更多嵌入式 Linux 学习资料，请关注：一口 Linux 回复关键字:1024

ADC 概述

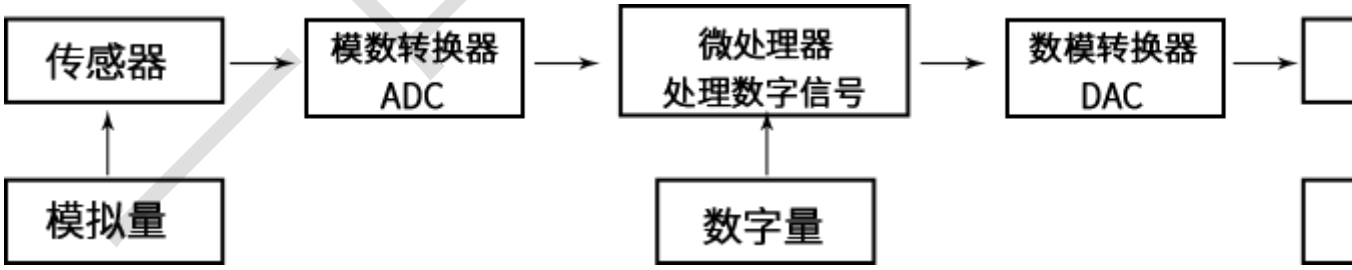
将时间上连续变化的模拟量转化为脉冲有无的数字量，这一过程就叫做数字化，实现数字化的关键设备是 ADC。

