Application Note

ATK-MS53L1M 模块使用说明

高性能激光测距模块（4 米）

使用说明

正点原子

广州市星翼电子科技有限公司

修订历史

版本	日期	原因
V1.0	2022/06/25	第一次发布
V1.1	2023/03/11	添加对阿波罗 STM32F429 开发板的阿波罗 STM32F767 开发板的支持
V1.2	2023/04/15	添加对阿波罗 STM32H743 开发板的支持
V1.3	2024/06/18	添加战舰/探索者/Mini V4 三款开发板的 IIC 引脚连接描述

目 录

1，硬件连接

1.1 正点原子 MiniSTM32F103 开发板

ATK-MS53L1M 模块可通过 UART 与正点原子 MiniSTM32F103 开发板进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS53L1M 模块	VCC	GND	TXD	RXD	SCL	SDA
MiniSTM32F103 开发板	3.3V/5V	GND	PD2	PC12	-	-

表 1.1.1 ATK-MS53L1M 模块与 MiniSTM32F103 开发板连接关系

ATK-MS53L1M 模块还可通过 IIC 与正点原子 MiniSTM32F103 开发板进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS53L1M 模块	VCC	GND	TXD	RXD	SCL	SDA
MiniSTM32F103 开发板	3.3V/5V	GND	-	-	PD2	PC12

表 1.1.2 ATK-MS53L1M 模块与 MiniSTM32F103 开发板连接关系

1.2 正点原子精英 STM32F103 开发板

ATK-MS53L1M 模块可通过 UART 与正点原子精英 STM32F103 开发板进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS53L1M 模块	VCC	GND	TXD	RXD	SCL	SDA
精英 STM32F103 开发板	3.3V/5V	GND	PB11	PB10	-	-

表 1.2.1 ATK-MS53L1M 模块与精英 STM32F103 开发板连接关系

1.3 正点原子战舰 STM32F103 开发板

ATK-MS53L1M 模块可通过 UART 与正点原子战舰 STM32F103 开发板进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS53L1M 模块	VCC	GND	TXD	RXD	SCL	SDA
战舰 STM32F103 开发板	3.3V/5V	GND	PB11	PB10	-	-

表 1.3.1 ATK-MS53L1M 模块与战舰 STM32F103 开发板连接关系

注意，若要使用正点原子战舰 STM32F103 开发板的 ATK MODULE 接口连接 ATK-MS53L1M 模块，需要用跳线帽将开发板板载的 P8 接线端子的 PB10(TX)和 GBC_RX 以及 PB11(RX)和 GBC_TX 用跳线帽进行短接，如下图所示：

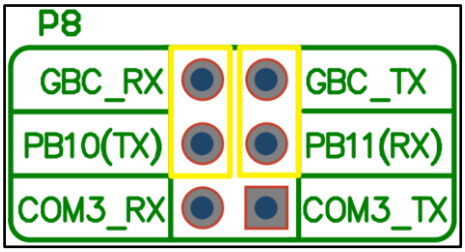

图 1.3.1 战舰 STM32F103 开发板 P8 接线端子

ATK-MS53L1M 模块还可通过 IIC 与正点原子战舰 STM32F103 开发板进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS53L1M 模块	VCC	GND	TXD	RXD	SCL	SDA
战舰 STM32F103 开发板	3.3V/5V	GND	-	-	PB10	PB11

表 1.3.2 ATK-MS53L1M 模块与战舰 STM32F103 开发板连接关系

1.4 正点原子探索者 STM32F407 开发板

ATK-MS53L1M 模块可通过 UART 与正点原子探索者 STM32F407 开发板进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS53L1M 模块	VCC	GND	TXD	RXD	SCL	SDA
探索者 STM32F407 开发板	3.3V/5V	GND	PB11	PB10	-	-

表 1.4.1 ATK-MS53L1M 模块与探索者 STM32F407 开发板连接关系

注意，若要使用正点原子探索者 STM32F407 开发板的 ATK MODULE 接口连接 ATK-MS53L1M 模块，需要用跳线帽将开发板板载的 P2 接线端子的 PB10(TX)和 GBC_RX 以及 PB11(RX)和 GBC_TX 用跳线帽进行短接，如下图所示：

