VxWorks 下声纳基阵稳定控制系统的软件实现

谢李李 乐慧康 陈义平 哈尔滨工程大学自动化学院(150001)

Abstract

This paper introduce the main communication manners among tasks in Vx Works at first.second.we introduce the main function,the tasks partion of the sonar array control system,and how to carry out this tasks and the communication manners among this tasks in VxWorks,.Third,we introduce the communication manners between Vx Works tasks and Zinc tasks.forth, analyze and compare the performance of the different communication manners.At last we do some summarization of this Communication manners.

Keywords:sonar arry,VxWorks,Zinc,communication manners

摘 要

介绍了 VxWorks 操作系统中任务间的通信,声纳基阵稳定控制系统的结构,主要功能,软件任务的划分,在 VxWorks 中这些任务的实现及任务间的通信。同时介绍了 VxWorks 任务与 Zinc 任务的通信及实现,并对 VxWorks 任务间通信机制的性能作了比较,分析和总结。

关键词:声纳基阵,VxWorks,Zinc,通信机制

1 声纳基阵控制

声纳基阵控制稳定系统是在猎雷艇航行并伴有纵摇、横摇与艏摇运动的情况下为声纳基阵提供稳定的姿态和方位俯仰角运动。声纳基阵包括识别阵和探测阵上下两个部分,它们都包含了三个自由度的控制稳定系统,从这两个基阵的结构和控制原理来说都基本相同,因此仅就识别阵的控制稳定系统的设计为例来说明。识别阵控制稳定系统由四个独立的分系统构成,分别是坐标变换计算机单元、方位伺服系统、横倾伺服系统和俯仰伺服系统。主要完成两种基本功能:一是根据舰艇的摇摆信号和指令角信号算出基阵绕自身三个轴应转动的 α、β、γ 角及其角速度;二是用三个伺服系统将基阵转到相应的 α、β、γ 位置上,并保持给定的精度。结构如图 1。

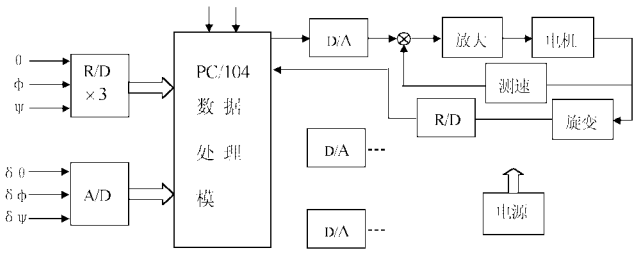

图 1 声纳基阵姿态稳定系统结构图

为了控制声纳基阵,在系统中必须是实时解算坐标变换,将舰船姿态角分解到基阵坐标系中。由于数据较多,且坐标变换都是完成三角函数和反三角函数的计算,处理比较复杂,所以采用 PC/104 作为处理器。PC/104 要完成姿态角的 A/D 采样,指令角的串行接收,然后进行坐标变换并将计算结果(α,β,γ)发送到后面的控制回路。

声纳基阵稳定控制系统软件部分要完成的主要功能是首先进行数据采样,然后对这些数据进行处理,将数字信号转化为角度和弧度信号,并将角度信息显示到屏幕上,再接收串口的命令角输入,进行坐标变换,得到期望的角度值,最后将期望值送到控制回路。另外,要求采样周期为 1ms,屏幕刷新周期为 1s。

2 基阵稳定控制系统任务的划分和任务间通信方式的实现

2.1 VxWorks 中的延迟和定时

系统要求每 1ms 进行一次数据采样,1s 进行一次屏幕刷新,这都是周期性的业务,需要用到延迟。在 VxWorks 中,常用的延迟有三种:直接调用函数 taskdelay()、WatchDog 和辅助时钟。

taskdelay()是最简单的延时方法,他的单位是 tick。所以其延时精度并不高,但对于延时 10ms 以上的系统足够了。利用 taskdelay(),可以将调用的任务从就绪态转到睡眠态但是不能用于中断服务程序中。特别的,可以通过调用 taskdelay(0),将 CPU 交给系统中其他相同优先级的任务。