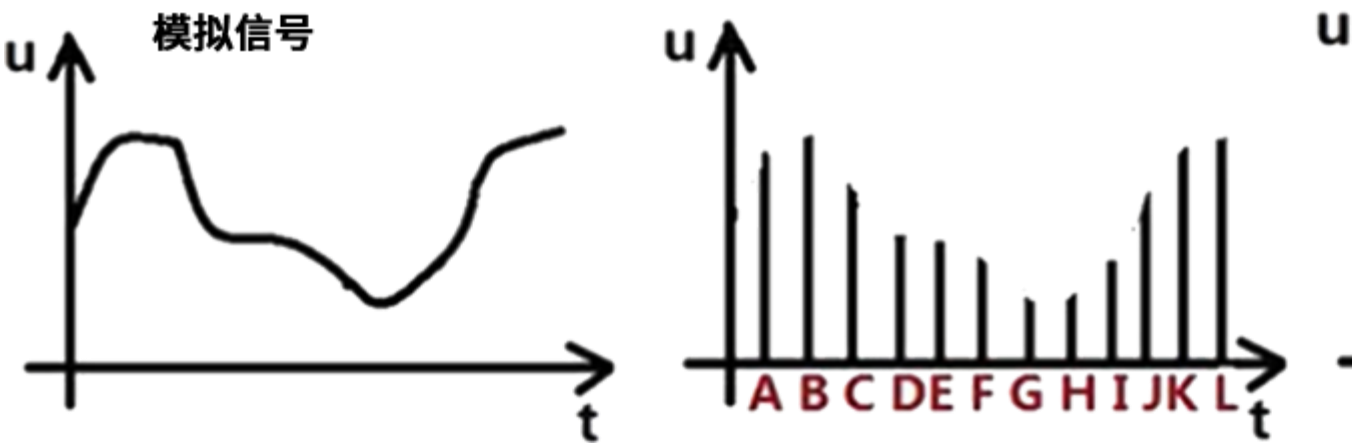
数字化： 将时间上连续变化的模拟量转化为脉冲有无的数字量（ADC）

ADC： 数模转换器，将时间和幅值连续的模拟量转化为时间和幅值离散的数字量

过程： 采样、保持、量化、编码

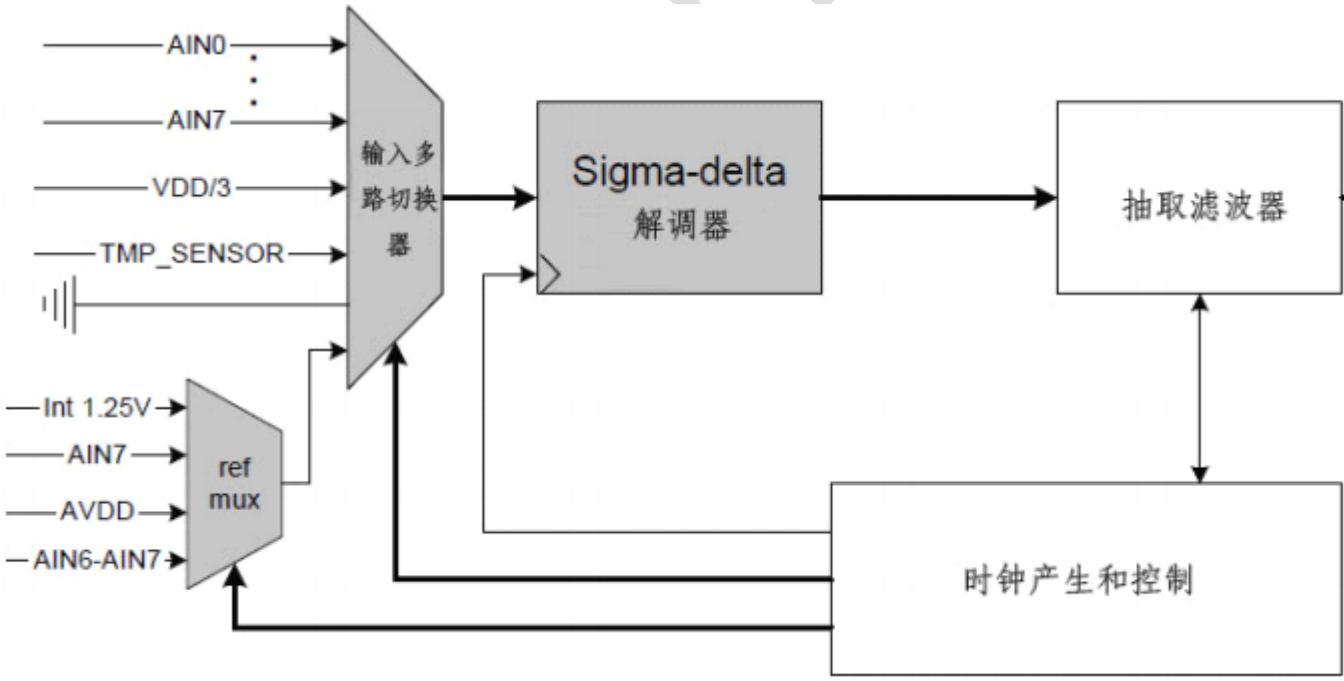
1. A/D 转换的基本工作原理





2. CC2530 的 A/D 转换模块

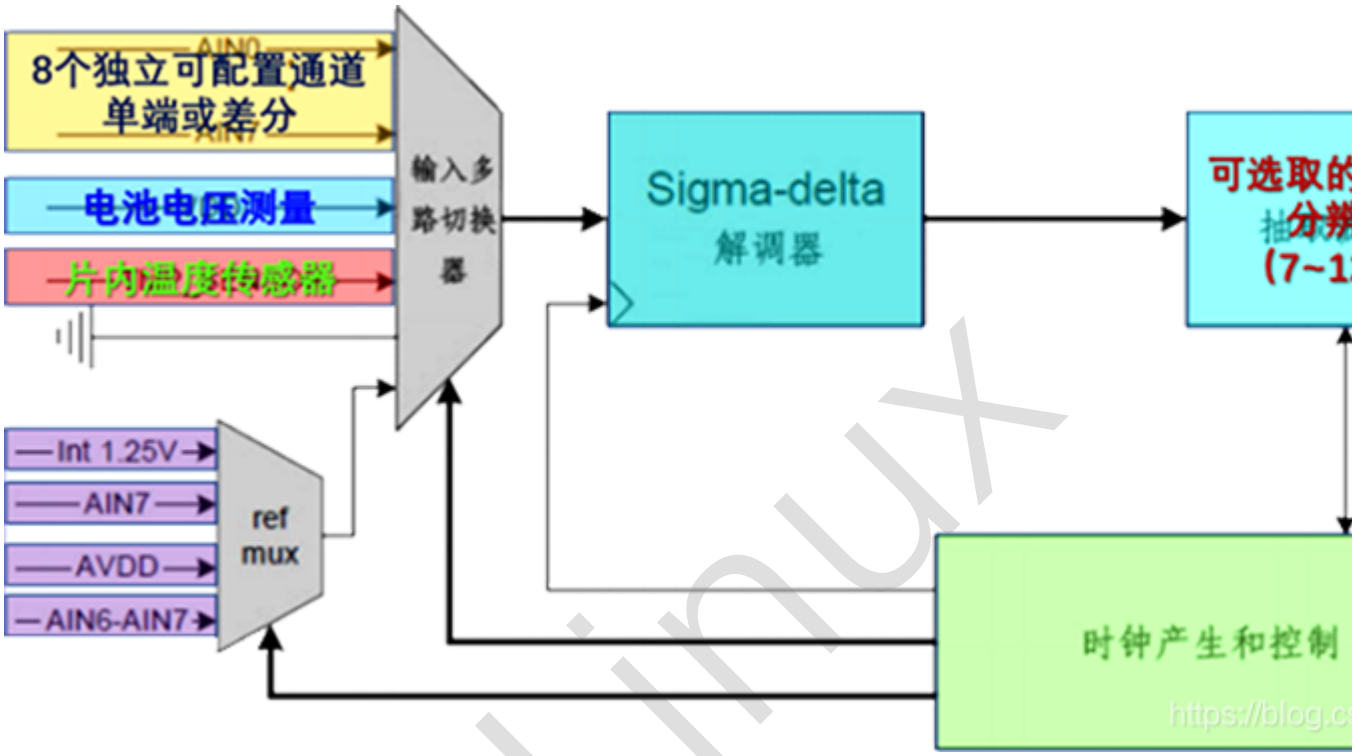
CC2530 的 ADC 模块支持最高 14 位二进制的模拟数字转换，具有 12 位的有效数据位，它包括一个模拟多路转换器，具有 8 个各自可配置的通道，以及一个参考电压发生器。



此 ADC 模块的主要特征：

1. 可选取的抽取率，设置分辨率（7~12 位）
2. 8 个独立的输入通道，可接收单端或差分信号
3. 参考电压可选为内部单端、外部单端、外部差分或 AVDD5
4. 单通道转换结束可产生中断请求

5. 序列转换结束可发出 DMA 触发
6. 可将片内温度传感器作为输入
7. 电池电压测量功能



ADC 模块的信号输入

1. 输入端可配置为单端输入或差分输入
2. 差分输入对： AIN0~AIN1、AIN2 ~ AIN3、AIN4 ~ AIN5 、AIN6 ~ AIN7
3. 片上温度传感器的输出也可以作为 ADC 的输入用于测量芯片的温度
4. **可以将一个对应 AVDD5/3 的电压作为 ADC 的输入，实现电池电压检测**
5. 负电压和大于 VDD 的电压都不能用于 P0 这些引脚
6. 单端电压输入 AIN0~ AIN7，以通道号码 0~7 表示
7. 四个差分输入对则以通道号码 8~11 表示
8. 温度传感器的通道号码为 14
9. AVDD5/3 电压输入的通道号码为 15

实验 1 将 1/3 电压值通过 ADC 转换为数字信号，再将其通过串口发送到 PC

我们设置 ADC 的源电压社为 1/3 电源电压（电源电压为 3.3V），然后通过 CC2530 的 ADC 功能将获得模拟电压值通过相应公式转换为数字信号，再由串口将数字信号打印到 PC 终端上。

