Workshop 4 (A):
Telemetry and Data Acquisition

Mahidol University

June 13, 2008

Paul Evenson
University of Delaware
Bartol Research Institute

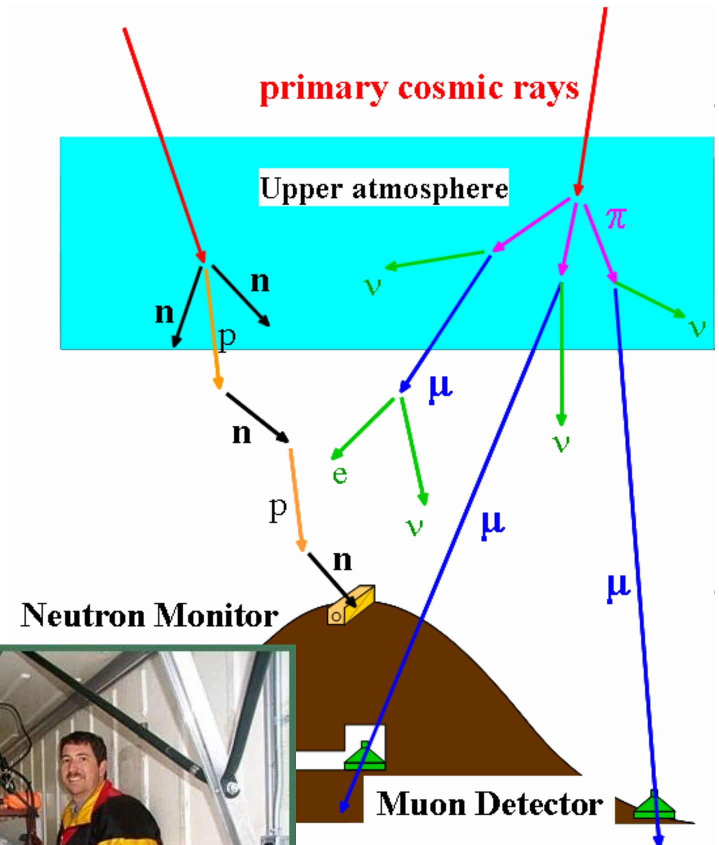
Workshop Series Idea

- 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

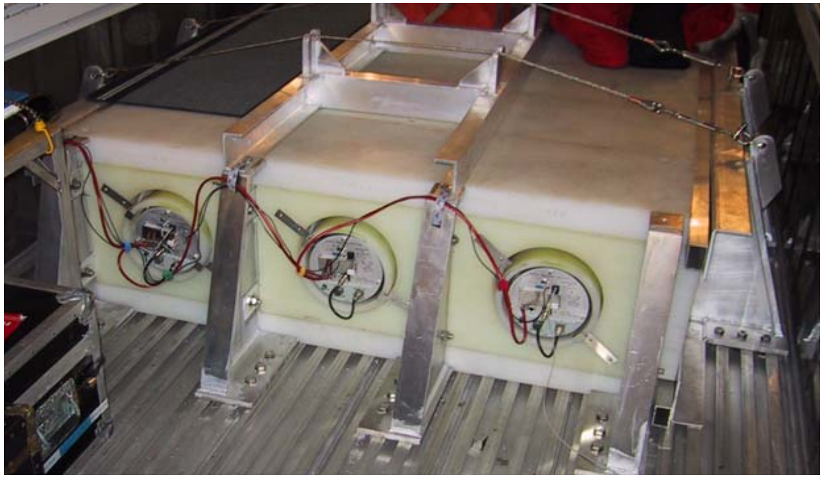


High energy cosmic rays are rare. Observing them at high time resolution requires a large detector.

Ground based instruments remain the state-of-the-art method for studying these elusive particles.

