uAPP225 (v1.0) July 25,2005

Setting up the CR1000 Data Logger and Microcom GTX Transmitter for TIMED DATA Transmissions

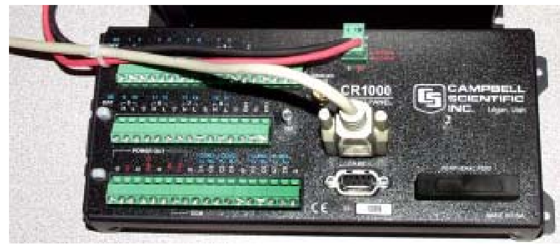
Author: Richard Schwarz

PRELIMINARY

SUMMARY

The Microcom Model GTX Satellite Transmitter and Data Collector works on GOES, GMS, ARGOS, SCD & METEOSAT systems. The GTX has some data logger functions built into it, including an SDI-12 and counter input. The GTX can interface to external data acquisition systems like the Campbell Scientific Programmable Data Logger via its RS-232 port.

Microcom GTX Satellite Transmitter



Campbell Scientific Programmable Data Logger

The GTX is able to transmit logged data at scheduled intervals beginning at a predetermined start time. These are called timed transmissions. The CR1000 Data Logger needs to be able to send commands to the GTX to set up the timed transmissions.

INTRODUCTION

This Application note covers the use of the CR1000 Data logger with the Microcom GTX Satellite Transmitter for timed transmissions. Specific data characteristics as well as transmission windows need to be ascertained to correctly set the timed transmissions up. An example along with code snippets for the CR1000 are shown in the application note to facilitate a timed transmission through the GTX from the CR1000.

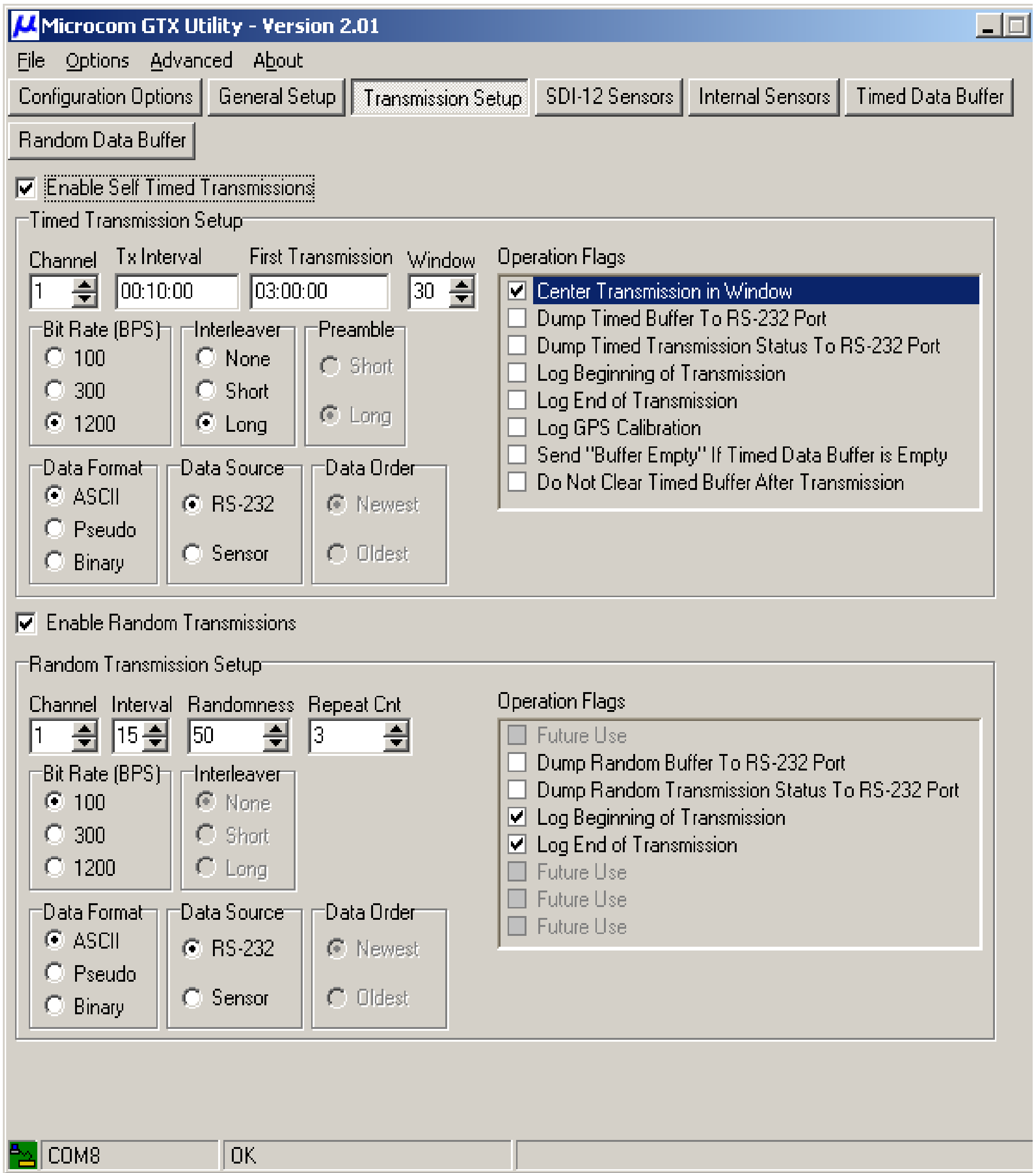
The basic setup and connections for the CR1000 and GTX is referenced in the Microcom Application Note uAPP222.