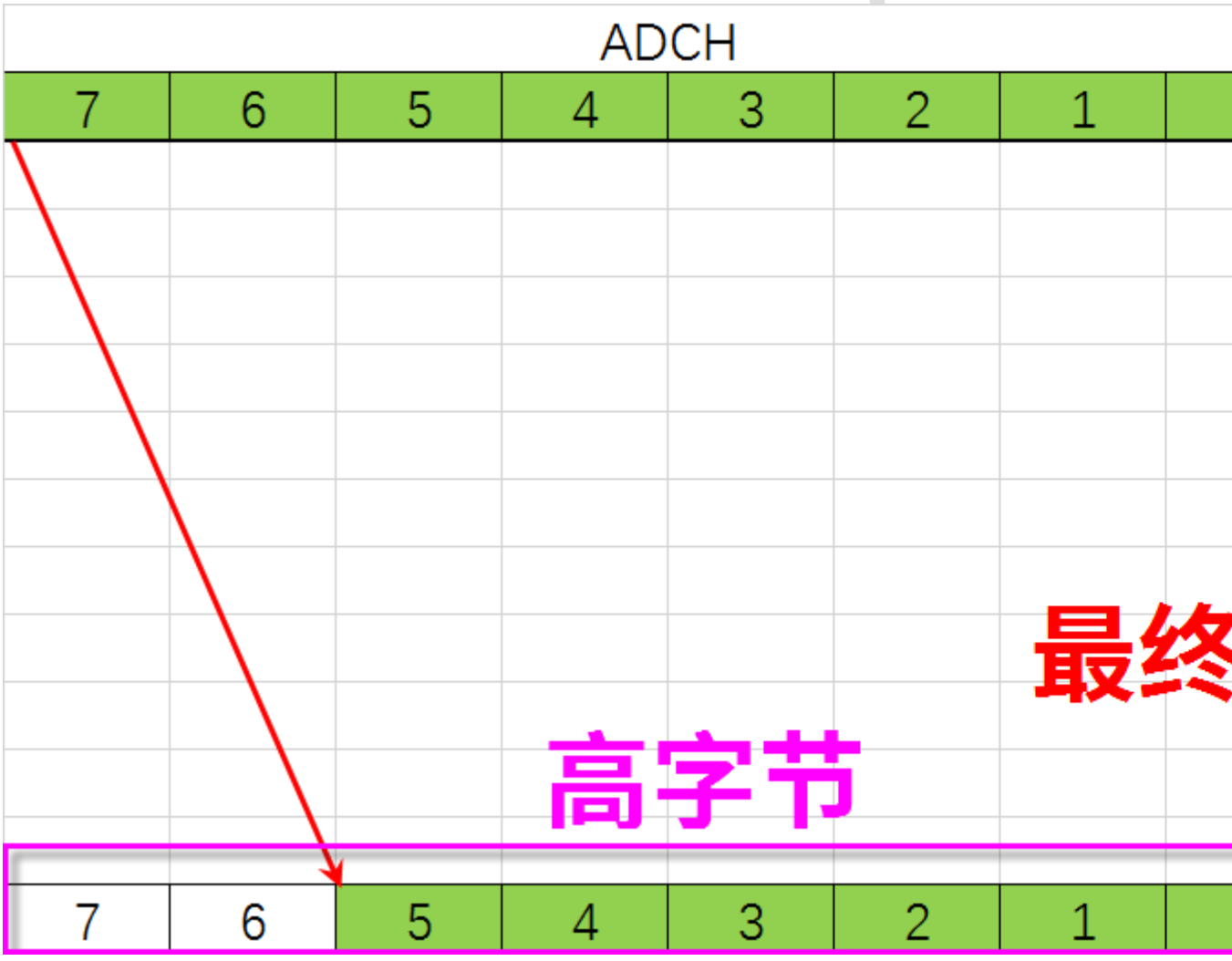
寄存器

ADCL (0xBA) - ADC 数据低位

位	名称	复
7:2	ADC[5:0]	000
1:0	-	00

ADCH (0xBB) - ADC 数据高位

位	名称	复
7:0	ADC[13:6]	0x00



ADCCON1 (0xB4) - ADC 控制 1

位	名称	复
7	EOC	0
6	ST	0
5:4	STSEL[1:0]	11

ADCCON3 (0xB6) - ADC 控制 3

位	名称	复位	R/W
7:6	EREF[1:0]	00	R/W
5:4	EDIV[1:0]	00	R/W
3:0	ECH[3:0]	0000	R/W

核心代码分析：

- (1) ADC 初始化函数函数原型：

```
void InitialAD(void)
{
 P1DIR = 0x03; //P1 控制 LED
 led1 = 1;
 led2 = 1; //关 LED
 ADCH &= 0X00; //清 EOC 标志
 ADCCON3=0xbf; //单次转换,参考电压为电源电压，对 1/3 VDD 进行 A/D 转换
 //14 位分辨率
 ADCCON1 = 0X30; //停止 A/D
 ADCCON1 |= 0X40; //启动 A/D
}
```