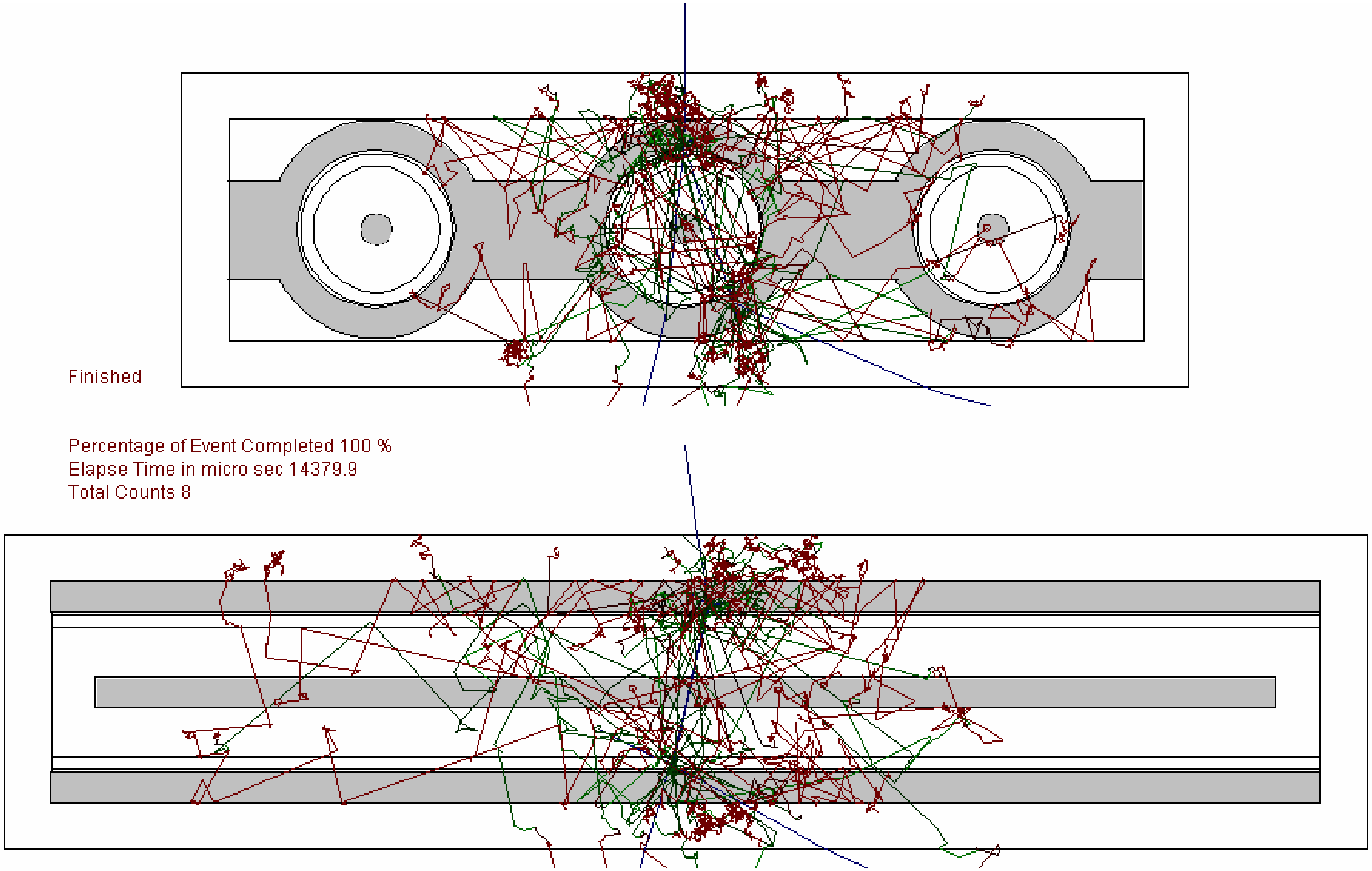
Neutron monitors on the surface record the byproducts of nuclear interactions of high energy primary cosmic rays with Earth's atmosphere.

