WF-NEO-6M 用户手册

——野火 GPS 定位模块

修订历史

日期	版本	更新内容
2014/9/25	1.0.0	-

文档说明

本手册旨在帮助用户正确构建 WF-NEO-6M 模块的使用环境，引导用户快速使用该模块。

关于 WF-NEO-6M 模块的特性参数、硬件资源、原理图、机械尺寸等说明请参考《WF-NEO-6M 数据手册》。

关于核心模 WF-NEO-6M 的硬件参数请参考文档《NEO-6_DataSheet_(GPS.G6-HW-09005).pdf》。

关于核心模块 WF-NEO-6M 的寄存器配置及传输协议请参考文档《u-blox6_ReceiverDescriptionProtocolSpec_GPS.G6-SW-10018-C.pdf》。

目录

WF-NEO-6M 用户手册 1
文档说明 2
目录 3
1. 准备开发环境 5
1.1 安装 u-blox 6 GPS Receiver 驱动 5
1.2 野火多功能调试助手 6
1.2.1 软件简介 6
1.2.2 使用方法 6
2. 硬件测试 9
3. NMEA-0183 协议 10
3.1 NMEA-0183 简介 10
3.2 NMEA-0183 常用语句格式说明 10
3.2.1 GPGGA 11
3.2.2 GPGSA 11
3.2.3 GPGSV 12
3.2.4 GPRMC 12
3.2.5 GPVTG 13
3.2.6 GPGLL 13
3.2.7 GPZDA 14
3.3 NMEA 解码库 14
4. 使用单片机系统控制 WF-NEO-6M 模块 15
4.1 通用控制说明 15
4.2 野火 STM32 开发板控制说明 16
4.2.1 连接模块 16
4.2.2 程序简介 17
4.2.3 实验现象 18
5. 代码分析 20
5.1 GPS_Decode_SDCard 例程 20
5.1.1 实验描述及工程文件清单 20
5.1.2 解码流程 20
5.1.3 结构体 nmeaPARSER 和 nmeaINFO 23
5.1.4 分配堆栈空间 25
5.2 GPS_Decode_USART 例程 25
5.2.2 配置 DMA 串口 26
6. u-center 软件 31
6.1 软件简介 31
6.2 软件操作 32
6.2.1 连接 GPS 模块 32
6.2.2 查看可视化信息 32
6.2.3 配置 GPS 模块寄存器 33

7. 产品更新及售后支持.....35

1. 准备开发环境

使用电脑来测试 WF-NEO-6M 模块非常方便，为此，我们首先要准备好软件环境。主要是安装好 u-blox 6 GPS Receiver 驱动(USB 转串口驱动)及野火 GSM 串口调试助手。

1.1 安装 u-blox 6 GPS Receiver 驱动

WF-NEO-6M 模块提供内嵌的 USB 转串口功能，它把 USB 协议转换成串口协议，使得电脑能直接用 USB 线与模块通讯，但在使用前，需要给电脑安装相应的驱动。

在模块配套资料里找到 USB 驱动文件夹，找到 u-blox_GPS_Receiver_drv.zip 压缩包驱动安装包，解压后双击 u-blox_GPS_Receiver_drv_1206.exe.exe 文件即可安装。

安装成功后，使用 Micro USB 线连接电脑与 WF-NEO-6M 模块，模块获得电源后，它的红色时间脉冲指示灯会亮起来，此时在电脑设备管理器界面的 COM 设备中会看到 u-blox 6 GPS Receiver 设备，见图 1-1，图中的设备指示使用的串口号是 COM9，在后面的串口调试助手中，我们根据这里提示来选择对应的串口号。

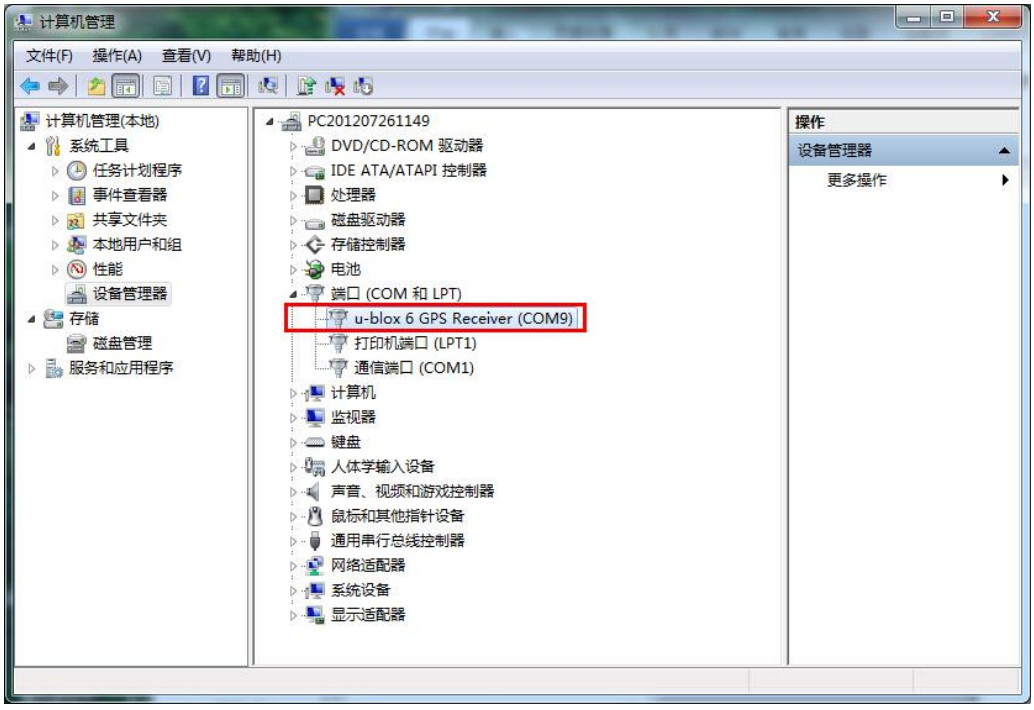

图 1-1 设备管理器中的 u-blox 6 GPS Receiver 设备

1.2 野火多功能调试助手

1.2.1 软件简介

为了方便用户调试及使用 WF-NEO-6M 模块，野火提供了多功能调试助手软件。配合 WF-NEO-6M 模块，该软件能显示 GPS 模块通过 USB 线传输的原始信息，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 GPS 模块变得更为简单直观。

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 1-2。

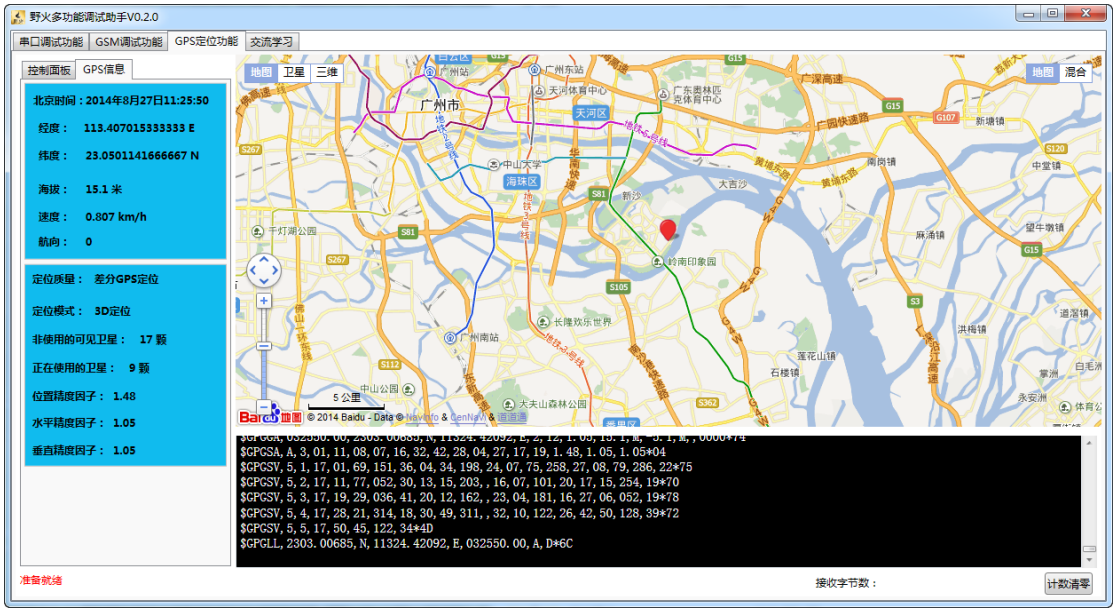

图 1-2 野火多功能调试助手(GPS 界面)

