ADC 模数转换

1. 实验目的

- 掌握 STC15W4K32S4 系列单片机 ADC 外设的原理。
- 掌握 ADC 采样硬件电路设计及相关应用程序设计。

2. 实验内容

- 编写程序，配置 ADC 相关寄存器，实现串口调试助手显示 ADC 转换原始数值。
- 编写程序，配置 ADC 相关寄存器，实现串口调试助手显示实时电压值。
- 编写程序，使用 NTC 光敏电阻算法，实现串口调试助手显示光强度信息值。

3. 硬件设计

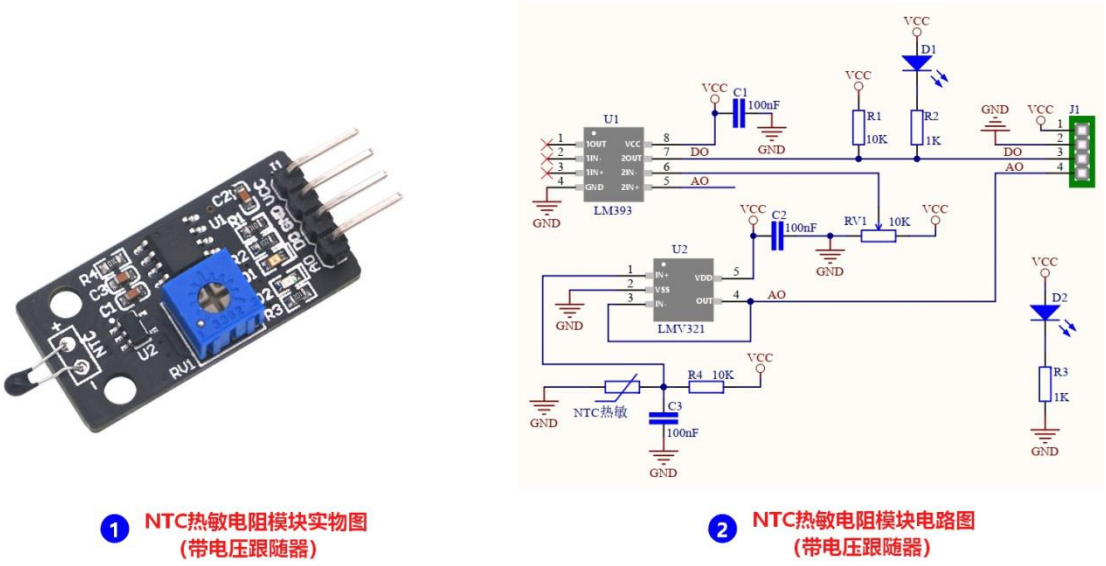
3.1. ADC 概念介绍

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，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。逐次逼近型 ADC 又称为逐位比较型 ADC，是模数转换器中应用很广泛的一种实现方式。其原理就是将输入的模拟信号与不同的参考电压按规律进行多次比较，使转换所得的数字量在数值上逐次逼近输入模拟量，最后得到转换后的数字量。供比较的数字量之间的最小变化值决定了 ADC 转换器精度。

ADC 转换器有几个常见且重要的性能参数，如转换精度、转换速度和通道数等。现市场上多数 MCU 芯片内部是集成了 ADC 外设的，也有专用 ADC 芯片，比如 TI 公司的 TLC548、ADI 公司的 AD7705 等。一般使用外部 ADC 芯片会提高 ADC 采集的精度，但成本会大大增加。片外 ADC 芯片与单片机之间的通信接口多是串行口（SPI 或 I2C 接口），这样占用的就是单片机的串行口外设资源。（不再是单片机的 ADC 通道了）

待采集的模拟信号无论是进单片机的 ADC 引脚还是接入片外专用 ADC 芯片的采集引脚，理论上都是要求具有尽可能大的输出阻抗。尤其是市场上微处理器厂家多、型号多，其片内 ADC 外设的结构会各不相同，这些 ADC 输入口对地的等效电阻数量级上也有差别，如果输入的待采集信号输出阻抗也在这个数量级上，那么很容易影响采集效果。所以，很多情况下会采用在待采集信号后加电压跟随器的办法解决这个问题。下面截图 NTC 热敏电阻模块（带电压跟随器）的原理图帮助大家直观理解下。