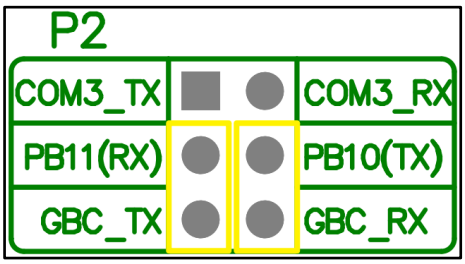

图 1.4.1 探索者 STM32F407 开发板 P2 接线端子

ATK-MS53L1M 模块还可通过 IIC 与正点原子探索者 STM32F407 开发板进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS53L1M 模块	VCC	GND	TXD	RXD	SCL	SDA
探索者 STM32F407 开发板	3.3V/5V	GND	-	-	PB11	PB10

表 1.4.2 ATK-MS53L1M 模块与探索者 STM32F407 开发板连接关系

1.5 正点原子 F407 电机控制开发板

ATK-MS53L1M 模块可通过 UART 与正点原子 F407 电机控制开发板进行连接，具体的

连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS53L1M 模块	VCC	GND	TXD	RXD	SCL	SDA
F407 电机控制开发板	3.3V/5V	GND	PC11	PC10	-	-

表 1.5.1 ATK-MS53L1M 模块与 F407 电机控制开发板连接关系

1.6 正点原子 MiniSTM32H750 开发板

ATK-MS53L1M 模块可通过 UART 与正点原子 MiniSTM32H750 开发板通过 UART 进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS53L1M 模块	VCC	GND	TXD	RXD	SCL	SDA
Min STM32H750 开发板	3.3V/5V	GND	PA3	PA2	-	-

表 1.6.1 ATK-MS53L1M 模块与 MiniSTM32H750 开发板连接关系

1.7 正点原子阿波罗 STM32F429 开发板

ATK-MS53L1M 模块可通过 UART 与正点原子阿波罗 STM32F429 开发板进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS53L1M 模块	VCC	GND	TXD	RXD	SCL	SDA
阿波罗 STM32F429 开发板	3.3V/5V	GND	PB11	PB10	-	-

表 1.7.1 ATK-MS53L1M 模块与阿波罗 STM32F429 开发板连接关系

注意，若要使用正点原子阿波罗 STM32F429 开发板的 ATK MODULE 接口连接 ATK-MS53L1M 模块，需要用跳线帽将开发板板载的 P9 接线端子的 PB10(TX)和 GBC_RX 以及 PB11(RX)和 GBC_TX 用跳线帽进行短接，如下图所示：

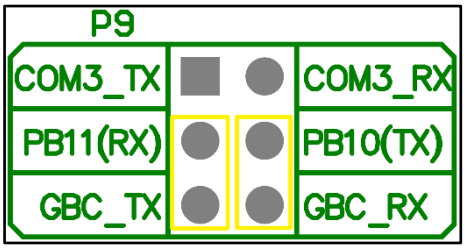

图 1.7.1 阿波罗 STM32F429 开发板 P9 接线端子

1.8 正点原子阿波罗 STM32F767 开发板

ATK-MS53L1M 模块可通过 UART 与正点原子阿波罗 STM32F767 开发板进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS53L1M 模块	VCC	GND	TXD	RXD	SCL	SDA
阿波罗 STM32F767 开发板	3.3V/5V	GND	PB11	PB10	-	-

表 1.8.1 ATK-MS53L1M 模块与阿波罗 STM32F767 开发板连接关系

注意，若要使用正点原子阿波罗 STM32F767 开发板的 ATK MODULE 接口连接 ATK-MS53L1M 模块，需要用跳线帽将开发板板载的 P9 接线端子的 PB10(TX)和 GBC_RX 以及 PB11(RX)和 GBC_TX 用跳线帽进行短接，如下图所示：

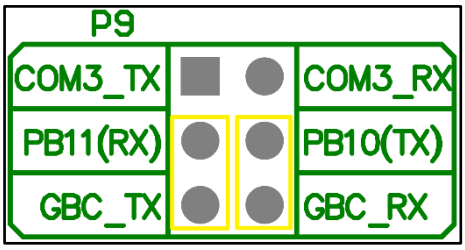

图 1.8.1 阿波罗 STM32F767 开发板 P9 接线端子

1.9 正点原子阿波罗 STM32H743 开发板

ATK-MS53L1M 模块可通过 UART 与正点原子阿波罗 STM32H743 开发板进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS53L1M 模块	VCC	GND	TXD	RXD	SCL	SDA
阿波罗 STM32H743 开发板	3.3V/5V	GND	PB11	PB10	-	-

