WeatherBox

Paul Beeken
Knowledge Software Consulting
info@knowsoft.com
http://www.knowsoft.com

Introduction

WeatherBox is a freeware program I developed for my own use to retrieve information from my Weather Monitor II® and Weather Link® equipment. Davis originally distributed a program called Maclink with the box. MacLink did the job but it lacked AppleScripting capability[1] I wanted the means to retrieve information and place it in databases and more significantly retrieve the information via a cgi on a network. The first phase of this ability was to develop an AppleScript capable application.

I developed this program on my own time and with my own resources thanks to the public release of the of the Weather Link addresses. Except for these numbers I built everything else as a hobby and for my own needs. Keep this in mind when you find a bug. I am donating this software for the public benefit and have every desire to make it robust and useful. My time is somewhat limited so these improvements, when they come may be slower than a commercial application.

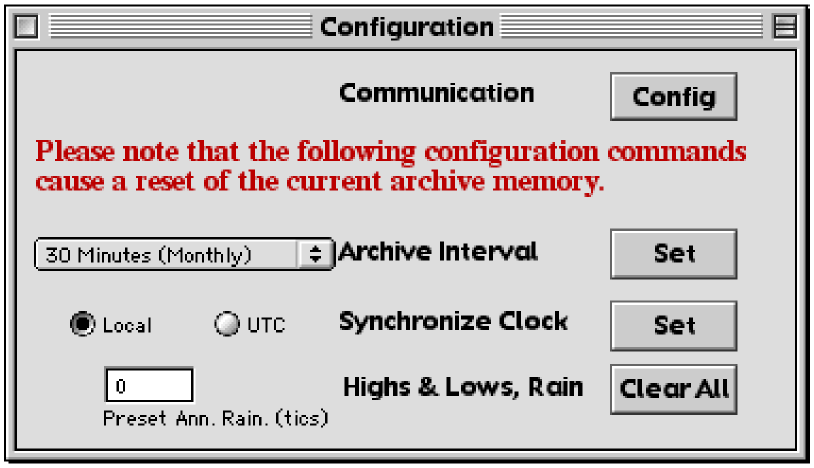
I only own a Weather Monitor II so the software is developed exclusively for that box. Launching the application will test the communication to the Weather Monitor and report the connection status on the splash screen. If the message indicates that the connection failed then the only menu options that will be presented are Quit and Port Setup.

Paul Beeken
Knowledge Software Consulting
1204 Hall Ave
White Plains, NY 10604
info@knowsoft.com
http://www.knowsoft.com

Legalese

[1] This is not a slight to Davis. At the time, AppleScript was a fairly new concept and one that was very difficult to implement.

Port Setup


Configuration

Communication [Config]

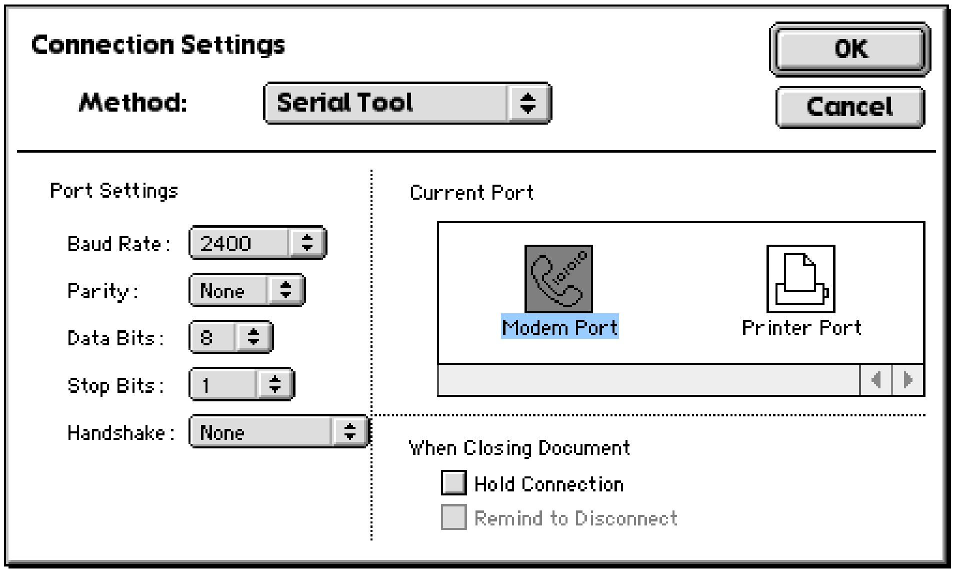
Please note that the following configuration commands cause a reset of the current archive memory.

