Application Note

Collecting Ethernet Data from the LI-7700 CH_4 Analyzer on a CR3000[1] or CR1000[1] Datalogger

The LI-7700 Open Path Methane Analyzer is a high speed, high precision methane analyzer designed for eddy covariance applications. It can be used as a standalone device or in conjunction with the LI-7550 Analyzer Interface Unit. When used with the LI-7550, data output options include SDM (Synchronous Devices for Measurement), RS-232, Ethernet, high speed Digital-to-Analog Converters (DACs) and on-board data logging to a USB flash drive. However, when operated as a standalone device, only an Ethernet connection is available from the LI-7700.

In this application note we describe methods for collecting the Ethernet output from an LI-7700 with Campbell Scientific, Inc. dataloggers. The example CRBasic code given here should be applicable to both the CR1000 and CR3000 dataloggers.

Networking

A physical network connection is required from the LI-7700 to the datalogger to allow data collection. On the LI-7700, the network connection is accessed through an eight-pin Turck® connector on the bottom of the instrument. Network connections for the CR1000 and CR3000 are accessed through the instrument's peripheral port using Campbell Scientific's NL115 Ethernet and CompactFlash® Module.

The LI-7700 and datalogger can be connected directly together or via a local area network, but **a direct connection is recommended**. Direct connections eliminate the possibility of additional network traffic, which can slow data transfer and lead to missed data packets by the logger. However they are connected, the logger and the LI-7700 should be configured to operate with static IP addresses on the same subnet. An example network configuration is given in Table 1.

[1] Campbell® Scientific, Inc. Logan, Utah USA

Table 1: Example network configuration. *Ethernet settings are set on the datalogger using the Device Configuration utility in LoggerNet (see the NL115 user manual) and are set on the LI-7700 in the Manual Controls window of the Windows interface software (see the LI-7700 user manual).*

Device	Datalogger	LI-7700
IP Address	172.24.23.60	172.24.23.61
Subnet Mask	255.255.0.0	255.255.0.0
Gateway	0.0.0.0	0.0.0.0
Port	6785	7700

Within the datalogger program, the logger's IP port must be enabled and configured to look for data packets sent from the LI-7700's IP address. This is done using the `TCPOpen` command, which is given below:

```
Public tcpip_socket_status As Long
Dim socket As Long

BeginProg
 TCPClose (101)
 Scan (100,mSec,300,0)
 …
 NextScan

SlowSequence
 Scan (5,Sec,3,0)
  tcpip_socket_status = SerialInChk (socket)
  If (tcpip_socket_status = -1) Then
   socket = TCPOpen ("172.24.23.61",7700,527)
  EndIf
NextScan
```

The value `7700` in the `TCPOpen` command is the IP port where the LI-7700 is located. This is set at the factory for the instrument and will always be 7700. The IP address of this LI-7700 is `172.24.23.61` and is set by the user in the Manual Controls window of the instrument's Windows® interface software. For the datalogger to talk to the LI-7700 the first three octets of their IP addresses should match (e.g. `172.24.23.nn`), but the last octet must be unique to each device. The subnet mask and gateway should be set the same on both devices.

Data output and collection

Data output from the LI-7700 is in the form of tab delimited ASCII text strings followed by a line feed (Figure 1). Upon connecting to the instrument, a set of headers containing labels for the variables included in each record type are sent from the LI-7700 to the datalogger. DATAH and DATASTATH list the variables included in the DATA and DATASTAT records, respectively. Any time the output rate is set to a value greater than zero, new DATA records will be output at the current output rate. DATASTAT is turned off by default in the LI-7700, but if needed it can be enabled by sending the command:

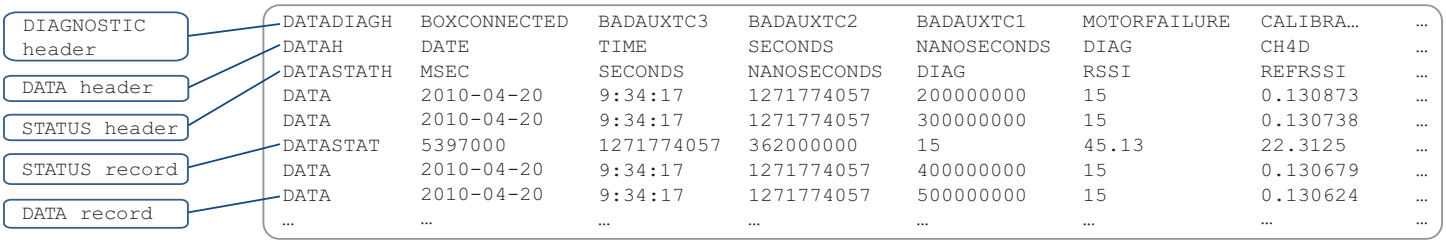

Figure 1. Data output format from the LI-7700. *In this example DATASTAT has been enabled.*

```
<licor><li7700><output><status>true
</status></output></li7700></licor>
```

followed by a line feed. The maximum output frequency for DATASTAT is fixed at 2Hz. When the instrument's output rate is configured for greater than 2Hz, a DATASTAT record will not be output following each DATA record. Most variables included in the DATASTAT record are duplicated by flags in the diagnostic variable included in each DATA record, so it is generally not necessary to continuously collect DATASTAT. DATASTAT is most useful for troubleshooting purposes.

When the output rate is set to a value greater than zero, new DATA records are continuously output from the LI-7700. With the instrument configured this way, the datalogger can be set up to act like a terminal that captures each new DATA record, as in the example code given below:

```
Public LI7700_time(3) As Long
Public LI7700(22)
Public tcpip_socket_status As Long
Dim socket As Long
Dim DATA_string As String * 237
Dim NBR As Long

DataTable (Ethernet_data,TRUE,-1)
  Sample (3,LI7700_time(1),Long)
  Sample (19,LI7700(4),IEEE4)
EndTable

BeginProg
  TCPClose (101)
  Scan (10,mSec,300,0)
    SerialInRecord (socket,DATA_string,&h44,0,&h0A,NBR,01)
    SplitStr (LI7700_time(1),DATA_string,CHR(09),3,4)
    SplitStr (LI7700(1),DATA_string,CHR(09),22,4)

    If NBR>0 Then
      CallTable Ethernet_data
    EndIf

NextScan

SlowSequence
  Scan (5,Sec,3,0)
    tcpip_socket_status = SerialInChk (socket)
    If (tcpip_socket_status = -1) Then
      socket = TCPOpen ("172.24.23.61",7700,527)
  EndIf
NextScan
```

In this example, `SerialInRecord` looks for ASCII strings starting with "`D`" `(&h44)` to define when to start writing data to the variable *`DATA_string`*. `SplitStr` then parses out individual data points from *`DATA_string`* based the occurrence of tab charters `(CHR(09))` in the ASCII string. The first three values included in the DATA record sent from the LI-7700 are 32 bit time stamps. These must be treated differently from the rest of the variables parsed from *`DATA_string`*, otherwise when the datalogger converts them from string to numeric variables they will be rounded to 24 bits by default.

The scan interval used in this example, `Scan(`*`10`*`,mSec,300,0)`, is much faster than the rate at which DATA records would normally be collected and the variables resulting from `SplitStr` are only written to the data table when a new record is received. Data from the LI-7700 should be collected this way because of timing asynchrony between the instrument and the datalogger. In a continuous data collection mode the timing between DATA records is controlled by the LI-7700, but the time between collecting DATA records is controlled by the scan interval of the logger. For a DATA record to be successfully collected by the logger, the record must be output within a certain window when the execution of a scan occurs. Since time keeping is handled separately by the two devices there is no mechanism ensuring this will happen. By using a scan interval that is much faster than the output rate of the LI-7700 the likelihood of missing a record is significantly reduced (Figure 2).

The LI-7700 also supports a polled mode, where the datalogger requests a new DATA record with each scan. When polled, the LI-7700 will return the next DATA record collected by the instrument at its base rate (40Hz) after the poll command is received. This will result in a baseline jitter in the data of 25 milliseconds (Figure2C).

Polling is enabled by setting the output rate to zero and sending the command:

```
<licor><li7700><cmd><poll>true
</poll></cmd></li7700></licor>
```

followed by a line feed whenever a new DATA record is desired.

