Using Multiple Service Requests to Control Concurrent DriverLINX® Tasks

by
Dave Sherman
Keithley Instruments, Inc.

Introduction

DriverLINX is a high-performance 32-bit driver for performing data acquisition tasks in Windows®. It is capable of running multitasking applications involving different subsystems within the same piece of data acquisition hardware. Creating a data acquisition task in DriverLINX involves using a service request control to set up an operation, such as: starting a task, stopping a task, or reading the status of a task. When creating multiple tasks, it is necessary to create more than one service request; otherwise, problems will arise, as this document will describe.

Multiple ActiveX® Controls

When programming in Microsoft® Visual Basic, the ActiveX control used for programming in DriverLINX is called the DriverLINXSR control. To run multiple tasks simultaneously from within one program, one DriverLINXSR control must be in the program for each task that will be running.

Multiple Tasks - The Wrong Way

As an example, suppose the application requires the following tasks: polled analog input and two polled counter/timer tasks. The first counter task will be used to generate a 1000Hz square wave, and the second will be used for event counting. This program was run on a Keithley KPCI-3104 data acquisition board, with a voltage source connected to channel 0, and the output of counter/timer 0 connected to the clock input of counter/timer 1. To demonstrate the problem, all of these tasks will be controlled with a single Service Request control.

Figure 1. Example Program Window

If the "Read Voltage" button is clicked, the voltage on channel 0 will be displayed correctly. If the "Start Square Wave" button is clicked, the square wave will start, and it will stop if the "Stop Square Wave" button is clicked.

Problems arise if the "Start Square Wave" button is clicked, then the "Start Counter" button is clicked. If the "Stop Square Wave" button is then clicked, the square wave won't stop. If the "Stop Counter" button is then clicked, the following warning message will appear: "Polled Counter/Timer: Active service request not found." What happened? Each service request has a property called task ID, which is assigned when a service request is started. This task ID is what DriverLINX uses to keep track of the running tasks. When the counter was started, the task ID for the square wave was lost, because it was replaced by a new task ID for the counter. When the "Stop Square Wave" button was clicked, the counter was being stopped,

because that Task ID was the only one of which it was aware. When the "Stop Counter" button was clicked, DriverLINX knew that the last task ID assigned to the service request (for the counter) had been stopped, so it didn't know the counter hardware was still running. If the "Read Voltage" button had been clicked, the same problem would occur, because the task ID would be overwritten by the task ID for the polled analog input.

Multiple Tasks - The Right Way

The solution to this problem is to add two more SR controls to the project. Only one control need be used to initialize the single board. The other two controls simply need to have the ".Req_DLL_name" property set to the appropriate driver name, and the device number needs to be set to the device number of the board in use. The code for the command buttons is changed, so that the "Start Square Wave" and "Stop Square Wave" buttons refer to their dedicated Service Request control. Likewise, the "Start," "Stop," and "Read Counter" buttons are modified to refer to the third Service Request control. Once these changes are implemented, each task can be run simultaneously, without conflicts occurring between them.

Example Programs

The following code examples illustrate the right way and the wrong way to perform multiple service requests. A downloadable version of the "right way" example program is available as an example program in the download center part of the Keithley web site.

Multsr1.frm (the wrong way)

```
Private Sub Read_volts_Click()
With DriverLINXSR1
.Req_op = DL_START
.Req_mode = DL_POLLED
.Req_subsystem = DL_AI
.Evt_Str_type = DL_COMMAND
.Evt_Tim_type = DL_NULLEVENT
.Evt_Stp_type = DL_TCEVENT
.Sel_chan_start = 0
.Sel_buf_N = 0
.Sel_chan_startGainCode = .DLGain2Code(-1) 'Use bipolar unity gain for AI
.Sel_chan_format = DL_tNATIVE
.Sel_chan_N = 1
.Refresh
showmessage DriverLINXSR1 'show errors, if any
Text1.Text = Str(.DLCode2Volts(.Res_Sta_ioValue)) 'Convert counts to volts and display in Text1
End With
End Sub

Private Sub Start_sq_Click()
With DriverLINXSR1
.Req_op = DL_START
.Req_mode = DL_POLLED
.Req_subsystem = DL_CT
.Evt_Str_type = DL_COMMAND
.Evt_Tim_type = DL_RATEEVENT
.Evt_Tim_rateMode = DL_SQWAVE
.Evt_Tim_rateChannel = 0
.Evt_Tim_rateClock = DL_INTERNAL1
.Evt_Tim_rateGate = DL_DISABLED
.Evt_Tim_ratePeriod = .DLSecs2Tics(DL_INTERNAL1, 0.001) 'Make a square wave of period .001 seconds
.Sel_chan_N = 0
.Sel_buf_N = 0
.Evt_Stp_type = DL_COMMAND
.Refresh
End With
showmessage DriverLINXSR1 'show errors, if any
Start_sq.Enabled = False
Stop_sq.Enabled = True
End Sub

Private Sub Start_cnt_Click()
With DriverLINXSR1
.Req_op = DL_START
.Req_mode = DL_POLLED
.Req_subsystem = DL_CT
.Evt_Str_type = DL_COMMAND
.Evt_Tim_type = DL_RATEEVENT
```

```
.Evt_Tim_rateChannel = 1
.Evt_Tim_rateMode = DL_COUNT 'Use event counting for the mode
.Evt_Tim_rateClock = DL_EXTERNAL
.Evt_Tim_rateGate = DL_DISABLED
.Evt_Tim_rateOnCount = 0
.Evt_Tim_ratePeriod = 0
.Evt_Tim_ratePulses = 0
.Evt_Stp_type = DL_COMMAND
.Sel_chan_N = 0
.Sel_buf_N = 0
.Refresh
End With
showmessage DriverLINXSR1 'show errors, if any
Start_cnt.Enabled = False
Stop_cnt.Enabled = True
Read_cnt.Enabled = True

End Sub

Private Sub Read_cnt_Click()
With DriverLINXSR1
.Req_op = DL_STATUS 'Use the status service request to get the value of the event counter
.Refresh
Text2.Text = Str(.Res_Tim_count) 'Get the counts and display in Text2
End With
showmessage DriverLINXSR1 'show errors, if any
End Sub

Private Sub Stop_sq_Click()
With DriverLINXSR1
.Req_op = DL_STOP 'stop the square wave
.Refresh
End With
showmessage DriverLINXSR1 'show errors
Stop_sq.Enabled = False
Start_sq.Enabled = True
End Sub

Private Sub Stop_cnt_Click()
With DriverLINXSR1
.Req_op = DL_STOP 'stop the event counter
.Refresh
End With
showmessage DriverLINXSR1
Stop_cnt.Enabled = False
Read_cnt.Enabled = False
Start_cnt.Enabled = True
End Sub

Private Sub Form_Load()
With DriverLINXSR1
.Req_DLL_name = "kpci3100" 'Use the kpci3100 driver
.Req_device = 0 'and use it with device 0
.Req_op = DL_INITIALIZE
.Req_subsystem = DL_DEVICE
.Req_mode = DL_OTHER
.Refresh
End With
showmessage DriverLINXSR1
End Sub

Sub showmessage(sr As Control)
sr.Req_op = DL_MESSAGEBOX 'Display a messagebox if there are errors
sr.Refresh
End Sub
```

