



BH-ATGM332D 用户手册

——秉火 GPS/北斗双模定位模块

修订历史

日期	版本	更新内容
2017/5/2	1.0.0	





文档说明

本手册旨在帮助用户正确构建 BH-ATGM332D 模块的使用环境，引导用户快速使用该模块。

关于核心模块 ATGM332D 的硬件参数请参考文档《ATGM332D-5N 卫星导航模块用户手册.pdf》。

关于 NMEA 传输协议请参考文档《CASIC 多模卫星导航接收机协议规范.pdf》。

关于如何修改模块的配置见文档《GNSSToolKit_Lite 简装版程序说明.pdf》。



目录

BH-ATGM332D 用户手册	1
文档说明.....	2
目录	3
1. BH-ATGM332D 模块说明	5
1.1 BH-ATGM332D 简介	5
1.2 产品特性参数.....	5
1.3 BH-ATGM332D 模块引脚说明	6
1.4 模块资源及配套天线.....	7
2. 模块原理图.....	9
3. 模块尺寸图.....	10
4. 硬件测试.....	10
4.1 准备开发环境.....	10
4.1.1 安装 USB 转串口驱动	10
4.1.2 秉火多功能调试助手（必须为 1.2.0 版本或以上）	11
4.2 测试 BH-ATGM332D 模块	16
4.3 坐标系及纠偏.....	17
4.3.1 坐标系不同而导致的偏移问题	17
4.3.2 秉火多功能调试助手纠偏说明	17
5. NMEA-0183 协议.....	19
5.1 NMEA-0183 简介	19
5.2 NMEA-0183 常用语句格式说明	20
5.2.1 协议帧总说明:	21
5.2.2 GGA	21
5.2.3 RMC	22
5.2.4 VTG	23
5.2.5 GLL	23
5.2.6 ZDA	23
5.2.7 GSA	24
5.2.8 GSV	24
5.3 NMEA 解码库	25
6. 使用单片机系统控制 BH-ATGM332D 模块.....	26
6.1 通用控制说明.....	26
6.2 秉火 STM32 开发板控制说明.....	26
6.2.1 连接模块.....	27
6.2.2 程序简介.....	30
6.2.3 实验现象.....	31
7. 代码分析.....	34
7.1 GPS_Decode_SDCard 例程.....	34



- 7.1.1 实验描述及工程文件清单34
 - 7.1.2 解码流程.....34
 - 7.1.3 结构体 nmeaPARSER 和 nmeaINFO37
 - 7.1.4 分配堆栈空间.....39
- 7.2 GPS_Decode_USART 例程40
 - 7.2.2 配置 DMA 串口.....40
- 8. 修改模块配置.....45
- 9. 常见问题.....45
- 10. 产品更新及售后支持.....47



1. BH-ATGM332D 模块说明

1.1 BH-ATGM332D 简介

BH-ATGM332D 是秉火设计的高性能、低功耗 GPS、北斗双模定位模块。它采用中科微电子公司的 ATGM332D-5N-31 模组方案，可以通过串口向单片机系统和电脑输出 GPS 及北斗定位信息，使用简单方便，其外观见图 1-1。

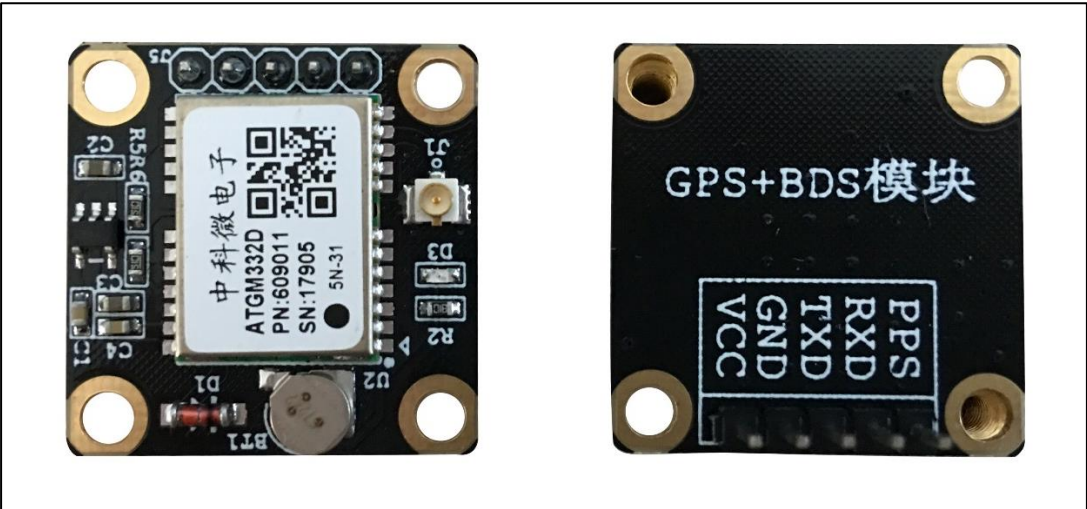


图 1-1 BH-ATGM332D GPS 定位模块

1.2 产品特性参数

BH-ATGM332D 模块产品特性参数见表 1-1。

表 1-1 BH-ATGM332D 模块产品特性

特性	说明
基本功能	<input type="checkbox"/> 三维位置定位(经纬度、海拔) <input type="checkbox"/> 测速 <input type="checkbox"/> 授时
导航系统	GPS、北斗 BDS（双模） 支持辅助 GNSS
位置定位精度	2.5 米(圆概率误差 CEP50)
测速精度	<0.1m/s
航向角精度	0.5 度
授时精度	<30ns
射频通道数目	支持全星座北斗 BDS、GPS 同时接收
定位时间	冷启动: ≤32s (各启动方式见 <input type="checkbox"/> 表 1-2) <input type="checkbox"/> 热启动: ≤1s



	<input type="checkbox"/> 重捕获: $\leq 1s$
冷启动捕获灵敏度	-148dBm
热启动捕获灵敏度	-156dBm
重捕获灵敏度	-160dBm
跟踪灵敏度	-162dBm
导航信息最高更新速率	1Hz (默认), 最大 10Hz
串口	<input type="checkbox"/> 预留有 TTL 电平标准的串口, 支持与使用 3.3/5V 电平标准的系统通讯 <input type="checkbox"/> 支持传输速率: 4800、9600、115200bps, 默认为 9600bps
协议	NMEA0183
输出的经纬度坐标系	WGS-84 坐标系
最大高度	18000m
最大速度	515m/s
最大加速度	4g
电源	<input type="checkbox"/> 通过模块引出的电源引脚 3.3~5V 供电 <input type="checkbox"/> 可充电电池, 在主电源断电后给备份域供电
工作温度	-40 到+85 摄氏度
保存温度	-40 到+125 摄氏度
功耗	连续运行<25mA (@3.3V)

表 1-2 GPS 模块启动方式

启动方式	说明
冷启动	定位模块是在完全无任何数据的状况下启动, 称为冷启动。 典型方式: <input type="checkbox"/> 初次使用时 <input type="checkbox"/> 电池耗尽导致星历信息丢失时 <input type="checkbox"/> 关机状态下将接收机移动 1000 公里以上距离
热启动	定位模块启动时仍然拥有有效的星历和年历(时间)数据的启动称为热启动。 典型方式: 与上次关机位置相同, 没有移动定位模块, 且备份域电池还在供电(一般为 2 小时之内)

1.3 BH-ATGM332D 模块引脚说明

BH-ATGM332D 模块的引脚说明见图 1-2 及表 1-3。



图 1-2 BH-ATGM332D 模块引脚说明

表 1-3 BH-ATGM332D 模块引脚说明

编号	名称	说明
1	VCC	电源线，正常电压范围为：3.3~5V
2	GND	地线
3	TXD	串口数据发送信号线，使用 TTL 电平
4	RXD	串口数据接收信号线，使用 TTL 电平
5	PPS	时间脉冲信号线，模块接收到 GPS 时间信息后，输出可调节的脉冲信号，默认为 1Hz，脉冲上升沿与 UTC 时间对齐

1.4 模块资源及配套天线

BH-ATGM332D 定位模块性能稳定、器件接口布局美观、方便使用，其资源描述见图 1-3。

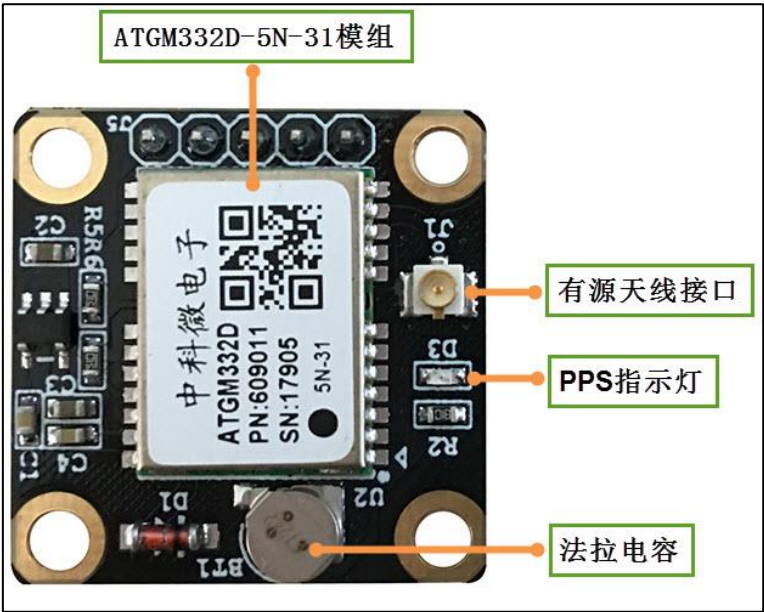


图 1-3 BH-ATGM332D GPS 定位模块资源描述图



表 1-4 模块资源描述表

编号	名称	说明
1	ATGM332D-5N-31 模块	ATGM332D-5N-31 是由中科微电子有限公司设计的 GPS+北斗双模定位方案，是本定位模块的核心。它具有快速的搜星能力，精准的定位效果，非常适合符合高性能、低功耗的应用场合。
2	XH414 法拉电容	XH414 法拉电容，参数为：3.3V 0.07F。它的功能和锂电池一样，在主电源掉电的时候可以为定位模块的 RTC 部分供电，以使定位模块在下次启动时能快速搜索到卫星，一般可持续供电 1 小时。
3	有源天线 IPX 接口	IPX 接口用于连接有源天线，本模块配套的有源天线见图 1-4 及表 1-5。
4	时间脉冲指示灯	模块上电后，时间脉冲指示灯即亮，在定位模块接收到时间信息后，时间脉冲信号指示灯会默认以 1Hz 的频率闪烁，该信号频率可以调节。



图 1-4 BH-ATGM332D GPS 模块及其配套的有源天线

表 1-5 BH-ATGM332D 配套有源天线参数

GPS 有源天线		规格
陶瓷	中心频率	1575±1MHz
	带宽	CF±8MHz
	极性	RHCP
	增益	5dB(Zenith)
	V. S. W. R	<1.5
	阻抗	50 Ω
	Axial Ratio	3dB(max)
	Dimensio	25*25*4mm
放大器	增益	28±2dB
	噪声系数	<1.0
	Filter Insertion Loss	<3dB
	Ex-band Attenuation	35dB@CF±50MHz/50dB@CF±100MHz
	供电电压	2.7~5V DC
	供电电流	16mA
	V. S. W. R	<2.0



机械	线材	RG174 96 编 铜喷锡
	接头形式	SMA
	Radome Material	PC
	Mounting Method	Magnet/Adhesive
工作环境	工作温度	-40℃~ +85℃
	Relative Humidity	Up to 95%
	防水等级	IP65~IP67
	Vibration	10 to 55Hz with 1.5mm amplitude 2hours
	Environmentally Friendly	ROHS 环保

