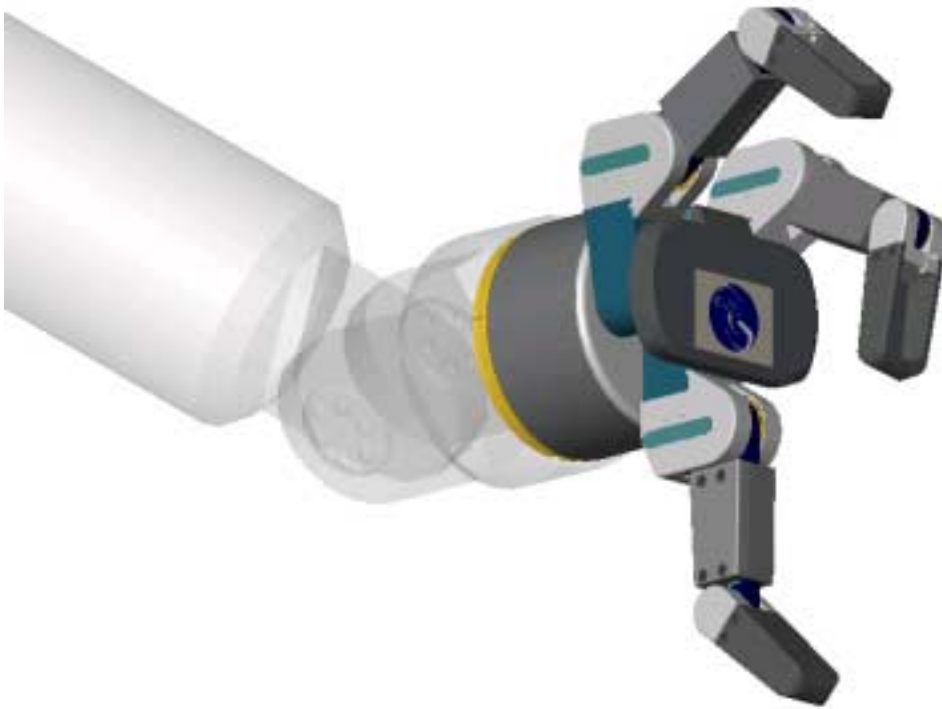


BarrettHand

BHControl Interface Manual



Barrett Technology Inc.

TABLE OF CONTENTS

LIST OF FIGURES	2
LIST OF TABLES	2
LIST OF EQUATIONS	2
<u>1 INTRODUCTION.....</u>	<u>3</u>
<u>2 CONFIGURE CONTROL WINDOW.....</u>	<u>5</u>
2.1 CONFIGURATION PANEL	6
2.2 TEST DELAYS PANEL.....	8
2.3 DOWNLOAD FIRMWARE PANEL.....	9
<u>3 SUPERVISE CONTROL WINDOW.....</u>	<u>10</u>
3.1 TERMINAL PANEL.....	11
3.2 CONTROL PANEL	12
<u>4 REALTIME CONTROL WINDOW.....</u>	<u>14</u>
4.1 REALTIME CONTROL PANEL.....	15
4.1.1 EXPRESSION SYNTAX	17
4.1.2 USING A DATA FILE	19
4.2 PLOT DATA PANEL	20
<u>5 VISUAL CONTROL WINDOW</u>	<u>22</u>
5.1 VISUAL FINGER SELECTION	23
5.2 VISUAL FINGER CONTROL.....	23
INDEX	25

LIST OF FIGURES

FIGURE 1 - BHCONTROL INTERFACE WINDOW	4
FIGURE 2 - CONFIGURE CONTROL WINDOW	5
FIGURE 3 - CONFIGURATION PANEL	6
FIGURE 4 - TEST DELAYS PANEL.....	9
FIGURE 5 - DOWNLOAD FIRMWARE PANEL	9
FIGURE 6 - SUPERVISE CONTROL WINDOW	10
FIGURE 7 - TERMINAL PANEL.....	11
FIGURE 8 - CONTROL PANEL	12
FIGURE 9 - REALTIME CONTROL WINDOW	14
FIGURE 10 - REALTIME CONTROL PANEL	15
FIGURE 11 - FILE DATA AS CONTROL VELOCITY	19
FIGURE 12 - PLOT DATA PANEL	20
FIGURE 13 - VISUAL CONTROL WINDOW	22
FIGURE 14 - FULL OPEN SLIDER.....	23
FIGURE 15 - MOVE TO POSITION SLIDER	24
FIGURE 16 - MOVE TO POSITION AND TRACK SLIDER	24
FIGURE 17 - TRACK VELOCITY SLIDER	24

LIST OF TABLES

TABLE 1 - GENERAL SETTINGS.....	6
TABLE 2 - COM PORT TIMEOUTS	7

LIST OF EQUATIONS

EQUATION 1 - REALTIME VELOCITY CONTROL.....	15
---	----

1 Introduction

BHControl is a Windows 95/98/NT application providing a graphical user interface (GUI) for control of the BarrettHand. It was developed using the Barrett Technology Inc. C-Function Library (Version 4.0) for all communication with the hardware. However, the C-Function Library is not required. The goal is to expose all the functionality provided by the hardware and the interface library in an easy-to-use graphical environment, without writing C code.

The functions are grouped into four control windows:

1. Configure – set communication parameters, test transmission delays, download firmware
2. Supervise – issue Supervisory mode commands, test command sequences, load previously saved command sequences and generate C++ code for standalone programs.
3. RealTime – issue RealTime mode commands, display feedback and control signals over time, test control laws, load previously saved control laws and generate C++ code for standalone programs.
4. Visual – control the fingers visually via customized sliders superimposed on a 3D image of the hand

The BHControl Interface functions like most other window-based programs. The large blue area at the top of the window serves as a title bar; you can move the window by pressing and holding the mouse anywhere in that area. The three small system buttons (top right) display a help file, minimize the window, and exit the program respectively, see Figure 1

Use the four larger buttons at the top right corner to switch between the different control panels, see Figure 1. When the BHControl Interface is first opened, only the *Configure* button will be visible. Once you have initialized the library, as described in Section 2, the other three buttons will become visible. You can activate the buttons in any sequence after the library has been initialized.

The program requires your monitor to be set to a resolution of at least 1024x768; otherwise, parts of the panel will be hidden. If you have changed the Windows color scheme from the default settings, the program restores the default dialog color settings during operation and restores customized colors on exit. However, while the program is running, the color in dialog boxes belonging to other applications will be affected.