1 NTC热敏电阻模块实物图（带电压跟随器）

2 NTC热敏电阻模块电路图（带电压跟随器）

图 1：NTC 热敏电阻模块（带电压跟随器）

✧ 注：集成电路 U2 实现了电压跟随功能，AO 电平信号输出阻抗极大，可以满足不同微处理器 ADC 口对输入阻抗的要求。

3.2. STC15W4K32S4 系列单片机 ADC 外设介绍

关于 AD 和 ADC 的区别：一般说 ADC 指的就是模数转换器，不会有歧义。而说到 AD，除了可能表示模数转换外（少数文档使用），还用来表示单片机访问外部扩展存储器的地址总线和数据总线合在一起的简称。举个例子，STC15W4K32S4 系列单片机的 P0 口有复用功能是 AD0~AD7，AD0~AD7 除了表示访问外部扩展存储器的地址总线 A0~A7 外，还可以表示访问外部扩展存储器的地址总线低 8 位 D0~D7。关键要看下手册描述，注意推敲语境，一般在 STC 的手册中 AD 指的就是地址总线和数据总线合在一起的简称，ADC 指的才是模数转换器，所以看芯片原理图时 AD0 和 ADC0 是完全不同的含义。

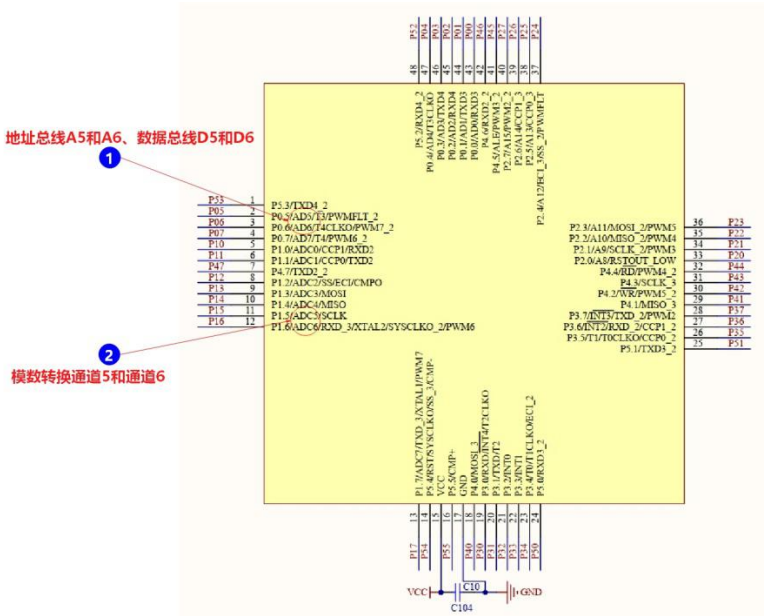


图 2：STC15W4K32S4 系列单片机引脚定义图

✧ 注：STC15W4K32S4 系列单片机各个引脚的功能都有表示，可以看到 AD0~AD7，还有 ADC0~ADC7，含义完全不同，须知。

STC15W4K32S4 系列单片机集成的是 8 路 10 位逐次逼近型 ADC，速度可达 30 万次/秒。该系列单片机没有专用的 ADC 电源脚和电压基准引脚，其参考电压源即是芯片供电引脚 VCC。下表介绍下 ADC 通道关联的引脚分布情况。

表 1：STC15W4K32S4 系列单片机 ADC 引脚分配

序号	ADC 通道	对应 IO 口	功能描述	备注
1	ADC0	P1.0	ADC 通道 0	单片机同一时刻只能选择一个 ADC 通道用于模拟量采集。多个通道采集多路模拟量信号时需分时复用。
2	ADC1	P1.1	ADC 通道 1	
3	ADC2	P1.2	ADC 通道 2	
4	ADC3	P1.3	ADC 通道 3	
5	ADC4	P1.4	ADC 通道 4	
6	ADC5	P1.5	ADC 通道 5	
7	ADC6	P1.6	ADC 通道 6	
8	ADC7	P1.7	ADC 通道 7	

