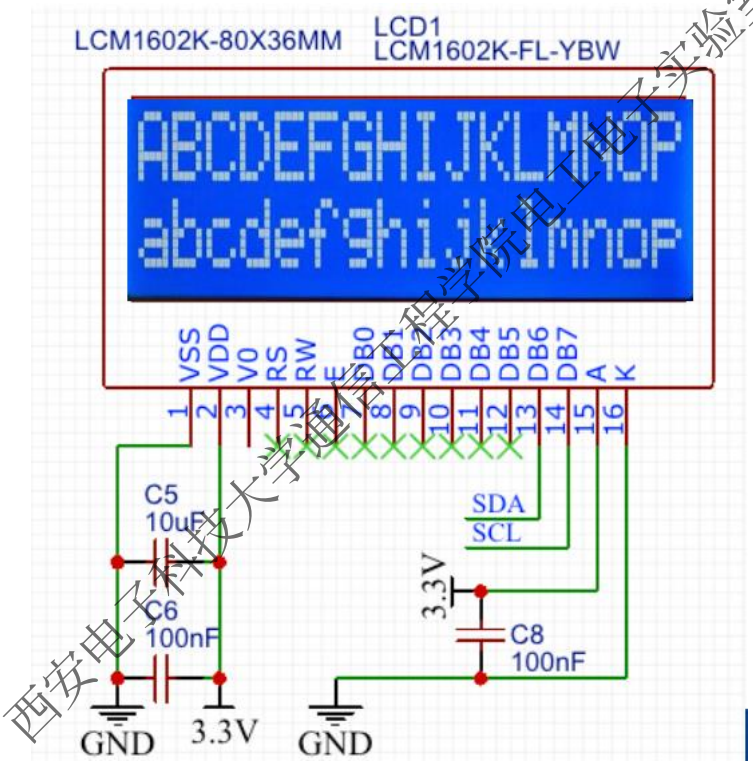
STM32 开发（I2C 和液晶操作）

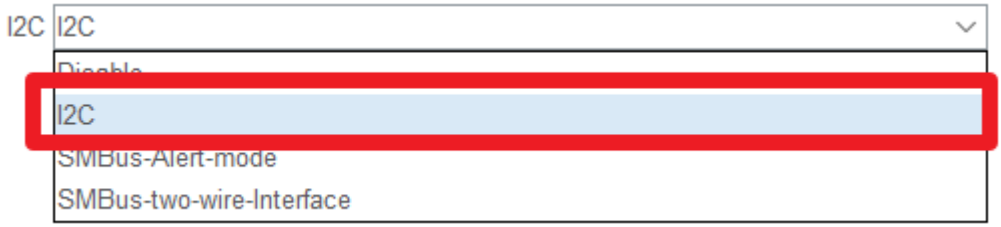


西安电子科技大学

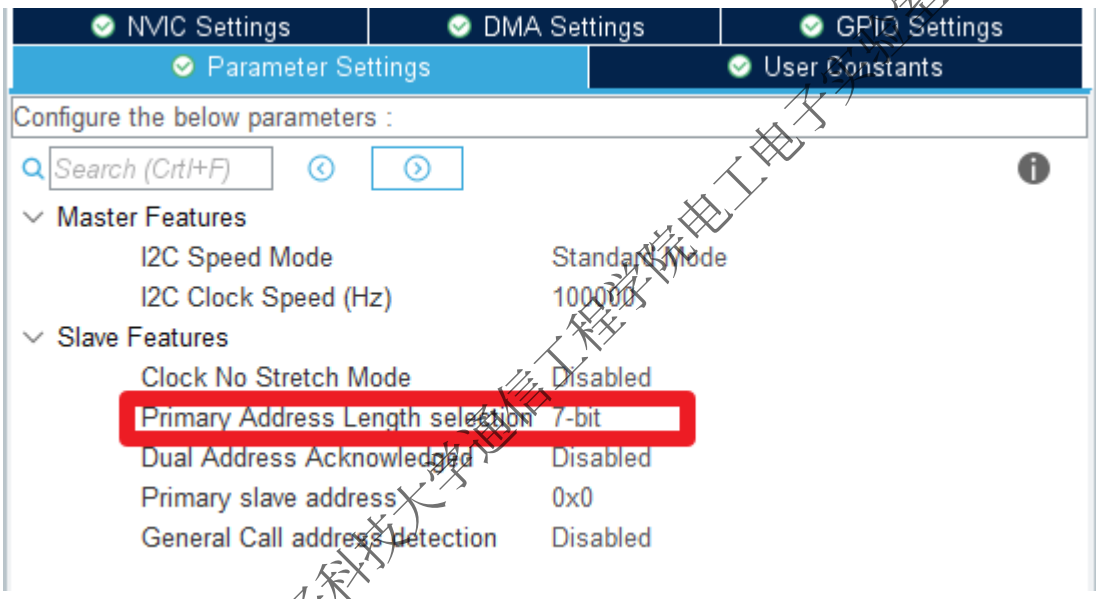
通信工程学院

I2C 设置

CUBE MX 中打开 I2C



设置参数:



I2C 操作函数

1 、 HAL_StatusTypeDef HAL_I2C_IsDeviceReady(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint32_t Trials, uint32_t Timeout) // Checks if target device is ready for communication.
用途：检查设备是否准备好
*hi2c，I2C 设备句柄
DevAddress，设备地址
Trials 尝试次数
Timeout，超时时间

2、HAL_I2C_StateTypeDef HAL_I2C_GetState(I2C_HandleTypeDef *hi2c) //Return the I2C handle state. 总线工作状态

用途：检查总线状态
*hi2c，I2C 设备句柄
eg：
flag=HAL_I2C_GetState(&hi2c1);

3、HAL_StatusTypeDef HAL_I2C_Mem_Write(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t *pData, uint16_t Size, uint32_t Timeout)
用途：给 I2C 带地址批量写数据
*hi2c，I2C 设备句柄
DevAddress，设备地址
MemAddress 存储地址
MemAddSize，数据位宽
pData，写入数据的首地址
Size，写入数据的长度
Timeout，超时时间
eg:
HAL_I2C_Mem_Write(&hi2c1, ADDRESS_W, MPU_PWR_MGMT1_REG, 1, &pdata, 1, HAL_MAX_DELAY);

