

GENERAL OVERVIEW

The OBD-121 provides several methods to control data flow between it and the host. In this application note, we will present and discuss examples of how a host communicates with the OBD-121 using hardware handshaking. Examples will be given in Visual Basic using the MSCOMM control. A complete simple application can be downloaded from the Bowser Electronics website.

PORT SETTINGS

The OBD-121 uses a fixed set of communication parameters. The host com port must be set the same. The following shows the initial MSCOMM properties.

- 'ComPort' – Whatever port your PC uses
- 'Settings' – '57600,N,8,1'
- 'Handshaking' – 'comRTS'
- 'InBufferSize' – 2048
- 'OutBufferSize' – 512
- 'InputMode' – 'comInputModeText'
- 'InputLen' – 0
- 'SThreshold' – 0
- 'RThreshold' – 0

All data is sent and received as strings. Therefore, each byte of binary data is represented as a character.

FLOW CONTROL

The OBD-121 implements the CTS signal on pin 6. This signal needs to be buffered and inverted using external circuitry to be understood by the host (Refer to application note AN-001). The minimum time to hold the CTS Signal low can be set by using command '0x14'. The default time is 4.096ms. If your PC misses the 'comEvCTS' Events, increase this value by 5mSec at a time until your PC no longer misses the event.

Host hardware control is implemented in the MSCOMM control using the 'Handshaking' property and setting it to 'comRTS' (3) prior to opening the com port. The CTS signal only has an effect on data transmission. When sending data from host to chip, transmission will only take place when the 'CTSHolding' property is 'True'. Care must be taken to ensure the prior request has been completed before loading the next request into the transmit buffer. If more than one message is transmitted to the OBD-121 at once, it will be seen as a transmission error and both requests will be ignored and return an invalid message length error. It is always good practice to clear the send and receive buffers as well as verify the 'CTSHolding' property is 'True' before loading data into the transmit buffer. This will ensure that all responses apply to the request made.

Data may be retrieved in several ways:

1. First, after an 'OnComm' Event where the 'CommEvent' property is equal to 'comEvCTS' and the 'CTSHolding' property is equal to 'True', then, the complete response string will be in the receive buffer.
2. Next, if the response message length is known, the 'Rthreshold' property can be set to the receive length and the 'comEvReceive' Event will be triggered when that number of bytes is received.
3. Finally, non-event programming may be used by implementing a sequence of loops to monitor the 'CTSHolding' property directly after the request message is loaded into the transmit buffer. Loop-one waits to verify CTS goes low after transmit, loop-two waits to verify when CTS goes high. For error detection, some timeout should be implemented to insure the system doesn't loop endlessly. After CTS returns to high, the response message is in the receive buffer.

Once the response message is in the receive buffer, it can be copied to a string variable and decoded. A '0xFF' character will separate multiple response messages from a vehicle. The CRC/Checksum should be verified to ensure the correct end position is picked as data returned from the vehicle may also contain a '0xFF' character.

PROGRAMMING EXAMPLES

Methods one and two, above, are seldom practical as they require prior knowledge about the data returned as well requiring a fixed location to return the data to. Implementing communications as a self contained function allows data to be requested and returned from/to any point anywhere within your program, and provides the most flexibility. The OBD-121FunctionLibrary, in the example application available from the Bowser Electronics website, provides a sample communication function, supporting functions, as well as functions for calculating CRC and Checksum.

Below are examples of code to transmit and receive data to/from the OBD-121. If these are to be implemented in a single function, code should be included to wait for the CTSHolding to transition to 'False' between the transmit and receive blocks.

Sending Data to the OBD-121

```
Dim X, Y, SendStream As String
'Convert Text type Hex String to Binary type data
'SendStream would contain '05686AF10100'
Y = ""
For X = 1 To Len(SendStream) Step 2
    Y = Y & Chr(Hex2Dec(Mid$(SendStream, X, 2)))
Next
'Clear Input and Output Buffers
MSComm1.InBufferCount = 0
MSComm1.OutBufferCount = 0
'Set a start time reference
X = Timer
'Wait for chip ready to accept data
Do
'Exit if more than 5 seconds
If Timer > X + 5 Then Exit Function
Loop Until MSComm1.CTSHolding = True
'Move data to output buffer
MSComm1.Output = Y
```

Reading Data from the OBD-121

```
Dim X, Y, Z
'Set a start time reference
X = Timer
'Wait for chip finished
Do
'Exit if more than 5 seconds
If Timer > X + 5 Then Exit Function
Loop Until MSComm1.CTSHolding = True
'Read Input buffer
X = MSComm1.Input
'Convert to Text type data
Y = ""
For Z = 1 To Len(X)
    Y = Y & Dec2Hex(Asc(Mid$(X, Z, 1)))
Next
```