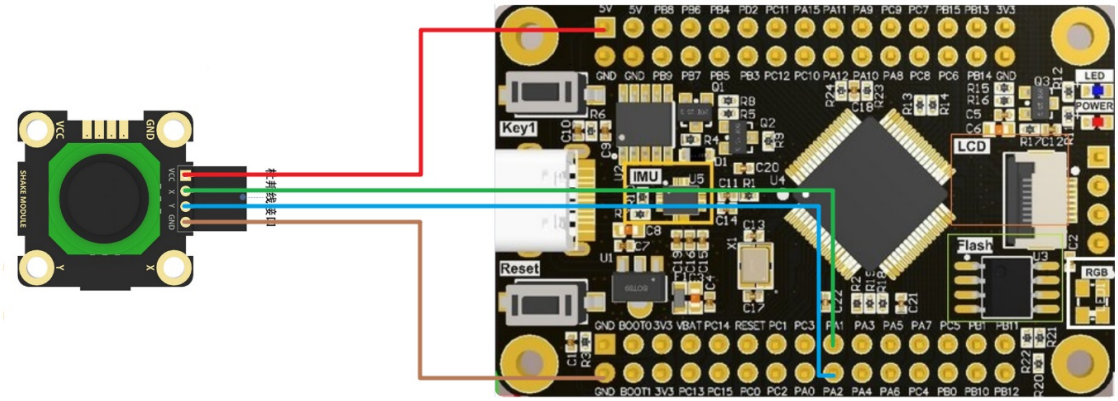
Joystick module: ADC multichannel conversion

Hardware wiring



Joystick module	STM32F103RCT6
VCC	5V/3.3V
X	PA1
Y	PA2
GND	GND

Brief principle

The joystick module can output analog values for the X and Y axes.

The XY axis data of the joystick module is obtained through ADC conversion, the data range is 0-4095, and the direction of the joystick module is simply divided by the data conversion formula, and the specific content can be seen in the code.

Main code

main.c

```
#include "stm32f10x.h"
#include "SysTick.h"
#include "UART.h"
#include "ADC.h"

int main(void)
{
    SysTick_Init();//滴答定时器初始化
    UART1_Init();//UART1初始化
    ADC1_Init();//ADC1初始化

    printf("JoyStick!\n");//打印JoyStick!
```

```
    while(1)
    {
        JoyStick_X_Y_Key_Data();//获取JoyStick X Y轴数据 并打印相关信息到串口
    }
}
```

SysTick.c

```
#include "SysTick.h"

unsigned int Delay_Num;

void SysTick_Init(void)//滴答定时器初始化
{
    while(SysTick_Config(72));//设置重装载值 72 对应延时函数为微秒级
    //若将重装载值设置为72000 对应延时函数为毫秒级
    SysTick->CTRL &= ~(1 << 0);//定时器初始化后关闭，使用再开启
}

void Delay_us(unsigned int NCount)//微秒级延时函数
{
    Delay_Num = NCount;
    SysTick->CTRL |= (1 << 0);//开启定时器
    while(Delay_Num);
    SysTick->CTRL &= ~(1 << 0);//定时器初始化后关闭，使用再开启
}

void SysTick_Handler(void)
{
    if(Delay_Num != 0)
    {
        Delay_Num--;
    }
}
```

SysTick.h

```
#ifndef __SYSTICK_H__
#define __SYSTICK_H__

#include "stm32f10x.h"

void SysTick_Init(void);//滴答定时器初始化
void Delay_us(unsigned int NCount);//微秒级延时函数

#endif
```

