# 3.4.ADC acquisition

## 1. learning target

1.Understand the basics of ADC.

2.Learn the basic use of ADC, and display the collected values on the serial port assistant.

### ADC introduction of STM32F1:

ADC (analog to digital converter) is an analog-to-digital converter, which can convert analog signals into digital signals. According to its conversion principle, it is mainly divided into three types: successive approximation type, double integral type, and voltage-frequency conversion type.

STM32 has 1~3 ADCs (STM32F101/102 series has only 1 ADC), these ADCs can be used independently or in dual mode (increased sampling rate). The ADC of STM32 is a 12-bit successive approximation analog-to-digital converter. It has 18 channels and can measure 16 external and 2 internal sources. A/D conversion for each channel can be performed in one-shot, continuous, sweep or intermittent mode. The ADC result can be stored in the 16-bit data registers either left-justified or right-justified. An analog watchdog feature allows the application to detect if the input voltage exceeds user-defined high/low thresholds.

The STM32F103 series has at least 2 ADCs, and the STM32F103C8T6 we chose contains 2 12-bit ADCs. The maximum conversion rate of the STM32 ADC is 1Mhz, that is, the conversion time is 1us (obtained under ADCCLK=14M, the sampling period is 1.5 ADC clocks), do not let the ADC clock exceed 14M, otherwise the accuracy of the result will be reduced.

The ADC of STM32 only performs one conversion in single conversion mode. This mode can be started by the ADON bit of the ADC_CR2 register (only applicable to regular channels), or it can be started by an external trigger (applicable to regular channels and injection channels). Yes CONT bit is 0.

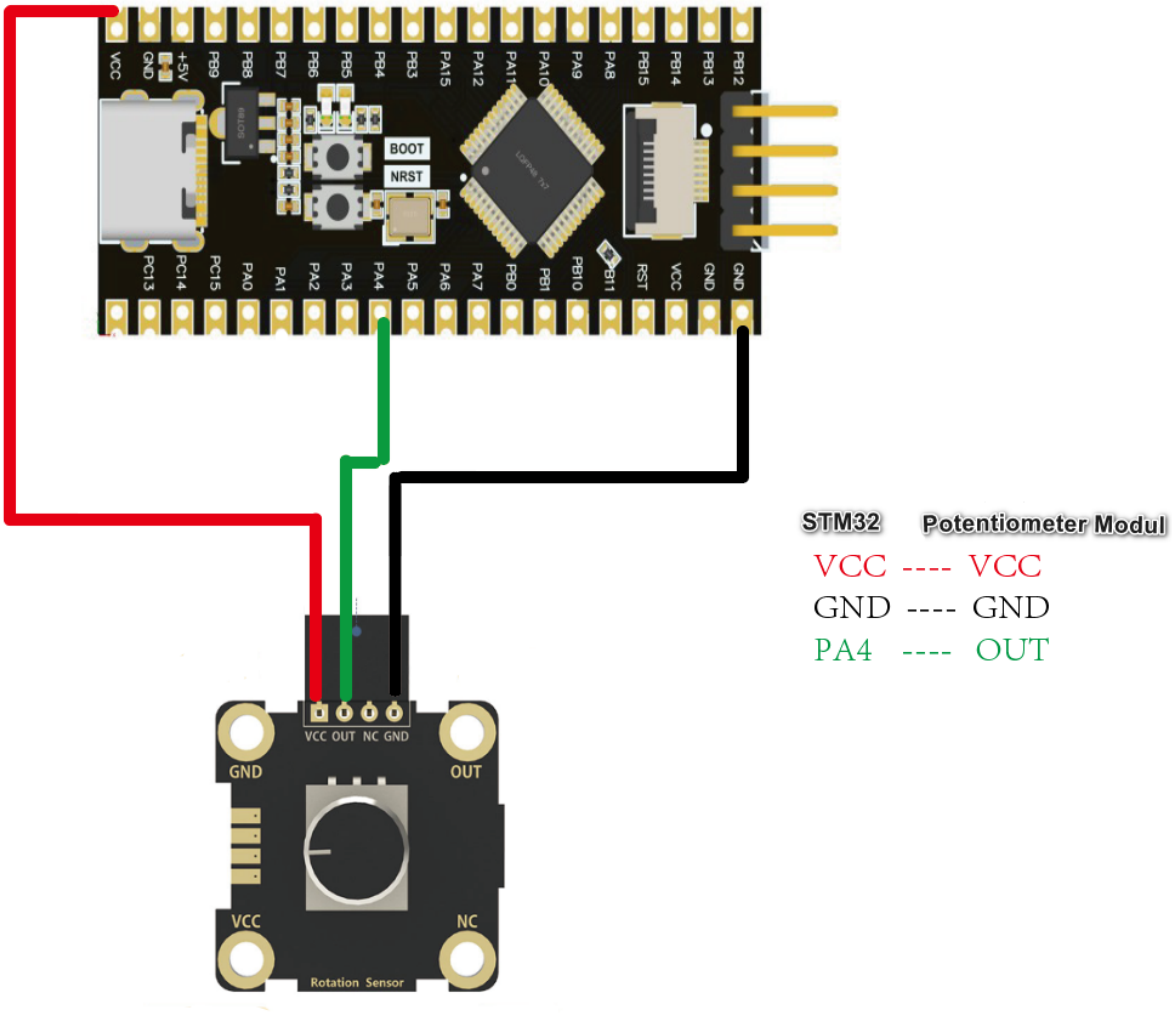
Taking the regular channel as an example, once the selected channel conversion is completed, the conversion result will be stored in the ADC_DR register, the EOC (end of conversion) flag will be set, and if EOCIE is set, an interrupt will be generated. The ADC will then stop until the next start.

