Application Note

ATK-MS53L0 模块使用说明

高性能激光测距模块

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	2023/03/07	新增阿波罗 F429 与 F767 硬件连接描述

目 录

1，硬件连接

1.1 正点原子 MiniSTM32F103 开发板

ATK-MS53L0 模块可直接与正点原子 MiniSTM32F103 开发板板载的 ATK 模块接口（ATK MODULE）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-MS53L0 模块	VCC	GND	SCL	SDA	INT	XSH
MiniSTM32F103 开发板	3.3V/5V	GND	PD2	PC12	-	PA4

表 1.1.1 ATK-MS53L0 模块与 MiniSTM32F103 开发板连接关系

1.2 正点原子精英 STM32F103 开发板

ATK-MS53L0 模块可直接与正点原子精英 STM32F103 开发板板载的 ATK 模块接口（ATK MODULE）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-MS53L0 模块	VCC	GND	SCL	SDA	INT	XSH
精英 STM32F103 开发板	3.3V/5V	GND	PB11	PB10	-	PA15

表 1.2.1 ATK-MS53L0 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-MS53L0 模块可直接与正点原子战舰 STM32F103 开发板板载的 ATK 模块接口（ATK MODULE）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-MS53L0 模块	VCC	GND	SCL	SDA	INT	XSH
战舰 STM32F103 开发板	3.3V/5V	GND	PB11	PB10	-	PA15

表 1.3.1 ATK-MS53L0 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-MS53L0 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：

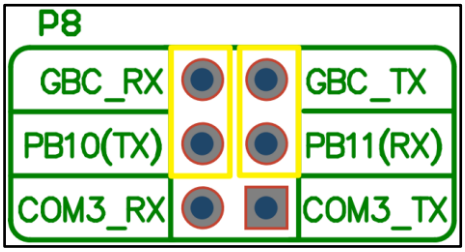

图 1.3.1 战舰 STM32F103 开发板 P8 接线端子

1.4 正点原子探索者 STM32F407 开发板

ATK-MS53L0 模块可直接与正点原子探索者 STM32F407 开发板板载的 ATK 模块接口（ATK MODULE）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-MS53L0 模块	VCC	GND	SCL	SDA	INT	XSH
探索者 STM32F407 开发板	3.3V/5V	GND	PB11	PB10	-	PC0

表 1.4.1 ATK-MS53L0 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-MS53L0 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：

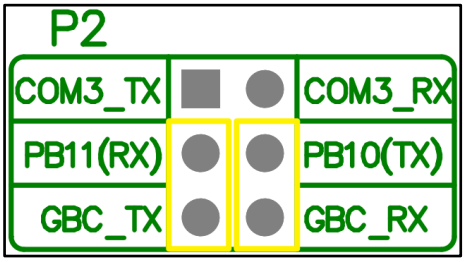

图 1.4.1 探索者 STM32F407 开发板 P2 接线端子

1.5 正点原子 F407 电机控制开发板

ATK-MS53L0 模块可直接与正点原子 F407 电机控制开发板板载的 ATK 模块接口（ATK MODULE）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-MS53L0 模块	VCC	GND	SCL	SDA	INT	XSH
F407 电机控制开发板	3.3V/5V	GND	PC11	PC10	-	PI11

表 1.5.1 ATK-MS53L0 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-MS53L0 模块可直接与正点原子 MiniSTM32H750 开发板板载的 ATK 模块接口（ATK MODULE）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-MS53L0 模块	VCC	GND	SCL	SDA	INT	XSH
MiniSTM32H750 开发板	3.3V/5V	GND	PA3	PA2	-	PC3

表 1.6.1 ATK-MS53L0 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-MS53L0 模块可直接与正点原子阿波罗 STM32F429 开发板板载的 ATK 模块接口（ATK MODULE）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-MS53L0 模块	VCC	GND	SDA	SCL	INT	NC
阿波罗 STM32F429 开发板	3.3V/5V	GND	PB11	PB10	-	-

