6 个数码管显示内容
67.             LEDseg_DispUpdata(LEDSEG_7,adcbuff[2],LEDSEG_DP_OFF);//更新第 7 个数码管显示内容
68.             LEDseg_DispUpdata(LEDSEG_8,adcbuff[3],LEDSEG_DP_OFF);//更新第 8 个数码管显示内容
69.         }
70.     }
71.}
```

4.1.3. 硬件连接

本实验程序的编写都是基于 IO 模式，所以 J24 端子需要使用短路帽将该端子第 1 引脚和第 2 引脚短接，即选择为 IO 模式。同时因为控制 PCF8591 执行不同操作用到的是按键 S4、S5、S8 和 S9，所以 J6 端子第 2 引脚和第 3 引脚需跳线帽短接，即选择矩阵按键。

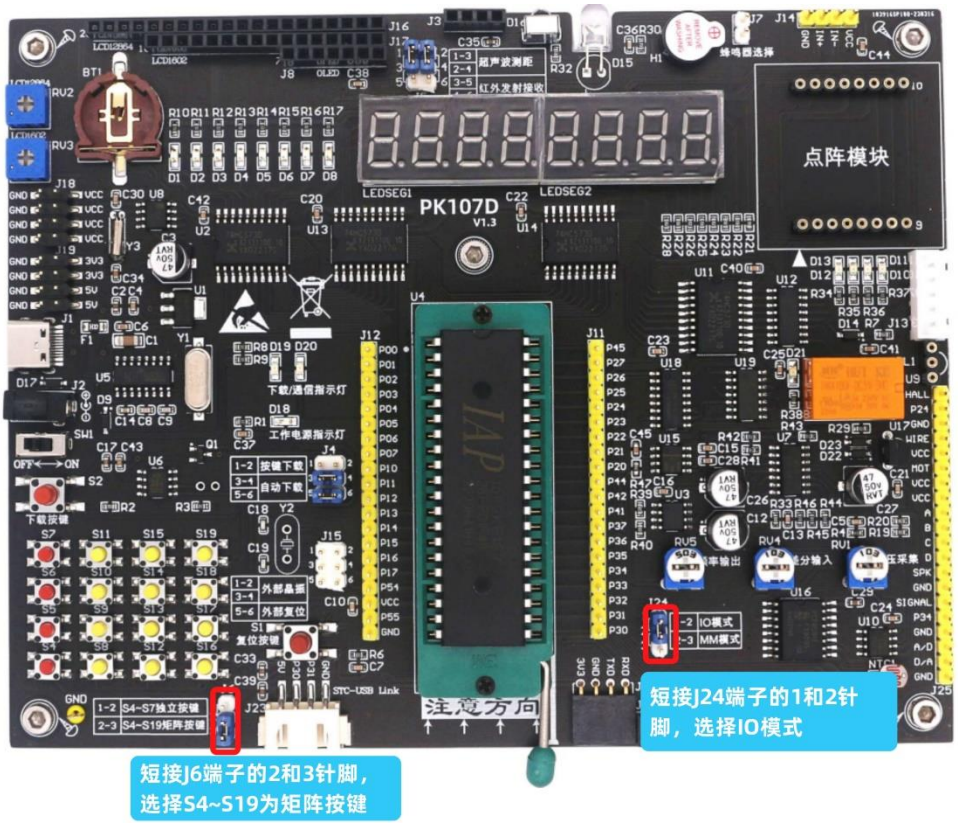

图 6：跳线帽短接

4.1.4. 实验步骤

1) 解压“…\第 3 部分：配套例程源码”目录下的压缩文件“实验 2-14-1：PCF8591 芯片 - ADC 测试 - 数码管显示”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”（这样做的目的是为了防止中文路径或者工程存放的路径过深导致打开工程出现问题）。
2) 双击“…\pcf8591\project”目录下的工程文件“pcf8591.uvproj”。
3) 点击编译按钮编译工程，编译成功后生成的 HEX 文件“pcf8591.hex”位于工程的“…

\pcf8591\Project\Object”目录下。

4) 打开 STC-ISP 软件下载程序，下载使用内部 IRC 时钟，IRC 频率选择：12 MHz。
5) 电脑上打开串口调试助手，选择开发板对应的串口号，将波特率设置为 9600bps。
6) 程序运行后，按下 S5 按键可读取光敏电阻 NTC1 分压值并通过数码管显示，按下 S8 按键可读取 LM324 第 7 引脚输出信号电压值并通过数码管显示，按下 S9 按键可读取电位器 RV1 抽头电压值并通过数码管显示（可用螺丝刀更改 RV1 抽头电压值观察显示值有没有变化）。

我们也编写好了串口调试助手显示 PCF8591 采集模拟量信息的例程，该例程在资料的“…\第 3 部分：配套例程源码目录下，其实验名称如下，读者在编写的过程中可以参考。

- 实验 2-14-2：PCF8591 芯片 - ADC 测试 - 串口显示。

4.2. PCF8591 数模转换 D/A 实验（数码管显示）

✧ 注：本节的实验是在“实验 2-14-1：PCF8591 芯片 - ADC 测试 - 数码管显示”的基础上修改，本节对应的实验源码是：“实验 2-14-3：PCF8591 芯片 - DAC 测试”。

4.2.1. 实验内容

配置 IAP15F2K61S2/IAP15W4K61S4 单片机的 I2C 引脚，模拟 I2C 时序写函数，通过模拟 I2C 总线访问 PCF8591，完成对指定通道的模拟量采集和模拟量输出控制。

4.2.2. 代码编写

1. 新建一个名称为“i2c_hw.c”的文件及其头文件“i2c_hw.h”保存到工程的“Source”文件夹，并将“i2c_hw.c”加入到 Keil 工程中的“SOURCE”组。该文件用于存放 I2C 硬件操作相关的函数。
2. 新建一个名称为“pcf8591.c”的文件及其头文件“pcf8591.h”保存到工程的“Source”文件夹，并将“pcf8591.c”加入到 Keil 工程中的“SOURCE”组。该文件用于存放 PCF8591 芯片操作相关的函数。
3. 引用头文件

因为在“main.c”文件中使用了“pcf8591.c”文件中的函数，所以需要引用下面的头文件“pcf8591.h”。

代码清单：引用头文件

