开源共享　　携手共进

深圳普中科技有限公司

官方网站：www.prechin.cn

技术论坛：www.prechin.net

技术 QQ：2489019400

咨询电话：0755-61139052

PZ-S1216F8-BD GPS 北斗定位模块开发手册

本手册我们将向大家介绍 PZ-S1216F8-BD GPS 北斗定位模块及其在普中 STM32F4 开发板上的使用。本手册我们将使用 PZ-S1216F8-BD GPS 北斗定位模块实现 GPS/北斗定位。本章分为如下几部分内容：

1 PZ-S1216F8-BD GPS 北斗定位模块介绍

2 硬件设计

3 软件设计

4 实验现象

1 PZ-S1216F8-BD GPS 北斗定位模块介绍

1.1 特性参数

PZ-S1216F8-BD GPS/北斗模块是一款高性能 GPS/北斗双模定位模块。该模块主要特点包括：

①模块采用 S1216F8-BD 模组，体积小巧，性能优异。

②模块可通过串口进行各种参数设置，并可保存在内部 FLASH，使用方便。

③模块自带 IPX 接口，可以连接各种有源天线，建议连接 GPS/北斗双模有源天线。

④模块兼容 3.3V/5V 电平，方便连接各种单片机系统。

⑤模块自带可充电后备电池，可以掉电保持星历数据（在主电源断开后，后备电池可以维持半小时左右的 GPS/北斗星历数据的保存，以支持温启动或热启动，从而实现快速定位）。

模块通过串口与外部系统连接，串口波特率支持 4800、 9600、 19200、 38400（默认）、57600、 115200、 230400 等不同速率，兼容 5V/3.3V 单片机系统，可以非常方便的与您的产品进行连接。该模块各参数如图所示：

项目	说明
接口特性	TTL，兼容 3.3V/5V 单片机系统
支持波特率	4800、 9600、 19200、 38400（默认）、 57600、 115200、 230400
接收特性	167 通道， 支持 QZSS， WAAS，MSAS， EGNOS， GAGAN
定位精度	2.5 mCEP （ SBAS： 2.0mCEP）
更新速率	1/2/4/5/8/10/20 Hz
捕获时间	冷启动(1)： 29S（最快） 温启动： 27S 热启动： 1S
冷启动灵敏度	-148dBm
捕获追踪灵敏度	-165dBm
通信协议	NMEA-0183 V3.01
工作温度	-40℃~85℃
模块尺寸	19.66mm*44.75mm

工作电压（ VCC）	DC3.3V~5.0V
工作电流	45mA
Voh	2.4V(Min)
Vol	0.4V(Max)
Vih	2V(Min)
Vil	0.8V(Max)
TXD/RXD 阻抗(2)	120欧

注（1）： 冷启动是指模块所有保存的 GPS/北斗接收历史信息都丢失了（相当于主电源和后备电池都没电了），这种情况下重启，称之为冷启动。温启动是指模块保存了 GPS/北斗接收历史信息，但是当前可视卫星的信息和保存的信息不一致了,这样的条件下重启,称之为温启动。热启动则是指在模块保存了 GPS/北斗接收历史信息且与当前可视卫星信息一致，这样的条件下重启，称之为热启动。

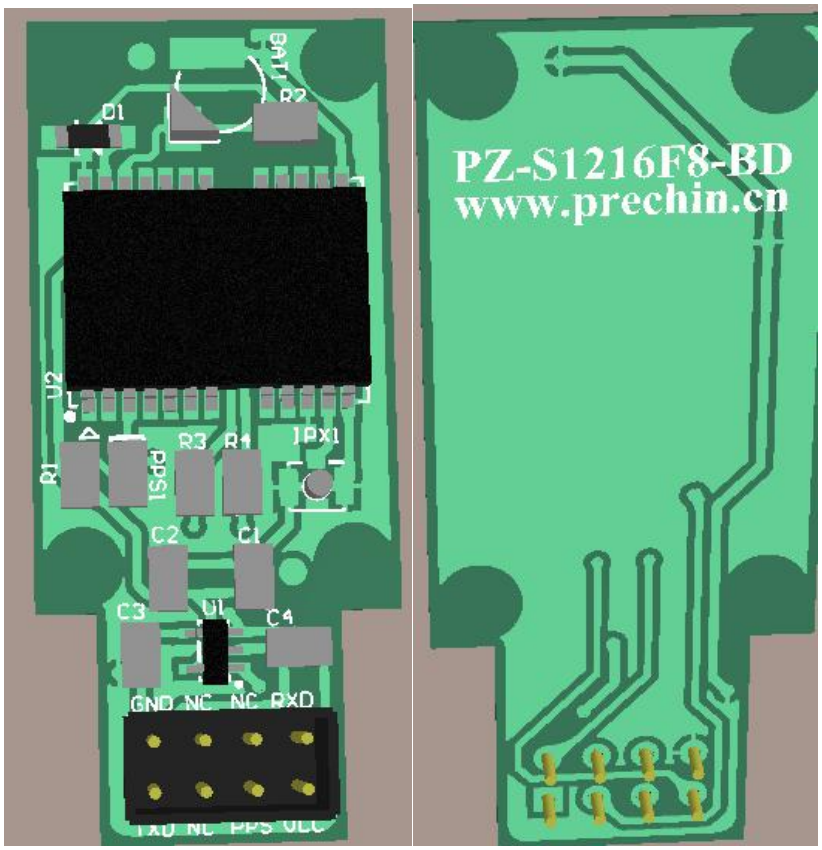
注（2）： 模块的 TXD 和 RXD 脚内部接了 120 欧电阻，做输出电平兼容处理，所以在使用的时候要注意，导线电阻不可过大(尤其是接 USB 转 TTL 串口模块的时候,如果模块的 TXD、 RXD 上带了 LED，那就会有问题)，否则可能导致通信不正常。

PZ-S1216F8-BD GPS/北斗模块支持多种通信波特率， 通过串口进行设置，并可以保存在模块内部 FLASH,模块默认波特率为：38400（ 8 位数据位， 1 位停止位，无奇偶校验），详细的设置方法，我们会在后面介绍。

1.2 模块说明

1.2.1 模块引脚说明

PZ-S1216F8-BD GPS/北斗模块非常小巧（19.66mm*44.75mm），模块通过 1 个 2*4 的间距为 2.54mm 的排针与外部连接， 模块上有 4 个安装孔，方便大家安装到自己的设备里面， 模块外观如图所示：



从图中可以看到，从右到左依次为模块引出的 8 个脚，其中 NC 代表未与模块相连即待扩展脚，可用的管脚只有 5 个，各引脚的详细描述如图所示：

管脚名称	功能说明
VCC	电源（ 3.3V~5.0V）
GND	地
TXD	模块串口发送脚（ TTL 电平，不能直接接 RS232 电平！）， 可接单片机的 RXD
RXD	模块串口接收脚（ TTL 电平，不能直接接 RS232 电平！）， 可接单片机的 TXD
PPS	时钟脉冲输出脚

其中 PPS 引脚同时连接到了模块自带了的状态指示灯：PPS1，该引脚连接在 S1216F8-BD 模组的 1PPS 端口，该端口的输出特性可以通过程序设置。PPS1 指示灯（即 PPS 引脚），在默认条件下（没经过程序设置），有 2 个状态：

