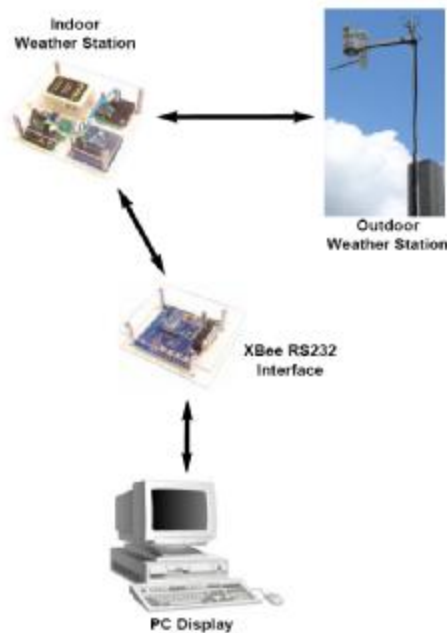


# Control Your World

## “Build a Wireless Weather System”

### Part 4

By Michael Simpson



In the last three installments you built the three satellites shown in Figure 2. I showed you how to transmit and collect basic data. In this final installment I am going to detail the protocol that I created that will allow you to transmit various readings. These readings can be weather related, or from various other pieces of telemetry you may want to add to the mesh network.

The reason I created a special protocol instead of just sending text messages was so that simple telemetry messages could be sent to any device. You will be able to add any kind of display system and pick out just the messages you are interested in.



**Figure 2**

## The Wireless Weather Protocol

The basic protocol format in its simplest form is shown in Table 1. The first byte in the packet is the **Start Of Packet (SOP)**. This is always a value of 2. The SOP byte is followed by the packet size. The packet size is the number of bytes that follow. The remaining bytes are the actual data bytes followed by a 2 byte checksum.

Byte	Example	Description
1	2	Start of packet indicator.
2	8	Number of bytes to follow
3	100	Data byte
4	101	Data byte
5	102	Data byte
6	103	Data byte
7	104	Data byte
8	105	Data byte
9 & 10	111 & 102	2 byte checksum

**Table 1**

There are some additional rules that must be followed:

### Rule 1

- There must always be at least 2 data bytes and 2 checksum bytes for a packet size of 4. More are ok, but no less.

### **Rule 2**

- If any of the data byte values are 0-3, then they are preceded by a 3. The value is then added to 100. For example, a value of 1 would be represented as 3,101

### **Rule 3**

- If either byte in the checksum is less than 4, it is added to 100. Note that unlike the data bytes, the checksum bytes will not be preceded by a 3 when this happens.

While this may sound a little like overkill, it does solve a few potential problems.

1. If a satellite is started, it won't start looking at any packets until it gets a true valid start of packet.
2. If a collision takes place between two satellites, the checksum won't match and the message will be discarded.
3. If any message gets corrupted, it will be discarded.

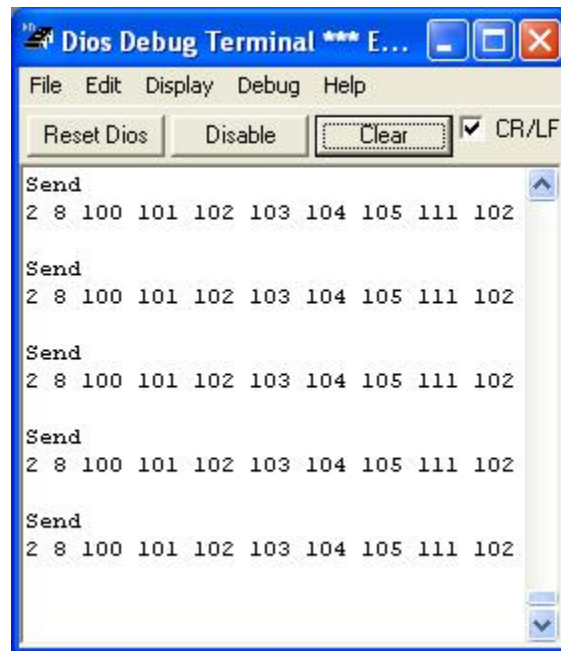
One other point I need to make is that I set the baud rate to 9600. Originally, I wanted to use 115200 as the baud rate, but the XBee modules had a problem with the actual timing needed to duplicate this baud rate.