```
1. //引用 pcf8591 的头文件
2.  #include    " pcf8591.h"
```

4. PCF8591 地址的确定

开发板上 PCF8591 的硬件电路中将引脚 A2、A1 和 A0 连接到了 GND，此前已分析 PCF8591 读写地址：0xA0（写地址）、0xA1（读地址），如下图所示。

代码中，PCF8591 地址定义如下：

代码清单：定义 PCF8591 地址

```
1.  #define PCF8591_ADDR_W  0x90            //I2C 从机写地址
2.  #define PCF8591_ADDR_R  0x91            //I2C 从机读地址
```

5. 控制 PCF8591 芯片 D/A 转换输出端输出模拟信号

代码清单：向 PCF8591 写入数字量以输出模拟信号

```
/*********************************************************************************
 * 描  述 : 向 PCF8591 写入数字量以在 DA 口输出模拟信号
 * 参  数 : dat：写入的数据
 * 返回值 : 无
 *********************************************************************************/
void PCF8591_write_byte(u8 dat)
{

    I2C_Start();                        //发送起始命令，产生起始条件

    I2C_SendData(PCF8591_ADDR_W);       //发送器件地址（写）
    I2C_WaitACK();                      //接收 ACK
    I2C_SendData(PCF8591_DAC);          //发送 DAC 输出模式命令
    I2C_WaitACK();                      //接收 ACK
    I2C_SendData(dat);                  //发送 DAC 输出模式下的数字信息
    I2C_WaitACK();                      //接收 ACK
    I2C_Stop();                         //发送停止命令，产生停止条件
}
```

编写好 D/A 转换输出模拟量函数后，我们就可以通过该函数控制 PCF8591 芯片 D/A 转换输出端输出预设的模拟量电压值，具体功能如下。

- 按下 S4 按键：控制 PCF8591 芯片 D/A 转换输出端输出约 0V 电压信号。
- 按下 S5 按键：控制 PCF8591 芯片 D/A 转换输出端输出约 1.6V 电压信号。
- 按下 S8 按键：控制 PCF8591 芯片 D/A 转换输出端输出约 3.3V 电压信号。
- 按下 S9 按键：控制 PCF8591 芯片 D/A 转换输出端输出约 5V 电压信号。
- 为方便演示，开发板会将芯片 D/A 转换输出端与芯片 0 通道的模拟量采集端短接，从而实现将 D/A 转换输出模拟量电压值在数码管上显示。

代码清单：主函数

```
/*************************************************************************
功能描述：主函数
入口参数：无
返 回 值：int 类型
*************************************************************************/
int main(void)
{
    u8 btn_val;
    u16 temp=0;
    u16 adc_value=0;
    static u8 temp_value=0;
```

