文章编号:1001 - 9081(2006)02 - 0505 - 03

Windows CE 系统中卫星通信的实现

李韶芳,朱宝山,文 江,曹 闻
(信息工程大学 测绘学院,河南 郑州 450052)
yyx0712@ sina. com)

摘 要:介绍了 Windows CE 系统下嵌入式设备与卫星信号接收机之间进行通信的一般方法,详细设计了一个实用的多线程实时通信解决方案,并以 GPS 系统的通信格式为例实现了一种提取其定位信息的方法。实验证明,该方法在 PDA 上能够实时、正确地获取 GPS 的定位信息,从而证明该方案是可行的、正确的。

关键词:卫星通信;嵌入式;多线程;信息提取
中图分类号: TP393 **文献标识码**:A

Solution of satellite communication in Windows CE

LI Shao-fang, ZHU Bao-shan, WEN Jiang, CAO Wen
(*Institute of Surveying and Mapping, Information Engineering University, Zhengzhou Henan* 450052, *China*)

Abstract: A common method of communication between embedded devices and satellite receivers was introduced. To resolve the problem of real-time communication, this paper designed a practical multithreading scheme and realized a method which can extract information from satellite signals, using the communication format of GPS. The result of the test in PDA proves that the method can correctly extract the position information of the GPS signals, and can satisfy the requirement of real-time communication.

Key words: satellite communication; embedded; multithreading; information extracting

1 嵌入式设备与卫星接收机的通信

通常 GPS 定位信息接收系统主要由 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。由于 GPS 定位信息内容较少,因此一般使用 RS-232 串口将定位信息(NEMA0183 语句)从 GPS 接收机传送到嵌入式设备中进行信息提取处理。实现串口通信有多种方法[1],本文使用 Win32 API 函数对其进行编程处理。

在 Windows CE 系统下不允许直接对硬件端口进行控制操作,所有的端口均被视为“文件”,因此在对串口进行侦听之前需要用以下方式来打开串口:

```
HANDLE hCom = CreateFile("COM1:", GENERIC_READ |
    GENERIC_WRITE, 0, NULL, OPEN_EXISTING, NULL,
    NULL);
```

如果该函数成功返回,hCom 即为串行端口 COM1 的句柄。注意以下两点与 VC++6.0 中有所不同[2]:

1) 第一个参数是串口名的字符串,必须加“:”;

2) 第 6 个参数不能为 FILE_FLAG_OVERLAPPED,只能是 NULL,这是因为该函数在 CE 系统中不支持重叠 I/O。尽管如此,我们仍可以使用多个线程来执行同样效果的重叠操作。因此串口信息的发送和接收必须以多线程的方法加以实现。

对串口的配置主要是对串口状态参数设置、超时结构设置和串口事件掩码的设置。串口状态参数可以由一个 DCB 结构获得。常用的可能更改的参数一般是数据传输率、数据位、停止位和校验位。串口的超时参数可以通过一个 COMMTIMEOUTS 超时结构来设置。和 DCB 结构一样,超时参数的设置也是先获得串口的超时结构,再更改其中需要修改的值,最后再把结构赋值回串口。串口响应什么事件是通过函数 SetCommMask 设置串口事件掩码来实现的,在这里设置为 EV_RXCHAR,它表示串口接收到一个字符就激活串口事件,使用函数 WaitCommEvent 即可接收到该事件。

串口的读写使用 ReadFile 函数和 WriteFile 函数。由于 CE 下串口不支持重叠 I/O,因此它们的最后一个参数(指向异步结构变量的指针)都必须为空(NULL)。

串口的关闭比较简单,使用 CloseHandle( )函数即可。

2 多线程解决方案

在卫星导航系统与 GSM 网络联合使用的情况下,串口不仅要接收卫星数据,还要向外发送数据。

由于串口数据的接收和发送是不规律的,所以可能会有在短时间内大量数据高速涌入的情况发生,再加上定位信息的解析计算量比较大,嵌入式设备的性能又比较差,此时系统可能来不及处理所有信息,从而导致信息丢失。

鉴于以上考虑,本文设计了三个辅助线程和两个数据缓冲区来解决这一问题。两个数据缓冲区分别是数据输入缓冲区和数据输出缓冲区。输入缓冲区中存储接收到的但未解析的串口数据;输出缓冲区储存已生成好的但未发送的数据。三个辅助线程分别是监视线程、解析线程和发送线程。监视线程用于监视串口状态,实时、完整地接收数据;解析线程把

