

Using SR Libraries with C++

This note covers calling Symmetric Research core library software from a C++ program. The core library software for our products is supplied in 32 bit DLL format so all users can have high level software access to our products. In order to use these libraries correctly from your C++ code, there are a few things you must do. The concepts are the same for using the SR libraries with g++ under Linux, although the details vary a little as discussed at the end of this note.

The examples given in the following discussion use the files and directories appropriate for the SR PAR4CH 24 bit, 4 channel A/D. Be sure to substitute the correct names for the product you are actually using.

First, you must add the extern "C" statement as shown below whenever you include the library header in your C++ source code. This lets the compiler know that the A/D library functions are written in standard C and prevents it from adding C++ specific "decoration" to the function names.

```
extern "C" {
#include "par4ch.h"
}
```

Second, you must use the DLL's "import library" when linking. This allows the linker to correctly resolve references to the functions inside the DLL. When linking from the command line, just add the .lib import library file to the end of the link statement. The link statement would look something like this:

```
link yourcode.obj \sr\par4ch\lib\par4ch.lib
```

Finally, remember that for your program to run successfully, it must be able to find the DLL at run time. The simplest way to do that is to add the location of the DLL file to your path as shown below. You will have to do this every time you reboot the computer, unless you add that statement to your autoexec.bat file.

```
set path=%path%;\sr\par4ch\lib
```

You may want to look at the scope software for an example of using the A/D library functions from C++ code. It is written in C++ and uses the Microsoft Foundation Classes in a manner more in keeping with the MS Visual C++ integrated development environment.

Under Linux, the terminology is a little different, but the ideas are all the same. The SR core software for Linux is supplied in .so shared library rather than DLL format. First, you must use the same extern "C" statement shown above to avoid name mangling. Next, you must include the library when compiling and linking your program. Finally, you must add the location of the shared

library to your library path so your program can find it at run time. The compile/link statement and the library path statement would look something like this:

```
g++ yourcode.cpp -l4ch  
LD_LIBRARY_PATH=/usr/local/sr/par4ch/lib:$LD_LIBRARY_PATH ; \  
export LD_LIBRARY_PATH
```