1.2.2 使用方法

在模块配套资料包的“测试软件”文件夹可找到野火多功能调试助手，打开该软件后，选择到 GPS 调试功能界面，把 WF-NEO-6M 模块通过 USB 线连接到电脑，见图 1-3，该软件即可检测到新的 COM 口(若没有检测到 COM 口，请检查 u-blox 6 GPS Receiver 驱动程序是否正确安装)，选择 WF-NEO-6M 模块所用的 COM 口，及默认波特率 9600，然后点击“打开串口”按钮，即可接收到 GPS 模块传回的信息，见图 1-4。

除了解码实时由 GPS 模块传回的信息，本软件还支持对 GPS 日志文件解码。GPS 日志文件即保存了原始 GPS 数据信息的文件，使用本软件从 GPS 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，在野火

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 1-5。

图 1-3 模块通过 USB 线与电脑连接

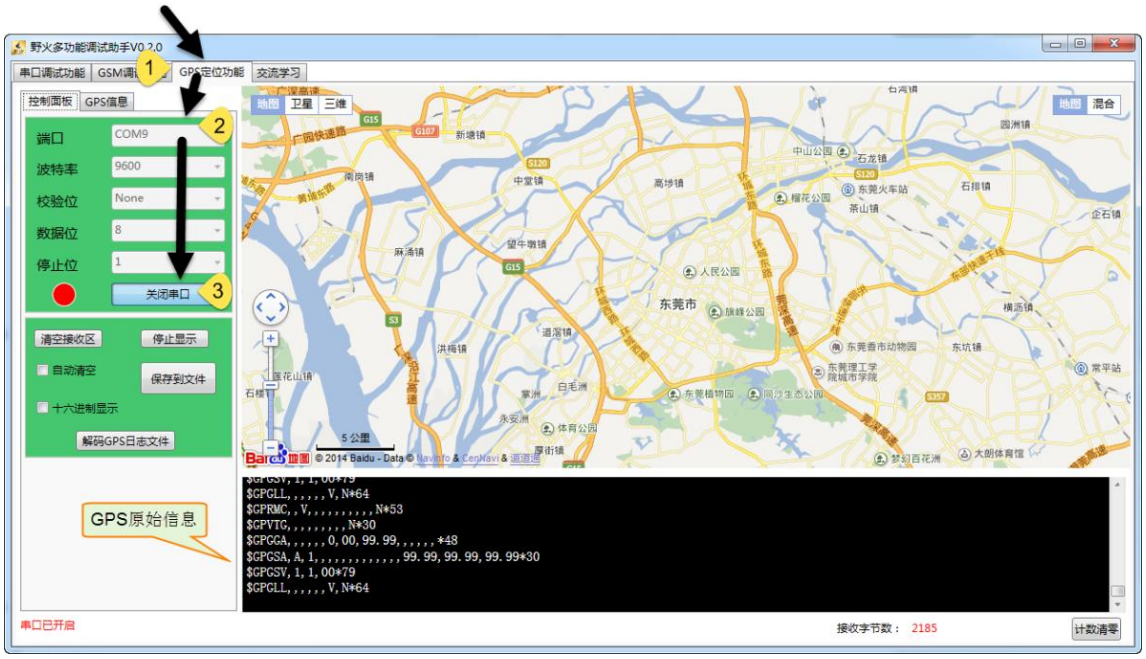

图 1-4 打开 GPS 模块配套的串口

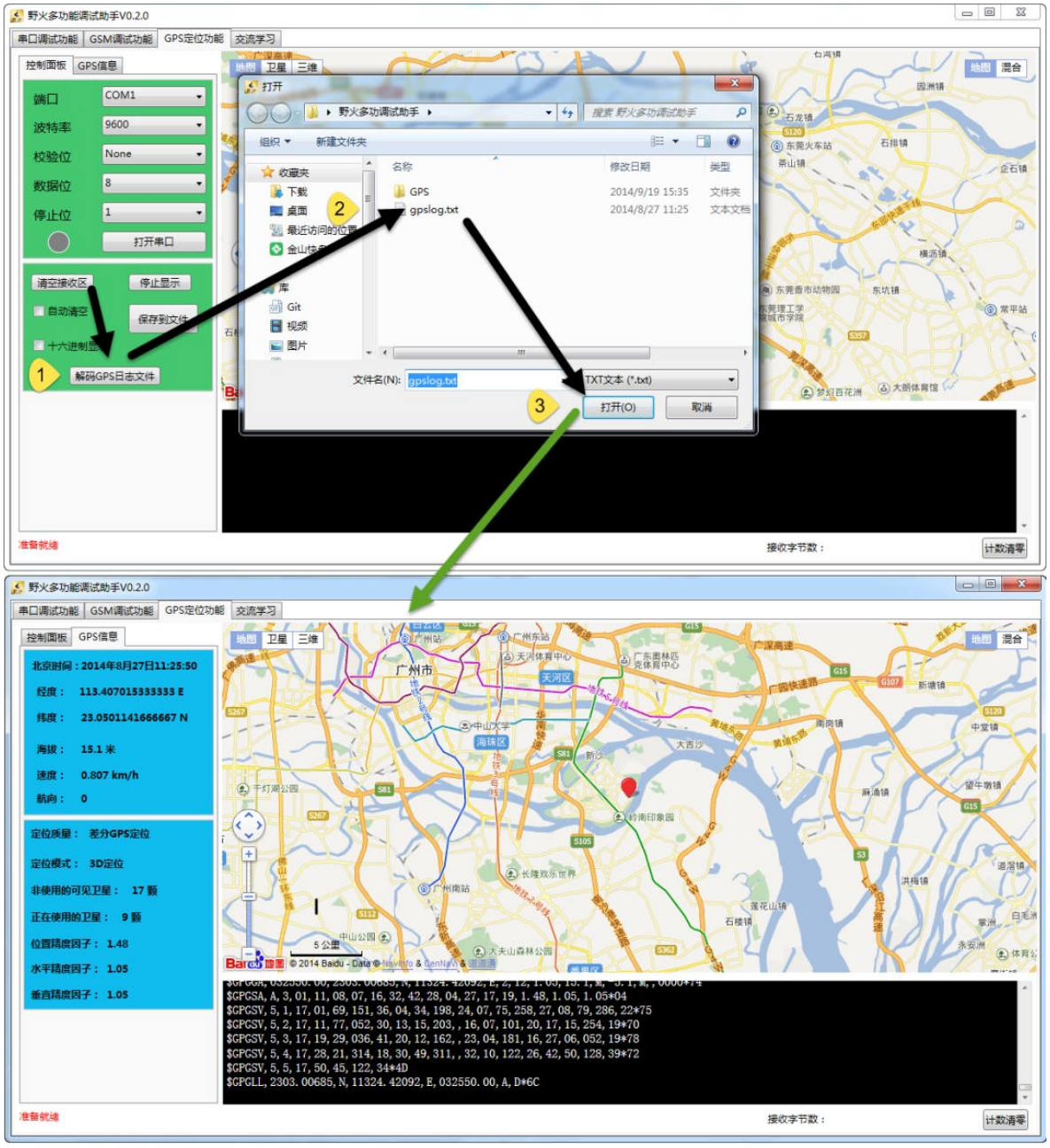

图 1-5 解码 GPS 日志文件

2. 硬件测试

使用野火多功能调试助手可方便地测试 WF-NEO-6M 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 u-blox 6 GPS Receiver 驱动
   - ❑ 使用 GPS 日志文件检验野火多功能调试助手是否正常运行(如不能正常运行，请到野火论坛反馈，我们会持续改进，谢谢支持~)
2) 使用 USB 线连接电脑与 WF-NEO-6M 模块，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 WF-NEO-6M 对应的串口，它的数据输出窗口会输出 GPS 的原始数据(下一小节将解析这些数据格式)。输出的数据一般会出现两种情况，见图 2-1。