4、HAL_StatusTypeDef HAL_I2C_Mem_Read(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t *pData, uint16_t Size, uint32_t Timeout) // Read an amount of data in blocking mode from a specific memory address
用途：给 I2C 带地址批量读数据
*hi2c，I2C 设备句柄
DevAddress，设备地址
MemAddress 存储地址
MemAddSize，数据位宽，MemAddSize 只能填 1or2，代表 8 位或者 16 位。
pData，存储数据的首地址
Size，读入数据的长度
Timeout，超时时间
eg:
HAL_I2C_Mem_Read(&hi2c1, ADDRESS_R, MPU_DEVICE_ID_REG, 1, &pdata, 1, HAL_MAX_DELAY);
#define I2C_MEMADD_SIZE_8BIT 0x00000001U // 1 代表 8bit。
#define I2C_MEMADD_SIZE_16BIT 0x00000010U // 2 代表 16bit。

5、 HAL_I2C_Master_Receive（）；// STM32 主机接收，不需要用到寄存器地址
和前面不同的是这里没有寄存器地址
6、 HAL_I2C_Master_Receive_IT（）；//中断 IIC 接收

7、 HAL_I2C_Master_Receive_DMA（）；// DMA 方式的 IIC 接收

8、HAL_I2C_Master_Transmit_IT(); //中断 IIC 发送

9、HAL_I2C_Master_Transmit_DMA();　　// DMA 方式的 IIC 发送

10 、 HAL_I2C_Master_Transmit(&hi2c2,salve_add,PA_BUFF,sizeof(PA_BUFF),0x10); //STM32 主机发送

