乐升半导体
LEVETOP

LT269 串口屏演示模块

(M00269-35-0480320-SRX-LT-V12)

使 用 说 明 书

V1.0

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

版本记录

版 本	日 期	说 明
V1.0	2024/5/21	初版
-	-	-

版权说明

目　录

图 附 录

1. 模块基本介绍

1.1. 模块外观

LT269 串口屏演示模块（M00269-35-0480320-SRX-LT-V12）为 3.5”分辨率 480×320 带 RTP 电阻触控屏的串口显示模块，PCB 尺寸为 100.0 * 55.0 mm，其外观如下图：

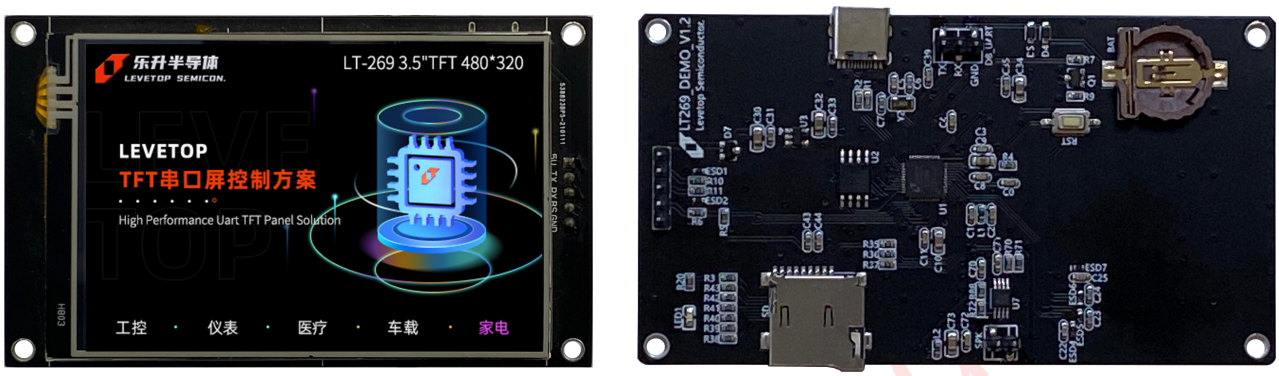

图 1-1：演示模块外观图

主要组件与接口如下所示：

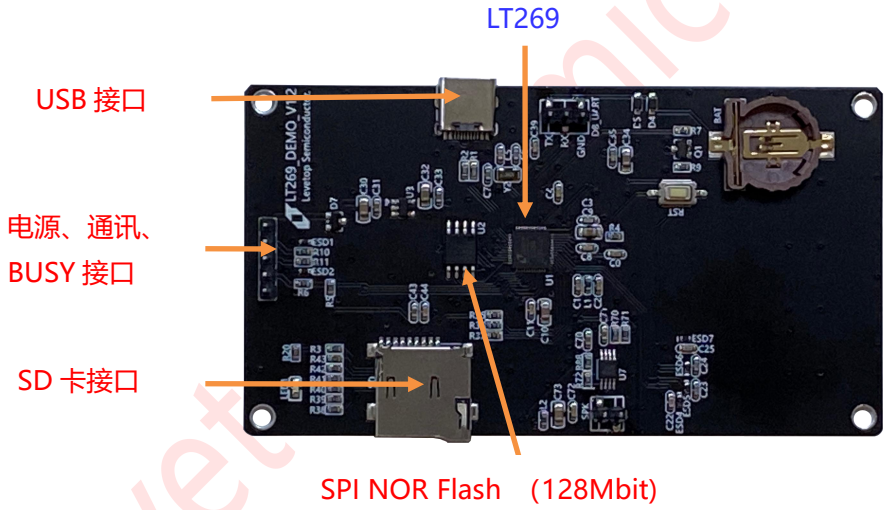

图 1-2：模块主要组件与接口

1.2. 原理图

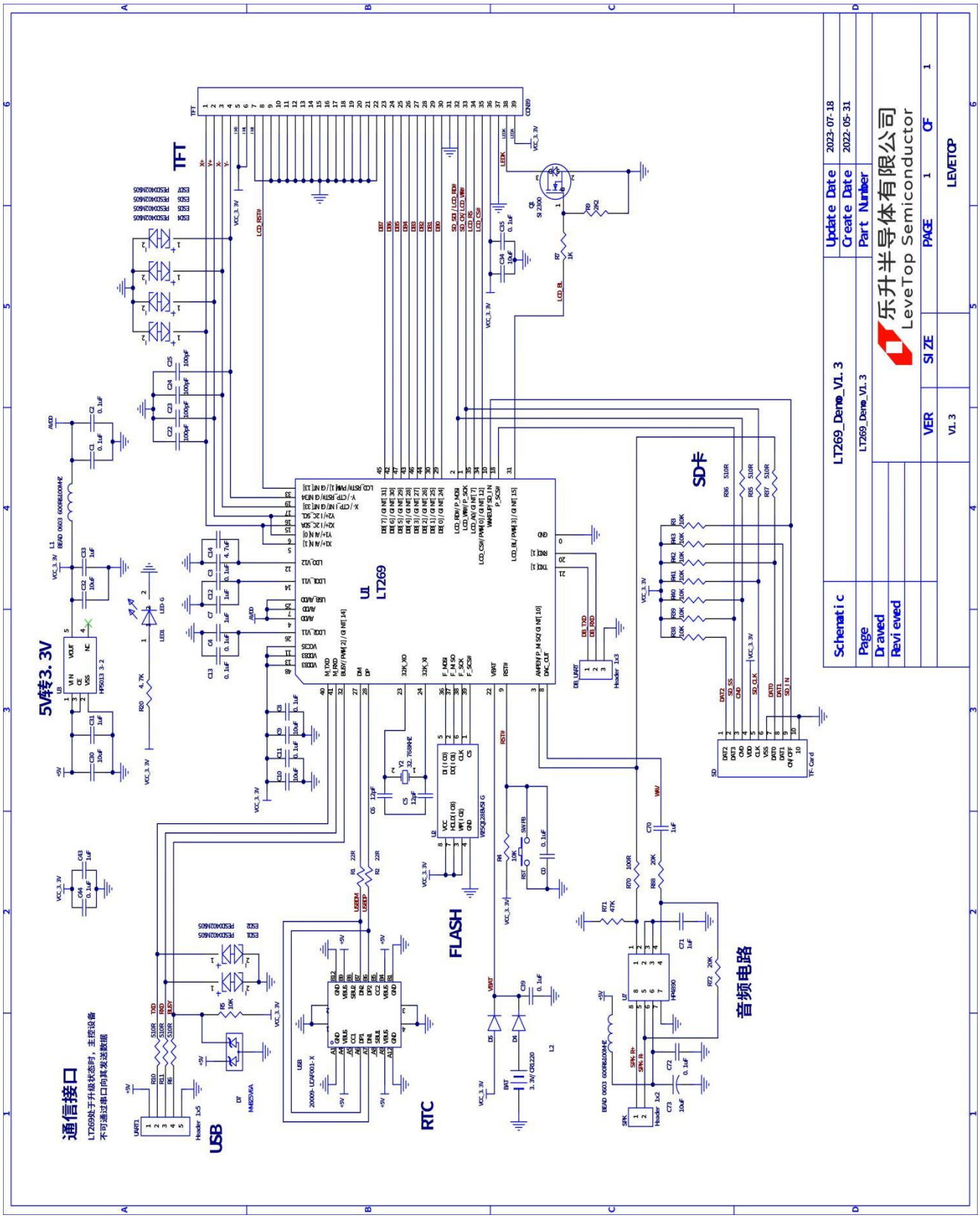

图 1-3：原理图

2. 使用方式

2.1. 上电演示

此 LT269 串口屏演示模块可以直接用 USB 线引入电源直接操作，将带电的 USB 线直接插入 USB 接口就可以看到演示画面，然后根据画面出现的显示 UI 进行触控操作，当然也可以通过“电源、通讯、BUSY 接口”的 VCC 与 GND 引入 5V 电源进行操作。

图 2-1：出厂的 UI 演示画面范例

