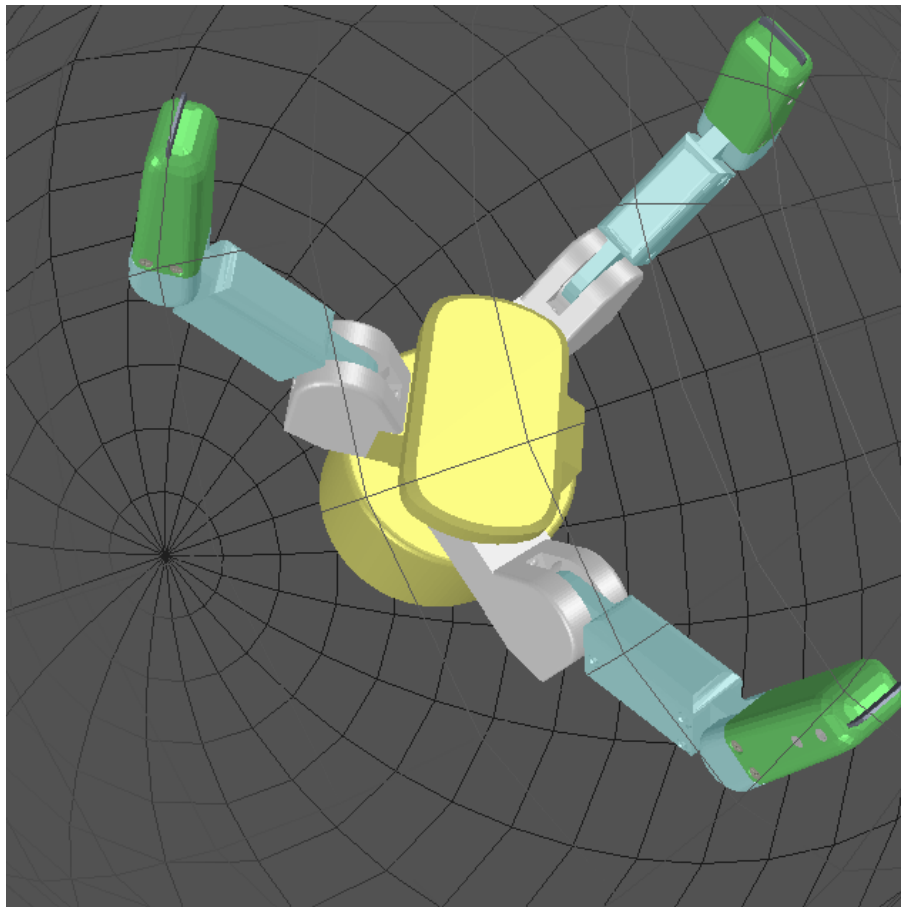


BarrettHand

Control GUI Manual



Barrett Technology Inc.

Table of Contents

<i>List of Figures/Equations.....</i>	<i>3</i>
1 INTRODUCTION.....	4
2 CONFIGURE TAB.....	5
2.1 CONFIGURATION.....	6
2.2 TEST DELAYS.....	7
2.3 UPLOAD FIRMWARE TO HAND.....	8
3 SUPERVISE TAB.....	10
3.1 TERMINAL.....	11
3.2 CONTROL PANEL.....	13
3.2.1 Parameters.....	15
4 REAL TIME TAB.....	16
4.1 REAL TIME CONTROL.....	17
4.1.1 Expression Syntax.....	20
4.1.2 Data From File.....	22
5 REAL TIME PLOT DATA TAB.....	24
5.1 PLOT DATA.....	24
6 VISUAL TAB.....	26
6.1 VISUAL CONTROL.....	27

List of Figures/Equations

Figure 1 – Configure Tab.....	5
Figure 2 – Configure Tab - Configuration.....	6
Figure 3 – Configure Tab – Test Delays.....	7
Figure 4 – Configure Tab – Delay Histogram.....	8
Figure 5 – Configure Tab – Upload Firmware to Hand.....	8
Figure 6 - Configure Tab – Upload Firmware Dialog Box.....	9
Figure 7 – Configure Tab – Message Prompt.....	9
Figure 8 – Supervise Tab.....	10
Figure 9 – Supervise Tab - Terminal.....	11
Figure 10 – Supervise Tab – Control Panel.....	13
Figure 11 – Supervise Tab – Parameters Box.....	15
Figure 12 – Real Time Tab.....	16
Figure 13 – Real Time Tab – Real Time Control Box.....	17
Equation 1 – Real Time Tab - Real Time Velocity Control.....	17
Figure 14 – Real Time Tab – Data From File Box.....	22
Figure 15 – Real Time Tab – Data From File as Control Velocity.....	22
Figure 16 – Real Time Plot Data Tab.....	24
Figure 17 – Real Time Plot Data Tab – Save Plot.....	25
Figure 18 – Visual Tab.....	26
Figure 19 – Visual Tab – Multiple Selection.....	27
Figure 20 – Visual Tab – OpenGL Window Features.....	28

1 Introduction

BHControl is a cross-platform software application designed for the BarrettHand. This software provides functionality of the BarrettHand API through an intuitive graphical user interface (GUI). BHControl can be used to help develop, test, and debug projects related to the BarrettHand.

This application includes five sections:

1. **Configure** – Initialize communication with the hand, test transmission delays, and upload firmware.
2. **Supervise** – Issue supervisory mode commands, test command sequences, load previously saved command sequences, and generate C++ code along with a makefile for standalone programs.
3. **Real Time** – Issue RealTime mode commands, test control laws, load previously saved control laws, and generate C++ code along with a makefile for standalone programs.
4. **Real Time Plot Data** – Run RealTime mode commands specified in the Real Time tab, graph real-time plot data based on feedback and control signals, and save plot data to a .csv file.
5. **Visual** – Visualize the hand and interact with hand components. Control fingers simultaneously by selecting multiple components.

BHControl is programmed using the wxWidgets Library. Toolbars, buttons, text windows, etc. are native to the operating system currently being used. BHControl contains tabs corresponding to each section, allowing for easy navigation within the GUI. When the application starts GUI will not be able to do much until the hand is initialized. Once the hand is initialized, the tabs and buttons will be enabled.

Note: The minimum recommended resolution for BHControl is 1024x768.

2 Configure Tab

The screenshot shows the 'Configure' tab of the BHControl software. The interface is divided into several sections:

- Configuration:** Includes a 'Model Num' dropdown set to 'BH8-262', radio buttons for 'CAN' and 'Serial' (with 'Serial' selected), a checked 'Auto-Init Hand' checkbox, a 'Com Port' dropdown set to 'Com 1', and buttons for 'Initialize Library' and 'Reset Defaults'.
- Upload Firmware to Hand:** Displays 'Bootloader configured for BH8-262 Hand'. It has a 'Current Path' field with a 'Browse for firmware file...' button, a 'Current File' field showing 'n/a' with a 'Browse' button, and a 'Firmware On Hand' field showing 'n/a' with an 'Upload' button.
- Test Delays:** Contains two sub-sections: 'Motors' with checkboxes for 'One', 'Two', 'Three', and 'Spread'; and 'Control' with checkboxes for 'Velocity' and 'Prop. Gain'. Below these are 'Feedback' checkboxes for 'Velocity', 'Position', 'Strain', and 'Delta Pos'. A 'Test Delays' button is present.
- Results (ms):** A table showing test results:

Results (ms)			
Out/In:	n/a	Min:	n/a
Avg.:	n/a	Max:	n/a
- Delay Histogram:** A plot with 'Y (count)' on the vertical axis (0.00 to 1.00) and 'X (ms)' on the horizontal axis (0.10 to 0.90). The plot area is currently empty.