在 VxWorks 中,看门狗定时器作为系统时钟中断服务程序的一部分来维护。因此,与看门狗定时器相联系的函数运行在系统时钟中断级,延时单位为 tick。如果应用程序需要多个看门狗函数,使用 wdCreate( )为每个需求产生独立的看门狗 ID。因为对于给定的看门狗 ID,只有最近的 wdStart()有效。利用看门狗定时器,调用的任务不会被阻塞:因为 wdStart()调用是立即返回的。但他也一般只适用于延时 10ms 以上的系统。

要想达到 1ms 甚至 μs 级的延时就需要使用辅助时钟。cpu 上至少有两个 Timer,Timer1 用于系统时钟,处理系统级任务,时钟频率一般不能太高。Timer2 用于辅助时钟,利用它可以得到精确的延时。

2.2 周期性采样和屏幕刷新任务的实现

为提高精度屏幕刷新用“看门狗”定时器来实现。由于“看门狗”定时器对应的 C 调用过程执行优先级高,因此不适合在此调用过程中对业务进行处理,因为这样会影响具有高优先级的任务的运行。可以在另一个任务中对业务进行处理,而 Watchdog 只是周期性地唤醒这个任务:

```
void renovationWatchdog()   //task1,spawn renovation task each 1s
   { event.type=RENAVATION_EVENT;
ZafEventManager->Put(event,Q_BEGIN);
wdStart (wdIdRenovation,renovation_delay, (FUNCPTR)renovation-Watchdog,1);
 } //在 zinc 队列里放入一个事件,唤醒屏幕刷新任务
```

“看门狗”定时器一般只能实现 10ms 以上的定时,这里 1ms 的采样使用了 Vxworks 的辅助时钟,用 sysAuxClkCon-

nect((FUNCPTR)readSpawn, 0)函数来实现,由于与他相连的程序也属于系统级时钟中断,同样具有太高的优先级,所以和看门狗一样,只用 readSpawn 每 1ms 唤醒采样程序。具体实现如下:

```
  void readSpawn()
  {   semGive(semIdForRead);
} //释放一个信号量,唤醒读任务
    void reading()     //readingTask
     {   .....
sysAuxClkDisable();/* Disable system aux clock     */
  sysAuxClkConnect((FUNCPTR) readSpawn, 0);
   sysAuxClkRateSet(100);   /* Set system  aux clock rate    */
   sysAuxClkEnable();
   for(;;)
    {  semTake  (semIdForRead,WAIT_FOREVER);//waiting  for
watchdog giving semphore
       semGive(semIdDeal);
      semTake (semIdDeal,WAIT_FOREVER);    //信号量 semId-
Deal,保护全局变量
          .....   //reading
         semGive(semIdDeal);
         return;}
        }
```

采样完成后,要把它们分别转换成角度和弧度信号,并且判断其有效性;而坐标变换都是三角函数和反三角函数的计算,他们的计算量都比较大,所以又把它们分成独立的两个任务:Dealing,进行角度和弧度变换,并判断数据的有效性;Coordinate,用于坐标变换。在这两个任务中,都要用到一些全局变量,为了确保这些全局变量在这个时刻不被其他的任务改变,用了信号量 semIdDeal、semIdData 来保护这些全局变量。经过处理的数据最后还要变成数字信号送到控制系统,这就又有了一个任务 Sending。