Archive Interval: 30 Minutes (Monthly) [Set]

Synchronize Clock: [✓ Local] UTC [Set]

Highs & Lows, Rain: Preset Ann. Rain. (tics) 0 [Clear All]

This is a simple dialog that allows you to configure the port, clear the WeatherLink memory, set the time, set the archiving interval and clear Highs and Lows. The interval, clock and Highs and Lows button wont be available if the port is not set up correctly. If this is the first time you launch the program then the first time your should configure the port.


Connection Settings

Method: Serial Tool

[OK] [Cancel]

Port Settings

Baud Rate: 2400

Parity: None

Data Bits: 8

Stop Bits: 1

Handshake: None

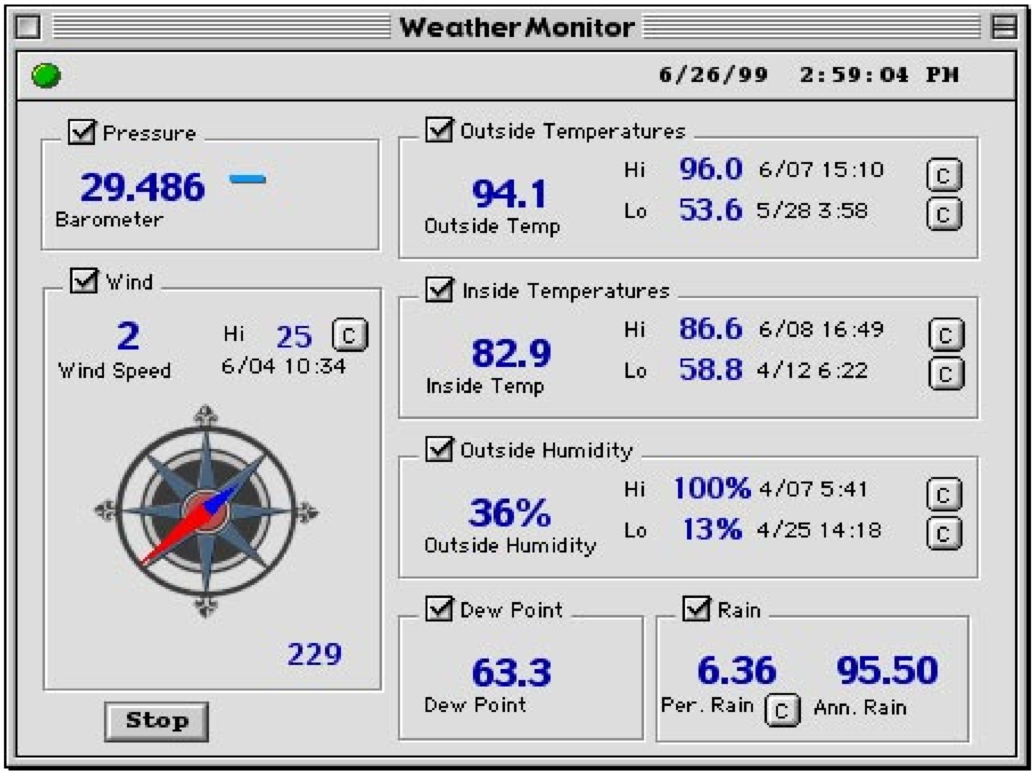
Current Port: [✓ Modem Port] Printer Port

When Closing Document

☐ Hold Connection

☐ Remind to Disconnect

I use the Communications Manager to set up the port so if you don't have this toolkit installed (MacOS 7.1 or greater) you wont be able to set up the port. The program wouldn't launch if you didn't have the CommToolBox, so if you didn't have it you wouldn't get this far. You must configure the connection for 2400 baud, no parity, 8 bits, no handshaking, and set the port to the correct input. The advantage of this application is that it will work with any Mac (even the USB Macs that have a USB/Serial adaptor). Click OK. .If the line was working then the buttons, previously disabled will now be enabled. If not check your cables.