Figure 1 – Configure Tab

2.1 Configuration

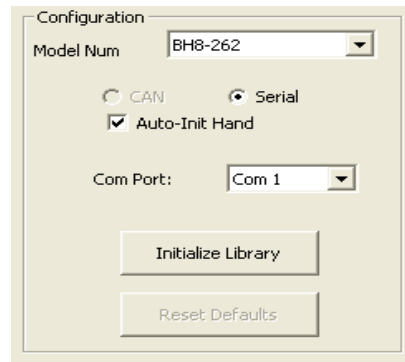


Figure 2 – Configure Tab - Configuration

The Configuration Box Initializes the BarrettHand and BarrettHand library. The library must be initialized before any communication to and from the hand can begin.

Model Num:

The user must select the model number of the hand in this listbox that will be used for communicating with the correct hardware.

Auto-Init Hand:

If Auto-Init Hand is checked, the BarrettHand will automatically initialize its motors after initializing the library. If Auto-Init Hand is unchecked, motors will need to be initialized manually through the Supervisory Tab.

Com Port:

The Com Port tells BHControl which serial port to use to communicate to the BarrettHand. Once the BarrettHand is initialized, the Com Port selection will be locked until the application is closed.

Baud Rate:

The Baud Rate specifies the bits of data per second sent over serial. The default value for the Baud Rate is 9600.

Initialize Library:

This button initializes the BarrettHand Library. Once clicked, it initializes communication port parameters, resets internal variables, clears all buffers, and starts a low-level communications thread.

Reset Defaults:

This button issues a reset command to the hand and resets the Baud Rate to the default value of 9600. Resetting the hand does not re-initialize the library, however. Motors will need to be re-initialized if the BarrettHand is reset; BHControl will automatically reinitialize motors if Auto-Init Hand is checked when Reset Defaults is clicked.

2.2 Test Delays

Results (ms)	
Out/In:	5/5
Avg.:	22.39
Min:	19
Max:	31

Figure 3 – Configure Tab – Test Delays

The Test Delays Box records communication delay with the BarrettHand in RealTime mode. Communication delay is the period between a control block being sent and a feedback block being received. Library serial communications, CPU processing, and hand processing are also part of the communication delay. One-hundred delays are measured. The delay times will vary depending on which parameters are selected within the Test Delays Box. When recording is over, a Delay Histogram is rendered to display a distribution of delay times.

Motors:

These checkboxes specify which motors will be enabled in RealTime mode while the communication delays are being tested. Selected motors will send and receive data.

Feedback:

These checkboxes specify parameters the BarrettHand can return in a feedback block. Checked parameters will be sent from the BarrettHand in RealTime mode.

Control:

These checkboxes allow for velocity and proportional gain control to be sent to the BarrettHand. These control settings can be enabled by checking the boxes. Invalid input in the text window for velocity control will default to 0. Proportional Gain is currently set at 8.

Test Delays:

In order to use Test Delays, first select desired motors. Next, enable desired feedback and control parameters. Once Test Delays is clicked, BHControl will enable RealTime mode on the BarrettHand and measure one-hundred delays. RealTime mode is stopped once the delays are measured.

Results:

The Results Box displays how many characters are sent and received (*Out/In*), average delay time (*Avg.*), minimum delay time (*Min*), and maximum delay time (*Max*).

Delay Histogram:

The Delay Histogram displays a bar graph representing the number of delays recorded at each time interval. The graph features panning, zooming, scaling, etc. All features are shown by right clicking anywhere in the Delay Histogram window and selecting “Show Mouse Commands”.

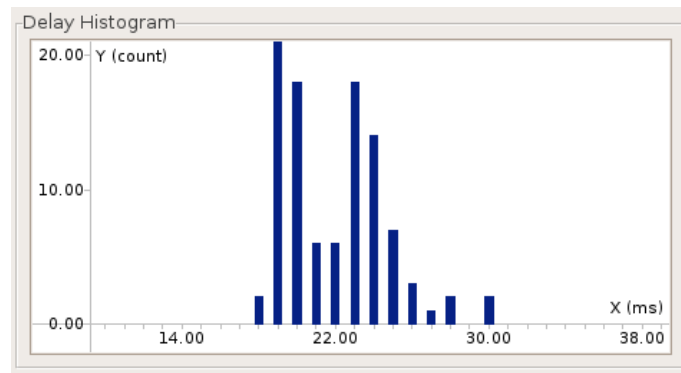


Figure 4 – Configure Tab – Delay Histogram

2.3 Upload Firmware to Hand

The figure shows a dialog box titled "Upload Firmware to Hand". It contains three input fields: "Current Path:" with a text box containing "Browse for firmware file...", "Current File:" with a text box containing "n/a", and "Firmware On Hand:" with a text box containing "n/a". To the right of these fields are two buttons: "Browse" and "Upload". Below the input fields is a progress bar consisting of 16 small squares, all of which are currently empty.

Figure 5 – Configure Tab – Upload Firmware to Hand

During library initialization, the program will show an error message if the BarrettHand has an outdated or invalid .S19 file. If an older version of the firmware is detected or the memory on the BarrettHand has decayed, valid firmware will be required for the BarrettHand to operate correctly.

Browse:

This button will open a file dialog box. Once the file dialog is open, the user can find and select a firmware file (.S19) to upload to the hand. The Current Path and Current File fields in the GUI will update accordingly.

Upload:

This button will upload firmware to the hand. When clicked, a dialog box will prompt the user to power off the BarrettHand. Once the BarrettHand is powered off and the user clicks "Ok", a message will appear, signaling the user to power the BarrettHand back on.

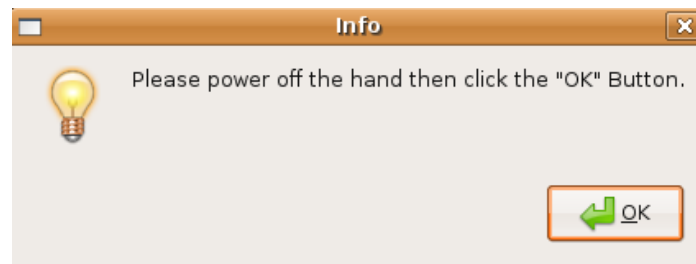


Figure 6 - Configure Tab – Upload Firmware Dialog Box

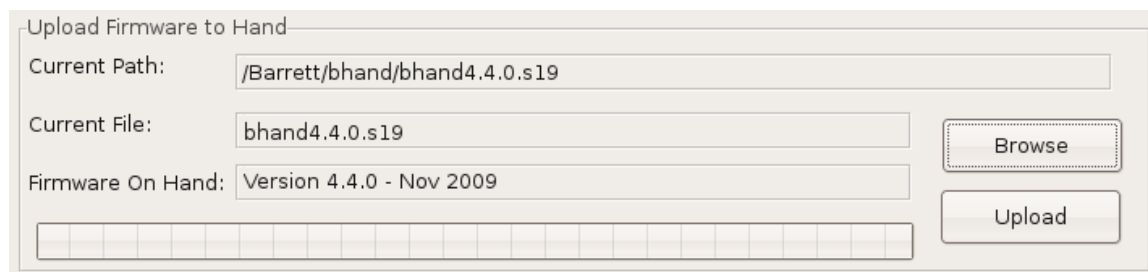


Figure 7 – Configure Tab – Message Prompt

Once the user powers the BarrettHand back on, the GUI will disable all features and the progress bar will update accordingly. Do NOT attempt to power off the hand or close the program while the firmware is uploading to the BarrettHand as potential damage could be caused to the hardware. Once the firmware upload is complete, the GUI will be re-activated. If the library was not initialized before uploading firmware to the hand, click the Initialize Library button after the firmware has been uploaded. Once the library is initialized, the Firmware On Hand Box will display the current version of the firmware on the BarrettHand.