To make life easier, I have created both packet builder and decoder functions in Zeus for the Pocket PC and Desktop, as well as functions for the DiosPro chips.

Let's try a little experiment by using the indoor satellite to transmit a message to the PC Satellite.

### **Step 1**

Power down the outdoor satellite. This will keep it from interfering with our protocol test. Program the Indoor Weather Satellite with the Dios code shown in Program 1. (DiosPacKeSent.txt) The main program calls the **sendWpacket** function once every 5 seconds. The **sendWpacket** function is the heart of the program. You pass from 2 to 16 data bytes and the function will build and transmit a weather packet for you. I also included a couple of print statements so the debug window will report the bytes in the packet that are sent as shown in Figure 3.



**Figure 3**

```

DiosPro
'DiosPacketSend.txt
'Packet Test Program
func main()
  hsersetup baud,HBAUD9600,start,txon,inwait,500

loop:
  print "Send"
  sendWpacket(100,101,102,103,104,105)
  print
  pause 5000
  goto loop

endfunc

'-----
'Send a Weather Packet
' Pass
' up to 16 bytes. Each byte must be value of 0-255
'-----
func sendWpacket(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16)
  dim sum as integer
  dim x as integer

```

```

dim dat as integer
dim ct as integer
dim ect as integer
ct = OPP8
sum = 0
ect = 0

if ct < 2 or ct > 16 then
  print "Invalid Number of Data bytes. Send 2-16"
  exit 0
endif

'First see how many escapes
for x = 0 to ct - 1
  dat = x1(x)
  if dat < 4 then
    inc ect
  endif
next

sum = sum + ct+2+ect
hserout 2,ct+2+ect
print 2," ",ct+2+ect," ";
for x = 0 to ct - 1
  dat=x1(x)
  sum = sum + dat
  if dat < 4 then
    sum = sum + 3 + 100
    hserout 3,dat+100
    print 3," ",dat+100," ";
  else
    hserout dat
    print dat," ";
  endif
next

if sum.byte(0) < 4 then sum.byte(0) = sum.byte(0)+100
if sum.byte(1) < 4 then sum.byte(1) = sum.byte(1)+100
hserout sum.byte(0),sum.byte(1)
print sum.byte(0)," ",sum.byte(1)
exit 1
endfunc

```

**Program 1**

## Step 2

Now load up a copy of ZeusPro and load the code shown in Program 2. (ZeusRXPacket.txt) You will need to change the com port to that of your X-CTU (PC Satellite). Once you compile and run the program you should see the bytes received by each packet as shown in Figure 4. This program calls the function called **getweatherpacket**. In Zeus, the **getweatherpacket** function returns a string containing the packet data bytes.

