千寻魔方 Native SDK

集成开发指南

免责声明：

千寻位置网络有限公司拥有随时修改本手册的权利，内容如有更改，恕不另行通知。包括但不限于对产品特定用途适用性和适销性的隐含保证。

千寻位置网络有限公司对本手册中包含的错误或对本手册的使用所带来的偶然或继起损害不承担任何责任。

文档修改记录

版本	修改日期	描述
V1.0	2018-05-08	初版发布
V1.1	2018-07-04	修正部分函数、结构体命名
V1.2	2018-07-06	增加 DR 相关接口、新增完好性输出参数
V1.3	2018-10-25	新增算法场景、专网配置参数说明

目录

1 SDK 接口结构

```
typedef struct
{
    size_t   size;
    int      (*init)(QXWZSdkCallbacks *callbacks, QXWZSdkConfig *config);
    int      (*start)(void);
    int      (*stop)(void);
    int      (*cleanup)(void);
    void     (*update_conn_status)(QXWZNetworkStatus networkType);
    void     (*set_position_mode)(QXWZPosMode mode, float frequency);
    void     (*delete_aiding_data)(QXGNSSAidingData flags);
    int      (*inject_gnss_data)(unsigned char *buffer, size_t len);
    int      (*inject_gnss_pps)(QXWZGNSSPPS *pps);
    int      (*inject_gnss_ort)(QXWZGNSSOrt *ort);
    int      (*inject_sensor_config)(QXWZSensorConfig *config);
    int      (*inject_sensor_data)(QXWZSensorData *sensorData);
    int      (*inject_map_matching_para)(QXWZMapMatchingPara *mapParams);
    int      (*set_lever_arm_para)(QXWZLeverArmParams *lam);
} QXWZSdkInterface;
```

蓝色标注的是 DR 相关接口，若无需注入外部传感器数据（如车速），可暂不实现。

1.1 获取 SDK 接口

getQXWZSdkInterface

定义	const QXWZSdkInterface* getQXWZSdkInterface(void);
参数	无
返回值	指向 QXWZSdkInterface 结构体的指针，包含 SDK 各接口。

说明	获取千寻 SDK 接口。

1.2 接口介绍

size

size	类型 size_t，表示结构体大小。

init

定义	int (*init)(QXWZSdkCallbacks *callbacks, QXWZSdkConfig *config);
参数	QXWZSdkCallbacks 结构体指针，回调函数集合。 QXWZSdkConfig 结构体指针，账号、SDK 工作目录等配置参数。
返回值	成功：0，失败：-1
说明	初始化千寻 SDK，注册回调函数。

start

定义	int (*start)(void);
参数	void
返回值	成功：0，失败：-1
说明	启动 SDK 定位服务。

stop

定义	int (*stop)(void);
参数	void

返回值	成功：0，失败：-1
说明	停止 SDK 定位服务。

cleanup

定义	int (*cleanup)(void);
参数	无
返回值	成功：0，失败：-1
说明	注销千寻 SDK，回收资源。

update_conn_status

定义	void (*update_conn_status)(QXWZNetworkStatus networkType);
参数	网络状态类型，具体参照 QXWZNetworkStatus 定义。
返回值	无
说明	在设备网络连接状态发生变化时及时通知 SDK 网络状态。

inject_gnss_data

定义	int (*inject_gnss_data)(unsigned char *buffer, size_t len);
参数	buffer: 存储 GNSS 数据的 buffer；len: 存储 GNSS 消息有效长度。
返回值	成功：0，失败：-1
说明	注入 GNSS 数据，至少包含 RXM-RAWX 等消息的 ubx 消息数据流。

inject_gnss_pps

定义	int (*inject_gnss_pps)(QXWZGNSSPPS *pps);