表 1.7.1 ATK-MS53L0 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-MS53L0 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：

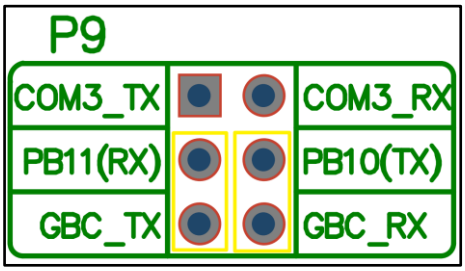

图 1.7.1 阿波罗 STM32F429 开发板 P9 接线端子

1.8 正点原子阿波罗 STM32F767 开发板

ATK-MS53L0 模块可直接与正点原子阿波罗 STM32F767 开发板板载的 ATK 模块接口（ATK MODULE）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-MS53L0 模块	VCC	GND	SDA	SCL	INT	NC
阿波罗 STM32F767 开发板	3.3V/5V	GND	PB10	PB11	-	-

表 1.8.1 ATK-MS53L0 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-MS53L0 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：

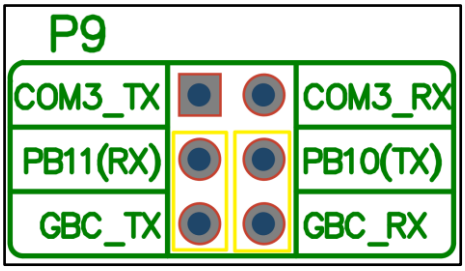

图 1.8.1 阿波罗 STM32F767 开发板 P9 接线端子

2，实验功能

2.1 ATK-MS53L0 模块单次测量测试实验

2.1.1 功能说明

在本实验中，开发板主控芯片通过模拟 IIC 与 ATK-MS53L0 模块进行通讯，从而对 ATK-MS53L0 模块进行配置、控制开启测量和获取测量结果，并将结果通过串口打印至串口调试助手。

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_MS53L0 子文件夹，该文件夹中就包含了 ATK-MS53L0 模块的驱动文件和 ST 官方针对 VL53L0X 提供的 API 库，如下图所示：

```
./Drivers/BSP/ATK_MS53L0/
|-- VL53L0X_1.0.4
|   |-- Api
|   |-- ApiExample
|   |-- LinuxDriverMassMarket_1.0.7
|   `-- doc
|-- atk_ms53l0.c
|-- atk_ms53l0.h
|-- atk_ms53l0_iic.c
`-- atk_ms53l0_iic.h
```

图 2.1.2.1 ATK-MS53L0 模块驱动代码

2.1.2.1 ATK-MS53L0 模块接口驱动

在图 2.1.2.1 中，atk_ms53l0_iic.c 和 atk_ms53l0_iic.h 是开发板与 ATK-MS53L0 模块通讯而使用的模拟 IIC 驱动文件，关于模拟 IIC 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模拟 IIC 对应的章节。

2.1.2.2 ATK-MS53L0 模块驱动

在图 2.1.2.1 中，atk_ms53l0.c 和 atk_ms53l0.h 是 ATK-MS53L0 模块的驱动文件，包含了 ATK-MS53L0 模块的硬件初始化、和硬件复位函数。

1. 函数 atk_ms53l0_hw_init()

该函数用于初始化与 ATK-MS53L0 模块相关的硬件并对 ATK-MS53L0 模块进行硬件复位，具体的代码，如下所示：

```
/**
 * @brief   ATK-MS53L0 模块硬件初始化
 * @param   无
 * @retval  无
 */
void atk_ms53l0_hw_init(void)
{
```

```
    GPIO_InitTypeDef gpio_init_struct = {0};

    ATK_MS53L0_XSH_GPIO_CLK_ENABLE();

    gpio_init_struct.Pin    = ATK_MS53L0_XSH_GPIO_PIN;
    gpio_init_struct.Mode   = GPIO_MODE_OUTPUT_PP;
    gpio_init_struct.Pull   = GPIO_PULLUP;
    gpio_init_struct.Speed  = GPIO_SPEED_FREQ_HIGH;
    HAL_GPIO_Init(ATK_MS53L0_XSH_GPIO_PORT, &gpio_init_struct);

    atk_ms53l0_hw_reset();
}
```

