Application Note

ATK-MO1218 模块使用说明

高性能 GPS/北斗模块

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

目 录

1，硬件连接

1.1 正点原子 MiniSTM32F103 开发板

ATK-MO1218 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-MO1218 模块	VCC	GND	TXD	RXD	PPS
MiniSTM32F103 开发板	5V	GND	PD2	PC12	PC4

表 1.1.1 ATK-MO1218 模块与 MiniSTM32F103 开发板连接关系

1.2 正点原子精英 STM32F103 开发板

ATK-MO1218 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-MO1218 模块	VCC	GND	TXD	RXD	PPS
精英 STM32F103 开发板	5V	GND	PB11	PB10	PA4

表 1.2.1 ATK-MO1218 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-MO1218 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-MO1218 模块	VCC	GND	TXD	RXD	PPS
战舰 STM32F103 开发板	5V	GND	PB11	PB10	PA4

表 1.3.1 ATK-MO1218 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-MO1218 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：

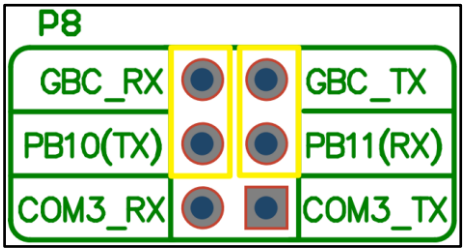

图 1.3.1 战舰 STM32F103 开发板 P8 接线端子

1.4 正点原子探索者 STM32F407 开发板

ATK-MO1218 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-MO1218 模块	VCC	GND	TXD	RXD	PPS
探索者 STM32F407 开发板	5V	GND	PB11	PB10	PF6

表 1.4.1 ATK-MO1218 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-MO1218 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：

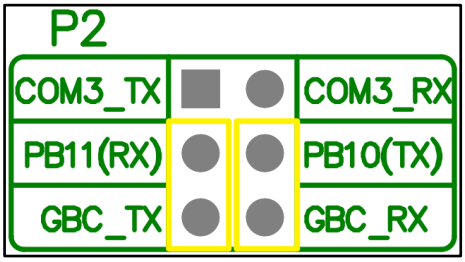

图 1.4.1 探索者 STM32F407 开发板 P2 接线端子

1.5 正点原子 F407 电机控制开发板

ATK-MO1218 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-MO1218 模块	VCC	GND	TXD	RXD	PPS
F407 电机控制开发板	5V	GND	PC11	PC10	PI10

表 1.5.1 ATK-MO1218 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-MO1218 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-MO1218 模块	VCC	GND	TXD	RXD	PPS
MiniSTM32H750 开发板	5V	GND	PA3	PA2	PC2

表 1.6.1 ATK-MO1218 模块与 MiniSTM32H750 开发板连接关系

2，实验功能

2.1 ATK-MO1218 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 接口与 ATK-MO1218 模块进行通讯，从而完成对 ATK-MO1218 模块的配置、数据获取等功能，在配置好 ATK-MO1218 模块后，会获取并解析 ATK-MO1218 模块输出的信息，从而获取到 UTC 时间、海拔高度、速度、位置经纬度、卫星数量等信息，并将这些信息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_MO1218 子文件夹，该文件夹中就包含了 ATK-MO1218 模块的驱动文件，如下图所示：

```
./Drivers/BSP/ATK_MO1218/
|-- atk_mo1218.c
|-- atk_mo1218.h
|-- atk_mo1218_bin_msg.c
|-- atk_mo1218_bin_msg.h
|-- atk_mo1218_nmea_msg.c
|-- atk_mo1218_nmea_msg.h
|-- atk_mo1218_uart.c
`-- atk_mo1218_uart.h
```

图 2.1.2.1 ATK-MO1218 模块驱动代码

2.1.2.1 ATK-MO1218 模块接口驱动

在图 2.1.2.1 中，atk_mo1218_uart.c 和 atk_mo1218_uart.h 是开发板与 ATK-MO1218 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-MO1218 模块通过 UART 发送给主控芯片的数据的长度是不固定的，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-MO1218 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的传输是否完成，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-MO1218 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-MO1218 模块 SkyTraq binary 协议驱动

