Workshop 4 (A): Telemetry and Data Acquisition

Mahidol University

June 13, 2008

Paul Evenson
University of Delaware
Bartol Research Institute
Workshop Series Idea

• Introduce students to technical aspects of neutron monitor operation
• Rotating workshop series that will repeat every two years at a two per year rate
• Independent enough so students can join at any point
• Accommodate wide skill range with an emphasis on “hands on” experience and individual discussion
Workshop Series Plan

1. Detector operation (i.e. proportional reading schematics, and analog electronics (through the PHA)
2. Digital logic (including theory of the peripheral devices like the barometers).
3. Microcontrollers (including data transfer methods within the electronics system)
4. Real time data acquisition (i.e. the Visual Basic program handling real-time data)
   A. Telemetry and Data Acquisition
   B. Data conversion and manipulation
Neutron Monitors

High energy cosmic rays are rare. Observing them at high time resolution requires a large detector.
Ground based instruments remain the state-of-the-art method for studying these elusive particles.

Neutron monitors on the surface record the byproducts of nuclear interactions of high energy primary cosmic rays with Earth's atmosphere.
Neutron Monitor Principle

- An incoming hadron interacts with a nucleus of lead to produce several low energy neutrons.
- These neutrons thermalize in polyethylene or other material containing a lot of hydrogen.
- Thermal neutrons cause fission reaction in $^{10}$B ($^{7}$Li + $^{4}$He) or $^{3}$He ($^{3}$H + p) in a gas proportional counter.
- The large amount of energy released in the fission process dominates that of all penetrating charged particles. There is essentially no background.
- Unfortunately some leptons can interact with the lead, most notably low energy negative muons in muonic atoms. About 6% of the counts come from this source.
- This causes the monitor to have some sensitivity to atmospheric structure. (Thunderstorm effect?)
Simulated interaction in a neutron monitor
Starting Point for Today:

• Numbers describing one second of operation of the (“300 chip”) micromonitor have been collected:
  – Accumulated Counts (24 bits)
  – Sequence timer (32 bits)
  – Voltages (4 x 16 bits)
  – Temperatures (3 x 16 bits)
  – Pulse Heights (16 x 16 bits)
Bits and Bytes

- Binary notation: 01010011
- Octal: 123 (01 010 011)
- Hexadecimal: 53 (0101 0011)
- ASCII Character: S
### The ASCII Code Table

