Global Logger II Communication Protocol

May 12 2006
Roi Peers

Overview

The Global Logger II Communication Protocol is designed to use a subset of the RS232 serial specification. However, as an alternative to a traditional RS232 serial connection, the logger may contain a USB interface that, along with a Virtual Com Port (VCP) driver on the host PC, allows a USB connection to behave as a traditional RS232 port from the point of view of the Logger Interface Software.

The logger RS232 connector is wired as a DCE device (Data Communications Equipment) and can be directly connected to a PC serial port, which is a DTE port (Data Terminal Equipment). The RS232 connection uses 4 wires: Ground, Transmit, Receive, and Data Terminal Ready (DTR). The DTR signal is used to signal the logger that a host wants to communicate with it. When DTR is at ground or below, the logger will allow itself to power down between taking samples. When DTR is asserted high, the logger will awaken and be listening to its RS232 input.

While DTR is normally used to be the “wake” signal to the logger, other active high signals can be used. For a modem, we have successfully used the CD (Carrier Detect) signal to perform this function with a customized cable assembly.

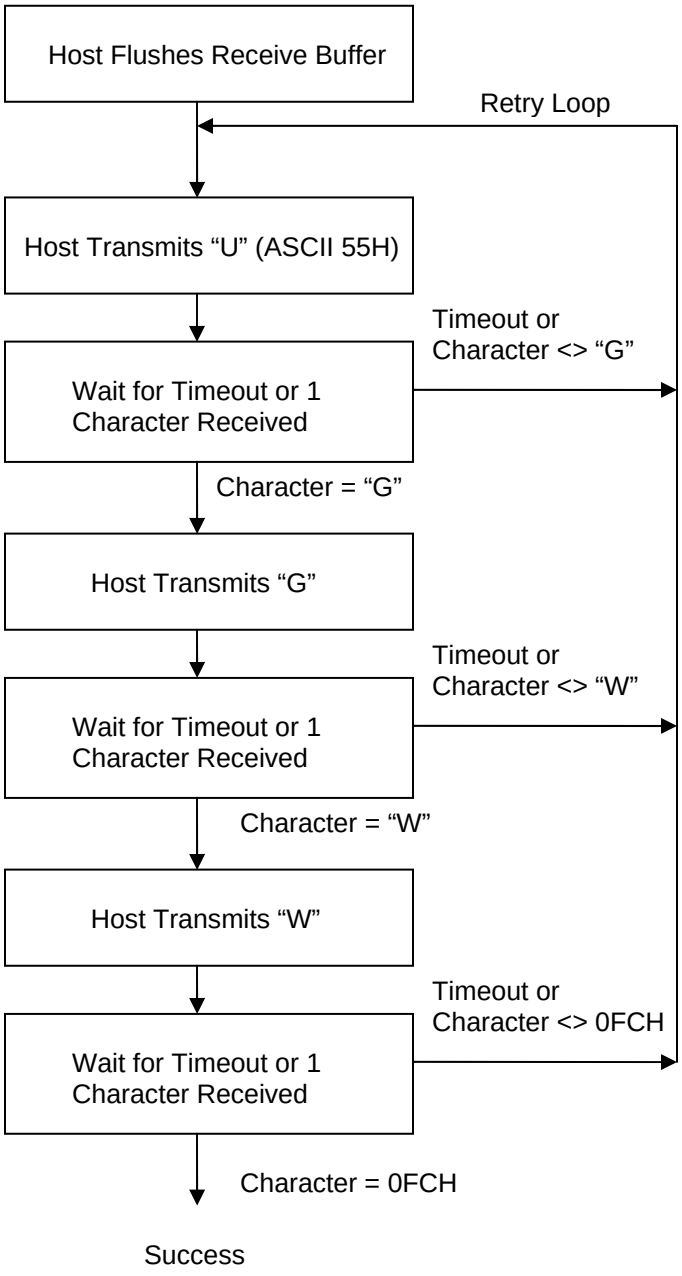
While in host connection mode (DTR asserted), the logger will still perform its scheduled logging functions as much as possible. In high-speed sampling mode, some samples may be missed.

The logger usually is configured to automatically detect the baud rate used by the host (see section on Auto Baud Rate Detect below). However, for some applications, a logger can be configured for a fixed baud rate (9600, 19200, 28800, 38400, 57600 or 115200).

The communication protocol is binary, half-duplex. The host (PC) always initiates each transaction. There is no hardware or software (XON/XOFF) flow control. The host serial port should be configured for 8 Bits, No Parity, and 1 Stop Bit.