- 设置与 LED 相关的 I/O， P1_0 和 P_1，将其方向设为输出，接下来关闭两个亮灯
- 将 0x00 写入 ADCH 寄存器，清 EOC 标志，由数据手册 ADCCON1 寄存器可知，当 ADCH，被读取的时候， ADCCON1.EOC 将被清除
- 将 0xbf 写入寄存器 ADDCON3，设置 ADC 为单次采样、 14 位的分辨率，参考电压为电源电压，对 1/3VDD 进行 AD 转换
- 将 0x30 写入 ADDCON1，停止 A/D 转换
- 将 0x40 写入 ADDCON1，启动 A/D 转换
- (2)主函数代码分析
- (a)获取采样值，将 ADCL， ADCH 分别写入数组 temp 中

```
temp[1] = ADCL;
temp[0] = ADCH;
```

- (b)从 ADCH 和 ADCL 中获取有效的数据

```
temp[1] = temp[1]>>2;
temp[1] |= temp[0]<<6;
temp[0] = temp[0]>>2; //数据处理
temp[0] &= 0x3f;
```

因为我们要获取 ADCL[0-5],所以右移两位将无效的两位移除，在左移即可获得 ADCL[0-5]。 同理获得 ADCH 有效位。

- (c)将 ADC 转换得到的数值进行处理，存入数组 adcdata 中

```
num = (temp[0]*256+temp[1])*3.3/8192;
adcdata[1] = (char)(num)%10+48;
```

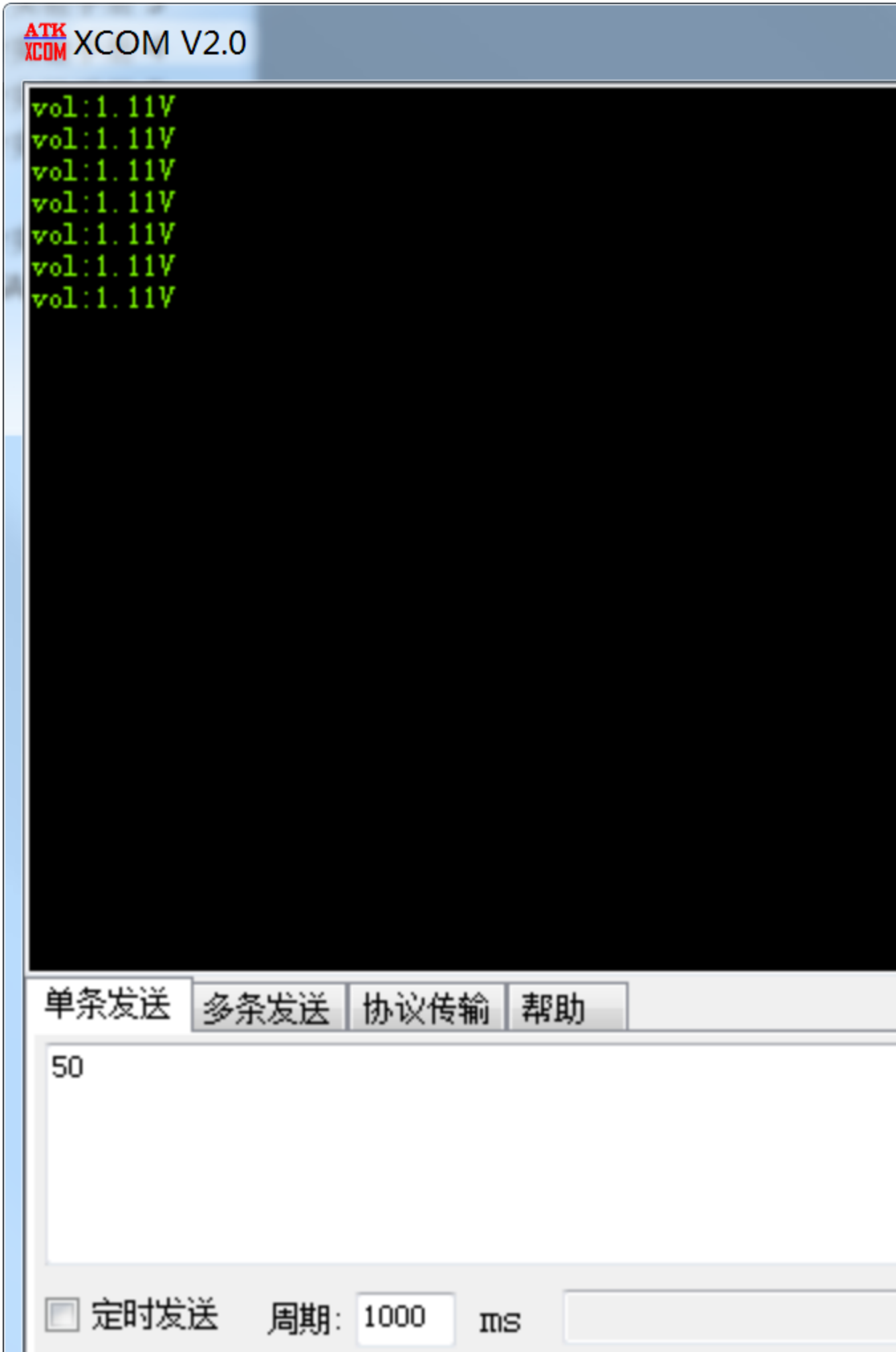
```
adcdata[3] = (char)(num*10)%10+48;
```