Weather Monitor

6/26/99 2:59:04 PM

☑ Pressure
29.486
Barometer

☑ Wind
2
Wind Speed
Hi 25 C
6/04 10:34
229

Stop

☑ Outside Temperatures
94.1
Outside Temp
Hi 96.0 6/07 15:10 C
Lo 53.6 5/28 3:58 C

☑ Inside Temperatures
82.9
Inside Temp
Hi 86.6 6/08 16:49 C
Lo 58.8 4/12 6:22 C

☑ Outside Humidity
36%
Outside Humidity
Hi 100% 4/07 5:41 C
Lo 13% 4/25 14:18 C

☑ Dew Point
63.3
Dew Point

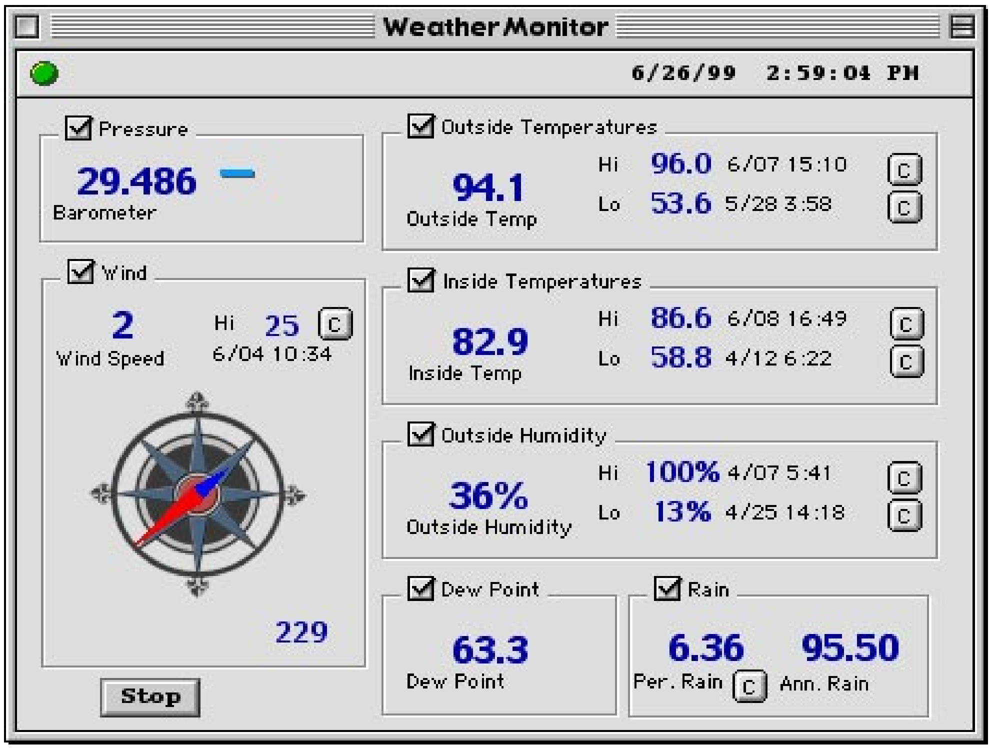
☑ Rain
6.36
Per. Rain C
95.50
Ann. Rain

Monitor Window

Once the connection has been setup, Three sub-menus will appear under the 'New' Menu item under the File Menu. The first of these is the Window responsible for communicating to the Weather Monitor II itself. This is window continuously polls the Weather Monitor for its status. Polling the weather monitor is slow so it takes a while to cycle through all the items. To further slow things down I delay the polling of the interface to allow other processes time for activity. Currently, the Mac is a 'cooperative multitasking environment' which means the programmer must be nice and yield time for the processor to do other things. I also designed this program to act as a kind of background server so I didn't want it to be hogging all the processor time.