此 LT269 串口屏演示模块通电后出现图 2-1 画面，因为没有主控通过串口发送讯息到这个模块，所以用触控屏来模拟进行画面的切换，基本演示操作说明如下图 2-2、图 2-3、图 2-4，用户可以按下这些触控区域来观察图标或是画面的变化；详细操作说明也可以到乐升官网的应用视频区观看或是下载（乐升官网→解决方案→应用视频→基本功能展示→LT269 应用-基本功能展示，如图 2-5）。

注意：此 LT269 串口屏演示模块是采用 RTP 电阻触控屏，因此操作时要稍微用力压下才能达到触控效果。

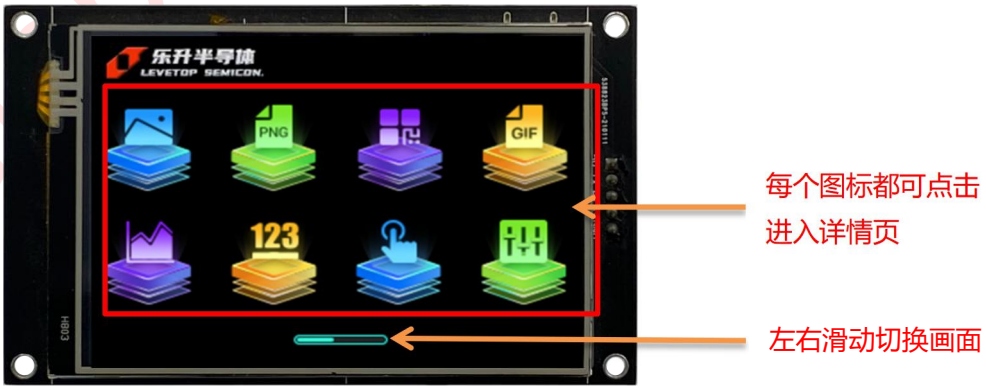

图 2-2：LT269 应用-基本功能展示演示画面 1

图 2-3：LT269 应用-基本功能展示演示画面 2

图 2-4：LT269 应用-基本功能展示演示画面 3

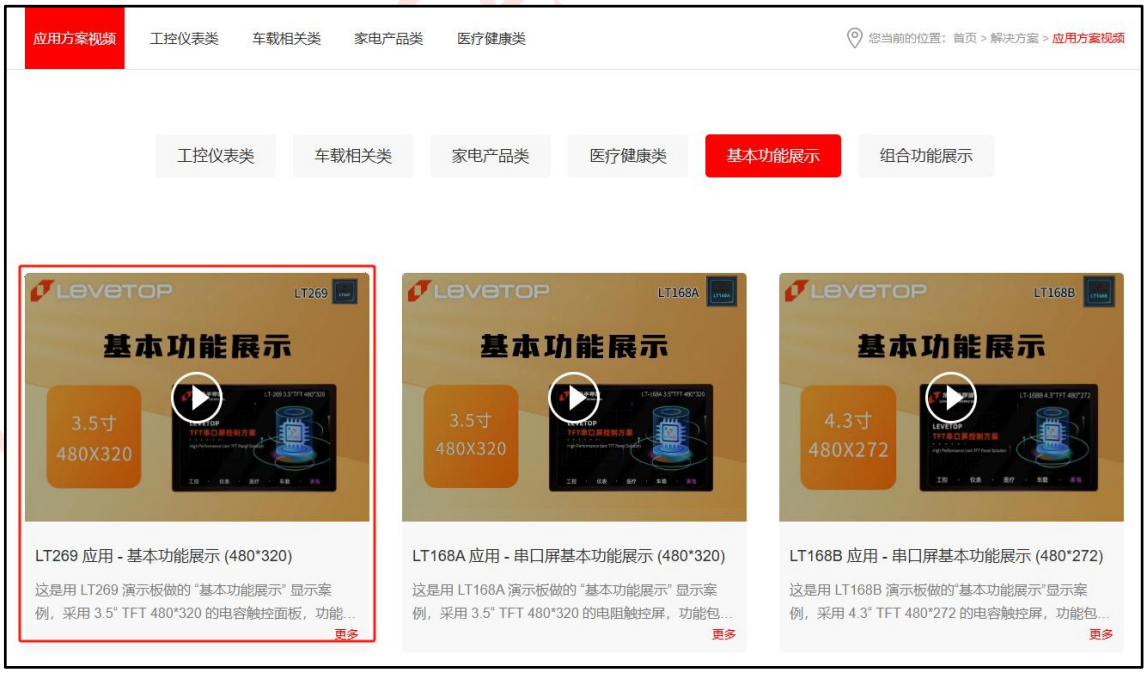

图 2-5：LT269 应用-基本功能展示演示视频官网位置

2.2. 工程下载与更新

上一节提到此 LT269 串口屏演示的工程与用到的软件都可以在深圳市乐升半导体有限公司官网下载专区下载：

图 2-6：官网下载区

2.2.1. 通过 LT_VCOM_GUI 软件更新 LT269

可更新的文件：MCU_Code.bin 和 UartTFT-II_Flash.bin。

LT269 具有 USB 接口，同时内部含有 Bootloader，**LT_VCOM_GUI** 可以让用户通过 USB 线更新 MCU_Code.bin 及 UartTFT_Flash.bin。具体软件可在深圳市乐升半导体有限公司(levetop.cn)官网下载专区下载。