在图 2.1.2.1 中，atk_mo1218_bin_msg.c 和 atk_mo1218_bin_msg.h 是 ATK-MO1218 模块的 SkyTraq binary 协议驱动文件，该协议主要用于配置 ATK-MO1218 模块，以及获取 ATK-MO1218 模块的各项配置参数，SkyTraq binary 协议的具体内容，请见《ATK-MO1218 模块用户手册》。

2.1.2.3 ATK-MO1218 模块 NMEA-0183 协议驱动

在图 2.1.2.1 中，atk_mo1218_nmea_msg.c 和 atk_mo1218_nmea_msg.h 是 ATK-MO1218 模块的 NMEA-0183 协议驱动文件，该协议主要用于获取 ATK-MO1218 模块输出的各种定

位信息，其中就包括 UTC 时间、海拔高度、速度、位置经纬度、卫星数量等信息，NMEA-0183 协议的具体内容，请见《ATK-MO1218 模块用户手册》。

2.1.2.4 ATK-MO1218 模块驱动

在图 2.1.2.1 中，atk_mo1218.c 和 atk_mo1218.h 是 ATK-MO1218 模块的驱动文件，包含了 ATK-MO1218 模块初始化以及获取 ATK-MO1218 模块输出的数据信息这两个 API 函数，下面分别介绍这两个函数。

1. 函数 atk_mo1218_init()

该函数用于初始化 ATK-MO1218 模块，具体的代码，如下所示：

```
/**
 * @brief   ATK-MO1218 初始化
 * @param   baudrate: ATK-MO1218 UART 通讯波特率
 * @retval  ATK_MO1218_EOK      : ATK-MO1218 初始化成功
 *          ATK_MO1218_ERROR    : ATK-MO1218 初始化失败
 */
uint8_t atk_mo1218_init(uint32_t baudrate)
{
    uint8_t ret;
    atk_mo1218_sw_version_t version;

    atk_mo1218_uart_init(baudrate);
    ret = atk_mo1218_get_sw_version(&version);
    if (ret != ATK_MO1218_EOK)
    {
        return ATK_MO1218_ERROR;
    }

    return ATK_MO1218_EOK;
}
```

从上面的代码中可以看出，函数 atk_mo1218_init()首先就是初始化与 ATK-MO1218 模块通讯的 UART，接着通过尝试获取 ATK-MO1218 模块的软件版本来判断与 ATK-MO1218 模块的通讯是否正常。

2. 函数 atk_mo1218_update()