<table>
<thead>
<tr>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>(null)</td>
<td>00000</td>
<td>0x20</td>
<td></td>
<td>(sp)</td>
<td>0040</td>
<td>0x20</td>
<td></td>
<td>$</td>
<td>0100</td>
<td>0x60</td>
<td></td>
</tr>
<tr>
<td>(soh)</td>
<td>0001</td>
<td>0x31</td>
<td></td>
<td>!</td>
<td>0041</td>
<td>0x21</td>
<td>A</td>
<td>A</td>
<td>0101</td>
<td>0x61</td>
<td></td>
</tr>
<tr>
<td>(stx)</td>
<td>0002</td>
<td>0x32</td>
<td></td>
<td>&quot;</td>
<td>0042</td>
<td>0x22</td>
<td>B</td>
<td>B</td>
<td>0102</td>
<td>0x62</td>
<td></td>
</tr>
<tr>
<td>(etx)</td>
<td>0003</td>
<td>0x33</td>
<td>#</td>
<td></td>
<td>0043</td>
<td>0x23</td>
<td>C</td>
<td>C</td>
<td>0103</td>
<td>0x63</td>
<td></td>
</tr>
<tr>
<td>(eot)</td>
<td>0004</td>
<td>0x34</td>
<td>$</td>
<td></td>
<td>0044</td>
<td>0x24</td>
<td>D</td>
<td>D</td>
<td>0104</td>
<td>0x64</td>
<td></td>
</tr>
<tr>
<td>(enq)</td>
<td>0005</td>
<td>0x35</td>
<td>%</td>
<td></td>
<td>0045</td>
<td>0x25</td>
<td>E</td>
<td>E</td>
<td>0105</td>
<td>0x65</td>
<td></td>
</tr>
<tr>
<td>(ack)</td>
<td>0006</td>
<td>0x36</td>
<td>&amp;</td>
<td></td>
<td>0046</td>
<td>0x26</td>
<td>F</td>
<td>F</td>
<td>0106</td>
<td>0x66</td>
<td></td>
</tr>
<tr>
<td>(bel)</td>
<td>0007</td>
<td>0x37</td>
<td>'</td>
<td></td>
<td>0047</td>
<td>0x27</td>
<td>G</td>
<td>G</td>
<td>0107</td>
<td>0x67</td>
<td></td>
</tr>
<tr>
<td>(bs)</td>
<td>0010</td>
<td>0x30</td>
<td>(</td>
<td></td>
<td>0040</td>
<td>0x20</td>
<td>H</td>
<td>H</td>
<td>0110</td>
<td>0x6a</td>
<td></td>
</tr>
<tr>
<td>(ht)</td>
<td>0011</td>
<td>0x31</td>
<td>)</td>
<td></td>
<td>0041</td>
<td>0x21</td>
<td>I</td>
<td>I</td>
<td>0111</td>
<td>0x6b</td>
<td></td>
</tr>
<tr>
<td>(nl)</td>
<td>0012</td>
<td>0x32</td>
<td>*</td>
<td></td>
<td>0042</td>
<td>0x22</td>
<td>J</td>
<td>J</td>
<td>0112</td>
<td>0x6c</td>
<td></td>
</tr>
<tr>
<td>(vt)</td>
<td>0013</td>
<td>0x33</td>
<td>+</td>
<td></td>
<td>0043</td>
<td>0x23</td>
<td>K</td>
<td>K</td>
<td>0113</td>
<td>0x6d</td>
<td></td>
</tr>
<tr>
<td>(np)</td>
<td>0014</td>
<td>0x34</td>
<td>,</td>
<td></td>
<td>0044</td>
<td>0x24</td>
<td>L</td>
<td>L</td>
<td>0114</td>
<td>0x6e</td>
<td></td>
</tr>
<tr>
<td>(cr)</td>
<td>0015</td>
<td>0x35</td>
<td>-</td>
<td></td>
<td>0045</td>
<td>0x25</td>
<td>M</td>
<td>M</td>
<td>0115</td>
<td>0x6f</td>
<td></td>
</tr>
<tr>
<td>(so)</td>
<td>0016</td>
<td>0x36</td>
<td>.</td>
<td></td>
<td>0046</td>
<td>0x26</td>
<td>N</td>
<td>N</td>
<td>0116</td>
<td>0x70</td>
<td></td>
</tr>
<tr>
<td>(si)</td>
<td>0017</td>
<td>0x37</td>
<td>/</td>
<td></td>
<td>0047</td>
<td>0x27</td>
<td>O</td>
<td>O</td>
<td>0117</td>
<td>0x71</td>
<td></td>
</tr>
<tr>
<td>(dle)</td>
<td>0020</td>
<td>0x30</td>
<td>0</td>
<td></td>
<td>0040</td>
<td>0x20</td>
<td>P</td>
<td>P</td>
<td>0120</td>
<td>0x70</td>
<td></td>
</tr>
<tr>
<td>(dc1)</td>
<td>0021</td>
<td>0x31</td>
<td>1</td>
<td></td>
<td>0041</td>
<td>0x21</td>
<td>Q</td>
<td>Q</td>
<td>0121</td>
<td>0x71</td>
<td></td>
</tr>
<tr>
<td>(dc2)</td>
<td>0022</td>
<td>0x32</td>
<td>2</td>
<td></td>
<td>0042</td>
<td>0x22</td>
<td>R</td>
<td>R</td>
<td>0122</td>
<td>0x72</td>
<td></td>
</tr>
<tr>
<td>(dc3)</td>
<td>0023</td>
<td>0x33</td>
<td>3</td>
<td></td>
<td>0043</td>
<td>0x23</td>
<td>S</td>
<td>S</td>
<td>0123</td>
<td>0x73</td>
<td></td>
</tr>
<tr>
<td>(dc4)</td>
<td>0024</td>
<td>0x34</td>
<td>4</td>
<td></td>
<td>0044</td>
<td>0x24</td>
<td>T</td>
<td>T</td>
<td>0124</td>
<td>0x74</td>
<td></td>
</tr>
<tr>
<td>(nak)</td>
<td>0025</td>
<td>0x35</td>
<td>5</td>
<td></td>
<td>0045</td>
<td>0x25</td>
<td>U</td>
<td>U</td>
<td>0125</td>
<td>0x75</td>
<td></td>
</tr>
<tr>
<td>(syn)</td>
<td>0026</td>
<td>0x36</td>
<td>6</td>
<td></td>
<td>0046</td>
<td>0x26</td>
<td>V</td>
<td>V</td>
<td>0126</td>
<td>0x76</td>
<td></td>
</tr>
<tr>
<td>(etb)</td>
<td>0027</td>
<td>0x37</td>
<td>7</td>
<td></td>
<td>0047</td>
<td>0x27</td>
<td>W</td>
<td>W</td>
<td>0127</td>
<td>0x77</td>
<td></td>
</tr>
<tr>
<td>(can)</td>
<td>0030</td>
<td>0x30</td>
<td>8</td>
<td></td>
<td>0048</td>
<td>0x28</td>
<td>X</td>
<td>X</td>
<td>0130</td>
<td>0x78</td>
<td></td>
</tr>
<tr>
<td>(em)</td>
<td>0031</td>
<td>0x31</td>
<td>9</td>
<td></td>
<td>0049</td>
<td>0x29</td>
<td>Y</td>
<td>Y</td>
<td>0131</td>
<td>0x79</td>
<td></td>
</tr>
<tr>
<td>(sub)</td>
<td>0032</td>
<td>0x32</td>
<td>:</td>
<td></td>
<td>0040</td>
<td>0x20</td>
<td>Z</td>
<td>Z</td>
<td>0132</td>
<td>0x7a</td>
<td></td>
</tr>
<tr>
<td>(esc)</td>
<td>0033</td>
<td>0x33</td>
<td>;</td>
<td></td>
<td>0051</td>
<td>0x31</td>
<td>a</td>
<td>a</td>
<td>0133</td>
<td>0x7b</td>
<td></td>
</tr>
<tr>
<td>(fs)</td>
<td>0034</td>
<td>0x34</td>
<td>&lt;</td>
<td></td>
<td>0060</td>
<td>0x38</td>
<td>b</td>
<td>b</td>
<td>0134</td>
<td>0x7c</td>
<td></td>
</tr>
<tr>
<td>(gs)</td>
<td>0035</td>
<td>0x35</td>
<td>=</td>
<td></td>
<td>0110</td>
<td>0x6a</td>
<td>c</td>
<td>c</td>
<td>0135</td>
<td>0x7d</td>
<td></td>
</tr>
<tr>
<td>(rs)</td>
<td>0036</td>
<td>0x36</td>
<td>&gt;</td>
<td></td>
<td>0220</td>
<td>0x14</td>
<td>d</td>
<td>d</td>
<td>0136</td>
<td>0x7e</td>
<td></td>
</tr>
<tr>
<td>(us)</td>
<td>0037</td>
<td>0x37</td>
<td>?</td>
<td></td>
<td>0260</td>
<td>0x1e</td>
<td>e</td>
<td>e</td>
<td>0137</td>
<td>0x7f</td>
<td></td>
</tr>
</tbody>
</table>

