WAM™ Arm

User's Manual

Barrett Technology Inc.

Document: D1001
Version: AH.00

Table of Contents

List of Figures

List of Tables

List of Equations

1 System Description

1.1 Standard WAM* System Components

1.1.1 System Features

Thank you for choosing the Whole Arm Manipulator (WAM). The WAM is designed to overcome the lack of backdrivability in conventional robotic arms while maintaining durability, low power usage, light moving weight, and ease of use. The standard WAM is a four degree-of-freedom (4-DOF) arm. The optional WAM Wrist adds three more degrees of freedom – see Section 1.2.1. The WAM has the dexterity and slenderness to easily navigate around obstacles placed in its path and is endowed with smooth and precise joint motion, contributing to the WAM's popularity in robotics control research and emerging applications.

The compactness, low system weight, and extraordinarily low power consumption make the WAM arm uniquely portable and so an ideal choice for use with mobile platforms. Its low mass and the absence of a controller cabinet ensure that it is significantly easier to mount than robotic arms of the same size. The WAM can also be powered directly from small batteries on a mobile platform without power conditioning, accepting a wide range of voltages from batteries.

While no robotic arm should be considered "safe" and all precautions should be taken as each application demands, the backdrivability and the multiple-layer safety system of the WAM make it one of the safest robotic arms available. Because of its 95% backdrivable cable drives (in contrast to poorly-backdrivable gear and harmonic-based drives), the WAM can react naturally and lightly as it contacts walls and people. These attributes have contributed to the WAM's success as one of the first robotic arms in surgery and rehabilitation.

We hope that you enjoy the versatility and functionality of the WAM. Please never hesitate to give feedback and to ask for advice as needed. US+617-252-9000, support@barrett.com, or http://www.barrett.com/.

1.1.2 Documentation

The WAM comes with six separate pieces of documentation:

1. User's Manual (this manual)
2. Quick Start Guide
3. WAM Cable Maintenance Guide
4. Wrist Cable Maintenance Guide
5. Inertial Specifications Manual
6. Support Reference Sheet

1. The User's Manual (this manual) covers:
   - System components and options
   - Safety instructions
   - System setup and operation
   - Troubleshooting
   - Technical specifications
   - Frequently asked questions

2. The Quick Start Guide is a single page guide that covers the essentials of operating the WAM and some basic demonstrations for the WAM. This includes turning on and initializing the WAM, home position, fault recovery, and the gravity compensation and teach-and-play demos.

* "Whole Arm Manipulator" and "WAM" are trademarks of Barrett Technology® Inc.

3. The WAM Cable Maintenance Guide covers the cabling of joints 1 through 4 and the general rules for handling cables, and should be used in the event that a cable in the 4-DOF WAM is damaged or is coming loose.

4. The Wrist Cable Maintenance Guide covers the cabling of joints 5 and 6 and should be used with the introductory sections in the WAM Cable Maintenance Guide in the event that a cable in the Wrist is damaged

5. The Inertial Specifications Manual presents the inertial data for the WAM robotic arm (the 4-DOF version and both 7-DOF versions) in support of computed-torque control techniques.

6. The Support Reference Sheet lists the variety of support options Barrett provides and assists the customer in using the resource most likely to answer their question most effectively.

1.1.3 WAM Arm

The standard four-degree-of-freedom WAM Arm, shown in Figure 1, has four cable-driven joints labeled J1 through J4, and four brushless DC motors labeled M1 through M4. M1 controls the yaw (J1) of the WAM and is located in the base of the WAM. Using Barrett's patented cable differential, M2 and M3 together control the pitch (J2) and roll (J3) of the WAM, and are contained in the shoulder of the WAM. M4 controls the bend of the "elbow" (J4), and is located just above M2 in the shoulder. Using Barrett's Puck technology, all motor encoders, power amplifiers, and controllers are located in a single compact package adjacent to each motor, eliminating the need for a controller cabinet and heavy electrical cables.

Figure 1 - WAM Arm

The standard WAM Arm comes with an internal computer and a blank outer link (shown attached to the end of the elbow in Figure 1 and shown in full in Figure 2). However, several options (see

Section 1.2) such as an external computer and the WAM Wrist (Figure 11) are available to replace these (though both standard options will still be included so that you may always switch between configurations at any time).

Figure 2 - Blank Outer Link

The WAM's internal computer has the following specifications:
Mainboard: PFM-540I
Processor Type: AMD Geode LX800 x86 @ ~500MHz
Memory: 256MB (4MB of that is used as a video framebuffer).
Hard drive: 2GB, CompactFlash
Kernel: Linux 2.6.16.57 patched with Xenomai 2.4-rc5
Distro: Slax 5.1.8
Expansion Card: PEAK-System PC/104 CAN Card, 2 ports
Wireless: Wi-Fi 802.11 b/g
Ethernet: 10/100 Base-T

1.1.4 Tool-End Attachments

The WAM™ comes with three endpoint (tool-end) attachments that can be used to change the setup and shape of the WAM™ Arm: the Haptic Ball, the Tool Plate, and the CAN Termination. These items in general will be attached to either the Outer Link (Figure 2) or WAM Wrist (Figure 11). Figure 3 shows the end plates of both the Wrist and Outer Link. The wires and connector coming out of the centers are for running the optional Barrett Hand. If using the Outer Link *without* the Barrett Hand, the wiring can simply be pushed inside the large open cavity. If using the Wrist *without* the Barrett Hand, the wires should be coiled neatly in the small cavity and the Wrist Plug installed to keep the wires from getting pinched.

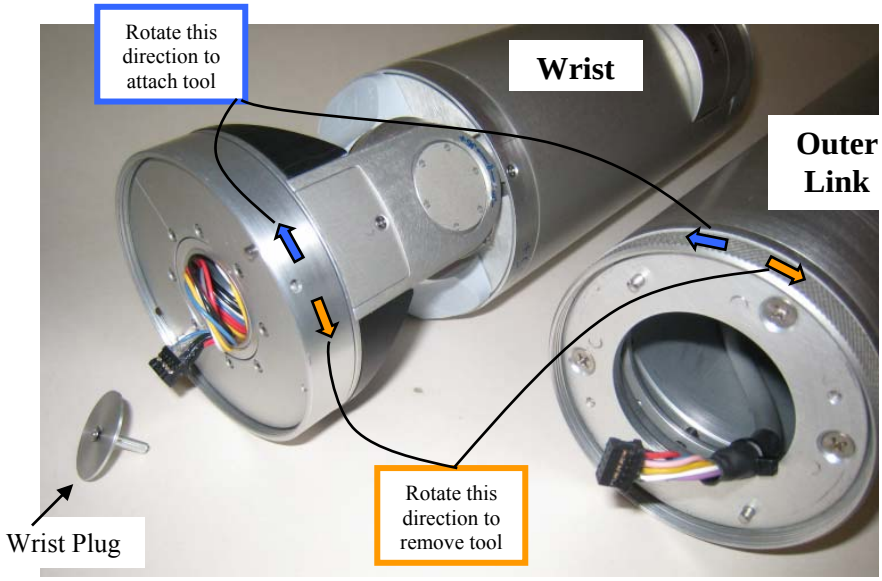

Figure 3 - End Plates of Wrist and Outer Link

The Haptic Ball (see Figure 4) can be attached to the end of the Outer Link or the Wrist. It is used to assist in haptic scenes by providing an easy-to-grasp, definable endpoint for the user. Systems are generally shipped with the Haptic Ball pre-installed.

The Tool Plate (see Figure 5) can be installed on the end of the Outer Link, Wrist, or Elbow Plate (if neither the Outer Link nor Wrist modules are installed). It provides a flat base that can be used to attach parts other than the ones designed by Barrett. The Tool Plate has four M6 tapped holes, evenly distributed in a circle, a 6-mm diameter dowel-pin hole, a quick-connect pin hole and slot, and two holes to connect the CAN Termination (see Appendix B for Tool Plate hole labels and dimensions).

The CAN Termination (see Figure 6) must be used if neither the Outer Link nor the Wrist is attached at the end of the elbow. The CANbus must be terminated at each end with a 120 Ohm resistor to minimize signal reflections. The termination can be attached to either the end plate of the elbow or to the Tool Plate (if attached to the elbow) by the two small holes, which are the same distance apart as the holes on the Termination.

Figure 4 – Haptic Ball

Figure 5 – Tool Plate

Figure 6 – CAN Termination

1.1.5 Power Supply

The WAM system generally requires an input voltage of 48 VDC. However, if you require an alternate input voltage, contact Barrett Technology. The power requirements for both 4-DOF and 7-DOF WAMs are summarized in Section 3.6.1. Any DC power sources may be connected directly to the WAM. Improper grounding can cause severe damage to the WAM electronics. If you are concerned that your voltage source may damage the WAM electronics, do not hesitate to contact Barrett for technical guidance.

The Power Supply that ships with the WAM, shown in Figure 7, can be plugged into any regular AC power source. It provides up to 10A of direct current at 48V. This Power Supply switches automatically to local voltage standards (100-120 & 200-240 VAC at 50-60Hz) around the globe and contains built-in surge protection.

Figure 7 – Power Supply

1.1.6 Safety Pendants

The WAM Arm system comes with two safety pendants: a control pendant and a display pendant (shown in Figure 8). Both pendants show the present safety status of the WAM Arm, with status lights for the velocity, torque, voltage, and heartbeats of the robot. There is also a 7-segment LED single-character display which shows additional information related to any existing errors. Each pendant has a large mushroom-type emergency stop button, which can be reset (popped up) by rotating the button clockwise for one quarter of a full turn. The control pendant has three additional buttons: Shift, Reset/Idle, and Activate.

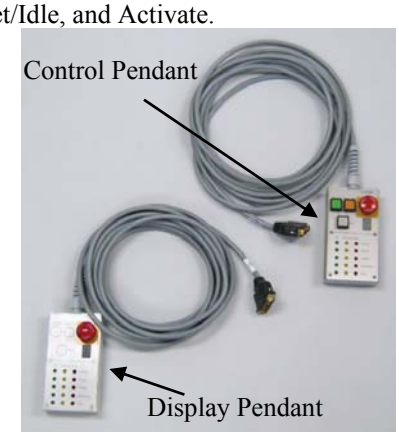

Figure 8 – Safety Pendants

1.1.7 Electrical Cables

All necessary electrical cables are included with the basic WAM System, shown in Figure 9. An AC Line Cord connects the Power Supply to a wall source. A blue DC Power Cable connects the Power Supply or another DC power source to the WAM. If you purchased an external WAM PC, a purple CANbus cable is provided for CAN communication with the WAM. The pendant cables (shown attached to the pendants in Figure 8) connect directly to the WAM. An Ethernet cable is also provided for the option of Ethernet communication with the WAM.

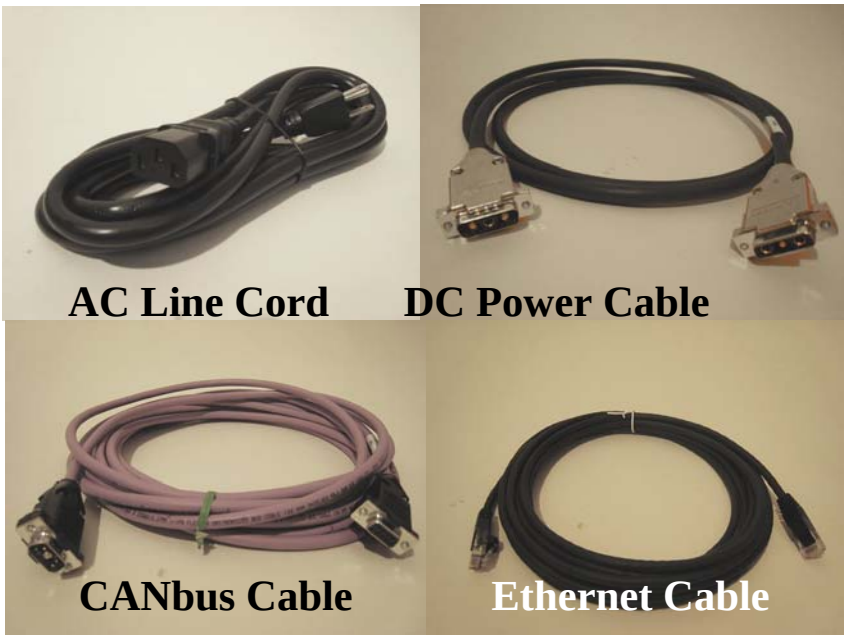

Figure 9 – Electrical Cables

1.1.8 Control Software and Firmware

The WAM's internal computer has firmware and software that are preloaded before it is shipped.

Included with a copy of the source code in electronic form are:
1. User's Manual (this manual)
2. Quick Start Guide
3. Cable Maintenance Guide
4. Inertial Specifications Manual
5. Support Resources Reference Sheet

For more information about software and firmware, please see Section 4.

1.1.9 Maintenance Kit

Included in each WAM package is a maintenance kit (shown in Figure 10). Use the maintenance kit in accordance with the instructions in Section 3 and the Cable Maintenance Guide. The maintenance kit includes the following:

- 1 Tool Plate (normally shipped attached to the Outer Link (or Wrist if purchased))
- 4 M10 Screws, Washers, and Nuts to secure the base
- 1 Set of Metric Hex Wrenches
- 1 Packet of Loctite 222
- 1 10-mm Combination Wrench

- 1 Pair of Tweezers
- 2 Clamps (for clamping the WAM to a table top)
- 1 Roll of 13-mm (1/2") Masking Tape
- 1 Roll of Kapton Tape
- 1 Slotted Screwdriver
- 1 Phillips-Head Screwdriver
- 1 Push-Pull Hook
- 1 Set Spare Mechanical Cables (Packaged separately from the rest of the Maintenance Kit)

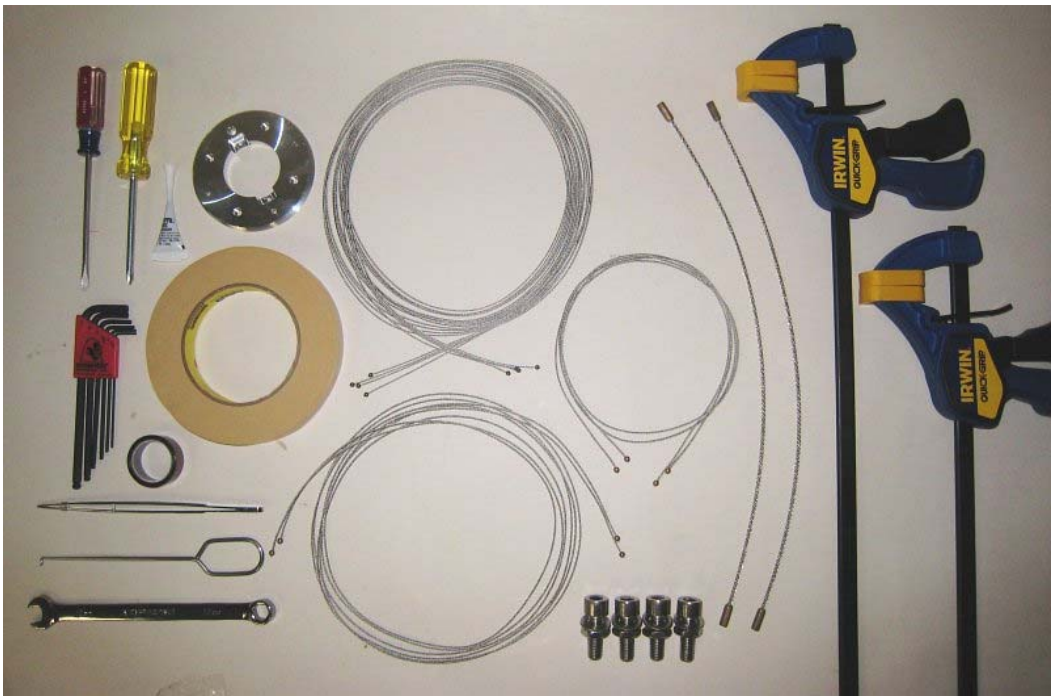

Figure 10 – Maintenance Kit

1.2 System Options

1.2.1 WAM Wrist

The WAM Wrist module, shown in Figure 11, replaces the outer link of a four-degree-of-freedom WAM, adding joints 5, 6, and 7, for another three degrees of freedom. Mechanical and electrical quick-connect features make it easy to swap between the WAM's outer link and the Wrist module to convert between 4-DOF and 7-DOF configurations.

In order to minimize the inertial effects of the motors on the host robot arm, the servomotors for joints 5 & 6 are located at the base of the wrist. The final roll joint in the WAM Wrist, motor joint 7, is the only geared axis. At very small scales, such as at joint 7, gears have slightly better performance than cables in terms of compactness.

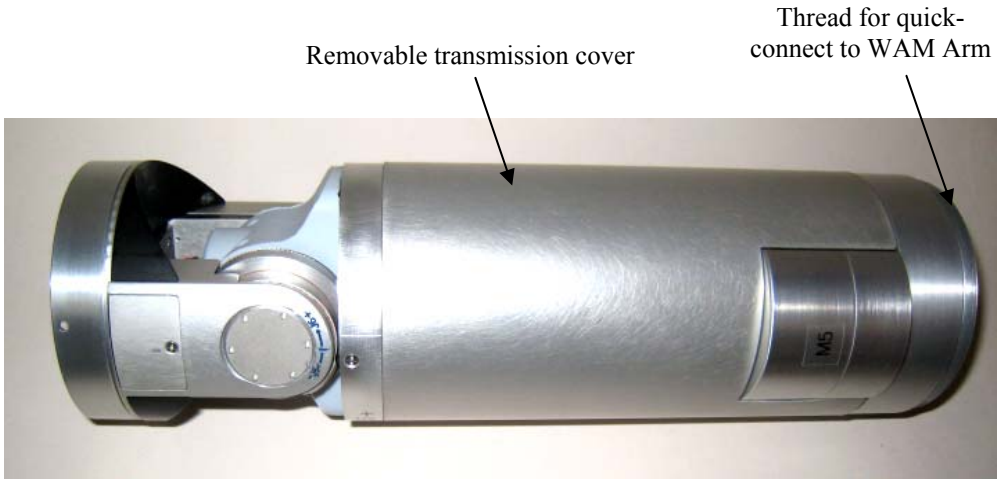

Figure 11 – WAM Wrist

The WAM Wrist comes with its own separate maintenance kit (shown in Figure 12), which contains:

- 1 Roll 6-mm (¼") wide Masking Tape
- 1 Wrist Tension Tool
- 1 Joint-7 Spanner Tool
- 1 Pull Hook
- 1 Wrist Safety Termination (Packaged separately from the rest of the Maintenance Kit)

Figure 12 – Wrist Maintenance Kit

1.2.2 Passive Gimbals

The passive gimbals option, shown in Figure 13, replaces the Outer Link on the WAM and adds three additional unpowered degrees of freedom. The links of the gimbals are designed to provide a large mounting area at the intersection point between all three joint axes. A vertical bar grip comes standard and is designed primarily for life-size haptics, exercise, and rehabilitation; but other grip and mounting options are available. The gimbals use Barrett Technology's quick-connect system so that it may easily be interchanged with other end effectors.

A 6-axis force/torque (FT) sensor can be mounted at the base of the gimbals grip for gathering additional data. This sensor is redundant with the built-in Cartesian Force/Torque control capabilities of the WAM, but enables a separate confirmation of values at slightly higher precision.

Figure 13 – Gimbals Option

1.2.3 External WAM PC

Barrett Technology offers an external PC (see Figure 14) for advanced control applications that require more processing power or more memory than the WAM's internal PC can provide. A standard keyboard and mouse, CANbus cable, and 2-m AC line cord are provided for use with the external PC. A monitor (not provided) is needed only during the initial setup of the computer. The external PC has the following specifications:

Mainboard: Intel 945G Micro ATX
Processor Type: Intel Core 2 Duo E6600 Conroe @ ~2.4GHz
Memory: 2GB (4MB of that is used as a video framebuffer).
Hard drive: 80GB
Optical: DVD Drive
Kernel: Linux 2.6.24.2 patched with Xenomai
Distro: Ubuntu 7.10
Expansion Card: PEAK-System PCI ISO dual-port CAN card
Ethernet: 10/100/1000 Base-T

Figure 14 – External WAM PC

Figure 15 – Back of External WAM PC

1.2.4 Control Software/Firmware Upgrades

Barrett Technology releases software and firmware upgrades periodically (see Section 4.5 for details on firmware upgrades). Upgrades are available free of charge for customers of Barrett's support subscription service. Customers with an expired support contracts can choose to renew their contract, or purchase the upgrades separately. See the enclosed Support Reference Sheet for more information about your support subscription service.

2 Safety and Cautions

2.1 Safety Instructions

PLEASE READ THIS SECTION IN ITS ENTIRETY BEFORE USING YOUR WAM.

Following these safety instructions will help prevent user injury and equipment damage.

- Proper precautions should be taken when selecting the location and setup of your WAM system. DO NOT set up the system such that any part of the robot's workspace (resembling a sphere with an approximately one meter radius, as shown in Figure 16) reaches into a pedestrian pathway in the lab space. The WAM is an unusually quiet mechanism – it generates little servo or transmission noise – thereby providing very little intrinsic warning of its enabled state.

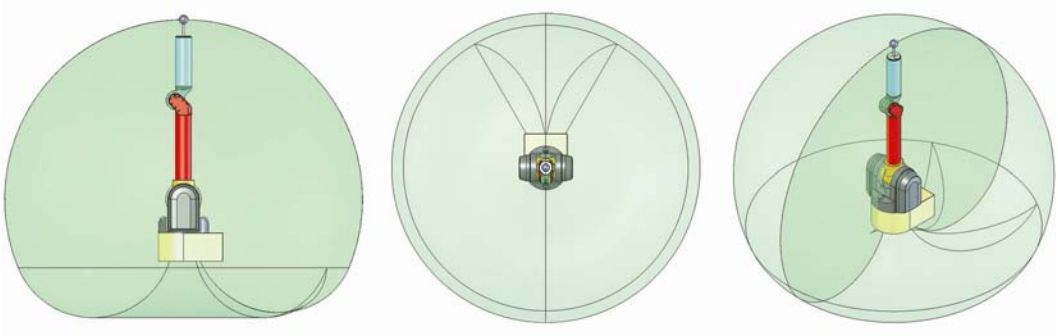

Figure 16 – WAM Workspace

- In addition, as with any piece of robotic equipment, it is ultimately up to you to be aware of your surroundings during robot operation. Although the WAM is intrinsically safer than other robotic systems, you may wish to integrate standard safety measures such as mats, gates, light curtains, etc. into the lab space surrounding the WAM.

