目 录

[主函数 1](#_Toc19016)

[485驱动 5](#_Toc5219)

[OLED驱动 10](#_Toc16631)

[CRC16校验 19](#_Toc24957)

[PH表驱动 21](#_Toc30808)

[溶氧仪驱动 29](#_Toc18674)

# 主函数

#include "main.h"

#include "rs485.h"

#include "PHMeter.h"

#include "DissolvedOxygenMeter.h"

#include "string.h"

/\* Private typedef -----------------------------------------------------------\*/

/\* Private define ------------------------------------------------------------\*/

/\* Private macro -------------------------------------------------------------\*/

/\* Private variables ---------------------------------------------------------\*/

//static GPIO\_InitTypeDef GPIO\_InitStruct;

I2C\_HandleTypeDef I2cHandle;

\_\_IO uint32\_t UserButton1Status = 0;

/\* Private function prototypes -----------------------------------------------\*/

static void SystemClock\_Config(void);

static void Error\_Handler(void);

void CMDSwitch(uint8\_t roundCnt);

void DisplaySwitch(uint8\_t roundCnt);

/\* Private functions ---------------------------------------------------------\*/

/\* Buffer used for transmission \*/

extern uint8\_t aTxBuffer[];

/\* Buffer used for reception \*/

extern uint8\_t aRxBuffer[];

extern uint8\_t aRxBufferBackUp[];

int main(void)

{

uint8\_t roundCnt=0x00;

HAL\_Init();

/\* Configure the system clock to 48 MHz \*/

SystemClock\_Config();

RS485\_Init(9600);

OLED\_Init();

BSP\_LED\_Init(LED1);

BSP\_LED\_Init(LED2);

BSP\_LED\_Init(LED3);

OLED\_Clear();

DOMeterDisplay();

RS485\_Receive\_Data(aRxBuffer,RXBUFFERSIZE);

while (1)

{

CMDSwitch(roundCnt);

DisplaySwitch(roundCnt);

RS485\_Check();

PHMeterCheck();

DOMeterCheck();

HAL\_GPIO\_TogglePin(LED2\_GPIO\_PORT, LED2\_PIN);

/\* Insert delay 100 ms \*/

HAL\_Delay(100);

roundCnt++;

}

}

static void SystemClock\_Config(void)

{

RCC\_ClkInitTypeDef RCC\_ClkInitStruct;

RCC\_OscInitTypeDef RCC\_OscInitStruct;

/\* No HSE Oscillator on Nucleo, Activate PLL with HSI/2 as source \*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_NONE;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSI;

RCC\_OscInitStruct.PLL.PREDIV = RCC\_PREDIV\_DIV1;

RCC\_OscInitStruct.PLL.PLLMUL = RCC\_PLL\_MUL12;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct)!= HAL\_OK)

{

Error\_Handler();

}

/\* Select PLL as system clock source and configure the HCLK, PCLK1 clocks dividers \*/

RCC\_ClkInitStruct.ClockType = (RCC\_CLOCKTYPE\_SYSCLK | RCC\_CLOCKTYPE\_HCLK | RCC\_CLOCKTYPE\_PCLK1);

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_1)!= HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @param None

\* @retval None

\*/

static void Error\_Handler(void)

{

/\* User may add here some code to deal with this error \*/

while(1)

{

}

}

void CMDSwitch(uint8\_t roundCnt){

uint8\_t cntPerRound=0xff;

if(roundCnt%cntPerRound==0){

DOMeterRequestData();

HAL\_Delay(8000);

}else if(roundCnt%cntPerRound==10){

PHMeterRequestT();

HAL\_Delay(1000);

}else if(roundCnt%cntPerRound==20){

PHMeterRequestORP();

HAL\_Delay(1000);

}else if(roundCnt%cntPerRound==100){

PHMeterRequestPH();

HAL\_Delay(1000);

}

}

void DisplaySwitch(uint8\_t roundCnt){

if(roundCnt%100==0){

OLED\_Clear();

PHMeterDisplay();

}else if(roundCnt%100==50){

OLED\_Clear();

DOMeterDisplay();

}

}

/\*\*

\* @brief EXTI line detection callbacks

\* @param GPIO\_Pin: Specifies the pins connected EXTI line

\* @retval None

\*/

void HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin)

{

if(GPIO\_Pin == USER\_BUTTON1\_PIN)

{

UserButton1Status = 1;

}

if(GPIO\_Pin == USER\_BUTTON2\_PIN)

{

// UserButton1Status = 1;

}

}

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

void assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* User can add his own implementation to report the file name and line number,

ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* Infinite loop \*/

while (1)

{

}

}

#endif

# 485驱动

#include "sys.h"

#include "rs485.h"

#include "delay.h"

#include "string.h"

#include "PHMeter.h"

#include "DissolvedOxygenMeter.h"

UART\_HandleTypeDef UartHandle;

\_\_IO ITStatus UartReady = RESET;

\_\_IO uint32\_t UserButtonStatus = 0; /\* set to 1 after User Button interrupt \*/

static void Error\_Handler(void);

/\* Buffer used for transmission \*/

uint8\_t aTxBuffer[TXBUFFERSIZE];

/\* Buffer used for reception \*/

uint8\_t aRxBuffer[RXBUFFERSIZE];

uint8\_t allZero[TXBUFFERSIZE];

uint8\_t aRxBufferBackUp[RXBUFFERSIZE];

uint8\_t RS485Reg=0x00;

/\*\*

\* @brief Init RS485.

\* @param bound:BaudRate

\* This parameter can be one of the following values:

\* @arg 9600 115200 etc.

\* @note None

\* @retval None

\*/

void RS485\_Init(u32 bound)

{

GPIO\_InitTypeDef gpioinitstruct;

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

gpioinitstruct.Pin = RS485\_2\_RE\_PIN;

gpioinitstruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

gpioinitstruct.Pull = GPIO\_NOPULL;

gpioinitstruct.Speed = GPIO\_SPEED\_FREQ\_HIGH;

HAL\_GPIO\_Init(RS485\_2\_RE\_GPIO\_PORT, &gpioinitstruct);

UartHandle.Instance = USART2;

UartHandle.Init.BaudRate = bound;

UartHandle.Init.WordLength = UART\_WORDLENGTH\_8B;

UartHandle.Init.StopBits = UART\_STOPBITS\_1;

UartHandle.Init.Parity = UART\_PARITY\_NONE;

UartHandle.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

UartHandle.Init.Mode = UART\_MODE\_TX\_RX;

UartHandle.AdvancedInit.AdvFeatureInit = UART\_ADVFEATURE\_NO\_INIT;

if(HAL\_UART\_DeInit(&UartHandle) != HAL\_OK)

{

Error\_Handler();

}

if(HAL\_UART\_Init(&UartHandle) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief Send data with RS485.

