千寻差分数据
嵌入式 SDK

开发指南

V1.4.3

千寻位置网络有限公司

2017 年 6 月·上海

法律声明

千寻位置网络有限公司

联系邮箱：service@qxwz.com

官方网站：www.qxwz.com

目 录

第一章 概述

1.1 产品简介

千寻位置网差分数据嵌入式 SDK 为基于嵌入式平台的应用提供自动注册差分账号、自动获取差分数据的服务接口包，专注于为广大开发者提供便捷的高精度位置服务，这里简称为**差分数据 SDK**。

该 SDK 免费对外开放，适用于大批量高精度终端的差分账号智能分配和差分数据获取的繁琐操作。使用该 SDK 获取差分数据后，差分数据传输、芯片写入工作完全交由开发者完成，为开发者提供最大灵活性。用户在获取这些服务的同时，不用考虑数据存储，数据安全和繁琐的网络交互等细节。

千寻位置网差分数据 SDK 主要提供以下功能：

用户使用自己的高精度芯片差分算法， 使用 SDK 可以获取差分数据，用户可以使用这些差分数据灌进相关定位芯片或算法。

第二章 开发指南

2.1 开发准备

2.1.1 注册为开发者

开发者需要入驻千寻位置网，在千寻位置官网（qxwz.com）注册为千寻位置开发者，注册成功并通过个人/企业用户认证后，便可成为千寻位置网的开发者。详细信息请见千寻官网帮助文档。

2.1.2 购买千寻 FindM 或 FindCM 服务

2.1.3 进入管理控制台-控制中心-千寻跬步 / 千寻知寸-服务实例，选择服务实例对应管理进行开发者配置工作

1. 选择服务实例接入方式为 SDK；
2. 选择设备服务号绑定方式：自动绑定或手动绑定；

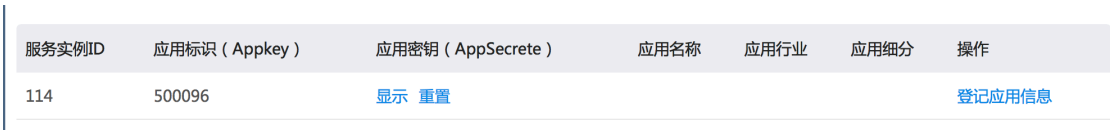
   若选择手动绑定，则需对需要连接千寻服务的设备进行绑定操作；
3. 选择设备服务号激活方式：自动激活或手动激活或终端激活方式；

   若选择手动激活，则需在千寻官网管理控制台中进行激活后使用；

   若选择终端激活方式，则需在终端调用千寻激活接口后进行激活后使用。

2.1.4 进入管理控制台-控制中心-千寻跬步 / 千寻知寸-设备服务号，创建设备服务号

2.1.5 前往控制中心-千寻跬步 / 千寻知寸-应用列表获取 AppKey, AppSecret



服务实例ID	应用标识（Appkey）	应用密钥（AppSecrete）	应用名称	应用行业	应用细分	操作
114	500096	显示 重置				登记应用信息

2.1.6 集成 SDK，写入 AppKey, AppSecret, DeviceID（设备 SN），DeviceType（设备类型）

2.1.7 绑定与激活操作（可选）

1. 若服务实例配置为需要绑定设备，则需要在控制中心-千寻跬步 / 千寻知寸-设备服务号进行绑定工作，写入 SDK 的 DeviceID 和 DeviceType

2. 若服务实例配置为需要激活，前往控制中心-千寻跬步 / 千寻知寸-设备服务号对当前设备进行激活操作。

2.2 接入流程图

SDK 使用流程大体如下：

1，工程引入 SDK lib 包，调用对应的 API 接口完成操作。

2，由于 API 有先后依赖关系， 列出相关 API 的调用流程，见 2.4.1。

2.3 配置工程

2.3.1 lib 和头文件引入

1，准备千寻位置网服务邮箱提供的 lib 包。例如“rtcm.lib”