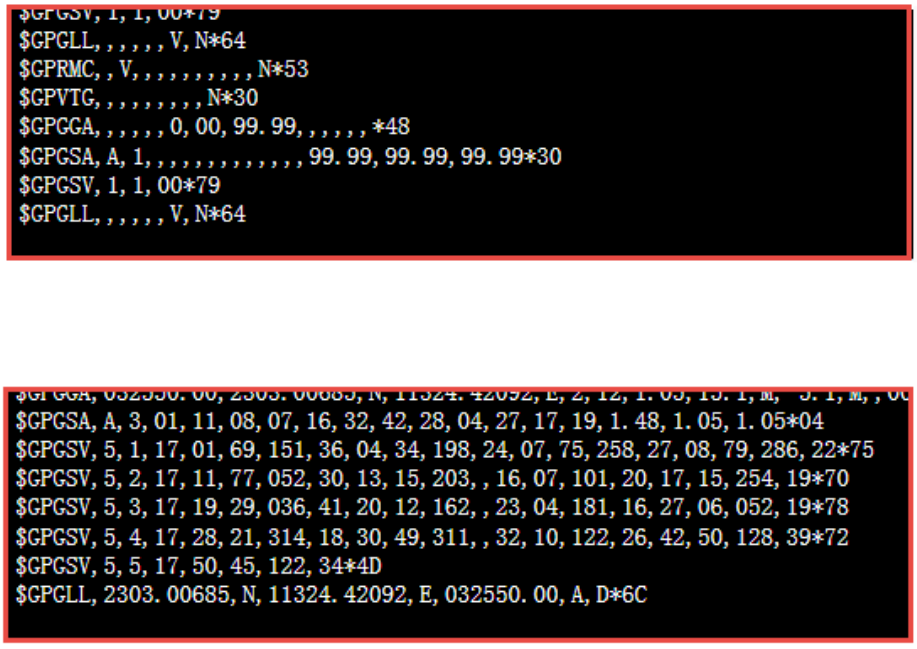

图 2-1 信号好与信号差时的 GPS 数据信息

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 GPS 模块的位置，一般在室内 GPS 信号会比较差，可到室外空旷的地方测试(如楼顶、阳台、窗边)，如果使用了有源天线，则应检查一下有源天线是否接触不良，并把天线接收器置于室外。另外，野火多功能调试助手加载的百度地图是需要在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 GPS 时可把 GPS 数据以文件格式保存起来，在联网的情况下再加载 GPS 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 开头的数据时，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！

3. NMEA-0183 协议

3.1 NMEA-0183 简介

WF-NEO-6M 模块通过串口及 USB 接口输出 GPS 定位数据信息，这些信息默认采用 NMEA-0183 协议，输出的信息形前面的图 2-1 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，NMEA3.0 协议还扩展了北斗导航系统的版本。

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 GPS 接收机和最通用的数据输出格式。

3.2 NMEA-0183 常用语句格式说明

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 $GPGGA、$GPGSA、$GPGSV、$GPRMC、$GPVTG、$GPGLL、$GPZDA 等。下面给出这些常用 NMEA-0183 语句的字段定义解释。

命令	说明	最大帧长
$GPGGA	全球定位数据	72
$GPGSA	卫星 PRN 数据	65
$GPGSV	卫星状态信息	210
$GPRMC	推荐最小数据	70
$GPVTG	地面速度信息	34
$GPGLL	大地坐标信息	
$GPZDA	UTC 时间和日期	

协议帧总说明：

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

<1> “$”——帧命令起始位

<2> aaccc——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

3.2.1 GPGGA

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

格式：

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

例子：$GPGGA,092725.00,4717.11399,N,00833.91590,E,1,8,1.01,499.6,M,48.0,M,,0*5B

<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> 大地椭球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。

3.2.2 GPGSA

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

格式：

$GPGSA,<1>,<2>,<3>,<4>,<5>,<6>,<7>,<8>,<9>,<10>,<11>,<12>,<13>,<14>,<15>,<16>,<17>*<18><CR><LF>

例子：$GPGSA,A,3,23,29,07,08,09,18,26,28,,,,,1.94,1.18,1.54*0D

<1> 模式 2：M = 手动， A = 自动
<2> 模式 1：定位型式 1 = 未定位，2 = 二维定位，3 = 三维定位
<3> 第 1 信道正在使用的卫星 PRN 码编号（Pseudo Random Noise，伪随机噪声码），01 至 32（前导位数不足则补 0，最多可接收 12 颗卫星信息）

<4> 第 2 信道正在使用的卫星 PRN 码编号
<5> 第 3 信道正在使用的卫星 PRN 码编号
<6> 第 4 信道正在使用的卫星 PRN 码编号
<7> 第 5 信道正在使用的卫星 PRN 码编号
<8> 第 6 信道正在使用的卫星 PRN 码编号
<9> 第 7 信道正在使用的卫星 PRN 码编号
<10> 第 8 信道正在使用的卫星 PRN 码编号
<11> 第 9 信道正在使用的卫星 PRN 码编号
<12> 第 10 信道正在使用的卫星 PRN 码编号
<13> 第 11 信道正在使用的卫星 PRN 码编号
<14> 第 12 信道正在使用的卫星 PRN 码编号
<15> PDOP 综合位置精度因子（0.5 - 99.9）
<16> HDOP 水平精度因子（0.5 - 99.9）
<17> VDOP 垂直精度因子（0.5 - 99.9）
<18> 校验和

3.2.3 GPGSV

可视卫星状态输出语句 (GNSS Satellites in View)。
格式：$GPGSV, <1>,<2>,<3>,<4>,<5>,<6>,<7>,...,<4>,<5>,<6>,<7>*<8><CR><LF>
例子：$GPGSV,3,1,10,23,38,230,44,29,71,156,47,07,29,116,41,08,09,081,36*7F

<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，01 至 32
<5> 卫星仰角，00 至 90 度
<6> 卫星方位角，000 至 359 度。实际值
<7> 信噪比（C/No），00 至 99dB；无表示未接收到讯号
<8> 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。

3.2.4 GPRMC

推荐最小数据量的 GPS 信息(Recommended Minimum Specific GPS/TRANSIT Data)。
格式：
$GPRMC,<1>,<2>,<3>,<4>,<5>,<6>,<7>,<8>,<9>,<10>,<11>,<12>*<13><CR><LF>
例子：$GPRMC,083559.00,A,4717.11437,N,00833.91522,E,0.004,77.52,091202,,,A*57

<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> 校验和。

3.2.5 GPVTG

地面速度信息(Course over ground and Ground speed)。
格式：$GPVTG,<1>,<2>,<3>,<4>,<5>,<6>,<7>,<8>,<9>*<10><CR><LF>
例子：$GPVTG,77.52,T,,M,0.004,N,0.008,K,A*06

<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

3.2.6 GPGLL

定位地理信息(Latitude and longitude, with time of position fix and status)
格式：$GPGLL,<1>,<2>,<3>,<4>,<5>,<6>,<7>*<8><CR><LF>
例子：$GPGLL,4717.11364,N,00833.91565,E,092321.00,A,A*60

<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

3.2.7 GPZDA

当前时间信息：(Time and Date)
格式：$GPZDA,<1>,<2>,<3>,<4>,<5>,<6>*<7><CR><LF>
例子：$GPZDA,082710.00,16,09,2002,00,00*64

<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

3.3 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。在野火提供的 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，野火提供的 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。

4. 使用单片机系统控制 WF-NEO-6M 模块

4.1 通用控制说明

WF-NEO-6M 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。本小节以野火 STM32 开发板为例子说明如何使用 STM32 来控制 WF-NEO-6M 模块。