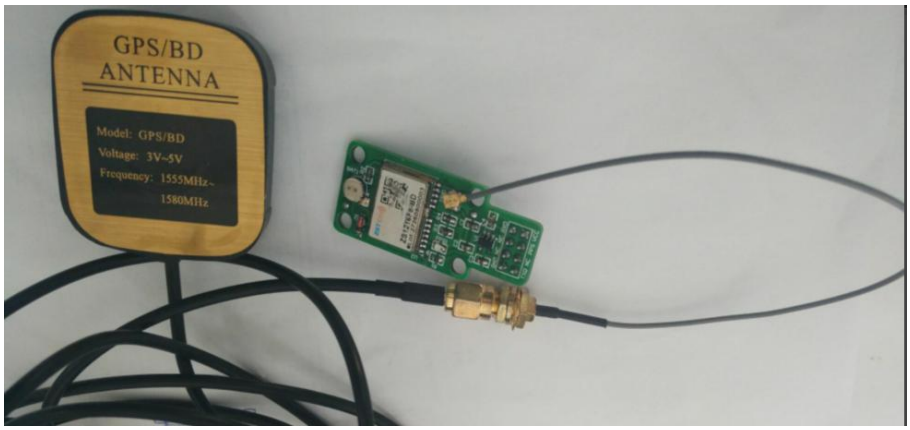
1， 常亮，表示模块已开始工作，但还未实现定位。

2， 闪烁（ 100ms 灭， 900ms 亮），表示模块已经定位成功。

这样， 通过 PPS1 指示灯，我们就可以很方便的判断模块的当前状态，方便大家使用。

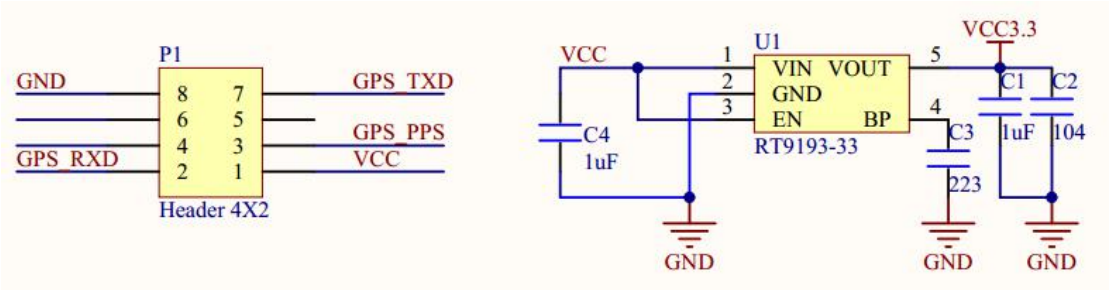
另外，模块的右下角的 IPX1 接口，用来外接一个有源天线，通过外接有源天线，我们就可以把模块放到室内，天线放到室外，实现室内定位。

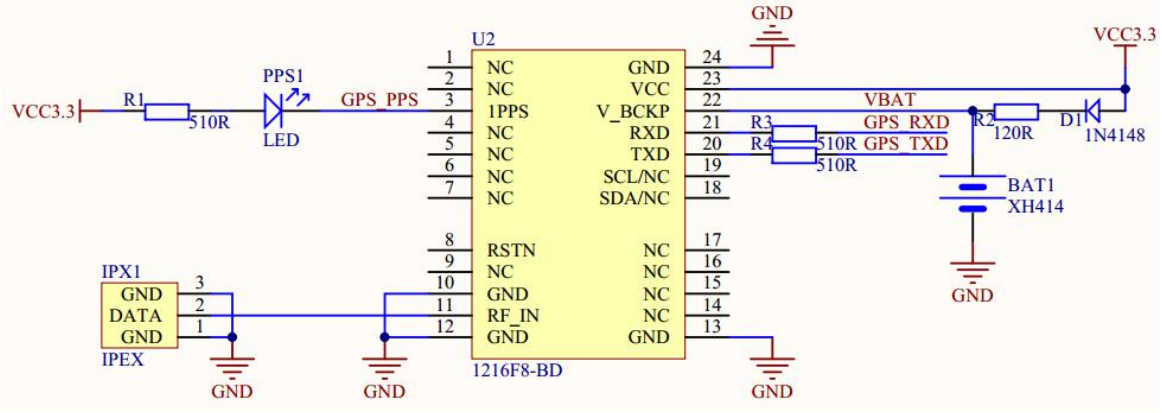
一般 GPS 有源天线都是采用 SMA 接口，我们需要准备一根 IPX（ IPEX）转 SMA 的连接线，从而连接 PZ-S1216F8-BD GPS/北斗模块与有源天线，如图所示：



PZ-S1216F8-BD GPS/北斗模块默认采用 NMEA-0183 协议输出 GPS/北斗定位数据，并可以通过 SkyTraq 协议对模块进行配置，NMEA-0183 协议将在后面会详细介绍，SkyTraq 配置协议，请参考《 Binary Messages of SkyTraq Venus 8 GNSSReceiver.pdf》 。

使用 PZ-S1216F8-BD GPS/北斗模块，任何单片机（ 3.3V/5V 电源）都可以很方便的实现 GPS/北斗定位。PZ-S1216F8-BD GPS/北斗模块的原理图如图所示：





1.2.2 模块使用说明

PZ-S1216F8-BD GPS/北斗模块同外部设备的通信接口采用 UART（串口）方式，输出的 GPS/北斗定位数据采用 NMEA-0183 协议（默认），控制协议为 SkyTraq 协议（该协议的详细介绍请看 Binary Messages of SkyTraq Venus 8 GNSS Receiver.pdf 这个文档）。这里我们主要向大家介绍 NMEA-0183 协议。

（1）NMEA-0183 协议简介

NMEA 0183 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。目前业已成了 GPS/北斗导航设备统一的 RTCM（ Radio TechnicalCommission for Maritime services）标准协议。

NMEA-0183 协议采用 ASCII 码来传递 GPS 定位信息，我们称之为帧。

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（ aa），后三位为语句名（ ccc）

3、ddd…ddd： 数据

4、“ *”： 校验和前缀（也可以作为语句数据结束的标志）

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)： 帧结束，回车和换行符

NMEA-0183 常用命令如图所示：

序号	命令	说明	最大帧长
1	$GNGGA	GPS/北斗定位信息	72
2	$GNGSA	当前卫星信息	65
3	$GPGSV	可见 GPS 卫星信息	210
4	$BDGSV	可见北斗卫星信息	210
5	$GNRMC	推荐定位信息	70
6	$GNVTG	地面速度信息	34
7	$GNGLL	大地坐标信息	--
8	$GNZDA	当前时间(UTC[1])信息	--

注 1： 即协调世界时，相当于本初子午线(0 度经线)上的时间，北京时间比 UTC 早 8 个小时。

接下来我们分别介绍这些命令。

1.$GNGGA（ GPS 定位信息， Global Positioning System Fix Data）

$GNGGA 语句的基本格式如下（其中 M 指单位 M， hh 指校验和， CR 和 LF 代表回车换行，下同）：

$GNGGA,(1),(2),(3),(4),(5),(6),(7),(8),(9),M,(10),M,(11),(12)*hh(CR)(LF)

