深 圳 大 学 实 验 报 告

课程名称：微型计算机技术

实验项目名称：片内 A/D 的使用

学院：医学院

专业：生物医学工程

指导教师：徐海华、刘昕宇

报告人：陈焕鑫 学号：2016222042 班级：生工 2 班

实验时间：2018-12-18

实验报告提交时间：2018-12-25

教务处制

一、实验目的

1、 片内 A/D 是单片机中最重要的功能之一，大部分数据都是靠 A/D 采集；

2、掌握如何设置 A/D 端口的寄存器等内容；

3、掌握如何将采集的数据进行处理；

二、实验仪器

微机原理实验板

三、实验内容

1、编写程序，实现以下功能

（1）首先设置串口 1，使用独立波特率作为串口 1 的波特率发生器，并且将波特率设置为 9600 bps。

（2）将 P1.1 设为 ADC1 功能，采集 P1.1 上的电压，将 8 位 AD 值通过串口 1 发送给 PC 上位机。每两次采集之间加入大概 50ms 的延时。

（3）运行 STC-ISP 软件，接收 AD 数据，同时转动实验箱上的旋钮，应该可以看见软件上接收缓冲区的数据会跟着旋钮的转动响应的变化。

（4）将 AD 数据显示实时显示到实验平台上的数码管上。

四、实验原理

1、STC 片内 A/D

STC 单片机带有片内 A/D 转换器，使用方法参照芯片手册第 9 章，设置步骤如下：

1） 设置 P1ASF 寄存器，将 P1.1 引脚设置为 A/D 输入功能引脚

2） 设置 ADC_CONTR 寄存器：打开 A/D 电源；将速度设置为 90 个时钟周期转换一次；设置模拟输入通道；

3） 读取 A/D 转换结果，STC 是 10 位的 A/D，我们只取其中的 8 位，也就是只读取 ADC_RES 中的数据，如下图所示。

当AUXR.1/ADRJ = 0时，A/D转换结果寄存器格式如下：

ADC_RES[7:0]

ADC_B9	ADC_B8	ADC_B7	ADC_B6	ADC_B5	ADC_B4	ADC_B3	ADC_B2

-	-	-	-	-	-	ADC_B1	ADC_B0

ADC_RESL[1:0]

2、独立波特率发生器

传统 51 单片机（包括我们做的实验 9），使用串行口需要占用一个定时器资源，对于很多应用场合，定时器的资源本来就非常紧张，所以 STC 单片机增加了一个独立波特率发生器，专门作为串行口的波特率发生器。

使用独立波特率发生器，需要对 AUXR 寄存器以及 BRT 寄存器。

具体设置方法，可参考数据手册第 8 章。

五、实验方法及步骤

本实验中用到了 AD、串口还有八段数码管模块，分别对应了 ADC、Uart 和 Seg8LED_Disp 三个文件对，在 ADC.h 声明了 InitADC 函数，.c 文件中对 ADC 相关的寄存器 P1ASF, ADC_RES, ADC_RESL, AUXR1, 还有最关键的 ADC_CONTR 进行了设置，在其中断服务函数中，首先判断 ADC 转换完成标志位是否为 1，如果为 1，则表明单片机完成了 AD 转换，将转换后的值存放在 result 中，将 RES 的值通过 SBUF 发送，然后重新配置 ADC_CONTR，getRst 函数可以把 result 的值返回出去。在使用串口的时候，我们使用的是独立波特率发生器，设置独立波特率发生器的寄存器是 AUXR 和 BRT，其中，BRT 寄存器用来存放自动重装载值。

在 main.c 文件中，调用函数初始化了串口和 ADC 之后，在 while 循环里面不断地对八段数码管进行扫描。而每次完成了 AD 转换之后，单片机会自动跳转到 AD 的中断服务程序中，完成数值的更新。