```
12.     static u8 dac_value=0;
13.     static u8 adcbuff[5];
14.
15.     P2M1 &= 0x1F;   P2M0 |= 0xE0;      //设置 P2.5、P2.6、P2.7 为推挽输出
16.     P0M1 &= 0x00;   P0M0 |= 0xFF;      //设置 P0.0 ~ P0.7 为推挽输出
17.     P3M1 &= 0xF3;   P3M0 &= 0xF3;      //设置 P3.2 ~ P3.3 为准双向口
18.     P4M1 &= 0xEB;   P4M0 &= 0xEB;      //设置 P4.2、P4.4 为准双向口
19.
20.     SEG_off();                      //控制 8 位数码管/点阵不显示
21.     leds_off();                     //熄灭 D1~D8 指示灯
22.     ULN2003_off();                  //控制 ULN2003 输出高电平，关闭蜂鸣器、继电器等
23.     delay_ms(10);                   //延时
24.
25.     I2C_init();                       //IIC 初始化
26.     timer2_init();                    //timer2 初始化
27.     timer2_start();                   //启动 timer2
28.     EA = 1;                           //使能总中断
29.     delay_ms(10);                     //初始化后延时
30.
31.     while(1)
32.     {
33.         btn_val=keyboard_scan();              //获取开发板矩阵按键检测值
34.
35.         //PCF8591 是 8 位 DAC，参考电压是 5V，故输出电压=5V*输出数字值/(256-1)=5000mV*输出数字值
    /255 ≈ 19.61mV*输出数字值
36.         //按下 S4：DAC 输出通道赋值 0x01
37.         if(btn_val == BUTTON1_PRESSED)
38.         {
39.             dac_value = 0x01;                       //DAC 输出电压：19.61mV*1 = 0.019V
40.         }
41.         //按下 S5：DAC 输出通道赋值 0x56
42.         else if(btn_val == BUTTON2_PRESSED)
43.         {
44.             dac_value = 0x56;                       //DAC 输出电压：19.61mV*86 = 1.686V
45.         }
46.         //按下 S8：DAC 输出通道赋值 0xAB
47.         else if(btn_val == BUTTON5_PRESSED)
48.         {
49.             dac_value = 0xAB;                     //DAC 输出电压：19.61mV*171 = 3.353V
50.         }
51.         //按下 S9：DAC 输出通道赋值 0xFF
52.         else if(btn_val == BUTTON6_PRESSED)
53.         {
```

