SkyAlert Functions (USB version)

SkyAlert uses a serial port to communicate with the PC.

When a keyword is issued by the computer, the board responds with the current raw data from the sensors in standard text format.
While some of the sensor values are usable as is, others, such as wind speed need to be modified with a formula to produce a useable value.

Sample output directly from the unit

Data-2.8m
18.04
-26.97
1010
739
38.12
519.00
1
101316.22

Values (in order from top to bottom) are:
"Data" (Used for communications check) and firmware version
Ambient temp
Sky Temperature
Dampness
Darkness
Humidity (absolute percentage)
Wind Speed
Power check (Boolean)
Barometric Pressure (hPa * 100)

Barometric Pressure

Usage

Communication parameters:

BaudRate = 9600
Parity = None
DataBits = 8
StopBits = One
WriteTimeout = 500
ReadTimeout = 500

Transmitting the word "send" (without the quotes) followed by a hash tag and carriage return, will invoke the board to respond with the current weather information.
The board will respond with several lines of data which will look something like this:

Data-2.8m
18.04
-26.97
1010
739
38.12
519.00
1
101316.22

Each line is followed by a carriage return.

The first line "Data" is used for checking communications and the firmware version as reported by the SkyAlert firmware code.

The second line is the current ambient temperature (in Celsius).

The third line is sky temperature (in Celsius)

The fourth line is dampness value (Arbitrary value ranging from ~0 to 1000 with 1000 indicating dry and lower values indicating higher amounts of dampness)

The fifth line is brightness value (Arbitrary value ~ 0 to 1000 with 1000 being full brightness and lower values indicating dimmer light levels)

The sixth line is humidity (in percentage)

The seventh line is wind speed (raw value) * see below

The eighth line is power check Boolean 0 or 1 (Used for optional power failure detection) 1 =power is ok, 0 =power has failed

The ninth line is barometric pressure (hectopascal * 100)

SkyAlert-E (Ethernet version)

The Ethernet version of SkyAlert transmits the data in a single line with white spaces separating the data elements.

Sample output directly from the unit:

Data-2.4mE 13.67 -3.91 1010 147 38.30 485.00 1 101353.94

From left to right, the element values are:

The word "Data" followed by a dash mark and the firmware version of the SkyAlert board.

Ambient temperature
Sky temperature
Dampness value
Brightness value
Humidity
Wind speed
Power check Boolean
Barometric pressure

The data is transmitted as standard text which readable by accessing the LAN address of the SkyAlert unit (data can be read with any browser).
(No query string is necessary)

DHCP information from the unit can be attained by a 9600 baud serial port connection made through the unit's native USB connector. When the keyword req# is sent, the unit will respond with the DHCP assigned IP address, the MAC ID and the

LAN Port number.

SkyAlert Remote

The SkyAlert Remote (and SkyAlertRemoteP) programs can be connected directly to the weather unit and will transmit values in a text file which can
be read and parsed by the SkyAlert program when in the 'Remote' mode.

Sample remote text file from SkyAlert RemoteP:

Data
7.92
-4.21
1012
362
49.99
522
1
2.0m
0
30.07

The values contained in the SkyAlert Remote text file are identical to the output generated by the SkyAlert with the exception of an additional entry
of the temperature value generated by an option 'Phidget' temperature sensor which is inserted as the second to last entry. In the sample, the Phidget
temperature value is 0.

Data Definitions:

The dampness value can vary from roughly 400 to 1000 where 1000 is dry and 400 is wet.
Generally speaking, values above 990 or so can be considered dry, a value between 970 and 990 is considered damp and anything below 970 is wet.
These value thresholds should be made user-adjustable to facilitate calibration of the rain sensor.

The brightness value default parameters are: 250 or less=Dark; 250-500=Dim; 500 or higher=Day

---

The wind speed sensor reports wind speed as a non-linear value which can be converted to MPH using the following formula.

Code snippet example:

```
'**************************Calculate Wind Speed*****************************

   If firmwareVersion.Contains("g") = False Then              'using Modern Device solid state sensor
      windSpeedVolts = windspeedReading * 0.0048828125
      ZeroWind_ADunits = (Math.Pow(Ambient_Temperature, 2) * 0.0568) - (5.4311 * Ambient_Temperature) + 491.5
      ZWV = (ZeroWind_ADunits * 0.0048828125)
      zerowindvolts = ((ZeroWind_ADunits * 0.0048828125) - (autoZeroAdjust))
```

```
        wind_speed = Math.Pow(((windSpeedVolts - zerowindvolts) / 0.23), 2.7265)
        wind_speed = ((Math.Pow(wind_speed, 2) * (-0.0063)) + (0.9515 * wind_speed))
        wind_speed = wind_speed + (wind_speed * 0.4)
        Dim windCurveFactor As Double = Form8.WindSpeedCalNumericUpDown.Value * 0.01      ‘user set calibration
        wind_speed = wind_speed + (wind_speed * windCurveFactor)
        If Double.IsNaN(wind_speed) Or wind_speed < 0 Then   'in case resultant value is a negative or NAN value
            wind_speed = 0          'Wind speed in mph
        End If

        '*****************************************Auto calibrate wind sensor*********************************

        If windCalSet = False And wind_speed < 2 And loopIteration > 4 Then     'if wind speed is between 1 and 3,
force the zero volts to less than zero
            zerowindvolts = zerowindvolts + 0.8 : windCalSet = True
        End If

        If loopIteration > 4 And (windSpeedVolts < zerowindvolts) And windspeedReading > 300 And
    windspeedReading < 600 Then

            autoZeroAdjust = Math.Round(ZWV - windSpeedVolts, 3)
            zerowindvolts = (ZeroWind_ADunits * 0.0048828125) - autoZeroAdjust
            My.Settings.autoZero = autoZeroAdjust : My.Settings.Save()
        End If

        zeroWindOffset = Math.Round(windspeedReading - zerowindvolts, 3)

        If Now >= windCalTime.AddHours(1) And windCalSet = True Then     'reset the calibrator every hour
            windCalSet = False
            windCalTime = Now
        End If

        '************************in case wind sensor gets wet and goes nuts**********************
```

```
        If dampness <> 0 And wind_speed > previousWindSpeed + 20 Then
            If previousWindSpeed < wind_speed Then wind_speed = previousWindSpeed
        End If

        If wind_speed > 45 Then             'in case sensor gets wet even if it is not raining
            If previousWindSpeed <= 45 Then
                wind_speed = previousWindSpeed
            Else
                wind_speed = 44
            End If
        End If
        previousWindSpeed = wind_speed

    End If     'end of firmware contains g = false
```