- NEVER connect or disconnect any electrical cables while the Power Supply is turned on. Failure to follow this instruction could impart irreparable damage to the onboard electronics or put you at risk of electrical shock.

- Always plug the Power Supply into a properly grounded wall source. Failure to do so could damage the WAM electronics and put you at risk of electrical shock.

- Do not allow the WAM to be exposed to liquids that may cause electrical short-circuit and put you at risk of electrical shock.

- Keep dirt and debris away from the exposed cable drives located at the joints.

- Do not exceed the load limit of the arm: 4 kg for a 4-DOF WAM and 3 kg for a 7-DOF WAM. Consider all loading situations including accelerated loads, cantilever loads from long objects, robot collisions, active loads, etc.

- Remove/replace the WAM Wrist only as instructed in Section 3.4.

- Monitor the operating temperature of the WAM so that it remains between 0 °C and 70 °C. Under normal conditions, the Pucks operate between 40 and 60 °C. Idling the motors, when possible, will help keep the temperature lower.

2.2 Safety System: Pendants

2.2.1 Safety States

The WAM Arm has three safety states: E-STOP, IDLE, and ACTIVATED:

E-STOP means there is no motor bus voltage, in fact the motor bus power and ground lines are tied together, resulting in a "resistive braking" effect on the joints of the WAM Arm. The motor controllers are off line and do not keep track of their motor positions in this state. E-STOP is achieved by pressing the E-STOP button on either pendant. In this state, neither the IDLE nor ACTIVATE button lights are lit on the control pendant.

IDLE means there is voltage applied to the motor bus and the motor controllers are on line and keeping track of their motor positions, but they are commanded to tie their motor phase leads together (also resulting in a braking effect), and they will ignore any command torque sent to them. To put the WAM Arm into the IDLE state (which will also reset any existing faults), press and hold the Shift button on the control pendant, then press the Reset/Idle button (yellow) and release both buttons. The yellow Idle button will light up, indicating that the WAM Arm is now in the IDLE state. Make sure both E-STOP buttons are reset (popped up) before attempting to change modes.

ACTIVATED means the motor controllers are actively applying any commanded torque they receive from the control PC. To put the WAM into the ACTIVATED state, press and hold the Shift button on the control pendant, then press the Activate button (green) and release both buttons. This state may only be reached when *all* of the status lights are showing OK (green). All warnings or faults must be cleared before activating the WAM. The green Activate button will light up, indicating that the WAM Arm is now in the ACTIVATED state.

2.2.2 Status Lights

The WAM Arm has five sets of status lights: Velocity, Torque, Voltage, Heartbeat, and Other.

Velocity: Before the WAM Arm's joint positions are initialized by the PC control software, the velocity status lights indicate the state of the WAM's angular joint speed. By default, there is a yellow LED warning when any joint exceeds 0.5 radians/sec and a red LED fault when any joint exceeds 2 radians/sec. The joint number responsible for the warning/fault is indicated by the single-character display on each pendant. After the WAM Arm's joint positions are initialized, the safety system begins calculating and monitoring the WAM's elbow and arm endpoint velocities in Cartesian space instead of monitoring individual joint velocities. By default, there is a warning when either the elbow (single-character "E") or arm endpoint (single-character "A") exceeds 0.5 m/s and a fault when either one exceeds 2 m/s. These defaults are modifiable in software (see 6.3).

Torque: The torque status lights indicate the state of the torque commands being received by the WAM Arm from the PC control software. If the PC sends non-zero torques while the WAM is in the IDLE state (the yellow Reset/Idle button is lit), the safety system will display a torque warning- prohibiting the WAM Arm from being activated. If the WAM Arm is in the ACTIVATED state (the green Activate button is lit) and the PC sends torques which exceed the default torque warning or fault levels, the torque warning or fault light will be lit- and the offending motor number will be shown in the single-character display. If a torque limit is

exceeded, the safety system automatically idles all motors (puts the WAM in the IDLE safety state). See Section 6.3 for more information about how to set the torque limits.

Sometimes, when you exit a WAM control program, the torque warning light will stay on. That just means that the last torque that the safety system saw go by was beyond the warning level for the present state (> 0 for E-STOP or IDLE, or > TL1 for ACTIVATED). As long as you send one or more zero-torque commands to the pucks, the warning light should extinguish.

Voltage: The voltage status lights indicate the state of the WAM Arm's motor bus voltage. When the system is first powered up, the bus is off (there is no motor power), and the safety system registers a voltage fault. This fault is cleared by pressing Shift-Idle on the control pendant. Placing the WAM Arm into the IDLE state applies a DC voltage on the motor bus and clears the fault. If the voltage approaches the limits, the voltage warning light is lit. If the voltage exceeds the limits, the voltage fault light is lit.

Heartbeat: The heartbeat status lights indicate the state of the communication between the PC and each motor controller in the WAM Arm. If the WAM is in the IDLE state and no control loop is active between the PC and the motor controllers, the pendants will display a heartbeat warning. If the WAM is in the ACTIVATED state, the computer must request encoder values in a constant loop until you press Shift-Idle again. If PC or any motor controller fails to issue any communication for longer than 16ms, the safety system will register a heartbeat fault and shut down the WAM.

Other: The "Other" status lights presently only indicate whether an E-STOP has occurred. If this is the case, the fault light will be lit and the single-character display will show "E".

2.3 Handling Safety Faults

If the safety system registers a fault during WAM operation, the fault must be cleared before continuing to use the WAM. The most common safety faults (over velocity, over torque) leave the WAM safety system in the IDLE state (yellow button is lit). To reset the fault from this state (when using the btdiag example application):

1) Stop any running Teach & Play playback (press '/')
2) Set the position controller to IDLE mode (press 'p' to toggle mode)
3) Turn off gravity compensation (press 'g' and enter 0)
4) Reset the safety fault (press <Shift+Reset/Idle> on the control pendant)
5) Re-activate the WAM (press <Shift+Activate> on the control pendant)
6) Turn on gravity compensation (press 'g')

If the safety system encountered a critical fault that resulted in an E-STOP of the WAM (no pendant buttons are lit), follow these steps to recover from the fault:

1) Exit the application (press 'x')
2) Reset the safety fault (press <Shift+Reset/Idle> on the control pendant)
3) Grab the WAM and move it back to the home position (the joint positions are lost after an E-STOP)
4) Re-launch the control program and follow the on-screen instructions

		Error occurs in state		
		E-STOP	IDLE	ACTIVE
ERROR	Velocity exceeds VL1	No action	Warn	Warn
	Velocity exceeds VL2	No action	Fault, E-STOP	Fault, IDLE, Wait 1/4s
	Torque exceeds TL1	Warn for non-zero	Warn for non-zero	Warn
	Torque exceeds TL2	Fault	Fault	Fault, IDLE
	Heartbeat missing	Warn	Warn	Fault, E-STOP
	Voltage lower than VOLTL1	No action	Warn	Warn
	Voltage lower than VOLTL2	Fault	Fault	Fault, IDLE
	Voltage higher than VOLTH1	No action	Warn, Bleed voltage	Warn, Bleed voltage
	Voltage higher than VOLTH2	No action	Fault, E-STOP	Fault, E-STOP
	E-Stop pressed	Fault, E-STOP	Fault, E-STOP	Fault, E-STOP
	NOTE: Warnings are cleared automatically, critical faults are cleared through a RESET			
		Request occurs in state		
		E-STOP	IDLE	ACTIVE
REQUEST	E-STOP	E-STOP	E-STOP	E-STOP
	RESET / IDLE	Clear faults, power up bus, enumerate, IDLE	Clear faults	IDLE
	ACTIVATE	No Action	If no warnings or faults, ACTIVE	No Action

Figure 17 – WAM Responses to Errors and Requests

3 System Setup

3.1 Mounting

The mounting-surface for the WAM should be designed to handle the large reaction forces generated at the base of the arm during high-acceleration operation. The WAM can be fastened to a prepared mounting-surface in several ways. If you use four M10 or 3/8" screws, remove the base cover to access the four bolt holes in the base plate. The holes are located on the base of the WAM according to Figure 18 and Figure 19. The plate thickness is 8mm. If you use M12 screws, you can screw upwards from the opposite direction, eliminating the need to removing the base cover. Alternatively, the WAM can be mounted using the clamps provided in the maintenance kit. It is important that the mounting surface NOT be grounded to earth ground (see "Grounding" section below).

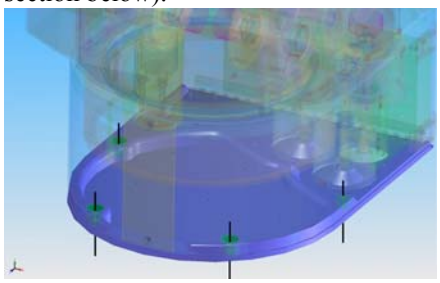

Figure 18 – Screw-hole Locations

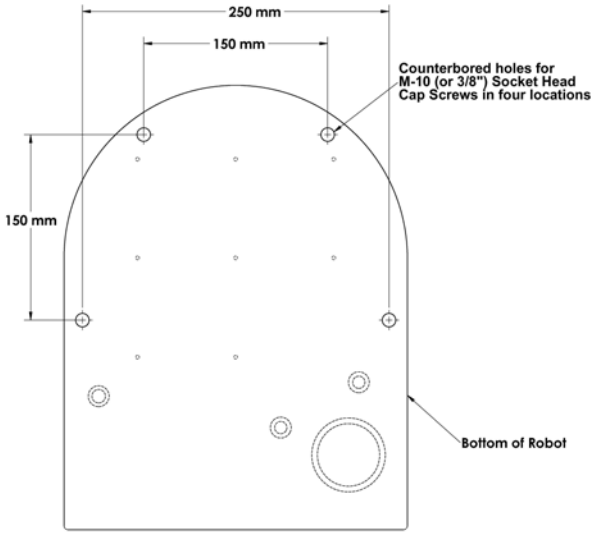

Figure 19 – Mounting-Hole Measurements

3.2 Grounding

Grounding of all hardware must be considered when locating the WAM and its peripherals. Multiple grounding paths between the external power supply, the WAM chassis, the Barrett Hand power supply, an external computer and earth ground can cause internal communications problems. It is important to mount the WAM itself on a non-grounded surface. However, if you need to physically mount it to a grounded structure (upside-down to a grounded I-Beam, or mounted to a grounded metal table, for example) contact Barrett Technology for instructions on making a slight modification to the external wiring to avoid multiple ground paths. Additionally, it is preferable that the metal chassis of the External Power Supply, the metal chassis of the Barrett Hand Supply (if applicable) and the WAM chassis are not in electrical contact with each other. If this is not possible, contact Barrett Technology for instructions. Finally, if using an external computer connected to either the WAM through the CANBus or the external Barrett Hand Supply, the computer *should* be grounded. Again, contact Barrett if this poses a problem. A computer connected to the Ethernet port can either be grounded or not grounded.

3.3 Installing the External PC (Optional)

3.3.1 Physical Installation

Install the PC shipped with the WAM as you would any other PC. The CANbus cable connects to the lower port of the PCI CAN card, not the built-in serial port. You will need to add a monitor while performing initial setup of the PC. Afterwards, you may use the PC for development or remove the monitor and use remote terminal software (such as *ssh*) to operate the PC remotely.

3.3.2 Software installation

The PC comes with the WAM software loaded. You will need to edit /etc/network/interfaces to set an IP address that is compatible with your network. We do not recommend using DHCP, as we have found that periodic address renegotiation interferes with the real-time operation of the WAM. The Barrett Technology software library uses *syslogd* to log all error messages to a file. It is highly recommended to make sure that *syslogd* is running. Error messages may be found in the */var/log/syslog* text file.

3.4 WAM Wrist (Optional)

To avoid damaging the WAM electronics, please make sure the WAM is powered off before detaching or attaching any outer link. In addition, if the Wrist was shipped separately from the WAM, a Safety Termination will be attached to the base connector of the Wrist, and must be removed before mounting the Wrist. If the Wrist is ever removed from the WAM, ensure that the Safety Termination is attached as soon as the Wrist is disconnected to prevent damage to the Wrist Pucks.

Figure 21 shows that the outer link can be unscrewed and removed from the WAM to accommodate the WAM Wrist (if the 7-DOF WAM is purchased). After removing the outer link, connect the WAM Wrist:

1. Align the D-sub connector of the Wrist with the mating connector on the end of the "elbow" of the WAM such that both have the same orientation.
2. Connect the alignment pins on the end of the elbow with the mating connections on the Wrist.
3. Thread the quick-connect ring onto the wrist base.

No special wrenches, fasteners, or tools are required, however, it may be necessary to shake the outer link (or Wrist) gently back and forth while tightening or loosening the ring. This ring should only be hand-tightened. It is normal for there to be a small number of threads left when fully tightened. This single operation also makes all electrical connections to the WAM Wrist. Figure 20 shows the electrical connector at the end of the WAM.

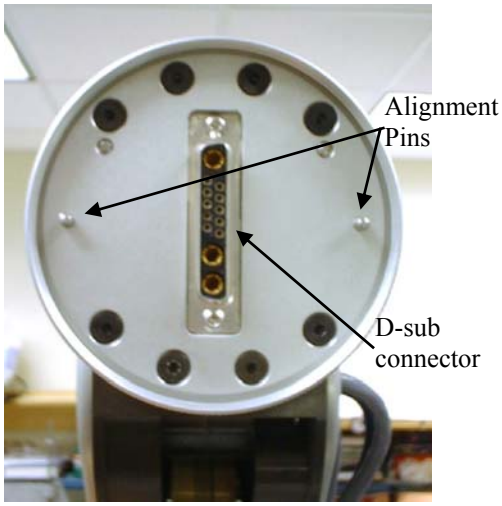

Figure 20 – Wrist Connector

Figure 21 – Separating the Outer Link

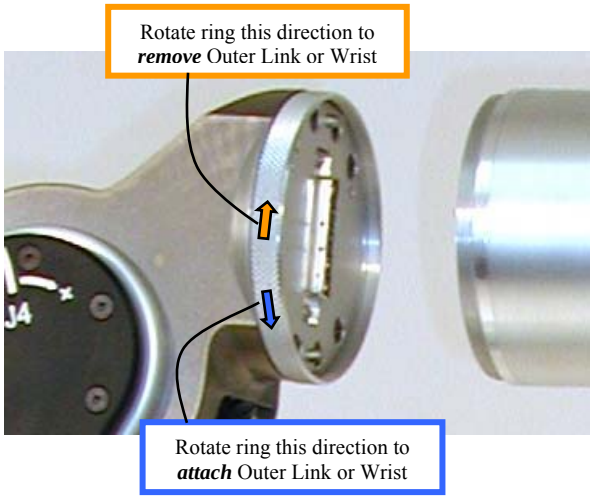

Figure 22: Close-up of Outer Link connection

3.5 Safety Board Settings

For different methods of communication (and different connections to the BarrettHand™, if attached) certain switches must be changed. To access these switches, turn off the WAM™, remove the 4-mm Hex screw in the upper-left corner of the back of the WAM™ (Figure 24), unlatch the base cover, and rotate the back plate of the WAM™ out so that it is in a horizontal position. (Figure 25). If you are facing the WAM™ from the back, the relevant switches will be on the lower-left corner (see Figure 23). If a switch is "out" the raised part of that pin is on the side closest to the edge of the safety board, and the opposite is true if the switch is "in".

Table 1 – WAM Communication Settings

Mode*:	Switch 1-2	Switch 1-3
Ethernet	IN	IN
CAN (from External PC)	OUT	OUT

Table 2 – Hand Communication and Power Settings

Modes*:	Switch 2	Switch 1-1
Internal Power & Communications	IN	IN
External Power & Communications	OUT	OUT

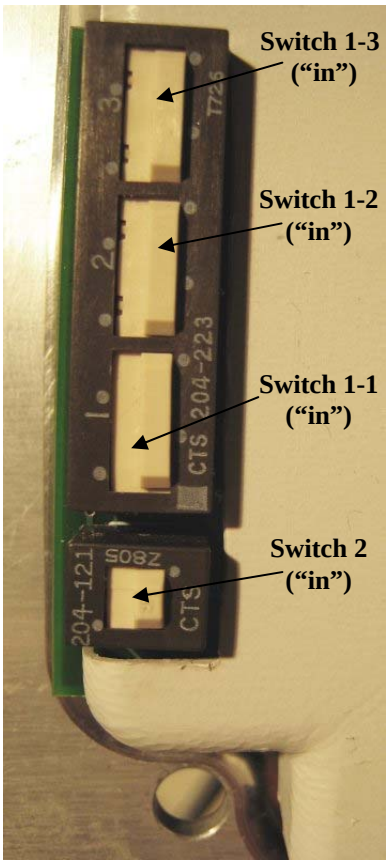

Figure 23 – Safety Board Switches

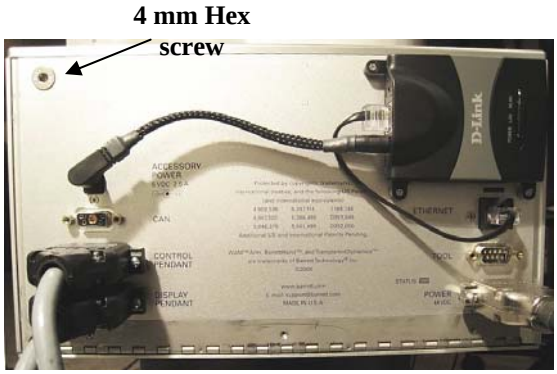

Figure 24 – WAM Backplate

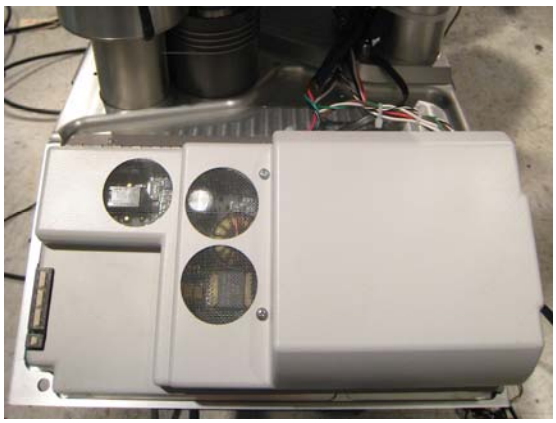

Figure 25 – Safety Board Cover

* Contact Barrett for additional modes of operation, for example: running 2 WAMs off of a single PC-104, or running a BarrettHand off of internal power with external (non-PC-104) communications.

3.6 Electrical Connections

3.6.1 Power Source

For applications using a standard (50-60 Hz, 100-120 or 200-240V) AC outlet:

- Place the Power Supply on a flat, secure surface and verify that the Power Supply is **OFF**.
- Attach the Power Supply Line Cord into any convenient outlet and also into the corresponding Power Supply socket.
- Attach the DC power cable from the socket on the back of the Power Supply to the socket on the WAM labeled "DC Power", as shown in Figure 11. Tighten the strain-relief screws on both ends of the cable using the screwdriver provided in the maintenance kit.

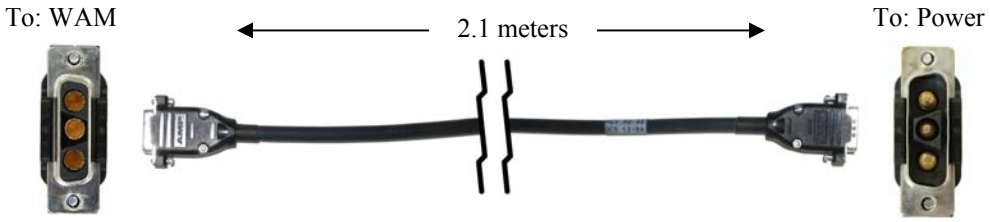

Figure 26 – DC Power Cable (blue)

For mobile applications or applications that require alternate DC power sources, simply connect the DC power cable directly from the power source to the socket on the WAM labeled "DC Power" as shown in Figure 26 (see Table 3 for details on DC power requirements). Tighten the strain-relief screws using the screwdriver provided in the maintenance kit.

Table 3 – DC Power Requirements

4-DOF	7-DOF
Quiescent: 18W	Quiescent: 27W
Typical: 28W	Typical: 45W
Peak: 600W	Peak: 800W

(Quiescent = powered up, no torques applied; Typical = operation with 2-kg payload)

3.6.2 Pendants

Of the two pendant cables, connect the pendant box with three colored buttons and a red E-Stop to the socket on the WAM marked "Pendant - Control". Connect the other pendant box (with only a red E-Stop) to the socket on the WAM marked "Pendant - Display". See Figure 27 for both pendants. Tighten the strain-relief screws using the screwdriver provided in the maintenance kit.

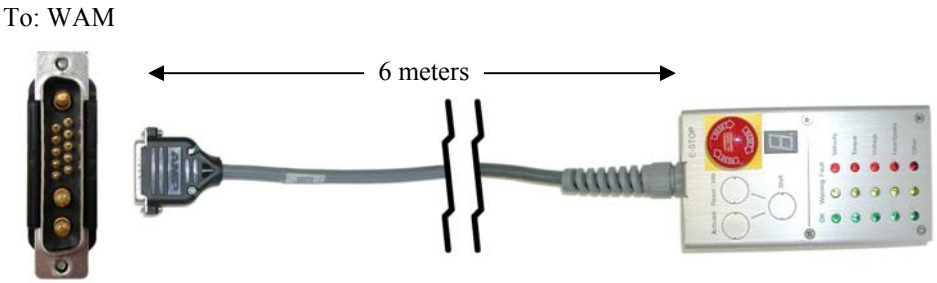

Figure 27 – Pendant Cables

3.6.3 Communications

You can control the WAM over wired Ethernet, wireless Ethernet, or CANbus.

