A76xx Series Open SDK_ GNSS_应用指导

LTE 模组

芯讯通无线科技(上海)有限公司
上海市长宁区临虹路289号3号楼芯讯通总部大楼
电话：86-21-31575100
技术支持邮箱：support@simcom.com
官网：www.simcom.com

名称：	A76xx Series Open SDK_GNSS_应用指导
版本：	V1.00
类别：	应用文档
状态：	已发布

版权声明

芯讯通无线科技(上海)有限公司
上海市长宁区临虹路289号3号楼芯讯通总部大楼
电话：86-21-31575100
邮箱：simcom@simcom.com
官网：www.simcom.com

了解更多资料，请点击以下链接：
http://cn.simcom.com/download/list-230-cn.html

技术支持，请点击以下链接：
http://cn.simcom.com/ask/index-cn.html 或发送邮件至 support@simcom.com

Version History

Version	Date	Owner	What is new
V1.00	2021-11-17		第一版

About this Document

本文档适用于 A1803S open 系列、A1603 open 系列、A1606 open 系列。

目录

1 GNSS 介绍

GNSS（Global Navigation Satellite System，是对北斗系统（BDS）、GPS、GLONASS、Galileo 系统等卫星导航定位系统的统称，也可指代他们的增强型系统。很长时间以来，GNSS 有两个译名：全球卫星导航系统和全球导航卫星系统，直到 1992 年 5 月，国际民航组织（(International Civil Aviation Organization，ICAO）在航空导航会议上才确定为全球导航卫星系统。

从本质上讲，GNSS 是一个全球性位置与时间的测定系统，包括诸多卫星星座、接收机与监测系统。也就是说它是由多个卫星导航定位及其增强型系统所拼凑组成的大系统。

从功能上讲，GNSS 是以人造卫星作为“航向标”的无线电导航系统，为全球陆、海、空、天的各类载体提供全天候、高精度的位置、速度和时间信息（Positioning, Navigation and Timing, PNT），因为它又称为天基定位、导航和授时系统。

很多人提及 GPS，就想到卫星导航，实质上这是对 GPS 的一个误解，同样也是对 GNSS 的混淆。GPS 是由美国国防部研制建立的一种具有全方位、全天候、全时段、高精度的卫星导航系统，能为全球用户提供低成本、高精度、有差别的 PNT 服务。GPS 可以说是卫星通信技术在导航领域的一个极其经典的应用案例，不仅提高了全球的信息化水平，同时也促进了数字地球的发展。

但是 GPS 并不是 GNSS，或者说 GPS 并不能完全代表 GNSS，理论上 GPS 是 GNSS 系统的一个重要组成部分。大家经常使用微信的“定位”功能，也经常使用各种地图进行导航（如百度和高德），实质上都是 GNSS 导航功能的典型应用，定位精度一般为米级。这里导航功能的实现并非单纯的 GPS，实质上是多种卫星系统组合定位的结果，但这并非 GNSS 的全部功能，其“定位”功能一般需要专业仪器才能实现，比如测绘中常用的 RTK 放样与静态测量等。正如有人所指出的那样，我们人类生活在一个四维的世界（x、y、z、t），其中 t 就是时间。GNSS 系统则可以提供我们经常使用的时间功能，目前手机上这个功能通常采用网络时间同步来实现，一旦网络中断，我们就无法得到高精度的时间（机械手表除外），而 GNSS 则可以实时提供我们所需要的三维位置与时间信息。不知不觉，GNSS 已经深入到我们生活的点点滴滴，也在潜移默化中改变我们的生活方式，不可否认，GNSS 就是信息地球的重要基础！

目前，我们获取到的 NMEA 原始数据属于 WGS-84 坐标系。

2 GNSS API 接口函数

模块支持如下几个 API 接口

2.1 sAPI_GnssPowerStatusSet

接口	SC_Gnss_Return_Code sAPI_GnssPowerStatusSet(SC_Gnss_Power_Status power);
输入	power：设置 GNSS 电源开/关
输出	无
返回值	0:成功　-1:失败
NOTE	这个接口用着设置 GNSS 电源状态，GNSS 默认电平状态是关闭状态。ASR1603 和 ASR1803 平台的 GNSS 上电后需要等待 9 秒时间让 GNSS 动态加载（给 GPS 芯片升级软件版本），加载完成后 GNSS 波特率变成 115200。

2.2 sAPI_GnssPowerStatusGet

接口	SC_Gnss_Power_Status sAPI_GnssPowerStatusGet(void);
输入	无
输出	无
返回值	0: GNSS 关闭状态；1: GNSS 开启状态
NOTE	该接口用于获取 GNSS 的开启状态。

2.3 sAPI_GnssNmeaDataGet

接口	SC_Gnss_Return_Code sAPI_GnssNmeaDataGet(SC_Gnss_Output_Control ctl, SC_Gnss_Nmea_Data_Get mode);
输入	ctl: 控制是否输出数据； mode: 通过 nmea 端口或者 urc 方式获取 nmea 数据
输出	无
返回值	0:成功　-1:失败
NOTE	GNSS 上电后才能使用该接口。该接口用于设置使用哪种方式是否输出 nmea 数据。

