文章编号:1005-8451(2005)04-0043-03

GPS 定位系统中数据读取和坐标的转换

李拥军
(长沙铁路总公司,长沙 410001)

摘 要:介绍了一种在 VC 下实现对 GPS 全球定位系统定位信息的接收以及其定位参数的提取和 WGS-84 的大地坐标转换为高斯平面坐标的方法。

关键词:全球定位系统;串口通讯;WGS—84 坐标;公式
中图分类号:TP39 文献标识码:A

Data reading and coordinate changing in GPS

LI Yong-jun
(Passanger Trasportation Department of Changsha Railway Co, Changsha 410001, China)

Abstract: It was introduced the method to receive the position information of GPS, extracted the position parameter, changed the WGS-84 ground coodrinate to plan coordinate based on VC.

key words: Global Positoning System; series communication; WGS-84 coordinate; formula

全球定位系统(GPS)是近年来开发的最具有开创意义的高新技术之一,其全球性、全能性和全天候性的导航定位、定时和测速优势必然会在诸多领域中得到越来越广泛的应用。在实际应用中,GPS 接收机输出的定位信息是通过 RS232 串口传递给计算机,计算机主程序需要将 GPS 定位信息进行判别并提取所需要的有用数据(如目标当前的经纬度坐标、海拔、速度和时间等)。由于 GPS 使用的坐标系 WGS-84 与我国采用的坐标系不同,因此还需要将经纬度坐标进行坐标变换使其适应当地坐标系,再将当前目标显示在电子地图上。本文就针对当前比较普及的 GPS,对其卫星定位信息的接收及其定位参数提取的实现和坐标转换的方法予以介绍。

1 定位信息的接收和提取

GPS 接收机主要由 GPS 接收天线、变频器、信号通道、微处理器、存储器以及电源等部分组成。GPS 接收机只要处于工作状态就会按照指令把接收并计算出的 GPS 导航定位信息(NEMA0183 语句)通过串口传送到计算机中。计算机从串口读取数据有多种方法,Windows 中提供了一个串口通讯控件(MSComm),MSComm 控件可以采用轮询或事件驱动的方法从端口获取数据。比较常用的事件驱动方法:有事件(如接收到数据)时通知程序。在程序中需要捕获并处理这些通讯事件。这样可以很简单地利用串口进行通讯。在使用它之前,应将控件加在应用程序的对话框上。然后再用 ClassWizard 生成相应的对象。

1.1 初始化串口

该控件有很多自己的属性,可以通过它的属性窗口来设置,也可以用程序设置。建议采用程序设置,这样更灵活。

```
if(m_ComPort.GetPortOpen())//设置串口配置信息前,先要关闭串口;
    m_ComPort.SetPortOpen(FALSE);
m_ComPort.SetCommPort(1); //指定使用的串口为 com1;
m_ComPort.SetInBufferSize(1024);//设置输入缓冲区的大小;
m_ComPort.SetOutBufferSize(512);//设置输出缓冲区的大小;
m_ComPort.SetInputMode(1); //设置输入方式为二进制方式;
m_ComPort.SetSettings("9600,n,8,1");//设置波特率等参数;
m_ComPort.SetRThreshold(1);//设置为每接收一
```

收稿日期:2004-09-23
作者简介:李拥军,工程师。

个字符就触发一个OnComm事件;

```
m_ComPort.SetInputLen(0); //设置为0时程序将读取缓冲区的全部字符;
if(!m_ComPort.GetPortOpen())//打开串口;
m_ComPort.SetPortOpen(TRUE);
```

1.2 定位信息的接收

在设置通讯口后,采取效率比较高的事件触发方式完成对GPS定位信息的接收。在使用事件驱动法设计程序时,每当有新字符到达,或端口状态改变,或发生错误时,MSComm控件将触发On-Comm事件,而应用程序在捕获该事件后,通过检查MSComm控件的CommEvent属性可以获知所发生的事件或错误,从而采取相应的操作。这种方法的优点是程序响应及时,可靠性高。代码如下:

```
void CCommDlg:: OnCommCom1()
{
  // TODO: Add your control notification handler code here
   VARIANT m_input1;
   COleSafeArraym_input2;
   LONG Length,i;
   BYTEdata1[1024];
   if(m_ComPort.GetCommEvent()==2)    //comEvReceiv事件,有数据到达
   {
     m_input1= m_ComPort.GetInput(); //读缓冲区
     m_input2=m_input1;
     Length=m_input2.GetOneDimSize(); //接收缓冲区的字符数目
     for(i=0;i<Length;i++) //将数据转换为BYTE型数组
       m_input2.GetElement(&i,data1+i);
     AddToData1(data1,Length);//自定义函数,将接收的字符存入缓存
   }
}
```

1.3 定位信息的提取

前面的代码只负责从串口接收数据并将其放置于缓存,这些信息必须通过程序分解处理,才能提取出有用的定位信息数据。对GPS进行信息提取必须首先了解信息的数据格式,GPS接收机使用的是NMEA-0183的传输协议,NMEA-0183的信息格式一般如下所示:$aaaaa,df1,df2,....[CR][LF]

所有的信息由$开始,以换行结束,紧跟着$后的5个字符解释了信息的基本类型,多重的信息之间用逗号隔开。不需要了解NMEA-0183通讯协议的全部信息,仅需要从中挑选出所需要的那部分定位数据,最常见的几种类型为:GPGGA(GPS定位数据)、GPGLL(地址位置和经纬度)、GPZDA(日期和时间)、GPVTG(方位角对地速度)、GPRMC(GPS推荐的最短数据,有经纬度、日期和时间、天线移动速度)等。对于通常的情况,定位数据如经纬度、速度、时间等均可以从"$GPRMC"帧中获取得到,该帧的结构及各字段释义如下:

