syntouch® LLC

SynTouch® JACO Integration Manual

Updated: July 11, 2014, V2

Authors: Gary Lin & Raymond Peck

Table of Contents

1 CAN-USB Hardware

1.1 ST-JACO board

20-pin flexible circuit connector, pin map:

Pin 1~3	+24V
Pin 4~8	Not Connection
Pin 9~16	GND
Pin 17~18	Not Connection
Pin 19	CANH
Pin 20	CANL

1000 Vrms isolation on DC regulation

5000 Vrms isolation on CANH and CANL signal lines

1.2 Customer Cable

Modified from original JACO cable, pin map:

Pin 2	CANL
Pin 3	GND
Pin 7	CANH

A 120 Ohm resistor was connected between CANH and CANL according to CAN Bus Protocol.

1.3 Peak CAN USB

Product information: http://www.peak-system.com/PCAN-USB.199.0.html?&L=1

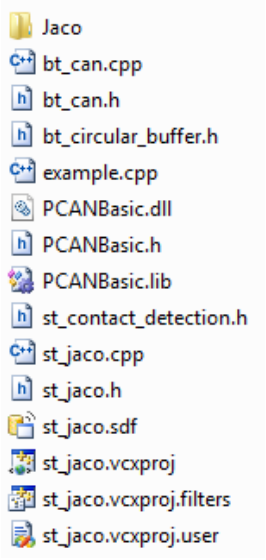
2 CAN-USB Software

2.1 Pre-request

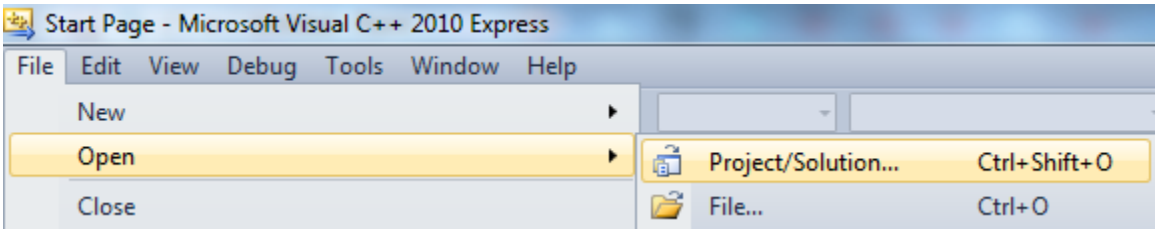
- **Peak CAN Driver**
  Please download and install from
  http://www.peak-system.com/fileadmin/media/files/usb.zip
  Or from product information website
  http://www.peak-system.com/PCAN-USB.199.0.html?&L=1
- **JACO Driver**
  Please download JACO Research Edition SDK from
  http://kinovarobotics.com/wp-content/uploads/2014/04/JACO_RESEARCH.zip
  Or product support website
  http://kinovarobotics.com/support/
- **Microsoft Visual C++ 2010**
- **Boost C++ Library**
  The BioTac Demo software needs boost C++ Library. Install the boost C++ library from
  http://www.boost.org/ (Version: 1.55.0)