If the library was already initialized before uploading firmware, the BarrettHand motors need to be initialized again (they can be initialized through the Supervise Tab).

3 Supervise Tab

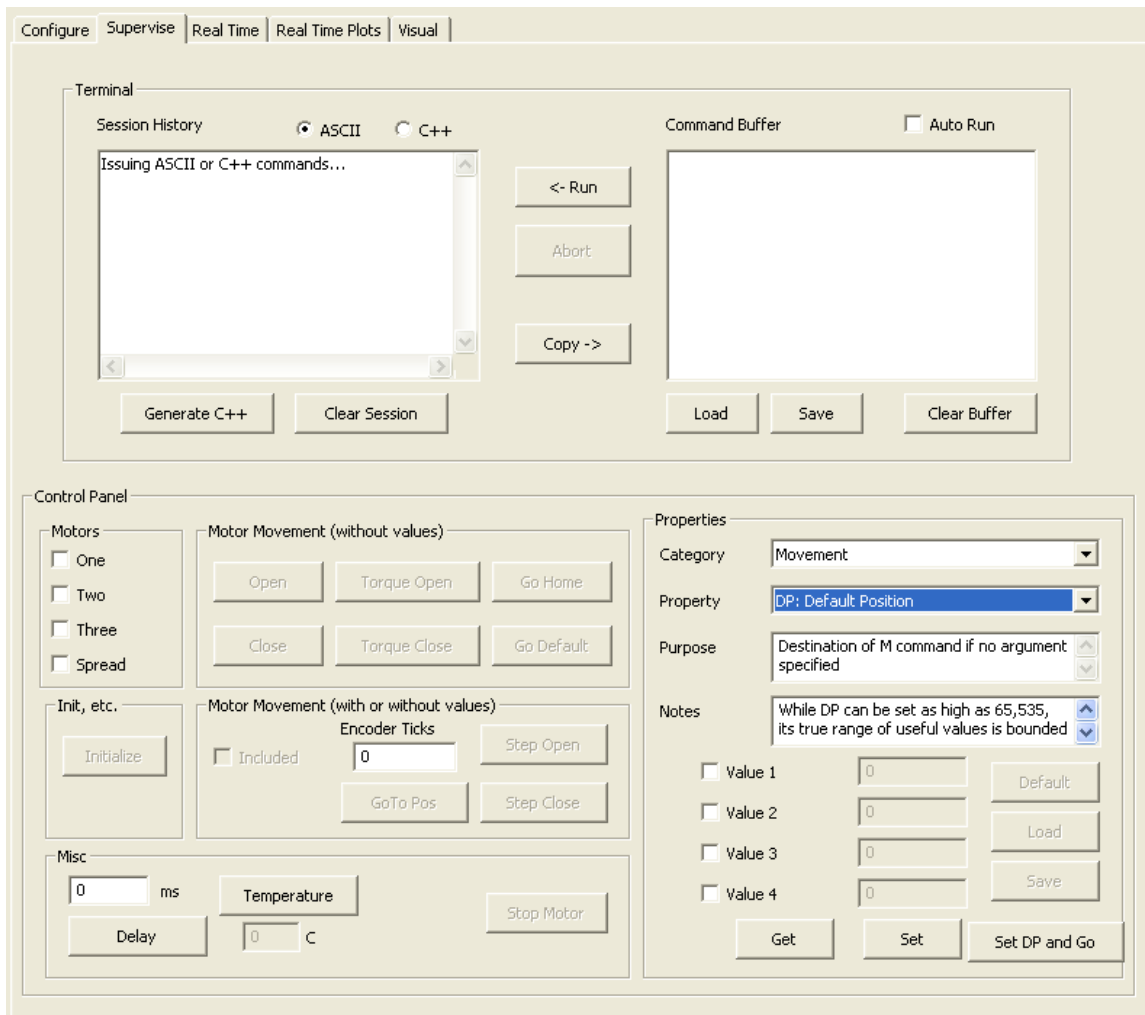


Figure 8 – Supervise Tab

3.1 Terminal

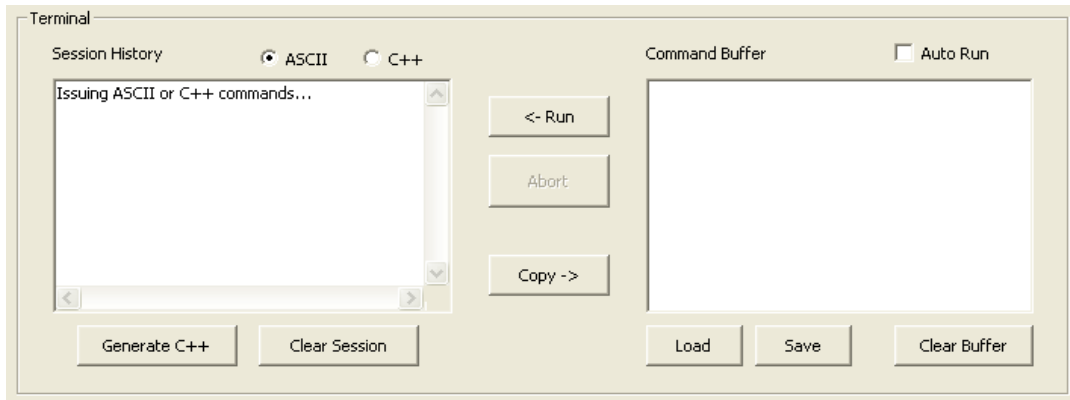


Figure 9 – Supervise Tab - Terminal

From the Terminal Box the user can send supervisory commands in the Command Buffer text window (non – RealTime commands) to the BarrettHand. Supervisory commands, in ASCII format, are parsed and call BarrettHand functions.

Command Buffer:

The Command Buffer text window handles all information sent to the hand. The user can populate the window with commands by either clicking buttons in the Supervise Control Panel or typing in commands manually. Each time a command is executed, it will be transferred from the Command Buffer window to the Session History window.

Session History:

The Session History text window records all commands issued from the Command Buffer window. The Session History window can toggle between ASCII or C++ version of the supervisory commands.

Run:

The Run button will execute all commands from the Command Buffer window one at a time. Supervisory commands will continue executing until the Command Buffer window is empty or an error is returned.

Abort:

This button will stop the command buffer from running any more commands.