2.4 sAPI_GnssStartMode

接口	SC_Gnss_Return_Code sAPI_GnssStartMode(SC_Gnss_Start_Mode mode);
输入	mode: 选择 GNSS 的启动模式
输出	无
返回值	0:成功 -1:失败
NOTE	GNSS 上电后才能使用该接口。该接口用于设置 GNSS 的启动模式。ASR1601 平台只支持冷启动和热启动。ASR1603、ASR1606 和 ASR1803 平台支持冷启动、温启动和热启动。

2.5 sAPI_GnssBaudRateSet

接口	SC_Gnss_Return_Code sAPI_GnssBaudRateSet(SC_Gnss_Baud_Rate baudrate);
输入	Baudrate： GNSS 的波特率
输出	无
返回值	0:成功 -1:失败
NOTE	GNSS 上电后才能使用该接口。该接口用于设置 GNSS 的波特率。ASR1601 平台支持：4800, 9600, 19200, 38400, 57600, 115200。ASR1603、ASR1803 和 ASR1606（除 A7680C 系列以外）平台支持：9600, 115200, 230400。ASR1606 的 A7680C 系列只支持 115200。

2.6 sAPI_GnssBaudRateGet

接口	SC_Gnss_Baud_Rate sAPI_GnssBaudRateGet(void);
输入	无
输出	无
返回值	GNSS 波特率
NOTE	GNSS 上电后才能使用该接口。该接口用于设置 GNSS 的波特率。ASR1601 平台支持：4800, 9600, 19200, 38400, 57600, 115200。ASR1603、ASR1803 和 ASR1606（除 A7680C 系列以外）平台支持：9600, 115200, 230400。ASR1606 的 A7680C 系列只支持 115200。

2.7 sAPI_GnssModeSet

接口	SC_Gnss_Return_Code sAPI_GnssModeSet(SC_Gnss_Mode mode);
输入	mode: GNSS 的定位模式

输出	无
返回值	0:成功　-1:失败
NOTE	GNSS 上电后才能使用该接口。该接口用于设置 GNSS 的定位模式。ASR1601 平台和 ASR1606 平台的 A7680C 系列支持：GPS, BDS, GPS+BDS, GLONASS, GPS+GLONASS, BDS+GLONASS, GPS+BDS+GLONASS。ASR1603、ASR1803 和 ASR1606（除 A7680C 系列以外）平台国内版本支持：GPS+BDS+QZSS, BDS, GPS+QZSS。ASR1603 和 ASR1803 平台国外版本支持：GPS+SBAS+QZSS, BDS, GPS+GLONASS+GALILEO+SBAS+QZSS，GPS+BDS+GALILEO+SBAS+QZSS。

2.8 sAPI_GnssModeGet

接口	SC_Gnss_Mode　sAPI_GnssModeGet(void);
输入	无
输出	无
返回值	GNSS 的定位模式
NOTE	GNSS 上电后才能使用该接口。该接口用于获取 GNSS 的定位模式。ASR1601 平台和 ASR1606 平台的 A7680C 系列支持：GPS, BDS, GPS+BDS, GLONASS, GPS+GLONASS, BDS+GLONASS, GPS+BDS+GLONASS。ASR1603、ASR1803 和 ASR1606（除 A7680C 系列以外）平台国内版本支持：GPS+BDS+QZSS, BDS, GPS+QZSS。ASR1603 和 ASR1803 平台国外版本支持：GPS+SBAS+QZSS, BDS, GPS+GLONASS+GALILEO+SBAS+QZSS，GPS+BDS+GALILEO+SBAS+QZSS。

2.9 sAPI_GnssNmeaRateSet

接口	SC_Gnss_Return_Code　sAPI_GnssNmeaRateSet(SC_Gnss_Nmea_Rate rate);
输入	无
输出	无
返回值	0:成功　-1:失败
NOTE	GNSS 上电后才能使用该接口。该接口用于设置 GNSS 的更新频率。ASR1601 平台支持：1Hz, 2Hz, 4Hz, 5Hz, 10Hz。ASR1603、ASR1803 和 ASR1606（除 A7680C 系列以外）平台支持：1Hz, 2Hz, 5Hz。ASR1606 平台的 A7680C 系列只支持 1Hz。

2.10 sAPI_GnssNmeaRateGet

接口	SC_Gnss_Nmea_Rate sAPI_GnssNmeaRateGet(void);
输入	无
输出	无
返回值	GNSS 的更新频率
NOTE	GNSS 上电后才能使用该接口。该接口用于获取 GNSS 的更新频率。ASR1601 平台支持：1Hz, 2Hz, 4Hz, 5Hz, 10Hz。ASR1603、ASR1803 和 ASR1606（除 A7680C 系列以外）平台支持：1Hz, 2Hz, 5Hz。ASR1606 平台的 A7680C 系列只支持 1Hz。