具体程序代码如下：

```
//Uart.h
#ifndef _UART_H_
#define _UART_H_

typedef unsigned char BYTE;  //定义 BYTE
typedef unsigned int  WORD;  //定义 WORD

void InitUart();   //初始化 Uart

#endif

//Uart.c
#include "STC12C5A60S2.h"
#include "Uart.h"

#define BRTR_OPEN    0x10
#define S1BRS_BRT_ON 0x01

static void __Uart1_Send_ISR();
static void __Uart1_Recv_ISR();

void InitUart()
{
  SCON = 0x50;         //串口配置寄存器
  AUXR = BRTR_OPEN | S1BRS_BRT_ON; //独立波特率发生器寄存器
  BRT = 0xFD;          //独立波特率发生器的自动重装载值
```

```
  EA = 1;             //开启系统中断
  ES = 1;             //开启串口中断
}

void Uart1_ISR() interrupt 4    //串口中断服务函数
{
  if(TI == 1)   //需要发送
  {
    TI = 0;     //清除标志位
    __Uart1_Send_ISR();
  }
  else if(RI == 1) //需要接收
  {
    RI = 0;     //清除标志位
    __Uart1_Recv_ISR();
  }
}

static void __Uart1_Send_ISR()
{

}

static void __Uart1_Recv_ISR()
{
  BYTE Uart_Tmp;
  Uart_Tmp = SBUF;
  SBUF = Uart_Tmp;
}

//Seg8LED_Disp.h
#ifndef _SEG8LED_DISP_H_
#define _SEG8LED_DISP_H_

void ShowNum(unsigned int num);   //扫描数码管

#endif

//Seg8LED_Disp.c
#include "Seg8LED_Disp.h"
#include "STC12C5A60S2.h"

sbit SM_G1 = P2^0;
sbit SM_G2 = P2^1;
```

```
sbit SM_G3 = P2^2;
sbit SM_G4 = P2^3;

#define PUT_OUT 10

unsigned short LEDNum[11] = {0x03, 0x9F, 0x25, 0x0d, 0x99, 0x49, 0x41,
0x1F, 0x01, 0x09, 0xFF};

static void ShowEachBit(unsigned char LEDbit, unsigned char num);

static void ShowEachBit(unsigned char LEDbit, unsigned char num)
{
  switch(LEDbit)  //四个数码管中的某一位
  {
    case 4:
      SM_G1 = 0;
      SM_G2 = 1;
      SM_G3 = 1;
      SM_G4 = 1;
      break;
    case 3:
      SM_G1 = 1;
      SM_G2 = 0;
      SM_G3 = 1;
      SM_G4 = 1;
      break;
    case 2:
      SM_G1 = 1;
      SM_G2 = 1;
      SM_G3 = 0;
      SM_G4 = 1;
      break;
    case 1:
      SM_G1 = 1;
      SM_G2 = 1;
      SM_G3 = 1;
      SM_G4 = 0;
      break;
    default:
      break;
  }
  if(LEDbit == 3)
  {
    P0 = LEDNum[num] - 1;  //减 1 就可以带上小数点
```

```
  }
  else
  {
    P0 = LEDNum[num];      //不带小数点
  }
}

void ShowNum(unsigned int num)
{
  unsigned long int i;
  unsigned char bit1, bit2, bit3, bit4;

  //取出数字的各个位
  if(num > 9999)
  {
    num = 9999;
  }

  bit4 = num / 1000;
  bit3 = (num % 1000) / 100;
  bit2 = (num % 100) / 10;
  bit1 = num % 10;

  ShowEachBit(4, PUT_OUT);  //第一个数码管固定是熄灭
  for(i = 0; i < 50; i++);
  ShowEachBit(3, bit3);     //第二个数码管固定带小数点
  for(i = 0; i < 50; i++);
  ShowEachBit(2, bit2);
  for(i = 0; i < 50; i++);
  ShowEachBit(1, bit1);
  for(i = 0; i < 50; i++);
}

//ADC.h
#ifndef _ADC_H_
#define _ADC_H_

#include "STC12C5A60S2.h"
#include "Uart.h"

#define ADC_POWER   0x80
#define ADC_FLAG    0x10
#define ADC_START   0x08
#define ADC_SPEEDHH 0x60
```

```
#define ADC_CHANNEL 0x01
#define ADRJ_       0x04

sbit LED = P2^4;

void InitADC();     //初始化 ADC
void Delay(WORD n); //延时 n 个 10ms
WORD getRst();      //获取转换的值

#endif

//ADC.c
#include "ADC.h"

static WORD result = 0;

void InitADC()
{
  P1ASF = 0x02;   //配置 AD 的端口，此处为 P1.1
  ADC_RES = 0;    //初始化存放高 8 位的寄存器
  ADC_RESL = 0;   //初始化存放低 2 位的寄存器
  AUXR1 &= ~ADRJ_;//设置高 8 位存在 RES，低 2 位存放在 RESL
  //ADC 配置寄存器
  ADC_CONTR = ADC_POWER | ADC_SPEEDHH | ADC_START | ADC_CHANNEL;
  //延时 20ms
  Delay(2);
  //打开 AD 中断开关
  EADC = 1;
}

void Delay(WORD n)
{
  WORD i, j, k;

  for(k = 0; k < n; k++)
    for(j = 0; j < 78; j++)
      for(i = 0; i < 25; i++);
}

void adc_isr()interrupt 5
{
  if(ADC_CONTR & ADC_FLAG)
  {
    ADC_RESL &= 0x03;
```

```
    //将结果转换为16位的值
    result = ADC_RES * 4 + ADC_RESL;
    SBUF = ADC_RES;
    ADC_CONTR = ADC_POWER | ADC_SPEEDHH | ADC_START | ADC_CHANNEL;
  }
}

WORD getRst()
{
  return (WORD)(result/1024.0*5.0*100.0);
}


//main.c
#include "Uart.h"
#include "ADC.h"
#include "Seg8LED_Disp.h"

void main()
{
  InitUart();  //初始化串口
  InitADC();   //初始化ADC

  while(1)
  {
    ShowNum(getRst());  //扫描八段数码管
  }
}
```