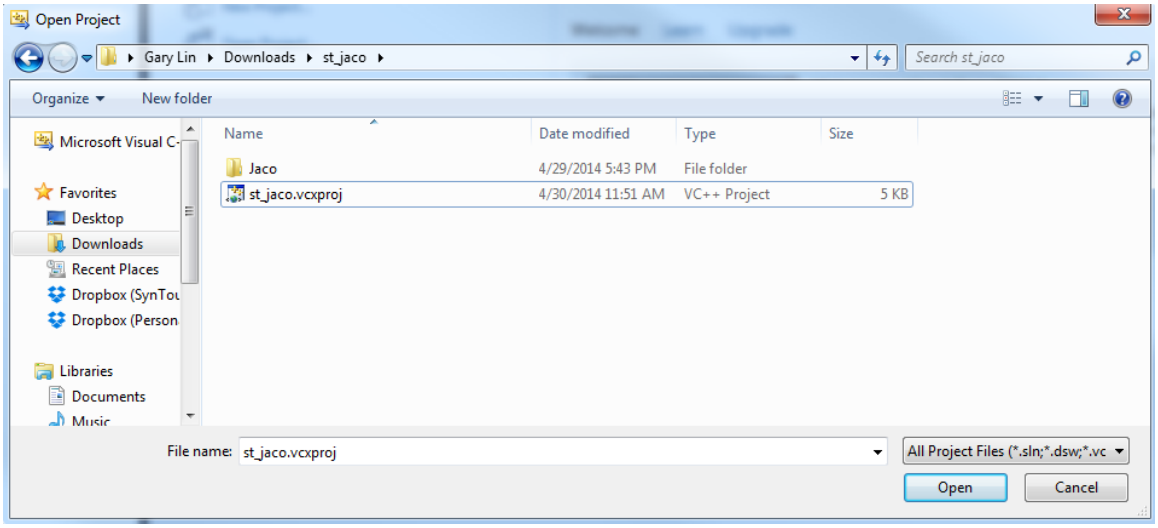
2.2 Installation Guide

1. Download the file “st_jaco.zip” to desired location (it will be Downloads folder in this document)
2. Unzip the file “st_jaco.zip” and in the folder we will get

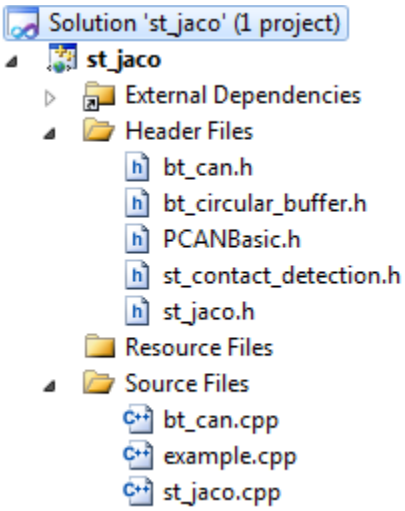


3. Open Microsoft Visual C++ and open the object file “st_jaco.vcxproj”





4. We will see following files in the project



5. Add boost library path into the project environment (see Appendix)
6. Compile and run example demo
7. Use joystick to move the hand to proper position and hit key “c” on keyboard to close the hand.

NOTE: Please do NOT press any key while the arm is controlled by the joystick. According to JACO API documentation, the commands from joystick have higher priority than those sending from PC. So the example API program cannot control the arm when it is manipulated by joystick.

8. Hit key “c” again to toggle the hand open/closing function to open the hand.
9. Hit key “r” to reset the position memory.
10. Use joystick to move the hand position to next desired position and key “c” to grasp another object.

2.3 Software Description

Key “c” is used to control finger open/close. The program will remember the position of the arm while key “c” is been pressed.
WARNING: After key “c” being entered, if the arm position has been modified by joystick
Key “q” save the circular buffer to the same folder “st_jaco” and quits the program
Key “r” resets the position memory

2.4 Functions

In example.cpp

2.4.1 Main function (main)

```
int main(  int argc,
           const char* argv[] );
```

Details

- In this function, it creates two threads for running the bt_main and stjaco::main programs in parallel.

- It listens to any new data in the shared memory `pdc_shared_data` and executes the contact detection algorithm.

- It listens to any value change from the keyboard. If the user presses any key, it passes the pressed key information to two shared memories (`biotac_keyboard` and `jaco_keyboard`).

In bt_can.cpp

2.4.2 Save data (bt_save_buffer_data)

```
void bt_save_buffer_data(   const char *file_name,
                            bt_circular_buffer<bt_data>*data_save,
                            int num_samples);
```

Save BioTac data which is in the circular buffer to file, including batch, frame, and channel ids, raw values, parity check, etc.