在主程序中发起各任务并启动"看门狗"定时器:

```
  mainTask()
   {    .....
     tIdReading =taskSpawn  (" treading" ,100,VX_SUPERVI-
SOR_MODE,stackSize,
(FUNCPTR) reading,0,0,0,0,0,0,0,0,0,0);
    tIdDealing=taskSpawn("tdealing",110,VX_SUPERVISOR_MODE,
stackSize,
(FUNCPTR) dealing,0,0,0,0,0,0,0,0,0,0);
tIdCoordinate =taskSpawn  (" tcoordinate" ,120,VX_SUPERVI-
SOR_MODE,stackSize,
(FUNCPTR) coordinate,0,0,0,0,0,0,0,0,0,0);
tIdSending =taskSpawn  (" tsending" ,130,VX_SUPERVISOR_MODE,
stackSize,
(FUNCPTR) sending,0,0,0,0,0,0,0,0,0,0);
wdStart(wdIdRenovation,renovation_delay,
(FUNCPTR)renovationWatchdog,1); //startup renovationWatchdog
task each 1s.
          .....
     }
```

2.3 Zinc 与外部任务的通信

在这个项目里,有两个地方涉及到图形处理,周期性的刷新屏幕,当数据无效时要立即在屏幕上显示。因为 VxWorks 本身并不支持图形显示,图形显示是用另一个相对独立的软件包 Zinc 来实现的。由于 Zinc 的相对独立,而且,在一个实时系统中 GUI 任务应该运转在一个足够低的优先级上,以便系统能够及时响应外部事件,所以,一般不在外部任务里直接处理 GUI,而是在一个专用的 GUI 任务里执行所有的 GUI 处理。但是 Zinc 本身是不可重入的,因此不能直接在外部任务里调用 Zinc 函数。当然也可以通过编译和处理使其变为可重入的,可是这样会加大系统开销,付出较大的代价,所以在这个项目中采用了 Zinc 入口点的方式来通信,即用 ZafEventManager::Put () 函数在 Zinc 事件队列上放置一个事件。当事件队列为空时,Zinc 任务挂起,这样可不浪费 CPU 时间。

另外,也可用消息队列来实现外部任务和 Zinc 之间的通信,只是要注意,这里的消息队列只能是单向的,而且要用一个 derived 设备来监听这个消息队列。

```
//derived 设备的构造函数
  Communicator::Communicator(ZafDeviceState _state) :
              ZafDevice(E_MOUSE, _state), mailToSend(false)
                  { installed = true; }
  //derived 设备的析构函数
  Communicator::~Communicator(void)
                  { installed = false; }
 //该类的轮询函数。事件管理器每次在 ZAF 事件队列中检查新消息
的时候,为每个附着  的设备调用此轮询函数。
              void Communicator::Poll(void)
             { if (!installed || DeviceState() == D_OFF)
                  return;
 //检查客户消息队列,如果有一个消息的话,就邮寄给 ZAF 的队列
              char queueMessage[MAX_MSG_LEN];
            if (msgQReceive(mesForFailtureId, queueMessage,
                  MAX_MSG_LEN, NO_WAIT) != ERROR)
               { ZafEventStruct qEvent(RECEIVE_MAIL);
                 qEvent.text=strdup(queueMessage);
                 zafEventManager->Put(qEvent); }
```

3 结束语

从总的来讲,如果考虑平均时间,信号量机制要优于消息队列。信号量机制在最好的情况下所需要的时间为 8μs,而最坏的情况下,比如任务在排队队列中时,为 48μs;消息队列最好的情况为 20μs,最坏的情况约为 50 多 μs。但对于实时系统,有限的响应时间是需要考虑的最重要的因素,无论选用什么样通信机制,必须考虑在最坏情况下的延迟,以及任务间切换引起的延迟。从实际应用中看到,在最差的情况下,信号量比消息队列好不了多少。

参考文献

1 罗国庆,等.VxWorks 与嵌入式软件开发.北京.机械工业出版社,2003

2 [美]Wind River 著.王金刚,高伟,苏琪,丁大尉,姜平译.VxWorks 程序员指南.北京.清华大学出版社,2003

3 [美]Wind River .Zinc Programmer's Guide.1996

4 [美]Wind River .VxWorks Programmer's Guide.1996

[收稿日期:2004.7.7]