从上面的代码中可以看出，该函数初始化了与 ATK-MS53L0 模块 XSH 引脚连接的 GPIO，该引脚能够控制 ATK-MS53L0 模块进行硬件复位，接着就是控制该 GPIO 对 ATK-MS53L 模块进行硬件复位。

2.1.2.3 ST 官方 API 库

在图 2.1.1 中，子文件夹 VL53L0X_1.0.4 下包含的是 ST 官方针对 VL53L0X 提供的 API 库，使用该 API 库可以很方便的驱动并使用 ATK-MS53L0 模块板载的 VL53L0X 芯片，为了能够在不同的开发板上使用该 API 库驱动 ATK-MS53L0 模块，已经对该 API 库进行了移植，下面介绍如何对该 API 库进行移植。

移植该 API 库仅需涉及两个文件，这两个文件均位于 API 库的./Api/platform/src 目录下，分别为 vl53l0x_platform.c 文件和 vl53l0x_i2c_platform.c。

1. 文件 vl53l0x_platform.c

该文件主要用于提供 API 库通过 IIC 访问 VL53L0X 的操作函数，不用做很大修改，主要要在 vl53l0x_i2c_platform.c 文件中实现该文件调用到的函数，因此具体的修改内容，请查看实验工程中移植好的 vl53l0x_platform.c 文件。

2. 文件 vl53l0x_i2c_platform.c

该文件主要用于为 vl53l0x_platform.c 文件提供 IIC 的操作函数，其他与 IIC 无关的操作函数可不实现，需要实现的函数，如下表所示：

函数	描述
VL53L0X_comms_initialise()	初始化 IIC 通讯端口
VL53L0X_write_multi()	通过 IIC 写多字节数据
VL53L0X_read_multi()	通过 IIC 读多字节数据
VL53L0X_write_byte()	通过 IIC 写 1 字节数据
VL53L0X_write_word()	通过 IIC 写 2 字节数据
VL53L0X_write_dword()	通过 IIC 写 4 字节数据
VL53L0X_read_byte()	通过 IIC 读 1 字节数据
VL53L0X_read_word()	通过 IIC 读 2 字节数据
VL53L0X_read_dword()	通过 IIC 读 4 字节数据