Arguments

`file_name`	pointer of the string of file name such as “output.txt”
`data_save`	pointer to access the circular buffer
`num_samples`	numbers of sampling data saved into the file

Return Value

This function does not return any value.

Details

This function saves data in a text file whose name is `file_name`. By default, a saved data format is as follows:

```
time, batch_index, frame_index, channel_id, value[0],
bt_parity[0], value[1], bt_parity[1], value[2],
bt_parity[2]
```

Note:

`time`: the relative time to the very first sample after the power is on

`batch_index`: in this project a batch contains one frame, so the batch index is equal to the frame index

`frame_index`: in this project a frame contains one sample, so the frame index increases one for every sample

`channel_id`: is saved as integer (0-3, 15-35) by default

`value[0], value[1], value[2]`: BioTac data from three fingers

`bt_parity[0], bt_parity[1], bt_parity[2]`: The parity check for each BioTac data. Logic 0 means good parity check and data is valid. Logic 1 means bad parity and data is invalid.

2.4.3 BioTac main function (bt_main)

```
int bt_main(void);
```

Arguments

None

Return Value

Return 0 if it is successfully terminated
Return 1 if there is any error related to Peak CAN-USB

Details

- This is the main function to handle all the functions that initialize the settings related to Peak USB-CAN device and listen to BioTac samples and collect data in a circular buffer.

- In this bt_main function, it pushes the sensing data (by default, PDC channel) to a shared memory (`pdc_shared_data`) for contact detection purpose. The following is the code:

```
// Example of using PDC data for contact detection
if(data.channel_id == CD_CHANNEL)
{
	// Lock the data first to prevent data accessing conflicting
	boost::unique_lock<boost::mutex> pdc_lock(pdc_shared_data_mutex);
	pdc_shared_data.push(data);
}
```

  The sensing channel can be modified through the variable `CD_CHANNEL` in st_contact_detection.h file. See the Appendix for all the possible channels.

- In this bt_main function, it listens to any changes on the keyboard. If it detects the key “q” pressed by user, it will save all the collected data in circular buffer to the file (please refer to the save data function), terminates the Peak CAN-USB connection, and exit from the bt_main function.

In st_jaco.cpp

2.4.4 Initialize JACO ARM and load its API (initialization)

```
bool stjaco::initialization(void);
```

Loads Kinova API and initialize function pointers.

Arguments

None

Return Value

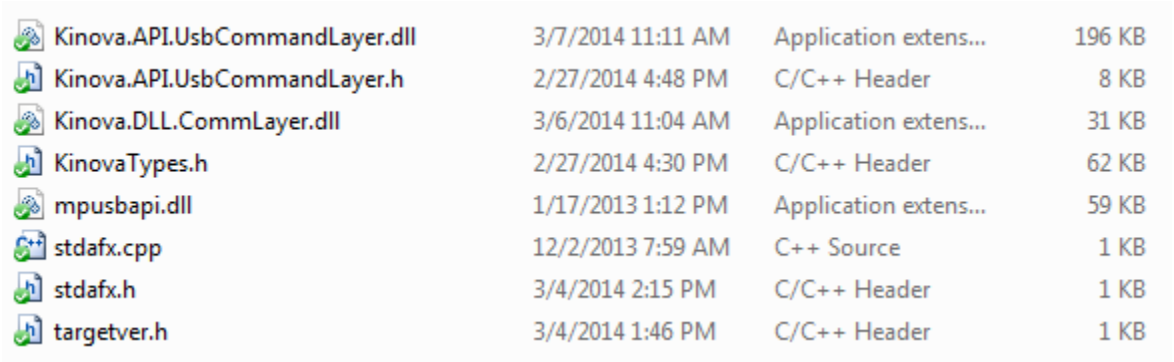
Return true (1) if library loaded and pointers created without any problem

Return false (0) if there is any error loading the library or creating function pointers

Details

For detail information for the Kinova API, please refer to Kinova programming guide.