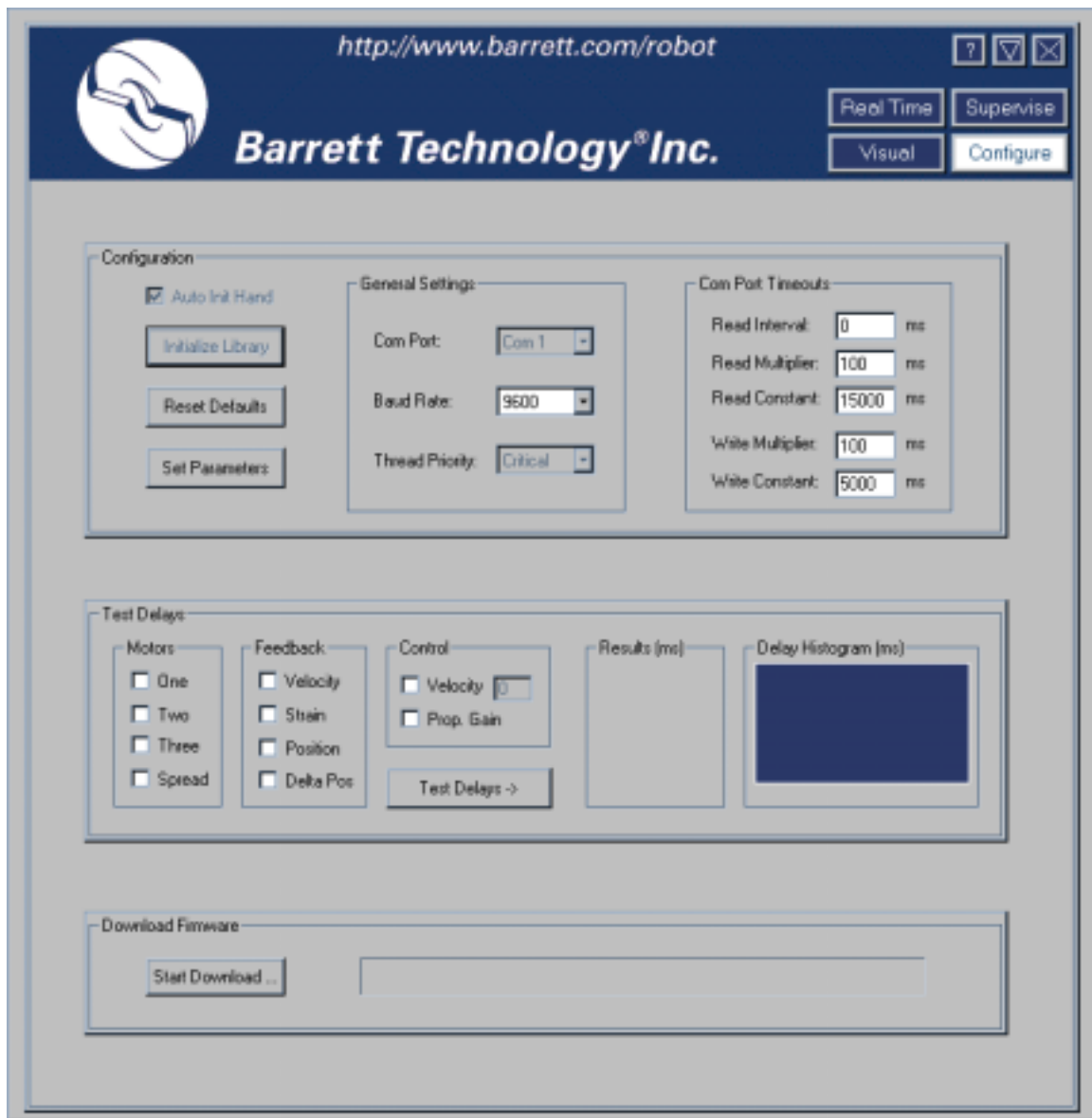


Figure 1 - BHControl Interface Window

2 Configure Control Window

This window is divided into 3 panels, see Figure 2:

1. Configuration Panel – set communication parameters and initializes the library.
2. Test Delays Panel – measure transmission delays for different parameter settings in RealTime mode.
3. Download Firmware Panel – download the firmware (.S19) file to the hand.

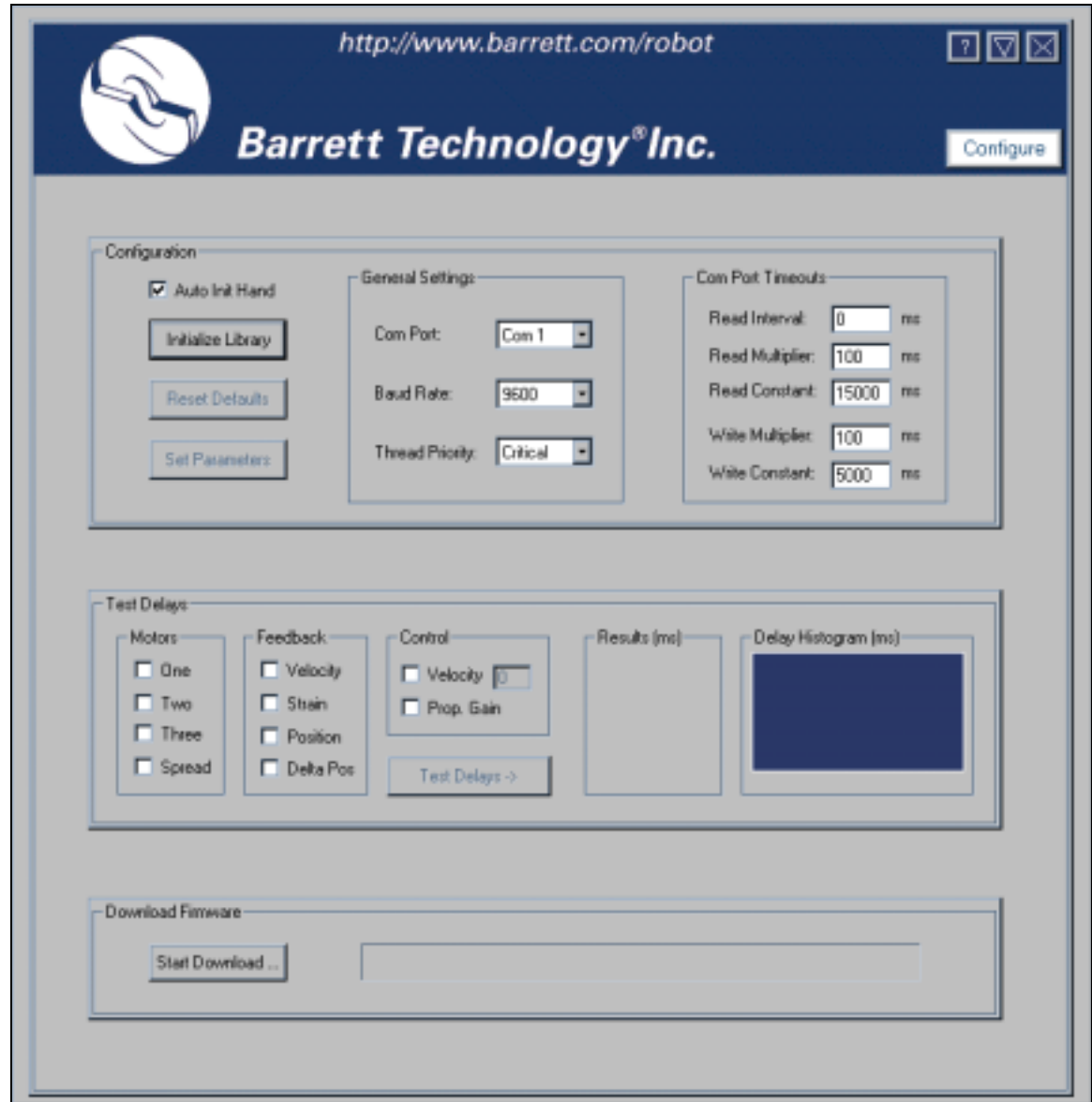


Figure 2 - Configure Control Window

2.1 Configuration Panel

The Configuration Panel seen in Figure 3 allows you to change the communication parameters, reset the default parameters and initialize the library. The BarrettHand must be plugged in and turned on before issuing any commands.

Configuration

☒ Auto Init Hand

Initialize Library

Reset Defaults

Set Parameters

General Settings

Com Port: Com 1

Baud Rate: 9600

Thread Priority: Critical

Com Port Timeouts

Read Interval: 0 ms

Read Multiplier: 100 ms

Read Constant: 15000 ms

Write Multiplier: 100 ms

Write Constant: 5000 ms

Figure 3 - Configuration Panel

The default parameters are displayed when the program starts and can be changed before initializing the library. The parameters listed in the *General Settings* allow you to change your communications port, set the communications speed and the low-level thread priority, see Table 1 for parameter details.