2.11 sAPI_GnssNmeaSentenceSet

接口	SC_Gnss_Return_Code sAPI_GnssNmeaSentenceSet(unsigned short mask);
输入	mask: nmea 语句类型的掩码，1 表示启用相应字段，0 表示禁用相应字段，ASR1601 平台范围为 0-13311，ASR1603 平台范围为 0-255，ASR1606 平台（除 A7680C 系列以外）范围为 0-511。
输出	无
返回值	0:成功 -1:失败
NOTE	GNSS 上电后才能使用该接口。该接口用于设置 GNSS 的 nmea 语句类型。 ASR1601 平台默认值是 255(00000011111111)，支持的 nmea 字段为： bit0-GGA, default is 1 bit1-GLL, default is 1 bit2-GSA, default is 1 bit3-GSV, default is 1 bit4-RMC, default is 1 bit5-VTG, default is 1 bit6-ZDA, default is 1 bit7-ANT, default is 1 bit8-DHV, default is 0 bit9-LPS, default is 0(nonsupport) bit10-res1, default is 0 bit11-res2, default is 0 bit12-UTC, default is 0(nonsupport) bit13-GST, default is 0 ASR1603、ASR1803 和 ASR1606（除 A7680C 系列以外）默认值是 63(00111111)，支持的 nmea 字段为： bit0-GGA, default is 1 bit1-GLL, default is 1 bit2-GSA, default is 1 bit3-GSV, default is 1

	bit4-RMC, default is 1 bit5-VTG, default is 1 bit6-ZDA, default is 0 bit7-GST, default is 0 ASR1606d A7680C 系列默认值是 511(111111111)，支持的 nmea 字段为： bit1-VTG,default is 1 bit2-GGA,default is 1 bit3-GSA,default is 1 bit4-GSV,default is 1 bit5-GLL,default is 1 bit6-ZDA,default is 1 bit7-GST,default is 1 bit8-TXT,default is 1

2.12 sAPI_GnssNmeaSentenceGet

接口	UINT8* sAPI_GnssNmeaSentenceGet(void);
输入	无
输出	无
返回值	Nmea 语句类型开关的情况
NOTE	GNSS 上电后才能使用该接口。该接口用于获取 GNSS 的 nmea 语句类型。 ASR1601 平台默认值是 255(00000011111111)，支持的 nmea 字段为： bit0-GGA, default is 1 bit1-GLL, default is 1 bit2-GSA, default is 1 bit3-GSV, default is 1 bit4-RMC, default is 1 bit5-VTG, default is 1 bit6-ZDA, default is 1 bit7-ANT, default is 1 bit8-DHV, default is 0 bit9-LPS, default is 0(nonsupport) bit10-res1, default is 0 bit11-res2, default is 0 bit12-UTC, default is 0(nonsupport) bit13-GST, default is 0 ASR1603、ASR1803 和 ASR1606（除 A7680C 系列以外）默认值是 63(00111111)，支持的 nmea 字段为： bit0-GGA, default is 1 bit1-GLL, default is 1 bit2-GSA, default is 1 bit3-GSV, default is 1 bit4-RMC, default is 1

	bit5-VTG, default is 1 bit6-ZDA, default is 0 bit7-GST, default is 0 ASR1606d A7680C 系列默认值是 511(111111111)，支持的 nmea 字段为： bit1-VTG,default is 1 bit2-GGA,default is 1 bit3-GSA,default is 1 bit4-GSV,default is 1 bit5-GLL,default is 1 bit6-ZDA,default is 1 bit7-GST,default is 1 bit8-TXT,default is 1

2.13 sAPI_GpsInfoGet

接口	SC_Gnss_Return_Code sAPI_GpsInfoGet(UINT8 period);
输入	period: 范围为 0-255，单位为秒。设置 period 后，GPS 解析信息将会每 period 秒输出，0 表示停止上报 GPS 解析信息。
输出	无
返回值	0:成功 -1:失败
NOTE	GNSS 上电后才能使用该接口。该接口用于获取并上报 GPS 的解析信息。GPS 信息将会在 cus_urc.c 里面通过 urc 方式上报。

2.14 sAPI_GnssInfoGet

接口	SC_Gnss_Return_Code sAPI_GnssInfoGet(UINT8 period);
输入	period: 范围为 0-255，单位为秒。设置 period 后，GNSS 解析信息将会每 period 秒输出，0 表示停止上报 GNSS 解析信息。
输出	无
返回值	0:成功 -1:失败
NOTE	GNSS 上电后才能使用该接口。该接口用于获取并上报 GNSS 的解析信息。GNSS 信息将会在 cus_urc.c 里面通过 urc 方式上报。

2.15 sAPI_SendCmd2Gnss

接口	SC_Gnss_Return_Code sAPI_SendCmd2Gnss(char *string);
输入	string: GNSS 的 nmea 格式的命令,如$CFGMSG,0,1,0
输出	无
返回值	0:成功 -1:失败

NOTE	GNSS 上电后才能使用该接口。该接口用于发送 GNSS 的命令到内部 gps 芯片。 1606 平台（A7680C 系列除外）和 1803 平台常见问题： 1. 漂移问题，请发送$CFGDYN,h02,1,100，其中参数 100 是设置的静态保持模式下速度门限，单位为厘米/秒，该值为 0 会关闭静态保持模式。该指令在 GPS 芯片型号为 UC6226NIS 是可以掉电保存的，但在型号为 UC6228CI 是无法掉电保存。 2. 版本号查看：$PDTINFO，发送后可以在 nmea 数据中查看到 GPS 的软件版本