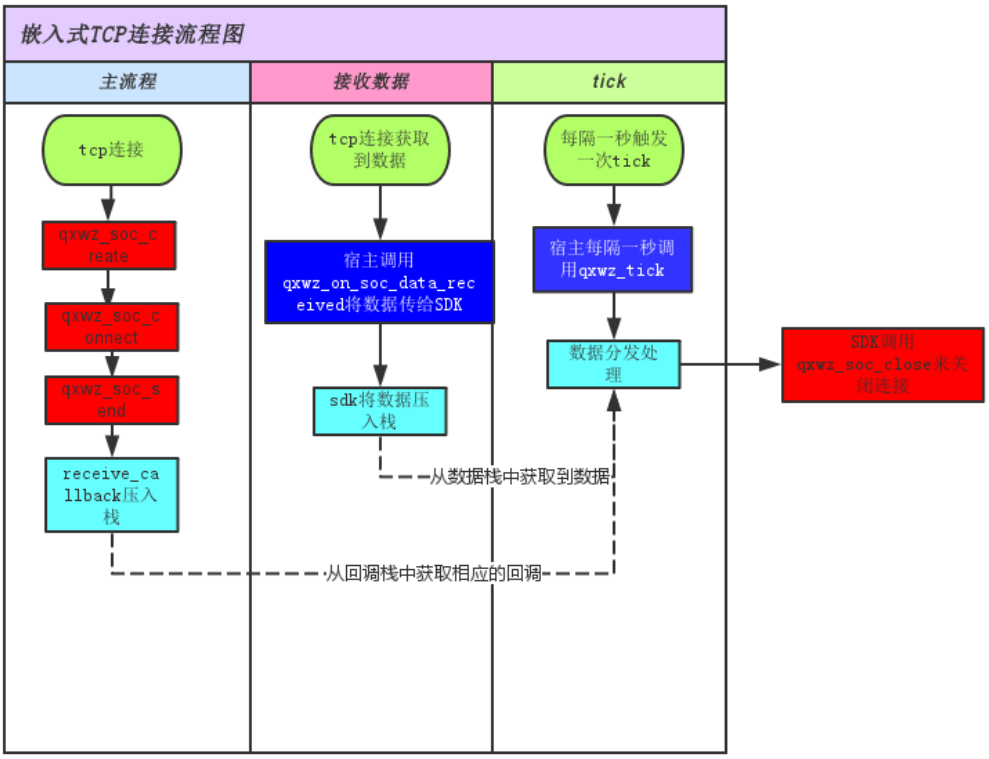
2，工程引入头文件，“qxwz_rtcm.h” “socket.h” “prefes.h”

2.4 网络模块交互设计

2.4.1 设计流程

差分数据 SDK 的网络请求和数据回传，外部需重写(参考 socket.h 头文件)，由于 SDK 将网络模块的功能剥离出来，因此需要开发者自己实现网络模块功能。（备注：SDK 在设计的时候已经是多通道设计，但是开发者在实现网络的时候可以按照单通道来实现）

SDK 对于网络模块的详细调用流程（备注：这些步骤需要 SDK 以及宿主互相配合才能组成一个完整的调用流程）参考下图：



备注：上图红色背景黑字的部分是需要宿主自己来实现提供给 SDK 调用的，而蓝色背景白色字体的部分是需要宿主调用的。

开发者如果网络是异步实现的，需要在程序中设置 flag 使能异步 socket 功能。

qxwz_prefs_flags_set(QXWZ_PREFS_FLAG_SOCKET_ASYN);

如果开发者板子具备打印 LOG 功能，可以通过调用

qxwz_prefs_flags_set(QXWZ_PREFS_FLAG_ENABLE_LOG);

使能打印 SDK LOG 功能，方便问题跟踪。

2.4.2 网络模块实现流程

1，由开发者实现 qxwz_soc_create 方法，该方法主要是用来创建一个 TCP，如果宿主是单通道设计的，那么可以返回一个固定的大于等于 1 的数字。

2，由开发者实现 qxwz_soc_connect 方法，该方法主要是由 SDk 调用来连接服务端。如果宿主使用的是异步 socket(非循环等待 AT 指令应答的实现)当宿主检测到连接完成，调用 qxwz_soc_connect_complete 方法 将网络的状态信息传给 SDK。如果是同步 socket，不能调用 qxwz_soc_connect_complete 方法。