Table 1 - General Settings

Parameter	Description	Possible Values	Default Value
<i>Com Port</i>	Communications Port	1,2,3,4	Com 1
<i>Baud Rate</i>	Communications Rate	9600, 19200, 38400	9600 BPS
<i>Thread Priority</i>	Priority of low level thread	Critical, High, Normal, Low	Critical

The parameters listed in *Com Port Timeouts* determine the communications timeouts, see Table 2 for parameter details.

Table 2 - Com Port Timeouts

Parameter	Description	Default Value
<i>Read Interval</i>	Specifies the maximum time, in milliseconds, allowed to elapse between the arrival of two characters on the communications line. A value of zero indicates that timeouts are not used.	0 ms
<i>Read Multiplier</i>	Specifies the multiplier, in milliseconds, used to calculate the total timeout period for read operations. For each read operation, this value is multiplied by the requested number of bytes to be read.	100 ms
<i>Read Constant</i>	Specifies the constant, in milliseconds, used to calculate the total timeout period for read operations. For each read operation, this value is added to the product of the <i>Read Multiplier</i> and the requested number of bytes. A value of zero for both the <i>Read Multiplier</i> and <i>Read Constant</i> indicates that total timeouts are not used for read operations	15000 ms
<i>Write Multiplier</i>	Specifies the multiplier, in milliseconds, used to calculate the total timeout period for write operations. For each write operation, this value is multiplied by the number of bytes to be written.	100 ms
<i>Write Constant</i>	Specifies the constant, in milliseconds, used to calculate the total timeout period for write operations. For each write operation, this value is added to the product of the <i>Write Multiplier</i> and the number of bytes to be written. A value of zero for both the <i>Write Multiplier</i> and <i>Write Constant</i> indicates that total timeouts are not used for write operations.	5000 ms

Note: Barrett recommends using the default Com Port Timeouts unless you are familiar with serial port communications. The *Thread Priority* should not be changed unless you are familiar with the Windows multithreading mechanism.

There are two types of initialization: library initialization and BarrettHand initialization. If you would like the BarrettHand to be automatically initialized after initializing the library, verify the *Auto Init Hand* check box is checked before

pressing the *Initialize Library* button. After all parameters have been set, press the *Initialize Library* button. This will initialize all communications port parameters, reset all internal variables, clear all buffers and start the low-level serial communications thread. If the *Auto Init Hand* check box is checked it will initialize the BarrettHand. After initialization, the *Reset Default* and *Set Parameters* buttons will become active. It is possible to restore all default parameters by pressing the *Reset* button. To make further changes, edit the corresponding text field, and press the *Set Parameters* button. The *Thread Priority* and *Com Port* dialog boxes can not be modified after the library has been initialized.

If the *Auto Init Hand* check box is unchecked, the BarrettHand will need to be initialized before issuing motor commands.

After initializing the library, the Supervise, RealTime and Visual Control Panel buttons will become visible.

2.2 Test Delays Panel

This panel, shown in Figure 4, will test the communication delay between sending a control block and receiving feedback information. This delay is measured from the instant the command is issued on the host computer to the time the feedback data is available to the user. Following is a description of the different control and feedback sections:

Motors - Select which motors will be controlled and have feedback data returned.

Feedback - Select the type of desired feedback. Refer to the C-Function Library Manual for more information on feedback parameters.

Control - Select the type of desired control. If the *Velocity* check box is checked, fill in the desired velocity. Check the *Proportional Gain* check box to send proportional gain. The BarrettHand will move according to the control law described in Equation 1 in Section 4.1.

Results - Displays how many characters are sent (*Out*) and received (*In*), average delay time (*Average*), minimum delay time (*Minimum*) and maximum delay time (*Maximum*). The delay times include CPU processing necessary to initiate the transmission, hand processing delays and serial communication delays in both directions.

Delay Histogram - Shows the distribution of delay times.

Select the desired motors, control and feedback data and press the *Test Delays* button. The program will start a RealTime loop with the hand, sending the

specified control blocks for all activated motors, and receiving specified the feedback values. The loop runs for 100 repetitions, and then displays statistics for the delay times measured. See Figure 4 for an example of results from Test Delays.

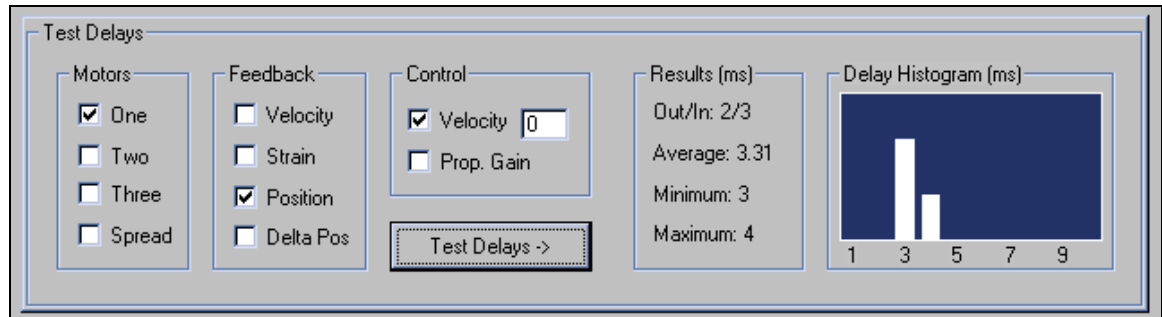


Figure 4 - Test Delays Panel

2.3 Download Firmware Panel

During library initialization, the program will show an error message if the hand has an outdated or invalid .S19 file. If an older version of the firmware is detected or the memory on the hand has decayed, it will be necessary to download a more recent version using the Download Firmware Panel.

Press the *Start Download* button to begin the download process, see Figure 5. Locate and select the appropriate .S19 file. For information on specific firmware files, refer to the User Manual. If the status bar prints the message "HIT RESET BUTTON TO START DOWNLOAD", press the red *Reset* button on the back of the Power Supply to start downloading code. Otherwise, the code will begin to download automatically. The download is finished when the status bar fills completely and then clears.

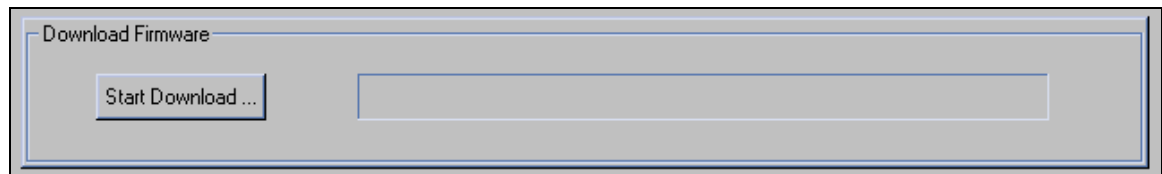


Figure 5 - Download Firmware Panel

If the library was not initialized before downloading the code, press the *Initialize Library* button. After downloading firmware, it will be necessary to initialize the BarrettHand in the Supervisory Window, before issuing motion commands.