SkyAlert devices that have an optional mechanical wind speed anemometer can be identified by having the letter “g” contained in the firmware version .
Two differing styles could be installed which are virtually identical except for the method in which the wind speed data is formatted. One device produces a voltage while the other produces a pulse signal.
Refer to the following code snippet to identify the type of anemometer in use and the respective calibration method needed.

```
 If firmwareVersion.Contains("g") = True Then     ‘using external wind anemometer

        Form8.btnCalibrate.Visible = False : Form8.lblCalibrateWind.Visible = False : Form8.WindCalLabel.Visible =
False : Form8.lblAdaWind.Visible = True
        Form8.WindSpeedCalNumericUpDown.DecimalPlaces = 0 : Form8.WindSpeedCalNumericUpDown.Increment
= 1

        If loopIteration > 2 And windspeedReading < 70 Then newAnemometerType = True
```

```
If newAnemometerType = False Then     'old style anemometer

    Dim voltageMax As Double = 2.0
    Dim windSpeedMax As Double = 32

    sensorVoltage = windspeedReading * 0.004882814


    If sensorVoltage > 0.3 And sensorVoltage < 0.5 Then    'always seek the lowest reading
        If sensorVoltage < voltageMin Then
            voltageMin = sensorVoltage ' + 0.01 '.01    offset for the periodic small error it makes
        End If
        If (loopIteration > 4) And windCalSet = False Then
            voltageMin = voltageMin + 0.01 : windCalSet = True '   Re-calibrate it every hour if possible
            windCalTime = Now
        End If
        If Now >= windCalTime.AddHours(1) Then    'reset the calibrator every hour
            windCalSet = False
        End If
    End If

    Form9.Label14.Text = "Sensor Voltage=" & sensorVoltage
    Form9.Label6.Text = "VoltageMin=" & voltageMin
    Form9.lblWind.Text = CStr(Math.Round(sensorVoltage, 3)) & " v"
    If windspeedReading <= voltageMin Then
        wind_speed = 0
    Else
        wind_speed = (sensorVoltage - voltageMin) * windSpeedMax / (voltageMax - voltageMin)
    End If
    wind_speed = (wind_speed * 2.2369) 'convert m/s to mph

    If wind_speed < 0 Then wind_speed = 0
```

```
            wind_speed = (wind_speed * 1.25)     'Needs to be boosted by 25% to read properly
            wind_speed = wind_speed + (wind_speed * (Form8.WindSpeedCalNumericUpDown.Value * 0.01))
        End If      'end of new anemometertype

        '*************************************************************************************************************

        If newAnemometerType = True Then     'new style anemometer
            voltageMin = 0
            Dim voltageMax As Double = 5.0
            Dim windSpeedMax As Double = 32.4
            sensorVoltage = windspeedReading * 0.004882814

            If sensorVoltage > 0 And sensorVoltage < 0.09 Then     'always seek the lowest reading
                voltageMin = sensorVoltage ' + 0.01 '.01     offset for the periodic small error it makes
                If (loopIteration > 4) And windCalSet = False Then
                    voltageMin = voltageMin + 0.01 : windCalSet = True     'Re-calibrate it every hour if possible
                    windCalTime = Now
                End If
                If Now >= windCalTime.AddHours(1) Then     'reset the calibrator every hour
                    windCalSet = False
                End If
            End If


            Form9.Label14.Text = "Sensor Voltage=" & sensorVoltage
            Form9.Label6.Text = "VoltageMin=" & voltageMin
            Form9.lblWind.Text = CStr(Math.Round(sensorVoltage, 3)) & " v"
            If windspeedReading <= voltageMin Then
                wind_speed = 0
            Else
                wind_speed = (sensorVoltage - voltageMin) * windSpeedMax / (voltageMax - voltageMin)    'wind speed in
meters/second
            End If
```

```
            wind_speed = (wind_speed * 2.2369)      'convert m/s to mph

            If wind_speed < 0 Then wind_speed = 0


            wind_speed = wind_speed + (wind_speed * (Form8.WindSpeedCalNumericUpDown.Value * 0.01))
'calibrate value with value in calibration dialog


        End If      'end of new anemometertype
    End If      'end of firmware g'
```

Cloud cover conditions can be calculated by subtracting the sky temperature from the ambient temperature and comparing the result to a constant value of 17 degrees C subtracted from the ambient temperature.
Offset value should be adjustable by user to compensate for altitude, change of season and other variables.