Wired/Wireless Ethernet

If you are using wired Ethernet to communicate between your personal computer and the internal WAM PC, attach the Ethernet cable from the network port on the WAM to a port on an Ethernet switch or hub. If you are using wireless Ethernet, connect the provided wireless access point (AP) to the network port of the WAM. When the WAM turns on, it will request an IP address from a DHCP server.

The provided AP will return an address of the form: 192.168.nnn.100, and the ssid of the AP is WAM-nnn, where nnn is the serial number of your WAM. You can use a laptop PC or other wireless-enabled device to connect to the AP. You can also log in and configure the AP by going to http://192.168.nnn.30. Username is "admin", password is "WAM". Once logged in to the AP, you can check the IP addresses assigned by DHCP by clicking on the "DHCP" button on the left-hand side of the web page. Your PC's network name will be listed there, and there should be an entry without a name- that is the WAM PC and its corresponding IP address. This is the address you will use to control the WAM.

You may want to add a static entry for the WAM in your DHCP server to make future log-ins easier.

Figure 28 – Wireless Access Point on WAM Backplate

CANbus

If you are using the optional external PC, attach the CANbus cable from the socket on the WAM labeled "CAN" to the PC as shown in Figure 29. Either attach a monitor and keyboard to the PC to operate the PC directly or connect the PC to a personal computer with remote terminal software to operate the PC remotely. Tighten the strain-relief screws using the screwdriver provided in the maintenance kit.

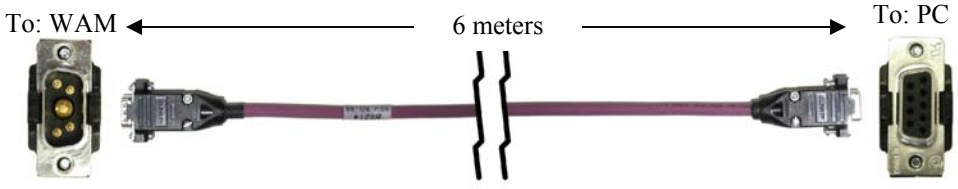

Figure 29 – CAN cable (purple)

3.7 Power-Up Sequence

Perform the following checks before powering up the WAM:

1. If you are using Ethernet or CANbus communication, verify that the appropriate cable is plugged into the desired communications port.
2. Confirm that the WAM safety board is properly configured for the communication option you have chosen.
3. Verify that the Control and Display Pendant cords are plugged into the correct sockets on the back of the WAM.
4. If you are using an AC power source, verify that the AC Line Cord is plugged into a valid power source (see Appendix B) and into the power outlet on the back of the Power Supply.
5. If you are using the external WAM PC, verify that the computer line cord is plugged into a power source and into the power outlet on the back of the computer.

Once the previous steps are complete, your WAM is ready for start-up. Power up the system according to the instructions below:

1. Turn on the power source to the WAM.
2. If using the external PC, turn on the power source to the external PC.
3. If using a personal computer to control the WAM, connect to the WAM PC (From a *nix PC, use ssh -l root <wam-ip-address>, or from a Windows PC, use the free PuTTY ssh client to connect – Putty can be downloaded at http://www.putty.nl/download.html – username = root, password = WAM).
4. Launch control program (btclient/src/btdiag/btdiag is standard)
5. Make sure both E-Stop buttons are reset (up). If they are not, rotate them one quarter of a turn clockwise.
6. Press <Shift+Reset/Idle> on the control pendant.
7. Place the WAM into its home position (see Appendix C for details on the home position).
8. Press <Shift+Activate> on the control pendant.

The WAM is now ready for use.

3.8 Code Examples

In the code examples, the PC closes a 500 Hz position/torque control loop with the WAM over the CAN bus. The PC asks the motor controllers for their present positions, converts the received encoder counts into joint angles, calculates the desired joint torques, converts these into motor torque commands, then sends out the calculated torques to the motors. All force or position control is calculated on the PC and converted to motor torques as a final step – the WAM itself is entirely motor torque controlled. For a source code example of this process, see the WAMControlThread() function in src/btwam/btwam.c.

These examples are preloaded onto the WAM PC to help demonstrate the capabilities of the WAM Arm and the functionality of the WAM library. The examples and their functions are as follows:

Example 1: A minimalist program for the WAM that prints out the position of the end point of the WAM.
Example 2: A minimalist program for the WAM that calculates and sends gravity compensation torques.
Example 3: This program demonstrates the datalogging, timing, and WAM control loop callback functions of the library.
Example 4: Demonstrates haptic (force-feedback) interaction with the WAM.
Example 5: Demonstrates the MoveWAM() function in joint space and Cartesian space.
Example 7: Opens the serial port at the end of the outer link, initializes the BarrettHand, and makes the Hand move.

4 PC & Control Software

4.1 Library Overview

This provides a brief overview of the WAM library. It is not an exhaustive feature list, but it gives a sampling of the WAM library's functionality.

Every WAM comes with its own internal PC running real-time Linux. Many WAM control demos and example applications are pre-loaded on the PC, along with their full ANSI-C source code. Each WAM application is linked against the WAM control library (source included, with C++ wrappers). This library offers easy access to many properties of the WAM, including:

- Joint torques
- Joint positions

Once it receives the joint positions from the WAM, the library uses the forward kinematics and mass parameters from an easily configured file (wam.conf) to calculate:

- Tool position and orientation (XYZ, RxRyRz)
- Jacobians for each link
- Mass (inertia) matrix

The WAM is a fundamentally torque-controlled robot. The typical control loop asks for joint positions and calculates new joint torques at 500 Hz. This control rate is adjustable (up to 1 kHz) to meet varying requirements. The WAM library saturates packed torques at 14-bits (from –8191 to +8191). To assist with application development, the WAM library offers:

- Adjustable control loop sample time
- Real-time data logging of any application data up to the control frequency
- Control loop callback hooks to perform calculations on the WAM data in real-time

Due to the WAM's tension element (cable) drive system, motor torque is directly related to joint torque with zero backlash and near-zero friction. This makes it possible for the WAM library to perform online real-time joint torque calculations (using the Jacobian transpose) to generate any desired link forces. The WAM's "gravity compensation" routines use this feature to make the robot arm appear weightless by constantly calculating and applying the correct upward force on each link to negate the effect of gravity. Gravity compensation makes it very easy to perform lead-through teaching and precision playback. Real-time link forces are also useful for creating "haptic scenes" or interactive virtual objects that a user can "feel" within the workspace of the WAM. The WAM library gives you access to:

- Joint torques calculated to yield gravity-compensation forces
- Teach & Play functionality with a simple keypress
- Simple haptic scene primitives

Even though a key feature of the WAM is its ability to apply forces in Cartesian (XYZ) coordinates without requiring a force sensor, the WAM library also provides a simple position controller (PID) in order to perform joint-space or Cartesian-space position moves like other robot arms. Therefore, the WAM library also provides:

- Point-to-point moves in joint or Cartesian space
- Adjustable acceleration and max velocity (for trapezoidal moves)
- Continuous-trajectory path-following in joint or Cartesian space
- Adjustable PID gains (control stiffness)
- Joint torque saturation limits
- Cartesian force saturation limits

Because the WAM was designed to be used with people in its workspace, safety is paramount. The WAM library allows you to adjust the sensitivity levels in the WAM's embedded safety system. This circuitry actively monitors the state of the robot, even if the control PC crashes. Using the WAM library, you can adjust:

- Torque limits to trigger a shutdown
- Joint velocity limits to trigger a shutdown
- Endpoint and elbow velocity limits to trigger a shutdown

For a more in-depth description of the WAM library API, please see the Doxygen documentation provided with the source code.

4.2 File System Layout

The Barrett Technology robot control client software (btclient) is divided into multiple parts:

examples/ - example source code for robot control software development
doc/ - source code documentation, Doxygen (HTML/RTF) **
lib/ - location of library binaries
include/ - common include directory for libraries
src/btsystem/ - software library with general robot routines*
src/btwam/ - software library with WAM-specific routines*
src/btutil/ - utility for enumerating the motor controllers, restoring defaults, updating firmware
src/btdiag/ - the primary robot diagnostic application (and demo program)

*If you edit the btsystem or btwam libraries, the procedure to recompile is:

```
# make lib; make install
```

**To rebuild the Doxygen documentation:

```
# cd btclient/src; doxygen config.dox
```

4.3 The Configuration File (wam.conf)

The configuration file (wam.conf) is read by the OpenWAM() function to set up the operating parameters for the WAM. This file must be located in the same directory as your executable application and may be a symbolic link to a configuration file that describes your particular WAM parameters. If you add or remove the wrist, or change any link or tool masses, you will need to edit your active configuration file. You should take a look at this file to familiarize yourself with its contents. It contains the default home position, the Denavit-Hartenberg parameters, and the inertial parameters, and many other useful robot parameters.

4.4 Operating Modes

The WAM has a number of operating modes (key commands apply to the btdiag application):

IDLE/POSITION (press 'p' to toggle)
In IDLE mode, the WAM does not try to maintain its present position. In POSITION mode, it will try to maintain its position using a PID controller.

JOINT/CARTESIAN (press <tab> to toggle)
In JOINT mode, the WAM will report joint angles (in radians) and control joint torques (in Nm). In CARTESIAN mode, it will report the end-effector position in world coordinates (in meters) and control tool forces (in Newtons). In both cases, the Move command executes a trapezoidal velocity profile and generates a stream of torque commands based on position error.

Move command arguments for JOINT mode must be entered in units of radians. In CARTESIAN mode, the units are meters from the world frame origin. All arguments should be specified and comma-separated: 4 arguments for a 4-DOF JOINT Move, 7 arguments for a 7-DOF JOINT Move, 12 arguments for any CARTESIAN Move. See examples/ex5-SimpleMove for more detail.

GRAVITY-COMPENSATION (press 'g', enter a scaling value)

If you set the gravity scaling value set to 1.0 and have appropriate mass parameters defined in "wam.conf", then the WAM will apply the necessary joint torques to "float" in normal Earth gravity.

4.5 Updating Firmware

Updating the firmware occurs in two steps: updating the Pucks™, and updating the Safety Board. However, both of these processes must be completed in the order below; do not do one without the other. The following instructions are for WAMs which have the most current version of safety boards (the wireless router is mounted on the outside). If you are trying to update a WAM without the above characteristics, or you need to download some of the files mentioned in the following steps please see instructions on the Barrett wiki at: http://wiki.barrett.com/index.php/Firmware_update_instructions.

4.5.1 Pucks™

1. Move to the btutil directory (*cd ~/btclient/src/btutil*)
2. Make the btutil application (*make clean; make*)
3. Turn on the WAM power supply
4. Press <SHIFT+RESET/IDLE> on the control pendant
5. Determine which version of firmware you are upgrading from (*./btutil -g 1*)
6. Note the VERS (version) and CTS (counts) parameters.
7. Repeat the following steps (replace <id> with a Puck™ number to update, 1 – 7):
   a. For WAMs with optical encoders (if CTS = 40960)
      - Motors 1-4: *./btutil -d <id> -f puck2.tek.r39b*
      - Motors 5-6: *./btutil -d <id> -f puck2.tek.r39b.wdiff*
      - Motor 7: *./btutil -d <id> -f puck2.tek.r39b.wroll*

   b. For WAMs with magnetic encoders (if CTS = 4096), all motors: *./btutil -d <id> -f puck2.tek.r109*
   c. Wait for the download to complete
   d. Set the default parameters for the new firmware (*./btutil -p <id> -l <id>*)
      - "*-l*" is a dash-ell, not dash-one
      - example for puck 2: *./btutil -p 2 -l 2*
8. Switch off the WAM power supply when you have finished updating all the pucks.
9. Update the Safety Board firmware using the instructions on the following page (this is not optional).

4.5.2 Safety Board

1. Ensure that main power to the WAM is off.
2. Remove the wrap-around base cover by releasing the four latches that secure it to the back plate of the WAM.
3. Use a 4-mm hex key to remove the screw that keeps the back plate closed.
4. Remove the 4-6 small Philips screws securing the safety board cover (see Figure 30).
5. Connect the serial cable (Figure 31) to the connector near the safety puck and flip the nearby switch (both halves) to the "ON" position. See Figure 32 for the final setup.
6. Connect the other side of the serial cable to COM1 of a Windows PC.
7. Turn on main power to the WAM.
8. Under MS Windows, simultaneously drag puck2mon.out.r6 and puck2.out.r102.enet.mag onto F28xxConsole.exe.
9. Wait for the download to complete (about 4 minutes).
10. Turn off main power to the WAM.
11. Flip the nearby switch (both halves) to the "OFF" position (opposite of Figure 32).
12. Launch TeraTerm/Hyperterminal (9600 baud, no parity, 8 bits, 1 stop bit, no flow control)
13. Turn on main power to the WAM. You should get a prompt (=>). Type the following commands in sequence, each followed by <Enter>. A mis-typed command will NOT result in an error message, so type carefully!
    a. *SET SAFE 4* (this is necessary; do not skip this step)
    b. *SET SAFE 5*
    c. *FIND VBUS*
    d. *SET SAFE 0*
    e. *SET VOLTL1 36*
    f. *SET VOLTL2 30*
    g. *SET VOLTH1 54*
    h. *SET VOLTH2 57*
    i. *SET GRPA 1*
    j. *SET GRPB 2*
    k. *SET GRPC 3*
    l. *SAVE*
    m. *RESET*
    n. *GET VOLTL1* (should be 36, else SET and SAVE it again)
    o. *GET VOLTL2* (should be 30, else SET and SAVE it again)
    p. *GET VOLTH1* (should be 54, else SET and SAVE it again)
    q. *GET VOLTH2* (should be 57, else SET and SAVE it again)
    r. *GET GRPA* (should be 1, else SET and SAVE it again)
    s. *GET GRPB* (should be 2, else SET and SAVE it again)
    t. *GET GRPC* (should be 3, else SET and SAVE it again)
14. Turn off main power to WAM

Figure 30 – Safety Board Cover

Figure 31 – Serial Cable

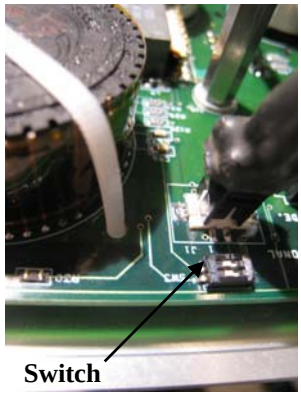

Figure 32 – Serial Connection with Switch in "On" Position

5 WAM Commands List

For the full WAM Application Programming Interface (API), please refer to the fully indexed and cross-referenced doxygen-style documentation in the btclient/doc/html directory.

Command: getProperty
Name: Get Property
Purpose: Obtains the properties of a single Puck™.
Arguments: Bus (0), Node [1,31] (0x400 is not valid), property [0,108] inclusive (use the enumerations in btcan.h), &reply
Example:
Notes:

getProperty(bus, node, property, &reply)
bus = 0 (no other value has been tested)
node =
property =
&reply = pointer to a long integer

If you want to get the properties of several pucks at one time, you will have to write your own routine using an array for the replies along with canSendMsg() and canReadMsg() - see the getPositions() function in src/btsystem/btcan_esd.c as an example.

Otherwise, you can use getProperty to get a single property of a single puck at a time.

Command: setProperty
Name: Set Property
Purpose: Sets the properties of Pucks™ (single or group)
Arguments: Bus (0), Node [1,31] (0x400 is not valid), property [0,108] inclusive (use the enumerations in btcan.h), &reply
Example:
Notes:

The setProperty() function, in contrast, DOES allow nodeIDs OR group messageIDs to be used interchangeably. So setProperty(0,0x400,STAT,FALSE,STATUS_READY) is okay.

However, if the 4th parameter is set to TRUE (verify=TRUE), then it is effectively calling getProperty() with the 0x400, which is illegal. Calling getProperty() with 0x400 will prompt at least 4 responses (one from each puck), but getProperty() will only read the first one, leaving 3 in the queue to be read by some later call to getProperty() (or canReadMsg()). In order to prevent confusion after the illegal call (which could occur even if things are done correctly), reboot the system to clear out these messages. Alternatively, you could call canReadMessage() with a non-blocking read until no messages are found – see getBusStatus() for an example.

Most properties may be read and written via getProperty() and setProperty() messages.

6 WAM Properties List

6.1 Common Properties

These properties are common to both the motor encoders and the Safety Module.

Property:	**ADDR**		
Definition:	Address to peek/poke		
Values (Units):		*Read/Write:*	R/W
Default:	0	*Saved:*	No
Notes:			

Property:	**ANA0, ANA1**		
Definition:	Analog input values from the on-board DAC		
Values (Units):	0 – 4095	*Read/Write:*	R/-
Default:	None	*Saved:*	No
Notes:	0-3.0 V only		

Property:	**BAUD**		
Definition:	RS-232 serial baud rate		
Values (Units):	Bits/sec	*Read/Write:*	R/W
Default:	9600	*Saved:*	No
Notes:			

Property:	**CMD**		
Definition:	Execute a special command over CANbus, instead of the typical Get/Set command.		
Values (Units):		*Read/Write:*	-/W
Default:	None	*Saved:*	No
Notes:	Set CMD to the enumerated value of the desired command. Can only be used with commands that don't take values: RESET, HOME, KEEP, PASS, LOOP, HI, IC, IO, TC, TO, C, O, T		

Property:	**DEF**		
Definition:	Default command for CAN		
Values (Units):		*Read/Write:*	-/W
Default:	None	*Saved:*	No
Notes:	Resets all parameters to their defaults (use SAVE to store them permanently in EEPROM)		

Property:	**DIG0**		
Definition:	Digital I/O, PWM-capable		
Values (Units):	-1 to 100	*Read/Write:*	R/W
Default:	0	*Saved:*	No
Notes:	0-3.3 V only (no optical isolation, reverse voltage, or overvoltage protection, be careful). -1 = Input, 0 = Output low, 1 = Output high, 2-100 = PWM duty cycle of n%, Pin 41 (Pin 44 = GND)		

Property:	**DIG1**		
Definition:	Digital I/O		
Values (Units):	-1, 0, 1	*Read/Write:*	R/W
Default:	0	*Saved:*	No
Notes:	0-3.3 V only (no optical isolation, reverse voltage, or overvoltage protection, be careful). -1 = Input, 0 = Output low, 1 = Output high, Pin 43 (Pin 44 = GND)		

Property:	**ERROR**		
Definition:	Reports the last error encountered by the Puck		
Values (Units):		*Read/Write:*	R/-
Default:	None	*Saved:*	No
Notes:	Not yet implemented		

Property:	**FET0**		
Definition:	Tensioner output (0 = Off, 1 = On)		
Values (Units):	0, 1	*Read/Write:*	R/W
Default:	0	*Saved:*	No
Notes:	Pin 33 floats when FET0 = 0, it is grounded when FET0 = 1 (Pin 34 = 5V, current source)		

Property:	**FET1**		
Definition:	Brake output (0 = Off, 1 = On)		
Values (Units):	0, 1	*Read/Write:*	R/W
Default:	0	*Saved:*	No
Notes:	Pin 31 floats when FET1 = 0, it is grounded when FET1 = 1 (Pin 32 = 5V, current source)		

Property:	**FIND**		
Definition:	Find command for CAN		
Values (Units):		*Read/Write:*	-/W
Default:	None	*Saved:*	No
Notes:	Used for calibration. Set FIND to the value you want to calibrate: setProperty(0, FIND, FALSE, MOFST);		

Property:	**GRPA, GRPB, GRPC**		
Definition:	The communication groups to which the puck belongs		
Values (Units):	0 to 31	*Read/Write:*	R/W
Default:	None	*Saved:*	Yes
Notes:	A Puck can be addressed by ID, or as part of a group		

Property:	**ID**		
Definition:	CANbus ID		
Values (Units):	0 to 31	*Read/Write:*	R/W
Default:	None	*Saved:*	Yes
Notes:	Each Puck on a CANbus must have a unique ID		

Property:	**ILOGIC**		
Definition:			
Values (Units):		*Read/Write:*	R/-
Default:	None	*Saved:*	No
Notes:			

Property:	**IMOTOR**		
Definition:			
Values (Units):		*Read/Write:*	R/-
Default:	None	*Saved:*	No
Notes:			

Property:	**LOAD**		
Definition:	Load a property from non-volatile memory into active puck memory via the CAN bus		
Values (Units):		*Read/Write:*	-/W
Default:	None	*Saved:*	No
Notes:	Example of use: setProperty(0,4,LOAD,FALSE,MT)		

Property:	**LOCK**		
Definition:	A combination lock to allow write access to locked properties		
Values (Units):		*Read/Write:*	-/W
Default:	None	*Saved:*	No
Notes:	These properties are locked: ROLE, SN, PTEMP, UPSECS, OD		

Property:	**MODE**		
Definition:	The control method for the motor controller		
Values (Units):	0, 2, 3, 4, 5	*Read/Write:*	R/W
Default:	0	*Saved:*	No
Notes:	0 = Idle, 2 = Torque, 3 = PID, 4 = Velocity, 5 = Trapezoidal		

Property:	**OTEMP**		
Definition:	Over temperature limit		
Values (Units):		*Read/Write:*	R/W
Default:	82	*Saved:*	No
Notes:	MT is reduced gradually 16C before OTEMP. When TEMP >= OTEMP, MT will be zero.		

Property:	**PTEMP**		
Definition:	Peak temperature recorded		
Values (Units):		*Read/Write:*	R/W
Default:	None	*Saved:*	Yes
Notes:			

Property:	**ROLE**		
Definition:	A value used by the firmware to determine the function of the device		
Values (Units):		*Read/Write:*	R/W
Default:	None	*Saved:*	Yes
Notes:	Bitfield: F=Feedback, R=Role: XXXX FFFF XXXX RRRR ROLE: 0 = Tater, 1 = Gimbals, 2 = Safety, 3 = Wraptor, 4 = Trigger FEEDBACK: 0 = Optical encoder, 1 = Magnetic encoder		

Property:	**SAVE**		
Definition:	After setting a non-volatile property, save it to non-volatile Puck™ memory		
Values (Units):		*Read/Write:*	-/W
Default:	None	*Saved:*	No
Notes:	Example: setProperty(0, 4, SAVE, FALSE, MT);		

Property:	**SG**		
Definition:	Analog reading of the strain gage peripheral		
Values (Units):	0-4095	*Read/Write:*	R/-
Default:	None	*Saved:*	No
Notes:	The 12-bit value 0-4095 corresponds to an analog reading of 0-3.0V		

Property:	**SN**		
Definition:	The unique manufacturing serial number for this Puck		
Values (Units):		*Read/Write:*	R/W
Default:		*Saved:*	Yes
Notes:			

Property:	**STAT**		
Definition:	0 = Reset/Monitor, 2 = Ready/Main		
Values (Units):	0, 2	*Read/Write:*	R/W
Default:		*Saved:*	No
Notes:	The Puck firmware is divided into two parts: Monitor and Main. The Puck initially boots into Monitor where there is a very limited command/property set. From here, you can download new Main firmware, or SET STAT 2 to execute the Main firmware. Setting STAT=0 forces a firmware reset, which will leave the motors free-spinning (no torque, no braking).		

