基于 VB 的多线程实时测控系统的设计与实现

高春雷,高新陵

(河海大学水利水电工程学院,江苏 南京 210098)

摘要:介绍在 VB 环境下基于多线程的压力机床测控系统的设计与实现过程,对该测控系统的技术特点、软件设计方法进行了具体论述,并给出在 VB 中用 ActiveX 技术实现多线程实时测控的系统框架.该系统已成功地应用于一大型汽车制造企业的生产过程.

关键词:Visual Basic;多线程;实时测控

中图分类号:TH122 **文献标识码**:A **文章编号**:1000-1980(2002)01-0049-05

发动机镶缸套压力机床是汽车发动机生产线的重要设备之一,对该压力机床工作过程的压力信号实时检测,是控制发动机镶缸套质量的重要手段,检测信号的分析结果可控制压力机床的运动,同时可为调整生产线前部相关工序提供参考.该数据测量系统需要对 4 路压力信号和对应的高度信号进行持续的在线实时测量,数据范围是压力值 0 ~ +65 kN、高度值 -100 ~ 180 mm,测量误差允许范围小于或等于 0.8%,测量频率为 4 s 之内完成一路信号的检测和处理.系统必须对 290 多个采样数据或基于采样数据的统计参数实现实时超界报警.

由于在采集界面下需要在较短的时间内完成数据采集、去噪声、数据整理与修复、压力数据实时图表显示、数据分析计算、超界实时报警以及数据存储与管理等任务,如果仅使用单线程模型来设计系统,就不能很好地完成数据采集和监控的任务,因此考虑采用多线程模型,利用多个线程分别完成各项任务.在多种可视化编程语言中,VB 因可以让开发者快速创建强大的应用程序而免除不必要的细节,已经在测控系统中得到了广泛的应用.但如何用 VB 开发基于多线程的测控系统尚有一定的难度.本文就 VB 如何用于多线程测控系统的软件设计作了较为系统的阐述,并给出了实现该系统的框架.

1 多线程 Visual Basic 应用程序的设计

1.1 线程概述

应用程序的一个运行实例称作进程,每个进程可以包括一个或多个线程.所有属于该进程的线程都共享进程的资源,如 CPU、地址空间以及对象句柄等.利用多线程机制可以实现进程内的各个子任务并行执行,从而提高系统的实时性能.

1.2 VB 中多线程应用程序的实现

VB 中的线程模型有两种主要形式:自由线程模型(Free Thread Model)和单元线程模型(Apartment Thread Model).在自由线程模型中,每一线程可访问全部进程的数据区域,而且所有线程共享应用程序的全局变量.这样的线程模型,优点是各线程之间通讯比较方便,缺点是不利于实现各线程之间的同步.在单元线程模型中,一个单元中可以有一个或多个线程,同一单元中的线程可以互相通讯,而不同单元之间的线程则不可以.这种线程模型有利于实现线程之间的同步,而不利于线程之间的通讯(主要是指不同的单元中封装的线程之间的通讯).总的来说,单元线程模型比自由线程模型能更安全些,因此在实际系统设计中运用较广泛.

VB6 可以创建多线程 ActiveX 组件,包括多线程 ActiveX EXE 组件和多线程 ActiveX DLL 组件.此外,VB6 也可以很方便地创建多线程普通应用程序,程序中可以选择不同的线程模型,但必须具备以下 3 个条件:

a. 应用程序必须是用 Thread Per Object 设置编译的 ActiveX EXE 服务器;

收稿日期:2000-12-29

作者简介:高春雷(1973—),男,江苏武进人,硕士研究生,机械设计及理论专业,主要从事计算机网络及应用研究.

b. 在不同的线程中运行的任务代码必须嵌在某个使用 MultiUse 的类中；

c. 创建新对象时，必须用 CreateObject 函数而不是 New 操作创建新对象.(有关本限制条件的说明：(a)这一点是保证由同一个运行组件提供对象，在执行时就不会产生组件的其他副本.(b)由于使用 New 操作时，VB 会用更有效的但不含任何限制的结构来创建对象，从而绕过 COM，这样就不能保证对象是可创建的(MultiUse)[1].)