✧ 注：STC15W4K32S4 系列单片机每一路 ADC 都有且只有一个 IO 端口供选择使用。

■ STC15W4K32S4 系列单片机 ADC 内部结构图

STC15W4K32S4 系列单片机 ADC 由多路选择开关、比较器、逐次比较寄存器、10 位 DAC、转换结果寄存器以及控制寄存器构成。

STC15W4K32S4 系列单片机 ADC 是逐次比较型 ADC，逐次比较型 ADC 由一个比较器和 DA 转换器构成，通过逐次比较逻辑，从最高位开始，依次对每一输入电压与内置 DA 转换器输出进行比较，经过多次比较，使转换所得的数字量逐次逼近输入模拟量对应值。逐次比较型 AD 转换具有速度高、功耗低的优点。

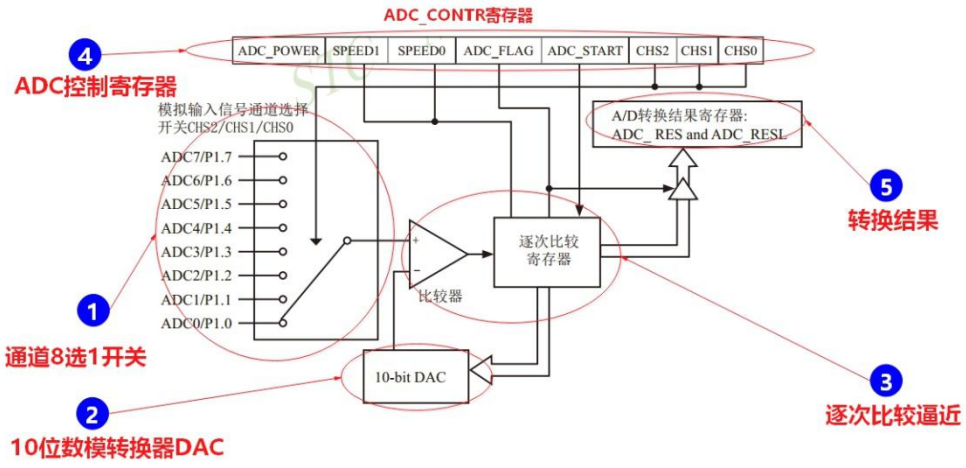


图 3：STC15W4K32S4 系列单片机 ADC 内部结构图

✧ 注：从上图 ADC 内部结构图可知 DA 转换器是 10 位的，所以了解了 ADC 逐次比较的原理后不难得出这样的结论：正是这个内部的 10 位 DA 转换器精度决定了 ADC 精度也是 10 位的。

ADC 内部结构图的多路选择开关是由 ADC 控制寄存器 ADC_CONTR 的 B0~B2 位控制，该寄存器 B3 位和 B7 位分别控制 ADC 转换启动和 ADC 电源，该寄存器 B5~B6 位控制 ADC 转换速度，该寄存器 B4 位是转换结束标志位（需软件清零），如下图所示。

ADC_CONTR：ADC控制寄存器

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
ADC_CONTR	BCH	name	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0

1 寄存器名

对ADC_CONTR寄存器进行操作，建议直接用MOV赋值语句，不要用‘与’和‘或’语句。

2 ADC电源控制位

ADC_POWER：ADC 电源控制位。
0：关闭ADC 电源；
1：打开A/D转换器电源.

3 ADC转换速度控制位

SPEED1，SPEED0：模数转换器转换速度控制位

SPEED1	SPEED0	A/D转换所需时间
1	1	90个时钟周期转换一次，CPU工作频率27MHz时，A/D转换速度约300KHz（=27MHz÷90）
1	0	180个时钟周期转换一次
0	1	360个时钟周期转换一次
0	0	540个时钟周期转换一次

ADC转换结束标志位

4 ADC_FLAG：模数转换器转换结束标志位，当A/D转换完成后，ADC_FLAG = 1，要由软件清0。

5 ADC转换启动位

ADC_START：模数转换器(ADC)转换启动控制位，设置为“1”时，开始转换，转换结束后为0。

CHS2/CHS1/CHS0：模拟输入通道选择，CHS2/CHS1/CHS0

6 ADC通道选择（8选择1）

CHS2	CHS1	CHS0	Analog Channel Select（模拟输入通道选择）
0	0	0	选择 P1.0 作为A/D输入来用
0	0	1	选择 P1.1 作为A/D输入来用
0	1	0	选择 P1.2 作为A/D输入来用
0	1	1	选择 P1.3 作为A/D输入来用
1	0	0	选择 P1.4 作为A/D输入来用
1	0	1	选择 P1.5 作为A/D输入来用
1	1	0	选择 P1.6 作为A/D输入来用
1	1	1	选择 P1.7 作为A/D输入来用