表 2.1.2.3.1 vl53l0x_i2c_platform 文件中移植需要实现的函数

上表列出的函数中，API 库已经调用通过 IIC 读/写多字节数据的函数实现了通过 IIC 读/写 1/2/4 字节数据的函数，因此仅需实现初始化 IIC 通讯端口和通过 IIC 读/写多字节数据的函数即可，如下所示：

```
int32_t VL53L0X_comms_initialise(  uint8_t comms_type,
```

```
                                    uint16_t comms_speed_khz)
{
    atk_ms53l0_iic_init();

    return STATUS_OK;
}

int32_t VL53L0X_write_multi(    uint8_t address,
                                uint8_t index,
                                uint8_t *pdata,
                                int32_t count)
{
    int32_t i;

    atk_ms53l0_iic_start();
    atk_ms53l0_iic_send_byte((address << 1) | 0);
    if (atk_ms53l0_iic_wait_ack() == 1)
    {
        atk_ms53l0_iic_stop();
        return STATUS_FAIL;
    }
    atk_ms53l0_iic_send_byte(index);
    if (atk_ms53l0_iic_wait_ack() == 1)
    {
        atk_ms53l0_iic_stop();
        return STATUS_FAIL;
    }
    for (i=0; i<count; i++)
    {
        atk_ms53l0_iic_send_byte(pdata[i]);
        if (atk_ms53l0_iic_wait_ack() == 1)
        {
            atk_ms53l0_iic_stop();
            return STATUS_FAIL;
        }
    }
    atk_ms53l0_iic_stop();
    return STATUS_OK;
}

int32_t VL53L0X_read_multi(uint8_t address,
                           uint8_t index,
                           uint8_t *pdata,
                           int32_t count)
```

```
{
    atk_ms53l0_iic_start();
    atk_ms53l0_iic_send_byte((address << 1) | 0);
    if (atk_ms53l0_iic_wait_ack() == 1)
    {
        atk_ms53l0_iic_stop();
        return STATUS_FAIL;
    }
    atk_ms53l0_iic_send_byte(index);
    if (atk_ms53l0_iic_wait_ack() == 1)
    {
        atk_ms53l0_iic_stop();
        return STATUS_FAIL;
    }
    atk_ms53l0_iic_start();
    atk_ms53l0_iic_send_byte((address << 1) | 1);
    if (atk_ms53l0_iic_wait_ack() == 1)
    {
        atk_ms53l0_iic_stop();
        return STATUS_FAIL;
    }
    while (count)
    {
        *pdata = atk_ms53l0_iic_read_byte((count > 1) ? 1 : 0);
        count--;
        pdata++;
    }
    atk_ms53l0_iic_stop();
    return STATUS_OK;
}
```

可以看出，以上函数都是由 2.1.2.1 小节“ATK-MS53L0 模块接口驱动”中的 IIC 驱动函数实现的。

2.1.2.4 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;
    VL53L0X_RangingMeasurementData_t data;
```

```
    atk_ms53l0_hw_init();                                   /* 硬件相关初始化 */
    VL53L0X_comms_initialise(NULL, NULL);                   /* 通讯接口初始化 */
    demo_detect_device(&demo_dev, DEMO_DEV_IIC_ADDDR);      /* 检测设备 */
    demo_config_device(&demo_dev);                          /* 配置设备 */

    printf("ATK-MS53L0 Config Succedded!\r\n");

    while (1)
    {
        /* 启动设备测量 */
        VL53L0X_StartMeasurement(&demo_dev);

        /* 等待设备测量完毕 */
        do {
            VL53L0X_GetMeasurementDataReady(&demo_dev, &ret);
        } while (ret != 1);

        /* 清除中断标志 */
        VL53L0X_ClearInterruptMask(&demo_dev, 0);

        /* 获取测量值 */
        VL53L0X_GetRangingMeasurementData(&demo_dev, &data);

        printf("Distance: %dmm\r\n", data.RangeMilliMeter);
    }
}
```

从上面的代码中可以看出，实验测试代码基本都是调用 ST 官方 API 库中的函数，可见 ST 针对 VL53L0X 提供的这个 API 库还是很完善的。实验代码中首先对 ATK-MS53L0 模块进行硬件复位，然后初始化与 ATK-MS53L0 模块通讯的 IIC 接口，接着调用函数 demo_detect_device()和函数 demo_config_device()对 ATK-MS53L0 模块进行检测和配置，检测无误并配置完成后，就能够使用 ATK-MS53L0 模块进行测距了。

下面看一下函数 demo_detect_device()，其代码其下所示：

```
/**
 * @brief   检测设备
 * @param   dev: 设备
 *          iic_addr: 设备 IIC 通讯地址
 * @retval  0: 设备无误
 *          1: 设备有误
 */
static void demo_detect_device(VL53L0X_DEV dev, uint8_t iic_addr)
{
    uint16_t module_id = 0;
```

```
    /* 获取设备模块 ID */
    VL53L0X_RdWord(dev, VL53L0X_REG_IDENTIFICATION_MODEL_ID, &module_id);
    if (module_id != ATK_MS53L0_MODULE_ID)
    {
        printf("ATK-MS53L0 Detect Failed!\r\n");
        while (1)
        {
            LED0_TOGGLE();
            delay_ms(200);
        }
    }