Handshaking and Auto Baud Detect

When DTR goes from low to high, the logger will enter a handshaking mode in which it tries to detect the baud rate. The supported baud rates are 9600, 19200, 28800, 38400, 57600 and 115200. The handshake sequence is as follows:



The timeout referred to in the above flowchart can be 100-200 msec typically. For a modem connection, 200-300 msec may be required. Note that each time DTR is de-asserted and subsequently asserted, then the handshake sequence must be repeated.

As an alternative to Auto Baud Rate Detect mode, a logger can be programmed to always use a fixed baud rate. In this case, there is no handshake sequence after DTR is asserted.

Message Encoding

The Global Logger uses binary data rather than ASCII encoded data. Seven values are reserved for special commands characters rather than data.

Special Command Characters

Name	Decimal	Hex	Function
SOT	248	F8	Start of Transmission. The first byte of a new message. Aborts any message in progress being received.
EOT	249	F9	End of Transmission. The last byte of a message.
ACK	250	FA	Acknowledge from Logger. The logger sends this after a legally formatted message is received.
NAK	251	FB	Negative Acknowledge from Logger. The logger sends this if a message had an invalid CRC, or had an unknown command.
IDLE	252	FC	A message has been received and acted upon successfully.
ERROR	253	FD	An error occurred while trying to process a message.
ESCAPE	254	FE	Used to encode data that would otherwise be interpreted as a command, using two bytes: 0FEH + 000H encodes 0F8H as data 0FEH + 001H encodes 0F9H as data . . 0FEH + 006H encodes 0FEH as data

A standard logger packet (in either direction) is formatted as follows:

<SOT> Start of Transmission Byte
<count> Number of bytes of data (may be less than the number transmitted). The <count> includes all the data bytes of the message, *plus two* CRC (cyclic-redundance-checksum) bytes:

<first message data byte>
.
.
<last message data byte>
<crc low data byte>
<crc high data byte>

<EOT> End of Transmission Byte

The CRC is calculated on the <count>-2 data bytes of the original message. An appendix of this document shows a Visual Basic and Pseudo-C implementation of the CRC algorithm.

The count byte can be in the range 3 to 247. Since the <count> includes the 2-byte CRC, this means that the message size can range from 1 to 245 bytes.

The message data bytes and CRC data bytes are subject to escape encoding. If a data byte is in the range 0F8H-0FEH it would be mistaken as a special command byte if transmitted unmodified. These characters are transmitted as two bytes: the ESCAPE command byte (0FEH) followed by a byte with a value of 0 to 6 (to encode 0F8H to 0FEH respectively). Thus the number of bytes of data transmitted may be up to two times the number of bytes in the un-encoded message, and up to two times the number of bytes specified by the <count>

The logger will respond with either an ACK character or NAK character after receiving the EOT based on whether the CRC is correct and the command number is valid.

After the message is processed, the logger will issue a response of either an ERROR character, or IDLE, or a multi-byte response message if data is being transmitted back to the host.

Table of Command Messages

Command Number	Command Name	Description
0	GetTime	Reads the date and time from the logger's timer chip.
1	SetTime	Sets the date and time of the logger's timer chip.
2	Not Implemented	
3	Not Implemented	
4	ResetHistory	Sets the state of the logger's history data buffer to empty.
5	Not Used	
6	EraseBlocks	Erase some or all of the flash memory.
7	WriteFlash	Write data to the flash memory.
8	ReadFlash	Read data from the flash memory.
9	RequestSample	Ask the logger to take a sample.
10	SendSample	Ask the logger to return the sample data from the previous request.
11	Startup	Start the logger with its currently programmed configuration, including alarms, if enabled.
12	GetStatus	Ask the logger to return the number of records in the flash memory, plus state information.
13	GetRecords	Ask the logger to transmit 240 bytes of historical record data.
14	ResetPulseAccs	Resets the current contents of the pulse accumulators.
15	GetVersion	Returns the firmware revision number, number of analog and digital channels, and whether the logger supports sample-on-demand.
16	GetNRecords	Ask the logger to transmit a specific number of historical records.