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 10B (7Li + 4He) or 3He (3H + 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

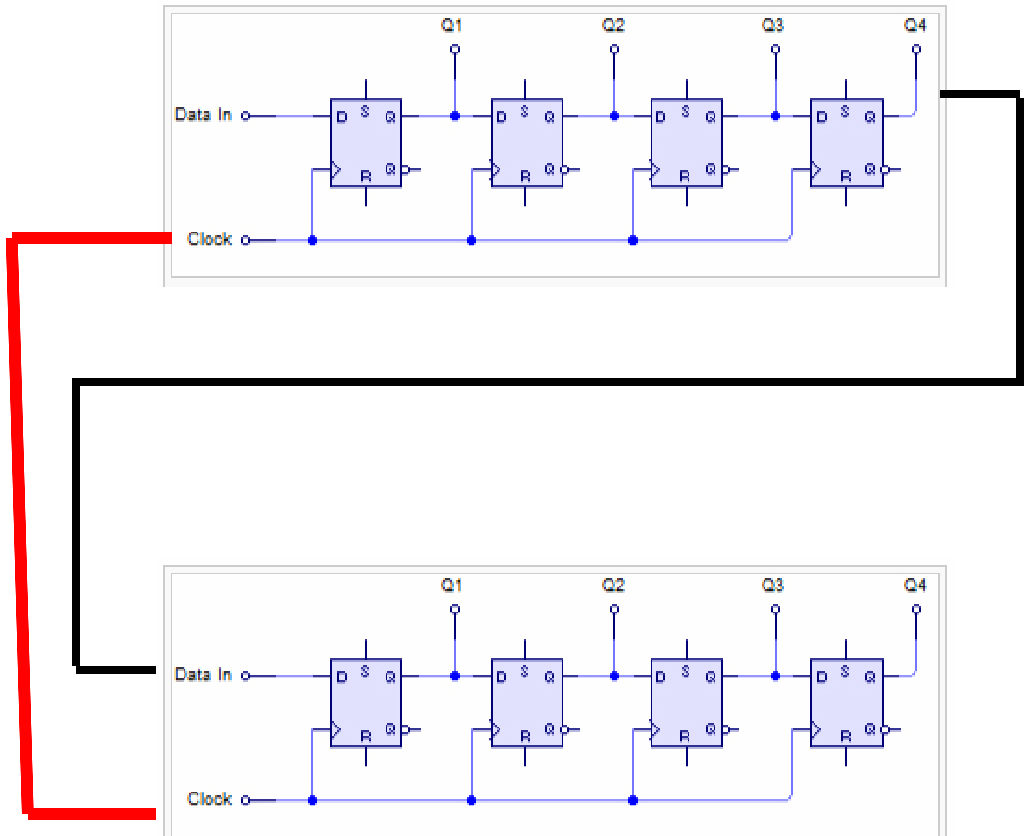
The ASCII Code Table

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(nul)	0	0000	0x00	(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
(soh)	1	0001	0x01	!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
(stx)	2	0002	0x02	"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
(etx)	3	0003	0x03	#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
(eot)	4	0004	0x04	$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
(enq)	5	0005	0x05	%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
(ack)	6	0006	0x06	&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
(bel)	7	0007	0x07	'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(bs)	8	0010	0x08	(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
(ht)	9	0011	0x09	)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
(nl)	10	0012	0x0a	*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
(vt)	11	0013	0x0b	+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
(np)	12	0014	0x0c	,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
(cr)	13	0015	0x0d	-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
(so)	14	0016	0x0e	.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
(si)	15	0017	0x0f	/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
(dle)	16	0020	0x10	0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
(dc1)	17	0021	0x11	1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
(dc2)	18	0022	0x12	2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
(dc3)	19	0023	0x13	3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
(dc4)	20	0024	0x14	4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
(nak)	21	0025	0x15	5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
(syn)	22	0026	0x16	6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
(etb)	23	0027	0x17	7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
(can)	24	0030	0x18	8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
(em)	25	0031	0x19	9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
(sub)	26	0032	0x1a	:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
(esc)	27	0033	0x1b	;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
(fs)	28	0034	0x1c	<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
(gs)	29	0035	0x1d	=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
(rs)	30	0036	0x1e	>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
(us)	31	0037	0x1f	?	63	0077	0x3f	_	95	0137	0x5f	(del)	127	0177	0x7f

ASCII Name	Description
nul	null byte
bel	bell character
bs	backspace
ht	horizontal tab
np	formfeed
nl	newline
cr	carriage return
vt	vertical tab
esc	escape
sp	space