2. 模块原理图

本模块的原理图见图 2-1，可放大查看。模块的原理图已整理到独立的 PDF 文档，也可查看配套资料里的《BH-ATGM332D 定位模块原理图.pdf》文件。

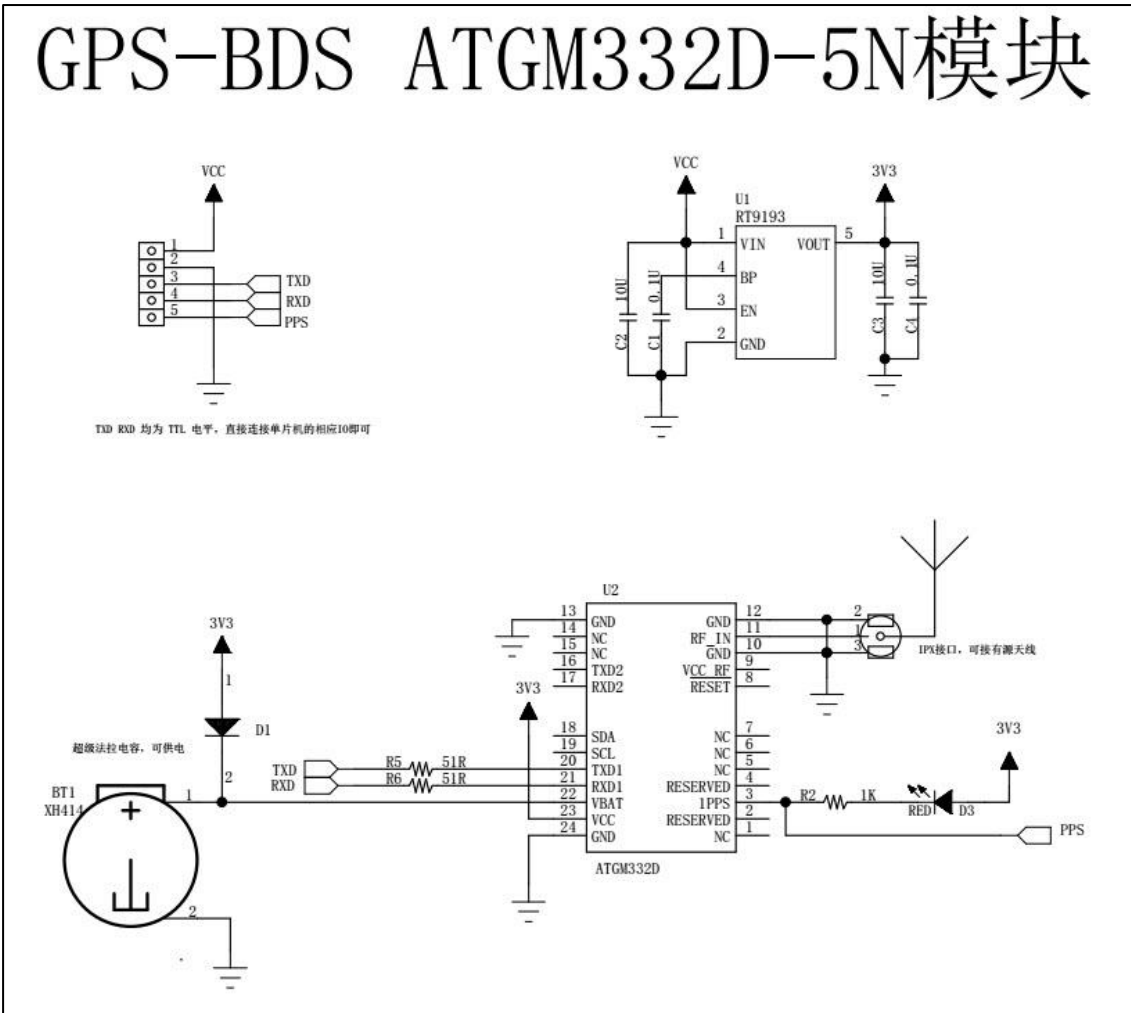


图 2-1 秉火 GPS 模块原理图



3. 模块尺寸图

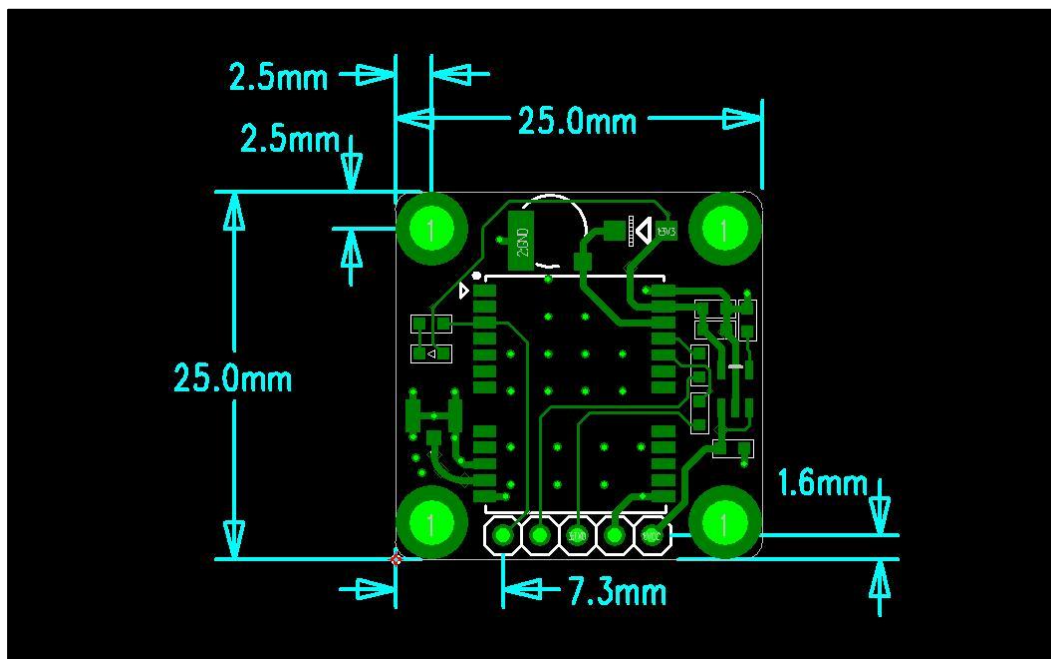


图 3-1 BH-ATGM332D 模块尺寸图

4. 硬件测试

使用电脑来测试 BH-ATGM332D 模块非常方便, 为此, 我们首先要准备好软硬件环境。主要是准备好 USB 转串口 TTL 模块和对应的驱动及秉火多功能调试助手。

4.1 准备开发环境

测试时需要使用 USB 转串口 TTL 模块连接电脑与 BH-ATGM332D 模块, 然后使用秉火多功能调试助手进行定位。

4.1.1 安装 USB 转串口驱动

BH-ATGM332D 模块使用串口通讯, 且通讯引脚电平为 TTL 类型, 所以与电脑通讯时需要使用 USB 转串口 TTL 的串口线作为媒介, 下面以我们推荐的 CH340 USB 转 TTL 线为例进行说明, 见图 4-1。



图 4-1 USB 转串口 TTL 线

该串口线的驱动芯片为 CH340，在使用前需要给电脑安装好驱动，在 BH-ATGM332D 模块配套资料压缩包的“配套软件”文件夹下可找到该驱动的安装文件，该驱动支持 XP、WIN7、WIN10，非 XP 用户使用用管理员身份安装即可。

安装完成后，把 USB 转 TTL 线接入电脑，如果 USB 转串口驱动安装成功，那么可在计算机->管理->设备管理器->端口中可查看识别到串口，具体见图 4-2。

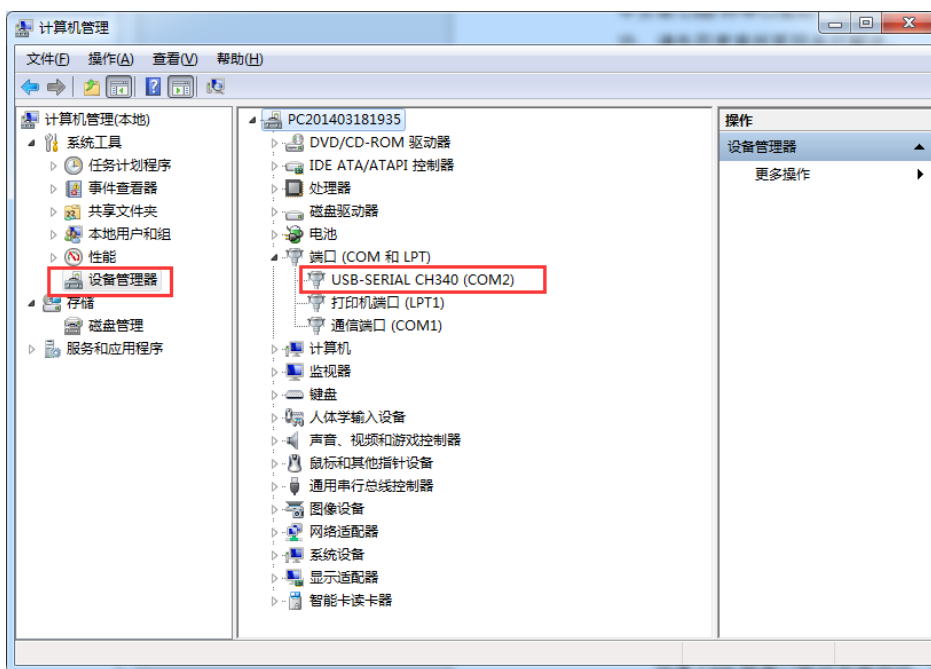


图 4-2 USB 转串口驱动安装成功

如果识别不了串口，请接到电脑的另一个 USB 接口，并重新安装驱动。

4.1.2 秉火多功能调试助手（必须为 1.2.0 版本或以上）

1. 软件简介

为了方便用户调试及使用 BH-ATGM332D 模块，秉火提供了多功能调试助手软件。配合 BH-ATGM332D 模块，该软件能显示定位模块通过 USB 转 TTL 线传回的原始信息，并对这些信息进行解码，得到时间、经纬度、海拔等数据，并根据解码得的结果在百度地图上标注实时位置，使应用定位模块变得更简单直观。



该软件是绿色免安装的, 直接打开即可使用, 如果无法打开, 请先安装 Windows 系统组件.Net Framework4.0。

软件中使用的百度地图需要联网使用, 在没有网络的情况下, 软件中的百度地图部分会加载异常。软件正常运行界面见图 4-3, **该软件有多个版本, 请注意必须使用 1.2.0 或以上的版本才能支持带北斗功能的定位模块。**

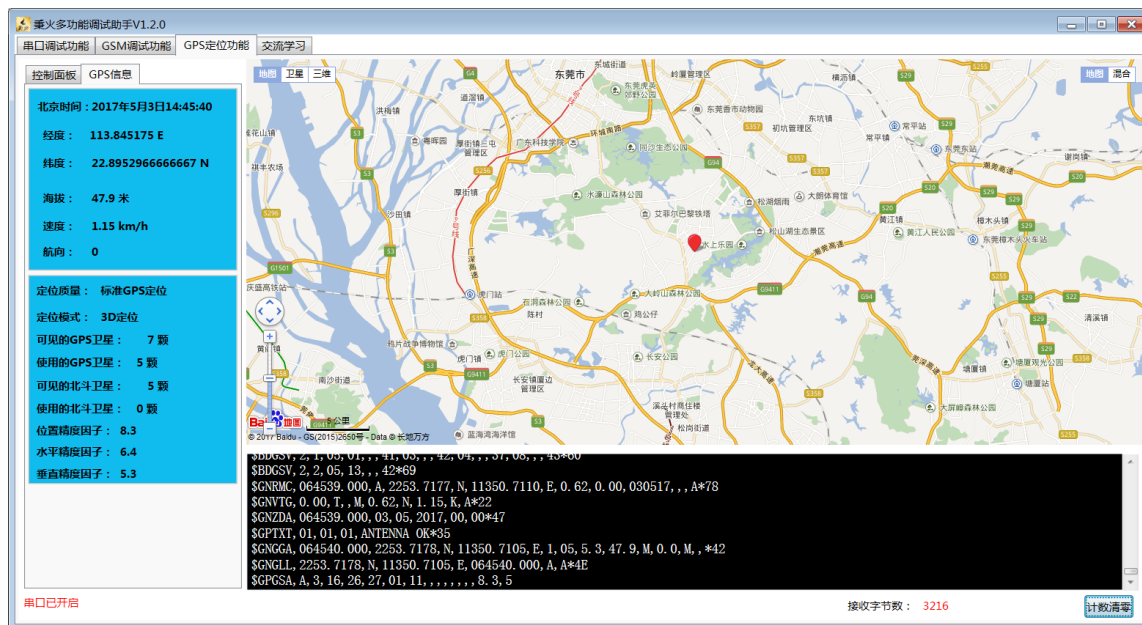


图 4-3 秉火多功能调试助手 1.2.0 版本(GPS 界面)

2. 使用方法

在模块配套资料包的“配套软件”文件夹可找到秉火多功能调试助手, 打开该软件后, 选择到 GPS 调试功能界面, 把 BH-ATGM332D 模块通过 USB 转串口 TTL 线连接到电脑, 见图 4-4, 该软件即可检测到新的 COM 口(若没有检测到 COM 口, 请检查 CH340 驱动程序是否正确安装), 选择 USB 转串口 TTL 线所用的 COM 口, 及默认**波特率 9600**, 然后点击“打开串口”按钮, 即可接收到定位模块传回的信息, 见图 4-5。