11、HAL_I2C_Slave_Receive();// STM32 从机机接收，不需要用到寄存器地址

12、HAL_I2C_Slave_Transmit();// STM32 从机机发送，不需要用到寄存器地址

13、HAL_I2C_Slave_Receive_IT();

14、HAL_I2C_Slave_Receive_DMA();

15、HAL_I2C_Slave_Transmit_IT();

16、HAL_I2C_Slave_Transmit_DMA();

示例

LCD1602.H

```
#ifndef _LCD1602_IIC_H
#define _LCD1602_IIC_H

#include "i2c.h"

/*
**以下为本文件所有函数的声明，方便调用：
*/

//所用 LCD 为 JXD1602-4
#define LCD1602_ADDR 0x78

void LCD_Init(void);
void LCD_DisplayLine(unsigned char line, uint8_t *str);
void I2C_ss_SendCmd(uint8_t cmd);
void I2C_ss_SendDat(uint8_t Dat);
void I2C_ss_SendMultipleDat(uint8_t * pData, uint8_t length);
#endif
```

LCD1602.C

```
#include "lcd1602_iic.h"

/**************************************************************************
* 名      称：I2C_ss_SendCmd()
* 功      能：I2C 发送液晶命令
* 入口参数：uint8_t cmd，需发送的命令
* 出口参数：无
**************************************************************************/
void I2C_ss_SendCmd(uint8_t cmd)
{
        HAL_I2C_Mem_Write(&hi2c1, LCD1602_ADDR, 0x80, I2C_MEMADD_SIZE_8BIT, & cmd, 1, 100);
}

/**************************************************************************
* 名      I2C_ss_SendDat(uint8_t Dat)
* 功      能：I2C 发送给液晶一字节数据
* 入口参数：uint8_t Dat，需发送的数据
* 出口参数：无
**************************************************************************/
void I2C_ss_SendDat(uint8_t Dat)
{
        HAL_I2C_Master_Transmit(&hi2c1, LCD1602_ADDR, &Dat, 1, 100);
}

/**************************************************************************
* 名      I2C_ss_SendDat(uint8_t Dat)
* 功      能：I2C 发送给液晶一字节数据
* 入口参数：uint8_t * pData，需发送的数据首地址；
* 入口参数：uint8_t length，  需发送的数据总长度
* 出口参数：无
**************************************************************************/
void I2C_ss_SendMultipleDat(uint8_t * pData, uint8_t length)
{
        HAL_I2C_Master_Transmit(&hi2c1, LCD1602_ADDR, pData, length, 100);
}


/**************************************************************************
* 名      称：LCD_Init()
* 功      能：LCD 初始化
* 入口参数：无
* 出口参数：无
**************************************************************************/
void LCD_Init(void)
{
        I2C_ss_SendCmd(0x38);// Function set
        I2C_ss_SendCmd(0x0C);// Display ON/OFF
        I2C_ss_SendCmd(0x01);// Clear display
        HAL_Delay(2);
        I2C_ss_SendCmd(0x06);// Entry mode set
}

/**************************************************************************
* 名      称：LCD_DisplayLine()
* 功      能：LCD 显示字符串
* 入口参数：unsigned char line，行(0-1)
* 入口参数：uint8_t *str，需显示的字符串
* 出口参数：无
**************************************************************************/
void LCD_DisplayLine(unsigned char line, uint8_t *str)
{
        //I2C 数据缓冲区
        uint8_t I2C_BUF[20],cnt;
        uint8_t * pdata;
        pdata = str;
        //判别行数，得到显示起始地址
        uint8_t nLineAddr = line ? (0xC0U) : (0x80U);
        //0x80,表示后面一字节是命令
        I2C_BUF[0]=0x80;
        //写入首地址
        I2C_BUF[1]=nLineAddr;
        //0x80,表示后面是数据
        I2C_BUF[2]=0x40;
        //16 字节显示数据
        for(cnt=3;cnt<19;cnt++)
        {
                I2C_BUF[cnt]=*pdata;
                pdata++;
        }
        //一次性发送给 LCD
        I2C_ss_SendMultipleDat(I2C_BUF,19);
}
```