3，由开发者实现 qxwz_soc_send 方法 ,该方法主要是由 SDK 调用来将数据发送给服务端。

5，如果使能异步 socket 方式，当宿主检测到数据发送完成，调用 qxwz_soc_send_complete 方法，将数据发送的状态信息传给 SDK。同步 socket 实现不能调用该方法。

6，当宿主获取到数据的时候，将获取到的数据通过调用 qxwz_on_soc_data_received 方法传给 SDK。

7，由开发者实现 qxwz_soc_close 方法，该方法主要是由 SDK 调用来关闭 TCP 连接。

8，当宿主检测到 SOCKET 关闭完成，调用 qxwz_soc_close_complete 方法，将 close 状态传给 SDK。同步 socket 实现不能调用该方法。

9，任何出现的 socket 异常，比如：接收数据失败、TCP 连接意外断开等都必须调用 qxwz_soc_error 方法,以便通知 SDK TCP 连接出现异常。

第三章 SDK 接口说明

3.1 配置 SDK

3.1.1 接口定义

```
/**
  * 设置rtcm配置
  * @param config   config包括appKey、secret、device_ID、device_Type，appKey和appsecret到官网申请到的,device_ID是可以填写mac地址、商品编号，device_Type是设备类型，可以填入类似sony、hitargt等，这四个参数必须填入。
*/

void qxwz_setting(qxwz_config* config);
```

3.1.2 示例代码

```
qxwz_config config;
config.appkey = "/*申请的 appKey*/";
config.appsecret= "/*申请的 secret*/";
config.device_ID = "00:11:67:44:41:6A"; //设备的唯一 ID，建议使用设备的序列号
config.device_Type = "Test_device";
qxwz_setting(&config);
```

3.2 启动 RTCM

3.2.1 接口定义

```
/**
 *  启动RTCM
 * qxwz_rtcm_response   RTCM数据获取函数回调指针
 * qxwz_status_response SDK状态获取函数回调指针
 */

      void qxwz_start(qxwz_rtcm_response rtcm_rsp, qxwz_status_response status_rsp);
```

3.2.2 示例代码

```
      qxwz_start(qxwz_rtcm_response_callback,qxwz_status_response_callback);
```

3.3 获取 RTCM 数据

3.3.1 接口定义

```
/**
* 当收到RTCM数据时，向外部程序通知收到的RTCM数据
*
*   @param rtcm         rtcm数据
*   @param length       rtcm数据长度
*
*/
typedef void(*qxwz_rtcm_response)(char* rtcm, size_t length);
```

3.4 获取 SDK 返回状态

3.4.1 接口定义

```
/**
 *   RTCM服务状态码回调函数
 *   @param qxwz_status_response
 */
typedef void (*qxwz_status_response)(qxwz_rtcm_status code);
```

备注：返回状态码见 3.7 状态码表

3.5 获取用户账号状态信息

3.5.1 接口定义

```
/**
*
* 查询用户账号信息
*
* @return    qxwz_account_info*
*
*/
qxwz_account_info*    getqxwzAccount(void);
```

通过 getqxwzAccount 接口获取用户状态信息，当距离账号过期还有 10 天时，我们会通过状态信息回调的方式，通知用户，状态码为 2011。当用户账号已过期时，相应的状态码为 2010。

```
/**
* SDK 账号信息
*/
typedef struct {
        char *appkey;
        char *deviceID;
        char *deviceType;
        char *serviceType;
        time_t expire_time;    /*自1970年1月1日的秒数*/
        char *NtripUserName;
        char *NtripPassword;
} qxwz_account_info;
```

