Ron Chambers, N5QQ

A "Phidgets-Based" Antenna Control System

It started 10 years ago when Brent, W5WW, and I decided to throw in together on a contest station. It was Jack Lemmon and Walter Matthau all over again. Brent was an avid phone op who wouldn't go near a CW paddle, and I a confirmed brass pounder who wouldn't touch a microphone. The basics were to have two *independent* operating positions — his and mine — with *lots* of antennas. At first the station was not designed to be a multi-two or multi-multi, but the thought was there. KC5FU was born.

We chose to scatter "band towers" around the 11 acre aluminum ranch; they're known to grow easily in North Texas. There are four towers, each dedicated to one of the bands 10 through 40 meters, plus a smaller "multiplier" or "power-split" tower. Three high stacks are in place for 10 and 15 and two high stacks for 20 and 40. Combined on the multiplier tower are a homebrew 40 meter Moxon and a Cushcraft X7. Wires for 80 and 160 wires strung from the taller 40 and 15 meter towers (an overview of the station's RF scheme and additional images are available on the *NCJ* Web site, **www.ncjweb.com/bonus.php**).

This makes 14 antennas, six bands and four StackMatches to select, combine, power-split and connect to either of two radios at either of two operating positions. And don't forget, every StackMatch offers seven possible choices. Of course we wouldn't consider simplifying into 10/15/20 at Station 1 and 40/80/160 at Station 2. That would be too simple. We wanted *all* antenna combinations on *all* bands at *all* radios. In short — a switching nightmare!

To wit, Brent would operate a phone contest, followed by my wanting to play CW a few weeks later. All of this meant moving the StackMatch and SixPak controllers and rotor controls and feed lines from "his" desk and to "mine." Wires were everywhere.

So, how to solve this nightmare? I looked at what others had done, knowing we weren't alone. I played with relays, I played with switches, I played with band decoders, I played with *BASIC* stamp controllers and more. Then one slow S&P night, Brent threw out the idea to look at Phidgets (**www.phidgets.com**), a USB-connected controller capable of being programmed in *Visual Basic*, *.NET*, *C++*, *Delphi*, *Labview*, *Java*, *COBOL 33*, *Fortran* and other languages. I was hooked. Brent, a software guru, turned into my *VB6* Elmer. He tossed a few *VB6* examples my way, gave me hints and kinks, handed me a stack of *Visual Basic 6* books and off I went. This was in 2006 I believe.

It didn't take me long to realize that the coming up with the necessary switching logic was the easy part. The hard part was the 64 to 128 hardware I/Os needed to get the job done and finding hardware to do it. I'll never forget a comment from Tim, K5RA: "As an engineer it's easy to sit at a desk and dream up something, but then can you actually get it built? You have to engage RAPD, readily available product design. Sometimes, you just have to work backwards, find a product that's readily available, at a reasonable price and then make your design fit it." The Phidgets answered that call. They were modular, networkable, and compact, had various forms of I/O, were not too expensive, offered a choice of operating systems and languages and were USB-connected.

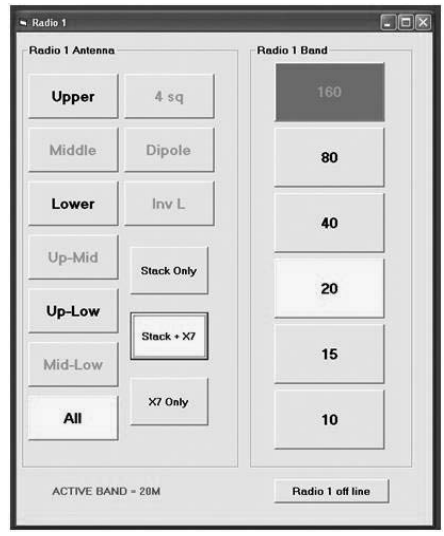

Figure 1 — The 20 meter screen

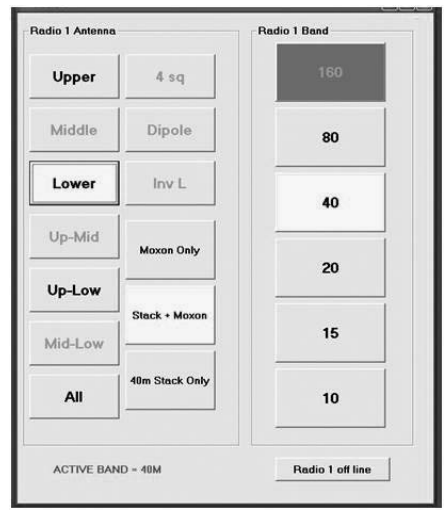

Figure 2 — The 40 meter screen

We developed a plan. We wanted the operator interface to be a typical Microsoft *Windows* screen that could be placed on any PC, now or in the future, throughout the station over our LAN. The operator would have access to *all* antennas, period.

