Lab exercise E1: Operation of a Campbell Scientific CR3000 datalogger

Field Phenomics Workshop, Maricopa, Arizona
Tuesday, April 8, 2014 13:30
Kevin Wang (xuwang@k-state.edu), Matthew Conley (matthew.conley@ars.usda.gov)

Objectives:

1. Study the basics of a Campbell Scientific CR3000 datalogger
2. Develop basic Campbell Scientific datalogger programming skills
3. Log a MaxBotix ultrasonic ranging sensor outputs using CR3000
4. Logging GPS receiver's NMEA string output using a CR3000

Introduction:

The Campbell Scientific CR3000 datalogger is a versatile stand-alone data acquisition system with signal-processing and control capabilities. CR3000 dataloggers provide precision measurement capabilities in a rugged, battery-operated package and provides serial communications to data storage peripherals, GPS receivers, modem devices, and computers etc.

During this lab exercise, we will review some basic programming skills needed to operate a CR3000 datalogger.

Equipment:

1. Campbell Scientific CR3000 datalogger
2. USB toRS232 interface adaptor cable
3. MaxBotix #7364 ultrasonic ranging sensor
4. GPS simulator NMEA Generator
5. LoggerNet datalogger support software
6. Laptop computer
7. 12V battery

Procedure:

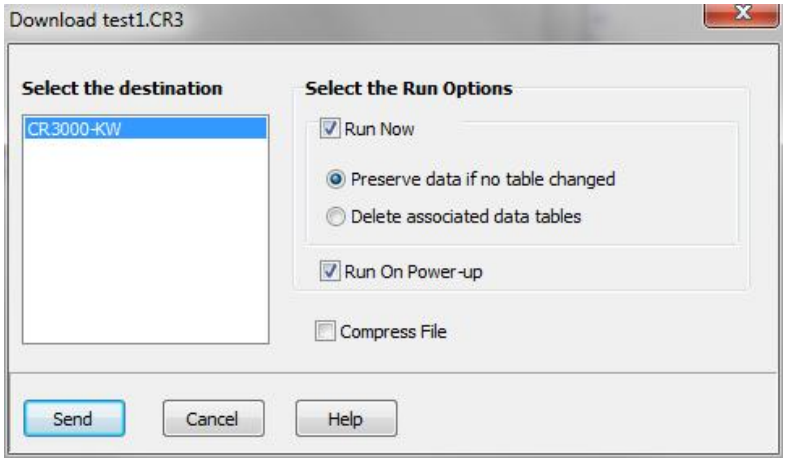
1. **Set up a CR3000 datalogger:**
   a. Connect RS-232 port from CR3000 to the laptop computer. Figure out which COM port is currently being used.
   b. Apply 12VDC power to CR3000.
   c. Open LoggerNet, you should see the following main menu:

d. Click on the “Setup” button to initialize connection and click “Add”. Select “CR3000”, and then choose “Direct Connect”. Choose your active COM port and choose 9600 as your baud rate.
e. Go back to the main menu, click on “Program” → “CRBasicEditor” to open the CRBasic Editor IDE.
f. The following program will appear in the editor. This program samples the panel temperature and the power supply level every second and stores the data in a data table called “Test” every 15 seconds.

```
'CR3000 Series Datalogger
'To create a different opening program template, type in new
'instructions and select Template | Save as Default Template
'date:
'program author:

'Declare Public Variables
'Example:
Public PTemp, batt_volt

'Declare Other Variables
'Example:
'Dim Counter

'Declare Constants
'Example:
'CONST PI = 3.141592654

'Define Data Tables
DataTable (Test,1,1000)
  DataInterval (0,15,Sec,10)
  Minimum (1,batt_volt,FP2,0,False)
  Sample (1,PTemp,FP2)
EndTable

'Define Subroutines
'Sub
  'EnterSub instructions here
'EndSub

'Main Program
BeginProg
  Scan (1,Sec,0,0)
    PanelTemp (PTemp,250)
    Battery (Batt_volt)
    'Enter other measurement instructions
    'Call Output Tables
    'Example:
    CallTable Test
  NextScan
EndProg
```

