VB Windows Communications

Carl Franklin, Software Engineer
Crescent Software, Inc.
(203) 438-5300 CIS: 70662,2605

Prerequisites: Familiarity with the Visual Basic 3.0 MSCOMM control.

Demo code: FRANKLIN.ZIP

Opening the Comm Port

In QuickBASIC:

```
OPEN "COM2:9600,N,8,1" FOR RANDOM AS #1
```

In Visual Basic:

```
Comm1.CommPort = 2
Comm1.Settings = "9600,N,8,1"
Comm1.PortOpen = True
```

Sending Data Out The Port

In QuickBASIC:

```
PUT #1, , “HELLO WORLD”
```

In Visual Basic:

```
Comm1.Output = “HELLO WORLD”
```

Processing Received Data

Data is received automatically by MSCOMM (or any other comm tool).

All received data goes into a receive buffer, whose size is defined by the InBufferSize property (up to 32K).

Your application must read the data from the receive buffer. The first byte received is the first byte read.

Reading Data from the buffer

The Input Property reads one or more characters from the receive buffer.

The InputLen Property determines how many characters are read with each assigning of the Input property:

```
'--- Tell MSCOMM to read 1 character
Comm1.InputLen = 1
'--- Read One Character
Char$ = Comm1.Input
'--- Tell MSCOMM to read all data in buffer
Comm1.InputLen = 0
'--- Read all the data
AllData$ = Comm1.Input
```

When is it safe to read data?

There are two methods to determine when data has been received.

POLLING METHOD

EVENT DRIVEN METHOD

Polling for Received Data

Polling requires you to monitor the status of the receive buffer, and read data as it is received.

This requires a loop.

With any comm tool, you must constantly check the number of characters that have been received.

The InBufferCount property returns the number of characters currently waiting in the receive buffer for you to read.

Polling for Received Data

This code continually polls the receive buffer, reads in data, and adds it to a string variable.

```
Sub Main ()
   Form1.Show
   Do While DoEvents()
       If Form1.Comm1.InBufferCount Then
           Received$ = Received$ & Comm1.Input
       End If
   Loop
End Sub
```

The form is displayed, and the loop occurs whenever the system is free.

Use the WAITFOR routine

The sample code includes a routine called WaitFor.

WaitFor waits a number of seconds to receive a specific string over a comm port, and returns all received data.

WaitFor is the best way to poll for data when you know what the received data will be, or you know the value of the last character (footer).

Event-Driven Data Processing

- You can tell MSCOMM to fire an event whenever data is received.
- You can set a window, or a number of characters that must be received before the event is fired.
- The RThreshold property provides this function.
  - Setting RThreshold to 1 tells MSCOMM to fire an event after every character has been received, so that you can read it from the receive buffer.
  - Setting RThreshold to 0 disables event-driven received data processing.

The OnComm Event

The OnComm Event is fired whenever a comm event or error occurs.

The CommEvent Property holds the number of the event or error when OnComm fires.

The CommEvent Property is set to MSCOMM_EV_RECEIVE (2) when characters are received.

Event-Driven Received Data Example:

```
Sub Comm1_OnComm ()
  Select Case Comm1.CommEvent
     Case MSCOMM_EV_RECEIVE
        ‘--- Add new data to Received$
        Received$ = Received$ & Comm1.Input
        CR = Instr(Received$, Chr$(13))
        If CR Then
           ‘--- Carriage Return was received
           LastLine$ = Left$(Received$, CR)
           Received$ = Mid$(Received$, CR + 1)
        End If
  End Select
End Sub
```

Common Problems with MSCOMM.VBX

MSCOMM seems to not receive any data.
Out Of Stack Space Error.
Carrier Detect is not always accurate.
The RING event doesn’t always fire.

Causes of Data Not Being Received

- Windows 3.1 Comm Notification.
- RTSEnable Property set to False.
- Handshaking Property set incorrectly.
- RThreshold set to 0.
- TEST:
  - Set RThreshold to 1.
  - Place a Beep in the OnComm event.

Comm Notification

- Windows 3.1 internal method in which the application (MSCOMM) does not poll for received data, but instead is notified by Windows when data is received.
- Not as stable as the Windows 3.0 polling method.
- The first MSCOMM.VBX supported ONLY the Notification method.
- The current MSCOMM.VBX supports both Notification AND polling.

RTSEnable Property Set False

- For devices that use RTS hardware handshaking, the RTSEnable property must be set True.
- Most modems sold today use RTS handshaking.
- The original version of MSCOMM defaults to False.
- The current version of MSCOMM defaults to True

Handshaking Set Incorrectly

Two devices that communicate via serial port MUST use the same handshaking method.

The Handshaking property must be set to reflect the method of handshaking used.

RThreshold Set To 0

The RThreshold property determines the number of characters that must be received before MSCOMM fires the OnComm event (with CommEvent Set To MSCOMM_EV_RECEIVE).

Setting RThreshold to 0 disables the firing of OnComm due to received data.

The data is still received, but if you are counting of OnComm being fired for you to read it, it will appear as if data is not being received.

Out Of Stack Space Error

- Caused by calling DoEvents within the OnComm event procedure.
- DoEvents allows OnComm to be fired, pushing the code address on the stack.
- Eventually, VB runs out of stack space.
- Solution: Temporarily disable events by setting RThreshold to 0 if you must call DoEvents in OnComm. Reset to 1 after exiting the loop.

Carrier Detect

Some modems and serial devices in combination with Windows do not successfully report the status of the CD (Carrier Detect) line.

There is no 100% reliable way to know the status of CD under Windows without writing a new COMM.DRV communications driver or using an existing 3rd party driver.

RING Events do not always fire.

Like Carrier Detect, the RING Indicator line is not always reflected accurately by the Windows COMM.DRV communications driver.

Recommended Methods 1:

Script Procedures

A Script is a dialogue between two computers or devices, in which data is sent, an answer is received, and based on the received data, more data is sent or action is taken.

Use WaitFor

The WaitFor routine in FRANKLIN.BAS is perfect for this kind of dialogue.

An example follows:

Script Example Using WaitFor:

Here is an example script to log onto CompuServe in pseudocode:

```
Send “ATDT 555-1234”
WaitFor “Host Name:”
Send “CIS”
WaitFor “User ID:”
Send “70062,2605”
WaitFor “Password:”
Send “FORGET + IT”
```

Recommended Methods 2 :

- Reading Data from a hardware device that sends data in fixed-length blocks:
  - Some devices spit out data infixed-length blocks of data. For example, a scanner may transmit data in 4K blocks.
- Use OnComm
  - Use OnComm to read a block of data by setting RThreshold to the size of the block. In the above example, set RThreshold to 4096. Also set InputLen to 4096. Then, every time 4K of data is received, the OnComm event will fire (with CommEvent set to MSCOMM_EV_RECEIVE) at which point you can read the data into a string via the Input property.

Sample Code - FRANKLIN.BAS

WaitFor$ - Function that waits for a particular string to be received and returns all received data.

AddCaret and RemoveCaret - Subroutines that convert control codes (^M) to ascii chars within a string.

Pause - Subroutine that pauses for a number of seconds (calls DoEvents).

FRANKLIN.MAK - VB Demo of WaitFor$.

Text #1 - VBPJ Comm Article

- This recently published article contains all the information in this slide show and more.
- Talks about Windows COMM under the hood, as well as Modem Basics.

Text #2 - QB45 to VB Xref

- QuickBasic to Visual Basic Comm Function Cross Reference
- Convert DOS Basic code to Visual Basic
- Quickly Look up functions when programming in Visual Basic