2.16 sAPI_GnssAgpsSeviceOpen

接口	SC_Gnss_Return_Code sAPI_ GnssAgpsSeviceOpen (void);
输入	无
输出	无
返回值	0:AGNSS 成功。 -1：GNSS 电源关闭。 101:打开套接字失败。 102:获取 AGNSS 服务器失败。 103:连接到 AGNSS 服务器失败。 104:向套接字写入信息失败。 105:从套接字读取 AGNSS 数据失败。 106:获取 AGNSS 数据错误。 107:发送 AGNSS 数据失败。
NOTE	ASR1601 平台下，使用 AGPS 时只需要在 GNSS 上电后并有网的情况下调用该接口。 ASR1603、ASR1606 和 ASR1803 平台使用该接口之前需要使用 ntp 更新系统时间，然后在 GNSS 上电后并有网的情况下调用该接口。 该接口用于从 AGNSS 服务器获取 AGNSS 数据，以进行辅助快速定位。访问 AGNSS 服务器的地址、用户名、密码等已经封装在该 API，使用 AGNSS 功能只需要调用该接口就可以了。

2.17 sAPI_GnssApFlashSet

接口	SC_Gnss_Return_Code sAPI_GnssApFlashSet(SC_Gnss_Ap_Flash_Status ctl);
输入	ctl: 设置 GNSS AP_Flash 的软件热启动标志的开/关
输出	无

返回值	0:成功　-1:失败
NOTE	ASR1601 和 ASR1606 平台不支持该功能。 此应用程序接口用于设置 GNSS AP_Flash 热启动标志状态，默认值为关闭，通过设置 SC_GNS_AP_Flash_ON 启用 GNSS AP_Flash 热起动。要使用 AP_Flash 热启动，首先将 AP_Flash 热起动标志设置为 SC_GNSS_ap_flash_ON，然后打开 GNSS。

2.18 sAPI_GnssApFlashGet

接口	SC_Gnss_Ap_Flash_Status sAPI_GnssApFlashGet(void);
输入	无
输出	无
返回值	0: GNSS AP_Flash 热启动关闭　1: GNSS AP_Flash 热启动开启
NOTE	ASR1601 和 ASR1606 平台不支持该功能。 该接口用于获取 GNSS AP_Flash 热启动标志状态。

3 GNSS API 使用步骤

请参考 demo_gps.c

1. sAPI_GnssPowerStatusSet(SC_GNSS_POWER_ON);
2. sAPI_GnssNmeaDataGet(SC_GNSS_START_OUTPUT_NMEA_DATA, SC_GNSS_NMEA_DATA_GET_BY_PORT);
3. sAPI_GnssStartMode(SC_GNSS_START_HOT);
4. sAPI_ GnssAgpsSeviceOpen ().