System Overview

After creating the control program in the language of choice, we installed the executable version on each operating position's PC. When executed, it brings up a window showing the band and antenna choices (see Figures 1 and 2). If one position is SO2R, then two windows would appear on the same screen.

The operator can manually select the band by mouse click or by using an independent rotary encoder. Or, taking advantage of automation, the operator can simply accept band information directly from the radio into the control program. The program decodes the 4 bit info into discrete outputs, similar to any of the common band decoders on the market. In either case, once a band is selected, the antennas available to that band are presented onscreen, including the power split option to add an X7 in parallel with the stack in use, with any part of the stack or use it by itself (Figure 1 shows a typical 20 meter screen with the X7 as an optional antenna). A yellow highlight indicates the selection, while a grayed-out selection indicates the band is in use at another radio and not available to the operator. It is locked out, period.

Similarly Figure 2 shows a typical 40 meter screen with the Moxon shown as a power split option. Note that our 40 meter stack is only two high, so some of the three high stack options are grayed out and unavailable. If another station is using the Moxon or X7, the option to use either of those is locked out from all other operator positions.

Antenna Choices

Let's go through one band as an example of what is available to the operator. The 10 meter tower is a three-high 5 element OWA stack with antennas at 35, 70 and 100 feet. The X7 is on a separate tower at 60 feet. All are rotatable. The operator has these available choices:

- 10 meter stack (all three)
- 10 meter upper
- 10 meter middle
- 10 meter lower
- 10 meter upper and lower
- 10 meter upper and middle
- 10 meter middle and lower

- 10 meter stack (all three) + X7
- 10 meter upper + X7
- 10 meter middle + X7
- 10 meter lower + X7
- 10 meter upper and lower + X7
- 10 meter upper and middle + X7
- 10 meter middle and lower + X7
- X7 alone

Software

The program was written in *Visual Basic 6* only because I was a homeschooled rookie programmer with limited experience in *BASIC*. *NET* or *C++* or any of the more current languages would be more appropriate in the long run as *VB6* is being shoved aside. For now, all this runs on *Windows XP* machines. Table 1 shows a bit of the *VB6* code.

This is a 100 percent custom piece of software, written to match our specific station requirements. This does not preclude its being modified to fit any other station's design, however. It just has to be done within the code itself. A more sophisticated programmer at the helm could add set-up screens, so the user could reassign outputs, add outputs, change antenna configurations, change colors and sizing of the windows, and so forth. I also assume data exchange or communication with logging programs, rotor controls, RS-232, RS-485 and wireless communication all are possible.

Hardware

The Phidget hardware is straightforward and easy to use. The Phidget boards in this system are nothing but various configurations of input/output devices that communicate over USB cabling. One board has *sink* outputs, another *source* outputs, and a third makes use of 10 A mechanical relay contacts. So, with a little planning and coordination you can control most popular antenna equipment that requires +12 V dc signals, such as Array Solutions' Stack Match and SixPak — or with a *sink* output board you can directly control Top Ten Devices A/B switches or six-way relay boxes.

After experimenting with the various boards I settled on the eight-relay board, or, as the Phidget people call it, the Model 1017 PhidgetInterfaceKit 0/0/8. With that board I can set it up for +12 V dc outputs, +5 V dc outputs or *sink* outputs, all dependent on which piece of equipment I need to control, now or in the future. Thus, a single board handles software communication and interfaces to higher load and voltage requirements without needing to have another interface relay board. Future plans call for using the LED64, which has 64 +5 V dc outputs.