    /* 设置设备 IIC 通讯地址 */
    if (iic_addr != dev->I2cDevAddr)
    {
        VL53L0X_SetDeviceAddress(dev, iic_addr << 1);
        dev->I2cDevAddr = iic_addr;
    }

    /* 设备一次性初始化 */
    VL53L0X_DataInit(dev);
}
```

从上面的代码可以看出，该函数会先获取 ATK-MS53L0 模块的模块 ID，该模块 ID 应为固定的 0xEEAA，如果获取到的模块 ID 与该固定的值不符，说明检测到模块出错了，如果模块无误，那么接下来就配置 ATK-MS53L0 模块的 IIC 通讯地址，这一步不是必须的，因为 ATK-MS53L0 在上电时，其 IIC 通讯地址为固定的 0x29，如果在同一 IIC 总线上连接了多个 ATK-MS53L0 模块时，那么就需要修改其 IIC 的通讯地址了，函数的最后是调用函数 VL53L0X_DataInit()来对 ATK-MS53L0 模块进行初始化。

下面看一下函数 demo_detect_config()，其代码其下所示：

```
/**
 * @brief   配置设备
 * @param   dev: 设备
 * @retval  0: 配置成功
 *          1: 配置失败
 */
static void demo_config_device(VL53L0X_DEV dev)
{
    uint8_t vhvsettings;
    uint8_t phasecal;
    uint32_t refspadcount;
    uint8_t isaperturespads;

    /* 设备基础初始化 */
    VL53L0X_StaticInit(dev);
```

```
    /* 参考校准 */
    VL53L0X_PerformRefCalibration(dev, &vhvsettings, &phasecal);

    /* 参考 Spad 管理 */
    VL53L0X_PerformRefSpadManagement(dev, &refspadcount, &isaperturespads);

    /* 设置设备模式 */
    VL53L0X_SetDeviceMode(dev, DEMO_DEVICE_MODE);