表 1.9.1 ATK-MS53L1M 模块与阿波罗 STM32H743 开发板连接关系

注意，若要使用正点原子阿波罗 STM32H743 开发板的 ATK MODULE 接口连接 ATK-MS53L1M 模块，需要用跳线帽将开发板板载的 P9 接线端子的 PB10(TX)和 GBC_RX 以及 PB11(RX)和 GBC_TX 用跳线帽进行短接，如下图所示：

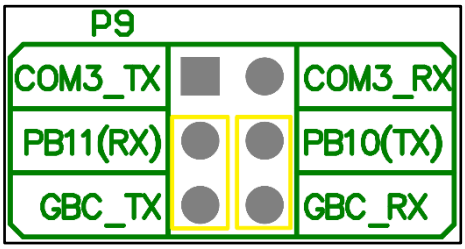

图 1.9.1 阿波罗 STM32H743 开发板 P9 接线端子

2，实验功能

2.1 ATK-MS53L1M 模块测试实验

2.1.1 功能说明

在本实验中，开发板主控芯片通过 UART 与 ATK-MS53L1M 模块进行通讯，通过按键配置 ATK-MS53L1M 模块的工作模式，并根据 ATK-MS53L1M 模块的工作方式，使用不同的通讯方式（Normal 或 Modbus）获得 ATK-MS53L1M 测量的距离数据，并将其在串口调试助手和 LCD 上进行显示。

2.1.2 源码解读

打开本实验的工程文件夹，能够在./Drivers/BSP 目录下看到 ATK_MS53L1M 子文件夹，该文件夹中就包含了 ATK-MS53L1M 模块的驱动文件，如下图所示：

```
./Drivers/BSP/ATK_MS53L1M/
|-- atk_ms53l1m.c
|-- atk_ms53l1m.h
|-- atk_ms53l1m_uart.c
`-- atk_ms53l1m_uart.h
```

图 2.1.2.1 ATK-MS53L1M 模块驱动代码

2.1.2.1 ATK-MS53L1M 模块接口驱动

在图 2.1.2.1 中，atk_ms53l1m_uart.c 和 atk_ms53l1m_uart.h 是开发板与 ATK-MS53L1M 模块通讯而使用的 UART 驱动文件，关于 UART 的驱动介绍，请查看正点原子各个开发板对应的开发指南中 UART 对应的章节。

值得一提的是，由于 ATK-MS53L1M 模块通过 UART 发送给主控芯片的数据长度是不固定的，因此主控芯片就无法直接通过接收到数据的长度来判断 ATK-MS53L1M 模块传来的一帧数据是否完成。对于这种通过 UART 接收不定长数据的情况，可以通过 UART 总线是否空闲来判断一帧数据是否传输完成，恰巧 STM32 的 UART 提供了总线空闲中断功能，因此可以开启 UART 的总线空闲中断，并在中断中做相应的处理，具体的实现过程可以查看 ATK-MS53L1M 模块的模块接口驱动代码，这里不做过多的描述。

2.1.2.2 ATK-MS53L1M 模块驱动

在图 2.1.2.1 中，atk_ms53l1m.c 和 atk_ms53l1m.h 是 ATK-MS53L1M 模块的驱动文件，包含了 ATK-MS53L1M 模块初始化、通过 UART 读写数据的相关 API 函数。函数比较多，下面仅介绍几个重要的 API 函数。

1. 函数 atk_ms53l1m_init()