The `SerialOutBlock` instruction can be used to issue the string and should be placed in the main datalogger program following the *`SerialInRecord`* instruction, as given on the following page:

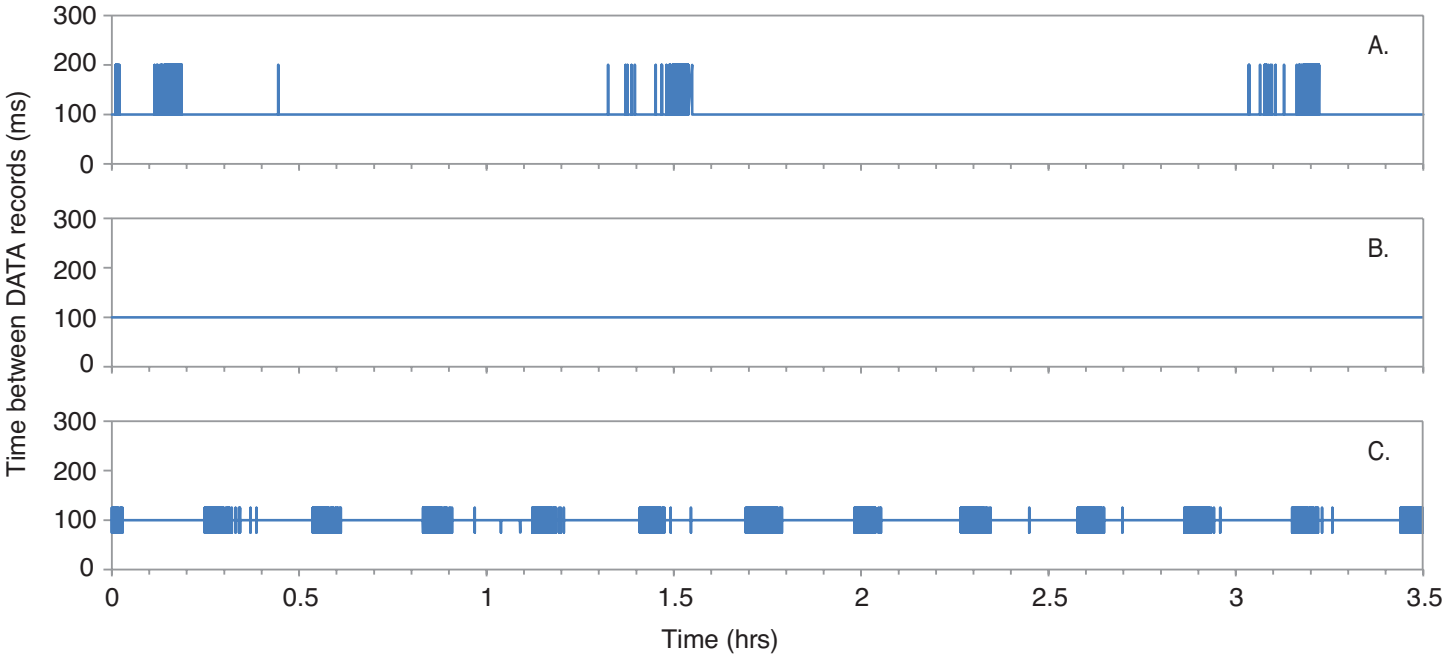

Figure 2: The effect of various scan interval and output rate combinations. *Time between DATA records is based on the LI-7700's time stamp, not that of the datalogger. Panel A shows a 100 ms scan interval and 10 Hz output rate. Using this combination, records were skipped as the two clocks moved in and out of phase, resulting in a 100 to 200 ms jitter in the data. Panel B shows a 10 ms scan interval with a 10 Hz output rate and conditional sampling. No jitter occurred in this configuration. The data in panel C, collected using polling and a 100 ms scan interval, shows considerable base line jitter of 25 ms.*

```
Public LI7700_time(3) As Long
Public LI7700(22)
Public tcpip_socket_status As Long
Dim socket As Long
Dim DATA_string As String * 237
Dim NBR As Long

DataTable (Ethernet_data,TRUE,-1)
  DataInterval (0,0,Sec,100)
  Sample (3,LI7700_time(1),Long)
  Sample (19,LI7700(4),IEEE4)
EndTable

BeginProg
  TCPClose (101)
  Scan (100,mSec,300,0)
    SerialInRecord (socket,DATA_string,&h44,0,&h0A,NBR,01)
    SerialOutBlock (socket,"<licor><li7700><cmd><poll>true</poll></cmd></li7700></
licor>"+CHR(10),61)
    SplitStr (LI7700_time(1),DATA_string,CHR(09),3,4)
    SplitStr (LI7700(1),DATA_string,CHR(09),22,4)
    CallTable Ethernet_data
  NextScan

SlowSequence
  Scan (5,Sec,3,0)
    tcpip_socket_status = SerialInChk (socket)
    If (tcpip_socket_status = -1) Then
      socket = TCPOpen ("172.24.23.61",7700,527)
    EndIf
NextScan
```