Either way, in the end the goal is to get all of the Phidget boards you're using connected to your PCs using standard USB cables. There is no magic here; you can plug them directly to USB ports on a desktop, you can incorporate 4:1 or 6:1 USB hubs, and one Phidget board even has a built-in 4:1 hub, which turns out to

Table 1
Sample *VB6* code

```
==================== STACK + X7 ==============================

    If X7(1).Value = True Then      'Selects "Stack + X7"(modeled as LOWER_MIDDLE)
    '===========

      For i = 0 To 2
        X7(i).BackColor = &H8000000F      'reset all graphic backgrounds to same for this radio
      Next

      X7(1).BackColor = &HFFFF&           'Now highlight only the selection

      '====== reset the Stack Match buttons =============
      For i = 0 To 6
        cmd1a(i).Enabled = True           'reset ALL the stack selection buttons
      Next

      If BandR1a = 20 Then                'hide the 20m stack selections n/a for 2 high
       cmd1a(1).Enabled = False
       cmd1a(3).Enabled = False
       cmd1a(5).Enabled = False
      End If

      cmd1a(last_antenna_A) = 1

      Dict2.Add "X7_Station_1", 1         'broadcast status of X7 for this radio
      Exit Sub

    End If

================================================
```

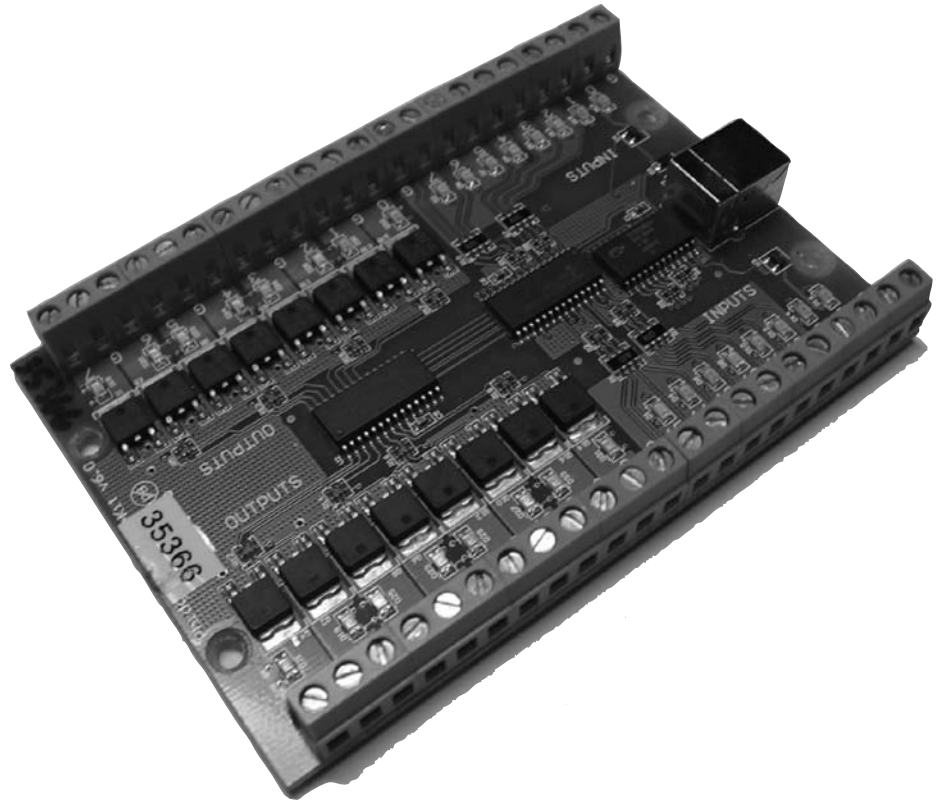

Figure 3 — The model 1012 PhidgetInterfaceKit 0/16/16

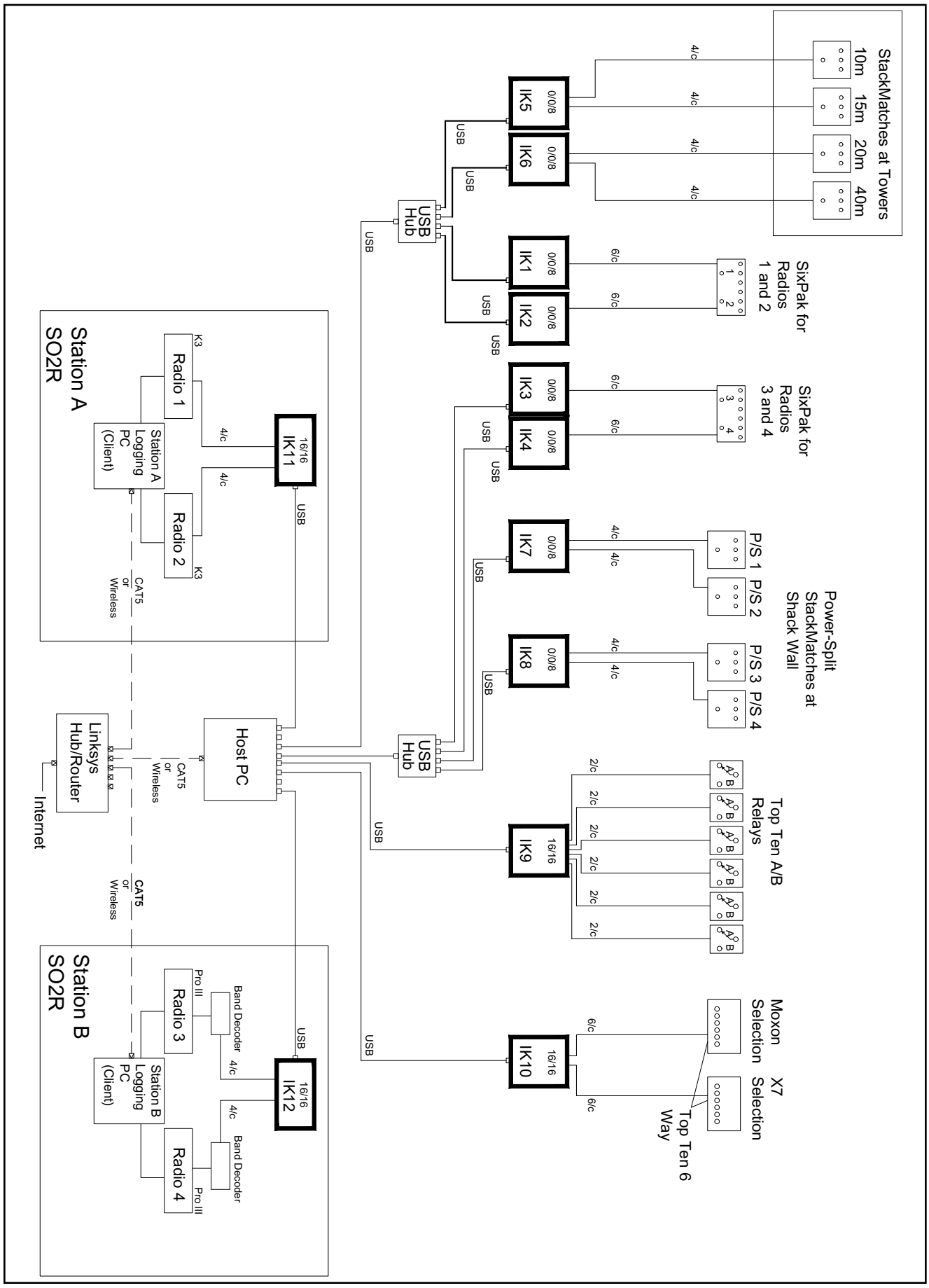

Figure 4 — A simplified control schematic

be quite convenient. As long as your PCs are networked, it doesn't matter where any board is connected. They can easily communicate over the network. The output wiring is small gauge (#18 to #24) control wiring from each Phidget board's terminal strips to the equipment to be controlled.

