乐升半导体
LEVETOP

UI_Editor-II

主控端串口通讯程序范例

V1.1

www.levetop.cn
Levetop Semiconductor Co., Ltd.

版本记录

版 本	日 期	说 明
V1.0	2023/12/01	初版
V1.1	2024/6/13	修订版

版权说明

目 录

1. 前言

在 UI_Editor-II 的串口协议下，主控端 MCU 必须透过 Uart 通讯接口将数据依照串口指令结构与串口屏进行沟通，而为让主控端 MCU 程序开发者能节省开发时间，本范例提供了一个完整的指令发送程序，将数据写入到指定的变量地址内。

2. 串口屏指令结构

下图为乐升半导体串口屏芯片通讯的指令基本结构：

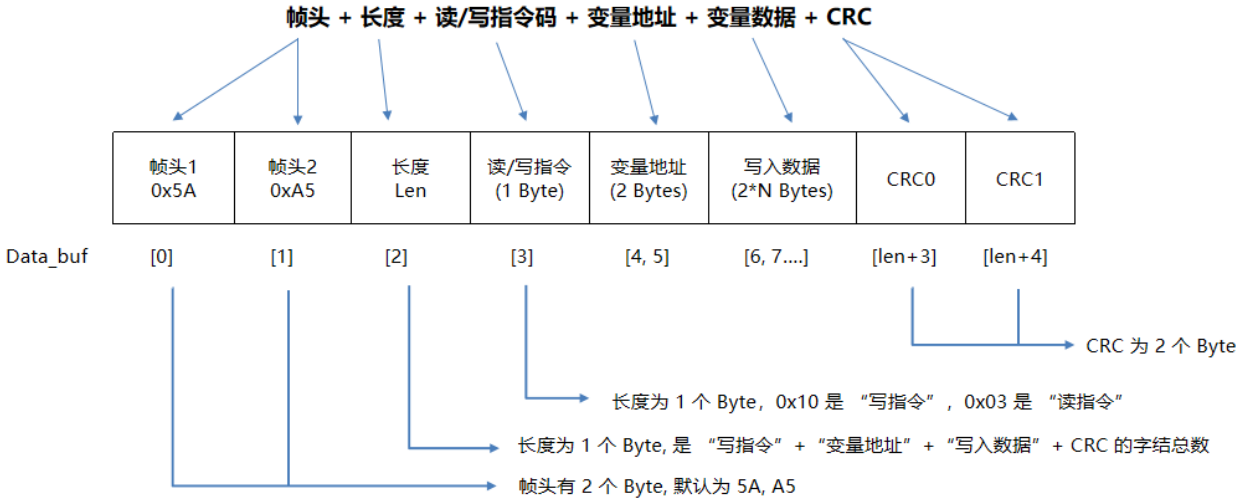

图 1：串口通讯指令结构图

本演范例中使用的主控 MCU 为 STM32F103RCT6，将 STM32F103RCT6 的 PA9、PA10 引脚分别设为 USART1_TX 和 USART1_RX，下图为 MCU 与 LT7689 串口芯片的接线模式。