Auxiliary sensor data

There are four single ended analog input channels and three thermocouple input channels available on the LI-7700. In applications where it is necessary to have auxiliary sensor data synchronized with the data from the LI-7700 (e.g. wind speed data in an eddy covariance application), the sensor should be connected via these inputs. This will compensate for the clock asynchrony between the datalogger and LI-7700, ensuring that the auxiliary sensor is sampled at the same rate as the LI-7700.

It is important to note that the sensor should be connected directly to the LI-7700, whether operating in polled mode where timing jitter is obvious between records, or when operating in continuous mode with conditional sampling even though there is no jitter between DATA records. This is because in continuous mode, there is an apparent drift in the time between when DATA records are written to the final storage table, which is equal to the scan interval. In the example used here, the datalogger's time stamp shows a 10 ms jitter roughly every five minutes (Figure 3). Using the LI-7700 as the primary interface for the auxiliary sensor ensures this jitter does not affect data quality.

Note: If adding the LI-7700 to a flux station where carbon dioxide and water vapor are measured, it may be necessary to split the output from the sonic anemometer so that it can be sampled by both the datalogger directly and the LI-7700.

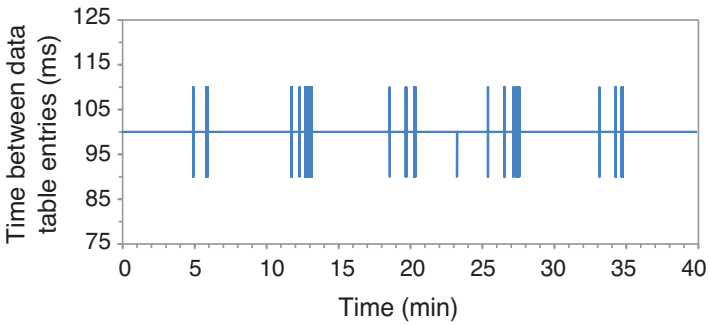

Figure 3: Time between data table entries from the datalogger clock. *Note that while there is a 10 ms jitter, the time between DATA records from the LI-7700 clock is always 100 ms (Figure 2B).*