该函数用于获取 ATK-MO1218 模块输出的各项最新数据，其中就包括 UTC 时间、位置经纬度、海拔高度、速度等信息，具体的代码，如下所示：

```
/**
 * @brief   获取并更新 ATK-MO1218 模块数据
 * @param   utc                 : UTC 时间
 *          position            : 位置信息（经纬度扩大 100000 倍）
 *          altitude            : 海拔高度（扩大 10 倍），单位：米
 *          speed               : 地面速度（扩大 10 倍），单位：千米/时
 *          fix_info            : 定位信息
 *          gps_satellite_info  : 可见 GPS 卫星信息
 *          beidou_satellite_info : 可见北斗卫星信息
 *          timeout             : 等待超时时间，单位：1 毫秒
```

```
 * @retval  ATK_MO1218_EOK      : 获取并更新 ATK-MO1218 模块数据成功
 *          ATK_MO1218_EINVAL   : 函数参数错误
 *          ATK_MO1218_ETIMEOUT : 等待超时
 */
uint8_t atk_mo1218_update(
    atk_mo1218_time_t *utc,
    atk_mo1218_position_t *position,
    int16_t *altitude,
    uint16_t *speed,
    atk_mo1218_fix_info_t *fix_info,
    atk_mo1218_visible_satellite_info_t *gps_satellite_info,
    atk_mo1218_visible_satellite_info_t *beidou_satellite_info,
    uint32_t timeout)
{
    uint8_t ret;
    uint8_t *buf;
    uint8_t *nmea;
    struct
    {
        atk_mo1218_nmea_gga_msg_t msg;
        uint8_t done;
    } gngga;
    struct
    {
        atk_mo1218_nmea_gsa_msg_t msg;
        uint8_t done;
    } gngsa;
    struct
    {
        atk_mo1218_nmea_gsv_msg_t msg;
        uint8_t done;
    } gpgsv;
    struct
    {
        atk_mo1218_nmea_gsv_msg_t msg;
        uint8_t done;
    } bdgsv;
    struct
    {
        atk_mo1218_nmea_rmc_msg_t msg;
        uint8_t done;
    } gnrmc;
    struct
    {
```

```
        atk_mo1218_nmea_vtg_msg_t msg;
        uint8_t done;
    } gnvtg;
    uint8_t satellite_index;

    if ( (utc == NULL) &&
         (position == NULL) &&
         (altitude == NULL) &&
         (speed == NULL) &&
         (fix_info == NULL) &&
         (gps_satellite_info == NULL) &&
         (beidou_satellite_info == NULL))
    {
        return ATK_MO1218_EINVAL;
    }

    gngga.done = 0;
    gngsa.done = 0;
    gpgsv.done = 0;
    bdgsv.done = 0;
    gnrmc.done = 0;
    gnvtg.done = 0;

    atk_mo1218_uart_rx_restart();
    while (timeout > 0)
    {
        buf = atk_mo1218_uart_rx_get_frame();
        if (buf != NULL)
        {
            /* GNGGA */
            if (((altitude != NULL) || (fix_info != NULL)) && (gngga.done == 0))
            {
                ret = atk_mo1218_get_nmea_msg_from_buf(
                        buf,
                        ATK_MO1218_NMEA_MSG_GNGGA,
                        0,
                        &nmea);
                if (ret == ATK_MO1218_EOK)
                {
                    ret = atk_mo1218_decode_nmea_xxgga(nmea, &gngga.msg);
                    if (ret == ATK_MO1218_EOK)
                    {
                        gngga.done = ~0;
                        if (altitude != NULL)
```

```
                {
                    *altitude = gngga.msg.altitude;
                }
                if (fix_info != NULL)
                {
                    fix_info->quality = gngga.msg.gps_quality;
                    fix_info->satellite_num = gngga.msg.satellite_num;
                }
            }
        }
    }
    else
    {
        gngga.done = ~0;
    }

    /* GNGSA */
    if ((fix_info != NULL) && (gngsa.done == 0))
    {
        ret = atk_mo1218_get_nmea_msg_from_buf(
                buf,
                ATK_MO1218_NMEA_MSG_GNGSA,
                0,
                &nmea);
        if (ret == ATK_MO1218_EOK)
        {
            ret = atk_mo1218_decode_nmea_xxgsa(nmea, &gngsa.msg);
            if (ret == ATK_MO1218_EOK)
            {
                gngsa.done = ~0;
                fix_info->type = gngsa.msg.type;
                for (   satellite_index=0;
                        satellite_index<12;
                        satellite_index++)
                {
                    fix_info->satellite_id[satellite_index] =
                            gngsa.msg.satellite_id[satellite_index];
                }
                fix_info->pdop = gngsa.msg.pdop;
                fix_info->hdop = gngsa.msg.hdop;
                fix_info->vdop = gngsa.msg.vdop;
            }
        }
    }
```

```
        else
        {
            gngsa.done = ~0;
        }

        /* GPGSV */
        if ((gps_satellite_info != NULL) && (gpgsv.done == 0))
        {
            ret = atk_mo1218_get_nmea_msg_from_buf(
                    buf,
                    ATK_MO1218_NMEA_MSG_GPGSV,
                    0,
                    &nmea);
            if (ret == ATK_MO1218_EOK)
            {
                ret = atk_mo1218_decode_nmea_xxgsv(nmea, &gpgsv.msg);
                if (ret == ATK_MO1218_EOK)
                {
                    gpgsv.done = ~0;
                    gps_satellite_info->satellite_num =
                                                gpgsv.msg.satellite_view;
                    for (   satellite_index=0;