Please make sure the following files are under the subfolder “st_jaco\Jaco”



2.4.5 Go to home position (homePosition)

```
void stjaco::homePosition(void);
```

Change the robot arm to position control and make it to go back to home position.

Arguments

None

Return Value

This function does not return any value.

2.4.6 Command finger position (fingerPosition)

```
void stjaco::fingerPosition(int finger_status);
```

Setup the goal position of the finger depending on the input argument.

Arguments

`finger_status` indicate the goal position of the finger is open or close. 0: finger open. 1: finger close

Return Value

This function does not return any value.

2.4.7 Freeze the arm motion (stopMotion)

```
void stjaco::stopMotion(void);
```

Stop the motion of the arm.

Arguments

None

Return Value

This function does not return any value.

Details

This function stops the motion of the arm, waits 200 millisecond for arm stabilization, and records the arm Cartesian position.

2.4.8 SynTouch JACO main function (stjaco::main)

```
int stjaco::main(void);
```

Initialize the JACO Arm related program and control its movement depending on the joystick and keyboard input.

Arguments

None

Details

- This function calls the initialization function for JACO Arm and load its API

- It checks the contact detection shared memory (cd_shared_data) and stops three finger movements individually if a contact event has happened to the finger.

- It monitors any change on the key board:
  Key “c” is used to control finger open/close. The program will remember the position of the arm while key “c” is been pressed.
  Key “r” resets the position memory
  Key “q” stops the motion of the arm, close the Kinova API, and exit the main function.

  WARNING: After key “c” is entered at time 1, if the arm position is modified by joystick and “c” is pressed again, the arm position will move back to the original position where “c” was pressed at time 1.

In bt_circular_buffer.h

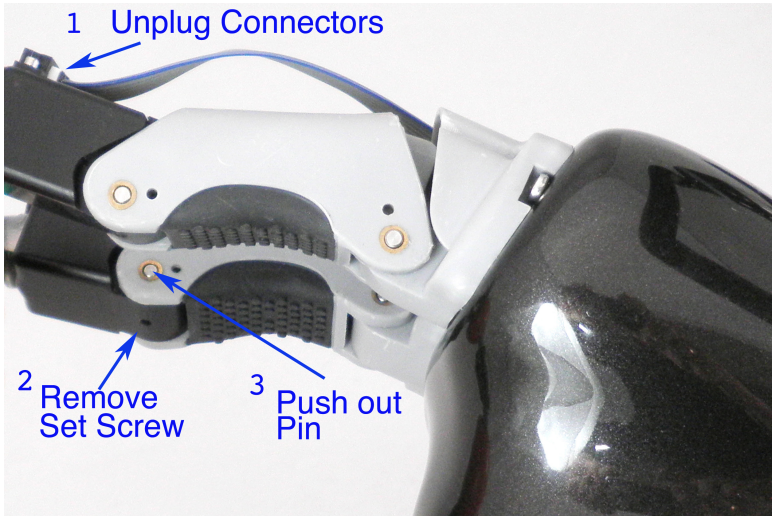
Please do not modify any code here except `BUFFER_SIZE` . The constant `BUFFER_SIZE` is used to setup the size of the circular buffer. By default, `BUFFER_SIZE = 132000` and can store about forty (40) seconds of data. If you prefer a bigger size of circular buffer, you can increase its size, there are about 3500 samples per second.