2. Download a program to your CR3000 datalogger:

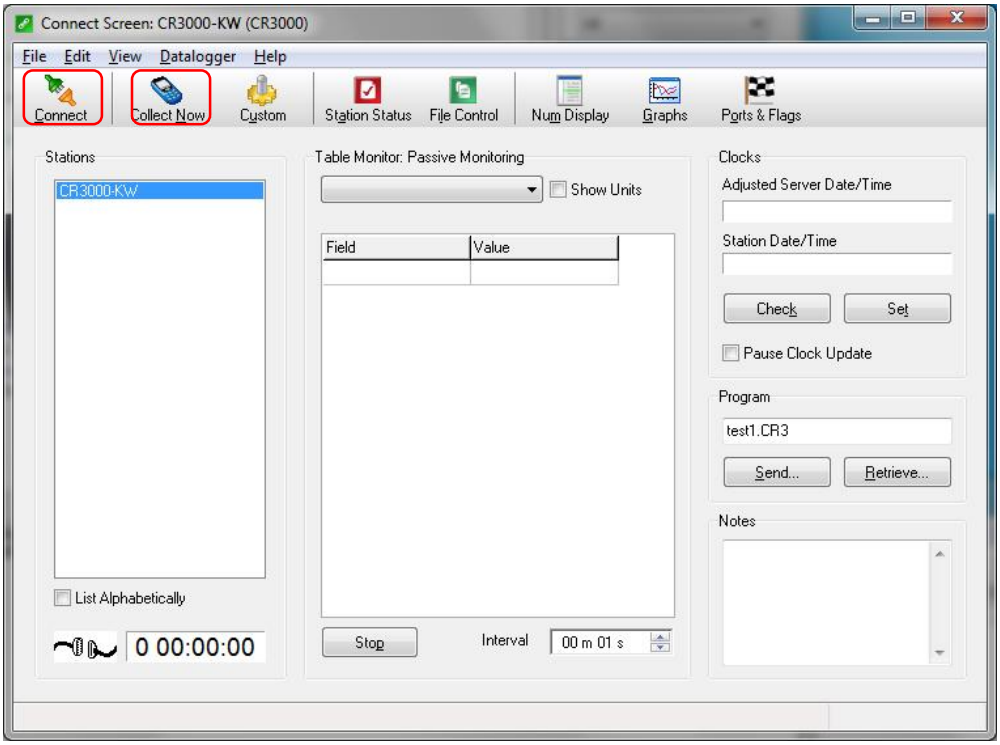
a. In the CRBasic Editor, open your program and click “Compile” check for errors and then “Compile, Save and Send”. Choose a file name then save your program on the laptop. You will then see the following dialog box:



b. Check “Run Now” to start your program immediately or Run On Power-up to run the program when you turn on the CR3000.
c. Click “Send” to start downloading the program.

3. Observe measurement data on a laptop:

a. In the LoggerNet main menu, click “Main” → “Connect”, you will see the following dialog box:

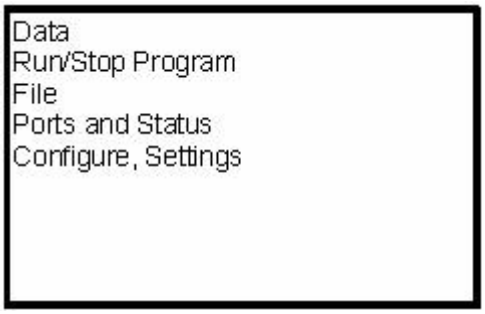


b. Click the “Connect” button, wait until it is connected, and then click “Collect Now” to upload the data. The data can be stored in a text file, which can then be opened by any text editor or Excel.
c. You can also monitor real time data in the “Num Display” or “Graphs” format, as shown in the “Connect Screen CR3000” dialog box.

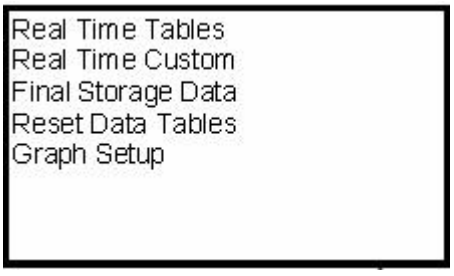
4. Run the datalogger in stand-alone mode:

The datalogger will operate without a laptop. , We review running the datalogger in a “stand-alone” mode in the step-by-step instructions below.

a. Start and stop the program:
    i. On the keypad, press “8” to enter the main menu display screen as shown below.



    ii. Note that you can always use “2” or “8” to navigate to your desired option (up or down arrows).
    iii. To run a program, select “Run/Stop Program”, then “Enter”.
    iv. To stop running a program, select “Stop, Retain Data”, press “Enter”. You will see a “*” showing up in front of your selection. You should then navigate to “Execute” and press “Enter”. You will see: “Program will be stopped, Continue? Yes, No?” on the display screen. Select “Yes”, then press “Enter”.
    v. To restart a program, select “Restart, Retain Data”.
    vi. You can always use “Esc” to cancel your operation and move to the next higher-level menu.
b. Observe the measurement data:
    i. Press “8” to enter the main menu display screen
    ii. Move the cursor to “Data” and press “Enter”. You should see the following menu.

