Programming a CR1000 Datalogger

CRBasic is the programming language which controls the functionality of a Campbell Scientific Datalogger. The program controls all processes in the logger related to the collection of data, including variable names, data table layouts, transducer definitions and ranging, and control port operation.

In order to begin a program in CRBasic, it is recommended to first create a program in short cut, available through loggernet, which contains a wizard and allows for the creation of the basic structure of a program including table and sensor declarations. Once the program has been created in short cut, it can be edited in CRBasic to add in any advanced functionality required. This guide covers the basic functionality of CRBasic as well as some specifics to the programming in the test site at St.Lawrence College. More detailed documentation for coding is available in the CR1000 manual found on the Campbell Scientific website.

In any CRBasic code, there are three major sections: i) variable declaration ii) table definition iii)collection

Contents

1 Variable Declaration
2 Table declaration
3 Data collection
4 Most recent code from the SEARC OTF

Variable Declaration

The first portion of the code defines all the variables which will be collected by the datalogger, using the Public command. Singular variables are defined by inputting the variable name and array variables, which can be used to store multiple values under indexed variable names, are defined by adding the length of the array in brackets after the variable name. For example:

```
Public Temp_C_2(25)
```

Declares a variable Temp_C_2 which has 25 indexable values. This is useful when using a multiplexer, as each index can represent a multiplexer port. These indexed variables can become an issue, however, as it is difficult to distinguish which physical sensor they are attached to. Because of this it is useful to use the Alias command to re-name each value inside the array to represent the sensor it represents. For example:

```
Alias Temp_C_2(5)=T1.5
```
Renames the fifth value in Temp_C_2 to show that this is measuring the thermocouple in the fifth position of the first row of the array.

**Table declaration**

Once the variables have defined, there is the option to divide them into multiple data tables for easier data manipulation. In this case, all the variables are output in a single table, called Table1. In addition, it is possible to time average data inside the tables. This would allow for collection of the data, for example, at 1 minute intervals but only storing of the 5 minute averages of this data in the table. The advantage of this is a high resolution of collection, while reducing the size of the collected datafile, making it more manageable for storage, transmission and post-processing.

In the case of this program data is sampled and collected at 5 minute intervals. The reason for this is the inclusion of multiplexers in this system, which have limited cycle lives. Therefore, the choice was made to reduce the number of cycles experienced by the multiplexer by only sampling at 5 minute intervals, which gives a good compromise of cycle times and data resolution.

The data table is declared using the following two lines:

```plaintext
DataTable(Table1, True, -1)
```

```plaintext
DataInterval(0, 5, Min, 10)
```

Which defines the memory usage and recording interval for the data table. An important attribute of tables in a Campbell Scientific system is the use of loop memory. The base memory on the unit is relatively limited at 4MB to store the program and all data. In the case that the collected data exceeds the available memory space on the system, the data will loop to the top of the data table, erasing the oldest data first. The memory status of the station can be seen through the device configuration utility, or the connect screen.

**Data collection**

Collection of data occurs in a repeating loop which runs until the program is stopped manually by the controller through loggernet. The beginning of this loop is signified by the following two lines of code:

```plaintext
BeginProg
```

```plaintext
Scan(5, min, 1, 0)
```

Which initiates the loop, and sets a scan to occur every 5 minutes, according to the datalogger clock. Every command following this is executed sequentially when the scan is initiated.
Each measurement is initiated by its own line in the program, which declares the type of measurement to be made, the input channel it is being read from and any special post processing required for the measurement. For example:

\[
\text{VoltDiff(SlrW,1,mV25,1,True,0,60Hz,1,0)}
\]

Reads the differential voltage measurement from channel 1 and stores it in the variable SlrW, with a voltage range of 25 mV. The remainder of the arguments in this declaration relate to specifics of how the value is collected, and a full description of these can be found in the CRBasic help. Also, right clicking on a measurement declaration will bring up a window which describes each input parameter, and details standards for each value.