Appendix A: Example CRBasic program for unprompted data collection. *The example code can be copied and pasted directly into CRBasic, and should compile correctly for both the CR1000 and CR3000.*

```
PipeLineMode

Const Output_interval = 30   'Diagnostic data table output interval.
Const Buffer_Size = 527
Const NBE = 237 'Number of bytes expected

Public LI7700_time(3) As Long
Public LI7700(22)
Public diag_bits(16) As Boolean
Public tcp_close As Boolean
Public tcp_open As Boolean
Public tcpip_socket_status As Long

Alias LI7700_time(1) = milliseconds
Alias LI7700_time(2) = seconds
Alias LI7700_time(3) = nanoseconds
Alias LI7700(4) = Diagnostic
Alias LI7700(5) = CH4_density
Alias LI7700(6) = CH4_mole_fraction
Alias LI7700(7) = Temperature
Alias LI7700(8) = Pressure
Alias LI7700(9) = RSSI
Alias LI7700(10) = Drop_rate
Alias LI7700(11) = Aux(8)
Alias LI7700(19) = TC(3)
Alias LI7700(22) = DATA_checksum
Alias diag_bits(1) = box_connected
Alias diag_bits(2) = bad_aux_tc3
Alias diag_bits(3) = bad_aux_tc2
Alias diag_bits(4) = bad_aux_tc1
Alias diag_bits(5) = motor_failure
Alias diag_bits(6) = calibrating
Alias diag_bits(7) = bottom_heater_on
Alias diag_bits(8) = top_heater_on
Alias diag_bits(9) = pump_on
Alias diag_bits(10) = motor_spinning
Alias diag_bits(11) = block_tmpr_unregulated
Alias diag_bits(12) = laser_tmpr_unregulated
Alias diag_bits(13) = bad_tmpr
Alias diag_bits(14) = ref_unlocked
Alias diag_bits(15) = no_signal
Alias diag_bits(16) = not_ready

Units milliseconds = ms
Units seconds = s
Units nanoseconds = ns
Units CH4_density = mmol/m^3
Units CH4_mole_fraction = umol/mol
Units Temperature = C
Units Pressure = kPa
Units RSSI = %
Units Drop_rate = %
Units TC() = C

Dim socket As Long
Dim DATA_string As String * NBE
Dim NBR As Long 'Number of bytes returned in DATA_string
```

```
Dim checksum_datalogger
Dim checksum_flag As Boolean
Dim diag_work As Long
Dim n

DataTable (Ethernet_data,TRUE,-1)
  Sample (3,milliseconds,Long)
  Sample (19,Diagnostic,IEEE4)
  Sample (1,checksum_datalogger,IEEE4)
EndTable

DataTable (Diagnostic_flags,TRUE,-1)
  DataInterval (0,Output_interval,Min,100)
  FieldNames ("nnd_7700_Tot")'No new data (sensor not connected or powered)
  Totalize (1,n,IEEE4,NBR<>0 IMP checksum_flag)
  FieldNames ("checksum_err_7700_TOT")'Checksum error
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (box_connected))
  FieldNames ("box_connected _TOT")'LI-7550 connected or not
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (bad_aux_tc3))
  FieldNames ("bad_aux_tc3_TOT")'Bad reading at TC3
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (bad_aux_tc2))
  FieldNames ("bad_aux_tc2_TOT")'Bad reading at TC2
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (bad_aux_tc1))
  FieldNames ("bad_aux_tc1_TOT")'Bad reading at TC1
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (motor_failure))
  FieldNames ("motor_failure_TOT")'Mirror spin motor failure
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (calibrating))
  FieldNames ("calibrating_TOT")'Calibration routine enabled
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (bottom_heater_on))
  FieldNames ("bottom_heater_on_TOT")'Bottom mirror heater on
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (top_heater_on))
  FieldNames ("top_heater_on_TOT")'Top mirror heater on
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (pump_on))
  FieldNames ("pump_on_TOT")'Washer pump activated
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (motor_spinning))
  FieldNames ("motor_spinning_TOT")'Bottom mirror spinning
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (block_tmpr_unregulated))
  FieldNames ("block_tmpr_unregulated_TOT")'Block temp not at set point
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (laser_tmpr_unregulated)))
  FieldNames ("laser_tmpr_unregulated_TOT")'Laser temp not at set point
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (bad_tmpr))
  FieldNames ("bad_tmpr_TOT")'Bad TC in optical path
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (ref_unlocked))
  FieldNames ("ref_unlocked_TOT")'Reference signal not line locked
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (no_signal))
  FieldNames ("no_signal_TOT")'No laser signal detected
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (not_ready))
  FieldNames ("not_ready_TOT")'LI-7700 not ready
EndTable

BeginProg
  TCPClose (101)
  n = 1
  Scan (10,mSec,300,0)
    SerialInRecord (socket,DATA_string,&h44,0,&h0A,NBR,01)
    SplitStr (LI7700_time(1),DATA_string,CHR(09),3,4)
    SplitStr (LI7700(1),DATA_string,CHR(09),22,4)
    checksum_flag = (DATA_checksum EQV (CheckSum ("D"&DATA_string,7,NBR-2)))
    checksum_datalogger = CheckSum ("D"&DATA_string,7,NBR-2)
```

```
    'Break up the Diagnostic into 16 separate bits.
    If ( (NBR <> 0) AND (checksum_flag) ) Then
      diag_work = Diagnostic
      box_connected = diag_work AND &h0001
      bad_aux_tc3 = diag_work AND &h0002
      bad_aux_tc2 = diag_work AND &h0004
      bad_aux_tc1 = diag_work AND &h0008
      motor_failure = diag_work AND &h0010
      calibrating = diag_work AND &h0020
      bottom_heater_on = diag_work AND &h0040
      top_heater_on = diag_work AND &h0080
      pump_on = diag_work AND &h0100
      motor_spinning = diag_work AND &h0200
      block_tmpr_unregulated = diag_work AND &h0400
      laser_tmpr_unregulated = diag_work AND &h0800
      bad_tmpr = diag_work AND &h1000
      ref_unlocked = diag_work AND &h2000
      no_signal = diag_work AND &h4000
      not_ready = diag_work AND &h8000
    Else
      Move (milliseconds,3,-99999,1)
      Move (LI7700(1),21,NaN,1)
      Move (diag_bits(1),16,TRUE,1)
    EndIf

    If NBR<>0 Then
      CallTable Ethernet_data
    EndIf
    CallTable Diagnostic_flags
  NextScan

SlowSequence
  Scan (5,Sec,3,0)
    tcpip_socket_status = SerialInChk (socket)

    If ( tcp_close ) Then
      tcp_close = FALSE
      TCPClose (socket)
    EndIf

    If ( (tcpip_socket_status = -1) OR tcp_open) Then
      tcp_open = FALSE
      socket = TCPOpen ("172.24.23.61",7700,Buffer_size)
    EndIf
  NextScan
```