该函数用于初始化 ATK-MS53L1M 模块，具体的代码，如下所示：

```
/**
 * @brief   ATK-MS53L1M 初始化
 * @param   baudrate    : ATK-MS53L1M UART 通讯波特率
 *          id          : ATK-MS53L1M 的设备 ID
 * @retval  ATK_MS53L1M_EOK     : ATK-MS53L1M 初始化成功，函数执行成功
 *          ATK_MS53L1M_ERROR   : ATK-MS53L1M 初始化失败，函数执行失败
```

```
 */
uint8_t atk_ms53l1m_init(uint32_t baudrate, uint16_t *id)
{
    uint8_t i;

    /* ATK-MS53L1M UART 初始化 */
    atk_ms53l1m_uart_init(baudrate);

    /* 获取设备地址 */
    i = 0;
    while ( atk_ms53l1m_read_data( 0xFFFF,
                                   ATK_MS53L1M_FUNCODE_IDSET,
                                   2,
                                   id)
            != ATK_MS53L1M_EOK)
    {
        delay_ms(100);
        if (++i == 5)
        {
            return ATK_MS53L1M_ERROR;
        }
    }

    /* 设置 ATK-MS53L1M 模块的工作模式为 Modbus 模式 */
    i = 0;
    while ( atk_ms53l1m_write_data(*id,
                                   ATK_MS53L1M_FUNCODE_WORKMODE,
                                   ATK_MS53L1M_WORKMODE_MODBUS)
            != ATK_MS53L1M_EOK)
    {
        delay_ms(100);
        if (++i == 5)
        {
            return ATK_MS53L1M_ERROR;
        }
    }

    return ATK_MS53L1M_EOK;
}
```

从上面的代码中可以看出，函数 atk_ms53l1m_init()会先初始化与 ATK-MS53L1M 模块的进行通讯的 UART，然后获取 ATK-MS53L1M 模块的设备地址，因为在一个主控芯片上可能连接多个 ATK-MS53L1M 模块，因此需要根据设备的地址来进行区分，最后就是讲 ATK-MS53L1M 模块的工作模式默认配置为 Modbus 工作模式。

2. 函数 atk_ms53l1m_read_data()和 atk_ms53l1m_write_data()

这两个函数是主控芯片与 ATK-MS53L1M 模块通过 UART 通讯的核心，分别用于主控芯片对 ATK-MS53L1M 模块的读操作和写操作，具体的代码如下所示：

```
/**
 * @brief   根据模块功能码读取数据
 * @param   addr        : 设备地址
 *          fun_code    : 功能码
 *          len         : 数据长度，取值范围：1 或 2
 *          dat         : 读取到的数据
 * @retval  ATK_MS53L1M_EOK         : 没有错误
 *          ATK_MS53L1M_ETIMEOUT    : 接收数据超时
 *          ATK_MS53L1M_EFRAME      : 帧错误
 *          ATK_MS53L1M_ECRC        : CRC 校验错误
 *          ATK_MS53L1M_EOPT        : 操作错误
 */
uint8_t atk_ms53l1m_read_data( uint16_t addr,
                               uint8_t fun_code,
                               uint8_t len,
                               uint16_t *dat)
{
    uint8_t  ret;
    uint16_t check_sum;
    uint8_t  buf[9];

    buf[0] = ATK_MS53L1M_MASTER_FRAME_HEAD;         /* 标志头 */
    buf[1] = ATK_MS53L1M_SENSOR_TYPE;               /* 传感器类型 */
    buf[2] = (uint8_t)(addr >> 8);                  /* 传感器地址，高 8 位 */
    buf[3] = (uint8_t)(addr & 0xFF);                /* 传感器地址，低 8 位 */
    buf[4] = ATK_MS53L1M_OPT_READ;                  /* 读操作 */
    buf[5] = fun_code;                              /* 功能码 */
    buf[6] = len;                                   /* 数据长度 */

    check_sum = atk_ms53l1m_crc_check_sum(buf, 7); /* 计算 CRC 校验和 */

    buf[7] = (uint8_t)(check_sum >> 8);             /* CRC 校验码，高 8 位 */
    buf[8] = (uint8_t)(check_sum & 0xFF);           /* CRC 校验码，低 8 位 */

    atk_ms53l1m_uart_rx_restart();                 /* 准备重新开始接收新的一帧数据 */
    atk_ms53l1m_uart_send(buf, 9);                  /* 发送数据 */
    ret = atk_ms53l1m_unpack_recv_data(dat);        /* 解析应答数据 */

    return ret;
}

/**
```

```
 * @brief   根据模块功能码写入 1 字节数据
 * @param   addr        : 设备地址
 *          fun_code    : 功能码
 *          dat         : 待写入的 1 字节数据
 * @retval  ATK_MS53L1M_EOK         : 没有错误
 *          ATK_MS53L1M_ETIMEOUT    : 接收数据超时
 *          ATK_MS53L1M_EFRAME      : 帧错误
 *          ATK_MS53L1M_ECRC        : CRC 校验错误
 *          ATK_MS53L1M_EOPT        : 操作错误
 */
uint8_t atk_ms53l1m_write_data(uint16_t addr, uint8_t fun_code, uint8_t dat)
{
    uint8_t  ret;
    uint8_t  buf[10];
    uint16_t check_sum;

    buf[0] = ATK_MS53L1M_MASTER_FRAME_HEAD;          /* 标志头 */
    buf[1] = ATK_MS53L1M_SENSOR_TYPE;                /* 传感器类型 */
    buf[2] = (uint8_t)(addr >> 8);                   /* 传感器地址，高 8 位 */
    buf[3] = (uint8_t)(addr & 0xFF);                 /* 传感器地址，低 8 位 */
    buf[4] = ATK_MS53L1M_OPT_WRITE;                  /* 写操作 */
    buf[5] = fun_code;                               /* 功能码 */
    buf[6] = 0x01;                                   /* 数据长度 */
    buf[7] = dat;                                    /* 数据 */

    check_sum = atk_ms53l1m_crc_check_sum(buf, 8);   /* 计算 CRC 校验和 */

    buf[8] = (uint8_t)(check_sum >> 8);              /* CRC 校验码，高 8 位 */
    buf[9] = (uint8_t)(check_sum & 0xFF);            /* CRC 校验码，低 8 位 */

    atk_ms53l1m_uart_rx_restart();                   /* 准备重新开始接收新的一帧数据 */
    atk_ms53l1m_uart_send(buf, 10);                  /* 发送数据 */
    ret = atk_ms53l1m_unpack_recv_data(NULL);        /* 解析应答数据 */

    return ret;
}
```