    /* 设置测量时间 */
    VL53L0X_SetMeasurementTimingBudgetMicroSeconds(dev, DEMO_BUDGET_TIME);
}
```

可以看出，该函数就是针对本实验的实验需求，调用 API 库中的函数来配置 ATK-MS53L0 模块，用户在实际使用时，也应针对实际的应用需求来配置 ATK-MS53L0 模块，这里可以参考 ST 提供的 VL53L0 用户手册“um2039-world-smallest-timeofflight-ranging-and-gesture-detection-sensor-application-programming-interface-stmicroelectronics.pdf”和 API 的相关手册“VL53L0X_API_v1.0.4.4960_externalx.pdf”。

2.1.3 实验现象

将 ATK-MS53L0 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：

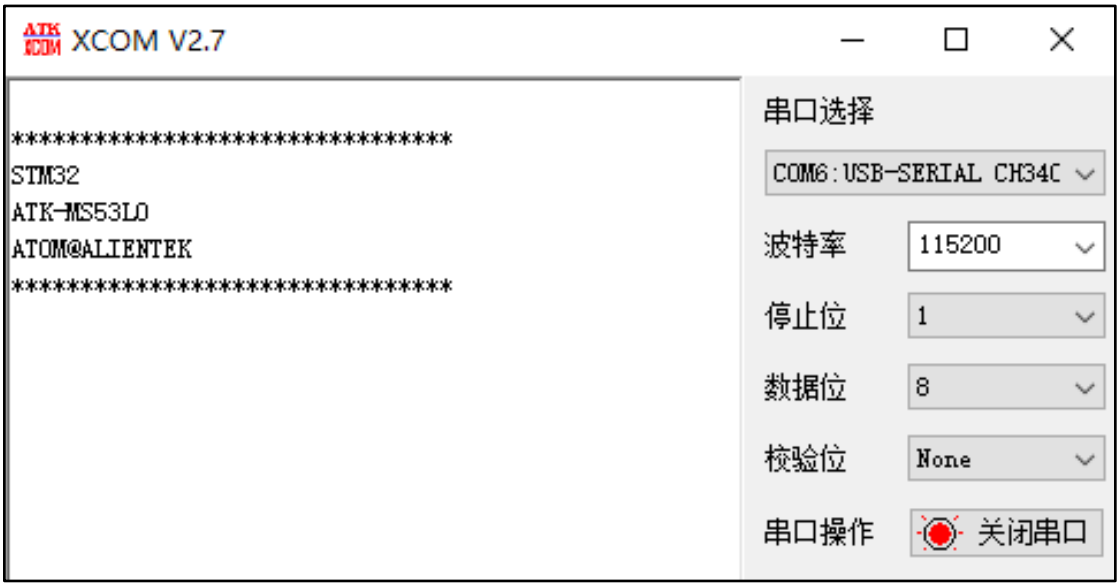

图 2.1.3.2 串口调试助手显示内容一

接下来，如果 ATK-MS53L0 模块初始化并配置成功，则会在串口调试助手上显示相应的提示，如下图所示：

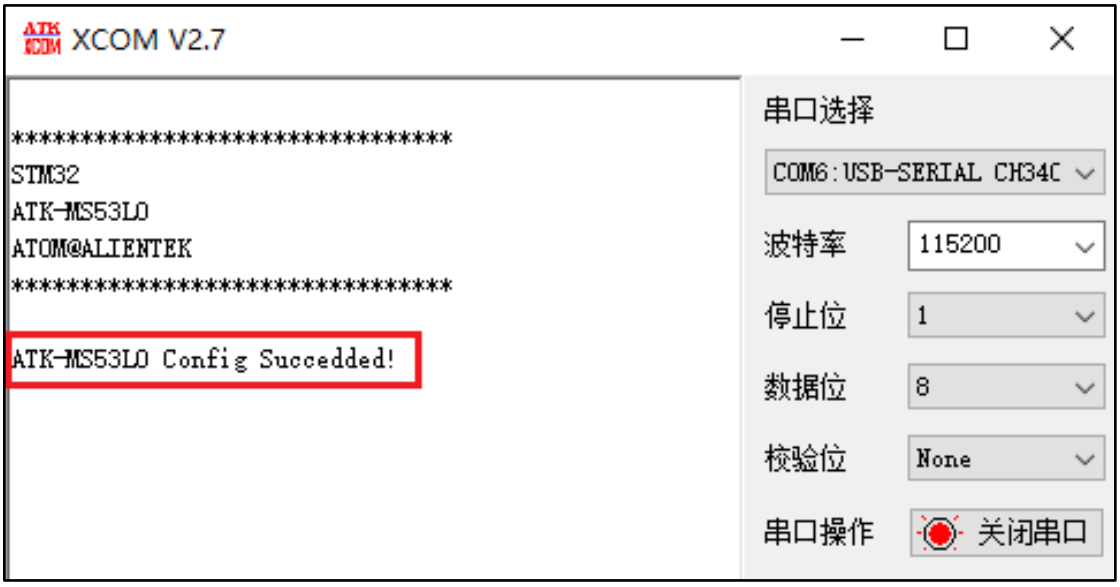

图 2.1.3.3 ATK-MS53L0 模块初始化并配置成功

接下来就能够通过 ATK-MS53L0 模块进行测距了，如下图所示：

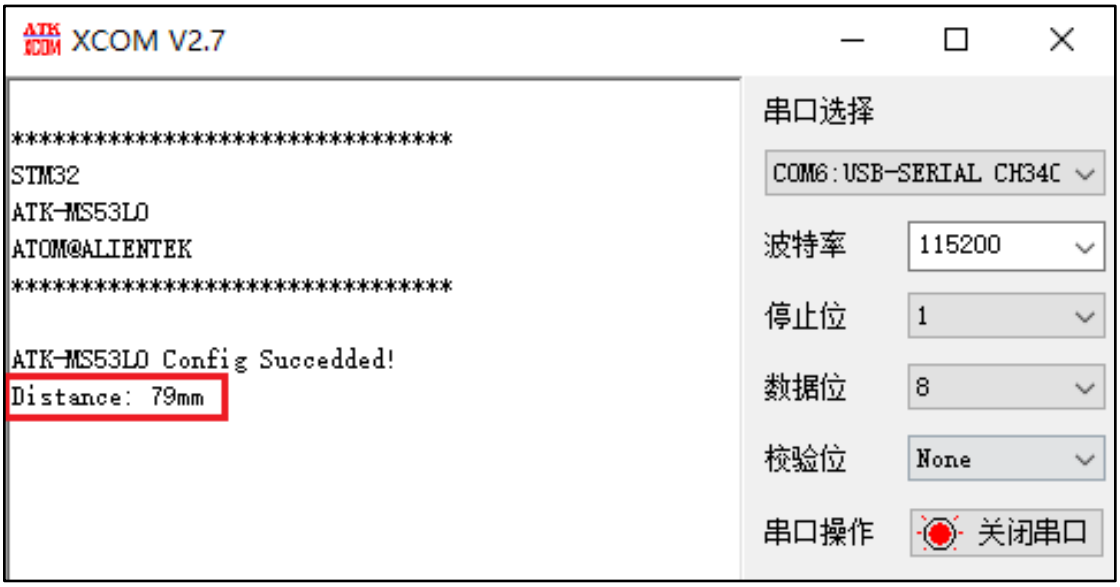

图 2.1.3.4 通过 ATK-MS53L0 模块进行测距

2.2 ATK-MS53L0 模块连续测量测试实验

2.2.1 功能说明

在本实验中，开发板主控芯片通过模拟 IIC 与 ATK-MS53L0 模块进行通讯，从而对 ATK-MS53L0 模块进行配置、控制开启测量和获取测量结果，并将结果通过串口打印至串口调试助手。

2.2.2 源码解读

2.2.2.1 ATK-MS53L0 模块接口驱动

本实验中，ATK-MS53L0 模块接口的驱动代码与 2.1 小节中“ATK-MS53L0 模块单次测量测试实验”中的接口驱动代码一致，请见 2.1.2.1 小节“ATK-MS53L0 模块接口驱动”。

2.2.2.2 ATK-MS53L0 模块驱动