1、接线说明，如下图所示，进行烧录前，**先将标注 1 处的 busy 引脚和 GND 互相连接，然后通过标注 2 处 USB 口连接电脑供电和通信。**

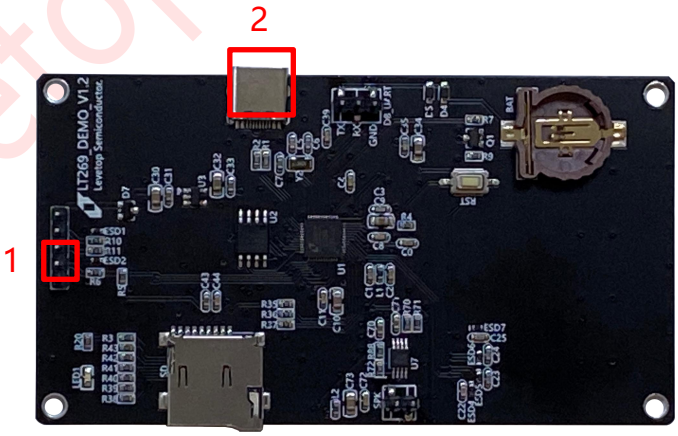

图 2-7：LT269 接线示意图

2、打开电脑软件 **LT_VCOM_GUI_Vxx.exe**，软件会自动辨识是否连接到 LT269 及自动获取通讯串口号，如果没有可以尝试点击 Open Comm。（若选择错误的串口号，无法进行下一步操作，防止选错串口号），如图。