单片机系统通过串口与 WF-NEO-6M 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，见

表 4-1 及图 4-1。

表 4-1 单片机与 WF-NEO-6M 模块连接引脚表

单片机系统	WF-NEO-6M 模块
5V	WF-NEO-6M_VCC
GND	WF-NEO-6M_GND
串口 RXD	WF-NEO-6M_TXD
串口 TXD	WF-NEO-6M_RXD

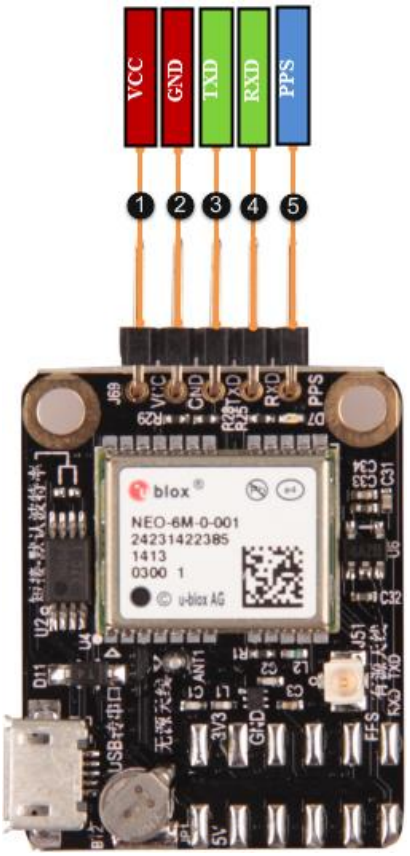

图 4-1 单片机与 WF-NEO-6M 模块连接引脚表

4.2 野火 STM32 开发板控制说明

WF-NEO-6M 模块配套有适用于野火 STM32-ISO 及 STM32-ISO-MINI 开发板的源码，用户可以参考它来编写自己的应用。

4.2.1 连接模块

野火 ISO 及 ISO-MINI 板子配套的例程，都是通过 STM32 的 USART2 外设控制 WF-NEO-6M 模块的，连接引脚说明见表 4-2 、图 4-2 及图 4-3。

表 4-2 野火开发板与 WF-NEO-6M 模块连接

ISO 及 ISO-MINI	(模块引脚编号)WF-SIM900A 模块
5V	WF-NEO-6M_5V
GND	WF-NEO-6M_GND
PA3/USART2_RX	WF-NEO-6M_TXD
PA2/USART2_TX	WF-NEO-6M_RXD

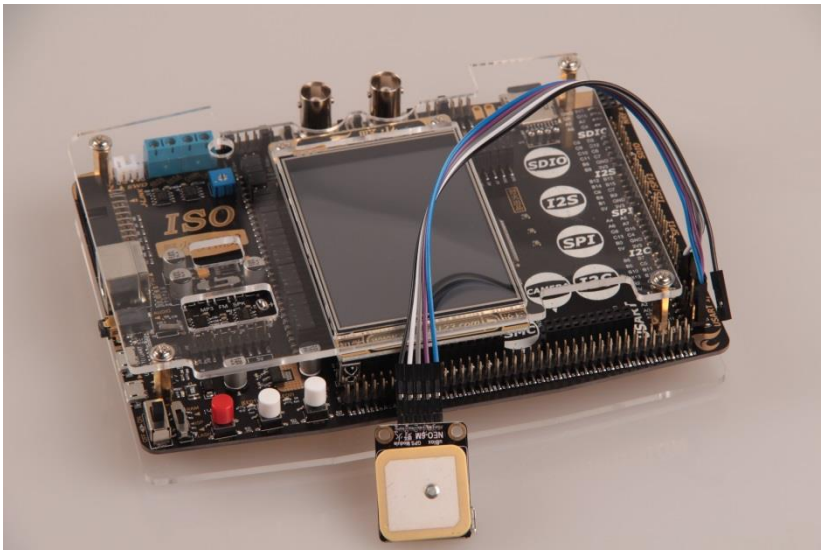

图 4-2 野火 STM32-ISO 开发板与 WF-NEO-6M 模块连接

注意，由于 STM32-ISO 开发板的 PA3/USART2_RX 及 PA2/USART2_TX 引脚与开发板上的 485 芯片相连，为了防止引脚共用的影响，请把 ISO 板子左侧在表 4-3 中的跳线帽拔掉。

表 4-3 要拔掉的跳线帽

编号	丝印
1	CAN/485 5V
4	485R A3
5	485D A2

另外，在使用 WF-NEO-6M 模块的时候不要接入摄像头。

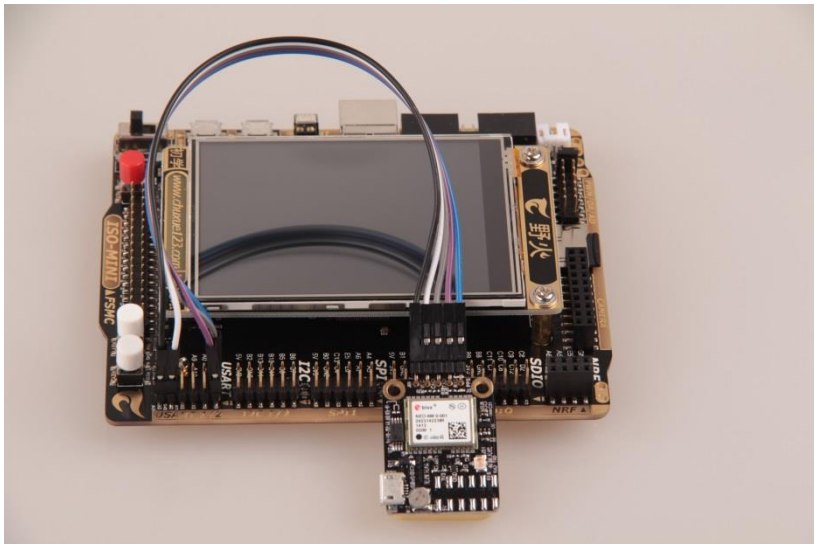

图 4-3 野火 STM32-ISO-MINI 开发板与 WF-NEO-6M 模块连接

除了摄像头，ISO-MINI 板子上的其它外设没有使用到 USART2，直接连接模块即可。

4.2.2 程序简介

解压野火 WF-NEO-6M 资料后，在如下路径可以找到配套野火 ISO 及 ISO-MINI 的例程：野火 GPS 模块资料\STM32 控制代码\ 。ISO 及 ISO-MINI 例程的功能和使用方法是完全一样的，根据自己使用的开发板，下载对应的程序即可，具体说明见图 4-4 及表 4-4。

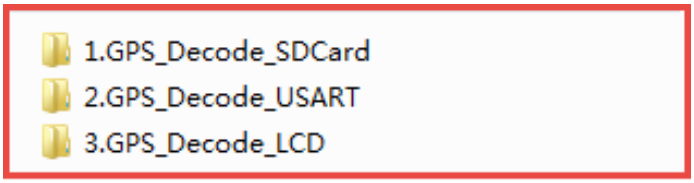

图 4-4 野火 STM32-ISO 及 ISO-MINI 开发板配套 WF-NEO-6M 例程

表 4-4 野火 WF-NEO-6M 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 usart2 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 WF-NEO-6M 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 usart2 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。

注：液晶例程使用了 emWin 软件库，其详细介绍可参考野火教程《emWin 实战指南》。

4.2.3 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 4-5，串口调试助手显示了接收到的解码信息。

图 4-5 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 GPS 模块接入到开发板上，GPS 信号良好时，串口调试助手会显示出准确的定位信息，见图 4-6。