### ASCII Name Description

- **nul**: null byte
- **bcl**: bell character
- **bs**: backspace
- **ht**: horizontal tab
- **np**: formfeed
- **nl**: newline
- **cr**: carriage return
- **vt**: vertical tab
- **esc**: escape
- **sp**: space
Telemetry: Reproduce a specific bit sequence in a remote location

- Explicitly clocked data
  - Simple and fast
  - No need for accurate frequency
  - Needs at least two “wires”
  - Shift registers
- Self clocked data
  - Next slide …
RZ (Return to Zero) Self Clocked Data

Manchester Encoding: Manchester Encoding translates a '1' into a low to high transition [01], and a '0' is translated into a high to low transition [10]. Also called Biphasic Code. Used with the Ethernet interface. More on Manchester Encoding

- Clock is implicit
- Accurate frequency is essential to detection

http://www.interfacebus.com/Definitions.html
NRZ (Non Return to Zero) Data

"A type of 'null' encoding, where a logical 'zero' is represented by a particular line state, and a logical 'one' by another. In other words, there is no encoding, as distinct from RZ encoding." NRZ is used with RS-232 and CANbus.

Non-return to zero encoding is used in slow speed synchronous and asynchronous transmission interfaces. With NRZ, a logic 1 bit is sent as a high value and a logic 0 bit is sent as a low value [really no encoding at all]. The receiver may lose synchronization when using NRZ to encode a synchronous link which may have long runs of consecutive bits with the same value [no changes in voltage]. Other problems with NRZ include; Data sequences containing the same number of 1's and 0's will produce a DC level, and NRZ requires a large bandwidth, from 0Hz [for a sequence containing only 1's or only 0's] to half of the data rate [for a sequence of 10101010].
Computer COM Ports Use RS 232

- Asynchronous Framing [Known data width, 8bits] with NRZ encoding
- The stop bit is used to bring [or insure] the signal rests at a logic high following the end of the frame
- Must have some framing gaps
- Still need internal patterns for synchronization of character strings
- For strict ASCII data, <carriage return> and/or <line feed> are typical
So what does the micromonitor actually transmit?