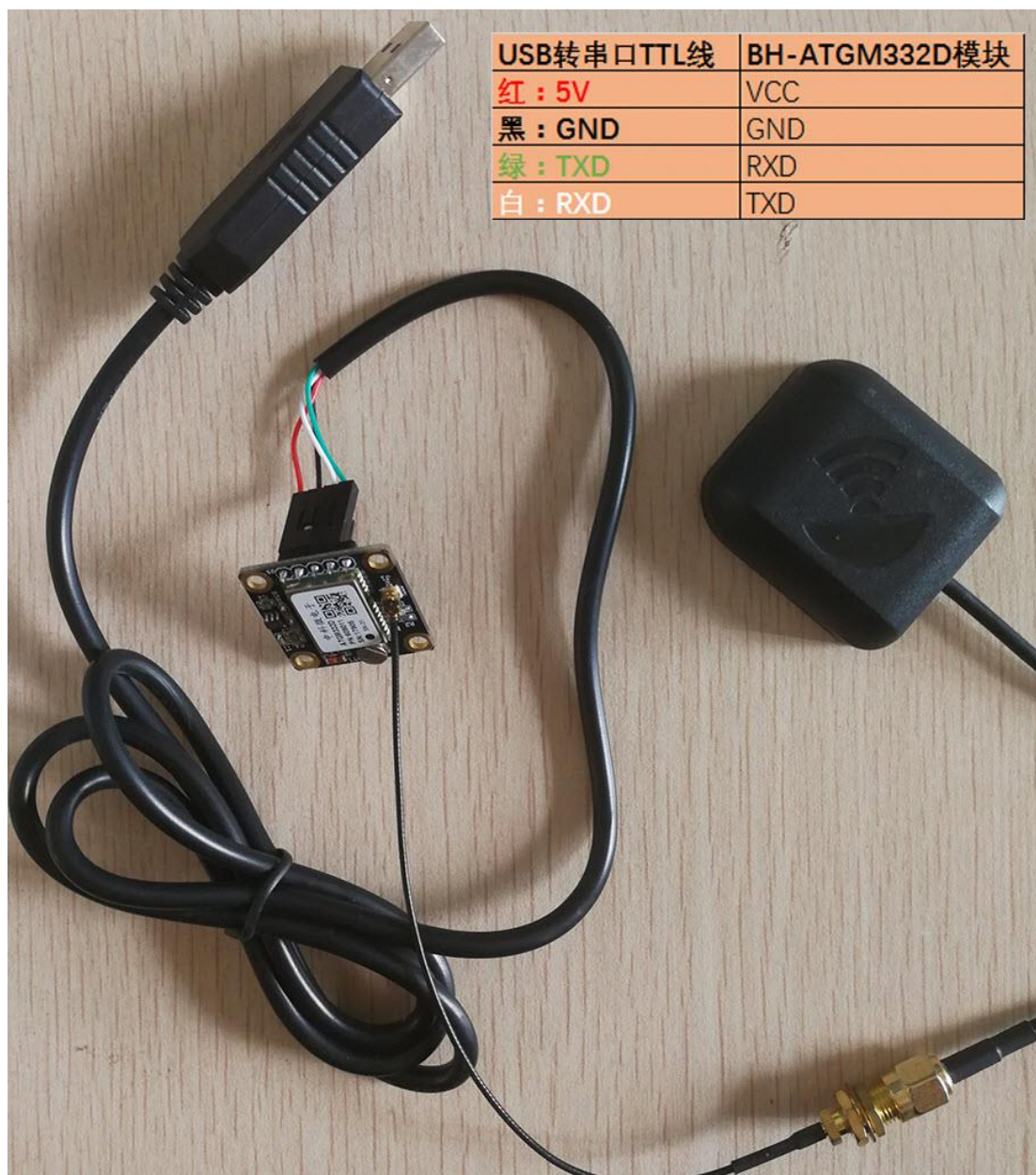


图 4-4 模块通过 USB 转串口 TTL 线与电脑连接

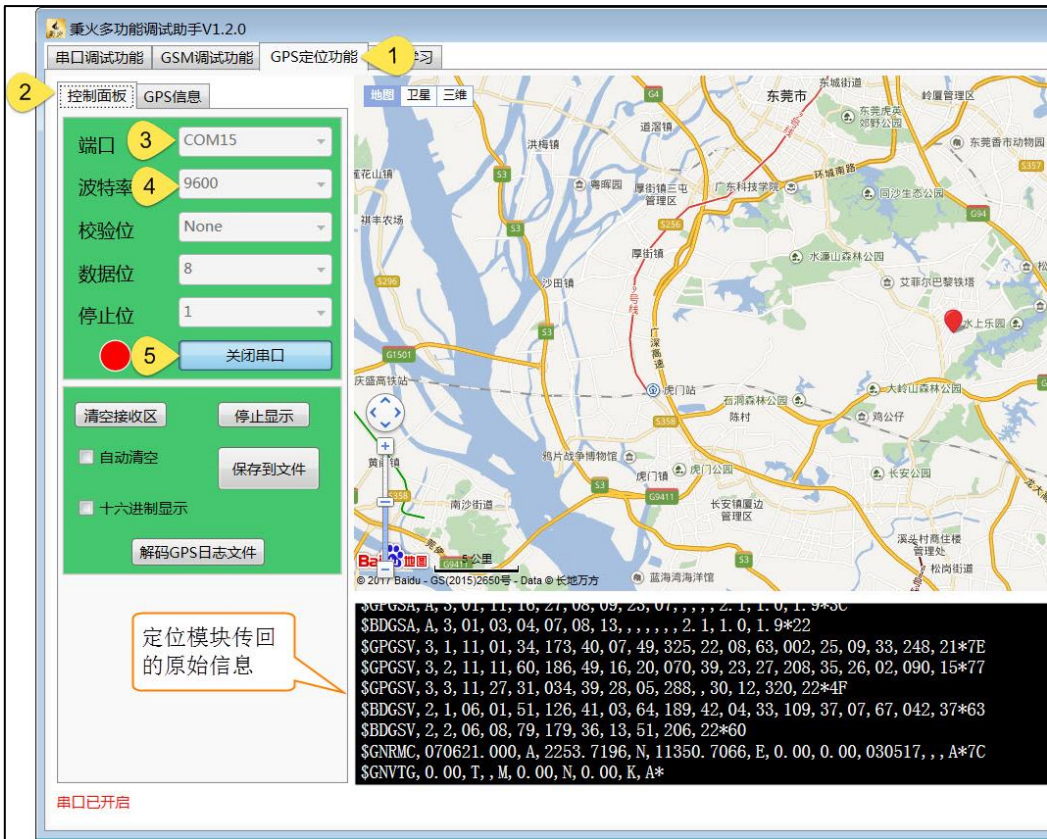


图 4-5 打开 USB 转串口 TTL 线对应的串口

除了解码实时由定位模块传回的信息，本软件还支持对 GPS 日志文件（也支持北斗，下同）解码。GPS 日志文件即保存了原始定位数据信息的文件，使用本软件从定位模块接收到这些信息后，可以把这些原始数据以 TXT 文本格式保存起来，得到 GPS 日志文件，方便以后使用。在使用日志文件解码时，点击控制面板中的“解码 GPS 日志文件”加载该文件即可，在秉火多功能调试助手软件目录中提供了一个 GPS 日志文件“gpslog.txt”，用户可以使用它来测试软件的功能，加载该文件后，软件会输出 GPS 解码结果并在地图上标注日志中记录的位置，见图 4-6。

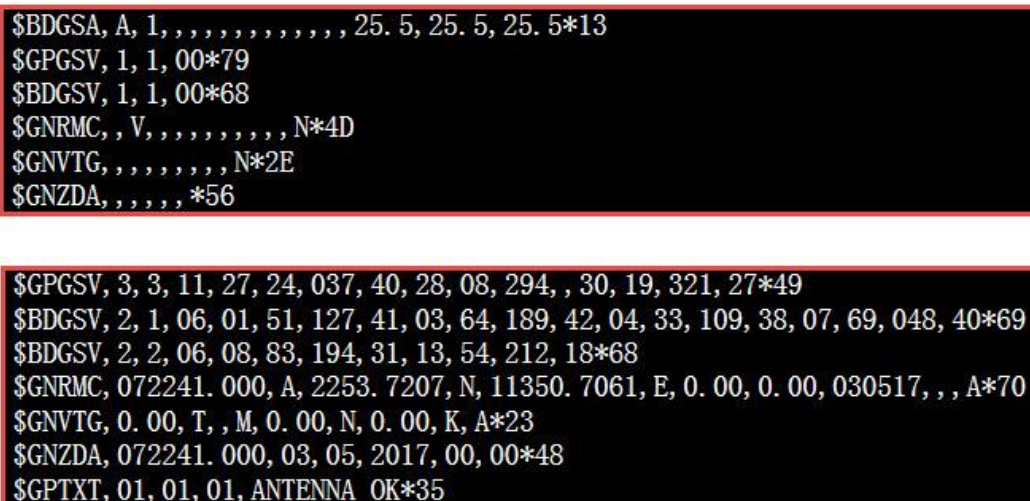




4.2 测试 BH-ATGM332D 模块

使用秉火多功能调试助手可方便地测试 BH-ATGM332D 模块是否正常, 测试步骤如下:

1. 确保开发环境正常
 - ❑ 检查是否正常安装好 CH340 USB 转串口 TTL 驱动
 - ❑ 使用 GPS 日志文件检验秉火多功能调试助手是否正常运行
2. 使用 USB 转串口 TTL 线连接电脑与 BH-ATGM332D 模块, 并给模块连接上有源天线。
正常时, 模块上的红色时间脉冲指示灯亮, 在调试助手软件上打开 USB 转串口 TTL 线对应的串口, 它的数据输出窗口会输出定位的原始数据(这些原始数据使用 NMEA0183 协议格式)。输出的数据一般会出现两种情况, 见图 4-7。



```
$BDGSA, A, 1, , , , , , , , , 25. 5, 25. 5, 25. 5*13
$GPGSV, 1, 1, 00*79
$BDGSV, 1, 1, 00*68
$GNRMC, , V, , , , , , , N*4D
$GNVTG, , , , , , , N*2E
$GNZDA, , , , , , *56

$GPGSV, 3, 3, 11, 27, 24, 037, 40, 28, 08, 294, , 30, 19, 321, 27*49
$BDGSV, 2, 1, 06, 01, 51, 127, 41, 03, 64, 189, 42, 04, 33, 109, 38, 07, 69, 048, 40*69
$BDGSV, 2, 2, 06, 08, 83, 194, 31, 13, 54, 212, 18*68
$GNRMC, 072241. 000, A, 2253. 7207, N, 11350. 7061, E, 0. 00, 0. 00, 030517, , , A*70
$GNVTG, 0. 00, T, , M, 0. 00, N, 0. 00, K, A*23
$GNZDA, 072241. 000, 03, 05, 2017, 00, 00*48
$GPTXT, 01, 01, 01, ANTENNA OK*35
```

图 4-7 信号差与信号良好时的定位数据信息

这两图的区别是, 上图中的 GPS 数据信息数据间有很多连续的“逗号”, 而下图中逗号与逗号之间一般是有数字的, 它们分别对应了 GPS 信号差与 GPS 信号良好的状况。

如果模块上电后输出的数据长期处于第一种状态, 应考虑转移一下定位模块天线的位置, 一般在室内卫星信号会比较差, 可到室外空旷的地方测试(如楼顶、阳台、窗边), 另外, 秉火多功能调试助手加载的百度地图是需要联网的时候才能正常使用的, 所以在室外无网络的地方, 测试时可把定位数据以文件格式保存起来, 在联网的情况下再加载定位日志文件进行解码。

实际上, 当调试助手的信息窗口显示接收到连续的以 \$GPXXX、\$BDXXX、\$GNXXX 开头的数字时, 已经说明 GPS 模块正常了, 当然, 软件在地图上标注出当前地点才是我们追求的目标!

特别地, BH-ATGM332D 模块还会输出当前的天线连接状态, 天线存在图 4-8 中的三种状态, 该数据形如 “\$GPTXT, 01, 01, 01, ANTENNA 状态*无关数字”的格式。



天线开路，没连接天线
\$GPTXT, 01, 01, 01, ANTENNA OPEN*25
天线短路
\$GPTXT, 01, 01, 01, ANTENNA SHORT*63
天线连接良好
\$GPTXT, 01, 01, 01, ANTENNA OK*35

图 4-8 定位模块的天线状态

图中的三种状态分别为开路（OPEN）、SHORT（短路）及 OK（正常），测试时请确保天线处于 OK 状态。

4.3 坐标系及纠偏

4.3.1 坐标系不同而导致的偏移问题

BH-ATGM332D 模块采用 NMEA-0183 协议输出定位信息，这些定位的经纬度信息采用的是 WGS-84 坐标系，把解码得到的经纬度数据输入到谷歌地图，可在地图上定位到正确的位置。

出于安全考虑，我国发布的电子地图一般是经过加密和增加了非线性位置偏移的，通常把这些地图称为“火星坐标系”，若直接把定位模块输出的基于 WGS-84 坐标系得到的经纬度输入到这些电子地图中查询位置，查到的地图定位会与实际位置存在较大的偏差，这并不是定位模块定位不精准，而是坐标系不一致而产生的问题。

为解决上述问题，需要找到对应电子地图对外提供的 API，如百度地图开放平台提供了 API，见：http://lbsyun.baidu.com/jsdemo.htm#a5_2

调用“原始坐标转换成百度坐标”的 API，并给 API 输入 WGS-84 坐标系的经纬度，才可以在地图上标注出正确的地点，资料里提供的“秉火多功能调试助手”软件中的定位功能就是这样实现的，关于该软件的源码可在如下帖子找到：