3 Supervise Control Window

This window is divided into 2 parts (see Figure 6):

1. Terminal Panel – emulate an ASCII terminal sending commands to the BarrettHand, show corresponding C++ function calls, enable saving and loading of control sequences and generate C++ code for standalone programs
2. Control Panel – allow the user to issue all possible commands by pressing buttons and allow hand parameters to be set and retrieved

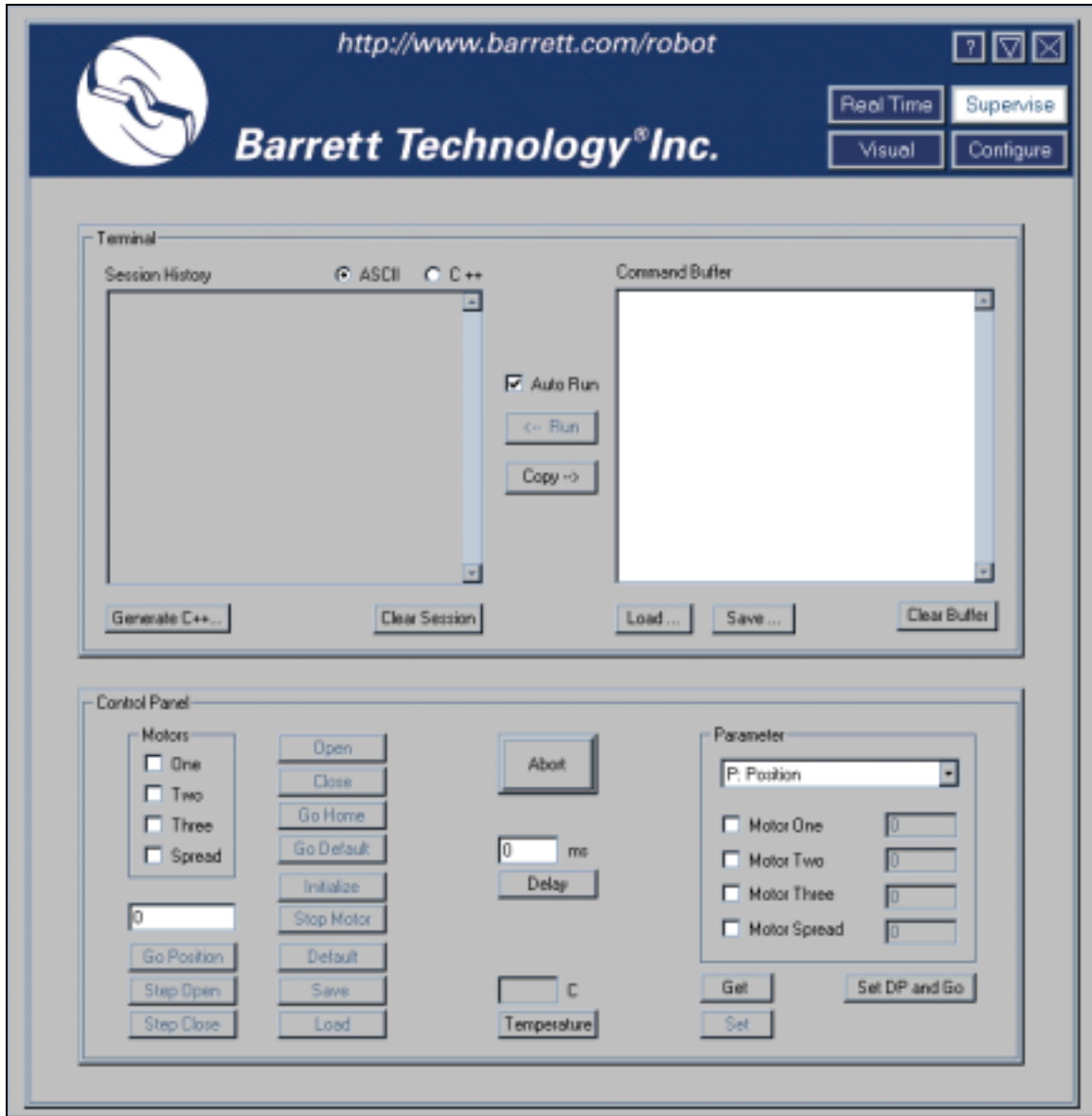


Figure 6 - Supervise Control Window

3.1 Terminal Panel

The Terminal Panel of the Supervise Control Window is used to send commands to the BarrettHand and view the response, see Figure 7.

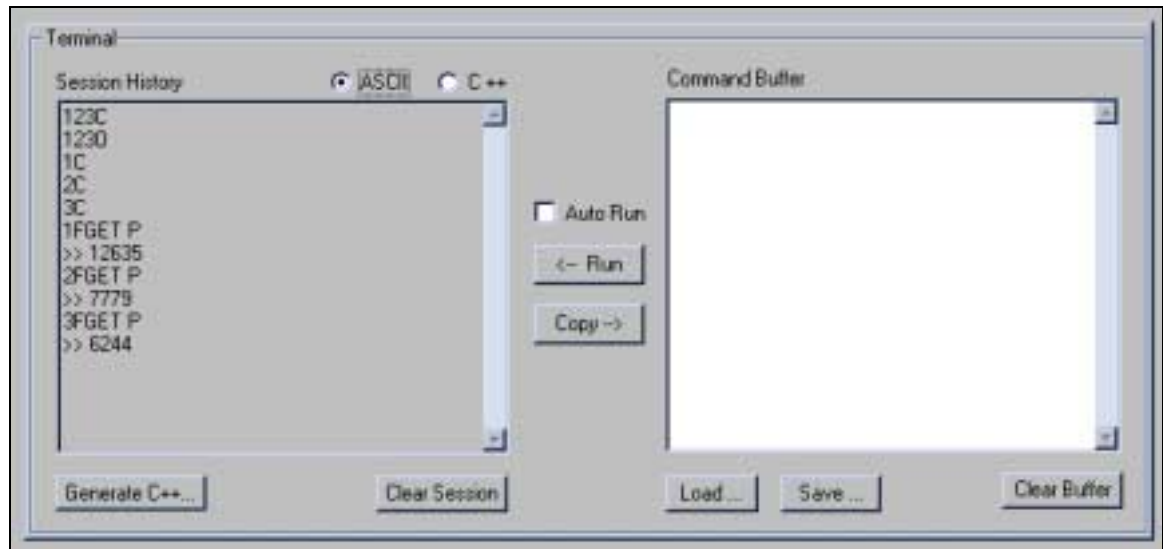


Figure 7 - Terminal Panel

This panel functions in two different modes depending on the setting of the *Auto Run* check box. When checked, pressing buttons in the Control Panel or typing a command string into the *Command Buffer* and pressing <Enter>, will cause immediate execution. For a list of valid ASCII control sequences see the BarrettHand User Manual. When *AutoRun* is not checked, the program accumulates commands in the *Command Buffer* window. You can execute the commands by pressing the <-Run button. The *Command Buffer* window can be cleared using the *Clear Buffer* button.

The *Session History* window displays the commands issued and the responses from the BarrettHand. Choosing the *ASCII* radio button will display the BarrettHand ASCII commands issued. Choosing the *C++* radio button will display the C++ code used to send the BarrettHand commands. The code displayed contains the functions available in the C-Function Library. The *Session History* window can be cleared using the *Clear Session* button.

The *Copy* → button can be used to copy the selected commands in the *Session History* into the *Command Buffer*. The *Copy* → button will not copy the responses from the BarrettHand. Commands can also be copied and pasted from the *Session History* to the *Command Buffer* window using <CTRL-C> and <CTRL-V>, respectively. The commands will be executed after the <-Run button is pressed or the <Enter> key is pressed within the *Command Buffer* window.

Sequences of ASCII command strings can also be saved into files and loaded. The *Save...* button will write the contents of the *Command Buffer* into a file, the *Load...* button will replace the *Command Buffer* with the contents of the specified file.

The *Generate C++...* button will generate a C++ program that contains the sequence of C++ commands appearing in the *Session History* window, when the C++ radio button is selected. The program generated will also add error control and the necessary initialization parameters. It is possible to edit the *Session History* by adding or removing commands before pressing the *Generate C++* button. You can then compile and link the autogenerated program with the C-Function Library, *BHand.lib*, to obtain an executable which is equivalent to issuing the sequence of ASCII commands.