User's of this application note should be familiar with, or have access to basic coding techniques, user's manuals and software associated with using the GTX and CR1000, including:

1) Microcom GTX User's Manual
2) Microcom GTX GUI Software
3) Campbel Scientific CR1000 User's Manual
4) Campbell Scientific LoggerNet Software

Setting up the GTX for TIMED DATA

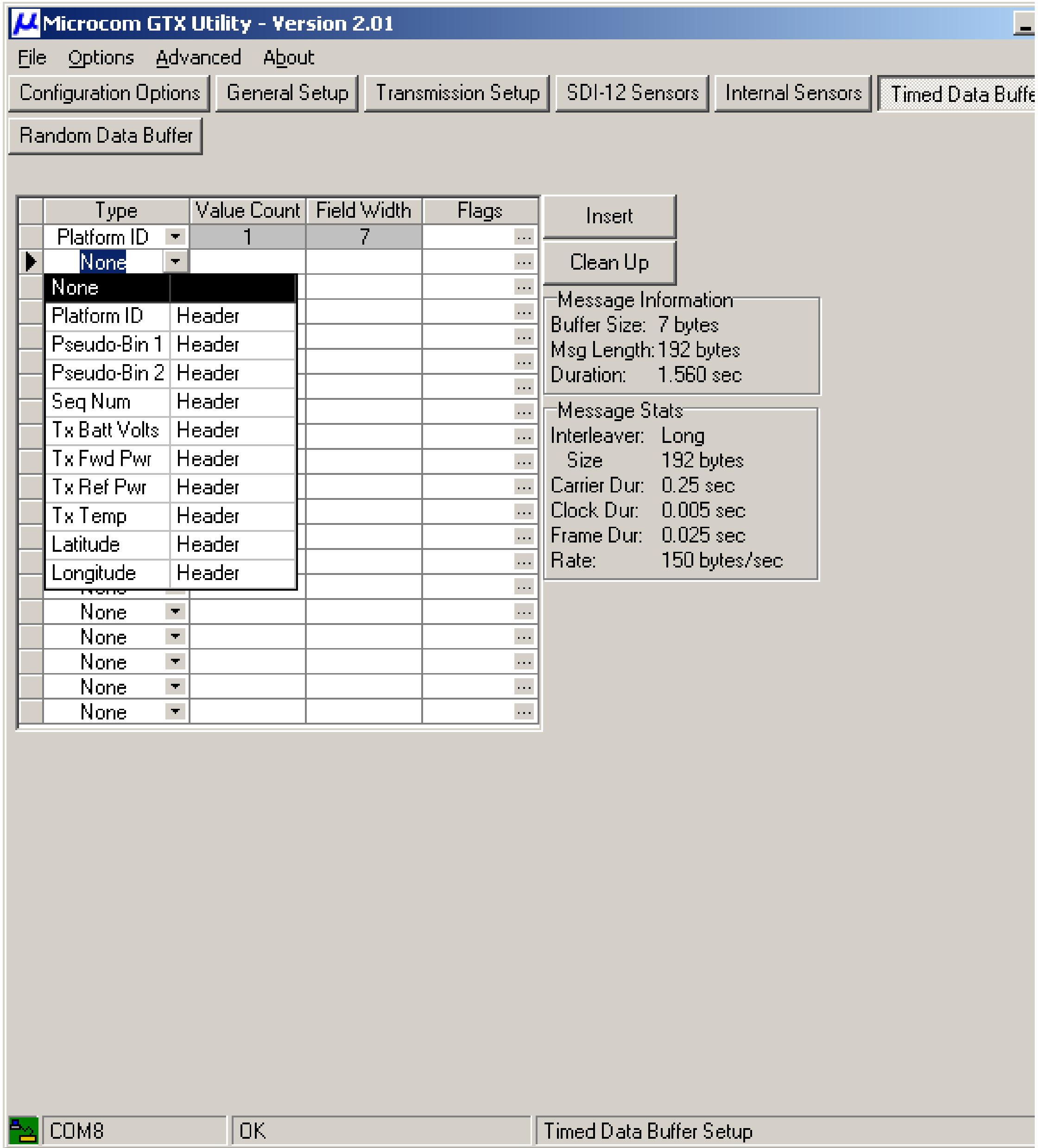
In order to send timed GTX data there are additional considerations which must be taken into account. Examining the GTX Timed Transmission Setup dialogs:



As can be seen the GTX Timed Transmission requires the system engineer to know

1) The GTX TX data rate
2) The GTX TX data Interval
3) The GTX Interleaver Setting
4) The GTX TX HEADER OVERHEAD

The GTX Software automatically calculated the overhead time requirements as can be seen from the screen shot below:



The GTX GUI allows the user to set up custom headers and will produce the required time for the header. In the example shown 1.560 seconds are required for the header portion of the transmission.