检查代码无误之后，分别编译、链接、生成 Hex 文件，将 Hex 文件通过串口烧进实验平台中，观察实验现象。

六、 实验现象

烧进程序之后，可以观察到实验现象：

刚开始的时候，旋钮往左拧尽，此时数码管显示的数字是 0.00（数码管上显示的数值时转换成电压之后的值），当我们稍微拧动一下旋钮，发现显示 0.12，如图 1 所示，拧到一半时，如图 2 所示，当向右拧到尽头时，数码管显示 3.28，如图 3 所示。同时，打开 STC-ISP 软件中的串口，可以看见串口缓存区接收到 AD 采集到的数据如图 4 所示。证明程序设计正确。

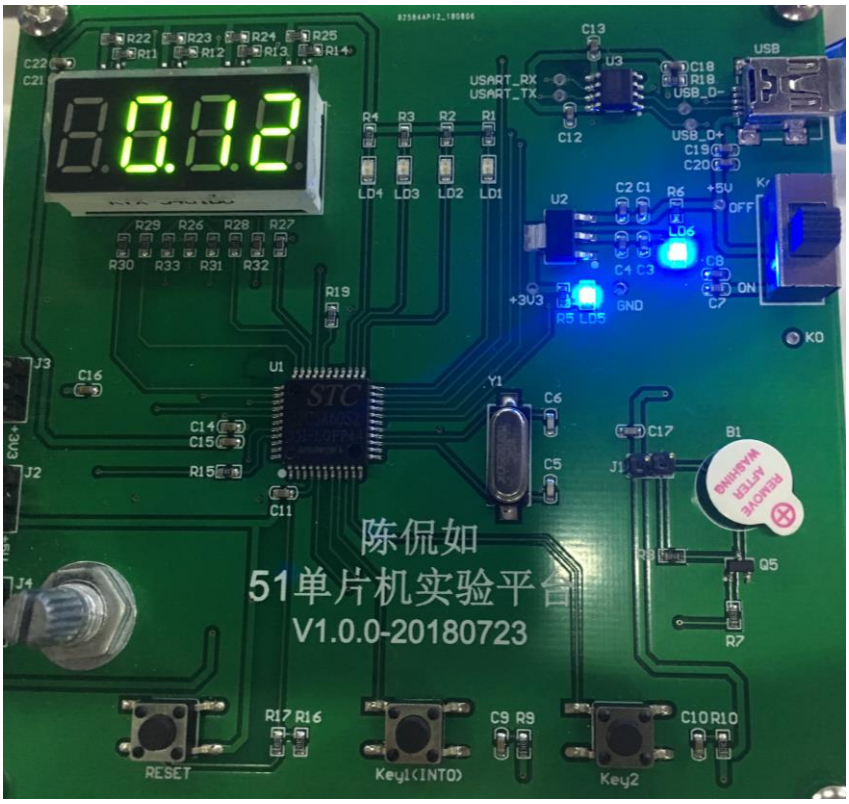

图 1 稍微拧一点旋钮

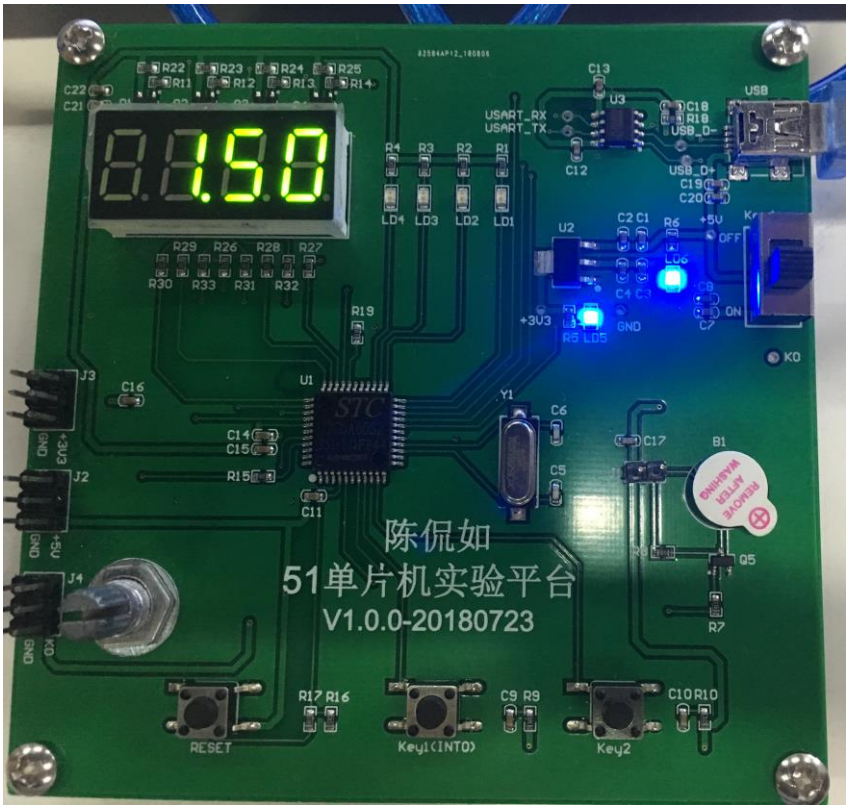

