SinoMCU

MC51F003A4

串口收发单总线说明

广东晟矽微电子有限公司

Guangdong SinoMCU Microelectronics Co., Ltd.

目录

一、简介

1. 应用环境：

应用电路中，由于空间限制，无法采用两线通信接口，只允许单线通信，而采用单总线（1-wire）协议，存在程序繁琐、速度慢等问题，采用自定义单总线协议也不可取；因此我们提供该问题解决方法，使用通用硬件串口，将原来 TX、RX 合为一线通信；程序上与原来使用方法一样，只需简单修改 IO 操作，实现最大通信波特率可达 500K（最大波特率由 MCU 决定）。

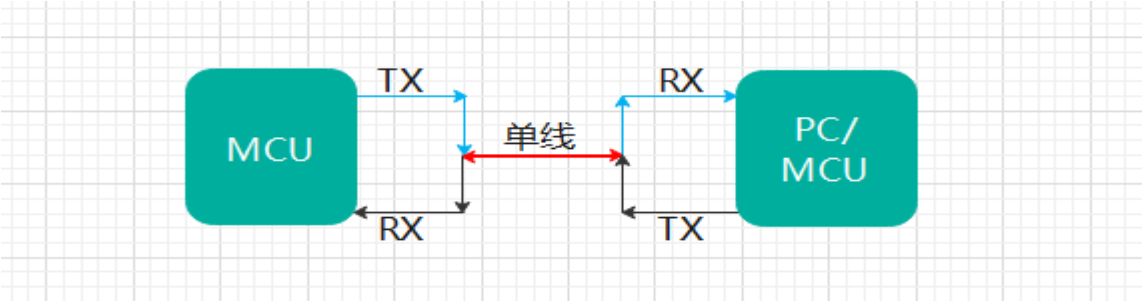
2. 软件实现：

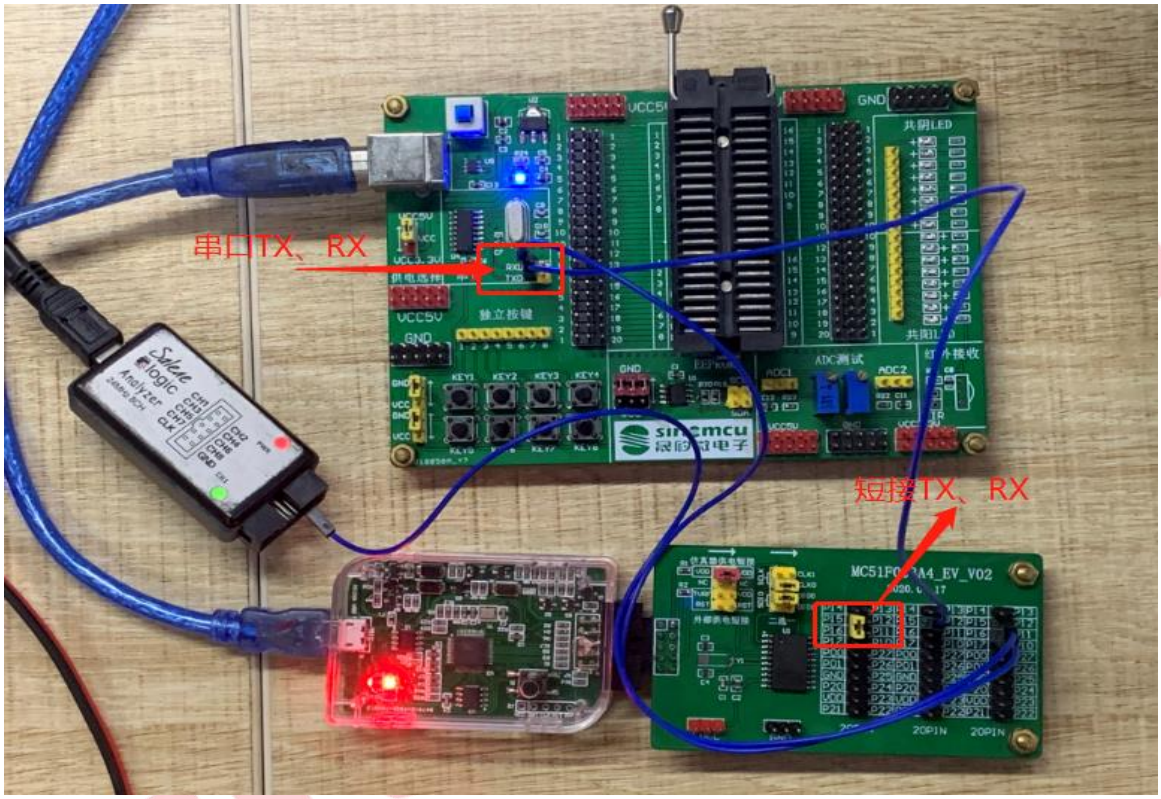
只有一线，不能同时收发，此时必须存在一个主机 A，和从机 B；开始 A 处于发送状态，B 处于接收状态，在 A 发送后，A 切换为接收状态，不再发送，B 收到数据后，B 开始发送，B 发送完成切换为接收状态；如果 A 发送后，A 在一定时间内没有收到 B 返回的数据，可以在 A 程序中添加超时，A 再次发送；