2 系统线程的设计与实现

通过分析系统需求，为了实现高速采集和处理数据并完成实时显示及监控实际压力数据，本系统主界面创建为 ActiveX EXE 服务器类型，采用单元线程模型，共使用了 3 个线程：(a)主线程，负责管理通常的前台界面处理，并接收和处理来自控制机柜面板上的按钮信号；(b)数据采集线程，负责数据的采集、平滑和去噪，并送入数据缓冲区供其他线程使用；(c)数据处理线程，负责对数据缓冲区中的最新数据进行显示实时压力曲线，并负责实时超界报警.下面对 3 个线程作进一步的介绍.

2.1 主线程的设计

主线程设计的主要任务是创建窗口，管理主界面上的键盘和鼠标输入，创建两个工作子线程，并控制子线程的存活.在多线程 ActiveX 组件中，无论何时创建新线程都会执行 Sub Main 过程.因此，程序中必须能够识别出在创建的线程是否为第一个组件线程.如果是，则需要显示窗体.主要有两种识别方法，第一种基于 FindWindow API 函数，判断窗体是否已经存在.第二种是基于 FindAtom API 函数，利用 Windows API 提供的对原型的操作来测试已知原型是否存在，判断线程是否为第一个组件线程.

本系统采用第二种方法，为此需要创建 CThread 类，用于创建原型，并在退出程序时摧毁原型.具体 CThread 类的实现如下：

```
'有关原型操作的 Windows API 函数的申明.
Private Declare Function FindAtom Lib "kernel32" Alias "FindAtomA" (ByVal atomName As String) As Integer
Private Declare Function AddAtom Lib "kernel32" Alias "AddAtomA" (ByVal atomName As String) As Integer
Private Declare Function DeleteAtom Lib "kernel32" (ByVal atomName As Integer) As Integer
Private atomID As Integer
' 线程初始化.
Private Sub Class-Initialize()
    Dim atomName As String
    ' 创建应用实例的原型唯一名字.
    atomName = App.EXEName & App.hInstance
    ' 如果原型不存在，就创建该原型.
    If FindAtom(atomName) = 0 Then atomID = AddAtom(atomName)
End Sub
' 在线程终止时，删除该原型.
Private Sub Class-Terminate()
     If atomID Then DeleteAtom atomID
End Sub
' 函数，用于判断创建原型的线程是否为第一个线程.
Function IsFirstThread() As Boolean
     IsFirstThread = (atomID < > 0)
End Function
```

在 Sub Main 过程中，CThread 类的使用以及多线程应用程序的架构如下：

```
' 申明全局变量 Thread .
Public Thread As New CThread
Public blThreadContain As Boolean
```

```
' 全局存活标志变量,用于控制子线程的同步退出.
' 默认值为 TRUE ,主线程退出时置为 FALSE.
Sub Main()
    If Thread.IsFirstThread Then
        '判断是不是系统创建的第一个线程.
       If App.StartMode = vbSModeAutomation Then
            Err.Raise 9999, , "Unable to be instantiated as a component"
        End If
        ' 创建系统主界面 frmMainForm.
        frmMainForm.Show
        ' 创建两个工作子线程.
        Dim CollectData As CCollect, ProcessData as CProcess
        Set CollectData = CreateObject("MThrApp.CCollect")
        Set ProcessData = CreateObject("MThrApp.CProcess")
        blThreadContain = TRUE ' 设置全局存活变量为 TRUE.
        CallByName CollectData, "CollectStart", VbMethod ' 激活对象
        CallByName ProcessData, "ProcessStart", VbMethod ' 激活对象
    Else
        ' 不是系统创建的第一个线程.
    End If
End Sub
```