参数	GNSS PPS 秒脉冲。
返回值	成功：0，失败：-1
说明	注入 GNSS PPS 秒脉冲。当第一次采集到秒脉冲之后，必须保持注入频率为 1Hz。GPS 不定位时同样需要注入，需将 QXWZGNSSPPS 结构体 Valid 字段设为 0。

inject_gnss_ort

定义	int (*inject_gnss_ort)(QXWZGNSSOrt *ort);
参数	双天线参数。
返回值	成功：0，失败：-1
说明	注入 QXWZGNSSOrt 双天线参数。与 inject_gnss_pps 接口保持同频率调用。

set_position_mode

定义	void (*set_position_mode)(QXWZPosMode mode, float frequency);
参数	mode 表示启动定位服务类型, frequency 表示上报位置信息的频率。
返回值	无
说明	设置 SDK 启动定位服务类型，包括仅使用 RTK 服务、使用 RTK+DR 定位服务；配置上报定位结果的频率，例如 frequency 配置 1hz 表示每秒上报、0.2hz 表示每五秒上报定位结果。

delete_aiding_data

定义	void (*delete_aiding_data)(QXGNSSAidingData flags);
参数	辅助数据删除的标志集，QXGNSSAidingData 详见头文件。
返回值	无
说明	控制删除指定的辅助数据。

inject_sensor_config

定义	int (*inject_sensor_config)(QXWZSensorConfig *config);
参数	config：传感器类型配置结构体指针。
返回值	成功：0，失败：-1
说明	配置传感器类型信息。

inject_sensor_data

定义	int (*inject_sensor_data)(QXWZSensorData *sensorData);
参数	QXWZSensorData 传感器原始数据。
返回值	成功：0，失败：-1
说明	注入传感器数据，注入频率以配置的传感器速率为准，大于或等于此速率的传感器数据以配置的速率注入，小于此速率的传感器数据以实际采样速率注入。将传感器数据填入 QXWZSensorData 结构体中，m_Mask 字段设置为传入传感器类型按位或的结果，m_Time 字段记录系统时间。 示例如下： DR 所需传感器数据类型为加速度计、陀螺仪、四轮速三类，配置传感器数据频率为 200Hz。则设备实际加速度计注入频率为 200Hz，陀螺仪 200Hz，四轮速 10Hz。加速度计、陀螺仪、四轮速的注入频率比为 20:20:1，则每 20 笔数据中，仅有一笔包含四轮速，其余 19 笔均只包含加速度计和陀螺仪。20 笔数据，m_Mask 字段取值有两类，一为加速度计、陀螺仪位或的结果 (m_Mask = QX_SENSOR_TYPE_ACCEL | QX_SENSOR_TYPE_GYRO)，表明传入的 QXWZSensorData 结构体中包含加速度计、陀螺仪两类数据；二为加速度计、陀螺仪、四轮速三者位或的结果 (m_Mask = QX_SENSOR_TYPE_ACCEL | QX_SENSOR_TYPE_GYRO | QX_SENSOR_TYPE_OSPD)，表明传入的 QXWZSensorData 结构体中包含三类传感器数据。 **传感器频率和所需传感器数据需根据不同项目进行沟通和调节。**

inject_map_matching_para

定义	int (*inject_map_matching_para)(QXWZMapMatchingPara *mapParams);
参数	QXWZMapMatchingPara 表示使用第三方地图的信息。
返回值	成功：0，失败：-1
说明	指定使用第三方离线地图匹配的信息。

set_lever_arm_para

定义	int (*set_lever_arm_para)(QXWZLeverArmParams *lam);
参数	QXWZLeverArmParams 表示杆臂配置。
返回值	成功：0，失败：-1
说明	设置杆臂补偿参数。此接口需在魔方初始化后调用。

1.3 网络状态定义

```
typedef enum {
    /** unknown network, the initial value. */
    QXWZ_NET_TYPE_UNKNOWN,
    /** none network. */
    QXWZ_NET_TYPE_NONETWORK,
    /** wifi network. */
    QXWZ_NET_TYPE_WIFI,
    /** gsm network. */
    QXWZ_NET_TYPE_GSM
} QXWZNetworkStatus;
```