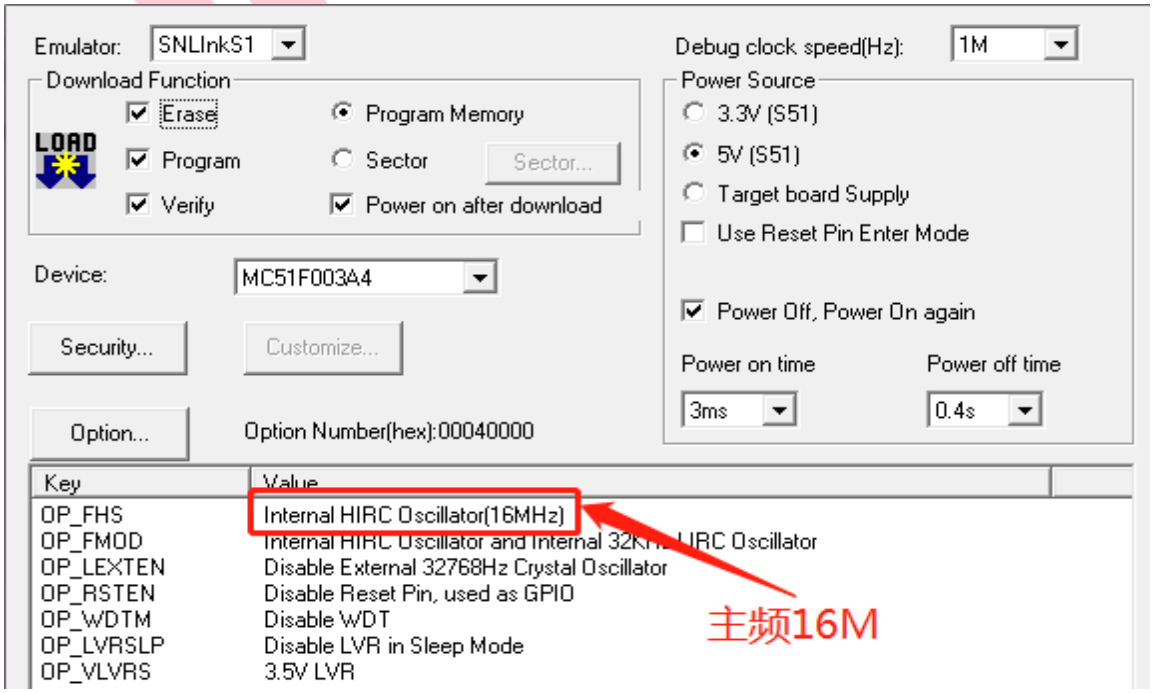
由于 TX、RX 接在一起，所以在 RX 接收前，应初始化串口，TX IO 设置为输入、不上拉状态，避免影响信号；在 TX 发送前，应关闭串口接收、关闭串口中断，TX IO 设置为推挽输出，RX IO 设置为输入不上拉，避免影响信号。

二、仿真环境配置，仿真器、串口连接

短接 MCU TX、RX（P15、P16），连接到电脑 TX、RX。


MCU
TX
RX
单线
RX
TX
PC/
MCU


串口TX、RX
短接TX、RX


Emulator: SNLInkS1
Debug clock speed(Hz): 1M
Download Function
Erase
Program Memory
Program
Sector
Sector...
Verify
Power on after download
Power Source
3.3V (S51)
5V (S51)
Target board Supply
Use Reset Pin Enter Mode
Power Off, Power On again
Power on time
Power off time
3ms
0.4s
Device: MC51F003A4
Security...
Customize...
Option...
Option Number(hex):00040000
Key
Value
OP_FHS
Internal HIRC Oscillator(16MHz)
OP_FMOD
OP_LEXTEN
Disable External 32768Hz Crystal Oscillator
OP_RSTEN
Disable Reset Pin, used as GPIO
OP_WDTM
Disable WDT
OP_LVRSLP
Disable LVR in Sleep Mode
OP_VLVRS
3.5V LVR
主频16M