从上面的代码中可以看出，与 ATK-MS53L1M 模块的通讯有一定的帧格式，具体帧格式的介绍，请见《ATK-MS53L1M 模块用户手册》。

3. 函数 atk_ms53l1m_normal_get_data()

该函数用于 ATK-MS53L1M 模块在 Normal 工作模式下，获取 ATK-MS53L1M 模块测量的距离数据，具体的代码如下所示：

```
/**
 * @brief   ATK-MS53L1M Normal 工作模式获取测量值
 * @param   dat : 获取到的测量值
```

```
 * @retval  ATK_MS53L1M_EOK     : 获取测量值成功
 *          ATK_MS53L1M_ERROR   : UART 未接收到数据，获取测量值失败
 */
uint8_t atk_ms53l1m_normal_get_data(uint16_t *dat)
{
    uint8_t  *buf = NULL;
    uint8_t  i = 0;
    char     *p;
    uint16_t dat_tmp = 0;

    atk_ms53l1m_uart_rx_restart();
    while (buf == NULL)
    {
        buf = atk_ms53l1m_uart_rx_get_frame();
        if (++i == 10)
        {
            return ATK_MS53L1M_ERROR;
        }
        delay_ms(100);
    }

    p = strstr((char *)buf, "d:");
    while (*p != 'm')
    {
        if (*p >= '0' && *p <= '9')
        {
            dat_tmp = dat_tmp * 10 + (*p - '0');
        }
        p++;
    }

    *dat = dat_tmp;

    return ATK_MS53L1M_EOK;
}
```

因为在 Normal 工作模式下，ATK-MS53L1M 模块会自动不断地输出测量到的距离数据，因此，在 Normal 工作模式下，无需向 ATK-MS53L1M 模块发送任何数据，即可直接从 UART 获取的数据中解析出 ATK-MS53L1M 模块测量到的距离数据。

4. 函数 atk_ms53l1m_modbus_get_data()