Appendix B: Example CRBasic program for polled data collection. *The example code can be copied and pasted directly into CRBasic, and should compile correctly for both the CR1000 and CR3000.*

```
PipeLineMode

'Measurement Rate               5 Hz      10 Hz     20 Hz
Const Scan_interval = 100     '200 mSec   100 mSec   50 mSec
Const Output_interval = 30    'Diagnostic data table output interval.

Const Buffer_Size = 527
Const NBE = 237 'Number of bytes expected

Public LI7700_time(3) As Long
Public LI7700(22)
Public diag_bits(16) As Boolean
Public tcpip_socket_status As Long

Alias LI7700_time(1) = milliseconds
Alias LI7700_time(2) = seconds
Alias LI7700_time(3) = nanoseconds
Alias LI7700(4) = Diagnostic
Alias LI7700(5) = CH4_density
Alias LI7700(6) = CH4_mole_fraction
Alias LI7700(7) = Temperature
Alias LI7700(8) = Pressure
Alias LI7700(9) = RSSI
Alias LI7700(10) = Drop_rate
Alias LI7700(11) = Aux(8)
Alias LI7700(19) = TC(3)
Alias LI7700(22) = DATA_checksum
Alias diag_bits(1) = box_connected
Alias diag_bits(2) = bad_aux_tc3
Alias diag_bits(3) = bad_aux_tc2
Alias diag_bits(4) = bad_aux_tc1
Alias diag_bits(5) = motor_failure
Alias diag_bits(6) = calibrating
Alias diag_bits(7) = bottom_heater_on
Alias diag_bits(8) = top_heater_on
Alias diag_bits(9) = pump_on
Alias diag_bits(10) = motor_spinning
Alias diag_bits(11) = block_tmpr_unregulated
Alias diag_bits(12) = laser_tmpr_unregulated
Alias diag_bits(13) = bad_tmpr
Alias diag_bits(14) = ref_unlocked
Alias diag_bits(15) = no_signal
Alias diag_bits(16) = not_ready

Units milliseconds = ms
Units seconds = s
Units nanoseconds = ns
Units CH4_density = mmol/m^3
Units CH4_mole_fraction = umol/mol
Units Temperature = C
Units Pressure = kPa
Units RSSI = %
Units Drop_rate = %
Units TC() = C

Dim socket As Long
Dim DATA_string As String * NBE
```

```
Dim NBR As Long 'Number of bytes returned in DATA_string
Dim checksum_datalogger
Dim checksum_flag As Boolean
Dim diag_work As Long
Dim n

DataTable (Ethernet_data,TRUE,-1)
  DataInterval (0,0,Sec,100)
  Sample (3,milliseconds,Long)
  Sample (19,Diagnostic,IEEE4)
  Sample (1,checksum_datalogger,IEEE4)
EndTable


DataTable (Diagnostic_flags,TRUE,-1)
  DataInterval (0,Output_interval,Min,100)
  FieldNames ("nnd_7700_Tot")'No new data (sensor not connected or powered)
  Totalize (1,n,IEEE4,NBR<>0 IMP checksum_flag)
  FieldNames ("checksum_err_7700_TOT")'Checksum error
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (box_connected))
  FieldNames ("box_connected _TOT")'LI-7550 connected or not
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (bad_aux_tc3))
  FieldNames ("bad_aux_tc3_TOT")'Bad reading at TC3
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (bad_aux_tc2))
  FieldNames ("bad_aux_tc2_TOT")'Bad reading at TC2
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (bad_aux_tc1))
  FieldNames ("bad_aux_tc1_TOT")'Bad reading at TC1
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (motor_failure))
  FieldNames ("motor_failure_TOT")'Mirror spin motor failure
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (calibrating))
  FieldNames ("calibrating_TOT")'Calibration routine enabled
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (bottom_heater_on))
  FieldNames ("bottom_heater_on_TOT")'Bottom mirror heater on