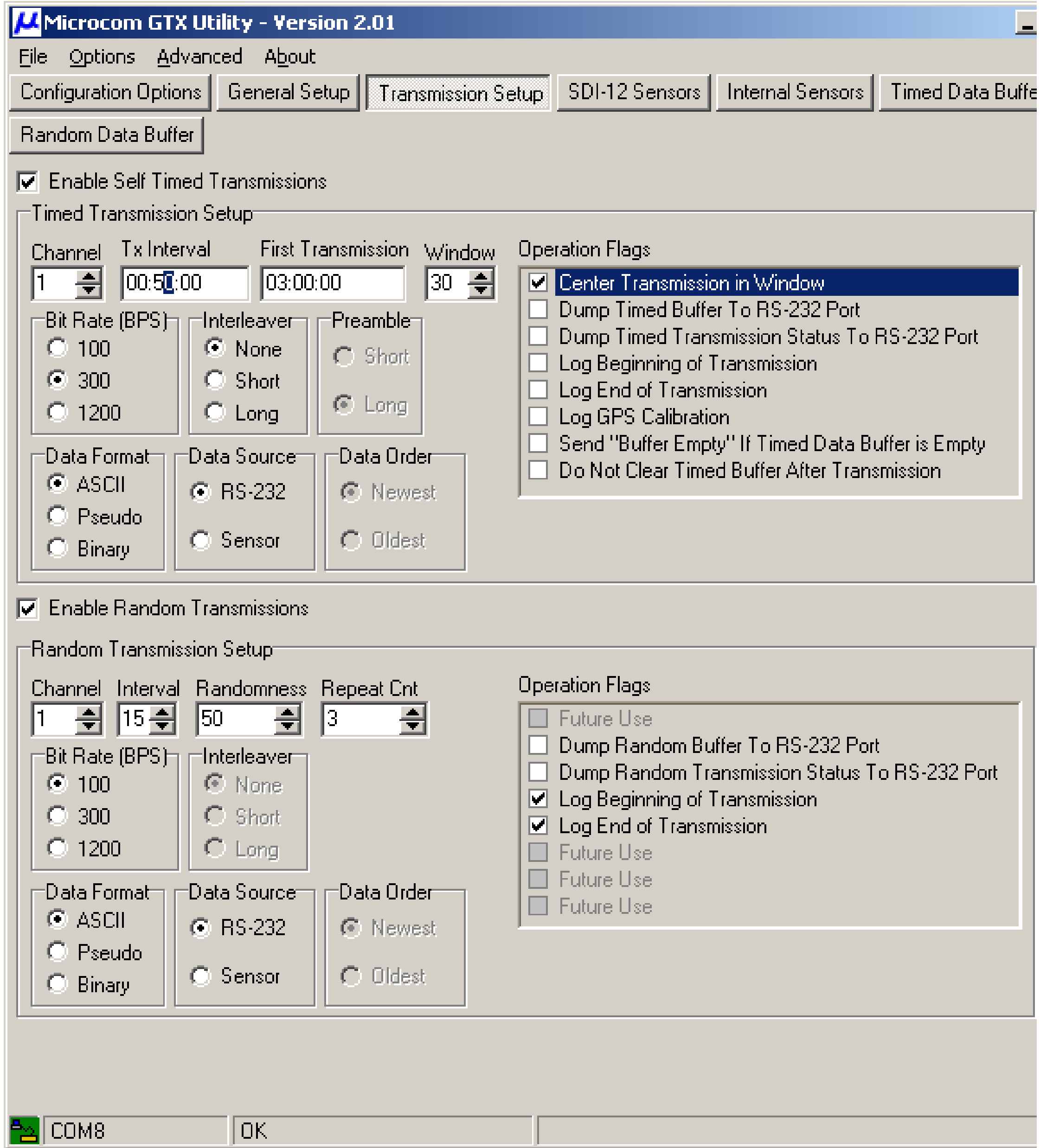
The person setting up a CR1000 / GTX transmission needs to be sure that the collected and forwarded data does not exceed the transmit window provided in the GTX setup.

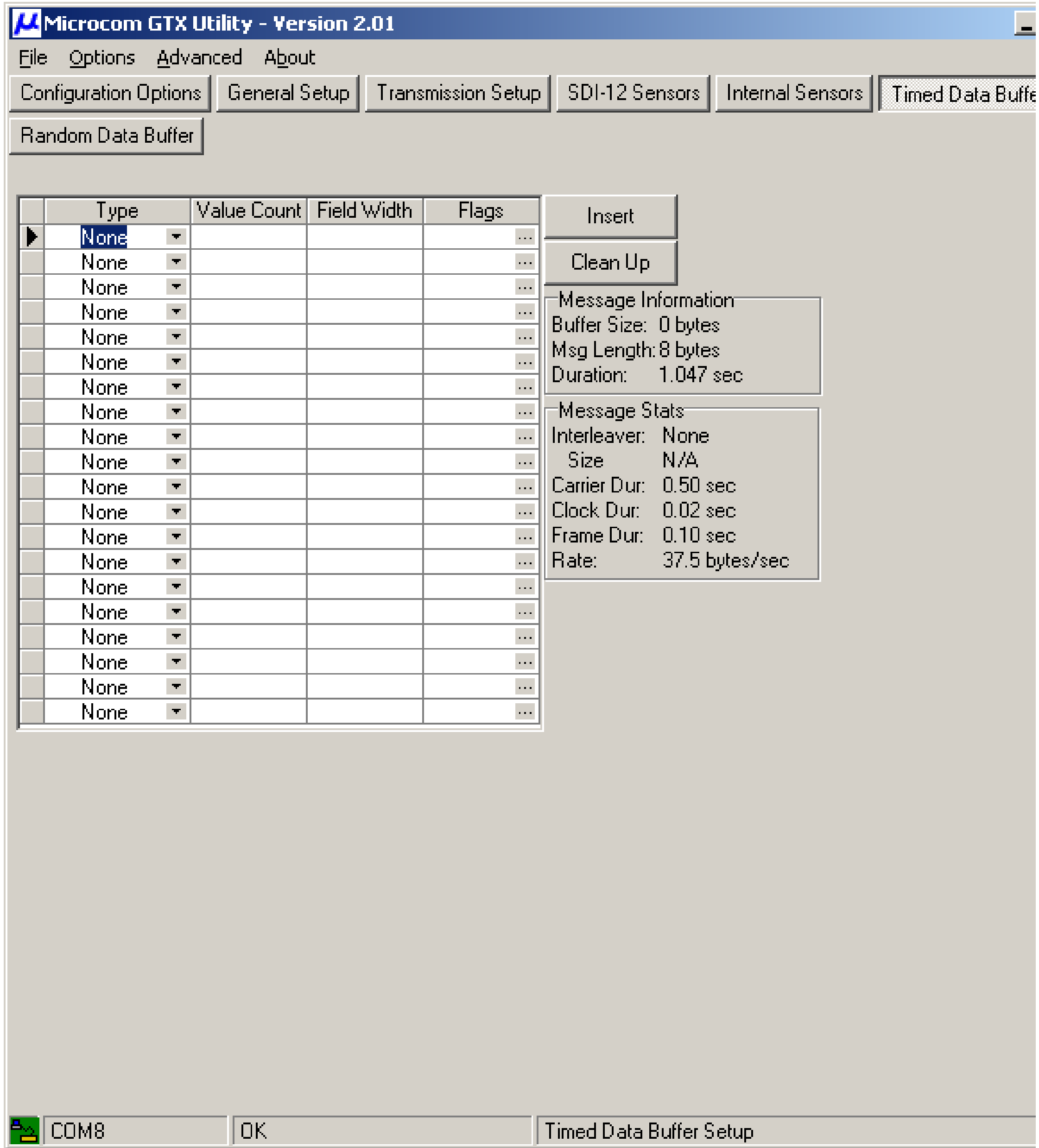
For example, if we sent ten bytes every second we will sending 10*8=80 bits every second. If for example our transmission window was set to 10 seconds, and our TX data interval is 10 minutes, running at a TX rate of 100 bps It is clear that we will exceed (overrun) our data buffer, since we have 80*10*60=48000 bits needing to be sent at 100 bps which would require 480 seconds transmission window (and that's before any header information is calculated.) and we only have a 10 second TX window set.