该函数用于 ATK-MS53L1M 模块在 Modbus 工作模式下，获取 ATK-MS53L1M 模块测量的距离数据，具体的代码如下所示：

```
/**
 * @brief   ATK-MS53L1M Modbus 工作模式获取测量值
 * @param   dat : 获取到的测量值
```

```
 * @retval  ATK_MS53L1M_EOK     : 获取测量值成功
 *          ATK_MS53L1M_ERROR   : UART 未接收到数据，获取测量值失败
 */
uint8_t atk_ms53l1m_modbus_get_data(uint16_t id, uint16_t *dat)
{
    uint8_t ret;
    uint16_t dat_tmp;

    ret = atk_ms53l1m_read_data(id, ATK_MS53L1M_FUNCODE_MEAUDATA, 2, &dat_tmp);
    if (ret != 0)
    {
        *dat = 0;
        return ATK_MS53L1M_ERROR;
    }
    else
    {
        *dat = dat_tmp;
        return ATK_MS53L1M_EOK;
    }
}
```

从上面的代码中可以看出，该函数就是调用函数 atk_ms53l1m_read_data()获取 ATK-MS53L1M 模块的距离测量数据。

2.1.2.3 实验测试代码

实验的测试代码为文件 demo.c，在工程目录下的 User 子目录中。测试代码的入口函数为 demo_run()，具体的代码，如下所示：

```
/**
 * @brief   例程演示入口函数
 * @param   无
 * @retval  无
 */
void demo_run(void)
{
    uint8_t  ret;
    uint8_t  key;
    uint16_t id;
    uint8_t  is_normal = 0;

    /* 初始化 ATK-MS53L1M */
    ret = atk_ms53l1m_init(115200, &id);
    if (ret != 0)
    {
        printf("ATK-MS53L1M init failed!\r\n");
        while (1)
        {
```

```
            LED0_TOGGLE();
            delay_ms(200);
        }
    }

    /* ATK-MS53L1M 初始化成功，显示设备地址 */
    demo_show_id(id);

    while (1)
    {
        key = key_scan(0);

        switch (key)
        {
            case KEY0_PRES:
            {
                /* 获取 ATK-MS53L1M 测量值 */
                demo_key0_fun(is_normal, id);
                break;
            }
            case KEY1_PRES:
            {
                /* 切换 ATK-MS53L1M 工作模式 */
                demo_key1_fun(&is_normal, id);
                break;
            }
            default:
            {
                break;
            }
        }

    delay_ms(10);
    }
}
```

本实验的应用代码中，首先对 ATK-MS53L1M 模块进行初始化，如果初始化成功，那么就根据按下的按键调用不同的函数，其中按键 0 按下调用的函数为 demo_key0_fun()，该函数会自动根据 ATK-MS53L1M 模块的工作状态，读取 ATK-MS53L1M 在该工作模式下测量出的距离数据。按键 1 按下调用的函数为 demo_key1_fun()，该函数用于切换 ATK-MS53L1M 模块的工作模式（Normal 工作模式或 Modbus 工作模式）。

2.1.3 实验现象

将 ATK-MS53L1M 模块按照第一节“硬件连接”中介绍的连接方式与开发板连接，并将实验代码编译烧录至开发板中，如果此时开发板连接 LCD，那么 LCD 显示的内容，如下图所示：