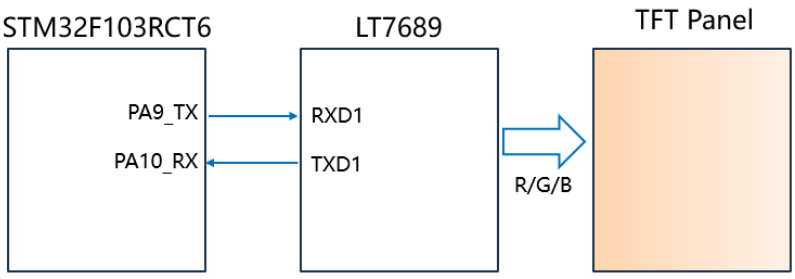

图 2：主控端 MCU (STM32F103RCT6)用串口与 LT7689 串口屏芯片通讯

3. CRC 码的生成

每个串口通讯的结尾都有 2 个 CRC 的校验码，是由读/写指令、变量地址、变量数据及一些参数表的数据所产生，其参考代码（CRC.h）如下：

```
/***** CRC.h *****/

#include "stm32f10x.h"                  // Device header
/* CRC 校验 */
//高位字节的 CRC 值
const 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};

//低位字节的 CRC 值
const char 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};
```

```
unsigned short CRC16(uint8_t *puchMsg,uint16_t usDataLen)
/* 函数以 unsigned short 类型返回 CRC */
{
    uint8_t uchCRCHi = 0xFF ;          // CRC 的高字节初始化
    uint8_t uchCRCLo = 0xFF ;          // CRC 的低字节初始化
    uint16_t uIndex ;                  // CRC 查询表索引
    while (usDataLen--)                // 完成整个报文缓冲区
        {
            uIndex = uchCRCLo ^ *puchMsg++ ;            // 计算 CRC
            uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex] ;   // 通过数组获取进行 CRC 低位
            uchCRCHi = auchCRCLo[uIndex] ;              // 通过数组获取进行 CRC 高位
        }
    return (uchCRCHi << 8 | uchCRCLo);
}
```


4. UART 串口配置

如前节所述，本演范例将使用STM32F103RCT6作为主控MCU，通过数据手册可将STM32F103RCT6的PA9、PA10 引脚分别设为 USART1_TX 和 USART1_RX 引脚。本次演示只进行一写指令操作，因此只需要使用 PA9 引脚与串口屏的 RXD1 引脚进行连接即可实现切换显示页面的操作。UART 串口输出程序代码（Uart.h）如下：

```
/***** Uart.h *****/
#include "sys.h"
#include "usart.h"

void uart_init(u32 bound)
{
    GPIO_InitTypeDef GPIO_InitStructure;            //GPIO 端口设置
    USART_InitTypeDef USART_InitStructure;
    NVIC_InitTypeDef NVIC_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1|RCC_APB2Periph_GPIOA,  ENABLE);  // 使 能 USART1，GPIOA 时钟
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9; //PA.9-USART1_TX
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;   //复用推挽输出
    GPIO_Init(GPIOA, &GPIO_InitStructure);//初始化 GPIOA.9
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;//PA10-USART1_RX
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;//浮空输入
    GPIO_Init(GPIOA, &GPIO_InitStructure);//初始化 GPIOA.10

    //-----------Usart1 NVIC 配置---------------
    NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority=3 ;//抢占优先级 3
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 3;          //子优先级 3
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;             //IRQ 通道使能
    NVIC_Init(&NVIC_InitStructure); //根据指定的参数初始化 VIC 寄存器

    //-----------USART 初始化设置---------------
    USART_InitStructure.USART_BaudRate = bound;//串口波特率
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;//字长为 8 位数据格式
    USART_InitStructure.USART_StopBits = USART_StopBits_1;//一个停止位
    USART_InitStructure.USART_Parity = USART_Parity_No;//无奇偶校验位
    USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;//无硬件数据流控制
    USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx; //收发模式

    USART_Init(USART1, &USART_InitStructure); //初始化串口 1
    //USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);//开启串口接受中断
    USART_Cmd(USART1, ENABLE);                          //使能串口 1
}
```



```
typedef struct
{
    uint8_t Flag;              // Data sending completion flag
    uint8_t Buf[TX_SIZE];
    uint16_t Count;
} USART_TX_INFO;

void STM_SingleByteToPc(uint8_t Data)
{
   while((USART1->SR&0X40)==0){};//循环发送,直到发送完毕
   USART1->DR = Data;
}
```

5. 主函数编写进行指令传输

以下范例为主控端 MCU(STM32F103RCT6) 将变量地址 0x7000 写入 0x0001 数据，实现切换显示页面、将变量地址 0x7001 写入 0x0020 数据，实现调整背光亮度，及修改 RTC 时钟日期，其流程与程序编写如下：