Property:	**TEMP**		
Definition:	Puck™ temperature		
Values (Units):		*Read/Write:*	R/-
Default:		*Saved:*	No
Notes:	The Puck temperature should not exceed 70 ˚C for extended periods.		

Property:	**THERM**		
Definition:	Thermistor (motor) temperature		
Values (Units):		*Read/Write:*	R/-
Default:		*Saved:*	No
Notes:	The motor temperature should not exceed 125 ˚C for extended periods.		

Property:	**VALUE**		
Definition:	Value to poke/peek		
Values (Units):		*Read/Write:*	R/W
Default:		*Saved:*	No

Notes:	Use with ADDR to read/write data from the Puck memory.
	Example: Read the TMS320F2812 DSP's GPBDAT register
	setProperty(0, 4, ADDR, FALSE, 28900);
	getProperty(0, 4, VALUE, &longVal);		

Property:	**VBUS**		
Definition:	The bus voltage on a Puck™		
Values (Units):		*Read/Write:*	R/-
Default:		*Saved:*	No
Notes:	The standard Puck is designed for operation between 18V and 90V. However, contact Barrett Technology if operating the bus at a voltage other than 48 V.		

Property:	**VERS**		
Definition:	Firmware version (or the monitor version, if the Puck™ is in reset)		
Values (Units):		*Read/Write:*	R/W
Default:		*Saved:*	No
Notes:			

Property:	**VLOGIC**		
Definition:			
Values (Units):		*Read/Write:*	R/-
Default:		*Saved:*	No
Notes:	Not yet implemented		

Property:	**X0, X1, X2**		
Values (Units):		*Read/Write:*	R/W
Default:		*Saved:*	Yes
Default:	N/A		
Notes:			

Property:	**X3, X4, X5**		
Definition:			
Values (Units):		*Read/Write:*	R/W
Default:		*Saved:*	Yes
Notes:			

Property:	**X6, X7**		
Definition:			
Values (Units):		*Read/Write:*	R/W
Default:		*Saved:*	Yes
Notes:			

6.2 Motor Properties

These properties are used only by the motor controllers (not the safety module).

Property:	**ACCEL**		
Definition:			
Values (Units):		*Read/Write:*	R/W
Default:	2048	*Saved:*	Yes
Notes:			

Property:	**CT**		
Definition:	32-Bit Close Target		
Values (Units):		*Read/Write:*	R/W
Default:	30000	*Saved:*	Yes
Notes:			

Property:	**CTS**		
Definition:	32-Bit Counts per revolution		
Values (Units):		*Read/Write:*	R/W
Default:	4096	*Saved:*	Yes
Notes:	CTS		

Property:	**DP**		
Definition:	32-Bit Default Position		
Values (Units):		*Read/Write:*	R/W
Default:	0	*Saved:*	Yes
Notes:			

Property:	**DS**		
Definition:			
Values (Units):		*Read/Write:*	R/W
Default:	25000	*Saved:*	Yes
Notes:			

Property:	**E**		
Definition:	32-Bit Endpoint		
Values (Units):		*Read/Write:*	R/W
Default:	0	*Saved:*	No
Notes:			

Property:	**ECMAX**		
Definition:	Encoder Correction Maximum Value		
Values (Units):		*Read/Write:*	R/W
Default:	0	*Saved:*	No
Notes:			

Property:	**ECMIN**		
Definition:	Encoder Correction Minimum Value		
Values (Units):		*Read/Write:*	R/W
Default:	0	*Saved:*	No
Notes:			

Property:	**EN**		
Definition:			
Values (Units):		*Read/Write:*	R/W
Default:		*Saved:*	No
Notes:			

Property:	**HALLH**		
Definition:	32-Bit Hall History Bitfield		
Values (Units):		*Read/Write:*	R/W
Default:	0	*Saved:*	No
Notes:			

Property:	**HALLS**		
Definition:	Hall feedback bitfield: CBA		
Values (Units):		*Read/Write:*	R/W
Default:		*Saved:*	No
Notes:			

Property:	**HOLD**		
Definition:	Flag to hold position after move		
Values (Units):		*Read/Write:*	R/W
Default:	0	*Saved:*	Yes
Notes:			

Property:	**HSG**		
Definition:			
Values (Units):		*Read/Write:*	R/W
Default:	255	*Saved:*	Yes
Notes:			

Property:	**IKCOR**		
Definition:	Current sense correction factor		
Values (Units):		*Read/Write:*	R/W
Default:	1638	*Saved:*	Yes
Notes:			

Property:	**IKI**		
Definition:	Current sense integral gain		
Values (Units):		*Read/Write:*	R/W
Default:	3276	*Saved:*	Yes
Notes:			

Property:	**IKP**		
Definition:	Current sense proportional gain		
Values (Units):		*Read/Write:*	R/W
Default:	8192	*Saved:*	Yes
Notes:			

Property:	**IOFF**		
Definition:	32-Bit Intialization Offset		
Values (Units):		*Read/Write:*	R/W
Default:		*Saved:*	Yes
Notes:			

Property: **IOFST**
Definition: Current Offset Calibration
Values (Units): *Read/Write:* R/W
Default: *Saved:* Yes
Notes:

Property: **IPNM**
Definition: The puck torque unit to Nm (Command Current / Nm) conversion is stored here
Values (Units): *Read/Write:* R/W
Default: 2755 *Saved:* Yes
Notes: IPNM is set at the factory.
IPNM is never actually used in the pucks themselves. It is simply a non-volatile place to store the conversion constant so that the controlling PC can read it upon startup and use it to convert between puck units and SI units of torque for the end user.

Property: **IVEL**
Definition:
Values (Units): *Read/Write:* R/W
Default: 20 *Saved:* Yes
Notes:

Property: **JIDX**
Definition:
Values (Units): *Read/Write:* R/W
Default: *Saved:* Yes
Notes:

Property: **KD**
Definition:
Values (Units): *Read/Write:* R/W
Default: 8000 *Saved:* Yes
Notes:

Property: **KI**
Definition:
Values (Units): *Read/Write:* R/W
Default: 0 *Saved:* Yes
Notes:

Property: **KP**
Definition:
Values (Units): *Read/Write:* R/W
Default: 2000 *Saved:* Yes
Notes:

Property: **LCTC**
Definition:
Values (Units): *Read/Write:* R/W
Default: 1 *Saved:* No
Notes:

Property:	**LCVC**		
Definition:			
Values (Units):		*Read/Write:*	R/W
Default:	1	*Saved:*	No
Notes:			

Property:	**LFLAGS**		
Definition:			
Values (Units):		*Read/Write:*	R/W
Default:	0	*Saved:*	No
Notes:			

Property:	**LSG**		
Definition:			
Values (Units):		*Read/Write:*	R/W
Default:	0	*Saved:*	Yes
Notes:			

Property:	**M**		
Definition:	32-Bit Mechanical Angle in ENCODER counts?		
Values (Units):		*Read/Write:*	-/W
Default:		*Saved:*	No
Notes:			

Property:	**MCV**		
Definition:	In counts/ms		
Values (Units):		*Read/Write:*	R/W
Default:	1500	*Saved:*	Yes
Notes:			

Property:	**MDS**		
Definition:	Max Duty Sum for Power Limiting		
Values (Units):		*Read/Write:*	R/W
Default:	1650	*Saved:*	Yes
Notes:			

Property:	**MECH**		
Definition:	Mechanical encoder angle		
Values (Units):		*Read/Write:*	R/-
Default:		*Saved:*	No
Notes:	For magnetic encoders, MECH is the raw encoder feedback, absolute within one rotation of the motor. For optical encoders, MECH is the number of encoder counts between the present location and the index pulse. (See Appendix C for details about MECH with optical encoders).		

Property:	**MOFST**		
Definition:	Mechanical Offset Calibration		
Values (Units):		*Read/Write:*	R/W
Default:		*Saved:*	Yes
Notes:	Calibrated at lab MOFST is the encoder reading at the start of an electrical cycle. (See Appendix C for details about MOFST with Optical Encoders).		

Property:	**MOV**		
Definition:	Counts/ms		
Values (Units):		*Read/Write:*	R/W
Default:	1500	*Saved:*	Yes
Notes:			

Property:	**MPE**		
Definition:			
Values (Units):		*Read/Write:*	R/W
Default:	5	*Saved:*	Yes
Notes:			

Property:	**MT**		
Definition:	The maximum torque a Puck™ will apply, even if commanded to apply more (in milliamps).		
Values (Units):		*Read/Write:*	R/W
Default:	4700	*Saved:*	Yes
Notes:	OpenWAM() calls InitializeSystem() (see the Doxygen documentation) which sets MT to 4731 for each puck. A value of 1024 corresponds to 1.00A of current in the motor windings. Let's say MT = 750, TL1 = 2500, TL2 = 4000. You send a T of 2800. The safety system is okay with this, it just shows the warning light on the pendant. The puck notices that abs(2800) exceeds 750, so it sets its own torque output to 750 * sign(2800).		

Property:	**MV**		
Definition:	Maximum Velocity in counts/millisecond		
Values (Units):		*Read/Write:*	R/W
Default:	1500	*Saved:*	Yes
Notes:			

Property:	**OD**		
Definition:			
Values (Units):		*Read/Write:*	R/W
Default:		*Saved:*	Yes
Notes:			

Property:	**OT**		
Definition:	32-Bit Open Target		
Values (Units):		*Read/Write:*	R/W
Default:	0	*Saved:*	Yes
Notes:			

Property:	**P**		
Purpose:	motor position in encoder counts.		
Values (Units):		*Read/Write:*	R/W
Default:		*Saved:*	Yes
Notes:	You can set P in a puck only when the puck's MODE = 0 (MODE = MODE_IDLE). Just be careful to set IFAULT (see Section 6.3) in the safety module before you do this, else you will likely get a velocity fault		

Property:	**PIDX**		
Definition:	Puck™ Index for Torque		
Values (Units):		*Read/Write:*	R/W
Default:		*Saved:*	Yes
Notes:			

Property:	**POLES**		
Definition:	Number of magnets on rotor		
Values (Units):		*Read/Write:*	R/W
Default:	12, 8, or 6, depending on motor type	*Saved:*	Yes
Notes:			

Property:	**T**		
Definition:	Set the Puck™ torque (in milliamps)		
Values (Units):		*Read/Write:*	R/W
Default:	0	*Saved:*	No
Notes:	The pucks will apply the last received torque until a new torque is sent, or the MODE changes to MODE_IDLE.		

Property:	**TENSO**		
Definition:			
Values (Units):		*Read/Write:*	R/W
Default:		*Saved:*	Yes
Notes:			

Property:	**TENST**		
Definition:			
Values (Units):		*Read/Write:*	R/W
Default:		*Saved:*	Yes
Notes:			

Property:	**TIE**		
Definition:	Flag to tie inner and outer links		
Values (Units):		*Read/Write:*	R/W
Default:	0	*Saved:*	Yes
Notes:			

Property:	**TSTOP**		
Definition:	Time until considered stopped		
Values (Units):		*Read/Write:*	R/W
Default:	1000	*Saved:*	Yes
Notes:			

Property:	**UPSECS**		
Definition:	The total power-up time for this WAM		
Values (Units):		*Read/Write:*	R/W
Default:		*Saved:*	Yes
Notes:			

Property:	**V**		
Definition:	Velocity in counts/milliseconds		
Values (Units):		*Read/Write:*	R/W
Default:	0	*Saved:*	No
Notes:			

6.3 Safety-Module Properties

The Safety Module is an additional Puck located in the base of the WAM (not attached to a motor). The Safety Module has a puck ID of 10, and is identified in the software as SAFETY_MODULE. Its job is to listen to the CANbus traffic and shut the WAM down in the event of an over-torque, over-velocity, voltage problem, or heartbeat error (missing communication from any puck or the PC). It also controls the pendant lights and switches. The properties below are only applicable to the Safety Module. All safety module properties apply to the whole arm on a global basis. For example, if the torque of ANY motor exceeds TL2, the safety system will force the arm into IDLE mode (all phase leads tied together).

Property: **IFAULT**
Definition: If IFAULT is greater than 0 and a fault is observed by the safety system, IFAULT is decremented by one, and the fault is ignored.
Values (Units): 0 to 20 (typical) *Read/Write:*
Default: 0 *Saved:*
Notes: To set the initial positions of the pucks, you must also set IFAULT (Ignore Fault) at that time to prevent the giant instantaneous position change from causing an over-velocity fault in the safety module. See the DefineWAMpos() function in the Doxygen documentation for an example.

Property: **MAXPWR**
Definition:
Values (Units): *Read/Write:*
Default: *Saved:*
Notes:

Property: **PEN**
Definition:
Values (Units): *Read/Write:*
Default: *Saved:*
Notes:

Property: **PWR**
Definition:
Values (Units): *Read/Write:*
Default: *Saved:*
Notes:

Property: **SAFE**
Definition:
Values (Units): *Read/Write:*
Default: *Saved:*
Notes:

Property: **TL1, TL2**
Definition: The safety system's motor torque warning, fault levels (Puck™ torque units)
Values (Units): *Read/Write:*
Default: *Saved:*
Notes: TL1 is the minimum torque that will cause a warning; TL2 is the minimum torque that will cause a fault.

Property: **VL1, VL2**
Definition: VL1 is the minimum velocity that will cause a warning; VL2 is the minimum velocity that will cause a fault.
Values (Units): (rad/s or m/s) *Read/Write:*
Default: 0.5, 1.0 (respectively) *Saved:*
Notes: Before calling DefineWAMpos() (otherwise known as zeroing the WAM), VL1 and VL2 limit the *angular* velocity (rad/s) of every joint, and the 7-segment LED display on the pendants will show which joint exceeded the velocity limit if a velocity warning or fault occurs. After DefineWAMpos() is called, the WAM switches to limiting the *Cartesian* velocity (m/s) of the elbow and 4-DOF endpoint, instead of each joint's velocity. In the event of a fault after DefineWAMpos() is called, the 7-segment LED display on the pendants will show "E" or "A" to indicate whether the Elbow or the Arm endpoint exceeded the velocity limit. The value of these limits can be changed at any time, but both limits are always applicable to the entire robot – the limits cannot be applied only to particular joints or points on the WAM.
If you want to set these properties directly (using SetProperty()), please note that they are in Q4.12 notation, so take whatever you want the value to be and multiply it by 4096 before you set it. Q4.12 means "4 bits of integer, 12 bits of fraction". 0x1000 = 1.0, 0x1800 = 1.5, 0x7800 = 7.5. If you go much higher, you'll go negative (do not do this). See the source code for SetSafetyLimits() in src/btwam/btwam.c for an example.

Property: **VNOM**
Definition:
Values (Units): *Read/Write:*
Default: *Saved:*
Notes:

Property: **VOLTH1, VOLTH2**
Definition: The safety system's high voltage warning, fault limit
Values (Units): Volts *Read/Write:*
Default: 54, 57 *Saved:*
Notes: If the bus voltage exceeds VOLTH1, the safety system will begin bleeding off bus voltage through a resistor, and the pendants will display a voltage warning with the letter 'H'.
If the bus voltage exceeds VOLTH2, the safety system will disconnect the bus and clamp it, and the pendants will display a voltage fault with the letter 'H'.

Property: **VOLTL1, VOLTL2**
Definition: The safety system's low voltage warning, fault limit.
Values (Units): Volts *Read/Write:*
Default: 36, 30 *Saved:*
Notes: If the bus voltage falls below VOLTL1, the pendants will display a voltage warning with the letter 'L'.
If the bus voltage falls below VOLTL2, the safety system will disconnect the bus and clamp it, and the pendants will display a voltage fault with the letter 'L'.

Property:	**ZERO**
Definition:	After sending the pucks their known initial position, set the safety system's ZERO property to 1 so it can start calculating Cartesian velocities.
Values (Units):	*Read/Write:*
Default:	*Saved:*
Notes:	Use the DefineWAMpos() function in src/btwam/btwam.c to set the initial positions of the other motor controllers. This is called automatically during the normal OpenWAM() call.

6.4 Key to Property Table

Key	Param.
0	VERS
1	ROLE
2	SN
3	ID
4	ERROR
5	STAT
6	ADDR
7	VALUE
8	MODE
9	TEMP
10	PTEMP
11	OTEMP
12	BAUD
13	LOCK
14	DIG0
15	DIG1
16	FET0
17	FET1
18	ANA0
19	ANA1
20	THERM
21	VBUS
22	IMOTOR
23	VLOGIC
24	ILOGIC
25	SG
26	GRPA
27	GRPB
28	GRPC
29	CMD
30	SAVE
31	LOAD
32	DEF
33	FIND
34	X0
35	X1
36	X2
37	X3
38	X4
39	X5
40	X6
41	X7
42	T
43	MT
44	V
45	MV
46	MCV
47	MOV
48	P
49	P2
50	DP
51	DP2
52	E
53	E2
54	OT
55	OT2
56	CT
57	CT2
58	M
59	M2
60	DS
61	MOFST
62	IOFST
63	UPSECS
64	OD
65	MDS
66	MECH
67	MECH2
68	CTS
69	CTS2
70	PIDX
71	HSG
72	LSG
73	IVEL
74	IOFF
75	IOFF2
76	MPE
77	EN
78	TSTOP
79	KP
80	KD
81	KI
82	ACCEL
83	TENST
84	TENSO
85	JIDX
86	IPNM
87	HALLS
88	HALLH
89	HALLH2
90	POLES
91	IKP
92	IKI
93	IKCOR
94	HOLD
95	TIE
96	ECMAX
97	ECMIN
98	LFLAGS
99	LCTC
100	LCVC

7 CANbus Communication Specifications

7.1 Data Link Specifications

1Mbaud CANbus
8 time quanta per bit
75% sampling point
Sync jump width = 1 time quanta (TQ)
11-bit MsgID (standard CAN)
Proprietary protocol, not DeviceNet or CANopen
Recommended reading: *Controller Area Network* by Konrad Etschberger

7.2 CANbus Timing

- 75µS to ask for position
- 75µS per puck to respond with the positions
- 125µS to send a packed torque to the lower 4DOF
- 125µS to send a packed torque to the wrist
- Control-side processing time on PC

For the 4DOF, it is: 75+(4*75)+125+PC = 500µS + PC
For the 7DOF, it is: 75+(7*75)+(2*125)+PC = 850µS + PC

These numbers are limited by the 1 Mbps CANbus. Each message has a 47-bit frame (47µS), plus payload data (3 bytes, 24µS typ). CANbus transceivers are not rated above 1 Mbps due to slew-rate limitations.

7.3 ID Specifications

7.3.1 Message IDs

[GFFFFFTTTTT] (11 bits, binary)
G: Group, 0 = Directed message, 1 = Group broadcast
F: From ID, Host = 00000, Motor N = N
T: To ID or group

Examples:
00000000011 => Directed message from host to motor 3 (3 = 00011, binary)
10001100100 => Group broadcast from motor 3 to group 4

7.3.2 Motor IDs and Groups

Each motor in the robot has a unique communication ID, can be a part of any three groups (GRPA, GRPB, GRPC), and will listen for and process messages bound for its ID or any of its groups. There are a total of 32 possible groups (from 00000 to 11111). Motors 1 to 4 belong to groups 0, 1, and 4, motors 5 to 7 belong to groups 0, 2, and 5, and the host belongs to groups 3 and 6 by default. The default groups are:
0 = All actuators
1 = Lower arm torques (motors 1-4)
2 = Upper arm torques (motors 5-7)
3 = Position feedback
4 = Lower arm property
5 = Upper arm property
6 = Property feedback

7.4 CANbus Frame Data Payload

7.4.1 Standard CANbus Message Format

CAN specifies a maximum of 8 bytes/frame payload – our typical payload consists of 4-6 bytes:
[APPPPPPP] [00000000] [LLLLLLLL] [mmmmmmmm] [MMMMMMMM] [HHHHHHHH]
A: Action, 0 = Get property, 1 = Set property
P: Property (128 possible values, 0..127, 0000000..1111111), see Motor Controller Properties and Safety Module Properties below.
0: Second byte (almost) always set to zero (see exceptions below)
L: Low byte of data value
m: mid-low byte of data value
If sending a 16-bit integer value, the following are not used:
M: Mid-high byte of data value
H: High byte of data value

The CAN frame data length code (DLC) is set to the number of bytes being transmitted.

7.4.2 Exceptions

1) The Position property (P) is a 22-bit, 2's complement number
It is packed into a 3-byte frame payload [00MMMMMM] [mmmmmmmm] [LLLLLLLL]
It is always sent to Group 3

2) Command torque can be sent as a set of four 14-bit, 2's complement numbers
It is sent to the motor controllers in 8 bytes (max):

0	1	2	3	4	5	6	7
APPPPPPP	BBBBBBbb	bbbbbbCC	CCCCcccc	ccccDDDD	DDdddddd	ddEEEEEE	eeeeeeee

A = Action (0:Get 1:Set)
P = Property
B = Upper 6 bits of first value
b = Lower 8 bits of first value
C = Upper 6 bits of second value
c = Lower 8 bits of second value
D = Upper 6 bits of third value
d = Lower 8 bits of third value
E = Upper 6 bits of fourth value
e = Lower 8 bits of fourth value

Each motor has a property (PIDX: 1-4), which tells it which torque to use from the set of 4

7.5 Full Communication Example

This example contains:

- 3 motors with IDs of 5, 6, and 7
- A host with an ID of 0

Host sends:
MsgID [10000000000] → Group 0
Data [10000101] [00000000] [00000010] [0000000] → Set property 5 (STAT) to 2 (STATUS_READY)
The motors start up with STAT = 0 (STATUS_RESET)
Setting STAT to READY gets the motors ready to receive additional data
Motors will only respond to STAT and VERS commands while in RESET (for safety)

Host sends:
MsgID [10000000000] → Group 0
Data [10001000] [00000000] [00000010] [00000000] → Set property 8 (MODE) to 2 (MODE_TORQUE)
The motors default to MODE = 0 (MODE_IDLE)
Setting MODE to MODE_TORQUE tells the motors to apply any torque sent to them
When MODE = MODE_IDLE, motors will ignore any torque commands sent and apply braking
When using a WAM, the safety system will set the MODE when you press the IDLE/ACTIVATE buttons
Do not try to bypass the WAM's safety system by setting the MODE directly, this will cause undesired operation.