图 4：ADC 控制寄存器 ADC_CONTR

3.3. ADC 配置步骤

针对 STC15W4K32S4 系列单片机 ADC，软件的配置过程如下：

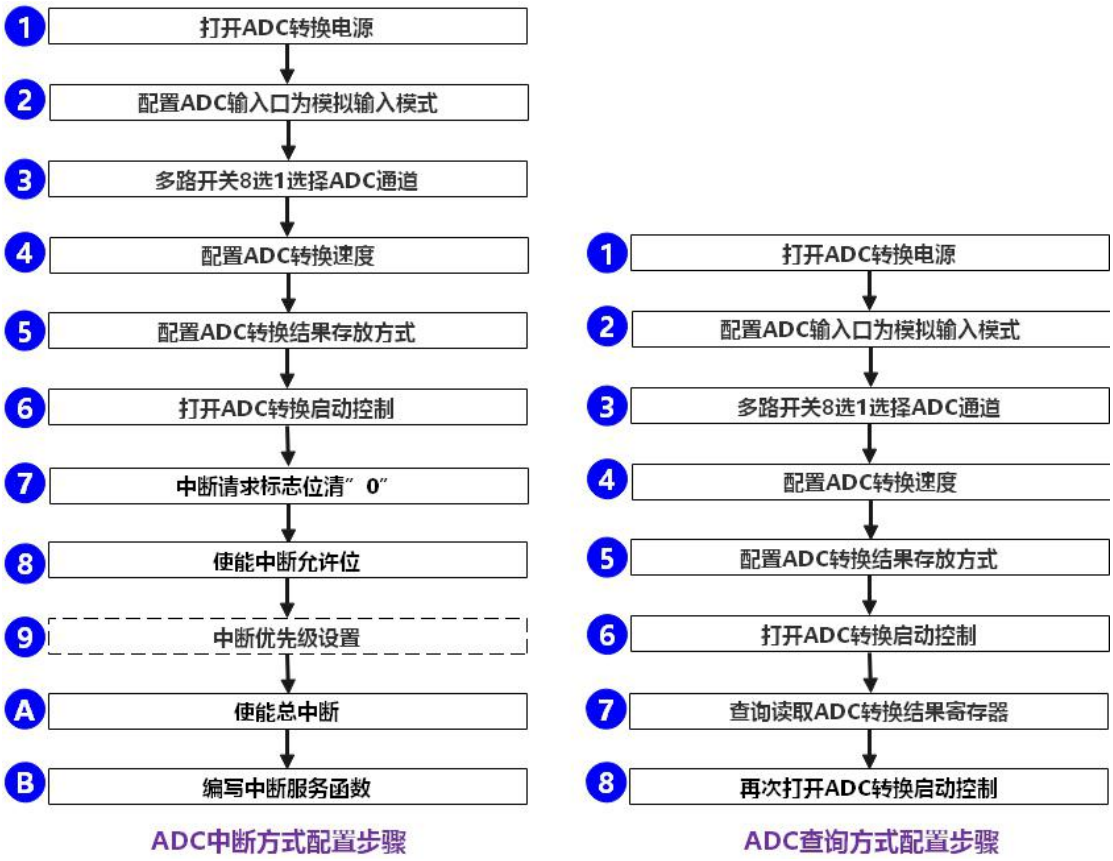


图 5：ADC 中断方式和查询方式软件配置步骤

✧ 注：ADC 中断方式对 ADC 转换结果的读取一般是在中断服务函数中进行，艾克姆提供的例程是 ADC 查询方式。

4. 软件设计

4.1. ADC 寄存器汇集

STC15W4K32S4 系列单片机进行 ADC 操作时会用到 7 个寄存器，如下表所示：

表 2：STC15W4K32S4 系列 ADC 操作使用寄存器汇总

序号	寄存器名	读/写	功能描述
1	IE	读/写	中断允许寄存器。
2	IP	读/写	中断优先级控制寄存器。
3	ADC_CONTR	读/写	ADC 控制寄存器。
4	P1ASF	只写	P1 口模拟功能控制寄存器。
5	CLK_DIV（或 PCON2）	读/写	时钟分频寄存器。
6	ADC_RES	读/写	ADC 转换结果寄存器。
7	ADC_RESL	读/写	

