## Detection of disconnect/reconnect 2016

MKY

The code at the next page used to detect a disconnect while a serial port was open. It was implemented in MKY.IO.Serial.SerialPort.aliveMonitor\_Elapsed().

As commented, this detection is by far not easy to achieve, given the variety of converters, drivers and operating systems. This document summarizes the findings, in order to keep this information for future reference.

#### Summary of findings

* IsOpen() doesn't work with Prolific drivers, it will return 'true' even after a disconnect.
* BytesToRead() works also with Prolific drivers, but leads to an additional async 'IOException' in 'mscorlib'.
* Async 'IOException' in 'mscorlib' still occurs 🡪 Handle it in YAT.Controller.Main. isfsdf

#### Conclusion

Instead of this implementation, simply rely on System.IO.Ports.SerialPort.GetPortNames()!  
Why didn't I think of this earlier…?!?

#### Code extracted from MKY.IO.Serial.SerialPort of SVN #1094 as of 2016-06-29

try

**{**

// Detect whether port has been shut down, e.g. USB-to-serial-converter

// device has been disconnected.

//

// This is not that straight-forward to achieve, as different devices

// or their driver behave differently. Used to only check for 'IsOpen',

// but that is not sufficient for e.g. Prolific driver which will still

// indicate 'IsOpen' even when device has long been disconnected.

// With device disconnected, debugger shows exceptions for...

// ...BytesToRead/BytesToWrite, or...

// ...CtsHolding/DsrHolding/CDHolding properties.

// Hardware pin state may again depend on device or driver, thus using

// one of the pure software properties.

if **(!**this**.**port**.**IsOpen**)**

**{**

DebugMessage**(**"AliveMonitorElapsed() has detected shutdown of port as it is no longer open."**);**

RestartOrResetPortAndThreadsAndNotify**();**

return**;**

**}**

int byteToReadDummy **=** this**.**port**.**BytesToRead**;** // Force e.g. 'IOException', see above.

UnusedLocal**.**PreventAnalysisWarning**(**byteToReadDummy**);**

// Attention:

//

// On an internal port that is open and in use, accessing 'BytesToRead' will first properly lead to an 'IOException',

// but later an additional 'ObjectDisposedException' will happen on a separate thread!

// > Message : "Safe handle has been closed"

// > Source : "mscorlib"

// > Stack : at System.StubHelpers.StubHelpers.SafeHandleC2NHelper(Object pThis, IntPtr pCleanupWorkList)

// at Microsoft.Win32.UnsafeNativeMethods.GetOverlappedResult(SafeFileHandle hFile,

NativeOverlapped\* lpOverlapped, Int32& lpNumberOfBytesTransferred, Boolean bWait)

// at System.IO.Ports.SerialStream.EventLoopRunner.WaitForCommEvent()

// at System.Threading.ExecutionContext.Run(ExecutionContext executionContext, ContextCallback callback,

Object state)

// at System.Threading.ThreadHelper.ThreadStart()

//

// A couple of workarounds have been considered:

// > Detecting a suspend request from the operating system.

// => Not a solution, as .NET doesn't provide this functionality and thus the implementation would get OS dependent.

// Note that SystemEvents.PowerModeChanged is located in Microsoft.Win32 and therefore also OS dependent.

// > Ignoring 'ObjectDisposedException' from "mscorlib" in the 'currentDomain\_UnhandledException' (Controller.Main).

// => Not a solution, as the exception already happened and will have closed the port.

// > Preventing such exception in 'best-effort', by skipping the access to 'BytesToRead' for internal ports COM1 and COM2.

// => Not really a solution, doesn't work for other than COM1/COM2 (e.g. Microchip MCP2221 USB-to-UART/I2C Bridge)

// > Make AliveMonitor configurable and handle the exception above in the 'ThreadException'.

**}**

catch **(**IOException ex**)** // The best way to detect a disconnected device is handling this exception...

**{**

DebugEx**.**WriteException**(**GetType**(),** ex**,** "AliveMonitorElapsed() has detected shutdown of port as it is no longer accessible."**);**

RestartOrResetPortAndThreadsAndNotify**();**

**}**

catch **(**Exception ex**)**

**{**

DebugEx**.**WriteException**(**GetType**(),** ex**,** "AliveMonitorElapsed() has caught an unexpected exception!

Restarting the port to try fixing the issue..."**);**

RestartOrResetPortAndThreadsAndNotify**();**

**}**