3 Removal of JACO-BioTac Mechanical Adapter

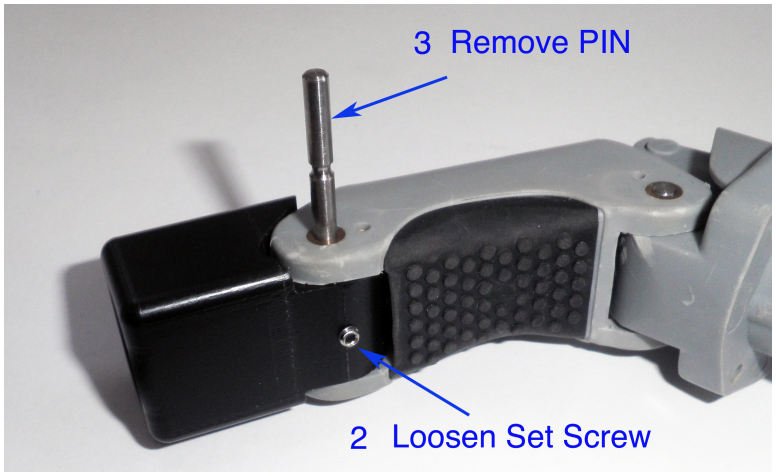
This section describes the steps needed to remove the JACO-BioTac Mechanical Adapter

3.1 Unplug the Connectors

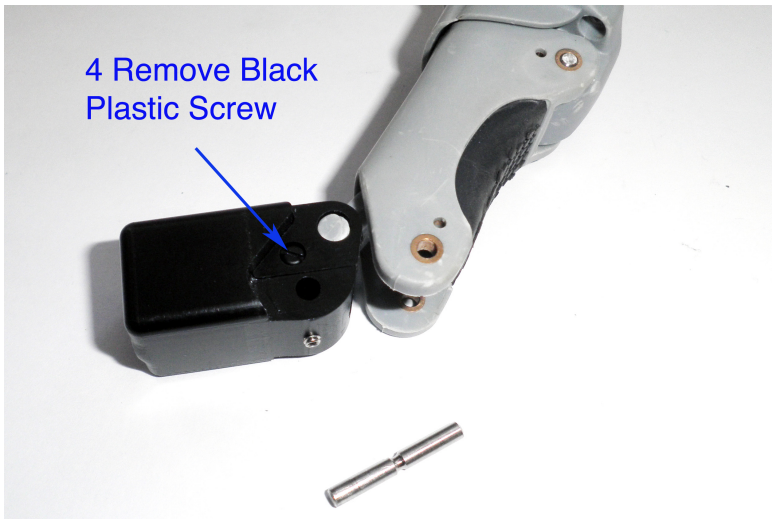
➢ *Warning: unplug connectors carefully to avoid damage to the BioTac*