                            satellite_index<gpgsv.msg.satellite_view;
                            satellite_index++)
                    {
gps_satellite_info->satellite_info[satellite_index].satellite_id =
                gpgsv.msg.satellite_info[satellite_index].satellite_id;
gps_satellite_info->satellite_info[satellite_index].elevation =
                gpgsv.msg.satellite_info[satellite_index].elevation;
gps_satellite_info->satellite_info[satellite_index].azimuth =
                gpgsv.msg.satellite_info[satellite_index].azimuth;
gps_satellite_info->satellite_info[satellite_index].snr =
                gpgsv.msg.satellite_info[satellite_index].snr;
                    }
                }
            }
        }
        else
        {
            gpgsv.done = ~0;
        }

        /* BDGSV */
        if ((beidou_satellite_info != NULL) && (bdgsv.done == 0))
```

```
        {
            ret = atk_mo1218_get_nmea_msg_from_buf(
                    buf,
                    ATK_MO1218_NMEA_MSG_BDGSV,
                    0,
                    &nmea);
            if (ret == ATK_MO1218_EOK)
            {
                ret = atk_mo1218_decode_nmea_xxgsv(nmea, &bdgsv.msg);
                if (ret == ATK_MO1218_EOK)
                {
                    bdgsv.done = ~0;
                    beidou_satellite_info->satellite_num =
                                                bdgsv.msg.satellite_view;
                    for (   satellite_index=0;
                            satellite_index<bdgsv.msg.satellite_view;
                            satellite_index++)
                    {
beidou_satellite_info->satellite_info[satellite_index].satellite_id =
                bdgsv.msg.satellite_info[satellite_index].satellite_id;
beidou_satellite_info->satellite_info[satellite_index].elevation =
                bdgsv.msg.satellite_info[satellite_index].elevation;
beidou_satellite_info->satellite_info[satellite_index].azimuth =
                bdgsv.msg.satellite_info[satellite_index].azimuth;
beidou_satellite_info->satellite_info[satellite_index].snr =
                bdgsv.msg.satellite_info[satellite_index].snr;
                    }
                }
            }
        }
        else
        {
            bdgsv.done = ~0;
        }

        /* GNRMC */
        if (((utc != NULL) || (position != NULL)) && (gnrmc.done == 0))
        {
            ret = atk_mo1218_get_nmea_msg_from_buf(
                    buf,
                    ATK_MO1218_NMEA_MSG_GNRMC,
                    0,
                    &nmea);
            if (ret == ATK_MO1218_EOK)
```

```
        {
            ret = atk_mo1218_decode_nmea_xxrmc(nmea, &gnrmc.msg);
            if (ret == ATK_MO1218_EOK)
            {
                gnrmc.done = ~0;
                if (utc != NULL)
                {
                    utc->year = gnrmc.msg.utc_date.year;
                    utc->month = gnrmc.msg.utc_date.month;
                    utc->day = gnrmc.msg.utc_date.day;
                    utc->hour = gnrmc.msg.utc_time.hour;
                    utc->minute = gnrmc.msg.utc_time.minute;
                    utc->second = gnrmc.msg.utc_time.second;
                    utc->millisecond = gnrmc.msg.utc_time.millisecond;
                }
                if (position != NULL)
                {
                    position->latitude.degree =
                                        gnrmc.msg.latitude.degree;
                    position->latitude.indicator =
                                        gnrmc.msg.latitude.indicator;
                    position->longitude.degree =
                                        gnrmc.msg.longitude.degree;
                    position->longitude.indicator =
                                        gnrmc.msg.longitude.indicator;
                }
            }
        }
    }
    else
    {
        gnrmc.done = ~0;
    }

    /* GNVTG */
    if ((speed != NULL) && (gnvtg.done == 0))
    {
        ret = atk_mo1218_get_nmea_msg_from_buf(
                buf,
                ATK_MO1218_NMEA_MSG_GNVTG,
                0,
                &nmea);
        if (ret == ATK_MO1218_EOK)
        {
```

```
                    gnvtg.done = ~0;
                    ret = atk_mo1218_decode_nmea_xxvtg(nmea, &gnvtg.msg);
                    if (ret == ATK_MO1218_EOK)
                    {
                        *speed = gnvtg.msg.speed_kph;
                    }
                }
            }
            else
            {
                gnvtg.done = ~0;
            }
        }

        if (    (gngga.done != 0) &&
                (gngsa.done != 0) &&
                (gpgsv.done != 0) &&
                (bdgsv.done != 0) &&
                (gnrmc.done != 0) &&
                (gnvtg.done != 0))
        {
            return ATK_MO1218_EOK;
        }

        timeout--;
        delay_ms(1);
    }

    return ATK_MO1218_ETIMEOUT;
}
```