\* @param buf: first address of the data buffer to be sent

len: length of data to be sent. in Byte

\* @arg

\* @note None

\* @retval None

\*/

void RS485\_Send\_Data(u8 \*buf,u8 len)

{

RS485\_2\_RE\_HIGH();

/\* The board sends the message and expects to receive it back \*/

/\*##-2- Start the transmission process #####################################\*/

/\* While the UART in reception process, user can transmit data through

"aTxBuffer" buffer \*/

if(HAL\_UART\_Transmit(&UartHandle,(uint8\_t\*)buf, len,1000) != HAL\_OK)

{

Error\_Handler();

}

/\*##-3- Wait for the end of the transfer ###################################\*/

/\* Reset transmission flag \*/

UartReady = RESET;

RS485\_2\_RE\_LOW();

}

/\*\*

\* @brief Get data received by RS485.

\* @param buf: first address of the data buffer used to store data received

len: length of data read. in Byte

\* @arg

\* @note None

\* @retval None

\*/

void RS485\_Receive\_Data(u8 \*buf,u8 len)

{

/\*##-4- Put UART peripheral in reception process ###########################\*/

while(HAL\_UART\_Receive\_IT(&UartHandle, (uint8\_t \*)buf, len) != HAL\_OK)

{

Error\_Handler();

}

}

void RS485\_Check(void){

uint8\_t ReceiveDataAddr=0x00;

if(RS485Reg){

memcpy(aRxBufferBackUp,aRxBuffer,RXBUFFERSIZE);

memcpy(aRxBuffer,allZero,RXBUFFERSIZE);

if(RS485Reg&RS485\_2\_REC){

ReceiveDataAddr=aRxBufferBackUp[0];

switch (ReceiveDataAddr){

case PHMeterAddr:

memcpy(PHMeterDataBuf,aRxBufferBackUp,PHMETER\_DATABUF\_SIZE);

PHMeterReg|=PHMETER\_RBUF\_UPDATE;

break;

case DOMeterAddr:

memcpy(DOMeterDataBuf,aRxBufferBackUp,DOMETER\_DATABUF\_SIZE);

DOMeterReg|=DOMETER\_RBUF\_UPDATE;

break;

default: ;

}

memcpy(aRxBufferBackUp,allZero,RXBUFFERSIZE);

}

RS485Reg&=~RS485\_2\_REC;

}

}

/\*\*

\* @brief Tx Transfer completed callback

\* @param UartHandle: UART handle.

\* @note This example shows a simple way to report end of IT Tx transfer, and

\* you can add your own implementation.

\* @retval None

\*/

void HAL\_UART\_TxCpltCallback(UART\_HandleTypeDef \*UartHandle)

{

/\* Set transmission flag: trasfer complete\*/

UartReady = SET;

}

/\*\*

\* @brief Rx Transfer completed callback

\* @param UartHandle: UART handle

\* @note This example shows a simple way to report end of DMA Rx transfer, and

\* you can add your own implementation.

\* @retval None

\*/

void HAL\_UART\_RxCpltCallback(UART\_HandleTypeDef \*UartHandle)

{

/\* Set transmission flag: trasfer complete\*/

UartReady = SET;

RS485Reg|=RS485\_2\_REC;

RS485\_Receive\_Data(aRxBuffer,RXBUFFERSIZE);

}

/\*\*

\* @brief UART error callbacks

\* @param UartHandle: UART handle

\* @note This example shows a simple way to report transfer error, and you can

\* add your own implementation.

\* @retval None

\*/

void HAL\_UART\_ErrorCallback(UART\_HandleTypeDef \*UartHandle)

{

Error\_Handler();

}

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @param None

\* @retval None

\*/

static void Error\_Handler(void)

{

/\* Turn LED2 on \*/

BSP\_LED\_On(LED2);

while(1)

{

/\* Error if LED2 is slowly blinking (1 sec. period) \*/

BSP\_LED\_Toggle(LED2);

HAL\_Delay(1000);

}

}

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

void assert\_failed(uint8\_t\* file, uint32\_t line)

{

/\* User can add his own implementation to report the file name and line number,

ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* Infinite loop \*/

while (1)

{

}

}

#endif

# OLED驱动

#include "oled.h"

#include "stdlib.h"

#include "oledfont.h"

#include "delay.h"

//[0]0 1 2 3 ... 127

//[1]0 1 2 3 ... 127

//[2]0 1 2 3 ... 127

//[3]0 1 2 3 ... 127

//[4]0 1 2 3 ... 127

//[5]0 1 2 3 ... 127

//[6]0 1 2 3 ... 127

//[7]0 1 2 3 ... 127

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//IIC Start

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//IIC Start

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void IIC\_Start(void)

{

OLED\_SCLK\_Set() ;

OLED\_SDIN\_Set();

OLED\_SDIN\_Clr();

OLED\_SCLK\_Clr();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//IIC Stop

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void IIC\_Stop(void)

{

OLED\_SCLK\_Set() ;

// OLED\_SCLK\_Clr();

OLED\_SDIN\_Clr();

OLED\_SDIN\_Set();

}

void IIC\_Wait\_Ack(void)

{

//GPIOB->CRH &= 0XFFF0FFFF;

//GPIOB->CRH |= 0x00080000;

// OLED\_SDA = 1;

// delay\_us(1);

//OLED\_SCL = 1;

//delay\_us(50000);

/\* while(1)

{

if(!OLED\_SDA)

{

//GPIOB->CRH &= 0XFFF0FFFF;

//GPIOB->CRH |= 0x00030000;

return;

}

}

\*/

OLED\_SCLK\_Set() ;

OLED\_SCLK\_Clr();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// IIC Write byte

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Write\_IIC\_Byte(unsigned char IIC\_Byte)

{

unsigned char i;

unsigned char m,da;

da=IIC\_Byte;

OLED\_SCLK\_Clr();

for(i=0;i<8;i++)

{

m=da;

// OLED\_SCLK\_Clr();

m=m&0x80;

if(m==0x80)

{OLED\_SDIN\_Set();}

else OLED\_SDIN\_Clr();

da=da<<1;

OLED\_SCLK\_Set();

OLED\_SCLK\_Clr();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// IIC Write Command

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Write\_IIC\_Command(unsigned char IIC\_Command)

{

IIC\_Start();

Write\_IIC\_Byte(0x78); //Slave address,SA0=0

IIC\_Wait\_Ack();

Write\_IIC\_Byte(0x00); //write command

IIC\_Wait\_Ack();

Write\_IIC\_Byte(IIC\_Command);

IIC\_Wait\_Ack();

IIC\_Stop();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// IIC Write Data

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Write\_IIC\_Data(unsigned char IIC\_Data)

{

IIC\_Start();

Write\_IIC\_Byte(0x78); //D/C#=0; R/W#=0

IIC\_Wait\_Ack();

Write\_IIC\_Byte(0x40); //write data

IIC\_Wait\_Ack();

Write\_IIC\_Byte(IIC\_Data);

IIC\_Wait\_Ack();

IIC\_Stop();

}

void OLED\_WR\_Byte(unsigned dat,unsigned cmd)

{

if(cmd)

{

Write\_IIC\_Data(dat);

}

else {

Write\_IIC\_Command(dat);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// fill\_Picture

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void fill\_picture(unsigned char fill\_Data)