1.4 回调函数

```
typedef struct
{
    size_t      size;
    void        (*acquire_wakeLock)(int flag);
    void        (*fill_raw_data)(unsigned char *buf, int len);
    void        (*fill_position)(QXWZGnssLocation *pos);
    void        (*fill_nmea_info)(QXGnssUtcTime time, const char* nmea, int len);
    void        (*status_response)(QXWZSdkStatus status);
} QXWZSdkCallbacks;
```

acquire_wakeLock

定义	void (*acquire_wakeLock)(int flag);
参数	flag 表示 SDK 获取、释放系统唤醒锁，1 表示获取，0 表示释放。
返回值	无
说明	用于获取系统唤醒锁。 如果 SDK 使用者不管理唤醒锁，则 SDK 通过该接口获取系统唤醒锁。

fill_raw_data

定义	void (*fill_raw_data)(unsigned char *buf, int len);
参数	buf: 存储发送给千寻魔方的数据；len: 存储发送数据的长度。
返回值	无
说明	通过回调函数通知集成方将 buf 中存储的内容发送给千寻魔方。

fill_position

定义	void (*fill_position)(QXWZGnssLocation *pos);
参数	QXWZGnssLocation 结构体指针
返回值	无
说明	上报定位结果。

fill_nmea_info

定义	void (*fill_nmea_info)(QXGnssUtcTime time, const char* nmea, int len);
参数	time: NMEA 对应 UTC 时间。 nmea: NMEA 语句字符串。 len: NMEA 语句字符串长度。
返回值	无
说明	上报 NMEA 信息。

status_response

定义	void (*status_response)(QXWZSdkStatus status);
参数	status 表示上报的状态码，QXWZSdkStatus 详见头文件。
返回值	无
说明	上报 SDK 关键状态或异常状态。

1.5 上报 Location 结构体

```
typedef struct
{
    /**
     * set to sizeof(QXWZGnssLocation)
     */
```

```
size_t              size;
/**
 * position fusion flag
 */
QXGnssPosFlag     posflag;
/**
 * Represents latitude in degrees.
 */
double              latitude;
/**
 * Represents longitude in degrees.
 */
double              longitude;
/**
 * Represents altitude in meters above the WGS 84 reference ellipsoid.
 */
double              altitude;
/**
 * Represents speed in meters per second.
 */
float               speed;
/**
 * Represents heading in degrees.
 */
float               bearing;
/**
 * Represents expected accuracy in meters.
 */
float               accuracy;
/**
 * Timestamp for the location fix.
```

```
     */
    QXGnssUtcTime      timestamp;
    /**
     * Average CN0
     */
    float              avg_cno;
    /**
     * HDOP
     */
    float              hdop;
    /**
     * Number of satellites used for positioning.
     */
    unsigned char              sat_used;
    /**
     * Confidence coefficient.
     */
    QXWZGnssConfidenceParams confidence_param;
} QXWZGnssLocation;
```

1.6 配置信息

app_key 和 app_secret 需从 qxwz 官网购买服务获取。
device_id 需要是保证唯一的设备 ID。
device_type 由调用者配置，一般取硬件对应的类型。
root_dir 为千寻可以存储数据的目录，需要保证具有读写权限。
log_enable 千寻日志功能开关，1 表示开启，0 表示关闭。
socket_dir 指定创建 AF_UNIX 的 socket 目录，该目录需要有可读写和执行权限。
apply_scenario 指定算法使用场景，具体参数参考头文件和 demo code。
adapter_config 为保留字段，目前请忽略，初始化时置空即可。

server_config 专网环境配置 hostname 和 port。

```
typedef struct
{
    size_t    size;
    char      app_key[32];          //应用 appkey
    char      app_secret[128];      //应用密钥
    char      device_id[64];        //设备唯一编号
    char      device_type[64];      //设备类型
    char      root_dir[256];        //设置 sdk 工作根目录
    char      log_enable;           //日志功能开关
    char      socket_dir[256];      //指定创建 AF_UNIX Socket 的目录
    char      cfg_filename[256];    //指定算法配置文件目录
    QXGnssApplyScene   apply_scenario; //算法场景设置
    QXWZUartAdaptConf adapter_config;    // SDK uart adapter 配置
    QXWZServerConfig   server_config;     //配置 hostname 和 port
} QXWZSdkConfig;
```