(1) UTC 时间，格式为 hhmmss.ss；

(2) 纬度，格式为 ddmm.mmmmm（ 度分格式）；

(3) 纬度半球， N 或 S（ 北纬或南纬）；

(4) 经度，格式为 dddmm.mmmmm（度分格式）；

(5) 经度半球， E 或 W（ 东经或西经）；

(6) GPS 状态， 0=未定位， 1=非差分定位， 2=差分定位；

(7) 正在使用的用于定位的卫星数量（ 00~12）

(8) HDOP 水平精确度因子（ 0.5~99.9）

(9) 海拔高度（ -9999.9 到 9999.9 米）

(10) 大地水准面高度（ -9999.9 到 9999.9 米）

(11) 差分时间（从最近一次接收到差分信号开始的秒数，非差分定位，此项为空）

(12) 差分参考基站标号（ 0000 到 1023， 首位 0 也将传送，非差分定位，此项为空）

例如：

$GNGGA,095528.000,2318.1133,N,11319.7210,E,1,06,3.7,55.1,M,-5.4,M,,0000*69

2.$GNGSA（ 当前卫星信息）

$GNGSA 语句的基本格式如下：

$GNGSA,(1),(2),(3),(3),(3),(3),(3),(3),(3),(3),(3),(3),(3),(3),(4),(5),(6)*hh(CR)(LF)

(1) 模式， M = 手动， A = 自动。

(2) 定位类型， 1=未定位， 2=2D 定位， 3=3D 定位。

(3) 正在用于定位的卫星号（ 01~32）

(4) PDOP 综合位置精度因子（ 0.5-99.9）

(5) HDOP 水平精度因子①（ 0.5-99.9）

(6) VDOP 垂直精度因子（ 0.5-99.9）

例如：

$GNGSA,A,3,14,22,24,12,,,,,,,,,4.2,3.7,2.1*2D

$GNGSA,A,3,209,214,,,,,,,,,,,4.2,3.7,2.1*21

注①： 精度因子值越小，则准确度越高。

3.$GPGSV（ 可见卫星数， GPS Satellites in View）

$GPGSV 语句的基本格式如下：

$GPGSV, (1),(2),(3),(4),(5),(6),(7),...,(4),(5),(6),(7)*hh(CR)(LF)

(1) GSV 语句总数。

(2) 本句 GSV 的编号。

(3) 可见卫星的总数（ 00~12，前面的 0 也将被传输）。

(4) 卫星编号（ 01~32，前面的 0 也将被传输）。

(5) 卫星仰角（ 00~90 度，前面的 0 也将被传输）。

(6) 卫星方位角（ 000~359 度，前面的 0 也将被传输）

(7) 信噪比（ 00~99dB，没有跟踪到卫星时为空）。

注：每条 GSV 语句最多包括四颗卫星的信息，其他卫星的信息将在下一条 $GPGSV 语句中输出。

例如：

$GPGSV,3,1,11,18,73,129,19,10,71,335,40,22,63,323,41,25,49,127,06*78

$GPGSV,3,2,11,14,41,325,46,12,36,072,34,31,32,238,22,21,23,194,08*76

$GPGSV,3,3,11,24,21,039,40,20,08,139,07,15,08,086,03*45

4.$BDGSV（ 可见卫星数， GPS Satellites in View）

$BDGSV 语句的基本格式如下：

$BDGSV, (1),(2),(3),(4),(5),(6),(7),...,(4),(5),(6),(7)*hh(CR)(LF)

(1) GSV 语句总数。

(2) 本句 GSV 的编号。

(3) 可见卫星的总数（ 00~12，前面的 0 也将被传输）。

(4) 卫星编号（ 01~32，前面的 0 也将被传输）。

(5) 卫星仰角（ 00~90 度，前面的 0 也将被传输）。

(6) 卫星方位角（ 000~359 度，前面的 0 也将被传输）

(7) 信噪比（ 00~99dB，没有跟踪到卫星时为空）。

注：每条 GSV 语句最多包括四颗卫星的信息，其他卫星的信息将在下一条 $BDGSV 语句中输出。

例如：

$BDGSV,1,1,02,209,64,354,40,214,05,318,40*69

5.$GNRMC（ 推荐定位信息， Recommended Minimum Specific GPS/Transit Data）

$GNRMC 语句的基本格式如下：

$GNRMC,(1),(2),(3),(4),(5),(6),(7),(8),(9),(10),(11),(12)*hh(CR)(LF)

(1) UTC 时间， hhmmss（时分秒）

(2) 定位状态， A=有效定位， V=无效定位

(3) 纬度 ddmm.mmmmm（度分）

(4) 纬度半球 N（北半球）或 S（南半球）

(5) 经度 dddmm.mmmmm（度分）

(6) 经度半球 E（东经）或 W（西经）

(7) 地面速率（ 000.0~999.9 节）

(8) 地面航向（ 000.0~359.9 度，以真北方为参考基准）

(9) UTC 日期， ddmmyy（日月年）

(10) 磁偏角（ 000.0~180.0 度，前导位数不足则补 0）

(11) 磁偏角方向， E（东）或 W（西）

(12) 模式指示（ A=自主定位， D=差分， E=估算， N=数据无效）

例如：

$GNRMC,095554.000,A,2318.1327,N,11319.7252,E,000.0,005.7,081215,,,A*73

6.$GNVTG（ 地面速度信息， Track Made Good and Ground Speed）

$GNVTG 语句的基本格式如下：

$GNVTG,(1),T,(2),M,(3),N,(4),K,(5)*hh(CR)(LF)

(1) 以真北为参考基准的地面航向（ 000~359 度，前面的 0 也将被传输）

(2) 以磁北为参考基准的地面航向(000~359 度，前面的 0 也将被传输)

(3) 地面速率(000.0~999.9 节，前面的 0 也将被传输)

(4) 地面速率(0000.0~1851.8 公里/小时，前面的 0 也将被传输)

(5) 模式指示（ A=自主定位， D=差分， E=估算， N=数据无效）

例如：

$GNVTG,005.7,T,,M,000.0,N,000.0,K,A*11

7. $GNGLL（ 定位地理信息， Geographic Position）

$GNGLL 语句的基本格式如下：

$GNGLL,(1),(2),(3),(4),(5),(6),(7)*hh(CR)(LF)

(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）

例如：

$GNGLL,2318.1330,N,11319.7250,E,095556.000,A,A*4F

8. $GNZDA（ 当前时间信息）

$GNZDA 语句的基本格式如下：

$GNZDA,(1),(2),(3),(4), (5), (6)*hh(CR)(LF)

(1) UTC 时间： hhmmss（时分秒）

(2) 日

(3) 月

(4) 年

(5) 本地区域小时（ NEO-6M 未用到，为 00）

(6) 本地区域分钟（ NEO-6M 未用到，为 00）

例如：

$GNZDA,095555.000,08,12,2015,00,00*4C

NMEA-0183 协议命令帧部分就介绍到这里，接下来我们看看 NMEA-0183 协议的校验，通过前面的介绍，我们知道每一帧最后都有一个 hh 的校验和，该校验和是通过计算$与*之间所有字符 ASCII 码的异或运算得到，将得到的结果以 ASCII 字符表示就是该校验（ hh）。

例如语句： $GNZDA,095555.000,08,12,2015,00,00*4C，校验和（红色部分参与计算）计算方法为：

0X47 xor 0X4E xor 0X5A xor 0X44 xor 0X41 xor 0X2C xor 0X30 xor 0X39 xor 0X35 xor 0X35 xor 0X35 xor 0X35 xor 0X2E xor 0X30 xor 0X30 xor 0X30 xor 0X2C xor 0X30 xor 0X38 xor 0X2C xor 0X31 xor 0X32 xor 0X2C xor 0X32 xor 0X30 xor 0X31 xor 0X35 xor 0X2C xor 0X30 xor 0X30 xor 0X2C xor 0X30 xor 0X30