To measure a multiplexer, more complex code is required. The basic steps in measuring a multiplexer are to:

i) Initialize the unit by setting the res port to high

\[
\text{PortSet(2,1)}
\]

\[
\text{Delay(0,150,mSec)}
\]

ii) Enter into a scan loop and send a pulse to the CLK port which sets the relay to the first input

\[
\text{SubScan(0,uSec,32)}
\]

\[
\text{PulsePort(1,10000)}
\]

\[
\text{Delay(0,100,uSec)}
\]

iii) Read the measured value into the correct slot of the array. Note that because two single ended measurements are being taken per differential channel, each declaration of VoltSe records two sequential values into two sequential slots in the measurement array. This is done by setting the 'repetitions' field (second field in this declaration) to 2, which takes a measurement from the declared channel (in this case 13) and puts it into the first slot of the array, and then takes a measurement from the next channel, in this case 14, and puts it into the next slot of the array.

\[
\text{VoltSe(SEvolt(LCount),2,mV5000,13,True,0,60Hz,1,0)}
\]

iv) Return to the beginning and continue to repeat the scan until all ports on the multiplexer have been read

\[
\text{NextSubScan}
\]

v) Set the RES port to low to deactivate the datalogger
At the end of the program, the tables must be called which stores all the variables which have been collected permanently in the pre-defined tables.

**Most recent code from the SEARC OTF**

The following is the most recent code being used at the SEARC OTF. Note than any line prefixed with ' represents a code comment.

'Program running in St.Lawrence CR1000 datalogger since January, 2011

'Created by Short Cut (2.8)

Declare Variables

Public BattV

Public SlrW

Public SlrkJ_2

Public SlrW_2

Public SlrkJ

Public SlrW_3

Public SlrkJ_3

Public SR50A(2)

Public TCDT

Public DBTCDT

Public RTempC

Public Temp_C(25)

Public RTempC_2

Public Temp_C_2(25)

Public AirTC

Public RH

Public WS_ms
Public WindDir
Public SEvolt(64)
Public SEvolt_2(64)
Public Sensor_Height
Public Snow_Depth
Public LCount

'Alias variables to make data easier to identify. These are the names which will be recorded in the data table.