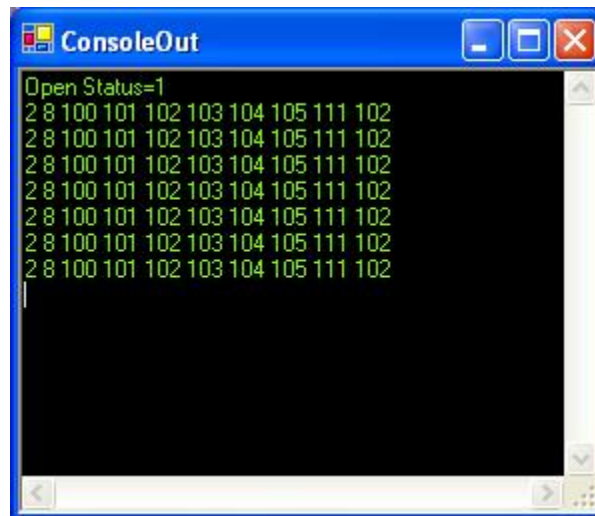


Figure 4

```
'Zeus Receive packet test
'ZeusRXPacket.txt
func main()
  dim x as integer
  dim tstr as string

  x=ComOpen(1,baud=9600,port=5) '<--- Set Com Port here
  print "Open Status=";x
  ComBGSuspend(1,0)

loop:
  sleep(1)
  doevents()
  tstr = getweatherpacket()
```

```

goto loop

endfunc

'-----
' This looks for a single packet of data.
' Returns empty string if no or bad packet
' Returns packet inside string if success
'-----

func getweatherpacket() as string
    dim sum as integer
    dim state as integer
    dim count as integer
    dim plen as integer
    dim dat as integer
    dim byte0 as integer
    dim byte1 as integer
    dim sbyte0 as integer
    dim sbyte1 as integer
    dim x as integer
    dim tstr as string
    tstr=""

    x=ComWaitforByte(1,100)
    if x <> 2 then exit("")
    print 2+" ";

    'Get packet Length
    plen = ComWaitforByte(1,5000)
    if plen = -1 then
        print "Len Error"
        exit("")
    endif

    print plen+" ";

    sum = plen
    for x = 1 to plen
        dat = ComWaitforByte(1,5000)
        if dat = -1 then
            print "Byte Error ",x
            exit("")
        endif

```

```

print dat+" ";

if x < plen - 1 then
  sum = sum + dat
endif

if state = 1 then
  state = 0
  dat = dat - 100
  tstr = tstr + chr(dat)
  goto cont
endif

if dat = 3 then
  state = 1
else
  tstr = tstr + chr(dat)
endif

cont:
next

print
'print sum
byte0 = sum & 255
if byte0 < 4 then byte0 = byte0+100
byte1 = sum & 65280 / 256
if byte1 < 4 then byte1 = byte1+100
'print byte0,byte1
sbyte0 = asc(mid(tstr,len(tstr)-1,1))
sbyte1 = asc(mid(tstr,len(tstr),1))
'print sbyte0,sbyte1
if byte0 <> sbyte0 or byte1 <> sbyte1 then
  print "Sum Error"
  exit("")
else
  exit(left(tstr,len(tstr)-2))
endif
endfunc

```

## Program 2

There is a bit more needed in order to put the protocol to work in our network. The first two data bytes have been predefined as detailed in Table 2. The data field is the sending satellite ID. Each satellite will have a unique ID that you will assign. In the case of our Indoor Weather Satellite it is 30. When a display device decodes the receive packet it will be possible to tell where the signal is coming from. The second data field is the type of reading. This value is important as it tells the display satellite the kind of data that is coming.

<b>Data Field</b>	<b>Byte</b>	<b>Description</b>
1	3	Satellite ID of Sender
2	4	Reading Type
3	5	Data byte 1
4	6	Data byte 2
5	7	Data byte 3
6	8	Data byte 4

**Table 2**

Let's take our experiment to the next level.

### **Step 3**

Program the Indoor Weather Satellite with the Dios code shown in Program 3 (DiosSendTemp.txt). In this program we take a 1Wire temperature reading from the DS1820. The result is passed to the **sendWpacket** function. We send our fromid and the reading type of 72. We then send all four bytes that make up the floating point temperature.

There are a few things you need to know in order for this program to work. First, make sure the DS1820 is plugged into the satellite as outlined in Part 2 of this series. You then need to program the **IndoorNetworksearch.txt** program into the DiosPro. This program will display all the network 1Wire serial numbers (roms) connected to the satellite as shown in Figure 5. The reading we are interested in for this test is the DS1820 Thermometer. Use that reading to populate the 8 Hex numbers in the table command at top of Program 3. One other thing to keep in mind is that I added a bit of code to pull port 13 high because we use that port to supply 5v to the 1Wire network.

```

DiosPro
'DiosSendTemp.txt
'Send the Indoor Temp from Indoor Weather Satellite
func main()

table InTEMPROM: $10,$4F,$51,$41,$00,$08,$00,$EB 'DS1820 Thermometer