秉火多功能调试助手上位机开源！（C#/WPF--VS2013 开发环境）

<http://www.firebbs.cn/forum.php?mod=viewthread&tid=11985&fromuid=64>

4.3.2 秉火多功能调试助手纠偏说明

下面以秉火多功能调试助手的纠偏方式进行说明。

如使用秉火多功能调试助手解码日志文件数据：

“\$GPGLL,2303.00685,N,11324.42092,E,032550.00,A,D*6C”的解码结果见图 4-9。



调试助手的“GPS 信息”面板中显示的是 **WGS84 坐标系 度.度** 格式：经度
113.407015333333E 纬度 23.0501141667

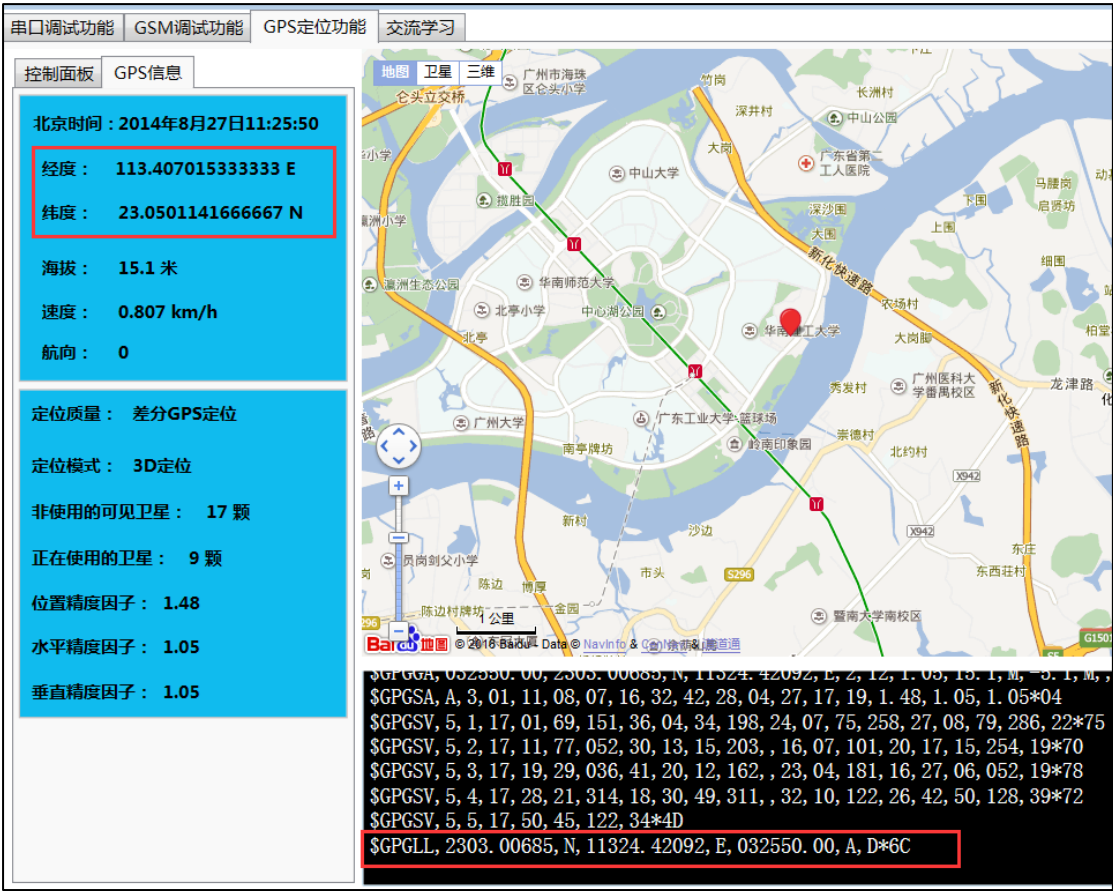


图 4-9 使用秉火多功能调试助手解码并定位

直接把该 **WGS84 坐标系**结果输入到百度地图反查，与调试助手的定位结果明显有差异，调试助手的是正常的，反查的有错误。



图 4-10 直接使用百度提供的坐标反查系统定位

出现定位偏差的原因是：输入反查系统的是 WGS-84 坐标系，而该系统使用百度自身的坐标系（类似火星坐标系），坐标系不对应。

要解决这个问题，只能通过百度提供的 javascript 地图接口纠偏，见：

http://lbsyun.baidu.com/jsdemo.htm#a5_2

火星坐标 API 转换，这些是百度的 javascript api，具体应用请查百度文档。

```
1 var gpsPoint = new BMap.Point(Longitude, Latitude);
2
3 //gps 坐标纠偏
4 //真实经纬度转成百度坐标
5 BMap.Convertor.translate(gpsPoint, 0, translateCallback);
6
```

秉火多功能调试助手就是把这个纠偏输出结果放到地图定位的。

5. NMEA-0183 协议

5.1 NMEA-0183 简介

BH-ATGM332D 模块通过 TTL 串口输出定位数据信息，这些信息默认采用 NMEA-0183 4.0 协议，输出的信息形前面的图 4-7 所示。



NMEA 是美国国家海洋电子协会（National Marine Electronics Association）为海用电子设备制定的标准格式，目前已经成为了 GPS 导航设备统一的 RTCM 标准协议，本模块使用的 NMEA 4.0 版本协议支持 GPS、北斗、海格纳斯等定位系统。

NMEA-0183 是一套定义接收机输出的标准信息，有几种不同的格式，每种都是独立相关的 ASCII 格式，使用逗号隔开数据，数据流长度从 30-100 字符不等，通常以每秒间隔选择输出，最常用的格式为"GGA"，它包含了定位时间，纬度，经度，高度，定位所用的卫星数，DOP 值，差分状态和校正时段等，其他的有速度，跟踪，日期等。NMEA 实际上已成为所有的定位接收机中最通用的数据输出格式。

5.2 NMEA-0183 常用语句格式说明

定位模块使用 NMEA 协议输出的原始数据形如：

“\$GNZDA,012840.000,14,01,2017,00,00*47”，这些语句都使用表 5-1 中的框架。

表 5-1 NMEA 协议框架

\$	<地址段>	{,数值}	*<校验和>	<CR><LF>
起始符	地址段	数据段	校验和段	结束符
每条语句以'\$'开始	分为发送器标识符和语句类型	以','开始，后面的数据值长度有可变或定长的类型	对'\$'和'*'之间的数据（不包括这两个字符）按字节进行异或运算的结果，用十六进制格式表示	每条语句都以换符<CR><LF>结束
示例：	\$GNZDA,012840.000,14,01,2017,00,00*47			
\$	GNZDA	,012840.000,14,01,2017,00,00	*47	<CR><LF>
语句开始	发送器标识符为：GN 语句类型为：ZDA	ZDA 语句主要表示时间和日期信息，此处表示：2017 年 01 月 14 日 01 时 28 分 40 秒 000 毫秒	校验和为 47	语句结束

NMEA 语句的数据段为信息主体，不同类型的语句用于传输不同类型的定位信息，其语句类型又分为两部分，如 GNZDA 前面两个字符 GN 用于区分定位系统，其它类型的还有中国北斗（BD）、美国 GPS（GP）、俄罗斯 GLONASS（GL），见表 5-2。

表 5-2 发送器类型

发送器	标识符
北斗导航卫星系统（BDS）	BD
全球定位系统（GPS、SBAS、QZSS）	GP
全球导航卫星系统（GLONASS）	GL



全球导航卫星系统（GNSS）	GN
自定义信息	P

其中 GN 标识符比较特殊，当发送器具有多模功能时（即同时支持一个以上的定位系统），系统会把各系统的信息整合、处理后，再把这些综合信息采用 GN 作为标识符发送出来，如前面的时间日期信息，使用 GNZDA 语句，在这样的系统中，GP、BD 等标识符仅用于表示对应系统的卫星信息，如 GPGSA 和 BDGSA 语句分别用于表示美国 GPS 系统和北斗系统的卫星信息。

NMEA-0183 协议定义的语句非常多，但是常用的或者说兼容性最广的语句只有 GGA、RMC、VTG、GLL、ZDA、GSA、GSV 等。下面给出这些常用 NMEA-0183 语句的字段定义解释。

表 5-3 语句类型

命令	说明
GGA	全球定位数据
RMC	推荐最小数据
VTG	地面速度信息
GLL	大地坐标信息
ZDA	UTC 时间和日期
GSA	卫星 PRN 数据
GSV	卫星状态信息

5.2.1 协议帧总说明：

该协议采用 ASCII 码。帧格式形如：\$aacc,ddd,ddd,...,ddd*hh<CR><LF>

- <1> “\$”——帧命令起始位
- <2> aacc——地址域，前两位为识别符，后三位为语句名
- <3> ddd...ddd——数据
- <4> “*”——校验和前缀
- <5> hh——校验和（check sum），\$与*之间所有字符 ASCII 码的校验和（各字节做异或运算，得到校验和后，再转换 16 进制格式的 ASCII 字符。）
- <6> <CR><LF>——CR（Carriage Return）+ LF（Line Feed）帧结束，回车和换行

5.2.2 GGA

GPS 固定数据输出语句(Global positioning system fix data)。

格式：

\$GNGGA,<1>,<2>,<3>,<4>,<5>,<6>,<7>,<8>,<9>,<10>,<11>,<12>,<13>,<14>*<15><CR>
<LF>

例子：\$GNGGA,012842.000,2253.7220,N,11350.7025,E,1,11,1.5,44.8,M,0.0,M,,*44

- <1> UTC 时间，格式为 hhmmss.sss



- <2> 纬度, 格式为 ddmm.mmmm (前导位数不足则补 0)
- <3> 纬度半球, N 或 S (北纬或南纬)
- <4> 经度, 格式为 dddmm.mmmm (前导位数不足则补 0)
- <5> 经度半球, E 或 W (东经或西经)
- <6> 定位质量指示, 0=定位无效, 1=标准定位, 2=差分定位, 6=估算
- <7> 使用卫星数量, 从 00 到 12 (前导位数不足则补 0)
- <8> 水平精确度, 0.5 到 99.9
- <9> 天线离海平面的高度, -9999.9 到 9999.9 米
- <10> 高度单位, M 表示单位米
- <11> 大地椭圆面相对海平面的高度 (-999.9 到 9999.9)
- <12> 高度单位, M 表示单位米
- <13> 差分 GPS 数据期限 (RTCM SC-104), 最后设立 RTCM 传送的秒数量
- <14> 差分参考基站标号, 从 0000 到 1023 (前导位数不足则补 0)
- <15> 校验和。

5.2.3 RMC

推荐最小数据量的 GPS 信息(Recommended Minimum Specific GPS/TRANSIT Data)。

格式:

\$ GNRMC,<1>,<2>,<3>,<4>,<5>,<6>,<7>,<8>,<9>,<10>,<11>,<12>,<13>*<14><CR><L

F>

例子: \$GNRMC,012841.000,A,2253.7220,N,11350.7025,E,0.00,0.00,140117,,,A*7B

- <1> UTC (Coordinated Universal Time) 时间, hhmmss (时分秒) 格式
- <2> 定位状态, A=有效定位, V=无效定位
- <3> Latitude, 纬度 ddmm.mmmm (度分) 格式 (前导位数不足则补 0)
- <4> 纬度半球 N (北半球) 或 S (南半球)
- <5> Longitude, 经度 dddmm.mmmm (度分) 格式 (前导位数不足则补 0)
- <6> 经度半球 E (东经) 或 W (西经)
- <7> 地面速率 (000.0~999.9 节, Knot, 前导位数不足则补 0)
- <8> 地面航向 (000.0~359.9 度, 以真北为参考基准, 前导位数不足则补 0)
- <9> UTC 日期, ddmmyy (日月年) 格式
- <10> Magnetic Variation, 磁偏角 (000.0~180.0 度, 前导位数不足则补 0)
- <11> Declination, 磁偏角方向, E (东) 或 W (西)
- <12> Mode Indicator, 模式指示 (仅 NMEA0183 3.00 版本输出, A=自主定位, D=差分, E=估算, N=数据无效)
- <13> NavStatus, 导航状态标示符 (V 表示系统不输出导航状态信息)
- <14> 校验和。



5.2.4 VTG

地面速度信息(Course over ground and Ground speed)。

格式: \$ GNVTG,<1>,<2>,<3>,<4>,<5>,<6>,<7>,<8>,<9>*<10><CR><LF>

例子: \$GNVTG,0.00,T,,M,0.00,N,0.00,K,A*23

- <1> 以真北为参考基准的地面航向
- <2> T, 表示“真”
- <3> 以磁北为参考基准的地面航向
- <4> M, 表示“磁场”
- <5> 地面速率
- <6> N, 表示“节”
- <7> 地面速率
- <8> K, 表示“千米/小时”
- <9> 模式指示 (A=自主定位, D=差分, E=估算, N=数据无效)
- <10> 校验和

5.2.5 GLL

定位地理信息(Latitude and longitude, with time of position fix and status)

格式: \$ GNGLL,<1>,<2>,<3>,<4>,<5>,<6>,<7>*<8><CR><LF>

例子: \$GNGLL,2253.7220,N,11350.7025,E,012842.000,A,A*4D

- <1> 纬度 ddmm.mmmmm (度分)
- <2> 纬度半球 N (北半球) 或 S (南半球)
- <3> 经度 dddmm.mmmmm (度分)
- <4> 经度半球 E (东经) 或 W (西经)
- <5> UTC 时间: hhmmss (时分秒)
- <6> 定位状态, A=有效定位, V=无效定位
- <7> 模式指示 (A=自主定位, D=差分, E=估算, N=数据无效)
- <8> 校验和

5.2.6 ZDA

当前时间信息: (Time and Date)

格式: \$ GNZDA,<1>,<2>,<3>,<4>,<5>,<6>*<7><CR><LF>

例子: \$GNZDA,012841.000,14,01,2017,00,00*46

- <1> UTC 时间: hhmmss (时分秒, 格林威治时间)
- <2> 日
- <3> 月
- <4> 年



- <5> 本地区域小时 (NEO-6M 不支持, 为 00)
- <6> 本地区域分钟 (NEO-6M 不支持, 为 00)
- <7> 校验和

5.2.7 GSA

GPS 精度指针及使用卫星 (GNSS DOP and Active Satellites)。

格式:

\$XXGSA,Smode,FS{,SVID},PDOP,HDOP,VDOP*CS<CR><LF>

例子: \$GPGSA,A,3,05,13,02,30,15,24,,,,,,,,2.2,1.5,1.6*35

\$BDGSA,A,3,01,03,04,08,12,,,,,,,,2.2,1.5,1.6*2D

- <1> 模式 1: 定位型式 1 = 未定位, 2 = 二维定位, 3 = 三维定位
- <2> FS: 定位状态标志
- <3> {,SVID}: 用于定位的卫星编号, 该字段共显示 12 颗可用卫星编号, 多于 12 颗时只输出前 12 颗, 不足 12 颗时不足的区域补空
- <4> PDOP 综合位置精度因子 (0.5 - 99.9)
- <5> HDOP 水平精度因子 (0.5 - 99.9)
- <6> VDOP 垂直精度因子 (0.5 - 99.9)
- <7> systemId: NMEA 所定义的 GNSS 系统 ID 号
- <8> 校验和