作业：

读取温度传感器，读取 DHT11/DS18B20 温度传感器数据，在液晶屏上显示温度数据。

DHT11 参考示例

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_ADC1_Init();
MX_USART1_UART_Init();
MX_TIM3_Init();
MX_I2C1_Init();
/* USER CODE BEGIN 2 */
    //???TIM3????
__HAL_TIM_CLEAR_IT(&htim3, TIM_IT_UPDATE);
//??TIM3??
HAL_TIM_Base_Start_IT(&htim3);
 __HAL_RCC_I2C1_CLK_ENABLE();

    printf("System Ready!");
LCD_Init();
    LCD_DisplayLine(0,line1);
    LCD_DisplayLine(1,line2);

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{

        HAL_GPIO_TogglePin(LED0_GPIO_Port,LED0_Pin);
        DHT11_Read_TempAndHumidity(&DHT11_Data);
        humidity=DHT11_Data.humidity;                   //湿度
        temperature=DHT11_Data.temperature;       //温度
        printf("H=%3.1f,T=%3.1f\r\n",humidity,temperature);
        memset(line1,0x20,16);
        memset(line2,0x20,16);
        sprintf((char *) line1, "Humidity=%3.1f",humidity);
        sprintf( (char *) line2, "Temperature=%3.1f",temperature);
        LCD_DisplayLine(0,line1);
        LCD_DisplayLine(1,line2);
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
        HAL_Delay(1000);
}
/* USER CODE END 3 */
```

```
#ifndef __DHT11_H
#define __DHT11_H

#include "main.h"

typedef struct
{
        uint8_t   humi_high8Bit;                    //原始数据：湿度高 8 位
        uint8_t   humi_low8bit;                     //原始数据：湿度低 8 位
        uint8_t   temp_high8bit;                    //原始数据：温度高 8 位
        uint8_t   temp_low8bit;                     //原始数据：温度高 8 位
        uint8_t   check_sum;                        //校验和
  float     humidity;                                                    //实际湿度
  float     temperature;                                                 //实际温度
} DHT11_Data_TypeDef;

enum DHT11_Pin_mode_enum
{
        Intput=1,
        Output,
};

#define DHT11_DATA_Pin GPIO_PIN_1
#define DHT11_DATA_GPIO_Port GPIOA
#define Read_DHT11_DATA()   HAL_GPIO_ReadPin(DHT11_DATA_GPIO_Port,DHT11_DATA_Pin)
#define DHT11_DATA_SET()    HAL_GPIO_WritePin(DHT11_DATA_GPIO_Port,DHT11_DATA_Pin,GPIO_PIN_SET)
#define DHT11_DATA_RESET() HAL_GPIO_WritePin(DHT11_DATA_GPIO_Port,DHT11_DATA_Pin,GPIO_PIN_RESET)

uint8_t DHT11_Read_TempAndHumidity(DHT11_Data_TypeDef *DHT11_Data);
#endif
```

```
#include "DHT11.h"

#define Delay_ms(x)   HAL_Delay(x)

static void DHT11_Delay(uint16_t time)
{
  while(time)
  {
                unsigned char i;
        i = 5;
        while (--i)
        {
        }
    time--;
  }
}

void DHT11_PIN_MODE(uint8_t mode)                    //初始化 DHT11_DATA PIN
{
        GPIO_InitTypeDef GPIO_InitStruct = {0};
        if(mode==Intput)
        {
                GPIO_InitStruct.Pin = DHT11_DATA_Pin;
                GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
                GPIO_InitStruct.Pull = GPIO_PULLUP;
                GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
                HAL_GPIO_Init(DHT11_DATA_GPIO_Port, &GPIO_InitStruct);
        }
        else
        {
                GPIO_InitStruct.Pin = DHT11_DATA_Pin;
                GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
                GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
                HAL_GPIO_Init(DHT11_DATA_GPIO_Port, &GPIO_InitStruct);
        }
}