Host sends:
MsgID [10000000000] → Group 0
Data [00110000] → Get property 48 (P)

Motors send:
MsgID [10010100011] → From ID 5 to Group 3
Data [00000000] [00000000] [0000010] → My position is 2 encoder counts

MsgID [10011000011] → From ID 6 to Group 3
Data [00000000] [00000000] [0000111] → My position is 7 encoder counts

MsgID [10011100011] → From ID 7 to Group 3
Data [00111111] [11111111] [11111110] → My position is -2 encoder counts

Host uses these positions to calculate a torque.
Host sends:
MsgID [10000000010] → Group 2
Data [10101010] [AAAAAAaa] [aaaaaaBB] [BBBBbbbb] [bbbbCCCC] [CCcccccc] [cc000000] [00000000]
Set torques to new values AAAAAAaaaaaaaa, etc.

8 Troubleshooting

8.1 Checking the Error Log

One of the first steps to diagnosing many WAM problems is to examine the error log file /var/log/syslog. Here is a normal startup log for a 7-DOF WAM:

```
Jul 29 20:27:41 WAM WAM: ...Starting btdiag program...
Jul 29 20:27:44 WAM WAM: Waking all pucks
Jul 29 20:27:44 WAM WAM: getBusStatus(): canReadMsg returned error
Jul 29 20:27:45 WAM last message repeated 22 times
Jul 29 20:27:45 WAM WAM: getBusStatus: Only status != -1 is shown.
Jul 29 20:27:45 WAM WAM: getBusStatus: status[1] = 2
Jul 29 20:27:45 WAM WAM: getBusStatus: status[2] = 2
Jul 29 20:27:45 WAM WAM: getBusStatus: status[3] = 2
Jul 29 20:27:45 WAM WAM: getBusStatus: status[4] = 2
Jul 29 20:27:45 WAM WAM: getBusStatus: status[5] = 2
Jul 29 20:27:45 WAM WAM: getBusStatus: status[6] = 2
Jul 29 20:27:45 WAM WAM: getBusStatus: status[7] = 2
Jul 29 20:27:45 WAM WAM: getBusStatus: status[10] = 2
Jul 29 20:27:45 WAM WAM: About to allocate space for 8 nodes
Jul 29 20:27:45 WAM WAM: getBusStatus(): canReadMsg returned error
Jul 29 20:27:45 WAM last message repeated 22 times
Jul 29 20:27:45 WAM WAM: getBusStatus: Only status != -1 is shown.
Jul 29 20:27:45 WAM WAM: getBusStatus: status[1] = 2
Jul 29 20:27:45 WAM WAM: getBusStatus: status[2] = 2
Jul 29 20:27:45 WAM WAM: getBusStatus: status[3] = 2
Jul 29 20:27:45 WAM WAM: getBusStatus: status[4] = 2
Jul 29 20:27:45 WAM WAM: getBusStatus: status[5] = 2
Jul 29 20:27:45 WAM WAM: getBusStatus: status[6] = 2
Jul 29 20:27:45 WAM WAM: getBusStatus: status[7] = 2
Jul 29 20:27:45 WAM WAM: getBusStatus: status[10] = 2
Jul 29 20:27:45 WAM WAM: Puck: ID=1 CTS=4096 IPNM=2700.00 PIDX=0 GRPB=1
Jul 29 20:27:45 WAM WAM: Puck: ID=2 CTS=4096 IPNM=2562.00 PIDX=1 GRPB=1
Jul 29 20:27:45 WAM WAM: Puck: ID=3 CTS=4096 IPNM=2562.00 PIDX=2 GRPB=1
Jul 29 20:27:45 WAM WAM: Puck: ID=4 CTS=4096 IPNM=2700.00 PIDX=3 GRPB=1
Jul 29 20:27:45 WAM WAM: Puck: ID=5 CTS=4096 IPNM=4961.00 PIDX=0 GRPB=2
Jul 29 20:27:45 WAM WAM: Puck: ID=6 CTS=4096 IPNM=4961.00 PIDX=1 GRPB=2
Jul 29 20:27:45 WAM WAM: Puck: ID=7 CTS=4096 IPNM=17474.00 PIDX=2 GRPB=2
Jul 29 20:27:45 WAM WAM: Actuator data dump:
Jul 29 20:27:45 WAM WAM: [0]:Bus-0,ID-1,G-1,O-0,M-0,Off-0,Enc-4096
Jul 29 20:27:45 WAM WAM: [1]:Bus-0,ID-2,G-1,O-1,M-0,Off-0,Enc-4096
Jul 29 20:27:45 WAM WAM: [2]:Bus-0,ID-3,G-1,O-2,M-0,Off-0,Enc-4096
Jul 29 20:27:45 WAM WAM: [3]:Bus-0,ID-4,G-1,O-3,M-0,Off-0,Enc-4096
Jul 29 20:27:45 WAM WAM: [4]:Bus-0,ID-5,G-2,O-0,M-0,Off-0,Enc-4096
Jul 29 20:27:45 WAM WAM: [5]:Bus-0,ID-6,G-2,O-1,M-0,Off-0,Enc-4096
Jul 29 20:27:45 WAM WAM: [6]:Bus-0,ID-7,G-2,O-2,M-0,Off-0,Enc-4096
Jul 29 20:27:45 WAM WAM: Data for Bus 0
Jul 29 20:27:45 WAM WAM: Bus Data: There were 7 Pucks sorted by ID
Jul 29 20:27:45 WAM WAM: [0]: Actuator 0 Puck 1
Jul 29 20:27:45 WAM WAM: [1]: Actuator 1 Puck 2
Jul 29 20:27:45 WAM WAM: [2]: Actuator 2 Puck 3
Jul 29 20:27:45 WAM WAM: [3]: Actuator 3 Puck 4
Jul 29 20:27:45 WAM WAM: [4]: Actuator 4 Puck 5
Jul 29 20:27:45 WAM WAM: [5]: Actuator 5 Puck 6
Jul 29 20:27:45 WAM WAM: [6]: Actuator 6 Puck 7
Jul 29 20:27:45 WAM WAM: Bus Data: There were 2 Groups
Jul 29 20:27:45 WAM WAM: Group 1: A0 P1 A1 P2 A2 P3 A3 P4
Jul 29 20:27:45 WAM WAM: Group 2: A4 P5 A5 P6 A6 P7 A-1 P0
Jul 29 20:27:45 WAM WAM: device_name=WAM7
Jul 29 20:27:45 WAM WAM: wam->name=WAM7
Jul 29 20:27:45 WAM WAM: bus=0, num_actuators=7
```

```
Jul 29 20:27:45 WAM WAM: OpenWAM(): for link from 0 to 7
Jul 29 20:27:45 WAM WAM: parseGetVal: key not found [WAM7.link[7].rotorI]
Jul 29 20:27:45 WAM WAM: Motor[0] is joint 0
Jul 29 20:27:45 WAM WAM: Motor[1] is joint 1
Jul 29 20:27:45 WAM WAM: Motor[2] is joint 2
Jul 29 20:27:45 WAM WAM: Motor[3] is joint 3
Jul 29 20:27:45 WAM WAM: Motor[4] is joint 4
Jul 29 20:27:45 WAM WAM: Motor[5] is joint 5
Jul 29 20:27:45 WAM WAM: Motor[6] is joint 6
Jul 29 20:27:45 WAM WAM: WAM zeroed by application
Jul 29 20:27:45 WAM WAM: About to set safety limits, VL2 = 46
Jul 29 20:27:45 WAM WAM: WAMControl period Sec:0.002000, ns: 2000000
```

8.2 Common Problems

The symptoms repeated in this section were either generated by Barrett's own lab WAMs or were reported by Barrett's customers.

Problem: **Can not log in to the WAM PC**

Reason(s):

1. Someone changed the password
   Solution(s):
   a) Ask the people in your lab for the new password
2. The drive is full (external PC)
   Solution(s):
   a) If the drive is full, the system is not able to record your login event, and the login will fail. It is likely that the /var/log/syslog files are very large, especially if you are logging data or errors from within the WAM control loop. Boot from a bootable CD (like sysresccd.org), and delete the syslog files (or other files) to make some room.
3. The network connection is not active (logging in over the network)
   Solution(s):
   a) Make sure the Ethernet cable is plugged into the PC
   b) Make sure your DHCP server is active (if using DHCP)
   c) Check that the Ethernet driver is loaded (from local terminal): lspci |grep –i eth; dmesg |grep –i eth; lsmod
4. The drive is corrupted or damaged due to jarring, poor ventilation, or a software bug.
   a) Try a new hard drive (external PC). O/S installation instructions are on wiki.barrett.com.
   b) Try a new CompactFlash (internal PC). Contact Barrett for a replacement.

Problem: **WAM control application crashes/segfaults**

Reason(s):

1. The WAM PC can not communicate with the Safety Board at program launch
   Solution(s):
   a) Turn on the WAM power supply.
   b) Make sure that the CANbus cable is securely plugged in.
   c) Make sure that the CANbus cable is plugged into the correct port on the PC (external WAM PCs)
   d) Make sure that the CANbus cable is properly terminated- there is a wrist or blank outer link attached.
   e) Make sure the CANbus card is properly seated in its PCI slot (external WAM PCs)
   f) Make sure the CAN driver is installed and loaded correctly: dmesg |grep –i pcan, cat /proc/pcan, lsmod
   g) Check for a broken CANbus or power wire, usually in a connector or at the safety puck.
   h) Check for a loose CANbus or power crimp in the connectors.

i) Attach the Puck serial cable to the safety board to verify that the safety puck is functional. Could it have overheated?

2. You are accessing data outside of the program's memory space (not likely with standard example programs).
   Solution(s):
   a) Use gdb to determine the offending line of source code.

Problem: **The pendants do not initialize when the WAM power supply is turned on**
Reason(s):

1. The safety board fuse is blown
   Solution(s):
   a) Remove, test, and replace the fuse on the safety board
2. There is no power
   Solution(s):
   a) Check for proper voltage at the source (wall or battery)
   b) Check for any custom current-limiting circuits you may be using
   c) Check the output voltage of the power supply (should be 48 VDC, nominal)
3. There is some other electrical problem
   Solution(s):
   a) Contact Barrett for repair

Problem: **The pendant initialization sequence is incorrect when the WAM power supply is turned on**
Reason(s):

1. The Data/Clock/Latch signals to the pendants are weak
   Solution(s):
   a) Make sure both pendant cables are plugged securely into the WAM
2. There is some other electrical problem
   Solution(s):
   a) Contact Barrett for repair

Problem: **Some joints have resistive braking, some do not. The angles that btdiag returns for the joints without resistive braking are incorrect. This is easiest to check for at the home position.**
Reason(s):

1. The Pucks without resistive braking are not powering up correctly. NOTE: It is normal to have no resistive braking in all joints after turning on the WAM and pressing Shift-Idle, but before you launch a WAM control program.
   Solution(s):
   a) Check for a broken CANbus or power wire, usually in a connector near the Puck.
   b) Check for a loose CANbus or power crimp in the connectors near the Puck.
   c) Check for a loose CANbus or power wire at the Puck itself.
   d) Check the syslog for clues (compare it with the syslog above)
   e) Contact Barrett for repair

Problem: **Joint readings "bounce" from reasonable/actual values to values that are not/cannot be true and back again.**
Reason(s):

1. Realtime control violation
   Solution(s):
   a) Make sure the control loop avoids system calls that are incompatible with realtime control: No UI such as printf() or getch(), no I/O such as read() or write()- just about anything except for loops and pure math.
   b) Make sure your total WAM control loop time, including the WAMCallback(), does not exceed the time allowed by the control rate. Keep in mind that the data time-of-flight

on the CANbus is approximately 850 uS for a 7-DOF and 500 uS for a 4-DOF. This takes up a significant amount of the total control loop time.

c) If you built your own WAM PC, a system hardware interrupt might be causing a realtime glitch. Check your installation against: http://www.rtai.dk/cgi-bin/gratiswiki.pl?Latency_Killer

d) Extra getProperty() or setProperty() calls could be exceeding the CANbus bandwidth. setProperty() takes 75 uS without verification, getProperty() takes 150 uS. You might try staggering these function calls across multiple control cycles.

2. CANbus communication problem
   Solution(s):
   a) Use the btutil utility application to verify that the Puck firmware versions match for all motor Pucks. Load matching firmware onto all motor Pucks, if necessary.
   b) Ensure that CANbus is properly terminated to minimize signal reflections. There is a 120 Ohm termination resistor in the wrist module (7-DOF) and blank outer link (4-DOF). If you are using the internal WAM PC, switches 1-2 and 1-3 should be "out" (see Section 3.5). This terminates the CANbus at the Safety Board with a 120 Ohm resistor. If you are using an external WAM PC, the purple CANbus cable has a 120 Ohm resistor in the connector at the PC end. In this case, switches 1-2 and 1-3 should be "in".
   c) Use btutil to ensure that there are no conflicting Puck IDs on the CANbus. Each Puck must have a unique ID. If the Pucks are not enumerating correctly, disconnect them from the CANbus on-by-one until you find the one causing the conflict. Use btutil to set a new Puck ID for that puck.
3. Grounding problem
   Solution(s):
   a) Eliminate CANbus ground loops. The CANbus shield is connected to Earth ground on the Safety Board. If your PC is also grounded, you should use an opto-isolated CANbus card to prevent ground loops.
   b) Eliminate power bus ground loops. The frame of the WAM is connected to Earth ground through the blue DC power cable. The frame of the 48 VDC supply is also Earth-grounded. If there is an additional electrical connection between the frame of the WAM and the frame of the power supply (such as a metal table or mounting bracket), this can cause a ground loop and undesired operation.

Problem: **Running "sh makeall" causes a number of error messages appear, including "cannot find –lntcan".**

Reason(s):

1. The linker is looking for the wrong CAN driver
   Solution(s):
   a) Edit your btclient/config.mk file to specify the correct CAN driver
2. The CAN driver is not installed
   Solution(s):
   a) Install the correct CAN driver for your CAN hardware

Problem: **A mechanical cable is fraying, birdnesting, or has come loose**

Reason(s):

1. The maximum recommended payload was exceeded
   Solution(s):
   a) Note that the payload specification does not take into account accelerated loads. If you attach a 3 kg load to the 4-DOF and accelerate it at 2 g's, the WAM experiences a 6 kg load, exceeding the recommended payload. The cable should be replaced before further use.
2. A brass termination slipped off the end of the cable.
   Solution(s):
   a) This is a manufacturing defect. Contact Barrett to get the cable replaced.

Problem: **During operation, you hear a "popping" sound accompanied by a distinct nasty smell.**

Reason(s):

1. An electrical component burned up.
   Solution(s):
   a) Carefully record all events leading up to the failure. Contact Barrett for repair.

Problem: **The WAM returns to home position before starting a new trajectory.**

Reason(s):

1. The first point in the trajectory is the home position.
   Solution(s):
   a) Inspect the trajectory file with a text editor to determine if this is the case. Every trajectory file is just a comma-delimited file with time (in seconds) in the left column and (depending on the mode) the joint positions or the Cartesian positions in the remaining columns.

Problem: **A slight knocking can be heard when moving a particular cable circuit of the WAM (it may also be felt if backdriving the WAM by hand). As the knock occurs, a slight backlash may be felt.**

Reason(s):

1. Debris in a joint bearing.
   Solution(s):
   a) Apply several drops of oil (3-in-1, or WD-40 in liquid form or another light viscosity mineral oil should work). The debris has about a 50% chance of working itself out eventually.
2. Loose ball-bearing retainer
   Solution(s):
   a) Eliminated in newer designs, usually does not interfere with operation except for clicking noise.
3. Insufficient motor shaft axial preload/shimming – the rotor/shaft should not be able to move axially inside the motor.
   Solution(s):
   a) Return to Barrett for repair.

Problem: **The robot fails to follow scaled trajectories.**

Reason(s):

1. The WAM does not have the latest OS/firmware/software.
   Solution(s):
   a) Follow the directions on wiki.barrett.com to update your firmware/software.

Problem: **Gravity Compensation does not operate correctly.**

Reason(s):

1. The wam.conf file links to the incorrect configuration file based on the present WAM setup
   Solution(s):
   a) Change the link in wam.conf to the correct configuration file: WAM4 (4-DOF), WAM7 (4DOF with Wrist), or WAMG (4DOF with gimbals)
2. The parameters in the configuration file do not match your system's actual configuration.
   Solution(s):
   a) Check the linked configuration file for the following:
   - Correct the tool center point (DH parameters d, a, alpha). These are offsets from the final link's frame.
   - Correct the tool Center-Of-Mass (COM), relative to the tool center point
   - Correct the tool mass
   - In pre-2007 WAMs, Motor 4 (M4) was located over M3. After 2007, M4 was reversed and located over M2. The transformation matrices need to match your

WAM's M4 configuration. Basically, the sign changed from + to – for the M4 transmission.

- The latest files in config/ will not necessarily match your WAM. You should always use the config file that shipped with your WAM. Update that file with *new features* from the latest config files, but be careful not to change the kinematic and dynamic constants that are specific to your WAM.

3. Mass parameters are inaccurate due to model errors, unmodeled parts, or machinist tolerances.
   Solution(s):
   a) On-line calibration (either static or dynamic) can yield better data than the model. Barrett is working on calibration software and will notify the WAM User List when it is released.
4. Electrical wiring stiffness requires additional joint torque to overcome pulling effect near joint limits
   Solution(s):
   a) This requires complex modeling algorithms that Barrett has not yet developed.
5. We depend on a sampled motor torque constant to be accurate across a batch of motors in a manufacturing run. In fact, the torque constants could vary slightly.
   Solution(s):
   a) Calibration software could reduce errors due to inaccurate torque constants. Barrett is working on calibration software and will notify the WAM User List when it is released.

Problem: **Safety Board is randomly rebooting. The pendants reinitialize themselves.**
Reason(s):

1. There are electrical noise spikes/dips in the power circuitry of the Safety Board.
   Solution(s):
   a) Contact Barrett for a replacement board.

Problem: **Pendant lights do not come on (specifically the IDLE light), but Pucks are online anyways, but do not make PWM sound when <SHIFT+ACTIVATE> is pressed.**
Reason(s):

1. The pendant is electrically damaged
   Solution(s):
   a) Contact Barrett for repair.

Problem: **One or more joints of the WAM vibrate during trajectory following.**
Reason(s):

1. The PID control gains for the joint are too high. The default PID control gains assume some non-zero payload in gravity. If you have no payload, or the joint is facing the floor or the ceiling (that is, if gravity is not a factor and there is no externally-applied torque on the joint), then the joint control gains could be too high.
   Solution(s):
   a) Lower the PID gains for that joint (in the wam.conf file). Start by cutting P and D in half.

Problem: **One or more joints of the WAM oscillate or otherwise go unstable during trajectory following.**
Reason(s):

1. The PID control gains for the joint are too low.
   Solution(s):
   a) Raise the PID gains for that joint (in the wam.conf file).
2. The payload is too heavy, the motor torque is saturating at Max Torque (MT), and the control can not keep up.
   Solution(s):

a) Reduce the payload. Remember that you must take into account accelerated loads while staying under the max payload specification. For example, the max payload for the 7DOF is 3 kg. This means you can accelerate 1.5 kg at 2G. Or you can hold 3 kg statically at the max reach.
b) Increase the MT parameter for the motors. However, the default MT values are set to stay within the breaking strength of the steel cables. Increasing these values increases the likelihood of cable failure.

3. The simple PID control that ships with the example code is not robust enough to handle loads offset significantly from the tool center point or loads with unusual inertial characteristics.
   Solution(s):
   a) Design a more robust control method for your application

Problem: **WAM will not enter Idle mode—the WAM starts with a low voltage fault, and when <Shift+Reset/Idle> is pressed, the pendants reinitialize instead of entering Idle mode.**

Reason(s):

1. The Puck power bus is shorted together. This could be due to a stuck relay, loose wire, metallic debris on Safety Board, or a damaged Puck.
   Solution(s):
   a) Contact Barrett for repair

9 Theory of Operation

9.1 Electronic Architecture

Figure 33 – WAM System Components

The following major components are part of your WAM system:

- The WAM Arm robot with embedded motor controllers (Pucks™).
- An internal PC/104 form-factor computer loaded with the Linux operating system.
- An internal safety system.
- An external power supply for the WAM.
- A Control Pendant and Display Pendant.
- (Optional) an external PC loaded with Linux, a CAN communications card, a CANbus cable

Figure 33 shows a schematic of the WAM system subcomponents that are important to the WAM programmer. The power supply is used with AC outlets, as specified in Section 3.6.1. It provides 48 VDC to the safety module which in turn provides the necessary voltage to the motors and the internal PC.

The internal computer is responsible for closing the control loop and providing high-level command of the WAM system. The control loop is the repeated reading of motor angles and commanding of motor torques to provide a smooth and safe motion of the WAM Arm robot. High-level command is deciding when and where to move to robot and providing an interface for the user of the WAM system to accomplish their task.

The external PC fulfills the same function as the internal computer, but it provides extra processing power and hard drive space. The external PC uses CAN communication to connect to the WAM. If the external PC is used, communications and commands bypass the internal PC.

The WAM safety system consists of the pendants and the safety board/module, and it monitors joint torque and velocity, communication between the computer and the Pucks™ (heartbeats), voltage levels to the WAM, and safety states (see Section 2 for details). The safety module will shut down the WAM and cause the motors to resistively brake if it senses a dangerous situation (see Section 2.2).