5.2.8 GSV

可视卫星状态输出语句 (GNSS Satellites in View)。

格式: \$XXGSV,NumMsg,MsgNo,NumSv{,SVID,ele,az,cn0} *CS<CR><LF>

例子: \$GPGSV,3,1,09,02,42,118,49,05,38,041,47,06,05,128,39,13,74,039,41*77

\$GPGSV,3,2,09,15,68,244,28,20,45,325,21,24,15,180,28,29,47,278,23*72

- <1> 总的 GSV 语句电文数
- <2> 当前 GSV 语句号
- <3> 可视卫星总数, 00 至 12
- <4> 卫星编号{,SVID,ele,az,cn0}, 01 至 32
- <5> 信噪比 (C/No), 00 至 99dB; 无表示未接收到讯号
- <6> 校验和。

注: 每条语句最多包括四颗卫星的信息, 每颗卫星的信息有四个数据项, 即: 卫星编号、卫星仰角、卫星方位角、信噪比。



5.3 NMEA 解码库

了解了 NMEA 格式有之后,我们就可以编写相应的解码程序了,而程序员 Tim (xtimor@gmail.com)提供了一个非常完善的 NMEA 解码库,在以下网址可以下载到:
<http://nmea.sourceforge.net/>, 直接使用该解码库,可以避免重复发明轮子的工作。在秉火提供的 GPS 模块资料的“NMEA0183 解码库源码”文件夹中也包含了该解码库的源码,秉火提供的 STM32 程序就是使用该库来解码 NMEA 语句的。

该解码库目前最新为 0.5.3 版本,它使用纯 C 语言编写,支持 windows、winCE、UNIX 平台,支持解析 GPGGA, GPGSA, GPGSV, GPRMC, GPVTG 这五种语句(这五种语句已经提供足够多的 GPS 信息),解析得的 GPS 数据信息以结构体存储,附加了地理学相关功能,可支持导航等数据工作,除了解析 NMEA 语句,它还可以根据随机数产生 NMEA 语句,方便模拟。在该解码库之上,秉火扩展了其对 NMEA-0183 4.0 版本的支持。



6. 使用单片机系统控制 BH-ATGM332D 模块

6.1 通用控制说明

BH-ATGM332D 支持 TTL 电平的串口通讯标准，非常方便使用单片机系统来控制。本小节以秉火 STM32 开发板为例子说明如何使用 STM32 来控制 BH-ATGM332D 模块。

单片机系统通过串口与 BH-ATGM332D 模块通讯，与模块连接时，只要通过模块引出的排针连接好如下四根线即可，注意**模块与单片机的收发引脚要交叉连接**，见

表 6-1 及图 6-1。

表 6-1 单片机与 BH-ATGM332D 模块连接引脚表

单片机系统	BH-ATGM332D 模块
5V 或 3.3V	VCC
GND	GND
串口 RXD	TXD
串口 TXD	RXD
可不接	PPS (信号灯输出引脚)

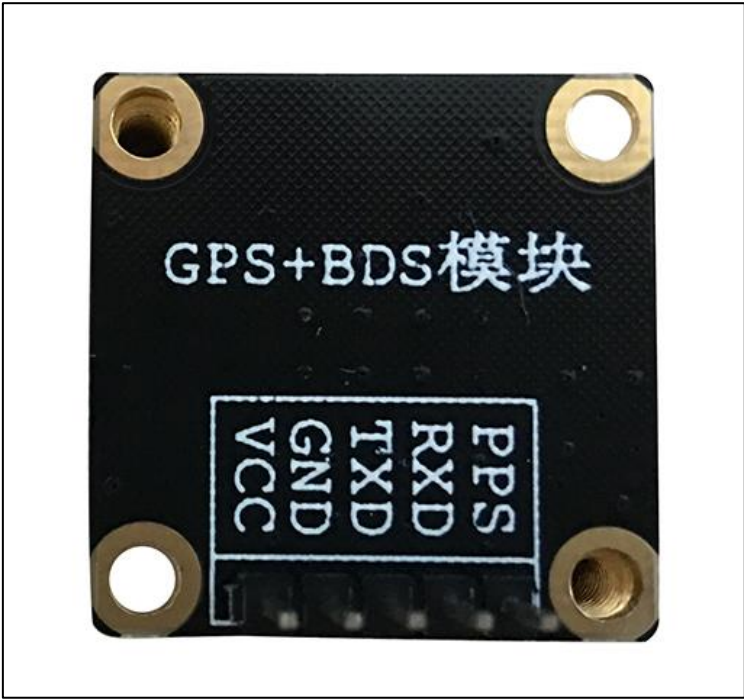


图 6-1 单片机与 BH-ATGM332D 模块连接引脚表

6.2 秉火 STM32 开发板控制说明

BH-ATGM332D 模块配套有适用于秉火 STM32 开发板的源码，用户可以参考它来编写自己的应用。



6.2.1 连接模块

1. 与秉火 F103 霸道、F103 指南者及 F407 霸天虎系列开发板的连接

秉火 F103 霸道、F103 指南者及 F407 霸天虎板子配套的例程，都是通过 STM32 的 USART2 外设控制 BH-ATGM332D 模块的，连接引脚说明见表 6-2、图 6-2 及图 6-4。

表 6-2 秉火开发板与 BH-ATGM332D 模块连接

F103 霸道/F103 指南者/F407 霸天虎	BH-ATGM332D 模块
5V/3.3V	GPS_5V
GND	GPS_GND
PA3/USART2_RX	GPS_TXD
PA2/USART2_TX	GPS_RXD

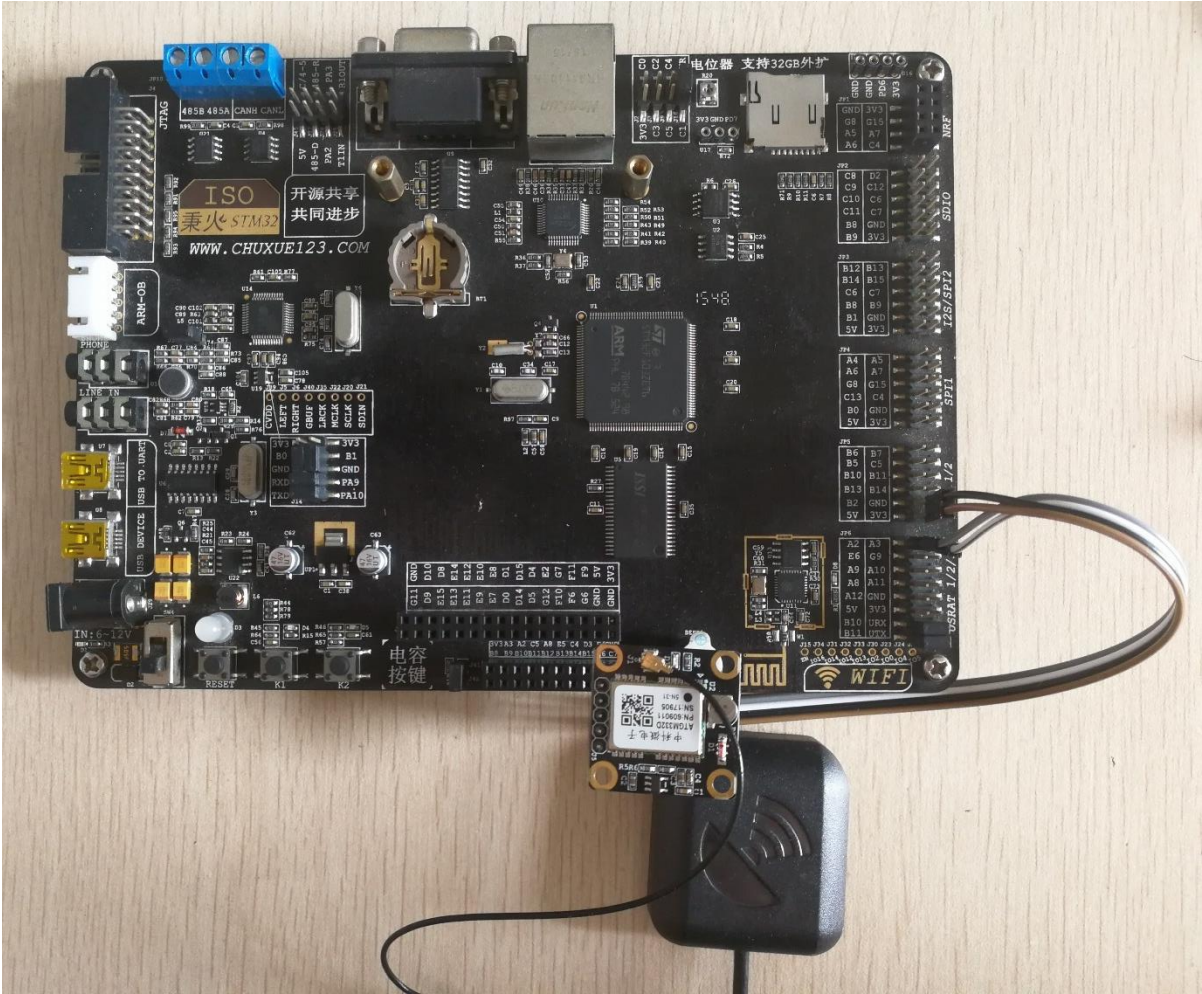


图 6-2 秉火 F103 霸道开发板与 BH-ATGM332D 模块连接

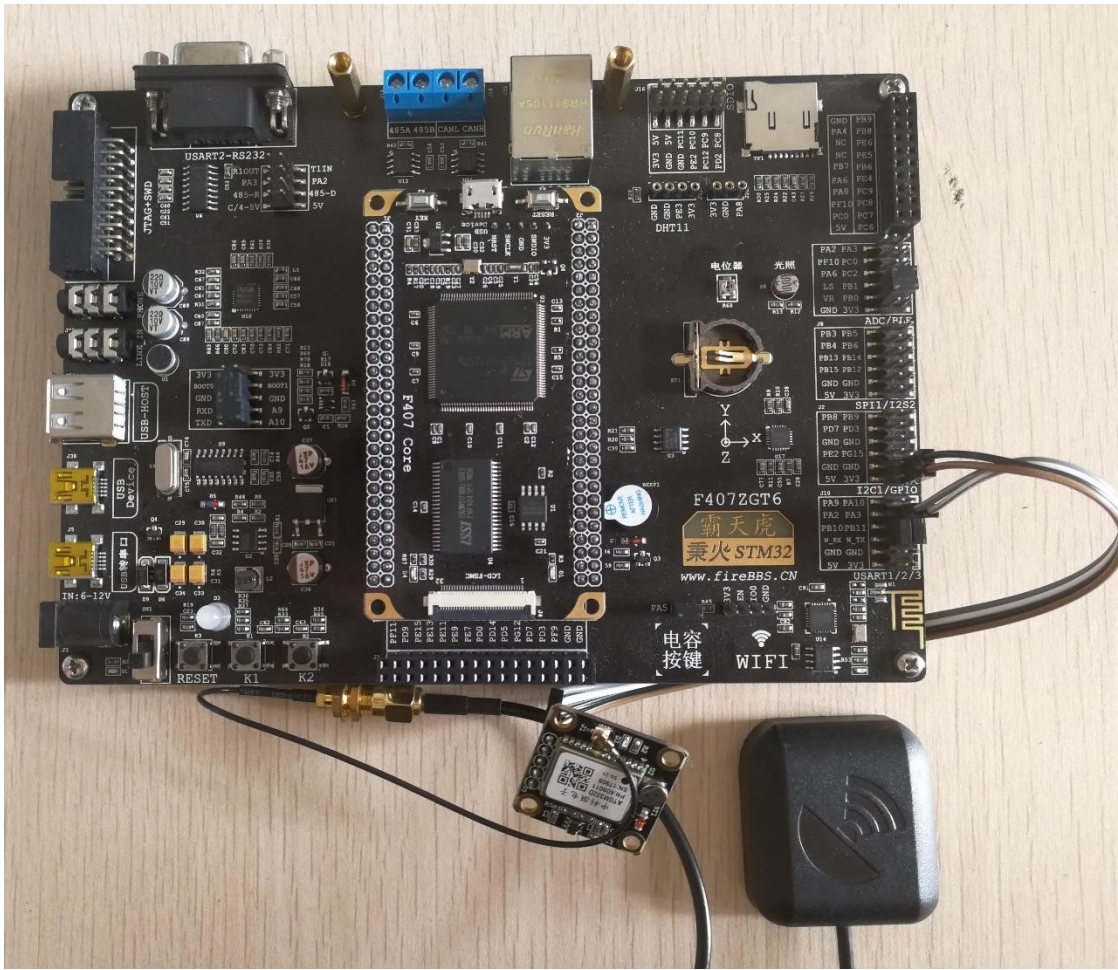


图 6-3 秉火 F407 霸天虎开发板与 BH-ATGM332D 模块连接

注意：由于 F103 霸道及 F407 霸天虎开发板的 PA3/USART2_RX 及 PA2/USART2_TX 引脚与开发板上的 MAX3232 串口芯片相连，为了防止引脚共用的影响，请把 ISO 板子左上角在表 6-3 中的跳线帽拔掉。