Auto Run:

If the Auto Run checkbox is enabled, hitting <Enter> key while in the Command Buffer window will automatically run all supervisory commands currently listed. Also, any buttons clicked from the Control Panel Box will automatically execute. If the Auto Run checkbox is not enabled, buttons clicked from the Control Panel Box will accumulate in the Command Buffer window. The Command Buffer's contents can be executed by clicking the Run button.

Clear Session / Clear Buffer:

Each button clears its respective window.

Load / Save:

These buttons load and save “.bh” files, which are a list of ASCII commands. Commands can be loaded to or saved from the Command Buffer in this format. Loading a file will completely overwrite the contents previously stored in the Command Buffer window.

Copy:

Copying of files can be done in two methods. Keyboard shortcuts for copy <Ctrl-C> and then paste <Ctrl-V> can be used to copy text from the Session History window to the Command Buffer window. Note, the Session History window is read-only. The alternative method is using the Copy button. If no text is selected, pressing the copy button will copy all the contents of the Session History window into the Command Buffer. Otherwise, the copy button will only copy selected text into the Command Buffer window. The Copy button is usable only when the ASCII radio button is selected.

Generate C++:

This button will generate a C++ program that executes the sequence of supervisory commands listed in the Session History window. The program will initialize the BarrettHand and the library, along with handling errors accordingly. A corresponding Makefile will be generated as well. For example, if a “file.cpp” is generated, a Makefile named “file.mak” will also be created.

The Makefile links automatically to the location of the BHControl's BarrettHand Library (assuming BHControl was run from BHControl, BHControl/bin/Debug, BHControl/bin/Release, or BHControl/bin). To make the file, navigate to where the “file.cpp” is located (via a terminal or a windows command prompt) and simply type in:

make -f file.mak (for Linux)

or

mingw32-make -f file.mak ARCH=windows (for Windows)

3.2 Control Panel

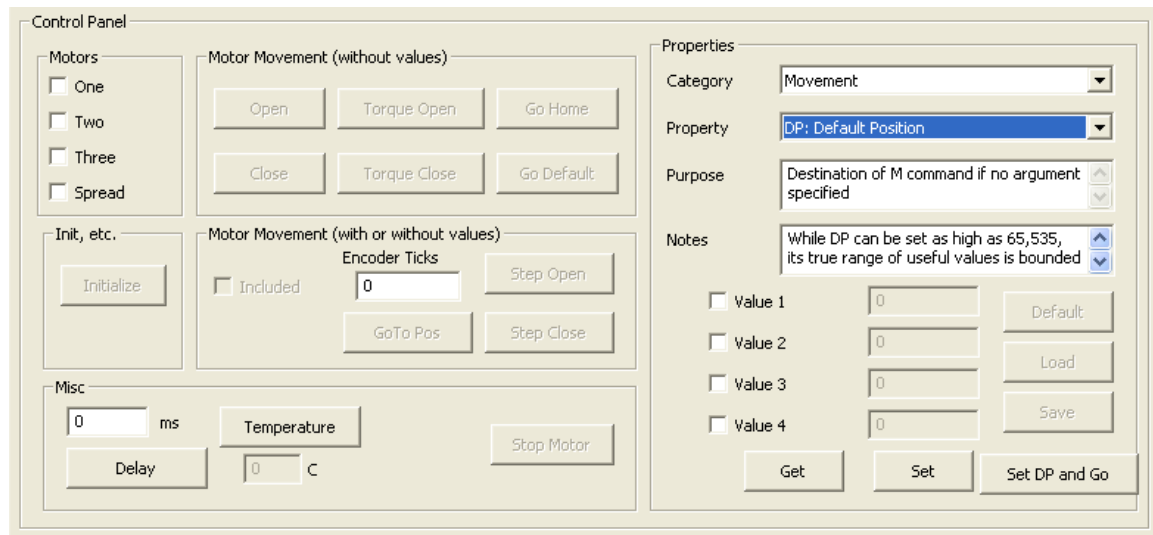


Figure 10 – Supervise Tab – Control Panel

The Control Panel contains all the buttons corresponding to Supervisory mode command provided by the BarrettHand library. The buttons on the left side of the Control Panel contain the commands that move the motors. First, select which motors to control and then press the desired command button. At least one motor must be selected to enable the command buttons.

<u>Buttons:</u>	Function
<i>Initialize</i>	Initializes the selected motors
<i>Stop Motor</i>	Idles the selected motors
<i>Open</i>	Actuates selected motors to open corresponding fingers or spread
<i>Close</i>	Actuates selected motors to close corresponding fingers or spread
<i>Torque Open</i>	Commands velocity of motors that opens the corresponding fingers or spread with control of motor torque at stall
<i>Torque Close</i>	Commands velocity of motors that closes the corresponding fingers or spread with control of motor torque at stall
<i>Go Home</i>	Opens all fingers and the spread, regardless of motors selected
<i>Go Default</i>	Moves selected motors to the default position, DP
<i>Default</i>	Loads the default parameters from the EEPROM for the selected motors

<u>Buttons (cont'd):</u>	Function
<i>Load</i>	Loads the active parameters from the EEPROM for the selected motors
<i>Save</i>	Saves the active parameters to EEPROM for the selected motors
<i>Go Position</i>	Moves the selected motors to the position indicated by the above text window
<i>Step Open</i>	Moves the selected motors in the open direction by the number of encoder counts in the above text window
<i>Step Close</i>	Moves the selected motors in the close direction by the number of encoder counts in the above text window
<i>Delay</i>	Inserts a delay sequence equal to the number of milliseconds specified by the corresponding text window
<i>Temperature</i>	Shows the present temperature of the internal electronic boards of the BarrettHand

3.2.1 Parameters

Figure 11 – Supervise Tab – Parameters Box

The Parameters Box is able to get and set BarrettHand parameters. Available parameters are listed within the list box. A desired parameter can be chosen by clicking on the list box to display a pull down menu. The user can enable/disable selected motors to get and set by clicking the checkboxes.

Get:

This button gets parameters for the selected motors. Regardless of the Auto Run checkbox state, the selected motors' text windows will update accordingly with the value of the selected parameter. If Auto Run is unchecked, the supervisory commands to get the values will appear in the command buffer as well.

Set:

This button sets parameters for the selected motors. If Auto Run is selected, the values of selected motors' chosen parameter will be set to the values in the text windows. Otherwise, the command buffer will append the supervisory commands to set the motors' chosen parameter. Some parameters are read-only; the Set Button will be enabled only if the user is allowed to change the value of the chosen parameter.

Set DP and Go:

This button functions the same as the Set button. It is only usable if the Default Position (DP) parameter is selected. Set DP and Go will both set the selected motors' default position to values specified and then the selected motors will move to the new default position immediately.

4 Real Time Tab

The screenshot shows the 'Real Time' tab selected in the software's navigation bar. The interface is divided into several sections:

- Real Time Control:** Contains a tip about loading 'RealTime.bh' files. It includes buttons for 'Load Code', 'Generate C++', 'Save Code', 'Clear Code', 'Execute Before', 'Load', 'Execute After', and 'Load'. There are also checkboxes for 'Control Velocity' (checked) and 'Control Proportional Gain' (unchecked).
- One, Two, Three, Spread:** Four rows of checkboxes and input fields for configuring different control modes.
- Real Time Parameters:** A section with checkboxes for 'Position', 'Strain', 'Velocity', and 'Delta'. Below these are input fields for 'FVEL Multiplier', 'Delta Multiplier', and 'Cvel Multiplier', all set to '1'.
- Termination Condition:** A section with a single input field.
- Run/Abort:** Two buttons for executing or aborting the real-time control.
- Data from File:** A section with a 'File:' label, an input field containing 'n/a', and a 'Load' button.