```

```
gconst fromid 30 'MY ID.
```

```
gconst ownw1port 12 'Port for 1Wire Network 1
```

```
'Port 13 used for 1K pullup
```

```
output 13
```

```
high 13
```

```
pause 100
```

```
global temperature as float
```

```
hsersetup baud,HBAUD9600,start,txon,inwait,500
```

```
loop:
```

```
temperature = DS1820readtemp(ownw1port,3,InTEMPROM)
```

```
print "Send Temp of ",temperature
```

```
sendWpacket(fromid,72,temperature.byte(0),temperature.byte(1),_  
temperature.byte(2),temperature.byte(3))
```

```
pause 5000
```

```
goto loop
```

```
endfunc
```

```
'-----  
'Send a Weather Packet  
' Pass  
' up to 16 bytes. Each byte must be value of 0-255  
'-----
```

```
func sendWpacket(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16)
```

```
dim sum as integer
```

```
dim x as integer
```

```
dim dat as integer
```

```
dim ct as integer
```

```
dim ect as integer
```

```
ct = OPP8
```

```
sum = 0
```

```
ect = 0
```

```
'First see how many escapes
```

```
for x = 0 to ct - 1
```

```
dat = x1(x)
```

```
if dat < 4 then
```

```
inc ect
```

```
endif
```

```
next
```

```

if ct < 2 or ct > 16 then
  print "Invalid Number of Data bytes. Send 2-16"
  exit 0
endif

sum = sum + ct+2+ect
hserout 2,ct+2+ect
print 2," ",ct+2+ect," ";
for x = 0 to ct - 1
  dat=x1(x)
  sum = sum + dat
  if dat < 4 then
    sum = sum + 3 + 100
    hserout 3,dat+100
    print 3," ",dat+100," ";
  else
    hserout dat
    print dat," ";
  endif
next

if sum.byte(0) < 4 then sum.byte(0) = sum.byte(0)+100
if sum.byte(1) < 4 then sum.byte(1) = sum.byte(1)+100
hserout sum.byte(0),sum.byte(1)
print sum.byte(0)," ",sum.byte(1)
exit 1
endfunc

include \lib\Dios1820.lib

```

### Program 3



Figure 5

#### Step 4

Load up the program ZeusRXTemp.txt into Zeus Pro and compile the program. The program is rather large so the listing in Program 1 only represents the most important portions of the code. In this program we make a call to the **getweatherpacket** function. If a packet of data is received a call to the **procweatherpacket** function is called. This function parses the packet data and pulls the transmitting satellite ID and reading type and places them in the fromid and cmd variables. A test is then performed to see if the reading type is 72 (Indoor Temp). If it is, the 4 floating point bytes are pulled. Floating point variables on the PC are not stored the same way as they are on the DiosPro, so a call to the **DiostoPCfloat** function is made to convert the 4 received bytes to a single precision floating point compatible with the PC. Finally, a call is made to convert the DS1820's Celsius reading to Fahrenheit. The data is then displayed using the print command.

This is how all readings are parsed. Once parsed, it's up to the particular satellite display program to utilize this data and convert it into the appropriate format so that it can be used.

```

'Zeus Receive Indoor Temperature
'ZeusRXTemp.txt
func main()
  dim x as integer
  dim tstr as string

  x=ComOpen(1,baud=9600,port=8) '<--- Set Com Port here
  print "Open Status=";x
  ComBGSuspend(1,0)

loop:
  sleep(1)