UART.c

```
#include "UART.h"

void UART1_Init(void)//UART1初始化
{
    USART_InitTypeDef USART_InitStructure;
    GPIO_InitTypeDef GPIO_InitStructure;

    /* Enable GPIO clock */
    /* 使能GPIOA AFIO时钟 TXD(PA9) RXD(PA10) */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_AFIO, ENABLE);

    /* Enable USART1 Clock */
    /* 使能串口1 */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1 , ENABLE);

    /* Configure USART1 Rx as input floating */
    /* 配置RXD引脚 PA10 浮空输入模式 */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    /* Configure USART1 Tx as alternate function push-pull */
    /* 配置TXD引脚 PA9 复用推挽输出模式 */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    /* USART1 configured as follow:
    - BaudRate = 115200 baud
    - Word Length = 8 Bits
    - One Stop Bit
    - No parity
    - Hardware flow control disabled (RTS and CTS signals)
    - Receive and transmit enabled
    */
    USART_InitStructure.USART_BaudRate = 115200;//波特率设置
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;//8位数据位
    USART_InitStructure.USART_StopBits = USART_StopBits_1;//1位停止位
    USART_InitStructure.USART_Parity = USART_Parity_No;//无奇偶校验位
    USART_InitStructure.USART_HardwareFlowControl =
USART_HardwareFlowControl_None;//无需硬件流控
    USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;//全双工 即发送
也接受
    USART_Init(USART1, &USART_InitStructure);

    /* Enable USART1 Receive and Transmit interrupts */
    USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
    USART_ITConfig(USART1, USART_IT_TXE, ENABLE);

    /* Enable the USART1 */
    /* 使能USART1 */
    USART_Cmd(USART1, ENABLE);
```

```
}

void NVIC_UART1_Init(void)//UART1 NVIC配置
{
    NVIC_InitTypeDef NVIC_InitStructure;

    /* Configure the NVIC Preemption Priority Bits */
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);

    /* Enable the USART1 Interrupt */
    NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}


void USART_SendString(USART_TypeDef* USARTx, char *pt)//给指定串口发送字符串
{
    while(*pt)
    {
        while(USART_GetFlagStatus(USARTx, USART_FLAG_TXE) == RESET);
        USART_SendData(USARTx,*pt);
        while(USART_GetFlagStatus(USARTx, USART_FLAG_TC) == RESET);
        pt++;
    }
}

int fputc(int c, FILE *pt)//printf重定向
{
    USART_TypeDef* USARTx = USART1;
    while(USART_GetFlagStatus(USARTx, USART_FLAG_TXE) == RESET);
    USART_SendData(USARTx, c);
    while(USART_GetFlagStatus(USARTx, USART_FLAG_TC) == RESET);
    return 0;
}

void USART1_IRQHandler(void)
{
    unsigned char ch;
    while(USART_GetFlagStatus(USART1, USART_FLAG_RXNE) == SET)//接收到数据
    {
        ch = USART_ReceiveData(USART1);
        printf("%c\n",ch);
    }
}
```

UART.h

```
#ifndef __UART_H__
#define __UART_H__

#include "stm32f10x.h"
#include "stdio.h"

void UART1_Init(void);//UART1初始化
void USART_SendString(USART_TypeDef* USARTx, char *pt);//给指定串口发送字符串
int fputc(int c, FILE *pt);//printf重定向
void NVIC_UART1_Init(void);//UART1 NVIC配置

#endif
```