图 4-6 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准确的定位信息，见图 4-7。

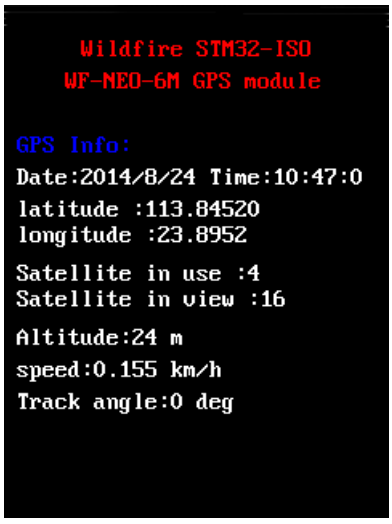

图 4-7GPS_Decode_LCD 实验屏幕截图

5. 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，野火提供的 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。

5.1 GPS_Decode_SDCard 例程

5.1.1 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.c** nmealib /**generate.c** nmealib /**generator.c** nmealib /**gmath.c** nmealib/**info.c** nmealib /**parse.c** nmealib /**parser.c** nmealib /**sentence.c** nmealib /**time.c** nmealib /**tok.c**
用户编写的文件	USER/**main.c** USER/**stm32f10x_it.c** USER/**bsp_usart1.c** USER/**bsp_sdio_sdcard.c** USER/**bsp_usart2.c** USER/**gps_config.c** USER/**nmea_decode_test.c**
Fatfs 文件系统	Fatfs 文件系统 R0.09 源码文件
ST 固件库	所有 ST3.5 版本标准库文件

5.1.2 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 5-1。

代码清单 5-1 GPS_Decode_SDCard main 文件

```
/**
  ******************************************************************
  * @file    main.c
  * @author  fire
  * @version V1.0
  * @date    2014-08-xx
  * @brief   gps 解码演示程序
  ******************************************************************
  * @attention
  *
  * 实验平台:野火 ISO-STM32-MINI 开发板
  * 论坛     :http://www.chuxue123.com
  * 淘宝     :http://firestm32.taobao.com
  *
  ******************************************************************
  */
#include "stm32f10x.h"
#include "bsp_usart1.h"
#include "gps_config.h"
#include "bsp_sdio_sdcard.h"
#include "diskio.h"
#include "ff.h"
#include <string.h>

extern void nmea_decode_test(void);

/*
 * 本工程用于对 SD 卡内的 GPS 日志文件进行解码
 *
 */
int main(void)
{
    /* 配置 USART1 用于向电脑 printf 调试信息*/
    USART1_Config();

    printf("\r\n 野火 WF-NEO-6M GPS 模块测试例程\r\n");

    /* GPS 解码测试 */
    nmea_decode_test();

    while (1);
}
```

main 函数先是初始化了 usart1，便于输出 GPS 解码结果。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 5-2。

代码清单 5-2 nmea_decode_test 函数

```

FATFS fs;
FIL log_file;
FRESULT res;
UINT br, bw;                        /* File R/W count */

/**
  * @brief  nmea_decode_test 解码 GPS 文件信息
  * @param  无
  * @retval 无
  */
void nmea_decode_test(void)
```

```
13 {
14
15     nmeaINFO info;          //GPS 解码后得到的信息
16     nmeaPARSER parser;      //解码时使用的数据结构
17
18     nmeaTIME beiJingTime;    //北京时间
19
20     char buff[2048];
21
22     /* 注册盘符 */
23     f_mount(0,&fs);
24
25     /* 打开记录有 GPS 信息的文件 */
26     res = f_open(&log_file,"0:gpslog.txt", FA_OPEN_EXISTING|FA_READ);
27
28     if (!(res == FR_OK)) {
29         printf("\r\n 打开 gpslog.txt 文件失败，请检查 SD 卡的根目录是否存放了
                                                    gpslog.txt 文件!\r\n");
30         return ;
31     }
32
33     /* 设置用于输出调试信息的函数 */
34     nmea_property()->trace_func = &trace;
35     nmea_property()->error_func = &error;
36
37     /* 初始化 GPS 数据结构 */
38     nmea_zero_INFO(&info);
39     nmea_parser_init(&parser);
40
41     while (!f_eof(&log_file)) {
42         f_read(&log_file, &buff[0], 100, &br);
43
44         /* 进行 nmea 格式解码 */
45         nmea_parse(&parser, &buff[0], br, &info);
46
47         /* 对解码后的时间进行转换，转换成北京时间 */
48         GMTconvert(&info.utc,&beiJingTime,8,1);
49
50         /* 输出解码得到的信息 */
51         printf("\r\n 时间%d,%d,%d,%d,%d,%d\r\n", beiJingTime.year+1900,
                   beiJingTime.mon+1,beiJingTime.day,beiJingTime. hour,
52                                     beiJingTime.min,beiJingTime.sec);
53         printf("\r\n 纬度：%f,经度%f\r\n",info.lat,info.lon);
54         printf("\r\n 正在使用的卫星：%d,可见卫星：%d",info.satinfo.inuse,
                                                    info.satinfo.inview);
55         printf("\r\n 海拔高度：%f 米 ", info.elv);
56         printf("\r\n 速度：%f km/h ", info.speed);
57         printf("\r\n 航向：%f 度", info.direction);
58     }
59
60     f_lseek(&log_file, f_size(&log_file));
61
62     /* 释放 GPS 数据结构 */
63     nmea_parser_destroy(&parser);
64
65     /* 关闭文件 */
66     f_close(&log_file);
67 }
```

从第 41 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。

5.1.3 结构体 nmeaPARSER 和 nmeaINFO

代码清单 5-2 的第 45 行中，在调用 nmea_parse 函数解码时，输入了四个参数，其说明见表 5-1。

表 5-1 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 参数我们都非常熟悉，从文件中读取数据时，GPS 原始数据就存储在 buff 中，且 buff 的大小为 br。

1. nmeaPARSER

而 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 5-3。

代码清单 5-3 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，我们调用了 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 5-3 的第 39 和 63 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 5-4。

代码清单 5-4 nmeaINFO 结构体定义

```
/**
 * Summary GPS information from all parsed packets,
 * used also for generating NMEA stream
 * @see nmea_parse
 * @see nmea_GPGGA2info,  nmea_...2info
 */
typedef struct _nmeaINFO {
    int      smask;      /**< Mask specifying types of packages from which
                                                    data have been obtained */

    nmeaTIME utc;        /**< UTC of position */

    int      sig;        /**< GPS quality indicator (0 = Invalid; 1 = Fix;
                                                2 = Differential, 3 = Sensitive) */
    int      fix;    /**< Operating mode, used for navigation (1 = Fix not
                                                  available; 2 = 2D; 3 = 3D) */

    double   PDOP;   /**< Position Dilution Of Precision */
    double   HDOP;   /**< Horizontal Dilution Of Precision */
    double   VDOP;   /**< Vertical Dilution Of Precision */

    double   lat;    /**< Latitude in NDEG - +/-[degree][min].[sec/60] */
    double   lon;    /**< Longitude in NDEG - +/-[degree][min].[sec/60] */
    double   elv;    /**< Antenna altitude above/below mean sea level
                                                        (geoid) in meters */
    double   speed;       /**< Speed over the ground in kilometers/hour */
    double   direction;   /**< Track angle in degrees True */
    double   declination; /**< Magnetic variation degrees (Easterly var.
                                                  subtracts from true course) */

    nmeaSATINFO satinfo; /**< Satellites information */

} nmeaINFO;
```

表 5-2 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 ±[度][分].[秒/60] N
lon	double	经度，格式为 ±[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
satinfo	nmeaSATINFO 结构	包含了可见卫星，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。

5.1.4 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堆空间，所以我们需要在 STM32 的启动文件 startup_stm32f10x_hd.s 文件中对堆空间进行修改，本工程中设置的堆空间大小设置为 0x0000 1000，见代码清单 5-5。

代码清单 5-5 启动文件中对堆空间的配置

```
; Amount of memory (in bytes) allocated for Stack
; Tailor this value to your application needs
; <h> Stack Configuration
;   <o> Stack Size (in Bytes) <0x0-0xFFFFFFFF:8>
; </h>

Stack_Size      EQU     0x00000400

                AREA    STACK, NOINIT, READWRITE, ALIGN=3
Stack_Mem       SPACE   Stack_Size
__initial_sp

; <h> Heap Configuration
;   <o>  Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
; </h>

Heap_Size       EQU     0x00001000

                AREA    HEAP, NOINIT, READWRITE, ALIGN=3
__heap_base
Heap_Mem        SPACE   Heap_Size
__heap_limit

                PRESERVE8
                THUMB
```