{

unsigned char m,n;

for(m=0;m<8;m++)

{

OLED\_WR\_Byte(0xb0+m,0); //page0-page1

OLED\_WR\_Byte(0x00,0); //low column start address

OLED\_WR\_Byte(0x10,0); //high column start address

for(n=0;n<128;n++)

{

OLED\_WR\_Byte(fill\_Data,1);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Delay\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_50ms(unsigned int Del\_50ms)

{

unsigned int m;

for(;Del\_50ms>0;Del\_50ms--)

for(m=6245;m>0;m--);

}

void Delay\_1ms(unsigned int Del\_1ms)

{

unsigned char j;

while(Del\_1ms--)

{

for(j=0;j<123;j++);

}

}

void OLED\_Set\_Pos(unsigned char x, unsigned char y)

{ OLED\_WR\_Byte(0xb0+y,OLED\_CMD);

OLED\_WR\_Byte(((x&0xf0)>>4)|0x10,OLED\_CMD);

OLED\_WR\_Byte((x&0x0f),OLED\_CMD);

}

void OLED\_Display\_On(void)

{

OLED\_WR\_Byte(0X8D,OLED\_CMD);

OLED\_WR\_Byte(0X14,OLED\_CMD); //DCDC ON

OLED\_WR\_Byte(0XAF,OLED\_CMD); //DISPLAY ON

}

void OLED\_Display\_Off(void)

{

OLED\_WR\_Byte(0X8D,OLED\_CMD);

OLED\_WR\_Byte(0X10,OLED\_CMD); //DCDC OFF

OLED\_WR\_Byte(0XAE,OLED\_CMD); //DISPLAY OFF

}

void OLED\_Clear(void)

{

u8 i,n;

for(i=0;i<8;i++)

{

OLED\_WR\_Byte (0xb0+i,OLED\_CMD);

OLED\_WR\_Byte (0x00,OLED\_CMD);

OLED\_WR\_Byte (0x10,OLED\_CMD);

for(n=0;n<128;n++)OLED\_WR\_Byte(0,OLED\_DATA);

}

}

void OLED\_On(void)

{

u8 i,n;

for(i=0;i<8;i++)

{

OLED\_WR\_Byte (0xb0+i,OLED\_CMD);

OLED\_WR\_Byte (0x00,OLED\_CMD);

OLED\_WR\_Byte (0x10,OLED\_CMD);

for(n=0;n<128;n++)OLED\_WR\_Byte(1,OLED\_DATA);

}

}

//x:0~127

//y:0~63

void OLED\_ShowChar(u8 x,u8 y,u8 chr,u8 Char\_Size)

{

unsigned char c=0,i=0;

c=chr-' ';

if(x>Max\_Column-1){x=0;y=y+2;}

if(Char\_Size ==16)

{

OLED\_Set\_Pos(x,y);

for(i=0;i<8;i++)

OLED\_WR\_Byte(F8X16[c\*16+i],OLED\_DATA);

OLED\_Set\_Pos(x,y+1);

for(i=0;i<8;i++)

OLED\_WR\_Byte(F8X16[c\*16+i+8],OLED\_DATA);

}

else {

OLED\_Set\_Pos(x,y);

for(i=0;i<6;i++)

OLED\_WR\_Byte(F6x8[c][i],OLED\_DATA);

}

}

u32 oled\_pow(u8 m,u8 n)

{

u32 result=1;

while(n--)result\*=m;

return result;

}

void OLED\_ShowNum(u8 x,u8 y,u32 num,u8 len,u8 size2)

{

u8 t,temp;

u8 enshow=0;

for(t=0;t<len;t++)

{

temp=(num/oled\_pow(10,len-t-1))%10;

if(enshow==0&&t<(len-1))

{

if(temp==0)

{

OLED\_ShowChar(x+(size2/2)\*t,y,' ',size2);

continue;

}else enshow=1;

}

OLED\_ShowChar(x+(size2/2)\*t,y,temp+'0',size2);

}

}

void OLED\_ShowString(u8 x,u8 y,u8 \*chr,u8 Char\_Size)

{

unsigned char j=0;

while (chr[j]!='\0')

{ OLED\_ShowChar(x,y,chr[j],Char\_Size);

x+=8;

if(x>120){x=0;y+=2;}

j++;

}

}

void OLED\_ShowCHinese(u8 x,u8 y,u8 no)

{

u8 t,adder=0;

OLED\_Set\_Pos(x,y);

for(t=0;t<16;t++)

{

OLED\_WR\_Byte(Hzk[2\*no][t],OLED\_DATA);

adder+=1;

}

OLED\_Set\_Pos(x,y+1);

for(t=0;t<16;t++)

{

OLED\_WR\_Byte(Hzk[2\*no+1][t],OLED\_DATA);

adder+=1;

}

}

void OLED\_DrawBMP(unsigned char x0, unsigned char y0,unsigned char x1, unsigned char y1,unsigned char BMP[])

{

unsigned int j=0;

unsigned char x,y;

if(y1%8==0) y=y1/8;

else y=y1/8+1;

for(y=y0;y<y1;y++)

{

OLED\_Set\_Pos(x0,y);

for(x=x0;x<x1;x++)

{

OLED\_WR\_Byte(BMP[j++],OLED\_DATA);

}

}

}

void OLED\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct;

I2Cx\_SDA\_GPIO\_CLK\_ENABLE();

/\* -2- Configure IOs in output push-pull mode to drive external LEDs \*/

GPIO\_InitStruct.Pin = I2Cx\_SCL\_PIN|I2Cx\_SDA\_PIN;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_PULLUP;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_HIGH;

HAL\_GPIO\_Init(I2Cx\_SCL\_GPIO\_PORT, &GPIO\_InitStruct);

OLED\_SCLK\_Set();

OLED\_SDIN\_Set();

delay\_ms(800);

OLED\_WR\_Byte(0xAE,OLED\_CMD);//--display off

OLED\_WR\_Byte(0x00,OLED\_CMD);//---set low column address

OLED\_WR\_Byte(0x10,OLED\_CMD);//---set high column address

OLED\_WR\_Byte(0x40,OLED\_CMD);//--set start line address

OLED\_WR\_Byte(0xB0,OLED\_CMD);//--set page address

OLED\_WR\_Byte(0x81,OLED\_CMD); // contract control

OLED\_WR\_Byte(0xFF,OLED\_CMD);//--128

OLED\_WR\_Byte(0xA1,OLED\_CMD);//set segment remap

OLED\_WR\_Byte(0xA6,OLED\_CMD);//--normal / reverse

OLED\_WR\_Byte(0xA8,OLED\_CMD);//--set multiplex ratio(1 to 64)