- One-per-second burst of 51 characters
- Framing gap is time between bursts (records)
- Record format:
  - “LSMN” in ASCII
  - Master board number (1 binary byte)
  - Remote within master (1 binary bytes)
  - Accumulated Counts (3 binary bytes)
  - Sequence timer (4 binary bytes)
  - Voltages (4 x 2 binary bytes)
  - Temperatures (3 x 2 binary bytes)
  - Pulse Heights (16 x 2 binary bytes)
  - `<cr><lf>` in ASCII
What does a COM port do?

- Electronically searches for start and stop bits
- Places each eight bit byte at the end of an “input buffer” (exact timing is now lost)
- Keeps track of the length of the buffer
- Via the “driver” lets a program know the length of the buffer
- Allows a program to remove bytes from the start of the buffer
- Keeps track of various error conditions
Visual Basic Interacts with the Driver via the MSComm Control

• A Control is a subroutine that can be “called” either by the program or by the system
• Parameters are termed Properties
• Calls by the system are termed Events
• Calls by the program are termed Methods
• Setting a Property can also result in an action
**MSComm Control Properties**

(Note that several of these could also be called Methods)

The **MSComm** control provides the following two ways for handling communications:

- Event-driven communications is a very powerful method for handling serial port interactions. In many situations you want to be notified the moment an event takes place, such as when a character arrives or a change occurs in the Carrier Detect (CD) or Request To Send (RTS) lines. In such cases, use the **MSComm** control’s **OnComm** event to trap and handle these communications events. The **OnComm** event also detects and handles communications errors. For a list of all possible events and communications errors, see the **CommEvent** property.

- You can also poll for events and errors by checking the value of the **CommEvent** property after each critical function of your program. This may be preferable if your application is small and self-contained. For example, if you are writing a simple phone dialer, it may not make sense to generate an event after receiving every character, because the only characters you plan to receive are the OK response from the modem.

Although the **MSComm** control has many important properties, there are a few that you should be familiar with first.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CommPort</td>
<td>Sets and returns the communications port number.</td>
</tr>
<tr>
<td>Settings</td>
<td>Sets and returns the baud rate, parity, data bits, and stop bits as a string.</td>
</tr>
<tr>
<td>PortOpen</td>
<td>Sets and returns the state of a communications port. Also opens and closes a port.</td>
</tr>
<tr>
<td>Input</td>
<td>Returns and removes characters from the receive buffer.</td>
</tr>
<tr>
<td>Output</td>
<td>Writes a string of characters to the transmit buffer.</td>
</tr>
</tbody>
</table>
The COM Port “Event”

OnComm Event

See Also  Example  Applies To

The OnComm event is generated whenever the value of the CommEvent property changes, indicating that either a communication event or an error occurred.

Syntax

Private Sub object_OnComm ()

The OnComm event syntax has these parts:

<table>
<thead>
<tr>
<th>Part</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>An object expression that evaluates to an object in the Applies To list.</td>
</tr>
</tbody>
</table>

Remarks

The CommEvent property contains the numeric code of the actual error or event that generated the OnComm event. Note that setting the RThreshold or SThreshold properties to 0 disables trapping for the comEvReceive and comEvSend events, respectively.

OnComm Event Example

The following example shows how to handle communications errors and events. You can insert code after each related Case statement, to handle a particular error or event.

Private Sub MSComm_OnComm ()
    Select Case MSComm1.CommEvent
        ' Handle each event or error by placing code below each case statement
        '
        ' Errors
        Case comEventBreak ' A Break was received.
        Case comEventFrame ' Framing Error
        Case comEventOverrun ' Data Lost.
        Case comEventRwOver ' Receive buffer overflow.
        Case comEventRxParity ' Parity Error.
        Case comEventTxFull ' Transmit buffer full.
        Case comEventDCB ' Unexpected error retrieving DCB.
        '
        ' Events
        Case comEvCD ' Change in the CD line.
        Case comEvCTS ' Change in the CTS line.
        Case comEvDSR ' Change in the DSR line.
        Case comEvRing ' Change in the Ring Indicator.
        Case comEvReceive ' Received RThreshold # of characters in the transmit buffer.
        Case comEvSend ' There are SThreshold number of characters in the transmit buffer.
        Case comEvEof ' An EOF character was found in the input stream
    End Select
End Sub
What we will do next (I Hope!)

• See the full program operate
• Look at the input data on an oscilloscope
• Make a “splitter” to enter the data into several computers
• Write a simple Visual Basic program together

Tomorrow we will go through the operation of the full program, and you can try to change your copy of this
Our Simple VB Program

• Start a new project with one form and one module
• Put a “Label Control” on the form
• Make it change color between red and green
• Put on a second label
• Make it say “red” or “green”
• Put on a Comm control
• Open or close it as the label goes green or red
• Read data
• Convert data to “hex” format
• Make the label display the latest data