Alias SR50A(1)=DT
Alias SR50A(2)=Q
Alias SEvolt(1)=V1.1V
Alias SEvolt(2)=V1.2V
Alias SEvolt(3)=V1.3V
Alias SEvolt(4)=V1.4V
Alias SEvolt(5)=V1.5V
Alias SEvolt(6)=V1.6V
Alias SEvolt(7)=V1.7V
Alias SEvolt(8)=V1.8V
Alias SEvolt(9)=V2.1V
Alias SEvolt(10)=V2.2V
Alias SEvolt(11)=V2.3V
Alias SEvolt(12)=V2.4V
Alias SEvolt(13)=V2.5V
Alias SEvolt(14)=V2.6V
Alias SEvolt(15)=V2.7V
Alias SEvolt(16)=V2.8V
Alias SEvolt(17)=V3.1V
Alias SEvolt(18)=V3.2V
Alias SEvolt(19)=V3.3V
<table>
<thead>
<tr>
<th>Alias SEvolt</th>
<th>Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>3.4V</td>
</tr>
<tr>
<td>21</td>
<td>3.5V</td>
</tr>
<tr>
<td>22</td>
<td>3.6V</td>
</tr>
<tr>
<td>23</td>
<td>3.7V</td>
</tr>
<tr>
<td>24</td>
<td>3.8V</td>
</tr>
<tr>
<td>25</td>
<td>4.1V</td>
</tr>
<tr>
<td>26</td>
<td>4.2V</td>
</tr>
<tr>
<td>27</td>
<td>4.3V</td>
</tr>
<tr>
<td>28</td>
<td>4.4V</td>
</tr>
<tr>
<td>29</td>
<td>4.5V</td>
</tr>
<tr>
<td>30</td>
<td>4.6V</td>
</tr>
<tr>
<td>31</td>
<td>4.7V</td>
</tr>
<tr>
<td>32</td>
<td>4.8V</td>
</tr>
<tr>
<td>33</td>
<td>C1.1C</td>
</tr>
<tr>
<td>34</td>
<td>C1.2C</td>
</tr>
<tr>
<td>35</td>
<td>C1.3C</td>
</tr>
<tr>
<td>36</td>
<td>C1.4C</td>
</tr>
<tr>
<td>37</td>
<td>C1.5C</td>
</tr>
<tr>
<td>38</td>
<td>C1.6C</td>
</tr>
<tr>
<td>39</td>
<td>C1.7C</td>
</tr>
<tr>
<td>40</td>
<td>C1.8C</td>
</tr>
<tr>
<td>41</td>
<td>C2.1C</td>
</tr>
<tr>
<td>42</td>
<td>C2.2C</td>
</tr>
<tr>
<td>43</td>
<td>C2.3C</td>
</tr>
<tr>
<td>44</td>
<td>C2.4C</td>
</tr>
<tr>
<td>45</td>
<td>C2.5C</td>
</tr>
<tr>
<td>46</td>
<td>C2.6C</td>
</tr>
<tr>
<td>47</td>
<td>C2.7C</td>
</tr>
</tbody>
</table>
Alias SEvolt(48)=2.8C
Alias SEvolt(49)=3.1C
Alias SEvolt(50)=3.2C
Alias SEvolt(51)=3.3C
Alias SEvolt(52)=3.4C
Alias SEvolt(53)=3.5C
Alias SEvolt(54)=3.6C
Alias SEvolt(55)=3.7C
Alias SEvolt(56)=3.8C
Alias SEvolt(57)=4.1C
Alias SEvolt(58)=4.2C
Alias SEvolt(59)=4.3C
Alias SEvolt(60)=4.4C
Alias SEvolt(61)=4.5C
Alias SEvolt(62)=4.6C
Alias SEvolt(63)=4.7C
Alias SEvolt(64)=4.8C
Alias SEvolt_2(1)=V1.9V
Alias SEvolt_2(2)=V1.10V
Alias SEvolt_2(3)=V1.11V
Alias SEvolt_2(4)=V1.12V
Alias SEvolt_2(5)=V1.13V
Alias SEvolt_2(6)=V1.14V
Alias SEvolt_2(7)=V1.15V
Alias SEvolt_2(8)=V1.16V
Alias SEvolt_2(9)=V2.9V
Alias SEvolt_2(10)=V2.10V
Alias SEvolt_2(11)=V2.11V
Alias SEvolt_2(12)=V2.12V
Alias SEvolt_2(13)=V2.13V
Alias SEvolt_2(14)=V2.14V
Alias SEvolt_2(15)=V2.15V
Alias SEvolt_2(16)=V2.16V
Alias SEvolt_2(17)=V3.9V
Alias SEvolt_2(18)=V3.10V
Alias SEvolt_2(19)=V3.11V
Alias SEvolt_2(20)=V3.12V
Alias SEvolt_2(21)=V3.13V
Alias SEvolt_2(22)=V3.14V
Alias SEvolt_2(23)=V3.15V
