

TechNote TN-121102-50
Date: December 11, 2002
Applies To: Greenleaf CommX 1.1 and below.
Subject: IncomingData Event – Best Practices for Using

Background

The CommX port control provides for individual communications events to be fired for each of the discrete events that Win32 can create. There are a total of 13 mask bits defined in WinBase.h, not all of which are useful and some are defined "just in case."

In traditional programming using the Win32 Comm API or when using MSComm control, you would call SetCommMask to enable one or more events, and you would need to check in your interrupt handler using GetCommMask to determine what event Windows fired. We take that grunt work out of your hands and into ours. There is a flipside to this. That is the topic of the rest of this TechNote.

IncomingData Event

When you set myPort.IssueRxDataEvent true, when one or more characters have come into the port and are in the buffer you can read from, the IncomingData event is fired by CommX. You would need to code an event handler to react to this event if you want to use this functionality.

If the event continues to be enabled while your event handler code is executing, the result will be recursive calls to the handler. This can cause errors, timeouts, and high CPU usage.

Catch-22 and How to Avoid It

The catch is that CommX does not (and cannot logically) set IssueRxDataEvent false just after firing the event because your handler may or may not exist and may need to do little or nothing. Were CommX to disable the event, how would it know when to re-enable it?

The fix (workaround, solution, etc.) is as follows:

1. First thing in your IncomingData event handler, set IssueRxDataEvent = false
2. Then process whatever you need to do. The event won't be fired again while you're doing this. Were it again fired, the result would be recursive calls to your handler and the result would be confusion and possible "CPU burn."
3. Just before returning from your event handler, set IssueRxDataEvent = true again.