While running a sequence of commands, the user can interrupt the program at any time by pressing the *Abort* button in the lower Control Panel.

3.2 Control Panel

The Control Panel provides an easy method of controlling the BarrettHand, see Figure 8.

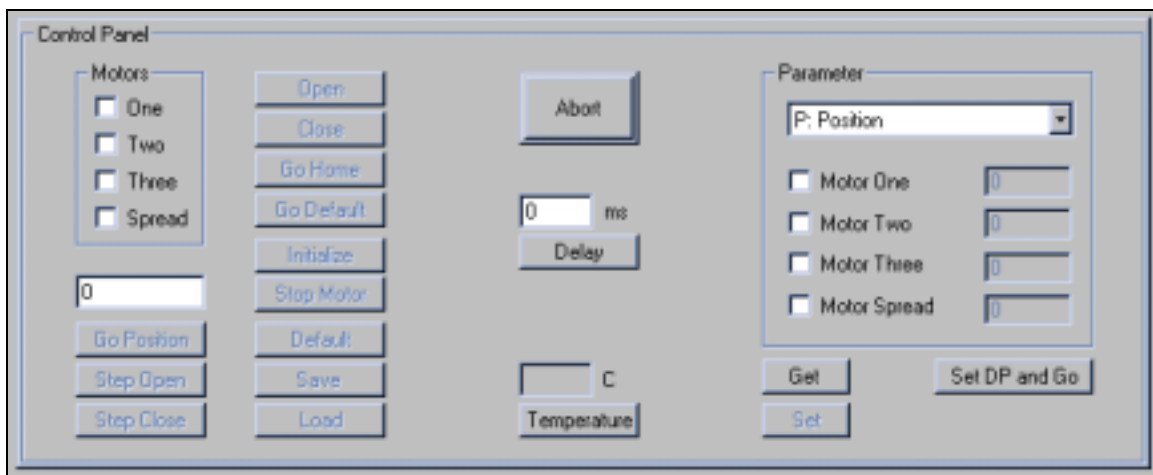


Figure 8 - Control Panel

The Control Panel contains all the buttons corresponding to Supervisory mode commands provided by the C-Function 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.

Following is a list of the buttons and their functions:

Button	Function
<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>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>Initialize</i>	Initializes the selected motors.
<i>Stop Motor</i>	Idle the selected motors.
<i>Default</i>	Loads the default parameters stored in the EEPROM for the selected motors.
<i>Save</i>	Saves the active parameters to EEPROM for the selected motors.
<i>Load</i>	Loads the parameters from the EEPROM into the active parameter list for the selected motors.
<i>Go Position</i>	Moves the selected motors to the position indicated in the above text box.
<i>Step Open</i>	Moves the selected motors in the open direction by the number of encoder counts in the above text box.
<i>Step Close</i>	Moves the selected motors in the close direction by the number of encoder counts in the above text box.

For a more in depth description of the commands, see the BarrettHand User Manual.

The middle column contains the *Abort*, *Delay* and *Temperature* buttons. The *Abort* button is used to abort a command being issued to the BarrettHand. This button can be pressed even if the mouse pointer is a wait cursor. The *Delay* button will insert a delay in the command sequence equal to the number of milliseconds in the above text box. The *Temperature* button will show the present temperature on the hand electronics boards in degrees Celsius.

The right group of buttons provides access to all hand parameters. Choose the parameter of interest from the pull down menu and specify the desired motor by checking the appropriate Motor check box. Press the *Get* button to get the value of the parameter displayed in the pull down menu. If the parameter can be set, the *Set* button will be enabled. Enter the desired value of the parameter in the corresponding motor text box and press the *Set* button.

The *Set DP and Go* button is used to move each motor to a different position simultaneously. First, select the parameter DP from the pull down menu and the desired motors to control. Type in the desired motor position for each motor. Press the *Set DP and Go* to move each of the selected motors to the position specified.

Note that *Get* and *Set* parameter are always executed immediately, even if *AutoRun* is not checked. The results are updated in the edit boxes, and the command sequences will appear in the *Session History* window.

4 RealTime Control Window

This panel is divided into 2 parts (see Figure 9):

1. RealTime Control Panel – issue RealTime mode commands, test control laws, load previously saved control laws and generate C++ code for standalone programs.
2. Plot Data Panel – plot the feedback and control data and save plots to files

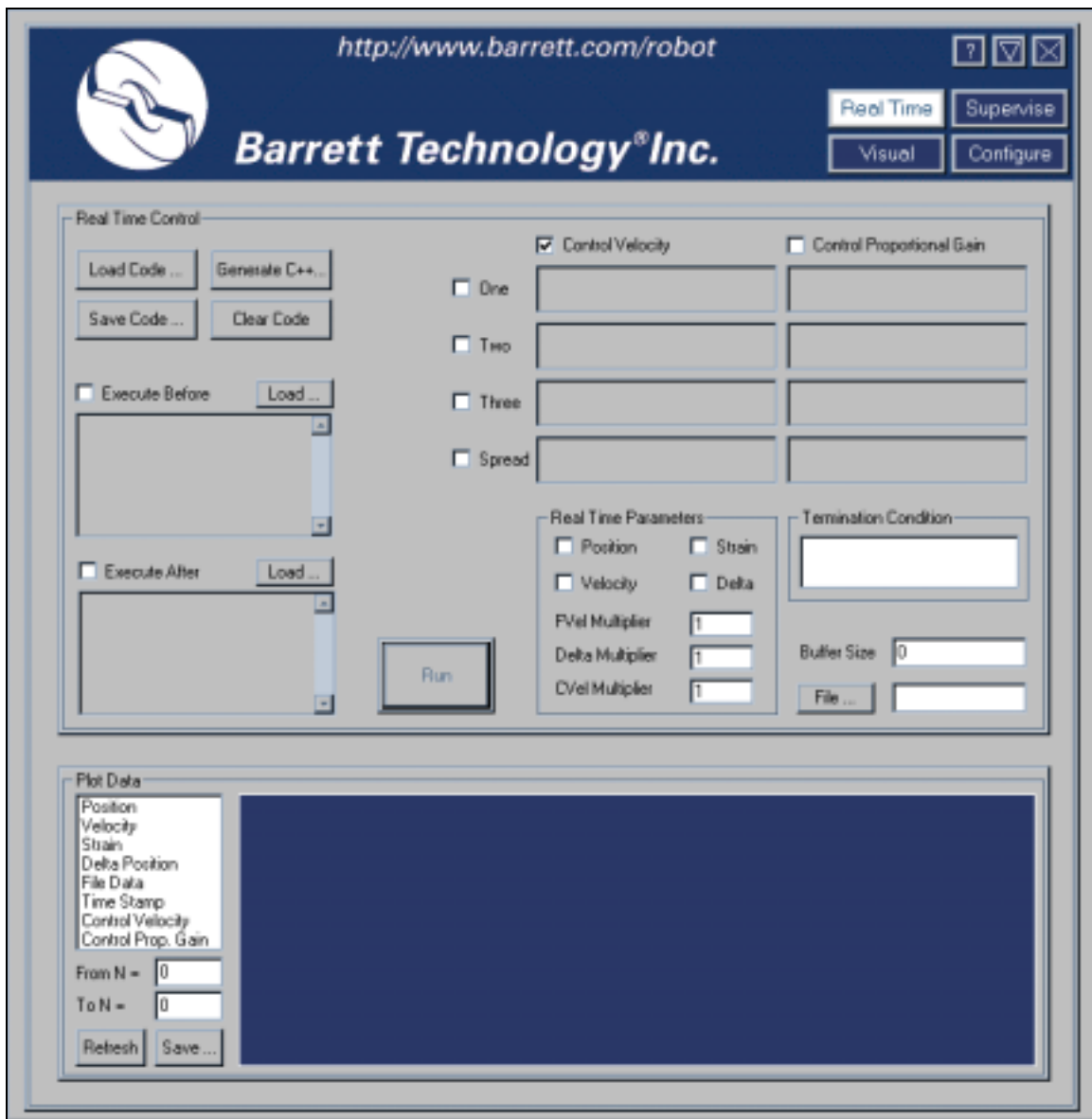


Figure 9 - RealTime Control Window

4.1 RealTime Control Panel

The RealTime Control Panel is used to create control laws for the fingers and spread, see Figure 10. RealTime control is useful when immediate control of the motor velocity is desired. Unlike the Supervisory mode, the RealTime mode allows you to change the velocity while the motors are still moving.