For example:

Let us now set the GTX TX Duration given the following:

GTX TX Interval: 50 minutes
GTX TX Window: 30 seconds
GTX Data Rate: 300 bps
GTX Interleaver: NONE
GTX HEADER: 1.047 seconds


Microcom GTX Utility - Version 2.01
File Options Advanced About
Configuration Options
General Setup
Transmission Setup
SDI-12 Sensors
Internal Sensors
Timed Data Buffe
Random Data Buffer
Enable Self Timed Transmissions
Timed Transmission Setup
Channel
Tx Interval
First Transmission
Window
1
00:50:00
03:00:00
30
Bit Rate (BPS)
100
300
1200
Interleaver
None
Short
Long
Preamble
Short
Long
Data Format
ASCII
Pseudo
Binary
Data Source
RS-232
Sensor
Data Order
Newest
Oldest
Operation Flags
Center Transmission in Window
Dump Timed Buffer To RS-232 Port
Dump Timed Transmission Status To RS-232 Port
Log Beginning of Transmission
Log End of Transmission
Log GPS Calibration
Send "Buffer Empty" If Timed Data Buffer is Empty
Do Not Clear Timed Buffer After Transmission
Enable Random Transmissions
Random Transmission Setup
Channel
Interval
Randomness
Repeat Cnt
1
15
50
3
Bit Rate (BPS)
100
300
1200
Interleaver
None
Short
Long
Data Format
ASCII
Pseudo
Binary
Data Source
RS-232
Sensor
Data Order
Newest
Oldest
Operation Flags
Future Use
Dump Random Buffer To RS-232 Port
Dump Random Transmission Status To RS-232 Port
Log Beginning of Transmission
Log End of Transmission
Future Use
Future Use
Future Use
COM8
OK



That basically leaves us with ~28 seconds of data we can take (at a 300 bps rate) before we overrun our time window:

This means we can only accept 300 * 28 = 8400 bits of information.

Setting the CR1000 up for Timed Transmission

If we set our CR1000 to periodically transmit the following data :

ABC1234567 (ten bytes=80 bits) every 30 seconds or 2.66 bps, then in the 5 minute window we would accumulate 50*60= 3000 seconds * 1.33 = 7980 bits which is less than the 8400 bits we calculated.

Our CR1000 MAIN routine is shown below with the SCAN statement set to 30 seconds.

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'Main Program''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
BeginProg

  CALL Initialize()

' Infinite loop to be executed every thirty seconds
        Scan (30,Sec,0,0)

'  Send CRs until getting Com4 > " >"
                SerialOut (Com4,CR,">",100,100)
' Send Test MessAGE

  SizeOfGTXData = TMessLen
  GTXData = TMess

   call GTXTimedDataSend()

                ' Get Time from GTX
                Call GetGTXTime()
                ' Set CR1000 Station CLock to the new values
                ClockSet (myTime())

        NextScan
EndProg
```

As can be seen from our code snippet the CR1000 calls a subroutine which we wrote called GTXTimedDataSend. This subroutine is shown below:

```
Sub GTXTimedDataSend
        SerialOut(Com4,"TimedData=","",0,100)
        SerialOut(Com4,GTXData,"",0,100)
         SerialOut (Com4, CRLF,"",0,100)
                If DebugOnOff = 1 Then
          SerialOut (Com1, CRLF,"",0,100)
           SerialOut(Com1,Mess1,"",0,0)
                  SerialOut(Com1,"TimedData=","",0,0)
                         SerialOut (Com1,GTXData,"",0,100)
                          SerialOut (Com1, CRLF,"",0,100)
                EndIf
```