```

```

doevents()
tstr = getweatherpacket()
strif tstr <> "" then
    procweatherpacket(tstr)
endif
goto loop

endfunc

'-----
'Main Weather Processor
'-----
func procweatherpacket(tstr as string) as integer
    dim b0 as integer
    dim b1 as integer
    dim b2 as integer
    dim b3 as integer

    dim slen as integer
    slen = len(tstr)
    if slen < 2 then exit(0)

    global cmd as integer
    global fromid as integer

'-- Weather Variables
    global itemperature as float

    fromid = asc(mid(tstr,1,1))
    cmd = asc(mid(tstr,2,1))

'-- Test for the reading type
    select integer cmd
    case 72
        b0 = asc(mid(tstr,3,1))
        b1 = asc(mid(tstr,4,1))
        b2 = asc(mid(tstr,5,1))
        b3 = asc(mid(tstr,6,1))
        otemperature= cvtDiostoPCfloat(b3,b2,b1,b0)
        if otemperature <> 255 then otemperature = 1.8 * otemperature + 32
        print "From "+fromid+": ", "Inside Temperature="+otemperature+"F"
    endselect

endfunc

```

## Program 4

### Indoor Weather Satellite Firmware

The program called **IndoorWeather.txt** program takes three readings. Indoor Humidity, Barometer, and Indoor Temperature. An internal timer is setup to count milliseconds and each reading is taken at 1 minute intervals. As each reading is taken, the packet is sent. Once all four readings are taken, a status packet is sent. Take a look at the beginning of this program and you will see a definition of each reading packet that is sent.

Again you will need to populate the Rom tables at the start of the program with the ones that map to your particular 1-Wire devices.

### Outdoor Weather Satellite Firmware

The program called **Outdoorweather.txt** works much the same way as the indoor program. The major difference is the number and type of sensors that are read. There is also processing done to calculate the wind speed variations. My particular outdoor satellite monitors is an AAG wind speed and wind dir sensor. It monitors a temperature sensor (DS1820) and a humidity sensor. It watches a HobbyBoard humidity gauge. It also takes 4 10bit AtoD readings and a modified rain gauge. All the code to handle and display these sensors are include in the outdoor and PC satellite software. Even if you don't utilize them, they will send empty packets so you can just ignore the data.

There is quite a bit of program code associated with this program. Several libraries are included in the download. They consist of the following libraries:

- **WLhum.lib**
- **WLrain.lib**
- **WLtemp.lib**
- **WLdir.lib**
- **WLlight.lib**
- **WLSpeed.lib**
- **WLtimers.lib**

### PC Monitor Satellite

On the PC side I have included a program called PCSatellite.txt. This program can handle several reading types from both the indoor and outdoor satellites. I have also included a special version called PCSatelliteForm.txt that allows you to select the com

port. I have also included a compiled version of the form version so you don't have to worry about programming the code. The output for both programs is shown in Figure 6. In this particular example I don't have any sensors connected to the Outdoor Satellite, so it shows empty packets.

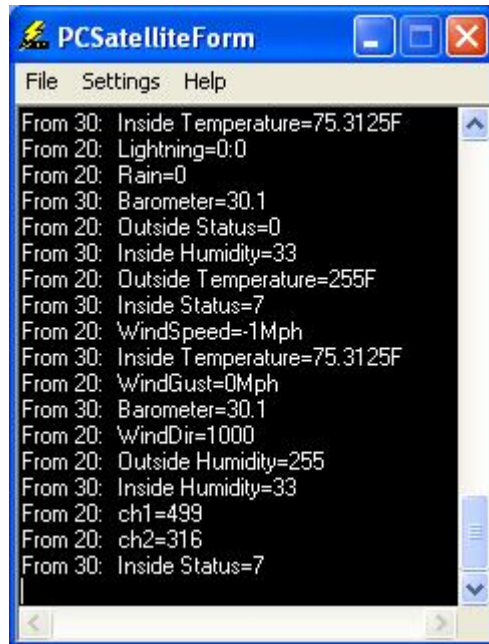


Figure 6

## Final Thoughts

I have been asked by a few individuals about the MaxStream XBee series 2 modules. I have not been able to get my hands on any of these, but I can tell you that they will not work with this project without certain code modifications, nor are they compatible with the original XBee series.