4 GNSS 实例代码

请参考 demo_ntp.c 和 demo_gps.c

```
void ntpUpdataTime()
{
    sMsgQRef ntpUIResp_msgq;
    SCsysTime_t currUtcTime;
    if(NULL == ntpUIResp_msgq)
   {
        SC_STATUS status;
        status = sAPI_MsgQCreate(&ntpUIResp_msgq, "htpUIResp_msgq", sizeof(SIM_MSG_T), 4,
SC_FIFO);
        if(SC_SUCCESS != status)
        {
            sAPI_Debug("[CNTP]msgQ create fail");
        }
    }
    memset(&currUtcTime,0,sizeof(currUtcTime));
    sAPI_GetSysLocalTime(&currUtcTime);
    sAPI_Debug("[CNTP] sAPI_GetSysLocalTime %d - %d - %d
- %d : %d : %d    %d",currUtcTime.tm_year,currUtcTime.tm_mon,currUtcTime.tm_mday,currUtcTime.tm_h
our,currUtcTime.tm_min,currUtcTime.tm_sec,currUtcTime.tm_wday);

    ret = sAPI_NtpUpdate(SC_NTP_OP_SET, "ntp3.aliyun.com", 32, NULL); //Unavailable addr may
cause long time suspend
    sAPI_Debug("[CNTP] func[%s] line[%d] ret[%d]", __FUNCTION__,__LINE__,ret);
    ret = sAPI_NtpUpdate(SC_NTP_OP_GET, buff, 0, NULL);
    sAPI_Debug("[CNTP] func[%s] line[%d] ret[%d] buff[%s]", __FUNCTION__,__LINE__,ret, buff);
    ret = sAPI_NtpUpdate(SC_NTP_OP_EXC, NULL, 0, ntpUIResp_msgq);
    sAPI_Debug("[CNTP] func[%s] line[%d] ret[%d] ", __FUNCTION__,__LINE__,ret );
    do
    {
        sAPI_MsgQRecv(ntpUIResp_msgq, &ntp_result, SC_SUSPEND);
        if(SC_SRV_NTP != ntp_result.msg_id ) //wrong msg received
        {
            sAPI_Debug("[CNTP] ntp_result.msg_id =[%d], ntp_result.msg_id ");
            ntp_result.msg_id = SC_SRV_NONE;                    //para reset
            ntp_result.arg1 = -1;
            ntp_result.arg3 = NULL;
            continue;
```

```
        }
        if(SC_NTP_OK == ntp_result.arg1)                    //it means update succeed
        {
            sAPI_Debug("[CNTP] successfully update time! ");
            PrintfResp("\r\nNTP Update Time Successful!\r\n");
            break;
        }
        else
        {
            sAPI_Debug("[CNTP] failed to update time! result code: %d", ntp_result.arg1);
            PrintfResp("\r\nNTP Update Time Failed!\r\n");
            break;
        }
    }while(1);
    memset(&currUtcTime,0,sizeof(currUtcTime));
    sAPI_GetSysLocalTime(&currUtcTime);
    sAPI_Debug("[CNTP] sAPI_GetSysLocalTime %d - %d - %d
- %d : %d : %d     %d",currUtcTime.tm_year,currUtcTime.tm_mon,currUtcTime.tm_mday,
currUtcTime.tm_hour,currUtcTime.tm_min,currUtcTime.tm_sec,currUtcTime.tm_wday);
    return;
}

int gps_on(void)
{
    int ret = 0;
    const int delay = 20;

    if( sAPI_GnssPowerStatusGet() == SC_GNSS_POWER_ON )
    {
        return 0;
    }
    Else
    {
        sAPI_GnssPowerStatusSet(SC_GNSS_POWER_ON);
    }
    sAPI_TaskSleep(90*delay);//等待 9 秒使得 GPS 动态加载升级软件版本完成(ASR1601 和 ASR1606 平台不需要该步骤，只需要 sAPI_TaskSleep(delay);)

    if(SC_GNSS_RETURN_CODE_ERROR == sAPI_GnssStartMode(SC_GNSS_START_HOT))
    {
        sAPI_Debug("%s: gps start mode set fail!",__func__);
    }
    sAPI_TaskSleep(delay);//  延迟 100ms 后再操作 GPS 的下一个操作，防止一连串的设置 GPS 反应不过来

    if(SC_GNSS_DEMO_RETURN_CODE_OK !=
```

```
(SC_Gnss_DEMO_Return_Code)sAPI_GnssAgpsSeviceOpen())
    {
        sAPI_Debug("%s: failed to use AGPS.!",__func__);
    }
    sAPI_TaskSleep(delay);

    if (SC_GNSS_RETURN_CODE_ERROR ==
sAPI_GnssNmeaDataGet(SC_GNSS_START_OUTPUT_NMEA_DATA,
SC_GNSS_NMEA_DATA_GET_BY_URC))//设置通过 URC 的方式获取 GPS 原始数据，此处设置后需要在
cus_urc.c 中去获取 GPS 原始数据
    {
        sAPI_Debug("%s: URC report error!",__func__);
    }
    //ret =
sAPI_GnssNmeaDataGet(SC_GNSS_START_OUTPUT_NMEA_DATA,SC_GNSS_NMEA_DATA_GET_B
Y_PORT);//设置通过 nmea 端口的方式查看 GPS 原始数据
    return ret;
}

int main()
{
    while(1);    //获取网络状态
    {
      sAPI_NetworkGetCgreg(&pGreg);
      if(1 != pGreg);
      {
          sAPI_Debug("[AGPS] NETWORK STATUS IS [%d] ", pGreg);
          sAPI_TaskSleep(10*300);
      }
      else
      {
           sAPI_Debug("[AGPS] NETWORK STATUS IS NORMAL");
           break;
       }
    }

    ntpUpdataTime();//ntp 更新系统时间

    gps_on();//开启 GPS
    return 0;
}
```

5 GNSS demo 演示步骤

用 ASR1603 A7670C_FASL 带动态加载版本的模块为示例（不带动态加载版本类似）：

1) 模块正常开机后从全功能 Uart 口（uart1）输出 UI 界面，输入 24 选择 GNSS 功能：

```
Please select an option to test from the items listed below.

************************************************************
1. NETWORK                    2. SIMCARD
3. SMS                        4. UART
5. USB                        6. GPIO
7. PMU                        8. I2C
9. AUDIO                      10. FILE SYSTEM
11. TCPIP                     12. HTTP
13. FTP                       14. MQTT
15. SSL                       16. FOTA
17. LBS                       18. NTP
19. HTP                       20. INTERNET SERVICE
21. TTS                       22. CALL
23. WIFI                      24. GNSS
25. LCD                       26. RTC
27. FLASH                     29. SPI
30. CAM                       31. APP UPDATE
32. LE CLIENT                 33. SPI NOR
************************************************************
```

2) 进入 GNSS 功能界面后，输入 1 进入 GNSS 芯片电源控制：

```
Please select an option to test from the items listed below, demo
just for GNSS.

************************************************************
1. GNSS power status          2. Get NMEA data
3. GNSS start mode            4. GNSS baud rate
5. GNSS mode                  6. GNSS nmea rate
7. GNSS nmea sentence         8. GPS information
9. GNSS information           10. Send command to GNSS
11. AGPS                      99. back
************************************************************
```