表 6-3 要拔掉的跳线帽

编号	丝印
1	R1OUT <---/---> PA3
2	T1IN <---/---> PA2

另外，在 F103 霸道开发板平台，在使用 BH-ATGM332D 模块的时候不要接入摄像头。

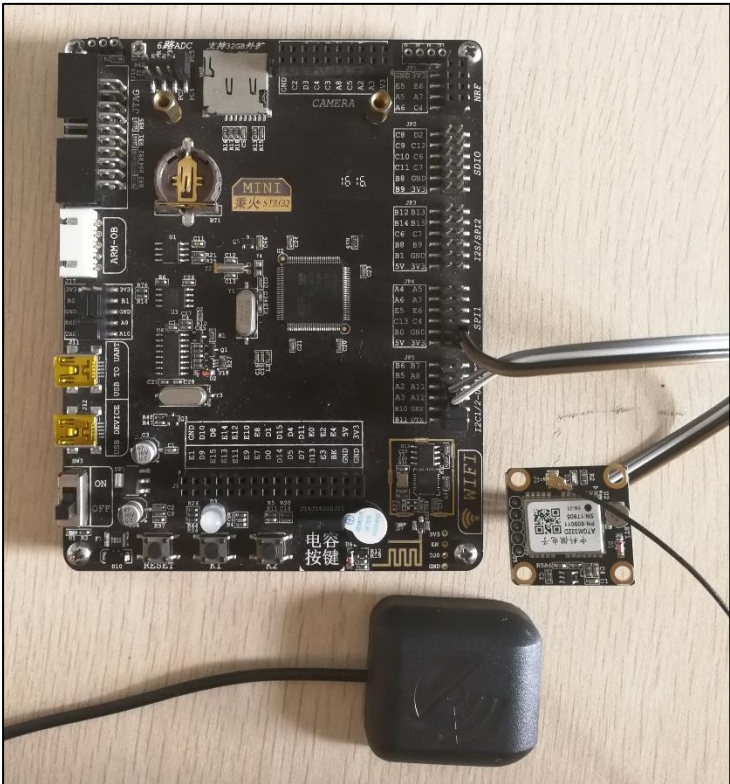


图 6-4 秉火 STM32 指南者开发板与 BH-ATGM332D 模块连接

除了摄像头，F103 指南者板子上的其它外设没有使用到 USART2，直接连接模块即可。

2. 与秉火 F429 挑战者开发板的连接

秉火 F429 板子配套的例程，是通过 STM32 的 **USART3** 外设控制 BH-ATGM332D 模块的，连接引脚说明见表 6-2 及图 6-5。使用时根据该表与开发板引出的相应排针连接即可。

表 6-4 秉火开发板与 BH-ATGM332D 模块连接

F429 挑战者开发板	BH-ATGM332D 模块
5V/3.3V	GPS_5V
GND	GPS_GND
PB10 /USART3_RX	GPS_TXD
PB11 /USART3_TX	GPS_RXD

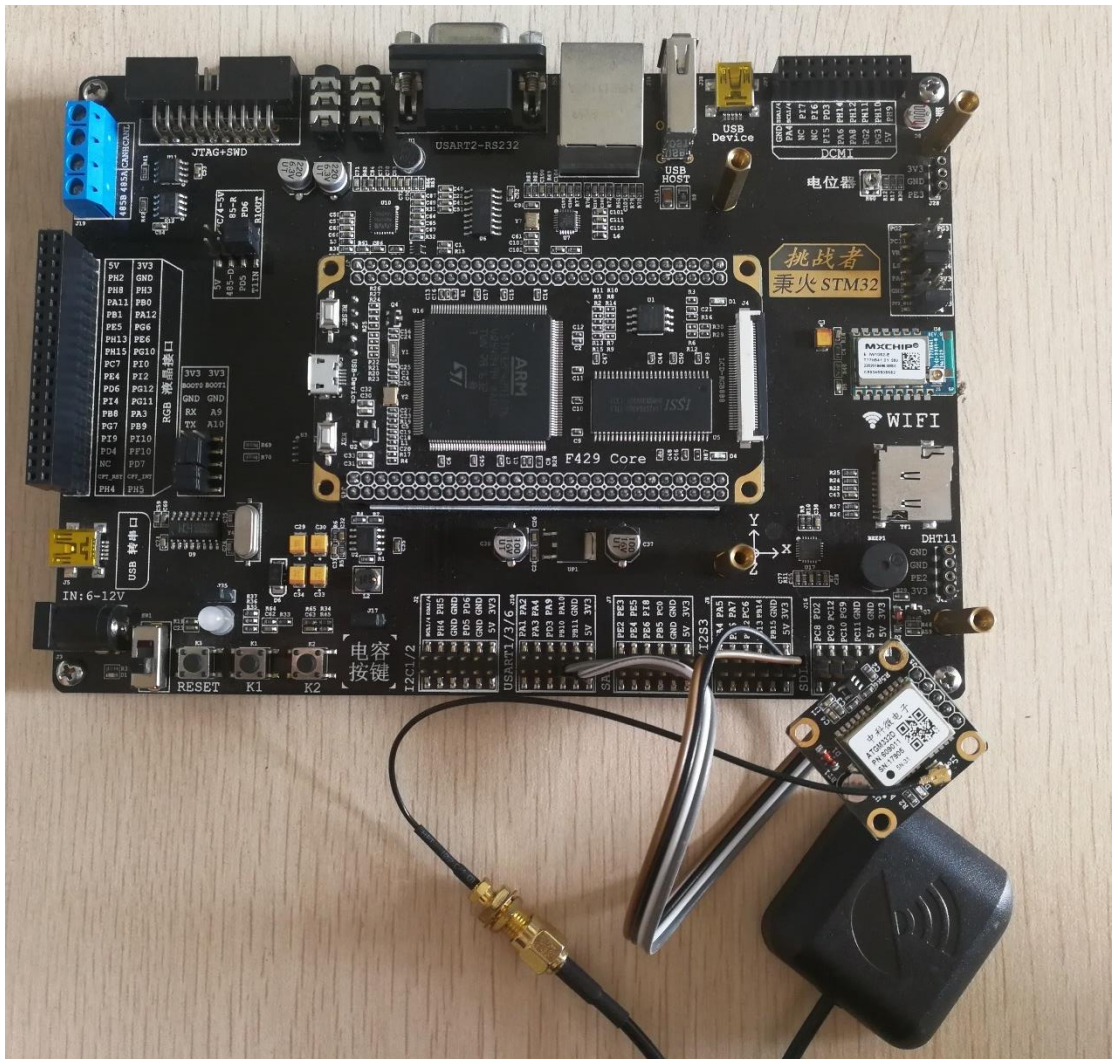


图 6-5 定位模块与 F429 挑战者开发板连接

6.2.2 程序简介

下面以 F103 系列开发板的程序为例进行介绍，F4 的代码类似。

解压秉火 BH-ATGM332D 资料后，在如下路径可以找到配套各个开发板的例程：
ATGM332D\2-开发板配套例程。各个开发板配套例程的功能和定位模块的驱动是基本一致的，只是不同平台使用的引脚、液晶显示部分稍有不同，根据自己使用的开发板，下载对应的程序即可，各平台下配套的代码见图 6-6 及表 6-5。

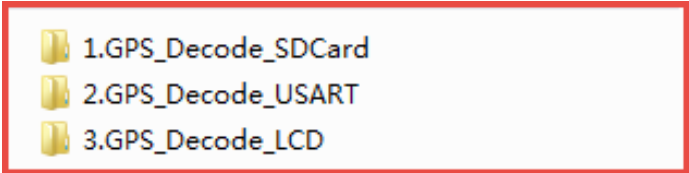


图 6-6 秉火各开发板配套 BH-ATGM332D 例程



表 6-5 秉火 BH-ATGM332D 模块配套例程说明

例程名称	功能简介	使用方法
1.GPS_Decode_SDCard	使用 STM32 开发板对 SD 卡内的 GPS 日志文件进行解码，并把结果通过 usart1 输出到电脑的串口调试助手上。	不需要 GPS 模块，把程序目录下的 gpslog.txt 文件复制到 SD 卡的根目录，并把该 SD 卡接入开发板，然后在电脑端使用串口调试助手(115200-N-8-1)可接收开发板返回的 GPS 解码结果。
2.GPS_Decode_USART	使用 STM32 开发板通过与 GPS 模块连接的 usart 接口接收 GPS 模块输出的 NMEA 信息，并把结果通过 usart1 输出到电脑的串口调试助手上。	不需要 SD 卡，把 BH-ATGM332D 模块按说明接入 STM32 开发板。然后使用串口调试助手可接收(115200-N-8-1)可接收开发板返回的 GPS 解码结果，在信号良好的情况下，会输出准确的时间、经纬度等信息。
3.GPS_Decode_LCD	使用 STM32 开发板通过与 GPS 模块连接的 usart 接口接收 GPS 模块输出的 NMEA 信息，把结果输出到板载的液晶上。	使用方法同上，信号良好时，板载的液晶屏会输出准确的时间、经纬度等信息。

6.2.3 实验现象

1) GPS_Decode_SDCard 实验

本程序对板子上 SD 卡的 gpslog.txt 文件进行解码(请确保卡内有该文件)，实验时把 USB 线接入开发板的 USB TO UART 接口可接收开发板对 GPS 日志的解码信息，见图 6-7，串口调试助手显示了接收到的解码信息。



图 6-7 GPS_Decode_SDCard 实验串口调试助手输出的解码结果

2) GPS_Decode_USART 实验

本程序需要把定位模块接入到开发板对应的串口引脚上，卫星信号良好时，串口调试助手会显示出 STM32 解码后的实时定位信息，见图 6-8。



图 6-8 GPS_Decode_USART 实验串口调试助手提示信息

3) GPS_Decode_LCD 实验

本程序同上，需要把 GPS 模块接入到开发板上，GPS 信号良好时，串口调试助手及板载的液晶屏会显示出实时的定位信息，见图 6-9。

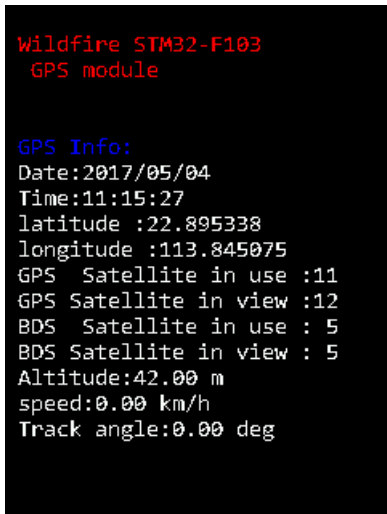


图 6-9GPS_Decode_LCD 实验屏幕截图



7. 代码分析

在本小节中我们将分析如何使用 NMEA 库解码 GPS 数据信息，秉火提供的 GPS_Decode_SDCard 及 GPS_Decode_USART 这两个例程区别在于 GPS 数据信息的来源，前者从 SD 卡文件中获取，后者从 GPS 通过串口模块获取，而它们获取信息后的解码过程都是类似的。

7.1 GPS_Decode_SDCard 例程

7.1.1 实验描述及工程文件清单

1. 实验描述

从板载 SD 卡的 gpslog.txt 文件加载 GPS 数据信息，使用 NMEA 库解码，并把解码结果通过 usart1 输出。

2. 主要工程文件

工程名称	GPS_Decode_SDCard
NMEA 库文件	nmealib/context nmealib/generate nmealib/generator nmealib/gmath nmealib/info nmealib/parse nmealib/parser nmealib/sentence nmealib/time nmealib/tok
用户编写的文件	USER/main USER/gps_config USER/nmea_decode_test

7.1.2 解码流程

以下按照程序的执行流程，从 main 文件开始分析，见代码清单 7-1。

代码清单 7-1 GPS_Decode_SDCard main 文件

```
1 #include "stm32f10x.h"
2 #include "../usart/bsp_usart.h"
3 #include "../led/bsp_led.h"
4 #include "../gps/gps_config.h"
5
6 extern void nmea_decode_test(void);
7
8 /**
9  * @brief 主函数
10  * @param 无
```




```
11  * @retval 无
12  */
13  int main(void)
14  {
15      /* LED 端口初始化 */
16      LED_GPIO_Config();
17
18      LED_BLUE;
19
20      /*串口初始化*/
21      USART_Config();
22
23      GPS_Config();
24
25      printf("\r\n 秉火 GPS 模块测试例程\r\n");
26
27      printf("\r\n 本程序对 SD 卡内的 gpslog.txt 文件定位日志数据进行解码\r\n");
28      printf("\r\n 若需要对 GPS 模块串口传回的数据解码, ");
29      printf("可注释掉 gps_config.h 文件的宏 __GPS_LOG_FILE \r\n");
30
31      /* GPS 解码测试 */
32      nmea_decode_test();
33
34      while (1);
35
36 }
```

main 函数先是用 USART_Config 初始化了调试串口, 便于输出 GPS 解码结果, 使用 GPS_Config 初始化与 GPS 模块连接的串口, 然后就开始执行函数 nmea_decode_test, 它完成了整个解码流程, nmea_decode_test 函数内容见代码清单 7-2。