示例代码：

```
qxwz_account_info *p_account_info = NULL;
void    get_qxwz_sdk_account_info(void)
{
  p_account_info = getqxwzAccount();
  if(p_account_info->appkey != NULL) {
        printf("appkey=%s\n",p_account_info->appkey);
  }
  if(p_account_info->NtripPassword != NULL) {
```

```
        printf("NtripPassword=%s\n",p_account_info->NtripPassword);
    }
    if(p_account_info->NtripUserName != NULL) {
        printf("NtripUserName=%s\n",p_account_info->NtripUserName);
    }
    if(p_account_info->NtripPassword != NULL) {
        printf("NtripPassword=%s\n",p_account_info->NtripPassword);
    }
    printf("expire_time=%d\n",p_account_info->expire_time);
}
```

3.6 发送 GGA

3.6.1 接口定义

```
/**
 *  向ntrip服务器发送GGA字符串用来获取rtcm数据
 *  @param ggastring NMEA协议里面GGA语句
 */

void qxwz_send_gga(char* ggastring);
```

备注：每隔一段时间（建议根据设备使用场景中 1 秒移动的距离，时间可以是 2 秒，建议不要大于 1 分钟）调用 qxwz_send_gga 方法，将原始的 GGA 字符串传给 SDK。

3.6.2 示例代码

```
char* gga = "$GPGGA,000001,3112.518576,N,12127.901251,E,1,8,1,0,M,-32,M,3,0*4B\r\n");
qxwz_send_gga(gga);
```

3.7 固定频率使 SDK 工作

3.7.1 接口定义

```
/**
* 可在此方法内执行具体的任务，由外部程序定时调用，由于网路超时等情况，外部程序可能不能及时调用
* *  @param system_time 系统时间
```

```
*/
    void qxwz_tick(int32_t system_time);
```

备注：每隔一段时间（建议不要大于 1 秒）调用 qxwz_tick 方法，并且传入一个表示应用启动到调用 qxwz_tick 为止的一个毫秒级的时间，（例如：第一次调用传入 0，第二次调用传入 1000，第三次传入 2000，以此类推）。

3.8 返回状态码

状态码	状态说明
1000	ntrip 连接到服务器
1001	ntrip 断开服务器
1002	APP KEY 认证失败
1003	APP KEY 认证成功
1004	网络异常
1005	NTRIP 用户已经达到上限
1006	NTRIP 用户不存在
1011	非法 GGA
1013	正在连接 ntrip 服务器
1014	ntrip 正在接受数据
1015	非法 APP KEY
1016	非法 APP SECRET
1017	deviceType 错误
1018	deviceid 错误
1020	SDK 内部错误

1021	Ntrip 播发数据正常
1022	Ntrip 认证失败
1035	setting 错误
1036	没有调用 setting 函数
2001	缺少参数
2002	账号不存在
2003	账号重复
2004	错误密码
2005	账号未激活
2006	没有有效的账号
2007	POPUser 不存在
2008	服务端内部错误
2010	账号已过期，需续费
2011	账号即将过期
2012	目前的账号没有绑定

第四章 常见问题(FAQ)

4.1 用户使用过程中的问题

Q1 :文档只给出了相应的函数调用 ,终端实现高精度定位的一个具体的流程，请给出说明？

A1：终端实现高精度定位的一个具体的流程:大致思路是，

STEP1：去官网申请 appkey 和 appsecret，完成用户认证，申请配额，订阅相关服务，绑定设备 sn。具体可参考官网帮助文档。

STEP2：调用 void qxwz_setting(qxwz_config* config);这个方法，将用户配置信息传入 SDK，参考接入文档 3.1.2

STEP3:调用 void qxwz_start(qxwz_rtcm_response rtcm_rsp, qxwz_status_response status_rsp);方法，该方法的两个参数是回调函数的函数指针，这两个回调函数需由用户实现。

STEP4 :每秒调用 qxwz_tick()方法 ,并传入系统时间 ,具体请参考接入文档 3.6。

此外，每秒将接收到的 GGA 数据发送给 SDK，由 SDK 发送给千寻服务器，然后服务器下发 RTCM 差分数据包 ,用户将接收到的数据包回写到 GPS 模块,然后硬件会做纠偏，后面再吐出的 GGA 等卫星数据就是纠偏过的，更加精准的。

