Application Note #0108

Developing TNET Applications with the MSCOMM Control

Introduction

Many TNET data collection applications have been (and will continue to be) written using the popular Visual Basic development system. The “Professional” and “Enterprise” versions of Visual Basic provide the MSCOMM control to simplify serial communications with a COM port. This application note will explain some basic concepts for using the MSCOMM control when developing an application for the TIM1B TNET controller based on Visual Basic version 6 (VB6). It is assumed the reader is already familiar with the TNET system, the TIM1B controller, VB6 programming, and the Windows operating system. No attempt will be made to explain VB6 programming or application software design.

TIM1B Setup

The heart of the TNET data collection system is the TIM1B controller module. This device connects to a host computer via an RS-232 serial port. A utility program (TIM1BUP.EXE) is provided with the controller which is used to configure the TIM1B and test the connected TNET devices. Typically, the TIM1B is configured for the default host/controller protocol (SR3 = 1). Each device on the TNET network has a unique address and must be included in the TIM1B polling list (see the TIM1B documentation for information on the controller).

Host/TIM1B Protocol

Communications between the TIM1B and the host computer is accomplished using a simple ASCII string packet protocol. Each packet consists of device address, a field delimiter (default is comma), the data field and a string terminator (default is <CR><LF>). The contents of the data field can include special characters or control codes unique to the TNET device receiving the data (see the individual TNET device documentation for details).

Transmitting data to a TNET device from the host application is accomplished by simply sending the TIM1B controller the ASCII packet. To display the message “Hello World” on a TransTerm terminal device the data field would normally include a “clear screen” code (hex 0C <FF>) and a “terminal reenable” code (hex 0F <SI>). For example, if the terminal device address is 65 the packet string would be:

```
65,<FF>Hello World<SI><CR><LF>
```

The TIM1B receives data packets from the attached TNET devices and forwards them one at a time to the host computer. Each received device packet sent by the TIM1B to the host computer must be acknowledged by the application software using the <ACK> (hex 06) code. When the TIM1B receives the acknowledgement from the host it is free to forward the next available packet. This provides a simple communications protocol between the TIM1B and the applications software.

MSCOMM Initialization

The MSCOMM control can be initialized at design time or at run time under program control. Several properties must be set before the port is opened for use. Typically, as a minimum, the following properties must be initialized:

```
MSComm1.ComPort = 1
MSComm1.Settings = “9600,N,8,1”
MSComm1.InputLen = 1
MSComm1.DTREnable = True
MSComm1.RTSEnable = True
```

In addition, the programmer must decide if the received data packets will be captured using an event driven model or a polled model. Normally, the event driven model would be preferred and the receive buffer threshold is set to trigger an event when one or more bytes arrive in the receive buffer.

```
        MSComm1.Rtheshold = 1
```

MSCOMM Event Function
The MSCOMM control provides a single event function called “OnComm”. The example below demonstrates a typical “OnComm” event function used to capture TIM1B packets. This function must determine what event triggered the routine and process it accordingly. When a complete packet is received the application software must parse out the device address and process the data field. In this example, only the receive data and error events are processed, other events could be included as needed.

```
Private Sub MSComm1_OnComm()
   Static srx As String
   Dim cx As String

   Select Case MSComm1.CommEvent
      Case comEvReceive            ‘Receive buffer has at least 1 byte
         Do While MSComm1.InBufferCount > 0
            cx = MSComm1.Input     'Get one byte
            Select Case cx
               Case Chr(13)        ‘<CR> packet terminator
                  TnetInput srx    ‘process packet
                  srx = ""         ‘reset packet buffer
                  MSComm1.Output = Chr(6)          'send <ACK>
               Case Chr(10)        'ignore <LF> terminators
               Case Else
                  srx = srx & cx   ‘capture packet bytes
            End Select
         Loop
     Case Is > 1000                ‘detected port error
         ComError MSComm1.CommEvent
   End Select
End Sub
```

Send TNET Packets
Sending a packet to a TNET device is very easy with the MSCOMM control. The example below shows a simple subroutine used to create an output packet and send it to the TIM1B controller for delivery to the specified device. Notice that this example includes the “terminal reenable” code as part of the packet.

```
'Tnet Output
'On Call:
'   ida = device address (1 - 250 or 255)
'   src = string to deliver
Public Sub TnetOutput(ida As Integer, src As String)
   Dim cs As String

   cs = ida & "," & src & Chr(15) & vbCr
   MSComm1.Output  cs
End Sub
```

Opening the Serial Port
The MSCOMM control must be “opened” at run time before any serial communications can take place. After it is fully initialized (either at design time or run time) it can be opened as follows:

```
MSComm1.PortOpen = True
```

If for any reason the MSCOMM control can not open the port a error event will be triggered. As a matter of good programming practice the port should be “closed” when the application program terminates or the port is no longer needed. This can be accomplished as follows:

```
MSComm1.PortOpen = False
```