Barrett Technology has created specially designed motor controllers called "Pucks™", which are mounted directly to the motors they control. A Puck™ serves as a power source for the motors and commands a smooth, continuous torque based on the digital torque command. It also serves as an encoder for the motor angle, a power amplifier, and contains precision internal current and temperature sensing as well as high-speed CANbus communications.

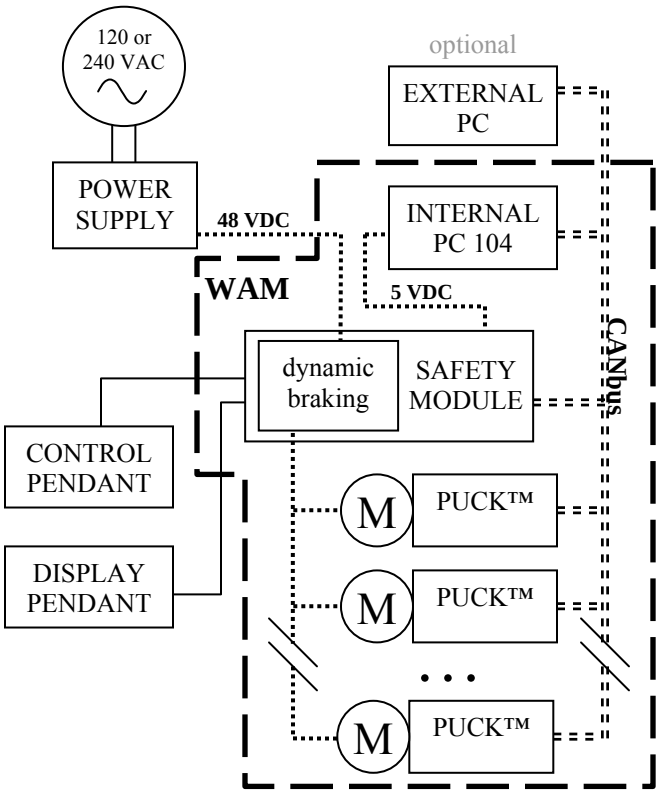

Figure 34 – WAM System Schematic

9.2 Kinematics, Transmission Ratios, and Joint Ranges

9.2.1 4 DOF and 7 DOF

A good introduction to coordinate frames, transformations and kinematics is beyond the scope of this document. There are several good introductory robotics books available. We recommend *Spong, M.; Hutchinson, S.; Vidyasagar, M. Robot Modeling and Control; 2006 John Wiley & Sons* as we use the variant of the Denavit-Hartenberg (D-H) method that is from this book to establish the coordinate frames.

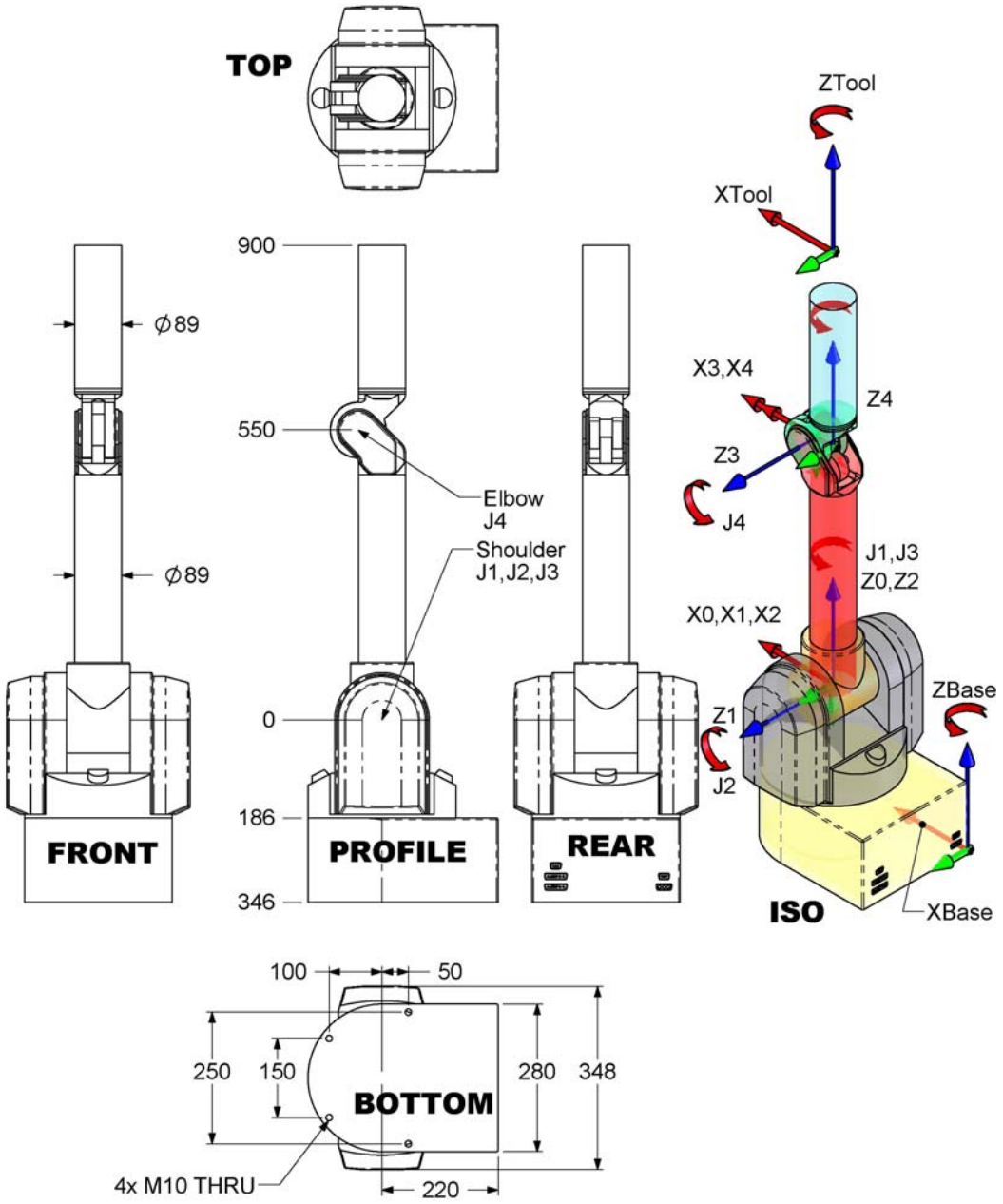

Figure 35 – WAM 4-DOF dimensions and D-H frames

Figure 35 shows the 4-DOF WAM system in the zero position. A positive joint motion is based on the right hand rule for each axis. Figure 37 through Figure 40 on the following pages show explicitly each of the four joint kinematic parameters and joint limits.

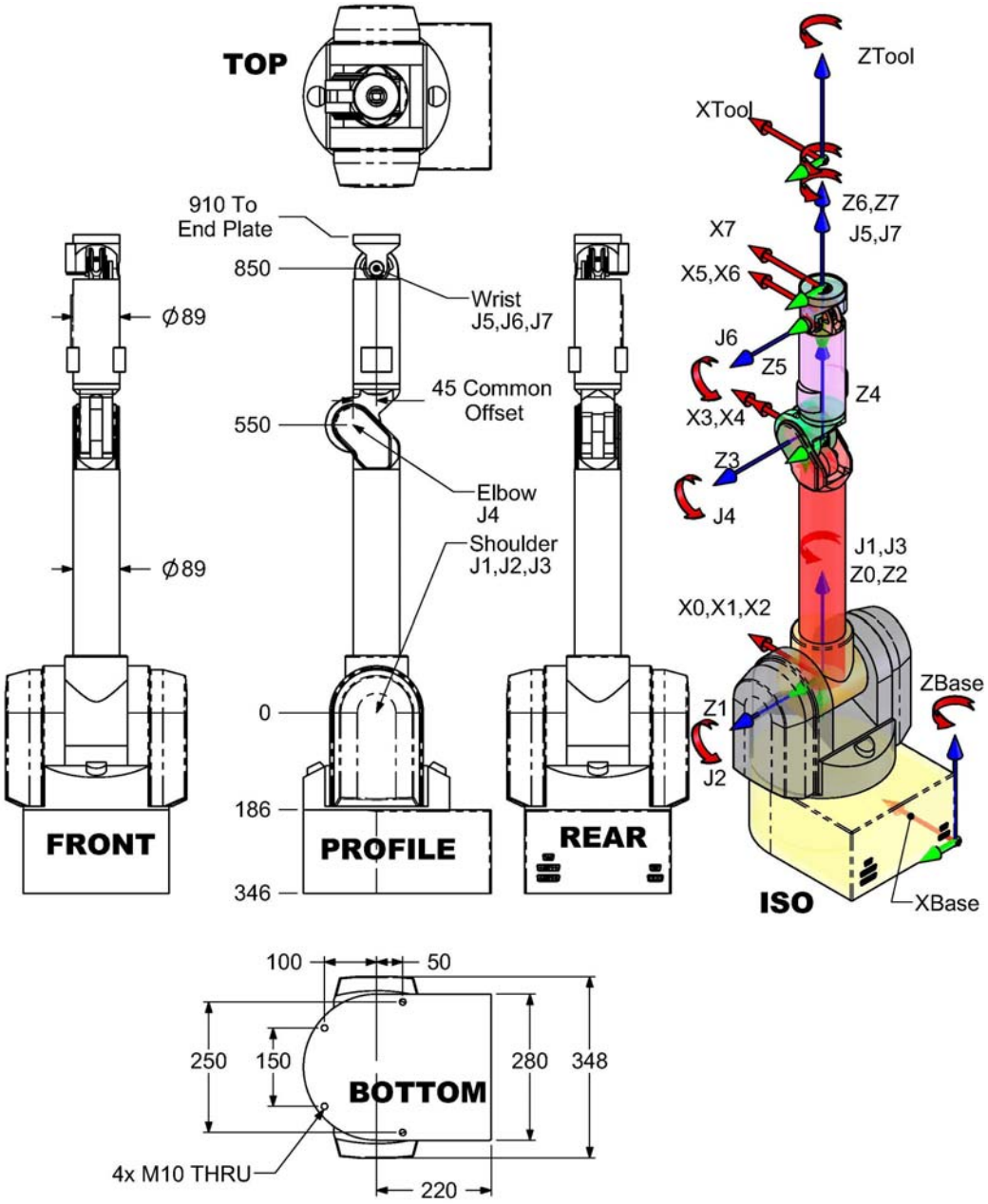

Figure 36 – WAM 7-DOF dimensions and D-H frames

Figure 36 shows the entire 7-DOF WAM system in the zero position. A positive joint motion is based on the right hand rule for each axis. Figure 37 through

Figure **43** on the following pages show explicitly each of the seven joint kinematic parameters and joint limits.