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (top_heater_on))
  FieldNames ("top_heater_on_TOT")'Top mirror heater on
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (pump_on))
  FieldNames ("pump_on_TOT")'Washer pump activated
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (motor_spinning))
  FieldNames ("motor_spinning_TOT")'Bottom mirror spinning
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (block_tmpr_unregulated))
  FieldNames ("block_tmpr_unregulated_TOT")'Block temp not at set point
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (laser_tmpr_unregulated)))
  FieldNames ("laser_tmpr_unregulated_TOT")'Laser temp not at set point
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (bad_tmpr))
  FieldNames ("bad_tmpr_TOT")'Bad TC in optical path
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (ref_unlocked))
  FieldNames ("ref_unlocked_TOT")'Reference signal not line locked
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (no_signal))
  FieldNames ("no_signal_TOT")'No laser signal detected
  Totalize (1,n,IEEE4,checksum_flag IMP NOT (not_ready))
  FieldNames ("not_ready_TOT")'LI-7700 not ready
EndTable

BeginProg
  TCPClose (101)
  n = 1
  Scan (Scan_interval,mSec,300,0)
    SerialInRecord (socket,DATA_string,&h44,0,&h0A,NBR,01)
    SerialOutBlock (socket,"<licor><li7700><cmd><poll>true</poll></cmd></li7700></
licor>"+CHR(10),61)
```

```
    SplitStr (LI7700_time(1),DATA_string,CHR(09),3,4)
    SplitStr (LI7700(1),DATA_string,CHR(09),22,4)
    checksum_flag = (DATA_checksum EQV (CheckSum ("D"&DATA_string,7,NBR-2)))
    checksum_datalogger = CheckSum ("D"&DATA_string,7,NBR-2)

    'Break up the Diagnostic into 16 separate bits.
    If ( (NBR <> 0) AND (checksum_flag) ) Then
      diag_work = Diagnostic
      box_connected = diag_work AND &h0001
      bad_aux_tc3 = diag_work AND &h0002
      bad_aux_tc2 = diag_work AND &h0004
      bad_aux_tc1 = diag_work AND &h0008
      motor_failure = diag_work AND &h0010
      calibrating = diag_work AND &h0020
      bottom_heater_on = diag_work AND &h0040
      top_heater_on = diag_work AND &h0080
      pump_on = diag_work AND &h0100
      motor_spinning = diag_work AND &h0200
      block_tmpr_unregulated = diag_work AND &h0400
      laser_tmpr_unregulated = diag_work AND &h0800
      bad_tmpr = diag_work AND &h1000
      ref_unlocked = diag_work AND &h2000
      no_signal = diag_work AND &h4000
      not_ready = diag_work AND &h8000
    Else
      Move (milliseconds,3,-99999,1)
      Move (LI7700(1),21,NaN,1)
      Move (diag_bits(1),16,TRUE,1)
    EndIf

    CallTable Ethernet_data
    CallTable Diagnostic_flags
  NextScan

SlowSequence
  Scan (5,Sec,3,0)
    tcpip_socket_status = SerialInChk (socket)

    If ( tcpip_socket_status = -1) Then
      socket = TCPOpen ("172.24.23.61",7700,Buffer_size)
    EndIf
  NextScan
```

LI-COR®
Biosciences

4647 Superior Street • P.O. Box 4425 • Lincoln, Nebraska 68504
North America: 800-447-3576 • International: 402-467-3576 • FAX: 402-467-2819
envsales@licor.com • envsupport@licor.com • www.licor.com

In Germany – LI-COR GmbH:
+49 (0) 6172 17 17 771 • envsales-gmbh@licor.com • envsupport-gmbh@licor.com

In UK, Ireland, and Scandinavia – LI-COR Biosciences UK Ltd.:
+44 (0) 1223 422102 1 • envsales-UK@licor.com • envsupport-UK@licor.com

979-11300 05/10