从上面的代码中可以看出，函数 atk_mo1218_update()是通过解析 NMEA-0183 协议语句来获取 ATK-MO1218 模块输出的各种数据信息的，通过该函数就能够很方便的获取到 ATK-1218 模块输出的各种数据信息，值得一提的是，调用该函数时，并不用传入所有入参，只需要根据需要的数据，在对应的入参传入相应数据结构变量的地址即可，不需获取的数据信息传入“NULL”即可。

2.1.2.5 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;

    /* 初始化 ATK-MO1218 模块 */
    ret = atk_mo1218_init(38400);
    if (ret != 0)
    {
        printf("ATK-MO1218 init failed!\r\n");
        while (1)
        {
            LED0_TOGGLE();
            delay_ms(200);
        }
    }

    /* 配置 ATK-MO1218 模块 */
    ret  = atk_mo1218_factory_reset(ATK_MO1218_FACTORY_RESET_REBOOT);
    ret += atk_mo1218_config_output_type(   ATK_MO1218_OUTPUT_NMEA,
                                            ATK_MO1218_SAVE_SRAM_FLASH);
    ret += atk_mo1218_config_nmea_msg(  1,
                                        1,
                                        1,
                                        1,
                                        1,
                                        1,
                                        0,
                                        ATK_MO1218_SAVE_SRAM_FLASH);
    ret += atk_mo1218_config_position_rate(ATK_MO1218_POSITION_RATE_5HZ,
                                           ATK_MO1218_SAVE_SRAM_FLASH);
    ret += atk_mo1218_config_gnss_for_navigation(
                                               ATK_MO1218_GNSS_GPS_BEIDOU,
                                               ATK_MO1218_SAVE_SRAM_FLASH);
    if (ret != 0)
    {
        printf("ATK-MO1218 configure failed!\r\n");
        while (1)
        {
            LED0_TOGGLE();
            delay_ms(200);
        }
    }

    while (1)
    {
```

```
        uint8_t ret;
        atk_mo1218_time_t utc;
        atk_mo1218_position_t position;
        int16_t altitude;
        uint16_t speed;
        atk_mo1218_fix_info_t fix_info;
        atk_mo1218_visible_satellite_info_t gps_satellite_info = {0};
        atk_mo1218_visible_satellite_info_t beidou_satellite_info = {0};
        uint8_t satellite_index;

        while (1)
        {
            /* 获取并更新 ATK-MO1218 模块数据 */
            ret = atk_mo1218_update(    &utc,
                                        &position,
                                        &altitude,
                                        speed,
                                        &fix_info,
                                        NULL,
                                        NULL,
                                        5000);
            if (ret == ATK_MO1218_EOK)
            {
                /* 打印数据，代码省略
                 * 具体请查看工程
                 */
            }
            else
            {
                /* ATK-MO1218 模块未定位时，
                 * 不输出 NMEA 协议的 GSV 语句，
                 * 导致因获取不到可见 GPS、北斗卫星的信息而超时失败，
                 * 此时可将函数 atk_mo1218_update()的入参
                 * gps_satellite_info 和 beidou_satellite_info
                 * 传入 NULL，从而获取未定位时的其他数据
                 */
                printf("Error!\r\n");
            }