OLED\_WR\_Byte(0x3F,OLED\_CMD);//--1/32 duty

OLED\_WR\_Byte(0xC8,OLED\_CMD);//Com scan direction

OLED\_WR\_Byte(0xD3,OLED\_CMD);//-set display offset

OLED\_WR\_Byte(0x00,OLED\_CMD);//

OLED\_WR\_Byte(0xD5,OLED\_CMD);//set osc division

OLED\_WR\_Byte(0x80,OLED\_CMD);//

OLED\_WR\_Byte(0xD8,OLED\_CMD);//set area color mode off

OLED\_WR\_Byte(0x05,OLED\_CMD);//

OLED\_WR\_Byte(0xD9,OLED\_CMD);//Set Pre-Charge Period

OLED\_WR\_Byte(0xF1,OLED\_CMD);//

OLED\_WR\_Byte(0xDA,OLED\_CMD);//set com pin configuartion

OLED\_WR\_Byte(0x12,OLED\_CMD);//

OLED\_WR\_Byte(0xDB,OLED\_CMD);//set Vcomh

OLED\_WR\_Byte(0x30,OLED\_CMD);//

OLED\_WR\_Byte(0x8D,OLED\_CMD);//set charge pump enable

OLED\_WR\_Byte(0x14,OLED\_CMD);//

OLED\_WR\_Byte(0xAF,OLED\_CMD);//--turn on oled panel

}

# CRC16校验

#include "CRC16.h"

static uint8\_t auchCRCHi[] = {

0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,

0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,

0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,

0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,

0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,

0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,

0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,

0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,

0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,

0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,

0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,

0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,

0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,

0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,

0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,

0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,

0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,

0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,

0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,

0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,

0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,

0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,

0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,

0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,

0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,

0x80, 0x41, 0x00, 0xC1, 0x81, 0x40

} ;

static int8\_t auchCRCLo[] = {

0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06,

0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,

0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,

0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,

0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4,

0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,

0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3,

0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,

0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,

0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,

0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED,

0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,

0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60,

0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,

0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,

0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,

0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E,

0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,

0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71,

0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,

0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,

0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,

0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B,

0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,

0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42,

0x43, 0x83, 0x41, 0x81, 0x80, 0x40

} ;

uint16\_t CRC16(uint8\_t \*puchMsg, uint16\_t usDataLen)

{

uint8\_t uchCRCHi = 0xFF ;

uint8\_t uchCRCLo = 0xFF ;

unsigned uIndex ;

while (usDataLen--)

{

uIndex = uchCRCHi ^ \*puchMsg++ ;

uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex] ;

uchCRCLo = auchCRCLo[uIndex] ;

}

return (uchCRCHi << 8 | uchCRCLo) ;

}

# PH表驱动

#include "PHMeter.h"

#include "oled.h"

uint8\_t PHMeterReg=0x00;

uint8\_t PHMeterCMDBuf[PHMETER\_CMDBUF\_SIZE];

uint8\_t PHMeterDataBuf[PHMETER\_DATABUF\_SIZE];

uint8\_t PHMeteErrFC=0x00;

uint8\_t PHMeteErrCODE=0x00;

uint16\_t PHData=0x0000;

uint16\_t TData=0x0000;

uint16\_t ORPData=0x0000;

uint16\_t PHThreshold=0x0000;

uint16\_t TThreshold=0x0000;

uint16\_t ORPThreshold=0x0000;

uint16\_t PHMeterCode=0x0000;

void PHMeterErrorHandle(void){

OLED\_ShowString(PHMETER\_DISP\_TITAL\_X,PHMETER\_DISP\_TITAL\_Y,"PHERR",16);

}

void PHMeterTOHandle(void){

OLED\_ShowString(PHMETER\_DISP\_TITAL\_X,PHMETER\_DISP\_TITAL\_Y,"PHTO",16);

}

void PHMeterCRCHandle(void){

OLED\_ShowString(PHMETER\_DISP\_TITAL\_X,PHMETER\_DISP\_TITAL\_Y,"PHCRC",16);

}

void PHMeterDisplay(void){

OLED\_ShowString(PHMETER\_DISP\_TITAL\_X,PHMETER\_DISP\_TITAL\_Y,"PHMeter",16);

OLED\_ShowString(PHMETER\_DISP\_ITEM\_X,PHMETER\_DISP\_ITEM\_Y,"ITEM",15);

OLED\_ShowString(PHMETER\_DISP\_VALUE\_X,PHMETER\_DISP\_VALUE\_Y,"VAL",15);

OLED\_ShowString(PHMETER\_DISP\_THRESHOLD\_X,PHMETER\_DISP\_THRESHOLD\_Y,"THR",15);

OLED\_ShowString(PHMETER\_DISP\_PH\_X,PHMETER\_DISP\_PH\_Y," PH",15);

OLED\_ShowString(PHMETER\_DISP\_T\_X,PHMETER\_DISP\_T\_Y," T",15);

OLED\_ShowString(PHMETER\_DISP\_ORP\_X,PHMETER\_DISP\_ORP\_Y,"ORP",15);

OLED\_ShowNum(PHMETER\_DISP\_CODE\_X,PHMETER\_DISP\_CODE\_Y,PHMeterCode,2,16);

OLED\_ShowNum(PHMETER\_DISP\_PHVALUE\_X,PHMETER\_DISP\_PHVALUE\_Y,PHData/100,2,15);

OLED\_ShowString(PHMETER\_DISP\_PHVALUE\_X+2\*6+1,PHMETER\_DISP\_PHVALUE\_Y,".",15);

OLED\_ShowNum(PHMETER\_DISP\_PHVALUE\_X+2\*6+6,PHMETER\_DISP\_PHVALUE\_Y,PHData%100/10,1,15);

OLED\_ShowNum(PHMETER\_DISP\_PHVALUE\_X+3\*6+6,PHMETER\_DISP\_PHVALUE\_Y,PHData%10,1,15);

OLED\_ShowNum(PHMETER\_DISP\_TVALUE\_X,PHMETER\_DISP\_TVALUE\_Y,TData/10,2,15);

OLED\_ShowString(PHMETER\_DISP\_PHVALUE\_X+2\*6+1,PHMETER\_DISP\_TVALUE\_Y,".",15);

OLED\_ShowNum(PHMETER\_DISP\_TVALUE\_X+2\*6+6,PHMETER\_DISP\_TVALUE\_Y,TData%10,1,15);

OLED\_ShowNum(PHMETER\_DISP\_ORPVALUE\_X,PHMETER\_DISP\_ORPVALUE\_Y,ORPData,4,15);

OLED\_ShowNum(PHMETER\_DISP\_PHTHRESHOLD\_X,PHMETER\_DISP\_PHTHRESHOLD\_Y,PHThreshold,4,15);

OLED\_ShowNum(PHMETER\_DISP\_TTHRESHOLD\_X,PHMETER\_DISP\_TTHRESHOLD\_Y,TThreshold,4,15);

OLED\_ShowNum(PHMETER\_DISP\_ORPTHRESHOLD\_X,PHMETER\_DISP\_ORPTHRESHOLD\_Y,ORPThreshold,4,15);

}

void PHMeterRequestData(void){

uint16\_t CRC16DATA=0X0000;

PHMeterCMDBuf[PHMETER\_ADDR\_OFF] =PHMeterAddr;

PHMeterCMDBuf[PHMETER\_FC\_OFF] =PHMETER\_FC;