```
 '  Get responce from GTX
            SerialIn (InString,Com4,1000,10,100) 'WAIT FOR RESPONSE
'  Echo stuff to Debug Terminal
            If DebugOnOff = 1 Then
       SerialOut (Com1, CRLF,"",0,100)
             SerialOut(Com1,Mess2,"",0,0)
                  SerialOut (Com1,InString,"",0,100)
                   SerialOut (Com1, CRLF,"",0,100)
            EndIf
'  Get responce from GTX
            SerialIn (InString,Com4,100,10,100)
'  Echo stuff to Debug Terminal
            If DebugOnOff = 1 Then
                  SerialOut (Com1, CRLF,"",0,100)
                  SerialOut(Com1,Mess2,"",0,100)
                  SerialOut (Com1,InString,"",0,100)
                   SerialOut (Com1, CRLF,"",0,100)
            EndIf

EndSub
```

Inside the subroutine it can be seen that a serial command sequence is sent to the GTX called TimedData=. This is a GTX command to send timed data. The specifics of this command are shown below:

TimedData=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (TDT,etx)

The **TDT** command appends host, formatted data to the Timed buffer when the **TimedDataSource** is RS-232. This command is not permitted and will return an error response (ERR) when the **TDS** is Sensor, or whenever the transmitter is disabled.

Prior to sending a timed transmission, the transmitter will insert the appropriate preamble and programmed DCP ID, any header information (e.g. HDR flag byte and/or sequence number), and for GOES operation append the appropriate EOT. If the **TimedDataFormat** is ASCII or Pseudo-Binary the transmitter will also insert the correct parity bit for each message character.

The maximum length of the formatted data can be up to 126000 bits, or 15750 bytes (see Section **Error! Reference source not found.**). However, the actual buffer size is calculated based on the **TimedWindowLength**, i.e. the transmitter will not accept more bytes than can be sent in the programmed window length at the configured BPS format.

If this command is received when a transmission is initiated and pending (approximately 5 seconds before the scheduled transmission) or during a timed transmission, the data will not be included in the current transmission but will be buffered for the next timed transmission. When a timed transmission is complete, the transmitted data will be automatically cleared from the timed buffer.

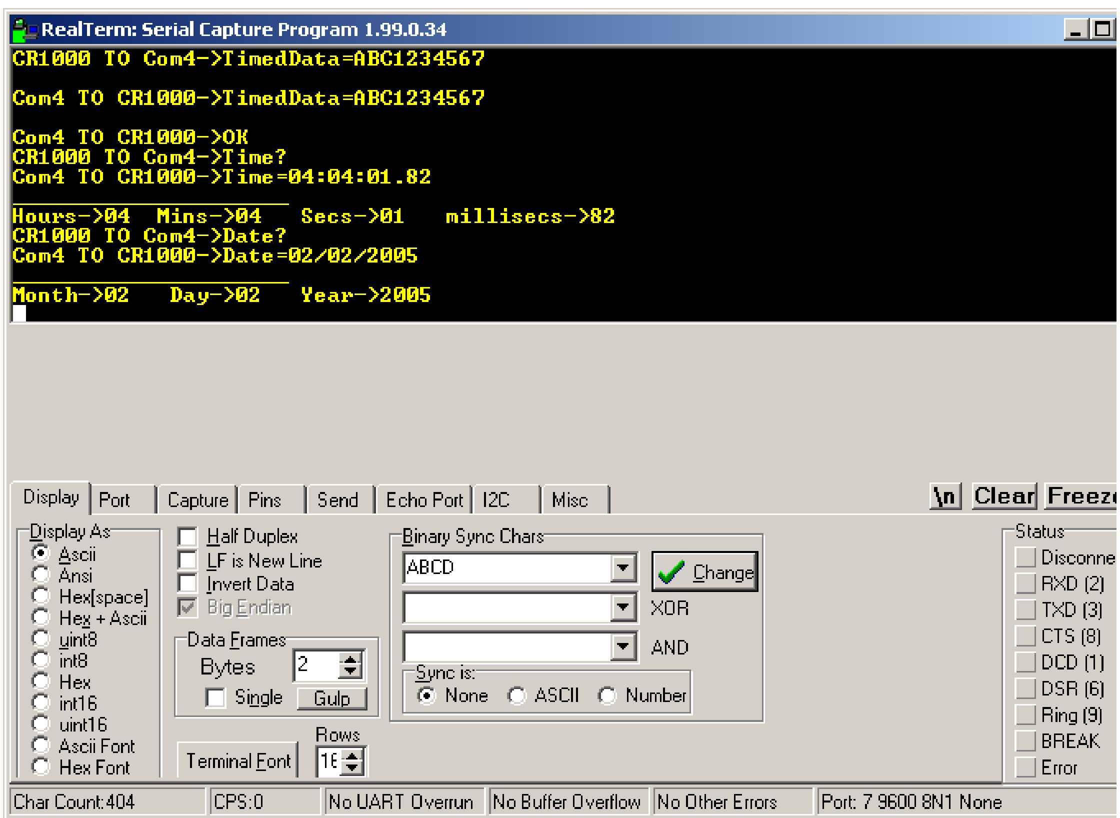
The transmitter responds with: **[CR][LF]>** if the data is accepted.
ERR[CR][LF]> if the buffer is full

Note that the transmitter will not prevent any prohibited ASCII characters from being loaded into the buffer. Instead, these characters will be replaced with a valid ASCII character before being transmitted if the unit is configured for ASCII or Pseudo-Binary operation.

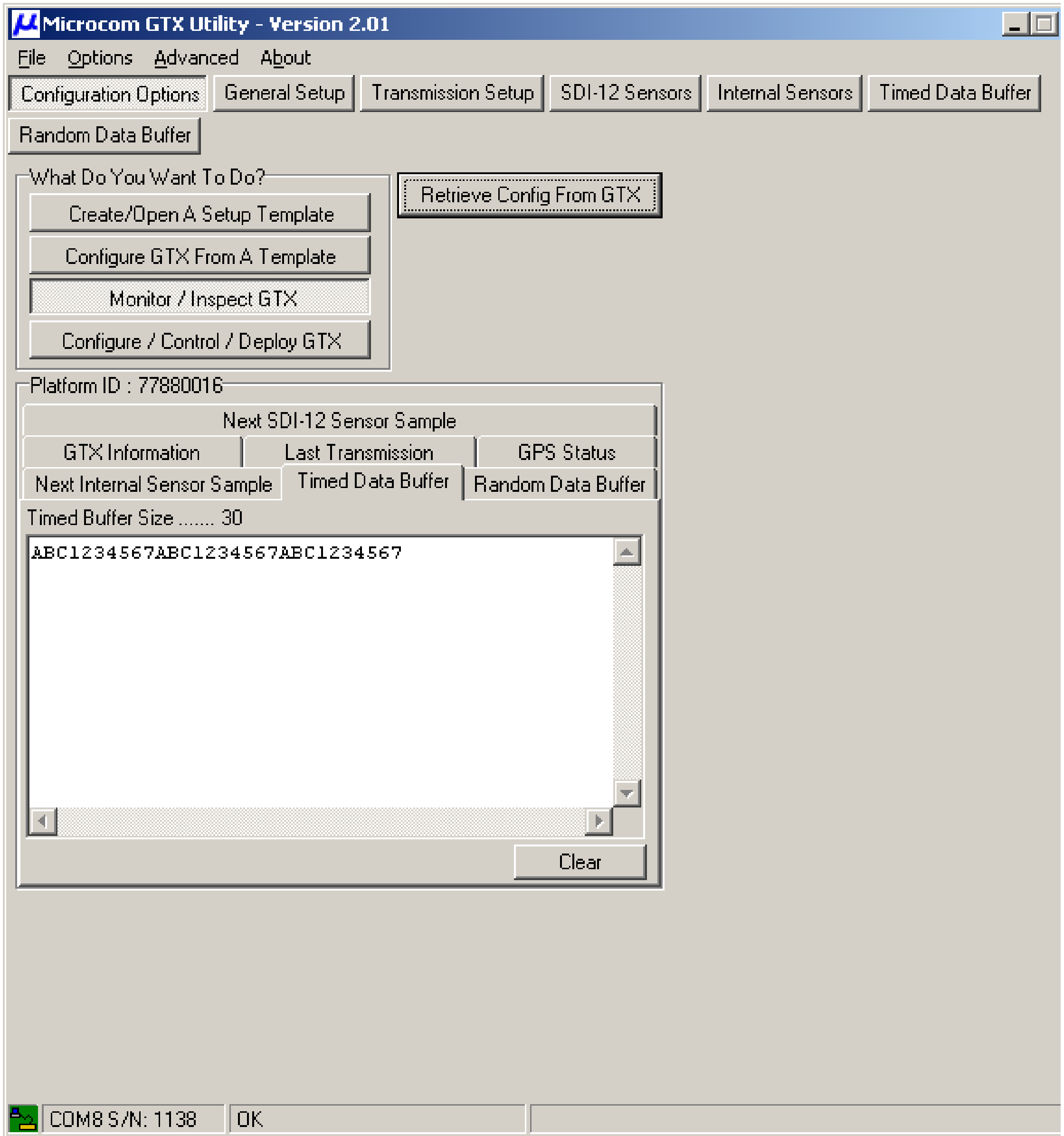
The unique nature of this command, requires several important distinctions from other commands to be noted.

- This command will only be accepted when the transmitter is enabled.
- Once the equals sign is received, the command itself may not be edited. In other words, backspacing to the point where the equals sign is deleted will terminate the command.
- While other commands have a predetermined number of parameters, the amount of data that can be loaded with this command is only limited by the buffer size as determined by the transmit window length.