Multsr2.frm (the right way)

```
Private Sub Read_volts_Click()
With DriverLINXSR1
.Req_op = DL_START
.Req_mode = DL_POLLED
.Req_subsystem = DL_AI
.Evt_Str_type = DL_COMMAND
.Evt_Tim_type = DL_NULLEVENT
.Evt_Stp_type = DL_TCEVENT
.Sel_chan_start = 0
.Sel_buf_N = 0
.Sel_chan_startGainCode = .DLGain2Code(-1) 'Use bipolar unity gain for AI
.Sel_chan_format = DL_tNATIVE
.Sel  chan  N = 1
```

```
.Refresh
showmessage DriverLINXSR1
Text1.Text = Str(.DLCode2Volts(.Res_Sta_ioValue))
End With
End Sub

Private Sub Start_sq_Click()
With DriverLINXSR2   ' notice the second control
.Req_op = DL_START
.Req_mode = DL_POLLED
.Req_subsystem = DL_CT
.Evt_Str_type = DL_COMMAND
.Evt_Tim_type = DL_RATEEVENT
.Evt_Tim_rateMode = DL_SQWAVE
.Evt_Tim_rateChannel = 0
.Evt_Tim_rateClock = DL_INTERNAL1
.Evt_Tim_rateGate = DL_DISABLED
.Evt_Tim_ratePeriod = .DLSecs2Tics(DL_INTERNAL1, 0.001)
.Evt_Stp_type = DL_COMMAND
.Sel_chan_N = 0
.Sel_buf_N = 0
.Refresh
End With
showmessage DriverLINXSR2
Start_sq.Enabled = False
Stop_sq.Enabled = True
End Sub

Private Sub Start_cnt_Click()
With DriverLINXSR3   ' notice the third control
.Req_op = DL_START
.Req_mode = DL_POLLED
.Req_subsystem = DL_CT
.Evt_Str_type = DL_COMMAND
.Evt_Tim_type = DL_RATEEVENT
.Evt_Tim_rateChannel = 1
.Evt_Tim_rateMode = DL_COUNT
.Evt_Tim_rateClock = DL_EXTERNAL
.Evt_Tim_rateGate = DL_DISABLED
.Evt_Tim_rateOnCount = 0
.Evt_Tim_ratePeriod = 0
.Evt_Tim_ratePulses = 0
.Evt_Stp_type = DL_COMMAND
.Sel_chan_N = 0
.Sel_buf_N = 0
.Refresh
End With
showmessage DriverLINXSR3
Start_cnt.Enabled = False
Stop_cnt.Enabled = True
Read_cnt.Enabled = True

End Sub

Private Sub Read_cnt_Click()
With DriverLINXSR3
.Req_op = DL_STATUS
.Refresh
Text2.Text = Str(.Res_Tim_count)
End With
showmessage DriverLINXSR3
End Sub

Private Sub Stop_sq_Click()
With DriverLINXSR2
.Req_op = DL_STOP
.Refresh
End With
showmessage DriverLINXSR2
Stop_sq.Enabled = False
Start_sq.Enabled = True
End Sub

Private Sub Stop_cnt_Click()
With DriverLINXSR3
.Req_op = DL_STOP
.Refresh
End With
showmessage DriverLINXSR3
Stop_cnt.Enabled = False
Read_cnt.Enabled = False
Start_cnt.Enabled = True
End Sub

Private Sub Form_Load()
With DriverLINXSR1
.Req_DLL_name = "kpci3100"
```

```
.Req_device = 0
.Req_op = DL_INITIALIZE
.Req_subsystem = DL_DEVICE
.Req_mode = DL_OTHER
.Refresh
showmessage DriverLINXSR1
End With
With DriverLINXSR2   ' notice the second control
.Req_DLL_name = "kpci3100"
.Req_device = 0
End With
With DriverLINXSR3    ' notice the third control
.Req_DLL_name = "kpci3100"
.Req_device = 0
End With
showmessage DriverLINXSR1
End Sub

Sub showmessage(sr As Control)
sr.Req_op = DL_MESSAGEBOX
sr.Refresh
End Sub
```