CRC16DATA=CRC16(PHMeterCMDBuf,PHMETER\_CRCLEN\_OFF);

PHMeterCMDBuf[PHMETER\_CRCLEN\_OFF] =CRC16DATA>>8;

PHMeterCMDBuf[PHMETER\_CRCLEN\_OFF+1] =CRC16DATA&0xff;

RS485\_Send\_Data(PHMeterCMDBuf,PHMETER\_CMDBUF\_SIZE);

}

void PHMeterRequestPH(void){

/\* uint8\_t Temp=0x00;

while(PHMeterReg){

Temp++;

if(Temp>>PHMeterTO){

PHMeterTOHandle();

return ;

}

}\*/

PHMeterCMDBuf[PHMETER\_OFFSET\_OFF+1] =PHMETER\_PH\_OFFSET&0xff;

PHMeterCMDBuf[PHMETER\_OFFSET\_OFF] =PHMETER\_PH\_OFFSET>>8;

PHMeterCMDBuf[PHMETER\_DATACNT\_OFF+1]=PHMETER\_PH\_DATACNT&0xff;

PHMeterCMDBuf[PHMETER\_DATACNT\_OFF]=PHMETER\_PH\_DATACNT>>8;

PHMeterReg|=PHMETER\_PH\_PEND;

PHMeterRequestData();

}

void PHMeterRequestT(void){

/\* uint8\_t Temp=0x00;

while(PHMeterReg){

Temp++;

if(Temp>>PHMeterTO){

PHMeterTOHandle();

return ;

}

}\*/

PHMeterCMDBuf[PHMETER\_OFFSET\_OFF+1] =PHMETER\_T\_OFFSET&0xff;

PHMeterCMDBuf[PHMETER\_OFFSET\_OFF] =PHMETER\_T\_OFFSET>>8;

PHMeterCMDBuf[PHMETER\_DATACNT\_OFF+1] =PHMETER\_T\_DATACNT&0xff;

PHMeterCMDBuf[PHMETER\_DATACNT\_OFF] =PHMETER\_T\_DATACNT>>8;

PHMeterReg|=PHMETER\_T\_PEND;

PHMeterRequestData();

}

void PHMeterRequestPHT(void){

/\* uint8\_t Temp=0x00;

while(PHMeterReg){

Temp++;

if(Temp>>PHMeterTO){

PHMeterTOHandle();

return ;

}

}\*/

PHMeterCMDBuf[PHMETER\_OFFSET\_OFF+1] =PHMETER\_PHT\_OFFSET&0xff;

PHMeterCMDBuf[PHMETER\_OFFSET\_OFF] =PHMETER\_PHT\_OFFSET>>8;

PHMeterCMDBuf[PHMETER\_DATACNT\_OFF+1] =PHMETER\_PHT\_DATACNT&0xff;

PHMeterCMDBuf[PHMETER\_DATACNT\_OFF] =(PHMETER\_PHT\_DATACNT)>>8;

PHMeterReg|=PHMETER\_PHT\_PEND;

PHMeterRequestData();

}

void PHMeterRequestORP(void){

/\* uint8\_t Temp=0x00;

while(PHMeterReg){

Temp++;

if(Temp>>PHMeterTO){

PHMeterTOHandle();

return ;

}

}\*/

PHMeterCMDBuf[PHMETER\_OFFSET\_OFF+1] =PHMETER\_ORP\_OFFSET&0xff;

PHMeterCMDBuf[PHMETER\_OFFSET\_OFF] =PHMETER\_ORP\_OFFSET>>8;

PHMeterCMDBuf[PHMETER\_DATACNT\_OFF+1] =PHMETER\_ORP\_DATACNT&0xff;

PHMeterCMDBuf[PHMETER\_DATACNT\_OFF] =PHMETER\_ORP\_DATACNT>>8;

PHMeterReg|=PHMETER\_ORP\_PEND;

PHMeterRequestData();

}

void PHMeterRequestORPT(void){

/\* uint8\_t Temp=0x00;

while(PHMeterReg){

Temp++;

if(Temp>>PHMeterTO){

PHMeterTOHandle();

return ;

}

}\*/

PHMeterCMDBuf[PHMETER\_OFFSET\_OFF+1] =PHMETER\_TORP\_OFFSET&0xff;

PHMeterCMDBuf[PHMETER\_OFFSET\_OFF] =PHMETER\_TORP\_OFFSET>>8;

PHMeterCMDBuf[PHMETER\_DATACNT\_OFF+1] =PHMETER\_TORP\_DATACNT&0xff;

PHMeterCMDBuf[PHMETER\_DATACNT\_OFF] =(PHMETER\_TORP\_DATACNT)>>8;

PHMeterReg|=PHMETER\_ORPT\_PEND;

PHMeterRequestData();

}

void PHMeterErrReceiveHandle(void){

uint8\_t Temp=0x00;

uint16\_t rCRC=0x0000; //readCRC

uint16\_t cCRC=0x0000; //calculateCRC

Temp=PHMeterReg&(~PHMETER\_RBUF\_UPDATE);

rCRC=PHMeterDataBuf[PHMETER\_ERRCRC\_OFF]<<8|PHMeterDataBuf[PHMETER\_ERRCRC\_OFF+1];

cCRC=CRC16(PHMeterDataBuf,PHMETER\_ERRCRC\_OFF);

if(rCRC!=cCRC){

PHMeterCRCHandle();//CRCERR

return ;

}

if(PHMeterDataBuf[PHMETER\_ADDR\_OFF]!=PHMeterAddr){

PHMeterErrorHandle();

}

PHMeteErrFC=PHMeterDataBuf[PHMETER\_FC\_OFF]&~0x80;

PHMeteErrCODE=PHMeterDataBuf[PHMETER\_ERRCRC\_OFF];

OLED\_Clear();

OLED\_ShowString(PHMETER\_DISP\_TITAL\_X,PHMETER\_DISP\_TITAL\_Y,"PHMeter",16); //L1

OLED\_ShowString(PHMETER\_DISP\_CODE\_X,PHMETER\_DISP\_CODE\_Y,"Error",16); //error

OLED\_ShowString(PHMETER\_DISP\_PH\_X,PHMETER\_DISP\_PH\_Y," ErrFC",15);

OLED\_ShowString(PHMETER\_DISP\_T\_X,PHMETER\_DISP\_T\_Y,"ErrCODE",15);

OLED\_ShowNum(PHMETER\_DISP\_PHVALUE\_X,PHMETER\_DISP\_PHVALUE\_Y,PHMeteErrFC,4,15);

OLED\_ShowNum(PHMETER\_DISP\_TVALUE\_X,PHMETER\_DISP\_TVALUE\_Y,PHMeteErrCODE,4,15);

switch (Temp){

case PHMETER\_PH\_PEND:

PHMeterReg&=(~PHMETER\_RBUF\_UPDATE);

PHMeterReg&=(~PHMETER\_PH\_PEND);

PHMeterRequestPH();

break;

case PHMETER\_T\_PEND:

PHMeterReg&=(~PHMETER\_RBUF\_UPDATE);