Alias SEvolt_2(24)=V3.16V
Alias SEvolt_2(25)=V4.9V
Alias SEvolt_2(26)=V4.10V
Alias SEvolt_2(27)=V4.11V
Alias SEvolt_2(28)=V4.12V
Alias SEvolt_2(29)=V4.13V
Alias SEvolt_2(30)=V4.14V
Alias SEvolt_2(31)=V4.15V
Alias SEvolt_2(32)=V4.16V
Alias SEvolt_2(33)=C1.9C
Alias SEvolt_2(34)=C1.10C
Alias SEvolt_2(35)=C1.11C
Alias SEvolt_2(36)=C1.12C
Alias SEvolt_2(37)=C1.13C
Alias SEvolt_2(38)=C1.14C
Alias SEvolt_2(39)=C1.15C
Alias SEvolt_2(40)=C4.13C
Alias SEvolt_2(41)=C2.9C
Alias SEvolt_2(42)=C2.10C
Alias SEvolt_2(43)=C2.11C
Alias SEvolt_2(44)=C2.12C
Alias SEvolt_2(45)=C2.13C
Alias SEvolt_2(46)=C2.14C
Alias SEvolt_2(47)=C2.15C
Alias SEvolt_2(48)=C4.14C
Alias SEvolt_2(49)=C3.9C
Alias SEvolt_2(50)=C3.10C
Alias SEvolt_2(51)=C3.11C
Alias SEvolt_2(52)=C3.12C
Alias SEvolt_2(53)=C3.13C
Alias SEvolt_2(54)=C3.14C
Alias SEvolt_2(55)=C3.15C
Alias SEvolt_2(56)=C4.15C
Alias SEvolt_2(57)=C4.9C
Alias SEvolt_2(58)=C4.10C
Alias SEvolt_2(59)=C4.11C
Alias SEvolt_2(60)=C4.12C
Alias Temp_C(1)=T3.4
Alias Temp_C(2)=T3.5
Alias Temp_C(3)=T3.6
Alias Temp_C(4)=T3.7
Alias Temp_C(5)=T4.1
Alias Temp_C(6)=T4.2
Alias Temp_C(7)=T4.3
Alias Temp_C(8)=T4.4
Alias Temp_C(9)=T4.5
Alias Temp_C(10)=T4.6
Alias Temp_C(11)=T4.7
Alias Temp_C(12)=T2.10
Alias Temp_C(13)=T2.11
Alias Temp_C(14)=T2.12
Alias Temp_C(15)=T2.13
Alias Temp_C(16)=T2.15
Alias Temp_C(17)=T3.11
Alias Temp_C(18)=T3.12
Alias Temp_C(19)=T3.13
Alias Temp_C(20)=T3.15
Alias Temp_C(21)=T4.15
Alias Temp_C(22)=Empty1
Alias Temp_C(23)=Empty2
Alias Temp_C(24)=Empty3
Alias Temp_C(25)=Empty4
Alias Temp_C_2(1)=T1.1
Alias Temp_C_2(2)=T1.2
Alias Temp_C_2(3)=T1.3
Alias Temp_C_2(4)=T1.4
Alias Temp_C_2(5)=T1.5
Alias Temp_C_2(6)=T1.6
Alias Temp_C_2(7)=T1.7
Alias Temp_C_2(8)=T1.8
Alias Temp_C_2(9)=T1.9
Alias Temp_C_2(10)=T1.10
<table>
<thead>
<tr>
<th>Alias</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temp_C_2(11)</td>
<td>T1.11</td>
</tr>
<tr>
<td>Temp_C_2(12)</td>
<td>T1.12</td>
</tr>
<tr>
<td>Temp_C_2(13)</td>
<td>T1.13</td>
</tr>
<tr>
<td>Temp_C_2(14)</td>
<td>T1.14</td>
</tr>
<tr>
<td>Temp_C_2(15)</td>
<td>T1.15</td>
</tr>
<tr>
<td>Temp_C_2(16)</td>
<td>T2.1</td>
</tr>
<tr>
<td>Temp_C_2(17)</td>
<td>T2.2</td>
</tr>
<tr>
<td>Temp_C_2(18)</td>
<td>T2.3</td>
</tr>
<tr>
<td>Temp_C_2(19)</td>
<td>T2.4</td>
</tr>
<tr>
<td>Temp_C_2(20)</td>
<td>T2.5</td>
</tr>
<tr>
<td>Temp_C_2(21)</td>
<td>T2.6</td>
</tr>
<tr>
<td>Temp_C_2(22)</td>
<td>T2.7</td>
</tr>
<tr>
<td>Temp_C_2(23)</td>
<td>T3.1</td>
</tr>
<tr>
<td>Temp_C_2(24)</td>
<td>T3.2</td>
</tr>
<tr>
<td>Temp_C_2(25)</td>
<td>T3.3</td>
</tr>
</tbody>
</table>