The window has a handful of controls but they fall into one of 2 categories. The check boxes above each group will disable the polling of that particular data item. The other control is a button that clears the extrema for that particular value. For example the check box above the Humidity group disables polling for the entire humidity group while the small "C" next to the Hi value will clear that particular 'hi' value.

"Stop" at the bottom of the window suspends all the polling. Note that AppleScript requests for information is not hindered by this suspension. In other words, you can still get data from the Monitor.



6/26/99 2:59:04 PM

Pressure
29.486
Barometer

Wind
2
Wind Speed
Hi 25 C
6/04 10:34
229

Outside Temperatures
94.1
Outside Temp
Hi 96.0 6/07 15:10 C
Lo 53.6 5/28 3:58 C

Inside Temperatures
82.9
Inside Temp
Hi 86.6 6/08 16:49 C
Lo 58.8 4/12 6:22 C

Outside Humidity
36%
Outside Humidity
Hi 100% 4/07 5:41 C
Lo 13% 4/25 14:18 C

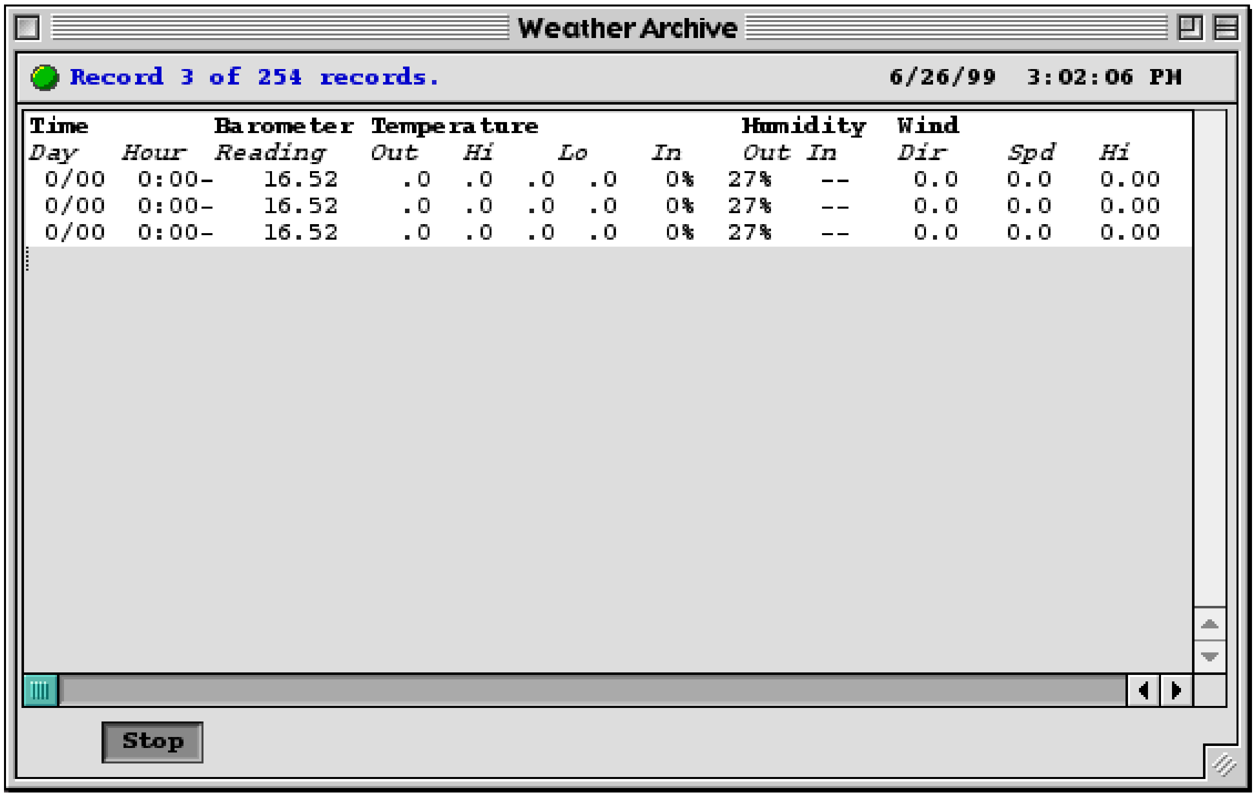
Dew Point
63.3
Dew Point

Rain
6.36 95.50
Per. Rain C Ann. Rain

Link Window

Like the Monitor Window described above this gets information directly from the WeatherLink communication device. Retrieving data from the Link is much faster and so the polling goes much quicker.. However, for the reasons stated above, I deliberately keep this rate low to allow the computer to do other things.

Like the Monitor window the Link window has buttons to stop polling for that particular piece of datum The controls are similar to those on the Weather Monitor Window.



```
Record 3 of 254 records.                         6/26/99   3:02:06 PM
Time         Barometer Temperature            Humidity  Wind
Day   Hour   Reading   Out   Hi      Lo    In    Out In   Dir     Spd    Hi
 0/00  0:00-    16.52    .0  .0  .0  .0    0%  27%   --    0.0    0.0    0.00
 0/00  0:00-    16.52    .0  .0  .0  .0    0%  27%   --    0.0    0.0    0.00
 0/00  0:00-    16.52    .0  .0  .0  .0    0%  27%   --    0.0    0.0    0.00
```

Archive Window

This window was difficult to design because I could not decide how best to present the information archived in the WeatherLink system. I finally opted for a window that polls the Weather Link slowly and builds up the historical data and presents it in a text edit window that can be copied to any other application. Like all the facilities in the application the real power comes from using AppleScript to get this information.Like the other two windows the polling is deliberately slow to allow lots of processor time for other things.