PHMeterReg&=(~PHMETER\_T\_PEND);

PHMeterRequestT();

break;

case PHMETER\_PHT\_PEND:

PHMeterReg&=(~PHMETER\_RBUF\_UPDATE);

PHMeterReg&=(~PHMETER\_PH\_PEND);

PHMeterReg&=(~PHMETER\_T\_PEND);

PHMeterRequestPHT();

break;

case PHMETER\_ORP\_PEND:

PHMeterReg&=(~PHMETER\_RBUF\_UPDATE);

PHMeterReg&=(~PHMETER\_ORP\_PEND);

PHMeterRequestORP();

break;

case PHMETER\_ORPT\_PEND:

PHMeterReg&=(~PHMETER\_RBUF\_UPDATE);

PHMeterReg&=(~PHMETER\_ORP\_PEND);

PHMeterReg&=(~PHMETER\_T\_PEND);

PHMeterRequestORPT();

break;

default: ;

}

PHMeterReg=0x00;

}

void PHMeterDataReceiveHandle(void){

uint8\_t Temp=0x00;

uint16\_t CRCData=0x0000;

Temp=PHMeterReg&(~PHMETER\_RBUF\_UPDATE);

if(PHMeterDataBuf[PHMETER\_ADDR\_OFF]!=PHMeterAddr){

PHMeterErrorHandle();

}

if(PHMeterDataBuf[PHMETER\_FC\_OFF]!=PHMETER\_FC){

PHMeterErrorHandle();

}

switch (Temp){

case PHMETER\_PH\_PEND:

CRCData=PHMeterDataBuf[PHMETER\_SCRCLEN\_OFF]<<8|PHMeterDataBuf[PHMETER\_SCRCLEN\_OFF+1];

if(PHMeterDataBuf[PHMETER\_DATALEN\_OFF]!=PHMETER\_PH\_DATALEN){

PHMeterErrorHandle();

}

if(CRCData!=CRC16(PHMeterDataBuf,PHMETER\_SCRCLEN\_OFF)){

PHMeterCRCHandle();

}

PHData=PHMeterDataBuf[PHMETER\_DATA\_OFF]<<8|PHMeterDataBuf[PHMETER\_DATA\_OFF+1];

PHMeterReg&=(~PHMETER\_RBUF\_UPDATE);

PHMeterReg&=(~PHMETER\_PH\_PEND);

break;

case PHMETER\_T\_PEND:

CRCData=PHMeterDataBuf[PHMETER\_SCRCLEN\_OFF]<<8|PHMeterDataBuf[PHMETER\_SCRCLEN\_OFF+1];

if(PHMeterDataBuf[PHMETER\_DATALEN\_OFF]!=PHMETER\_T\_DATALEN){

PHMeterErrorHandle();

}

if(CRCData!=CRC16(PHMeterDataBuf,PHMETER\_SCRCLEN\_OFF)){

PHMeterCRCHandle();

}

TData=PHMeterDataBuf[PHMETER\_DATA\_OFF]<<8|PHMeterDataBuf[PHMETER\_DATA\_OFF+1];

PHMeterReg&=(~PHMETER\_RBUF\_UPDATE);

PHMeterReg&=(~PHMETER\_T\_PEND);

break;

case PHMETER\_PHT\_PEND: CRCData=PHMeterDataBuf[PHMETER\_DCRCLEN\_OFF]<<8|PHMeterDataBuf[PHMETER\_DCRCLEN\_OFF+1];

if(PHMeterDataBuf[PHMETER\_DATALEN\_OFF]!=PHMETER\_PHT\_DATALEN){

PHMeterErrorHandle();

}

if(CRCData!=CRC16(PHMeterDataBuf,PHMETER\_DCRCLEN\_OFF)){

PHMeterCRCHandle();

} PHData=PHMeterDataBuf[PHMETER\_DATA\_OFF]<<8|PHMeterDataBuf[PHMETER\_DATA\_OFF+1];

TData=PHMeterDataBuf[PHMETER\_DATA\_OFF+2]<<8|PHMeterDataBuf[PHMETER\_DATA\_OFF+3];

PHMeterReg&=(~PHMETER\_RBUF\_UPDATE);

PHMeterReg&=(~PHMETER\_PH\_PEND);

PHMeterReg&=(~PHMETER\_T\_PEND);

break;

case PHMETER\_ORP\_PEND: CRCData=PHMeterDataBuf[PHMETER\_SCRCLEN\_OFF]<<8|PHMeterDataBuf[PHMETER\_SCRCLEN\_OFF+1];

if(PHMeterDataBuf[PHMETER\_DATALEN\_OFF]!=PHMETER\_ORP\_DATALEN){

PHMeterErrorHandle();

}

if(CRCData!=CRC16(PHMeterDataBuf,PHMETER\_SCRCLEN\_OFF)){

PHMeterCRCHandle();

}

ORPData=PHMeterDataBuf[PHMETER\_DATA\_OFF]<<8|PHMeterDataBuf[PHMETER\_DATA\_OFF+1];

PHMeterReg&=(~PHMETER\_RBUF\_UPDATE);

PHMeterReg&=(~PHMETER\_ORP\_PEND);

break;

case PHMETER\_ORPT\_PEND:

CRCData=PHMeterDataBuf[PHMETER\_DCRCLEN\_OFF]<<8|PHMeterDataBuf[PHMETER\_DCRCLEN\_OFF+1];

if(PHMeterDataBuf[PHMETER\_DATALEN\_OFF]!=PHMETER\_TORP\_DATALEN){

PHMeterErrorHandle();

}

if(CRCData!=CRC16(PHMeterDataBuf,PHMETER\_DCRCLEN\_OFF)){

PHMeterCRCHandle();

}

TData=PHMeterDataBuf[PHMETER\_DATA\_OFF]<<8|PHMeterDataBuf[PHMETER\_DATA\_OFF+1];

ORPData=PHMeterDataBuf[PHMETER\_DATA\_OFF+2]<<8|PHMeterDataBuf[PHMETER\_DATA\_OFF+3];

PHMeterReg&=(~PHMETER\_RBUF\_UPDATE);

PHMeterReg&=(~PHMETER\_ORP\_PEND);

PHMeterReg&=(~PHMETER\_T\_PEND);

break;

default: ;

}

PHMeterReg=0x00;

}

void PHMeterReceiveHandle(void){

if(PHMeterDataBuf[PHMETER\_FC\_OFF]&0x80){

PHMeterErrReceiveHandle();

}else{

PHMeterDataReceiveHandle();

}

}

void PHMeterCheck(void){

if(PHMeterReg&PHMETER\_RBUF\_UPDATE){

PHMeterReceiveHandle();

PHMeterReg&=~PHMETER\_RBUF\_UPDATE;

}

}

# 溶氧仪驱动

#include "DissolvedOxygenMeter.h"