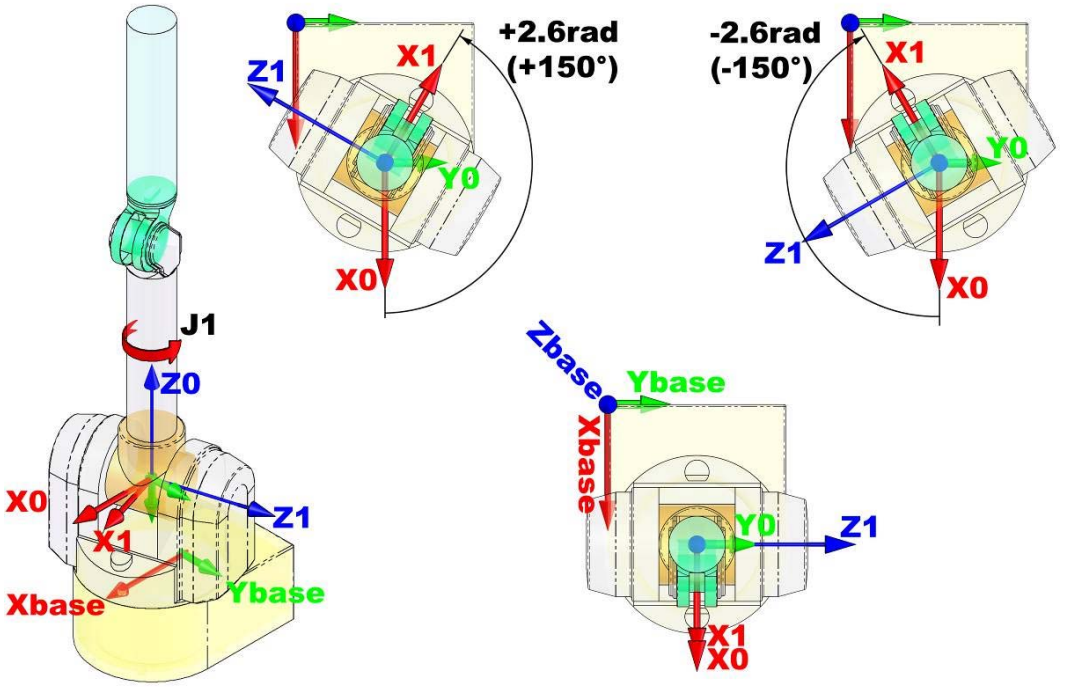

Figure 37 – WAM Arm Joint 1 Frames and Limits

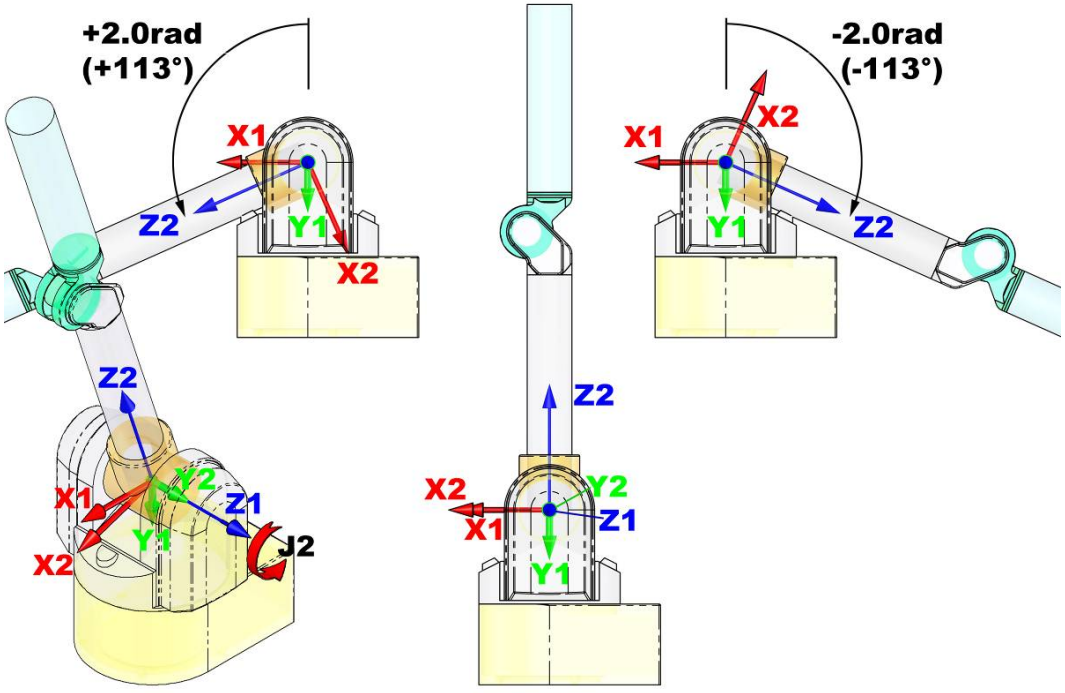

Figure 38 – WAM Arm Joint 2 Frames and Limits

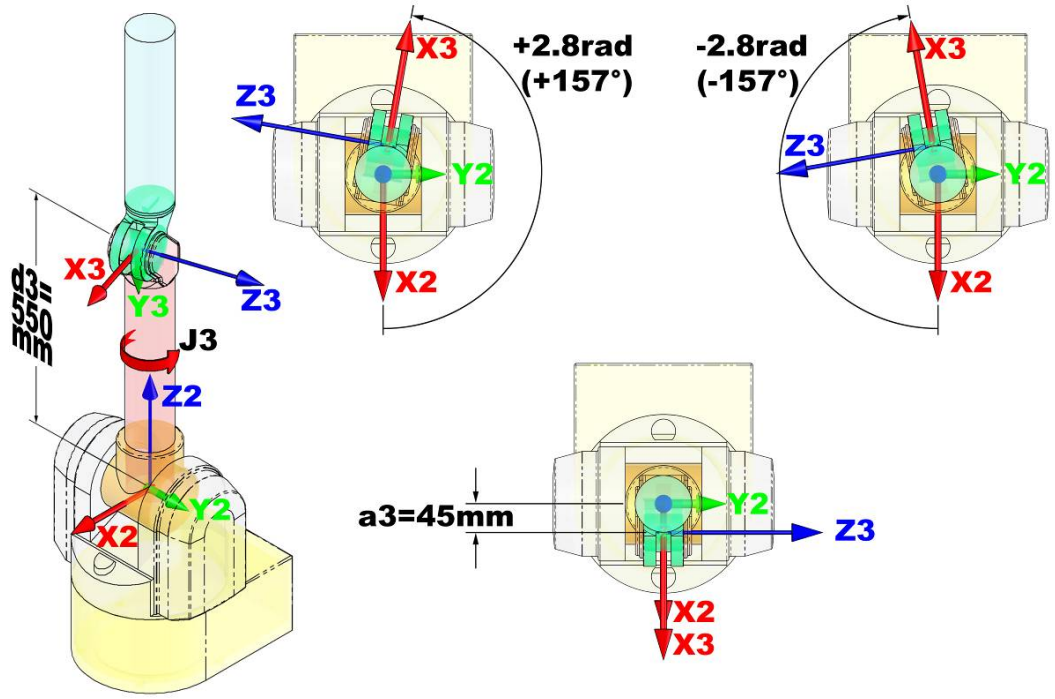

Figure 39 – WAM Arm Joint 3 Frames and Limits

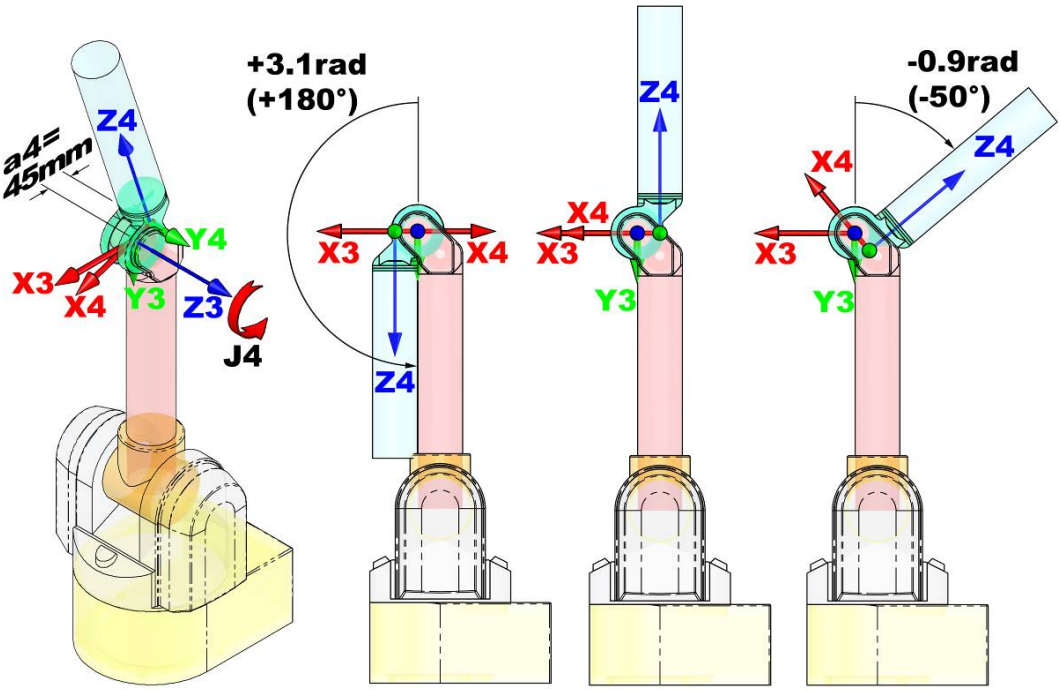

Figure 40 – WAM Arm Joint 4 Frames and Limits

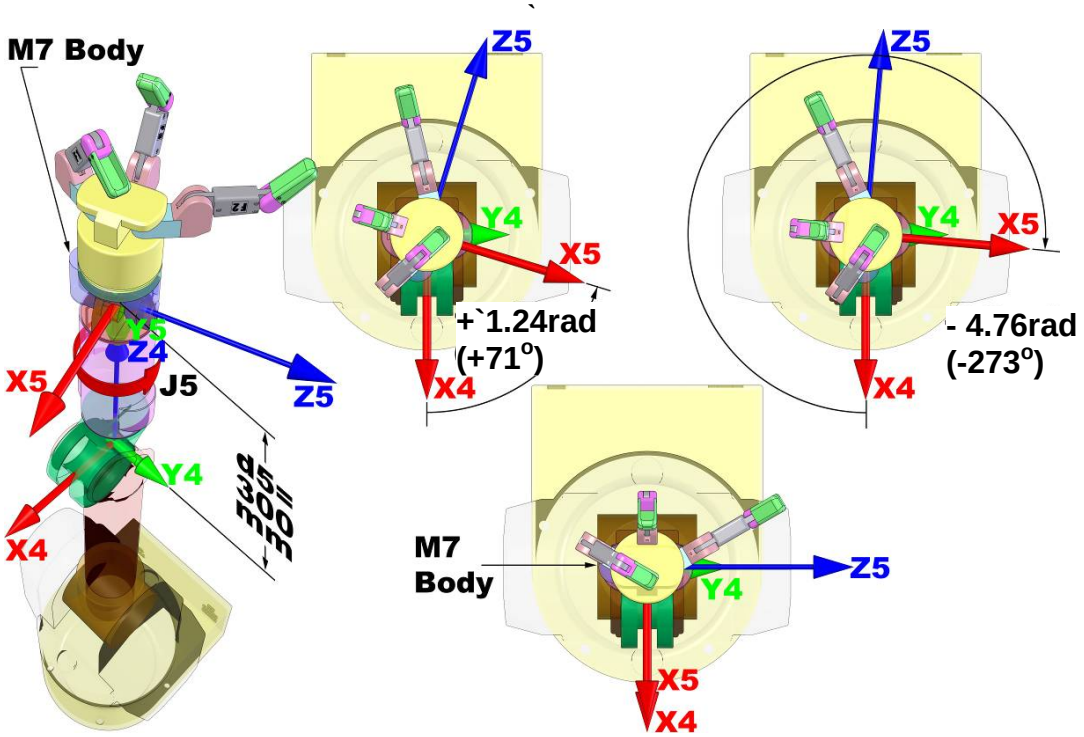

Figure 41 – WAM Arm Joint 5 Frames and Limits

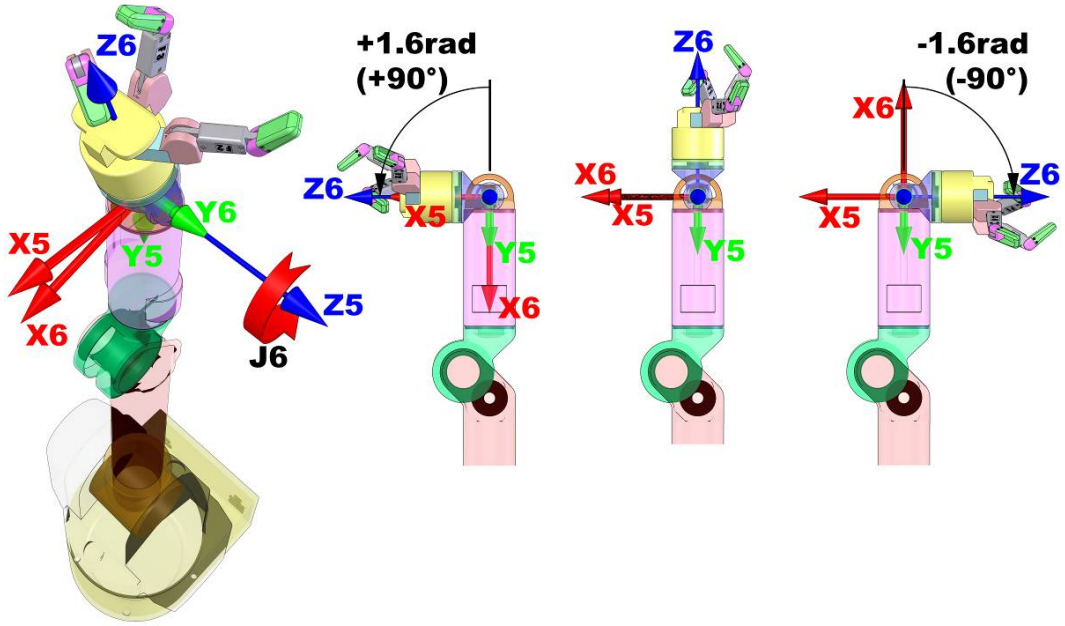

Figure 42 – WAM Arm Joint 6 Frames and Limits

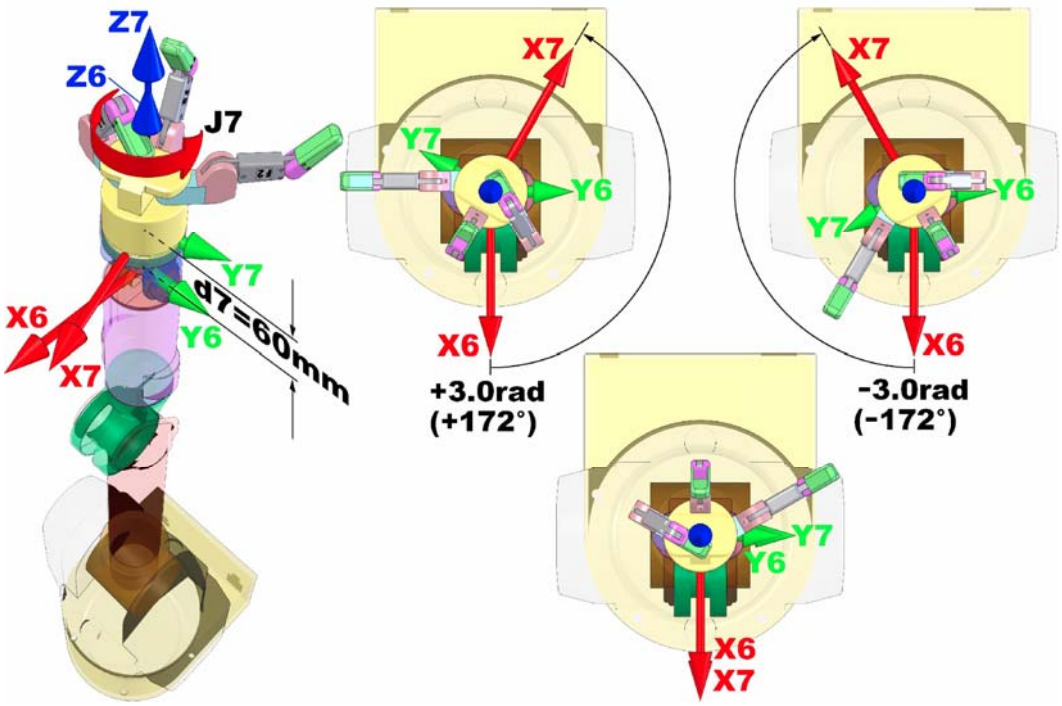

Figure 43 – WAM Arm Joint 7 Frames and Limits

Equation 1 below gives the transform between two adjacent D-H coordinate frames. The D-H parameters that were derived from this equation are located in Table 4 and Table 5 below. Note that ***c*** and ***s*** stand for *cos* and *sin* respectively.

$$ {}^{i-1}\mathrm{T}_i = \begin{pmatrix} \mathrm{c}\theta_i & -\mathrm{s}\theta_i\mathrm{c}\alpha_i & \mathrm{s}\theta_i\mathrm{s}\alpha_i & \mathrm{a}_i\mathrm{c}\theta_i \\ \mathrm{s}\theta_i & \mathrm{c}\theta_i\mathrm{c}\alpha_i & -\mathrm{c}\theta_i\mathrm{s}\alpha_i & \mathrm{a}_i\mathrm{s}\theta_i \\ 0 & \mathrm{s}\alpha_i & \mathrm{c}\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Equation 1 – D-H generalized transform matrix

Table 4 – 4-DOF WAM frame parameters (with blank outer link installed)

i	a_i	α_i	d_i	θ_i
1	0	–π/2	0	θ_1
2	0	π/2	0	θ_2
3	0.045	–π/2	0.55	θ_3
4	–0.045	π/2	0	θ_4
T	0	0	0.35	

Notes: – Units of meters and radians
– T = Tool frame

Table 5 – 7-DOF WAM frame parameters

i	a_i	α_i	d_i	θ_i
1	0	–π/2	0	θ_1
2	0	π/2	0	θ_2
3	0.045	–π/2	0.55	θ_3
4	–0.045	π/2	0	θ_4
5	0	–π/2	0.3	θ_5
6	0	π/2	0	θ_6
7	0	0	0.06	θ_7
T	0	0	0	

For example, to generate the transform from coordinate Frame 2 to coordinate Frame 1 (i.e. the position and orientation of Frame 2 described in terms of Frame 1 which is also a rotation about joint 2), use the parameters in the second row of Table 4 as follows:

$$ {}^{1}\mathrm{T}_2 = \begin{pmatrix} \cos(\theta_2) & -\sin(\theta_2)\cos(\pi/2) & \sin(\theta_2)\sin(\pi/2) & \cos(\theta_2)(0) \\ \sin(\theta_2) & \cos(\theta_2)\cos(\pi/2) & -\cos(\theta_2)\sin(\pi/2) & \sin(\theta_2)(0) \\ 0 & \sin(\pi/2) & \cos(\pi/2) & (0) \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ {}^{1}\mathrm{T}_2 = \begin{pmatrix} \cos(\theta_2) & 0 & \sin(\theta_2) & 0 \\ \sin(\theta_2) & 0 & -\cos(\theta_2) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Equation 2 – D-H Matrix Example

Each of the joints has a mechanical stop that limits the motion. Refer to Table 6 below for a complete listing of the joint limits for each axis.

Table 6 – Joint Limits

Joint	Positive Joint Limit *Rad (deg)*	Negative Joint Limit *Rad (deg)*
1	2.6 (150)	-2.6 (-150)
2	2.0 (113)	-2.0 (-113)
3	2.8 (157)	-2.8 (-157)
4	3.1 (180)	-0.9 (-50)
5	1.24 (71)	-4.76 (-273)
6	1.6 (90)	-1.6 (-90)
7	3.0 (172)	-3.0 (-172)

Forward Kinematics for the 4-DOF WAM

The forward kinematics of the 4-DOF WAM system is used to determine the end tip location and orientation. These transformations are generated using the parameters in Table 4 on page 64 and the matrix in Equation 1 on page 64.

$$ {}^{4}\mathrm{T}_{Tool} = \begin{pmatrix} u_x & v_x & w_x & p_x \\ u_y & v_y & w_y & p_y \\ u_z & v_z & w_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Equation 3 – Tool frame matrix

You define the ${}^{4}\mathrm{T}_{Tool}$ frame for your specific end-effector. The forward kinematics are determined for any frame on the robot by mulitplying all of the transforms up to and including the final frame. To determine the tool end tip location and orientation use the following equation:

$$ {}^{0}\mathrm{T}_{Tool} = {}^{0}\mathrm{T}_{1}\,{}^{1}\mathrm{T}_{2}\,{}^{2}\mathrm{T}_{3}\,{}^{3}\mathrm{T}_{4}\,{}^{4}\mathrm{T}_{Tool} $$

Equation 4 – Tool end tip position and orientation equation for the 4-DOF WAM

Forward Kinematics for the 7-DOF WAM

As with the previous example, you define the ${}^{7}\mathrm{T}_{Tool}$ frame for your specific end-effector. The forward kinematics are determined for any frame on the robot by mulitplying all of the transforms up to and including the final frame. To determine the end tip location and orientation use the following equation:

$$ {}^{0}\mathrm{T}_{Tool} = {}^{0}\mathrm{T}_{1}\,{}^{1}\mathrm{T}_{2}\,{}^{2}\mathrm{T}_{3}\,{}^{3}\mathrm{T}_{4}\,{}^{4}\mathrm{T}_{5}\,{}^{5}\mathrm{T}_{6}\,{}^{6}\mathrm{T}_{7}\,{}^{7}\mathrm{T}_{Tool} $$

Equation 5 – Tool end tip position and orientation equation for the 7-DOF WAM

9.2.2 4 DOF with Gimbals

A 4-DOF WAM Arm can be outfitted with optional 3-axis non-motorized gimbals that give precise angular feedback. The kinematics of the first 4 joints is identical to a 4-DOF WAM Arm. The kinematics for the additional 3 axes is shown in Figure 44.

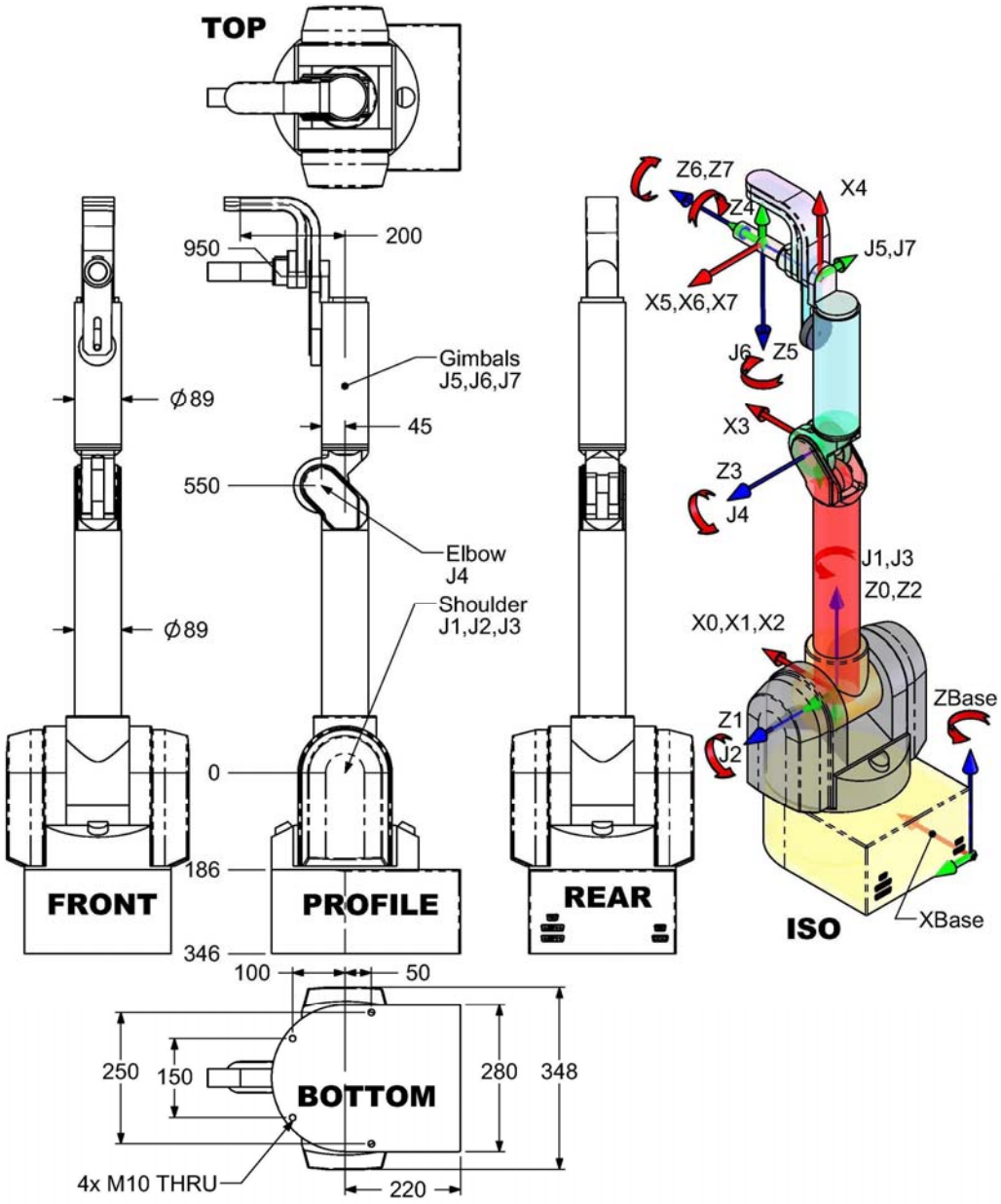

Figure 44 – Denavit-Hartenberg Frames – 4-DOF + Gimbals

Figure 44 shows the WAM Gimbals in the zero position. A positive joint motion is based on the right hand rule for each axis. The D-H parameters that were derived from Figure 44 are located in Table 7 below.

Table 7 – 4-DOF WAM + Gimbals DH Parameters

i	a_i	α_i	d_i	θ_i
1	0	$-\pi/2$	0	θ_1
2	0	$\pi/2$	0	θ_2
3	0.045	$-\pi/2$	0.55	θ_3
4	0.4	$-\pi/2$	0	θ_4
5	0	$\pi/2$	0	$\theta_5-\pi/2$
6	0	$-\pi/2$	0.155	$\theta_6-\pi/2$
7	0	0	0	θ_7

9.2.3 Motor-to-Joint Transformations

Motor-to-Joint Position Transformations

The following transformations show the change in joint positions as a function of motor positions. The input transmission ratios and the differential transmission ratios are calculated from known pulley, pinion, and cable diameters.

Table 8 – Arm Transmission Ratios

Parameter	Value
N_1	42.0
N_2	28.25
N_3	28.25
n_3	1.68
N_4	18.0
N_5	9.48
N_6	9.48
N_7	14.93
n_6	1

$$\begin{pmatrix} J\theta_1 \\ J\theta_2 \\ J\theta_3 \\ J\theta_4 \end{pmatrix} = \begin{pmatrix} \frac{-1}{N_1} & 0 & 0 & 0 \\ 0 & \frac{1}{2N_2} & \frac{-1}{2N_2} & 0 \\ 0 & \frac{-n_3}{2N_2} & \frac{-n_3}{2N_2} & 0 \\ 0 & 0 & 0 & \frac{-1}{N_4} \end{pmatrix} \begin{pmatrix} M\theta_1 \\ M\theta_2 \\ M\theta_3 \\ M\theta_4 \end{pmatrix}$$

Equation 6 – WAM Motor-to-Joint position transformations

$$\begin{pmatrix} J\theta_5 \\ J\theta_6 \\ J\theta_7 \end{pmatrix} = \begin{pmatrix} \frac{1}{2N_5} & \frac{1}{2N_5} & 0 \\ \frac{-n_6}{2N_5} & \frac{n_6}{2N_5} & 0 \\ 0 & 0 & \frac{-1}{N_7} \end{pmatrix} \begin{pmatrix} M\theta_5 \\ M\theta_6 \\ M\theta_7 \end{pmatrix}$$

Equation 7 – Wrist Motor-to-Joint position transformations

The motor position can also be derived from joint space by taking the inverse of the multiplying matrix. For convenience, they are as follows:

$$\begin{pmatrix} M\theta_1 \\ M\theta_2 \\ M\theta_3 \\ M\theta_4 \end{pmatrix} = \begin{pmatrix} -N_1 & 0 & 0 & 0 \\ 0 & N_2 & \frac{-N_2}{n_3} & 0 \\ 0 & -N_2 & \frac{-N_2}{n_3} & 0 \\ 0 & 0 & 0 & -N_4 \end{pmatrix} \begin{pmatrix} J\theta_1 \\ J\theta_2 \\ J\theta_3 \\ J\theta_4 \end{pmatrix}$$

Equation 8 – Arm Joint-to-Motor position transformations

$$\begin{pmatrix} M\theta_5 \\ M\theta_6 \\ M\theta_7 \end{pmatrix} = \begin{pmatrix} N_5 & \frac{-N_5}{n_6} & 0 \\ N_5 & \frac{N_5}{n_6} & 0 \\ 0 & 0 & -N_7 \end{pmatrix} \begin{pmatrix} J\theta_5 \\ J\theta_6 \\ J\theta_7 \end{pmatrix}$$

Equation 9 – Wrist Joint-to-Motor position transformation

Motor-to-Joint Torque Transformations
Similar to the position transformations the following equations determine the joint torque from the motor torque:

$$\begin{pmatrix} J\tau_1 \\ J\tau_2 \\ J\tau_3 \\ J\tau_4 \end{pmatrix} = \begin{pmatrix} -N_1 & 0 & 0 & 0 \\ 0 & N_2 & -N_2 & 0 \\ 0 & \frac{-N_2}{n_3} & \frac{-N_2}{n_3} & 0 \\ 0 & 0 & 0 & -N_4 \end{pmatrix} \begin{pmatrix} M\tau_1 \\ M\tau_2 \\ M\tau_3 \\ M\tau_4 \end{pmatrix}$$

Equation 10 – Arm Motor-to-Joint torque transformation

$$\begin{pmatrix} J\tau_5 \\ J\tau_6 \\ J\tau_7 \end{pmatrix} = \begin{pmatrix} N_5 & N_5 & 0 \\ \frac{-N_5}{n_6} & \frac{N_5}{n_6} & 0 \\ 0 & 0 & -N_7 \end{pmatrix} \begin{pmatrix} M\tau_5 \\ M\tau_6 \\ M\tau_7 \end{pmatrix}$$

Equation 11 – Wrist Motor-to-Joint transformations

The following equations determine motor torque from the joint torque:

$$\begin{pmatrix} M\tau_1 \\ M\tau_2 \\ M\tau_3 \\ M\tau_4 \end{pmatrix} = \begin{pmatrix} \frac{-1}{N_1} & 0 & 0 & 0 \\ 0 & \frac{1}{2N_2} & \frac{-n_3}{2N_2} & 0 \\ 0 & \frac{-1}{2N_2} & \frac{-n_3}{2N_2} & 0 \\ 0 & 0 & 0 & \frac{-1}{N_4} \end{pmatrix} \begin{pmatrix} J\tau_1 \\ J\tau_2 \\ J\tau_3 \\ J\tau_4 \end{pmatrix}$$

Equation 12 – Arm Joint-to-Motor torque transformations

$$\begin{pmatrix} M\tau_5 \\ M\tau_6 \\ M\tau_7 \end{pmatrix} = \begin{pmatrix} \frac{1}{2N_5} & \frac{-n_6}{2N_5} & 0 \\ \frac{1}{2N_5} & \frac{n_6}{2N_5} & 0 \\ 0 & 0 & \frac{-1}{N_7} \end{pmatrix} \begin{pmatrix} J\tau_5 \\ J\tau_6 \\ J\tau_7 \end{pmatrix}$$

Equation 13 – Wrist Joint-to-Motor torque transformations

Appendix A Integrating a BarrettHand™

The BarrettHand can be readily integrated into the WAM system. The Hand can be powered either internally using the built-in DC-DC converter on the Safety Board, or externally using the standard BarrettHand power supply. See Table 2 for the necessary Safety Board switch settings to power and control the Hand.

There are two methods by which Hand control can be integrated with Arm control. The first method allows the user to control the Hand from a separate window while the Arm is running. The second method is integration of Hand commands with Arm commands within the same program. Both methods require the following setup procedure.

Setup:

- Turn power to entire system OFF
- Plug in Hand connector that extends from the modular outer link or Wrist into base of Hand
- Use white washers and screws to secure the connector.
- Align pins on Tool Plate with holes on Hand Base Ring
- Attach Hand to modular outer link or Wrist by turning Tool Plate Attachment Ring clockwise (when viewed from beyond the palm of the hand).
- The pins must engage completely for Hand to be securely attached to Wrist
- Plug external hand cable from base of Arm to rear of Hand Power Supply Box
- Plug power cord into Power Supply Box
- Attach a standard serial cable from COM port to rear of Hand Power Supply Box

The Hand is attached and ready for operation. Read the BarrettHand User Manual before proceeding with Hand operation.