The STM32 ADC channel and GPIO correspondence table is as follows:

| | ADC1 | ADC2 | ADC3 |
|---|---|---|---|
| Channel | PA0 | PA0 | PA0 |
| Channel 1 | PA1 | PA1 | PA1 |
| Channel 2 | PA2 | PA2 | PA2 |
| Channel 3 | PA3 | PA3 | PA3 |
| Channel 4 | PA4 | PA4 | PF6 |
| Channel 5 | PA5 | PA5 | PF7 |
| Channel 6 | PA6 | PA6 | PF8 |
| Channel 7 | PA7 | PA7 | PF9 |
| Channel 8 | PB0 | PB0 | PF10 |
| Channel 9 | PB1 | PB1 | |
| Channel 10 | PC0 | PC0 | PC0 |
| Channel 11 | PC1 | PC1 | PC1 |
| Channel 12 | PC2 | PC2 | PC2 |
| Channel 13 | PC3 | PC3 | PC3 |
| Channel 14 | PC4 | PC4 | |
| Channel 15 | PC5 | PC5 | |
| Channel 16 | Temperature Sensor | | |
| Channel 17 | internal reference voltage | | |

## 2. Hardware wiring construction

The potentiometer module and DuPont line used in the ADC acquisition experiment need to be purchased separately, otherwise the course cannot be completed due to the lack of important accessories.




Module Wiring Diagram

# 3、Program code analysis

## ADC initialization code:

```
void Adc_Init(void)
{
  ADC_InitTypeDef ADC_InitStructure;
  GPIO_InitTypeDef GPIO_InitStructure;

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_ADC1 , ENABLE );
  RCC_ADCCLKConfig(RCC_PCLK2_Div6);


  GPIO_InitStructure.GPIO_Pin =GPIO_Pin_4;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  ADC_DeInit(ADC1);
  ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
  ADC_InitStructure.ADC_ScanConvMode = DISABLE;
  ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;
  ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
  ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
  ADC_InitStructure.ADC_NbrOfChannel = 1;
  ADC_Init(ADC1, &ADC_InitStructure);
  ADC_Cmd(ADC1, ENABLE);
  ADC_ResetCalibration(ADC1);
  while(ADC_GetResetCalibrationStatus(ADC1));
  ADC_StartCalibration(ADC1);
  while(ADC_GetCalibrationStatus(ADC1));
}
```

**Get the value of ADC:**

```
float Get_Measure_Volotage(void)
{
  u16 adcx;
  float temp;

  adcx=Get_Adc_Average(ADC_Channel_4, 10);
  temp=(float)adcx*(3.3/4096);
  return temp;
}
```

**Obtain multiple ADC averages:**

```
static u16 Get_Adc_Average(u8 ch, u8 times)
{
  u32 temp_val=0;
  u8 t;
  for(t=0;t<times;t++)
  {
    temp_val+=Get_Adc(ch);
    Delay_ms(5);
  }
  return temp_val/times;
}
```

**Get actual voltage:**

```
float Get_Measure_Volotage(void)
{
  u16 adcx;
  float temp;

  adcx=Get_Adc_Average(ADC_Channel_4, 10);
  temp=(float)adcx*(3.3/4096);
  return temp;
}
```