I do not, currently, save this information to a database file. I was in the process of adding this feature when I discovered that Davis made available their database structure. I want to be compatible so I may adopt it. Then again, I may design my own. I am still on the fence. Any suggestions?

AppleScript Control

This is where the real power of this application resides. Originally I wanted to make this program that was a 'Background Only' application. Soon, I realized that the only reasonable way to debug such a program was to allow some user interaction. Still the user interaction is slow and stilted. This was so that the program can co-reside with an HTTP server like Apple's Personal Web Server and provide information via the cgi handlers. The idea is that by loading a page you can post a simple request that then polls the Weather Link, composes the information and spits it back to the requesting client. It works astonishingly well.

I am not about to give a complete AppleScript tutorial here. Those who know applescript will probably not even need the overview I have provided here as most of the key bits and pieces are in the dictionary for WeatherBox. Suffice it to say that AppleScript is a fairly straight forward programming language for controlling programs running on the Mac. More than controlling programs it can also be used to retrieve and manipulate information. All that a programmer designing an application need do is to provide the interpretation of the signals sent to the application to do all this. The means to build these commands and the documentation that goes with it are contained in a data structure called the dictionary that can be viewed in the Script Editor.

The complete dictionary can be found by the usual methods. I will describe only the key elements that pertain to control of the Weather Monitor II here.

Class application: An application program

Elements:

```
        ArchiveWindow by numeric index, by name
        MonitorWindow by numeric index, by name
        LinkWindow by numeric index, by name
```

Properties:

```
        link active application [r/o] -- Tests the serial link status to the weatherlink device.
```

Is Connected: Is the computer connected to the weather link or weather monitor?

```
        Is Connected
        Result:  boolean -- status of connection
```

The Application has three elements or objects that it needs to communicate with the Monitor. Each corresponds to a window accessible to the interface. There is also a property which allows you to determine if the monitor is still connected[2] . As you might expect this simply returns a true or false depending on the current connection status.