Declare the units for each measurement

Units BattV=Volts
Units SlrW=W/m²
Units SlrkJ_2=kJ/m²
Units SlrW_2=W/m²
Units SlrkJ=kJ/m²
Units SlrW_3=W/m²
Units SlrkJ_3=kJ/m²
Units RTempC=Deg C
Units Temp_C=Deg C
Units RTempC_2=Deg C
Units Temp_C_2=Deg C
Units AirTC=Deg C
Units RH=%

Units WS_ms=meters/second

Units WindDir=Degrees

Define Data Tables, Table 1 is the only table used in this program.

The table is averaged over 5 minute intervals, and the collection of data occurs at the same interval, therefore the average values are actually the point measurement taken at that point in time

DataTable(Table1,True,-1)

DataInterval(0,5,Min,10)

Average(1,SlrW,FP2,False)

Totalize(1,SlrkJ_2,IEEE4,False)

Average(1,SlrW_2,FP2,False)

Totalize(1,SlrkJ,IEEE4,False)

Average(1,SlrW_3,FP2,False)

Totalize(1,SlrkJ_3,IEEE4,False)

Average(1,DT,FP2,False)

Average(1,Q,FP2,False)

Average(1,TCDT,FP2,False)

Average(1,DBTCDT,FP2,False)

Average(1,RTempC_2,FP2,False)

Average(1,Snow_Depth,FP2,False)

Average(1,Temp_C_2(1),FP2,False)

Average(1,Temp_C_2(2),FP2,False)

Average(1,Temp_C_2(3),FP2,False)

Average(1,Temp_C_2(4),FP2,False)

Average(1,Temp_C_2(5),FP2,False)

Average(1,Temp_C_2(6),FP2,False)

Average(1,Temp_C_2(7),FP2,False)