Figure 12 – Real Time Tab

4.1 Real Time Control

Figure 13 – Real Time Tab – Real Time Control Box

The Real Time Control Panel is used to create control laws for the fingers and spread. Real Time control is useful when immediate control of the motor velocity is desired. Unlike supervisory mode, the Real Time mode allows the user to change the velocity while the motors are still moving. The eight expression fields (top right) define the Control Velocity and Control Proportional Gain signals to be sent to the motors at each time step. Select the desired motors and control parameters by checking the motor number, Control Velocity, and Control Proportional Gain check boxes. Enter an expression in the enabled fields. The expressions can include feedback variables, time, constants, and numeric sequences loaded from a file, see Section 4.1.1. The velocity of a given motor will respond according to the control law in Equation 1.

$$MC_n = (K/4) * Y_n$$

Equation 1 – Real Time Tab - Real Time Velocity Control

Where:

MC_n is the motor command output.

K is the proportional gain calculated from the Control Proportional Gain expression.

Y_n is the (Control Velocity expression * CVEL Multiplier – Actual Velocity).

After the chosen expression windows have been filled in, select the desired feedback parameters and multipliers in the Real Time Parameters section. The following is a list of the parameters and their definitions:

<u>Parameter:</u>	Definition
<i>Position</i>	Motor Position Feedback
<i>Velocity</i>	Motor Velocity Feedback
<i>Strain</i>	Finger Strain Gauge Feedback
<i>Delta</i>	Motor Delta Position Feedback
<i>FVel Multiplier</i>	The BarrettHand divides the Actual Velocity by FVel Multiplier before sending the Velocity Feedback to the host computer. $\text{Velocity Feedback} = \text{Actual Velocity} / \text{FVel Multiplier}$
<i>Delta Multiplier</i>	The BarrettHand divides the Actual Delta Position by Delta Multiplier before sending the Delta Position Feedback to the host computer. $\text{Delta Position Feedback} = \text{Actual Delta Position} / \text{Delta Multiplier}$
<i>CVel Multiplier</i>	After receiving the Control Velocity, the BarrettHand multiplies the Control Velocity by CVel Multiplier to determine the Command Velocity to control the BarrettHand. $\text{Control Velocity} * \text{CVel Multiplier} = \text{Command Velocity}$

After the Real Time Parameters have been set, a Termination Condition must be provided. Real Time mode will stop running once the Termination Condition evaluates to TRUE. The Termination Condition can include all the variables supported by the Control Velocity and Control Proportional Gain expressions (refer to Section 4.1.1).

Data From File:

Values loaded from a file can be used in the expressions in Real Time mode. Refer to Section 4.1.2 for more information regarding file format.

Execute Before / Execute After:

These two text windows on the left side of the Real Time Control Box support supervisory ASCII commands. If Execute Before is checked, the supervisory commands in that text window will be executed before Real Time control is started. If Execute After is checked, the supervisory commands in that text window will be executed after Real Time control is complete. Commands can be entered into these text windows via typing directly or loading from a file created within the Supervise Tab. See the BarrettHand User Manual for more information on specific BarrettHand commands.

Run:

Once run is clicked, if Execute Before is checked, those supervisory commands will be run. The expressions will be parsed into a tree and variables will be substituted recursively through the nodes of the tree. If any syntax errors are found, Real Time mode will stop. At each time step the program evaluates the enabled expressions to obtain desired control signals, sends them to the hand, and reads the feedback variables that have been activated. The Termination Condition (always required) is evaluated at each time step. If the Termination Condition is true, real time mode is aborted. Then, if Execute After is checked, the corresponding supervisory commands are issued. If at any time any error occurs, the entire process will stop immediately and Run will need to be hit again to restart the process.

Abort:

The Abort button will stop the Run process immediately as well.

The combination of settings in the Real Time Control Box defines a user “program” that a built-in interpreter executes in real-time to control the hand.

Save Code:

This button saves the settings/expressions of the Real Time Control Box to a file.

Load Code:

This button loads the settings/expressions from a file into the Real Time Control Box.

Clear Code:

This button clears the settings/expressions of the Real Time Control Box to its defaults.

Generate C++:

This button will generate a C++ program that translates the Real Time Control Box settings into a C++ program along with Execute Before and Execute After supervisory commands. The program will initialize the BarrettHand and the library, along with handling errors accordingly. A corresponding makefile will be generated as well. For example, if “file.cpp” is generated, a makefile named “file.mak” will also be created. The makefile links automatically to the location of the BHControl's BarrettHand Library (assuming BHControl was run from BHControl, BHControl/bin, BHControl/bin/Debug, BHControl/bin/Release). To make the file, navigate to where the “file.cpp” is located via a terminal or a windows command prompt and simply type in:

```
make -f file.mak                                (for Linux)
or
mingw32-make -f file.mak ARCH=windows           (for Windows)
```

4.1.1 Expression Syntax

The expressions used for Real Time control follow a C-style syntax. The program will return an error if the syntax is incorrect. The expressions are checked when the user attempts to Run the program. All activated fields must have expressions entered.

The expressions are made of variables, functions, and operators. All variables (predefined) are internally stored as integers. At the beginning of each update, they are converted to double-precision floating point numbers used in all expression evaluations. The resulting evaluation is rounded to the nearest integer.

Variables:

D	Delta Positon
F	File Data (0 is no file is opened)
N	Update Number
P	Position
S	Strain
T	Time elapsed in milliseconds since RealTime control began
V	Velocity

Each variable can be used with a subscript. For example, P1 refers to the position of finger 1 in all expression fields. If the variable does not include a subscript, its value will correspond to the motor value of the text window it is located in. For example, if P is used in the Motor 2 Velocity Control expression field, then P refers to P2 – the position of motor 2. If a variable is used without a subscript in the Termination Condition, it will default to motor 1's value. Although lower case variables can be used, only upper case variables will correctly translate into a generated C++ file.

Functions:

abs(x)	Returns the absolute value of x
acos(x)	Returns the Arc-Cosine of x
asin(x)	Returns the Arc-Sin of x
atan(x)	Returns the Arc-Tan of x
ceil(x)	Returns the smallest integer greater than x
cos(x)	Returns the Cosine of x
exp(x)	Returns the value e^x
floor(x)	Returns the largest integer smaller than x
log(x)	Returns the Log, base 10, of x
ln(x)	Returns the Natural Log of x
random(x)	Generates a uniformly distributed real number between zero and x
round(x)	Returns the integer closest to x
sin(x)	Returns the Sine of x
sqrt(x)	Returns the square root of x
tan(x)	Returns the Tangent of x

Note: All trigonometric functions are in radians

Arithmetic Operators (in order of precedence):

^	Exponentiation
*	Multiplication
/	Division
%	Modulus (A % B = remainder of A divided by B) (works for floats)
+	Addition
-	Subtraction

Parentheses can be used to change precedence. Note, in order to negate values, an extra set of parentheses must be included around the value that is to be negated. For example, “5 + -5” must be entered in the expression window as “5 + (-5)”.