GetTime

Transmit	000H	
Acknowledge	ACK or NAK	
Response (7 bytes)	Second	BCD 00H-59H
	Minute	BCD 00H-59H
	Hour	Bit 6 should be clear, indicating 24-hour mode. Then, hour is BCD 00H-23H. If bit 6 is set, the timer is incorrectly in 12-hour mode. In 12-hour mode, bit 5 is the PM indicator.
	day of week	1-7 (not used)
	day of month	BCD 00H-31H
	Month	BCD 01H-12H
	Year	BCD 00H (2000) – 99H (2099)

SetTime

Transmit	001H	
	Second	BCD 00H-59H
	Minute	BCD 00H-59H
	Hour	BCD 00H-23H
	day of week	1-7 (not used)
	day of month	BCD 00H-31H
	Month	BCD 01H-12H
	Year	BCD 00H (2000) – 99H (2099)
Acknowledge	ACK or NAK	
Response	IDLE or ERROR	

ResetHistory

Note: this command resets the first and last pointers of the history buffer to the start of the history buffer, signaling an empty state. It does not actually erase the data.

Transmit	004H
	0A4H
	04AH
Acknowledge	ACK or NAK
Response	IDLE or ERROR

EraseBlocks

It is probably never necessary to use this command. A block of flash memory is 256 bytes. It may take up to 5 seconds to erase the entire flash memory.

Transmit	006H
	0A6H
	06AH
	adrs<15:8>
	adrs<19:16>
	count<7:0>
	count<15:8>
Acknowledge	ACK or NAK
Response	IDLE or ERROR

WriteFlash

Used to write configuration data to the logger flash memory. The number of data bytes is: nData = lastAdrs – firstAdrs + 1. The largest number of bytes that can be transferred in a single message is 238.

The logger will return an error if the last address is greater than or equal to 800H. This is to prevent accidentally overwriting the historical sample records.

Transmit	007H
	0A7H
	07AH
	firstAdrs<7:0>
	firstAdrs<15:8>
	lastAdrs<7:0>
	lastAdrs<15:8>
	data, data, ….
Acknowledge	ACK or NAK
Response	IDLE or ERROR

ReadFlash

Used to read configuration data from the logger flash memory. However, since a full 20-bit address is sent to the logger, we can read any location in the logger flash memory. The maximum number of bytes that can be transferred from the logger in a single message is 242. Note that the response message echoes the address information as an extra verification that the logger received the correct address.

Transmit	008H
	firstAdrs<7:0>
	firstAdrs<15:8>
	firstAdrs<19:16>
	Count, 1 to 242
Acknowledge	ACK or NAK
Response	firstAdrs<7:0>
	firstAdrs<15:8>
	firstAdrs<19:16>
	data, data, …

RequestSample

This command will return an error if a sample has already been requested but has not been yet processed by the logger (for example, if it is enforcing the warm-up timer). It is not necessary to issue a SendSample command after each RequestSample command. The logger will leave sensors powered for two seconds after a requested sample is taken.

Transmit	009H
	0: enforce warm-up time 1: skip warm-up time
Acknowledge	ACK or NAK
Response	IDLE or ERROR

SendSample

This command will return an error instead of the requested sample data if no sample is ready. The returned sample record has a 4-byte timestamp, followed by 8 unsigned words of analog samples, followed by 2 unsigned words of pulse accumulators. The returned record always has 8 analog and 2 digital values even if the channels are not implemented on a given logger. A logger with 3 analog channels will return data in the 1st, 2nd and 8th channels.

For more information on the format of a sample record, see the section below on Logger Sample Records.

Transmit	00AH
Acknowledge	ACK or NAK
Response	second ACK, or ERROR
	24-byte sample record (If not ERROR)

Startup

Use this command after modifying configuration information in order to have it take effect. The startup command will restart the alarm sequence, if enabled.

Transmit	00BH
	0ABH
	0BAH
Acknowledge	ACK or NAK
Response	IDLE or ERROR

GetStatus

Returns the number of records in the history buffer, and status flags.

Transmit	00CH
Acknowledge	ACK or NAK
Response	number records<7:0>
	number records<15:8>
	number records<23:16>
	status flags<7:0>
	status flags<15:8>

```
‘ example how to use the flags word to determine the logger state
If (StatusFlags And &H80) = 0 Then
      LoggerState = "Not Logging"
ElseIf (StatusFlags And &H800) <> 0 Then
      LoggerState = "Waiting for Alarm 1"
ElseIf (StatusFlags And &H2000) <> 0 Then
      LoggerState = "Stopped by Alarm 2"
ElseIf (StatusFlags And &H1000) <> 0 Then
      LoggerState = "Logging Until Alarm 2"
Else
      LoggerState = "Logging"
End If
```

GetRecords/GetNRecords

GetRecords will always transmit 240 bytes of record data. For an 8-channel logger, each record is 24 bytes, so 10 records are returned for each GetRecord message. For a 3-channel logger, each record is 12 bytes, so 20 records are returned for each GetRecord message (see section on Logger Sample Records, below).