5.2 GPS_Decode_USART 例程

1. 实验描述

GPS_Decode_USART 例程使用 USART2 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，控制器在处理数据的同时，串口源源不断地接收 GPS 数据，此时会造成串口数据丢失，而野火例程使用 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.c** nmealib /**generate.c** nmealib /**generator.c** nmealib /**gmath.c** nmealib/**info.c**

	nmealib /**parse.c** nmealib /**parser.c** nmealib /**sentence.c** nmealib /**time.c** nmealib /**tok.c**
用户编写的文件	USER/**main.c** USER/**stm32f10x_it.c** USER/**bsp_usart1.c** USER/**bsp_sdio_sdcard.c** USER/**bsp_usart2.c** USER/**gps_config.c** USER/**nmea_decode_test.c**
Fatfs 文件系统	Fatfs 文件系统 R0.09 源码文件
ST 固件库	所有 ST3.5 版本标准库文件

5.2.2 配置 DMA 串口

先来阅读 GPS_Decode_USART 例程的 main 文件，见代码清单 5-6，它与 GPS_Decode_SDCard 例程区别在代码的第 37 行，相对增加了 GPS_Config 函数，它对 USART2 接口初始化，以便于接收 GPS 模块的信息。

代码清单 5-6 GPS_Decode_USART 例程 main 文件

```
1  /**
2    ******************************************************************
3    * @file    main.c
4    * @author  fire
5    * @version V1.0
6    * @date    2014-08-xx
7    * @brief   gps 解码演示程序
8    ******************************************************************
9    * @attention
10   *
11   * 实验平台:野火 ISO-STM32-MINI 开发板
12   * 论坛     :http://www.chuxue123.com
13   * 淘宝     :http://firestm32.taobao.com
14   *
15   ******************************************************************
16   */
17 #include "stm32f10x.h"
18 #include "bsp_usart1.h"
19 #include "gps_config.h"
20 #include "bsp_sdio_sdcard.h"
21 #include "diskio.h"
22 #include "ff.h"
23 #include <string.h>
24
25 extern void nmea_decode_test(void);
26
27 /*
28  * 本工程用于对 SD 卡内的 GPS 日志文件进行解码
29  *
30  */
31 int main(void)
32 {
33     /* 配置 USART1 用于向电脑 printf 调试信息*/
34     USART1_Config();
```

```
35
36      /* 初始化 GPS 模块使用的接口 */
37      GPS_Config();
38
39      printf("\r\n 野火 WF-NEO-6M GPS 模块测试例程\r\n");
40
41      /* GPS 解码测试 */
42      nmea_decode_test();
43
44      while (1);
45  }
```

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

代码清单 5-7 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 的 USART2 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 5-8。

代码清单 5-8 GPS_DMA_Config 函数

```
1  /**
2    * @brief  GPS_DMA_Config gps dma 接收配置
3    * @param  无
4    * @retval 无
5    */
6  static void GPS_DMA_Config(void)
7  {
8      DMA_InitTypeDef DMA_InitStructure;
9
10     /*开启 DMA 时钟*/
11     RCC_AHBPeriphClockCmd(GPS_DMA_CLK, ENABLE);
12     /*设置 DMA 源：串口数据寄存器地址*/
13     DMA_InitStructure.DMA_PeripheralBaseAddr = GPS_DATA_ADDR;
14     /*内存地址(要传输的变量的指针)*/
15     DMA_InitStructure.DMA_MemoryBaseAddr = (u32)gps_rbuff;
16     /*方向：从内存到外设*/
17     DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
18     /*传输大小 DMA_BufferSize=SENDBUFF_SIZE*/
19     DMA_InitStructure.DMA_BufferSize = GPS_RBUFF_SIZE;
20     /*外设地址不增*/
21     DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
22     /*内存地址自增*/
```

```
23      DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
24      /*外设数据单位*/
25      DMA_InitStructure.DMA_PeripheralDataSize =
                                              DMA_PeripheralDataSize_Byte;
26      /*内存数据单位 8bit*/
27      DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
28      /*DMA 模式：不断循环*/
29      DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
30      /*优先级：中*/
31      DMA_InitStructure.DMA_Priority = DMA_Priority_Medium;
32      /*禁止内存到内存的传输  */
33      DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
34      /*配置 DMA 的通道*/
35      DMA_Init(GPS_DMA_CHANNEL, &DMA_InitStructure);
36      GPS_Interrupt_Config();
37      DMA_ITConfig(GPS_DMA_CHANNEL,DMA_IT_HT|DMA_IT_TC,ENABLE);   //配置 DMA
                                                              发送完成后产生中断
38      /*使能 DMA*/
39      DMA_Cmd (GPS_DMA_CHANNEL,ENABLE);
40      /* 配置串口 向 DMA 发出 RX 请求 */
41      USART_DMACmd(GPS_USART, USART_DMAReq_Rx, ENABLE);
42  }
```

本函数中使用到比较多的宏，部分定义见代码清单 5-9。GPS_DMA_Config 函数主要工作如下：设置了外设地址为 USART2 的数据寄存器，并把数据传输方向设置为从 USART2 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 37 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。

代码清单 5-9 GPS_DMA_Config 使用到的变量及宏

```
1
2  /* DMA 接收缓冲  */
3  uint8_t gps_rbuff[GPS_RBUFF_SIZE];
4
5  /* GPS 接口配置 使用不同的串口时，要修改对应的接口 */
6
7  #define USART2_DR_Base  (0x40004400+0x04)      // 串口 2 的数据寄存器地址
8
9  #define GPS_DATA_ADDR   USART2_DR_Base         //GPS 使用的串口的数据寄存器地址
10 #define GPS_RBUFF_SIZE            512                    //串口接收缓冲区大小
11 #define HALF_GPS_RBUFF_SIZE       (GPS_RBUFF_SIZE/2)     //串口接收缓冲区一半
```

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 5-10。

代码清单 5-10 GPS_ProcessDMAIRQ 函数

```
/**
  * @brief  GPS_ProcessDMAIRQ GPS DMA 中断服务函数
  * @param  None.
  * @retval None.
  */
void GPS_ProcessDMAIRQ(void)
{

    if (DMA_GetITStatus(GPS_DMA_IT_HT) ) {       /* DMA 半传输完成 */
        GPS_HalfTransferEnd = 1;                 //设置半传输完成标志位
        DMA_ClearFlag(GPS_DMA_FLAG_HT);
    } else if (DMA_GetITStatus(GPS_DMA_IT_TC)) { /* DMA 传输完成 */
        GPS_TransferEnd = 1;                     //设置传输完成标志位
        DMA_ClearFlag(GPS_DMA_FLAG_TC);

    }
}
```