In order to get information from the Weather Monitor you need to establish at least one of the three kinds of windows. This is usually done via a `make new <window class>` command Once you have established a window, you can poll that element of the Monitor. Although one instance of each of these windows can be created it only really makes sense to create one at a time.

[2] Some will argue that this shou be a directive, not a property since it initiates an action. The result of the action is clealy a property but since the property that is not cached and only determined at request it is one of those grey areas. I have provided a verb as well to satisfy the purists.

Basic Classes

Class weather record: record of current, high, and low values.

Properties:

```
current real -- current value
high real -- highest value since last reset
high date date -- date of highest value since last reset
low real -- lowest value since last reset
low date date -- date of lowest value since last reset
```

Class pressure record: record of pressure and trend.

Properties:

```
current real -- current value
trend Up/Steady/Down -- trend in value
```

Class wind record: record of wind values.

Properties:

```
current real -- current wind speed
direction integer -- current wind direction
high real -- high wind speed since last reset
high date date -- date of high since last reset
```

Class rain record: record of rainfall.

Properties:

```
periodic rain real -- periodic, resettable rainfall.
annual rain real -- accumulated rainfall for the year.
```

Class archive record: A single record of archival data from the weather link.

Properties:

```
date date [r/o] -- date and time for this archive record.
barometer real [r/o] -- barometric pressure at time of archive.
outside avg temperature real [r/o] -- average outside temperature at time of archive.
inside avg temperature real [r/o] -- average inside temperature during archive interval.
outside hi temperature real [r/o] -- high outside temperature during archive interval.
outside lo temperature real [r/o] -- low outside temperature during archive interval.
inside humidity integer [r/o] -- average inside humidity during archive interval.
outside humidity integer [r/o] -- average outside humidity during archive interval.
avg wind real [r/o] -- average wind during the archive interval.
hi wind real [r/o] -- wind gust during the archive interval.
wind direction integer [r/o] -- predominant wind direction during archive interval.
rain real [r/o] -- rainfalll in the archive interval.
```

The data returned by the objects are almost always associative records of the kind described above. This allows for the easy organization of the various characteristics. I have tried to group them logically so extracting the desired information is straight forward.

To access the information, as mentioned above, you first need to create and instance of particular type of window. The properties of the window are the data elements (since verbs are directed to the application and not the window). Many of the properties do indeed, initiate an action (fetching the data, for example) so you have to be prepared in the AppleScript code for a long wait (in computer time). The default response wait time of 120 seconds should be plenty for many requests but if you should get a timeout error (I haven't among my tests, but just in case) you should wrap your property requests with a `with timeout xxx seconds ... end timeout` group.

MonitorWindow

Class MonitorWindow: A window to the WeatherMonitor

Properties:

`<inherited>` `window` [r/o] *-- window*
`connected` `boolean` [r/o] *-- tests the connection to the device and returns its status.*
`polling` `boolean` *-- is the window polling?*
`polling time` `small integer` *-- the number of seconds between polling intervals.*
`barometer` `pressure record` [r/o] *-- the barometric pressure*
`inside temperature` `weather record` [r/o] *-- a record of inside temperatures*
`outside temperature` `weather record` [r/o] *-- a record of outside temperatures*
`dew points` `weather record` [r/o] *-- a record of dew points*
`inside humidity` `weather record` [r/o] *-- a record of inside humidity*
`outside humidity` `weather record` [r/o] *-- a record of outside humidity*
`wind` `wind record` [r/o] *-- a record of wind values*
`rain` `rain record` [r/o] *-- a record of rainfall*

Creating an object of this type will display a monitor window and will begin polling right away. You can stop polling at any time by setting the polling property to **false.** You can still access any and all properties when polling is off. This simply stops the automatic polling of the Weather Monitor. You can slow down the automatic polling by changing the polling time.

The 'connected' property is a true property unlike the 'link active' property of the application. This simply returns the current state of the communication based on the last poll. For example if polling is turned off and the Weather Monitor / Link becomes disconnected this may still return true.

Most of the data properties are self explanatory. They simply return the information from the Weather Monitor in terms of the previously described records.