adcdata[3] = (char)(num*10)%10+48;

- (d)将数据通过串口发送出去

```
UartTX_Send_String(adcdata,6);
```

实验现象



完整代码：

```
/*********************************************************************
* 文 件 名: main.c
* 作    者: Daniel Peng
* 修    订: 2022-4-10
* 版    本: 1.0
* 描    述: 将 1/3 电压值通过 ADC 转换为数字信号，再将其通过串口发送到 PC
*********************************************************************/
#include <stdio.h>
#include <string.h>
#include "UartTimer.h"


//定义控制灯的端口
#define led1 P1_0
#define led2 P1_1
/*********************************************************************
* 名    称: InitSensor()
* 功    能: AD 初始化函数
* 入口参数: 无
* 出口参数: 无
*********************************************************************/
void InitialAD(void)
{
 P1DIR = 0x03; //P1 控制 LED
 led1 = 1;
 led2 = 1; //关 LED

 ADCH &= 0X00; //清 EOC 标志
 ADCCON3=0xbf; //单次转换,参考电压为电源电压，对 1/3 VDD 进行 A/D 转换
 //14 位分辨率
 ADCCON1 = 0X30; //停止 A/D
 ADCCON1 |= 0X40; //启动 A/D
}
/*********************************************************************
* 程序入口函数
*********************************************************************/
void main(void)
{
 char temp[2];
 float num;
    char strTemp[12]={0};

 DISABLE_ALL_INTERRUPTS();      //关闭所有中断
```

```
InitClock();                    //设置系统主时钟为 32M
    InitUART();                                 //初始化串口
    InitialAD();

led1 = 1;
    while(1)
    {
      if(ADCCON1>=0x80)
      {
  led1 = 0; //转换完毕指示

        temp[1] = ADCL;
        temp[0] = ADCH;
        ADCCON1 |= 0x40; //开始下一转换

        temp[1] = temp[1]>>2;
        temp[1] |= temp[0]<<6;
        temp[0] = temp[0]>>2; //数据处理
        temp[0] &= 0x3f;

        num = (float)(temp[0]*256+temp[1])*3.3/8192; //有一位符号位,取 2^13;
        //定参考电压为 3.3V。 14 位精确度

  sprintf(strTemp,"vol:%.02fV", num);//将浮点数转成字符串
        UartSendString(strTemp,12); //串口送数
        //包括空格
        DelayMS(1000);
  led1 = 1; //完成数据处理
  DelayMS(1000);
      }
    }
}
```