Average(1,Temp_C_2(8),FP2,False)
Average(1, Temp_C_2(9), FP2, False)
Average(1, Temp_C_2(10), FP2, False)
Average(1, Temp_C_2(11), FP2, False)
Average(1, Temp_C_2(12), FP2, False)
Average(1, Temp_C_2(13), FP2, False)
Average(1, Temp_C_2(14), FP2, False)
Average(1, Temp_C_2(15), FP2, False)
Average(1, Temp_C_2(16), FP2, False)
Average(1, Temp_C_2(17), FP2, False)
Average(1, Temp_C_2(18), FP2, False)
Average(1, Temp_C_2(19), FP2, False)
Average(1, Temp_C_2(20), FP2, False)
Average(1, Temp_C_2(21), FP2, False)
Average(1, Temp_C_2(22), FP2, False)
Average(1, Temp_C_2(23), FP2, False)
Average(1, Temp_C_2(24), FP2, False)
Average(1, Temp_C_2(25), FP2, False)
Average(1, RTempC, FP2, False)
Average(1, Temp_C(1), FP2, False)
Average(1, Temp_C(2), FP2, False)
Average(1, Temp_C(3), FP2, False)
Average(1, Temp_C(4), FP2, False)
Average(1, Temp_C(5), FP2, False)
Average(1, Temp_C(6), FP2, False)
Average(1, Temp_C(7), FP2, False)
Average(1, Temp_C(8), FP2, False)
Average(1, Temp_C(9), FP2, False)
Average(1, Temp_C(10), FP2, False)
Average(1,Temp_C(1),FP2,False)
Average(1,Temp_C(11),FP2,False)
Average(1,Temp_C(12),FP2,False)
Average(1,Temp_C(13),FP2,False)
Average(1,Temp_C(14),FP2,False)
Average(1,Temp_C(15),FP2,False)
Average(1,Temp_C(16),FP2,False)
Average(1,Temp_C(17),FP2,False)
Average(1,Temp_C(18),FP2,False)
Average(1,Temp_C(19),FP2,False)
Average(1,Temp_C(20),FP2,False)
Average(1,Temp_C(21),FP2,False)
Average(1,Temp_C(22),FP2,False)
Average(1,Temp_C(23),FP2,False)
Average(1,Temp_C(24),FP2,False)
Average(1,Temp_C(25),FP2,False)
Average(1,SEvolt(1),FP2,False)
Average(1,SEvolt(2),FP2,False)
Average(1,SEvolt(3),FP2,False)
Average(1,SEvolt(4),FP2,False)
Average(1,SEvolt(5),FP2,False)
Average(1,SEvolt(6),FP2,False)
Average(1,SEvolt(7),FP2,False)
Average(1,SEvolt(8),FP2,False)
Average(1,SEvolt(9),FP2,False)
Average(1,SEvolt(10),FP2,False)
Average(1,SEvolt(11),FP2,False)
Average(1,SEvolt(12),FP2,False)
Average(1,SEvolt(13),FP2,False)
Average(1,SEvolt(14),FP2,False)
Average(1,SEvolt(15),FP2,False)
Average(1,SEvolt(16),FP2,False)
Average(1,SEvolt(17),FP2,False)
Average(1,SEvolt(18),FP2,False)
Average(1,SEvolt(19),FP2,False)
Average(1,SEvolt(20),FP2,False)
Average(1,SEvolt(21),FP2,False)
Average(1,SEvolt(22),FP2,False)
Average(1,SEvolt(23),FP2,False)
Average(1,SEvolt(24),FP2,False)
Average(1,SEvolt(25),FP2,False)
Average(1,SEvolt(26),FP2,False)
Average(1,SEvolt(27),FP2,False)
Average(1,SEvolt(28),FP2,False)
Average(1,SEvolt(29),FP2,False)
Average(1,SEvolt(30),FP2,False)
Average(1,SEvolt(31),FP2,False)
Average(1,SEvolt(32),FP2,False)
Average(1,SEvolt(33),FP2,False)
Average(1,SEvolt(34),FP2,False)
Average(1,SEvolt(35),FP2,False)
Average(1,SEvolt(36),FP2,False)
Average(1,SEvolt(37),FP2,False)
Average(1,SEvolt(38),FP2,False)
Average(1,SEvolt(39),FP2,False)
Average(1,SEvolt(40),FP2,False)
Average(1,SEvolt(41),FP2,False)
Average(1,SEvolt(42),FP2,False)
Average(1,SEvolt(43),FP2,False)
Average(1,SEvolt(44),FP2,False)
Average(1,SEvolt(45),FP2,False)
Average(1,SEvolt(46),FP2,False)
Average(1,SEvolt(47),FP2,False)
Average(1,SEvolt(48),FP2,False)
Average(1,SEvolt(49),FP2,False)
Average(1,SEvolt(50),FP2,False)
Average(1,SEvolt(51),FP2,False)
Average(1,SEvolt(52),FP2,False)
Average(1,SEvolt(53),FP2,False)
Average(1,SEvolt(54),FP2,False)
Average(1,SEvolt(55),FP2,False)