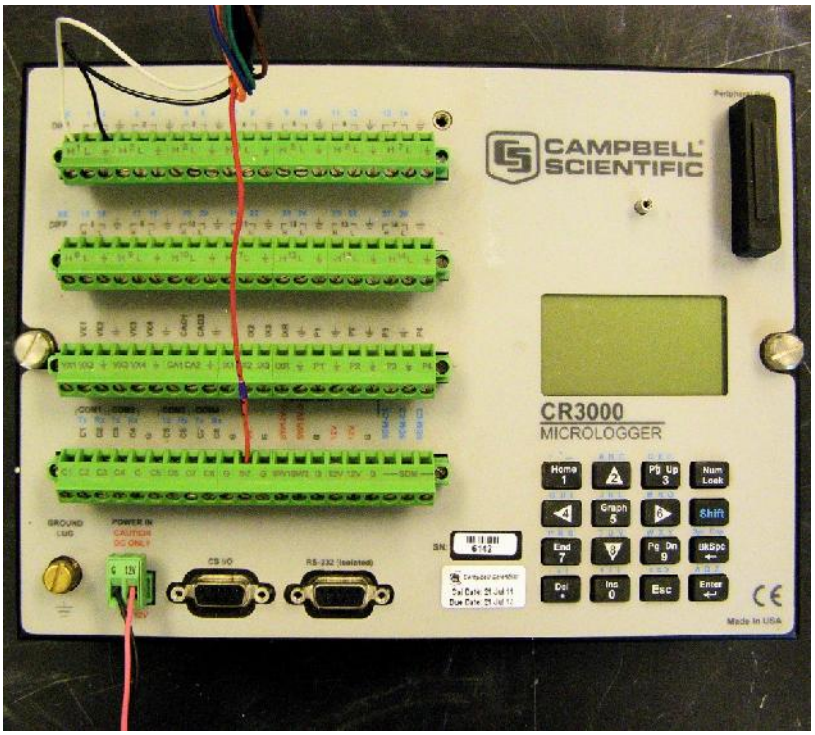


    iii. Select “Real Time Tables”, then press “Enter”.

iv. Select “Public” table and press “Enter”.
v. Now you can see two real-time variables and their values: PTemp and batt_volt, as they are measured.
vi. You can also select “Final Storage Data” to view stored data inside your customized table “Test”. Only stored data is retrievable for output and later analysis.