4.2. 寄存器解析

4.2.1. P1 口模拟功能控制寄存器 P1ASF

P1 口模拟功能控制寄存器 P1ASF 用于配置 P1 口的模式是否为 ADC 模拟输入模式，某种意义上可以理解为这种模式是独立于单片机 IO 口 4 种工作模式之外的一种模式。该寄存器的 8 位分别配置 P1.0~P1.7 口是否为 ADC 模拟输入模式。

图 6：P1 口模拟功能控制寄存器 P1ASF

4.2.2. 时钟分频寄存器 CLK_DIV

时钟分频寄存器 CLK_DIV 与单片机 ADC 转换相关的位是 ADRJ 位，该位控制 ADC 转换结果如何存放到 ADC 转换结果寄存器中。

图 7：时钟分频寄存器

✧ 注：时钟分频寄存器 CLK_DIV 中的每个位作用于单片机不同的功能外设，故在 ADC 程序设计时一定按位操作该寄存器，以免误操作其他位导致不可预知故障。

4.3. ADC 光敏电阻检测（读取原始值）

✧ 注：本节的实验源码是在“实验 2-8-1：串口 1 收发实验（P3.0 和 P3.1）”的基础上修改。本节对应的实验源码是：“实验 2-12-1：ADC 光敏电阻检测（读取原始值）”。

4.3.1. 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	delay	. c	包含用户自定义延时函数。

4.3.2. 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"
```

■ 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 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。

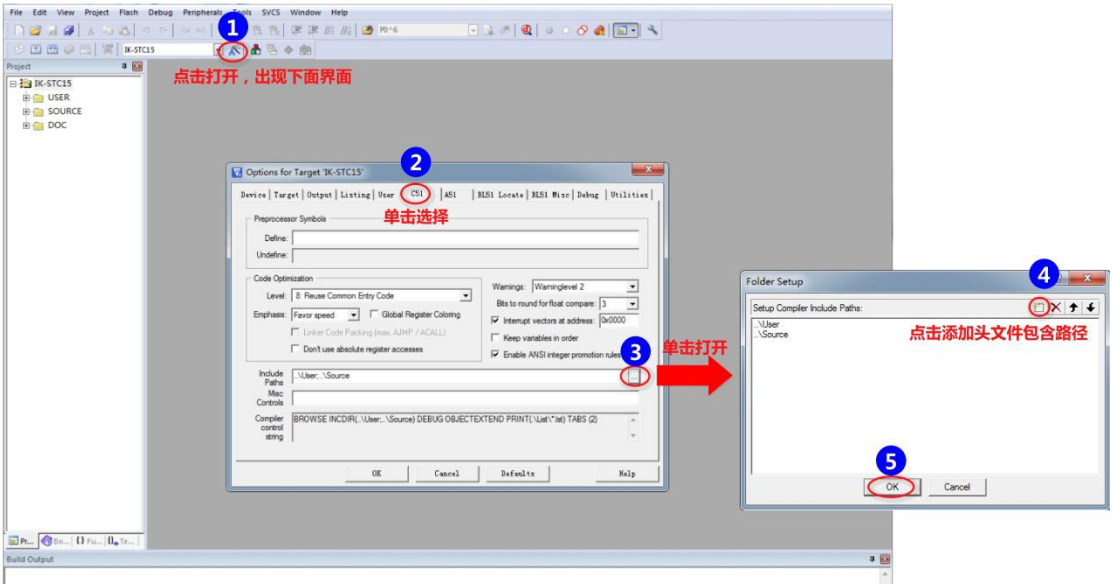


图 8：添加头文件包含路径

4.3.3. 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 基本操作函数汇集

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。

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