1.7 传感器类型配置

```
#define QX_SENSOR_TYPE_LEN      64
typedef struct {
    /** Set to sizeof(QXWZSensorConfig). */
    size_t     size;
    /** * Gyro type */
    char       gyro_type[QX_SENSOR_TYPE_LEN];
    /** Accelerometer type */
    char       acce_type[QX_SENSOR_TYPE_LEN];
    /** Magnetometer type */
    char       magn_type[QX_SENSOR_TYPE_LEN];
    /** Barometer model */
    char       press_type[QX_SENSOR_TYPE_LEN];
    /** Odometer type */
    char       odom_type[QX_SENSOR_TYPE_LEN];
    /** Thermometer type */
     char       temp_type[QX_SENSOR_TYPE_LEN];
    /** Four wheel speed type */
    char       ospd_type[QX_SENSOR_TYPE_LEN];
} QXWZSensorConfig;
```

1.8 传感器数据结构体

m_Mask 字段设置为传入传感器类型位或的结果；m_Time 字段记录该传感器数据采集时的开机时间，单位为 1 毫秒；m_4Spd 字段根据下标 0-4，顺序记录方向盘转角，左前轮速，右前轮速，左后轮速，右后轮速。

```
typedef struct
{
```

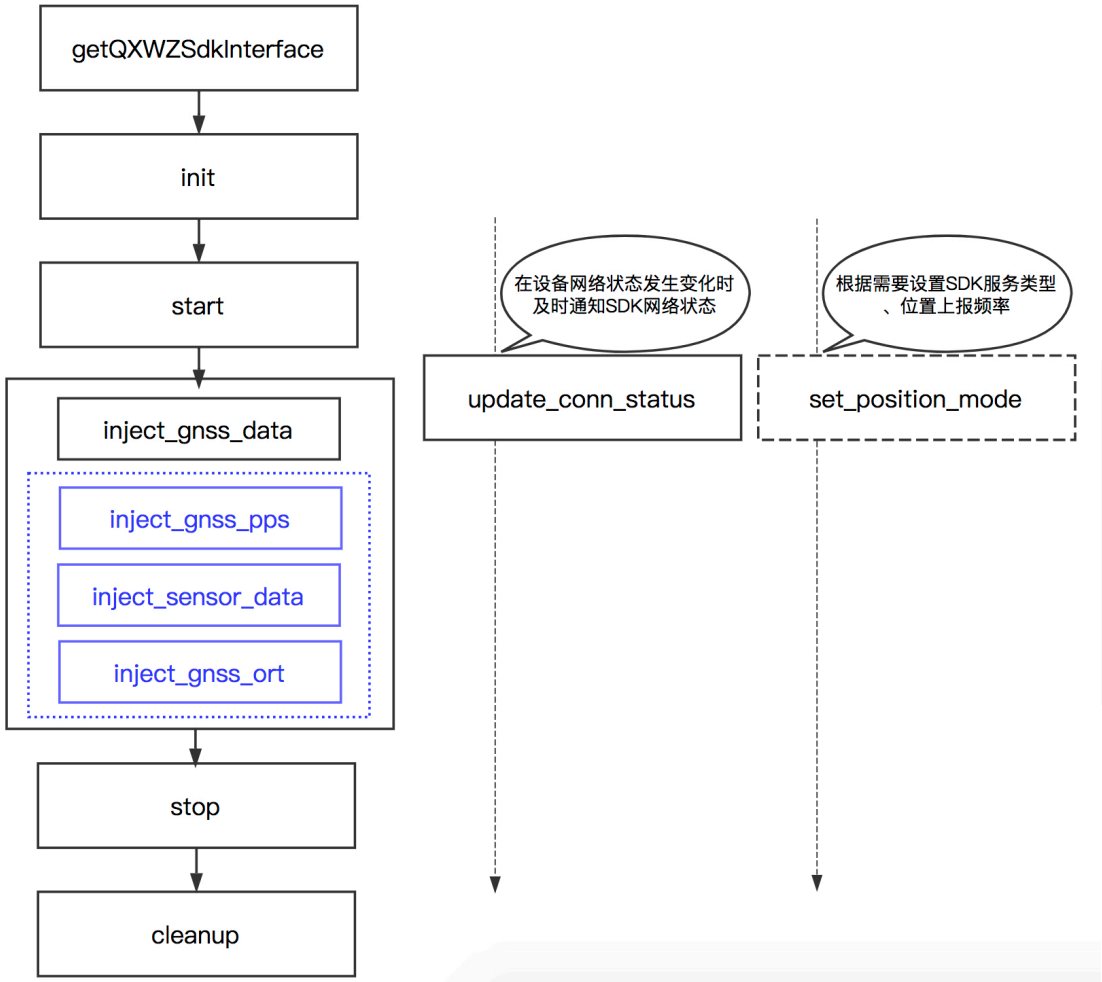
```
    unsigned int            m_Mask;         //传入传感器类型位或结果
    unsigned long long      m_Time;         //系统时间，单位 1 millisecond
    float                   m_ARate[3];     //陀螺仪，单位弧度每秒，rad/s
    float                   m_SForce[3];    //加速度，单位 m/s2
    float                   m_Mag[3];       //磁力计，单位 gauss
    float                   m_Baro;         //气压计，单位 Pa
    float                   m_Temp;         //温度计，单位 celsius
    float                   m_Speed;        //车速，单位 m/s
    float                   m_4Spd[5];      //方向盘转角，单位 degree 和四轮速，单位 m/s
} QXWZSensorData;
```