5. Measure distance using a MaxBotix ultrasonic ranging sensor:

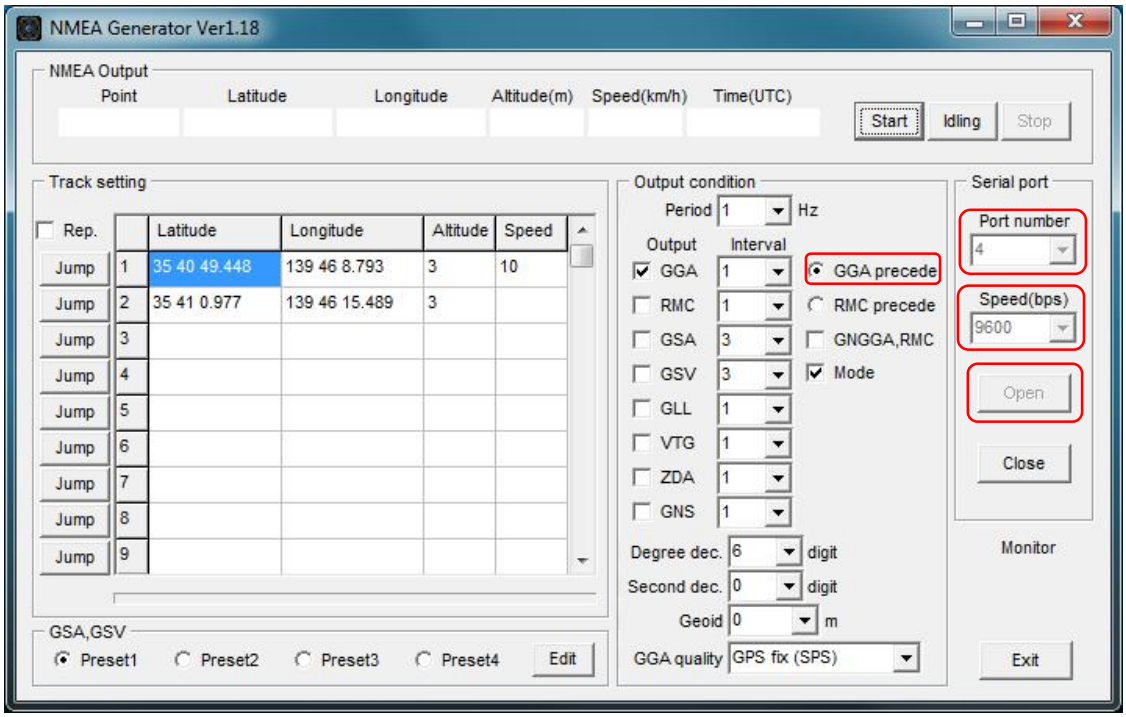
a. On your CR3000 datalogger, connect your MaxBotix ultrasonic ranging sensor to analog channel 1 as shown below:



b. Open the “UltraSonic_Training.CR3” program in the CRBasic Editor.
c. Compile, check for errors, save, and send the program to the CR3000 datalogger.
d. Using the “Connect” tools, observe the data in the “Num Display” or “Graphs” form.

6. Log GPS outputs through the RS232 port in CR3000:

a. Set up the GPS simulator
i. Open the “NMEA Generator”, and you should see the following window:



ii. Set the “Port number” to the output COM port in the laptop.
iii. Set the “Speed(bps)” (Baud rate) to 9600.
iv. Check the “GGA precede” only.

b. Open the “Logging_GPS_Training.CR3” program in the CRBasic Editor.
c. Compile, check for errors, save, and send the program to the CR3000 datalogger.
d. Go back to the “NMEA Generator”, click the “Open” button.
e. Observe the GPS outputs on the CR3000’s LCD screen.
f. Collect the data recorded in the CR3000, and see the data records in the laptop. Make sure you click the “Close” button in the “NMEA Generator”, or LoggerNet cannot connect to the CR3000.

7. An additional task for participants:

By completing procedures 1 thorugh 6, participants have acquired basic knowledge of how to use a CR3000 logger and some basic skills of CRBaisc programming. Next, participants can work out a simulated phenotyping program to log the canopy height trait (distance) with the spatial information (GPS coordinates).

Appendix A: NMEA message used in this lab exercise: the GGA message (GPS Fix Data)

GGA (GPS Fix Data)

The GGA message contains the time, position, and fix related data.

The GGA message structure is:

$GPGGA,151924,3723.454444,N,12202.269777,
W,2,09,1.9,–17.49,M,–25.67,M,1,0000*57

Table 4 describes these fields.

Table 4 GGA message fields

Field	Description
1	UTC of position fix in HHMMSS.SS format
2	Latitude in DD MM,MMMM format (0-7 decimal places)
3	Direction of latitude N: North S: South
4	Longitude in DDD MM,MMMM format (0-7 decimal places)
5	Direction of longitude E: East W: West
6	GPS Quality indicator 0: fix not valid 4: Real-time kinematic, fixed integers 1: GPS fix 5: Real-time kinematic, float integers 2: DGPS fix
7	Number of SVs in use, 00-12
8	HDOP
9	Antenna height, MSL reference
10	"M" indicates that the altitude is in meters
11	Geoidal separation
12	"M" indicates that the geoidal separation is in meters
13	Correction age of GPS data record, Type 1; Null when DGPS not used
14	Base station ID, 0000-1023