```
/*************************************************************************
功能描述：ADC 口初始化
入口参数：无
返回值：无
*************************************************************************/
void    ADC_config(void)
{
    ADC_CONTR|=0x80;          //开 AD 转换电源
    delay_ms(10);             //适当延时等待 AD 转换供电稳定
    P1ASF|=0x80;              //选择 P1.7 作为模拟功能 AD 使用
    ADC_CONTR|=0x07;          //选择 P1.7 作为 AD 转换通道输入使用
    ADC_CONTR|=0x60;          //AD 转换速度为 90 个时钟周期转换一次
    ADC_CONTR&=0xEF;          //清 AD 转换完成标志
    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;
}
```

然后，对光敏电阻引脚 ADC 原始值的处理方式是直接返回该原始值，处理函数代码如下。

程序清单： ADC 转换原始值处理函数

```
/*********************************************************************
功能描述：读取 ADC 采集的原始值
入口参数：无
返回值：实时原始值
 *********************************************************************/
uint16  HandleADC(void)
{
    uint16 Temp_signal;

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

    //返回采集的原始值
    return Temp_signal;
}
```

最后，在主函数中对串口 1 进行初始化，主循环中每 1000ms 通过串口 1 发送读取的 ADC 原始值。

代码清单：主函数

```
int main()
{
////////////////////////////////////////////////
//注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
//       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
//相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
//         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
////////////////////////////////////////////////
    P1M1 &= 0x7F;   P1M0 &= 0x7F;                   //设置 P1.7 为准双向口
    P3M1 &= 0xFC;   P3M0 &= 0xFC;                   //设置 P3.0~P3.1 为准双向口

    ADC_config();                                   //ADC 初始化
    Uart1_Init();                                   //串口 1 初始化
    EA = 1;                                         //使能总中断
    delay_ms(10);                                   //初始化后延时

  while (1)
    {
        printf("\r\n ADC 采集原始值：%d\r\n",HandleADC());      //串口打印上传的 ADC 采集原始值信息
```

```
20.          delay_ms(1000);
21.      }
22. }
```

4.3.4. 硬件连接

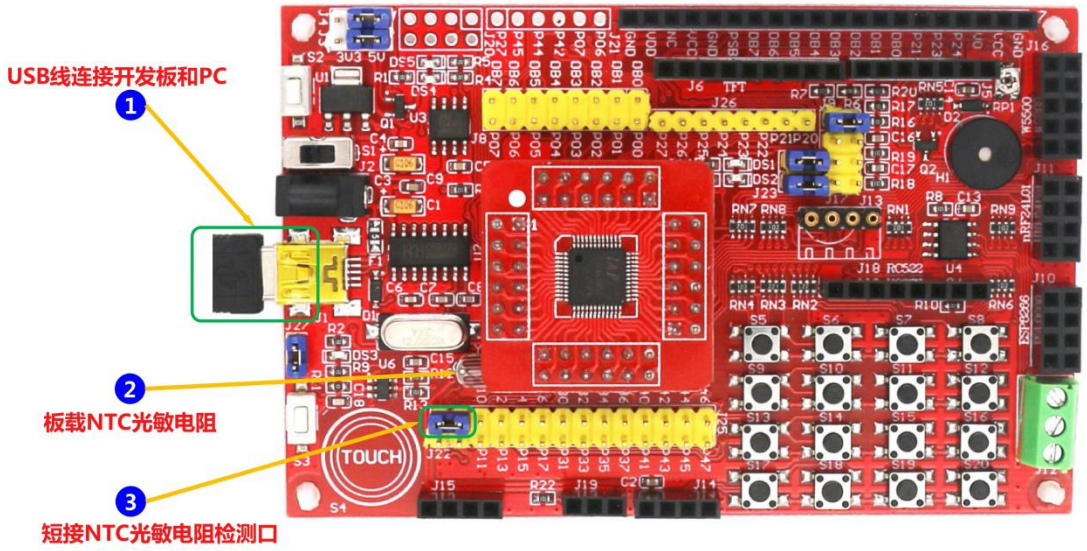


图 9：开发板连接图

4.3.5. 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 2-12-1：ADC 光敏电阻检测（读取原始值）”，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“…\ADC\project”目录下的工程“ADC.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“ADC.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 采集原始值： xxxx”内容。（xxxx 是实际的数字）

4.4. ADC 光敏电阻检测（电压值）

✧ 注：本节的实验源码是在“实验 2-12-1：ADC 光敏电阻检测（读取原始值）”的基础上修改。本节对应的实验源码是：“实验 2-12-2：ADC 光敏电阻检测（电压值）”。

4.4.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-12-1：ADC 光敏电阻检测（读取原始值）”部分。

4.4.2. 编写代码

在 adc.c 文件中编写操作 ADC 外设会用到的基本函数，请参考“实验 2-12-1：ADC 光敏电阻检测（读取原始值）”部分。

然后，对光敏电阻引脚 ADC 原始值的处理方式是根据返回的原始值来计算出对应 ADC 口的电压值，处理函数代码如下。

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。
    if((Temp_signal<TEMPMAX)&&(Temp_signal>TEMPMIN))
    {
    g_voltage=(3.3*Temp_signal)/1024;
    }

  //返回实测电压值
    return g_voltage;
}
```