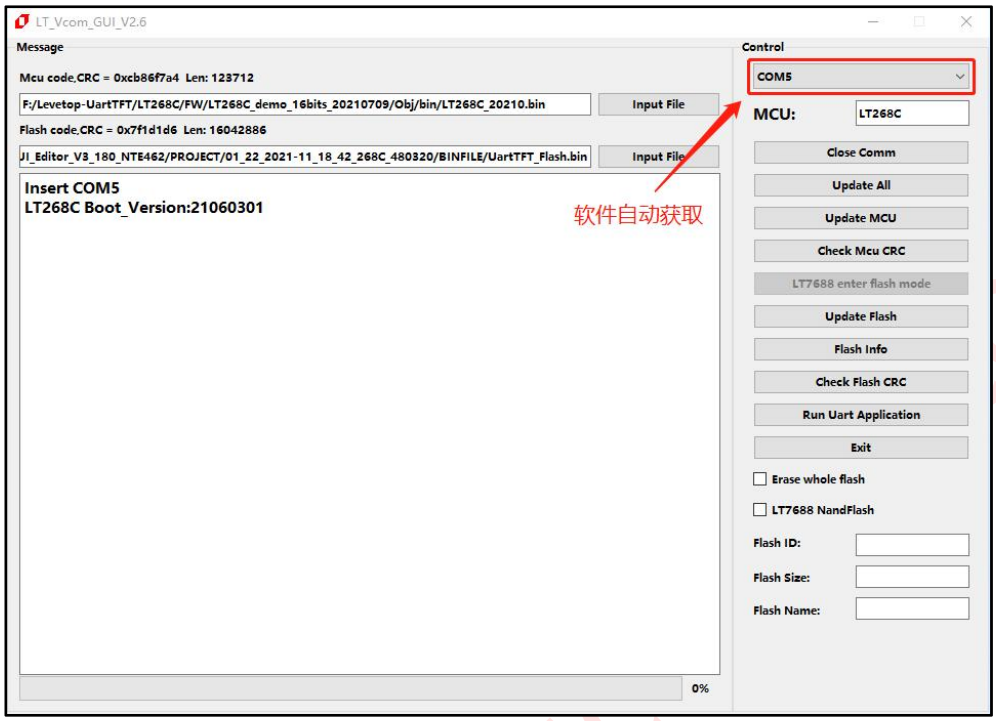

图 2-8：开启软件 LT_VCOM_GUI_Vxx.exe

3、导入需要更新的 MCU_Code.bin 或 UartTFT-II_Flash.bin，点击“Update XXX”开启更新，软件界面如下图所示：

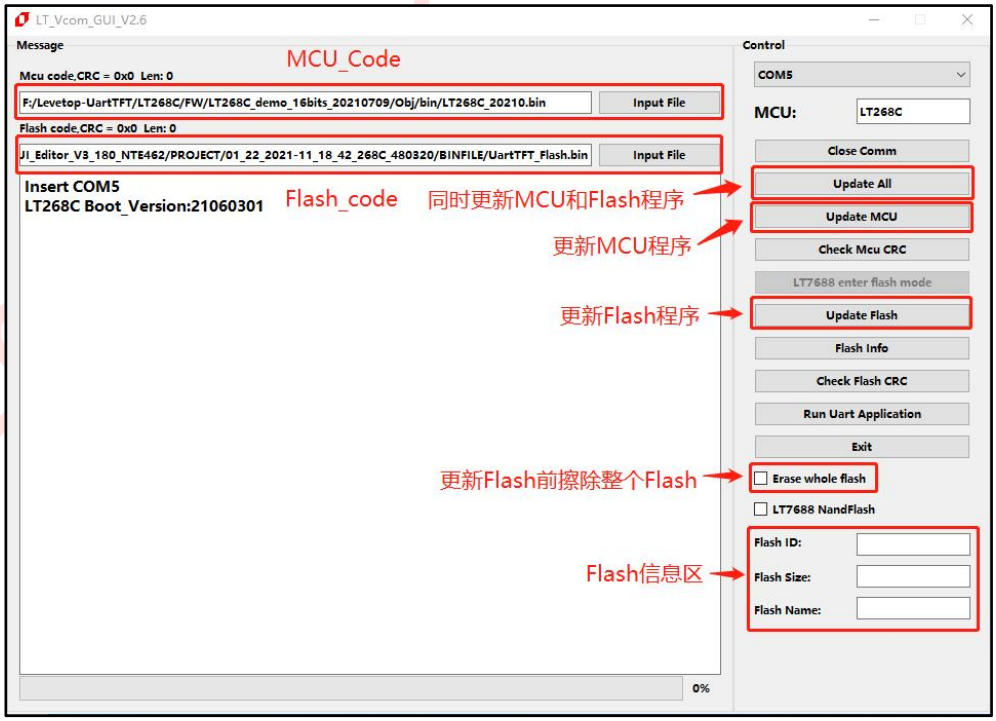

图 2-9：选择更新项目

4、更新完成后可点击“Run Uart Application”进行重置和运行程序，也可重新上电或复位进行重置和运行程序。（注：进行“Run Uart Application”操作时会使 MCU 退出更新模式，使软件不能识别串口，若要重新进入更新模式需按下 RST 按键进行复位。），如下图：

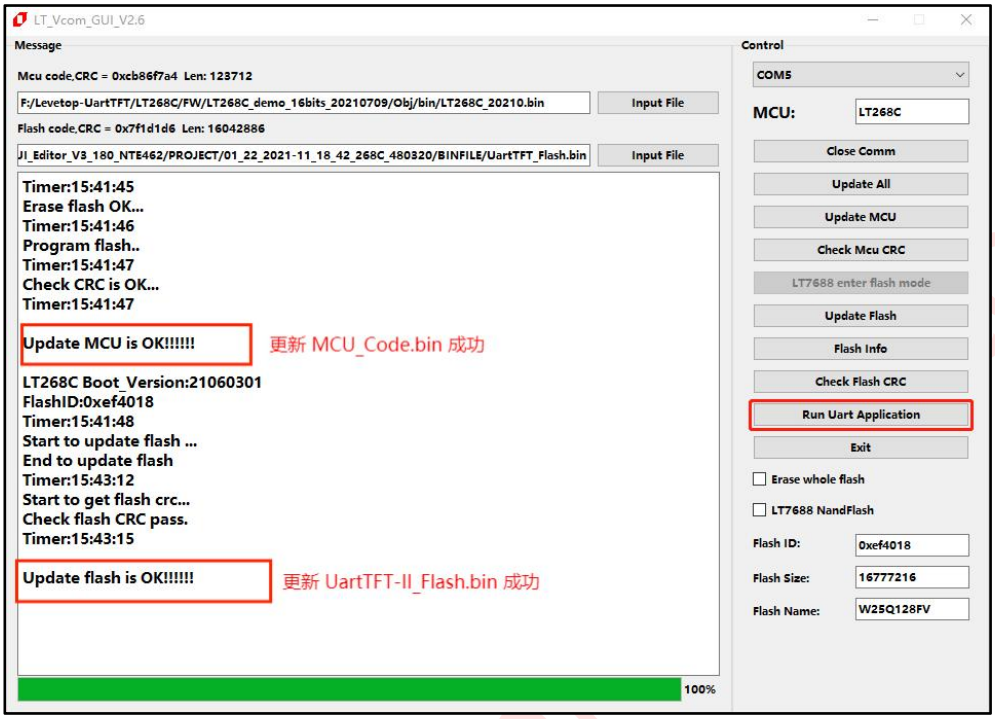

图 2-10：更新完成后进行重置和运行程序

2.2.2. 使用 SD 卡更新 LT269

可更新的文件：UartTFT-II_Flash.bin。

1、SD 卡要求：使用 SD 卡更新时，SD 卡需要 USB2.0 格式，2G-32G 容量，以 FAT32 方式格式化。在进行 SD 卡格式化时，建议使用快速格式化，分配单元大小选择默认配置，如下图所示：