收稿日期:2005 - 08 - 30;**修订日期**:2005 - 11 - 08

作者简介:李韶芳(1981 -),男,山西晋城人,硕士研究生,主要研究方向:数字图像处理、地理信息系统; 朱宝山(1964 -),男,河南平顶山人,副教授,主要研究方向:图像处理与应用; 文江 (1977 -),男,四川康定人,硕士研究生,主要研究方向:数字图像处理、地理信息系统、模式识别; 曹闻(1979 -),男,山东泰安人,助教,主要研究方向:遥感图像处理、地理信息系统.

串口数据解析为定位信息;发送线程把数据输出缓冲区中的数据发送至串口。它们的逻辑关系如图 1 所示。

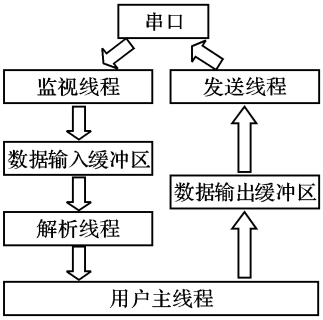

图 1 多线程流程关系图

当一个线程从某个共享资源中读取数据的时候,可能另外一个线程却要往该资源写入数据。为了不使读写操作相冲突,本文使用三个临界区变量来处理串口、输入缓冲区和输出缓冲区这类被多个线程共享的资源的使用协调问题,这里分别命名为 ComSync, InBufSync 和 OutBufSync。访问它们之前,要锁定临界区变量(EnterCriticalSection),以保证在本线程访问该资源时,其他线程不能更改该资源,在访问操作完成之后还必须要解锁临界区变量(LeaveCriticalSection)。

同时,为了协调各线程之间不同操作的执行顺序[3],还定义了 6 个事件,它们分别是:解析事件、解析完成事件、发送事件、发送完成事件、发送线程结束事件、解析线程结束事件。事件的激活使用 SetEvent 函数,关闭使用 ResetEvent 函数。

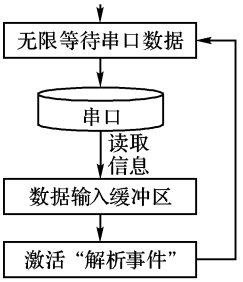

图 2 监视线程工作流程图

数据的接收由监视线程完成。其流程如图 2 所示。监视线程使用函数 WaitCommEvent 等待串口数据的到来。为了使监视线程占用共享资源的时间尽量少,在获得串口的使用权后,可将串口数据先复制到一个临时变量中,然后立刻放弃串口的使用权。同理,在获得输入缓冲区的使用权并把临时变量中的数据写入到输入缓冲区后,也要立刻放弃其使用权。然后,监视线程便激活解析事件,通知解析线程开始工作,监视线程则继续下一次等待串口数据的到来。若用户主线程要结束监视线程,可使用函数 CloseHandle 将其强制结束。

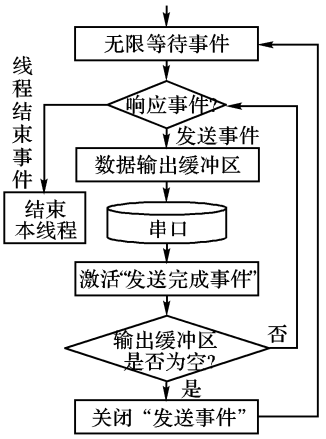

图 3 发送线程工作流程图

数据的发送由发送线程来完成。其流程如图 3 所示。发送线程使用函数 WaitForMultipleObjects 来等待发送事件或线程结束事件,其中前者的优先级较高。和监视线程一样,访问输出缓冲区和串口之前都应先锁定临界区变量,访问后再解锁临界区变量。发送线程如果等到发送事件后,则从输出缓冲区中取出数据,并使用函数 WriteFile 将该数据发送到串口,然后检查输出缓冲区是否为空,若是则关闭发送事件,等待下一次的发送事件,若否则继续把输出缓冲区中的数据发送至串口。如此循环直至输出缓冲区为空,然后再关闭发送事件。若用户主线程要结束发送线程,只需激活发送线程结束事件,发送线程会先将输出缓冲区中的数据全部发送到串口,然后再响应结束事件,退出循环,结束自己。