Before operating both Hand and Arm together the Hand should be tested on the end of the Arm. Place Arm in a configuration that allows the Hand to move through its full range of motion. Follow the procedures in the BarrettHand User Manual for testing Hand operation. If Hand operation is successful, the system is ready for use.

To operate the Hand separately from the Arm:

- Turn Hand Power Supply Box ON
- Open a serial terminal application (such as TeraTerm, HyperTerminal, minicom) and connect to the COM port of the BarrettHand at 9600 baud, No parity, 8 bits, One stop bit, No flow control
- Put Arm in a configuration that allows for full joint motion of the Hand
- Initialize Hand (Type "HI"). The Hand is now ready for operation while executing Arm programs
- Execute desired Arm program
- While Arm program is running, switch to the serial terminal window
- Type Hand commands (see the Grasper Control Language section of the BarrettHand manual

To operate Hand and Arm from the same program:

Integration of the Hand and the Arm in the same program requires opening and initializing the serial port from within your WAM control application, then sending the desired Grasper Control Language (GCL) commands from the application. See the directory of example code for details about how this is implemented.

Type Hand commands (see the Grasper Control Language section of the BarrettHand manual)

Appendix B Technical Specifications

Kinematics

Total number of joints: 4 or 7
Total number of motors: 4 or 7
Total joint friction: 3 Nm
Mechanical stiffness: $1.5*10^6$ N/m
Control stiffness: 5000 N/m
Percent backdrivability: > 95%

Range of Motion

Reach:
4-DOF WAM: 0.90 m to mounting plate
7-DOF WAM: 0.91 m to mounting plate

Joint	Positive Joint Limit	Negative Joint Limit
1	2.6 (150˚)	-2.6 (-150˚)
2	2.0 (113˚)	-2.0 (-113˚)
3	2.8 (157˚)	-2.8 (-157˚)
4	3.1 (180˚)	-0.9 (-50˚)
5	1.24 (71˚)	-4.76 (-273˚)
6	1.57 (90˚)	-1.57 (-90˚)
7	3.0 (172˚)	-3.0 (-172˚)

Weight

4-DOF WAM: 25.6 kg (56.5 lb)
7-DOF WAM: 27.4 kg (60.5 lb)

Maximum Payload

4-DOF WAM: 4.0 kg (8.8 lb) at endpoint
7-DOF WAM: 3.0 kg (6.6 lb) at endpoint

Motor Type

Neodymium Iron Boron, brushless, DC servo motors

Mechanisms

Cable drives integrated with a cable differential
Barrett quick-connect feature for outer-link
Puck™ ultra-miniature motor encoders

Power Requirements

Single phase AC electrical outlet with ground:
Load: 250 W
Phases: Single
Voltage: 100-120 & 200-240 VAC
Frequency: 50/60 Hz

Power Supply

Location: dry, stable surface
Voltage: 48 VDC output
Size: 279.4 x 127 x 63.5 mm (11 x 5 x 2.5 in)
Weight: 2.27 kg (5.0 lb)

Cables

AC Line Cord
DC Power Cable
Pendant Cables (Two)
Ethernet Cable
External CAN cable (optional)
External Hand control cable (optional)

Available Options

3-DOF Wrist
3-DOF Gimbals
External Computer

WAM Dimensions

NOTE: The dimensions for a 4 degree of freedom WAM are the same except that the 4 degree of freedom WAM does not have a joint at 850 mm and the "End Plate" dimension is 900 mm.

All measurements are in millimeters.

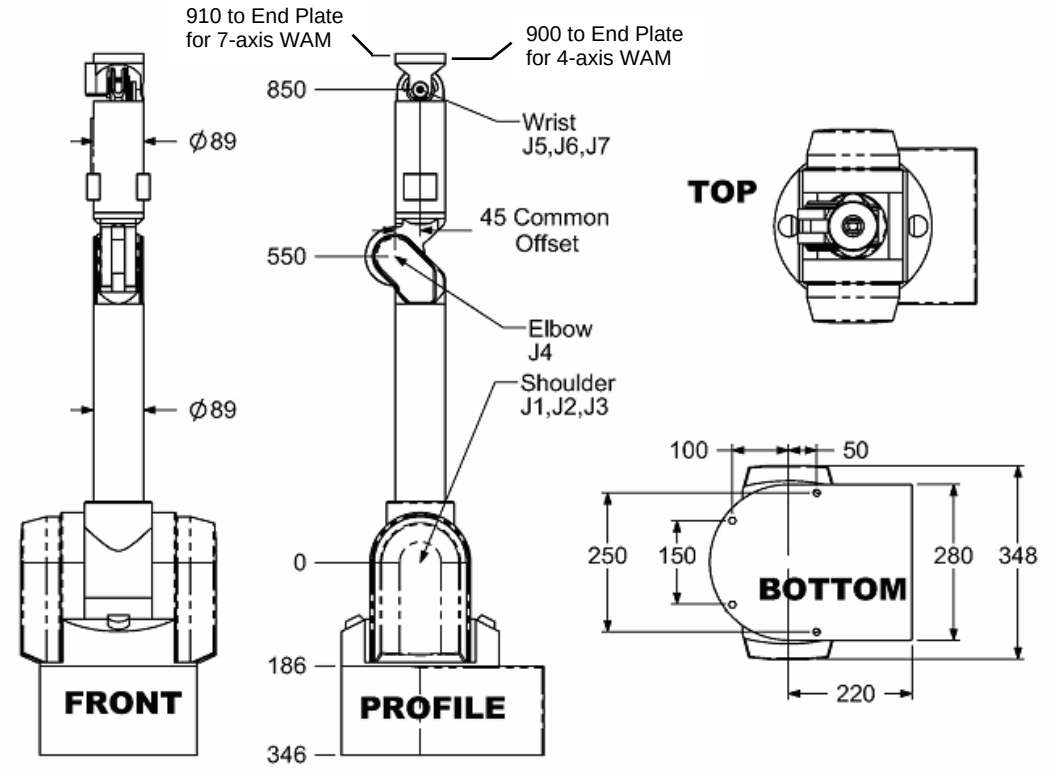

Figure 45 – WAM Dimensions

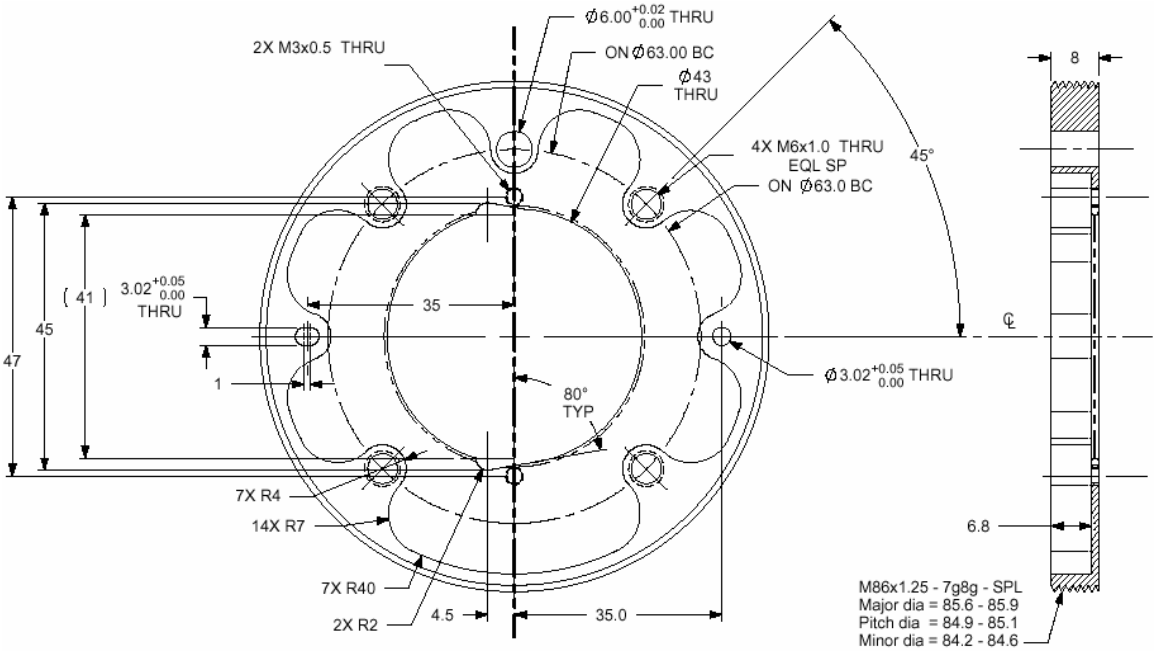

Figure 46 – Tool Plate Dimensions

US Patents (and international equivalents)

4,903,536 4,957,320
5,046,375 5,207,114
5,388,480 5,501,498
D351,849 D352,050
Additional US and international patents pending

Appendix C FAQ

Q1: What types of motions are possible with the WAM Arm, and what mechanisms are used to control them?

A1: The WAM Arm has a workspace that can be approx a sphere 1 meter in diameter centered between the WAM's "shoulders". It can reach any position and orientation that does not require the arm to intersect with itself or some obstacle in a workspace ____. While most of the joints are connected to standard cable drives, joints 2 and 3 (the roll and pitch of the Arm) are both controlled by motors 2 and 3 using Barrett's patented cable differential (see Figure 47). In addition, all motors and the Safety Board are controlled by Barrett's Puck™ miniature brushless motor controller (see Figure 48). The Puck™ serves a multitude of purposes (see Section 9.1 and Appendix D for details).

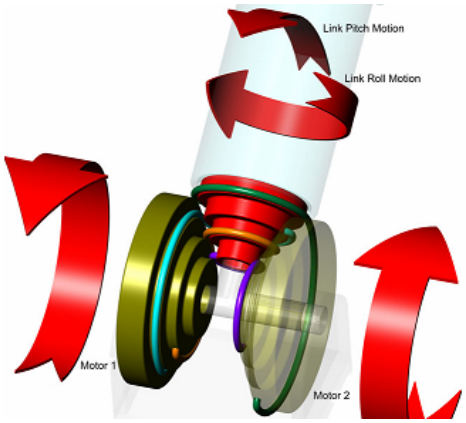

Figure 47 – Cable Differential

Figure 48 – Pucks™ – Miniature Brushless Motor Controllers

Q2: What materials is the WAM made of?

A2: The WAM (both the Arm and the Wrist) are made mostly from aircraft-grade aluminum, but also contains some steel (ball bearings, stainless drive cables, fasteners), carbon-fiber-epoxy internal structures, titanium (Ti-6Al-4V), PEEK, copper motor-stator coils and bus/signal wiring, cobalt motor laminations, ceramic rotor magnets, Teflon-impregnated ceramics for cable pulley surfaces, Kydex acrylic-polyvinyl chloride covers, RTV, FR4 phenolic, adhesives, beryllium-copper, and gold.

Q3: Does my WAM use optical or magnetic encoders?

A3: Your WAM uses magnetic encoders if it has a serial number greater than 20 (WAM-05020 or greater), or if you were notified that your encoders were updated to magnetic encoders during a repair or replacement. Otherwise your WAM has optical encoders. The property CTS can also be used to differentiate between optical and magnetic encoders -- if CTS is 40960, then the encoder is optical, if CTS is 4096 then the encoder is magnetic.

Q4: What qualities of a WAM depend on whether the WAM implements optical or magnetic encoders?

A4: The number of encoder ticks per motor revolution:

- Optical Encoders – There are 40960 encoder ticks per motor revolution for each motor.
- Magnetic Encoders – There are 4096 encoder ticks per motor revolution for each motor.

The procedure for motor controllers when the WAM is first powered on:

- Optical Encoders – When the WAM is first powered on, the motor controllers use Hall-effect sensors and six-step commutation for control until an initial hall transition occurs. Then they switch to using the incremental encoders for smoother commutation until the encoder index pulse is observed (once per motor revolution). Then they begin a factory-calibrated commutation loop for precise torque control.
- Magnetic Encoders – When the WAM is first powered on, the motor controllers use the absolute magnetic encoder to begin a factory-calibrated commutation loop for precise torque control.

MECH:

- Optical Encoders – MECH tells you how many encoder counts there are between the present location and the index pulse. Unless you want to write your own cogging or torque ripple cancellation code, these properties are not important. It is only important to know the orientation of the rotor magnets inside the stator for low-level commutation and any types of perturbation cancellation you may wish to implement
- Magnetic Encoders – MECH is the raw magnetic encoder feedback, absolute within one rotation of the motor.

MOFST:

- Optical Encoders – MOFST is the distance in encoder counts between the index pulse of the encoder and the start of an electrical cycle for that motor (a point where the magnets are aligned with a single phase coil).
- Magnetic Encoders – MOFST is the encoder reading at the start of an electrical cycle.

Q5: How do I calculate the desired Maximum Torque (MT) values for each joint?
A5:

Table 9 – Maximum Torque (MT) mA to Nm conversion table

(Note: Cable limits chosen to be approx 20% of the rated breaking strength)

Motor:	1	2	3	4	5	6	7
Stall (Nm):	1.49	1.49	1.49	1.49	0.356	0.356	0.091
Peak (Nm):	6.31	6.31	6.31	6.31	1.826	1.826	0.613
Cable limit (Nm):	1.8	1.8	1.8	1.6	0.6	0.6	N/A
Nm/A–published:	0.457	0.457	0.457	0.457	0.236	0.236	0.067
Nm/A–actual:	0.379	0.379	0.379	0.379	0.157	0.157	0.058

Example:
What should the motor 3 Max Torque (MT) be set to in order to not exceed the designed cable limit?
(1.8 Nm) / (0.379 Nm/A) * (1024 Units/A) = 4863

```
setProperty(bus, 1, MT, FALSE, 4860); // Cable limit = 4860
setProperty(bus, 2, MT, FALSE, 4860); // Cable limit = 4860
setProperty(bus, 3, MT, FALSE, 4860); // Cable limit = 4860
setProperty(bus, 4, MT, FALSE, 4320); // Cable limit = 4320
setProperty(bus, 5, MT, FALSE, 3900); // Cable limit = 3900
setProperty(bus, 6, MT, FALSE, 3900); // Cable limit = 3900
setProperty(bus, 7, MT, FALSE, 3200); // J7 Gears (max stall = 1600)
```

Appendix D GLOSSARY

Backdrivability - *Backdrivability* is the measure of how accurately a force or motion that is applied at the output end of a mechanical transition is reproduced at the input end. In a mechanical robot-like linkage, good backdrivability means that a person can grab the ending of the linkage and move it around effortlessly. Some robots show acceptable backdrivability for slow disturbances, but near-zero backdrivability when the disturbances happen suddenly.

Backdrivable Robot - A *robot* which exhibits greater than 95% backdrivability.

BarrettHand™ - The 1.2 kilogram dexterous robotic Grasper™ as described in the BarrettHand™ BH8-Series User's Manual.

Cables (mechanical tension elements) - A "wire rope" typically made from very fine strands of stainless steel.

CANbus Communications - *CANbus* is a robust, deterministic, and addressable 2-wire serial-communications protocol that operates up to 1 megabit/sec.

Cartesian Space - *Cartesian space* is a rectilinear, orthogonal description of three-dimensional space. It consists of three infinitely long straight lines that intersect at a mutual point, the origin. The lines are often associated with right-hand X, Y, and Z axes, and half of each line is designated positive. For the WAM, the world coordinate frame is right-handed and its origin is in the center of the shoulder with X pointing toward the front (rounded) part of the WAM, and Z pointing up (see frame docs in this manual).

Controller - The word *controller* in the field of motion controls has two meanings:

1. A motor controller takes sensor feedback of rotor position and performs (brushless-commutation) calculations to determine a set of winding currents for that position that will produce either a desired trajectory or a desired torque (in cases with very low Coulomb friction). Then, in most modern motor controllers, a set of PWM signals is generated that control a set of (typically 6) FETs (through charge pumps) arranged in a double-H bridge (in the 6-FET case).
2. A high-level motion controller coordinates the motions of one or more axes of a robot. A simple controller may apply a PID filter to a single joint to follow a trapezoidal function (velocity vs. time) with limits on acceleration/deceleration and velocity. More sophisticated controllers may control trajectories and forces in true Cartesian space and may exploit modeled observers, nonlinear functions, and adaptive control techniques.

Degrees of Freedom - *Degrees of freedom* is shorthand for *independent degrees of freedom*, which is almost always equal to the number of motors, as long as the motors drive independent axes. Generally a six-degree-of-freedom robotic arm can place its tool frame origin at any location within its reach and orient that frame arbitrarily anywhere within its dexterous workspace. However, for a fixed position and orientation of the tool frame the arm pose is fixed.

When there are more degrees of freedom than 6-space, the robot arm kinematics are called redundant. For example, a seven-degree-of-freedom robotic arm such as the WAM arm remains free to move even while holding a position and orientation.

Denavit-Hartenberg (D-H) - *Denavit-Hartenberg* is a technique for analyzing the type of serial-link kinematic chains used in robotic manipulators in which each frame is defined relative to its adjacent frame via two length dimensions and two rotations. The scheme is flexible enough to work with both revolute and prismatic joints. An important initial step in the analysis is to affix coordinate frames to each sequential link according to a protocol that is consistent with correct interpretation of the D-H parameters.

Denavit-Hartenberg Parameters - The *D-H parameters* form an *n* x 4 matrix of parameters that define the kinematic relationship between coordinate frames that are attached to links in a robot. Knowledge of the D-H parameters allows immediate construction of the transformation matrices. Multiplying the chain of transformation matrices enables the final tool frame to be described relative to the user's robot-base or room coordinates.

Encoders - *Encoders* come in two main types: incremental and absolute. The sensor types are usually optical, but may also be magnetic or electrical. In absolute encoders, when power is booted, the encoder knows its absolute location before there is any motion. Incremental encoders rely on A and B pulse trains in quadrature to report relative distances and direction. An optional index pulse on a third track allows the incremental encoder to calibrate and begin reporting absolute position only after the rotor rotates far enough (up to 1 full turn) to locate it.

Ethernet Communications - *Ethernet* is a high-speed non-deterministic serial communications protocol. There are versions for 2-, 4-, and 8-wire variants and optical physical layers, with bandwidth highest in the 8-wire and optical versions.

Firmware - Software that is embedded in a hardware device that allows reading and executing the software, but does not allow modification, e.g., writing or deleting data by an end user.

Gravity Compensation - *Gravity compensation* is a specialized type of feedforward control used in backdrivable robotic arms. The kinematics, relative orientation of the gravity vector, and instantaneous pose must be known or knowable. The link and payload masses must be known a priori or measured (in a process called identification). In these circumstances, only a few sines and cosines must be evaluated to determine the unique set of joint torques required to balance gravity precisely and these balancing torque change only gradually as the arm moves or is pushed around the workspace.

Haptic Device - An articulated machine designed to come in physical contact with people, usually through a handle that a person grasps, and to have the capability to control the forces that the person experiences.

Haptic Object- A *haptic object* is a virtual object that is suspended (and can move and morph) within the workspace of a haptic device. A person moving the handle, stylus, or thimble of the device senses resistance forces generated by the device to simulate collision with the membrane of the haptic object.

Haptics - *Haptics* is the study of the perception of touch in exploring and controlling objects and fluids that we can touch.

Home Position - The default *home position* for the WAM (specified in the Quick-Start Guide) is J2 folded about 2 rad back against its large rubber stop, and J4 folded in against the inner link (+pi rad). It looks like the WAM is trying to touch its shoulder. In order to change the default home position, you must change the defined home joint positions in wam.conf. The normal joint readings for home positions are as follows:

J1	J2	J3	J4	J5	J6	J7
0	-2	0	3.14	0	0	0

Jacobian - The *Jacobian* is a mathematical matrix representing the first derivative of kinematics and is very important in robotics. There are two interpretations of the Jacobian: one using the Jacobian-Inverse, and the other using the Jacobian-Transpose. The latter is only used in advanced robotics and is more powerful.

Jacobian-Inverse - *Jacobian-Inverse* is a computationally slow and burdensome operation in robotics, generally calculated off-line on a powerful processor, and virtually barring true realtime Cartesian control. It is a matrix inversion in the formal sense, which means that it must be well conditioned (the robot must be far from singularities) and it must be square (so only non-redundant robots can have a Jacobian-Inverse).

Jacobian-Transpose - A *Jacobian-Transpose* is derived from the Jacobian with no computation, allowing realtime Cartesian control of robots with redundant degrees-of-freedom. A mathematical matrix transpose affects only the order in which the matrix elements are written into and subsequently read from memory.

Kinematics - The science of motion which treats motion without regard to the forces which cause it, specifically all the geometrical and time-based properties of position.

Micro-Yielding - *Micro-yielding* occurs when instantaneous cable tension loads combined with bending-induced (tension) stresses in the outmost cable fibers exceed the yield strength in the material, causing some of the fibers to break.

Pretension - The process of adding additional tension to a cable during the assembly process.

Puck™ - An ultra-miniature brushless servo controller fit into an ultra-high-precision encoder. Developed over several years by Barrett Technology, this puck weighs only 44gms and is only 35mm (dia) x 17 (high) with connectors. The Puck™ fulfills both definitions of a Controller (see Controller).

Stiffness - The *stiffness* of a spring or similar mechanism (with units of force/distance, or force x distance in the case of trosional mechanisms) is defined as the slope of the curve of the restoring force relative to the displacement from equilibrium. So, in a robot stiffness can be measured at the endtip of the arm with the arm outstretched. In robotics high stiffness is always better.

Whole Arm Manipulation - There are two meanings to *Whole-Arm Manipulation*, both exemplified in Barrett's WAM arm:

1. The inherent capability of a robot to control contact forces with inherent safety and robustness all along its link and joint surfaces, from the base of the arm through its end. This capability is especially enabling when applied to a kinematically redundant arm.
2. A holistic approach to robot design enhances safety and performance while reducing intrinsic manufacturing, shipping, training, software, footprint, power, and servicing costs. For example, conventional robot designs continue today to depend on legacy drive technologies that introduce friction, backlash, torque fluctuations, or generate poor aspect ratios. Then conventional designers go to great lengths to attempt to mask these effects at the expense of safety or other performance metrics. The holistic approach leverages simple, (high-speed cable) drives that do not introduce these problems in the first place and yet exceed the highest standards of conventional performance.

Index