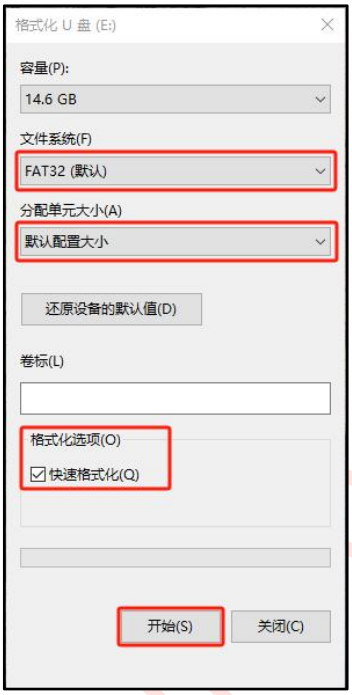

图 2-11：格式化 SD 卡

2、文件目录要求：格式化完成后，**在 SD 卡根目录下建立 UartTFT_Flash 文件夹，将需要更新的 UartTFT-II_Flash.bin 文件放入对应的文件夹（文件和文件夹名称不能修改）**如下图所示。

图 2-12：更新文档所在的储存目录

3、标注 1 处 USB 口连接电脑供电，上电后程序正常运行时将已经准备好的 SD 卡插入下图标注 2 卡槽内，程序会自动检测到 SD 卡插入，并进入升级模式。需要注意的是**标注 3 处不需要连接，若有连接需要断开。**

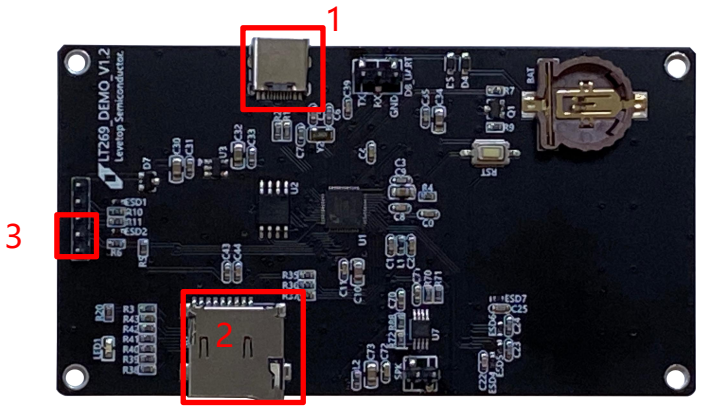

图 2-13：SD 卡更新 UartTFT-II_Flash.bin 示意图

4、给串口屏模组上电，等待显示正常后，将 SD 卡装入模组上的 SD 卡槽内，LT269 的程序检测到 SD 卡会进入更新倒计时画面，如下图。

图 2-14：SD 卡更新中

5、在倒计时 3 秒之后，LT269 就开始读取 SD 卡上的 UartTFT-II_Flash.bin 数据，开始进行对 Flash 的烧录动作，如下图所示，烧录结束后 LT268C 会自动重新启动。注意：SD 卡不支持更新 MCU_Code.bin。