Q2: 是不是我每一秒调用一次 void qxwz_tick(int32_tsystem_time)这个函数，SDK 就可以正常的工作了？

A2: 每一秒调用一次 void qxwz_tick(int32_tsystem_time)这个函数，这个是必要条件，此外 SDK 正常工作还需要网络模块和 GPS 模块正常工作，以及每秒发送正确的 GGA 数据给 SDK

Q3: 为什么接入后程序跑飞？

A3:SDK 需要 4K 字节以上的 STACK，需要用户设置，要不然可能会造成内存踩踏，导致跑飞。

Q4：当收到 RTCM 数据时，向外部程序通知收到的 RTCM 数据差分数据回调函数

void qxwz_rtcm_response_callback(char* rtcm, size_t length)

错误事件通知 RTCM 服务状态码回调函数

void qxwz_status_response_callback(qxwz_rtcm_status_code code)

发送 GGA 字符串，用来获取 rtcm 数据

void qxwz_send_gga(char* ggastring)

1. 这三个函数的用途
2. 是否需要由我们来编写

A4:前两个函数需要用户实现函数体，第三个用户只需要每秒调用即可。

Q5：获取差分数据和发送 GGA 的时间间隔可以调整不?

A5: 获取差分数据的时间是根据您的需求变化的，如果是 FindM 服务，那么可能是 15 秒播发或者 1 秒播发，如果是 FindCM 服务，一般就是每秒播发，发送 GGA 的时间可以调整，但最好不低于 5 秒一次。

Q6:如何实现 SDK 的 LOG 信息输出?

A6:目前 SDK 中的 LOG 输出调用的是 printf 方法 ,开启 SDK 的 LOG 输出，需要先设置标志位。可以调用该方法 qxwz_prefs_flags_set(QXWZ_PREFS_FLAG_ENABLE_LOG);

如果需要同时设置两个标志位 ,比如使能 LOG 输出以及使能异步 SOCKET 功能，需要采用如下方式实现:

qxwz_prefs_flags_set(QXWZ_PREFS_FLAG_ENABLE_LOG|QXWZ_PREFS_FLAG_SOCKET_ASYN);

另外，大部分嵌入式集成调试环境 IDE，不支持直接在 IDE 界面上输出 printf 的结果，需要重定向 LOG 输出，将其输出到串口。具体输出到哪个串口，是由用户实现的。以下是我们 demo 中的一个串口实现 LOG 输出的例子 ,是在 KEIL 中将 LOG 输出到串口 1 的一个例子:

```
#pragma import(__use_no_semihosting)
//标准库需要的支持函数
struct __FILE
{
    int handle;
};
```

```
FILE __stdout;
//定义 sys_exit()以避免使用半主机模式
_sys_exit(int x)
{
    x = x;
}
//重定义 fputc 函数
int fputc(int ch, FILE *f)
{
    while((USART1->SR&0X40)==0);//循环发送知道发送结束。
     USART1->DR = (u8) ch;
    return ch;
}
```

A7：通过集成千寻的 SDK，连接到服务器的主要流程以及会收到哪些数据？

Q7:目前连接到千寻服务器主要有以下流程:

STEP1：第一个 HTTP 请求（短连接），获取千寻服务器时间戳记。

STEP2：第二个 HTTP 请求（短连接），主要是对用户账号信息等进行鉴权、获取服务器配置、以及获取 ntrip 差分账号等。

STEP3：Ntrip 连接（长连接），主要是使用上一步骤获取的差分账号信息连接 Ntrip 服务器

STEP4：定时发送 GGA 到 Ntrip 服务器，Ntrip 会依据用户的位置信息，播发 RTCM 差分数据。

此外，可以参考我们使用 SIM900A 上网模块连接到千寻服务器的 LOG 信息，请参见以下 LOG 日志，  log_20161122_rtd_rtcm32_S_AT.txt 。

4.2 用户账号申请的问题

1.注意事项，device_Type 与 device_ID 填写时不要出现空格。

2.官网上的设备 sn 对应的是 deviceID，在用户程序中填写的必须与官网的保持一致。