Logical Operators:

!	Not
&&	And
	Or

Parentheses should be used to explicitly define expressions. For example, the expression `50 && 25 > 10` can evaluate to both TRUE and FALSE. The expression `(50 && 25) > 10` will yield FALSE and `50 && (25 > 10)` will yield TRUE.

Comparison Operators:

>=	Greater than or equal to
<=	Less than or equal to
!=	Not equal to
<	Less than
>	Greater than
==	Equal to


Once again, use Parentheses to explicitly define the two values being compared.

Conditional Expression Operator:

expression ? value1 : value2

If the expression is true, the statement evaluates to value1, otherwise to value2. Parentheses must be used to denote the fields expression, value1, and value2.

4.1.2 Data From File



Data From File

File:

Figure 14 – Real Time Tab – Data From File Box

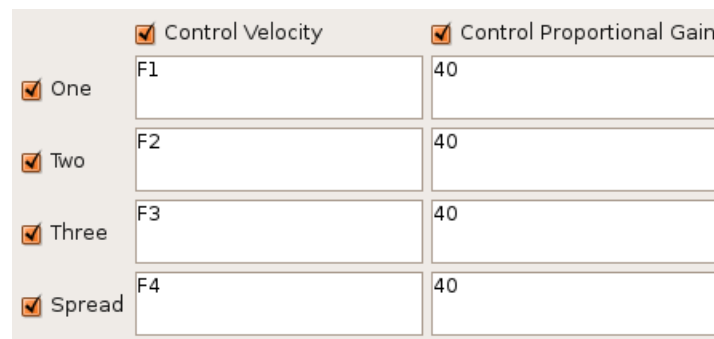
The user can specify a data file to be used in expressions by loading a file into the Real Time Tab. The format of the file, Real Time Data (.rtd), should contain a table of numbers with four tab-delimited columns and any number of rows, N. The variable F1 will evaluate to column 1, row number equal to the present update step. F2 will evaluate to column 2, row number equal to the present update step, etc. If the update step is larger than the last row, the update number = update number % total rows in the file.

Here is an example:

File Data:

30	100	0	35
30	100	0	35
50	100	0	35
50	40	0	35
50	40	0	35

Set Control Velocity / Proportional Gain Expressions accordingly:



	<input checked="" type="checkbox"/> Control Velocity	<input checked="" type="checkbox"/> Control Proportional Gain
<input checked="" type="checkbox"/> One	<input type="text" value="F1"/>	<input type="text" value="40"/>
<input checked="" type="checkbox"/> Two	<input type="text" value="F2"/>	<input type="text" value="40"/>
<input checked="" type="checkbox"/> Three	<input type="text" value="F3"/>	<input type="text" value="40"/>
<input checked="" type="checkbox"/> Spread	<input type="text" value="F4"/>	<input type="text" value="40"/>

Figure 15 – Real Time Tab – Data From File as Control Velocity

Note, there must be some Proportional Gain for the motors to move (refer to Equation 1). Also, not all expression windows have to be enabled to use Data From File variables. Each time control velocities are sent to the BarrettHand as follows:

Update Step	F1	F2	F3	F4
1	30	100	0	35
2	30	100	0	35
3	50	100	0	35
4	50	40	0	35
5	50	40	0	35
6	30	100	0	35
7	30	100	0	35
8	50	100	0	35
9	50	40	0	35
10	50	40	0	35
etc.	etc.	etc.	etc.	etc.

5 Real Time Plot Data Tab

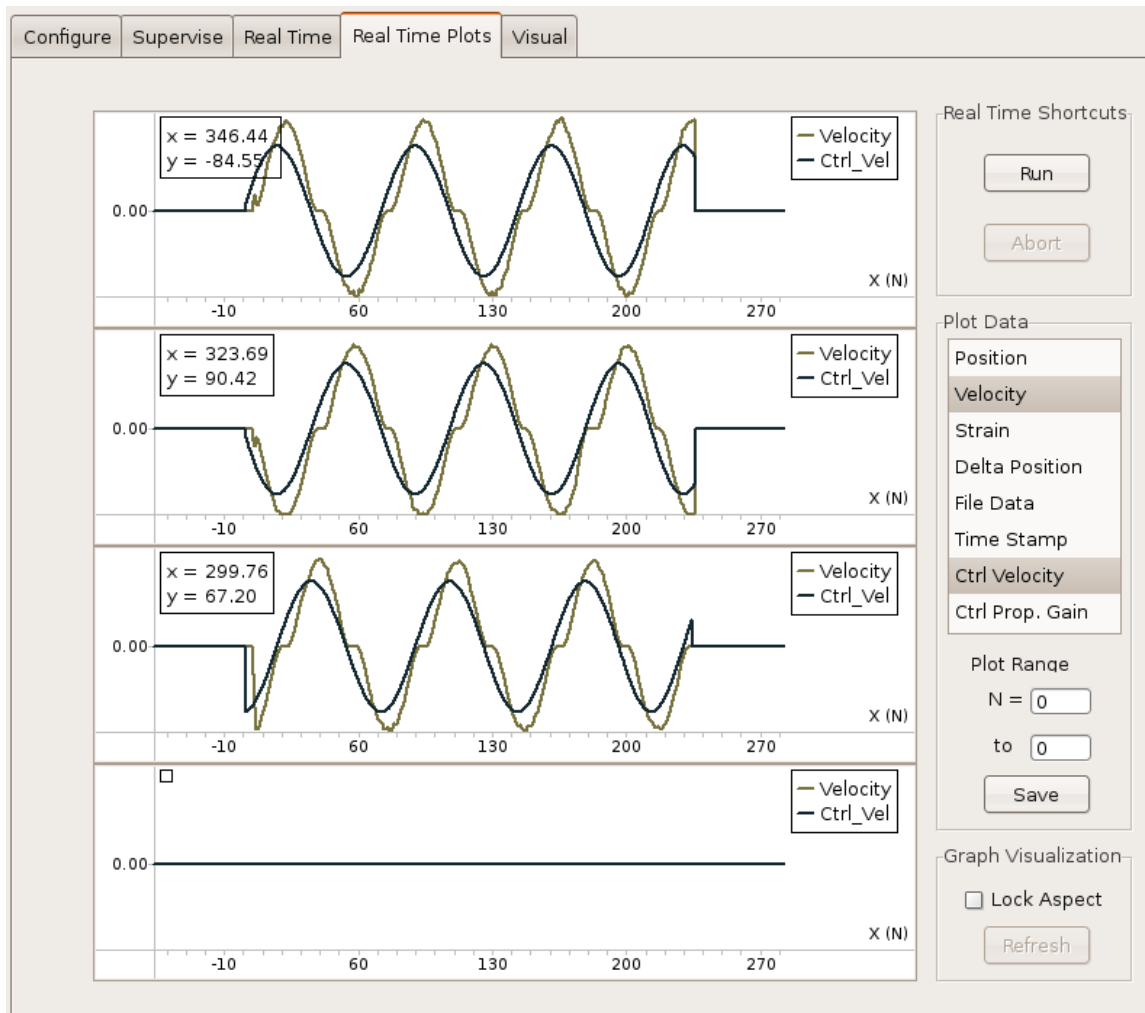


Figure 16 – Real Time Plot Data Tab

5.1 Plot Data

The Real Time Plot Data Tab is an optional feature usable with Real Time mode. This tab allows users to visualize data more readily. The Real Time Plot Data Tab plots Control and Feedback variables in Real Time as soon as data is received onto the four corresponding motor graphs. The top graph represents motor one, the second graph represents motor two, the third graph represents motor three, the fourth graph represents motor spread. The graphs resize themselves automatically to get the best fit of the plot data, but the user is free to pan, zoom, scale to suit his or her needs. Right click on any graph and select “Show mouse commands...” for instructions.