图 2.1.3.1 LCD 显示内容一

同时，通过串口调试助手输出实验信息，如下图所示：

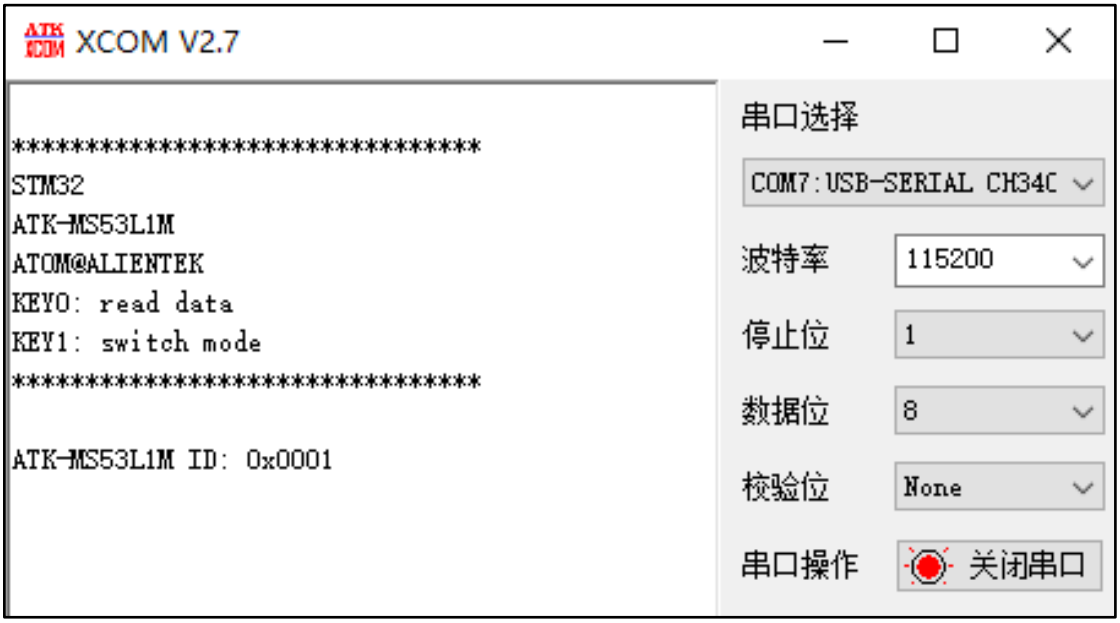

图 2.1.3.2 串口调试助手显示内容一

可以看到，在串口调试助手和 LCD 上已经显示了 ATK-MS53L1M 模块的设备地址，因为程序已经自动初始化了 ATK-MS53L1M，并成功读取到的 ATK-MS53L1M 模块的设备地址。

接下来按下按键 0，就可以读取 ATK-MS53L1M 模块测量到的距离数据，并将其通过串口输出到串口调试助手，因为程序在初始化 ATK-MS53L1M 模块的时候，默认将其工作模式初始化为 Modbus 工作模式，因此，此时按下按键 0，将会以 Modbus 工作模式的方式，读取 ATK-MS53L1M 模块测量到的距离数据，如下图所示：

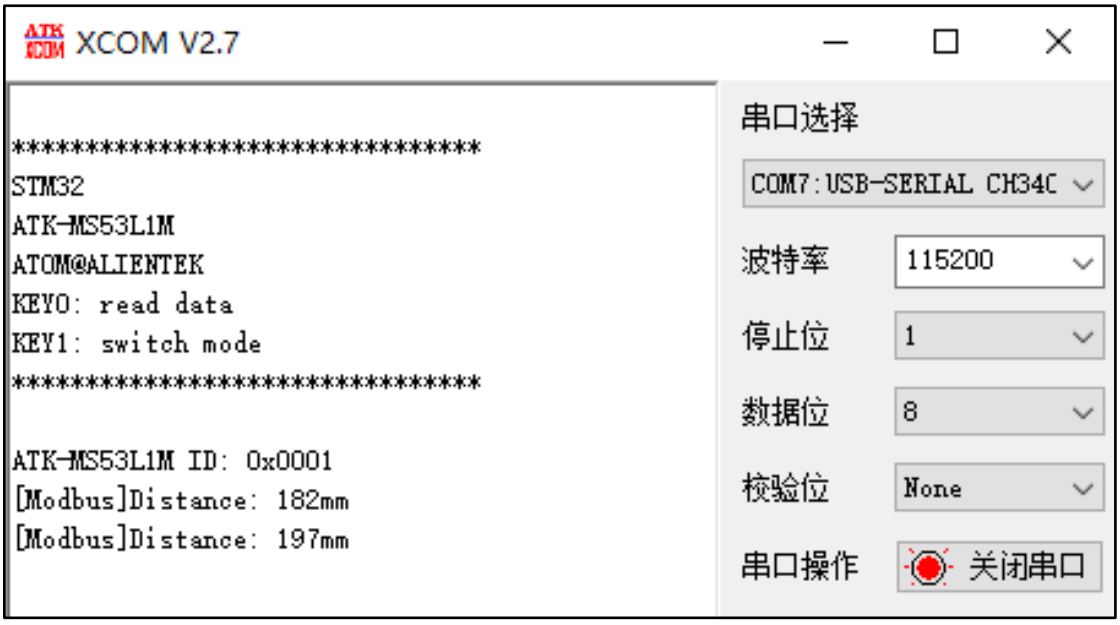

图 2.1.3.3 串口调试助手显示内容二

接下来可以通过按下按键 1 来切换 ATK-MS53L1M 模块的工作模式为 Normal 工作模式，并再次通过按键 0 来读取 ATK-MS53L1M 模块测量到的距离数据，如下图所示：