图 2-15：SD 卡更新完毕

2.3. 使用串口控制演示模块

烧录完成重新上电可以得到相同的工程画面，此步骤确认使用者可以透过更新回复到原先的工程，此用用户可以用电脑发送串口数据来控制这个演示模块，连接与通讯的方法如下：

1、通过串口与演示模块连接，之后使用**串口调试工具（UI_Debugger-II）**，进行通信控制。先按下图顺序添加设置好的测试串口指令，也可以跳过该步骤，自行添加指令。串口调试工具详细使用方法可以看**UI_Editor-II_CH 使用说明书**介绍中的 **9.2.节串口调试工具（UI_Debugger-II）使用说明。**

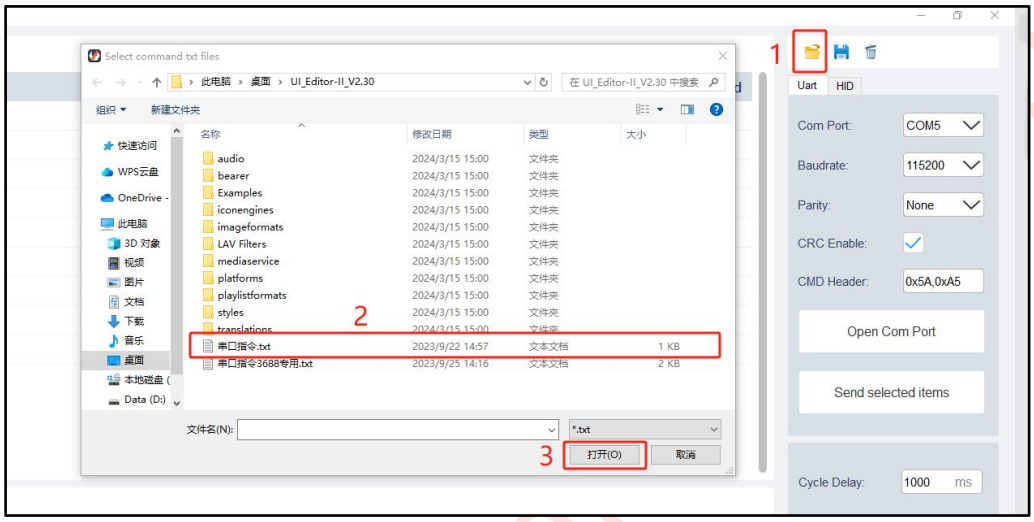

图 2-16：导入预设置的串口指令

2、导入指令后**选择端口和设置的波特率（需要与工程设置波特率对应）**，最后点击 Open Com Port 打开端口。

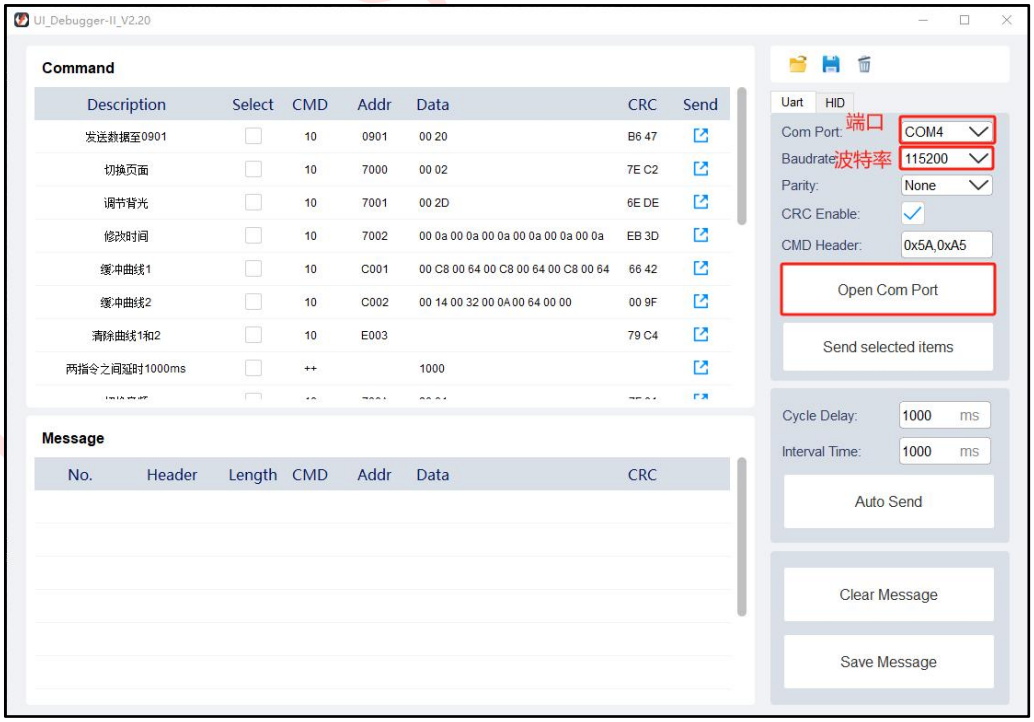

图 2-17：点击 Open Com Port 打开端口

3、连接后通过发送 Send 按钮发送对应设置好的指令，Message 处可以看到发送的完整指令以及反馈信息。

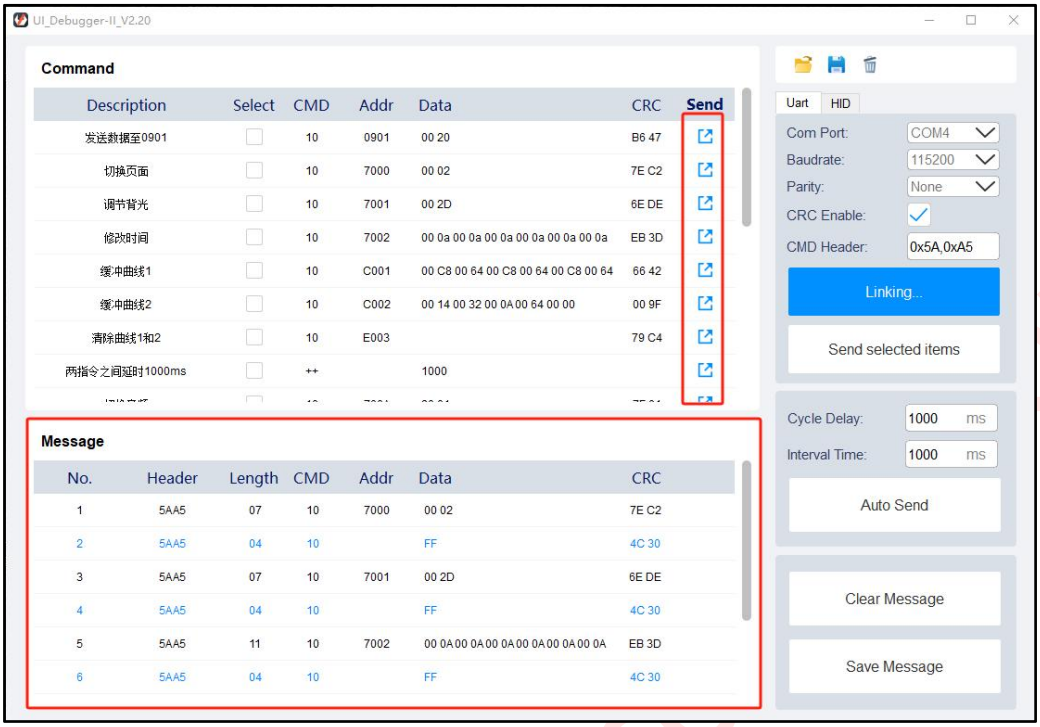

图 2-18：通过电脑与演示模块通讯

2.4. 新工程下载与更新

接下来可以试试更新另一个工程，例如在乐升半导体官网下载区下载相同分辨率为 480x320 的工程：

图 2-19：官网下载区另一个范例

同样透过 UartTFT-II 将工程编译后产生的 bin 档案（UartTFT-II_Flash.bin）烧录到 SPI Flash 内，烧录完成重新上电可以得到新的工程画面：

图 2-20：新的 UI 演示画面

注意，此 LT269 串口屏演示模块的分辨率为 480x320，Flash 是 NOR type，容量为 128Mbit (16Mbytes)，因此在 UI_Editor-II 设计的 UI 画面必须是符合相同的分辨率，同时工程编译后产生的 bin 档案（UartTFT-II_Flash.bin）不能超过演示模块的 Flash 容量大小。

此 LT269 串口屏演示模块烧录新工程通电后出现图 2-20 画面，其基本演示操作说明如下图 2-21、图 2-22；详细操作说明也可以到乐升官网的应用视频区观看或是下载（乐升官网→解决方案→应用视频→工控仪表类→LT269 应用-红酒柜，如图 2-23）。