得到的结果就是 0X4C，用 ASCII 表示就是 4C。

NMEA-0183 协议我们就介绍到这里，了解了该协议，我们就可以编写单片机代码，解析 NMEA-0183 数据，从而得到 GPS/北斗定位的各种信息了。

2 硬件设计

本实验使用到硬件资源如下：

（1）D1 指示灯

（2）K_UP 按键

（3）串口 1、串口 3

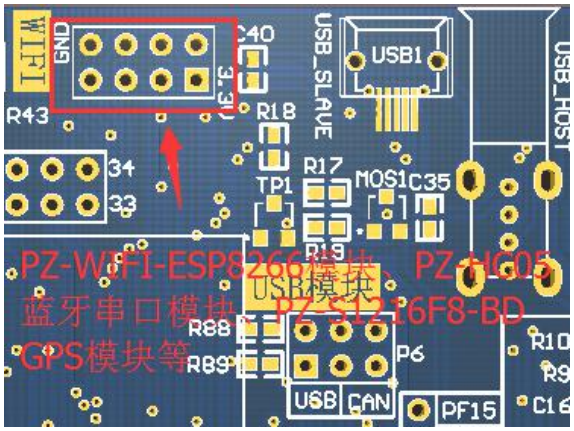
（4）TFTLCD 模块

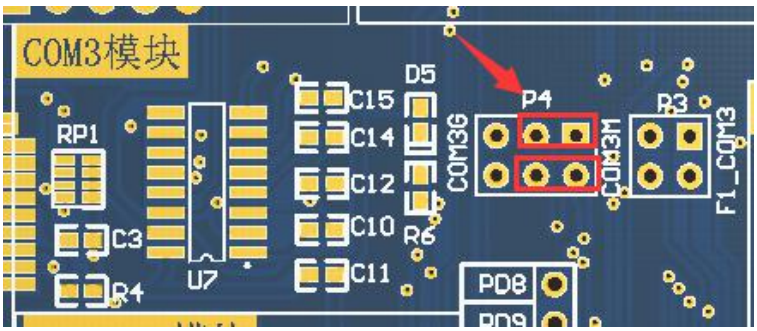
（5）PZ-S1216F8-BD GPS/北斗模块

前四部分电路在前面章节都介绍过，这里就不多说，前面我们介绍了该模块的接口管脚功能，下面我们来看下 PZ-S1216F8-BD GPS/北斗模块与开发板如何连接的。我们 STM32 开发板上板载了一个 WIFI 接口（适用于 PZ-WIFI-ESP8266 模块、PZ-HC05 蓝牙串口模块、PZ-S1216F8-BD GPS 模块等），直接将该模块插上 WIFI 模块接口处即可，其内部管脚连接关系如图所示：

PZ-S1216F8-BD GPS/北斗模块	普中STM32F4开发板（IO）
VCC	3.3V
GND	GND
RXD	PB10(TXD)
TXD	PB11(RXD)
PPS	PF6

注意：因为开发板上的 RS232 模块也是使用串口 3，所以使用 PZ-WIFI-ESP8266 模块、PZ-HC05 蓝牙串口模块、PZ-S1216F8-BD GPS 模块等需要将开发板 RS232 模块处的 P4 端子切换到 COM3M（丝印）处。如图所示：





如果是使用其他板子需要将模块连接到单片机管脚上的画，一定要注意模块的 TXD 要连接单片机的 RXD，模块的 RXD 要连接单片机的 TXD 管脚。

3 软件设计

本实验所实现的功能为：通过串口 3 连接 PZ-S1216F8-BD GPS/北斗模块，然后通过液晶显示 GPS/北斗信息，包括精度、纬度、高度、速度、用于定位的卫星数、可见卫星数、 UTC 时间等信息。另外通过 K_UP 按键，可以开启或关闭 NMEA 数据的上传（即输出到串口 1，方便开发调试）。

我们打开本实验工程，可以看到我们的工程 APP 列表中多了 usart3.c 和 gps.c 源文件，以及头文件 usart3.h、 gps.h。首先，我们来看 usart3.c 里面代码，如下：

```
#include "usart3.h"
#include "stdarg.h"
#include "stdio.h"
#include "string.h"
#include "time.h"


//串口接收缓存区
u8 USART3_RX_BUF[USART3_MAX_RECV_LEN];    // 接收缓冲，最大 USART3_MAX_RECV_LEN 个字节.
u8    USART3_TX_BUF[USART3_MAX_SEND_LEN]; // 发送缓冲，最大 USART3_MAX_SEND_LEN 字节


//通过判断接收连续 2 个字符之间的时间差不大于 10ms 来决定是不是一次连续的数据.
//如果 2 个字符接收间隔超过 10ms,则认为不是 1 次连续数据.也就是超过 10ms 没有接收到
//任何数据,则表示此次接收完毕.
//接收到的数据状态
//[15]:0,没有接收到数据;1,接收到了一批数据.
```

```
//[14:0]:接收到的数据长度
u16 USART3_RX_STA=0;

void USART3_Init(u32 bound)
{

   GPIO_InitTypeDef GPIO_InitStructure;
   USART_InitTypeDef USART_InitStructure;
   NVIC_InitTypeDef NVIC_InitStructure;

   RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB,ENABLE);  // 使 能
GPIOA 时钟
   RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART3,ENABLE);//  使  能
USART3 时钟

   //串口 3 对应引脚复用映射
   GPIO_PinAFConfig(GPIOB,GPIO_PinSource10,GPIO_AF_USART3);
   GPIO_PinAFConfig(GPIOB,GPIO_PinSource11,GPIO_AF_USART3);

   //USART3 端口配置
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10|GPIO_Pin_11 ;
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;//复用功能
   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;    //  速  度
50MHz
   GPIO_InitStructure.GPIO_OType = GPIO_OType_PP; //推挽复用输出
   GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP; //上拉
   GPIO_Init(GPIOB,&GPIO_InitStructure);

   //Usart3 NVIC 配置
```

```
    NVIC_InitStructure.NVIC_IRQChannel = USART3_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority=2 ;//抢占优先级 0
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 3;      //子优先级 0
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;         //IRQ 通道使能
    NVIC_Init(&NVIC_InitStructure); //根据指定的参数初始化 VIC 寄存器

        //USART3 初始化设置
    USART_InitStructure.USART_BaudRate = bound;//串口波特率
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;//字长为 8 位数据格式
    USART_InitStructure.USART_StopBits = USART_StopBits_1;//一个停止位
    USART_InitStructure.USART_Parity = USART_Parity_No;//无奇偶校验位
    USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;//无硬件数据流控制
    USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;  //收发模式
    USART_Init(USART3, &USART_InitStructure); //初始化串口 3

    USART_ITConfig(USART3, USART_IT_RXNE, ENABLE);//开启串口接受中断

    USART_Cmd(USART3, ENABLE);                    //使能串口 3
```

```
    TIM7_Int_Init(100-1,8400-1);    //10ms 中断一次
    USART3_RX_STA=0;     //清零
    TIM_Cmd(TIM7,DISABLE);          //关闭定时器 7
}

void USART3_IRQHandler( void )
{
    u8 res;
    if(USART_GetITStatus(USART3, USART_IT_RXNE) != RESET)//接收到数
据
    {
        res =USART_ReceiveData(USART3);
        if((USART3_RX_STA&(1<<15))==0)//接收完的一批数据,还没有被
处理,则不再接收其他数据
        {
            if(USART3_RX_STA<USART3_MAX_RECV_LEN) //还可以接收数据
            {
                TIM_SetCounter(TIM7,0);//计数器清空
    //计数器清空
                if(USART3_RX_STA==0)                //使能定时器 7 的
中断
                {
                    TIM_Cmd(TIM7,ENABLE);//使能定时器 7
                }
                USART3_RX_BUF[USART3_RX_STA++]=res;    //记录接收到
的值
            }
            else
            {
```

```
                USART3_RX_STA|=1<<15;                    //强制标记接收完
成
            }
        }
    }
}


//串口 3,printf 函数
//确保一次发送数据不超过 USART3_MAX_SEND_LEN 字节
void usart3_printf(char* fmt,...)
{
    u16 i,j;
    va_list ap;
    va_start(ap,fmt);
    vsprintf((char*)USART3_TX_BUF,fmt,ap);
    va_end(ap);
    i=strlen((const char*)USART3_TX_BUF);      //此次发送数据的长度
    for(j=0;j<i;j++)                           //循环发送数据
    {
        while(USART_GetFlagStatus(USART3,USART_FLAG_TC)==RESET); //
循环发送,直到发送完毕
        USART_SendData(USART3,USART3_TX_BUF[j]);
    }
}
```