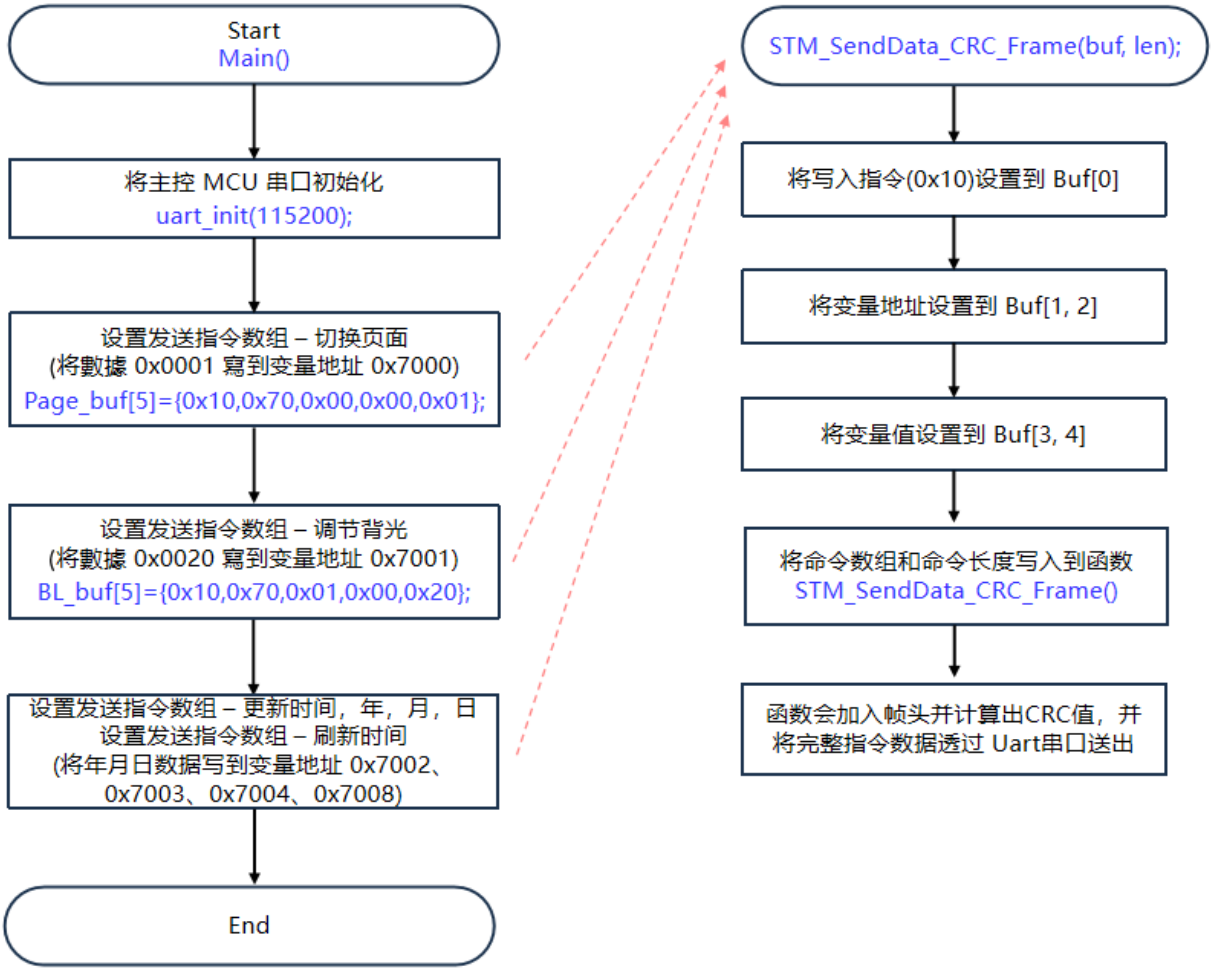

图 3：主控端发送串口指令的流程图

```
/***** main() *****/
#include "sys.h"
#include "delay.h"
#include "usart.h"

#define TX_SIZE (1024+ 2) // Macro used to control USART sending data

uint8_t SCI_C0 = 0x5A;                          // 设置帧头
uint8_t SCI_C1 = 0xA5;
uint8_t CRC_Enable_Flag = 1;                    // CRC 校验标志位

USART_TX_INFO gUsartTx; // For serial port reception
int main(void)
{
    uint8_t Page_buf[5]={0x10,0x70,0x00,0x00,0x01};                         //page1
    uint8_t Icon_buf[5]={0x10,0x00,0x22,0x00,0x01};                             //Icon
    uint8_t BL_buf[5]={0x10,0x70,0x01,0x00,0x20};                           //Backlight set
    uint8_t Time_buf[9]={0x10,0x70,0x02,0x00,0x17,0x00,0x0C,0x00,0x1D};     //TIME: YEAR,MONTH,DAY,
    uint8_t TimeFresh_buf[5]={0x10,0x70,0x08,0x00,0x01};                  //FreshTIME: YEAR,MONTH,DAY,
    uint8_t Number_buf[7]={0x10,0x00,0x24,0x00,0x00,0x30,0x39};             //Number:12345
    uint8_t TXT_buf[17]={0x10,0x00,0x28,0xB9,0xE3,0xB6,0xAB,0xCA,0xA1,
                         0xC9,0xEE,0xDB,0xDA,0xCA,0xD0,0x00,0x00};          //String:广东省深圳市
    SystemInit();                       // STM32 系统时钟初始化
    delay_init();                       // 延时初始化
    uart_init(115200);                  // 串口初始化
    //    printf("start!!!\r\n");
    //-----------切换页面-------------------
   STM_SendData_CRC_Frame(Page_buf, 5);
   delay_ms(1500);
    //-----------刷新 ICON 图标------------
   STM_SendData_CRC_Frame(Icon_buf, 5);
   delay_ms(1500);