#include "oled.h"

uint8\_t DOMeterReg=0x00;

uint8\_t DOMeterCMDBuf[DOMETER\_CMDBUF\_SIZE];

uint8\_t DOMeterDataBuf[DOMETER\_DATABUF\_SIZE];

uint8\_t DOMeterErrFC=0x00;

uint8\_t DOMeterErrCODE=0x00;

uint16\_t DOData=0x0000;

uint16\_t DOTData=0x0000;

uint16\_t HALMData=0x0000;

uint16\_t LALMData=0x0000;

uint16\_t LTCData=0x0000;

uint16\_t STAData=0x0000;

uint16\_t setHALM=0xffff;

uint16\_t setLALM=0x0000;

uint16\_t setLTC=0x00ff;

void DOMeterErrorHandle(void){

OLED\_ShowString(DOMETER\_DISP\_TITAL\_X,DOMETER\_DISP\_TITAL\_Y,"DOERR",16); //L1

}

void DOMeterCRCHandle(void){

OLED\_ShowString(DOMETER\_DISP\_TITAL\_X,DOMETER\_DISP\_TITAL\_Y,"DOCRC",16); //L1

}

void DOMeterTOHandle(void){

OLED\_ShowString(DOMETER\_DISP\_TITAL\_X,DOMETER\_DISP\_TITAL\_Y,"DOTO",16); //L1

}

void DOMeterDisplay(void){

OLED\_ShowString(DOMETER\_DISP\_TITAL\_X,DOMETER\_DISP\_TITAL\_Y,"DOMeter",16); //L1

OLED\_ShowNum(DOMETER\_DISP\_CODE\_X,DOMETER\_DISP\_CODE\_Y,STAData,2,16); //STA

OLED\_ShowString(DOMETER\_DISP\_ITEM\_X,DOMETER\_DISP\_ITEM\_Y,"ITEM",15); //L2

OLED\_ShowString(DOMETER\_DISP\_MIN\_X,DOMETER\_DISP\_MIN\_Y,"MIN",15);

OLED\_ShowString(DOMETER\_DISP\_VALUE\_X,DOMETER\_DISP\_VALUE\_Y,"VAL",15);

OLED\_ShowString(DOMETER\_DISP\_MAX\_X,DOMETER\_DISP\_MAX\_Y,"MAX",15);

OLED\_ShowString(DOMETER\_DISP\_DO\_X,DOMETER\_DISP\_DO\_Y," DO",15); //ITEM

OLED\_ShowString(DOMETER\_DISP\_T\_X,DOMETER\_DISP\_T\_Y," T",15);

OLED\_ShowString(DOMETER\_DISP\_LTC\_X,DOMETER\_DISP\_LTC\_Y,"LTC",15);

OLED\_ShowNum(DOMETER\_DISP\_DOMIN\_X,DOMETER\_DISP\_DOMIN\_Y,LALMData/10,2,15); //MIN

OLED\_ShowString(DOMETER\_DISP\_DOMIN\_X+2\*6+1,DOMETER\_DISP\_DOMIN\_Y,".",15);

OLED\_ShowNum(DOMETER\_DISP\_DOMIN\_X+3\*6,DOMETER\_DISP\_DOMIN\_Y,LALMData%10,1,15);

OLED\_ShowNum(DOMETER\_DISP\_DOVALUE\_X,DOMETER\_DISP\_DOVALUE\_Y,DOData/1000,1,15); //VALUE

OLED\_ShowString(DOMETER\_DISP\_DOVALUE\_X+6+1,DOMETER\_DISP\_DOVALUE\_Y,".",15);

OLED\_ShowNum(DOMETER\_DISP\_DOVALUE\_X+2\*6,DOMETER\_DISP\_DOVALUE\_Y,DOData%1000/100,1,15);

OLED\_ShowNum(DOMETER\_DISP\_DOVALUE\_X+3\*6,DOMETER\_DISP\_DOVALUE\_Y,DOData%100/10,1,15);

OLED\_ShowNum(DOMETER\_DISP\_DOVALUE\_X+4\*6,DOMETER\_DISP\_DOVALUE\_Y,DOData%10,1,15);

OLED\_ShowNum(DOMETER\_DISP\_TVALUE\_X,DOMETER\_DISP\_TVALUE\_Y,DOTData/100,2,15);

OLED\_ShowString(DOMETER\_DISP\_TVALUE\_X+2\*6+1,DOMETER\_DISP\_TVALUE\_Y,".",15);

OLED\_ShowNum(DOMETER\_DISP\_TVALUE\_X+3\*6,DOMETER\_DISP\_TVALUE\_Y,DOTData%100/10,1,15);

OLED\_ShowNum(DOMETER\_DISP\_TVALUE\_X+4\*6,DOMETER\_DISP\_TVALUE\_Y,DOTData/10,1,15);

OLED\_ShowNum(DOMETER\_DISP\_LTCVALUE\_X,DOMETER\_DISP\_LTCVALUE\_Y,LTCData/10,2,15);

OLED\_ShowString(DOMETER\_DISP\_LTCVALUE\_X+2\*6+1,DOMETER\_DISP\_LTCVALUE\_Y,".",15);

OLED\_ShowNum(DOMETER\_DISP\_LTCVALUE\_X+3\*6,DOMETER\_DISP\_LTCVALUE\_Y,LTCData%10,1,15);

OLED\_ShowNum(DOMETER\_DISP\_DOMAX\_X,DOMETER\_DISP\_DOMAX\_Y,HALMData/10,2,15); //MAX

OLED\_ShowString(DOMETER\_DISP\_DOMAX\_X+2\*6+1,DOMETER\_DISP\_DOMAX\_Y,".",15);

OLED\_ShowNum(DOMETER\_DISP\_DOMAX\_X+3\*6,DOMETER\_DISP\_DOMAX\_Y,HALMData%10,1,15);

}

void DOMeterRequestData(void){

uint16\_t CRCDATA=0X0000;

DOMeterCMDBuf[DOMETER\_ADDR\_OFF] =DOMeterAddr;

DOMeterCMDBuf[DOMETER\_FC\_OFF] =DOMETER\_REQ\_FC;

DOMeterCMDBuf[DOMETER\_START\_OFF] =0X00;

DOMeterCMDBuf[DOMETER\_START\_OFF+1] =0X00;

DOMeterCMDBuf[DOMETER\_REGCNT\_OFF] =0X00;

DOMeterCMDBuf[DOMETER\_REGCNT\_OFF+1] =0X06;

CRCDATA=CRC16(DOMeterCMDBuf,DOMETER\_REQCRC\_OFF);

DOMeterCMDBuf[DOMETER\_REQCRC\_OFF+1] =CRCDATA&0XFF;

DOMeterCMDBuf[DOMETER\_REQCRC\_OFF] =CRCDATA>>8;

RS485\_Send\_Data(DOMeterCMDBuf,DOMETER\_REQCRC\_OFF+2);

DOMeterReg|=DOMETER\_REQ\_PEND;

}

void DOMeterWriteReg(uint16\_t HALM,uint16\_t LALM,uint16\_t LTC){

uint16\_t CRCDATA=0X0000;

DOMeterCMDBuf[DOMETER\_ADDR\_OFF] =DOMeterAddr;