Average(1,SEvolt(56),FP2,False)
Average(1,SEvolt(57),FP2,False)
Average(1,SEvolt(58),FP2,False)
Average(1,SEvolt(59),FP2,False)
Average(1,SEvolt(60),FP2,False)
Average(1,SEvolt(61),FP2,False)
Average(1,SEvolt(62),FP2,False)
Average(1,SEvolt(63),FP2,False)
Average(1,SEvolt(64),FP2,False)
Average(1,SEvolt_2(1),FP2,False)
Average(1,SEvolt_2(2),FP2,False)
Average(1,SEvolt_2(3),FP2,False)
Average(1,SEvolt_2(4),FP2,False)
Average(1,SEvolt_2(5),FP2,False)
Average(1,SEvolt_2(6),FP2,False)
Average(1,SEvolt_2(7),FP2,False)
Average(1,SEvolt_2(8),FP2,False)
Average(1,SEvolt_2(9),FP2,False)
Average(1,SEvolt_2(10),FP2,False)
Average(1,SEvolt_2(11),FP2,False)
Average(1,SEvolt_2(12),FP2,False)
Average(1,SEvolt_2(13),FP2,False)
Average(1,SEvolt_2(14),FP2,False)
Average(1,SEvolt_2(15),FP2,False)
Average(1,SEvolt_2(16),FP2,False)
Average(1,SEvolt_2(17),FP2,False)
Average(1,SEvolt_2(18),FP2,False)
Average(1,SEvolt_2(19),FP2,False)
Average(1,SEvolt_2(20),FP2,False)
Average(1,SEvolt_2(21),FP2,False)
Average(1,SEvolt_2(22),FP2,False)
Average(1,SEvolt_2(23),FP2,False)
Average(1,SEvolt_2(24),FP2,False)
Average(1,SEvolt_2(25),FP2,False)
Average(1,SEvolt_2(26),FP2,False)
Average(1,SEvolt_2(27),FP2,False)
Average(1,SEvolt_2(28),FP2,False)
Average(1,SEvolt_2(29),FP2,False)
Average(1,SEvolt_2(30),FP2,False)
Average(1,SEvolt_2(31),FP2,False)
Average(1,SEvolt_2(32),FP2,False)
Average(1,SEvolt_2(33),FP2,False)
Average(1,SEvolt_2(34),FP2,False)
Average(1,SEvolt_2(35),FP2,False)
Average(1,SEvolt_2(36),FP2,False)
Average(1,SEvolt_2(37),FP2,False)
Average(1,SEvolt_2(38),FP2,False)
Average(1,SEvolt_2(39),FP2,False)
Average(1,SEvolt_2(40),FP2,False)
Average(1,SEvolt_2(41),FP2,False)
Average(1,SEvolt_2(42),FP2,False)
Average(1,SEvolt_2(43),FP2,False)
Average(1,SEvolt_2(44),FP2,False)
Average(1,SEvolt_2(45),FP2,False)
Average(1,SEvolt_2(46),FP2,False)
Average(1,SEvolt_2(47),FP2,False)
Average(1,SEvolt_2(48),FP2,False)
Average(1,SEvolt_2(49),FP2,False)
Average(1,SEvolt_2(50),FP2,False)
Average(1,SEvolt_2(51),FP2,False)
Average(1,SEvolt_2(52),FP2,False)
Average(1,SEvolt_2(53),FP2,False)
Average(1,SEvolt_2(54),FP2,False)
Average(1,SEvolt_2(55),FP2,False)
Average(1,SEvolt_2(56),FP2,False)
Average(1,SEvolt_2(57),FP2,False)
Average(1,SEvolt_2(58),FP2,False)
Average(1,SEvolt_2(59),FP2,False)
Average(1,SEvolt_2(60),FP2,False)
Average(1,SEvolt_2(61),FP2,False)
Average(1,SEvolt_2(62),FP2,False)
Average(1,SEvolt_2(63),FP2,False)
Average(1,SEvolt_2(64),FP2,False)
Average(1,AirTC,FP2,False)
Sample(1,RH,FP2)
Average(1,WS_ms,FP2,False)
Sample(1,WindDir,FP2)
EndTable
DataTable(Table2,True,-1)
DataInterval(0,1440,Min,10)
Minimum(1,BattV,FP2,False,False)
EndTable
'Main Program, this is where all measurements are made
BeginProg
'Set a scan to occur every 5 minutes
Scan(5,min,1,0)
Default Datalogger Battery Voltage measurement BattV
Battery(BattV)
'CMP22 direct Pyranometer measurements SlrkJ and SlrW
VoltDiff(SlrW,1,mV25,1,True,0,_60Hz,1,0)
If SlrW<0 Then SlrW=0
SlrkJ=SlrW*33.25942
SlrW=SlrW*105.9322
'CMP11 Pyranometer measurements SlrkJ_2 and SlrW_2
This sensor has been temporarily replaced with a test 'pyranometer, with a different callibration factor
VoltDiff(SlrW_2,1,mV25,2,True,0,_60Hz,1,0)
If SlrW_2<0 Then SlrW_2=0
SlrkJ_2=SlrW_2*33.25942
SrW_2=SlrW_2*105.5966  *****CMP 11 factor****
SrW_2=SlrW_2*142.857