$GPRMC,<1>,<2>,<3>,<4>,<5>,<6>,<7>,<8>,<9>,<10>,<11>*hh<CR><LF>

<1> 当前格林尼治时间(UTC),格式为 hhmmss

< 2 > 状态字,A:定位成功;V:目前没有定位

< 3 > 纬度格式为ddmm.mmmm

< 4 > 纬度的属性,南半球为 N,北半球为S

< 5 > 经度格式为dddmm.mmmm

< 6 > 经度的属性,东半球为 E,西半球为W

< 7 > 天线移动速度从 000.0到999.9 节

< 8 >相对地面方向, 000.0到 359.9度

< 9 >当前日期(UTC时间),格式为:ddmmyy

<10> 磁偏角 000.0到180.0度

<11> 磁偏方向 E or W

< * > 校验和标志

<hh> 表示校验和

在处理缓存数据时一般是通过搜寻"$GPRMC"来判断是否是一帧数据的帧头,在对帧头的类别进行识别后再通过对逗号个数的计数来判断出当前正在处理的是哪一种参数,并作出相应的处理。如需要从其他类型帧获取数据,处理方法是完全类似。下面对缓存Data中的数据进行解帧处理的主要代码,本文只提取时间、日期、经度、纬度、分别保存在CString型变量m_sTime、m_Data、m_sPositionY和m_sPositionX中。

```
CString   m_sTime,m_Data,m_sPositionY,m_sPositionX;
int CountID;//逗号计数器
BOOL  Flag;//帧头是"$GPRMC"时设为TRUE;
for(inti=0;i<Length;i++)
{
```

```
if(Data[i]=='$')//帧头
{
  if((Data[i+1]=='G')&&(Data[i+2]=='P')&&(Data
[i+3]=='R')&&
  (Data[i+4]=='M')&&(Data[i+5]=='C'))
  {
    Flag=TRUE;  //  帧头为"$GPRMC"
          CountID =0;//计数器清零
   }
}
if(Flag)
{
       if(Data[i]==',')  // 逗号计数
                CountID ++;
   else
   {
      switch(CountID)
     {
       case 1: //提取出时间
       m_sTime+=Data[i];
      break;
       case 2: //判断数据是否有效(有效时为A)
      if(Data[i]='A')
           GPSParam[m_nNumber].m_bValid=
TRUE;
     break;
       case 3: //提取出纬度
      m_sPositionY+=Data[i];
      break;
       case 5: //提取出经度
      m_sPositionX+=Data[i];
      break;
       case 9: //提取出日期
      m_sDate+=Data[i];
      break;
  default:
      break;
}
if((Data[i]='0x0D')&&(Data[i+1]='0x0A'))//帧尾
   Flag=FALSE;
       }
      }
}
```

2　坐标变换

目前 GPS 所使用的坐标系统基本都是 WGS－84 坐标系，而我们使用的地图大部分都属于 54 北京坐标系或 80 西安国家大地坐标系。要使 GPS 定位信息正确地显示在数字地图上，必须将 GPS 定位结果即大地坐标（L，B）转换为本地高斯平面坐标（x，y）。一般要通过两步转换，首先将 WGS－84 的大地坐标（L，B）转换为对应于 WGS－84 椭球的高斯坐标平面（x84，y84），然后再经过平面坐标转换，将高斯平面坐标（x84，y84）强制符合到本地高斯平面坐标系统，以实现 GPS 定位信息在数字地图中的正确匹配。由已知的参心大地坐标系中点的大地纬度和大地经度（B，L），求相应的高斯投影直角坐标（x，y）的公式，称为高斯投影正算公式。

设参考椭球的长半轴为 a，第 1 偏心率为 e，则高斯投影正算公式为：

$x=X_0^B+Ntm_0^2/2+(5-t^2+9n^2+4n^4)Ntm_0^4/24+(61-58t^2+t^4)Ntm_0^6/720$

$y=Nm_0+(l-t^2+n^2)Nm_0^3+(5-18t^2+t^4+14n^2 58n^2t^2)Nm_0^5/120$

式中：

$X_0^B=C_0B-\cos B(C_1\sin B+C_2\sin^3 B+C_3\sin^5 B)$

$t=\tan B$

$l=L-L_0$

$m_0=l\cos B$

$N=e/(1-e^2\sin^2 B)^{1/2}$

$n^2=e^2\cos^2 B/(1-e^2)$

L、B 为转换前的经纬度坐标；x、y 为转换后的高斯坐标；L_0为投影带的中央经线坐标；C0、C1、C2、C3 为与点位无关而只与椭球参数有关的常数。

3　结束语

为提高定位精度，通常采用差分 GPS 技术。由 GPS 卫星导航测到 GPS 接收机位置坐标数据、前进的方向都与实际行驶的路线轨迹存在一定误差，为修正这两者的误差，与地图上的路线统一，需采用地图匹配算法，对行驶的路线与电子地图上道路误差进行自动修正，得到在电子地图上的正确位置。GPS 导航系统与电子地图、无线电通信网络及计算机管理信息系统相结合，可以实现机车跟踪和管理等许多功能，对铁路安全生产具有积极的意义。