图 2 拧到一半

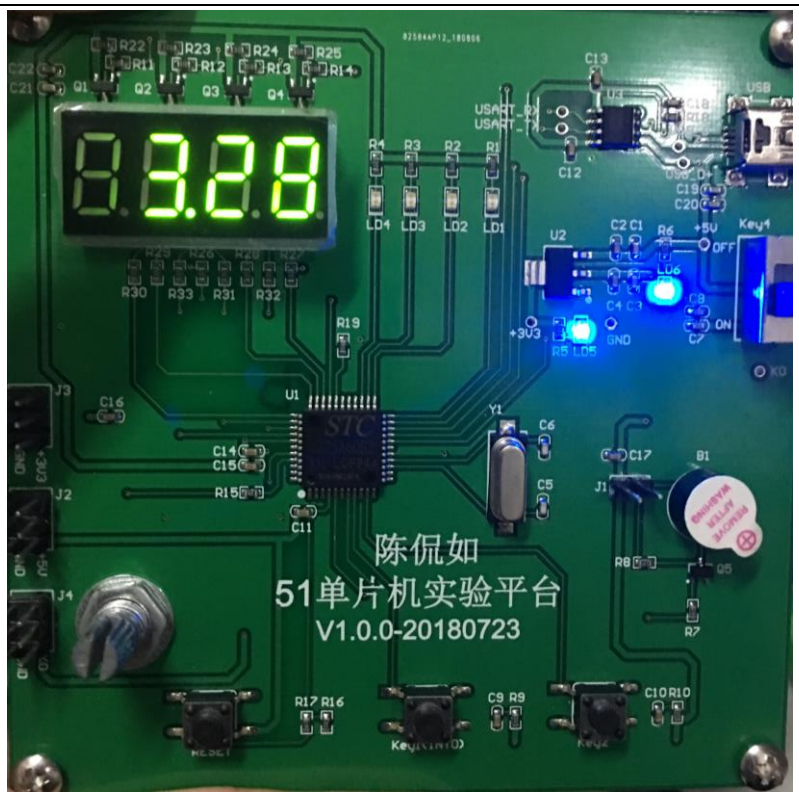

图 3 拧到尽头

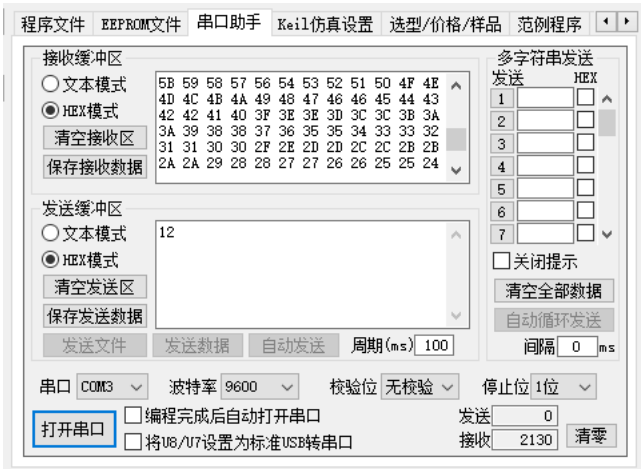

图 4 接收缓冲区中接收到了 AD 转换的值

七、实验结论

通过这次实验的学习，我懂得了片内 A/D 是单片机中最重要的功能之一，大部分数据都是靠 A/D 采集，掌握了如何设置 A/D 端口的寄存器和如何设置串口使用独立波特率发生器，掌握如何将采集的数据进行处理和显示。

指导教师批阅意见: 成绩评定: 指导教师签字: 年　　月　　日
备注:

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。