List Box:

The list box to the right shows all available variables that can be plotted. Variables in the list box can be toggled on and off at any time, even while Real Time control is running. Each variable is plotted in a different color; up to 8 variables can be plotted in the same graph at once.

Run:

The Run Button is a shortcut that will run programs created in the Real Time Control panel from within the Real Time Plots panel.

Abort:

Aborts Real Time mode immediately.

Lock Aspect:

This check box will lock/unlock the aspect ratio for all graphs. For variables such as Time Stamp and Position, it is best to have the aspect ratio unlocked. Lock Aspect is toggled off by default.

Save Plot:

Save Plot will save all the graph data into a comma separated value “.csv” file. If the *N* = and *To* text windows are non-zero, the “.csv” value will only contain values for that range of update steps. By default, all values from all update steps are saved to the file. Inactive motors will have the value of zero for all parameters.

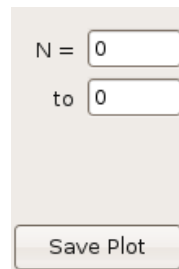


Figure 17 – Real Time Plot Data Tab – Save Plot

An example “.csv” file is as follows:

```
N, Ctrl_Vel1, Ctrl_Vel2, Ctrl_Vel3, Ctrl_Vel4, Ctrl_Gain1, Ctrl_Gain2, Ctrl_Gain3, Ctrl_Gain4
0, 6.503930, -5.606993, -64.217316, 0.000000, 40.000000, 40.000000, 40.000000, 0.000000
1, 8.012341, -7.115150, -64.058426, 0.000000, 40.000000, 40.000000, 40.000000, 0.000000
2, 9.677564, -8.780029, -63.841507, 0.000000, 40.000000, 40.000000, 40.000000, 0.000000
3, 11.336661, -10.438716, -63.581650, 0.000000, 40.000000, 40.000000, 40.000000, 0.000000
4, 12.988526, -12.090102, -63.279037, 0.000000, 40.000000, 40.000000, 40.000000, 0.000000
5, 14.684925, -13.785931, -62.922020, 0.000000, 40.000000, 40.000000, 40.000000, 0.000000
```

6 Visual Tab

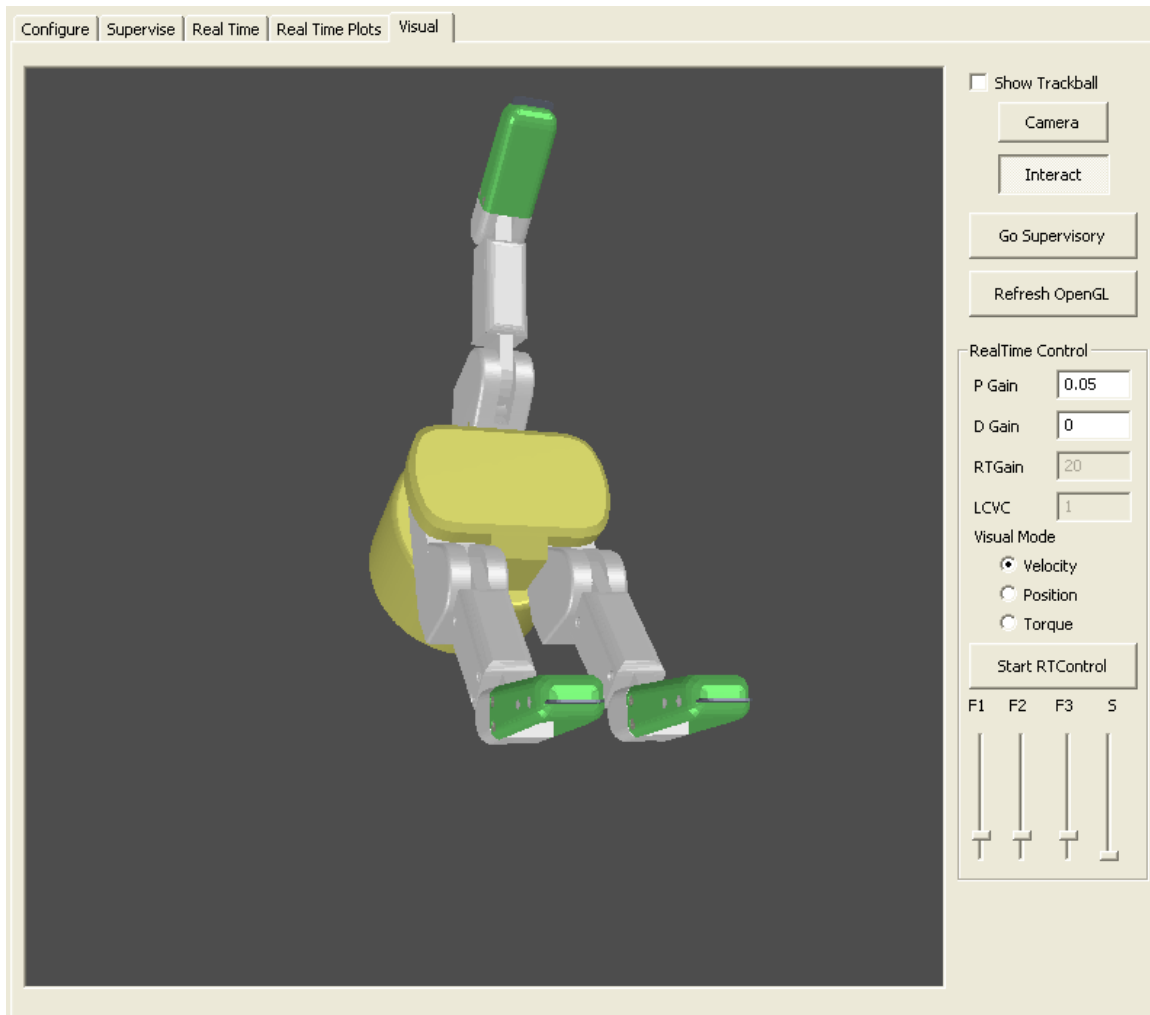


Figure 18 – Visual Tab

6.1 Visual Control

The Visual Tab provides an OpenGL window to interact with the hand. The BarrettHand OpenGL model moves and behaves in a similar fashion to the real BarrettHand. Simple bounding-sphere collision detection, a basic shading model, and rigid-body animation are implemented in this visualizer.

Users can move the fingers of the hand by right-clicking and holding on a given finger and then moving the mouse up or down. The finger will close if the mouse is moved up or vice versa. If the base of fingers 1 or 2 is right-clicked, moving the mouse left or right will open or close the spread for the hand in the visual, respectively. Release the right mouse button to have the spread go to the position shown in the visual.

Multiple fingers can be moved simultaneously as well. If the Control key is held while right clicking on a finger, the finger will change to a shade of blue. Control and right-clicking on a blue finger will revert the finger back to gray.