3) 进入 GNSS 芯片电源控制界面后，输入 1 给 GNSS 芯片上电：

```
************************************************************
1. power on                   2. power off
3. get power status           99. back
************************************************************

set power on!

************************************************************
1. power on                   2. power off
3. get power status           99. back
************************************************************
```

4) 给 GNSS 上电后输入 99 返回 GNSS 功能 UI 界面，输入 2 进入 GNSS 芯片数据输出控制功能：

```
Return to the previous menu!

*************************************************************
1. GNSS power status          2. Get NMEA data
3. GNSS start mode            4. GNSS baud rate
5. GNSS mode                  6. GNSS nmea rate
7. GNSS nmea sentence         8. GPS information
9. GNSS information           10. Send command to GNSS
11. AGPS                      99. back
*************************************************************
```

5) 进入 GNSS 芯片数据输出控制界面后输入 1 选择将 GNSS 芯片的数据输出到 NMEA 口：

```
*************************************************************
1. start get NMEA data by port2. stop get NMEA data by port
3. start get NMEA data by URC 4. stop get NMEA data by URC
99. back
*************************************************************

start get NMEA data by NMEA port!

*************************************************************
1. start get NMEA data by port2. stop get NMEA data by port
3. start get NMEA data by URC 4. stop get NMEA data by URC
99. back
*************************************************************
```

6) 此时便可在 NMEA 口中看到动态加载过程中 GNSS 反馈的数据接收信息（未解析的乱码）：

```
[14:29:25.827]收←◆□□□□
[14:29:25.907]收←◆□□□□□□□□
[14:29:26.019]收←◆□□□□□□□□□
[14:29:26.155]收←◆□□□□□□
[14:29:26.251]收←◆□□□□□□□□□□
[14:29:26.388]收←◆□□□□□□
[14:29:26.470]收←◆□□□□
[14:29:26.532]收←◆□□□□□□□□□□□□□
[14:29:26.719]收←◆□□□□□□□
[14:29:26.826]收←◆□□□□□□□□□□□□□□□□□□□
[14:29:27.095]收←◆□□□□
[14:29:27.157]收←◆□□□□□□□□□
[14:29:27.279]收←◆□□
[14:29:27.310]收←◆□□□□□□□□□
[14:29:27.454]收←◆□□□□□□□□□□□□□□□□□□□
[14:29:27.717]收←◆□□□□□□□□□□□□□□□□□□□□
## Total Size = 0x00042c00 = 273408 Bytes
## Start Addr = 0x00230000

[14:29:28.032]收←◆
 size of binary:0x42c□
[14:29:29.127]收←◆$GNRMC,062928.11,A,2929.45859,N,10638.06315,E,0.155,,220921,,,A,V*10
$GNGGA,062928.11,2929.45859,N,10638.06315,E,1,27,0.56,283.2,M,,M,,*5E
$GNGSA,A,3,05,10,13,15,18,23,24,32,194,199,195,,0.99,0.56,0.82,1*35
$GNGSA,A,3,06,08,09,13,16,19,20,22,29,35,36,01,0.99,0.56,0.82,4*0C
$GPGSV,3,1,12,05,21,089,36,10,19,301,36,13,17,043,33,15,45,036,42,0*6D
```

动态加载过程中GNSS芯片的响应乱码

动态加载完成

由于使用 UI 界面功能间切换产生的时间延迟，大多时候能看到的动态加载乱码如上，当乱码输出完后，动态加载功能完成，GNSS 芯片开始正常输出 NMEA 数据。整个动态加载过程产生的响应乱码如下（整个动态加载时间大致在 7-9 秒）：

```
[14:29:19.896]收←◆?C
[14:29:19.997]收←◆□
[14:29:20.097]收←◆□
[14:29:20.194]收←◆□
[14:29:20.291]收←◆□
[14:29:20.389]收←◆□
[14:29:20.485]收←◆□
[14:29:20.580]收←◆□
[14:29:20.676]收←◆□
[14:29:20.769]收←◆□
[14:29:20.865]收←◆□
[14:29:20.962]收←◆□
[14:29:21.058]收←◆□
[14:29:21.155]收←◆□□
[14:29:22.328]收←◆□
[14:29:23.442]收←◆□□□□□□□□□□□□
[14:29:23.641]收←◆□□□□□
[14:29:23.722]收←◆□□□□□□□□□□□□□□
[14:29:23.954]收←◆□□□
[14:29:24.015]收←◆□□□□□□□□
[14:29:24.124]收←◆□□□□□□□□□
[14:29:24.250]收←◆□□□
[14:29:24.310]收←◆□□□□□□□□□□□□□□□□
[14:29:24.564]收←◆□□
[14:29:24.607]收←◆□□□□□□□□□□□□□□□□□□
[14:29:24.890]收←◆□□□
[14:29:24.951]收←◆□□□□□□□□□□□□□□□□□□□
[14:29:25.250]收←◆□□□□□□□□□□□□□□□□□
[14:29:25.514]收←◆□□□□□
[14:29:25.596]收←◆□□□□□□□□□□□□□
[14:29:25.827]收←◆□□□□
[14:29:25.907]收←◆□□□□□□□□
[14:29:26.019]收←◆□□□□□□□□□
[14:29:26.155]收←◆□□□□□□
[14:29:26.251]收←◆□□□□□□□□□□
[14:29:26.388]收←◆□□□□□□
[14:29:26.470]收←◆□□□□
[14:29:26.532]收←◆□□□□□□□□□□□□□
[14:29:26.719]收←◆□□□□□□□
[14:29:26.826]收←◆□□□□□□□□□□□□□□□□□□
[14:29:27.095]收←◆□□□□
[14:29:27.157]收←◆□□□□□□□□□□
[14:29:27.279]收←◆□□
[14:29:27.310]收←◆□□□□□□□□□
[14:29:27.454]收←◆□□□□□□□□□□□□□□□□□□□
[14:29:27.717]收←◆□□□□□□□□□□□□□□□□□□□□
```