DOMeterCMDBuf[DOMETER\_FC\_OFF] =DOMETER\_WREG\_FC;

DOMeterCMDBuf[DOMETER\_START\_OFF] =0X00;

DOMeterCMDBuf[DOMETER\_START\_OFF+1] =0X00;

DOMeterCMDBuf[DOMETER\_REGCNT\_OFF] =0X00;

DOMeterCMDBuf[DOMETER\_REGCNT\_OFF+1] =0X03;

DOMeterCMDBuf[DOMETER\_DATANUM\_OFF] =0X06;

DOMeterCMDBuf[DOMETER\_DATANUM\_OFF+1]=HALM>>8;

DOMeterCMDBuf[DOMETER\_DATANUM\_OFF+2]=HALM&0XFF;

DOMeterCMDBuf[DOMETER\_DATANUM\_OFF+3]=LALM>>8;

DOMeterCMDBuf[DOMETER\_DATANUM\_OFF+4]=LALM&0XFF;

DOMeterCMDBuf[DOMETER\_DATANUM\_OFF+5]=LTC>>8;

DOMeterCMDBuf[DOMETER\_DATANUM\_OFF+6]=LTC&0XFF;

CRCDATA=CRC16(DOMeterCMDBuf,DOMETER\_DATANUM\_OFF+7);

DOMeterCMDBuf[DOMETER\_DATANUM\_OFF+7] =CRCDATA>>8;

DOMeterCMDBuf[DOMETER\_DATANUM\_OFF+8] =CRCDATA&0XFF;

RS485\_Send\_Data(DOMeterCMDBuf,DOMETER\_CMDBUF\_SIZE);

DOMeterReg|=DOMETER\_WRITE\_PEND;

}

void DOMeterErrReceiveHandle(void){

uint8\_t Temp=0x00;

uint16\_t rCRC=0x0000; //readCRC

uint16\_t cCRC=0x0000; //calculateCRC

Temp=DOMeterReg&(~DOMETER\_RBUF\_UPDATE);

rCRC=DOMeterDataBuf[DOMETER\_ERRCRC\_OFF]<<8|DOMeterDataBuf[DOMETER\_ERRCRC\_OFF+1];

cCRC=CRC16(DOMeterDataBuf,DOMETER\_ERRCRC\_OFF);

if(rCRC!=cCRC){

DOMeterCRCHandle();//CRCERR

return ;

}

if(DOMeterDataBuf[DOMETER\_ADDR\_OFF]!=DOMeterAddr){

DOMeterErrorHandle(); //Meter addr err

}

DOMeterErrFC=DOMeterDataBuf[DOMETER\_FC\_OFF]&~0x80;

DOMeterErrCODE=DOMeterDataBuf[DOMETER\_CODE\_OFF];

OLED\_Clear();

OLED\_ShowString(DOMETER\_DISP\_TITAL\_X,DOMETER\_DISP\_TITAL\_Y,"DOMeter",16); //L1

OLED\_ShowString(DOMETER\_DISP\_CODE\_X,DOMETER\_DISP\_CODE\_Y,"Error",16); //error

OLED\_ShowString(DOMETER\_DISP\_DO\_X,DOMETER\_DISP\_DO\_Y," ErrFC",15); //ITEM

OLED\_ShowString(DOMETER\_DISP\_T\_X,DOMETER\_DISP\_T\_Y,"ErrCODE",15);

OLED\_ShowNum(DOMETER\_DISP\_DOVALUE\_X,DOMETER\_DISP\_DOVALUE\_Y,DOMeterErrFC,4,15); //VALUE

OLED\_ShowNum(DOMETER\_DISP\_TVALUE\_X,DOMETER\_DISP\_TVALUE\_Y,DOMeterErrCODE,4,15);

switch (Temp){

case DOMETER\_REQ\_PEND:

DOMeterReg&=(~DOMETER\_RBUF\_UPDATE);

DOMeterReg&=(~DOMETER\_REQ\_PEND);

DOMeterRequestData();

break;

case DOMETER\_WRITE\_PEND:

DOMeterReg&=(~DOMETER\_RBUF\_UPDATE);

DOMeterReg&=(~DOMETER\_WRITE\_PEND);

DOMeterWriteReg(setHALM,setLALM,setLTC);

break;

default: ;

}

}

void DOMeterDataReceiveHandle(void){

uint16\_t rCRC=0x0000; //readCRC

uint16\_t cCRC=0x0000; //calculateCRC

if(DOMeterDataBuf[DOMETER\_ADDR\_OFF]!=DOMeterAddr){

DOMeterErrorHandle();

}

switch (DOMeterDataBuf[DOMETER\_FC\_OFF]){

case DOMETER\_REQ\_FC:

rCRC=DOMeterDataBuf[DOMETER\_DATACRC\_OFF]<<8|DOMeterDataBuf[DOMETER\_DATACRC\_OFF+1];

cCRC=CRC16(DOMeterDataBuf,DOMETER\_DATACRC\_OFF);

if(rCRC!=cCRC){

DOMeterCRCHandle();//CRCERR

return ;

}

DOData =DOMeterDataBuf[DOMETER\_DOH\_OFFSET]<<8|DOMeterDataBuf[DOMETER\_DOL\_OFFSET];

DOTData =DOMeterDataBuf[DOMETER\_TH\_OFFSET]<<8|DOMeterDataBuf[DOMETER\_TL\_OFFSET];

HALMData=DOMeterDataBuf[DOMETER\_HALMH\_OFFSET]<<8|DOMeterDataBuf[DOMETER\_HALML\_OFFSET];

LALMData=DOMeterDataBuf[DOMETER\_LALMH\_OFFSET]<<8|DOMeterDataBuf[DOMETER\_LALML\_OFFSET];

LTCData =DOMeterDataBuf[DOMETER\_LTCH\_OFFSET]<<8|DOMeterDataBuf[DOMETER\_LTCL\_OFFSET];

STAData =DOMeterDataBuf[DOMETER\_ALMSTA\_OFFSET];

DOMeterReg&=(~DOMETER\_RBUF\_UPDATE);

DOMeterReg&=(~DOMETER\_REQ\_PEND);

break;

case DOMETER\_WREG\_FC:

rCRC=DOMeterDataBuf[6]<<8|DOMeterDataBuf[6+1];

cCRC=CRC16(DOMeterDataBuf,6);

if(rCRC!=cCRC){

DOMeterCRCHandle();//CRCERR

return ;

}

DOMeterReg&=(~DOMETER\_RBUF\_UPDATE);

DOMeterReg&=(~DOMETER\_WRITE\_PEND);

break;

default: ;

}

}

void DOMeterReceiveHandle(void){

if(DOMeterDataBuf[DOMETER\_FC\_OFF]&0x80){

DOMeterErrReceiveHandle();

}else{

DOMeterDataReceiveHandle();

}

}

void DOMeterCheck(void){

if(DOMeterReg&DOMETER\_RBUF\_UPDATE){

DOMeterReceiveHandle();

DOMeterReg&=~DOMETER\_RBUF\_UPDATE;

}

}