代码清单 7-2 nmea_decode_test 函数

```
1
2  /*gps_config.h 文件中的宏*/
3  //定义这个宏, 对 SD 卡上的 gpslog.txt 文件进行解码;
4  //不定义的话使用串口接收 GPS 信息解码
5  #define __GPS_LOG_FILE
6
7  /*nmea_decode_test.c 文件*/
8  //对 SD 卡上的 gpslog.txt 文件进行解码; (需要在 sd 卡上存放 gpslog.txt 文件)
9  #ifndef __GPS_LOG_FILE
10
11  FATFS fs;
12  FIL log_file;
13  FRESULT res;
14  UINT br, bw; /* File R/W count */
15
16  /**
17   * @brief nmea_decode_test 解码 GPS 文件信息
18   * @param 无
19   * @retval 无
20   */
21  void nmea_decode_test(void)
22  {
23      double deg_lat;//转换成[degree].[degree]格式的纬度
24      double deg_lon;//转换成[degree].[degree]格式的经度
25
26      nmeaINFO info; /*GPS 解码后得到的信息
27      nmeaPARSER parser; /*解码时使用的数据结构
28
29      nmeaTIME beiJingTime; /*北京时间
30
```



```
31     char buff[2048];
32
33     /* 注册盘符 */
34     res = f_mount(&fs,"0:",1);
35
36     if (res != FR_OK) {
37         printf("\r\n! SD 卡挂载文件系统失败。( %d),
38             请给开发板接入 SD 卡\r\n",res);
39     }
40
41     /* 打开记录有 GPS 信息的文件 */
42
43     res = f_open(&log_file,"0:gpslog.txt", FA_OPEN_EXISTING|FA_READ);
44
45     if (!(res == FR_OK)) {
46         printf("\r\n 打开 gpslog.txt 文件失败,
47             请检查 SD 卡的根目录是否存放了 gpslog.txt 文件!\r\n");
48         return ;
49     }
50
51     /* 设置用于输出调试信息的函数 */
52     nmea_property()->trace_func = &trace;
53     nmea_property()->error_func = &error;
54     nmea_property()->info_func = &gps_info;
55
56     /* 初始化 GPS 数据结构 */
57     nmea_zero_INFO(&info);
58     nmea_parser_init(&parser);
59
60     while (!f_eof(&log_file)) {
61
62         f_read(&log_file, &buff[0], 100, &br);
63
64         /* 进行 nmea 格式解码 */
65         nmea_parse(&parser, &buff[0], br, &info);
66
67         /* 对解码后的时间进行转换, 转换成北京时间 */
68         GMTconvert (&info.utc,&beiJingTime,8,1);
69
70         /* 输出解码得到的信息 */
71         printf("\r\n 时间%d-%02d-%02d,%d:%d:%d\r\n",
72             beiJingTime.year+1900, beiJingTime.mon,beiJingTime.day,
73             beiJingTime.hour,beiJingTime.min,beiJingTime.sec);
74
75         //info.lat lon 中的格式为[degree][min].[sec/60],
76         //使用以下函数转换成[degree].[degree]格式
77         deg_lat = nmea_ndeg2degree(info.lat);
78         deg_lon = nmea_ndeg2degree(info.lon);
79
80         printf("\r\n 纬度: %f,经度%f\r\n",deg_lat,deg_lon);
81         printf("\r\n 海拔高度: %f 米 ", info.elv);
82         printf("\r\n 速度: %f km/h ", info.speed);
83         printf("\r\n 航向: %f 度", info.direction);
84
85         printf("\r\n 正在使用的 GPS 卫星: %d,可见 GPS 卫星: %d",
86             info.satinfo.inuse,info.satinfo.inview);
87
88         printf("\r\n 正在使用的北斗卫星: %d,可见北斗卫星: %d",
89             info.BDsatinfo.inuse,info.BDsatinfo.inview);
90         printf("\r\nPDOP: %f,HDOP: %f, VDOP: %f",
91             info.PDOP,info.HDOP,info.VDOP);
92
93     }
94 }
```




```
95
96     f_lseek(&log_file, f_size(&log_file));
97
98     /* 释放 GPS 数据结构 */
99     nmea_parser_destroy(&parser);
100
101     /* 关闭文件 */
102     f_close(&log_file);
103 }
104 #else          //对 GPS 模块传回的信息进行解码
105 /*此处省略...*/
106 #endif
```

nmea_decode_test.c 文件的主体使用了 “#if...#else...#endif”语句把代码分成了两部分，通过 gps_config.h 头文件中的宏 “__GPS_LOG_FILE”来区分使用哪一部分代码，当定义了该宏时，nmea_decode_test 函数对 SD 卡内的日志文件进行解码，若注释了该宏的定义，则 nmea_decode_test 函数对串口接收到的 GPS 实时信息解码，本资料提供的三个工程都支持使用这个宏切换工作方式，默认情况下 GPS_Decode_SDCard 工程定义了该宏，其余两个工程注释了该宏。

从第 60 行开始的 while 循环是本函数中最最重要的结构，每次循环开始前检查是否已到文件尾，没到文件尾即调用 f_read 函数读取 GPS 日志文件的内容，紧接着调用 NMEA 库函数 nmea_parse 进行解码，解码的结果存放在数据结构变量 info 中，由于解码结果得到的时间信息是格林威治时间，所以在输出解码结果前，调用了 GMTconvert 函数把它转化成北京时间。是的，这样就完成了 GPS 信息的解码，十分简单，关于 nmea_parse 函数在这里不再分析，有兴趣的读者可以自行阅读其源码。

见代码中的 75-80 行，代码使用了 nmea_ndeg2degree 函数把 info.lat 及 info.lon 参数转化到了 deg_lat 和 deg_lon 变量中。info.lat 及 info.lon 存储的就是纬度、经度信息，但它们的单位是[degree][min].[sec/60]格式，即 NMEA 语句解码后的原始 ddmm.mmmmm 表示的数据，通常在地图上使用的格式都是[degree].[degree]，所以一定要经过这样转换后再使用。

7.1.3 结构体 nmeaPARSER 和 nmeaINFO

代码清单 7-2 的第 65 行中，在调用 nmea_parse 函数解码时，输入了四个参数，其说明见表 7-1。

表 7-1 nmea_parse 的输入参数

变量名称	变量类型	作用
parser	nmeaPARSER	解码时使用的缓冲区
buff	char 型数组	将要解码的 GPS 原始数据
br	int	buff 的大小
info	nmeaINFO	存储解码结果

其中的 buff 及 br 参数都很容易获取，当使用文件系统函数 f_read 从文件中读取数据时，GPS 原始数据就存储在 buff 中，且读取到的 buff 大小为 br。

1. nmeaPARSER

上述参数中的 parser 及 info 变量的数据类型 nmeaPARSER 和 nmeaINFO 则是 NMEA 解码库特有的数据结构。其中 nmeaPARSER 的定义见代码清单 7-3。

代码清单 7-3 nmeaPARSER 数据结构定义



```

1 typedef struct _nmeaPARSER {
2     void *top_node;
3     void *end_node;
4     unsigned char *buffer;
5     int buff_size;
6     int buff_use;
7 }
8 } nmeaPARSER;

```

可以看到, nmeaPARSER 是一个链表, 在解码时, NMEA 库会把输入的 GPS 原始数据压入到 nmeaPARSER 结构的链表中, 便于对数据管理及解码。在使用该结构前, 需要调用了 nmea_parser_init 函数分配动态空间, 而解码结束时, 调用了 nmea_parser_destroy 函数释放分配的空间, 见代码清单 7-3 的第 58 和 99 行。

2. nmeaINFO

NMEA 解码库良好的封装特性使我们无需关注更深入的内部实现, 只需要再了解一下 nmeaINFO 数据结构即可, 所有 GPS 解码得到的结果都存储在这个结构中, 其结构体定义见代码清单 7-4。

代码清单 7-4 nmeaINFO 结构体定义

```

1 /**
2  * Summary GPS information from all parsed packets,
3  * used also for generating NMEA stream
4  * @see nmea_parse
5  * @see nmea_GPGGA2info, nmea_...2info
6  */
7 typedef struct _nmeaINFO {
8     int smask; /**< Mask specifying types of packages
9                from which data have been obtained */
10
11     nmeaTIME utc; /**< UTC of position */
12
13     int sig; /**< GPS quality indicator (0 = Invalid;
14             1 = Fix; 2 = Differential, 3 = Sensitive) */
15     int fix; /**< Operating mode, used for navigation
16             (1 = Fix not available; 2 = 2D; 3 = 3D) */
17
18     double PDOP; /**< Position Dilution Of Precision */
19     double HDOP; /**< Horizontal Dilution Of Precision */
20     double VDOP; /**< Vertical Dilution Of Precision */
21
22     double lat; /**< Latitude in NDEG - +/-[degree][min].[sec/60] */
23     double lon; /**< Longitude in NDEG - +/-[degree][min].[sec/60] */
24     double elv; /**< Antenna altitude
25                 above/below mean sea level (geoid) in meters */
26
27     double sog; /**< 数值 对地速度, 单位为节 */
28     double speed; /**< Speed over the ground in kilometers/hour */
29     double direction; /**< Track angle in degrees True */
30     double declination; /**< Magnetic variation degrees
31                         (Easterly var. subtracts from true course) */
32     char mode; /**< 字符 定位模式标志 (A = 自主模式, D = 差分模式,
33             E = 估算模式, N = 数据无效) */
34
35     nmeaSATINFO satinfo; /**< Satellites information */
36     nmeaSATINFO BDsatinfo; /**< 北斗卫星信息 */
37
38     int txt_level;
39     char *txt;
40 } nmeaINFO;

```



表 7-2 nmeaINFO 结构体说明

结构体成员	类型	存储的信息
smask	int	接收到的 GPS 信息包类型
utc	nmeaTIME 结构	包含年、月、日、时、分、秒信息，格林威治时间
sig	int	定位质量：0-无效，1-标准定位，2-差分定位
fix	int	导航模式：1-无效，2-2D，3-3D
PDOP	double	位置精度因子
HDOP	double	水平精度因子
VDOP	double	垂直精度因子
lat	double	纬度，格式为 \pm [度][分].[秒/60] N
lon	double	经度，格式为 \pm [度][分].[秒/60] E
elv	double	海拔，单位：m
speed	double	速度，单位：km/h
direction	double	航向，单位：度
declination	double	磁场方向信息
mode	char	定位模式：A-自主，D-差分，E-估算，N-数据无效
satinfo	nmeaSATINFO 结构	包含了 GSP 可见卫星，正在使用的卫星，卫星信号等信息
BDsatinfo	nmeaSATINFO 结构	包含了北斗可见卫星，正在使用的卫星，卫星信号等信息

在调用了 nmea_parse 函数之后，直接查询 nmeaINFO 结构的数据即可得到解码的结果。

7.1.4 分配堆栈空间

由于 NMEA 解码库在进行解码时需要动态分配较大的空间，所以我们需要在 STM32 的启动文件 startup_stm32f10x_hd.s 文件中对堆栈空间进行修改，本工程中设置的栈空间大小设置为 0x00004000，堆空间大小设置为 0x0000 1000，见代码清单 7-5。

代码清单 7-5 启动文件中对堆空间的配置

```

1 ; Amount of memory (in bytes) allocated for Stack
2 ; Tailor this value to your application needs
3 ; <h> Stack Configuration
4 ;   <o> Stack Size (in Bytes) <0x0-0xFFFFFFFF:8>
5 ; </h>
6
7 Stack_Size      EQU      0x00004000
8
9                 AREA     STACK, NOINIT, READWRITE, ALIGN=3
10 Stack_Mem      SPACE    Stack_Size
11 __initial_sp
12
13 ; <h> Heap Configuration
14 ;   <o> Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
15 ; </h>
16
17 Heap_Size       EQU      0x00001000
18
19                 AREA     HEAP, NOINIT, READWRITE, ALIGN=3
20 __heap_base
21 Heap_Mem        SPACE    Heap_Size
22 __heap_limit
23
24                 PRESERVE8
25                 THUMB

```



7.2 GPS_Decode_USART 例程

1. 实验描述

GPS_Decode_USART 例程使用与定位模块连接的 USART 串口获取 GPS 模块输出的原始信息，并把解码结果使用 USART1 输出。本例程与 GPS_Decode_SDCard 例程直接从 SD 卡获取信息的区别是，在控制器在处理数据的同时，串口会源源不断地接收 GPS 数据，因此需要协调好接收数据和解码数据的关系，秉火例程使用 DMA 串口缓冲区方案，解决了这个问题。

2. 主要工程文件

工程名称	GPS_Decode_USART
NMEA 库文件	nmealib/context nmealib/generate nmealib/generator nmealib/gmath nmealib/info nmealib/parse nmealib/parser nmealib/sentence nmealib/time nmealib/tok
用户编写的文件	USER/main USER/gps_config USER/nmea_decode_test

7.2.2 配置 DMA 串口

先来阅读 GPS_Decode_USART 例程的 main 文件，见代码清单 7-6，它与 GPS_Decode_SDCard 例程区别在代码的第 23 行，相对增加了 GPS_Config 函数，它对与定位模块连接的 USART 串口进行初始化，以便于接收 GPS 模块的信息。

代码清单 7-6 GPS_Decode_USART 例程 main 文件

```
1 #include "stm32f10x.h"
2 #include "../usart/bsp_usart.h"
3 #include "../led/bsp_led.h"
4 #include "../gps/gps_config.h"
5
6 extern void nmea_decode_test(void);
7
8 /**
9  * @brief 主函数
10  * @param 无
11  * @retval 无
12  */
13 int main(void)
14 {
15     /* LED 端口初始化 */
16     LED_GPIO_Config();
17
18     LED_BLUE;
19
20     /*串口初始化*/
```



```
21     USART_Config();
22
23     GPS_Config();
24
25     printf("\r\n 秉火 GPS 模块测试例程\r\n");
26
27     printf("\r\n 本程序对 GPS 模块串口传回的数据解码, ");
28     printf("实验时请给开发板接入 GPS 模块 \r\n");
29
30     /* GPS 解码测试 */
31     nmea_decode_test();
32
33     while (1);
34 }
```

而 GPS_Config 函数又分别调用了 GPS_USART_INIT 和 GPS_DMA_Config 函数, 分别初始化了串口及串口配套的 DMA 模式。

代码清单 7-7 GPS_Config 函数

```
1 /**
2  * @brief  GPS_Config gps 初始化
3  * @param  无
4  * @retval 无
5  */
6 void GPS_Config(void)
7 {
8     GPS_USART_INIT();
9     GPS_DMA_Config();
10
11 }
```

