

## A Modified Spinapi USB library for OSX

Included here you will find modified spinapi source code (incorporating changes made up to September 4, 2007), a functioning dynamic library, and an Xcode Tools project file. These instructions describe how to take the included, modified spinapi source code, and compile and link it into a dynamic library (dylib) under Apple OSX for use with a RadioProcessor USB board. After this is done, you will have the following capabilities:

- You should be able to compile, link, and use any pulse programs that use USB Radioprocessor boards. The only functions that I know definitely behave differently are *pb\_init* and *pb\_select\_board* (see below), but this should have no effect in most cases.
- You can write programs to control up to 8 boards; this can be increased in the file *spinapi.h* if desired – note that the default spinapi code comes with the maximum number of boards set to 32.

A couple of things to note:

- Because I built the library up over time based on functions I needed, the source files don't always have the same name as the spinapi source files. This could be confusing, please feel free to contact me with questions.
- The primary file that interfaces with USB under OSX here is *driver\_usb\_osx.c*. However, there are differences that show up in other source files; for example, the original spinapi uses the watchdog timer to tell if data transfer has timed out. Under OSX there are “timeout” versions of USB read (and write) statements that can be used instead, so these are used here rather than watchdog.
- OSX interfaces with devices differently than most unix os's, so these files (particularly *driver\_usb\_osx.c*) will not be helpful to develop versions of spinapi for other unix flavors.
- Some functions don't work exactly the same as the original spinapi code; for example, *pb\_init* initializes all of the boards it finds on the USB bus at once, you can then use *pb\_select\_board* to select any board at any time; note that initializing the boards involves for the most part initializing USB communication and finding the board capabilities – if a board is initialized it does not lose whatever was previously programmed. Also, the *debug* function prints to the terminal rather than to a file (i.e. if debug is set to 1, debugging info will be displayed).
- When *pb\_init()* is executed, the boards are ordered by their USB location id in the case of multiple boards. This gives them a software-independent ordering that I have found generally follows the port number when the boards are all connected through a USB hub (e.g. if two boards are connected, one to port 0 and one to port 3, port 0 will be board 0 and port 3 will be board 1).

The installation instructions that follow assume you have Xcode Tools installed on your Mac. This is included in the main installation disk for the operating system in case you don't have it already installed – it can also be downloaded from the Apple web site

<http://developer.apple.com>. It is also assumed you are familiar with entering unix commands from the Terminal, and that you are also familiar with unix file permissions.

1. Expand the included tar file. You should then have a directory **spinapi\_osx\_lib0907**, inside of which you will find a subdirectory **Build/Release**. The dynamic library file included there,

**libspinapi\_osx\_lib.dylib**, is a version of the modified spinapi library **for Intel processors only**. This library file does not need to be re-compiled and re-linked, but this could easily be done in Xcode if needed. I intended originally to make a build for PPC Macs as well, but ran into a few difficulties: the Radioprocessor board appears to only work with USB 2, and all of the PPC Macs I was able to borrow only had USB 1.1. If it were desired to make a PPC Mac library, I think it should be possible, but one needs to watch out for byte order effects: the byte ordering is reversed on PPC vs Intel processors. This will definitely have an effect on how the data are written (functions such as *pb\_write\_ascii*, etc), and may also affect the way frequencies and phases are loaded onto the board (this is because these functions load large integers onto the board one byte at a time) – unfortunately I didn't have a chance to check. If you do want to try compiling a PPC version, please feel free to contact me.

2. Before attempting to use the library, put the dylib file into a directory that is easily accessed by the compiler – I recommend /usr/local/lib; e.g. **cp libspinapi\_osx\_lib.dylib /usr/local/lib**. You can also create a symbolic link.

Now, you just need to compile your programs; note that *spinapi.h* needs to be in the same directory as the program you are compiling (or in a shared directory such as /usr/local/include) in order build the programs as written in the example code. For example, to build the excitation test program included in the RadioProcessor directory, for example:

```
cc excite_test.c -lspinapi_osx_lib -o excite_test
```

That is all that is required.

*Tom Pratum, Department of Chemistry, Western Washington University,  
September 30, 2007, pratum@chem.wvu.edu*