2.2　数据采集线程的设计

定义一个类 CCollect,把有关数据采集的动作封装在该类中,实现时必须保证数据采集类 CCollect 的 Instancing 类型为 MultiUse.其中,采集端口数据通过调用数据采集卡提供的 DLL 中的函数来实现.下面是 CCollect 类的大体框架:

```
Sub CollectStart()
    Static active As Boolean
    If active Then Exit Sub ' 防止重进入线程.
    active = True
    Do Until not (blThreadContain)
   'blThreadContain 全局存活标志变量,主线程退出时该变量被置为 FALSE,子线程退出.
    ' 调用 DLL 提供的采集函数,检测端口,采集数据.
    ' 对采集的数据进行平滑、去噪.
    ' 向数据缓冲区填充数据,填满缓冲区后,置缓冲区存取标志为 TRUE.
    Loop
    active = False
End Sub
```

2.3　数据处理线程的设计

定义一个类 CProcess,主要是循环测试缓冲区的存取标志,在存取许可的情况下,读取缓冲区的数据,进行处理即显示实时压力曲线和实时超界报警以及保存数据到数据库中,实现时必须保证数据处理类 CProcess 的 Instancing 类型为 MultiUse.下面是 CProcess 类的框架:

```
Sub ProcessStart()
    Static active As Boolean
    If active Then Exit Sub ' 防止重进入线程.
    active = True
```

```
    Do Until not (blThreadContain )
    ' blThreadContain 全局存活标志变量,主线程退出该变量置为 FALSE 子线程退出.
    ' 循环判断数据缓冲区的存取标志变量,若 BufferFull 为 TRUE,则进行数据处理.
If BufferFull1 Then
    ' 读取数据,显示实时压力变化曲线.
    ' 读取工艺参数,判断压力值是否在规定范围内,否则调用 DLL 中的函数,
    ' 控制报警灯亮,如果严重超限,控制警铃响.
        BufferFull1 = False ' 标识数据缓冲区中的数据已经处理.
    End If
    If BufferFull2 Then
    ' 读取数据,显示实时压力变化曲线.
    ' 读取工艺参数,判断压力值是否在规定范围内,否则调用 DLL 中的函数,
    ' 控制报警灯亮,如果严重超限,控制警铃响.
        BufferFull2 = False ' 标识数据缓冲区中的数据已经处理.
    End If
    Loop
    active = False
End Sub
```

3 多线程应用系统中数据缓冲区的设计

为了进一步提高数据采集处理的实时性,系统采用双缓冲区设计[2],实现中主要是要解决对缓冲区的并发和同步问题.

a. 双缓冲区的工作方式(见图 1).即缓冲区 1 中的数据正被数据处理线程使用时,数据采集线程已经开始采集下一批数据到缓冲区 2 了,如此反复交替.图 1 中实线箭头表示在周期 T 时两子线程的读写指针的位置,虚线箭头表示在周期 $T+1$ 时两子线程的读写指针的位置.