其中的 GPS_USART_INIT 函数主要是对 stm32 与定位模块连接的 USART 串口外设作了基本的初始化, 除了要注意把波特率配置为 9600, 其它跟普通串口配置无异。本例程重点在串口 DMA 的配置, GPS_DMA_Config 函数定义见代码清单 7-8。

代码清单 7-8 GPS_DMA_Config 函数

```
1
2 #define GPS_DR_Base          (USART2_BASE+0x04)    // 串口的数据寄存器地址
3 #define GPS_DATA_ADDR        GPS_DR_Base //GPS 使用的串口的数据寄存器地址
4 #define GPS_RBUFF_SIZE       512                //串口接收缓冲区大小
5 #define HALF_GPS_RBUFF_SIZE  (GPS_RBUFF_SIZE/2)   //串口接收缓冲区一半
6
7 /*****
8 #define GPS_DMA                DMA1
9 #define GPS_DMA_CLK            RCC_AHBPeriph_DMA1
10 #define GPS_DMA_CHANNEL        DMA1_Channel6
11 #define GPS_DMA_IRQn           DMA1_Channel6_IRQn    //GPS 中断源
12
13 /* 外设标志 */
14 #define GPS_DMA_FLAG_TC        DMA1_FLAG_TC6
15 #define GPS_DMA_FLAG_TE        DMA1_FLAG_TE6
16 #define GPS_DMA_FLAG_HT        DMA1_FLAG_HT6
17 #define GPS_DMA_FLAG_GL        DMA1_FLAG_GL6
18 #define GPS_DMA_IT_HT          DMA1_IT_HT6
19 #define GPS_DMA_IT_TC          DMA1_IT_TC6
20
21 /* 中断函数 */ //GPS 使用的 DMA 中断服务函数
22 #define GPS_DMA_IRQHANDLER     DMA1_Channel6_IRQHandler
23
24 /**
25  * @brief  GPS_DMA_Config gps dma 接收配置
26  * @param  无
```




```
27  * @retval 无
28  */
29 static void GPS_DMA_Config(void)
30 {
31     DMA_InitTypeDef DMA_InitStructure;
32
33     /*开启 DMA 时钟*/
34     RCC_AHBPeriphClockCmd(GPS_DMA_CLK, ENABLE);
35
36     /*设置 DMA 源: 串口数据寄存器地址*/
37     DMA_InitStructure.DMA_PeripheralBaseAddr = GPS_DATA_ADDR;
38
39     /*内存地址(要传输的变量的指针)*/
40     DMA_InitStructure.DMA_MemoryBaseAddr = (u32)gps_rbuff;
41
42     /*方向: 从内存到外设*/
43     DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
44
45     /*传输大小 DMA_BufferSize=SENDERBUFF_SIZE*/
46     DMA_InitStructure.DMA_BufferSize = GPS_RBUFF_SIZE;
47
48     /*外设地址不增*/
49     DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
50
51     /*内存地址自增*/
52     DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
53
54     /*外设数据单位*/
55     DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte;
56
57     /*内存数据单位 8bit*/
58     DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
59
60     /*DMA 模式: 不断循环*/
61     DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
62
63     /*优先级: 中*/
64     DMA_InitStructure.DMA_Priority = DMA_Priority_Medium;
65
66     /*禁止内存到内存的传输 */
67     DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
68
69     /*配置 DMA 的通道*/
70     DMA_Init(GPS_DMA_CHANNEL, &DMA_InitStructure);
71
72     GPS_Interrupt_Config();
73     /*配置 DMA 发送完成后产生中断
74     DMA_ITConfig(GPS_DMA_CHANNEL, DMA_IT_HT|DMA_IT_TC, ENABLE);
75
76     /*使能 DMA*/
77     DMA_Cmd (GPS_DMA_CHANNEL, ENABLE);
78
79     /* 配置串口 向 DMA 发出 TX 请求 */
80     USART_DMACmd(GPS_USART, USART_DMAReq_Rx, ENABLE);
81 }
```

本函数中使用到比较多的宏, 部分定义见**错误!未找到引用源。**。GPS_DMA_Config 函数主要工作如下: 设置了外设地址为 USART 的数据寄存器, 并把数据传输方向设置为从 USART 数据寄存器传输到内存变量 gps_rbuff 中, 该缓冲区数组大小为 512 字节。最关键的位置是第 74 行, 它设置了 DMA 半传输结束中断及全传输结束中断, 所以它实际把缓冲区分为成了大小相等的 A/B 两部分, 每次 DMA 接收了半个缓冲区大小的数据时(本程序为 256 字节), 就会引起中断。



得益于这个机制,可以设计程序当 DMA 使用缓冲区 A 存储数据时,控制 CPU 使用 B 中的数据进行 GPS 解码,当 DMA 使用 B 存储时,控制 CPU 使用 A 进行解码,只要缓冲区的大小设置合适,即可避免前面说到的数据丢失问题,这种处理方式也称“乒乓缓冲”,得名于它像打乒乓球一样,你来我往。

当 DMA 的半传输中断或全传输中断产生时,进入的中断服务函数调用了 GPS_ProcessDMAIRQ 函数,见代码清单 7-9。

代码清单 7-9 GPS_ProcessDMAIRQ 函数

```
1 /**
2  * @brief GPS_ProcessDMAIRQ GPS DMA 中断服务函数
3  * @param None.
4  * @retval None.
5  */
6 void GPS_ProcessDMAIRQ(void)
7 {
8
9     if (DMA_GetITStatus(GPS_DMA_IT_HT) ) {          /* DMA 半传输完成 */
10         GPS_HalfTransferEnd = 1;                    //设置半传输完成标志位
11         DMA_ClearFlag(GPS_DMA_FLAG_HT);
12     } else if (DMA_GetITStatus(GPS_DMA_IT_TC)) { /* DMA 传输完成 */
13         GPS_TransferEnd = 1;                        //设置传输完成标志位
14         DMA_ClearFlag(GPS_DMA_FLAG_TC);
15
16     }
17 }
```

在这个函数处理中,主要是在半传输和全传输结束引起中断时,对 GPS_HalfTransferEnd 和 GPS_TransferEnd 标志位进行标记,在解码流程中根据这两个标志使用不同的缓冲区进行处理,处理过程见代码清单 7-10。

代码清单 7-10 nmea_decode_test 函数

```
1 /*gps_config.h 文件*/
2 //定义这个宏,对 SD 卡上的 gpslog.txt 文件进行解码;
3 //不定义的话使用串口接收 GPS 信息解码
4 //define __GPS_LOG_FILE
5
6 /*nmea_decode_test.c 文件*/
7 //对 SD 卡上的 gpslog.txt 文件进行解码;
8 //(需要在 sd 卡上存放 gpslog.txt 文件)
9 #ifndef __GPS_LOG_FILE
10 /*此处省略 SD 卡日志文件解码部分...*/
11 #else //对 GPS 模块传回的信息进行解码
12
13 /**
14  * @brief nmea_decode_test 解码 GPS 模块信息
15  * @param 无
16  * @retval 无
17  */
18 int nmea_decode_test(void)
19 {
20     double deg_lat;//转换成[degree].[degree]格式的纬度
21     double deg_lon;//转换成[degree].[degree]格式的经度
22
23     nmeaINFO info; //GPS 解码后得到的信息
24     nmeaPARSER parser; //解码时使用的数据结构
25     uint8_t new_parse=0; //是否有新的解码数据标志
26 }
```



```
27     nmeaTIME beiJingTime;    //北京时间
28
29     /* 设置用于输出调试信息的函数 */
30     nmea_property()->trace_func = &trace;
31     nmea_property()->error_func = &error;
32     nmea_property()->info_func = &gps_info;
33
34     /* 初始化 GPS 数据结构 */
35     nmea_zero_INFO(&info);
36     nmea_parser_init(&parser);
37
38     while (1) {
39         if (GPS_HalfTransferEnd) { /* 接收到 GPS_RBUFF_SIZE 一半的数据 */
40             /* 进行 nmea 格式解码 */
41             nmea_parse(&parser, (const char*)&gps_rbuff[0],
42                       HALF_GPS_RBUFF_SIZE, &info);
43
44             GPS_HalfTransferEnd = 0;    //清空标志位
45             new_parse = 1;              //设置解码消息标志
46         } else if (GPS_TransferEnd) { /* 接收到另一半数据 */
47             nmea_parse(&parser, (const char*)&gps_rbuff[HALF_GPS_RBUFF_SIZE],
48                       HALF_GPS_RBUFF_SIZE, &info);
49
50             GPS_TransferEnd = 0;
51             new_parse = 1;
52         }
53
54         if (new_parse) {                //有新的解码消息
55             /* 对解码后的时间进行转换, 转换成北京时间 */
56             GMTconvert(&info.utctime, &beiJingTime, 8, 1);
57
58             /* 输出解码得到的信息 */
59             printf("\r\n 时间%d-%02d-%02d, %d:%d:%d\r\n",
60                   beiJingTime.year+1900, beiJingTime.mon, beiJingTime.day,
61                   beiJingTime.hour, beiJingTime.min, beiJingTime.sec);
62
63             //info.lat lon 中的格式为[degree][min].[sec/60],
64             //使用以下函数转换成[degree].[degree] 格式
65             deg_lat = nmea_ndeg2degree(info.lat);
66             deg_lon = nmea_ndeg2degree(info.lon);
67
68             printf("\r\n 纬度: %f, 经度%f\r\n", deg_lat, deg_lon);
69             printf("\r\n 海拔高度: %f 米 ", info.elv);
70             printf("\r\n 速度: %f km/h ", info.speed);
71             printf("\r\n 航向: %f 度", info.direction);
72
73             printf("\r\n 正在使用的 GPS 卫星: %d, 可见 GPS 卫星: %d",
74                   info.satinfo.inuse, info.satinfo.inview);
75
76             printf("\r\n 正在使用的北斗卫星: %d, 可见北斗卫星: %d",
77                   info.BDsatinfo.inuse, info.BDsatinfo.inview);
78             printf("\r\n PDOP: %f, HDOP: %f, VDOP: %f",
79                   info.PDOP, info.HDOP, info.VDOP);
80
81             new_parse = 0;
82         }
83     }
84 }
85
86 /* 释放 GPS 数据结构 */
87 // nmea_parser_destroy(&parser);
88 // return 0;
89 }
```



91 #endif

与 GPS_Decode_SDCard 函数的处理过程类似, 只是输入数据的方式有点区别, 它根据标志选择了不同缓冲区的数据进行解码, 解码后结果使用 USART1 输出。

解码串口接收到的实时信息时, 同样需要注意前面提到的经纬度单位格式。

8. 修改模块配置

BH-ATGM332D 模块支持多种配置, 如使用不同的波特率、定位更新频率以及设置只输出某种类型的 NMEA 语句等功能。

设置时需要使用官方配套的上位机, 资料目录: 3-配套软件\官方配套上位机 GNSSToolKitLite, 在该目录下还带有该上位机的使用文档《GNSSToolKit_Lite 简装版程序说明》, 直接按该文档说明操作即可。

9. 常见问题

1. Q:为什么把解码得到的经纬度数据输入到地图上定位有较大的偏差?

答: 因为坐标系的不同, 国内的电子地图采用“火星坐标系”, 而模块输出的是标准的 WGS-84 坐标系, 这并不是模块定位不精准, 解决方案见 4.3 小节。
2. Q:使用电脑上位机测试时, 为什么没有看到原始数据?

答: 请检查模块与 USB 转串口 TTL 线的连接, 特别注意串口线与模块的 TXD 与 RXD 要交叉相连 (即 TXD<-->RXD、RXD<-->TXD), 另外, 确认电脑上位机打开的 COM 口号与 USB 转 TTL 线对应, 以及波特率要求为 9600。
3. Q:使用电脑上位机测试时, 有原始数据, 但过了很长时间都没有定位成功?

答: 确认模块的天线连接良好, 可根据 4.2 中说明的 GPTXT 语句确认天线连接; 确认天线处在卫星信号良好的位置中, 天线需要正面朝上, 并置于室外 (窗台或阳台边亦可) 无遮挡物处。若把天线置于室内, 是无法定位的, 卫星信号在室内基本无信号 (手机在室内能够定位是因为它还使用了移动及 WiFi 网络辅助, 纯粹使用卫星信号也是定位不了的)
4. Q:运行 STM32 解码 SD 卡日志文件程序时, 为什么没有输出解码信息?

答: 确认调试助手打开的是 STM32 开发板调试用的 COM 口号及波特率为 115200, 确认开发板上已接入 SD 卡, 并且 SD 卡根目录下具有工程路径下的 gpslog.txt 日志文件。
5. Q:运行 STM32 解码程序时, 为什么没有输出解码信息? (串口程序没更新解码数据、液晶屏一直在初始界面)



答: 确认调试助手打开的是 STM32 开发板调试用的 COM 口号及波特率为 115200。检查模块与 STM32 对应开发板的硬件连接, 对于 F103 霸道及 F407 霸天虎开发板还要注意跳帽的问题, 见 6.2 小节。

6. Q: 运行 STM32 解码程序时, 为什么没有在上位机地图上标注定位地点?

答: 要使用在上位机标注定位地点的功能, 请直接使用 USB 转串口 TTL 线连接定位模块与电脑使用。STM32 解码程序仅向电脑发送解码结果, 不做标注。



10. 产品更新及售后支持

秉火的产品资料更新会第一时间发布到论坛: <http://www.firebbs.cn>

购买秉火产品请到秉火官方淘宝店铺: <http://fire-stm32.taobao.com>

在学习或使用秉火产品时遇到问题可在论坛发帖子与我们交流。