In the simplest mode, such as for one SO1R or SO2R station, one PC handles all normal logging chores, radio interfaces and Phidget antenna control software. With all the Phidget board USB cables connected directly to the PC, no network is necessary.

In our system, a dedicated *host* PC connects the Phidget boards via USB cables. The host then connects to our LAN, which includes two other PCs — one at each operating position. I haven't detected any latency in the system, so we have no reason to expect that the system will not be capable of handling multiple stations. For you software types, the Phidgets networking makes use of *key-value pairs* and *dictionaries*, which enable all client programs to communicate with one another.

At present the operator can control the switching by selecting the antenna of choice with a mouse click or by using the USB-connected Phidget model 1052 rotary encoder. I think the ultimate solution would be a touch screen, but I had neither the time nor the expertise to incorporate that element (*hint, hint*) just yet.

Summary

Solutions to antenna control vary all over the map. Each has to do with priorities, what you want to accomplish and what you have to work with, including time and money. I found the Phidgets easy to work with, and they certainly have been a lot of fun. They offer the flexibility of an off-the-shelf product that has helped me get where I wanted to go.

A schematic of the RF paths at KC5FU and a photo of the Phidget control system with the covers removed are on the *NCJ* Web site, **www.ncjweb.com\bonus.php**.

NCJ