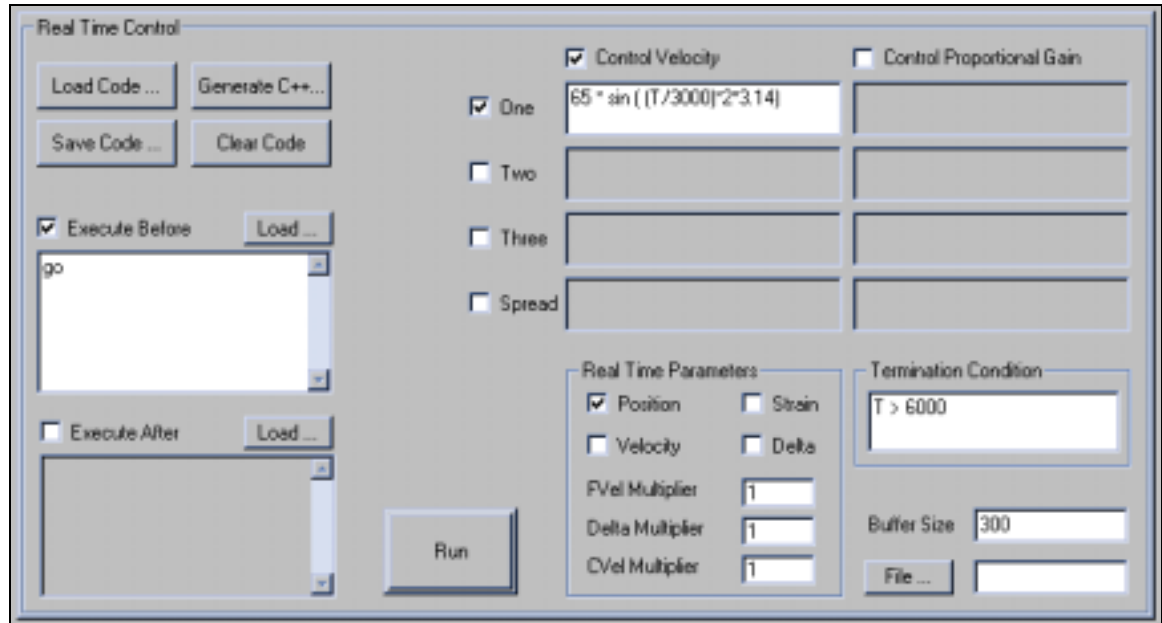


Figure 10 - RealTime Control Panel

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, or numeric sequences loaded from a file, see Section 4.1.1. The velocity of the motor will respond according to the control law in Equation 1.

$$MC_n = \left(\frac{K}{4} \right) * Y_n$$

Equation 1 - RealTime 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* from expression * *CVel Multiplier* - Actual Velocity).

Select the desired feedback parameters and multipliers in the RealTime Parameters section. Following is a list of the parameters and their definitions:

Parameter	Definition
<i>Position</i>	Motor Position Feedback
<i>Velocity</i>	Motor Velocity Feedback
<i>Strain</i>	Finger Strain Gage Feedback
<i>Delta</i>	Motor Delta Position Feedback
<i>FVel Multiplier</i>	The hand divides the Actual Velocity by <i>FVel Multiplier</i> before sending the <i>Velocity Feedback</i> to the host computer. (<i>Velocity Feedback</i> = Actual Velocity / <i>FVel Multiplier</i>)
<i>Delta Multiplier</i>	The hand divides the Actual Delta Position by <i>Delta Multiplier</i> before sending the <i>Delta Position Feedback</i> to the host computer. (<i>Delta Position Feedback</i> = Actual Delta Position / <i>Delta Multiplier</i>)
<i>CVel Multiplier</i>	After receiving the <i>Control Velocity</i> , the hand multiplies the <i>Control Velocity</i> by <i>CVel Multiplier</i> to determine the Command Velocity to control the hand. (<i>Control Velocity</i> * <i>CVel Multiplier</i> = Command Velocity)

Fill in a *Termination Condition* expression in field provided. When this expression evaluates TRUE, the RealTime control will stop. The expressions can include feedback variables, time, buffer limitations, or numeric sequences loaded from a file, see Section 4.1.1.

If data from a file is being used in the expressions for controlling motors, use the *File...* button to select the file. See Section 4.1.2 for information on the file format.

You can define a *Buffer Size* which allocates a circular buffer that saves all control and feedback variables. This data can later be saved to disk for further analysis, or plotted using the bottom panel, see Section 4.2. A data sample is recorded after each control block is sent. The size of the buffer is dependent on the duration of the RealTime control and the delay time of each control block. The duration of the RealTime control is dependent on the *Termination Condition* expression. The delay time is determined by size of the control and feedback blocks. The size of the buffer can be estimated by dividing the estimated duration of the RealTime control by the time delay measured in Test Delays Panel, see Section 2.2.

The two text fields on the left side of the RealTime Control Panel, *Execute Before* and *Execute After*, when activated, should contain sequences of BarrettHand ASCII commands to be sent to the hand. The commands in the *Execute Before* text field will be executed before the RealTime control is started. The commands in the *Execute After* text field will be executed after the RealTime control is

finished. It is possible to type the commands directly into the text fields, or they can be loaded from files created in the Supervise Control Window. See the BarrettHand User Manual for more information on specific BarrettHand commands.

When all of the desired fields have been filled and feedback selected, press the *Run* button. If there are any errors with the expressions or required fields are not entered, an error message will appear and the cursor will be placed at the first encountered error. At each time step the program evaluates the enabled expressions to obtain desired control signals, sends them to the hand, and reads the selected feedback variables that have been activated. The *Termination Condition* (always required) is evaluated at each time step, and the RealTime control aborts when evaluates TRUE. It is possible to terminate the program by pressing the *Abort* button (the *Run* button becomes an *Abort* button when the RealTime control is running).

The combination of settings in the RealTime Control panel defines a user “program” that a built-in interpreter executes in real-time to control the hand. The settings can be saved and loaded from files (with extension .RT) using the *Save Code...* and *Load Code...* buttons. Clear all of the parameters by using the *Clear Code* button. The *Generate C++...* button will translate the program into a C++ program the user can compile and link with the C-Function Library.

4.1.1 Expression Syntax

The expressions used for RealTime control follow a C-style syntax. The program will return an error if the syntax is incorrect, show some informative description of the problem, and position the cursor where the syntax error was encountered. 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 result of the evaluation is rounded to the nearest integer.

Variables:

D – delta position

F – file data (0 if no file is opened)

N – update number

P – position

S – strain

T – time (in ms) since the beginning of RT control

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 used with no subscript it will default to the corresponding motor number. For example, if P is used in Motor 2 *Velocity Control* expression field then it refers to finger 2. If used in the Termination Condition without a subscript, all variables default to finger one.