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 5-11。

代码清单 5-11 nmea_decode_test 函数

```
/**
  * @brief  nmea_decode_test 解码 GPS 模块信息
  * @param  无
  * @retval 无
  */
int nmea_decode_test(void)
{

    nmeaINFO info;          //GPS 解码后得到的信息
    nmeaPARSER parser;      //解码时使用的数据结构
    uint8_t new_parse=0;    //是否有新的解码数据标志

    nmeaTIME beiJingTime;    //北京时间

    /* 设置用于输出调试信息的函数 */
    nmea_property()->trace_func = &trace;
    nmea_property()->error_func = &error;

    /* 初始化 GPS 数据结构 */
    nmea_zero_INFO(&info);
    nmea_parser_init(&parser);

    while (1) {
        if (GPS_HalfTransferEnd) {  /* 接收到 GPS_RBUFF_SIZE 一半的数据 */
            /* 进行 nmea 格式解码 */
            nmea_parse(&parser, (const char*)&gps_rbuff[0],
                                    HALF_GPS_RBUFF_SIZE, &info);

            GPS_HalfTransferEnd = 0;    //清空标志位
            new_parse = 1;              //设置解码消息标志
        } else if (GPS_TransferEnd) { /* 接收到另一半数据 */

            nmea_parse(&parser, (const char*)
            &gps_rbuff[HALF_GPS_RBUFF_SIZE], HALF_GPS_RBUFF_SIZE, &info);

            GPS_TransferEnd = 0;
            new_parse =1;
```

```
36          }
37
38          if (new_parse ) {                //有新的解码消息
39              /* 对解码后的时间进行转换，转换成北京时间 */
40              GMTconvert(&info.utc,&beiJingTime,8,1);
41
42              /* 输出解码得到的信息 */
43              printf("\r\n 时间%d,%d,%d,%d,%d,%d\r\n", beiJingTime.year+1900,
                        beiJingTime.mon+1,beiJingTime.day, beiJingTime.hour,
44                                                  beiJingTime.min,beiJingTime.sec);
45              printf("\r\n 纬度：%f,经度%f\r\n",info.lat,info.lon);
46              printf("\r\n 正在使用的卫星：%d,可见卫星：%d",info.satinfo.inuse,
                                                        info.satinfo.inview);
47              printf("\r\n 海拔高度：%f 米 ", info.elv);
48              printf("\r\n 速度：%f km/h ", info.speed);
49              printf("\r\n 航向：%f 度", info.direction);
50
51              new_parse = 0;
52          }
53
54      }
55
56      /* 释放 GPS 数据结构 */
57      // nmea_parser_destroy(&parser);
58
59
60      //  return 0;
61  }
```

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。

6. u-center 软件

6.1 软件简介

野火多功能调试助手可以配合 GPS 模块在百度地图上进行精准的定位，方便应用。而 u-blox 公司提供的 u-center 软件则具有配置模块工作方式的功能，方便开发调试。

在野火提供的 GPS 模块资料如下目录可找到 u-center 的压缩包：野火 GPS 模块资料\5.测试软件\u-center。解压安装即可，在正常使用前还需要安装 u-blox 6 GPS Receiver 驱动。软件运行界面见图 6-1。

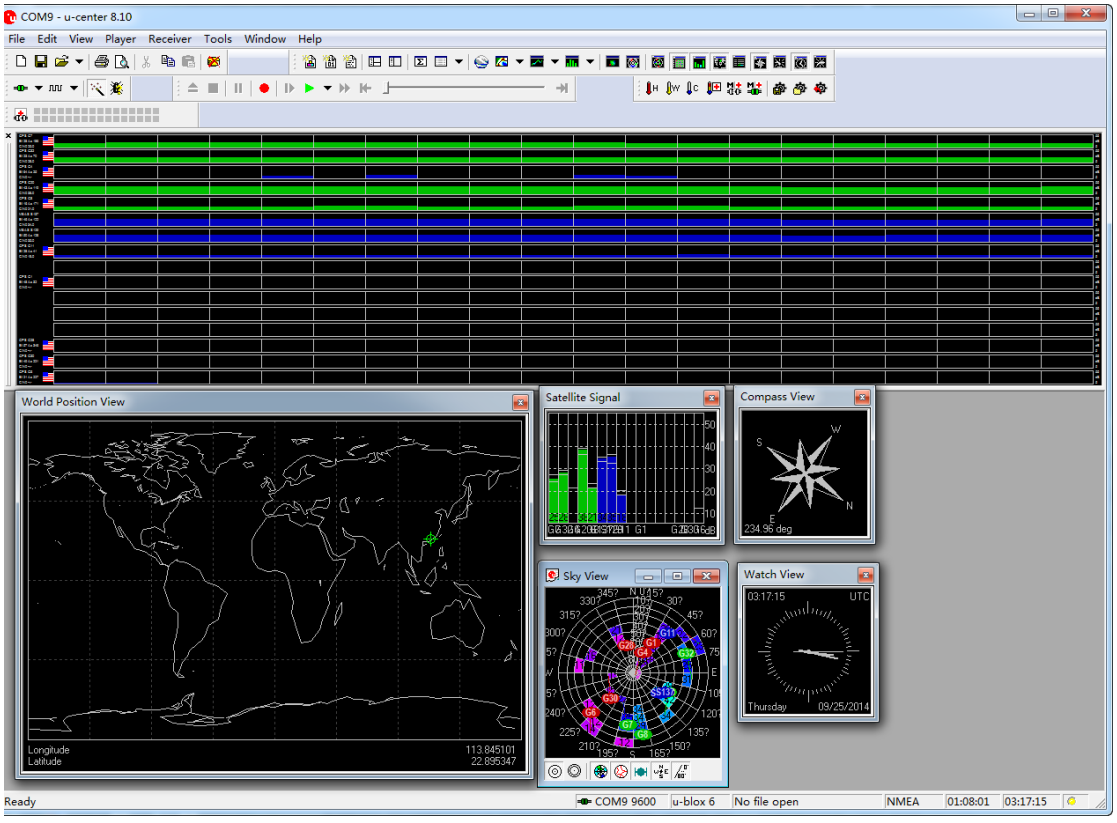

图 6-1 u-center 软件运行界面

该软件的主要功能如下：

- 对 GPS 模块进行性能测试
- 修改 GPS 模块的寄存器配置，控制其工作方式
- 更新 GPS 模块固件
- 测试附加的 u-blox AssistNow 服务(辅助系统，加快搜星速度)

6.2 软件操作

6.2.1 连接 GPS 模块

使用 u-center 软件时，把 WF-NEO-6M 模块通过 USB 线接入到电脑，若电脑已经安装 u-blox 6 GPS Receiver 驱动，该软件会找到模块对应的 com 口。然后像使用串口调试软件一样，配置好波特率及 com 口并连接，即可正常使用软件，见图 6-2 及图 6-3。