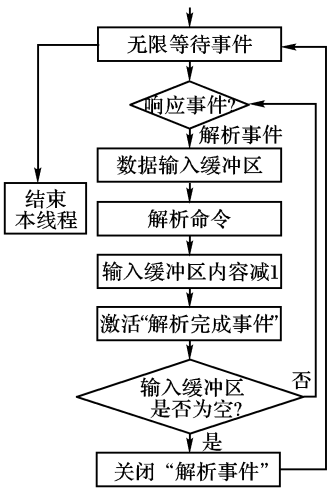

图 4 解析线程工作流程图

数据的解析由解析线程来实现。其流程如图 4 所示。解析线程等待到监视线程激活的解析事件后,便从数据输入缓冲区取得原始的串口数据,根据通讯格式将其中的信息提取出来,并激活解析完成事件以通知主线程接收信息,然后判断输入缓冲区是否为空,若是则关闭解析事件,等待下一次解析事件的激活,若不是则从输入缓冲区中取出下一条命令进行解析,直至缓冲区为空。解析线程的结束和发送线程一样,由用户主线程激活解析线程结束事件,解析线程会先将输入缓冲区中的数据全部解析完,然后再响应线程结束事件,退出循环,结束自己。

三个辅助线程的主循环分别如下。

1) 监视线程的主循环

```
While(监视线程未被结束){
  使用函数 WaitCommEvent 无限等待串口事件;
  If(是接收字符事件){
    锁定串口临界区变量 ComSync;
    使用 ClearCommError 获得串口数据长度;
    If(串口数据长度大于 0){
      使用 ReadFile 读取串口数据到临时变量;
      解锁串口临界区变量 ComSync;
      锁定输入缓冲区临界区变量 InBufSync;
      将临时变量复制到输入数据缓冲区中;
      解锁输入缓冲区临界区变量 InBufSync;
      激活解析事件;
    }
  }
  Else
    解锁串口临界区变量 ComSync;
  }
```

2) 发送线程的主循环

```
While(发送线程结束标志为假){
  使用 WaitForMultipleObjects 函数无限等待两个事件的发生;
  If(是发送事件)
    If(输出缓冲区不为空){
      锁定变量 OutBufSync;
      将输出缓冲区中数据复制到临时变量中;
      解锁 OutBufSync;
      锁定串口临界区变量 ComSync;
      使用 WriteFile 函数把临时变量中数据发送至串口;
      解锁串口临界区变量 ComSync;
      激活发送完成事件;
    }
  Else
    关闭发送事件;
  Else
    设置监视线程结束标志为 TRUE;
}
```

3) 解析线程的主循环

```
While(解析线程结束标志为假){
  使用 WaitForMultipleObjects 函数无限等待两个事件的发生;
  If(是解析事件){
    If(输入缓冲区不为空){
      锁定串口临界区变量 ComSync;
```

```
            使用 ReadFile 函数把串口数据复制到临时变量中;
            解锁串口临界区变量 ComSync;
            锁定变量 InBufSync;
            将临时变量中数据复制到输入缓冲区中;
            解锁 InBufSync;
            激活解析完成事件;
          }
          Else
            关闭解析事件;
        }
        Else
          设置解析线程结束标志为 TRUE;
    }
```

3　定位信息的提取

定位信息的提取由解析线程来完成。下面以 GPS 的 NEMA0183 协议语句为例介绍定位信息的提取方法。

GPS 接收机只要处于工作状态就会不断地把接收并计算出的 GPS 导航定位信息通过串口传送到 Windows CE 系统中。前面的代码只负责从串口接收数据并将其放置于缓存,在没有进一步处理之前缓存中是一长串字节流,这些信息在没有经过分类提取之前是无法加以利用的。因此,必须通过程序将各个字段的信息从缓存字节流中提取出来,将其转化成有实际意义的,可供高层决策使用的定位信息数据。同其他通讯协议类似,对 GPS 进行信息提取必须首先明确其帧结构,然后才能根据其结构完成对各定位信息的提取。对于本文所使用的 GARMIN GPS 天线板,其发送到计算机的数据主要由帧头、帧尾和帧内数据组成,根据数据帧的不同,帧头也不相同,主要有“$ GPGGA”、“$ GPGSA”、“$ GPGSV”以及“$ GPRMC”等。这些帧头标识了后续帧内数据的组成结构,各帧均以回车符和换行符作为帧尾标识一帧的结束。对于通常的情况,我们所关心的定位数据如经纬度、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[4]:

$ GPRMC,<1>,<2>,<3>,<4>,<5>,<6>,<7>,<8>,<9>,<10>,<11>,<12>,<13>CR LF

<1> UTC_TIME 24 小时制的标准时间,格式为“hhmmss”;

<2> 通信状态,A 为有效位置,V 为非有效接收警告,即当前天线视野上方的卫星个数少于 3 颗;

<3> 纬度值,格式为“dddmm.mmmm”,ddd 表示度(o),mm.mmmm 表示分(′);

<4> 标明南北半球,N 为北半球、S 为南半球;

<5> 经度值,格式为“dddmm.mmmm”,ddd 表示度(o),mm.mmmm 表示分(′);

<6> 标明东西半球,E 为东半球、W 为西半球;

<7> 地面上的速度,范围为 000.0 ~ 999.9,单位为海里/小时;

<8> 真实航向,范围为 000.0 ~ 359.9,单位为度(°);

<9> UTC_DATE 标准日期,格式为 ddmmyy,年份应该在 1970 ~ 2040 之间;

<10> 地磁偏移量,范围为 000.0 ~ 180.0,单位为度(°);

<11> 地磁偏转方向,为 E 表示东,为 W 表示西;

<12> 系统定位模式;

<13> 校验和域,格式为“..xx”,xx 表示校验和。

至于其他几种帧格式,除了特殊用途外,平时并不常用,虽然接收机也在源源不断地向主机发送各种数据帧,但在处理时一般先通过对帧头的判断而只对“$ GPRMC”帧进行数据的提取处理。如果情况特殊,需要从其他帧获取数据,处理方法与之也是完全类似的。由于帧内各数据段由逗号分割,因此在处理缓存数据时一般是通过搜寻 ASCII 码“$”来判断是否是帧头,在对帧头的类别进行识别后再通过对所经历逗号个数的计数来判断出当前正在处理的是哪一种定位导航参数,并作出相应的处理。下面就是对缓存 pBuf 中的数据进行解帧处理的主要代码,本文在此只提取地理坐标和时间。

```
For(i = 0;i < nDataLength;i++){
  If( pBuf[i]是"$"字符){                          //找到帧头
    nCommaID = 0;                                  //逗号计数器
    If( pBuf[i]是",")                              //找到逗号
      nCommaID++;
    Else{
      Switch(nCommaID){
      Case 1:                                      //读取时间
        依次把 pBuf 中 i ~ i +6 的内容转换为字符串,并加入到
        字符串 szTime 中;
        i = i + 6;
        break;
      Case 3:                                      //读取纬度
        依次把 pBuf 中 i ~ i +10 的内容转换为字符串,并加入
        到字符串 szLatitude 中;
        i = i + 10;
        break;
      Case 5:                                      //读取经度
        依次把 pBuf 中 i ~ i +10 的内容转换为字符串,并加入
        到字符串 szLongitude 中;
        i = i + 10;
        break;
      Default:
        break;
      }
    }
  }
}
```

现在已将所需信息提取到内存,即时间、日期以及经纬度分别保存在字符串型变量 szTime、szLatitude 和 szLongitude 中。将提取到的数据再稍作变换即可交给用户主线程了。

4　结语

通过本文的设计方法可以将 GPS 定位导航信息从 GPS 接收机完整接收,通过对定位参数的提取可将其应用于其他高层应用系统(如移动导航系统)中。本文设计的多线程解决方案经 eMbedded Visual C++3.0 编译通过,程序在型号为 HP hx2110 的 PDA 上测试,运行正常,并且达到了 PDA 从 GPS 接收机实时获取定位信息的要求。

参考文献:

[1] 龚建伟,雄光明. Visual C++/Turbo C 串口通信编程实践[M]. 北京:电子工业出版社,2004.

[2] BOLING D. Windows CE 程序设计[M]. 北京博彦科技发展有限公司,译. 北京:北京大学出版社,1999.

[3] BEVERIDGE J, WIENER R. Multithreading Applications in Win32 [M]. 候捷,译. 武汉:华中科技大学出版社,2002.

[4] 谭思亮,邹超群,等. Visual C++ 串口通信工程开发实例导航[M]. 第 1 版. 北京:人民邮电出版社,2003.