- If the timed data buffer is exceeded for a **TimedData** command, it is treated as receiving an [ESC], and none of the data received during the command will be loaded.

The terminal Debug output from our CR1000 program is shown below



The GTX Timed buffer can be examined in the GTX GUI software as shown below (here we have only sent 3 messages or 90 seconds worth)



If we do send data at too high a rate, the GTX will refuse to accept the data and will send an ERR instead of the OK message.

Interleaver

Interleaving is a key component of many digital communication systems involving forward error correction (FEC) coding. Interleaving the encoded symbols provides a form of time diversity to guard against localized corruption or bursts of errors. For example, if we interleave our data into rows and columns by taking in our data into a matrix in rows, and then read out our data in columns, we spread out any burst errors (errors which happen in successive bytes). Most Error Correction schemes have a limit on how many errors can be corrected per blocks or sections of data. For example an error correction scheme may be able to correct 3 errors per X bytes. Many data errors tend to happen in bursts, that is, grouped together. If we could spread out (interleave) the data (before applying the Error correction), any errors introduced in bursts onto an interleaved data transmission will be spread out when the data is de-interleaved at the receiver. As an example lets use words and letters: Lets assume our error corrector can only handle 1 error per word.

Using the ? to represent spaces:

The?quick?brown?fox?jumped?over?the?lazy?dogs?back

contains 50 characters. If we used a 5 by 10 interleaver the data would become

T h e ? q u I c k ?
b r o w n ? f o x ?
j u m p e d ? o v e
r ? t h e ? l a z y
? d o g s ? b a c k

Tbjr?hru?deomto?wphgqneesu?d??lf?lbcooaakxvzc??eyk

Well you can see that this is unrecognizable but once we deinterleave it with a 10 x 5

Tbjr?
hru?d
eomto
?wpeh
qnees
u?d??
if?lb
cooaa
kxvzc
??eyk

we get out:

The?quick?brown?fox?jumped?over?the?lazy?dogs?back

Now lets introduce 4 errors in a row

The quick brown fox XXXXed over the lazy dogs back (errors in 21,22,23,24)

It changed or masks the meaning totally.

however lets put the errors in the same place in the interleaved data stream

Tbjr?hru?deomto?wphgXXXXsu?d??lf?lbcooaakxvzc??eyk (errors in 21,22,23,24)

now lets deinterleave:

Tbjr?
hru?d
eomto
?wphg
XXXXs
u?d??
lf?lb
cooaa
kxvzc
??eyk

The Xuick browX fox jumpXd over thX lazy dogs back.

Now the errors are spread out over 4 words but each individual word only has one error and our error correction can handle that.

In the GTX Transmitter has two interleavers. The short interleaver is 96 long, while the long interleaver is 192 long.

The key thing to remember here is that as soon as the byte count goes over the interleaver length, a new interleaver block needs to be added to the byte count. In other words if we have 193 bytes to interleave, and we are using a long (192 long) interleaver, we need two interleavers or (2*192)= 384 bytes.

CODE EXAMPLE

We listed the MAIN function of the CR1000 code above. The entire program is listed below. It is worthy to note here that most applications are going to want to set the CR1000 clock from the more accurate GTX clock. In order to facilitate this a sub program was written for the CR1000 and is shown here:

Debug code is included but can be turned off by setting DebugOnOff = 0.

The entire code example is shown below:

```
' Program: GTXTimed.cr1
```

```
' Description: GTX Timed Data Transmission Program
' CR1000 Series Datalogger forwards a "TimedData="
' to the GTX in  Mode. Time and Date is read from the GTX using
' Subroutine GetGTXTime
'
'program author: R. Schwarz  Microcom Design Incorporated
'program date: Original Code May 1, 2005
'
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'Declare Public Variables''''''''''''''''''''''''''''''''''''''''''''''''''''
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''


Public PTemp,I,TMessLen
Public SizeOfGTXData
Public GTXData as string * 1000
Public CR as string * 1
Public ESC as string * 1
Public BS as string * 1
Public SLASH as string * 1

Public LF as string * 1
Public CRLF as string * 2
Public PROMPT as string * 1
Public myTime(7)
Public Dest(9)
Public InString as string * 100
Public TMess as string * 1000
Public ValStr(4) as string * 100
Public LINE as string * 100
Public Mess1 as string * 100
Public Mess2 as string * 100
Public Batt_volt
'
Units Batt_Volt=Volts
'
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'Constant Declarations''''''''''''''''''''''''''''''''''''''''''''''''''''''
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Const DebugOnOff = 1



''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'Alias Declarations''''''''''''''''''''''''''''''''''''''''''''''''''''''''
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Alias myTime(1) = Year          'assign the alias Year to rTime(1)
Alias myTime(2) = Month         'assign the alias Month to rTime(2)
Alias myTime(3) = Day 'assign the alias Day to rTime(3)
Alias myTime(4) = Hours         'assign the alias Hour to rTime(4)
Alias myTime(5) = Mins'assign the alias Minute to rTime(5)
```

```
Alias myTime(6) = Secs        'assign the alias Second to rTime(6)
Alias myTime(7) = uSecs       'assign the alias uSecond to rTime(7)


Sub GTXTimedDataSend
        SerialOut(Com4,"TimedData=","",0,100)
        SerialOut(Com4,GTXData,"",0,100)
         SerialOut (Com4, CRLF,"",0,100)
                If DebugOnOff = 1 Then
          SerialOut (Com1, CRLF,"",0,100)
           SerialOut(Com1,Mess1,"",0,0)
                  SerialOut(Com1,"TimedData=","",0,0)
                        SerialOut (Com1,GTXData,"",0,100)
                         SerialOut (Com1, CRLF,"",0,100)
                EndIf

 '  Get responce from GTX
                SerialIn (InString,Com4,1000,10,100) 'WAIT FOR RESPONSE
'  Echo stuff to Debug Terminal
                If DebugOnOff = 1 Then
          SerialOut (Com1, CRLF,"",0,100)
                  SerialOut(Com1,Mess2,"",0,0)
                        SerialOut (Com1,InString,"",0,100)
                         SerialOut (Com1, CRLF,"",0,100)
                EndIf
'  Get responce from GTX
                SerialIn (InString,Com4,100,10,100)
'  Echo stuff to Debug Terminal
                If DebugOnOff = 1 Then
                        SerialOut (Com1, CRLF,"",0,100)
                        SerialOut(Com1,Mess2,"",0,100)
                        SerialOut (Com1,InString,"",0,100)
                         SerialOut (Com1, CRLF,"",0,100)
                EndIf

EndSub


''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'Define Subroutines''''''''''''''''''''''''''''''''''''''''''''''''''''''
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Sub Initialize()
' Use Com1 as an ECHO output port to viwe on Com1
 SerialOpen (Com1,9600,0,0,10000)
' Setup Com4 serial port to GTX
 SerialOpen (Com4,9600,0,0,10000)
  CR=CHR(13)
  LF=CHR(10)
  BS=CHR(8)
  ESC=CHR(27)
```

```
  SLASH="/"
  CRLF=CHR(13)+CHR(10)
  PROMPT= ">"
  LINE="____________________"
  Mess1="CR1000 TO Com4->"
  Mess2="Com4 TO CR1000->"
  TMess="ABC1234567"
  TMessLen = 10
EndSub
''''''''''''''''''''''''''''''''''''''''''''''
'  Get the Microcom GTX Time
''''''''''''''''''''''''''''''''''''''''''''''
Sub GetGTXTime()
'   Request Time from GTX
                SerialOut(Com4,"Time?","",0,100)
                SerialOut (Com4, CRLF,"",0,100)
'   Echo stuff to Debug Terminal
                If DebugOnOff = 1 Then