LinkWindow

Class LinkWindow: A window into the WeatherLink

Properties:

`<inherited>` `window` [r/o] *-- window*
`connected` `boolean` [r/o]
`polling` `boolean` *-- is the window polling?*
`polling time` `small integer` *-- the number of seconds between polling intervals.*
`barometer` `pressure record` [r/o] *-- the barometric pressure*
`inside temperature` `weather record` [r/o] *-- a record of inside temperatures*
`outside temperature` `weather record` [r/o] *-- a record of outside temperatures*
`dew points` `weather record` [r/o] *-- a record of dew points*
`inside humidity` `weather record` [r/o] *-- a record of inside humidity*
`outside humidity` `weather record` [r/o] *-- a record of outside humidity*
`wind` `wind record` [r/o] *-- a record of wind values*
`rain` `rain record` [r/o] *-- a record of rainfall*

Creating an object of this type will display a link window and will begin polling right away. You can stop polling at any time by setting the polling property to **false.** You can still access any and all properties when polling is off. This simply stops the automatic polling of the Weather Link. You can slow down the automatic polling by changing the polling time.

The Weather Link is the additional electronic component attached to the Weather Monitor whose job is to provide communication and storage of weather information. The 'connected' property is a true property unlike the 'link active' property of the application. This simply returns the current state of the communication based on the last poll. For example if polling is turned off and the Weather Monitor / Link becomes disconnected this may still return true.

As with the MonitorWindow, most of the data properties are self explanatory. They simply return the information from the Weather Link in terms of the previously described records.

ArchiveWindow

Class ArchiveWindow: The window into the archived data from the WeatherLink.

Properties:

```
<inherited> window [r/o] -- window
connected boolean [r/o]
polling boolean -- is the window polling?
polling time small integer -- the number of seconds between polling intervals.
current index integer -- returns or sets the current archive record index.
maximum index integer [r/o] -- Last index available from the archive.
link archive archive record [r/o] -- the arcival record from the current index.
```

Creating an object of this type will display a monitor window and will begin polling right away. You can stop polling at any time by setting the polling property to **false.** You can still access any and all properties when polling is off. This simply stops the automatic polling of the Weather Monitor. You can slow down the automatic polling by changing the polling time.

This is the most interesting and perhaps useful of all the classes. With this you can interactively extract precise values of the Archive storage on the Weather Link.. With the wrap around memory feature it is even conceivable that you never need reset the archive. You simply use this on a daily launch to extract the last days records for storage, analysis or summaries. By providing this simple system you program the desired information retrieval to anything you like.

Conclusions

I hope you find the program useful. I certainly have and it has made my transition to the USB based Macs easier to stomach now that communications is centered on the CommToolBox. Further, with the AppleScript controls and data structures you can get information and organize it the way you want (even set up weather servers) and not how others wanted it set up.

Some elements of this work are still a work in progress but there have been no stability issues as yet. Because it is a hobby (although I am a professional developer) I don't currently support any more equipment than I own. Maybe some day I'll get different equipment.

I am open to suggestions but, as I have indicated, I'll get to them as I can. I will certainly try to correct any real problems as soon as possible.

Known Issues:

- ▫ Doesn't save archive data to file. This is in progress as I only just discovered the new database format from Davis on the Web. With AppleScripting this may not even be necessary.
- ▫ Occasionally dialog boxes like: `Exception thrown...` appear. These are generic error messages which I have been slowly eliminating in favor of more 'user friendly' boxes. When I see them I change the code to correct the condition or inform the user more completely. If you find one copy down exactly what it says and I'll try to eliminate it.
- ▫ Polling is still too slow for some functions. Particularly the archive window. I should be able to retrieve the data slightly faster than I do (block mode transfer from the Weather Link). I can do it but it ties the machine up too much. I need to experiment with the best balance between block size and number of iterations. I am working on this.
- ▫ Multithreading the polling interface. I have recently (re)discovered the wonders of threads through my work in Java and want to learn more about the LThread classes of PowerPlant. This wont speed up the communication with the weather monitor but will make the interface more responsive.