Appendix B: UltraSonic_Training.CR3

```
'Campbell Scientific CR3000 Series Datalogger instructional program
'Supporting FB-HTP training, from the USDA ARS and Kansas State University
'Conducted at MAC / ALARC in Maricopa, Arizona, USA, Spring 2014
'Program prepared by Matt Conley and Kevin Wang, 2/19/2014

'This simple program reads a MaxBotix Ultrasonic ranging sensor
'Comments follow the preceding apostrophe mark and explain program elements
'Comments are not executed by the logger

'Declare Public Variables, each variable used must be named
Public Panel_Temperature, Battery_Voltage
Public MB7364_Voltage
Public MB7364_cm, MB7364_inches


'Define the Data Table of variables to be collected and retained for analysis
'This data table is named "LearningOutput" and is set to maximum memory
DataTable (LearningOutput,1,-1)
  DataInterval (0,1,Sec,1) 'Data is collected each second (1Hz)
  Sample (1,Battery_Voltage,FP2) 'The 12v system voltage is useful for QC
  Sample (1,Panel_Temperature,FP2) 'The temperature of the logger hardware
  Sample (1,MB7364_Voltage,IEEE4) 'Store raw voltages at high precision
  Sample (1,MB7364_cm,FP2) 'The calculated distance in cm
  Sample (1,MB7364_inches, FP2) 'The calculated distance in inches
EndTable

'The actual program code starts here
BeginProg
  Scan (1,Sec,0,0) ' this is the program cycle time, set to 1Hz
    PanelTemp (Panel_Temperature,250) ' Read logger hardware temperature
    Battery (Battery_Voltage) ' Read instruction for logger system voltage

    'The Read instruction for the Ultrasonic Sensor on SE channel 1 at 5v
    VoltSe (MB7364_Volts,1,mV5000,1,False,0,250,1,0)
    MB7364_cm = MB7364_Volts / 10 'A conversion of voltage to distance in cm
    MB7364_inches = (MB7364_cm * 0.393700787) 'Conversion of cm to inches


    CallTable LearningOutput 'You must call the data table for it to write
  NextScan
EndProg
```