I have purchased an XBee Pro module and hope to add it to my network as the coordinator. The Pro module has much more range than the standard modules. It also has a much more sensitive receiver, so I should be able to extend my network by adding just one of these to the network. Once I do, I hope to provide some updates on the Kronos Robotics website.

I built the following satellites for my weather station:

- **LCD Display**
- **LED Sign**
- **Wood Stove Monitor**
- **X10 Interfaces**

I plan on offering these as future articles as time permits.

Be sure to check for updates at:

<http://www.kronosrobotics.com/Projects/wirelessweather.shtml>

## Parts

The following is a breakdown of the source for all the components needed for Parts 1-4 of this project.

### MaxStream

Starter Kit #XB24-DKS

[http://store.maxstream.net/index.cfm?fuseaction=product.display&Product\\_ID=1](http://store.maxstream.net/index.cfm?fuseaction=product.display&Product_ID=1)

Also available from Mouser at:

<http://www.mouser.com/search/ProductDetail.aspx?R=XB24-DKSvirtualkey61440000virtualkey888-XB24-DKS>

Or do a search for XB24-DKS

### Hobby Boards

Wind Instrument

[http://www.hobby-boards.com/catalog/product\\_info.php?cPath=22&products\\_id=92](http://www.hobby-boards.com/catalog/product_info.php?cPath=22&products_id=92)

Humidity Module (H3-R1-A)

[http://www.hobby-boards.com/catalog/product\\_info.php?cPath=22&products\\_id=46](http://www.hobby-boards.com/catalog/product_info.php?cPath=22&products_id=46)

Lightning Detector

[http://www.hobby-boards.com/catalog/product\\_info.php?cPath=22&products\\_id=65](http://www.hobby-boards.com/catalog/product_info.php?cPath=22&products_id=65)

### Spark Fun Electronics

XBee Breakout Board (Used to build various interface boards)

[http://www.sparkfun.com/commerce/product\\_info.php?products\\_id=8276](http://www.sparkfun.com/commerce/product_info.php?products_id=8276)

2mm connectors (You need 2 for each Breakout board)

[http://www.sparkfun.com/commerce/product\\_info.php?products\\_id=8272](http://www.sparkfun.com/commerce/product_info.php?products_id=8272)

## **Kronos Robotics**

DiosPro 28 chip

<http://www.kronosrobotics.com/xcart/customer/product.php?productid=16429>

Dios Carrier 1

<http://www.kronosrobotics.com/xcart/customer/product.php?productid=16170>

3.3v to 5v Interface Kit

<http://www.kronosrobotics.com/xcart/customer/product.php?productid=16537>

40-Pin Male Header

<http://www.kronosrobotics.com/xcart/customer/product.php?productid=16290>

1K resistors

<http://www.kronosrobotics.com/xcart/customer/product.php?productid=16178>

Regulator Kit

<http://www.kronosrobotics.com/xcart/customer/product.php?productid=16304>

Free Dios Compiler (Includes 1-Wire libraries)

<http://www.kronosrobotics.com/downloads/DiosSetup.exe>

## **KRMicros**

ZeusPro Compiler

<http://www.krmicros.com/Development/ZeusPro/ZeusPro.htm>

## **SchmartBoard**

Prototyping board (.1")

<http://www.schmartboard.com/index.asp?a=11&id=24>

Jumpers 5" Yellow

<http://www.schmartboard.com/index.asp?a=11&id=42>

Jumpers 3" Red

<http://www.schmartboard.com/index.asp?a=11&id=41>

## **Links**

Hobby Boards

<http://www.hobby-boards.com>

Spark Fun Electronics

[www.sparkfun.com](http://www.sparkfun.com)

Kronos Robotics

<http://www.kronosrobotics.com/xcart/customer/home.php>

ShmartBoard

[http://www.schmartboard.com/index.asp?a=11&page=a\\_products](http://www.schmartboard.com/index.asp?a=11&page=a_products)

Jameco Electronics

<http://www.jameco.com>