1.9 杆臂补偿参数结构体

杆臂补偿参数，以魔方安装在汽车上为例，以车正前方为 x 轴，车的正右为 y 轴，垂直车底盘向下为 z 轴，gnss2imu 取以 IMU 为这个坐标系原点，GNSS 天线在这个坐标系中的位置；imu2rearmiddle 取以车的两后轮中心连线的中点为原点，IMU 在坐标系中的位置；wheeltrack 取两后车轮的轮距（标量），即后两轮胎几何中心间的距离；imu2vehicle 为保留字段。

```
typedef struct
{
    size_t    size;
    float     gnss2imu[3];
    float     imu2rearmiddle[3];
    float     imu2vehicle[3];
    float     wheeltrack;
} QXWZLeverArmParams;
```

2 接口调用基本流程


getQXWZSdkInterface
init
start
inject_gnss_data
inject_gnss_pps
inject_sensor_data
inject_gnss_ort
stop
cleanup
在设备网络状态发生变化时
及时通知SDK网络状态
根据需要设置SDK服务类型
、位置上报频率
update_conn_status
set_position_mode

3 魔方 Native SDK 集成说明

魔方 NSDK 集成请参照 demo，大致步骤如下：

- 确保千寻魔方已集成到 Host 硬件平台，确定魔方在 Host 端的串口设备节点，并验证魔方串口工作正常。
- 千寻官网申请账号，替换 demo 中账号、串口设备节点信息，将 demo 编译成可执行程序。运行并检查当前目录产生的日志。控制台有 NMEA 等输出即表明 NSDK 正常工作。
- 基于 demo 进行集成开发，具体接口说明请阅读 API 说明。
- 嵌入式 Linux 平台检查一下”/proc/sys/net/unix/max_dgram_qlen”大小，如果该值过小会影响 Unix Socket 通信，确保该值大于 100，可通过命令 echo 512 > /proc/sys/net/unix/max_dgram_qlen 修改。