3.2 Remove set screw and push pin out.

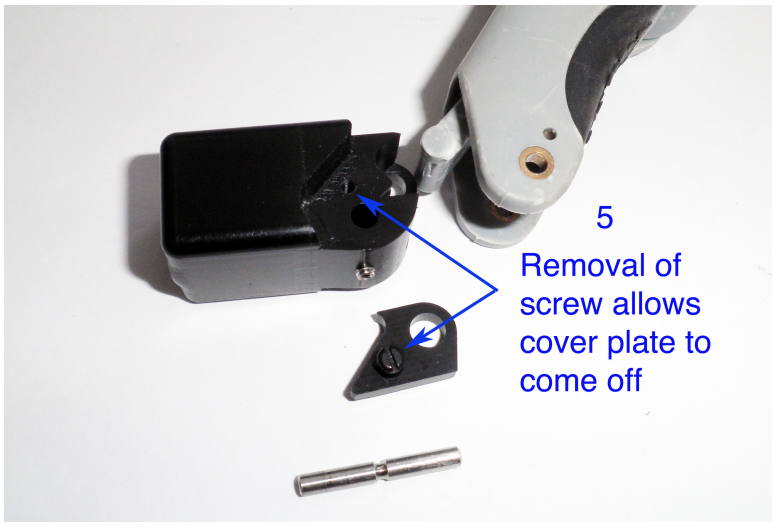


3.3 Remove black plastic screw



3.4 Remove adapter

Once screw is removed, cover plate will come off to allow adapter to separate



Appendix A: Troubleshooting Problems

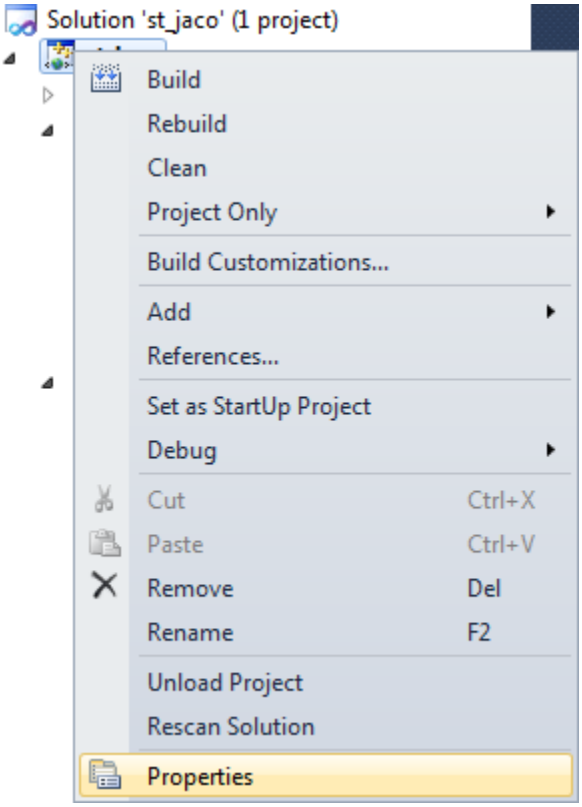
Error #1

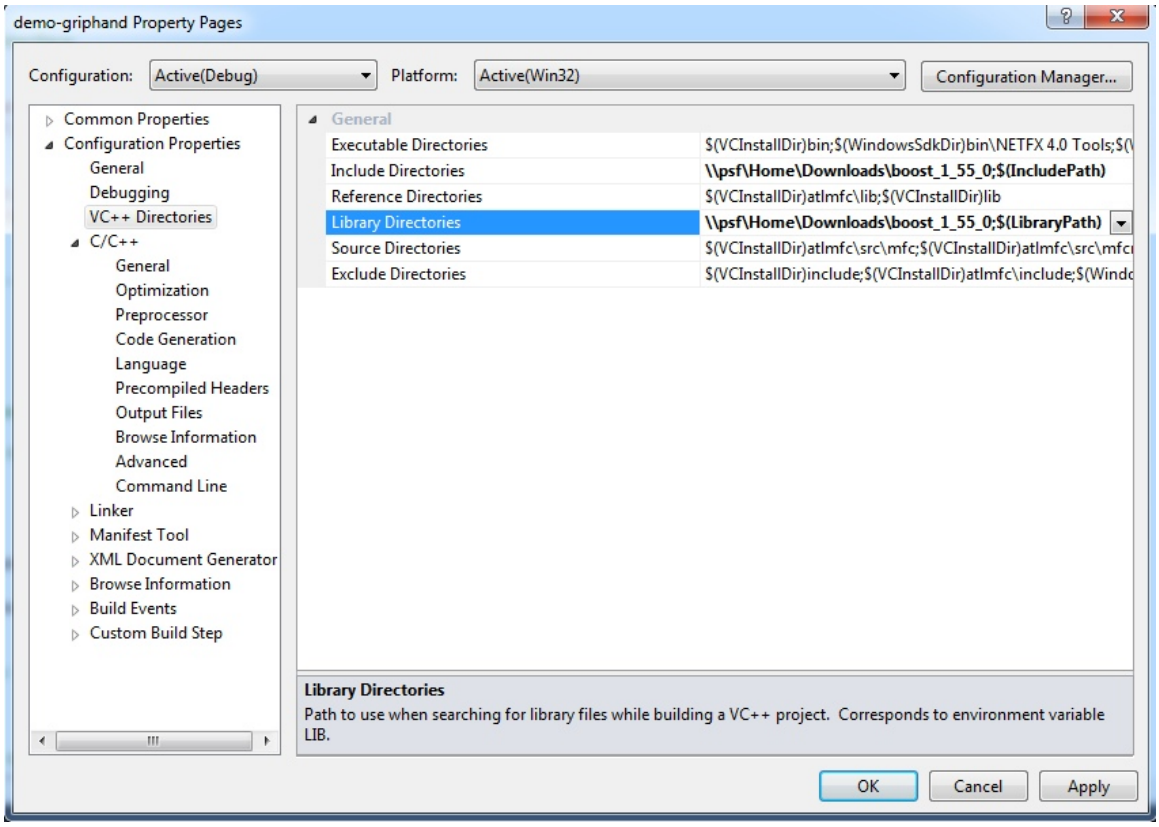
Compiling error:

fatal error C1083: Cannot open include file: 'boost/thread.hpp': No such file or directory

Solution:

Include the library path (\Downloads\boost_1_55_0) into the “VC++ Directories” under “project property”





Error #2

Compiling error:

fatal error LNK1104: cannot open file 'libboost_thread-vc100-mt-gd-1_55.lib'

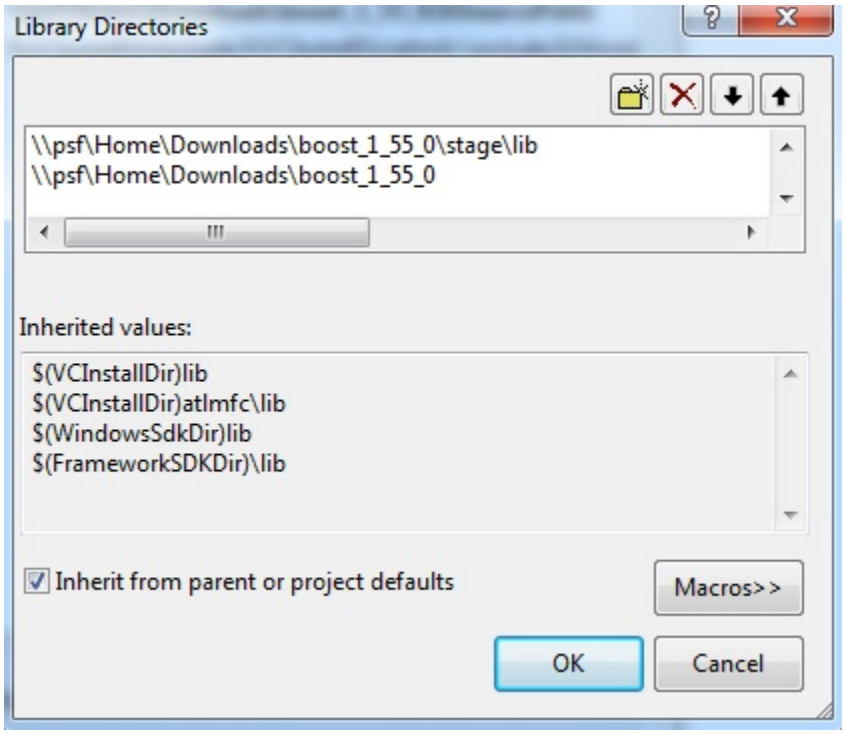
Solution:

Chapter 5.1 in http://www.boost.org/doc/libs/1_55_0/more/getting_started/windows.html

If you wish to build from source with Visual C++, you can use a simple build procedure described in this section. Open the command prompt and change your current directory to the Boost root directory. Then, type the following commands:

```
bootstrap
.\b2
```

The first command prepares the Boost.Build system for use. The second command invokes Boost.Build to build the separately-compiled Boost libraries. Please consult the Boost.Build documentation for a list of allowed options. This will compile to generate the **stage** folder