static uint8_t DHT11_ReadByte ( void )
{

        DHT11_PIN_MODE(Intput);
        uint8_t i, temp=0;
        for(i=0;i<8;i++)
        {
                /*每 bit 以 50us 低电平标置开始，轮询直到从机发出 的 50us 低电平 结束*/
                while(Read_DHT11_DATA()==GPIO_PIN_RESET);

                /*DHT11 以 26~28us 的高电平表示"0"，以 70us 高电平表示"1"，
                 *通过检测 x us 后的电平即可区别这两个状 ，x 即下面的延时
                 */
                DHT11_Delay(40); //延时 x us 这个延时需要大于数据 0 持续的时间即可

                if(Read_DHT11_DATA()==GPIO_PIN_SET)/* x us 后仍为高电平表示数据"1" */
                {
                                /* 等待数据 1 的高电平结束 */
                                while(Read_DHT11_DATA()==GPIO_PIN_SET);
                                temp|=(uint8_t)(0x01<<(7-i));  //把第 7-i 位置 1，MSB 先行
                }
                else            // x us 后为低电平表示数据"0"
                {
                                temp&=(uint8_t)~(0x01<<(7-i)); //把第 7-i 位置 0，MSB 先行
                }
        }
        return temp;
}

uint8_t DHT11_Read_TempAndHumidity(DHT11_Data_TypeDef *DHT11_Data)
{
        uint8_t temp;
        uint16_t humi_temp;

        /*输出模式*/
        DHT11_PIN_MODE(Output);
        /*主机拉低*/
        DHT11_DATA_RESET();
        /*延时 18ms*/
        Delay_ms(18);

        /*总线拉高 主机延时 30us*/
        DHT11_DATA_SET();

        DHT11_Delay(30);    //延时 30us

        /*主机设为输入 判断从机响应信号*/
        DHT11_PIN_MODE(Intput);

        /*判断从机是否有低电平响应信号 如不响应则跳出，响应则向下运行*/
        if(Read_DHT11_DATA()==GPIO_PIN_RESET)
        {
                /*轮询直到从机发出 的 80us 低电平 响应信号结束*/
                while(Read_DHT11_DATA()==GPIO_PIN_RESET);

                /*轮询直到从机发出的 80us 高电平 标置信号结束*/
                while(Read_DHT11_DATA()==GPIO_PIN_SET);

                /*开始接收数据*/
                DHT11_Data->humi_high8Bit= DHT11_ReadByte();
                DHT11_Data->humi_low8bit = DHT11_ReadByte();
                DHT11_Data->temp_high8bit= DHT11_ReadByte();
                DHT11_Data->temp_low8bit = DHT11_ReadByte();
                DHT11_Data->check_sum       = DHT11_ReadByte();

                /*读取结束，引脚改为输出模式*/
                DHT11_PIN_MODE(Output);
                /*主机拉高*/
                DHT11_DATA_SET();

                /* 对数据进行处理 */
                humi_temp=DHT11_Data->humi_high8Bit*100+DHT11_Data->humi_low8bit;
                DHT11_Data->humidity =(float)humi_temp/100;

                humi_temp=DHT11_Data->temp_high8bit*100+DHT11_Data->temp_low8bit;
                DHT11_Data->temperature=(float)humi_temp/100;

                /*检查读取的数据是否正确*/
                temp = DHT11_Data->humi_high8Bit + DHT11_Data->humi_low8bit +
                DHT11_Data->temp_high8bit+ DHT11_Data->temp_low8bit;
                if(DHT11_Data->check_sum==temp)
                {
                                return SUCCESS;
                }
                else
                                return ERROR;
        }
        else
                                return ERROR;
}
```