Appendix C: Logging_GPS_Training.CR3

```
'Campbell Scientific CR3000 Series Datalogger instructional program
'Supporting FB-HTP training, from Kansas State University and the USDA ARS
'Conducted at MAC / ALARC in Maricopa, Arizona, USA, Spring 2014
'Program prepared by Kevin Wang and Matt Conley, 3/26/2014
'This program logs the GPS outputs from a NMEA string generator
'Comments follow the preceding apostrophe mark and explain program elements
'Comments are not executed by the logger
'Declaration of the Public Variables
Public PTemp, Batt_Volt
Public GPShour, GPSminute, GPSsecond
Public GGAbytes, GGA_UTC_Az
'RealTime makes time variable outputs from the clock
Public rtime(9)' a nine varialbe time array is created
'The Alias command assigns a descriptive name to your variable
Alias rtime(1) = Year 'Logger Year
Alias rtime(2) = Month 'Logger Month
Alias rtime(3) = Day_of_Month 'Logger Day
Alias rtime(4) = Hour 'Logger Hour
Alias rtime(5) = Minute 'Logger Minute
Alias rtime(6) = Second 'Logger Second
Alias rtime(7) = Microseconds 'Logger Microsecond
Alias rtime(8) = Day_of_Week 'Looger week day
Alias rtime(9) = Day_of_Year 'Logger Julian DOY
'Constants are declared static numbers
Const Local_Time_Offset = 70000 ' Universial time offset for Arizona
'The Units command assigns units to your variable data output header line #3
'Otherwise this header line will come up blank
Units PTemp = Celsius 'Panel temperature of the logger
Units Batt_Volt = Volts 'Logger system voltage
Units Day_of_Year = Logger_DOY 'Derived from RealTime
Units GPShour = GGA_GPSTime 'Derived from the GGA GPS string
Units GPSminute = GGA_GPSTime 'Derived from the GGA GPS string
Units GPSsecond = GGA_GPSTime 'Derived from the GGA GPS string
'The NMEA $GPGGA GPS text string is arrayed and recorded
Public GGAString As String * 100 'Initial string has been striped down
Public GGA_Data(15) As String 'There are 16 variables in the string
Alias GGA_Data(2) = GGA_UTC       'UTC Current Time of GPS position fix - hhmmss.ss
Alias GGA_Data(3) = GGA_Lat       'Latitude degrees plus minutes
Alias GGA_Data(4) = GGA_NorS      'North or South Hemisphere indicator - N or S
Alias GGA_Data(5) = GGA_Long      'Longitude in degrees plus minutes
Alias GGA_Data(6) = GGA_EorW      'East or West indicator - E or W
Alias GGA_Data(7) = GGA_Quality   'Position Fix indicator - 0=invalid, 1=GPS only, 2=DGPS
Alias GGA_Data(8) = GGA_Sat       'Number of satellites used
Alias GGA_Data(9) = GGA_Horiz     'Horizontal dilution of precision (HDOP)
Alias GGA_Data(10) = GGA_Altitude 'Altitude relative to sea level (MSL)
Alias GGA_Data(11) = GGA_Meters   'Unit of altitude, Meters
Alias GGA_Data(12) = GGA_sep      'Geoidal separation
Alias GGA_Data(13) = GGA_Meters2  'Unit of separation, Meters
Alias GGA_Data(14) = GGA_age      'Age, in seconds, since last DGPS differential correction update
Alias GGA_Data(15) = GGA_DGPSref  'DGPS reference station ID#
'Descriptive units are added to GPS variables and will show in header line #3
Units GGAString = NMEA_$GPGGA
Units GGA_UTC = hhmmss.ss_hour_minute_second
Units GGA_UTC_Az = hhmmss.ss
Units GGA_Lat = ddmm.mm_degree_minute
Units GGA_NorS = hemisphere
Units GGA_Long = degrees_minutes
Units GGA_EorW = hemisphere
Units GGA_Quality = 1_GPS 2_DGPS
Units GGA_Sat = satellites
Units GGA_Horiz = HDCP
Units GGA_Altitude = meters
Units GGA_Meters = meters
Units GGA_sep = geoidal_separation
Units GGA_Meters2 = meters
Units GGA_age = seconds
Units GGA_DGPSref = station_ID
```

```
DataTable (GPS_and_Ptemp,1,-1)
  Sample (1,GPShour,UINT2) 'GGA derived hour of day
  Sample (1,GPSminute,UINT2) 'GGA derived minute of hour
  Sample (1,GPSsecond,UINT2) 'GGA derived second of minute
  Sample (1,Day_of_Year,UINT2) 'Realtime derived day of year
  Sample (1,PTemp,FP2)'Logger internal temperature
  Sample (1,Batt_Volt,FP2) 'Logger system voltage
  Sample (1,GGA_Lat,String) 'GGA derived positional latitude
  Sample (1,GGA_NorS,String) 'GGA derived North or South
  Sample (1,GGA_Long,String) 'GGA derived positional longitude
  Sample (1,GGA_EorW,String) 'GGA derived East or West
  Sample (1,GGA_Quality,UINT2) 'GGA derived GPS signal quality
  Sample (1,GGA_Sat,UINT2) 'GGA derived number of sattilites in view
EndTable

'Main Program Starts here
BeginProg
SerialOpen(ComRS232,9600,0,0,1000) 'Baud rate for the GPS simulator, Com?
Scan (1,Sec,5,0) '1Hz cycle rate

  RealTime(rtime) 'The RealTime time command
  PanelTemp (PTemp,250) 'Logger temperature command
  Battery (Batt_Volt) 'Logger system voltage command

  'The NMEA GGA serial string is read in based on start and stop characters
  '36 is the ASCII character for $
  '&H0D0A is the hexadecimal representation for carriage return/line feed
  SerialInRecord(ComRS232,GGAString,36,0,&H0D0A,GGAbytes,01)
  'The RMC string is split into 13 variables based on commas
  SplitStr(GGA_Data(),GGAString,",",15,5)

  'Convert universal time to local time
  GGA_UTC_Az = GGA_UTC
  If GGA_UTC_Az >= 70000
    GGA_UTC_Az = GGA_UTC_Az - Local_Time_Offset
  Else
    GGA_UTC_Az = GGA_UTC + 24 - Local_Time_Offset
  EndIf

  'Shift local time to account for an increase in number of digits (9 to 10)
  If GGA_UTC_Az >= 100000
    GPShour = Mid(GGA_UTC_Az,1,2)
    GPSminute = Mid(GGA_UTC_Az,3,2)
    GPSsecond = Mid(GGA_UTC_Az,5,2)
  EndIf
  If GGA_UTC_Az < 100000
    GPShour = Mid(GGA_UTC_Az,1,1)
    GPSminute = Mid(GGA_UTC_Az,2,2)
    GPSsecond = Mid(GGA_UTC_Az,4,2)
  EndIf

  'Call the data tables to write data
  CallTable GPS_and_Ptemp

NextScan
EndProg
```