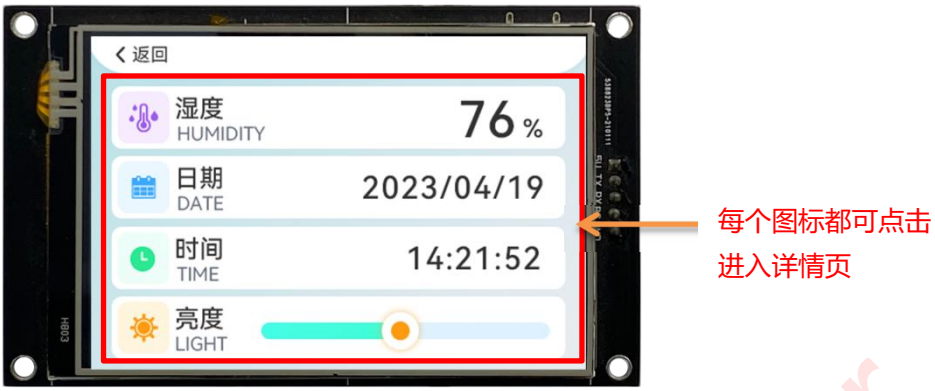

图 2-21：LT269 应用-串口屏基本功能展示画面 1

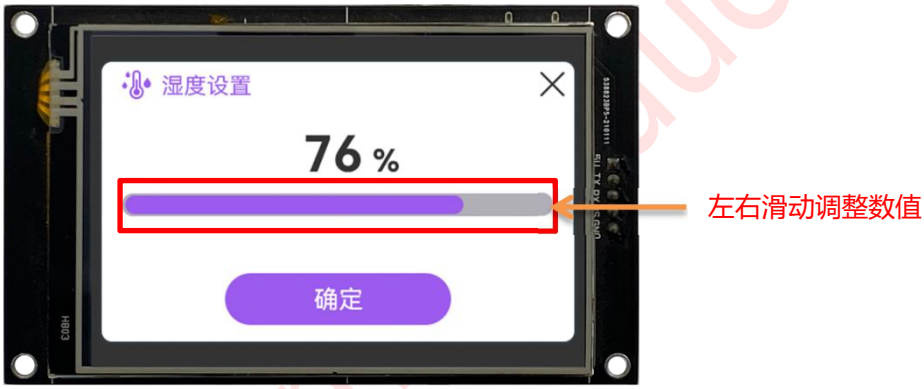

图 2-22：LT269 应用-串口屏基本功能展示画面 2

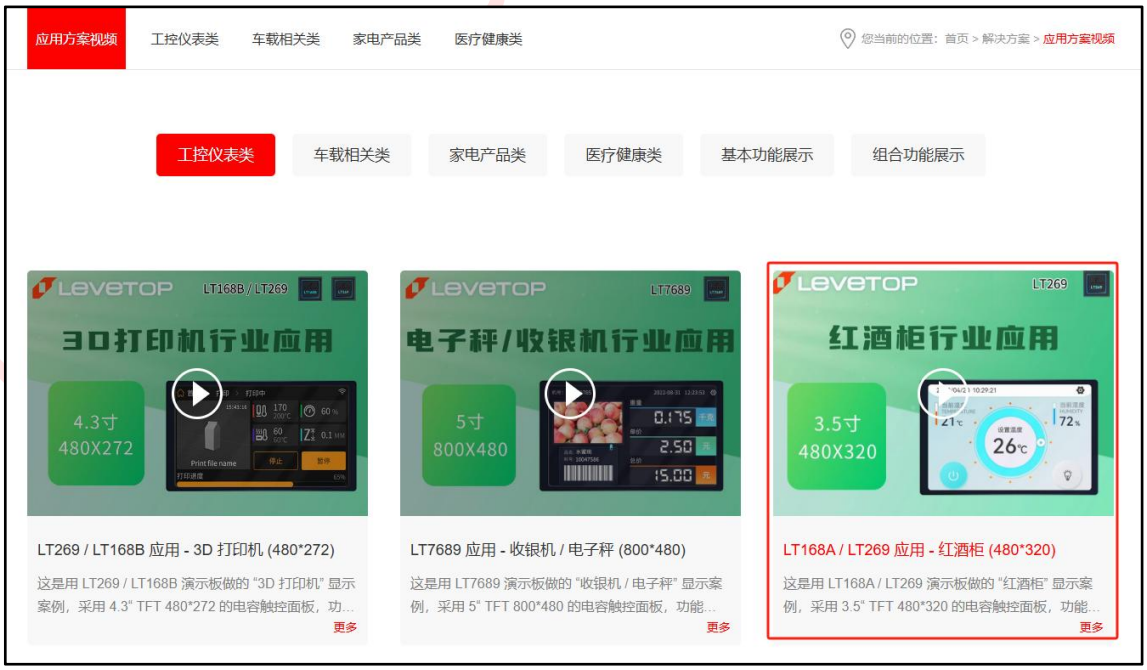

图 2-23：LT269 应用-串口屏基本功能展示视频官网位置

3. 主控端串口通讯程序范例

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。

3.1. 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：

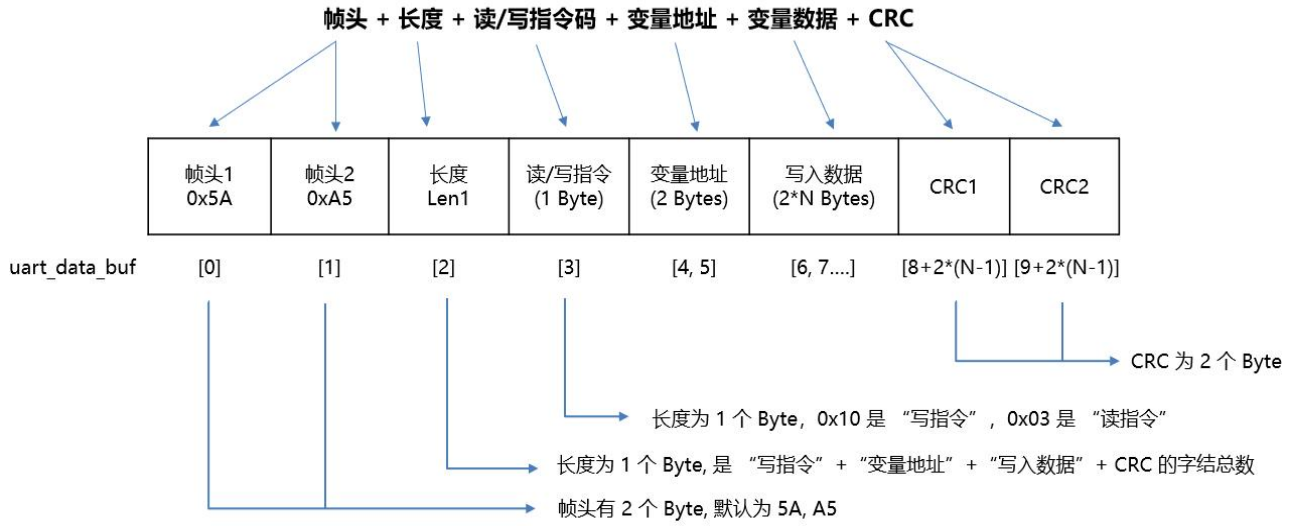

图 3-1：串口通讯指令结构图

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 LT269 串口芯片的接线模式。

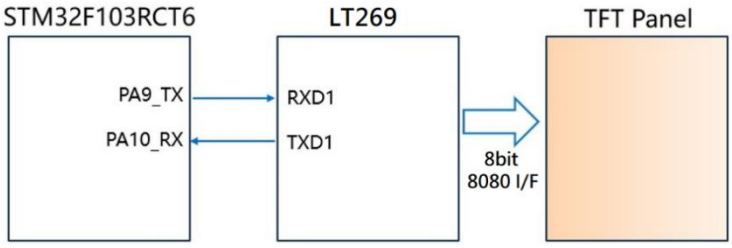

图 3-2：主控端 MCU（STM32F103RCT6）用串口与 LT269 串口屏芯片通讯

3.2. 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);
}
```

3.3. 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 "stm32f10x.h"                  // Device header
#include <stdio.h>
#include <stdarg.h>
void Uart_Init(void)                    // 串口初始化
{
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);

    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    USART_InitTypeDef USART_InitStructure;
    USART_InitStructure.USART_BaudRate = 115200;
    USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART_InitStructure.USART_Mode = USART_Mode_Tx;
    USART_InitStructure.USART_Parity = USART_Parity_No;
    USART_InitStructure.USART_StopBits = USART_StopBits_1;
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;
    USART_Init(USART1, &USART_InitStructure);

    USART_Cmd(USART1, ENABLE);
}

uint16_t UART_SendByte(uint8_t Byte)                    // 串口发送一个 Byte 数据
{
    USART_SendData(USART1, Byte);
    while (USART_GetFlagStatus(USART1, USART_FLAG_TXE) == RESET);
}

uint16_t UART_SendData(uint8_t *send_buf, uint16_t Length)   // 串口发送指令函数
{
    uint16_t ret;
    uint32_t i;

    for (i = 0; i < Length; i ++)
    {
        ret = UART_SendByte(send_buf[i]);
    }
    return ret;
}
```