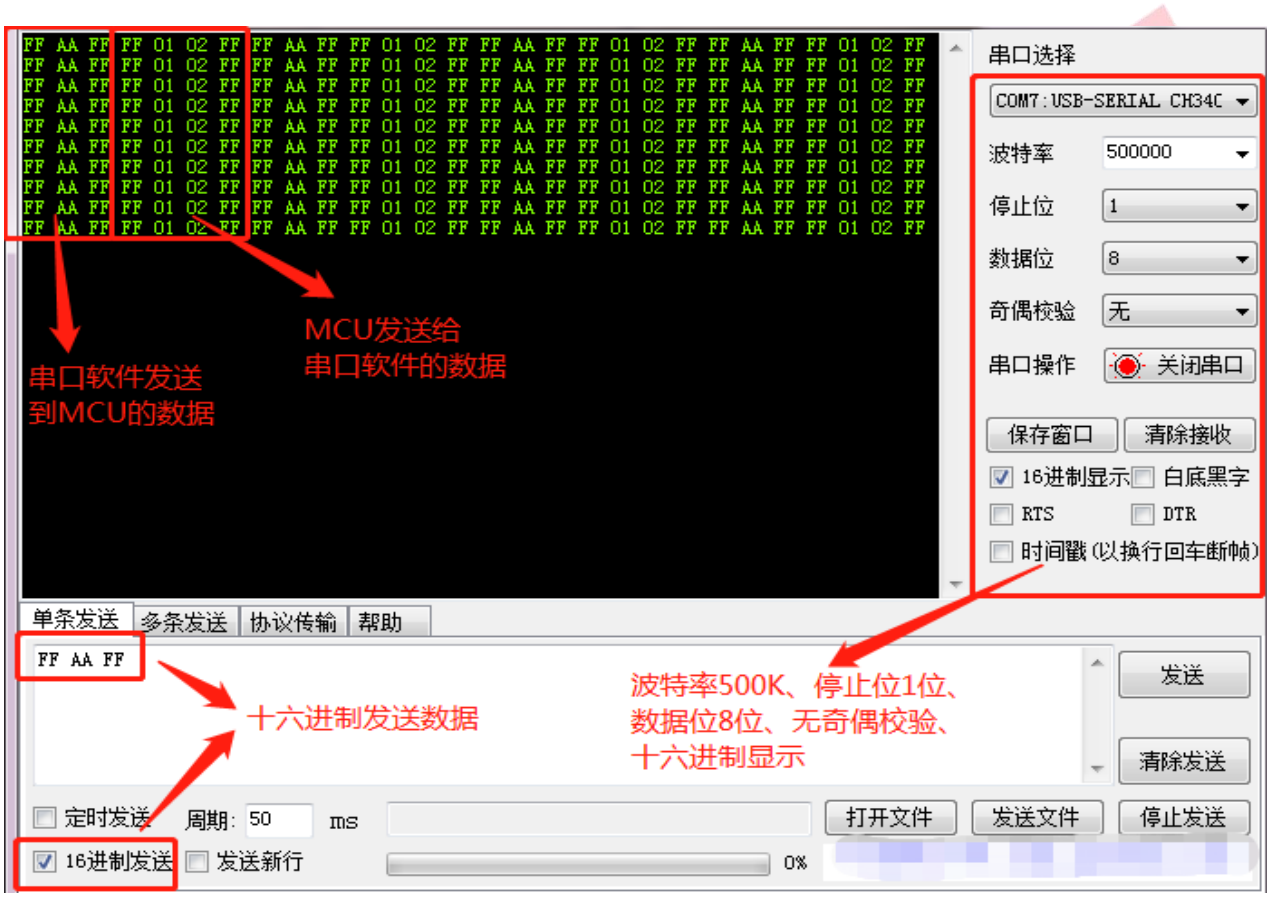
三、MCU硬件资源、功能说明

1. 硬件资源：

硬件串口、两个 IO、定时器波特率发生器；

2. 功能说明：

上电处于串口接收状态，当收到串口软件发送的 0xFF、0xAA、0xFF 数据后，MCU 发送 0xFF、0x01、0x02、0xFF；如下图：



三、软件介绍

1. 软件整体介绍

接收：TX 设置为输入、不上拉，防止 TX 影响 RX 数据；

发送：RX 设置为输入、不上拉，TX 设置为推挽输出；

接收到数据后切换为发送，发送完成后切换为接收；

2.mian.c 介绍

文件路径：\DEMO\USER\mian.c，代码如下：

main.c

```
40  void main(void)
41  {
42      DISABLE_GIE(); //关闭总中断
43      GPIO_Init();   //设置IO
44      UART1_Init();  //配置串口
45      ENABLE_GIE();  //打开总中断
46      delay(100);
47      Flag1.byte = 0; //清除标志位
48
49      //上电接收，不发送
50      P15_INPUTHIZ_MODE;
51      P1PU &= ~DEF_SET_BIT5; //TX输入  不上拉
52
53      while (1)
54      {
55          if (UART_RX_END_FLAG) //接收结束
56          {
57              if (UART_RX_buff[0] == 0xAA) //判断数据是否正确
58              {
59                  SCON_1 &= ~DEF_SET_BIT4; //禁止接收
60                  IE1 &= ~DEF_SET_BIT3;    //关闭UART1中断
61
62                  P16_INPUTHIZ_MODE;
63                  P1PU &= ~DEF_SET_BIT6; //RX 输入  不上拉
64                  P15_PUSHPULL_MODE;     //TX推挽输出
65                  P15D = 1;              //串口TX输出高
66
67                  UART_Send_Data(UART_TX_buff, sizeof(UART_TX_buff)); //发送数据
68
69                  UART1_Init(); //配置串口
70                  P15_INPUTHIZ_MODE;
71                  P1PU &= ~DEF_SET_BIT5; //TX输入  不上拉
72              }
73              UART_RX_END_FLAG = 0;
74              UART_RX_len = 0;
75              UART_RX_STA_FLAG = 0;
76          }
77      }
78  }
79
80
81
```

上电配置定时器、串口、IO；TX 输入、上拉，避免影响接收，while 中判断标志位（UART_RX_END_FLAG）等待串口数据接收完成；判断接收到数据是否为 0xAA，是 0xAA 则关闭串口接收、关闭串口中断，设置相关 IO，向串口软件发送数据，发送完成后配置重新配置串口，TX 输入、上拉，避免影响接收；

3.UART.c 介绍

文件路径：\DEMO\ HARDWARE\UART\UART.c，代码如下：

```
/*********************************************
;  *    @Function Name       : UART1_Init
;  *    @Description         : 串口1初始化
;  *    @IN_Parameter        :
;  *    @Return parameter    :
;  *********************************************/
void UART1_Init(void)
{
    PCON |= DEF_SET_BIT7; //使能UART1
    SCON_1 &= 0x3f;
    SCON_1 |= 0x50; //方式1  允许接收

    TMOD &= 0xCF;
    TMOD |= DEF_SET_BIT5; //内部时钟  方式2
    TL1 = T1_Cnt;