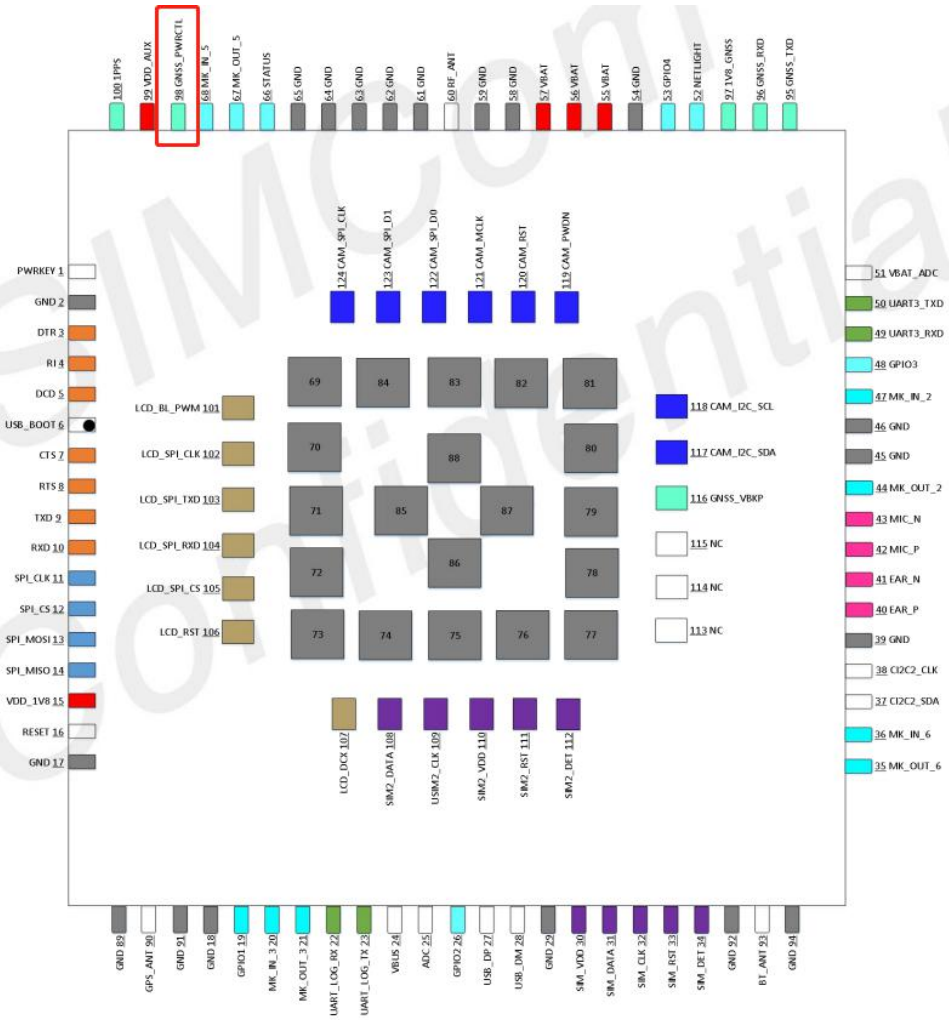
6 GNSS 常规问题文档合入

1) 给 GPS 上电后等待了一会未看到 nmea 语句输出，这是为什么？

答：1603 带动态加载的版本一般情况下是因为动态加载失败了，常见问题分为以下情况：

（1）在 GPS 的 TX RX 与 UART3 的 RX TX 之间的电阻没有替换为 1K，还是 10K 的电阻，此时替换电阻为 1K 就可解决。(相关 case：A7678-3033)

（2）二次开发中调用 GPS 电源 API 后，量下 GPS 的电源引脚是否上电 1.8v 的电压，引脚图如下：



（3）二次开发中，让客户检测下是否拉高了两次 GPS 电源引脚，GPS 电源引脚开机默认为低电平，上电两次会导致动态加载失败。

（4）客户自身代码时序设置或者代码设计等问题，因为 GPS 动态加载会使用 NVM 分区，而我们读写 nvm 文件系统的句柄 fd 只有 8 个，目前系统已经默认占用了几个，所以 GPS 动态加载需要注意和其他操作（网络注册，蓝牙开启等）分开来，不然会容易被抢占资源，最好执行完其他操作再开启 GPS 或者开启完 GPS 再执行其他操作。（相关 case A7678-10601、A7678-9621、A7678-9546、A7678-8311）