本实验中，ATK-MS53L0 模块的驱动代码与 2.1 小节中“ATK-MS53L0 模块单次测量测试实验”中的驱动代码一致，请见 2.1.2.2 小节“ATK-MS53L0 模块驱动”。

2.2.2.3 ST 官方 API 库

本实验中，ST 官方 API 库与 2.1 小节中“ATK-MS53L0 模块单次测量测试实验”中的 ST 官方 API 库一致，请见 2.1.2.3 小节“ST 官方 API 库”。

2.2.2.4 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;
    VL53L0X_RangingMeasurementData_t data;

    atk_ms53l0_hw_init();                                   /* 硬件相关初始化 */
    VL53L0X_comms_initialise(NULL, NULL);                   /* 通讯接口初始化 */
    demo_detect_device(&demo_dev, DEMO_DEV_IIC_ADDDR);      /* 检测设备 */
    demo_config_device(&demo_dev);                          /* 配置设备 */

    printf("ATK-MS53L0 Config Succedded!\r\n");

    /* 启动设备测量 */
    VL53L0X_StartMeasurement(&demo_dev);

    while (1)
    {
        /* 等待设备测量完毕 */
        do {
            VL53L0X_GetMeasurementDataReady(&demo_dev, &ret);
        } while (ret != 1);

        /* 清除中断标志 */
        VL53L0X_ClearInterruptMask(&demo_dev, 0);

        /* 获取测量值 */
        VL53L0X_GetRangingMeasurementData(&demo_dev, &data);

        printf("Distance: %dmm\r\n", data.RangeMilliMeter);
    }
}
```