这部分代码主要实现了串口 3 的初始化，以及实现了串口 3 的 printf 函数：usart3_printf，和串口 3 的接收处理。串口 3 的数据接收，采用了定时判断的方法，对于一次连续接收的数据，如果出现连续 10ms 没有接收到任何数据，则表示这次连续接收数据已经结束。

usart3.h 里面的代码我们就不在这里列出了，请大家参考对应源码。

下面我们来看下 gps.c 里面的代码，如下：

```
#include "gps.h"
#include "led.h"
#include "SysTick.h"
#include "usart3.h"
#include "stdio.h"
#include "stdarg.h"
#include "string.h"
#include "math.h"



const u32 BAUD_id[9]={4800,9600,19200,38400,57600,115200,230400,460800,921600}; //模块支持波特率数组

//从 buf 里面得到第 cx 个逗号所在的位置
//返回值:0~0XFE,代表逗号所在位置的偏移.
//       0XFF,代表不存在第 cx 个逗号
u8 NMEA_Comma_Pos(u8 *buf,u8 cx)
{
   u8 *p=buf;
   while(cx)
   {
      if(*buf=='*'||*buf<' '||*buf>'z')return 0XFF;//遇到'*'或者非法字符,则不存在第 cx 个逗号
      if(*buf==',')cx--;
```

```
        buf++;
    }
    return buf-p;
}

//m^n 函数
//返回值:m^n 次方.
u32 NMEA_Pow(u8 m,u8 n)
{
    u32 result=1;
    while(n--)result*=m;
    return result;
}

//str 转换为数字,以','或者'*'结束
//buf:数字存储区
//dx:小数点位数,返回给调用函数
//返回值:转换后的数值
int NMEA_Str2num(u8 *buf,u8*dx)
{
    u8 *p=buf;
    u32 ires=0,fres=0;
    u8 ilen=0,flen=0,i;
    u8 mask=0;
    int res;
    while(1) //得到整数和小数的长度
    {
        if(*p=='-'){mask|=0X02;p++;}//是负数
        if(*p==','||(*p=='*'))break;//遇到结束了
```

```
        if(*p=='.'){mask|=0X01;p++;}//遇到小数点了
        else if(*p>'9'||(*p<'0'))   //有非法字符
        {
            ilen=0;
            flen=0;
            break;
        }
        if(mask&0X01)flen++;
        else ilen++;
        p++;
    }
    if(mask&0X02)buf++;  //去掉负号
    for(i=0;i<ilen;i++)  //得到整数部分数据
    {
        ires+=NMEA_Pow(10,ilen-1-i)*(buf[i]-'0');
    }
    if(flen>5)flen=5; //最多取 5 位小数
    *dx=flen;         //小数点位数
    for(i=0;i<flen;i++)  //得到小数部分数据
    {
        fres+=NMEA_Pow(10,flen-1-i)*(buf[ilen+1+i]-'0');
    }
    res=ires*NMEA_Pow(10,flen)+fres;
    if(mask&0X02)res=-res;
    return res;
}

//分析 GPGSV 信息
//gpsx:nmea 信息结构体
```

```
//buf:接收到的 GPS 数据缓冲区首地址
void NMEA_GPGSV_Analysis(nmea_msg *gpsx,u8 *buf)
{
    u8 *p,*p1,dx;
    u8 len,i,j,slx=0;
    u8 posx;
    p=buf;
    p1=(u8*)strstr((const char *)p,"$GPGSV");
    len=p1[7]-'0';                              //得到 GPGSV 的条数
    posx=NMEA_Comma_Pos(p1,3);                  //得到可见卫星总数
    if(posx!=0XFF)gpsx->svnum=NMEA_Str2num(p1+posx,&dx);
    for(i=0;i<len;i++)
    {
        p1=(u8*)strstr((const char *)p,"$GPGSV");
        for(j=0;j<4;j++)
        {
            posx=NMEA_Comma_Pos(p1,4+j*4);

    if(posx!=0XFF)gpsx->slmsg[slx].num=NMEA_Str2num(p1+posx,&dx);  //
得到卫星编号
            else break;
            posx=NMEA_Comma_Pos(p1,5+j*4);

    if(posx!=0XFF)gpsx->slmsg[slx].eledeg=NMEA_Str2num(p1+posx,&dx);/
/得到卫星仰角
            else break;
            posx=NMEA_Comma_Pos(p1,6+j*4);

    if(posx!=0XFF)gpsx->slmsg[slx].azideg=NMEA_Str2num(p1+posx,&dx);/
```

```
/得到卫星方位角
                else break;
                posx=NMEA_Comma_Pos(p1,7+j*4);

    if(posx!=0XFF)gpsx->slmsg[slx].sn=NMEA_Str2num(p1+posx,&dx);   //
得到卫星信噪比
                else break;
                slx++;
            }
            p=p1+1;//切换到下一个 GPGSV 信息
        }
    }

    //分析 BDGSV 信息
    //gpsx:nmea 信息结构体
    //buf:接收到的 GPS 数据缓冲区首地址
    void NMEA_BDGSV_Analysis(nmea_msg *gpsx,u8 *buf)
    {
        u8 *p,*p1,dx;
        u8 len,i,j,slx=0;
        u8 posx;
        p=buf;
        p1=(u8*)strstr((const char *)p,"$BDGSV");
        len=p1[7]-'0';                              //得到 BDGSV 的条数
        posx=NMEA_Comma_Pos(p1,3);                  //得到可见北斗卫星总
数
        if(posx!=0XFF)gpsx->beidou_svnum=NMEA_Str2num(p1+posx,&dx);
        for(i=0;i<len;i++)
        {
```

```
            p1=(u8*)strstr((const char *)p,"$BDGSV");
            for(j=0;j<4;j++)
            {
                posx=NMEA_Comma_Pos(p1,4+j*4);

    if(posx!=0XFF)gpsx->beidou_slmsg[slx].beidou_num=NMEA_Str2num(p1+
posx,&dx);//得到卫星编号
                else break;
                posx=NMEA_Comma_Pos(p1,5+j*4);

    if(posx!=0XFF)gpsx->beidou_slmsg[slx].beidou_eledeg=NMEA_Str2num(
p1+posx,&dx);//得到卫星仰角
                else break;
                posx=NMEA_Comma_Pos(p1,6+j*4);

    if(posx!=0XFF)gpsx->beidou_slmsg[slx].beidou_azideg=NMEA_Str2num(
p1+posx,&dx);//得到卫星方位角
                else break;
                posx=NMEA_Comma_Pos(p1,7+j*4);