ADC.c

```
#include "ADC.h"

void ADC1_Init(void)//ADC1初始化
{
    GPIO_InitTypeDef GPIO_InitStructure;
    ADC_InitTypeDef ADC_InitStructure;

    /* ADCCLK = PCLK2/6 */
    RCC_ADCCLKConfig(RCC_PCLK2_Div6);

    /* Enable ADC1, and GPIOA clocks */
    /* 使能ADC1 GPIOA时钟 */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1| RCC_APB2Periph_GPIOA, ENABLE);

    /* Configure PA1 (ADC Channel) as analog inputs */
    /* 配置ADC 通道1 通道2 PA1 PA2 模拟输入引脚 */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1 | GPIO_Pin_2;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    /* ADC1 configuration */
    /* ADC1 配置 */
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;//独立模式
    ADC_InitStructure.ADC_ScanConvMode = DISABLE;//扫描模式
    ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;//单次转换
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;//软件触发
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;//数据右对齐
    ADC_InitStructure.ADC_NbrOfChannel = 2;//双通道
    ADC_Init(ADC1, &ADC_InitStructure);

    /* Enable ADC1 */
    /* 使能 ADC1 */
    ADC_Cmd(ADC1, ENABLE);

    /* Enable ADC1 reset calibration register */
    /* 使能ADC1复位校准寄存器 */
    ADC_ResetCalibration(ADC1);
    /* Check the end of ADC1 reset calibration register */
    /* 检查ADC1复位校准寄存器的末端 */
```

```
    while(ADC_GetResetCalibrationStatus(ADC1));

    /* Start ADC1 calibration */
    /* 启动 ADC1 校准 */
    ADC_StartCalibration(ADC1);
    /* Check the end of ADC1 calibration */
    /* 检查ADC1校准结束 */
    while(ADC_GetCalibrationStatus(ADC1));
}

uint16_t ADC1_Result(unsigned int Channel)//获取ADC通道转换数据
{
    /* ADC1 regular channels configuration */
    /* ADC1 规则通道配置 */
    ADC_RegularChannelConfig(ADC1, Channel, 1, ADC_SampleTime_55Cycles5);
    /* Start ADC1 Software Conversion */
    /* 启动 ADC1 软件转换 */
    ADC_SoftwareStartConvCmd(ADC1, ENABLE);

    while(ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC) == RESET);
    return (ADC_GetConversionValue(ADC1));
}

uint16_t ADC1_Result_Average(unsigned int Channel, unsigned int Times)//获取ADC通
道数据平均值
{
    int i;
    uint16_t Total = 0;

    for(i = 0; i < Times; i++)
    {
        Total += ADC1_Result(Channel);
                Delay_us(5000);
    }
     return (Total / Times);
}

void JoyStick_X_Y_Key_Data(void)//获取JoyStick X Y轴数据 并打印相关信息到串口
{
    uint16_t ADC1_X, ADC1_Y;
    int X, Y;

    ADC1_X = ADC1_Result_Average(ADC_Channel_1, 10);//获取ADC1 通道1 的10次数据平均
值
    ADC1_Y = ADC1_Result_Average(ADC_Channel_2, 10);//获取ADC1 通道2 的10次数据平均
值

    X = (ADC1_X * 3.3 / 4096);//转换数据 便于后面各个状态判断
    Y = (ADC1_Y * 3.3 / 4096);//转换数据 便于后面各个状态判断

        /* JoyStick XY轴状态判断 */
        if(X == 2 && Y == 3)
        {
            printf("%d\n",ADC1_X);
            printf("%d\n",ADC1_Y);
```

```
            printf("Up!\n");
        }
        else if(X == 2 && Y == 0)
        {
            printf("%d\n",ADC1_X);
            printf("%d\n",ADC1_Y);
            printf("Down!\n");
        }
        else if(X == 3 && Y == 2)
        {
            printf("%d\n",ADC1_X);
            printf("%d\n",ADC1_Y);
            printf("Right!\n");
        }
        else if(X == 0 && Y == 2)
        {
            printf("%d\n",ADC1_X);
            printf("%d\n",ADC1_Y);
            printf("Left!\n");
        }
        else if(X == 3 && Y == 3)
        {
            printf("%d\n",ADC1_X);
            printf("%d\n",ADC1_Y);
            printf("Right_Up!\n");
        }
        else if(X == 0 && Y == 3)
        {
            printf("%d\n",ADC1_X);
            printf("%d\n",ADC1_Y);
            printf("Left_Up!\n");
        }
        else if(X == 3 && Y == 0)
        {
            printf("%d\n",ADC1_X);
            printf("%d\n",ADC1_Y);
            printf("Right_Down!\n");
        }
        else if(X == 0 && Y == 0)
        {
            printf("%d\n",ADC1_X);
            printf("%d\n",ADC1_Y);
            printf("Left_Down!\n");
        }
}
```

ADC.h

```
#ifndef __ADC_H__
#define __ADC_H__

#include "stm32f10x.h"
#include "UART.h"
#include "SysTick.h"

void ADC1_Init(void);//ADC1初始化
uint16_t ADC1_Result(unsigned int Channel);//获取ADC 通道转换数据
uint16_t ADC1_Result_Average(unsigned int Channel, unsigned int Times);//获取ADC通道数据平均值
void JoyStick_X_Y_Key_Data(void);//获取JoyStick X Y轴数据 并打印相关信息到串口

#endif
```

Phenomenon

After downloading the program, press the Reset key once, and the downloaded program will run.

When you shake the stick module in different directions, the serial debug assistant prints the data acquired by the current ADC channel 1 and channel 2 and the direction based on the data conversion.

Note: If you find that the joystick is in the opposite direction, you can adjust the placement of the joystick module or modify the code yourself.

```
[2023-05-25 14:29:18.480]# RECV ASCII>
JoyStick!

[2023-05-25 14:29:22.282]# RECV ASCII>
3186
3748
Up!

[2023-05-25 14:29:22.391]# RECV ASCII>
3185
4095
Up!

[2023-05-25 14:29:22.484]# RECV ASCII>
2557
4095
Up!

[2023-05-25 14:29:22.592]# RECV ASCII>
26
4095
Left_Up!

[2023-05-25 14:29:22.686]# RECV ASCII>
8
3330
Left!

[2023-05-25 14:29:22.999]# RECV ASCII>
4095
3064
Right!

[2023-05-25 14:29:23.091]# RECV ASCII>
4095
3925
Right_Up!

[2023-05-25 14:29:23.200]# RECV ASCII>
3854
3964
Right_Up!

[2023-05-25 14:29:23.295]# RECV ASCII>
1095
4095
Left_Up!

[2023-05-25 14:29:29.821]# RECV ASCII>
2751
5
Down!
```