（5）模组芯片焊接是否有虚焊，模组是否有问题，硬件连接上是否正常，建议换块板子测试（相关 case：A76801606-1452，A7678-9087）

2) 连接了 GPS 天线，但是 GPS 很久不能定位，为什么？

答：（1）GPS 天线分有源天线和无源天线，ASR1603 模块的 TE 板默认只支持无源天线，如果想使用有源天线需要修改 TE 板，将 TE 板子的 GPS 天线端口处的 L1 短接了，参考如下图：

（2）天线检测无误后，查看 GPS 的信噪比（nmea 语句中的 GSV 字段）情况，信噪比太差表示环境影响，一般信噪比在 30 一下是不能定位的。

（3）模组芯片焊接问题、模组内部有些出厂缺料导致，建议换块板子试。

3) 动态加载后 GNSS 模块定位后还表现出静漂严重问题。

答：因为 GNSS 模块默认不是静态保持模式，GPS 正常运行后，标准版发送指令：AT+CGNSSCMD=0," $CFGDYN,h02,1,100"，二次开发调用 api: sAPI_SendCmd2Gnss("$CFGDYN,h02,1,100");，其中参数 100 是设置的静态保持模式下速度门限，单位为厘米/秒，该值为 0 会关闭静态保持模式。该指令在 GPS 芯片型号为 UC6226NIS 是可以掉电保存的，但在型号为 UC6228CI 是无法掉电保存。（相关 case：A7678-9999）

4) 如何查看目前 GPS 芯片的软件版本？

答：标准版：AT+CGNSSCMD=0,"$PDTINFO"

二次开发调用 api：sAPI_SendCmd2Gnss("$PDTINFO ");

发送后能够在 nmea 语句中查看到 GPS 芯片型号。

5) 如何查看 AGPS 是否下载成功到 GPS 芯片中？

答：方法 1 发送指令查看是否注入成功：

标准版：AT+CGNSSCMD=0,"$AIDINFO"

二次开发调用 api：sAPI_SendCmd2Gnss("$AIDINFO ");

详细字段解释如下：

输出辅助数据信息		
消息格式	$AIDINFO,GPSRS,GPSUS,BDSRS,BDSUS,GALRS,GALUS,GLORS,GLOUS,AType	
例子	$AIDINFO,003FFFFFF7,000000FA00,0000003F7F,0000001A3F,0000000000,0000000000,,,7	
描述	输出辅助数据的状态和辅助类型	
类型	输出	
参数定义		
参数名	类型	描述
GPSRS	UINT64	GPS 星历的接收状态，只要接收到的数据校验通过，则相应 bit 置 1， 如果 GPS 系统没有 enable，则此字段为空

GPSUS	UINT64	GPS 星历有效且可用于定位，则相应 bit 置 1，如果 GPS 系统没有 enable，则此字段为空
BDSRS	UINT64	BDS 星历的接收状态，只要接收到的数据校验通过，则相应 bit 置 1， 如果 BDS 系统没有 enable，则此字段为空
BDSUS	UINT64	BDS 星历有效且可用于定位，则相应 bit 置 1，如果 BDS 系统没有 enable，则此字段为空
GALRS	UINT64	GAL 星历的接收状态，只要接收到的数据校验通过，则相应 bit 置 1， 如果 GAL 系统没有 enable，则此字段为空
GALUS	UINT64	GAL 星历有效且可用于定位，则相应 bit 置 1，如果 GAL 系统没有 enable，则此字段为空
GLORS	UINT64	GLO 星历的接收状态，只要接收到的数据校验通过，则相应 bit 置 1， 如果 GLO 系统没有 enable，则此字段为空
GLOUS	UINT64	GLO 星历有效且可用于定位，则相应 bit 置 1，如果 GLO 系统没有 enable，则此字段为空
Atype	UINT	辅助类型 bit0-3：有 GPS/BDS/GAL/GLO 星历辅助 bit4：辅助位置有效 Bit5：使用辅助位置 Bit6-7：reserve Bit8：辅助时间有效 Bit9：使用辅助时间 Bit10-15：reserve

方法 2 抓取 catstudio log 查看是否成功，搜索关键词 aGPS_handle_thread，查看是否有 received data over, colse socket.打印，有打印是下载成功到 GPS 芯片中了的。

6) 使用 AGPS 后系统挂住了，这是为何？

答：这是因为客户使用了一分钟内调用了多次 AGPS，造成 agps 服务器概率性的拒绝访问回应，导致在发送给 GPS 星历数据过程进入了一个死循环，使得系统挂住。所以为了更好的服务质量，建议客户每天访问 AGPS 服务器次数不要超过 12 次，也就是每两小时一次。（相关 case：A7678-9271）

7) 调用休眠接口后使用 GPS 后系统概率性数据越界死机？

答：休眠前必须要关闭 GPS，否则调用休眠接口后串口时钟关闭，会存在串口工作不正常的情况，使得出现野指针导致系统死机。（相关 case: A7678-8140）