3.4. 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：

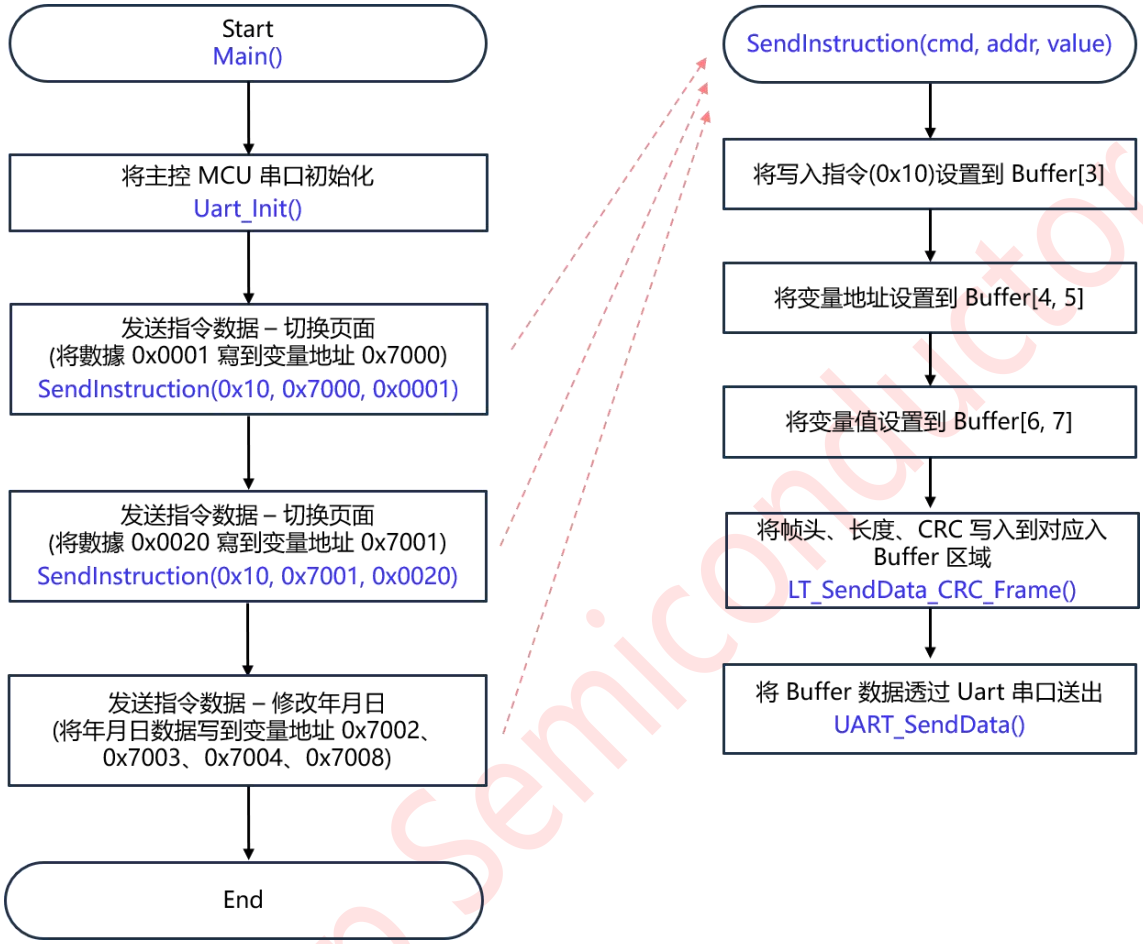

图 3-3：主控端发送串口指令的流程图

```
/***** main() *****/

#include "stm32f10x.h"                          // Device header
#include "Delay.h"
#include "Uart.h"
#include "CRC.h"

uint8_t SCI_C0 = 0x5A;                          // 设置帧头
uint8_t SCI_C1 = 0xA5;
uint8_t uart_data_buf[256];                     // 存放指令的数组
uint8_t len;                                    // 指令长度
uint8_t CRC_Enable_Flag = 1;                    // CRC 校验标志位
uint8_t CRC_Feedback_Flag = 1;

int main()
{
    Uart_Init();                                // 串口初始化
    SendInstruction(0x10, 0x7000, 0x0001);      // 发送指令数据 – 切换页面
    SendInstruction(0x10, 0x7001, 0x0020);      // 发送指令数据 – 调整背光亮度

    SendInstruction(0x10, 0x7002, 0x0017);      // 发送指令数据 – 修改年为 2023
    SendInstruction(0x10, 0x7003, 0x000B);      // 发送指令数据 – 修改月份 11
    SendInstruction(0x10, 0x7004, 0x001C);      // 发送指令数据 – 修改日为 28
    SendInstruction(0x10, 0x7008, 0x0001);      // 发送指令数据 – 确认年月日修改
}

void LT_SendData_CRC_Frame(uint8_t *buf, uint8_t len1)      // 获取长度及 CRC，并将帧头、长度、CRC
                                                            // 写入对应的 Buffer 区
{
    uint16_t TxToPc_crc;
    uint8_t crc[2] = {0};

    *(buf + 0) = SCI_C0;                                    // 将帧头写入到 Buffer[0, 1]
    *(buf + 1) = SCI_C1;
    if (CRC_Enable_Flag)
    {
        TxToPc_crc = CRC16(buf + 3, len1);                  // 进行 CRC 计算
        crc[0] = (uint8_t)(TxToPc_crc & 0x00ff);
        crc[1] = (uint8_t)((TxToPc_crc >> 8) & 0x00ff);

        len1 += 2;                                          // 加上 CRC（2 个 byte）后的长度
        *(buf + len1 + 1) = crc[0];                         // 将 CRC 写入到 Buffer 内
        *(buf + len1 + 2) = crc[1];
    }
    *(buf + 2) = len1;                  // 将长度(写指令+变量地址+变量数据+CRC 字节总数)
                                        // 写入到 Buffer[2]
    len = len1 + 3;                     // 完整的指令长度（再加上帧头 2byte 和 length1 个 byte）
}
```

```
void SendInstruction(uint8_t cmd, uint16_t addr, uint16_t value)
 {
      uart_data_buf[3] = cmd;                          // 设置功能码到 Buffer[3]
      uart_data_buf[4] = (uint8_t)(addr >> 8);         // 设置变量地址高位到 Buffer[4]
      uart_data_buf[5] = (uint8_t)addr;                // 设置变量地址低位到 Buffer[5]
      uart_data_buf[6] = (uint8_t)(value >> 8);        // 设置变量值高位到 Buffer[6]
      uart_data_buf[7] = (uint8_t)value;               // 设置变量值低位到 Buffer[7]
      LT_SendData_CRC_Frame(uart_data_buf, 5);         // 将帧头、长度、CRC 写入对应 Buffer 区
       UART_SendData(uart_data_buf, len);              // 通过 UART 串口将存在 Buffer 区内的指令数据
                                                       // 发送出去
      Delay_ms(1000);
}
```