Functions:

abs(x)	Returns the absolute value of x.
acos(x)	Returns the Arccosine of x.
asin(x)	Returns the Arcsine of x.
atan(x)	Returns the Arctangent 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 less than x.
log(x)	Returns the Log, base 10, of x
random(N)	Generates a uniformly distributed real number between zero and N.
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.

All trigonometric functions are in radians.

Arithmetic Operators (in order of precedence):

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

Use parenthesis, (), to change precedence.

Logical Operators (in order of precedence):

!	Not
&&	And
	Or

Use parenthesis, (), to change precedence.

Comparison Operators (read from left to right):

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

< Less than
> Greater than
== Equal to

Conditional Expression Operator:

expression ? value1 : value2 - if the expression is true, the statement evaluates to value1, otherwise to value2.

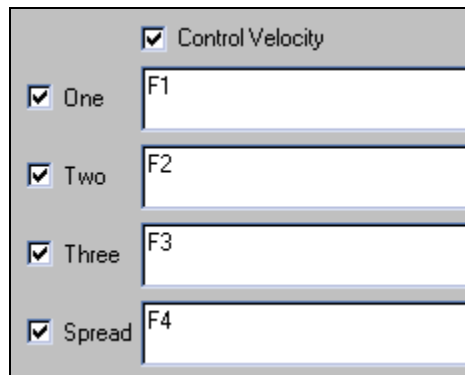
4.1.2 Using a data file

The user can specify a file with data to be used in expressions by using the *File...* button. The file selected 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, etc. If the last row is reached and the loop is still running, it rewinds and starts from the first row. Following 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 expressions according to Figure 11.



	Control Velocity
<input checked="" type="checkbox"/> One	F1
<input checked="" type="checkbox"/> Two	F2
<input checked="" type="checkbox"/> Three	F3
<input checked="" type="checkbox"/> Spread	F4

Figure 11 - File Data as Control Velocity

Each time control velocity is sent to the BarrettHand the values are 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

4.2 Plot Data Panel

It is possible to plot control and feedback data using this panel of the RealTime Control Window. To plot data, you must enter a non-zero number in the *Buffer Size* text field located in the RealTime Control Panel. This value is the number of update time steps to be recorded. The buffer is circular, meaning when the buffer is full, it will begin overwriting data at the beginning of the buffer. Fill in the desired control parameters in the RealTime Control Panel and Run the program, see Section 4.1 for a detailed description.

Select the desired parameters to plot by highlighting the parameters in the visible menu. Select the desired range to plot by inserting a value for *From N* and *To N*, as the first and last buffer locations respectively. Press the *Refresh* button. You will see as many columns of plots as the number of motors activated, and as many rows as the number of variables chosen from the list on the left. Figure 12 shows the plot of feedback position for one motor. When you move the mouse over a plot, a sign will appear in the top left corner indicating the variable you are viewing and where the cursor coordinates are.

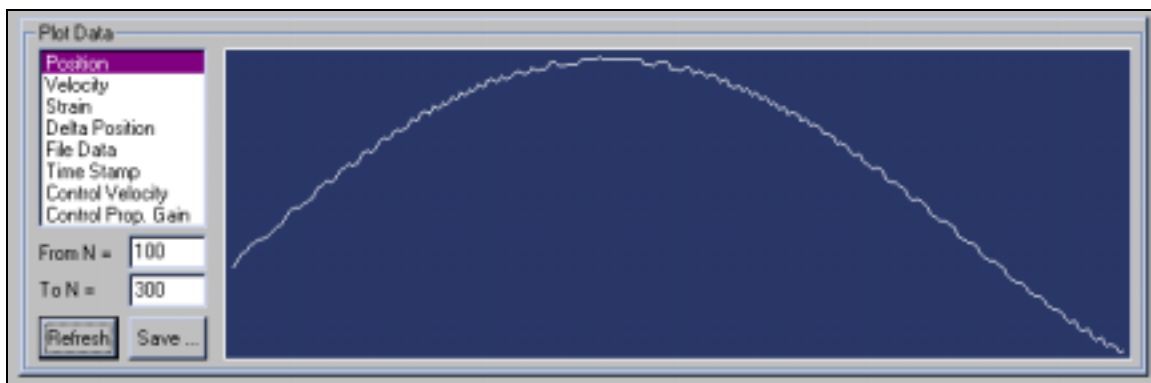


Figure 12 - Plot Data Panel

Pressing the *Save...* button will save all variables into a file. There will be four columns for each parameter selected which corresponds to the four motors. Inactive motors will be assigned the value 0.

5 Visual Control Window

This control panel allows the user to control the fingers and the spread motion of the BarrettHand by moving the mouse across an image. This panel shows a 3D image of the hand and a slider on each finger and spread, see Figure 13. There is only one slider corresponding to Spread, because that is a coupled degree-of-freedom. The spread slider is shown at the base of finger ONE only, but affects both fingers ONE and TWO. The sliders have a mark inside them showing the current position for each motor, which is updated after every operation.

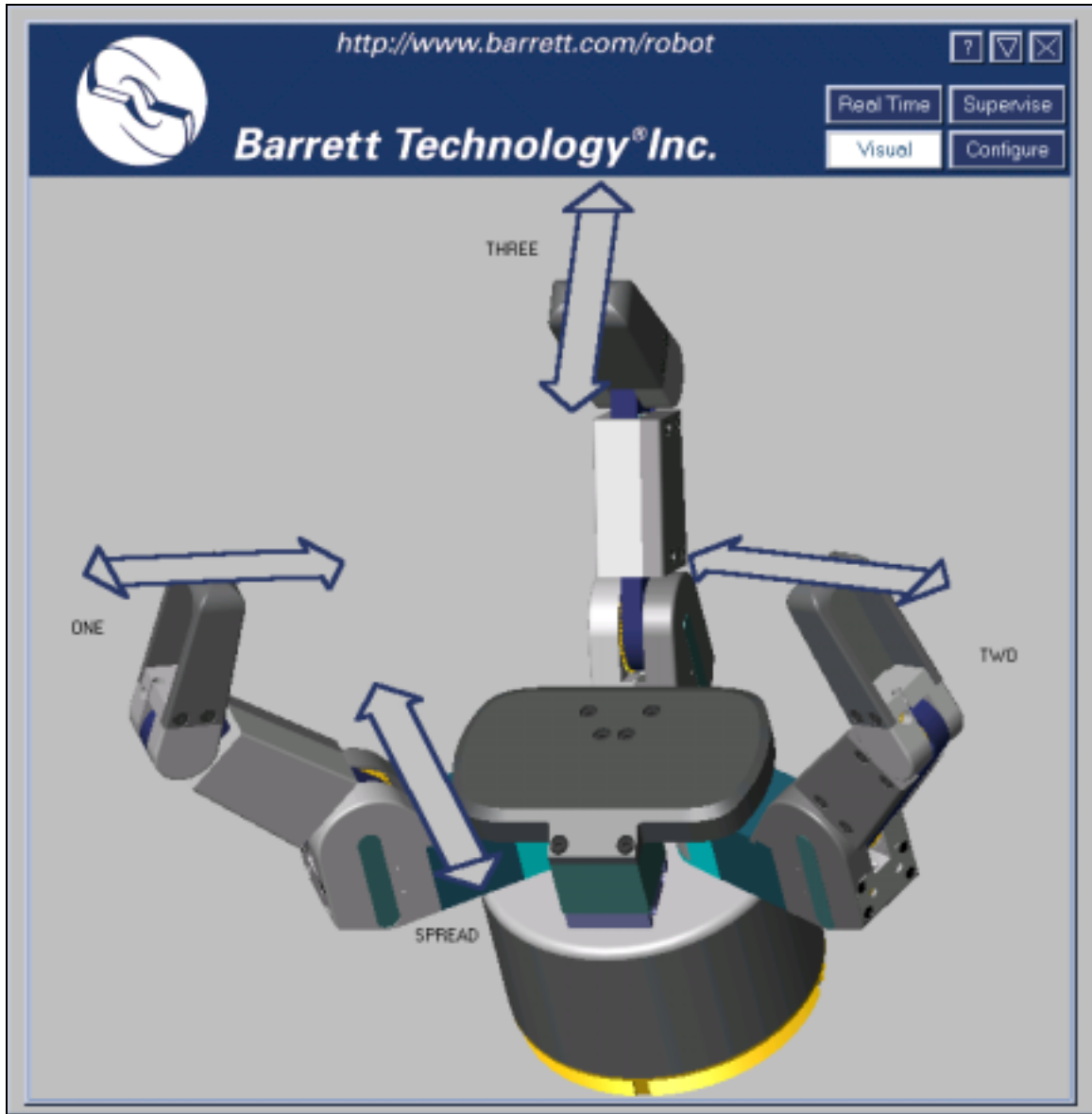


Figure 13 - Visual Control Window

The sliders have three color states:

- Gray (empty) – not selected, no action
- Red – the motor will be activated if you press a mouse button
- Green – the motor is currently activated

This control panel tracks the mouse continuously. When the mouse enters the sensitive area around any slider, it turns the slider red (i.e. selected and ready to be activated by a mouse click). The panel provides position control using the Left mouse button, and velocity control using the Right mouse button. You can control one finger at a time or select any combination of motors.

5.1 Visual Finger Selection

Commands apply to the finger under the mouse cursor (colored in red), allowing control of a single finger at a time. To control more than one finger simultaneously, additional fingers must be selected by moving the cursor over them and pressing Shift+Left mouse button. The selected finger will then remain red even after the mouse moves. The selected fingers will now move in unison. To deselect the fingers, press Shift+Left mouse button over that finger again. Note that at least one motor must be under the cursor to activate the hand, i.e. selecting a finger and moving away from it is not sufficient to issue control commands.

It is important to remember the fingers and the spread motors are geared differently. A cursor movement along the slider for the spread will result in a much larger motion than an equal cursor movement on a finger slider.

5.2 Visual Finger Control

It is possible to control the finger position in four different modes.

1. Open/Close – Left click on one of the arrows at the end of a slider (they become yellow when the cursor is positioned over them). The fingers or spread will open or close completely. See Figure 14



Figure 14 - Full Open Slider

2. MoveTo – Left click anywhere on the slider. The fingers will move to the position clicked on. See Figure 15.

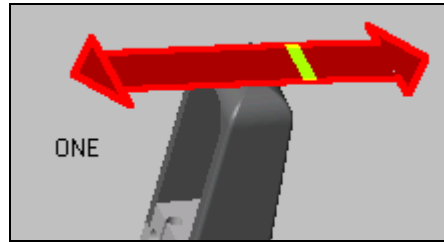


Figure 15 - Move to Position Slider

3. MoveTo and Track – Left click on the slider and hold. The corresponding finger(s) will first move to the specified position. The program will then enter a real-time loop, where the mouse position is sampled continuously, the difference between the current and desired positions is converted to a velocity command (with an internally specified gain) and sent to the hand. To stop tracking, release the left mouse button. See Figure 16



Figure 16 - Move to Position and Track Slider

4. Velocity Track - Right click on the slider and hold. The program enters a real-time loop, continuously sampling the mouse position, subtracting the center of the slider from it, and using that as a velocity command. To specify zero velocity, hold the cursor in the center of the slider. To move the finger at its maximum velocity, hold the cursor at the end of the slider. Release the button to end tracking.

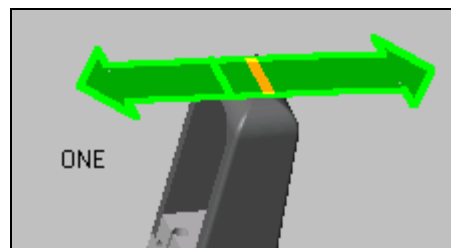


Figure 17 - Track Velocity Slider

INDEX

A

Arithmetic Operators	18
Addition	18
Division	18
Exponentiation	18
Modulus	18
Multiplication	18
Subtraction	18
ASCII terminal	10

B

Baud rate.....	6
Buffer.....	16
Buffer Size.....	16, 20
Buttons	
Abort.....	12, 13, 17
ASCII.....	11
Auto Init Hand	7, 8
Auto Run.....	11
C++	11
Clear Buffer	11
Clear Code... ..	17
Clear Session	11
Close	12
Configure	3
Copy.....	11
Default	13
Delay.....	13
File... ..	16, 19
Generate C++... ..	12, 17
Get	13
Go Default	13
Go Home.....	13
Go Position	13
Initialize	13
Initialize Library	8, 9
Load.....	13
Load Code... ..	17
Load... ..	12
Open.....	12
Refresh.....	20
Reset	8
Run.....	11, 17
Save	13
Save Code... ..	17
Save... ..	11, 21
Set	13
Set DP and Go	13
Set Parameters	8
Start Download	9

Step Close	13
Step Open.....	13
Stop Motor	13
Temperature	13
Test Delays.....	8

C

C++ code generation	10
Command Buffer	11
Communication parameters	5, 6
Communications port.....	6, 8
Comparison Operators	18
'?' Conditional Expression Operator....	19
Equal to	19
Greater than.....	19
Greater than or equal to	18
Less than	18
Less than or equal to	18
Not equal to	18
Configuration panel	6
Configure	3
Control laws.....	14, 15
Control Panel	10, 12
Control parameters.....	15
Control Proportional Gain.....	15
Control Velocity	15, 19

D

Default parameters	6
Delays	
Test Delays.....	5
Control.....	8
Feedback	8
Histogram	8
Motors	8
Results	8
Download Firmware	5, 9

E

Execute After	16
Execute Before.....	16
Expression syntax	17

F

Feedback	8
Feedback parameters.....	16
Firmware.....	9
Download	5
S19	5, 9
From N.....	20
Functions.....	17, 18

abs(x)	18
acos(x)	18
asin(x)	18
atan(x)	18
ceil(x)	18
cos(x)	18
exp(x)	18
floor(x)	18
log(x)	18
random(N)	18
round(x)	18
sin(x)	18
sqrt(x)	18
tan(x)	18

G

General settings.....	6
Generate C++.....	10, 14

I

Initialize library.....	6
Initialize Library	3, 5

L

Logical Operators	18
And.....	18
Not	18
Or	18

M

Monitor	3
Multipliers.....	16

O

Operators.....	17
----------------	----

P

Plot Data	14, 20
Position control.....	23

R

RealTime.....	3
RealTime Control Panel.....	15
RealTime mode.....	14
RealTime Parameters.....	16
CVel Multiplier	16
Delta.....	16
Delta Multiplier.....	16
FVel Multiplier	16

Position.....	16
Strain	16
Velocity	16
Reset (red button).....	9

S

S19	5, 9
Save plots	14
Session History	11
Slider	22
Supervise.....	3
Supervisory mode	12
Syntax	17

T

Terminal	10, 11
Termination Condition	16, 17, 18
Test Delays.....	8
Thread Priority	6, 7, 8
Timeouts	6, 7
To N	20
Trigonometric functions.....	18

V

Variables	17
Delta Position	17
File data	17
Position.....	17
Strain	17
Time	17
Update number	17
Velocity	17
Velocity control.....	23
Visual	3
Visual Finger Control.....	23
MoveTo	24
MoveTo and Track	24
Open/Close	23
Velocity Track.....	24
Visual Finger Selection	23

W

Windows	
Configure.....	5
RealTime Control	14
Supervise	10
Visual Control	22