```
54.             dac_value = 0xFF;                    //DAC 输出电压：19.61mV*255 = 5.000V
55.         }
56.
57.         //DAC 输出按键选择的电压值，并在数码管显示电压值
58.         if((dac_value)&&(temp_value!=dac_value))
59.         {
60.             temp_value=dac_value;
61.             PCF8591_write_byte(temp_value);
62.
63.             //ADC 通道选择时务必将 DAC 功能也打开
64.             temp =(u16)PCF8591_read_byte(0x00|PCF8591_DAC);
65.             temp =(u16)PCF8591_read_byte(0x00|PCF8591_DAC);  //软件滤波取第 2 次读取的值
66.             //PCF8591 是 8 位 ADC，参考电压是 5V，故采样电压=5V*采集值/(256-1)=5000mV*采集值
    /255 ≈ 19.61mV*采集值
67.             adc_value = 19.61*temp;  //将 ADC 采样值转换为电压(单位 mV)
68.
69.             adcbuff[0]=adc_value%10000/1000;
70.             adcbuff[1]=adc_value%10000%1000/100;
71.             adcbuff[2]=adc_value%10000%1000%100/10;
72.             adcbuff[3]=adc_value%10000%1000%100%10%10;
73.
74.             LEDseg_DispUpdata(LEDSEG_5,adcbuff[0],LEDSEG_DP_ON);//更新第 5 个数码管显示内容，带小数点，
    电压显示单位为 V
75.             LEDseg_DispUpdata(LEDSEG_6,adcbuff[1],LEDSEG_DP_OFF);//更新第 6 个数码管显示内容
76.             LEDseg_DispUpdata(LEDSEG_7,adcbuff[2],LEDSEG_DP_OFF);//更新第 7 个数码管显示内容
77.             LEDseg_DispUpdata(LEDSEG_8,adcbuff[3],LEDSEG_DP_OFF);//更新第 8 个数码管显示内容
78.         }
79.     }
80.}
```

4.2.3. 硬件连接

本实验程序的编写都是基于 IO 模式，所以 J24 端子需要使用短路帽将该端子第 1 引脚和第 2 引脚短接，即选择为 IO 模式。同时因为控制 PCF8591 输出不同模拟量值用到的是按键 S4、S5、S8 和 S9，所以 J6 端子第 2 引脚和第 3 引脚需跳线帽短接，即选择矩阵按键。

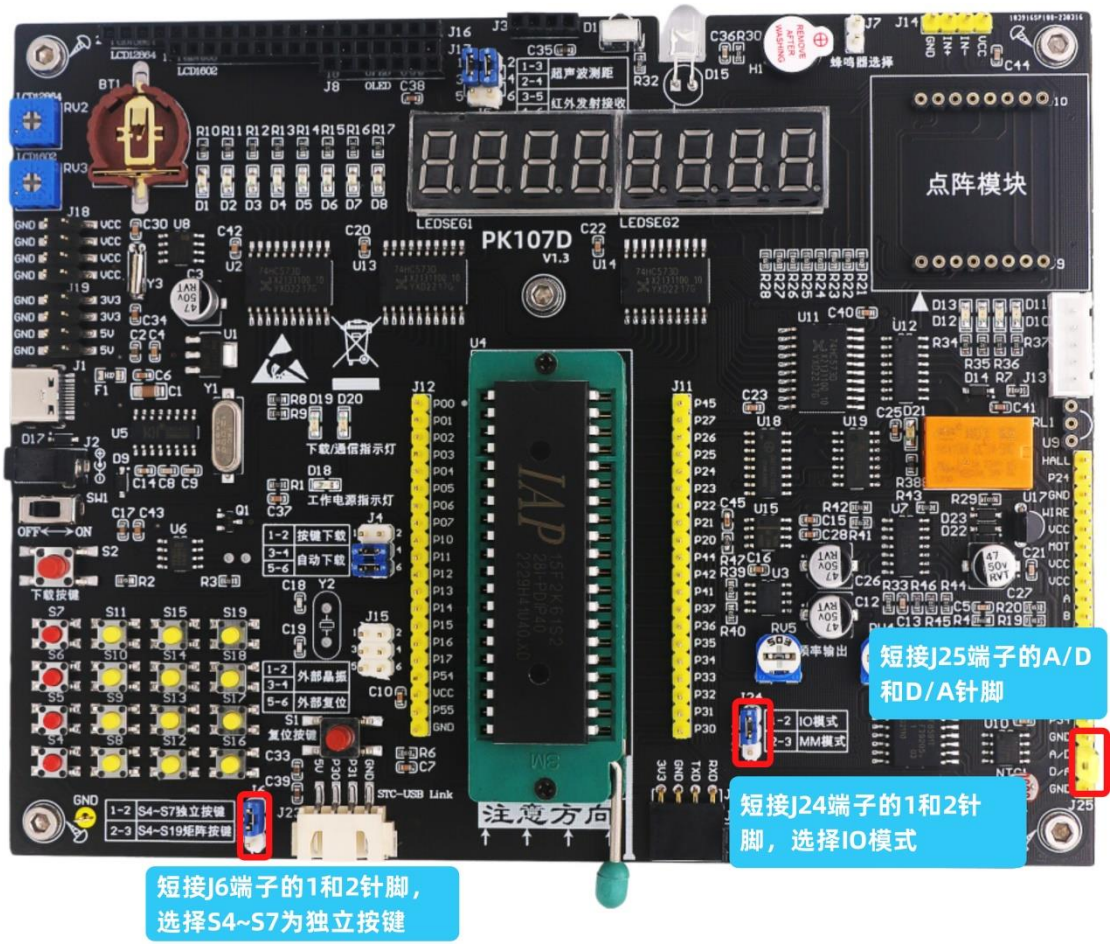

图 7：跳线帽短接

4.2.4. 实验步骤

1) 解压“…\第 3 部分：配套例程源码”目录下的压缩文件“实验 2-14-3：PCF8591 芯片 - DAC 测试”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”（这样做的目的是为了防止中文路径或者工程存放的路径过深导致打开工程出现问题）。
2) 双击“…\pcf8591\project”目录下的工程文件“pcf8591.uvproj”。
3) 点击编译按钮编译工程，编译成功后生成的 HEX 文件“pcf8591.hex”位于工程的“…\pcf8591\Project\Object”目录下。
4) 打开 STC-ISP 软件下载程序，下载使用内部 IRC 时钟，IRC 频率选择：12 MHz。
5) 电脑上打开串口调试助手，选择开发板对应的串口号，将波特率设置为 9600bps。
6) 程序运行后，按下 S4 按键可观察数码管显示的电压值约 0V，按下 S4 按键可观察数码管显示的电压值约 1.6V，按下 S4 按键可观察数码管显示的电压值约 3.3V，按下 S4 按键可观察数码管显示的电压值约 5V。