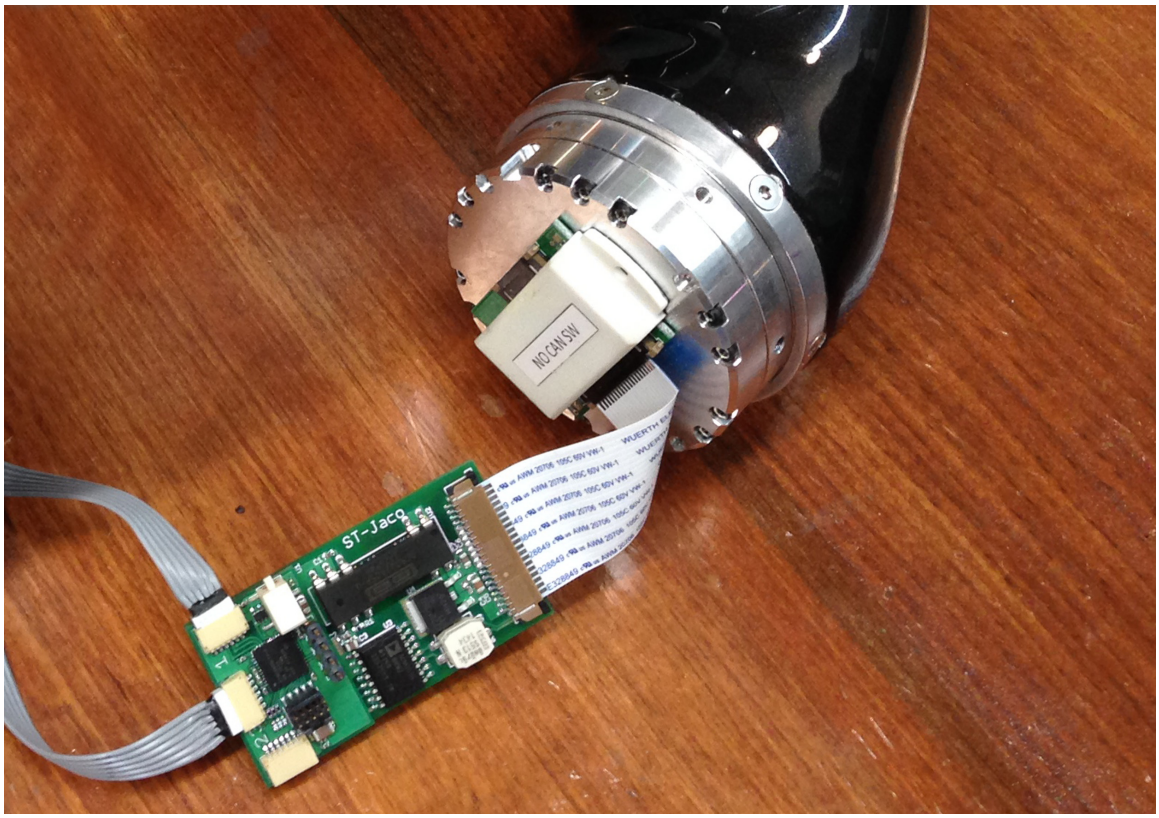
In *st_contact_detection.h* file, the magic number CD_CHANNEL can be modified to the following constant number:

```
BT_PAC_SAMPLING
BT_PDC_SAMPLING
BT_TAC_SAMPLING
BT_TDC_SAMPLING
BT_E01_SAMPLING
BT_E02_SAMPLING
BT_E03_SAMPLING
BT_E04_SAMPLING
BT_E05_SAMPLING
BT_E06_SAMPLING
BT_E07_SAMPLING
BT_E08_SAMPLING
BT_E09_SAMPLING
BT_E10_SAMPLING
BT_E11_SAMPLING
BT_E12_SAMPLING
BT_E13_SAMPLING
BT_E14_SAMPLING
BT_E15_SAMPLING
```

```
BT_E16_SAMPLING
BT_E17_SAMPLING
BT_E18_SAMPLING
BT_E19_SAMPLING
```

Appendix B: JACO Integration System Electronics

This photograph shows the JACO wrist connected to SynTouch electronics allowing up to three connected BioTacs.



CONTACT

SynTouch LLC
2222 S.Figueroa PH2
Los Angeles, CA 90007

213.973.4102
info@syntouchllc.com

www.syntouchllc.com