    if(posx!=0XFF)gpsx->beidou_slmsg[slx].beidou_sn=NMEA_Str2num(p1+p
osx,&dx);  //得到卫星信噪比
                else break;
                slx++;
            }
            p=p1+1;//切换到下一个 BDGSV 信息
        }
    }
```

```
//分析 GNGGA 信息
//gpsx:nmea 信息结构体
//buf:接收到的 GPS 数据缓冲区首地址
void NMEA_GNGGA_Analysis(nmea_msg *gpsx,u8 *buf)
{
    u8 *p1,dx;
    u8 posx;
    p1=(u8*)strstr((const char *)buf,"$GNGGA");
    posx=NMEA_Comma_Pos(p1,6);                          //得到 GPS
状态
    if(posx!=0XFF)gpsx->gpssta=NMEA_Str2num(p1+posx,&dx);
    posx=NMEA_Comma_Pos(p1,7);                          //得到用
于定位的卫星数
    if(posx!=0XFF)gpsx->posslnum=NMEA_Str2num(p1+posx,&dx);
    posx=NMEA_Comma_Pos(p1,9);                          //得到海
拔高度
    if(posx!=0XFF)gpsx->altitude=NMEA_Str2num(p1+posx,&dx);
}

//分析 GNGSA 信息
//gpsx:nmea 信息结构体
//buf:接收到的 GPS 数据缓冲区首地址
void NMEA_GNGSA_Analysis(nmea_msg *gpsx,u8 *buf)
{
    u8 *p1,dx;
    u8 posx;
    u8 i;
    p1=(u8*)strstr((const char *)buf,"$GNGSA");
    posx=NMEA_Comma_Pos(p1,2);                          //得到定
```

```
位类型
        if(posx!=0XFF)gpsx->fixmode=NMEA_Str2num(p1+posx,&dx);
        for(i=0;i<12;i++)                                    //得到定位卫
星编号
        {
            posx=NMEA_Comma_Pos(p1,3+i);
            if(posx!=0XFF)gpsx->possl[i]=NMEA_Str2num(p1+posx,&dx);
            else break;
        }
        posx=NMEA_Comma_Pos(p1,15);                          //  得  到
PDOP 位置精度因子
        if(posx!=0XFF)gpsx->pdop=NMEA_Str2num(p1+posx,&dx);
        posx=NMEA_Comma_Pos(p1,16);                          //  得  到
HDOP 位置精度因子
        if(posx!=0XFF)gpsx->hdop=NMEA_Str2num(p1+posx,&dx);
        posx=NMEA_Comma_Pos(p1,17);                          //  得  到
VDOP 位置精度因子
        if(posx!=0XFF)gpsx->vdop=NMEA_Str2num(p1+posx,&dx);
    }

    //分析 GNRMC 信息
    //gpsx:nmea 信息结构体
    //buf:接收到的 GPS 数据缓冲区首地址
    void NMEA_GNRMC_Analysis(nmea_msg *gpsx,u8 *buf)
    {
        u8 *p1,dx;
        u8 posx;
        u32 temp;
        float rs;
```

```
        p1=(u8*)strstr((const char *)buf,"$GNRMC");//"$GNRMC",经常有&
和 GNRMC 分开的情况,故只判断 GPRMC.
        posx=NMEA_Comma_Pos(p1,1);                                  //得到 UTC
时间
        if(posx!=0XFF)
        {
            temp=NMEA_Str2num(p1+posx,&dx)/NMEA_Pow(10,dx);       //得到
UTC 时间,去掉 ms
            gpsx->utc.hour=temp/10000;
            gpsx->utc.min=(temp/100)%100;
            gpsx->utc.sec=temp%100;
        }
        posx=NMEA_Comma_Pos(p1,3);                                  //得到纬
度
        if(posx!=0XFF)
        {
            temp=NMEA_Str2num(p1+posx,&dx);
            gpsx->latitude=temp/NMEA_Pow(10,dx+2); //得到°
            rs=temp%NMEA_Pow(10,dx+2);              //得到'

    gpsx->latitude=gpsx->latitude*NMEA_Pow(10,5)+(rs*NMEA_Pow(10,5-dx)
)/60;//转换为°
        }
        posx=NMEA_Comma_Pos(p1,4);                                  //南纬还
是北纬
        if(posx!=0XFF)gpsx->nshemi=*(p1+posx);
        posx=NMEA_Comma_Pos(p1,5);                                  //得到经
度
        if(posx!=0XFF)
```

```
        {
            temp=NMEA_Str2num(p1+posx,&dx);
            gpsx->longitude=temp/NMEA_Pow(10,dx+2);   //得到°
            rs=temp%NMEA_Pow(10,dx+2);              //得到'

    gpsx->longitude=gpsx->longitude*NMEA_Pow(10,5)+(rs*NMEA_Pow(10,5-
dx))/60;//转换为°
        }
        posx=NMEA_Comma_Pos(p1,6);                              // 东 经 还
是西经
        if(posx!=0XFF)gpsx->ewhemi=*(p1+posx);
        posx=NMEA_Comma_Pos(p1,9);                              //得到 UTC
日期
        if(posx!=0XFF)
        {
            temp=NMEA_Str2num(p1+posx,&dx);                     //得到 UTC
日期
            gpsx->utc.date=temp/10000;
            gpsx->utc.month=(temp/100)%100;
            gpsx->utc.year=2000+temp%100;
        }
    }

    //分析 GNVTG 信息
    //gpsx:nmea 信息结构体
    //buf:接收到的 GPS 数据缓冲区首地址
    void NMEA_GNVTG_Analysis(nmea_msg *gpsx,u8 *buf)
    {
        u8 *p1,dx;
```

```
        u8 posx;
        p1=(u8*)strstr((const char *)buf,"$GNVTG");

        posx=NMEA_Comma_Pos(p1,7);                              //得到地面速率
        if(posx!=0XFF)
        {
            gpsx->speed=NMEA_Str2num(p1+posx,&dx);