'CMP22 diffuse Pyranometer measurements SlrJ_3 and SlrW_3
VoltDiff(SlrW_3,1,mV25,3,True,0,-60Hz,1,0)
If SlrW_3<0 Then SlrW_3=0
SlrJ_3=SlrW_3*33.25942
SlrW_3=SlrW_3*110.8647

'SR50A Sonic Ranging Sensor (SDI-12 Output) measurements
'Sensor height was set to an arbitrary measurement for the beginning
Sensor_Height=2
SD12Recorder(SR50A(),7,"0","M1!",1,0)
TCDT=DT*SQR((AirTC+273.15)/273.15)
Snow_Depth=Sensor_Height-TCDT

'Type T (copper-constantan) Thermocouple measurements 'Temp_C_2() on the AM25T Multiplexer
AM25T('Temp_C_2(),25,mV2_5C,1,4,TypeT,RTempC_2,5,3,3,True,0,-60Hz,1,0)

'Type T (copper-constantan) Thermocouple measurements 'Temp_C() on the AM25T Multiplexer
AM25T('Temp_C(),25,mV2_5C,1,5,TypeT,RTempC,5,4,3,True,0,-60Hz,1,0)

'05103 Wind Speed & Direction Sensor measurements
' Measures Wind Speed
PulseCount(WS_ms,1,1,1,1,0.098,0)

'Measures wind direction, this measurement uses excitation 'port 1
BrHalf(WindDir,1,mV2500,11,1,1,2500,True,0,-60Hz,355,0)
If WindDir>=360 Then WindDir=0

'Turn both AM16/32 Multiplexers On
PortSet(2,1)
Delay(0,150,mSec)

'Initialize data counter
Lcount=1
Start scan of first multiplexer
SubScan(0,uSec,32)

' Switch to next AM16/32 Multiplexer channel
PulsePort(1,10000)
Delay (0,100,uSec)

'Generic Single Ended Voltage measurements SEVolt() on 'the AM16/32 Multiplexer: modify this code to read 'current as 0-2.5 and voltage as 0-5V
VoltSe(SEvolt(LCount),2,mV5000,13,True,0,_60Hz,1,0)
LCount=LCount+2
NextSubScan

'Scan western AYA multiplexer
LCount=1
SubScan(0,uSec,32)

'Switch to next AM16/32 Multiplexer channel
PulsePort(1,10000)
Delay (0,100,uSec)

'the last two differential channels of this multiplexer 'measure temperature and RH as well, and these IF' statements ensure that these is measured properly.
If Lcount=61
'Measure Air temperature
VoltDiff (AirTC,1,mV2500,8,True ,0,_60Hz,0.1,-40)
ElseIf LCount=63
'Generic Differential Voltage measurements RH:
VoltDiff (RH,1,mV2500,8,True ,0,_60Hz,0.1,0)
Else
'Generic Single Ended Voltage measurements SEVolt() on 'the AM16/32 Multiplexer: modify this code to read 'current as 0-2.5 and voltage as 0-5V
VoltSe(SEvolt_2(LCount),2,mV5000,15,True,0,_60Hz,1,0)
EndIf
LCount=LCount+2

NextSubScan

Turn AM16/32 Multiplexer Off

PortSet(2,0)

Delay(0,150,mSec)

`Call Data Tables and Store Data

CallTable(Table1)

CallTable(Table2)

NextScan

EndProg