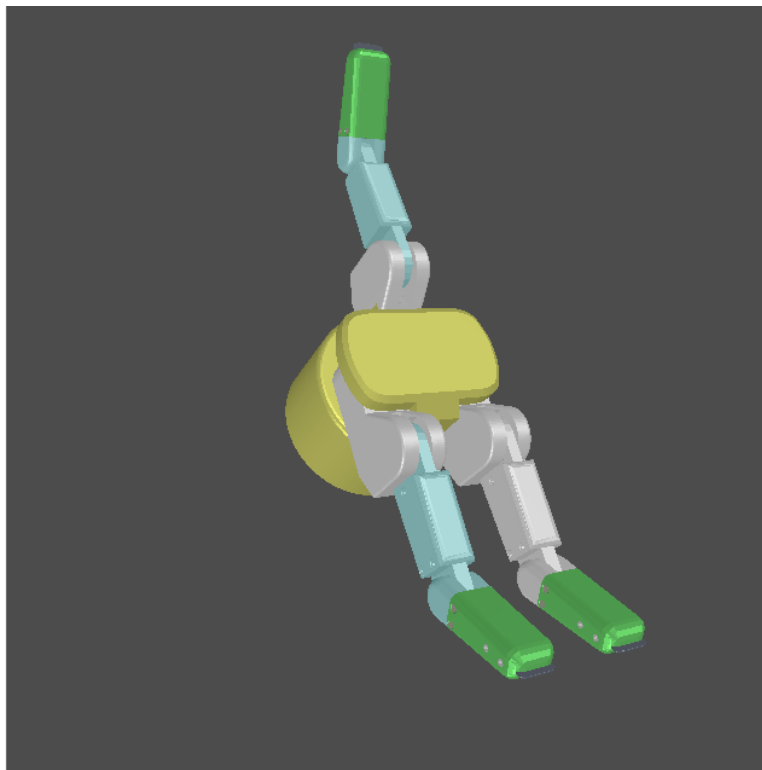


Figure 19 – Visual Tab – Multiple Selection

Show Trackball:

To help visualize the rotation, checking the Show Trackball box will display a sphere to show how mouse movement is associated with the three-dimensional rotations.

Camera/Interact:

The Camera and Interact toggle buttons determine the function of the scroll wheel. If the Interact button is toggled (on by default), scrolling the mouse wheel up will incrementally close all selected (blue) fingers. Likewise, scrolling the mouse wheel down will open all selected (blue) fingers. If the Camera button is toggled, the mouse scroll will zoom the camera viewing the OpenGL scene in or out.

The OpenGL window also implements trackball rotation, which allows the user to freely rotate the BarrettHand model in three-dimensions using two-dimensional mouse movement (implemented with quaternions). Left-clicking and dragging the mouse will rotate the BarrettHand model.

Here is a screen shot demonstrating camera zoom, multiple finger movement, camera rotation, and with the Show Trackball check box checked:

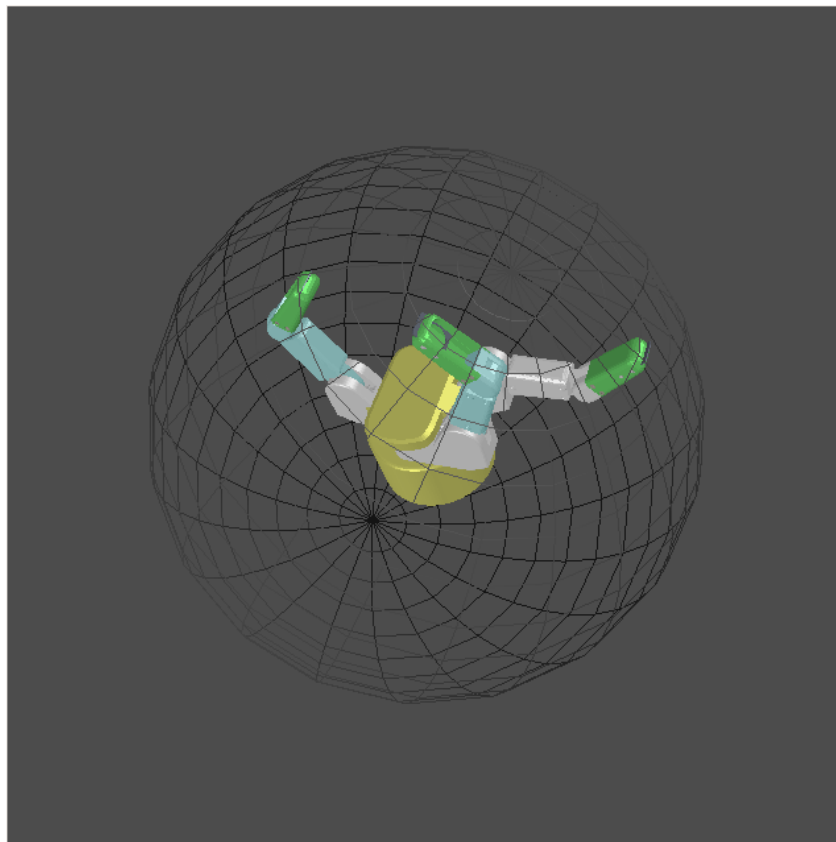


Figure 20 – Visual Tab – OpenGL Window Features

The OpenGL window can interact with the BarrettHand directly. There are two Supervisory buttons that allow the user to update the hand or the OpenGL model of the hand.

Go Supervisory:

This button will move the real-life BarrettHand to match the orientation of the BarrettHand OpenGL model.

Refresh OpenGL:

This button will Refresh the OpenGL window to match the orientation of the real-life BarrettHand.

The BarrettHand can also be controlled in the Visual tab while the OpenGL model updates in real-time. This mode is activated by clicking on the *Start RTControl* toggle button. The fingers are controlled in RealTime with position-mode PID controllers for fingers that compute desired control velocities. Spread is controlled in position mode by the microcontroller in the hand and is commanded to change position when the slider for spread is released or the right mouse button is released after the spread position has changed.

Start RTControl/Abort RTControl:

This button will first enter Real Time mode and move the real-life BarrettHand to match the orientation of the BarrettHand OpenGL model. Then, any movements made within the OpenGL model will be mimicked by the real-life BarrettHand. This button functions as a toggle button to start and abort RealTime control of the BarrettHand in the Visual tab. Parameters for RealTime control of the hand are updated when this toggle button is pressed to start RealTime control.

RealTime Control Parameters:

The user may set proportional and derivative (PD) gains for the position controller that is used to control the fingers. For the 262 hand RTGain is a proportional control gain for the hand motors that amplifies the actual control velocity. LCV (loop control velocity) is also a proportional control gain that will amplify the actual control velocity.

Visual mode:

In general, the hand should be controlled with velocities in the visual tab for smoothest movement. A 280 hand may be visually controlled in velocity, position, or torque mode. Reference positions for the model are submitted to another thread that executes with high priority so that the servo-rate is much higher than it would be otherwise. Position mode may have non-smooth movement because the model itself gets updated with slower mouse movements. Torque and position modes are mainly implemented to illustrate how the user may get started writing their own position or torque controllers in a separate thread.

RealTime Control Sliders:

Dragging these sliders will update the positions of the fingers in RealTime or give the spread a new position to go to. These sliders will also be updated with position changes to the hand when dragging the fingers and spread in the OpenGL window.