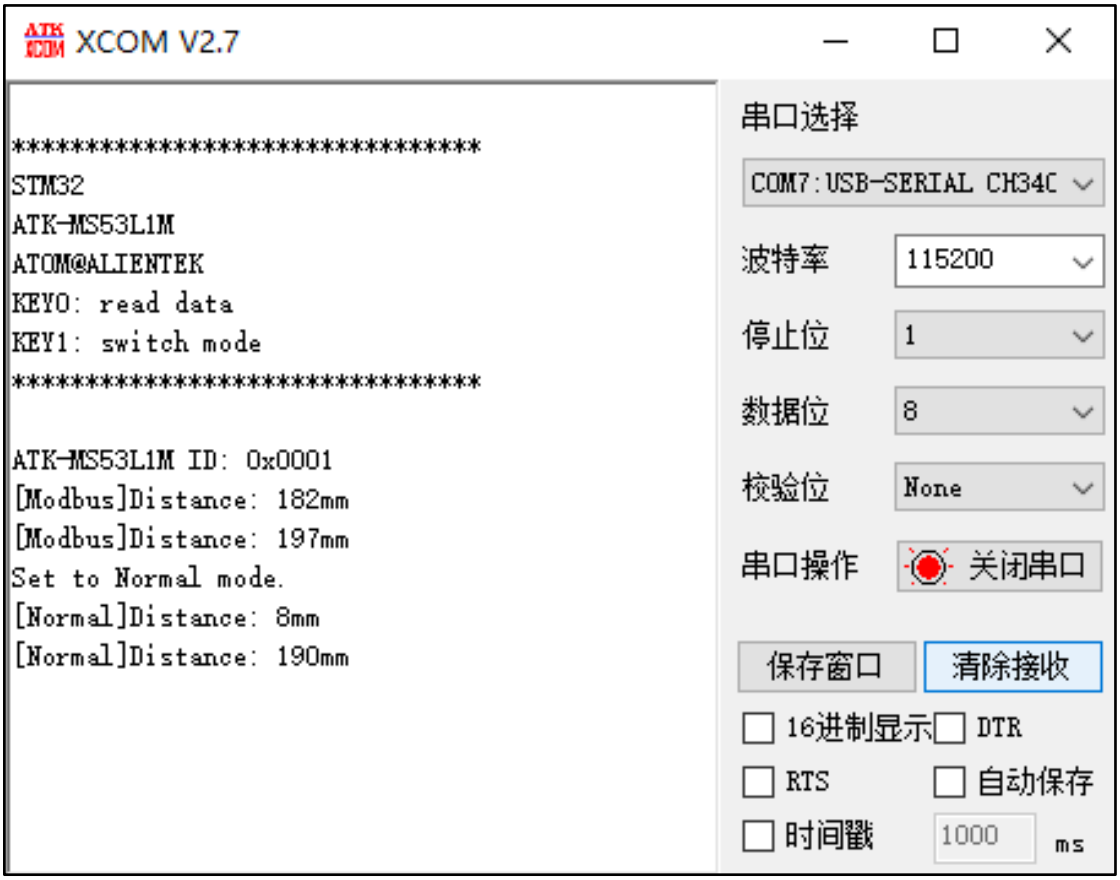

图 2.1.3.4 串口调试助手显示内容三

此时依然可以再次通过按键 1 来切换 ATK-MS53L1M 模块的工作模式。值得一提的时，根据测量场景的不同，测量出的结果也不一定相同。

3，其他

1、 购买地址：

天猫：https://zhengdianyuanzi.tmall.com

淘宝：https://openedv.taobao.com

2、 资料下载

模块资料下载地址：http://www.openedv.com/docs/modules/other/ATK-MS53L1M.html

3、 技术支持

公司网址：www.alientek.com

技术论坛：http://www.openedv.com/forum.php

在线教学：www.yuanzige.com

B 站视频：https://space.bilibili.com/394620890

传真：020-36773971

电话：020-38271790