最后，在主函数中对串口 1 进行初始化，主循环中每 1000ms 通过串口 1 发送读取的电压值。

代码清单：主函数

```
1.  int main()
2.  {
3.  ////////////////////////////////////////////////
4.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
5.  //       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
6.  //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
7.  //         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
8.  ////////////////////////////////////////////////
9.      P1M1 &= 0x7F;   P1M0 &= 0x7F;                    //设置 P1.7 为准双向口
10.     P3M1 &= 0xFC;   P3M0 &= 0xFC;                    //设置 P3.0~P3.1 为准双向口
11.
12.     ADC_config();                                    //ADC 初始化
13.     Uart1_Init();                                    //串口 1 初始化
14.     EA = 1;                                          //使能总中断
15.     delay_ms(10);                                    //初始化后延时
16.
17.   while (1)
18.     {
19.         printf("\r\n ADC 端口电压值:  %.1f\r\n",HandleADC());      //串口打印上传的
    电压值
20.         delay_ms(1000);
21.     }
22. }
```

4.4.3. 硬件连接

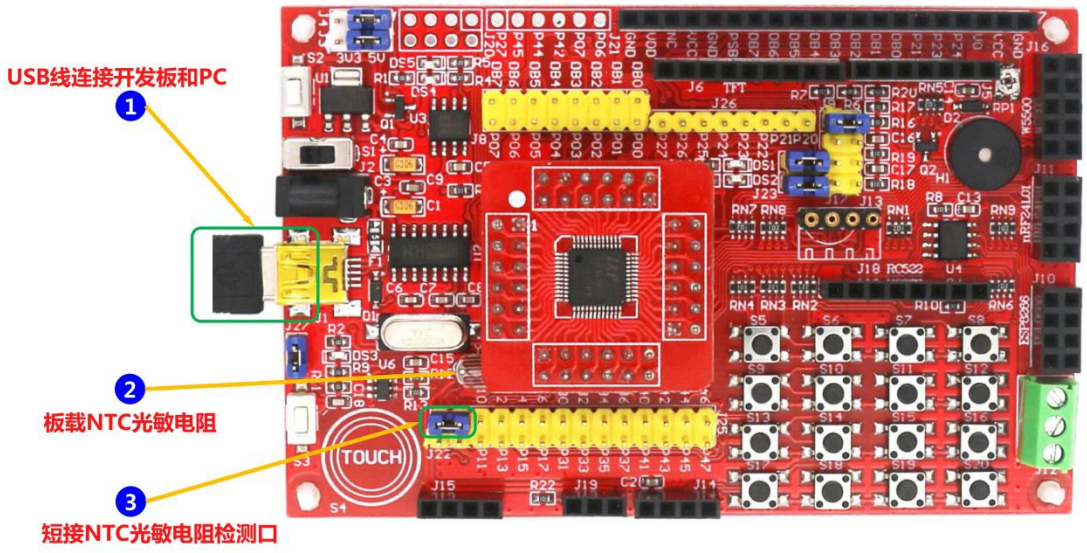


图 10：开发板连接图

4.4.4. 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 2-12-2：ADC 光敏电阻检测（电压值）”，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“…\ADC\project”目录下的工程“ADC.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“ADC.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 端口电压值： xxxx”内容。（xxxx 是实际的数字，带一位小数点）

4.5. ADC 光敏电阻检测（光强度）

✧ 注：本节的实验源码是在“实验 2-12-1：ADC 光敏电阻检测（读取原始值）”的基础上修改。本节对应的实验源码是：“实验 2-12-3：ADC 光敏电阻检测（光强度）”。

4.5.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-12-1：ADC 光敏电阻检测（读取原始值）”部分。