GetNRecords includes a byte that specifies the number of records to send. The limit is that the total size of all records is less than or equal to 240 bytes.

Both commands will return invalid records of zero data if reading beyond the number of records available in the history data. The count byte in the response message indicates how many records are valid.

Transmit	00DH
	0: send a set of records starting with first record 1: send next set of records 2: resend the previous set of records
Acknowledge	ACK or NAK
Response	number of valid records in data that follows
	240 bytes (12 or 24 records)

Transmit	010H
	number of records request ed
	0: send a set of records starting with first record 1: send next set of records 2: resend the previous set of records
Acknowledge	ACK or NAK
Response	number of valid records in data that follows
	12 or 24 * number requested

ResetPulseAcccs

Transmit	00EH	
	Mask	Bit 0: set in order to clear pulse accumulator #1 Bit 1: set in order to clear pulse accumulator #2
Acknowledge	ACK or NAK	
Response	IDLE or ERROR	

GetVersion

Returns the firmware version number, number of analog channels, number of digital channels, and whether the logger supports sample-on-demand.

Transmit	00FH	
Acknowledge	ACK or NAK	
Response	Firmware Version	8 bytes in response, 4 are reserved for future use.
	Number of analog channels (1-8)	
	Number of digital channels (0-2)	
	0: does not support sample-on-demand 1: does support sample-on-demand	
	Reserved Byte #1	
	Reserved Byte #2	
	Reserved Byte #3	
	Reserved Byte #4	

Flash Organization

Flash Address Range	Description
00000H-00007H	Logger ID. Used to identify that the logger has been initialized. The first 3 bytes will be "GL2" if the logger has been initialized. The next 3 bytes will be the ASCII version of the Global Logger II program that initialized the logger. The remaining 2 bytes are ASCII spaces.
00008H-007FFH	Logger Configuration Data / Application Configuration Data
00800H-FFFFFH	Logged Data

Configuration Data

Note: shaded entries show those fields that affect the operation of the logger. Un-shaded entries are used by the Global Logger II Interface software but not by the logger itself.

Flash Address Range	Description
00008H-00027H	Logger Name (32 ASCII characters). No leading spaces, padded with spaces.
00028H-00029H	Sample Interval, 0 to 65535. 0 is treated the same as 1.
0002AH	Sample Interval Units, 0: seconds, 1: minutes, 2: hours, 3: days.
0002BH-0002CH	<not used>
0002DH	Analog Channel Enable Flags (bit 0 is for channel 1,etc.) (1)
0002EH	Digital Channel Enable Flags (bit 0 is for Pulse Channel 1 etc).
0002FH	Bit 0: Memory Wrap Flag (0: don't wrap, 1: wrap). Bits [4:1]: Baud Rate: 000 Auto Baud Rate Detect 001 Fixed 9600 Baud 010 Fixed 19200 Baud 011 Fixed 28800 Baud 100 Fixed 38400 Baud 101 Fixed 57600 Baud 110 Fixed 115200 Baud 111 Fixed 230400 Baud (Not Used) An un-initialized logger (i.e., ID is not "GL2") will use Auto