从上面的代码中可以看出，本实验的实验测试代码与第 2.1 小节中“ATK-MS53L0 模块单次测量测试实验”的实验测试代码基本一致，不同之处在于，在函数 demo_config_device()中将 ATK-MS53L0 模块配置为连续测量模式，因此进仅需调用一次启动设备测量的函数，ATK-MS53L0 模块便会连续地进行距离测量。

2.2.3 实验现象

实验现象与第 2.1 小节中“ATK-MS53L0 模块单次测量测试实验”的实验现象基本一致，不同之处在于在 ATK-MS53L0 模块进行一次距离测量后，无需再次手动开启测量，ATK-MS53L0 模块会连续地进行距离测量。

2.3 ATK-MS53L0 模块连续定时测量测试实验

2.3.1 功能说明

在本实验中，开发板主控芯片通过模拟 IIC 与 ATK-MS53L0 模块进行通讯，从而对 ATK-MS53L0 模块进行配置、控制开启测量和获取测量结果，并将结果通过串口打印至串口调试助手。

2.3.2 源码解读

2.3.2.1 ATK-MS53L0 模块接口驱动

本实验中，ATK-MS53L0 模块接口的驱动代码与 2.1 小节中“ATK-MS53L0 模块单次测量测试实验”中的接口驱动代码一致，请见 2.1.2.1 小节“ATK-MS53L0 模块接口驱动”。

2.3.2.2 ATK-MS53L0 模块驱动

本实验中，ATK-MS53L0 模块的驱动代码与 2.1 小节中“ATK-MS53L0 模块单次测量测试实验”中的驱动代码一致，请见 2.1.2.2 小节“ATK-MS53L0 模块驱动”。

2.3.2.3 ST 官方 API 库

本实验中，ST 官方 API 库与 2.1 小节中“ATK-MS53L0 模块单次测量测试实验”中的 ST 官方 API 库一致，请见 2.1.2.3 小节“ST 官方 API 库”。

2.3.2.4 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;
    VL53L0X_RangingMeasurementData_t data;

    atk_ms53l0_hw_init();                                   /* 硬件相关初始化 */
    VL53L0X_comms_initialise(NULL, NULL);                   /* 通讯接口初始化 */
    demo_detect_device(&demo_dev, DEMO_DEV_IIC_ADDDR);      /* 检测设备 */
    demo_config_device(&demo_dev);                          /* 配置设备 */

    printf("ATK-MS53L0 Config Succedded!\r\n");

    /* 启动设备测量 */
    VL53L0X_StartMeasurement(&demo_dev);

    while (1)
    {
        /* 等待设备测量完毕 */
        do {
            VL53L0X_GetMeasurementDataReady(&demo_dev, &ret);
        } while (ret != 1);
```

```
        /* 清除中断标志 */
        VL53L0X_ClearInterruptMask(&demo_dev, 0);

        /* 获取测量值 */
        VL53L0X_GetRangingMeasurementData(&demo_dev, &data);

        printf("Distance: %dmm\r\n", data.RangeMilliMeter);
    }
}
```

从上面的代码中可以看出，本实验的实验测试代码与第 2.1 小节中“ATK-MS53L0 模块单次测量测试实验”的实验测试代码基本一致，不同之处在于，在函数 demo_config_device()中将 ATK-MS53L0 模块配置为连续定时测量模式，并配置了定时时间为 1000ms，因此进仅需调用一次启动设备测量的函数，ATK-MS53L0 模块便会连续地每 1000ms 就进行一次距离测量。

2.3.3 实验现象

实验现象与第 2.1 小节中“ATK-MS53L0 模块单次测量测试实验”的实验现象基本一致，不同之处在于 ATK-MS53L0 模块进行一次距离测量后，无需再次手动开启测量，ATK-MS53L0 模块会连续地每间隔 1000ms 就进行一次距离测量。

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-VL53L0X.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

正点原子