                        SerialOut(Com1,Mess1,"",0,100)
                        SerialOut(Com1,"Time?","",0,100)
                        SerialOut (Com1, CRLF,"",0,100)
                EndIf
'   Get GTX Responce
                SerialIn (InString,Com4,100,10,100)

'   Echo stuff to Debug Terminal
                If DebugOnOff = 1 Then
                        SerialOut(Com1,Mess2,"",0,100)
                        SerialOut (Com1,InString,"",0,100)
                EndIf

                SerialIn (InString,Com4,100,10,100)
'   Echo stuff to Debug Terminal
                If DebugOnOff = 1 Then
                        SerialOut(Com1,Mess2,"",0,100)
                        SerialOut (Com1,InString,"",0,100)
                        SerialOut (Com1, CRLF,"",0,100)
                        SerialOut (Com1,LINE,"",0,100)
                        SerialOut (Com1, CRLF,"",0,100)
                EndIf

                'Breakup Time String
                SplitStr (ValStr(1),InString,":",1,0)
                SplitStr (ValStr(2),InString,":",2,4)
                SplitStr (ValStr(4),ValStr(3),".",1,4)
                SplitStr (ValStr(3),ValStr(3),".",1,5)

           'Assign Hours
                Hours = ValStr(1)
```

```
If DebugOnOff = 1 Then
        SerialOut(Com1,"Hours->","",0,100)
        SerialOut(Com1,ValStr(1),"",0,100)
        SerialOut (Com1," ","",0,100)
EndIf

'Assign Mins
Mins = ValStr(2)

If DebugOnOff = 1 Then
        SerialOut(Com1,"Mins->","",0,100)
        SerialOut(Com1,ValStr(2),"",0,100)
        SerialOut (Com1," ","",0,100)
EndIf

'Assign Seconds
Secs = ValStr(3)

If DebugOnOff = 1 Then
        SerialOut(Com1,"Secs->","",0,100)
        SerialOut(Com1,ValStr(3),"",0,100)
        SerialOut (Com1," ","",0,100)
EndIf

'Assign Microsecs
uSecs = ValStr(4)

If DebugOnOff = 1 Then
        SerialOut(Com1,"millisecs->","",0,100)
        SerialOut(Com1,ValStr(4),"",0,100)
        SerialOut (Com1, CRLF,"",0,100)
EndIf

'Request Date from Com4
SerialOut(Com4,"Date?","",0,100)
SerialOut (Com4, CRLF,"",0,100)
SerialIn (InString,Com4,100,10,100)
SerialIn (InString,Com4,100,10,100)

If DebugOnOff = 1 Then
        SerialOut(Com1,Mess1,"",0,100)
        SerialOut(Com1,"Date?","",0,100)
        SerialOut (Com1, CRLF,"",0,100)
        SerialOut(Com1,Mess2,"",0,100)
        SerialOut (Com1,InString,"",0,100)
 SerialOut(Com1,Mess2,"",0,100)
 SerialOut (Com1,InString,"",0,100)
 SerialOut (Com1, CRLF,"",0,100)
 SerialOut (Com1,LINE,"",0,100)
 SerialOut (Com1, CRLF,"",0,100)
EndIf
```

```
                'Break up Date String
                SplitStr (ValStr(1),InString,"/",1,0)
                SplitStr (ValStr(2),InString,"/",2,4)

                'Assign Month
                Month = ValStr(1)

                If DebugOnOff = 1 Then
                        SerialOut(Com1,"Month->","",0,100)
                        SerialOut(Com1,ValStr(1),"",0,100)
                        SerialOut (Com1, "   ","",0,100)
                EndIf

                'Assign Day
                Day = ValStr(2)

                If DebugOnOff = 1 Then
                        SerialOut(Com1,"Day->","",0,100)
                        SerialOut(Com1,ValStr(2),"",0,100)
                        SerialOut (Com1, "   ","",0,100)
                EndIf

                'Assign Year
                Year = ValStr(3)

                If DebugOnOff = 1 Then
                        SerialOut(Com1,"Year->","",0,100)
                        SerialOut(Com1,ValStr(3),"",0,100)
                        SerialOut (Com1, CRLF,"",0,100)
                EndIf

EndSub



''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'Main Program''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

BeginProg

  CALL Initialize()

' Infinite loop to be executed every thirty seconds
        Scan (30,Sec,0,0)

'  Send CRs until getting Com4 > " >"
                SerialOut (Com4,CR,">",100,100)
' Send Test MessAGE

  SizeOfGTXData = TMessLen
```

```
            GTXData = TMess

             call GTXTimedDataSend()

                                   ' Get Time from GTX
                                   Call GetGTXTime()
                                   ' Set CR1000 Station CLock to the new values
                                   ClockSet (myTime())

                    NextScan
          EndProg
```

Conclusion

The Microcom Model GTX 1.0 is a highly versatile yet easy-to-use Satellite Transmitter intended for use in a wide variety of satellite based meteorological data collection applications. While the GTX transmitter can operate as a stand-alone data collection platform it can also be used with an external data logger, like the CR1000.

Data transmissions sent through the GTX can be scheduled for certain times. These timed transmissions require specific setup instructions to be sent to the GTX. The CR1000 data logger can be programmed to send these specific instructions to the data logger.

The transmitted data parameter settings such as data rate and interleaver settings as well as the amount of data being sent from the logger all need to be considered in order to set up a correct transmission.

Revision History

Date	Version	Revision
July 12, 2005	1.0	Initial Microcom Preliminary Release