            if(dx<3)gpsx->speed*=NMEA_Pow(10,3-dx);             //确保扩大1000倍
        }
    }

    //提取NMEA-0183信息
    //gpsx:nmea信息结构体
    //buf:接收到的GPS数据缓冲区首地址
    void GPS_Analysis(nmea_msg *gpsx,u8 *buf)
    {
        NMEA_GPGSV_Analysis(gpsx,buf); //GPGSV解析
        NMEA_BDGSV_Analysis(gpsx,buf); //BDGSV解析
        NMEA_GNGGA_Analysis(gpsx,buf); //GNGGA解析
        NMEA_GNGSA_Analysis(gpsx,buf); //GPNSA解析
        NMEA_GNRMC_Analysis(gpsx,buf); //GPNMC解析
        NMEA_GNVTG_Analysis(gpsx,buf); //GPNTG解析
    }

    ////////////////////////////////////////UBLOX 配置代码///////////////////////////////////
    ////检查CFG配置执行情况
```

```
////返回值:0,ACK 成功
////        1,接收超时错误
////        2,没有找到同步字符
////        3,接收到 NACK 应答
u8 S1216F8BD_Cfg_Ack_Check(void)
{
   u16 len=0,i;
   u8 rval=0;
   while((USART3_RX_STA&0X8000)==0 && len<100)//等待接收到应答
   {
      len++;
      delay_ms(5);
   }
   if(len<100)       //超时错误.
   {
      len=USART3_RX_STA&0X7FFF;   //此次接收到的数据长度
      for(i=0;i<len;i++)
      {
         if(USART3_RX_BUF[i]==0X83)break;
         else if(USART3_RX_BUF[i]==0X84)
         {
            rval=3;
            break;
         }
      }
      if(i==len)rval=2;                    //没有找到同步字符
   }else rval=1;                           //接收超时错误
    USART3_RX_STA=0;                       //清除接收
   return rval;
```

```
}
//配置 S1216F8-BD_GPS/北斗模块波特率
//baud_id:0~8 , 对 应 波 特
率,4800/9600/19200/38400/57600/115200/230400/460800/921600
//返回值:0,执行成功;其他,执行失败(这里不会返回 0 了)
u8 S1216F8BD_Cfg_Prt(u8 baud_id)
{
   S1216F8BD_baudrate *cfg_prt=(S1216F8BD_baudrate
*)USART3_TX_BUF;
   cfg_prt->sos=0XA1A0;     //引导序列(小端模式)
   cfg_prt->PL=0X0400;      //有效数据长度(小端模式)
   cfg_prt->id=0X05;        //配置波特率的 ID
   cfg_prt->com_port=0X00;  //操作串口 1
   cfg_prt->Baud_id=baud_id;  ////波特率对应编号
   cfg_prt->Attributes=1;   //保存到 SRAM&FLASH
   cfg_prt->CS=cfg_prt->id^cfg_prt->com_port^cfg_prt->Baud_id^cfg
_prt->Attributes;
   cfg_prt->end=0X0A0D;     //发送结束符(小端模式)
   S1216F8BD_Send_Date((u8*)cfg_prt,sizeof(S1216F8BD_baudrate));/
/发送数据给 S1216F8BD
   delay_ms(200);           //等待发送完成
   USART3_Init(BAUD_id[baud_id]); //重新初始化串口 2
   return S1216F8BD_Cfg_Ack_Check();//这里不会反回 0,因为 UBLOX 发
回来的应答在串口重新初始化的时候已经被丢弃了.
}
//配置 S1216F8BD_GPS 模块的时钟脉冲宽度
//width:脉冲宽度 1~100000(us)
//返回值:0,发送成功;其他,发送失败.
u8 S1216F8BD_Cfg_Tp(u32 width)
```

```
{
    u32 temp=width;
    S1216F8BD_pps_width                *cfg_tp=(S1216F8BD_pps_width
*)USART3_TX_BUF;
    temp=(width>>24)|((width>>8)&0X0000FF00)|((width<<8)&0X00FF000
0)|((width<<24)&0XFF000000);//小端模式
    cfg_tp->sos=0XA1A0;          //cfg header(小端模式)
    cfg_tp->PL=0X0700;          //有效数据长度(小端模式)
    cfg_tp->id=0X65   ;              //cfg tp id
    cfg_tp->Sub_ID=0X01;          //数据区长度为 20 个字节.
    cfg_tp->width=temp;          //脉冲宽度,us
    cfg_tp->Attributes=0X01;  //保存到 SRAM&FLASH
    cfg_tp->CS=cfg_tp->id^cfg_tp->Sub_ID^(cfg_tp->width>>24)^(cfg_
tp->width>>16)&0XFF^(cfg_tp->width>>8)&0XFF^cfg_tp->width&0XFF^cfg_tp
->Attributes;     //用户延时为 0ns
    cfg_tp->end=0X0A0D;         //发送结束符(小端模式)
    S1216F8BD_Send_Date((u8*)cfg_tp,sizeof(S1216F8BD_pps_width));/
/发送数据给 NEO-6M
    return S1216F8BD_Cfg_Ack_Check();
}
//配置 S1216F8BDF8-BD 的更新速率
//Frep:（取值范围:1,2,4,5,8,10,20,25,40,50）测量时间间隔，单位为
Hz，最大不能大于 50Hz
//返回值:0,发送成功;其他,发送失败.
u8 S1216F8BD_Cfg_Rate(u8 Frep)
{
    S1216F8BD_PosRate *cfg_rate=(S1216F8BD_PosRate *)USART3_TX_BUF;
    cfg_rate->sos=0XA1A0;          //cfg header(小端模式)
    cfg_rate->PL=0X0300;          //有效数据长度(小端模式)
```

```
        cfg_rate->id=0X0E;          //cfg rate id
        cfg_rate->rate=Frep;        //更新速率
        cfg_rate->Attributes=0X01;      //保存到 SRAM&FLASH   .
        cfg_rate->CS=cfg_rate->id^cfg_rate->rate^cfg_rate->Attributes;
//脉冲间隔,us
        cfg_rate->end=0X0A0D;         //发送结束符(小端模式)
        S1216F8BD_Send_Date((u8*)cfg_rate,sizeof(S1216F8BD_PosRate));/
/发送数据给 NEO-6M
        return S1216F8BD_Cfg_Ack_Check();
    }
    //发送一批数据给 S1216F8BDF8-BD，这里通过串口 3 发送
    //dbuf：数据缓存首地址
    //len：要发送的字节数
    void S1216F8BD_Send_Date(u8* dbuf,u16 len)
    {
       u16 j;
       for(j=0;j<len;j++)//循环发送数据
       {
          while((USART3->SR&0X40)==0);//循环发送,直到发送完毕
          USART3->DR=dbuf[j];
       }
    }
```

