CINs are out, Call Library Function Nodes are in!

Submitted by LabJack Support on Tue, 09/07/2010 - 12:07

Two months ago we documented how to create and use Code Interface Nodes (CINs) for LabVIEW on Mac OS X and Linux. It’s a great way to call C code that uses the Exodriver, but we discovered that LabVIEW 2010 does not support CINs. In this post, we describe how we migrated our CIN work to Call Library Function nodes.

We created a library (a .framework on Mac OS X and a .so on Linux), which is now the preferred way to access LabJack devices from LabVIEW on Mac OS X and Linux. The result is really better than the old, CIN-based way. You can find the finished product on the LabVIEW on Mac OS X and Linux support page.

Here’s how we built the library.

**Make a Mac OS X Framework**

1. Start a new project using Xcode, and select “Cocoa Framework” from the “Framework & Library Section”

2. After you’ve given your project a name, right click on the project, and select “Get Info”

3. In the section labeled “Architectures”, find the sub-section called Architectures and select “Other…” from the pull down menu. Enter “i386” as the new value.

4. Make sure the “Base SDK” subsection has your OS selected. In my case, “Mac OS X 10.6”.

5. Changed the “Valid Architectures” to “i386”
6. In the section “Linking”, find the subsection “Other Linker Flags” and set it to “-llabjackusb -llabviewcin -llvexports -lc”

7. In the section “Search Paths”, change the “Header Search Paths” sub-section to have “/usr/local/include” and the path to the cintools directory. In our case, “/Volumes/Macintosh\ HD/Applications/National\ Instruments/LabVIEW\ 2010/cintools”

8. Also in the “Search Paths” section, change the “Library Search Paths” to have “/usr/local/lib” and the path to the Mach-O directory inside the cintools directory. In our case, “/Volumes/Macintosh\ HD/Applications/National\ Instruments/LabVIEW\ 2010/cintools/Mach-O”

9. You can close that window, and add your .h/.c files.

Do NOT to use the .c files that LabVIEW offers to build for you. They use typedefs from LabVIEW’s extcode.h. LabVIEW’s “Import Shared Library” tool will not work, or even crash, on libraries that use extcode.h. Stick with C’s basic types: int, double, etc.

The other piece of advice we can give can be: Place all your includes in the .c and keep only your function prototypes in the .h. LabVIEW’s “Import Shared Library” tool will try to create functions that are defined in your includes, which is annoying.

**Make a Linux shared object (.so)**

Use this Makefile:

```bash
# Makefile to build a .so for LabVIEW

VERSION = 1.0

LIBFLAGS = -llabjackusb -lcin -llv -lc
CFLAGS += -m32 -I/usr/local/natinst/LabVIEW-2010/cintools
LIBTOOLFLAGS = -L/usr/local/natinst/LabVIEW-2010/cintools
LABJACKHEADERS = u6.o

%.o: %.c
    $(CC) $(CFLAGS) -c $<

TARGET = LabJackLabviewLibrary.so

all: $(TARGET)

$(TARGET): LabJackLabviewLibrary.o $(LABJACKHEADERS)
    $(CC) $(LIBTOOLFLAGS) -shared -Wl,-soname,LabJackLabviewLibrary.so -o $(TARGET) LabJackLabviewLibrary.o $(LABJACKHEADERS)

clean:
    rm -f $(TARGET) *.o ~
```
Be sure to check that your installation of LabVIEW is installed at “/usr/local/natinst/LabVIEW-2010”, or change CFLAGS and LIBTOOLFLAGS. Change LABJACKHEADERS if you use u3.h or ue9.h.

**How to call a Call Library Function node**

We’ve abstracted our calls to these nodes into sub-VIs, but here’s what one looks like:

```c
int readFloatingPointRegister(int deviceHandle, int Register, float* Value);
```