	Baud Rate Detect mode.
00030H-00031H	Sensor Warm-up Time in milliseconds. The logger adds 50 to this value, so a value of 0 corresponds to 50 milliseconds, the minimum warm-up time.
00032H	Alarm 1 Enable Flag (0: disabled, 1: enabled).
00033H	Alarm 2 Enable Flag (0: disabled, 1: enabled).
00034H	Sample-on-demand Enable Flag (0: disabled, 1: enabled).
00035H	Bits [1:0]: Logging Style: 00 Fixed Periodic 01 Logarithmic 10 Fast Mode (approximately 10 samples per second) Bit 7: 0: Exception Mode disabled, 1: enabled.
00036H-00037H	Exception Mode Threshold, 0-65535. In Exception Mode, a raw sample must be different than the previous raw sample by greater than this threshold in order to be logged.
00038H-00091H	Engineering Units Names, 9 bytes per channel, 10 channels (8 analog followed by 2 digital). (1)
00092H-00095H	Alarm 1 Time. 4 bytes: Byte 0: seconds BCD 00H-059H Byte 1: minutes BCD 00H-059H Byte 2: hours BCD 00H-023H Byte 3: must by 80H
00096H-00099H	Alarm 2 Time: 4 bytes (see above)
0009AH-000A3H	Displayed Precision, 0-6 decimal places, for 10 channels (8 analog followed by 2 digital). (1)
000A4H-000B3H	Calibrated Raw Low Values, 8 words (analog channels only). (1)
000B4H-000C3H	Calibrated Raw High Values, 8 words (analog channels only). (1)
000C4H-000C9H	Digital Pulse Scale Factors, floating point numbers stored in 3 bytes for each of 2 digital channels.
000CAH-000E1H	Calibrated Engineering Units Low Values, 8 floating point numbers stored in 3 bytes each for 8 analog channels. (1)
000E2H-000F9H	Calibrated Engineering Units High Values, 8 floating point numbers stored in 3 bytes each for 8 analog channels. (1)
000FAH-000FFH	<not used>
00100H-00017H	Meter Display Bottom Value, 8 floating point numbers stored in 3 bytes each for 8 analog channels. (1)
00118H-0012FH	Meter Display Top Value, 8 floating point numbers stored in 3 bytes each for 8 analog channels. (1)
00130H-001FFH	<not used>

Note (1): For a logger with 3 analog channels, the 1st, 2nd, and 8th analog channels are used.

Additional Configuration Data for Flow Monitor

00200H	Flow Monitor Mode: 0: no FM 1: Manning's Equation 2: Flume 3: Weirs 4: Table 255: Flow Monitor Parameters not programmed
00201H-00203H	Manning's Friction (3 bytes floating point)
00204H-00206H	Manning's Slope (3 bytes floating point)
00207H	Slope units: 0: rise/run ratio 1: percent
00208H-0020AH	Manning's Pipe Diameter (3 bytes floating point)
0020BH	Diameter units: 0: feet 1: inches 2: meters 3: cm
0020CH-00213H	Flume Tag: 8 bytes ASCII text identifying the flume type.
00214H-0021BH	Weir Tag: 8 bytes ASCII text identifying the weir type.
0021CH-0021EH	Level Offset (3 bytes floating point)
0021FH	Level Offset Units: 0: feet 1: inches 2: meters 3: cm
00220H	Display Flow Units: 0: cfs 1: gpm 2: mgd 3:cms 4:lps
00221H-00223H	Display Flow Meter Top (3 bytes floating point)
00224H	Display Velocity Units: 0: fps 1: mps
00225H-00227H	Display Velocity Meter Top (3 bytes floating point)
00228H-002A7H	128 bytes path to table file (string padded with spaces)

Floating Point Numbers in Configuration Data

Floating point numbers in the range 999999..-999999 are stored as a 24-bit integer, X[23:0], in 3 bytes:

Byte 0: bits X[7:0]
Byte 1: bits X[15:8]
Byte 2: bits X[23:16]

Then,

M = X[19:0] is the mantissa in the range 0..999999
E = X[22:20] is the negative of the exponent in the range 0..7
S = X[23] is the sign bit: 1 if the value is negative

Then,

$Y = (-1*S) * (M * 10^{(-E)})$

The value Y = 0 is always stored as X[23:0] = 0.
Sample Visual Basic Code is in the appendix showing conversion functions.

Logger Sample Records

Sample records come in two sizes, depending on the type of data logger.

A logger with 8 analog channels and 2 digital channels stores a record in 24 bytes:

- 4 bytes time stamp
- 8 unsigned words analog raw measurement, 12 significant bits left justified.
- 2 words digital pulse count

A logger with 3 analog channels and 1 digital channel stores a record in 12 bytes:

- 4 bytes time stamp
- 3 unsigned words analog raw measurement, 12 significant bits left justified.
- 1 word digital pulse count

Each measurement word is stored least significant byte first.

The time stamp is encoded in four bytes B0, B1, B2 and B3 as follows:

B0[5:0]	seconds, 0-59
B1[3:0] * 16 + B0[7:6]	minutes, 0-59
B2[0] * 16 + B1[7:4]	hours, 0-23
B2[5:1]	day, 0-31

The year and month is stored as a 10-bit number ranging from 0 to 1023 which is the number of months since January, 2000.

B3[7:0] * 4 + B2[7:6]	elapsed months since Jan, 2000

Thus,

Month (1-12) = (ElapsedMonths Mod 12) + 1

Sample Visual Basic code showing how to decode a time stamp is in the appendix.