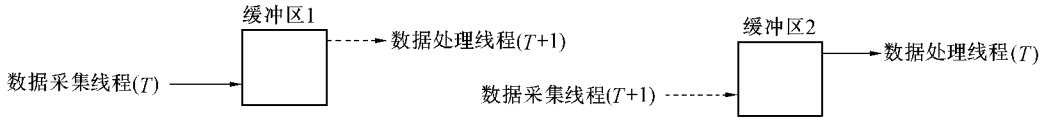

图 1 双缓冲区的工作方式示意图

Fig.1 Work mode with double-buffer

b. 数据缓冲区的并发和同步的解决.多线程应用程序基于优先级的可抢先调度和不可预测性,使得其同步问题变得非常重要.如何正确、高效地实现多线程系统中各个线程之间的通信,使得相关线程之间能够对临界区的访问达成同步,对提高多线程数据采集系统的效率有着重要的作用.在本系统中,由于采用的是单元线程模型,三个线程在同一个单元中,因此可以设置两个全局存取标志变量,当数据采集线程在填充缓冲区 1 前,首先判断 BufferFull1 是否为 FALSE,是则进行填充;当数据采集线程填满缓冲区 1 时,设置该缓冲区的 BufferFull1 为 TRUE,同理,设置缓冲区 2 的存取标志变量为 BufferFull2,用于控制对缓冲区 2 的访问.而数据处理线程则循环测试两个存取标志变量,只有当缓冲区全满后,即相应的存取标志变量为 TRUE 时才进入缓冲区读出数据,进行进一步处理,在处理完缓冲区中的所有数据后设置该缓冲区的存取标志变量为 FALSE,以释放缓冲区.由于 CPU 的运行速度远高于采集卡采集数据的速度,因此,数据处理线程的大部分时间是在等待读取缓冲区的数据,不会因为来不及处理缓冲区的数据而使数据采集线程因等待缓冲区的释放以致造成采集数据的丢失.这样,两个工作线程之间较好地实现了对数据缓冲区访问的并发和同步.

4 系统设计中的若干优化措施

由于本系统是一个工业生产中连续运行的实用系统,系统的可靠性非常重要,这就要求优化设计方案,

通过反复研究和实验，认为以下措施是有效的.

4.1　注意释放内存，避免线程阻塞

在系统的调试过程中，发现 Windows NT Workstation 环境下有一个缺陷，就是当某个任务持续长时间运行时，会导致所占用的存储空间逐渐膨胀，因此有时会因内存自由空间的过少而出现有关线程阻塞的现象，甚至出现死机现象.故在开发程序的过程中，必须考虑有效措施使程序能自动释放内存.

4.2　注意节约占用 CPU 的时间

系统规定数据采集线程和数据处理线程的优先级为最高，其余任务必须注意节约占用 CPU 的时间，否则会降低系统的运行效率.

4.3　合理规划线程内容，控制线程个数

虽然系统采用多线程模型可以有效地提高采集和监控效率，但系统中所拥有的线程不能太多.因为可运行的线程越多，对所有线程轮询一次所需的时间越长，系统延时越大.同时系统的吞吐量将相对减少，当系统总的延时超过一定的限制时，系统将变得不可使用[3].

5　结　束　语

该文所述的设计思想已成功地用于某大型汽车制造企业的发动机镶缸套压力机床的压力数据检测系统.实际运行结果表明，该文所述的基于 VB 的多线程测控系统的设计思想是可行的；整个系统对压力数据的检测迅速准确，系统的运行情况稳定可靠；系统所生成的图形图表为实时监测和控制本生产工序的质量状况提供了依据和手段，同时其数据可为生产线前部相关工序的工艺调整提供有效参考.

参考文献：

[1] Deborah Kurata. Doing objects in Visual Basic 6[M]. Indianapolis：SAMS Publishing，1999.327～348.
[2] 刘晓东，苏光大.一种基于中断的多线程高速图像采集系统[J].计算机工程与应用，2000(5)：8～11.
[3] 余华云，蔡莲红，张维.基于多线程的远程数据库电话访问系统的设计与实现[J].计算机工程，1999(4)：14～17.

Design and Realization of VB-Based Multi-Thread Real-Time Measurement and Control System

GAO Chun-lei，GAO Xin-ling

(*College of Water Conservancy and Hydropower Engineering，Hohai Univ.，Nanjing* 210098，*China*)

Abstract：Introduced is the design and realization of a multi-thread based real-time measurement and control system for pressure machine tools，and discussed are the technical features and programming methods of the system. A system framework is presented by use of the ActiveX technique in Visual Basic. The system is successfully used to a large automobile factory.

Key words：Visual Basic；multi-thread；real-time measurement and control