4.5.2. 编写代码

在 adc.c 文件中编写操作 ADC 外设会用到的基本函数，请参考“实验 2-12-1：ADC 光敏电阻检测（读取原始值）”部分。

然后，对光敏电阻引脚 ADC 原始值的处理方式是根据返回的原始值来计算出对应 ADC 口的光强度值，处理函数代码如下。

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光强度值
入口参数：无
返回值：根据已有公式计算的光强度值
 *********************************************************************************/
float HandleADC(void)
{
    uint16 Temp_signal;
    float g_voltage;

```

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

    //下面是通过曲线拟合得出的光强度与采集的原始值之间的关系，各系数是用户自己通过校准测试得出的（仅供参考）
    if((Temp_signal<TEMPMAX)&&(Temp_signal>TEMPMIN))
    {
    g_voltage=(6.84E-07)*pow(Temp_signal,3)+(3.794E-05)*pow(Temp_signal,2)+(9.743E-02)*Temp_signal+(3.604E+01);
    }

  //返回光强度值
    return g_voltage;
}
```

最后，在主函数中对串口 1 进行初始化，主循环中每 1000ms 通过串口 1 发送读取的 ADC 光强度值。

代码清单：主函数

```
int main()
{
//////////////////////////////////////////////
//注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
//       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
//相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
//         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
//////////////////////////////////////////////
    P1M1 &= 0x7F;   P1M0 &= 0x7F;                   //设置 P1.7 为准双向口
    P3M1 &= 0xFC;   P3M0 &= 0xFC;                   //设置 P3.0~P3.1 为准双向口

    ADC_config();                                   //ADC 初始化
    Uart1_Init();                                   //串口 1 初始化
    EA = 1;                                         //使能总中断
    delay_ms(10);                                   //初始化后延时

  while (1)
    {
        printf("\r\n NTC 光敏测光强度:  %.1f\r\n",HandleADC());      //串口打印上传的光强度值
        delay_ms(1000);
    }
}
```

4.5.3. 硬件连接

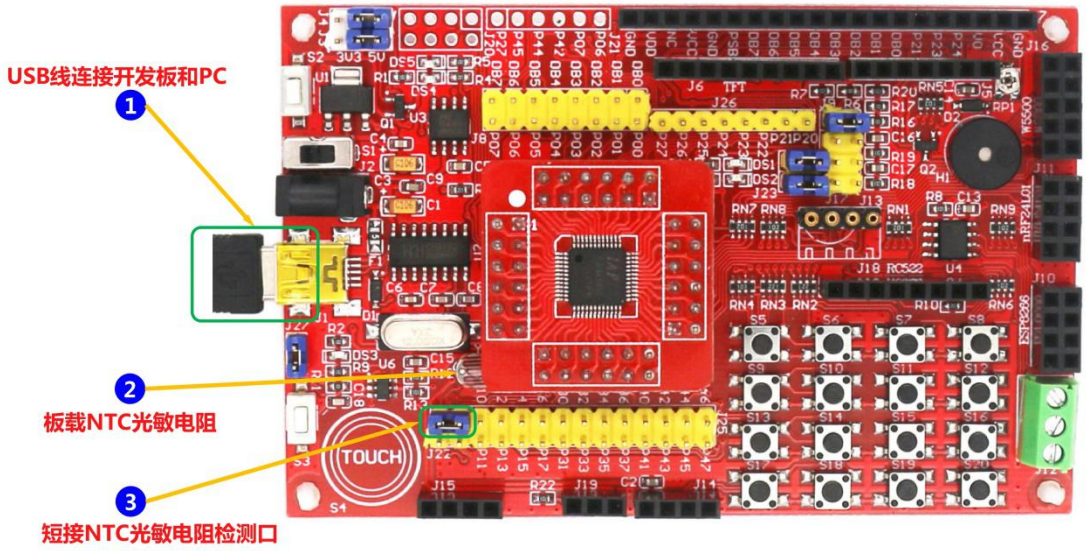


图 11：开发板连接图

4.5.4. 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 2-12-3：ADC 光敏电阻检测（光强度）”，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“…\ADC\project”目录下的工程“ADC.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“ADC.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“NTC 光敏测光强度值： xxxx”内容。（xxxx 是实际的数字，带一位小数点）