Appendix A - Visual Basic 6.0 Code to Unpack a Data Record Timestamp

```
Dim second as integer, minute as integer, hour as integer
Dim day as integer, month as integer, year as integer
Dim ts(0 to 3) as Byte  ‘time stamp

.
.
.

second = ts(0) And &H3F
minute = (ts(0) \ 64) + (ts(1) And &HF) * 4
hour = (ts(1) \ 16) + (ts(2) And &H1) * 16

day = (ts(2) \ 2) And &H1F

' there are 10 bits for months starting with jan 2000

month = (ts(2) \ 64) + (ts(3) * 4)
year = (month \ 12)
month = (month Mod 12) + 1

‘ Internationalized date conversion to text string

Dim dd as string, tt as string

dd = FormatDateTime(DateSerial(year + 2000, month, day))

‘ time displayed as 24 hour HH:MM:SS

tt = Format(hour, "00:") & Format(minute, "00:") & Format(second, "00")
```

Appendix B - Visual Basic 6.0 Code to Calculate a CRC for a Message

```
'-----------------------------------------------------
'               CalcCRC
'
' This function calculates a 2-byte CRC (stored as LONG)
' for a string of bytes.
'
' Input:
'   s()     array of bytes
'   b       index first byte to scan
'   e       index of last byte to scan
'
' Output:
'   crc value (0-65535) stored as LONG
'
' The CRC algorithm (in pseudo-C):
'
'   uint16 CalcCRC(uint8 *s, uint16 b, uint16 e)
'   {
'   uint16 crc = 0;
'   uint16 i,j;
'   for (i=b; i <= e; i++) {
'       crc ^= s[i];
'       for (j=0; j < 8; j++) {
'           if (crc & 1) {
'               crc >>= 1;
'               crc ^= 0xA001;
'               }
'           else crc >>= 1;
'           }
'       }
'-----------------------------------------------------

Public Function CalcCRC(s() As Byte, b As Integer, e As Integer) As
Long
    Dim i As Integer, j As Integer
    CalcCRC = 0
    For i = b To e
        CalcCRC = CalcCRC Xor s(i)
        For j = 0 To 7
            If (CalcCRC And 1) <> 0 Then
                CalcCRC = CalcCRC \ 2
                CalcCRC = CalcCRC Xor &HA001&
            Else
                CalcCRC = CalcCRC \ 2
            End If
        Next j
    Next i
End Function
```

Appendix C - Visual Basic 6.0 Code to Encode Floating Point to 24 Bits

```
' convert double float to 3-byte floating point format:
' bits 19:0    mantissa = 0 to 999999
' bits 22:20  -exponent
' bit  23      sign (1 if negative)

Public Function DoubleToLong(ByVal X As Double) As Long
    Dim sign As Long
    Dim exp As Integer
    Dim y As Long
    Dim t As String

    If X = 0 Then
        DoubleToLong = 0
        Exit Function
    End If

    sign = 0
    If X < 0 Then
        X = -X
        sign = &H800000
    End If
    If X > 999999 Then X = 999999

    exp = 0
    Do While X < 99999.95 And exp < 7
        X = 10 * X
        exp = exp + 1
    Loop

    ' The following mess seems to do rounding
    ' the way I want, namely, 2.49 rounds to 2
    ' and 2.50 rounds to 3.0. For some reason
    ' this seems to be a challenge for Visual Basic
    ' and trivial in C.

    X = X + 0.5
    t = CStr(X)   ' convert to string
    X = CDbl(t)   ' and back to double
    X = Int(X)    ' truncate
    y = CLng(X)   ' finally to long

    If y > 999999 Then y = 999999
    y = y + sign
    y = y + CLng(exp) * 1024 * 1024
    DoubleToLong = y
End Function
```

Appendix D - Visual Basic 6.0 Code to Decode Floating Point from 24 Bits

```
' convert 3-byte floating point number as a long to a double:
' bits 19:0    mantissa = 0 to 999999
' bits 22:20  -exponent
' bit  23      sign (1 if negative

Public Function LongToDouble(X As Long) As Double
    Dim y As Double
    Dim exp As Long

    If X = 0 Then
        LongToDouble = 0
        Exit Function
    End If

    y = CDbl(X And &HFFFFF)
    exp = X \ (1024& * 1024&)
    exp = exp And &H7
    Do While exp > 0
        y = y / 10#
        exp = exp - 1
    Loop
    If (X And &H800000) <> 0 Then
        y = -y
    End If
    LongToDouble = y
End Function
```