    //-----------调节背光亮度--------------
   STM_SendData_CRC_Frame(BL_buf, 5);
   delay_ms(1500);
    //-----------更新时间：年月日---------
    STM_SendData_CRC_Frame(Time_buf, 9);
   STM_SendData_CRC_Frame(TimeFresh_buf, 5);
   delay_ms(1500);
    //-----------更新数字------------------
   STM_SendData_CRC_Frame(Number_buf, 7);
   delay_ms(1500);
    //-----------更新字符串----------------
   STM_SendData_CRC_Frame(TXT_buf, 17);
   delay_ms(1500);
```



```
    while(1)
        {
            delay_ms(100);
        }
}

void STM_SendData_CRC_Frame(uint8_t *buf, uint8_t len)
{
    uint16_t i = 0;
    uint16_t TxToPc_crc;
    uint8_t crc[2] = {0};
    uint8_t rebuf[200+3] = {0};

    *(rebuf + 0) = SCI_C0;
    *(rebuf + 1) = SCI_C1;
    for (i = 0; i < len; i++)
        *(rebuf + 3 + i) =  *(buf + i);
    if (CRC_Enable_Flag)                    //命令带 CRC
    {
        //rebuf[2]=读/写指令(1)+变量地址(2)+数据(X)+CRC(2)
        *(rebuf + 2) = len + 2;
        //CRC 计算不带入帧头和长度
        TxToPc_crc = CRC16(buf, len);    //
        crc[0] = (uint8_t)(TxToPc_crc & 0x00ff);
        crc[1] = (uint8_t)((TxToPc_crc >> 8) & 0x00ff);

        *(rebuf + len + 3) = crc[0];
        *(rebuf + len + 4) = crc[1];
        for (i = 0; i < 5 + len; i++)
        {
            STM_SingleByteToPc(*(rebuf + i));
        }
    }
    else                                    //命令不带 CRC
    {
        //rebuf[2]=读/写指令(1)+变量地址(2)+数据(X)
        *(rebuf + 2) = len;
        for (i = 0; i < 3 + len; i++)
        {
            STM_SingleByteToPc(*(rebuf + i));
        }
    }
}
```