    TH1 = T1_Cnt;
    TR1 = 1; //打开T1

    IE1 |= DEF_SET_BIT3; //打开UART1中断
}
/*********************************************
;  *    @Function Name       : UART_Send_Byte
;  *    @Description         : 串口发送一个字节
;  *    @IN_Parameter        : Send_Data:发送的数据
;  *    @Return parameter    :
;  *********************************************/
void UART_Send_Byte(u8 send_data)
{
    uint16_t cnt = 0;
    SBUF_1 = send_data;      //写数据到UART数据寄存器
    while (!(SCON_1 & 0x02)) //等待前面的数据发送完成
    {
        cnt++;
        if (cnt >= 50000) //16M 大于62.5mS
        {
            NOP(20);
            break; //防止程序卡死
        }
    }
    SCON_1 &= ~DEF_SET_BIT1;
}
```

UART1_Init 串口 1、定时器 1 配置，串口 1 方式 1，定时器 1 方式 2 为串口 1 提供波特率，使能定时器、串口；

UART_Send_Byte 串口发送一个字节的数据；查询标志位时，防止死机，加入超时跳出；

```
/*********************************************
;   *    @Function Name        : UART_Send_Data
;   *    @Description          : 连续串口输出
;   *    @IN_Parameter         : *s:要发送的数据，len:数据长度
;   *    @Return parameter     :
;   *********************************************/
void UART_Send_Data(char *s, u8 len)
{
    while (len)
    {
        UART_Send_Byte(*s++);
        len--;
    }
}
/*********************************************
;   *    @Function Name        : UART_Send_String
;   *    @Description          : 连续串口输出
;   *    @IN_Parameter         : *s:要发送的数据
;   *    @Return parameter     :
;   *********************************************/
void UART_Send_String(char *s)
{
    while (*s) //检查字符串结束
    {
        UART_Send_Byte(*s++); //发送数据，地址+1
    }
}
/*********************************************
;   *    @Function Name        : putchar   (占用1kROM)
;   *    @Description          : 构建printf，重定义putchar
;   *    @IN_Parameter         :
;   *    @Return parameter     :
;   *********************************************/
char putchar(char tx_data)
{
    SBUF_1 = tx_data; //写数据到UART数据寄存器
    while (!(SCON_1 & 0x02)); //等待前面的数据发送完成
    SCON_1 &= ~DEF_SET_BIT1;
    return tx_data;
}
```

UART_Send_Data 串口输出指定长度数据，长度由形参 len 决定；

UART_Send_String 串口输出字符串，以'\0'结束；

Putchar 构建 printf，重定向 putchar；加入此函数可以直接使用 printf 输出。

版本修订记录

版本号	修订日期	修订内容
V1.0	2020-05-20	新建