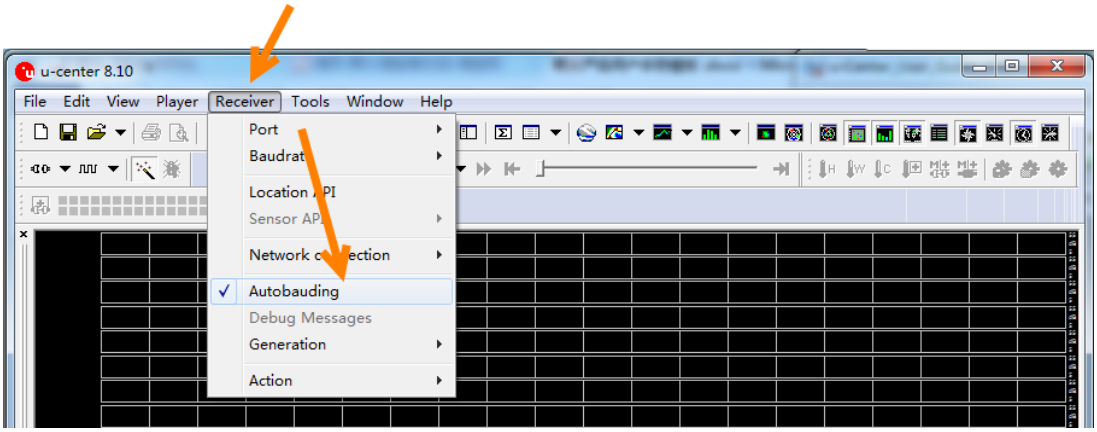

图 6-2 配置串口波特率为自动检测

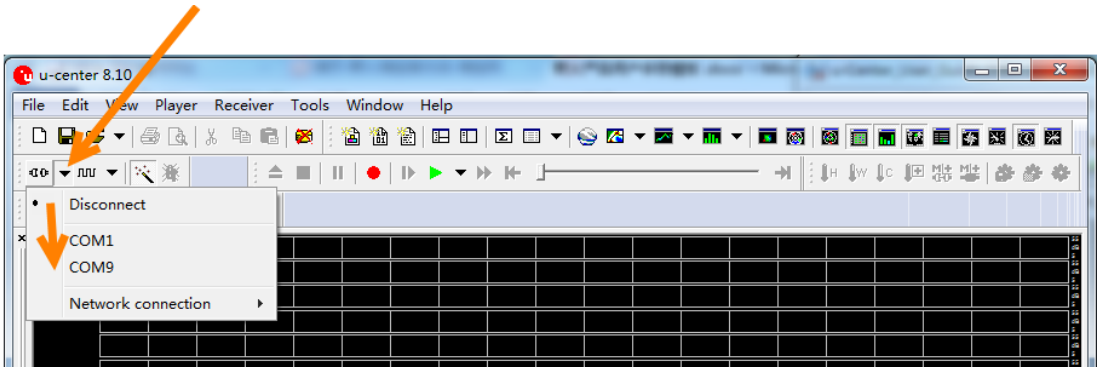

图 6-3 选择 WF-NEO-6M 对应的 COM 口并连接

6.2.2 查看可视化信息

u-center 软件提供了指南针、时钟、卫星信号质量等各种 GPS 信息可视化窗口，如果软件默认没有打开该窗口，可以在软件的工具栏中调出来，见图 6-4。

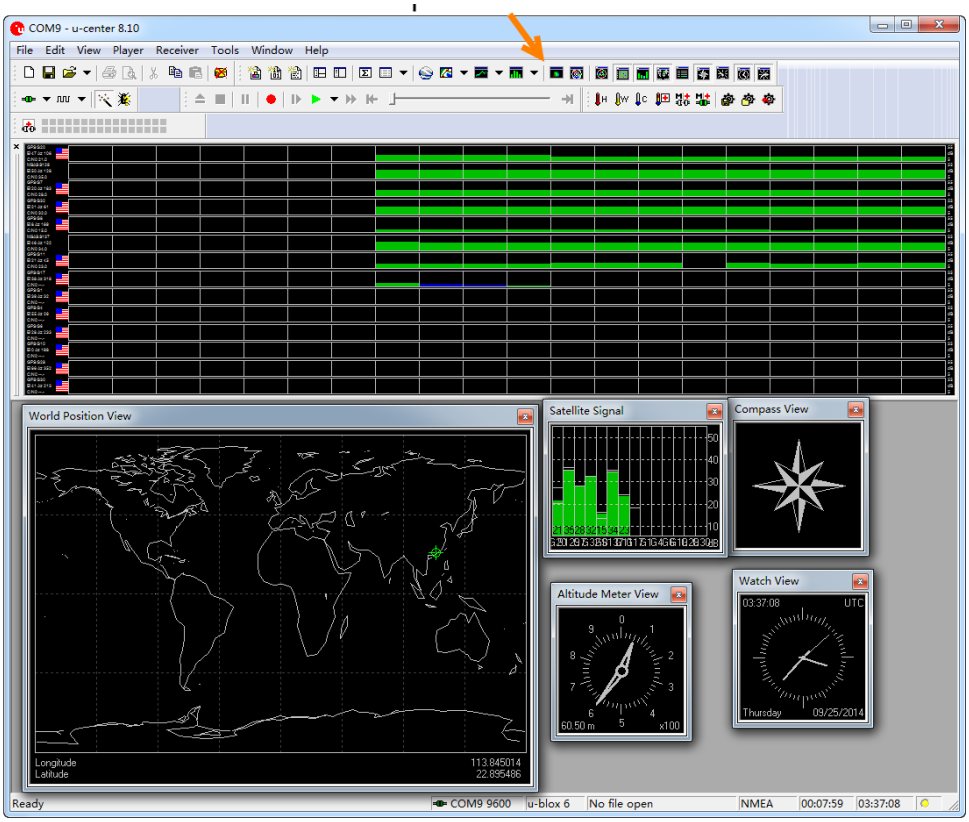

图 6-4 u-center 的各种 GPS 信息可视化窗口

6.2.3 配置 GPS 模块寄存器

使用 u-center 软件可以控制 GPS 模块的寄存器，从而改变它的工作方式，在软件的菜单栏：View->Messages View (F9)可以调出信息窗口。

我们来尝试调节一下时间脉冲的频率，见图 6-5。在调出的窗口中，选择到 UBX->CFG->TP(Timepulse)目录，在该配置选项中把 Pulse Period 由 1000ms 设置为 200ms，Pulse Length 配置为 100ms,然后点击窗口左下角的 send 按钮把配置发送到模块，然后就可以看到模块上的时间脉冲指示灯闪烁的频率明显加快了。关于其它寄存器的配置可以参考 NEO-6M 的操作说明《u-blox6_ReceiverDescriptionProtocolSpec_GPS.G6-SW-10018-C.pdf》。

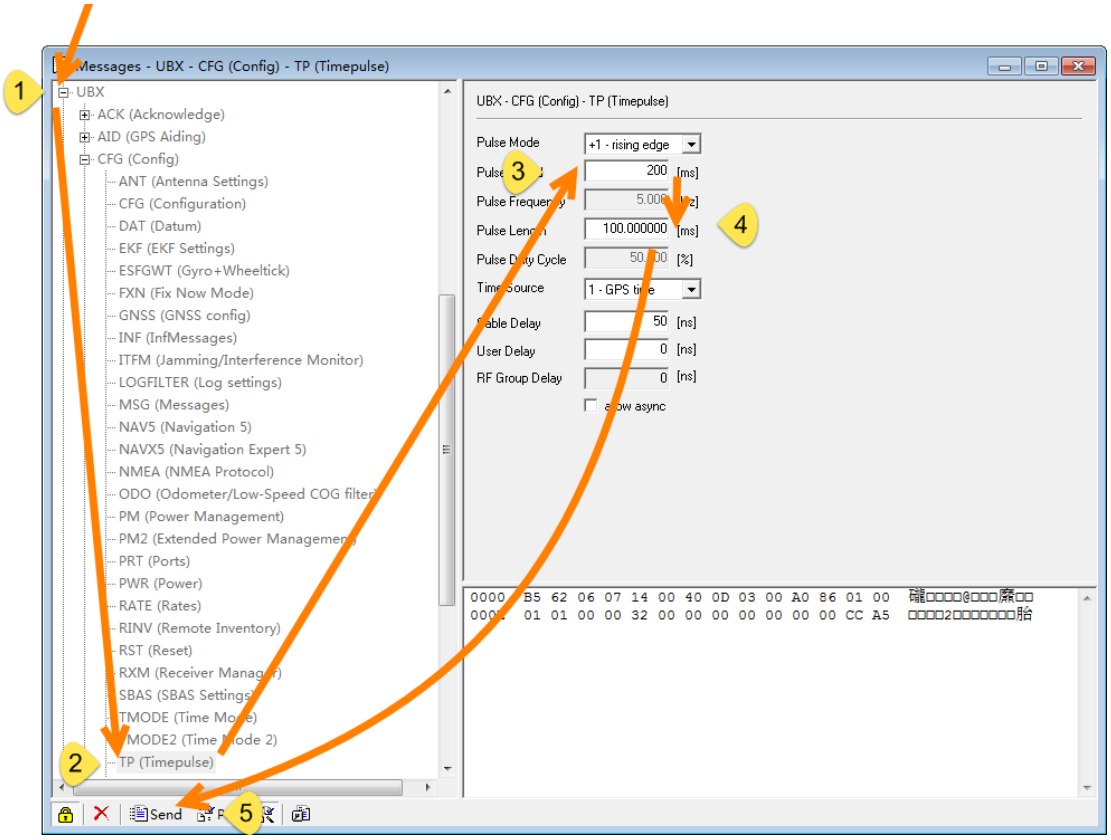

图 6-5 配置时间脉冲频率

关于 u-center 软件的基本使用就介绍到这里，更多详细的软件使用方法请参考文档《u-Center_User_Guide_(UBX-13005250).pdf》。

7. 产品更新及售后支持

野火的产品资料更新会第一时间发布到论坛：http://www.chuxue123.com

购买野火产品请到野火官方淘宝店铺：http://firestm32.taobao.com

在学习或使用野火产品时遇到问题可在论坛发帖子与我们交流。