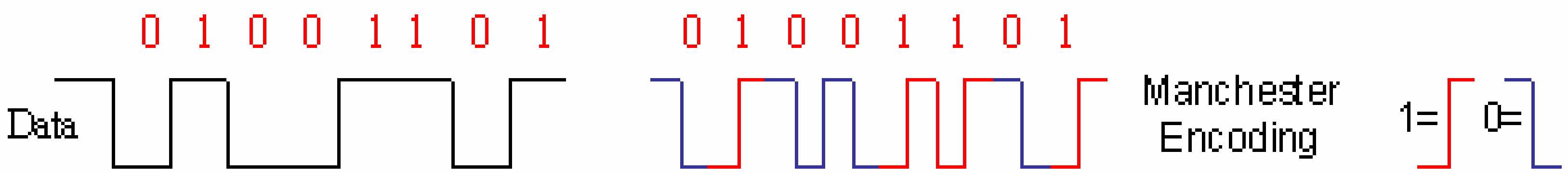
Telemetry: Reproduce a specific bit sequence in a remote location

- 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 Biphase Code. Used with the Ethernet interface. More on Manchester Encoding



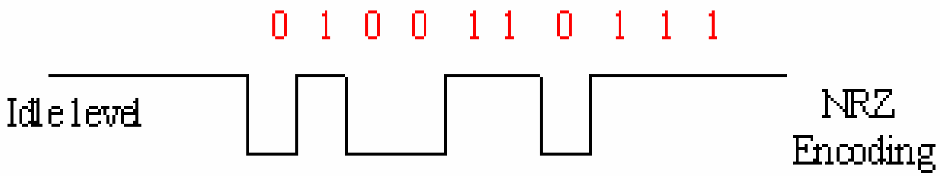
Manchester Encoding

- Clock is implicit
- Accurate frequency is essential to detection

http://www.interfacebus.com/Definitions.html

NRZ (Non Return to Zero) Data

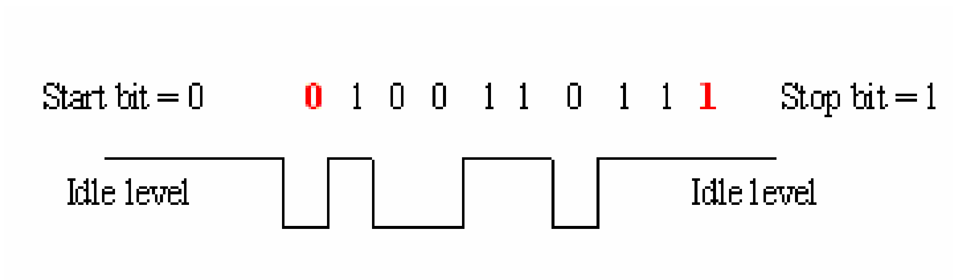
"A type of 'null' encoding, where a logical 'zero' is represented by a particular line state, and a logical 'one' by another. In other words, there is no encoding, as distinct from RZ encoding." NRZ is used with RS-232 and CANbus.

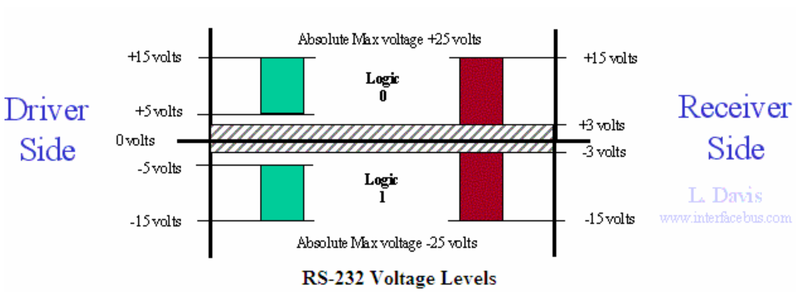


NRZ Encoding

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.

Properties	Description
CommPort	Sets and returns the communications port number.
Settings	Sets and returns the baud rate, parity, data bits, and stop bits as a string.
PortOpen	Sets and returns the state of a communications port. Also opens and closes a port.
Input	Returns and removes characters from the receive buffer.
Output	Writes a string of characters to the transmit buffer.

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:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.

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 comEventRxOver   ' 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
                          ' chars.
      Case comEvSend   ' There are SThreshold number of
                       ' characters in the transmit
                       ' buffer.
      Case comEvEof   ' An EOF charater 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