**display actual voltage:**

```
void voltage_warning(void)
{
    float fVoltage = 0.0f;
    fVoltage = Get_Measure_Volotage();
    printf("Voltage:%.2fV\r\n", fVoltage);
}
```

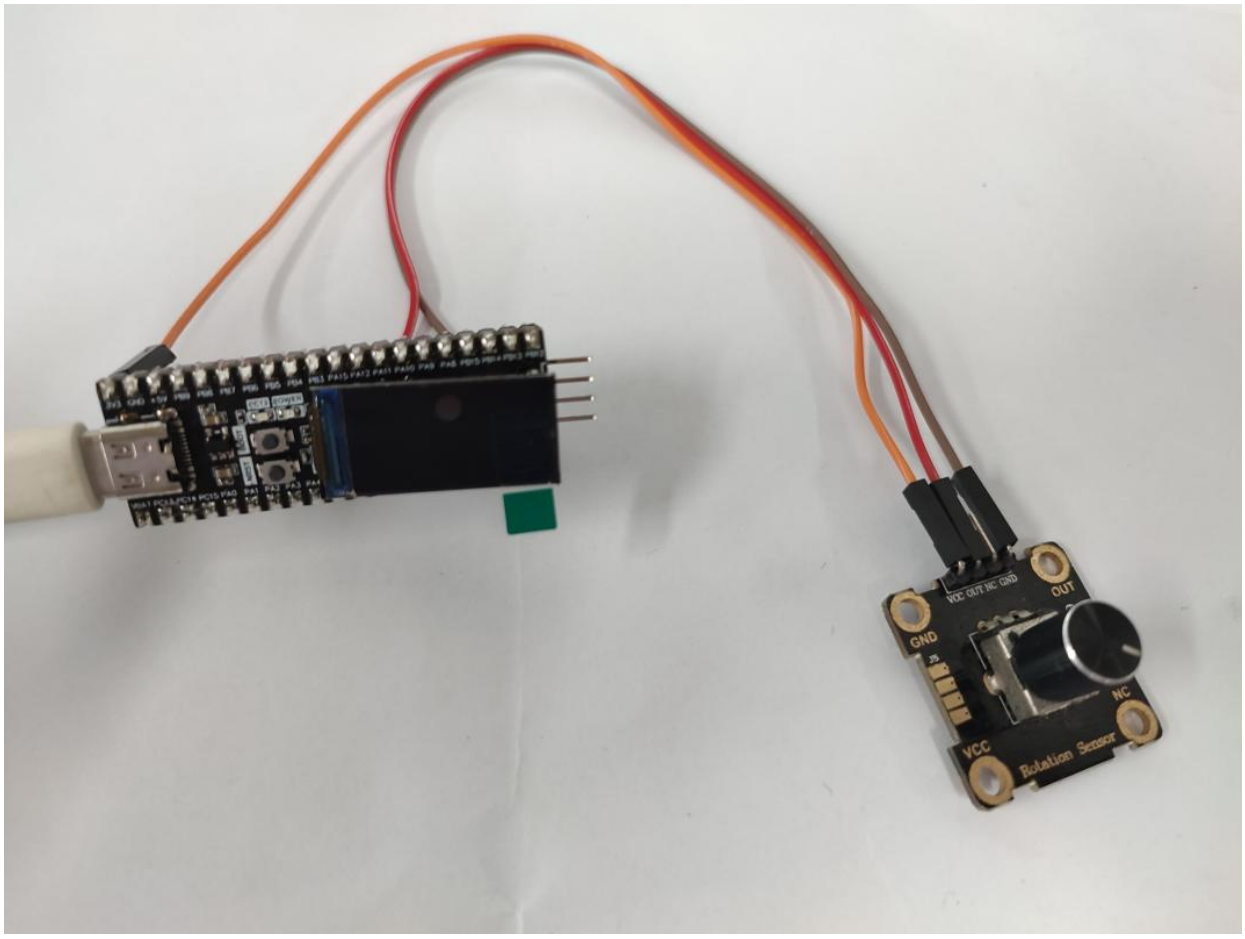
### main function:

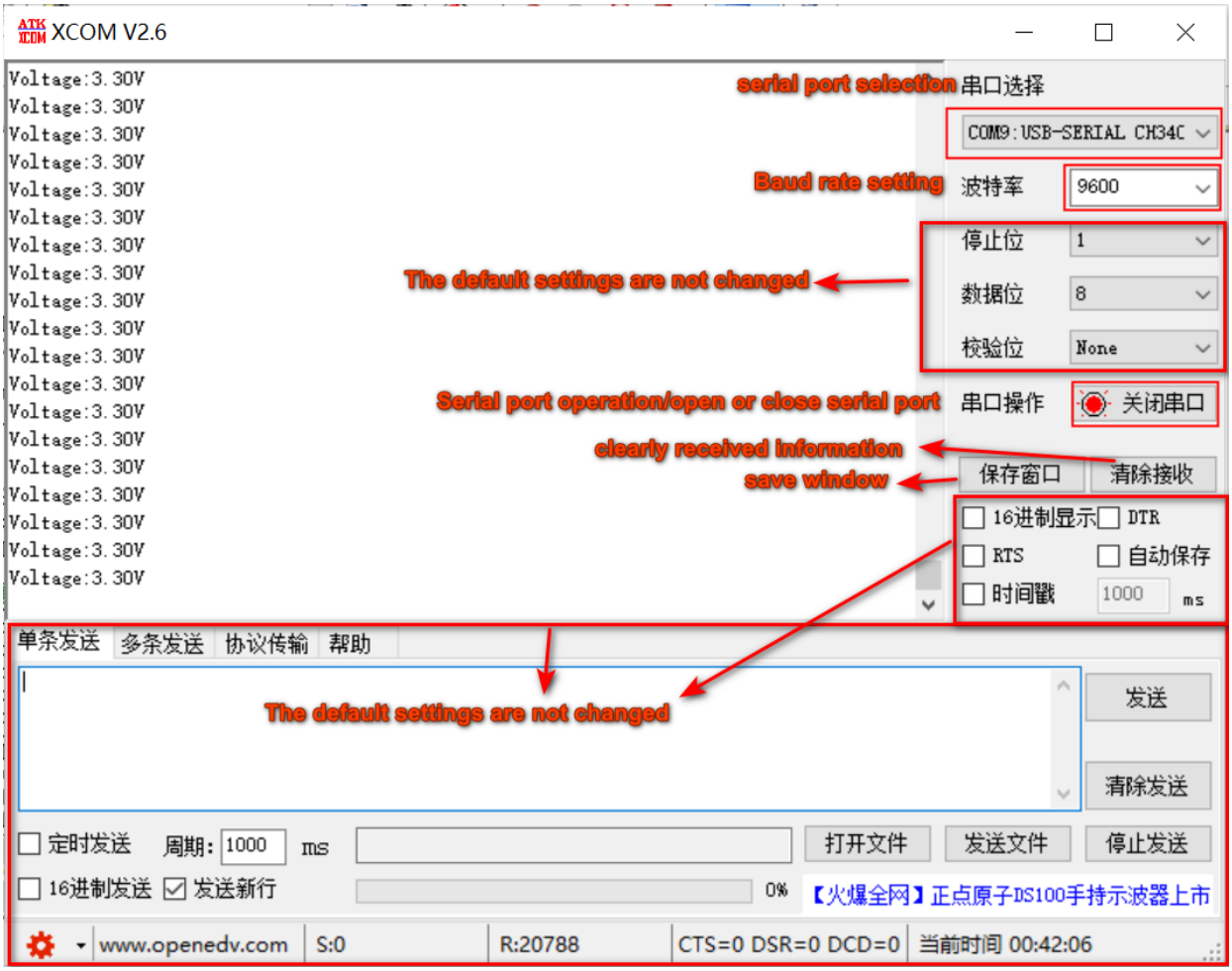
```
int main(void)
{
  bsp_init();
   while(1)
  {
    voltage_warning();
  }

}
```

## 4.Experimental phenomena

After the program download is complete, configure the serial port assistant as shown in the figure below, adjust the potentiometer knob, and you can see the real-time effect of the voltage changing between 0-3.3v in the serial port assistant receiving window.




XCOM V2.6
Voltage:3.30V
serial port selection
串口选择
COM9:USB-SERIAL CH340
Baud rate setting
波特率
9600
停止位
1
The default settings are not changed
数据位
8
校验位
None
Serial port operation/open or close serial port
串口操作
关闭串口
clearly received information
save window
保存窗口
清除接收
16进制显示
DTR
RTS
自动保存
时间戳
1000
ms
单条发送
多条发送
协议传输
帮助
The default settings are not changed
发送
清除发送
定时发送
周期:
1000
ms
打开文件
发送文件
停止发送
16进制发送
发送新行
0%
【火爆全网】正点原子DS100手持示波器上市
www.openedv.com
S:0
R:20788
CTS=0 DSR=0 DCD=0
当前时间 00:42:06