这部分代码可以分为 2 个部分，第一部分是 NMEA-0183 数据解析部分，另外一部分则是 S1216F8BDF8-BD 协议控制部分。

NMEA-0183 协议解析部分，这里利用了一个简单的数逗号方法来解析。我们知道 NMEA-0183 协议都是以类似$GPGSV 的开头，然后固定输出格式，不论是否有数据输出，逗号是肯定会有的，而且都会以‘ *’作为有效数据的结尾，所以，我们了解了 NMEA-0183 协议的数据格式之后，就可以通过数逗号的方法，来解析数据了。本代码实现了对 NMEA-0183 协议的$GNGGA、$GPGSA、$GNGSV、$BDGSV、

$GNRMC 和$GNVTG 等 6 类帧的解析，结果存放在通过 gps.h 定义的 nmea_msg 结构体内。

S1216F8BDF8-BD 协议控制部分，此部分我们只实现了 S1216F8BDF8-BD 模组常用的 3 个配置：串口波特率设置、 PPS 输出脉冲宽度设置、输出频率设置。

串口波特率设置，通过函数 S1216F8BD_Cfg_Prt 实现，该函数可以设置模块的波特率。

PPS 输出脉冲宽度设置，通过函数 S1216F8BD_Cfg_Tp 实现，可以设置脉冲宽度（ 1us~100ms）。

输出频率设置，通过函数 S1216F8BD_Cfg_Rate 实现，该函数可以设置模块的测量输出频率，最快可以达到 20Hz 的测量输出频率。

最后 S1216F8BD_Send_Date 函数，用于发送一批设置好的数据给串口 2，完成对 GPS 模块的配置。

gps.h 里面的代码我们也不列出了，请大家参考对应源码。

最后我们打开 main.c 文件，里面的代码如下：

```
#include "system.h"
#include "SysTick.h"
#include "led.h"
#include "key.h"
#include "usart.h"
#include "tftlcd.h"
#include "gps.h"
#include "usart3.h"
#include "string.h"



u8 USART1_TX_BUF[USART3_MAX_RECV_LEN];                    //串口 1,发送缓存区
nmea_msg gpsx;                                            //GPS 信息
```

```
    __align(4) u8 dtbuf[50];                                    //打印缓存器
    const u8*fixmode_tbl[4]={"Fail","Fail"," 2D "," 3D "}; //fix mode
字符串

    //显示 GPS 定位信息
    void Gps_Msg_Show(void)
    {
        float tp;
        FRONT_COLOR=BLUE;
        tp=gpsx.longitude;
        sprintf((char            *)dtbuf,"Longitude:%.5f            %1c
",tp/=100000,gpsx.ewhemi);  //得到经度字符串
        LCD_ShowString(10,120,200,16,16,dtbuf);
        tp=gpsx.latitude;
        sprintf((char            *)dtbuf,"Latitude:%.5f            %1c
",tp/=100000,gpsx.nshemi);  //得到纬度字符串
        LCD_ShowString(10,140,200,16,16,dtbuf);
        tp=gpsx.altitude;
        sprintf((char *)dtbuf,"Altitude:%.1fm     ",tp/=10);
        //得到高度字符串
        LCD_ShowString(10,160,200,16,16,dtbuf);
        tp=gpsx.speed;
        sprintf((char *)dtbuf,"Speed:%.3fkm/h     ",tp/=1000);
        //得到速度字符串
        LCD_ShowString(10,180,200,16,16,dtbuf);
        if(gpsx.fixmode<=3)
        //定位状态
        {
            sprintf((char                                    *)dtbuf,"Fix
```

```
Mode:%s",fixmode_tbl[gpsx.fixmode]);
          LCD_ShowString(10,200,200,16,16,dtbuf);
        }
        sprintf((char                *)dtbuf,"GPS+BD                Valid
satellite:%02d",gpsx.posslnum);         //用于定位的 GPS 卫星数
        LCD_ShowString(10,220,200,16,16,dtbuf);
        sprintf((char                *)dtbuf,"GPS                Visible
satellite:%02d",gpsx.svnum%100);            //可见 GPS 卫星数
        LCD_ShowString(10,240,200,16,16,dtbuf);

        sprintf((char                *)dtbuf,"BD                Visible
satellite:%02d",gpsx.beidou_svnum%100);           //可见北斗卫星数
        LCD_ShowString(10,260,200,16,16,dtbuf);

        sprintf((char          *)dtbuf,"UTC          Date:%04d/%02d/%02d
",gpsx.utc.year,gpsx.utc.month,gpsx.utc.date);    //显示 UTC 日期
        LCD_ShowString(10,280,200,16,16,dtbuf);
        sprintf((char          *)dtbuf,"UTC          Time:%02d:%02d:%02d
",gpsx.utc.hour,gpsx.utc.min,gpsx.utc.sec);   //显示 UTC 时间
       LCD_ShowString(10,300,200,16,16,dtbuf);
    }



    int main()
    {
       u16 i,rxlen;
       u16 lenx;
       u8 key=0XFF;
```

```
    u8 upload=0;

    SysTick_Init(168);
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);  //中断优先级分组 分 2 组
    LED_Init();
    KEY_Init();
    USART1_Init(9600);
    TFTLCD_Init();           //LCD 初始化
    USART3_Init(38400);

    FRONT_COLOR=RED;
    LCD_ShowString(10,10,tftlcd_data.width,tftlcd_data.height,16,"PRECHIN");
    LCD_ShowString(10,30,tftlcd_data.width,tftlcd_data.height,16,"www.prechin.com");
    LCD_ShowString(10,50,tftlcd_data.width,tftlcd_data.height,16,"S1216F8-BD GPS Test");
    LCD_ShowString(10,70,tftlcd_data.width,tftlcd_data.height,16,"K_UP:Upload NMEA Data SW");
    LCD_ShowString(10,90,tftlcd_data.width,tftlcd_data.height,16,"NMEA Data Upload:OFF");

    if(S1216F8BD_Cfg_Rate(5)!=0)    //设置定位信息更新速度为 5Hz,顺便判断 GPS 模块是否在位.
    {
        LCD_ShowString(10,120,200,16,16,"S1216F8-BD              GPS Setting...");
```

```
        do
        {
            USART3_Init(9600);              //初始化串口 3 波特率为 9600
            S1216F8BD_Cfg_Prt(3);           //重新设置模块的波特率
为 38400
            USART3_Init(38400);             //初始化串口 3 波特率为 38400
            key=S1216F8BD_Cfg_Tp(100000);   //脉冲宽度为 100ms
        }while(S1216F8BD_Cfg_Rate(5)!=0&&key!=0);//       配       置
S1216F8BDF8-BD 的更新速率为 5Hz
        LCD_ShowString(10,120,200,16,16,"S1216F8-BD     GPS     Set
Done!!");
        delay_ms(500);
        LCD_Fill(10,120,10+200,120+16,WHITE);//清除显示
    }
    while(1)
    {
        delay_ms(1);
        if(USART3_RX_STA&0X8000)     //接收到一次数据了
        {
            rxlen=USART3_RX_STA&0X7FFF; //得到数据长度
            for(i=0;i<rxlen;i++)USART1_TX_BUF[i]=USART3_RX_BUF[i];

            USART3_RX_STA=0;           //启动下一次接收
            USART1_TX_BUF[i]=0;            //自动添加结束符
            GPS_Analysis(&gpsx,(u8*)USART1_TX_BUF);//分析字符串
            Gps_Msg_Show();                //显示信息
            if(upload)printf("\r\n%s\r\n",USART1_TX_BUF);//发送接收
到的数据到串口 1
        }
```

```
            key=KEY_Scan(0);
            if(key==KEY_UP)
            {
                upload=!upload;
                FRONT_COLOR=RED;
                if(upload)LCD_ShowString(10,90,200,16,16,"NMEA      Data
Upload:ON ");
                else      LCD_ShowString(10,90,200,16,16,"NMEA      Data
Upload:OFF");
            }
            if((lenx%500)==0)
                led1=!led1;
            lenx++;
        }
    }
```

此部分代码比较简单， main 函数初始化硬件之后，通过 S1216F8BD_Cfg_Rate 函数判断模块是否在位，如果不在位，则尝试去设置模块的波特率为 38400，直到检测到模块在位为止。然后进入死循环，等待串口 3 接收 GPS/北斗数据。每次接收到 GPS/北斗模块发送过来的数据，就执行数据解析，数据解析后执行 GPS/北斗定位数据的显示，并可以根据需要（通过 K_UP 按键开启/关闭），将收到的数据通过串口 1 发送给上位机。

4 实验现象

将工程程序编译下载到开发板内并且将模块连接到开发板上，注意对于 STM32F4 开发板要将 RS232 模块的端子短接到 COM3M 测，这个在前面已经说过。可以看到 TFTLCD 上显示如下界面：

上图是我们的 GPS/北斗模块成功定位后的照片，可以得到当前地点的经纬度、高度、速度、定位模式、用于定位卫星数、可见卫星数和 UTC 日期时间等信息。此时我们的 PZ-S1216F8-BD GPS/北斗模块，用于定位的卫星达到 10 颗，可见的 GPS 和北斗卫星一共 17 颗。