            delay_ms(1000);
        }
    }
}
```

从上面的代码中可以看出，整个测试代码的逻辑还是比较简单的，主要就是先调用函数

atk_mo1218_init()进行初始化，接着就调用 SkyTraq binary 协议的 API 函数来配置 ATK-MO1218 模块，配置无误后，ATK-MO1218 模块便会根据配置的测量频率不断的输出数据，接下来就调用函数 atk_mo1218_update()来获取 ATK-MO1218 模块输出的各个数据信息，然后打印至串口调试助手。

2.1.3 实验现象

将 ATK-MO1218 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

同时，通过串口调试助手输出实验信息，如下图所示：

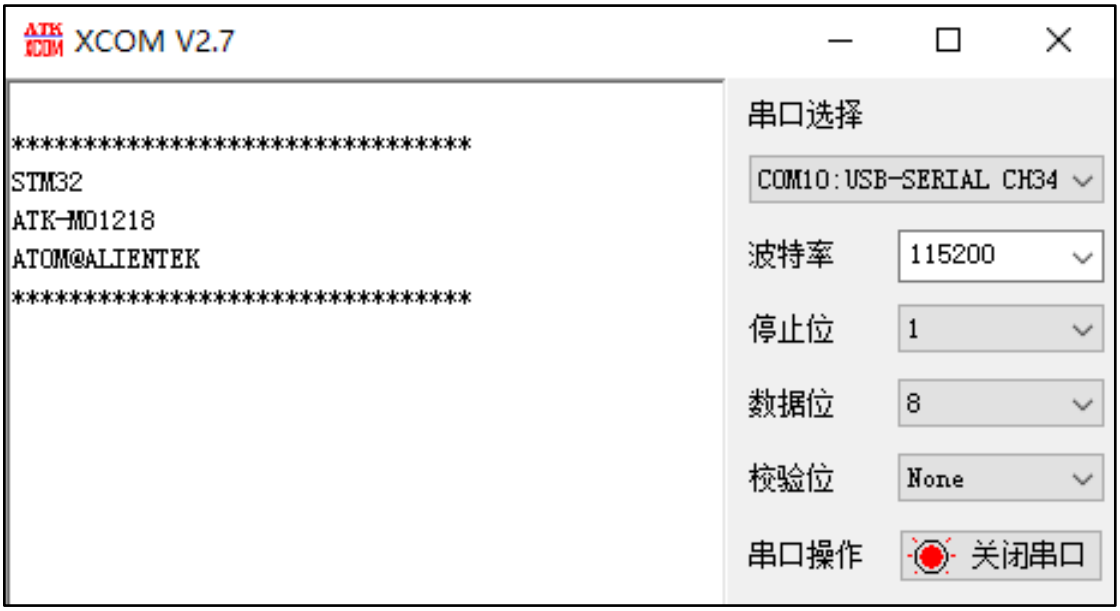

图 2.1.3.2 串口调试助手显示内容

接下来，程序会初始化 ATK-MO1218 模块并自动通过 SkyTraq binary 协议配置 ATK-MO1218 模块，配置无误后，就会自动解析 ATK-MO1218 模块输出的 NMEA-0183 协议语句，从而获取 ATK-MO1218 模块输出的各个数据信息，若解析无误，便会将获取到的数据信息打印至串口调试助手，如下图所示：

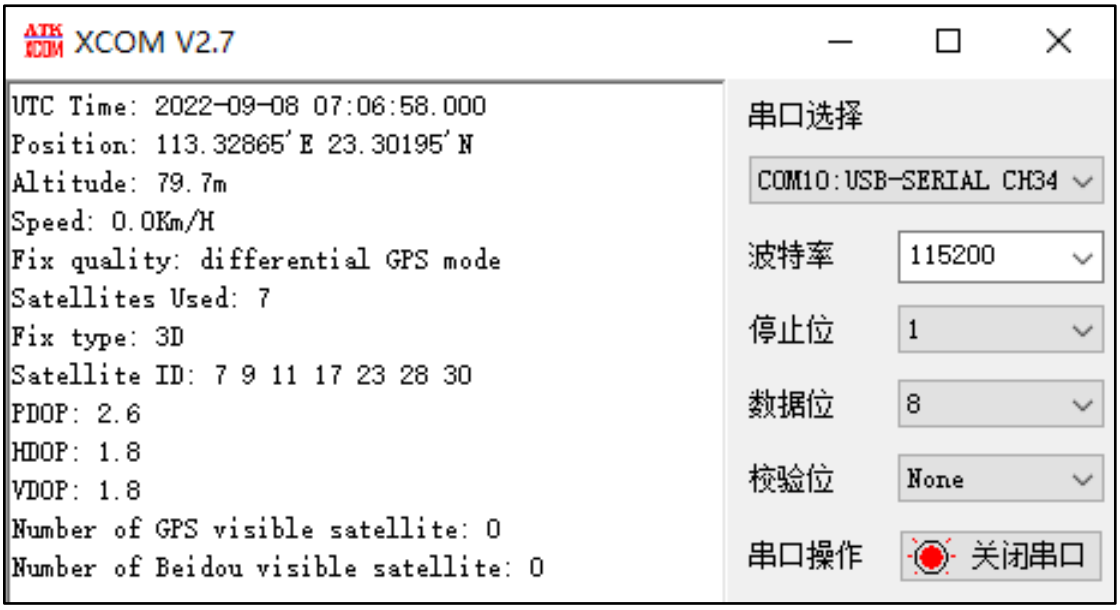

图 2.1.3.3 打印 ATK-MO1218 模块输出的各个数据信息

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-1218-BD.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